Analisi dei dati moderna

NetApp Solutions

NetApp
April 26, 2024

Sommario

Soluzioni NetApp per l'analisi dei dati moderna . . . 1
Gestione dei dati nel cloud con la dualità di file-object NetApp e AWS SageMaker . . . 1
Carichi di lavoro Apache Kafka con storage NetApp NFS . . . 30
Confluenza di Kafka con i controller di storage NetApp ONTAP . . . 78
Soluzioni storage NetApp per Apache Spark . . . 90
Analisi dei big data dati per l'intelligenza artificiale . . . 138
Best practice per Confluent Kafka . . . 182
Soluzioni dati di cloud ibrido NetApp: Spark e Hadoop in base ai casi di utilizzo dei clienti . . . 211
Analisi dei dati moderna: Soluzioni diverse per diverse strategie di analisi . . . 229
TR-4623: NetApp e-Series E5700 e Splunk Enterprise . . . 229
NVA-1157-DEPLOY: Workload Apache Spark con soluzione storage NetApp . . . 229

Soluzioni NetApp per l’analisi dei dati moderna

Gestione dei dati nel cloud con la dualità di file-object NetApp e AWS SageMaker

TR-4967: Gestione dei dati nel cloud con la dualità di NetApp file-object e AWS SageMaker

Karthikeyan Nagalingam, NetApp

I data scientist e i tecnici spesso devono accedere ai dati memorizzati nel formato NFS, ma l’accesso a questi dati direttamente dal protocollo S3 in AWS SageMaker può essere difficile perché AWS supporta solo l’accesso al bucket S3. Tuttavia, NetApp ONTAP offre una soluzione abilitando l’accesso a due protocolli per NFS e S3. Con questa soluzione, data scientist e ingegneri possono accedere ai dati NFS dai notebook AWS SageMaker tramite i bucket S3 di NetApp Cloud Volumes ONTAP. Questo approccio consente un facile accesso e condivisione degli stessi dati da NFS e S3 senza la necessità di software aggiuntivo.

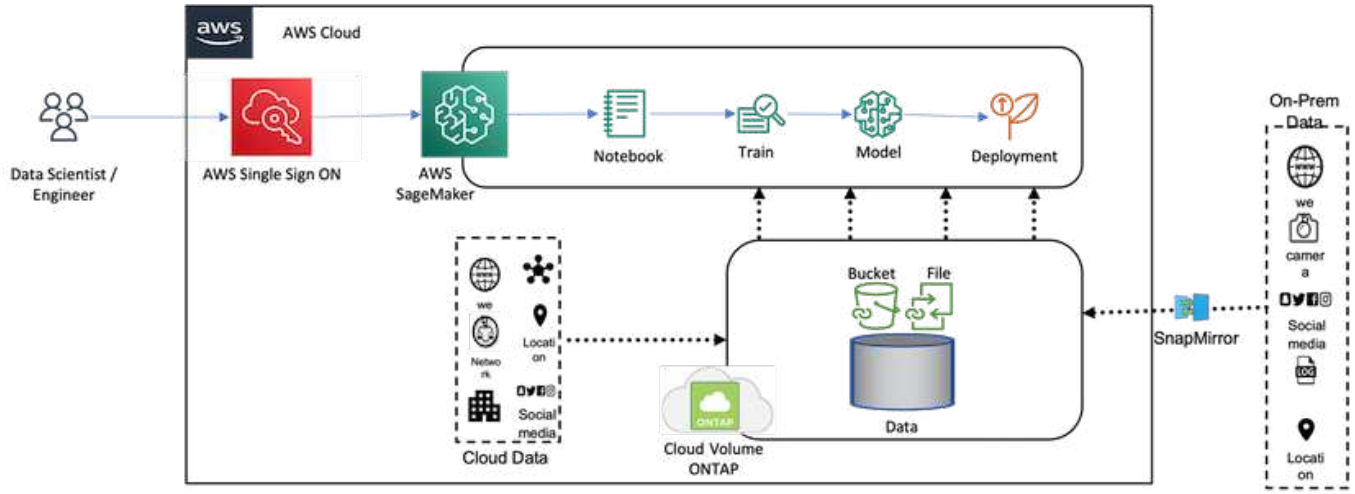
Tecnologia della soluzione

Questa soluzione utilizza le seguenti tecnologie:

- **AWS SageMaker notebook.** offre funzionalità di apprendimento automatico a sviluppatori e data scientist per creare, formare e implementare modelli ML di alta qualità in modo efficiente.
- **NetApp BlueXP.** consente il rilevamento, l’implementazione e il funzionamento dello storage on-premise, nonché su AWS, Azure e Google Cloud. Offre protezione dei dati contro la perdita di dati, le minacce informatiche e le interruzioni non pianificate e ottimizza lo storage e l’infrastruttura dei dati.
- **NetApp Cloud Volumes ONTAP.** offre volumi di storage di livello Enterprise con protocolli NFS, SMB/CIFS, iSCSI e S3 su AWS, Azure e Google Cloud, offrendo agli utenti una maggiore flessibilità nell’accesso e nella gestione dei dati nel cloud.

NetApp Cloud Volumes ONTAP è stato creato da BlueXP per memorizzare i dati ML.

La figura seguente mostra i componenti tecnici della soluzione.



Riepilogo del caso d'utilizzo

Un potenziale caso di utilizzo per l'accesso a due protocolli di NFS e S3 riguarda l'apprendimento automatico e la scienza dei dati. Ad esempio, un team di data scientist potrebbe lavorare a un progetto di machine learning utilizzando AWS SageMaker, che richiede l'accesso ai dati memorizzati nel formato NFS. Tuttavia, potrebbe essere necessario accedere e condividere i dati tramite i bucket S3 per collaborare con altri membri del team o per integrarli con altre applicazioni che utilizzano S3.

Utilizzando NetApp Cloud Volumes ONTAP, il team può memorizzare i dati in un'unica posizione e renderli accessibili con i protocolli NFS e S3. I data scientist possono accedere ai dati in formato NFS direttamente da AWS SageMaker, mentre altri membri del team o applicazioni possono accedere agli stessi dati tramite i bucket S3.

Questo approccio consente di accedere e condividere i dati in modo semplice ed efficiente senza la necessità di software aggiuntivo o migrazione dei dati tra diverse soluzioni di storage. Inoltre, consente di ottimizzare il workflow e la collaborazione tra i membri del team, consentendo uno sviluppo più rapido ed efficace dei modelli di machine learning.

Dualità dei dati per data scientist e altre applicazioni

I dati sono disponibili in NFS e accessibili da S3 da AWS SageMaker.

Requisiti tecnologici

I notebook NetApp BlueXP, NetApp Cloud Volumes ONTAP e AWS SageMaker sono necessari per il caso di utilizzo della doppia funzionalità dei dati.

Requisiti software

La seguente tabella elenca i componenti software necessari per implementare il caso d'utilizzo.

Software	Quantità
BlueXP	1
NetApp Cloud Volumes ONTAP	1
Notebook AWS SageMaker	1

Procedure di implementazione

L'implementazione della soluzione per la dualità dei dati comporta le seguenti attività:

- Connettore BlueXP
- NetApp Cloud Volumes ONTAP
- Dati per l'apprendimento automatico
- AWS SageMaker
- Apprendimento automatico validato dai notebook Jupyter

Connettore BlueXP

In questa convalida, abbiamo utilizzato AWS. È applicabile anche a Azure e Google Cloud. Per creare un connettore BlueXP in AWS, attenersi alla seguente procedura:

1. Abbiamo utilizzato le credenziali basate sull'abbonamento mcarl-marketplace in BlueXP.
2. Scegli la regione adatta al tuo ambiente (ad esempio, US-East-1 [N. Virginia]), quindi selezionare il metodo di autenticazione (ad esempio, assumere le chiavi role o AWS). In questa convalida, utilizziamo le chiavi AWS.
3. Fornire il nome del connettore e creare un ruolo.
4. Fornire i dettagli di rete, ad esempio VPC, subnet o coppia di chiavi, a seconda che sia necessario un IP pubblico o meno.
5. Fornire i dettagli per il gruppo di protezione, ad esempio l'accesso HTTP, HTTPS o SSH dal tipo di origine, ad esempio le informazioni su Anywhere e sull'intervallo IP.
6. Esaminare e creare BlueXP Connector.
7. Verificare che lo stato dell'istanza di BlueXP EC2 sia in esecuzione nella console AWS e controllare l'indirizzo IP dalla scheda **Networking**.
8. Accedere all'interfaccia utente del connettore dal portale BlueXP oppure utilizzare l'indirizzo IP per l'accesso dal browser.

NetApp Cloud Volumes ONTAP

Per creare un'istanza di Cloud Volumes ONTAP in BlueXP, attenersi alla seguente procedura:

1. Creare un nuovo ambiente di lavoro, selezionare il provider cloud e selezionare il tipo di istanza di Cloud Volumes ONTAP (ad esempio CVO singolo, ha o Amazon FSxN per ONTAP).
2. Fornire dettagli come il nome e le credenziali del cluster Cloud Volumes ONTAP. In questa convalida, abbiamo creato un'istanza di Cloud Volumes ONTAP chiamata `svm_sagemaker_cvo_sn1`.
3. Selezionare i servizi necessari per Cloud Volumes ONTAP. In questa convalida, abbiamo scelto di eseguire solo il monitoraggio, quindi abbiamo disattivato **Data Sense & Compliance** e **Backup to Cloud Services**.
4. Nella sezione **Location & Connectivity**, selezionare la regione AWS, VPC, subnet, gruppo di sicurezza, metodo di autenticazione SSH, e una password o una coppia di chiavi.
5. Scegliere il metodo di ricarica. Per questa convalida abbiamo utilizzato **Professional**.
6. È possibile scegliere un pacchetto preconfigurato, ad esempio **POC e piccoli carichi di lavoro**, **carichi di lavoro di produzione di dati applicativi e database**, **DR conveniente** o **carichi di lavoro di produzione dalle performance più elevate**. In questa convalida, scegliamo **POC e workload di piccole dimensioni**.
7. Creare un volume con una dimensione specifica, protocolli consentiti e opzioni di esportazione. In questa

convalida, abbiamo creato un volume chiamato `vol1`.

8. Scegliere un tipo di disco del profilo e una policy di tiering. In questa convalida, abbiamo disattivato **efficienza dello storage** e **SSD General- purpose – Dynamic Performance**.
9. Infine, esaminare e creare l'istanza di Cloud Volumes ONTAP. Quindi attendere 15-20 minuti affinché BlueXP crei l'ambiente di lavoro Cloud Volumes ONTAP.
10. Configurare i seguenti parametri per attivare il protocollo di dualità. Il protocollo di dualità (NFS/S3) è supportato da ONTAP 9. 12.1 e versioni successive.
    a. In questa convalida, abbiamo creato una SVM chiamata `svm_sagemaker_cvo_sn1` e volume `vol1`.
    b. Verificare che SVM disponga del supporto del protocollo per NFS e S3. In caso contrario, modificare la SVM per supportarla.

```
sagemaker_cvo_sn1::> vserver show -vserver svm_sagemaker_cvo_sn1
                                    Vserver: svm_sagemaker_cvo_sn1
                               Vserver Type: data
                            Vserver Subtype: default
                               Vserver UUID: 911065dd-a8bc-11ed-bc24-
e1c0f00ad86b
                                Root Volume:
svm_sagemaker_cvo_sn1_root
                                  Aggregate: aggr1
                                 NIS Domain: -
                  Root Volume Security Style: unix
                                LDAP Client: -
               Default Volume Language Code: C.UTF-8
                            Snapshot Policy: default
                              Data Services: data-cifs, data-
flexcache,
                                             data-iscsi, data-nfs,
                                             data-nvme-tcp
                                    Comment:
                               Quota Policy: default
                List of Aggregates Assigned: aggr1
 Limit on Maximum Number of Volumes allowed: unlimited
                        Vserver Admin State: running
                  Vserver Operational State: running
   Vserver Operational State Stopped Reason: -
                          Allowed Protocols: nfs, cifs, fcp, iscsi,
ndmp, s3
                       Disallowed Protocols: nvme
            Is Vserver with Infinite Volume: false
                           QoS Policy Group: -
                        Caching Policy Name: -
                                Config Lock: false
                               IPspace Name: Default
                         Foreground Process: -
                    Logical Space Reporting: true
                  Logical Space Enforcement: false
Default Anti_ransomware State of the Vserver's Volumes: disabled
            Enable Analytics on New Volumes: false
    Enable Activity Tracking on New Volumes: false

sagemaker_cvo_sn1::>
```

11. Creare e installare un certificato CA, se necessario.
12. Creare una policy sui dati del servizio.

```
sagemaker_cvo_sn1::*> network interface service-policy create -vserver
svm_sagemaker_cvo_sn1 -policy sagemaker_s3_nfs_policy -services data-
core,data-s3-server,data-nfs,data-flexcache
sagemaker_cvo_sn1::*> network interface create -vserver
svm_sagemaker_cvo_sn1 -lif svm_sagemaker_cvo_sn1_s3_lif -service-policy
sagemaker_s3_nfs_policy -home-node sagemaker_cvo_sn1-01 -address
172.30.10.41 -netmask 255.255.255.192

Warning: The configured failover-group has no valid failover targets for
the LIF's failover-policy. To view the failover targets for a LIF, use
         the "network interface show -failover" command.

sagemaker_cvo_sn1::*>
sagemaker_cvo_sn1::*> network interface show
Logical    Status     Network            Current       Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node          Port
Home
----------- ---------- ---------- ------------------ -------------
------- ----
sagemaker_cvo_sn1
            cluster-mgmt up/up    172.30.10.40/26    sagemaker_cvo_sn1-
01
                                                                    e0a
true
            intercluster up/up    172.30.10.48/26    sagemaker_cvo_sn1-
01
                                                                    e0a
true
            sagemaker_cvo_sn1-01_mgmt1
                         up/up    172.30.10.58/26    sagemaker_cvo_sn1-
01
                                                                    e0a
true
svm_sagemaker_cvo_sn1
            svm_sagemaker_cvo_sn1_data_lif
                         up/up    172.30.10.23/26    sagemaker_cvo_sn1-
01
                                                                    e0a
true
            svm_sagemaker_cvo_sn1_mgmt_lif
                         up/up    172.30.10.32/26    sagemaker_cvo_sn1-
01
                                                                    e0a
true
            svm_sagemaker_cvo_sn1_s3_lif
                         up/up    172.30.10.41/26    sagemaker_cvo_sn1-
```

```
01
                                                                    e0a
true
6 entries were displayed.

sagemaker_cvo_sn1::*>
sagemaker_cvo_sn1::*> vserver object-store-server create -vserver
svm_sagemaker_cvo_sn1  -is-http-enabled true -object-store-server
svm_sagemaker_cvo_s3_sn1 -is-https-enabled false
sagemaker_cvo_sn1::*> vserver object-store-server show

Vserver: svm_sagemaker_cvo_sn1

           Object Store Server Name: svm_sagemaker_cvo_s3_sn1
               Administrative State: up
                        HTTP Enabled: true
             Listener Port For HTTP: 80
                       HTTPS Enabled: false
     Secure Listener Port For HTTPS: 443
  Certificate for HTTPS Connections: -
                  Default UNIX User: pcuser
               Default Windows User: -
                            Comment:

sagemaker_cvo_sn1::*>
```

13. Controllare i dettagli dell’aggregato.

```
sagemaker_cvo_sn1::*> aggr show


Aggregate     Size Available Used% State   #Vols  Nodes            RAID
Status
--------- -------- --------- ----- ------- ------ ----------------
------------
aggr0_sagemaker_cvo_sn1_01
           124.0GB   50.88GB   59% online       1 sagemaker_cvo_
raid0,
                                                   sn1-01
normal
aggr1      907.1GB   904.9GB    0% online       2 sagemaker_cvo_
raid0,
                                                   sn1-01
normal
2 entries were displayed.

sagemaker_cvo_sn1::*>
```

14. Creare un utente e un gruppo.

```
sagemaker_cvo_sn1::*> vserver object-store-server user create -vserver
svm_sagemaker_cvo_sn1 -user s3user

sagemaker_cvo_sn1::*> vserver object-store-server user show
Vserver     User            ID        Access Key          Secret Key
----------- --------------- --------- -------------------
-------------------
svm_sagemaker_cvo_sn1
            root            0         -                   -
   Comment: Root User
svm_sagemaker_cvo_sn1
            s3user          1         0ZNAX21JW5Q8AP80CQ2E

PpLs4gA9K0_2gPhuykkp014gBjcC9Rbi3QDX_6rr
2 entries were displayed.

sagemaker_cvo_sn1::*>


sagemaker_cvo_sn1::*> vserver object-store-server group create -name
s3group -users s3user -comment ""

sagemaker_cvo_sn1::*>
sagemaker_cvo_sn1::*> vserver object-store-server group delete -gid 1
-vserver svm_sagemaker_cvo_sn1

sagemaker_cvo_sn1::*> vserver object-store-server group create -name
s3group -users s3user -comment "" -policies FullAccess

sagemaker_cvo_sn1::*>
```

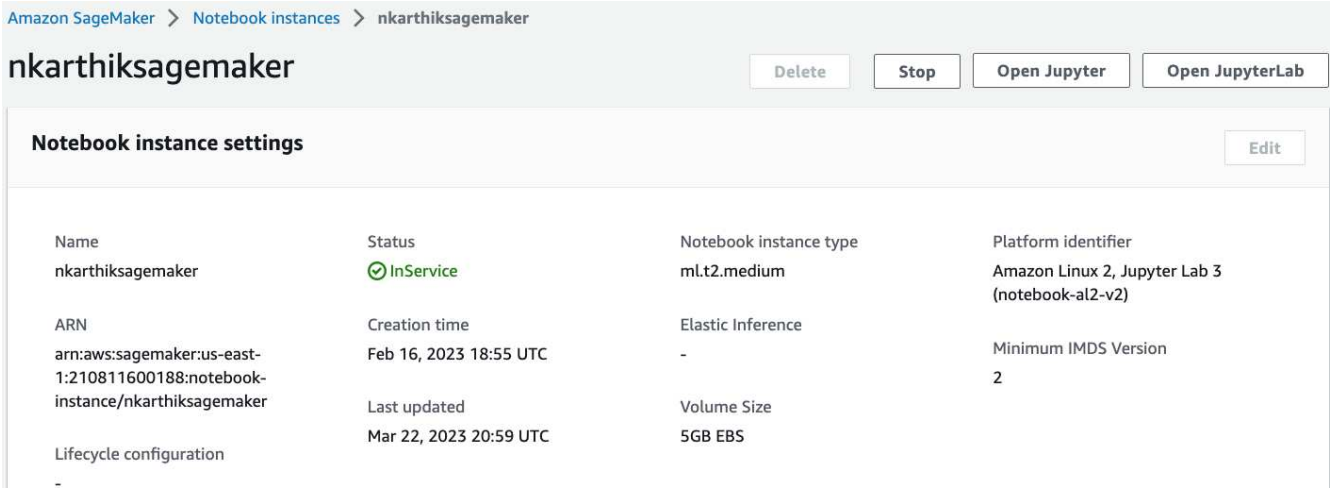
15. Creare un bucket sul volume NFS.

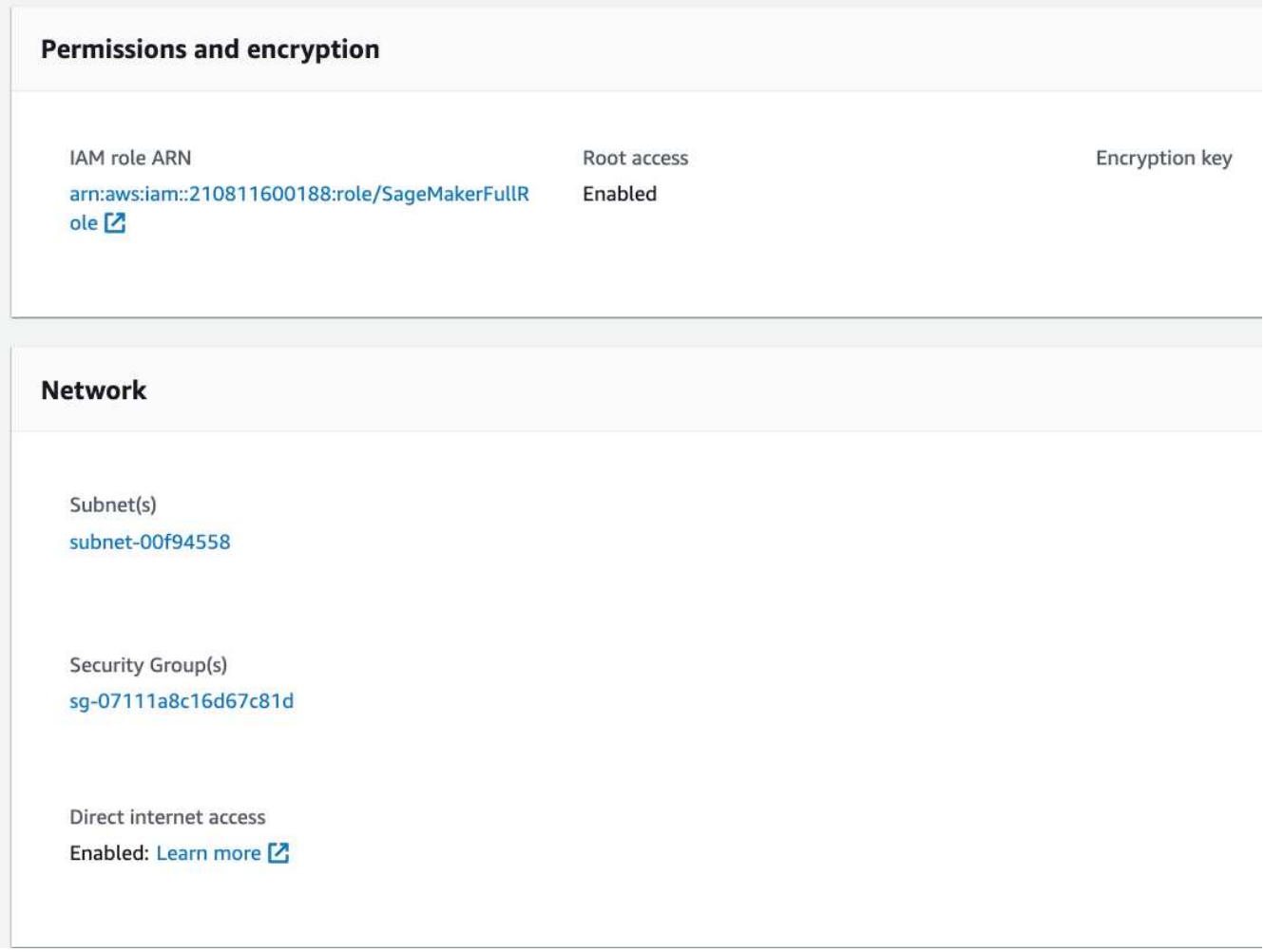
```
sagemaker_cvo_sn1::*> vserver object-store-server bucket create -bucket
ontapbucket1 -type nas -comment "" -vserver svm_sagemaker_cvo_sn1 -nas
-path /vol1
sagemaker_cvo_sn1::*> vserver object-store-server bucket show
Vserver     Bucket          Type     Volume            Size
Encryption Role       NAS Path
----------- --------------- -------- ----------------- ----------
---------- ---------- ----------
svm_sagemaker_cvo_sn1
            ontapbucket1    nas      vol1              -          false
-          /vol1
sagemaker_cvo_sn1::*>
```

AWS SageMaker

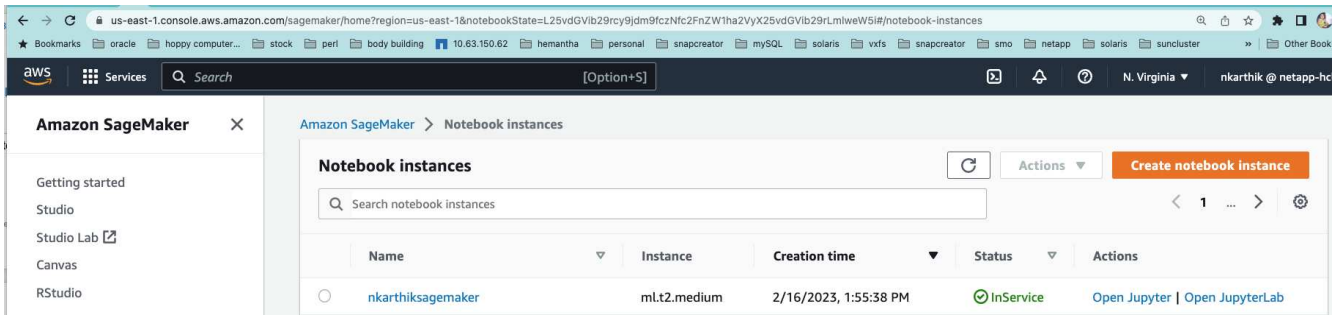
Per creare un notebook AWS da AWS SageMaker, attenersi alla seguente procedura:

1. Assicurarsi che l'utente che sta creando un'istanza di notebook disponga di un criterio IAM AmazonSageMakerFullAccess o faccia parte di un gruppo esistente che dispone dei diritti AmazonSageMakerFullAccess. In questa convalida, l'utente fa parte di un gruppo esistente.
2. Fornire le seguenti informazioni:
   - Nome dell'istanza del notebook.
   - Tipo di istanza.
   - Identificatore della piattaforma.
   - Selezionare il ruolo IAM che dispone dei diritti AmazonSageMakerFullAccess.
   - Root access (accesso root): Abilitare.
   - Encryption key (chiave di crittografia) - selezionare NO customed Encryption (
   - Mantenere le restanti opzioni predefinite.
3. In questa convalida, i dettagli dell'istanza di SageMaker sono i seguenti:

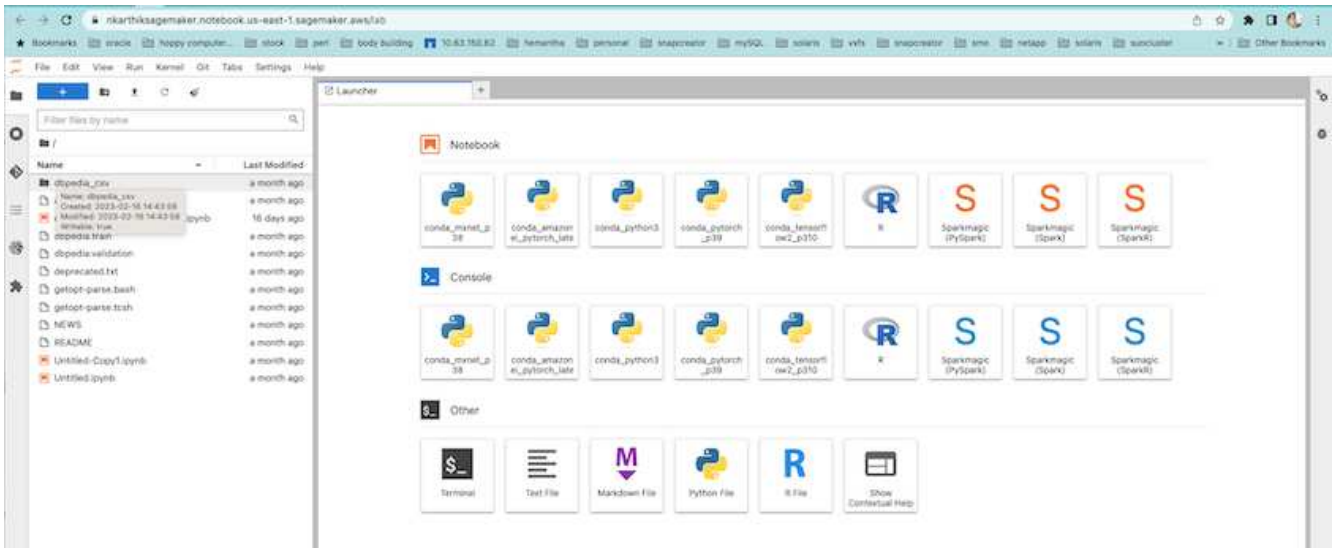




4. Avviare il notebook AWS.



5. Aprire il laboratorio Jupyter.



6. Accedere al terminale e montare il volume Cloud Volumes ONTAP.

```
sh-4.2$ sudo mkdir /vol1; sudo mount -t nfs 172.30.10.41:/vol1 /vol1
sh-4.2$ df -h
Filesystem          Size  Used Avail Use% Mounted on
devtmpfs            2.0G     0  2.0G   0% /dev
tmpfs               2.0G     0  2.0G   0% /dev/shm
tmpfs               2.0G  624K  2.0G   1% /run
tmpfs               2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/xvda1          140G  114G   27G  82% /
/dev/xvdf           4.8G   72K  4.6G   1% /home/ec2-user/SageMaker
tmpfs               393M     0  393M   0% /run/user/1001
tmpfs               393M     0  393M   0% /run/user/1002
tmpfs               393M     0  393M   0% /run/user/1000
172.30.10.41:/vol1  973M  189M  785M  20% /vol1
sh-4.2$
```

7. Controllare il bucket creato sul volume Cloud Volumes ONTAP utilizzando i comandi CLI AWS.

```
sh-4.2$ aws configure --profile netapp
AWS Access Key ID [None]: 0ZNAX21JW5Q8AP80CQ2E
AWS Secret Access Key [None]: PpLs4gA9K0_2gPhuykkp014gBjcC9Rbi3QDX_6rr
Default region name [None]: us-east-1
Default output format [None]:
sh-4.2$

sh-4.2$ aws s3 ls --profile netapp --endpoint-url
2023-02-10 17:59:48 ontapbucket1

sh-4.2$ aws s3 ls --profile netapp --endpoint-url  s3://ontapbucket1/


2023-02-10 18:46:44       4747 1
2023-02-10 18:48:32         96 setup.cfg

sh-4.2$
```

Dati per l'apprendimento automatico

In questa convalida, abbiamo utilizzato un set di dati di dbpedia, un'iniziativa della community basata su crowd, per estrarre contenuti strutturati dalle informazioni create in vari progetti Wikimedia.

1. Scaricare i dati dalla posizione di dbpedia GitHub ed estrarli. Utilizzare lo stesso terminale utilizzato nella sezione precedente.

```
sh-4.2$ wget
--2023-02-14 23:12:11--
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:  [following]
--2023-02-14 23:12:11--
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 68431223 (65M) [application/octet-stream]
Saving to: ‘dbpedia_csv.tar.gz’

100%[===================================================================
===================================================================
===================>] 68,431,223  56.2MB/s   in 1.2s

2023-02-14 23:12:13 (56.2 MB/s) - ‘dbpedia_csv.tar.gz’ saved
[68431223/68431223]

sh-4.2$ tar -zxvf dbpedia_csv.tar.gz
dbpedia_csv/
dbpedia_csv/test.csv
dbpedia_csv/classes.txt
dbpedia_csv/train.csv
dbpedia_csv/readme.txt
sh-4.2$
```

2. Copiare i dati nella posizione Cloud Volumes ONTAP e controllarli dal bucket S3 utilizzando l’interfaccia CLI AWS.

```
sh-4.2$ df -h
Filesystem          Size  Used Avail Use% Mounted on
devtmpfs            2.0G     0  2.0G   0% /dev
tmpfs               2.0G     0  2.0G   0% /dev/shm
tmpfs               2.0G  628K  2.0G   1% /run
tmpfs               2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/xvda1          140G  114G   27G  82% /
/dev/xvdf           4.8G   52K  4.6G   1% /home/ec2-user/SageMaker
tmpfs               393M     0  393M   0% /run/user/1002
tmpfs               393M     0  393M   0% /run/user/1001
tmpfs               393M     0  393M   0% /run/user/1000
172.30.10.41:/vol1  973M  384K  973M   1% /vol1
sh-4.2$ pwd
/home/ec2-user
sh-4.2$ cp -ra dbpedia_csv /vol1
sh-4.2$ aws s3 ls --profile netapp --endpoint-url  s3://ontapbucket1/
                           PRE dbpedia_csv/
2023-02-10 18:46:44       4747 1
2023-02-10 18:48:32         96 setup.cfg
sh-4.2$
```

3. Eseguire la convalida di base per assicurarsi che la funzionalità di lettura/scrittura funzioni sul bucket S3.

```
sh-4.2$ aws s3 cp  --profile netapp --endpoint-url  /usr/share/doc/util-
linux-2.30.2 s3://ontapbucket1/ --recursive
upload: ../../../usr/share/doc/util-linux-2.30.2/deprecated.txt to
s3://ontapbucket1/deprecated.txt
upload: ../../../usr/share/doc/util-linux-2.30.2/getopt-parse.bash to
s3://ontapbucket1/getopt-parse.bash
upload: ../../../usr/share/doc/util-linux-2.30.2/README to
s3://ontapbucket1/README
upload: ../../../usr/share/doc/util-linux-2.30.2/getopt-parse.tcsh to
s3://ontapbucket1/getopt-parse.tcsh
upload: ../../../usr/share/doc/util-linux-2.30.2/AUTHORS to
s3://ontapbucket1/AUTHORS
upload: ../../../usr/share/doc/util-linux-2.30.2/NEWS to
s3://ontapbucket1/NEWS
sh-4.2$ aws s3 ls --profile netapp --endpoint-url
s3://ontapbucket1/s3://ontapbucket1/

An error occurred (InternalError) when calling the ListObjectsV2
operation: We encountered an internal error. Please try again.
sh-4.2$ aws s3 ls --profile netapp --endpoint-url  s3://ontapbucket1/
                           PRE dbpedia_csv/
2023-02-16 19:19:27      26774 AUTHORS
```

```
2023-02-16 19:19:27      72727 NEWS
2023-02-16 19:19:27       4493 README
2023-02-16 19:19:27       2825 deprecated.txt
2023-02-16 19:19:27       1590 getopt-parse.bash
2023-02-16 19:19:27       2245 getopt-parse.tcsh
sh-4.2$ ls -ltr /vol1
total 132
drwxrwxr-x 2 ec2-user ec2-user  4096 Mar 29  2015 dbpedia_csv
-rw-r--r-- 1 nobody   nobody    2245 Apr 10 17:37 getopt-parse.tcsh
-rw-r--r-- 1 nobody   nobody    2825 Apr 10 17:37 deprecated.txt
-rw-r--r-- 1 nobody   nobody    4493 Apr 10 17:37 README
-rw-r--r-- 1 nobody   nobody    1590 Apr 10 17:37 getopt-parse.bash
-rw-r--r-- 1 nobody   nobody   26774 Apr 10 17:37 AUTHORS
-rw-r--r-- 1 nobody   nobody   72727 Apr 10 17:37 NEWS
sh-4.2$ ls -ltr /vol1/dbpedia_csv/
total 192104
-rw------- 1 ec2-user ec2-user 174148970 Mar 28  2015 train.csv
-rw------- 1 ec2-user ec2-user  21775285 Mar 28  2015 test.csv
-rw------- 1 ec2-user ec2-user       146 Mar 28  2015 classes.txt
-rw-rw-r-- 1 ec2-user ec2-user      1758 Mar 29  2015 readme.txt
sh-4.2$ chmod -R 777 /vol1/dbpedia_csv
sh-4.2$ ls -ltr /vol1/dbpedia_csv/
total 192104
-rwxrwxrwx 1 ec2-user ec2-user 174148970 Mar 28  2015 train.csv
-rwxrwxrwx 1 ec2-user ec2-user  21775285 Mar 28  2015 test.csv
-rwxrwxrwx 1 ec2-user ec2-user       146 Mar 28  2015 classes.txt
-rwxrwxrwx 1 ec2-user ec2-user      1758 Mar 29  2015 readme.txt
sh-4.2$ aws s3 cp --profile netapp --endpoint-url http://172.30.2.248/
s3://ontapbucket1/ /tmp --recursive
download: s3://ontapbucket1/AUTHORS to ../../tmp/AUTHORS
download: s3://ontapbucket1/README to ../../tmp/README
download: s3://ontapbucket1/NEWS to ../../tmp/NEWS
download: s3://ontapbucket1/dbpedia_csv/classes.txt to
../../tmp/dbpedia_csv/classes.txt
download: s3://ontapbucket1/dbpedia_csv/readme.txt to
../../tmp/dbpedia_csv/readme.txt
download: s3://ontapbucket1/deprecated.txt to ../../tmp/deprecated.txt
download: s3://ontapbucket1/getopt-parse.bash to ../../tmp/getopt-
parse.bash
download: s3://ontapbucket1/getopt-parse.tcsh to ../../tmp/getopt-
parse.tcsh
download: s3://ontapbucket1/dbpedia_csv/test.csv to
../../tmp/dbpedia_csv/test.csv
download: s3://ontapbucket1/dbpedia_csv/train.csv to
../../tmp/dbpedia_csv/train.csv
sh-4.2$
```

```
sh-4.2$ aws s3 ls --profile netapp --endpoint-url  s3://ontapbucket1/
                           PRE dbpedia_csv/
2023-02-16 19:19:27      26774 AUTHORS
2023-02-16 19:19:27      72727 NEWS
2023-02-16 19:19:27       4493 README
2023-02-16 19:19:27       2825 deprecated.txt
2023-02-16 19:19:27       1590 getopt-parse.bash
2023-02-16 19:19:27       2245 getopt-parse.tcsh
sh-4.2$
```

Convalida l’apprendimento automatico dai notebook Jupyter

La seguente convalida fornisce i modelli di creazione, formazione e implementazione dell’apprendimento automatico attraverso la classificazione del testo utilizzando l’esempio di SageMaker BlazingText riportato di seguito:

1. Installare i pacchetti boto3 e SageMaker.

   ```
   In [1]:  pip install --upgrade boto3 sagemaker
   ```

   Uscita:

   ```
   Looking in indexes: https://pypi.org/simple,
   https://pip.repos.neuron.amazo naws.com
   Requirement already satisfied: boto3 in /home/ec2-
   user/anaconda3/envs/pytho n3/lib/python3.10/site-packages (1.26.44)
   Collecting boto3
     Downloading boto3-1.26.72-py3-none-any.whl (132 kB)
        □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 132.7/132.7 kB 14.6 MB/s eta
   0: 00:00
   Requirement already satisfied: sagemaker in /home/ec2-
   user/anaconda3/envs/p ython3/lib/python3.10/site-packages (2.127.0)
   Collecting sagemaker
     Downloading sagemaker-2.132.0.tar.gz (668 kB)
        □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 668.0/668.0 kB 12.3 MB/s eta
   0:
   00:0000:01
     Preparing metadata (setup.py) ... done
   Collecting botocore<1.30.0,>=1.29.72
     Downloading botocore-1.29.72-py3-none-any.whl (10.4 MB)
        □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 10.4/10.4 MB 44.3 MB/s eta
   0: 00:0000:010:01
   Requirement already satisfied: s3transfer<0.7.0,>=0.6.0 in /home/ec2-
   user/a naconda3/envs/python3/lib/python3.10/site-packages (from boto3)
   (0.6.0)
   ```

```
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /home/ec2-
user/ana conda3/envs/python3/lib/python3.10/site-packages (from boto3)
(0.10.0)
Requirement already satisfied: attrs<23,>=20.3.0 in /home/ec2-
user/anaconda
3/envs/python3/lib/python3.10/site-packages (from sagemaker) (22.1.0)
Requirement already satisfied: google-pasta in /home/ec2-
user/anaconda3/env s/python3/lib/python3.10/site-packages (from
sagemaker) (0.2.0)
Requirement already satisfied: numpy<2.0,>=1.9.0 in /home/ec2-
user/anaconda
3/envs/python3/lib/python3.10/site-packages (from sagemaker) (1.22.4)
Requirement already satisfied: protobuf<4.0,>=3.1 in /home/ec2-
user/anacond a3/envs/python3/lib/python3.10/site-packages (from
sagemaker) (3.20.3)
Requirement already satisfied: protobuf3-to-dict<1.0,>=0.1.5 in
/home/ec2-u ser/anaconda3/envs/python3/lib/python3.10/site-packages
(from sagemaker)
(0.1.5)
Requirement already satisfied: smdebug_rulesconfig==1.0.1 in /home/ec2-
use r/anaconda3/envs/python3/lib/python3.10/site-packages (from
sagemaker) (1.
0.1) Requirement already satisfied: importlib-metadata<5.0,>=1.4.0 in
/home/ec2user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sagemaker)
(4.13.0)
Requirement already satisfied: packaging>=20.0 in /home/ec2-
user/anaconda3/ envs/python3/lib/python3.10/site-packages (from
sagemaker) (21.3)
Requirement already satisfied: pandas in /home/ec2-
user/anaconda3/envs/pyth on3/lib/python3.10/site-packages (from
sagemaker) (1.5.1)
Requirement already satisfied: pathos in /home/ec2-
user/anaconda3/envs/pyth on3/lib/python3.10/site-packages (from
sagemaker) (0.3.0)
Requirement already satisfied: schema in /home/ec2-
user/anaconda3/envs/pyth on3/lib/python3.10/site-packages (from
sagemaker) (0.7.5) Requirement already satisfied: python-
dateutil<3.0.0,>=2.1 in /home/ec2-use
r/anaconda3/envs/python3/lib/python3.10/site-packages (from
botocore<1.30.
0,>=1.29.72->boto3) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/ec2-
user/anac onda3/envs/python3/lib/python3.10/site-packages (from
botocore<1.30.0,>=1.2
9.72->boto3) (1.26.8) Requirement already satisfied: zipp>=0.5 in
```

```
/home/ec2-user/anaconda3/envs/p ython3/lib/python3.10/site-packages
(from importlib-metadata<5.0,>=1.4.0->s agemaker) (3.10.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /home/ec2-
user/a naconda3/envs/python3/lib/python3.10/site-packages (from
packaging>=20.0->s agemaker) (3.0.9)
Requirement already satisfied: six in /home/ec2-
user/anaconda3/envs/python
3/lib/python3.10/site-packages (from protobuf3-to-dict<1.0,>=0.1.5-
>sagemak er) (1.16.0)
Requirement already satisfied: pytz>=2020.1 in /home/ec2-
user/anaconda3/env s/python3/lib/python3.10/site-packages (from pandas-
>sagemaker) (2022.5)
Requirement already satisfied: ppft>=1.7.6.6 in /home/ec2-
user/anaconda3/en vs/python3/lib/python3.10/site-packages (from pathos-
>sagemaker) (1.7.6.6) Requirement already satisfied:
multiprocess>=0.70.14 in /home/ec2-user/anac
onda3/envs/python3/lib/python3.10/site-packages (from pathos->sagemaker)
(0.70.14)
Requirement already satisfied: dill>=0.3.6 in /home/ec2-
user/anaconda3/env s/python3/lib/python3.10/site-packages (from pathos-
>sagemaker) (0.3.6)
Requirement already satisfied: pox>=0.3.2 in /home/ec2-
user/anaconda3/envs/ python3/lib/python3.10/site-packages (from pathos-
>sagemaker) (0.3.2) Requirement already satisfied: contextlib2>=0.5.5 in
/home/ec2-user/anacond a3/envs/python3/lib/python3.10/site-packages
(from schema->sagemaker) (21.
6.0) Building wheels for collected packages: sagemaker
  Building wheel for sagemaker (setup.py) ... done
  Created wheel for sagemaker: filename=sagemaker-2.132.0-py2.py3-none-
any. whl size=905449
sha256=f6100a5dc95627f2e2a49824e38f0481459a27805ee19b5a06ec
83db0252fd41
  Stored in directory: /home/ec2-
user/.cache/pip/wheels/60/41/b6/482e7ab096
520df034fbf2dddd244a1d7ba0681b27ef45aa61
Successfully built sagemaker
Installing collected packages: botocore, boto3, sagemaker
  Attempting uninstall: botocore     Found existing installation:
botocore 1.24.19
    Uninstalling botocore-1.24.19:       Successfully uninstalled
botocore-1.24.19
  Attempting uninstall: boto3     Found existing installation: boto3
1.26.44
    Uninstalling boto3-1.26.44:
      Successfully uninstalled boto3-1.26.44
  Attempting uninstall: sagemaker     Found existing installation:
```

```
sagemaker 2.127.0
    Uninstalling sagemaker-2.127.0:
      Successfully uninstalled sagemaker-2.127.0
ERROR: pip's dependency resolver does not currently take into account
all t he packages that are installed. This behaviour is the source of
the followi ng dependency conflicts.
awscli 1.27.44 requires botocore==1.29.44, but you have botocore 1.29.72
wh ich is incompatible.
aiobotocore 2.0.1 requires botocore<1.22.9,>=1.22.8, but you have
botocore 1.29.72 which is incompatible. Successfully installed boto3-
1.26.72 botocore-1.29.72 sagemaker-2.132.0 Note: you may need to restart
the kernel to use updated packages.
```

2. Nella fase successiva, i dati (`dbpedia_csv`) viene scaricato dal bucket s3 `ontapbucket1` A un'istanza Jupyter notebook utilizzata nell'apprendimento automatico.

```
In [2]: import sagemaker
In [3]: from sagemaker import get_execution_role
In [4]:
import json
import boto3
sess = sagemaker.Session()
role = get_execution_role()
print(role)
bucket = "ontapbucket1"
print(bucket)
sess.s3_client = boto3.client('s3',region_name='',aws_access_key_id =
'0ZNAX21JW5Q8AP80CQ2E',  aws_secret_access_key =
'PpLs4gA9K0_2gPhuykkp014gBjcC9Rbi3QDX_6rr',
                              use_ssl = False, endpoint_url =
'http://172.30.10.41',

config=boto3.session.Config(signature_version='s3v4',
s3={'addressing_style':'path'}) )
sess.s3_resource = boto3.resource('s3',region_name='',aws_access_key_id
= '0ZNAX21JW5Q8AP80CQ2E', aws_secret_access_key =
'PpLs4gA9K0_2gPhuykkp014gBjcC9Rbi3QDX_6rr',
                              use_ssl = False, endpoint_url =
'http://172.30.10.41',

config=boto3.session.Config(signature_version='s3v4',
s3={'addressing_style':'path'}) )
prefix = "blazingtext/supervised"
import os
my_bucket = sess.s3_resource.Bucket(bucket)
my_bucket = sess.s3_resource.Bucket(bucket)
#os.mkdir('dbpedia_csv')
for s3_object in my_bucket.objects.all():
    filename = s3_object.key
#    print(filename)
#    print(s3_object.key)
    my_bucket.download_file(s3_object.key, filename)
```

3. Il codice seguente crea il mapping tra gli indici interi e le etichette delle classi utilizzate per recuperare il nome effettivo della classe durante l'inferenza.

```
index_to_label = {}
with open("dbpedia_csv/classes.txt") as f:
    for i,label in enumerate(f.readlines()):
        index_to_label[str(i + 1)] = label.strip()
```

L'output elenca i file e le cartelle in `ontapbucket1` Bucket utilizzati come dati per la convalida dell'apprendimento automatico AWS SageMaker.

```
arn:aws:iam::210811600188:role/SageMakerFullRole ontapbucket1
AUTHORS
AUTHORS
NEWS
NEWS
README README
dbpedia_csv/classes.txt dbpedia_csv/classes.txt dbpedia_csv/readme.txt
dbpedia_csv/readme.txt dbpedia_csv/test.csv dbpedia_csv/test.csv
dbpedia_csv/train.csv dbpedia_csv/train.csv deprecated.txt
deprecated.txt getopt-parse.bash getopt-parse.bash getopt-parse.tcsh
getopt-parse.tcsh
In [5]: ls
AUTHORS       deprecated.txt     getopt-parse.tcsh  NEWS
Untitled.ipynb dbpedia_csv/  getopt-parse.bash  lost+found/
README
In [6]: ls -l dbpedia_csv
total 191344
-rw-rw-r-- 1 ec2-user ec2-user       146 Feb 16 19:43 classes.txt
-rw-rw-r-- 1 ec2-user ec2-user      1758 Feb 16 19:43 readme.txt
-rw-rw-r-- 1 ec2-user ec2-user  21775285 Feb 16 19:43 test.csv
-rw-rw-r-- 1 ec2-user ec2-user 174148970 Feb 16 19:43 train.csv
```

4. Avviare la fase di pre-elaborazione dei dati per pre-elaborare i dati di training in un formato di testo tobenizzato, separato dallo spazio, che può essere utilizzato dall'algoritmo BlazingText e dalla libreria nltk per mettere in token le frasi di input dal set di dati dbpedia. Scarica il token nltk e altre librerie. Il `transform_instance` Applicato a ogni istanza di dati in parallelo utilizza il modulo multiprocessing Python.

```
ln [7]: from random import shuffle
import multiprocessing
from multiprocessing import Pool
import csv
import nltk
nltk.download("punkt")
def transform_instance(row):
    cur_row = []
    label ="__label__" + index_to_label [row[0]] # Prefix the index-ed
label with __label__
    cur_row.append (label)
    cur_row.extend(nltk.word_tokenize(row[1].lower ()))
    cur_row.extend(nltk.word_tokenize(row[2].lower ()))
    return cur_row
def preprocess(input_file, output_file, keep=1):
```

```
    all_rows = []
    with open(input_file,"r") as csvinfile:
        csv_reader = csv.reader(csvinfile, delimiter=",")
        for row in csv_reader:
            all_rows.append(row)
    shuffle(all_rows)
    all_rows = all_rows[: int(keep * len(all_rows))]
    pool = Pool(processes=multiprocessing.cpu_count())
    transformed_rows = pool.map(transform_instance, all_rows)
    pool.close()
    pool. join()
    with open(output_file, "w") as csvoutfile:
        csv_writer = csv.writer (csvoutfile, delimiter=" ",
lineterminator="\n")
        csv_writer.writerows (transformed_rows)

# Preparing the training dataset
# since preprocessing the whole dataset might take a couple of minutes,
# we keep 20% of the training dataset for this demo.
# Set keep to 1 if you want to use the complete dataset
preprocess("dbpedia_csv/train.csv","dbpedia.train", keep=0.2)
# Preparing the validation dataset
preprocess("dbpedia_csv/test.csv","dbpedia.validation")
sess = sagemaker.Session()
role = get_execution_role()
print (role) # This is the role that sageMaker would use to leverage Aws
resources (S3,  Cloudwatch) on your behalf
bucket = sess.default_bucket() # Replace with your own bucket name if
needed
print("default Bucket::: ")
print(bucket)
```

Uscita:

```
[nltk_data] Downloading package punkt to /home/ec2-user/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
arn:aws:iam::210811600188:role/SageMakerFullRole default Bucket:::
sagemaker-us-east-1-210811600188
```

5. Caricare il set di dati formattato e formativo in S3 in modo che possa essere utilizzato da SageMaker per eseguire i lavori di training. Quindi caricare due file nel bucket e nella posizione del prefisso utilizzando l'SDK Python.

```
ln [8]: %%time
train_channel = prefix + "/train"
validation_channel = prefix + "/validation"
sess.upload_data(path="dbpedia.train", bucket=bucket,
key_prefix=train_channel)
sess.upload_data(path="dbpedia.validation", bucket=bucket,
key_prefix=validation_channel)
s3_train_data = "s3://{}/{}".format(bucket, train_channel)
s3_validation_data = "s3://{}/{}".format(bucket, validation_channel)
```

Uscita:

```
CPU times: user 546 ms, sys: 163 ms, total: 709 ms
Wall time: 1.32 s
```

6. Impostare una posizione di output su S3 in cui viene caricato l'artefatto del modello in modo che gli artefatti possano essere l'output del lavoro di training dell'algoritmo. Creare un `sageMaker.estimator.Estimator` oggetto per avviare il lavoro di training.

```
In [9]: s3_output_location = "s3://{}/{}/output".format(bucket, prefix)
In [10]: region_name = boto3.Session().region_name
In [11]: container =
sagemaker.amazon.amazon_estimator.get_image_uri(region_name,
"blazingtext","latest")
print("Using SageMaker BlazingText container: {} ({})".format(container,
region_name))
```

Uscita:

```
The method get_image_uri has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
Defaulting to the only supported framework/algorithm version: 1.
Ignoring f ramework/algorithm version: latest.
Using SageMaker BlazingText container: 811284229777.dkr.ecr.us-east-
1.amazo naws.com/blazingtext:1 (us-east-1)
```

7. Definire il SageMaker `Estrimator` Con configurazioni delle risorse e hyperparameters per formare la classificazione del testo nel dataset dbpedia utilizzando la modalità supervisionata su un'istanza c4.4xlarge.

```
In [12]: bt_model = sagemaker.estimator.Estimator(
container,
role,
instance_count=1,
instance_type="ml.c4.4xlarge",
volume_size=30,
max_run=360000,
input_mode="File",
output_path=s3_output_location,
hyperparameters={
        "mode": "supervised",
        "epochs": 1,
        "min_count": 2,
        "learning_rate": 0.05,
        "vector_dim": 10,
        "early_stopping": True,
        "patience": 4,
        "min_epochs": 5,
        "word_ngrams": 2,
 },
     )
```

8. Preparare un handshake tra i canali dati e l'algoritmo. A tale scopo, creare `sagemaker.session.s3_input` oggetti dei canali dati e conservarli in un dizionario che l'algoritmo deve utilizzare.

```
ln [13]: train_data = sagemaker.inputs.TrainingInput(
    s3_train_data,
    distribution="FullyReplicated",
    content_type="text/plain",
    s3_data_type="S3Prefix",
)
validation_data = sagemaker.inputs.TrainingInput(
    s3_validation_data,
    distribution="FullyReplicated",
    content_type="text/plain",
    s3_data_type="S3Prefix",
)
data_channels = {"train": train_data, "validation": validation_data}
```

9. Al termine del lavoro, viene visualizzato il messaggio lavoro completato. Il modello addestrato si trova nel bucket S3 configurato come `output_path` nello stimatore.

```
ln [14]: bt_model.fit(inputs=data_channels, logs=True)
```

Uscita:

```
INFO:sagemaker:Creating training-job with name: blazingtext-2023-02-16-
20-3
7-30-748
2023-02-16 20:37:30 Starting - Starting the training job......
2023-02-16 20:38:09 Starting - Preparing the instances for
training......
2023-02-16 20:39:24 Downloading - Downloading input data
2023-02-16 20:39:24 Training - Training image download completed.
Training in progress... Arguments: train
[02/16/2023 20:39:41 WARNING 140279908747072] Loggers have already been
set up. [02/16/2023 20:39:41 WARNING 140279908747072] Loggers have
already been set up.
[02/16/2023 20:39:41 INFO 140279908747072] nvidia-smi took:
0.0251793861389
16016 secs to identify 0 gpus
[02/16/2023 20:39:41 INFO 140279908747072] Running single machine CPU
Blazi ngText training using supervised mode.
Number of CPU sockets found in instance is  1
[02/16/2023 20:39:41 INFO 140279908747072] Processing
/opt/ml/input/data/tr ain/dbpedia.train . File size: 35.0693244934082 MB
[02/16/2023 20:39:41 INFO 140279908747072] Processing
/opt/ml/input/data/va lidation/dbpedia.validation . File size:
21.887572288513184 MB
Read 6M words
Number of words:  149301
Loading validation data from
/opt/ml/input/data/validation/dbpedia.validati on
Loaded validation data.
-------------- End of epoch: 1 ##### Alpha: 0.0000  Progress: 100.00%
Million Words/sec: 10.39 ##### Training finished.
Average throughput in Million words/sec: 10.39
Total training time in seconds: 0.60
#train_accuracy: 0.7223
Number of train examples: 112000
#validation_accuracy: 0.7205
Number of validation examples: 70000
2023-02-16 20:39:55 Uploading - Uploading generated training model
2023-02-16 20:40:11 Completed - Training job completed
Training seconds: 68
Billable seconds: 68
```

10. Una volta completato il training, implementa il modello addestrato come endpoint in hosting in tempo reale Amazon SageMaker per fare previsioni.

```
In [15]: from sagemaker.serializers import JSONSerializer
 text_classifier = bt_model.deploy(
     initial_instance_count=1, instance_type="ml.m4.xlarge",
serializer=JSONS
)
```

Uscita:

```
INFO:sagemaker:Creating model with name: blazingtext-2023-02-16-20-41-
33-10
0
INFO:sagemaker:Creating endpoint-config with name blazingtext-2023-02-
16-20
-41-33-100
INFO:sagemaker:Creating endpoint with name blazingtext-2023-02-16-20-41-
33-
100
-------!
```

```
In [16]: sentences = [
    "Convair was an american aircraft manufacturing company which later
expanded into rockets and spacecraft.",
       "Berwick secondary college is situated in the outer melbourne
metropolitan suburb of berwick .",
]
# using the same nltk tokenizer that we used during data preparation for
training
tokenized_sentences = [" ".join(nltk.word_tokenize(sent)) for sent in
sentences]
payload = {"instances": tokenized_sentences} response =
text_classifier.predict(payload)
predictions = json.loads(response)
print(json.dumps(predictions, indent=2))
```

```
[
  {
    "label": [
      "__label__Artist"
    ],
    "prob": [
      0.4090951681137085
    ]
  },
  {
    "label": [
      "__label__EducationalInstitution"
    ],
    "prob": [
      0.49466073513031006
    ]
  }
]
```

11. Per impostazione predefinita, il modello restituisce una previsione con la maggiore probabilità. Per recuperare la parte superiore `k` previsioni, set `k` nel file di configurazione.

```
In [17]: payload = {"instances": tokenized_sentences, "configuration":
{"k": 2}}
 response = text_classifier.predict(payload)

 predictions = json.loads(response)
 print(json.dumps(predictions, indent=2))
```

```
[
  {
    "label": [
      "__label__Artist",
      "__label__MeanOfTransportation"
    ],
    "prob": [
      0.4090951681137085,
      0.26930734515190125
    ]
  },
  {
    "label": [
      "__label__EducationalInstitution",
      "__label__Building"
    ],
    "prob": [
      0.49466073513031006,
      0.15817692875862122
    ]
  }
]
```

12. Eliminare l'endpoint prima di chiudere il notebook.

```
In [18]: sess.delete_endpoint(text_classifier.endpoint)
WARNING:sagemaker.deprecations:The endpoint attribute has been renamed
in s agemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
INFO:sagemaker:Deleting endpoint with name: blazingtext-2023-02-16-20-
41-33
-100
```

Conclusione

In base a questa convalida, i data scientist e i tecnici possono accedere ai dati NFS dai notebook AWS SageMaker Jupyter tramite i bucket S3 di NetApp Cloud Volumes ONTAP. Questo approccio consente un facile accesso e condivisione degli stessi dati da NFS e S3 senza la necessità di software aggiuntivo.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Classificazione del testo con SageMaker BlazingText

  "https://sagemaker-examples.readthedocs.io/en/latest/introduction_to_amazon_algorithms/blazingtext_text_classification_dbpedia/blazingtext_text_classification_dbpedia.html"

- Supporto della versione di ONTAP per lo storage a oggetti S3

  "https://docs.netapp.com/us-en/ontap/s3-config/ontap-version-support-s3-concept.html"

Carichi di lavoro Apache Kafka con storage NetApp NFS

TR-4947: Carico di lavoro Apache Kafka con storage NetApp NFS - convalida funzionale e performance

Shantanu Chakole, Karthikeyan Nagalingam e Joe Scott, NetApp

Kafka è un sistema di messaggistica distribuito publish-subscribe con una solida coda in grado di accettare grandi quantità di dati dei messaggi. Con Kafka, le applicazioni possono scrivere e leggere i dati su argomenti in modo molto rapido. A causa della sua tolleranza agli errori e della sua scalabilità, Kafka viene spesso utilizzato nel grande spazio di dati come un modo affidabile per acquisire e spostare molti flussi di dati molto rapidamente. I casi di utilizzo includono elaborazione dello streaming, monitoraggio delle attività del sito Web, raccolta e monitoraggio delle metriche, aggregazione dei log, analisi in tempo reale e così via.

Anche se le normali operazioni di Kafka su NFS funzionano bene, il "ridenominazione sciocco" Il problema blocca l'applicazione durante il ridimensionamento o la partizione di un cluster Kafka in esecuzione su NFS. Si tratta di un problema significativo perché un cluster Kafka deve essere ridimensionato o ripartizionato a scopo di bilanciamento del carico o di manutenzione. Ulteriori dettagli sono disponibili "qui".

Questo documento descrive i seguenti argomenti:

- Il problema di ridenominazione e la convalida della soluzione
- Riduzione dell'utilizzo della CPU per ridurre i tempi di attesa i/O.
- Tempi di recovery più rapidi per i broker Kafka
- Performance nel cloud e on-premise

Perché utilizzare lo storage NFS per i workload Kafka?

I carichi di lavoro Kafka nelle applicazioni di produzione possono eseguire lo streaming di enormi quantità di dati tra le applicazioni. Questi dati vengono conservati e memorizzati nei nodi del broker Kafka nel cluster Kafka. Kafka è nota anche per la disponibilità e il parallelismo, che si ottiene suddividendo gli argomenti in partizioni e replicando le partizioni in tutto il cluster. Ciò significa che l'enorme quantità di dati che scorre attraverso un cluster Kafka è generalmente moltiplicata per dimensioni. NFS rende il ribilanciamento dei dati con il variare del numero di broker molto rapido e semplice. Per ambienti di grandi dimensioni, ribilanciamento dei dati in DAS quando il numero di broker cambia è molto lungo e, nella maggior parte degli ambienti Kafka, il numero di broker cambia frequentemente.

Altri vantaggi includono:

- **Maturità.** NFS è un protocollo maturo, il che significa che la maggior parte degli aspetti dell'implementazione, della protezione e dell'utilizzo sono ben noti.
- **Open.** NFS è un protocollo aperto e il suo continuo sviluppo è documentato nelle specifiche Internet come protocollo di rete libero e aperto.
- * Conveniente.* NFS è una soluzione a basso costo per la condivisione di file di rete facile da configurare perché utilizza l'infrastruttura di rete esistente.
- **Gestione centralizzata.** la gestione centralizzata di NFS riduce la necessità di aggiungere software e spazio su disco nei singoli sistemi utente.
- **Distributed.** NFS può essere utilizzato come file system distribuito, riducendo la necessità di dispositivi di storage su supporti rimovibili.

Perché scegliere NetApp per i workload Kafka?

L'implementazione NetApp NFS è considerata uno standard di riferimento per il protocollo e viene utilizzata in innumerevoli ambienti NAS aziendali. Oltre alla credibilità di NetApp, offre anche i seguenti vantaggi:

- Affidabilità ed efficienza
- Scalabilità e performance
- Alta disponibilità (partner ha in un cluster NetApp ONTAP)
- Protezione dei dati
  - **Disaster Recovery (NetApp SnapMirror).** il tuo sito non funziona o vuoi partire da un altro sito e continuare da dove hai interrotto.
  - Gestibilità del sistema storage (amministrazione e gestione con NetApp OnCommand).
  - **Bilanciamento del carico.** il cluster consente di accedere a volumi diversi da file di dati LIF ospitati su nodi diversi.
  - **Operazioni senza interruzioni.** le LIF o gli spostamenti dei volumi sono trasparenti per i client NFS.

Soluzione NetApp per problemi di ridenominazione sciocco per carichi di lavoro da NFS a Kafka

Kafka si basa sul presupposto che il file system sottostante sia conforme a POSIX, ad esempio XFS o Ext4. Il ribilanciamento delle risorse Kafka rimuove i file mentre l'applicazione li sta ancora utilizzando. Un file system conforme a POSIX consente di procedere con l'annullamento del collegamento. Tuttavia, il file viene rimosso solo dopo che tutti i riferimenti al file sono stati rimossi. Se il file system sottostante è collegato in rete, il client NFS intercetta le chiamate di sconnessione e gestisce il flusso di lavoro. Poiché il file non è collegato, il client NFS invia una richiesta di ridenominazione al server NFS e, all'ultima chiusura del file non collegato, effettua un'operazione di rimozione del file rinominato. Questo comportamento viene comunemente definito come ridenominazione sciocco di NFS ed è orchestrato dal client NFS.

Qualsiasi broker Kafka che utilizza lo storage da un server NFSv3 si verifica in problemi a causa di questo comportamento. Tuttavia, il protocollo NFSv4.x dispone di funzionalità per risolvere questo problema consentendo al server di assumersi la responsabilità dei file aperti e non collegati. I server NFS che supportano questa funzionalità opzionale comunicano la proprietà al client NFS al momento dell'apertura del file. Il client NFS interrompe quindi la gestione di unlink quando vengono aperte le porte in sospeso e consente al server di gestire il flusso. Sebbene la specifica NFSv4 fornisca linee guida per l'implementazione, fino ad ora

non esistevano implementazioni server NFS note che supportassero questa funzionalità opzionale.

Le seguenti modifiche sono necessarie per consentire al server NFS e al client NFS di risolvere il problema del ridenominazione sciocco:

- **Modifiche al client NFS (Linux).** al momento dell'apertura del file, il server NFS risponde con un flag, che indica la capacità di gestire lo scollegamento dei file aperti. Le modifiche al lato client NFS consentono al server NFS di gestire lo scollegamento in presenza del flag. NetApp ha aggiornato il client NFS Linux open-source con queste modifiche. Il client NFS aggiornato è ora generalmente disponibile in RHEL8.7 e RHEL9.1.
- **Modifiche al server NFS.** il server NFS tiene traccia dell'apertura. Lo scollegamento su un file aperto esistente è ora gestito dal server per corrispondere alla semantica POSIX. Quando l'ultima apertura viene chiusa, il server NFS avvia la rimozione effettiva del file ed evita così il processo di ridenominazione sciocco. Il server NFS di ONTAP ha implementato questa funzionalità nella sua ultima versione, ONTAP 9.12.1.

Con le suddette modifiche al client e al server NFS, Kafka può sfruttare in tutta sicurezza tutti i vantaggi dello storage NFS collegato alla rete.

Convalida funzionale - correzione del ridenominazione senza problemi

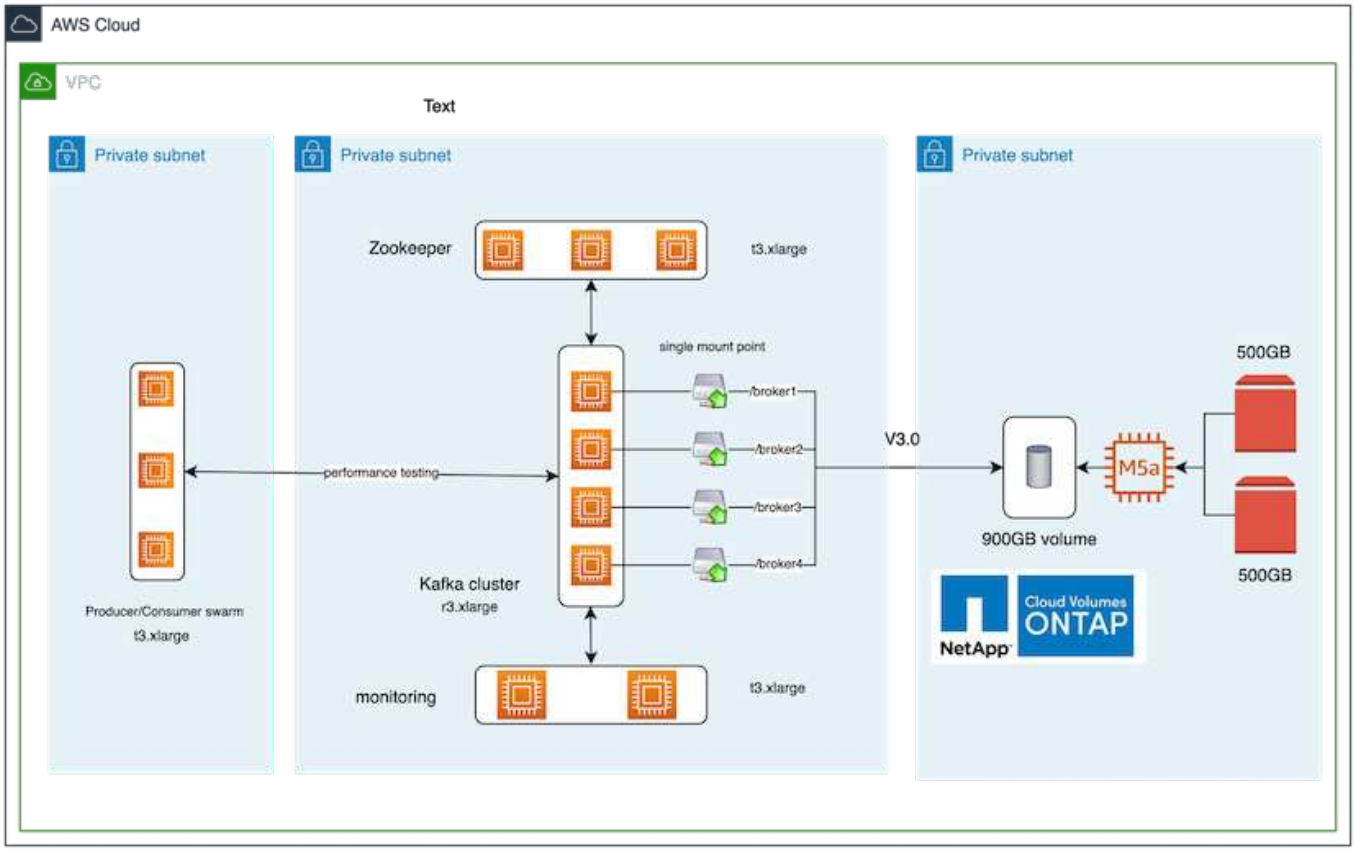
Per la convalida funzionale, abbiamo mostrato che un cluster Kafka con un montaggio NFSv3 per lo storage non esegue operazioni Kafka come la ridistribuzione delle partizioni, mentre un altro cluster montato su NFSv4 con la correzione può eseguire le stesse operazioni senza interruzioni.

Configurazione della convalida

La configurazione viene eseguita su AWS. La seguente tabella mostra i diversi componenti della piattaforma e la configurazione ambientale utilizzati per la convalida.

Componente della piattaforma	Configurazione dell'ambiente
Confluent Platform versione 7.2.1	• 3 zookeeper – t3.xlarge • 4 server di broker – r3.xlarge • 1 x Grafana – t3.xlarge • 1 centro di controllo – t3.xlarge • 3 x Produttore/consumatore
Sistema operativo su tutti i nodi	RHEL8.7o versione successiva
Istanza di NetApp Cloud Volumes ONTAP	Istanza a nodo singolo – M5.2xLarge

La figura seguente mostra la configurazione architetturale per questa soluzione.



Flusso architettonico

- **Compute.** abbiamo utilizzato un cluster Kafka a quattro nodi con un ensemble di zookeeper a tre nodi in esecuzione su server dedicati.
- **Monitoring.** abbiamo utilizzato due nodi per una combinazione Prometheus-Grafana.
- **Workload.** per la generazione dei workload, abbiamo utilizzato un cluster a tre nodi separato in grado di produrre e consumare da questo cluster Kafka.
- **Storage.** abbiamo utilizzato un'istanza NetApp Cloud Volumes ONTAP a nodo singolo con due volumi AWS-EBS GP2 da 500 GB collegati all'istanza. Questi volumi sono stati quindi esposti al cluster Kafka come singolo volume NFSv4.1 attraverso un LIF.

Le proprietà predefinite di Kafka sono state scelte per tutti i server. Lo stesso è stato fatto per lo sciame dello zooteeper.

Metodologia di test

1. Aggiornare `-is-preserve-unlink-enabled true` al volume kafka, come segue:

```
aws-shantanclastrecall-aws::*> volume create -vserver kafka_svm -volume
kafka_fg_vol01 -aggregate kafka_aggr -size 3500GB -state online -policy
kafka_policy -security-style unix -unix-permissions 0777 -junction-path
/kafka_fg_vol01 -type RW -is-preserve-unlink-enabled true
[Job 32] Job succeeded: Successful
```

2. Sono stati creati due cluster Kafka simili con la seguente differenza:
   - **Cluster 1.** il server NFS v4.1 di back-end con ONTAP versione 9.12.1 pronto per la produzione è stato ospitato da un'istanza CVO di NetApp. RHEL 8.7/RHEL 9.1 è stato installato sui broker.
   - **Cluster 2.** il server NFS back-end era un server Linux NFSv3 generico creato manualmente.
3. È stato creato un argomento dimostrativo su entrambi i cluster Kafka.

   Cluster 1:

```
[root@ip-172-30-0-160 demo]# kafka-topics --bootstrap-server=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:90
92,172.30.0.123:9092 --describe --topic __a_demo_topic
Topic: __a_demo_topic   TopicId: 2ty29xfhQLq65HKsUQv-pg PartitionCount: 4       ReplicationFactor: 2    Configs:
 min.insync.replicas=1,segment.bytes=1073741824
        Topic: __a_demo_topic   Partition: 0    Leader: 4       Replicas: 4,1   Isr: 4,1        Offline:
        Topic: __a_demo_topic   Partition: 1    Leader: 2       Replicas: 2,4   Isr: 2,4        Offline:
        Topic: __a_demo_topic   Partition: 2    Leader: 3       Replicas: 3,2   Isr: 3,2        Offline:
        Topic: __a_demo_topic   Partition: 3    Leader: 1       Replicas: 1,3   Isr: 1,3        Offline:
```

   Cluster 2:

```
[root@ip-172-30-0-198 demo]# kafka-topics --bootstrap-server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:90
92,172.30.0.204:9092 --describe --topic __a_demo_topic
Topic: __a_demo_topic   TopicId: AwQpsZTQShyeMIhaquCG3Q PartitionCount: 4       ReplicationFactor: 2    Configs:
 min.insync.replicas=1,segment.bytes=1073741824
        Topic: __a_demo_topic   Partition: 0    Leader: 2       Replicas: 2,3   Isr: 2,3        Offline:
        Topic: __a_demo_topic   Partition: 1    Leader: 3       Replicas: 3,1   Isr: 3,1        Offline:
        Topic: __a_demo_topic   Partition: 2    Leader: 1       Replicas: 1,4   Isr: 1,4        Offline:
        Topic: __a_demo_topic   Partition: 3    Leader: 4       Replicas: 4,2   Isr: 4,2        Offline:
```

4. I dati sono stati caricati in questi nuovi argomenti creati per entrambi i cluster. Questa operazione è stata eseguita utilizzando il toolkit Producer-perf-test incluso nel pacchetto predefinito di Kafka:

```
./kafka-producer-perf-test.sh --topic __a_demo_topic --throughput -1
--num-records 3000000 --record-size 1024 --producer-props acks=all
bootstrap.servers=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:9092,
172.30.0.123:9092
```

5. È stato eseguito un controllo dello stato di salute per il broker-1 per ciascuno dei cluster utilizzando telnet:
   - telnet `172.30.0.160 9092`
   - telnet `172.30.0.198 9092`

     Nella schermata successiva viene mostrato un controllo dello stato di salute dei broker su entrambi i cluster:

```
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.160 9092
Trying 172.30.0.160...
Connected to 172.30.0.160.
Escape character is '^]'.
^[

Connection closed by foreign host.
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.198 9092
Trying 172.30.0.198...
Connected to 172.30.0.198.
Escape character is '^]'.
^[
```

6. Per attivare la condizione di errore che causa il crash dei cluster Kafka che utilizzano i volumi di storage NFSv3, abbiamo avviato il processo di riassegnazione delle partizioni su entrambi i cluster. La riassegnazione delle partizioni è stata eseguita utilizzando `kafka-reassign-partitions.sh`. Il processo dettagliato è il seguente:
   a. Per riassegnare le partizioni per un argomento in un cluster Kafka, abbiamo generato la configurazione di riassegnazione proposta JSON (eseguita per entrambi i cluster).

   ```
   kafka-reassign-partitions --bootstrap
   -server=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:9092,172.30.
   0.123:9092 --broker-list "1,2,3,4" --topics-to-move-json-file
   /tmp/topics.json --generate
   ```

   b. Il JSON di riassegnazione generato è stato quindi salvato in `/tmp/reassignment- file.json`.
   c. Il processo di riassegnazione effettiva delle partizioni è stato attivato dal seguente comando:

   ```
   kafka-reassign-partitions --bootstrap
   -server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:9092,172.30.
   0.204:9092 --reassignment-json-file /tmp/reassignment-file.json
   -execute
   ```

7. Dopo alcuni minuti di completamento della riassegnazione, un altro controllo dello stato di salute dei broker ha dimostrato che il cluster che utilizza volumi di storage NFSv3 ha riscontrato un problema di ridenominazione sciocco e si è bloccato, mentre il cluster 1 utilizza volumi di storage NetApp ONTAP NFSv4.1 con la correzione delle operazioni senza interruzioni.

```
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.160 9092
Trying 172.30.0.160...
Connected to 172.30.0.160.
Escape character is '^]'.
^[

Connection closed by foreign host.
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.198 9092
Trying 172.30.0.198...
telnet: connect to address 172.30.0.198: Connection refused
telnet: Unable to connect to remote host
```

- Cluster1-Broker-1 è attivo.
- Cluster2-broker-1 è morto.

8. Dopo aver controllato le directory di log di Kafka, era chiaro che il cluster 1 che utilizzava i volumi di storage NetApp ONTAP NFSv4.1 con la correzione aveva un'assegnazione pulita delle partizioni, mentre il cluster 2 utilizzava lo storage NFSv3 generico non era dovuto a problemi di ridenominazione sciocco, che hanno portato al crash. La figura seguente mostra il ribilanciamento delle partizioni del cluster 2, che ha causato un problema di ridenominazione sciocco sullo storage NFSv3.

```
/demo/broker_demo_1/__a_demo_topic-1.b31a8dd60fd443b283ffda2ecca9c2b9-delete:
total 40
drwxr-xr-x.   2 nobody nobody  4096 Sep 19 10:37 .
drwxr-xr-x. 246 nobody nobody 32768 Sep 19 10:36 ..
-rw-r--r--.   1 nobody nobody     5 Sep 19 10:22 .nfs0000000025f9008400000045
-rw-r--r--.   1 nobody nobody     0 Sep 19 10:25 .nfs0000000025f91d6800000048

/demo/broker_demo_1/__a_demo_topic-2:
total 832592
drwxr-xr-x.   2 nobody nobody      4096 Sep 19 10:26 .
drwxr-xr-x. 246 nobody nobody     32768 Sep 19 10:36 ..
-rw-r--r--.   1 nobody nobody         5 Sep 19 10:22 .nfs0000000025f91d5500000046
-rw-r--r--.   1 nobody nobody         0 Sep 19 10:25 .nfs0000000025f91fce00000047
-rw-r--r--.   1 nobody nobody  10485760 Sep 19 10:24 00000000000000000000.index
-rw-r--r--.   1 nobody nobody 848113134 Sep 19 10:24 00000000000000000000.log
-rw-r--r--.   1 nobody nobody  10485756 Sep 19 10:24 00000000000000000000.timeindex
-rw-r--r--.   1 nobody nobody         0 Sep 19 10:16 leader-epoch-checkpoint
-rw-r--r--.   1 nobody nobody        43 Sep 19 10:16 partition.metadata
```

La figura seguente mostra un ribilanciamento pulito delle partizioni del cluster 1 utilizzando lo storage NetApp NFSv4.1.

```
/demo/broker_demo_1/__a_demo_topic-0:
total 710932
drwxr-xr-x.  2 nobody nobody      4096 Sep 19 10:26 .
drwxr-xr-x. 85 nobody nobody      8192 Sep 19 10:37 ..
-rw-r--r--.  1 nobody nobody  10485760 Sep 19 10:25 00000000000000000000.index
-rw-r--r--.  1 nobody nobody 724167522 Sep 19 10:25 00000000000000000000.log
-rw-r--r--.  1 nobody nobody  10485756 Sep 19 10:25 00000000000000000000.timeindex
-rw-r--r--.  1 nobody nobody         0 Sep 19 10:15 leader-epoch-checkpoint
-rw-r--r--.  1 nobody nobody        43 Sep 19 10:15 partition.metadata

/demo/broker_demo_1/__a_demo_topic-2:
total 780016
drwxr-xr-x.  2 nobody nobody      4096 Sep 19 10:35 .
drwxr-xr-x. 85 nobody nobody      8192 Sep 19 10:37 ..
-rw-r--r--.  1 nobody nobody  10485760 Sep 19 10:36 00000000000000000000.index
-rw-r--r--.  1 nobody nobody 794575786 Sep 19 10:36 00000000000000000000.log
-rw-r--r--.  1 nobody nobody  10485756 Sep 19 10:36 00000000000000000000.timeindex
-rw-r--r--.  1 nobody nobody         0 Sep 19 10:35 leader-epoch-checkpoint
-rw-r--r--.  1 nobody nobody        43 Sep 19 10:35 partition.metadata
```

Perché scegliere NetApp NFS per i workload Kafka?

Ora che esiste una soluzione per il problema del ridenominazione sciocco nello storage NFS con Kafka, è possibile creare solide implementazioni che sfruttano lo storage NetApp ONTAP per il carico di lavoro Kafka. Questo non solo riduce significativamente l'overhead operativo, ma offre anche i seguenti vantaggi ai cluster Kafka:

- **Utilizzo ridotto della CPU per i broker Kafka.** l'utilizzo dello storage NetApp ONTAP disaggregato separa le operazioni di i/o dei dischi dal broker e riduce così l'impatto della CPU.
- **Tempi di recovery più rapidi per i broker.** poiché lo storage NetApp ONTAP disaggregato è condiviso tra i nodi dei broker Kafka, una nuova istanza di calcolo può sostituire un broker difettoso in qualsiasi momento in una frazione del tempo rispetto alle implementazioni Kafka convenzionali senza ricostruire i dati.
- **Efficienza dello storage.** con il provisioning del layer di storage dell'applicazione tramite NetApp ONTAP, i clienti possono sfruttare tutti i vantaggi dell'efficienza dello storage forniti con ONTAP, come compressione dei dati in linea, deduplica e compaction.

Questi vantaggi sono stati testati e validati in casi di test di cui discutiamo in dettaglio in questa sezione.

Riduzione dell'utilizzo della CPU sul broker Kafka

Abbiamo scoperto che l'utilizzo complessivo della CPU è inferiore rispetto alla controparte DAS quando abbiamo eseguito carichi di lavoro simili su due cluster Kafka separati che erano identici nelle specifiche tecniche ma differivano nelle tecnologie di storage. Non solo l'utilizzo complessivo della CPU è inferiore quando il cluster Kafka utilizza lo storage ONTAP, ma l'aumento dell'utilizzo della CPU ha dimostrato un gradiente più delicato rispetto a un cluster Kafka basato su DAS.

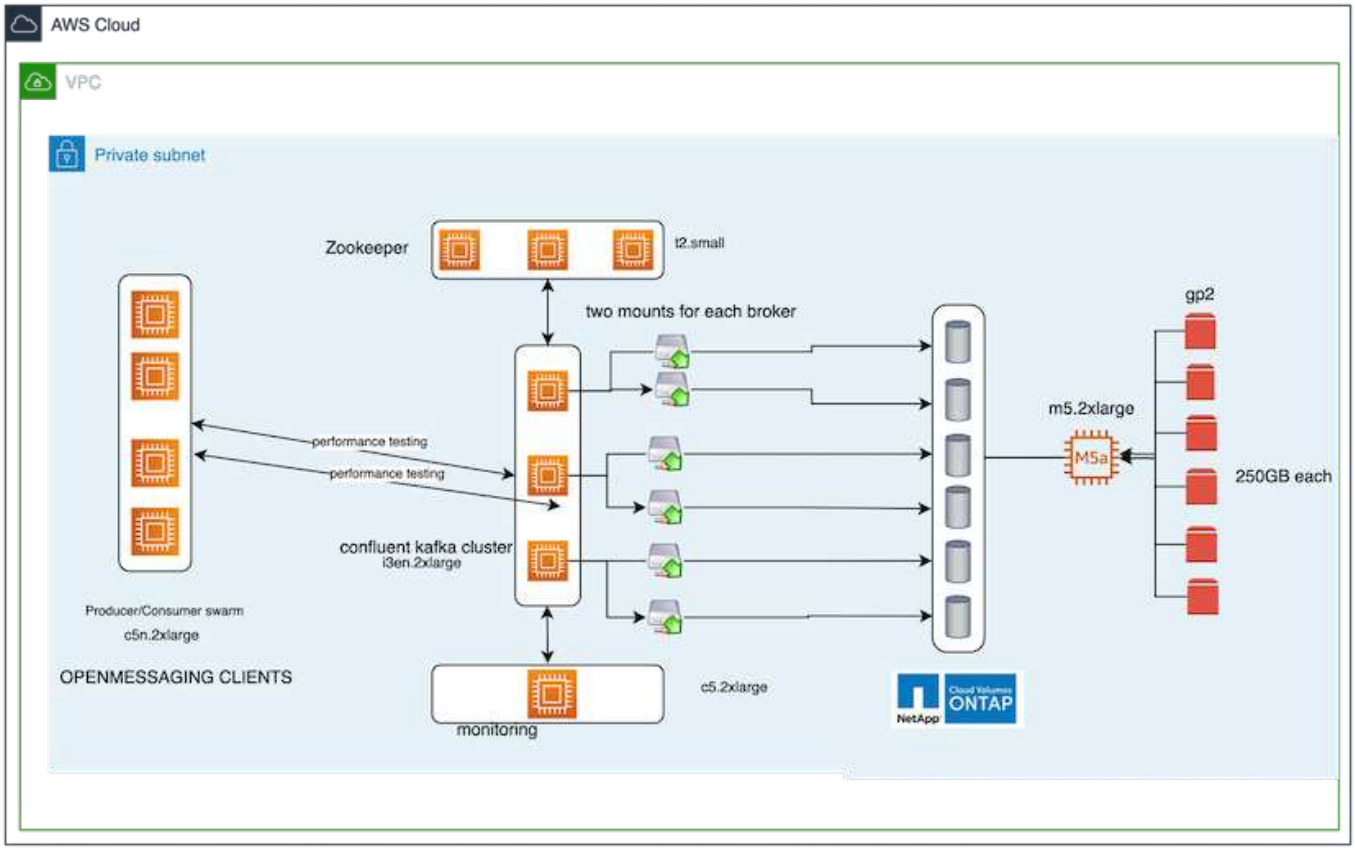
Configurazione architetturale

La seguente tabella mostra la configurazione ambientale utilizzata per dimostrare un utilizzo ridotto della CPU.

Componente della piattaforma	Configurazione dell'ambiente
Tool di benchmarking Kafka 3.2.3: OpenMessaging	• 3 zookeeper – t2.small • 3 server di broker – i3en.2xlarge • 1 x Grafana – c5n.2xlarge • 4 x Produttore/Consumer — c5n.2xlarge
Sistema operativo su tutti i nodi	RHEL 8.7 o versione successiva
Istanza di NetApp Cloud Volumes ONTAP	Istanza a nodo singolo – M5.2xLarge

Tool di benchmarking

Lo strumento di benchmarking utilizzato in questo caso di test è "OpenMessaging" framework. OpenMessaging è indipendente dal vendor e dal linguaggio; fornisce linee guida di settore per finanza, e-commerce, IoT e big data; aiuta a sviluppare applicazioni di messaggistica e streaming su sistemi e piattaforme eterogenee. La figura seguente mostra l'interazione dei client OpenMessaging con un cluster Kafka.



- **Compute.** abbiamo utilizzato un cluster Kafka a tre nodi con un ensemble di zookeeper a tre nodi in esecuzione su server dedicati. Ciascun broker disponeva di due mount point NFSv4.1 su un singolo volume sull'istanza CVO di NetApp attraverso una LIF dedicata.
- **Monitoring.** abbiamo utilizzato due nodi per una combinazione Prometheus-Grafana. Per la generazione dei carichi di lavoro, abbiamo un cluster a tre nodi separato che può produrre e consumare da questo cluster Kafka.

- **Storage.** abbiamo utilizzato un'istanza NetApp Cloud Volumes ONTAP a nodo singolo con sei volumi AWS-EBS GP2 da 250 GB montati sull'istanza. Questi volumi sono stati quindi esposti al cluster Kafka come sei volumi NFSv4.1 attraverso LIF dedicate.
- **Configurazione.** i due elementi configurabili in questo test case erano i broker Kafka e i carichi di lavoro OpenMessaging.
  - **Broker config.** per i broker Kafka sono state selezionate le seguenti specifiche. Abbiamo utilizzato il fattore di replica di 3 per tutte le misurazioni, come evidenziato di seguito.

```
broker.id=1
advertised.listeners=PLAINTEXT://172.30.0.185:9092
log.dirs=/mnt/data-1
zookeeper.connect=172.30.0.13:2181,172.30.0.108:2181,172.30.0.253:2181
num.replica.fetchers=8
message.max.bytes=10485760
replica.fetch.max.bytes=10485760
num.network.threads=8
default.replication.factor=3
replica.lag.time.max.ms=100000000
replica.fetch.max.bytes=1048576
replica.fetch.wait.max.ms=500
num.replica.fetchers=1
replica.high.watermark.checkpoint.interval.ms=5000
fetch.purgatory.purge.interval.requests=1000
producer.purgatory.purge.interval.requests=1000
replica.socket.timeout.ms=30000
replica.socket.receive.buffer.bytes=65536
```

- **OpenMessaging benchmark (OMB) workload config.** sono state fornite le seguenti specifiche. Abbiamo specificato un tasso di produzione target, evidenziato di seguito.

```
name: 4 producer / 4 consumers on 1 topic
topics: 1
partitionsPerTopic: 100
messageSize: 1024
payloadFile: "payload/payload-1Kb.data"
subscriptionsPerTopic: 1
consumerPerSubscription: 4
producersPerTopic: 4
producerRate: 40000
consumerBacklogSizeGB: 0
testDurationMinutes: 5
```

Metodologia di test

1. Sono stati creati due cluster simili, ciascuno con un proprio set di benchmark di sciami di cluster.
   - Cluster 1.* cluster Kafka basato su NFS.

    - Cluster 2.* cluster Kafka basato su DAS.
2. Utilizzando un comando OpenMessaging, carichi di lavoro simili sono stati attivati su ciascun cluster.

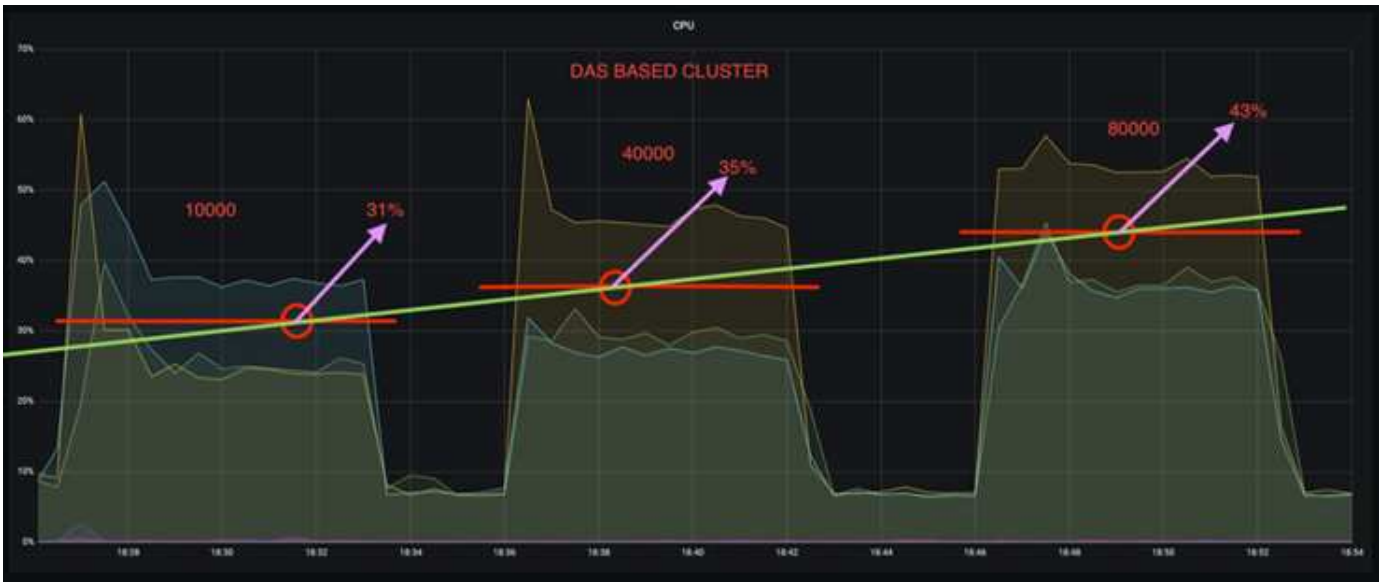
```
sudo bin/benchmark --drivers driver-kafka/kafka-group-all.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

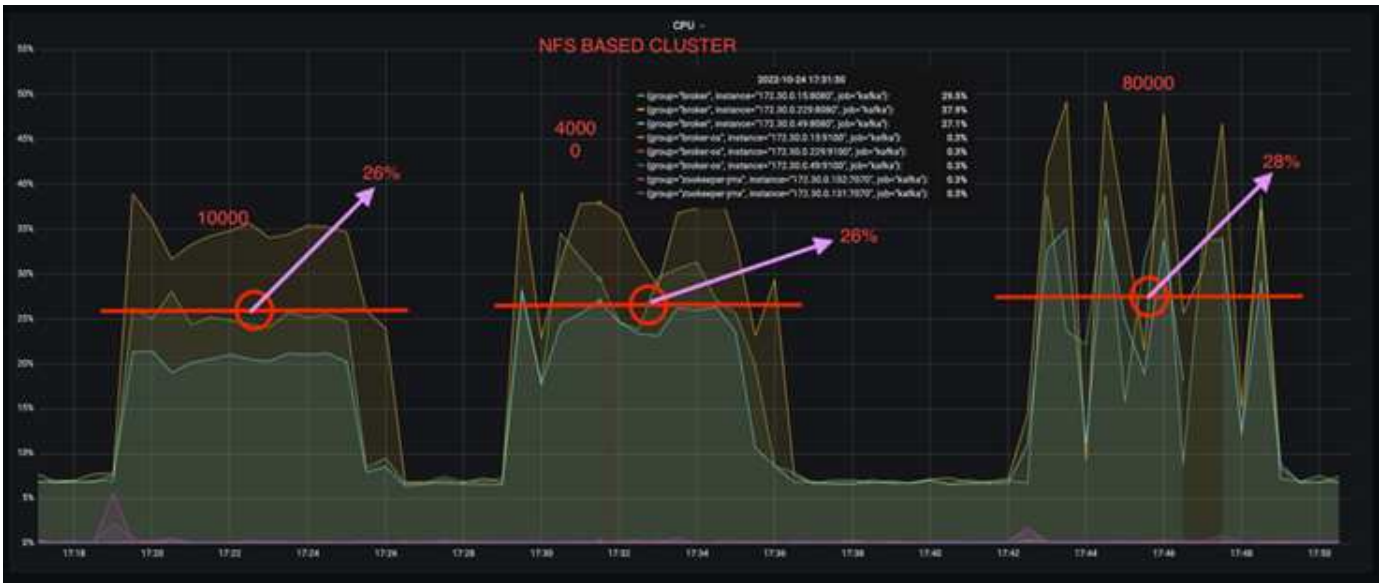
3. La configurazione del tasso di produzione è stata aumentata in quattro iterazioni e l'utilizzo della CPU è stato registrato con Grafana. Il tasso di produzione è stato impostato sui seguenti livelli:
    - 10,000
    - 40,000
    - 80,000
    - 100,000

Osservazione

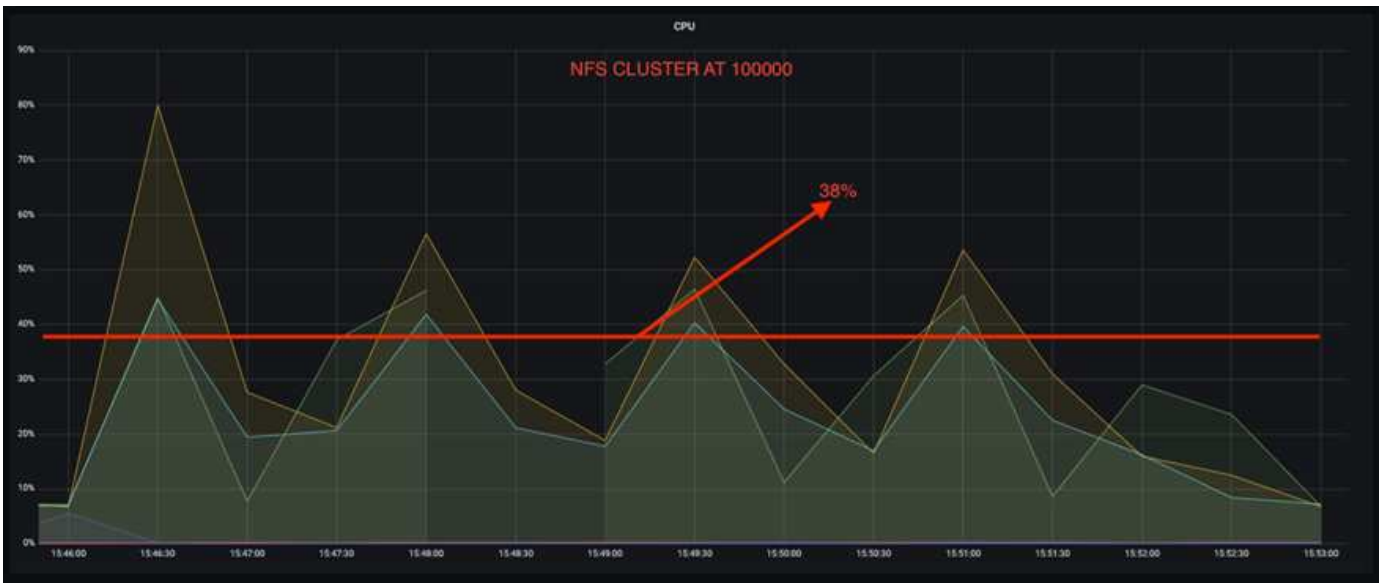
L'utilizzo dello storage NetApp NFS con Kafka offre due vantaggi principali:

- **È possibile ridurre l'utilizzo della CPU di quasi un terzo.** l'utilizzo complessivo della CPU con carichi di lavoro simili è stato inferiore per NFS rispetto agli SSD DAS; i risparmi variano dal 5% per velocità di produzione inferiori al 32% per velocità di produzione superiori.
- **Una riduzione di tre volte nella deriva di utilizzo della CPU a velocità di produzione più elevate.** come previsto, si è verificata una deriva verso l'alto per l'aumento dell'utilizzo della CPU con l'aumento dei tassi di produzione. Tuttavia, l'utilizzo della CPU sui broker Kafka che utilizzano DAS è aumentato dal 31% per il tasso di produzione inferiore al 70% per il tasso di produzione più elevato, un aumento del 39%. Tuttavia, con un backend di storage NFS, l'utilizzo della CPU è aumentato dal 26% al 38%, con un aumento del 12%.





Inoltre, con 100,000 messaggi, DAS mostra un utilizzo della CPU maggiore rispetto a un cluster NFS.



Recupero più rapido del broker

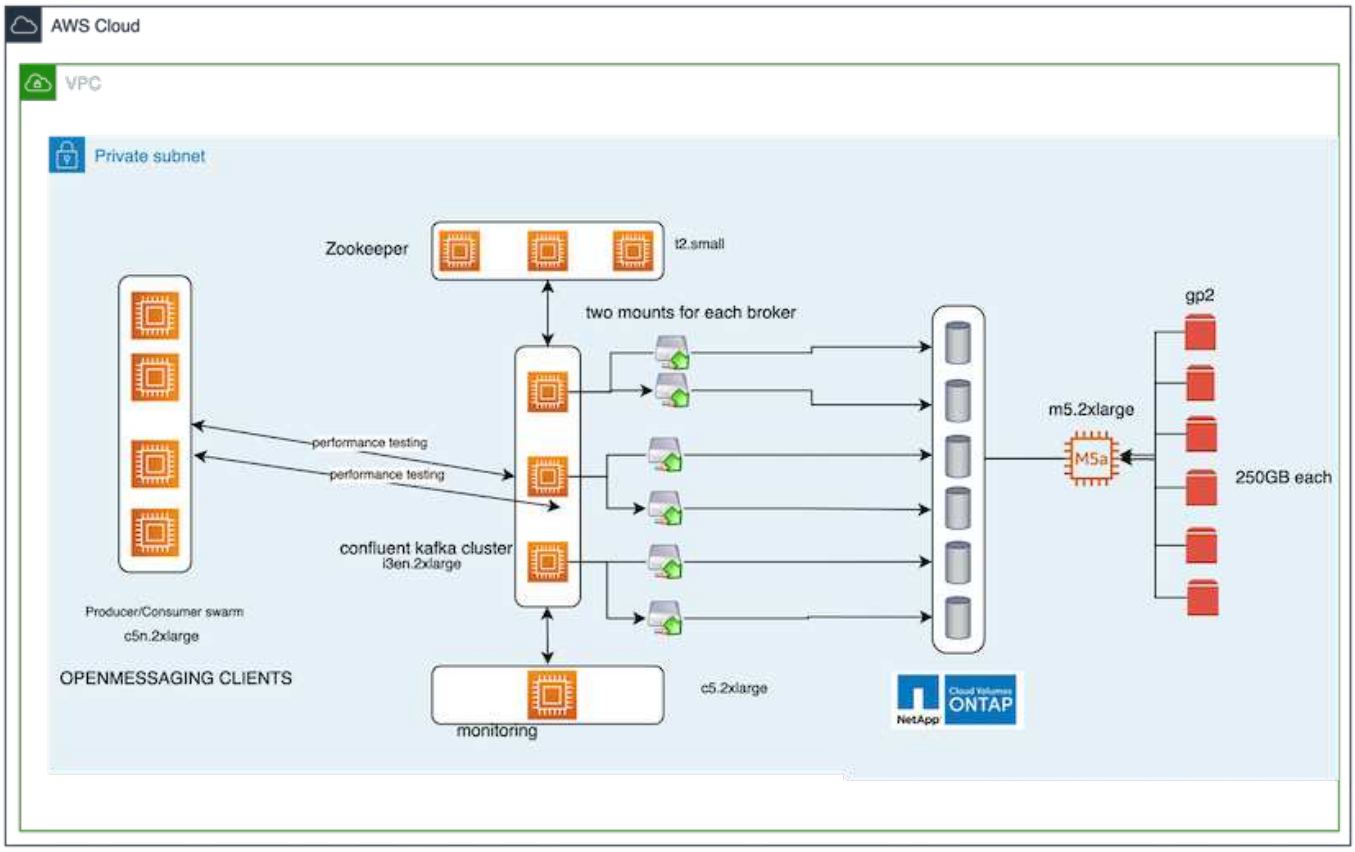
Abbiamo scoperto che i broker Kafka si ripristinano più velocemente quando utilizzano lo storage NetApp NFS condiviso. Quando un broker si blocca in un cluster Kafka, questo broker può essere sostituito da un broker sano con lo stesso ID broker. Dopo aver eseguito questo test case, abbiamo scoperto che, nel caso di un cluster Kafka basato su DAS, il cluster ricostruisce i dati su un nuovo broker sano aggiunto, il che richiede tempo. Nel caso di un cluster Kafka basato su NetApp NFS, il broker che sostituisce continua a leggere i dati dalla directory di log precedente e a ripristinarli molto più velocemente.

Configurazione architetturale

La seguente tabella mostra la configurazione ambientale per un cluster Kafka che utilizza NAS.

Componente della piattaforma	Configurazione dell'ambiente
Kafka 3.2.3	• 3 zookeeper – t2.small • 3 server di broker – i3en.2xlarge • 1 x Grafana – c5n.2xlarge • 4 x produttore/consumatore — c5n.2xlarge • 1 nodo Kafka di backup – i3en.2xlarge
Sistema operativo su tutti i nodi	RHEL8.7 o versione successiva
Istanza di NetApp Cloud Volumes ONTAP	Istanza a nodo singolo – M5.2xLarge

La figura seguente mostra l'architettura di un cluster Kafka basato su NAS.



- **Compute.** un cluster Kafka a tre nodi con un ensemble di zookeeper a tre nodi in esecuzione su server dedicati. Ciascun broker dispone di due punti di montaggio NFS per un singolo volume sull'istanza NetApp CVO tramite un LIF dedicato.
- **Monitoring.** due nodi per una combinazione Prometheus-Grafana. Per la generazione dei carichi di lavoro, utilizziamo un cluster a tre nodi separato in grado di produrre e utilizzare questo cluster Kafka.
- **Storage.** un'istanza NetApp Cloud Volumes ONTAP a nodo singolo con sei volumi GP2 AWS-EBS da 250 GB montati sull'istanza. Questi volumi vengono quindi esposti al cluster Kafka come sei volumi NFS attraverso LIF dedicate.
- **Configurazione Broker.** l'elemento configurabile in questo caso di test sono i broker Kafka. Per i broker Kafka sono state selezionate le seguenti specifiche. Il `replica.lag.time.mx.ms` È impostato su un valore alto perché questo determina la velocità con cui un determinato nodo viene estratto dall'elenco ISR. Quando si passa da un nodo cattivo a un nodo integro, non si desidera che l'ID broker sia escluso dall'elenco ISR.

```
broker.id=1
advertised.listeners=PLAINTEXT://172.30.0.185:9092
log.dirs=/mnt/data-1
zookeeper.connect=172.30.0.13:2181,172.30.0.108:2181,172.30.0.253:2181
num.replica.fetchers=8
message.max.bytes=10485760
replica.fetch.max.bytes=10485760
num.network.threads=8
default.replication.factor=3
replica.lag.time.max.ms=100000000
replica.fetch.max.bytes=1048576
replica.fetch.wait.max.ms=500
num.replica.fetchers=1
replica.high.watermark.checkpoint.interval.ms=5000
fetch.purgatory.purge.interval.requests=1000
producer.purgatory.purge.interval.requests=1000
replica.socket.timeout.ms=30000
replica.socket.receive.buffer.bytes=65536
```

Metodologia di test

1. Sono stati creati due cluster simili:
   - Un cluster confluente basato su EC2.
   - Un cluster confluente basato su NetApp NFS.
2. È stato creato un nodo Kafka di standby con una configurazione identica ai nodi del cluster Kafka originale.
3. Su ciascuno dei cluster è stato creato un argomento di esempio e sono stati popolati circa 110 GB di dati su ciascuno dei broker.
   - **Cluster basato su EC2.** Su Cui è mappata Una directory di dati del broker Kafka `/mnt/data-2` (Nella figura seguente, Broker-1 del cluster1 [terminale sinistro]).
   - **Cluster NetApp basato su NFS.** Una directory di dati del broker Kafka è montata su NFS point `/mnt/data` (Nella figura seguente, Broker-1 del cluster2 [terminale destro]).

4. In ciascuno dei cluster, il broker-1 è stato terminato per attivare un processo di recovery del broker non riuscito.
5. Una volta terminato il broker, l'indirizzo IP del broker è stato assegnato come IP secondario al broker di standby. Ciò era necessario perché un broker in un cluster Kafka è identificato da quanto segue:
   - **Indirizzo IP.** assegnato riassegnando l'IP del broker guasto al broker di standby.
   - **Broker ID.** questa opzione è stata configurata nel broker di standby `server.properties`.
6. Al momento dell'assegnazione IP, il servizio Kafka è stato avviato sul broker di standby.
7. Dopo un po', i log del server sono stati estratti per controllare il tempo impiegato per creare i dati sul nodo sostitutivo nel cluster.

Osservazione

Il recupero del broker Kafka è stato quasi nove volte più veloce. Il tempo necessario per ripristinare un nodo broker guasto è risultato notevolmente più veloce quando si utilizza lo storage condiviso NetApp NFS rispetto all'utilizzo di SSD DAS in un cluster Kafka. Per 1 TB di dati su argomenti, il tempo di ripristino per un cluster basato su DAS è stato di 48 minuti, rispetto a meno di 5 minuti per un cluster Kafka basato su NetApp-NFS.

Abbiamo osservato che il cluster basato su EC2 ha impiegato 10 minuti per ricostruire i 110 GB di dati sul nuovo nodo del broker, mentre il cluster basato su NFS ha completato il ripristino in 3 minuti. Abbiamo anche osservato nei log che gli offset consumer per le partizioni EC2 erano 0, mentre nel cluster NFS gli offset consumer sono stati rilevati dal broker precedente.

```
[2022-10-31 09:39:17,747] INFO [LogLoader partition=test-topic-51R3EWs-
0000-55, dir=/mnt/kafka-data/broker2] Reloading from producer snapshot and
rebuilding producer state from offset 583999 (kafka.log.UnifiedLog$)
[2022-10-31 08:55:55,170] INFO [LogLoader partition=test-topic-qbVsEZg-
0000-8, dir=/mnt/data-1] Loading producer state till offset 0 with message
format version 2 (kafka.log.UnifiedLog$)
```

Cluster basato SU DAS

1. Il nodo di backup è iniziato alle 08:55:53,730.

```
2  [2022-10-31 08:55:53,661] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotia
3  [2022-10-31 08:55:53,727] INFO Registered signal handlers for TERM, INT, HUP (or
4  [2022-10-31 08:55:53,730] INFO starting (kafka.server.KafkaServer)
5  [2022-10-31 08:55:53,730] INFO Connecting to zookeeper on 172.30.0.17:2181,172.3
6  [2022-10-31 08:55:53,755] INFO [ZooKeeperClient Kafka server] Initializing a new
```

2. Il processo di ricostruzione dei dati è terminato alle 09:05:24,860. L'elaborazione di 110 GB di dati richiede circa 10 minuti.

```
[2022-10-31 09:05:24,860] INFO [ReplicaFetcherManager on broker 1] Removed fetcher for
partitions HashSet(test-topic-qbVsEZg-0000-95, test-topic-qbVsEZg-0000-5,
test-topic-qbVsEZg-0000-41, test-topic-qbVsEZg-0000-23, test-topic-qbVsEZg-0000-11,
test-topic-qbVsEZg-0000-47, test-topic-qbVsEZg-0000-83, test-topic-qbVsEZg-0000-35,
test-topic-qbVsEZg-0000-89, test-topic-qbVsEZg-0000-71, test-topic-qbVsEZg-0000-53,
test-topic-qbVsEZg-0000-29, test-topic-qbVsEZg-0000-59, test-topic-qbVsEZg-0000-77,
test-topic-qbVsEZg-0000-65, test-topic-qbVsEZg-0000-17)
(kafka.server.ReplicaFetcherManager)
```

Cluster basato su NFS

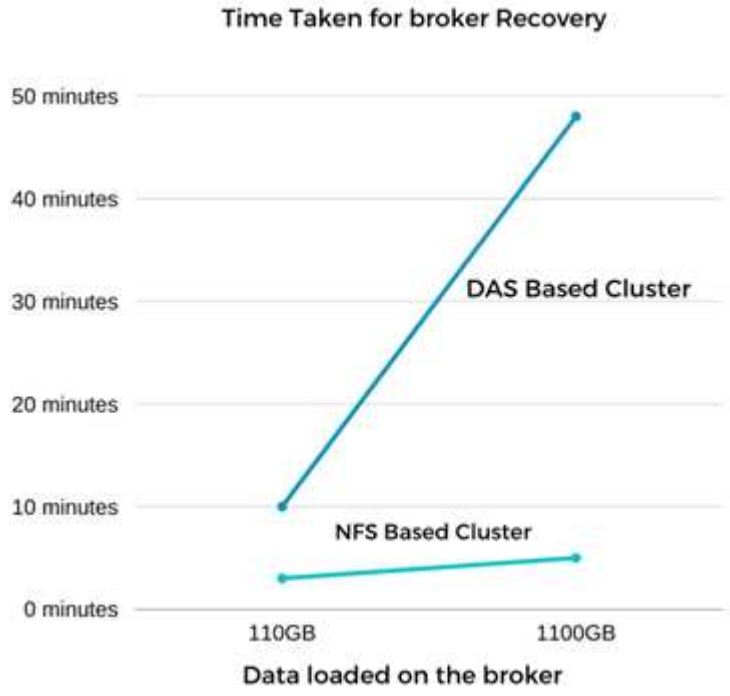
1. Il nodo di backup è stato avviato alle 09:39:17,213. La voce del registro di avvio viene evidenziata di seguito.

```
2  [2022-10-31 09:39:17,142] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiati
3  [2022-10-31 09:39:17,211] INFO Registered signal handlers for TERM, INT, HUP (org.
4  [2022-10-31 09:39:17,213] INFO starting (kafka.server.KafkaServer)
5  [2022-10-31 09:39:17,214] INFO Connecting to zookeeper on 172.30.0.22:2181,172.30.
6  [2022-10-31 09:39:17,238] INFO [ZooKeeperClient Kafka server] Initializing a new s
7  [2022-10-31 09:39:17,244] INFO Client environment:zookeeper.version=3.6.3--6401e4a
8  [2022-10-31 09:39:17,244] INFO Client environment:host.name=ip-172-30-0-110.ec2.in
9  [2022-10-31 09:39:17,244] INFO Client environment:java.version=11.0.17 (org.apache
```

2. Il processo di ricostruzione dei dati è terminato alle 09:42:29,115. L'elaborazione di 110 GB di dati richiede circa 3 minuti.

```
[2022-10-31 09:42:29,115] INFO [GroupMetadataManager brokerId=1] Finished loading offsets
and group metadata from __consumer_offsets-20 in 28478 milliseconds for epoch 3, of which
28478 milliseconds was spent in the scheduler.
(kafka.coordinator.group.GroupMetadataManager)
```

Il test è stato ripetuto per i broker contenenti circa 1 TB di dati, che hanno richiesto circa 48 minuti per il DAS e 3 minuti per NFS. I risultati sono illustrati nel seguente grafico.



Efficienza dello storage

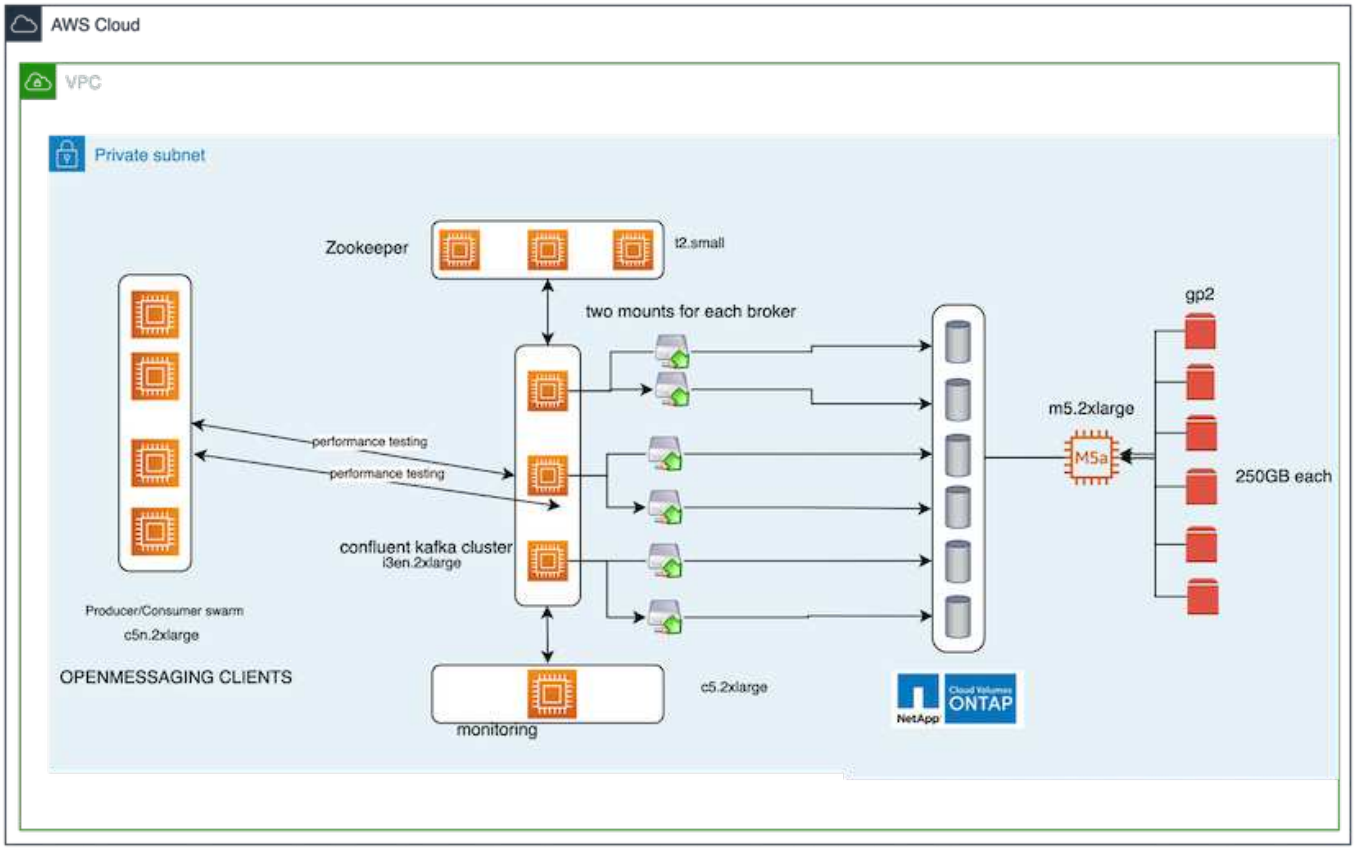
Poiché il provisioning del layer di storage del cluster Kafka è stato eseguito tramite NetApp ONTAP, abbiamo ottenuto tutte le funzionalità di efficienza dello storage di ONTAP. Questo è stato testato generando una quantità significativa di dati su un cluster Kafka con storage NFS fornito su Cloud Volumes ONTAP. Abbiamo potuto constatare che le funzionalità di ONTAP hanno ridotto notevolmente lo spazio.

Configurazione architetturale

La seguente tabella mostra la configurazione ambientale per un cluster Kafka che utilizza NAS.

Componente della piattaforma	Configurazione dell'ambiente
Kafka 3.2.3	• 3 zookeeper – t2.small • 3 server di broker – i3en.2xlarge • 1 x Grafana – c5n.2xlarge • 4 x produttore/consumatore — c5n.2xlargo *
Sistema operativo su tutti i nodi	RHEL8.7 o versione successiva
Istanza di NetApp Cloud Volumes ONTAP	Istanza a nodo singolo – M5.2xLarge

La figura seguente mostra l'architettura di un cluster Kafka basato su NAS.



- **Compute.** abbiamo utilizzato un cluster Kafka a tre nodi con un ensemble di zookeeper a tre nodi in esecuzione su server dedicati. Ciascun broker disponeva di due punti di montaggio NFS su un singolo volume sull'istanza NetApp CVO tramite un LIF dedicato.

- **Monitoring.** abbiamo utilizzato due nodi per una combinazione Prometheus-Grafana. Per la generazione dei carichi di lavoro, abbiamo utilizzato un cluster a tre nodi separato in grado di produrre e utilizzare questo cluster Kafka.
- **Storage.** abbiamo utilizzato un'istanza NetApp Cloud Volumes ONTAP a nodo singolo con sei volumi AWS-EBS GP2 da 250 GB montati sull'istanza. Questi volumi sono stati quindi esposti al cluster Kafka come sei volumi NFS attraverso LIF dedicate.
- **Configurazione.** gli elementi configurabili in questo test case erano i broker Kafka.

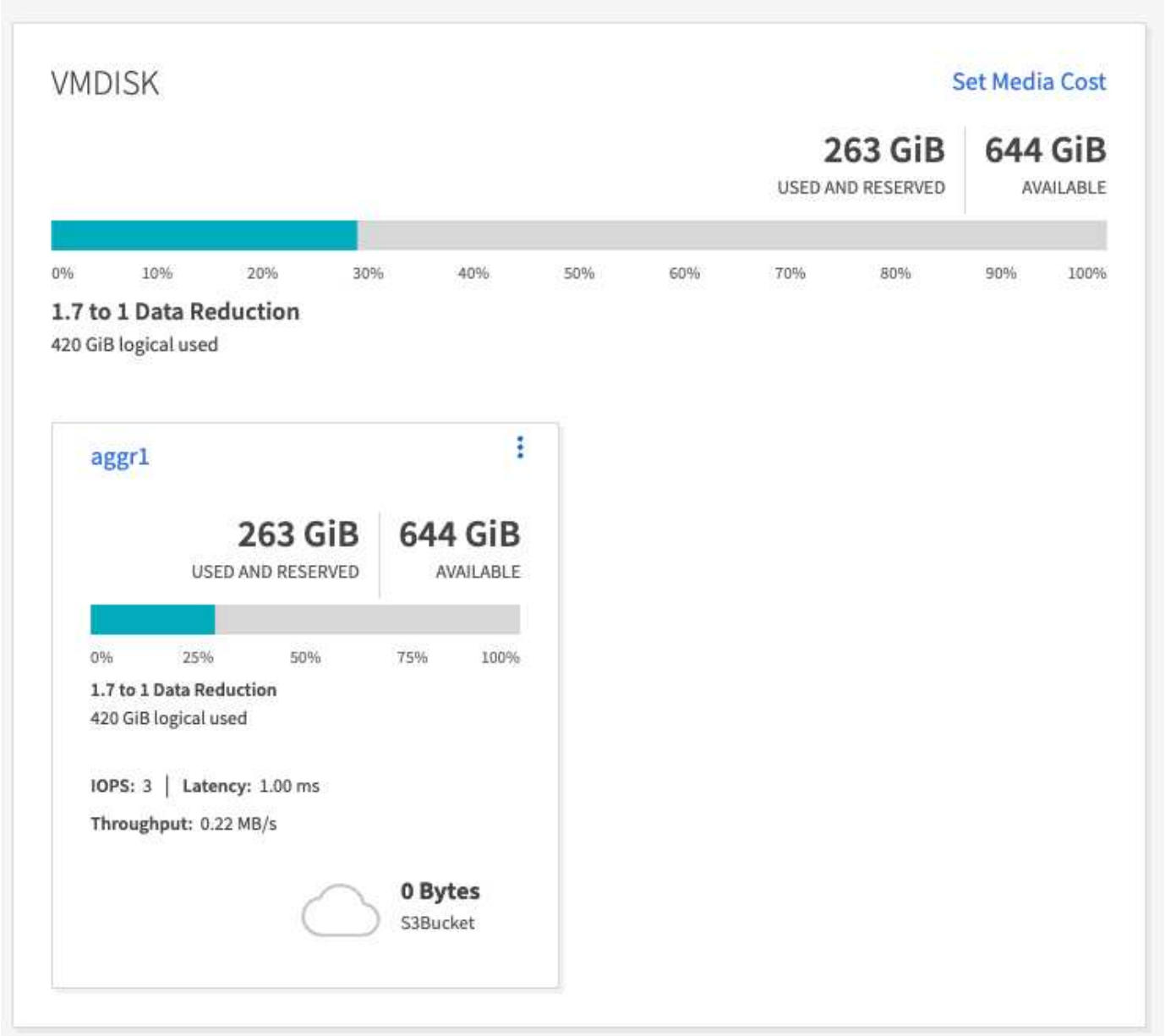
La compressione è stata disattivata alla fine del produttore, consentendo così ai produttori di generare un throughput elevato. L'efficienza dello storage è stata invece gestita dal livello di elaborazione.

Metodologia di test

1. È stato eseguito il provisioning di un cluster Kafka con le specifiche indicate in precedenza.
2. Sul cluster, sono stati prodotti circa 350 GB di dati utilizzando il tool OpenMessaging Benchmarking.
3. Una volta completato il carico di lavoro, le statistiche sull'efficienza dello storage sono state raccolte utilizzando Gestione di sistema di ONTAP e l'interfaccia CLI.

Osservazione

Per i dati generati con lo strumento OMB, abbiamo registrato un risparmio di spazio di ~33% con un rapporto di efficienza dello storage di 1.70:1. Come mostrato nelle figure seguenti, lo spazio logico utilizzato dai dati prodotti era di 420,3 GB e lo spazio fisico utilizzato per contenere i dati era di 281,7 GB.




```
shantanuCVOinstancenew::> df -h -S

Warning: The "-S" parameter is deprecated and may be removed in a future release. To show the efficiency ratio use "aggr show-efficiency"
         command.
Filesystem                 used       total-saved    %total-saved    deduplicated    %deduplicated    compressed    %compressed  Vserver
/vol/vol0/               7319MB                0B              0%              0B               0%            0B             0%  shantanuCVOinstancenew-01
/vol/kafka_vol/           281GB             138GB             33%           138GB              33%            0B             0%  svm_shantanuCVOinstancenew
/vol/svm_shantanuCVOinstancenew_root/
                          660KB                0B              0%              0B               0%            0B             0%  svm_shantanuCVOinstancenew
3 entries were displayed.
```

```
                    Name of the Aggregate: aggr1
              Node where Aggregate Resides: shantanuCVOinstancenew-01
            Total Storage Efficiency Ratio: 1.70:1
Total Data Reduction Efficiency Ratio Without Snapshots: 1.70:1
Total Data Reduction Efficiency Ratio without snapshots and flexclones: 1.70:1
        Logical Space Used for All Volumes: 420.3GB
       Physical Space Used for All Volumes: 281.7GB
```

Panoramica delle performance e validazione in AWS

Un cluster Kafka con il layer di storage montato su NetApp NFS è stato sottoposto a benchmark per le performance nel cloud AWS. Gli esempi di benchmarking sono descritti nelle sezioni seguenti.

Kafka nel cloud AWS con NetApp Cloud Volumes ONTAP (coppia ad alta disponibilità e nodo singolo)

Un cluster Kafka con NetApp Cloud Volumes ONTAP (coppia ha) è stato sottoposto a benchmark per le performance nel cloud AWS. Questo benchmarking è descritto nelle sezioni seguenti.

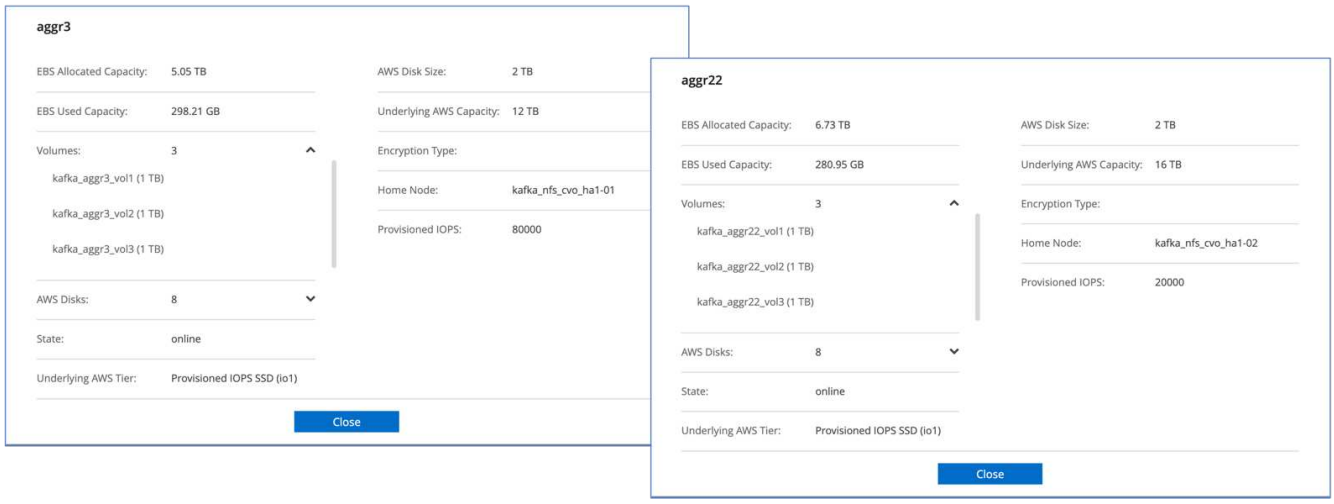
Configurazione architetturale

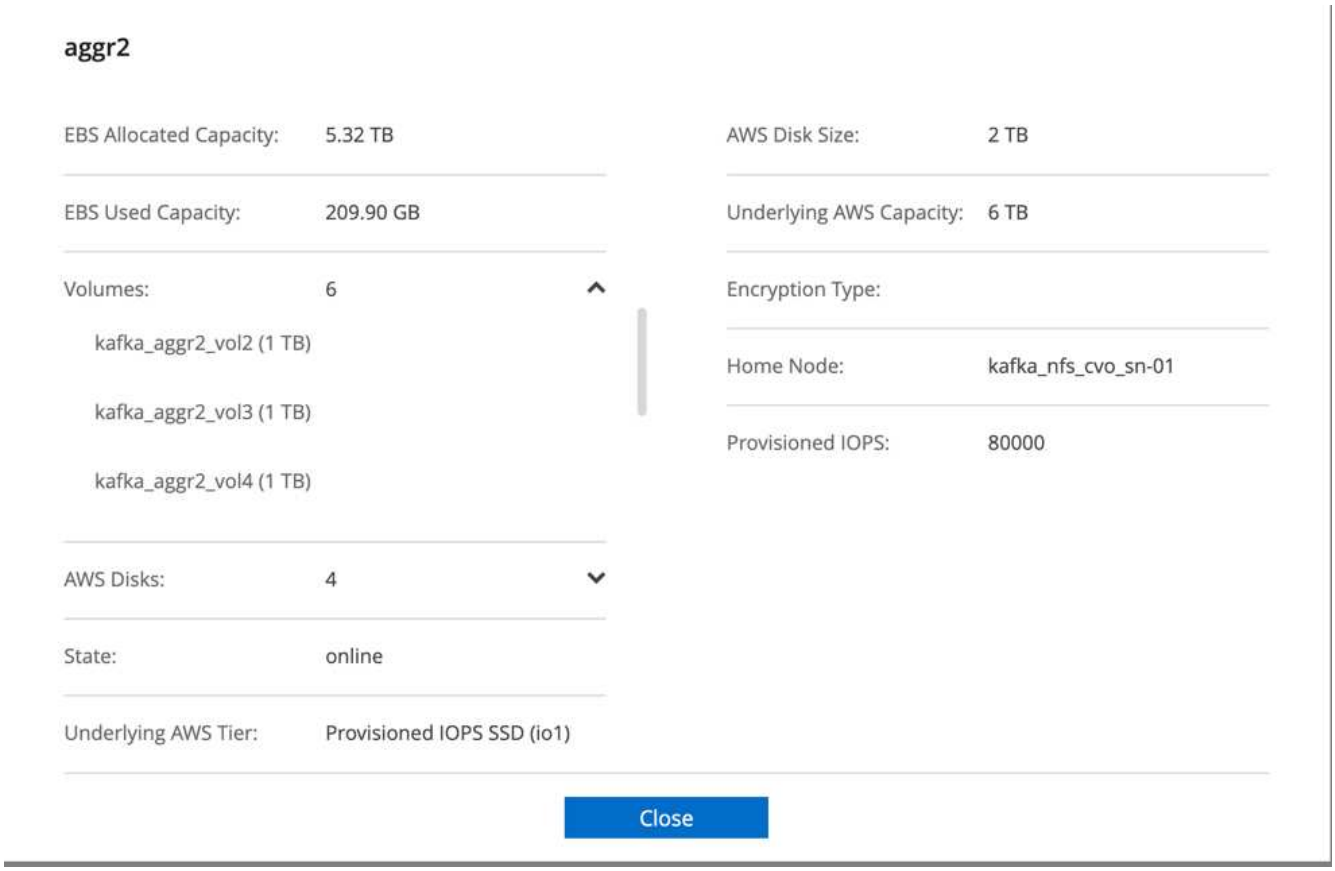
La seguente tabella mostra la configurazione ambientale per un cluster Kafka che utilizza NAS.

Componente della piattaforma	Configurazione dell'ambiente
Kafka 3.2.3	• 3 zookeeper – t2.small • 3 server di broker – i3en.2xlarge • 1 x Grafana – c5n.2xlarge • 4 x produttore/consumatore — c5n.2xlargo *
Sistema operativo su tutti i nodi	RHEL8.6
Istanza di NetApp Cloud Volumes ONTAP	Istanza coppia HA – m5dn.12xLarge x istanza nodo singolo 2node - m5dn.12xLarge x 1 nodo

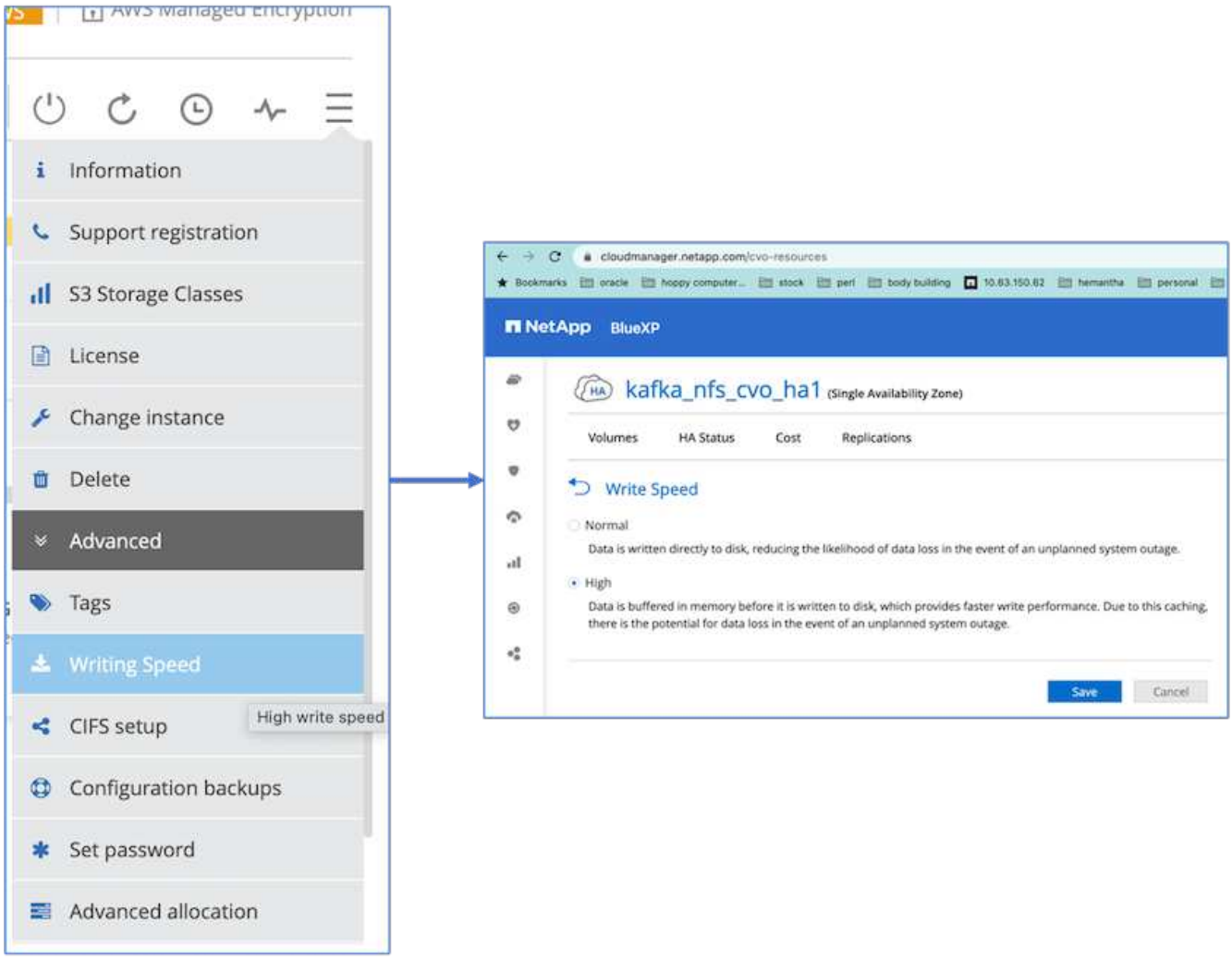
Configurazione di NetApp Cluster Volume ONTAP

1. Per la coppia Cloud Volumes ONTAP ha, abbiamo creato due aggregati con tre volumi su ciascun aggregato su ciascun controller di storage. Per il singolo nodo Cloud Volumes ONTAP, creiamo sei volumi in un aggregato.



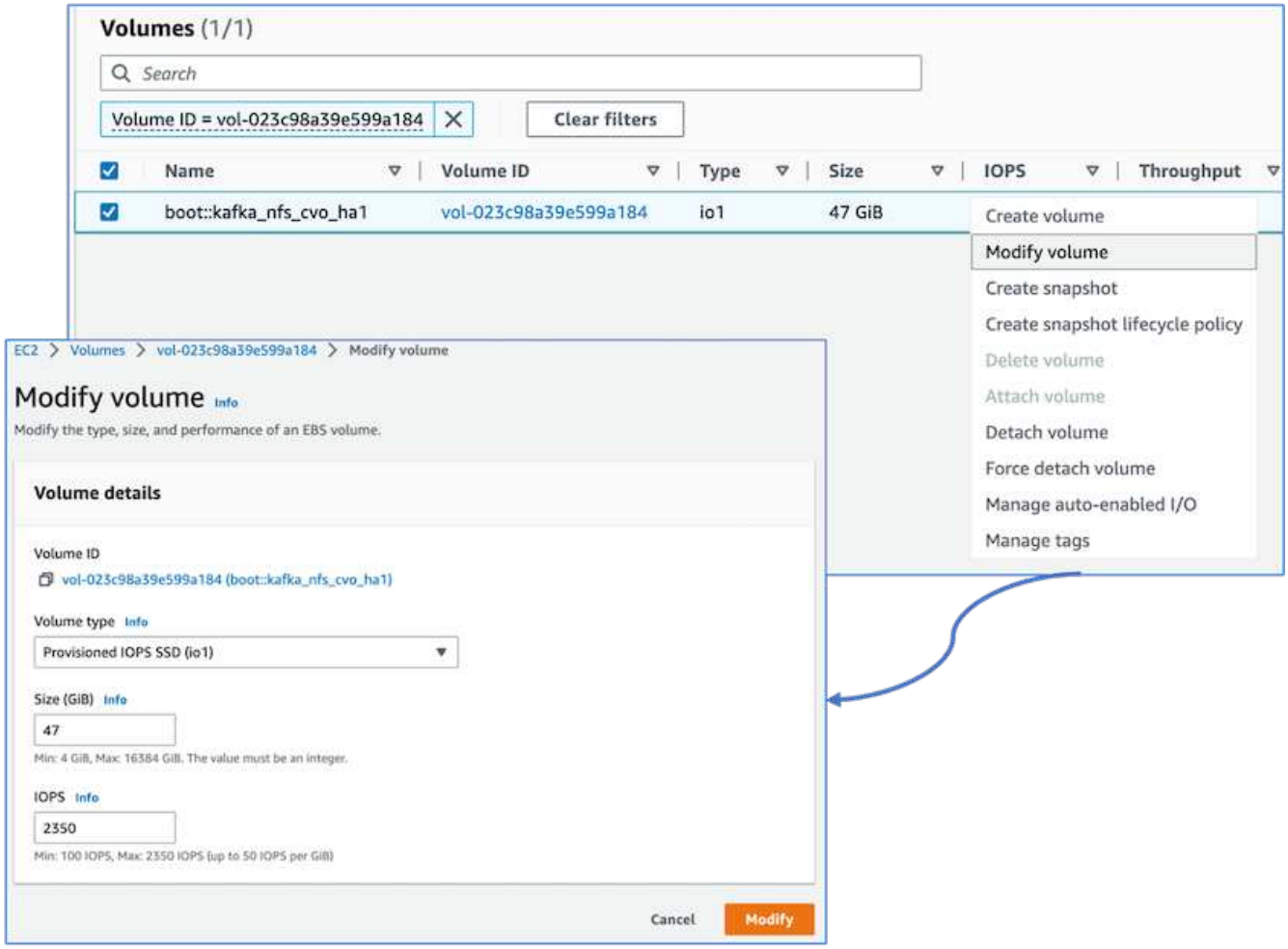


2. Per ottenere performance di rete migliori, abbiamo attivato il networking ad alta velocità sia per la coppia ha che per il singolo nodo.



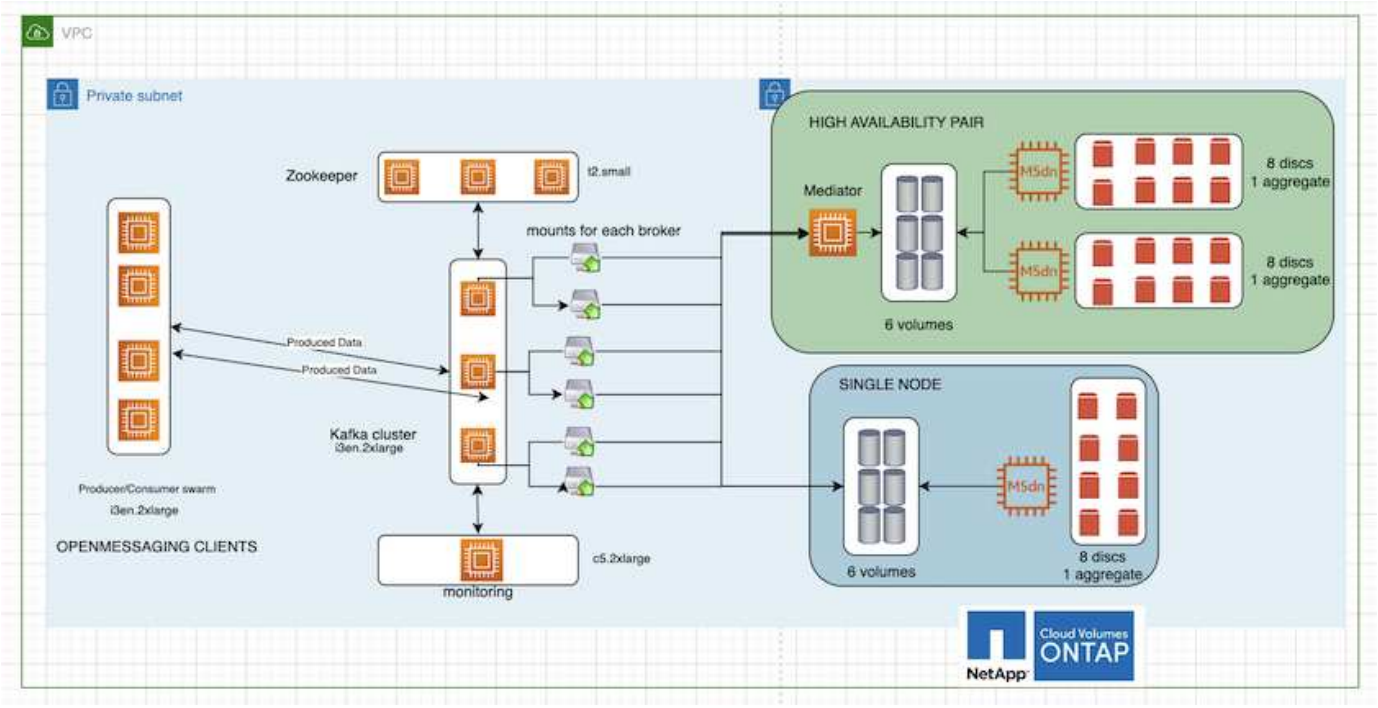
3. Abbiamo notato che la NVRAM ONTAP aveva più IOPS, quindi abbiamo modificato gli IOPS a 2350 per il volume root Cloud Volumes ONTAP. Il disco del volume root in Cloud Volumes ONTAP aveva una dimensione di 47 GB. Il seguente comando ONTAP è per la coppia ha e lo stesso passo è applicabile per il singolo nodo.

```
statistics start -object vnvram -instance vnvram -counter
backing_store_iops -sample-id sample_555
kafka_nfs_cvo_ha1::*> statistics show -sample-id sample_555
Object: vnvram
Instance: vnvram
Start-time: 1/18/2023 18:03:11
End-time: 1/18/2023 18:03:13
Elapsed-time: 2s
Scope: kafka_nfs_cvo_ha1-01
    Counter                                                     Value
    -------------------------------- --------------------------------
    backing_store_iops                                           1479
Object: vnvram
Instance: vnvram
Start-time: 1/18/2023 18:03:11
End-time: 1/18/2023 18:03:13
Elapsed-time: 2s
Scope: kafka_nfs_cvo_ha1-02
    Counter                                                     Value
    -------------------------------- --------------------------------
    backing_store_iops                                           1210
2 entries were displayed.
kafka_nfs_cvo_ha1::*>
```



La figura seguente mostra l'architettura di un cluster Kafka basato su NAS.

- **Compute.** abbiamo utilizzato un cluster Kafka a tre nodi con un ensemble di zookeeper a tre nodi in esecuzione su server dedicati. Ciascun broker disponeva di due punti di montaggio NFS su un singolo volume nell'istanza di Cloud Volumes ONTAP tramite un LIF dedicato.
- **Monitoring.** abbiamo utilizzato due nodi per una combinazione Prometheus-Grafana. Per la generazione dei carichi di lavoro, abbiamo utilizzato un cluster a tre nodi separato in grado di produrre e utilizzare questo cluster Kafka.
- **Storage.** abbiamo utilizzato un'istanza di ha-Pair Cloud Volumes ONTAP con un volume AWS-EBS GP3 da 6 TB montato sull'istanza. Il volume è stato quindi esportato nel broker Kafka con un montaggio NFS.



Configurazioni di benchmarking di OpenMessage

1. Per migliorare le performance NFS, abbiamo bisogno di più connessioni di rete tra il server NFS e il client NFS, che possono essere create utilizzando nconnect. Montare i volumi NFS sui nodi di broker con l'opzione nconnect eseguendo il seguente comando:

```
[root@ip-172-30-0-121 ~]# cat /etc/fstab
UUID=eaa1f38e-de0f-4ed5-a5b5-2fa9db43bb38/xfsdefaults00
/dev/nvme1n1 /mnt/data-1 xfs defaults,noatime,nodiscard 0 0
/dev/nvme2n1 /mnt/data-2 xfs defaults,noatime,nodiscard 0 0
172.30.0.233:/kafka_aggr3_vol1 /kafka_aggr3_vol1 nfs
defaults,nconnect=16 0 0
172.30.0.233:/kafka_aggr3_vol2 /kafka_aggr3_vol2 nfs
defaults,nconnect=16 0 0
172.30.0.233:/kafka_aggr3_vol3 /kafka_aggr3_vol3 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol1 /kafka_aggr22_vol1 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol2 /kafka_aggr22_vol2 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol3 /kafka_aggr22_vol3 nfs
defaults,nconnect=16 0 0
[root@ip-172-30-0-121 ~]# mount -a
[root@ip-172-30-0-121 ~]# df -h
Filesystem                       Size  Used Avail Use% Mounted on
devtmpfs                          31G     0   31G   0% /dev
tmpfs                             31G  249M   31G   1% /run
tmpfs                             31G     0   31G   0% /sys/fs/cgroup
/dev/nvme0n1p2                    10G  2.8G  7.2G  28% /
/dev/nvme1n1                     2.3T  248G  2.1T  11% /mnt/data-1
/dev/nvme2n1                     2.3T  245G  2.1T  11% /mnt/data-2
172.30.0.233:/kafka_aggr3_vol1   1.0T   12G 1013G   2% /kafka_aggr3_vol1
172.30.0.233:/kafka_aggr3_vol2   1.0T  5.5G 1019G   1% /kafka_aggr3_vol2
172.30.0.233:/kafka_aggr3_vol3   1.0T  8.9G 1016G   1% /kafka_aggr3_vol3
172.30.0.242:/kafka_aggr22_vol1  1.0T  7.3G 1017G   1%
/kafka_aggr22_vol1
172.30.0.242:/kafka_aggr22_vol2  1.0T  6.9G 1018G   1%
/kafka_aggr22_vol2
172.30.0.242:/kafka_aggr22_vol3  1.0T  5.9G 1019G   1%
/kafka_aggr22_vol3
tmpfs                            6.2G     0  6.2G   0% /run/user/1000
[root@ip-172-30-0-121 ~]#
```

2. Verificare le connessioni di rete in Cloud Volumes ONTAP. Il seguente comando ONTAP viene utilizzato dal singolo nodo Cloud Volumes ONTAP. Lo stesso passaggio si applica alla coppia Cloud Volumes ONTAP ha.

```
Last login time: 1/20/2023 00:16:29
kafka_nfs_cvo_sn::> network connections active show -service nfs*
-fields remote-host
node                cid        vserver              remote-host
```

```
------------------- ---------- -------------------- ------------
kafka_nfs_cvo_sn-01 2315762628 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762629 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762630 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762631 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762632 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762633 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762634 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762635 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762636 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762637 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762639 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762640 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762641 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762642 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762643 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762644 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762645 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762646 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762647 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762648 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762649 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762650 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762651 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762652 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762653 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762656 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762657 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762658 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762659 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762660 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762661 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762662 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762663 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762664 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762665 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762666 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762667 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762668 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762669 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762670 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762671 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762672 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762673 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762674 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762676 svm_kafka_nfs_cvo_sn 172.30.0.121
```

```
kafka_nfs_cvo_sn-01 2315762677 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762678 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762679 svm_kafka_nfs_cvo_sn 172.30.0.223
48 entries were displayed.

kafka_nfs_cvo_sn::>
```

3. Utilizziamo il seguente Kafka `server.properties` In tutti i broker Kafka per la coppia Cloud Volumes ONTAP ha. Il `log.dirs` la proprietà è diversa per ogni broker e le proprietà rimanenti sono comuni per gli broker. Per il broker1, il `log.dirs` il valore è il seguente:

```
[root@ip-172-30-0-121 ~]# cat /opt/kafka/config/server.properties
broker.id=0
advertised.listeners=PLAINTEXT://172.30.0.121:9092
#log.dirs=/mnt/data-1/d1,/mnt/data-1/d2,/mnt/data-1/d3,/mnt/data-
2/d1,/mnt/data-2/d2,/mnt/data-2/d3
log.dirs=/kafka_aggr3_vol1/broker1,/kafka_aggr3_vol2/broker1,/kafka_aggr
3_vol3/broker1,/kafka_aggr22_vol1/broker1,/kafka_aggr22_vol2/broker1,/ka
fka_aggr22_vol3/broker1
zookeeper.connect=172.30.0.12:2181,172.30.0.30:2181,172.30.0.178:2181
num.network.threads=64
num.io.threads=64
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
replica.fetch.max.bytes=524288000
background.threads=20
num.replica.alter.log.dirs.threads=40
num.replica.fetchers=20
[root@ip-172-30-0-121 ~]#
```

   - Per il broker2, il `log.dirs` il valore della proprietà è il seguente:

```
log.dirs=/kafka_aggr3_vol1/broker2,/kafka_aggr3_vol2/broker2,/kafka_a
ggr3_vol3/broker2,/kafka_aggr22_vol1/broker2,/kafka_aggr22_vol2/broke
r2,/kafka_aggr22_vol3/broker2
```

   - Per il broker3, il `log.dirs` il valore della proprietà è il seguente:

```
log.dirs=/kafka_aggr3_vol1/broker3,/kafka_aggr3_vol2/broker3,/kafka_a
ggr3_vol3/broker3,/kafka_aggr22_vol1/broker3,/kafka_aggr22_vol2/broke
r3,/kafka_aggr22_vol3/broker3
```

4. Per il singolo nodo Cloud Volumes ONTAP, Kafka `servers.properties` È uguale alla coppia Cloud Volumes ONTAP ha, ad eccezione di `log.dirs` proprietà.
   - Per il broker1, il `log.dirs` il valore è il seguente:

     ```
     log.dirs=/kafka_aggr2_vol1/broker1,/kafka_aggr2_vol2/broker1,/kafka_a
     ggr2_vol3/broker1,/kafka_aggr2_vol4/broker1,/kafka_aggr2_vol5/broker1
     ,/kafka_aggr2_vol6/broker1
     ```

   - Per il broker2, il `log.dirs` il valore è il seguente:

     ```
     log.dirs=/kafka_aggr2_vol1/broker2,/kafka_aggr2_vol2/broker2,/kafka_a
     ggr2_vol3/broker2,/kafka_aggr2_vol4/broker2,/kafka_aggr2_vol5/broker2
     ,/kafka_aggr2_vol6/broker2
     ```

   - Per il broker3, il `log.dirs` il valore della proprietà è il seguente:

     ```
     log.dirs=/kafka_aggr2_vol1/broker3,/kafka_aggr2_vol2/broker3,/kafka_a
     ggr2_vol3/broker3,/kafka_aggr2_vol4/broker3,/kafka_aggr2_vol5/broker3
     ,/kafka_aggr2_vol6/broker3
     ```

5. Il carico di lavoro nell'OMB viene configurato con le seguenti proprietà: `(/opt/benchmark/workloads/1-topic-100-partitions-1kb.yaml)`.

   ```
   topics: 4
   partitionsPerTopic: 100
   messageSize: 32768
   useRandomizedPayloads: true
   randomBytesRatio: 0.5
   randomizedPayloadPoolSize: 100
   subscriptionsPerTopic: 1
   consumerPerSubscription: 80
   producersPerTopic: 40
   producerRate: 1000000
   consumerBacklogSizeGB: 0
   testDurationMinutes: 5
   ```

   Il `messageSize` può variare in base al caso di utilizzo. Nel nostro test delle performance, abbiamo

utilizzato 3K.

Abbiamo utilizzato due diversi driver, Sync o throughput, da OMB per generare il carico di lavoro sul cluster Kafka.

- Il file yaml utilizzato per le proprietà del driver Sync è il seguente `(/opt/benchmark/driver-kafka/kafka-sync.yaml)`:

```
name: Kafka
driverClass:
io.openmessaging.benchmark.driver.kafka.KafkaBenchmarkDriver
# Kafka client-specific configuration
replicationFactor: 3
topicConfig: |
  min.insync.replicas=2
  flush.messages=1
  flush.ms=0
commonConfig: |

bootstrap.servers=172.30.0.121:9092,172.30.0.72:9092,172.30.0.223:909
2
producerConfig: |
  acks=all
  linger.ms=1
  batch.size=1048576
consumerConfig: |
  auto.offset.reset=earliest
  enable.auto.commit=false
  max.partition.fetch.bytes=10485760
```

- Il file yaml utilizzato per le proprietà del driver di throughput è il seguente `(/opt/benchmark/driver- kafka/kafka-throughput.yaml)`:

```
name: Kafka
driverClass:
io.openmessaging.benchmark.driver.kafka.KafkaBenchmarkDriver
# Kafka client-specific configuration
replicationFactor: 3
topicConfig: |
  min.insync.replicas=2
commonConfig: |

bootstrap.servers=172.30.0.121:9092,172.30.0.72:9092,172.30.0.223:909
2
  default.api.timeout.ms=1200000
  request.timeout.ms=1200000
producerConfig: |
  acks=all
  linger.ms=1
  batch.size=1048576
consumerConfig: |
  auto.offset.reset=earliest
  enable.auto.commit=false
  max.partition.fetch.bytes=10485760
```

Metodologia di test

1. È stato eseguito il provisioning di un cluster Kafka secondo le specifiche descritte in precedenza utilizzando Terraform e Ansible. Il terraform viene utilizzato per costruire l'infrastruttura utilizzando istanze AWS per il cluster Kafka e Ansible crea il cluster Kafka su di essi.
2. È stato attivato un carico di lavoro OMB con la configurazione del carico di lavoro descritta sopra e il driver Sync.

   ```
   Sudo bin/benchmark –drivers driver-kafka/kafka- sync.yaml workloads/1-
   topic-100-partitions-1kb.yaml
   ```

3. È stato attivato un altro carico di lavoro con il driver di throughput con la stessa configurazione del carico di lavoro.

   ```
   sudo bin/benchmark –drivers driver-kafka/kafka-throughput.yaml
   workloads/1-topic-100-partitions-1kb.yaml
   ```

Osservazione

Sono stati utilizzati due diversi tipi di driver per generare carichi di lavoro per confrontare le performance di un'istanza di Kafka in esecuzione su NFS. La differenza tra i driver è la proprietà di scaricamento dei log.

Per una coppia Cloud Volumes ONTAP ha:

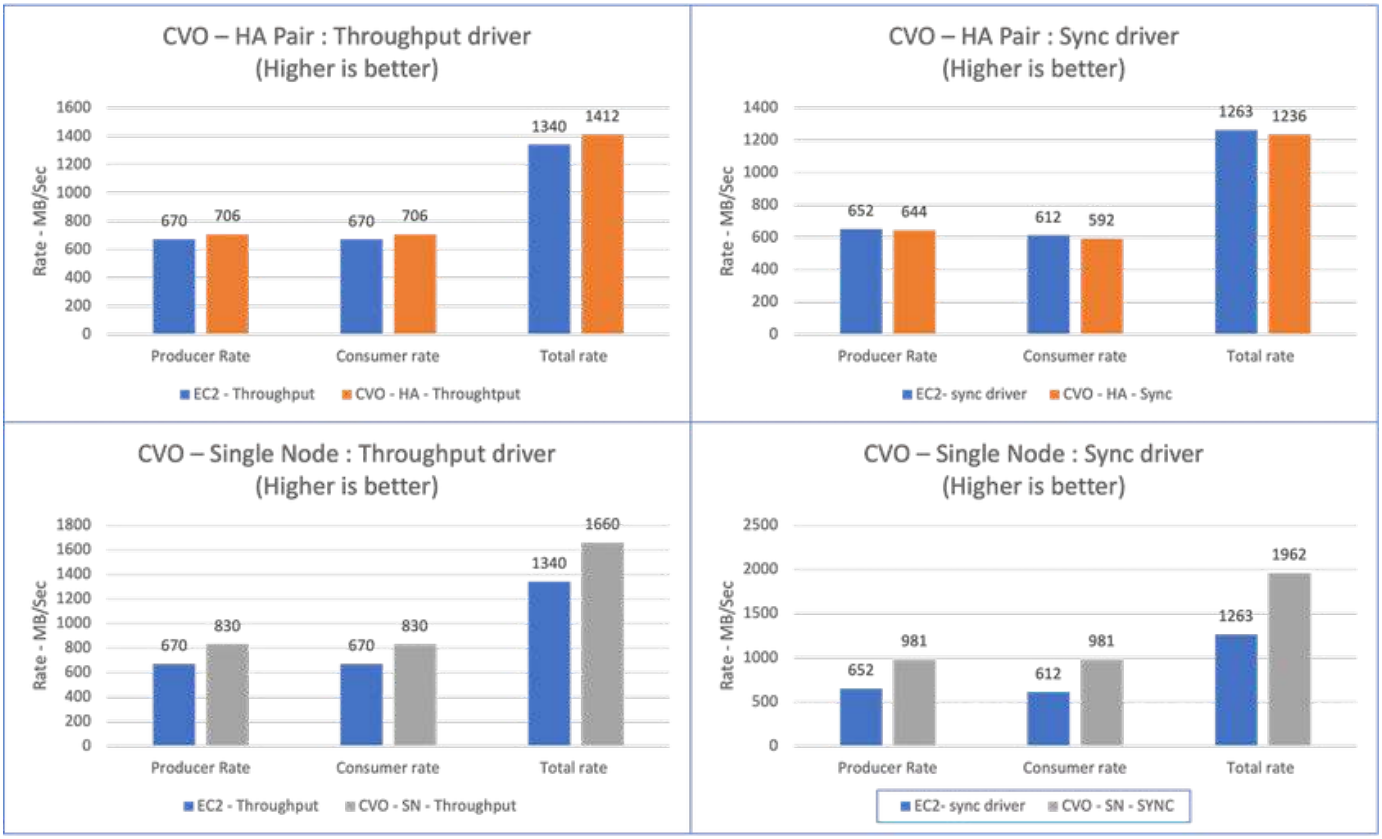
- Throughput totale generato in modo coerente dal driver Sync: ~1236 Mbps.
- Throughput totale generato per il driver di throughput: Picco ~1412 Mbps.

Per un singolo nodo Cloud Volumes ONTAP:

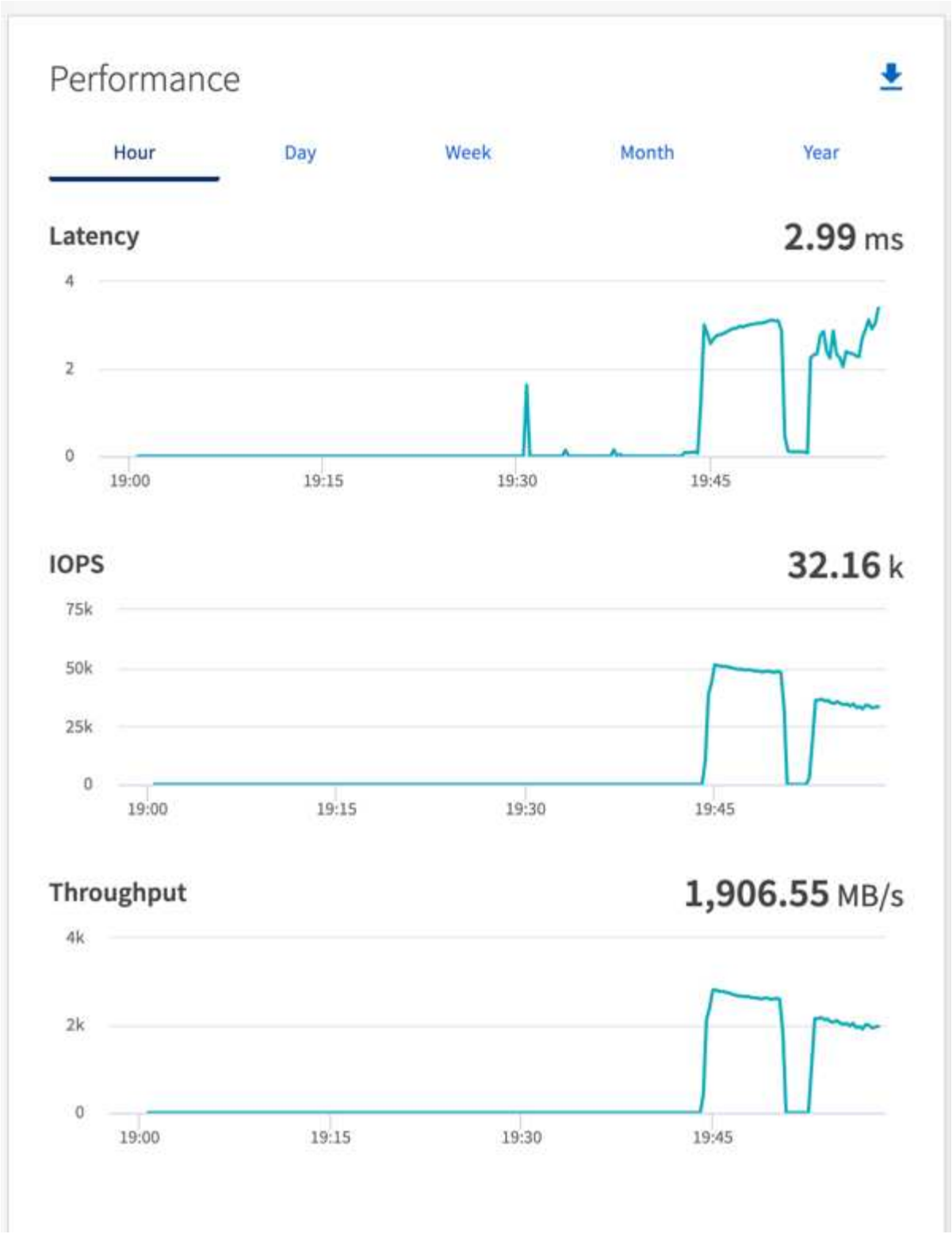
- Throughput totale generato in modo coerente dal driver Sync: ~ 1962 MBps.
- Throughput totale generato dal driver di throughput: Picco ~1660 MBps

Il driver Sync è in grado di generare un throughput coerente quando i log vengono trasferiti istantaneamente sul disco, mentre il driver di throughput genera burst di throughput quando i log vengono impegnati su disco in massa.

Questi numeri di throughput vengono generati per la configurazione AWS specificata. Per requisiti di performance più elevati, i tipi di istanze possono essere scalati e ottimizzati ulteriormente per ottenere numeri di throughput migliori. Il throughput totale o il tasso totale è la combinazione di un tasso di produttore e di consumo.



Verificare il throughput dello storage durante l'esecuzione del benchmarking del throughput o del driver di sincronizzazione.



Panoramica e convalida delle performance in AWS FSX per NetApp ONTAP

Un cluster Kafka con il layer di storage montato su NFS NetApp è stato sottoposto a benchmark per le performance in AWS FSX per NetApp ONTAP. Gli esempi di benchmarking sono descritti nelle sezioni seguenti.

Apache Kafka in AWS FSX per NetApp ONTAP

Network file System (NFS) è un file system di rete ampiamente utilizzato per la memorizzazione di grandi quantità di dati. Nella maggior parte delle organizzazioni i dati vengono sempre più generati da applicazioni di streaming come Apache Kafka. Questi carichi di lavoro richiedono scalabilità, bassa latenza e una solida architettura di acquisizione dei dati con moderne funzionalità di storage. Per consentire l'analisi in tempo reale e fornire informazioni utili, è necessaria un'infrastruttura ben progettata e dalle performance elevate.

Kafka di progettazione funziona con file system compatibile con POSIX e si affida al file system per gestire le operazioni sui file, ma quando si memorizzano i dati su un file system NFSv3, il client NFS del broker Kafka può interpretare le operazioni sui file in modo diverso da un file system locale come XFS o Ext4. Un esempio comune è il ridenominazione di NFS Silly, che ha causato il fallimento dei broker Kafka durante l'espansione dei cluster e la riallocazione delle partizioni. Per far fronte a questa sfida, NetApp ha aggiornato il client NFS open-source Linux con le modifiche ora generalmente disponibili in RHEL8.7, RHEL9.1 e supportate dall'attuale versione di FSX per NetApp ONTAP, ONTAP 9.12.1.

Amazon FSX per NetApp ONTAP offre un file system NFS completamente gestito, scalabile e dalle performance elevate nel cloud. I dati Kafka su FSX per NetApp ONTAP possono scalare per gestire grandi quantità di dati e garantire la tolleranza agli errori. NFS offre gestione dello storage centralizzata e protezione dei dati per set di dati critici e sensibili.

Questi miglioramenti consentono ai clienti AWS di sfruttare FSX per NetApp ONTAP quando eseguono carichi di lavoro Kafka su servizi di calcolo AWS. Questi vantaggi sono:
* Riduzione dell'utilizzo della CPU per ridurre i tempi di attesa i/O.
* Tempi di recovery più rapidi per i broker Kafka.
* Affidabilità ed efficienza.
* Scalabilità e performance.
* Disponibilità multi-Availability zone.
* Protezione dei dati.

Panoramica e convalida delle performance in AWS FSX per NetApp ONTAP

Un cluster Kafka con il layer di storage montato su NetApp NFS è stato sottoposto a benchmark per le performance nel cloud AWS. Gli esempi di benchmarking sono descritti nelle sezioni seguenti.

Kafka in AWS FSX per NetApp ONTAP

Un cluster Kafka con AWS FSX per NetApp ONTAP è stato sottoposto a benchmark per le performance nel cloud AWS. Questo benchmarking è descritto nelle sezioni seguenti.

Configurazione architetturale

La seguente tabella mostra la configurazione ambientale per un cluster Kafka che utilizza AWS FSX per NetApp ONTAP.

Componente della piattaforma	Configurazione dell'ambiente
Kafka 3.2.3	• 3 zookeeper – t2.small • 3 server di broker – i3en.2xlarge • 1 x Grafana – c5n.2xlarge • 4 x produttore/consumatore — c5n.2xlargo *
Sistema operativo su tutti i nodi	RHEL8.6

Componente della piattaforma	Configurazione dell'ambiente
AWS FSX per NetApp ONTAP	Multi-AZ con throughput di 4 GB/sec e 160000 IOPS

Configurazione di NetApp FSX per NetApp ONTAP

1. Per il test iniziale, abbiamo creato un file system FSX per NetApp ONTAP con 2 TB di capacità e 40000 IOPS per un throughput di 2 GB/sec.

   ```
   [root@ip-172-31-33-69 ~]# aws fsx create-file-system --region us-east-2
   --storage-capacity 2048 --subnet-ids <desired subnet 1> subnet-<desired
   subnet 2> --file-system-type ONTAP --ontap-configuration
   DeploymentType=MULTI_AZ_HA_1,ThroughputCapacity=2048,PreferredSubnetId=<
   desired primary subnet>,FsxAdminPassword=<new
   password>,DiskIopsConfiguration="{Mode=USER_PROVISIONED,Iops=40000"}
   ```

   Nel nostro esempio, stiamo implementando FSX per NetApp ONTAP attraverso l'interfaccia CLI AWS. Sarà necessario personalizzare ulteriormente il comando nell'ambiente in base alle esigenze. FSX per NetApp ONTAP può inoltre essere implementato e gestito tramite la console AWS per un'esperienza di implementazione più semplice e ottimizzata con meno input dalla riga di comando.

   Documentazione in FSX per NetApp ONTAP, il numero massimo di IOPS ottenibili per un file system con throughput di 2 GB/sec nella nostra area di test (US-Est-1) è 80,000 iops. Il totale massimo di iops per un file system FSX per NetApp ONTAP è di 160,000 iops, che richiede un'implementazione di throughput di 4 GB/sec per ottenere il risultato che verrà dimostrato più avanti in questo documento.

   Per ulteriori informazioni sulle specifiche delle prestazioni di FSX per NetApp ONTAP, visita la documentazione di AWS FSX per NetApp ONTAP qui: https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/performance.html .

   La sintassi dettagliata della riga di comando per FSX "create-file-system" è disponibile qui: https://docs.aws.amazon.com/cli/latest/reference/fsx/create-file-system.html

   Ad esempio, è possibile specificare una chiave KMS specifica invece della chiave master AWS FSX predefinita utilizzata quando non viene specificata alcuna chiave KMS.

2. Durante la creazione del file system FSX per NetApp ONTAP, attendere che lo stato "Lifecycle" (ciclo di vita) passi a "AVAILABLE" (DISPONIBILE) nel ritorno JSON dopo aver descritto il file system come segue:

   ```
   [root@ip-172-31-33-69 ~]# aws fsx describe-file-systems  --region us-
   east-1 --file-system-ids fs-02ff04bab5ce01c7c
   ```

3. Convalidare le credenziali effettuando l'accesso a FSX per SSH NetApp ONTAP con l'utente fsxadmin: Fsxadmin è l'account admin predefinito per FSX per i filesystem NetApp ONTAP al momento della creazione. La password per fsxadmin è la password che è stata configurata durante la prima creazione del file system nella console AWS o con l'interfaccia CLI AWS, come è stato completato nella fase 1.

```
[root@ip-172-31-33-69 ~]# ssh fsxadmin@198.19.250.244
The authenticity of host '198.19.250.244 (198.19.250.244)' can't be
established.
ED25519 key fingerprint is
SHA256:mgCyRXJfWRc2d/jOjFbMBsUcYOWjxoIky0ltHvVDL/Y.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '198.19.250.244' (ED25519) to the list of
known hosts.
(fsxadmin@198.19.250.244) Password:

This is your first recorded login.
```

4. Una volta convalidate le credenziali, creare la macchina virtuale di storage sul file system FSX per NetApp ONTAP

```
[root@ip-172-31-33-69 ~]# aws fsx --region us-east-1 create-storage-
virtual-machine --name svmkafkatest --file-system-id fs-
02ff04bab5ce01c7c
```

Una macchina virtuale per lo storage (SVM) è un file server isolato con le proprie credenziali amministrative ed endpoint per l'amministrazione e l'accesso ai dati in FSX per i volumi NetApp ONTAP e fornisce FSX per il multi-tenancy NetApp ONTAP.

5. Una volta configurata la macchina virtuale di storage primaria, SSH nel nuovo file system FSX per NetApp ONTAP e creare volumi nella macchina virtuale di storage utilizzando il comando di esempio riportato di seguito. Analogamente, creiamo 6 volumi per questa convalida. In base alla nostra convalida, mantenere il costituente predefinito (8) o un numero inferiore di costituenti che forniranno prestazioni migliori a kafka.

```
FsxId02ff04bab5ce01c7c::*> volume create -volume kafkafsxN1 -state
online -policy default -unix-permissions ---rwxr-xr-x -junction-active
true -type RW -snapshot-policy none  -junction-path /kafkafsxN1 -aggr
-list aggr1
```

6. Per i nostri test, abbiamo bisogno di capacità aggiuntiva nei nostri volumi. Estendere le dimensioni del volume a 2 TB e montarlo sul percorso di giunzione.

```
FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN1 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN1" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN2 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN2" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN3 -new-size +2TB
```

```
vol size: Volume "svmkafkatest:kafkafsxN3" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN4 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN4" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN5 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN5" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN6 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN6" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume show -vserver svmkafkatest -volume *
Vserver   Volume       Aggregate    State      Type       Size
Available Used%
--------- ------------ ------------ ---------- ---- ----------
---------- -----
svmkafkatest
          kafkafsxN1   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN2   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN3   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN4   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN5   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN6   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          svmkafkatest_root
                       aggr1        online     RW          1GB
968.1MB    0%
7 entries were displayed.

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN1 -junction
-path /kafkafsxN1

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN2 -junction
-path /kafkafsxN2
```

```
FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN3 -junction
-path /kafkafsxN3

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN4 -junction
-path /kafkafsxN4

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN5 -junction
-path /kafkafsxN5

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN6 -junction
-path /kafkafsxN6
```

In FSX per NetApp ONTAP, è possibile eseguire il thin provisioning dei volumi. Nel nostro esempio, la capacità totale del volume esteso supera la capacità totale del file system, quindi sarà necessario estendere la capacità totale del file system per sbloccare la capacità aggiuntiva del volume sottoposto a provisioning, come illustrato nella fase successiva.

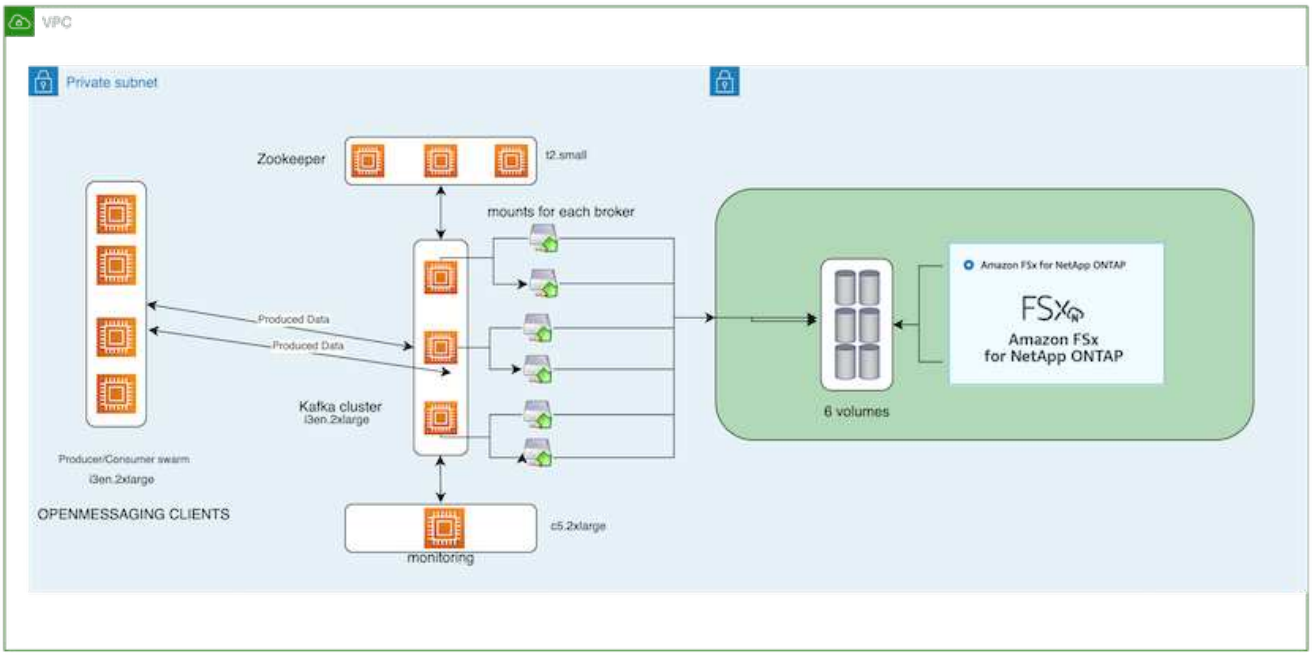
7. Inoltre, per ottenere maggiori performance e capacità, estendiamo la capacità di throughput FSX per NetApp ONTAP da 2 GB/sec a 4 GB/sec e IOPS a 160000 e la capacità a 5 TB

```
[root@ip-172-31-33-69 ~]# aws fsx update-file-system --region us-east-1
--storage-capacity 5120 --ontap-configuration
'ThroughputCapacity=4096,DiskIopsConfiguration={Mode=USER_PROVISIONED,Io
ps=160000}' --file-system-id fs-02ff04bab5ce01c7c
```

La sintassi dettagliata della riga di comando per FSX "update-file-system" è disponibile qui: https://docs.aws.amazon.com/cli/latest/reference/fsx/update-file-system.html

8. I volumi FSX per NetApp ONTAP sono montati con opzioni nconnect e predefinite nei broker Kafka

La seguente immagine mostra l'architettura finale di un cluster Kafka basato su FSX per NetApp ONTAP:



- Calcolo. Abbiamo utilizzato un cluster Kafka a tre nodi con un gruppo di zookeeper a tre nodi in esecuzione su server dedicati. Ciascun broker disponeva di sei punti di montaggio NFS su sei volumi nell'istanza FSX per NetApp ONTAP.
- Monitoraggio. Abbiamo utilizzato due nodi per una combinazione Prometheus-Grafana. Per la generazione dei carichi di lavoro, abbiamo utilizzato un cluster a tre nodi separato in grado di produrre e utilizzare questo cluster Kafka.
- Storage. Abbiamo utilizzato un FSX per NetApp ONTAP con sei volumi da 2 TB montati. Il volume è stato quindi esportato nel broker Kafka con un montaggio NFS. I volumi FSX per NetApp ONTAP sono montati con 16 sessioni Nconnect e opzioni predefinite nei broker Kafka.

Configurazioni di benchmarking di OpenMessage.

Abbiamo utilizzato la stessa configurazione utilizzata per NetApp Cloud Volumes ONTAP e i relativi dettagli sono qui - https://docs.netapp.com/us-en/netapp-solutions/data-analytics/kafka-nfs-performance-overview-and-validation-in-aws.html#architectural-setup

Metodologia di test

1. È stato eseguito il provisioning di un cluster Kafka in base alle specifiche descritte in precedenza utilizzando Terraform e ansible. Il terraform viene utilizzato per costruire l'infrastruttura utilizzando istanze AWS per il cluster Kafka e ansible crea il cluster Kafka su di essi.
2. È stato attivato un carico di lavoro OMB con la configurazione del carico di lavoro descritta sopra e il driver Sync.

```
sudo bin/benchmark -drivers driver-kafka/kafka-sync.yaml workloads/1-topic-100-partitions-1kb.yaml
```

3. È stato attivato un altro carico di lavoro con il driver di throughput con la stessa configurazione del carico di lavoro.

```
sudo bin/benchmark -drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

Osservazione

Sono stati utilizzati due diversi tipi di driver per generare carichi di lavoro per confrontare le performance di un'istanza di Kafka in esecuzione su NFS. La differenza tra i driver è la proprietà di scaricamento dei log.

Per un fattore di replica Kafka 1 e FSX per NetApp ONTAP:

- Throughput totale generato in modo coerente dal driver Sync: ~ 3218 Mbps e performance di picco in ~ 3652 Mbps.
- Throughput totale generato in modo coerente dal driver di throughput: ~ 3679 Mbps e performance di picco in ~ 3908 Mbps.

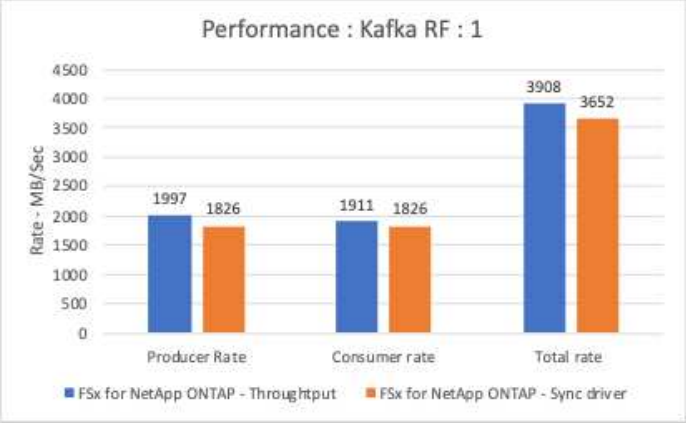
Per Kafka con fattore di replica 3 e FSX per NetApp ONTAP :

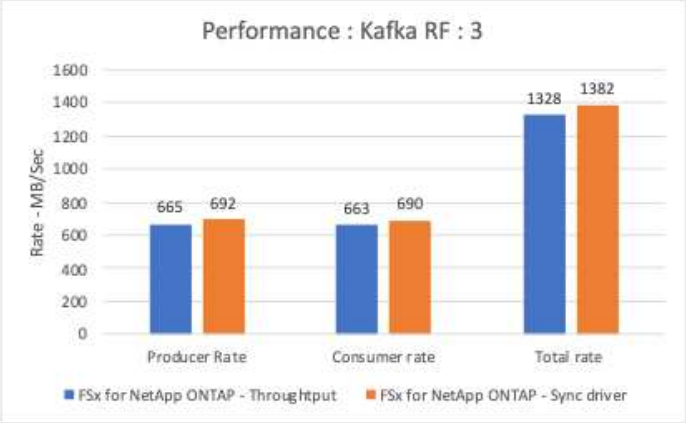
- Throughput totale generato in modo coerente dal driver Sync: ~ 1252 Mbps e performance di picco in ~ 1382 Mbps.
- Throughput totale generato in modo coerente dal driver di throughput: ~ 1218 Mbps e performance di picco in ~ 1328 Mbps.

Nel fattore 3 di replica di Kafka, l'operazione di lettura e scrittura è stata eseguita tre volte su FSX per NetApp ONTAP, nel fattore 1 di replica di Kafka, l'operazione di lettura e scrittura è una volta su FSX per NetApp ONTAP, quindi in entrambe le procedure di convalida, Siamo in grado di raggiungere il throughput massimo di 4 GB/sec.

Il driver Sync è in grado di generare un throughput coerente quando i log vengono trasferiti istantaneamente sul disco, mentre il driver di throughput genera burst di throughput quando i log vengono impegnati su disco in massa.

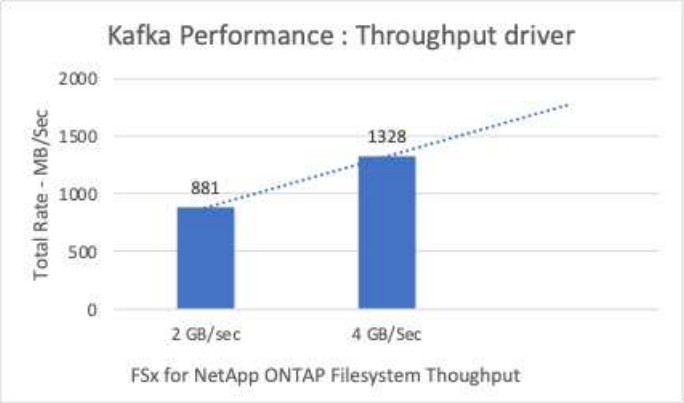
Questi numeri di throughput vengono generati per la configurazione AWS specificata. Per requisiti di performance più elevati, i tipi di istanze possono essere scalati e ottimizzati ulteriormente per ottenere numeri di throughput migliori. Il throughput totale o il tasso totale è la combinazione di un tasso di produttore e di consumo.

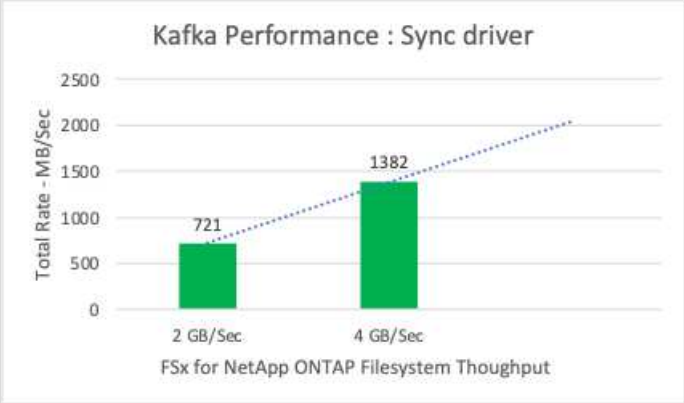




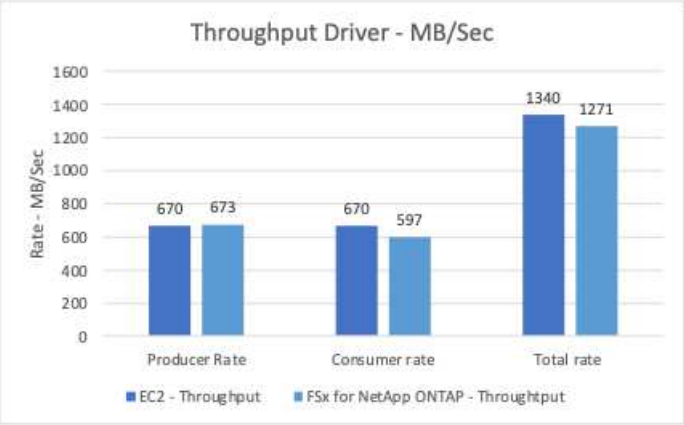
Il grafico riportato di seguito mostra le prestazioni FSX da 2 GB/sec per NetApp ONTAP e da 4 GB/sec per il fattore di replica Kafka 3. Il fattore di replica 3 esegue tre volte l'operazione di lettura e scrittura su FSX per lo

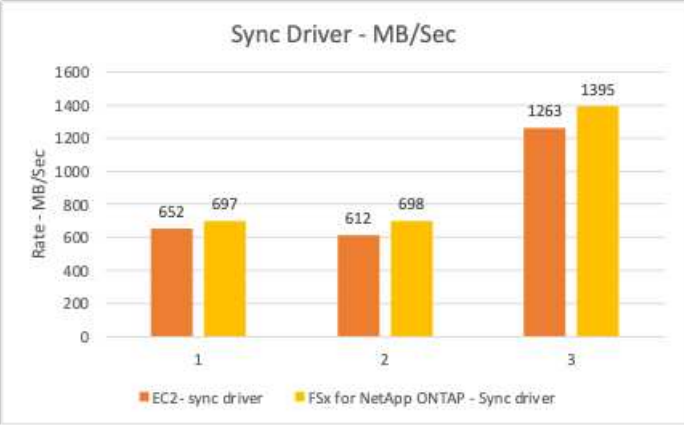
storage NetApp ONTAP. La velocità totale per il driver di throughput è di 881 MB/sec, che esegue operazioni di lettura e scrittura Kafka di circa 2.64 GB/sec sul file system FSX da 2 GB/sec per NetApp ONTAP, mentre la velocità totale per il driver di throughput è di 1328 MB/sec che esegue operazioni di lettura e scrittura kafka di circa 3.98 GB/sec. Le performance di Kafka sono lineari e scalabili in base al throughput FSX per NetApp ONTAP.





Il grafico seguente mostra le performance tra l'istanza EC2 e FSX per NetApp ONTAP (fattore di replica Kafka: 3)

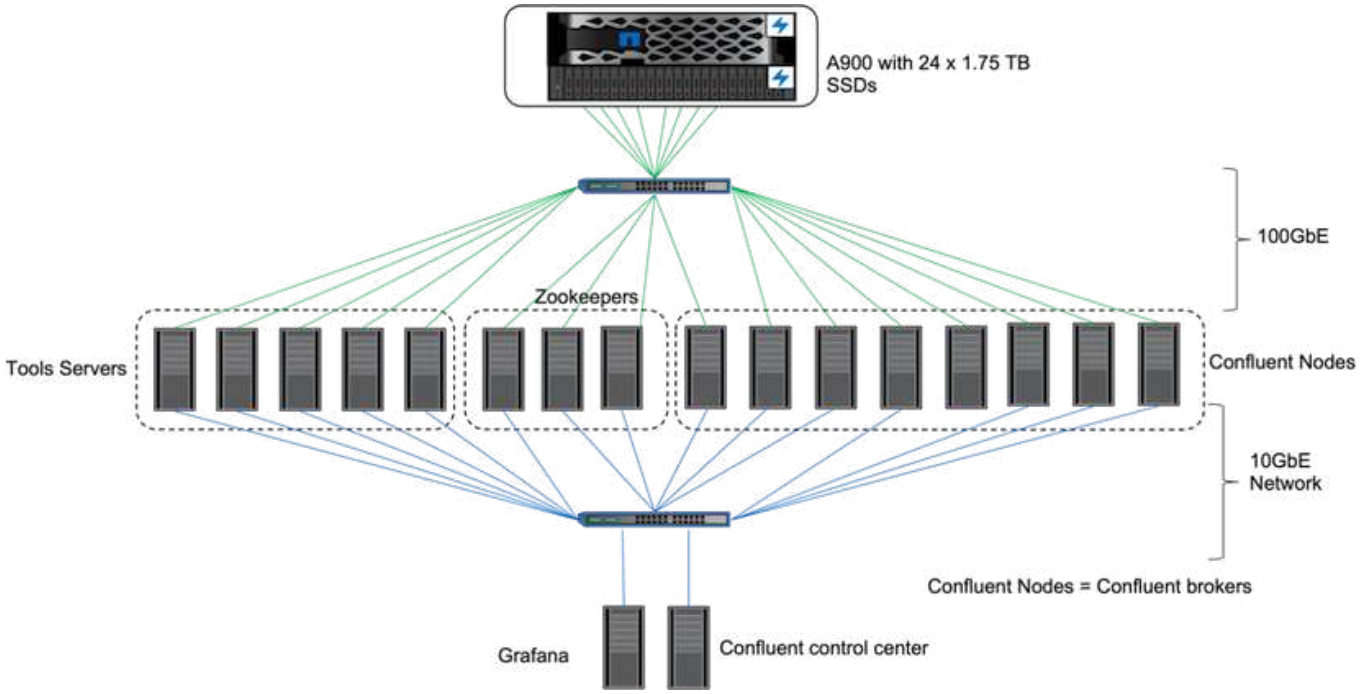




Panoramica e validazione delle performance con AFF A900 on-premise

On-premise, abbiamo utilizzato il controller di storage NetApp AFF A900 con ONTAP 9.12.1RC1 per convalidare le performance e la scalabilità di un cluster Kafka. Abbiamo utilizzato lo stesso tested delle Best practice per lo storage a più livelli precedenti con ONTAP e AFF.

Abbiamo utilizzato Confluent Kafka 6.2.0 per valutare AFF A900. Il cluster include otto nodi di broker e tre nodi di zookeeper. Per il test delle performance, abbiamo utilizzato cinque nodi di lavoro OMB.



Configurazione dello storage

Abbiamo utilizzato le istanze di NetApp FlexGroups per fornire un singolo namespace per le directory di log, semplificando il ripristino e la configurazione. Abbiamo utilizzato NFSv4.1 e pNFS per fornire accesso diretto al percorso ai dati del segmento di registro.

Tuning del client

Ogni client ha montato l'istanza di FlexGroup con il seguente comando.

```
mount -t nfs -o vers=4.1,nconnect=16 172.30.0.121:/kafka_vol01
/data/kafka_vol01
```

Inoltre, abbiamo aumentato il `max_session_slots` dal valore predefinito `64` a. `180`. Corrisponde al limite predefinito di slot di sessione in ONTAP.

Messa a punto del broker Kafka

Per massimizzare il throughput nel sistema sottoposto a test, abbiamo aumentato significativamente i parametri predefiniti per alcuni pool di thread chiave. Si consiglia di seguire le Best practice di Confluent Kafka per la maggior parte delle configurazioni. Questo tuning è stato utilizzato per massimizzare la concorrenza tra i/o in sospeso e storage. Questi parametri possono essere regolati in modo da corrispondere alle risorse di calcolo e agli attributi di storage del broker.

```
num.io.threads=96
num.network.threads=96
background.threads=20
num.replica.alter.log.dirs.threads=40
num.replica.fetchers=20
queued.max.requests=2000
```

Metodologia di test del generatore di workload

Abbiamo utilizzato le stesse configurazioni OMB utilizzate per i test cloud per il driver di throughput e la configurazione degli argomenti.

1. È stato eseguito il provisioning di un’istanza di FlexGroup utilizzando Ansible su un cluster AFF.

```
---
- name: Set up kafka broker processes
  hosts: localhost
  vars:
    ntap_hostname: ‘hostname’
    ntap_username: 'user'
    ntap_password: 'password'
    size: 10
    size_unit: tb
    vserver: vs1
    state: present
    https: true
    export_policy: default
    volumes:
      - name: kafka_fg_vol01
        aggr: ["aggr1_a", "aggr2_a", “aggr1_b”, “aggr2_b”]
        path: /kafka_fg_vol01
  tasks:
    - name: Edit volumes
      netapp.ontap.na_ontap_volume:
        state: "{{ state }}"
        name: "{{ item.name }}"
        aggr_list: "{{ item.aggr }}"
        aggr_list_multiplier: 8
        size: "{{ size }}"
        size_unit: "{{ size_unit }}"
        vserver: "{{ vserver }}"
        snapshot_policy: none
        export_policy: default
        junction_path: "{{ item.path }}"
        qos_policy_group: none
        wait_for_completion: True
        hostname: "{{ ntap_hostname }}"
        username: "{{ ntap_username }}"
        password: "{{ ntap_password }}"
        https: "{{ https }}"
        validate_certs: false
      connection: local
      with_items: "{{ volumes }}"
```

2. PNFS è stato attivato su ONTAP SVM.

```
vserver modify -vserver vs1 -v4.1-pnfs enabled -tcp-max-xfer-size 262144
```

3. Il carico di lavoro è stato attivato con il driver di throughput utilizzando la stessa configurazione del carico di lavoro di Cloud Volumes ONTAP. Vedere la sezione "Performance a stato stazionario" sotto. Il carico di lavoro utilizzava un fattore di replica di 3, il che significa che in NFS sono state mantenute tre copie di segmenti di log.

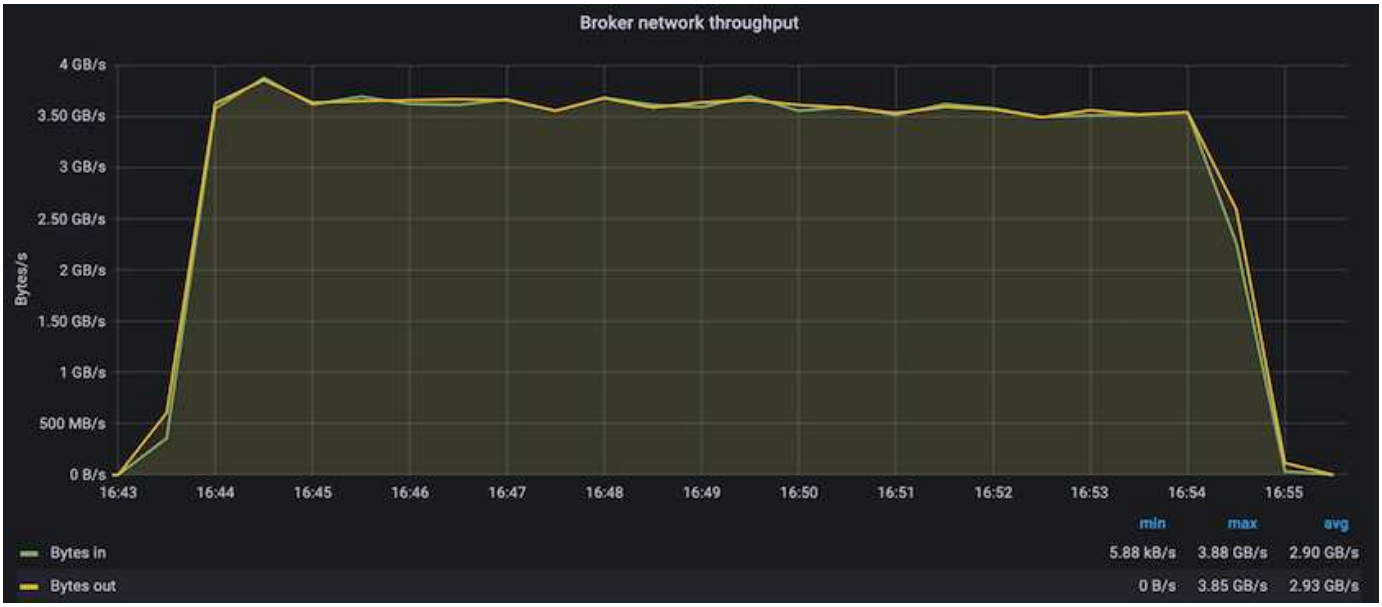
```
sudo bin/benchmark --drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

4. Infine, abbiamo completato le misurazioni utilizzando un backlog per misurare la capacità dei consumatori di recuperare i messaggi più recenti. OMB crea un backlog mettendo in pausa i consumatori durante l'inizio di una misurazione. Ciò produce tre fasi distinte: Creazione di backlog (traffico solo produttore), eliminazione dei backlog (una fase pesante per i consumatori in cui i consumatori si mettono al passo con gli eventi persi in un argomento) e lo stato stazionario. Vedere la sezione "analisi dei limiti dello storage" per ulteriori informazioni.

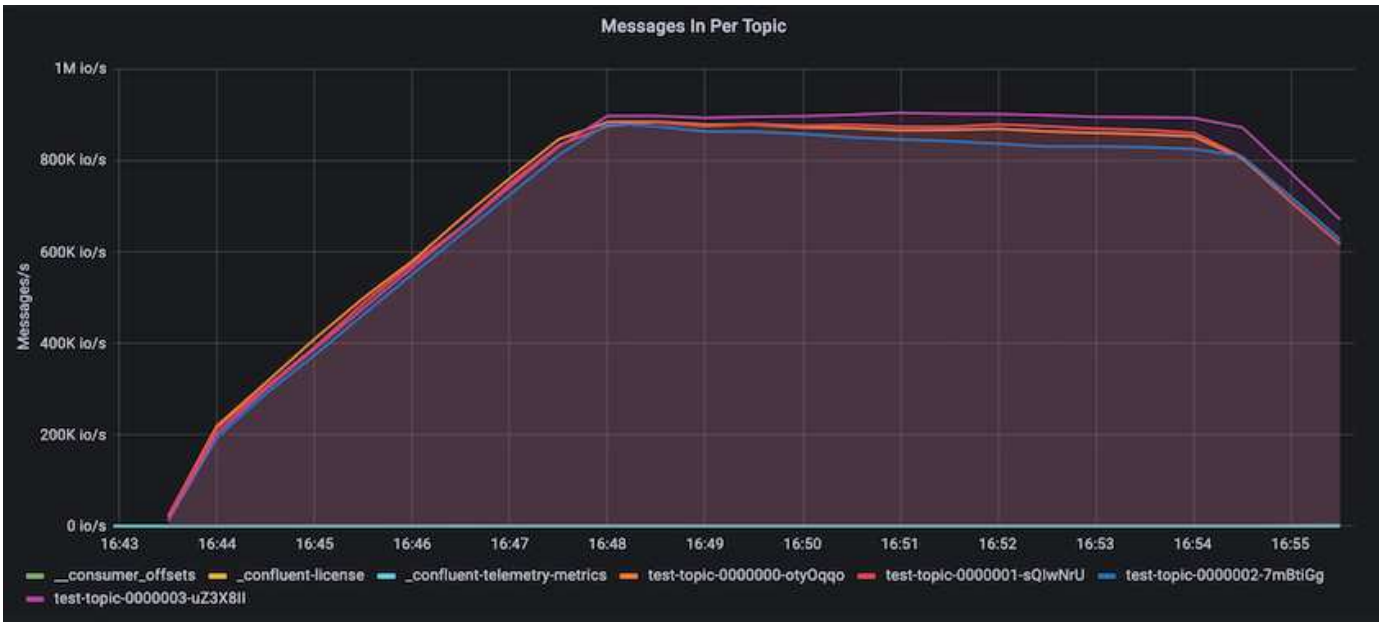
Performance a stato stazionario

Abbiamo valutato AFF A900 utilizzando il benchmark OpenMessaging per fornire un confronto simile a Cloud Volumes ONTAP in AWS e DAS in AWS. Tutti i valori delle performance rappresentano il throughput del cluster Kafka a livello di produttore e consumatore.

Le performance a stato stazionario con Confluent Kafka e AFF A900 hanno raggiunto un throughput medio di oltre 3,4 Gbps sia per i produttori che per i consumatori. Si tratta di oltre 3.4 milioni di messaggi nel cluster Kafka. Visualizzando il throughput sostenuto in byte al secondo per BrokerTopicMetrics, vediamo le eccellenti performance di stato stazionario e il traffico supportato da AFF A900.



Questo si allinea perfettamente con la visualizzazione dei messaggi inviati per argomento. Il seguente grafico fornisce una descrizione dettagliata per argomento. Nella configurazione testata abbiamo visto quasi 900.000 messaggi per argomento in quattro argomenti.

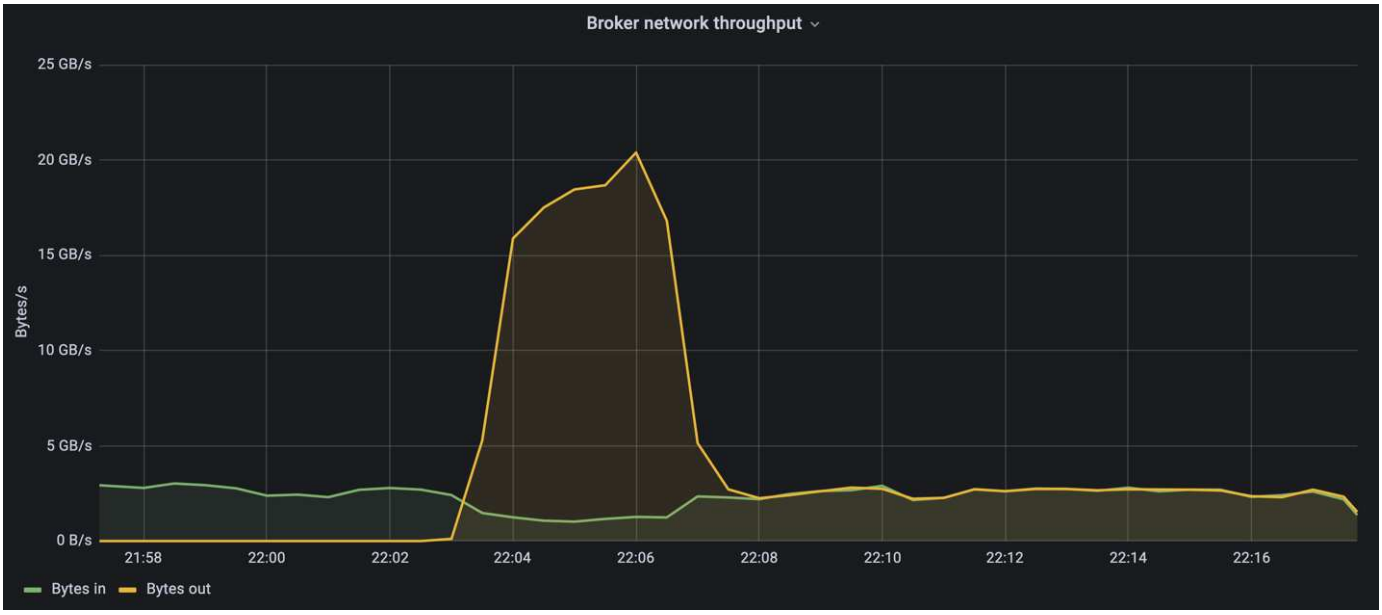


Performance estreme e analisi dei limiti dello storage

Per AFF, abbiamo anche testato con OMB utilizzando la funzionalità di backlog. La funzionalità di backlog mette in pausa gli abbonamenti consumer mentre nel cluster Kafka viene creato un backlog di eventi. Durante questa fase, si verifica solo il traffico del produttore, che genera eventi che vengono impegnati nei registri. In questo modo si emulano più da vicino i flussi di lavoro di elaborazione batch o di analisi offline; in questi flussi di lavoro, le sottoscrizioni consumer vengono avviate e devono leggere i dati storici che sono già stati rimossi dalla cache del broker.

Per comprendere le limitazioni dello storage sul throughput consumer in questa configurazione, abbiamo misurato la fase solo produttore per capire quanto traffico di scrittura potrebbe assorbire l'A900. Vedere la sezione successiva "Guida al dimensionamento" per capire come sfruttare questi dati.

Durante la parte solo produttore di questa misurazione, abbiamo riscontrato un throughput elevato che ha spinto i limiti delle performance di A900 (quando le altre risorse di broker non erano sature e il traffico consumer e dei produttori).



Abbiamo aumentato le dimensioni del messaggio a 16.000 per questa misurazione per limitare le spese generali per messaggio e massimizzare il throughput dello storage ai punti di montaggio NFS.

```
messageSize: 16384
consumerBacklogSizeGB: 4096
```

Il cluster Confluent Kafka ha raggiunto un picco di throughput dei produttori di 4,03 Gbps.

```
18:12:23.833 [main] INFO WorkloadGenerator - Pub rate 257759.2 msg/s /
4027.5 MB/s | Pub err     0.0 err/s …
```

Dopo che OMB ha completato il popolamento dell'eventbacklog, il traffico consumer è stato riavviato. Durante le misurazioni con il deflusso del backlog, abbiamo osservato un throughput dei consumatori di oltre 20 Gbps in tutti gli argomenti. Il throughput combinato per il volume NFS che memorizza i dati di log OMB si avvicinava a ~30 Gbps.

Guida al dimensionamento

Amazon Web Services offre un "guida al dimensionamento" Per il dimensionamento e la scalabilità dei cluster Kafka.

Questo dimensionamento fornisce una formula utile per determinare i requisiti di throughput dello storage per il cluster Kafka:

Per un throughput aggregato prodotto nel cluster di tcluster con un fattore di replica di r, il throughput ricevuto dallo storage del broker è il seguente:

```
t[storage] = t[cluster]/#brokers + t[cluster]/#brokers * (r-1)
          = t[cluster]/#brokers * r
```

Questo può essere ulteriormente semplificato:

```
max(t[cluster]) <= max(t[storage]) * #brokers/r
```

Questa formula consente di selezionare la piattaforma ONTAP appropriata per le tue esigenze di hot Tier Kafka.

La seguente tabella illustra il throughput previsto dal produttore per l'A900 con diversi fattori di replica:

Fattore di replica	Throughput produttore (GPPS)
3 (misurato)	3.4
2	5.1
1	10.2

Conclusione

La soluzione NetApp per il problema del ridenominazione sciocco offre una forma di storage semplice, economica e gestita centralmente per carichi di lavoro che in precedenza erano incompatibili con NFS.

Questo nuovo paradigma consente ai clienti di creare cluster Kafka più gestibili che siano più facili da migrare e eseguire il mirroring ai fini del disaster recovery e della protezione dei dati.
Abbiamo anche visto che NFS offre ulteriori vantaggi, come la riduzione dell'utilizzo della CPU e un tempo di recovery più rapido, un notevole miglioramento dell'efficienza dello storage e migliori performance grazie a NetApp ONTAP.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Che cos'è Apache Kafka?

  "https://www.confluent.io/what-is-apache-kafka/"

- Che cos'è un ridenominazione sciocco?

  "https://linux-nfs.org/wiki/index.php/Server-side_silly_rename"

- ONATP viene letto per le applicazioni di streaming.

  "https://www.netapp.com/blog/ontap-ready-for-streaming-applications/"

- Stupido: Rinominare il problema con Kafka.

  "https://sbg.technology/2018/07/10/kafka-nfs/"

- Documentazione sui prodotti NetApp

  "https://www.netapp.com/support-and-training/documentation/"

- Che cos'è NFS?

  "https://en.wikipedia.org/wiki/Network_File_System"

- Cos'è la riassegnazione delle partizioni Kafka?

  "https://docs.cloudera.com/runtime/7.2.10/kafka-managing/topics/kafka-manage-cli-reassign-overview.html"

- Che cos'è il benchmark OpenMessaging?

  "https://openmessaging.cloud/"

- Come migrare un broker Kafka?

  "https://medium.com/@sanchitbansal26/how-to-migrate-kafka-cluster-with-no-downtime-58c216129058"

- Come monitorate il broker Kafka con Prometheus?

  https://www.confluent.io/blog/monitor-kafka-clusters-with-prometheus-grafana-and-confluent/

- Piattaforma gestita per Apache Kafka

  https://www.instaclustr.com/platform/managed-apache-kafka/

- Supporto per Apache Kafka

  https://www.instaclustr.com/support-solutions/kafka-support/

- Servizi di consulenza per Apache Kafka

  https://www.instaclustr.com/services/consulting/

Confluenza di Kafka con i controller di storage NetApp ONTAP

TR-4941: Confluenza con i controller di storage NetApp ONTAP

Karthikeyan Nagalingam, Joe Scott, NetApp Rankesh Kumar, Confluent

Per rendere la piattaforma Confluent più scalabile ed elastica, l'IT deve essere in grado di scalare e bilanciare i carichi di lavoro molto rapidamente. Lo storage a più livelli consente di gestire l'archiviazione di enormi volumi di dati in Confluent riducendo il carico operativo.

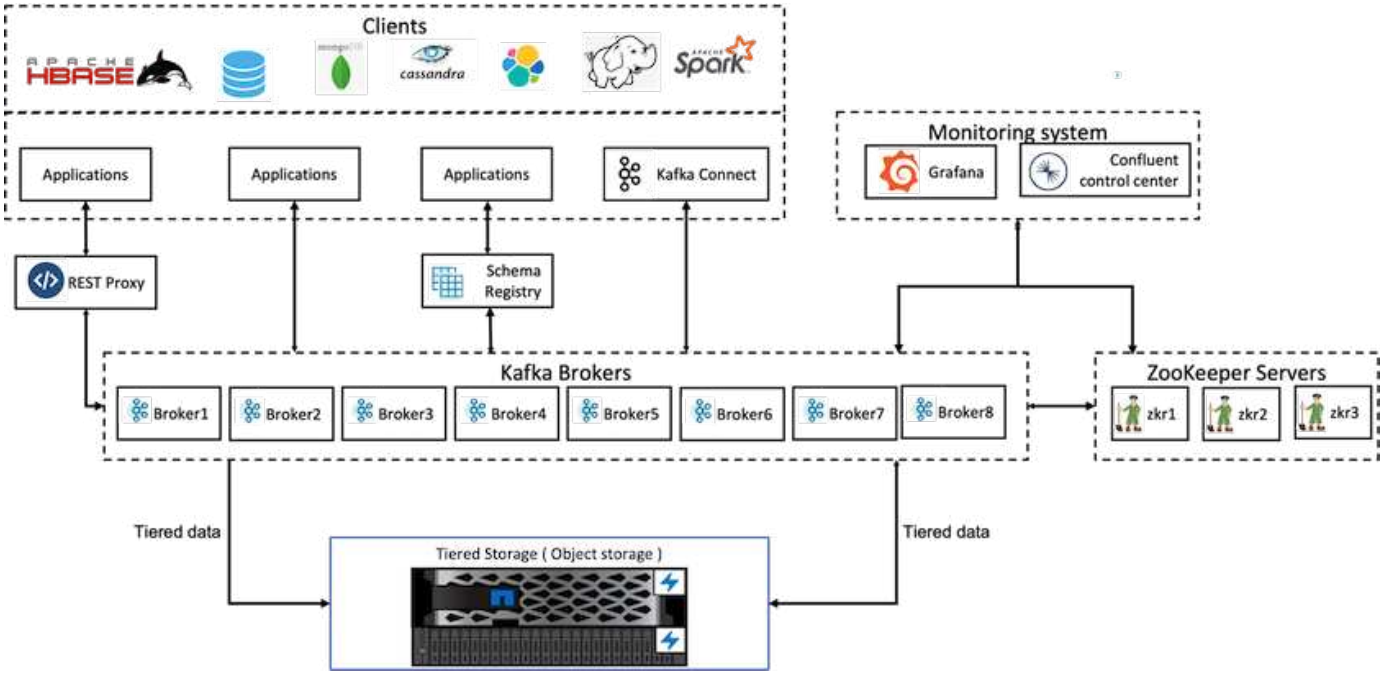
L'idea fondamentale è separare lo storage dei dati dall'elaborazione dei dati, il che rende molto più semplice la scalabilità di ciascuno di essi in modo indipendente.

Dotato di innovazioni leader del settore, il software per la gestione dei dati NetApp ONTAP offre a Confluent numerosi vantaggi ovunque i dati siano presenti.

Questo documento descrive i benchmark delle performance per la piattaforma Confluent su NetApp ONTAP utilizzando un kit di benchmarking per lo storage a più livelli.

Soluzione

I controller di storage Confluent e NetApp AFF A900 con tecnologia ONTAP sono sistemi distribuiti progettati per i flussi di dati. Entrambi sono scalabili orizzontalmente, tolleranti agli errori e offrono eccellenti prestazioni sotto carico. Si integrano a vicenda nello streaming di dati distribuiti e nell'elaborazione del flusso con costi di storage inferiori grazie a tecnologie per la riduzione dei dati che riducono al minimo l'impatto dei dati. Il controller di storage AFF A900 offre performance elevate, consentendo al contempo il disaccoppiamento delle risorse di calcolo e storage dei dati. Ciò semplifica l'amministrazione del sistema e consente di scalare le risorse in modo indipendente.



Dettagli sull'architettura della soluzione

Questa sezione descrive l'hardware e il software utilizzati per la verifica delle performance nell'implementazione della piattaforma confluente con NetApp ONTAP per lo storage su più livelli. La seguente tabella illustra l'architettura della soluzione e i componenti di base.

Componente della piattaforma	Configurazione dell'ambiente
Confluent Platform versione 6.2	• 3 zookeeper • 8 server di broker • 5 server di strumenti • 1 Grafana • 1 centro di controllo
Sistema operativo su tutti i nodi	Linux (Ubuntu 18.04)
NetApp ONTAP per i bucket Warm	• 1 coppia AFF A900 ad alta disponibilità (ha) • 4 SSD 24 x 800 • Protocollo S3 • 100 GbE
15 server Fujitsu PRIMERGY RX2540	• 2 CPU; 16 core fisici in totale • Intel Xeon • 256 GB di memoria fisica • Doppia porta 100 GbE

Panoramica della tecnologia

Questa sezione descrive la tecnologia utilizzata in questa soluzione.

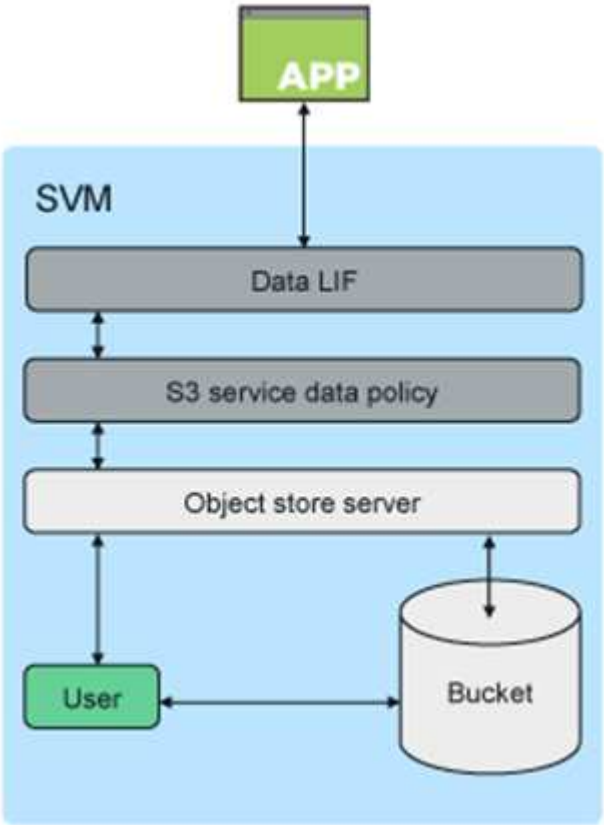
Storage controller NetApp ONTAP

NetApp ONTAP è un sistema operativo per lo storage Enterprise dalle performance elevate.

NetApp ONTAP 9.8 introduce il supporto per le API di Amazon Simple Storage Service (S3). ONTAP supporta un sottoinsieme di azioni API di Amazon Web Services (AWS) S3 e consente la rappresentazione dei dati come oggetti nei sistemi basati su ONTAP tra i cloud provider (AWS, Azure e GCP) e on-premise.

Il software NetApp StorageGRID è la soluzione NetApp di punta per lo storage a oggetti. ONTAP integra StorageGRID fornendo un punto di acquisizione e pre-elaborazione all'edge, espandendo il data fabric basato su NetApp per i dati a oggetti e aumentando il valore del portfolio di prodotti NetApp.

L'accesso a un bucket S3 viene fornito tramite applicazioni client e utente autorizzate. Il seguente diagramma mostra l'applicazione che accede a un bucket S3.



Casi di utilizzo principali

Lo scopo principale del supporto delle API S3 è fornire l'accesso agli oggetti su ONTAP. L'architettura di storage unificata di ONTAP ora supporta file (NFS e SMB), blocchi (FC e iSCSI) e oggetti (S3).

Applicazioni S3 native

Un numero crescente di applicazioni è in grado di sfruttare il supporto ONTAP per l'accesso a oggetti utilizzando S3. Sebbene sia adatto per carichi di lavoro di archiviazione ad alta capacità, la necessità di performance elevate nelle applicazioni S3 native sta crescendo rapidamente e include:

- Analytics
- Intelligenza artificiale
- Acquisizione edge-to-core
- Apprendimento automatico

I clienti possono ora utilizzare strumenti di gestione familiari come Gestore di sistema di ONTAP per eseguire rapidamente il provisioning dello storage a oggetti dalle performance elevate per lo sviluppo e le operazioni in ONTAP, sfruttando le efficienze e la sicurezza dello storage di ONTAP.

Endpoint FabricPool

A partire da ONTAP 9.8, FabricPool supporta il tiering dei bucket in ONTAP, consentendo il tiering ONTAP-ONTAP. Si tratta di un'opzione eccellente per i clienti che desiderano riutilizzare l'infrastruttura FAS esistente come endpoint dell'archivio di oggetti.

FabricPool supporta il tiering a ONTAP in due modi:

- **Tiering del cluster locale.** i dati inattivi vengono suddivisi in livelli in un bucket situato nel cluster locale utilizzando le LIF del cluster.
- **Tiering del cluster remoto.** i dati inattivi vengono suddivisi in livelli in un bucket situato su un cluster remoto in modo simile a un Tier cloud FabricPool tradizionale utilizzando le IC LIF sul client FabricPool e le LIF dei dati sull'archivio di oggetti ONTAP.

ONTAP S3 è adatto per le funzionalità S3 sui cluster esistenti senza hardware e gestione aggiuntivi. Per implementazioni superiori a 300 TB, il software NetApp StorageGRID continua a essere la soluzione NetApp di punta per lo storage a oggetti. Non è richiesta una licenza FabricPool quando si utilizza ONTAP o StorageGRID come livello cloud.

NetApp ONTAP per lo storage a più livelli confluente

Ogni data center deve garantire l'esecuzione delle applicazioni business-critical e la disponibilità e la sicurezza dei dati importanti. Il nuovo sistema NetApp AFF A900 è basato sul software ONTAP edizione Enterprise e su un design ad alta resilienza. Il nostro nuovo sistema di storage NVMe estremamente veloce elimina le interruzioni delle operazioni mission-critical, riduce al minimo l'ottimizzazione delle performance e protegge i dati dagli attacchi ransomware.

Dall'implementazione iniziale alla scalabilità del cluster Confluent, il tuo ambiente richiede un rapido adattamento alle modifiche senza interruzioni per le tue applicazioni business-critical. La gestione dei dati aziendali, la qualità del servizio (QoS) e le performance di ONTAP ti consentono di pianificare e adattarsi al tuo ambiente.

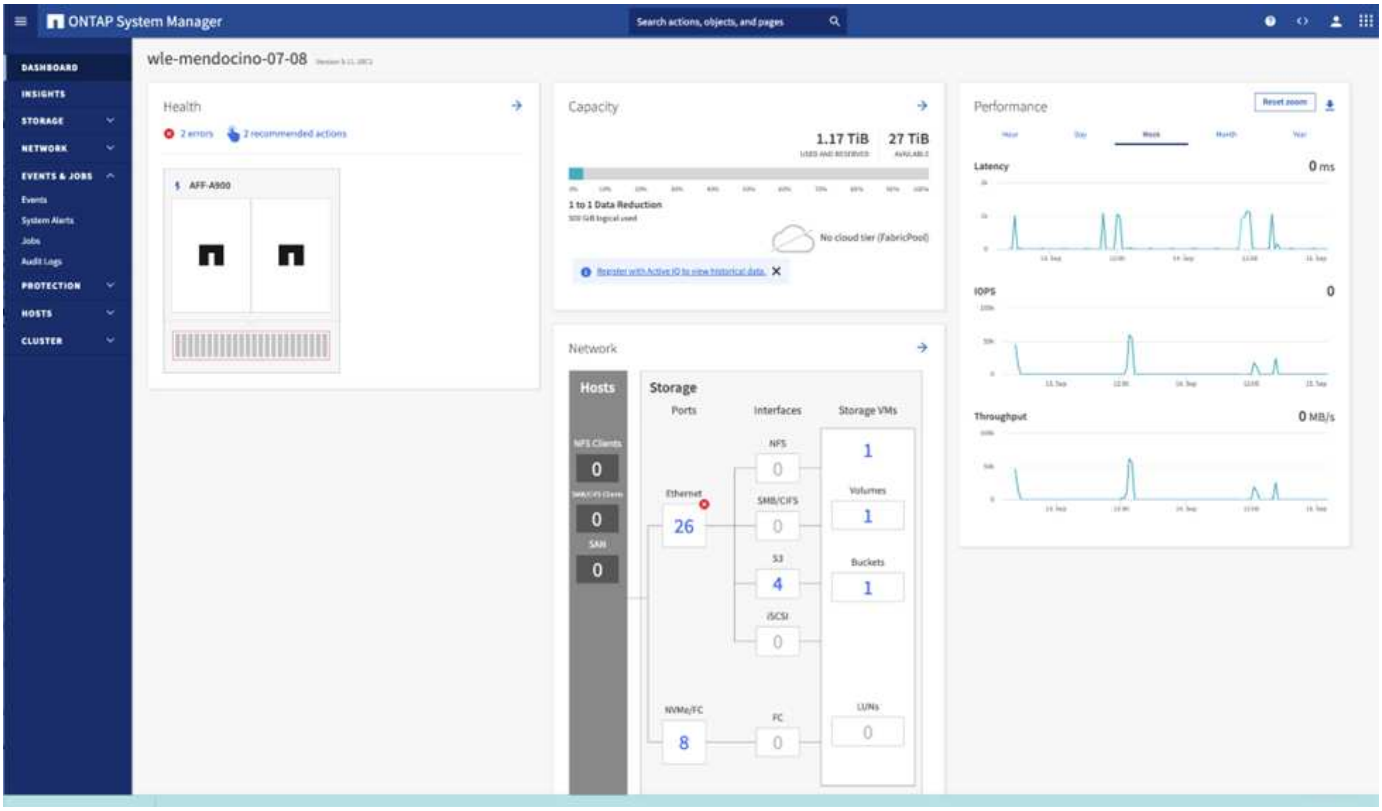
L'utilizzo di NetApp ONTAP e dello storage a più livelli confluente semplifica la gestione dei cluster Apache Kafka sfruttando ONTAP come destinazione di storage scale-out e consente una scalabilità indipendente delle risorse di calcolo e storage per Confluent.

Un server ONTAP S3 si basa sulle funzionalità di storage scale-out avanzate di ONTAP. La scalabilità del cluster ONTAP può essere eseguita senza problemi estendendo i bucket S3 per utilizzare i nuovi nodi aggiunti al cluster ONTAP.

Gestione semplice con Gestore di sistema di ONTAP

Gestore di sistema ONTAP è un'interfaccia grafica basata su browser che consente di configurare, gestire e monitorare il controller di storage ONTAP in ubicazioni distribuite a livello globale in un unico pannello di

controllo.



È possibile configurare e gestire ONTAP S3 con Gestore di sistema e l'interfaccia utente di ONTAP. Quando si attiva S3 e si creano bucket utilizzando Gestione sistema, ONTAP fornisce le Best practice predefinite per una configurazione semplificata. Se si configurano il server S3 e i bucket dalla CLI, è comunque possibile gestirli con System Manager, se lo si desidera, o viceversa.

Quando si crea un bucket S3 utilizzando Gestione di sistema, ONTAP configura un livello di servizio delle performance predefinito il più alto disponibile sul sistema. Ad esempio, in un sistema AFF, l'impostazione predefinita è estrema. I livelli di servizio delle performance sono gruppi di policy QoS adattivi predefiniti. Invece di uno dei livelli di servizio predefiniti, è possibile specificare un gruppo di criteri QoS personalizzato o nessun gruppo di criteri.

I gruppi di criteri QoS adattivi predefiniti includono:

- **Extreme.** utilizzato per applicazioni che richiedono la latenza più bassa e le performance più elevate.
- **Performance.** utilizzato per applicazioni con esigenze di performance e latenza modeste.
- **Valore.** utilizzato per applicazioni per le quali throughput e capacità sono più importanti della latenza.
- **Custom.** specificare un criterio QoS personalizzato o nessun criterio QoS.

Se si seleziona **Use for Tiering** (Usa per il tiering), non viene selezionato alcun livello di servizio delle performance e il sistema tenta di selezionare supporti a basso costo con performance ottimali per i dati a più livelli.

ONTAP tenta di eseguire il provisioning di questo bucket su Tier locali che dispongono dei dischi più appropriati, soddisfacendo il livello di servizio scelto. Tuttavia, se è necessario specificare quali dischi includere nel bucket, è consigliabile configurare lo storage a oggetti S3 dalla CLI specificando i Tier locali (aggregato). Se si configura il server S3 dalla CLI, è comunque possibile gestirlo con System Manager, se necessario.

Se si desidera specificare gli aggregati da utilizzare per i bucket, è possibile farlo solo utilizzando la CLI.

Confluente

Confluent Platform è una piattaforma per lo streaming di dati completa che consente di accedere, memorizzare e gestire facilmente i dati come flussi continui e in tempo reale. Creato dai creatori originali di Apache Kafka, Confluent amplia i vantaggi di Kafka con funzionalità di livello Enterprise, eliminando al contempo il peso della gestione o del monitoraggio di Kafka. Oggi, oltre il 80% dei Fortune 100 è basato su tecnologia di streaming dei dati e la maggior parte utilizza Confluent.

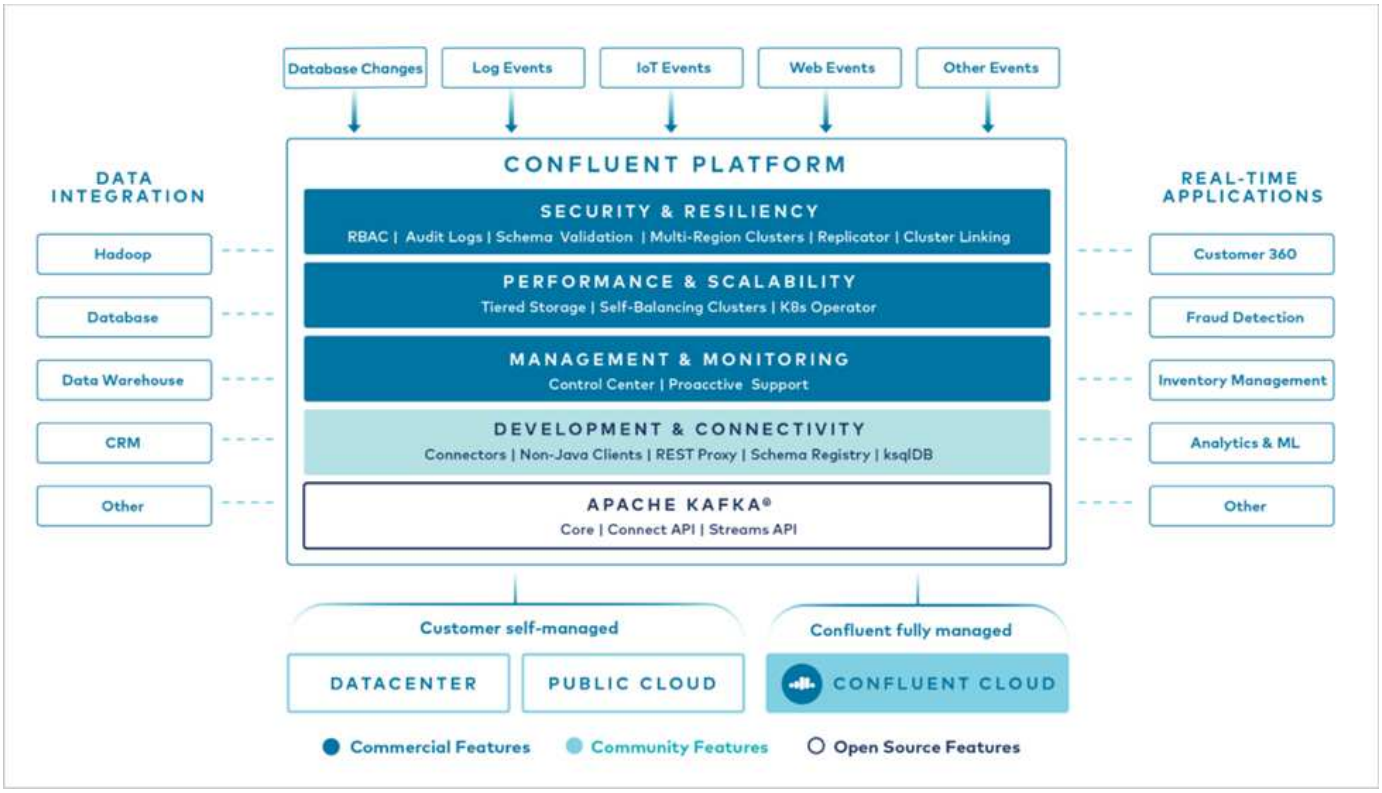
Perché confluente?

Integrando dati storici e in tempo reale in un'unica fonte di verità centrale, Confluent semplifica la creazione di una categoria completamente nuova di applicazioni moderne e basate sugli eventi, l'acquisizione di una pipeline universale di dati e lo sblocco di nuovi casi di utilizzo potenti con scalabilità, performance e affidabilità complete.

A cosa serve Confluent?

Confluent Platform ti consente di concentrarti su come ricavare il valore di business dai tuoi dati piuttosto che preoccuparsi delle meccaniche sottostanti, come ad esempio il modo in cui i dati vengono trasportati o integrati tra sistemi diversi. In particolare, Confluent Platform semplifica la connessione delle origini dati a Kafka, la creazione di applicazioni di streaming e la protezione, il monitoraggio e la gestione dell'infrastruttura Kafka. Attualmente, Confluent Platform viene utilizzata per un'ampia gamma di casi di utilizzo in numerosi settori, dai servizi finanziari, al retail omnichannel e alle auto autonome, al rilevamento delle frodi, ai microservizi e all'IoT.

La figura seguente mostra i componenti della piattaforma confluente.



Panoramica della tecnologia Confluent Event Streaming

Il fulcro della piattaforma confluente è "Kafka", la piattaforma di streaming distribuito open source più diffusa.

Le principali funzionalità di Kafka includono:

- Pubblicare e sottoscrivere flussi di record.
- Memorizzare i flussi di record in modo tollerante agli errori.
- Elaborazione di flussi di record.

Confluent Platform include anche il Registro di sistema dello schema, il proxy REST, oltre 100 connettori Kafka preintegrati e ksqlDB.

Panoramica delle funzionalità aziendali della piattaforma Confluent

- **Confluent Control Center.** sistema basato su interfaccia utente per la gestione e il monitoraggio di Kafka. Consente di gestire facilmente Kafka Connect e creare, modificare e gestire le connessioni ad altri sistemi.
- **Confluent per Kubernetes.** Confluent per Kubernetes è un operatore di Kubernetes. Gli operatori di Kubernetes estendono le funzionalità di orchestrazione di Kubernetes fornendo funzionalità e requisiti unici per una specifica applicazione della piattaforma. Per Confluent Platform, ciò include una notevole semplificazione del processo di implementazione di Kafka su Kubernetes e l'automazione delle attività tipiche del ciclo di vita dell'infrastruttura.
- * Connettori Kafka Connect.* i connettori utilizzano l'API Kafka Connect per connettere Kafka ad altri sistemi come database, archivi di valori chiave, indici di ricerca e file system. Confluent Hub dispone di connettori scaricabili per le fonti di dati e i sink più diffusi, incluse le versioni completamente testate e supportate di questi connettori con Confluent Platform. Ulteriori dettagli sono disponibili "qui".
- **Cluster con bilanciamento automatico.** offre bilanciamento del carico automatico, rilevamento degli errori e riparazione automatica. Fornisce inoltre supporto per l'aggiunta o la disattivazione di broker in base alle necessità, senza tuning manuale.
- **Collegamento di cluster confluente.** collega direttamente i cluster e esegue il mirroring degli argomenti da un cluster all'altro tramite un bridge di collegamento. Il collegamento dei cluster semplifica la configurazione di implementazioni di cloud ibrido, multi-cluster e multi-data center.
- **Confluent auto data balancer.** monitora il cluster per il numero di broker, la dimensione delle partizioni, il numero di partizioni e il numero di leader all'interno del cluster. Consente di spostare i dati per creare un carico di lavoro uniforme nel cluster, riducendo al contempo il ribilanciamento del traffico per ridurre al minimo l'effetto sui carichi di lavoro di produzione durante il ribilanciamento.
- **Confluent Replicator.** semplifica la gestione di più cluster Kafka in più data center.
- **Tiered storage.** offre opzioni per l'archiviazione di grandi volumi di dati Kafka utilizzando il tuo cloud provider preferito, riducendo così il carico operativo e i costi. Con lo storage a più livelli, puoi mantenere i dati su uno storage a oggetti conveniente e scalare i broker solo quando hai bisogno di più risorse di calcolo.
- **Confluent JMS client.** Confluent Platform include un client compatibile con JMS per Kafka. Questo client Kafka implementa l'API standard JMS 1.1, utilizzando i broker Kafka come backend. Questo è utile se si utilizzano applicazioni legacy con JMS e si desidera sostituire il message broker JMS esistente con Kafka.
- **Il proxy MQTT confluente.** offre un modo per pubblicare i dati direttamente su Kafka da dispositivi e gateway MQTT senza la necessità di un broker MQTT al centro.
- **I plug-in di sicurezza confluenti.** i plug-in di sicurezza confluenti vengono utilizzati per aggiungere funzionalità di sicurezza a vari strumenti e prodotti della piattaforma confluente. Attualmente, è disponibile un plug-in per il proxy REST confluente che consente di autenticare le richieste in entrata e propagare l'identità autenticata alle richieste a Kafka. Ciò consente ai client proxy REST confluenti di utilizzare le funzionalità di sicurezza multi-tenant del broker Kafka.

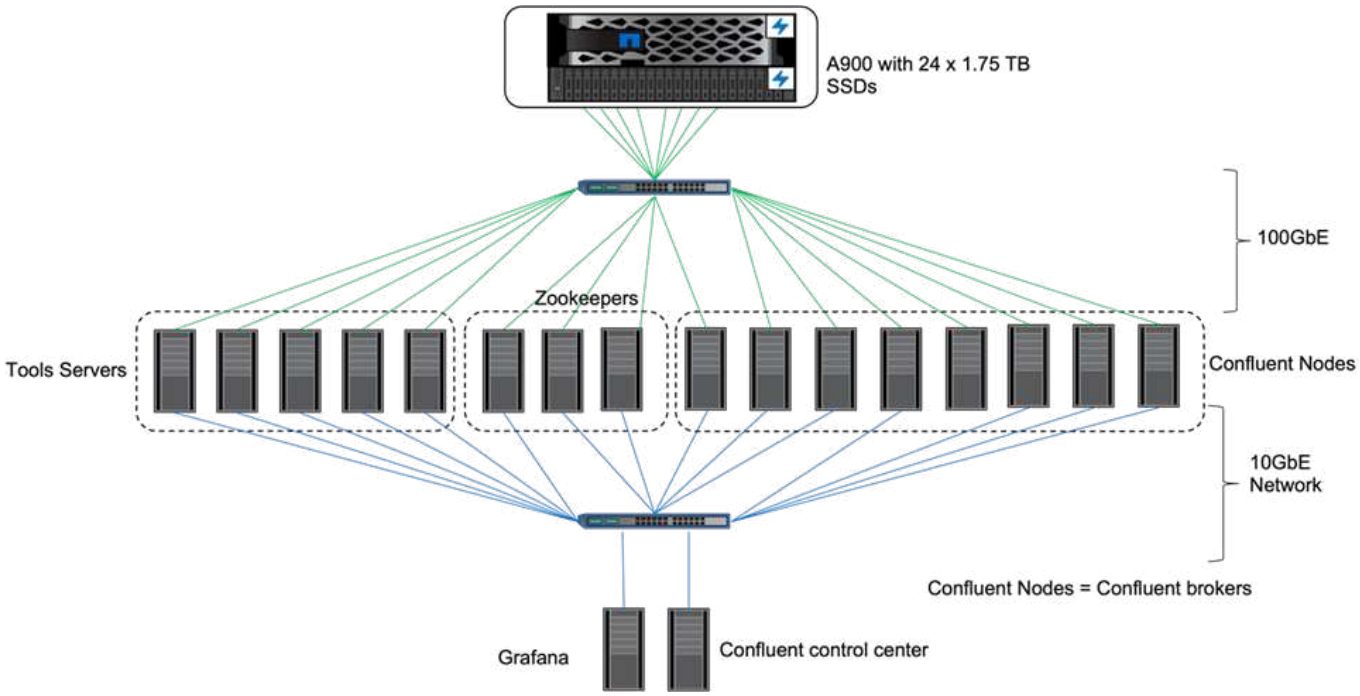
Convalida delle performance confluente

Abbiamo eseguito la verifica con la piattaforma confluente per lo storage su più livelli su NetApp ONTAP. I team NetApp e Confluent hanno lavorato insieme a questa verifica ed hanno eseguito i test case richiesti per l'IT.

Configurazione confluente

Per la configurazione, abbiamo utilizzato tre zookeeper, cinque broker e cinque server di test con 256 GB di RAM e 16 CPU. Per lo storage NetApp, abbiamo utilizzato ONTAP con una coppia AFF A900 ha. Lo storage e i broker sono stati connessi tramite connessioni a 100 GbE.

La figura seguente mostra la topologia di rete della configurazione utilizzata per la verifica dello storage su più livelli.



I server degli strumenti agiscono come client applicativi che inviano o ricevono eventi da o verso i nodi confluenti.

Configurazione dello storage a più livelli confluente

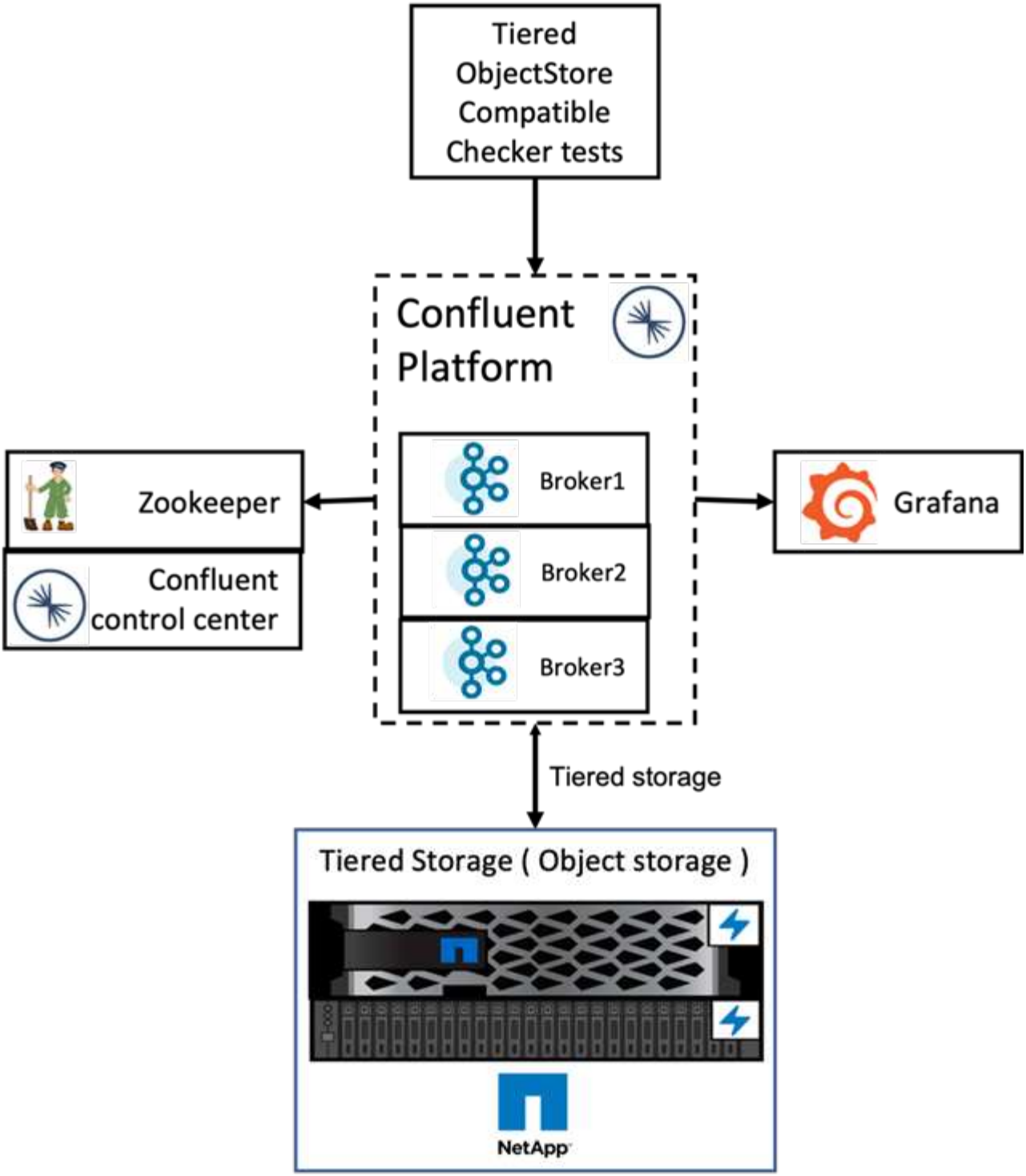
Abbiamo utilizzato i seguenti parametri di test:

```
confluent.tier.fetcher.num.threads=80
confluent.tier.archiver.num.threads=80
confluent.tier.enable=true
confluent.tier.feature=true
confluent.tier.backend=S3
confluent.tier.s3.bucket=kafkabucket1-1
confluent.tier.s3.region=us-east-1
confluent.tier.s3.cred.file.path=/data/kafka/.ssh/credentials
confluent.tier.s3.aws.endpoint.override=http://wle-mendocino-07-08/
confluent.tier.s3.force.path.style.access=true
bootstrap.server=192.168.150.172:9092,192.168.150.120:9092,192.168.150.164
:9092,192.168.150.198:9092,192.168.150.109:9092,192.168.150.165:9092,192.1
68.150.119:9092,192.168.150.133:9092
debug=true
jmx.port=7203
num.partitions=80
num.records=200000000
#object PUT size - 512MB and fetch 100MB - netapp
segment.bytes=536870912
max.partition.fetch.bytes=1048576000
#GET size is max.partition.fetch.bytes/num.partitions
length.key.value=2048
trogdor.agent.nodes=node0,node1,node2,node3,node4
trogdor.coordinator.hostname.port=192.168.150.155:8889
num.producers=20
num.head.consumers=20
num.tail.consumers=1
test.binary.task.max.heap.size=32G
test.binary.task.timeout.sec=3600
producer.timeout.sec=3600
consumer.timeout.sec=3600
```

Per la verifica, abbiamo utilizzato ONTAP con il protocollo HTTP, ma anche HTTPS ha funzionato. La chiave di accesso e la chiave segreta vengono memorizzate nel nome file fornito in `confluent.tier.s3.cred.file.path` parametro.

Storage controller NetApp – ONTAP

Abbiamo configurato una singola configurazione di coppia ha in ONTAP per la verifica.



Risultati della verifica

Abbiamo completato i seguenti cinque casi di test per la verifica. I primi due erano test di funzionalità e i restanti tre erano test di performance.

Test di correttezza dell'archivio di oggetti

Questo test esegue operazioni di base come GET, put ed DELETE nell'archivio di oggetti utilizzato per lo storage a più livelli utilizzando le chiamate API.

Test di correttezza delle funzionalità di tiering

Questo test verifica la funzionalità end-to-end dello storage a oggetti. Crea un argomento, crea un flusso di eventi per l'argomento appena creato, attende che i broker archivino i segmenti nello storage a oggetti, consuma il flusso di eventi e convalida le corrispondenze del flusso consumato con il flusso prodotto. Questo test è stato eseguito con e senza un'iniezione di errori dello store di oggetti. Abbiamo simulato il guasto del nodo arrestando il servizio di gestione dei servizi in uno dei nodi in ONTAP e convalidando che la funzionalità end-to-end funziona con lo storage a oggetti.

Benchmark Tier fetch

Questo test ha validato le prestazioni di lettura dello storage a più livelli e verificato l'intervallo di richieste di lettura di recupero sotto carico pesante dai segmenti generati dal benchmark. In questo benchmark, Confluent ha sviluppato client personalizzati per soddisfare le richieste di recupero del Tier.

Generatore di carichi di lavoro producete-consumate

Questo test genera indirettamente il carico di lavoro di scrittura nell'archivio di oggetti attraverso l'archiviazione dei segmenti. Il carico di lavoro di lettura (segmenti letti) è stato generato dallo storage a oggetti quando i gruppi di consumatori hanno recuperato i segmenti. Questo carico di lavoro è stato generato da uno script TOCC. Questo test ha verificato le prestazioni di lettura e scrittura sullo storage a oggetti in thread paralleli. Abbiamo eseguito test con e senza l'iniezione di errori del negozio di oggetti come abbiamo fatto per il test di correttezza della funzionalità di tiering.

Generatore di workload di conservazione

Questo test ha controllato le prestazioni di eliminazione di uno storage a oggetti con un carico di lavoro di conservazione degli argomenti pesante. Il carico di lavoro di conservazione è stato generato utilizzando uno script TOCC che produce molti messaggi in parallelo a un argomento di test. L'argomento del test è stato configurato con un'impostazione di conservazione aggressiva basata sulle dimensioni e sul tempo che ha causato la rimozione continua del flusso di eventi dall'archivio di oggetti. I segmenti sono stati quindi archiviati. Ciò ha portato a numerose eliminazioni nello storage a oggetti da parte del broker e alla raccolta delle performance delle operazioni di eliminazione degli archivi di oggetti.

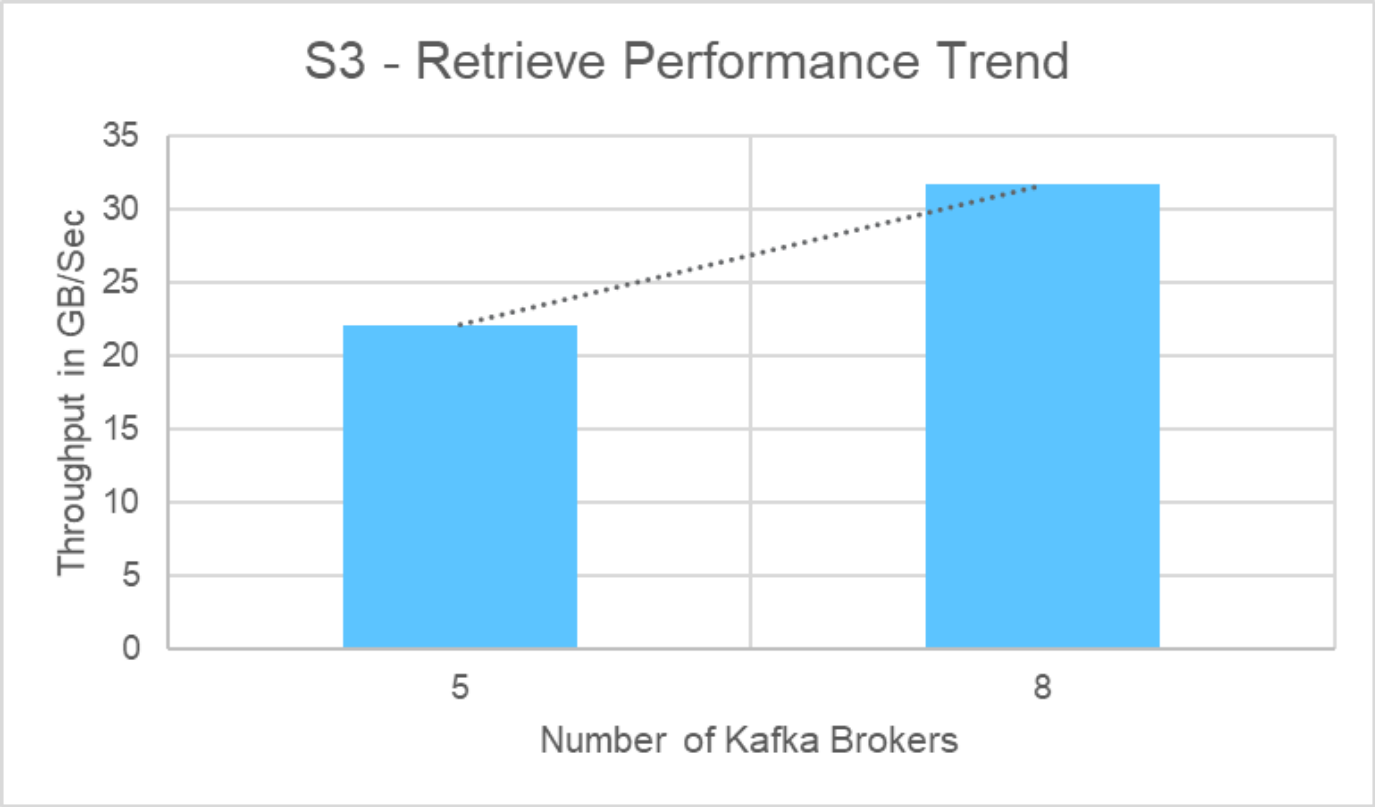
Per informazioni dettagliate sulla verifica, consultare "Confluente" sito web.

Test delle performance con generatore di carichi di lavoro producete-consumate

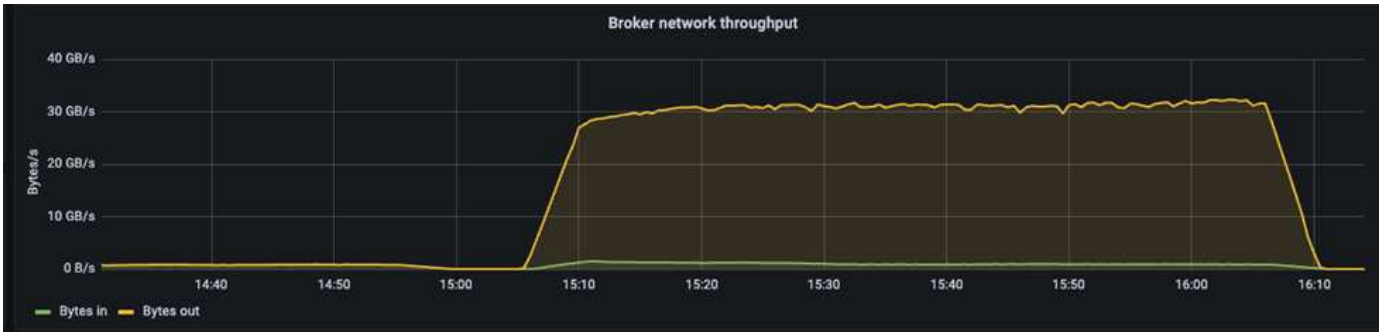
Abbiamo eseguito test dello storage su più livelli con cinque o otto nodi broker durante un carico di lavoro di produzione-consumo con un controller di storage NetApp a coppia AFF A900 ha. Secondo i nostri test, il tempo di completamento e i risultati delle performance sono stati scalati in base al numero di nodi broker fino a quando l'utilizzo delle risorse di AFF A900 non ha raggiunto il 100%. La configurazione del controller di storage ONTAP richiedeva almeno una coppia ha.

Le performance per l'operazione di recupero S3 sono aumentate linearmente in base al numero di nodi di broker confluenti. Lo storage controller ONTAP supporta fino a 12 coppie ha in una singola implementazione.

Il seguente grafico mostra il traffico di tiering S3 combinato con cinque o otto nodi broker. Abbiamo massimizzato le prestazioni della coppia ha singola AFF A900.



Il seguente grafico mostra il throughput di Kafka a circa 31,74 GBps.



Abbiamo anche osservato un throughput simile nel controller di storage ONTAP `perfstat` report.

```
object_store_server:wle-mendocino-07-08:get_data:34080805907b/ s
object_store_server:wle-mendocino-07-08:put_data:484236974b/ s
```

Linee guida sulle Best practice per le performance

Questa pagina descrive le Best practice per migliorare le performance di questa soluzione.

- Per ONTAP, quando possibile, utilizzare una dimensione GET >=1 MB.
- In aumento `num.network.threads` e. `num.io.threads` poll `server.properties` On broker Node consente di trasferire l'attività di tiering aumentata al Tier S3. Questi risultati sono con `num.network.threads` e. `num.io.threads` impostare su 32.

- I bucket S3 devono essere mirati a otto componenti per aggregato membro.
- I collegamenti Ethernet che guidano il traffico S3 devono utilizzare un MTU di 9k, quando possibile, sia sullo storage che sul client.

Conclusione

Questo test di verifica ha raggiunto 31,74 Gbps di throughput di tiering su Confluent con il controller di storage NetApp ONTAP.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Cos'è Confluent?

  "https://www.confluent.io/apache-kafka-vs-confluent/"

- Dettagli del parametro S3-sink

  "https://docs.confluent.io/kafka-connect-s3-sink/current/configuration_options.html#s3-configuration-options"

- Apache Kafka

  "https://en.wikipedia.org/wiki/Apache_Kafka"

- S3 in ONTAP Best practice

  https://www.netapp.com/pdf.html?item=/media/17219-tr4814.pdf

- Gestione dello storage a oggetti S3

  "https://docs.netapp.com/us-en/ontap/s3-config/s3-support-concept.html"

- Documentazione sui prodotti NetApp

  "https://www.netapp.com/support-and-training/documentation/"

Soluzioni storage NetApp per Apache Spark

TR-4570: Soluzioni storage NetApp per Apache Spark: Architettura, casi di utilizzo e risultati delle performance

Rick Huang, Karthikeyan Nagalingam, NetApp

Questo documento si concentra sull'architettura di Apache Spark, sui casi di utilizzo dei clienti e sul portfolio di storage NetApp relativi all'analisi dei big data e all'intelligenza artificiale (ai). Presenta inoltre diversi risultati di test utilizzando strumenti di ai standard di settore, machine learning (ML) e deep learning (DL) rispetto a un sistema Hadoop tipico, in modo da poter scegliere la soluzione Spark appropriata. Per iniziare, è necessaria

un’architettura Spark, componenti appropriati e due modalità di implementazione (cluster e client).

Questo documento fornisce anche casi di utilizzo per i clienti per risolvere i problemi di configurazione e illustra una panoramica del portfolio di storage NetApp relativo all’analisi dei big data e ai, ML e DL con Spark. Concludiamo con i risultati dei test derivati dai casi di utilizzo specifici di Spark e dal portfolio di soluzioni NetApp Spark.

Sfide per i clienti

Questa sezione si concentra sulle sfide dei clienti con analisi dei big data e ai/ML/DL nei settori di crescita dei dati come retail, digital marketing, banking, produzione discreta, produzione di processi, enti pubblici e servizi professionali.

Performance imprevedibili

Le implementazioni Hadoop tradizionali utilizzano generalmente hardware commodity. Per migliorare le performance, devi mettere a punto la rete, il sistema operativo, il cluster Hadoop, i componenti dell’ecosistema come Spark e l’hardware. Anche se si sintonizzano ciascun livello, può essere difficile raggiungere i livelli di performance desiderati perché Hadoop viene eseguito su hardware commodity non progettato per le performance elevate nel proprio ambiente.

Guasti dei supporti e dei nodi

Anche in condizioni normali, l’hardware commodity è soggetto a guasti. Se un disco su un nodo dati si guasta, il master Hadoop considera il nodo non integro per impostazione predefinita. Quindi, copia dati specifici da quel nodo in rete dalle repliche a un nodo integro. Questo processo rallenta i pacchetti di rete per qualsiasi job Hadoop. Il cluster deve quindi copiare di nuovo i dati e rimuovere i dati replicati in eccesso quando il nodo non integro torna allo stato integro.

Lock-in del vendor Hadoop

I distributori Hadoop dispongono di una propria distribuzione Hadoop con il proprio controllo delle versioni, che blocca il cliente a tali distribuzioni. Tuttavia, molti clienti richiedono supporto per l’analisi in-memory che non colleghi il cliente a specifiche distribuzioni Hadoop. Hanno bisogno della libertà di cambiare le distribuzioni e di portare con sé le loro analisi.

Mancanza di supporto per più di una lingua

I clienti spesso richiedono il supporto di più lingue oltre ai programmi Java MapReduce per eseguire i propri lavori. Opzioni come SQL e script offrono maggiore flessibilità per ottenere risposte, più opzioni per l’organizzazione e il recupero dei dati e metodi più rapidi per spostare i dati in un framework di analisi.

Difficoltà di utilizzo

Per qualche tempo, si lamenta che Hadoop è difficile da utilizzare. Anche se Hadoop è diventato più semplice e potente con ogni nuova versione, questa critica ha persistito. Hadoop richiede di comprendere i modelli di programmazione Java e MapReduce, una sfida per gli amministratori di database e le persone con competenze di scripting tradizionali.

Framework e strumenti complessi

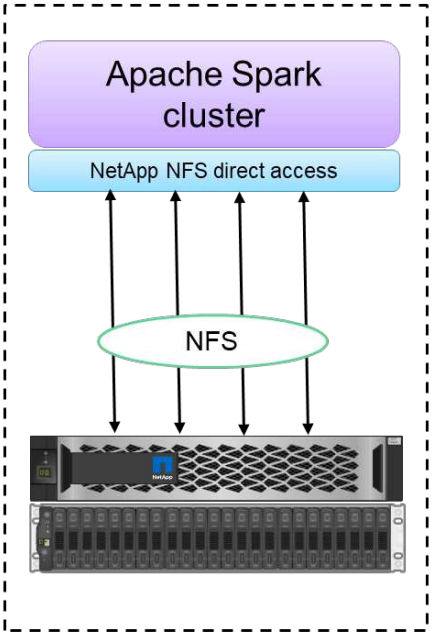
I team ai delle aziende devono affrontare diverse sfide. Anche con una conoscenza esperta di data science, gli strumenti e i framework per diversi ecosistemi di implementazione e applicazioni potrebbero non tradursi semplicemente da uno all’altro. Una piattaforma per la scienza dei dati dovrebbe integrarsi perfettamente con

le corrispondenti piattaforme per i big data basate su Spark con facilità di spostamento dei dati, modelli riutilizzabili, codice pronto all'uso e strumenti che supportino le Best practice per la prototipazione, la convalida, il controllo delle versioni, la condivisione, il riutilizzo, e implementazione rapida dei modelli in produzione.

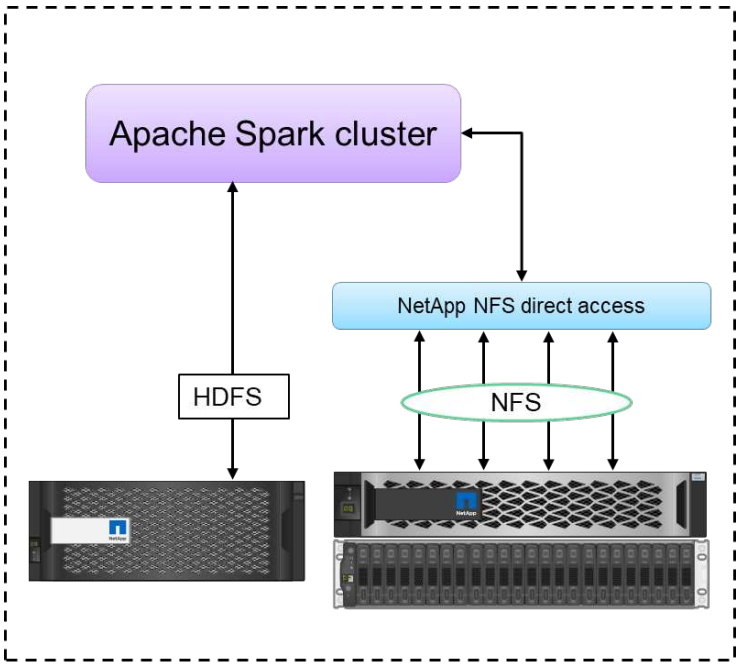
Perché scegliere NetApp?

NetApp può migliorare la tua esperienza con Spark nei seguenti modi:

- L'accesso diretto NetApp NFS (mostrato nella figura seguente) consente ai clienti di eseguire lavori di big data analytics sui dati NFSv3 o NFSv4 esistenti o nuovi senza spostare o copiare i dati. Impedisce più copie di dati ed elimina la necessità di sincronizzare i dati con un'origine.
- Storage più efficiente e minore replica dei server. Ad esempio, la soluzione NetApp e-Series Hadoop richiede due repliche anziché tre dei dati, mentre la soluzione FAS Hadoop richiede un'origine dati ma non una replica o copie dei dati. Le soluzioni di storage NetApp producono inoltre meno traffico server-server.
- Migliore comportamento del cluster e del job Hadoop durante il guasto di disco e nodo.
- Migliori performance di acquisizione dei dati.



Configuration 1: NFS as primary storage



Configuration 2: HDFS and NFS in single Spark cluster

Ad esempio, nel settore finanziario e sanitario, lo spostamento dei dati da un luogo all'altro deve soddisfare gli obblighi legali, il che non è un compito facile. In questo scenario, l'accesso diretto NetApp NFS analizza i dati finanziari e sanitari dalla posizione originale. Un altro vantaggio chiave è che l'utilizzo dell'accesso diretto NetApp NFS semplifica la protezione dei dati Hadoop utilizzando i comandi Hadoop nativi e abilitando i flussi di lavoro per la protezione dei dati con il ricco portfolio di gestione dei dati di NetApp.

L'accesso diretto NetApp NFS offre due tipi di opzioni di implementazione per i cluster Hadoop/Spark:

- Per impostazione predefinita, i cluster Hadoop o Spark utilizzano HDFS (Distributed file System) di Hadoop per lo storage dei dati e il file system predefinito. L'accesso diretto NetApp NFS può sostituire l'HDFS predefinito con lo storage NFS come file system predefinito, consentendo l'analisi diretta dei dati NFS.
- In un'altra opzione di implementazione, l'accesso diretto NetApp NFS supporta la configurazione di NFS come storage aggiuntivo insieme a HDFS in un singolo cluster Hadoop o Spark. In questo caso, il cliente

può condividere i dati attraverso le esportazioni NFS e accedervi dallo stesso cluster insieme ai dati HDFS.

I vantaggi principali dell'utilizzo dell'accesso diretto NetApp NFS includono:

- Analisi dei dati dalla posizione corrente, che impedisce il dispendioso in termini di tempo e performance dello spostamento dei dati di analisi in un'infrastruttura Hadoop come HDFS.
- Riduzione del numero di repliche da tre a uno.
- Consentendo agli utenti di separare calcolo e storage per scalare in modo indipendente.
- Protezione dei dati aziendali sfruttando le ricche funzionalità di gestione dei dati di ONTAP.
- Certificazione con la piattaforma dati Hortonworks.
- Implementazione di data analytics ibridi.
- Riduzione dei tempi di backup sfruttando la funzionalità multithread dinamica.

Vedere "TR-4657: Soluzioni dati di cloud ibrido NetApp - Spark e Hadoop in base ai casi di utilizzo dei clienti" Per il backup dei dati Hadoop, il backup e il disaster recovery dal cloud al on-premise, l'abilitazione di DevTest sui dati Hadoop esistenti, la protezione dei dati e la connettività multicloud e l'accelerazione dei carichi di lavoro di analytics.

Le sezioni seguenti descrivono le funzionalità di storage importanti per i clienti Spark.

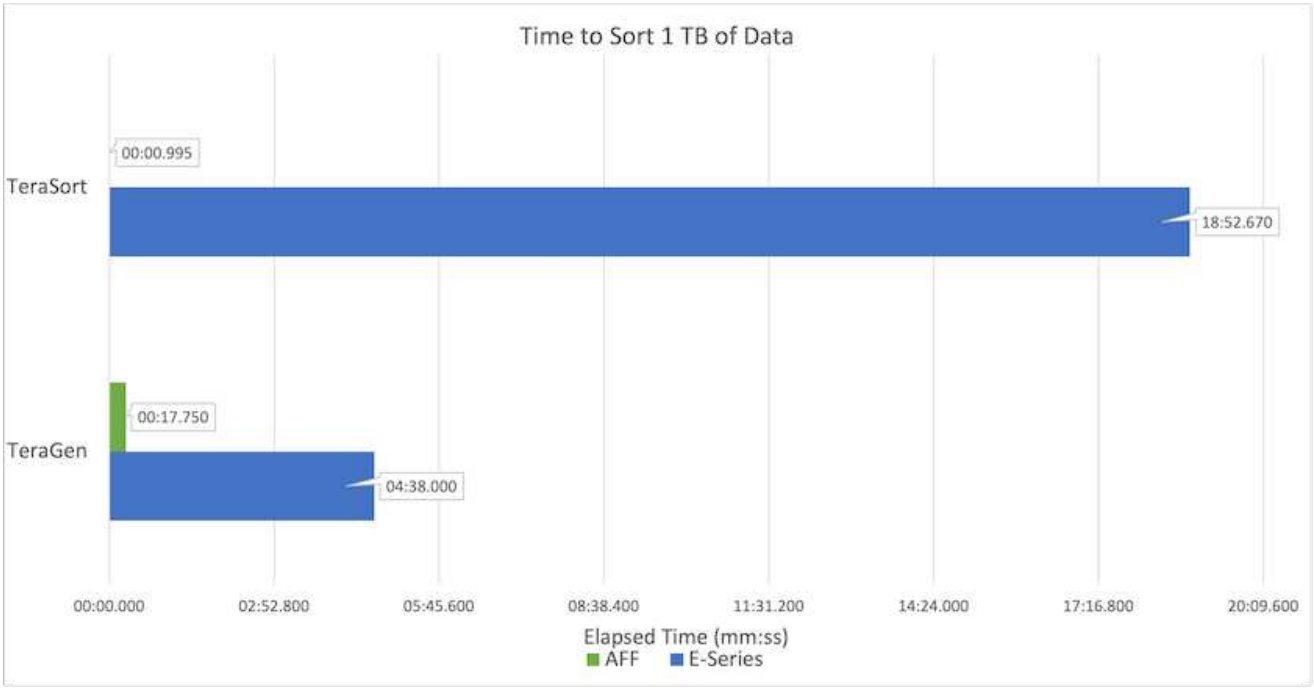
Tiering dello storage

Con il tiering dello storage Hadoop, è possibile memorizzare file con diversi tipi di storage in conformità con una policy di storage. I tipi di storage includono `hot`, `cold`, `warm`, `all_ssd`, `one_ssd`, e. `lazy_persist`.

<<<<<<< HEAD abbiamo eseguito la convalida del tiering dello storage Hadoop su un controller di storage NetApp AFF e su un controller di storage e-Series con unità SSD e SAS con diverse policy di storage. Il cluster Spark con AFF-A800 ha quattro nodi di lavoro di calcolo, mentre il cluster con e-Series ne ha otto. Questo serve principalmente a confrontare le prestazioni dei dischi a stato solido (SSD) rispetto ai dischi rigidi (HDD).

Abbiamo eseguito la convalida del tiering dello storage Hadoop su un controller di storage NetApp AFF e su un controller di storage e-Series con unità SSD e SAS con policy di storage diverse. Il cluster Spark con AFF-A800 ha quattro nodi di lavoro di calcolo, mentre il cluster con e-Series ne ha otto. Abbiamo fatto questo principalmente per confrontare le performance dei dischi a stato solido con quelle dei dischi rigidi. >>>>>> a51c9dddf73ca69e1120ce05edc7b0b9607b96eae

La figura seguente mostra le performance delle soluzioni NetApp per un SSD Hadoop.



- La configurazione baseline NL-SAS utilizzava otto nodi di calcolo e 96 dischi NL-SAS. Questa configurazione ha generato 1 TB di dati in 4 minuti e 38 secondi. Vedere "TR-3969 soluzione NetApp e-Series per Hadoop" per informazioni dettagliate sulla configurazione del cluster e dello storage.
- Utilizzando TeraGen, la configurazione SSD ha generato 1 TB di dati a una velocità di 15,66 volte superiore rispetto alla configurazione NL-SAS. Inoltre, la configurazione SSD utilizzava la metà del numero di nodi di calcolo e la metà del numero di dischi (24 unità SSD in totale). In base al tempo di completamento del lavoro, la velocità era quasi doppia rispetto alla configurazione NL-SAS.
- Utilizzando TeraSort, la configurazione SSD ha ordinato 1 TB di dati 1138.36 volte più rapidamente della configurazione NL-SAS. Inoltre, la configurazione SSD utilizzava la metà del numero di nodi di calcolo e la metà del numero di dischi (24 unità SSD in totale). Pertanto, per disco, la velocità era circa tre volte superiore rispetto alla configurazione NL-SAS. <<<<<<< TESTA
- La transizione da dischi rotanti a all-flash migliora le performance. Il numero di nodi di calcolo non era il collo di bottiglia. Con lo storage all-flash di NetApp, le performance di runtime sono perfettamente scalabili.
- Con NFS, i dati erano funzionalmente equivalenti a quelli del pool, il che può ridurre il numero di nodi di calcolo in base al carico di lavoro. Gli utenti del cluster Apache Spark non devono ribilanciare manualmente i dati quando cambiano il numero di nodi di calcolo.

- In sintesi, la transizione dai dischi rotanti a all-flash migliora le performance. Il numero di nodi di calcolo non era il collo di bottiglia. Con lo storage all-flash NetApp, le performance di runtime sono perfettamente scalabili.

- Con NFS, i dati erano funzionalmente equivalenti a quelli del pool, il che può ridurre il numero di nodi di calcolo in base al carico di lavoro. Gli utenti del cluster Apache Spark non devono ribilanciare manualmente i dati quando cambiano il numero di nodi di calcolo. >>>>>> a51c9dddf73ca69e1120ce05edc7b0b9607b96eae

Scalabilità delle performance - scalabilità orizzontale

Quando è necessaria una maggiore potenza di calcolo da un cluster Hadoop in una soluzione AFF, è possibile aggiungere nodi dati con un numero appropriato di controller storage. NetApp consiglia di iniziare con quattro nodi di dati per array di controller storage e di aumentare il numero fino a otto nodi di dati per controller storage, a seconda delle caratteristiche del carico di lavoro.

AFF e FAS sono perfetti per l'analisi in-place. In base ai requisiti di calcolo, è possibile aggiungere gestori di nodi, mentre le operazioni senza interruzioni consentono di aggiungere un controller di storage on-demand senza downtime. Offriamo funzionalità complete con AFF e FAS, come SUPPORTO multimediale NVME, efficienza garantita, riduzione dei dati, QOS, analisi predittiva, tiering del cloud, replica, implementazione del cloud e sicurezza. Per aiutare i clienti a soddisfare i propri requisiti, NetApp offre funzionalità come analisi del file system, quote e bilanciamento del carico on-box senza costi di licenza aggiuntivi. NetApp offre performance migliori in termini di numero di processi simultanei, latenza inferiore, operazioni più semplici e throughput di gigabyte al secondo superiore rispetto alla concorrenza. Inoltre, NetApp Cloud Volumes ONTAP viene eseguito su tutti e tre i principali cloud provider.

Scalabilità delle performance - scalabilità verticale

Le funzionalità di scale-up consentono di aggiungere dischi ai sistemi AFF, FAS ed e-Series quando è necessaria una capacità di storage aggiuntiva. Con Cloud Volumes ONTAP, la scalabilità dello storage a livello di PB è una combinazione di due fattori: Il tiering dei dati utilizzati di rado per lo storage a oggetti dallo storage a blocchi e lo stacking delle licenze Cloud Volumes ONTAP senza elaborazione aggiuntiva.

Protocolli multipli

I sistemi NetApp supportano la maggior parte dei protocolli per le implementazioni Hadoop, tra cui SAS, iSCSI, FCP, InfiniBand, E NFS.

Soluzioni operative e supportate

Le soluzioni Hadoop descritte in questo documento sono supportate da NetApp. Queste soluzioni sono certificate anche con i principali distributori Hadoop. Per ulteriori informazioni, consultare "MapR" sito, il "Hortonworks" E il Cloudera "certificazione" e. "partner" siti.

Pubblico di riferimento

Il mondo dell'analytics e della scienza dei dati tocca diverse discipline nell'IT e nel business:

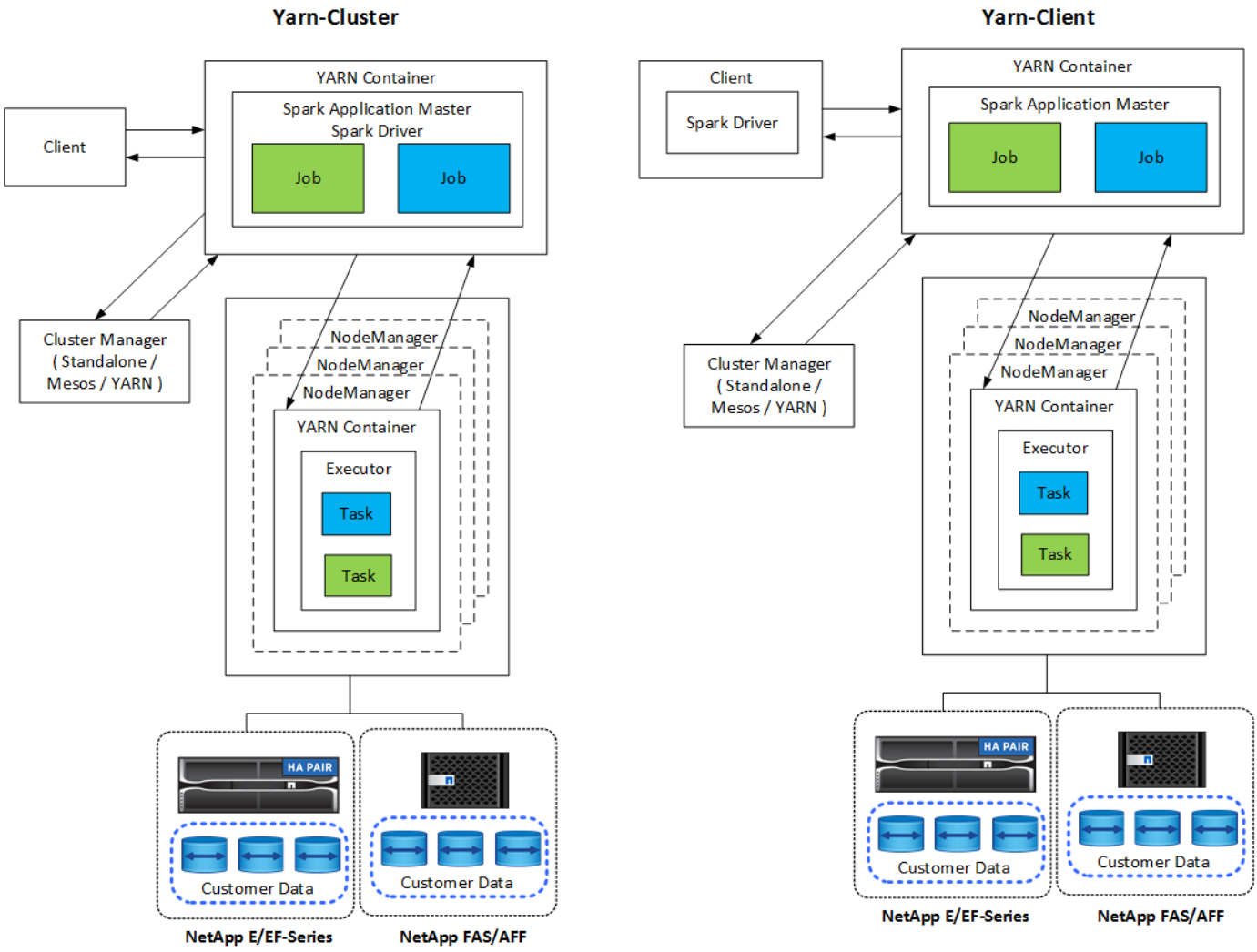
- Il data scientist ha bisogno della flessibilità necessaria per utilizzare i propri strumenti e le librerie preferite.
- Il data engineer deve sapere come i dati scorrono e dove risiedono.
- Un tecnico DevOps ha bisogno dei tool per integrare le nuove applicazioni ai e ML nelle pipeline ci e CD.
- Gli amministratori e gli architetti del cloud devono essere in grado di configurare e gestire le risorse del cloud ibrido.
- Gli utenti aziendali desiderano avere accesso alle applicazioni di analisi, ai, ML e DL.

In questo report tecnico, descriviamo come NetApp AFF, e-Series, StorageGRID, NFS direct access, Apache Spark, Horovod e keras aiutano ciascuno di questi ruoli a portare valore al business.

Tecnologia della soluzione

Apache Spark è un popolare framework di programmazione per la scrittura di applicazioni Hadoop che funziona direttamente con Hadoop Distributed file System (HDFS). Spark è pronto per la produzione, supporta l'elaborazione dei dati in streaming ed è più veloce di MapReduce. Spark dispone di un caching dei dati in-memory configurabile per un'iterazione efficiente e la shell Spark è interattiva per l'apprendimento e l'esplorazione dei dati. Con Spark, puoi creare applicazioni in Python, Scala o Java. Le applicazioni SPARK sono costituite da uno o più lavori che hanno uno o più compiti.

Ogni applicazione Spark dispone di un driver Spark. In modalità YARN-Client, il driver viene eseguito sul client localmente. In modalità YARN-Cluster, il driver viene eseguito nel cluster sul master dell'applicazione. In modalità cluster, l'applicazione continua a funzionare anche se il client si disconnette.



Esistono tre cluster manager:

- **Standalone.** questo gestore fa parte di Spark, che semplifica la configurazione di un cluster.

- **Apache Mesos.** questo è un gestore di cluster generale che esegue anche MapReduce e altre applicazioni.
- **Hadoop YARN.** questo è un resource manager in Hadoop 3.

Il dataset distribuito resiliente (RDD) è il componente principale di Spark. RDD ricrea i dati persi e mancanti dai dati memorizzati nel cluster e memorizza i dati iniziali provenienti da un file o creati a livello di programmazione. Le RDD vengono create da file, dati in memoria o da un altro RDD. La programmazione SPARK esegue due operazioni: Trasformazione e azioni. La trasformazione crea un nuovo RDD basato su uno esistente. Le azioni restituiscono un valore da un RDD.

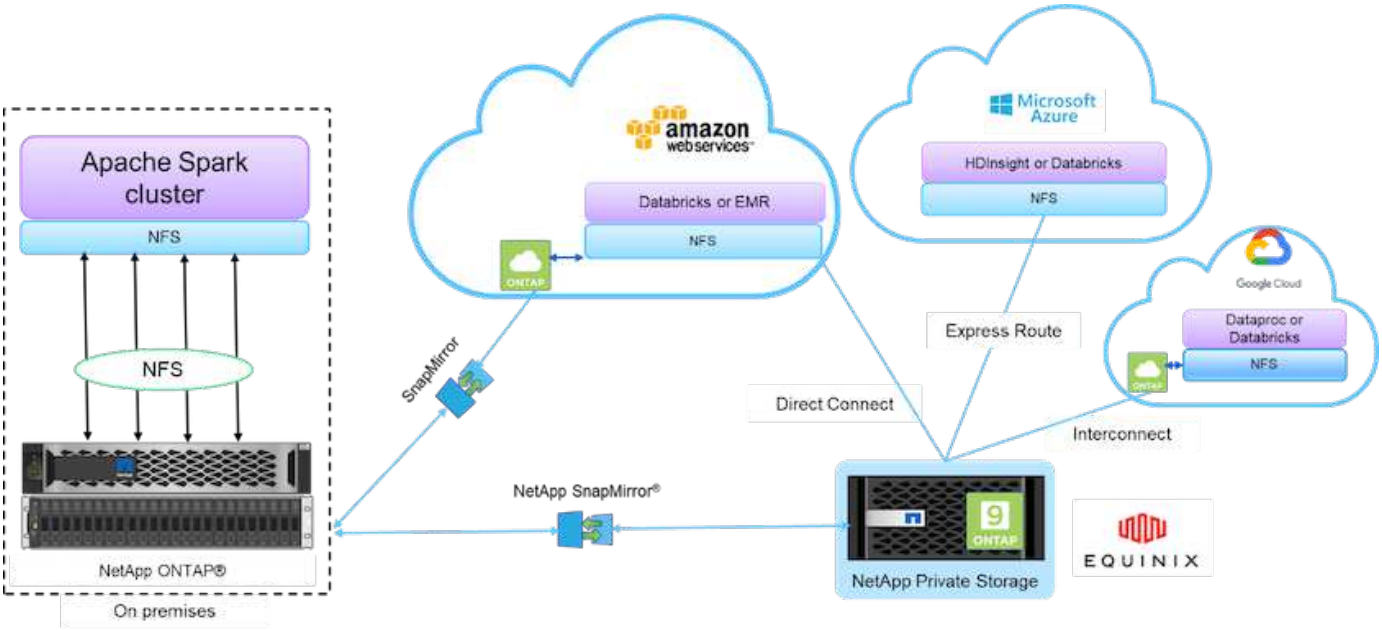
Le trasformazioni e le azioni si applicano anche ai DataSet e ai DataFrame di Spark. Un set di dati è una raccolta distribuita di dati che offre i benefici delle RDDs (tipizzazione forte, utilizzo delle funzioni lambda) con i benefici del motore di esecuzione ottimizzato di Spark SQL. È possibile costruire un dataset da oggetti JVM e manipolarlo utilizzando trasformazioni funzionali (mappa, mappa piatta, filtro e così via). Un DataFrame è un dataset organizzato in colonne denominate. È concettualmente equivalente a una tabella in un database relazionale o a un frame di dati in R/Python. I DataFrame possono essere costruiti da un'ampia gamma di origini come file di dati strutturati, tabelle in Hive/HBase, database esterni on-premise o nel cloud o RDDs esistenti.

Le applicazioni SPARK includono uno o più lavori Spark. I job eseguono task negli esecutori e gli esecutori vengono eseguiti in CONTAINER DI FILATI. Ogni esecutore viene eseguito in un singolo container e gli esecutori esistono per tutta la vita di un'applicazione. Un esecutore viene fissato dopo l'avvio dell'applicazione e IL FILATO non ridimensiona il container già allocato. Un esecutore può eseguire task contemporaneamente sui dati in-memory.

Panoramica delle soluzioni NetApp Spark

NetApp dispone di tre portfolio di storage: FAS/AFF, e-Series e Cloud Volumes ONTAP. Abbiamo validato AFF e e-Series con il sistema di storage ONTAP per le soluzioni Hadoop con Apache Spark.

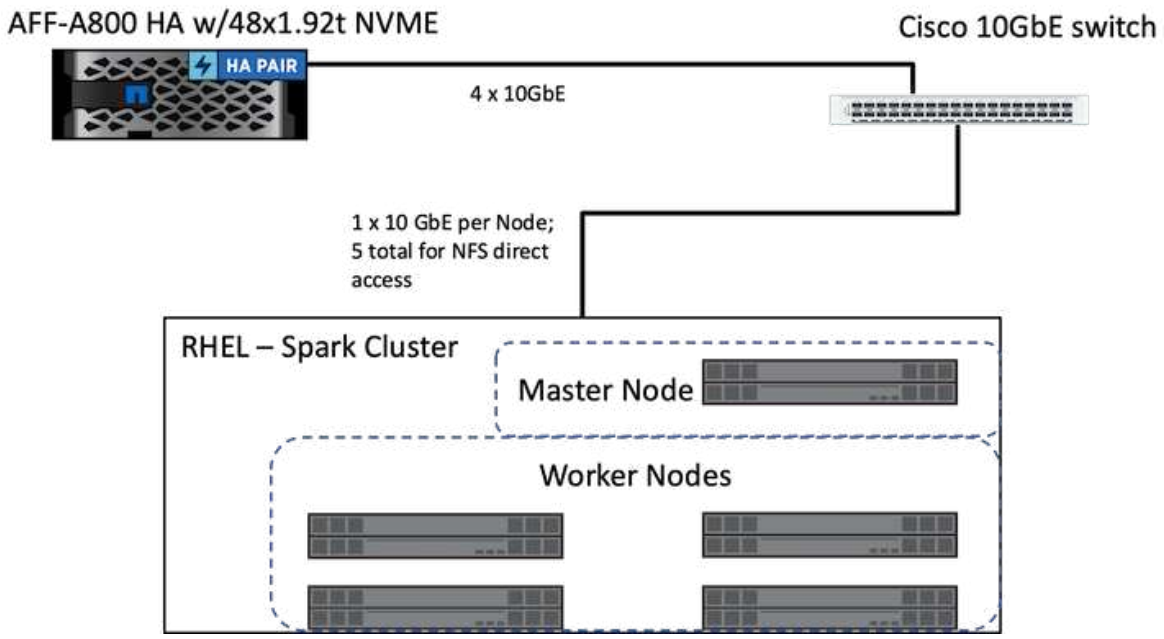
Il data fabric basato su NetApp integra i servizi e le applicazioni di gestione dei dati (building block) per l'accesso, il controllo, la protezione e la sicurezza dei dati, come mostrato nella figura seguente.



Gli elementi di base della figura precedente includono:

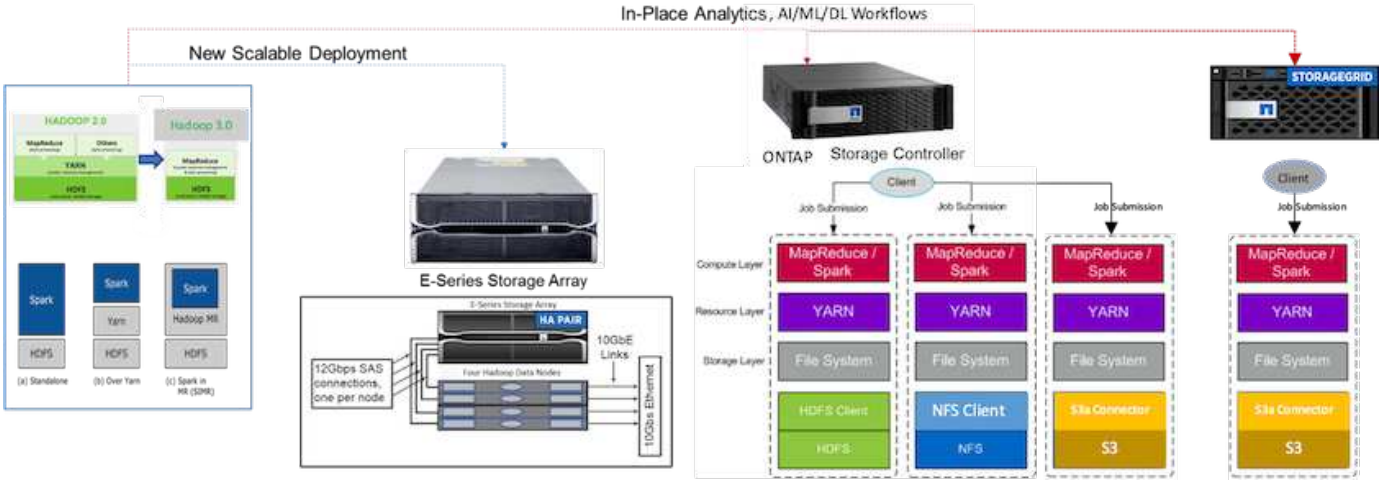
- **Accesso diretto NetApp NFS.** offre i più recenti cluster Hadoop e Spark con accesso diretto ai volumi NetApp NFS senza requisiti aggiuntivi di software o driver.
- **NetApp Cloud Volumes ONTAP e servizi di volume cloud.** storage connesso definito tramite software basato su ONTAP eseguito in AWS (Amazon Web Services) o Azure NetApp Files (ANF) nei servizi cloud Microsoft Azure.
- **La tecnologia NetApp SnapMirror.** offre funzionalità di protezione dei dati tra istanze cloud o NPS ONTAP on-premise e on-premise.
- **Cloud service provider.** questi provider includono AWS, Microsoft Azure, Google Cloud e IBM Cloud.
- **PaaS.** servizi di analisi basati sul cloud come Amazon Elastic MapReduce (EMR) e Databricks in AWS, Microsoft Azure HDInsight e Azure Databricks.

La seguente figura illustra la soluzione Spark con lo storage NetApp.

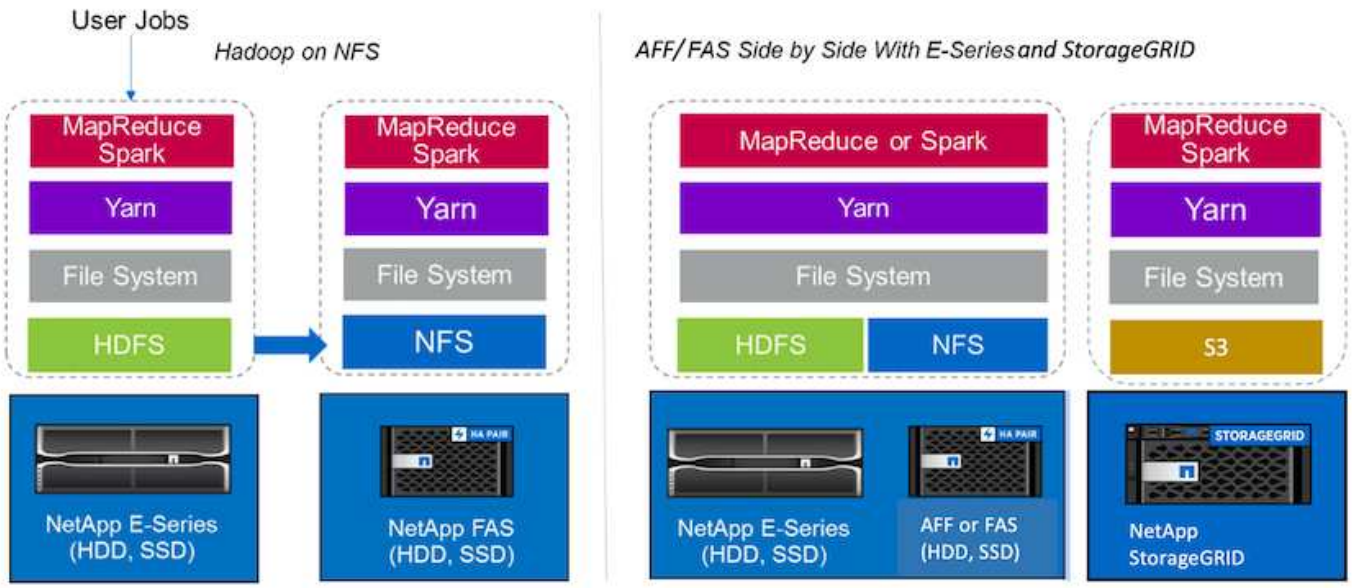


La soluzione Spark di ONTAP utilizza il protocollo di accesso diretto NetApp NFS per l'analisi in-place e i flussi di lavoro ai, ML e DL utilizzando l'accesso ai dati di produzione esistenti. I dati di produzione disponibili per i nodi Hadoop vengono esportati per eseguire lavori analitici in-place e ai, ML e DL. È possibile accedere ai dati da elaborare nei nodi Hadoop con l'accesso diretto NetApp NFS o senza di essi. In Spark con la versione standalone o. `yarn` Cluster manager, è possibile configurare un volume NFS utilizzando `file:///<target_volume`. Abbiamo validato tre casi di utilizzo con set di dati diversi. I dettagli di queste validazioni sono presentati nella sezione "risultati del test". (xref)

La seguente figura illustra il posizionamento dello storage NetApp Apache Spark/Hadoop.



Abbiamo identificato le caratteristiche esclusive della soluzione Spark e-Series, della soluzione Spark AFF/FAS ONTAP e della soluzione Spark StorageGRID, e abbiamo eseguito test e validazione dettagliati. In base alle nostre osservazioni, NetApp consiglia la soluzione e-Series per le installazioni in ambiente e le nuove implementazioni scalabili e la soluzione AFF/FAS per l'analisi in-place, i carichi di lavoro ai, ML e DL utilizzando i dati NFS esistenti e StorageGRID per ai, ML e DL e le analisi dei dati moderne quando è richiesto lo storage a oggetti.



Un data Lake è un repository di storage per set di dati di grandi dimensioni in formato nativo che può essere utilizzato per attività di analisi, ai, ML e DL. Abbiamo creato un repository di data Lake per le soluzioni e-Series, AFF/FAS e StorageGRID SG6060 Spark. Il sistema e-Series fornisce l'accesso HDFS al cluster Hadoop Spark, mentre i dati di produzione esistenti sono accessibili attraverso il protocollo di accesso diretto NFS al cluster Hadoop. Per i set di dati che risiedono nello storage a oggetti, NetApp StorageGRID offre accesso sicuro S3 e S3a.

Riepilogo del caso d'utilizzo

Questa pagina descrive le diverse aree in cui è possibile utilizzare questa soluzione.

Dati in streaming

Apache Spark è in grado di elaborare i dati in streaming, utilizzati per processi di estrazione, trasformazione e carico (ETL) in streaming, per l'arricchimento dei dati, per l'attivazione del rilevamento degli eventi e per analisi complesse delle sessioni:

- **Streaming ETL.** i dati vengono continuamente ripuliti e aggregati prima di essere inseriti negli archivi dati. Netflix utilizza lo streaming di Kafka e Spark per creare una soluzione di monitoraggio dei dati e consigli sui film online in tempo reale in grado di elaborare miliardi di eventi al giorno da diverse origini dati. Tuttavia, l'ETL tradizionale per l'elaborazione in batch viene trattato in modo diverso. Questi dati vengono letti per primi, quindi convertiti in un formato di database prima di essere scritti nel database.
- **Arricchimento dei dati.** lo streaming Spark arricchisce i dati live con dati statici per consentire un'analisi dei dati più in tempo reale. Ad esempio, gli inserzionisti online possono fornire annunci personalizzati e mirati, diretti in base alle informazioni sul comportamento dei clienti.
- **Rilevamento eventi trigger.** lo streaming Spark consente di rilevare e rispondere rapidamente a comportamenti anomali che potrebbero indicare problemi potenzialmente gravi. Ad esempio, gli istituti finanziari utilizzano i trigger per rilevare e arrestare le transazioni di frode, mentre gli ospedali utilizzano i trigger per rilevare i cambiamenti sanitari pericolosi rilevati nei segni vitali di un paziente.
- **Analisi complessa della sessione.** lo streaming di Spark raccoglie eventi come l'attività dell'utente dopo l'accesso a un sito Web o a un'applicazione, che vengono quindi raggruppati e analizzati. Ad esempio, Netflix utilizza questa funzionalità per fornire consigli sui filmati in tempo reale.

Per ulteriori informazioni su configurazione dei dati in streaming, verifica di Confluent Kafka e test delle performance, consulta "TR-4912: Linee guida sulle Best practice per lo storage a più livelli Confluent Kafka con NetApp".

Apprendimento automatico

Il framework integrato Spark consente di eseguire query ripetute sui set di dati utilizzando la libreria di apprendimento automatico (MLlib). MLlib viene utilizzato in aree come clustering, classificazione e riduzione delle dimensioni per alcune funzioni comuni dei big data, come l'intelligence predittiva, la segmentazione dei clienti per scopi di marketing e l'analisi del sentimento. MLlib viene utilizzato nella sicurezza di rete per eseguire ispezioni in tempo reale dei pacchetti di dati per indicazioni di attività dannose. Aiuta i provider di sicurezza a conoscere le nuove minacce e a restare al passo con gli hacker, proteggendo i propri clienti in tempo reale.

Apprendimento approfondito

TensorFlow è un framework di deep learning diffuso in tutto il settore. TensorFlow supporta la formazione distribuita su un cluster di CPU o GPU. Questo training distribuito consente agli utenti di eseguirlo su una grande quantità di dati con molti livelli profondi.

Fino a poco tempo fa, se volevamo utilizzare TensorFlow con Apache Spark, dovevamo eseguire tutte le ETL necessarie per TensorFlow in PySpark e quindi scrivere i dati nello storage intermedio. Tali dati verranno quindi caricati nel cluster TensorFlow per l'effettivo processo di training. Questo flusso di lavoro richiedeva all'utente di mantenere due diversi cluster, uno per ETL e uno per la formazione distribuita di TensorFlow. L'esecuzione e la manutenzione di più cluster erano in genere noiose e dispendiose in termini di tempo.

DataFrame e RDD nelle versioni precedenti di Spark non erano adatti per l'apprendimento approfondito perché l'accesso casuale era limitato. In Spark 3.0 con il progetto Hydrogen, è stato aggiunto il supporto nativo per i framework di deep learning. Questo approccio consente la pianificazione non basata su MapReduce sul cluster Spark.

Analisi interattiva

Apache Spark è abbastanza veloce da eseguire query esplorative senza campionamenti con linguaggi di sviluppo diversi da Spark, tra cui SQL, R e Python. SPARK utilizza strumenti di visualizzazione per elaborare dati complessi e visualizzarli in modo interattivo. Spark with Structured streaming esegue query interattive in base ai dati in tempo reale in analisi web che consentono di eseguire query interattive in base alla sessione corrente di un visitatore web.

Sistema consigliato

Nel corso degli anni, i sistemi di recommender hanno apportato enormi cambiamenti alla nostra vita, in quanto aziende e consumatori hanno risposto a cambiamenti drastici nello shopping online, nell'intrattenimento online e in molti altri settori. In effetti, questi sistemi sono tra i casi di successo più evidenti dell'ai in produzione. In molti casi pratici di utilizzo, i sistemi consigliati sono combinati con i chatbot o ai conversazionali interfacciati con un backend NLP per ottenere informazioni rilevanti e produrre utili inferenze.

Oggi, molti retailer stanno adottando modelli di business più recenti, come l'acquisto online e il ritiro in negozio, il ritiro in marciapiede, il pagamento automatico, la scansione e la partenza e molto altro ancora. Questi modelli sono diventati preminenti durante la pandemia di COVID-19, rendendo gli acquisti più sicuri e convenienti per i consumatori. L'ai è fondamentale per queste tendenze digitali in crescita, che sono influenzate dal comportamento dei consumatori e viceversa. Per soddisfare le crescenti esigenze dei consumatori, aumentare l'esperienza del cliente, migliorare l'efficienza operativa e aumentare i ricavi, NetApp aiuta i clienti aziendali e le aziende a utilizzare algoritmi di apprendimento automatico e di deep learning per progettare sistemi di raccomandazione più rapidi e precisi.

Esistono diverse tecniche utilizzate per fornire consigli, tra cui il filtraggio collaborativo, i sistemi basati sui contenuti, il modello DLRM (Deep Learning recommender Model) e le tecniche ibride. In precedenza, i clienti utilizzavano PySpark per implementare il filtraggio collaborativo per la creazione di sistemi di raccomandazione. Spark MLlib implementa Alternating Least Squares (ALS) per il filtraggio collaborativo, un algoritmo molto popolare tra le aziende prima dell'ascesa di DLRM.

Elaborazione del linguaggio naturale

L'intelligenza artificiale conversa, resa possibile dall'elaborazione del linguaggio naturale (NLP), è il ramo dell'intelligenza artificiale che aiuta i computer a comunicare con gli esseri umani. La tecnologia NLP è prevalente in tutti i mercati verticali del settore e in molti casi di utilizzo, dagli smart Assistant ai chatbot, alla ricerca con Google e al testo predittivo. Secondo a. "Gartner" Secondo le previsioni, entro il 2022, il 70% delle persone interagirà quotidianamente con le piattaforme di ai convergenti. Per una conversazione di alta qualità tra un essere umano e una macchina, le risposte devono essere rapide, intelligenti e naturali.

I clienti hanno bisogno di una grande quantità di dati per elaborare e formare i propri modelli NLP e ASR (Automatic Speech Recognition). Devono anche spostare i dati all'edge, al core e al cloud e hanno bisogno della potenza necessaria per eseguire l'inferenza in millisecondi per stabilire una comunicazione naturale con gli esseri umani. NetApp ai e Apache Spark sono la combinazione ideale per calcolo, storage, elaborazione dei dati, training sui modelli, messa a punto, e implementazione.

L'analisi del sentimento è un campo di studio all'interno di NLP in cui i sentimenti positivi, negativi o neutri vengono estratti dal testo. L'analisi del sentimento ha una varietà di casi di utilizzo, dalla determinazione delle performance dei dipendenti del centro di supporto nelle conversazioni con i chiamanti alla fornitura di risposte dei chatbot automatizzate appropriate. Inoltre, è stato utilizzato per prevedere il prezzo delle azioni di un'azienda in base alle interazioni tra i rappresentanti dell'azienda e il pubblico durante le chiamate trimestrali sui guadagni. Inoltre, l'analisi del sentimento può essere utilizzata per determinare la posizione del cliente sui prodotti, servizi o supporto forniti dal marchio.

Abbiamo utilizzato "NLP. Scintilla" libreria da "John Snow Labs" Per caricare pipeline preaddestrate e

rappresentazioni di encoder bidirezionali da modelli di Transformers (BERT), tra cui "sentimento di notizie finanziarie" e. "FinBERT", esecuzione di togenizzazione, riconoscimento di entità denominate, training sui modelli, analisi di adattamento e sentimento su larga scala. Spark NLP è l'unica libreria NLP open-source in produzione che offre trasformatori all'avanguardia come BERT, ALBERT, ELECTRA, XLNet, DistillBERT, Roberta, DeBERTA, XLM-Roberta, Longformer, ELMO, Universal sentence Encoder, Google T5, MarianMT e GPT2. La libreria funziona non solo in Python e R, ma anche nell'ecosistema JVM (Java, Scala e Kotlin) su larga scala estendendo Apache Spark in modo nativo.

Principali casi di utilizzo e architetture ai, ML e DL

I principali casi di utilizzo e la metodologia di ai, ML e DL possono essere suddivisi nelle seguenti sezioni:

Pipeline SPARK NLP e deduzione distribuita TensorFlow

Il seguente elenco contiene le librerie NLP open-source più diffuse che sono state adottate dalla community di data science con diversi livelli di sviluppo:

- "Natural Language Toolkit (NLTK)". Il toolkit completo per tutte le tecniche NLP. È stato mantenuto fin dai primi anni 2000.
- "TextBlob". Un tool NLP facile da usare API Python costruita su NLTK e Pattern.
- "Stanford Core NLP". Servizi e pacchetti NLP in Java sviluppati da Stanford NLP Group.
- "GENsim". Topic Modeling for Humans è iniziato come una raccolta di script Python per il progetto Czech Digital Mathematics Library.
- "Spacy". Workflow NLP industriali end-to-end con Python e Cython con accelerazione GPU per i trasformatori.
- "Fasttext". Una libreria NLP gratuita, leggera e open-source per l'apprendimento delle parole e la classificazione delle frasi creata dal laboratorio ai Research (FAIR) di Facebook.

Spark NLP è una soluzione singola e unificata per tutte le attività e i requisiti NLP che consente di ottenere software scalabile, dalle performance elevate e ad alta precisione basato su NLP per casi di utilizzo in produzione reali. Sfrutta l'apprendimento del trasferimento e implementa gli algoritmi e i modelli più recenti nella ricerca e nei vari settori. A causa della mancanza di supporto completo da parte di Spark per le librerie di cui sopra, Spark NLP è stato costruito su "SPARK ML" Sfruttare il motore di elaborazione dei dati distribuiti in-memory di Spark per scopi generali come libreria NLP di livello Enterprise per flussi di lavoro di produzione mission-critical. I suoi annotatori utilizzano algoritmi basati su regole, machine learning e TensorFlow per potenziare le implementazioni di deep learning. Questo copre i comuni compiti di NLP, inclusi, a titolo esemplificativo ma non esaustivo, la tokenizzazione, la lemmatizzazione, lo stemming, il tagging part-of-speech, il riconoscimento di entità nominate, controllo ortografico e analisi del sentimento.

Le rappresentazioni di encoder bidirezionali di Transformers (BERT) sono tecniche di apprendimento automatico basate su trasformatore per NLP. Ha reso popolare il concetto di pre-training e fine tuning. L'architettura dei trasformatori di BERT è nata dalla traduzione automatica, che modella le dipendenze a lungo termine meglio dei modelli di linguaggio basati su Neural Network (RNN) ricorrenti. Ha inoltre introdotto il task Masked Language Modeling (MLM), in cui il 15% casuale di tutti i token viene mascherato e il modello li prevede, consentendo una reale bidirezionalità.

L'analisi del sentimento finanziario è complessa a causa del linguaggio specializzato e della mancanza di dati etichettati in quel dominio. "FinBERT", Un modello linguistico basato su BERT preaddestrato, è stato adattato al dominio "Reuters TRC2", un corpus finanziario, e messo a punto con i dati etichettati ("Financial PhraseBank") per la classificazione del sentimento finanziario. I ricercatori hanno estratto 4, 500 frasi dagli articoli di notizie con termini finanziari. Poi 16 esperti e master studenti con background finanziari hanno

etichettato le frasi come positive, neutrali e negative. Abbiamo creato un workflow Spark end-to-end per analizzare il sentimento per le trascrizioni delle chiamate delle aziende Top-10 NASDAQ dal 2016 al 2020 utilizzando FinBERT e due altre pipeline preformate ("Analisi del sentimento per le notizie finanziarie", "Spiegare il documento DL") Di Spark NLP.

Il motore di deep learning sottostante per Spark NLP è TensorFlow, una piattaforma open source end-to-end per l'apprendimento automatico che consente la creazione di modelli semplici, una produzione ML solida ovunque e potenti sperimentazioni per la ricerca. Pertanto, quando si eseguono le nostre pipeline in Spark `yarn cluster` In pratica, abbiamo eseguito TensorFlow distribuito con parallelizzazione di dati e modelli tra un nodo master e più nodi di lavoro, oltre allo storage collegato alla rete montato sul cluster.

Formazione distribuita Horovod

La convalida Hadoop principale per le performance correlate a MapReduce viene eseguita con TeraGen, TeraSort, TeraValidate e DFSIO (lettura e scrittura). I risultati della convalida TeraGen e TeraSort sono presentati nella "TR-3969: Soluzioni NetApp per Hadoop" Per e-Series e nella sezione "Tiering dello storage" (xref) per AFF.

In base alle richieste dei clienti, riteniamo che la formazione distribuita con Spark sia uno dei casi di utilizzo più importanti. In questo documento, abbiamo utilizzato "Hovorod su Spark" Per convalidare le performance di Spark con le soluzioni di cloud ibrido, nativo e on-premise di NetApp utilizzando i controller di storage NetApp All Flash FAS (AFF), Azure NetApp Files e StorageGRID.

Il pacchetto Horovod on Spark offre un comodo wrapper intorno a Horovod che semplifica l'esecuzione di workload di training distribuiti nei cluster Spark, consentendo un loop di progettazione di modelli ristretti in cui l'elaborazione dei dati, la formazione sui modelli e la valutazione dei modelli vengono eseguite in Spark, dove risiedono i dati di formazione e deduzione.

Esistono due API per l'esecuzione di Horovod su Spark: Un'API di stima di alto livello e un'API di esecuzione di livello inferiore. Sebbene entrambi utilizzino lo stesso meccanismo sottostante per lanciare Horovod sugli esecutori di Spark, l'API Estimator astratta l'elaborazione dei dati, il loop di training del modello, il checkpoint del modello, la raccolta di metriche e il training distribuito. Abbiamo utilizzato gli stimatori di Horovod Spark, TensorFlow e keras per una preparazione dei dati end-to-end e un workflow di training distribuito basato su "Vendita al negozio Kaggle Rossmann" concorrenza.

Lo script `keras_spark_horovod_rossmann_estimator.py` si trova nella sezione "Script Python per ogni caso di utilizzo principale." Contiene tre parti:

- La prima parte esegue varie fasi di pre-elaborazione dei dati su un set iniziale di file CSV forniti da Kaggle e raccolti dalla community. I dati di input vengono separati in un set di training con un `Validation` e un dataset di test.
- La seconda parte definisce un modello di rete neurale profonda (DNN) con funzione di attivazione sigmoid logaritmica e un ottimizzatore Adam, ed esegue un training distribuito del modello utilizzando Horovod su Spark.
- La terza parte esegue la previsione sul set di dati di test utilizzando il modello migliore che riduce al minimo l'errore medio assoluto complessivo del set di convalida. Viene quindi creato un file CSV di output.

Vedere la sezione ""Apprendimento automatico"" per diversi risultati di confronto tra runtime.

Deep learning multi-worker con keras per la previsione CTR

Con i recenti progressi nelle piattaforme E nelle applicazioni ML, è ora molto importante concentrarsi sull'apprendimento su larga scala. Il tasso di click-through (CTR) è definito come il numero medio di click-through per cento impressioni di annunci online (espresso in percentuale). È ampiamente adottato come

parametro chiave in diversi mercati verticali e casi di utilizzo del settore, tra cui digital marketing, retail, e-commerce e service provider. Consulta la nostra "TR-4904: Formazione distribuita in Azure - previsione dei tassi click-through" Per ulteriori dettagli sulle applicazioni di CTR e un'implementazione del workflow ai cloud end-to-end con Kubernetes, ETL di dati distribuiti e training sui modelli con DAK e CUDA ML.

In questo report tecnico abbiamo utilizzato una variante di "Set di dati Click Logs Criteo Terabyte" (Vedere TR-4904) per l'apprendimento approfondito distribuito multi-worker che utilizza keras per creare un workflow Spark con modelli DCN (Deep and Cross Network), confrontando le sue performance in termini di funzione di errore di perdita di log con un modello di riferimento Spark ML Logistic Regression. DCN acquisisce in modo efficiente le interazioni efficaci delle funzioni di gradi limitati, apprende interazioni altamente non lineari, non richiede alcuna progettazione manuale delle funzioni o ricerca completa e ha un costo di calcolo basso.

I dati per i sistemi recommender su scala web sono per lo più discreti e categorici, il che porta a un ampio e sparso spazio di funzionalità che è difficile per l'esplorazione delle funzionalità. Questo ha limitato la maggior parte dei sistemi su larga scala a modelli lineari come la regressione logistica. Tuttavia, l'identificazione delle funzionalità spesso predittive e allo stesso tempo l'esplorazione di funzioni incrociate rare o invisibili è la chiave per fare buone previsioni. I modelli lineari sono semplici, interpretabili e facili da scalare, ma sono limitati nel loro potere espressivo.

Le funzionalità incrociate, d'altro canto, si sono dimostrate significative nel migliorare l'espressività dei modelli. Sfortunatamente, spesso richiede un'ingegneria delle funzionalità manuale o una ricerca completa per identificare tali funzionalità. La generalizzazione di interazioni di funzionalità non visibili è spesso difficile. L'utilizzo di una rete interneurale come DCN evita l'ingegneria delle funzionalità specifiche dell'attività, applicando esplicitamente il passaggio delle funzionalità in modo automatico. La rete incrociata è costituita da più livelli, in cui il più alto grado di interazione è determinato in modo probabile dalla profondità del livello. Ogni livello produce interazioni di ordine superiore in base a quelle esistenti e mantiene le interazioni dai livelli precedenti.

Una rete neurale profonda (DNN) ha la promessa di acquisire interazioni molto complesse tra le varie funzionalità. Tuttavia, rispetto alla rete DCN, richiede quasi un ordine di grandezza più parametri, non è in grado di formare funzioni incrociate in modo esplicito e potrebbe non riuscire ad apprendere in modo efficiente alcuni tipi di interazioni tra funzionalità. La rete è efficiente in termini di memoria e facile da implementare. La formazione congiunta dei componenti Cross e DNN acquisisce in modo efficiente le interazioni predittive delle funzionalità e offre performance all'avanguardia sul set di dati CTR Criteo.

Un modello DCN inizia con un livello di incorporamento e stacking, seguito da una rete incrociata e una rete profonda in parallelo. Questi a loro volta sono seguiti da un livello di combinazione finale che combina le uscite dalle due reti. I dati di input possono essere un vettore con funzioni sparse e dense. In Spark, entrambi "ml" e. "mllib" le librerie contengono il tipo `SparseVector`. È quindi importante che gli utenti distinguano i due e si ricordino quando chiamano le rispettive funzioni e metodi. Nei sistemi recommender su scala web come la previsione CTR, gli input sono per lo più caratteristiche categoriche, ad esempio `'country=usa'`. Tali caratteristiche sono spesso codificate come vettori one-hot, ad esempio, `'[0,1,0, …]'`. One-hot-encoding (OHE) con `SparseVector` è utile quando si gestiscono set di dati reali con vocabolari in continua evoluzione e in crescita. Abbiamo modificato gli esempi in "DeepCTR" Elaborare vocabolari di grandi dimensioni, creando vettori di incorporamento nel livello di incorporamento e stacking della nostra rete DCN.

Il "Dataset Criteo Display Ads" prevede il tasso di click-through degli annunci. Dispone di 13 caratteristiche intere e 26 caratteristiche categoriche in cui ogni categoria ha un'elevata cardinalità. Per questo set di dati, un miglioramento di 0.001 nella perdita di log è praticamente significativo a causa delle grandi dimensioni dell'input. Un piccolo miglioramento della precisione di previsione per una base di utenti di grandi dimensioni può potenzialmente portare a un aumento significativo dei ricavi di un'azienda. Il set di dati contiene 11 GB di log utente da un periodo di 7 giorni, che equivale a circa 41 milioni di record. Abbiamo utilizzato Spark `dataFrame.randomSplit()function` suddividere casualmente i dati per il training (80%), la convalida incrociata (10%) e il restante 10% per il test.

DCN è stato implementato su TensorFlow con keras. L'implementazione del processo di training del modello con DCN comprende quattro componenti principali:

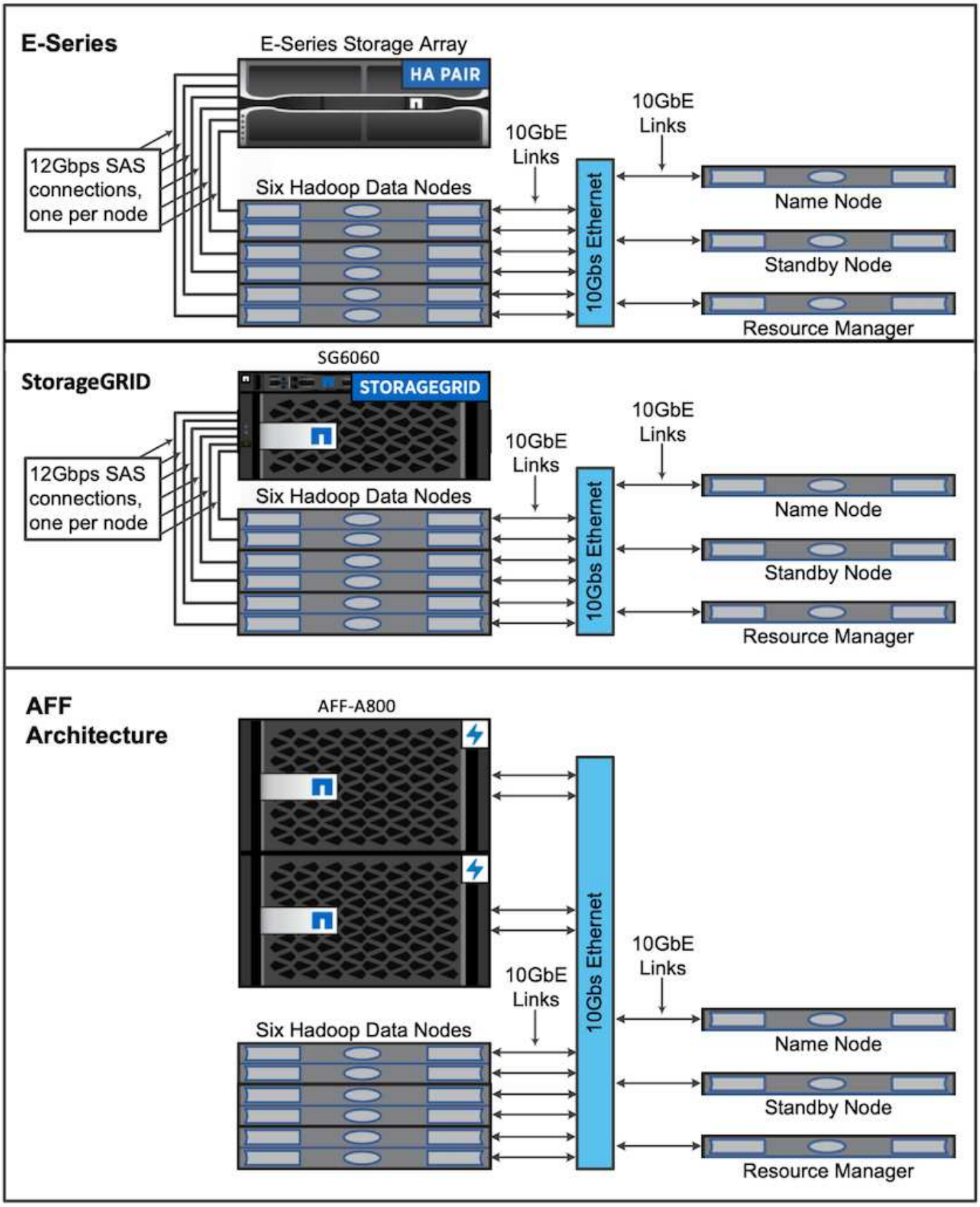
- **Elaborazione e incorporamento dei dati.** le funzionalità a valore reale vengono normalizzate applicando una trasformazione del log. Per le funzionalità categoriche, le funzionalità sono incorporate in vettori densi di dimensione 6×(categoria cardinalità)1/4. Concatenando tutte le incorporazioni si ottiene un vettore di dimensione 1026.
- **Optimization.** abbiamo applicato l'ottimizzazione stocastica mini-batch con Adam Optimizer. La dimensione del batch è stata impostata su 512. La normalizzazione batch è stata applicata alla rete profonda e la norma del gradiente clip è stata impostata su 100.
- **Regolarizzazione.** abbiamo utilizzato la sospensione anticipata, in quanto la regolarizzazione L2 o il dropout non sono stati trovati efficaci.
- **Hyperparameters.** i risultati vengono riportati in base a una ricerca in griglia sul numero di livelli nascosti, la dimensione del livello nascosto, la velocità di apprendimento iniziale e il numero di livelli incrociati. Il numero di livelli nascosti variava da 2 a 5, con dimensioni dei livelli nascosti comprese tra 32 e 1024. Per DCN, il numero di strati incrociati era da 1 a 6. Il tasso di apprendimento iniziale è stato ottimizzato da 0.0001 a 0.001 con incrementi di 0.0001. Tutti gli esperimenti hanno subito interrotto la fase di training 150,000, oltre la quale ha iniziato a verificarsi un overfitting.

Oltre a DCN, abbiamo anche testato altri modelli di deep-learning molto diffusi per la previsione CTR, tra cui "DeepFM", "XDeepFM", "Int. Auto", e. "DCN v2".

Architetture utilizzate per la convalida

Per questa convalida, abbiamo utilizzato quattro nodi di lavoro e un nodo master con una coppia ha AFF-A800. Tutti i membri del cluster erano connessi tramite switch di rete 10 GbE.

Per la convalida della soluzione NetApp Spark, abbiamo utilizzato tre diversi controller di storage: E5760, E5724 e AFF-A800. I controller di storage e-Series erano collegati a cinque nodi dati con connessioni SAS a 12 Gbps. Il controller di storage AFF ha-Pair offre volumi NFS esportati attraverso connessioni 10 GbE ai nodi di lavoro Hadoop. I membri del cluster Hadoop erano connessi tramite connessioni 10GbE nelle soluzioni e-Series, AFF e StorageGRID Hadoop.



Risultati del test

Abbiamo utilizzato gli script TeraSort e TeraValidate nello strumento di benchmarking

TeraGen per misurare la convalida delle performance Spark con le configurazioni E5760, E5724 e AFF-A800. Inoltre, sono stati testati tre casi di utilizzo principali: Pipeline SPARK NLP e training distribuito TensorFlow, training distribuito Horovod e deep learning multi-worker con keras per la previsione CTR con DeepFM.

Per la convalida di e-Series e StorageGRID, abbiamo utilizzato il fattore di replica Hadoop 2. Per la convalida AFF, abbiamo utilizzato una sola fonte di dati.

La seguente tabella elenca la configurazione hardware per la convalida delle prestazioni di Spark.

Tipo	Nodi di lavoro Hadoop	Tipo di disco	Dischi per nodo	Controller dello storage
SG6060	4	SAS	12	Singola coppia ad alta disponibilità (ha)
E5760	4	SAS	60	Coppia ha singola
E5724	4	SAS	24	Coppia ha singola
AFF800	4	SSD	6	Coppia ha singola

La seguente tabella elenca i requisiti software.

Software	Versione
RHEL	7.9
Ambiente di runtime OpenJDK	1.8.0
Server VM OpenJDK a 64 bit	25.302
Git	2.24.1
GCC/G++	11.2.1
Scintilla	3.2.1
PySpark	3.1.2
SparkNLP	3.4.2
TensorFlow	2.9.0
Ere	2.9.0
Horovod	0.24.3

Analisi del sentimento finanziario

Abbiamo pubblicato "TR-4910: Analisi del sentimento da Customer Communications con NetApp ai", In cui è stata costruita una pipeline di ai conversazionale end-to-end utilizzando "NetApp DataOps Toolkit", Storage AFF e sistema NVIDIA DGX. La pipeline esegue l'elaborazione del segnale audio batch, il riconoscimento vocale automatico (ASR), l'apprendimento del trasferimento e l'analisi del sentimento utilizzando il DataOps Toolkit, "SDK NVIDIA Riva"e il "Framework di Tao". Espandendo il caso d'uso dell'analisi del sentimento nel settore dei servizi finanziari, abbiamo creato un workflow SparkNLP, caricato tre modelli BERT per varie attività NLP, come il riconoscimento delle entità nominate, e ottenuto un sentimento a livello di frase per le chiamate trimestrali sui guadagni delle prime 10 aziende NASDAQ.

Il seguente script `sentiment_analysis_spark. py` Utilizza il modello FinBERT per elaborare le trascrizioni in HDFS e produrre conteggi di sentimenti positivi, neutri e negativi, come mostrato nella seguente tabella:

```
-bash-4.2$ time ~/anaconda3/bin/spark-submit
--packages com.johnsnowlabs.nlp:spark-nlp_2.12:3.4.3
--master yarn
--executor-memory 5g
--executor-cores 1
--num-executors 160
--conf spark.driver.extraJavaOptions="-Xss10m -XX:MaxPermSize=1024M"
--conf spark.executor.extraJavaOptions="-Xss10m -XX:MaxPermSize=512M"
/sparkusecase/tr-4570-nlp/sentiment_analysis_spark.py
hdfs:///data1/Transcripts/
> ./sentiment_analysis_hdfs.log 2>&1
real13m14.300s
user557m11.319s
sys4m47.676s
```

La seguente tabella elenca l'analisi del sentimento a livello di frase e di chiamata degli utili per le prime 10 aziende NASDAQ dal 2016 al 2020.

I conteggi dei sentimenti e la percentuale	Tutte le 10 aziende	AAAPL	AMD	N. AMZN	CSCO	GOOGL	INTC	MSFT	NVDA
Conteggi positivi	7447	1567	743	290	682	826	824	904	417
Conteggi neutrali	64067	6856	7596	5086	6650	5914	6099	5715	6189
Conteggi negativi	1787	253	213	84	189	97	282	202	89
Conteggi senza categoria	196	0	0	76	0	0	0	1	0
(conteggi totali)	73497	8676	8552	5536	7521	6837	7205	6822	6695

In termini di percentuali, la maggior parte delle frasi pronunciate dagli amministratori delegati e dai CFO è fattuale e quindi ha un sentimento neutrale. Durante una chiamata sui guadagni, gli analisti pongono domande che potrebbero trasmettere un sentimento positivo o negativo. Vale la pena di analizzare in maniera quantitativa il modo in cui il sentimento negativo o positivo influisce sui prezzi delle azioni nello stesso giorno o nel giorno successivo di negoziazione.

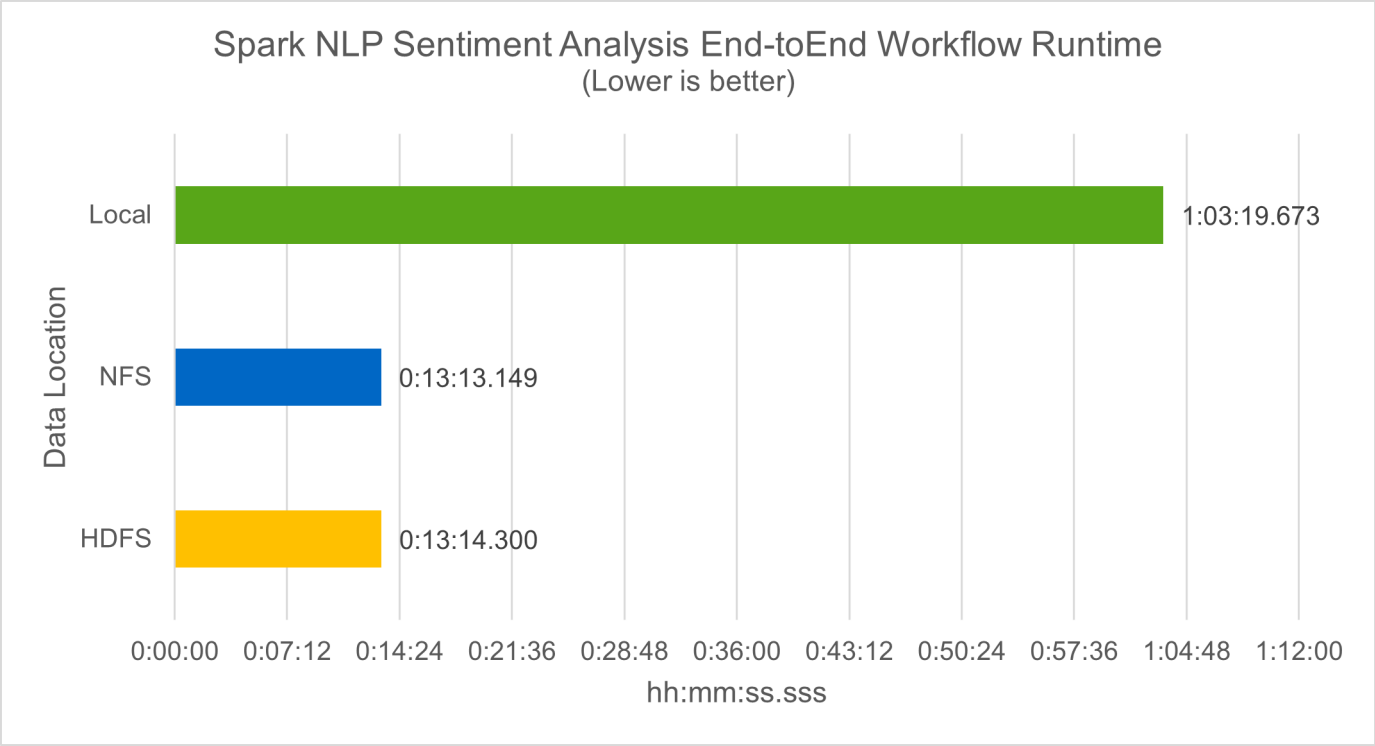
La seguente tabella elenca l'analisi del sentimento a livello di frase per le prime 10 aziende NASDAQ, espressa in percentuale.

Percentuale di sentimento	Tutte le 10 aziende	AAAPL	AMD	N. AMZN	CSCO	GOOGL	INTC	MSFT	NVDA
Positivo	10.13%	18.06%	8.69%	5.24%	9.07%	12.08%	11.44%	13.25%	6.23%
Neutro	87.17%	79.02%	88.82%	91.87%	88.42%	86.50%	84.65%	83.77%	92.44%
Negativo	2.43%	2.92%	2.49%	1.52%	2.51%	1.42%	3.91%	2.96%	1.33%
Senza categoria	0.27%	0%	0%	1.37%	0%	0%	0%	0.01%	0%

In termini di runtime del workflow, abbiamo riscontrato un significativo miglioramento di 4,78 volte `local` A un ambiente distribuito in HDFS e un ulteriore miglioramento del 0.14% grazie all'utilizzo di NFS.

```
-bash-4.2$ time ~/anaconda3/bin/spark-submit
--packages com.johnsnowlabs.nlp:spark-nlp_2.12:3.4.3
--master yarn
--executor-memory 5g
--executor-cores 1
--num-executors 160
--conf spark.driver.extraJavaOptions="-Xss10m -XX:MaxPermSize=1024M"
--conf spark.executor.extraJavaOptions="-Xss10m -XX:MaxPermSize=512M"
/sparkusecase/tr-4570-nlp/sentiment_analysis_spark.py
file:///sparkdemo/sparknlp/Transcripts/
> ./sentiment_analysis_nfs.log 2>&1
real13m13.149s
user537m50.148s
sys4m46.173s
```

Come mostrato nella figura seguente, il parallelismo dei dati e dei modelli ha migliorato l'elaborazione dei dati e la velocità di deduzione del modello TensorFlow distribuito. La posizione dei dati in NFS ha prodotto un runtime leggermente migliore perché il collo di bottiglia del workflow è il download di modelli preformati. Se aumentiamo le dimensioni del set di dati delle trascrizioni, il vantaggio di NFS è più evidente.



Formazione distribuita con performance Horovod

Il seguente comando ha prodotto informazioni di runtime e un file di log nel cluster Spark utilizzando un singolo `master` nodo con 160 esecutori ciascuno con un core. La memoria dell'esecutore era limitata a 5 GB per evitare errori di memoria esaurita. Vedere la sezione ""Script Python per ogni caso di utilizzo principale"" per ulteriori dettagli sull'elaborazione dei dati, sul training del modello e sul calcolo della precisione del modello in `keras_spark_horovod_rossmann_estimator.py`.

```
(base) [root@n138 horovod]# time spark-submit
--master local
--executor-memory 5g
--executor-cores 1
--num-executors 160
/sparkusecase/horovod/keras_spark_horovod_rossmann_estimator.py
--epochs 10
--data-dir file:///sparkusecase/horovod
--local-submission-csv /tmp/submission_0.csv
--local-checkpoint-file /tmp/checkpoint/
> /tmp/keras_spark_horovod_rossmann_estimator_local. log 2>&1
```

Il runtime risultante con dieci epoche di training è stato il seguente:

```
real43m34.608s
user12m22.057s
sys2m30.127s
```

Ci sono voluti più di 43 minuti per elaborare i dati di input, formare un modello DNN, calcolare la precisione e produrre checkpoint TensorFlow e un file CSV per i risultati delle previsioni. Abbiamo limitato il numero di epoche di training a 10, che in pratica è spesso impostato a 100 per garantire una precisione del modello soddisfacente. Il tempo di training in genere è in grado di scalare in modo lineare con il numero di epoche.

Successivamente, abbiamo utilizzato i quattro nodi di lavoro disponibili nel cluster ed eseguito lo stesso script in `yarn` Modalità con dati in HDFS:

```
(base) [root@n138 horovod]# time spark-submit
--master yarn
--executor-memory 5g
--executor-cores 1 --num-executors 160
/sparkusecase/horovod/keras_spark_horovod_rossmann_estimator.py
--epochs 10
--data-dir hdfs:///user/hdfs/tr-4570/experiments/horovod
--local-submission-csv /tmp/submission_1.csv
--local-checkpoint-file /tmp/checkpoint/
> /tmp/keras_spark_horovod_rossmann_estimator_yarn.log 2>&1
```

Il runtime risultante è stato migliorato come segue:

```
real8m13.728s
user7m48.421s
sys1m26.063s
```

Con il modello di Horovod e il parallelismo dei dati in Spark, abbiamo visto una velocità di runtime di 5,29x `yarn` contro `local` con dieci epoche di training. Questo è mostrato nella figura seguente con le legende `HDFS` e. `Local`. Il training sul modello DNN TensorFlow sottostante può essere ulteriormente accelerato con le GPU, se disponibili. Prevediamo di condurre questo test e di pubblicare i risultati in un report tecnico futuro.

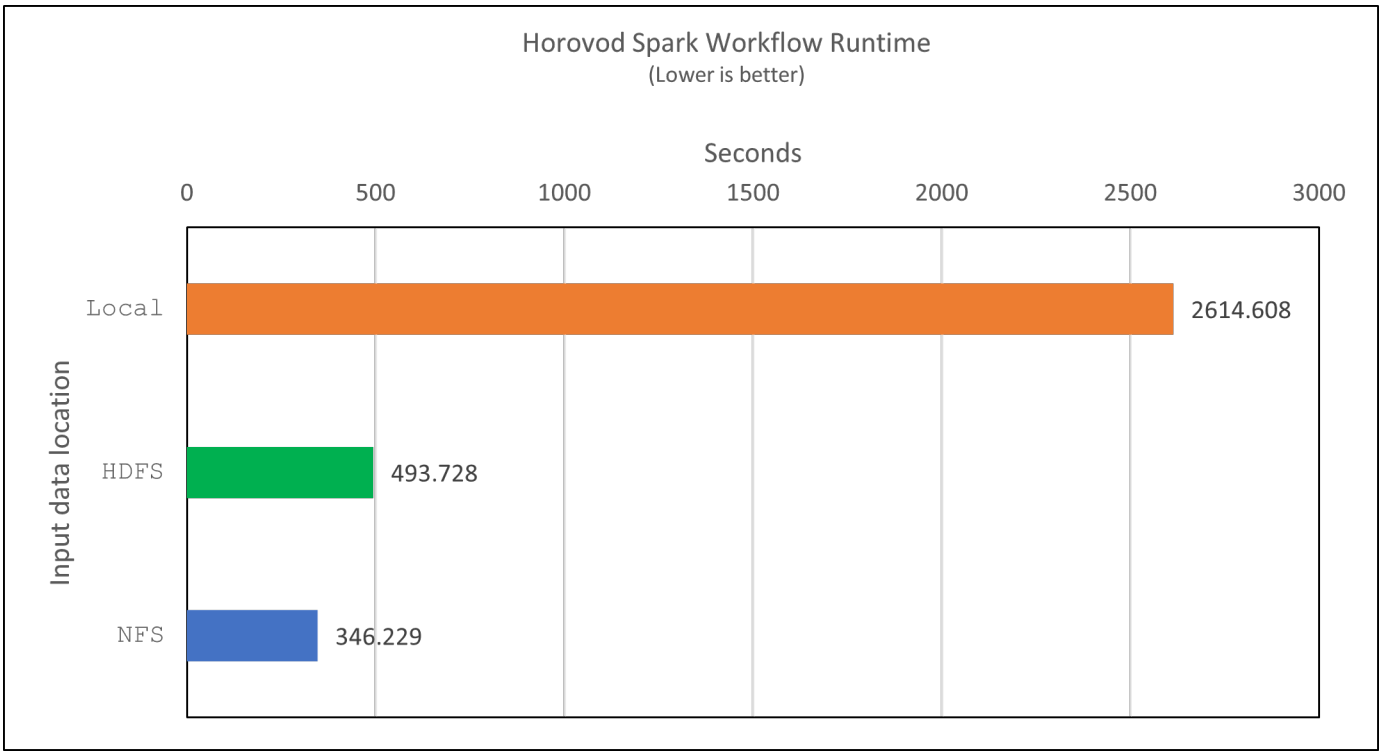
Il nostro test successivo ha confrontato i runtime con i dati di input che risiedono in NFS rispetto a HDFS. Il volume NFS su AFF A800 è stato montato `/sparkdemo/horovod` Tra i cinque nodi (un master, quattro dipendenti) nel cluster Spark. Abbiamo eseguito un comando simile a quello dei test precedenti, con `--data-dir` Parametro ora che punta al montaggio NFS:

```
(base) [root@n138 horovod]# time spark-submit
--master yarn
--executor-memory 5g
--executor-cores 1
--num-executors 160
/sparkusecase/horovod/keras_spark_horovod_rossmann_estimator.py
--epochs 10
--data-dir file:///sparkdemo/horovod
--local-submission-csv /tmp/submission_2.csv
--local-checkpoint-file /tmp/checkpoint/
> /tmp/keras_spark_horovod_rossmann_estimator_nfs.log 2>&1
```

Il runtime risultante con NFS è stato il seguente:

```
real 5m46.229s
user 5m35.693s
sys  1m5.615s
```

Si è verificato un ulteriore velocismo di 1,43 volte, come mostrato nella figura seguente. Pertanto, con uno storage all-flash NetApp collegato al cluster, i clienti possono usufruire dei vantaggi di un rapido trasferimento e distribuzione dei dati per i flussi di lavoro di Horovod Spark, ottenendo una velocità di 7,55 volte superiore rispetto all'esecuzione su un singolo nodo.



Modelli di deep learning per performance di previsione CTR

Per i sistemi di raccomandazione progettati per massimizzare il CTR, è necessario imparare sofisticate interazioni di funzionalità dietro i comportamenti degli utenti che possono essere calcolati matematicamente da basso ordine a alto ordine. Le interazioni di funzionalità di basso e alto ordine devono essere ugualmente importanti per un buon modello di deep learning senza polarizzare l'uno o l'altro. DeepFM (Deep Factorization Machine), una rete neurale basata su macchine per la fattorizzazione, combina macchine per la fattorizzazione per consigli e un apprendimento approfondito per l'apprendimento delle funzionalità in una nuova architettura di rete neurale.

Anche se le macchine convenzionali di fattorizzazione modellano le interazioni a coppie come prodotto interno di vettori latenti tra le funzionalità e possono teoricamente acquisire informazioni di ordine elevato, in pratica, i professionisti dell'apprendimento automatico di solito utilizzano solo le interazioni di funzionalità di secondo ordine a causa dell'elevata complessità di calcolo e storage. Varianti di rete neurali profonde come quelle di Google " modelli profondi" d'altro canto, impara sofisticate interazioni di funzionalità in una struttura di rete ibrida combinando un modello ampio lineare e un modello profondo.

Ci sono due input per questo modello ampio e profondo, uno per il modello ampio sottostante e l'altro per il deep, l'ultima parte del quale richiede ancora un esperto di ingegneria delle funzionalità e quindi rende la tecnica meno generalizzabile per altri domini. A differenza di Wide & Deep Model, DeepFM può essere addestrato in modo efficiente con funzionalità raw senza alcuna progettazione delle funzioni, perché la sua parte ampia e profonda condividono lo stesso input e lo stesso vettore di inclusione.

Abbiamo elaborato per la prima volta il Criteo `train.txt` (11 GB) in un file CSV denominato `ctr_train.csv` Memorizzato in un montaggio NFS `/sparkdemo/tr-4570-data` utilizzo di `run_classification_criteo_spark.py` dalla sezione ""Script Python per ogni caso di utilizzo principale"." All'interno di questo script, la funzione `process_input_file` esegue diversi metodi di stringa per rimuovere le schede e inserire `','` come delimitatore e. `'\n'` come novità. Tenere presente che è necessario elaborare solo l'originale `train.txt` una volta, in modo che il blocco di codice sia visualizzato come commenti.

Per i seguenti test di diversi modelli DL, abbiamo utilizzato `ctr_train.csv` come file di input. Nelle successive esecuzioni dei test, il file CSV di input è stato letto in un Spark DataFrame con schema contenente un campo di `'label'`, caratteristiche ad alta densità di numeri interi `['I1', 'I2', 'I3', …, 'I13']`e funzioni sparse `['C1', 'C2', 'C3', …, 'C26']`. Quanto segue `spark-submit` Command acquisisce un input CSV, allena i modelli DeepFM con una suddivisione del 20% per la convalida incrociata e sceglie il modello migliore dopo dieci epoche di training per calcolare l'accuratezza della previsione sul set di test:

```
(base) [root@n138 ~]# time spark-submit --master yarn --executor-memory 5g
--executor-cores 1 --num-executors 160
/sparkusecase/DeepCTR/examples/run_classification_criteo_spark.py --data
-dir file:///sparkdemo/tr-4570-data >
/tmp/run_classification_criteo_spark_local.log 2>&1
```

Tenere presente che dal file di dati `ctr_train.csv` È superiore a 11 GB, è necessario impostare un valore sufficiente `spark.driver.maxResultSize` maggiore della dimensione del set di dati per evitare errori.

```
 spark = SparkSession.builder \
    .master("yarn") \
    .appName("deep_ctr_classification") \
    .config("spark.jars.packages", "io.github.ravwojdyla:spark-schema-
utils_2.12:0.1.0") \
    .config("spark.executor.cores", "1") \
    .config('spark.executor.memory', '5gb') \
    .config('spark.executor.memoryOverhead', '1500') \
    .config('spark.driver.memoryOverhead', '1500') \
    .config("spark.sql.shuffle.partitions", "480") \
    .config("spark.sql.execution.arrow.enabled", "true") \
    .config("spark.driver.maxResultSize", "50gb") \
    .getOrCreate()
```

In quanto sopra `SparkSession.builder` anche la configurazione è stata abilitata "Freccia Apache", Che converte un DataFrame Spark in un DataFrame Pandas con `df.toPandas()` metodo.

```
22/06/17 15:56:21 INFO scheduler.DAGScheduler: Job 2 finished: toPandas at
/sparkusecase/DeepCTR/examples/run_classification_criteo_spark.py:96, took
627.126487 s
Obtained Spark DF and transformed to Pandas DF using Arrow.
```

Dopo la suddivisione casuale, nel set di dati di training sono presenti più di 36 M di righe e 9 M di esempi nel set di test:

```
Training dataset size =  36672493
Testing dataset size =  9168124
```

Poiché questo report tecnico è incentrato sul test della CPU senza utilizzare alcuna GPU, è fondamentale creare TensorFlow con i flag appropriati del compilatore. Questo passaggio evita di invocare librerie con accelerazione GPU e sfrutta al meglio le istruzioni AVX (Advanced Vector Extensions) e AVX2 di TensorFlow. Queste funzionalità sono progettate per calcoli algebrici lineari come addizione vettorizzata, moltiplicazioni di matrice all'interno di un training feed-forward o DNN back-propagation. L'istruzione FMA (Fused Multiply Add) disponibile con AVX2 che utilizza registri a virgola mobile (FP) a 256 bit è ideale per i tipi di dati e codice intero, con una velocità fino a 2 volte superiore. Per il codice FP e i tipi di dati, AVX2 raggiunge una velocità dell'8% su AVX.

```
2022-06-18 07:19:20.101478: I
tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary
is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the
appropriate compiler flags.
```

Per creare TensorFlow dall'origine, NetApp consiglia di utilizzare "Bazel". Per il nostro ambiente, abbiamo eseguito i seguenti comandi nel prompt della shell per l'installazione `dnf`, `dnf-plugins`E Bazel.

```
yum install dnf
dnf install 'dnf-command(copr)'
dnf copr enable vbatts/bazel
dnf install bazel5
```

È necessario abilitare GCC 5 o versioni successive per utilizzare le funzionalità C++17 durante il processo di creazione, fornito da RHEL con Software Collections Library (SCL). I seguenti comandi vengono installati `devtoolset` E GCC 11.2.1 sul nostro cluster RHEL 7.9:

```
subscription-manager repos --enable rhel-server-rhscl-7-rpms
yum install devtoolset-11-toolchain
yum install devtoolset-11-gcc-c++
yum update
scl enable devtoolset-11 bash
. /opt/rh/devtoolset-11/enable
```

Si noti che gli ultimi due comandi sono disponibili `devtoolset-11`, che utilizza `/opt/rh/devtoolset-11/root/usr/bin/gcc` (GCC 11.2.1). Inoltre, assicurarsi di `git` La versione è superiore alla 1.8.3 (fornita con RHEL 7.9). Fare riferimento a questo "articolo" per l'aggiornamento `git` a 2.24.1.

Supponiamo che tu abbia già clonato l'ultimo repo master TensorFlow. Quindi, creare un `workspace` directory con un `WORKSPACE` File per la creazione di TensorFlow dall'origine con AVX, AVX2 e FMA. Eseguire `configure` E specificare la posizione binaria di Python corretta. "CUDA" È disattivato per i test perché non abbiamo utilizzato una GPU. R `.bazelrc` il file viene generato in base alle impostazioni. Inoltre, abbiamo modificato il file e il set `build --define=no_hdfs_support=false` Per attivare il supporto HDFS. Fare riferimento a. `.bazelrc` nella sezione ""Script Python per ogni caso di utilizzo principale"," per un elenco completo di impostazioni e flag.

```
./configure
bazel build -c opt --copt=-mavx --copt=-mavx2 --copt=-mfma --copt=
-mfpmath=both -k //tensorflow/tools/pip_package:build_pip_package
```

Dopo aver creato TensorFlow con i flag corretti, eseguire il seguente script per elaborare il set di dati Criteo Display Ads, formare un modello DeepFM e calcolare l'area sotto la curva caratteristica operativa ricevitore (ROC AUC) in base ai punteggi di previsione.

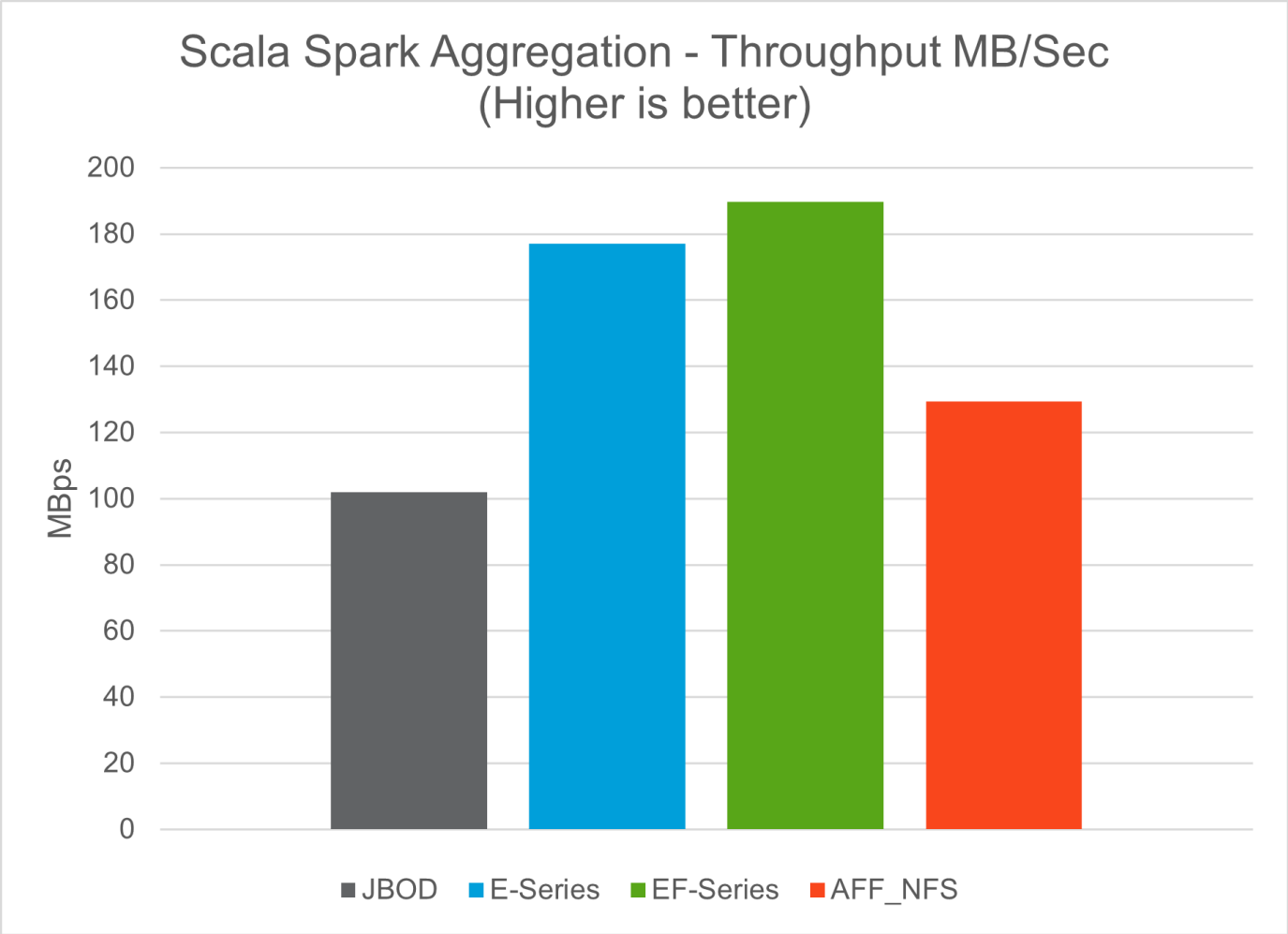
```
(base) [root@n138 examples]# ~/anaconda3/bin/spark-submit
--master yarn
--executor-memory 15g
--executor-cores 1
--num-executors 160
/sparkusecase/DeepCTR/examples/run_classification_criteo_spark.py
--data-dir file:///sparkdemo/tr-4570-data
> . /run_classification_criteo_spark_nfs.log 2>&1
```

Dopo dieci epoche di training, abbiamo ottenuto il punteggio AUC nel set di dati di test:

```
Epoch 1/10
125/125 - 7s - loss: 0.4976 - binary_crossentropy: 0.4974 - val_loss:
0.4629 - val_binary_crossentropy: 0.4624
Epoch 2/10
125/125 - 1s - loss: 0.3281 - binary_crossentropy: 0.3271 - val_loss:
0.5146 - val_binary_crossentropy: 0.5130
Epoch 3/10
125/125 - 1s - loss: 0.1948 - binary_crossentropy: 0.1928 - val_loss:
0.6166 - val_binary_crossentropy: 0.6144
Epoch 4/10
125/125 - 1s - loss: 0.1408 - binary_crossentropy: 0.1383 - val_loss:
0.7261 - val_binary_crossentropy: 0.7235
Epoch 5/10
125/125 - 1s - loss: 0.1129 - binary_crossentropy: 0.1102 - val_loss:
0.7961 - val_binary_crossentropy: 0.7934
Epoch 6/10
125/125 - 1s - loss: 0.0949 - binary_crossentropy: 0.0921 - val_loss:
0.9502 - val_binary_crossentropy: 0.9474
Epoch 7/10
125/125 - 1s - loss: 0.0778 - binary_crossentropy: 0.0750 - val_loss:
1.1329 - val_binary_crossentropy: 1.1301
Epoch 8/10
125/125 - 1s - loss: 0.0651 - binary_crossentropy: 0.0622 - val_loss:
1.3794 - val_binary_crossentropy: 1.3766
Epoch 9/10
125/125 - 1s - loss: 0.0555 - binary_crossentropy: 0.0527 - val_loss:
1.6115 - val_binary_crossentropy: 1.6087
Epoch 10/10
125/125 - 1s - loss: 0.0470 - binary_crossentropy: 0.0442 - val_loss:
1.6768 - val_binary_crossentropy: 1.6740
test AUC 0.6337
```

In modo simile ai casi di utilizzo precedenti, abbiamo confrontato il runtime del workflow Spark con i dati che

risiedono in posizioni diverse. La figura seguente mostra un confronto della previsione CTR di apprendimento approfondito per un runtime di workflow Spark.

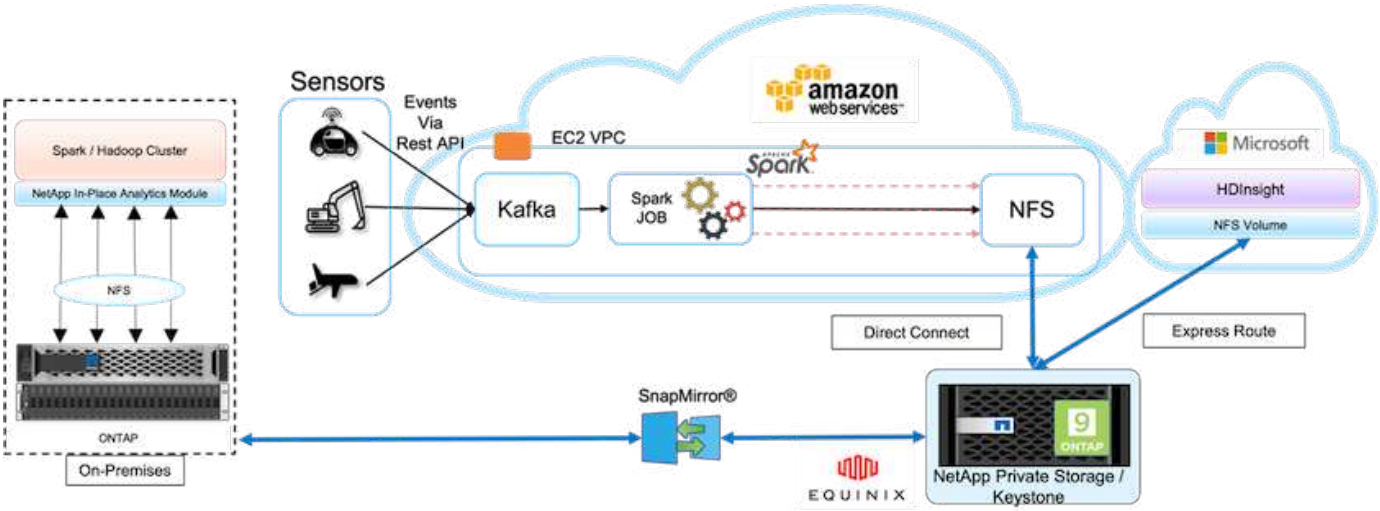


Soluzione di cloud ibrido

Un moderno data center aziendale è un cloud ibrido che connette più ambienti di infrastruttura distribuita attraverso un piano di gestione continua dei dati con un modello operativo coerente, on-premise e/o in più cloud pubblici. Per ottenere il massimo da un cloud ibrido, devi essere in grado di spostare perfettamente i dati tra ambienti on-premise e multi-cloud senza la necessità di conversioni di dati o refactoring delle applicazioni.

I clienti hanno indicato di iniziare il loro percorso nel cloud ibrido spostando lo storage secondario nel cloud per casi di utilizzo come la protezione dei dati o spostando meno carichi di lavoro business-critical come lo sviluppo delle applicazioni e DevOps nel cloud. Passano quindi a carichi di lavoro più critici. Hosting di contenuti e web, sviluppo di applicazioni e DevOps, database, analytics e applicazioni containerizzate sono tra i carichi di lavoro di cloud ibrido più diffusi. La complessità, i costi e i rischi dei progetti ai aziendali hanno storicamente ostacolato l'adozione dell'ai dalla fase sperimentale alla produzione.

Con una soluzione di cloud ibrido NetApp, i clienti possono beneficiare di strumenti integrati per la sicurezza, la governance dei dati e la conformità con un unico pannello di controllo per la gestione dei dati e del workflow in ambienti distribuiti, ottimizzando al contempo il costo totale di proprietà in base al consumo. La figura seguente è una soluzione di esempio di un partner di servizi cloud che ha il compito di fornire connettività multi-cloud per i dati di analisi dei big data dei clienti.



In questo scenario, i dati IoT ricevuti in AWS da diverse origini vengono memorizzati in una posizione centrale in NetApp Private Storage (NPS). Lo storage NPS è connesso ai cluster Spark o Hadoop situati in AWS e Azure, consentendo l'esecuzione di applicazioni di big data analytics in più cloud che accedono agli stessi dati. I requisiti e le sfide principali per questo caso di utilizzo includono:

- I clienti vogliono eseguire lavori di analisi sugli stessi dati utilizzando più cloud.
- I dati devono essere ricevuti da fonti diverse, ad esempio ambienti on-premise e cloud, attraverso diversi sensori e hub.
- La soluzione deve essere efficiente e conveniente.
- La sfida principale è quella di creare una soluzione conveniente ed efficiente in grado di offrire servizi di analisi ibridi tra diversi ambienti on-premise e cloud.

La nostra soluzione per la protezione dei dati e la connettività multicloud risolve il problema di avere applicazioni di analisi del cloud su più hyperscaler. Come mostrato nella figura precedente, i dati provenienti dai sensori vengono trasmessi e acquisiti nel cluster AWS Spark tramite Kafka. I dati vengono memorizzati in una condivisione NFS residente in NPS, che si trova all'esterno del cloud provider all'interno di un data center Equinix.

Poiché NetApp NPS è connesso ad Amazon AWS e Microsoft Azure rispettivamente tramite Direct Connect e Express Route Connections, i clienti possono sfruttare il modulo di analisi in-place per accedere ai dati da entrambi i cluster di analisi Amazon e AWS. Di conseguenza, poiché sia lo storage on-premise che NPS esegue il software ONTAP, "SnapMirror" Può eseguire il mirroring dei dati NPS nel cluster on-premise, fornendo analisi del cloud ibrido su cloud on-premise e multipli.

Per ottenere le migliori performance, NetApp consiglia di utilizzare più interfacce di rete e connessioni dirette o percorsi espressi per accedere ai dati dalle istanze cloud. Abbiamo altre soluzioni di data mover, tra cui "XCP" e. "Copia e sincronizzazione di BlueXP" Per aiutare i clienti a creare cluster Spark di cloud ibrido basati su applicazioni, sicuri e convenienti.

Script Python per ogni caso di utilizzo principale

I tre script Python riportati di seguito corrispondono ai tre principali casi di utilizzo testati. Il primo è `sentiment_analysis_sparknlp.py`.

```
# TR-4570 Refresh NLP testing by Rick Huang
```

```
from sys import argv
import os
import sparknlp
import pyspark.sql.functions as F
from sparknlp import Finisher
from pyspark.ml import Pipeline
from sparknlp.base import *
from sparknlp.annotator import *
from sparknlp.pretrained import PretrainedPipeline
from sparknlp import Finisher
# Start Spark Session with Spark NLP
spark = sparknlp.start()
print("Spark NLP version:")
print(sparknlp.version())
print("Apache Spark version:")
print(spark.version)
spark = sparknlp.SparkSession.builder \
     .master("yarn") \
     .appName("test_hdfs_read_write") \
     .config("spark.executor.cores", "1") \
     .config("spark.jars.packages", "com.johnsnowlabs.nlp:spark-
nlp_2.12:3.4.3")\
     .config('spark.executor.memory', '5gb') \
     .config('spark.executor.memoryOverhead','1000')\
     .config('spark.driver.memoryOverhead','1000')\
     .config("spark.sql.shuffle.partitions", "480")\
     .getOrCreate()
sc = spark.sparkContext
from pyspark.sql import SQLContext
sql = SQLContext(sc)
sqlContext = SQLContext(sc)
# Download pre-trained pipelines & sequence classifier
explain_pipeline_model = PretrainedPipeline('explain_document_dl',
lang='en').model#pipeline_sa =
PretrainedPipeline("classifierdl_bertwiki_finance_sentiment_pipeline",
lang="en")
# pipeline_finbert =
BertForSequenceClassification.loadSavedModel('/sparkusecase/bert_sequence_
classifier_finbert_en_3', spark)
sequenceClassifier = BertForSequenceClassification \
        .pretrained('bert_sequence_classifier_finbert', 'en') \
        .setInputCols(['token', 'document']) \
        .setOutputCol('class') \
        .setCaseSensitive(True) \
        .setMaxSentenceLength(512)
def process_sentence_df(data):
```

```
    # Pre-process: begin
    print("1. Begin DataFrame pre-processing...\n")
    print(f"\n\t2. Attaching DocumentAssembler Transformer to the
pipeline")
    documentAssembler = DocumentAssembler() \
        .setInputCol("text") \
        .setOutputCol("document") \
        .setCleanupMode("inplace_full")
        #.setCleanupMode("shrink", "inplace_full")
    doc_df = documentAssembler.transform(data)
    doc_df.printSchema()
    doc_df.show(truncate=50)
    # Pre-process: get rid of  blank lines
    clean_df = doc_df.withColumn("tmp", F.explode("document")) \
        .select("tmp.result").where("tmp.end !=
-1").withColumnRenamed("result", "text").dropna()
    print("[OK!] DataFrame after initial cleanup:\n")
    clean_df.printSchema()
    clean_df.show(truncate=80)
    # for FinBERT
    tokenizer = Tokenizer() \
        .setInputCols(['document']) \
        .setOutputCol('token')
    print(f"\n\t3. Attaching Tokenizer Annotator to the pipeline")
    pipeline_finbert = Pipeline(stages=[
        documentAssembler,
        tokenizer,
        sequenceClassifier
        ])
    # Use Finisher() & construct PySpark ML pipeline
    finisher = Finisher().setInputCols(["token", "lemma", "pos",
"entities"])
    print(f"\n\t4. Attaching Finisher Transformer to the pipeline")
    pipeline_ex = Pipeline() \
        .setStages([
           explain_pipeline_model,
           finisher
           ])
    print("\n\t\t\t ---- Pipeline Built Successfully ----")
    # Loading pipelines to annotate
    #result_ex_df = pipeline_ex.transform(clean_df)
    ex_model = pipeline_ex.fit(clean_df)
    annotations_finished_ex_df = ex_model.transform(clean_df)
    # result_sa_df = pipeline_sa.transform(clean_df)
    result_finbert_df = pipeline_finbert.fit(clean_df).transform(clean_df)
    print("\n\t\t\t ----Document Explain, Sentiment Analysis & FinBERT
```

```
Pipeline Fitted Successfully ----")
    # Check the result entities
    print("[OK!] Simple explain ML pipeline result:\n")
    annotations_finished_ex_df.printSchema()
    annotations_finished_ex_df.select('text',
'finished_entities').show(truncate=False)
    # Check the result sentiment from FinBERT
    print("[OK!] Sentiment Analysis FinBERT pipeline result:\n")
    result_finbert_df.printSchema()
    result_finbert_df.select('text', 'class.result').show(80, False)
    sentiment_stats(result_finbert_df)
    return
def sentiment_stats(finbert_df):
    result_df = finbert_df.select('text', 'class.result')
    sa_df = result_df.select('result')
    sa_df.groupBy('result').count().show()
    # total_lines = result_clean_df.count()
    # num_neutral = result_clean_df.where(result_clean_df.result ==
['neutral']).count()
    # num_positive = result_clean_df.where(result_clean_df.result ==
['positive']).count()
    # num_negative = result_clean_df.where(result_clean_df.result ==
['negative']).count()
    # print(f"\nRatio of neutral sentiment = {num_neutral/total_lines}")
    # print(f"Ratio of positive sentiment = {num_positive / total_lines}")
    # print(f"Ratio of negative sentiment = {num_negative /
total_lines}\n")
    return
def process_input_file(file_name):
    # Turn input file to Spark DataFrame
    print("START processing input file...")
    data_df = spark.read.text(file_name)
    data_df.show()
    # rename first column 'text' for sparknlp
    output_df = data_df.withColumnRenamed("value", "text").dropna()
    output_df.printSchema()
    return output_dfdef process_local_dir(directory):
    filelist = []
    for subdir, dirs, files in os.walk(directory):
        for filename in files:
            filepath = subdir + os.sep + filename
            print("[OK!] Will process the following files:")
            if filepath.endswith(".txt"):
                print(filepath)
                filelist.append(filepath)
    return filelist
```

```
def process_local_dir_or_file(dir_or_file):
    numfiles = 0
    if os.path.isfile(dir_or_file):
        input_df = process_input_file(dir_or_file)
        print("Obtained input_df.")
        process_sentence_df(input_df)
        print("Processed input_df")
        numfiles += 1
    else:
        filelist = process_local_dir(dir_or_file)
        for file in filelist:
            input_df = process_input_file(file)
            process_sentence_df(input_df)
            numfiles += 1
    return numfiles
def process_hdfs_dir(dir_name):
    # Turn input files to Spark DataFrame
    print("START processing input HDFS directory...")
    data_df = spark.read.option("recursiveFileLookup",
"true").text(dir_name)
    data_df.show()
    print("[DEBUG] total lines in data_df = ", data_df.count())
    # rename first column 'text' for sparknlp
    output_df = data_df.withColumnRenamed("value", "text").dropna()
    print("[DEBUG] output_df looks like: \n")
    output_df.show(40, False)
    print("[DEBUG] HDFS dir resulting data_df schema: \n")
    output_df.printSchema()
    process_sentence_df(output_df)
    print("Processed HDFS directory: ", dir_name)
    returnif __name__ == '__main__':
    try:
        if len(argv) == 2:
            print("Start processing input...\n")
    except:
        print("[ERROR] Please enter input text file or path to
process!\n")
        exit(1)
    # This is for local file, not hdfs:
    numfiles = process_local_dir_or_file(str(argv[1]))
    # For HDFS single file & directory:
    input_df = process_input_file(str(argv[1]))
    print("Obtained input_df.")
    process_sentence_df(input_df)
    print("Processed input_df")
    numfiles += 1
```

```
    # For HDFS directory of subdirectories of files:
    input_parse_list = str(argv[1]).split('/')
    print(input_parse_list)
    if input_parse_list[-2:-1] == ['Transcripts']:
        print("Start processing HDFS directory: ", str(argv[1]))
        process_hdfs_dir(str(argv[1]))
    print(f"[OK!] All done. Number of files processed = {numfiles}")
```

Il secondo script è `keras_spark_horovod_rossmann_estimator.py`.

```
# Copyright 2022 NetApp, Inc.
# Authored by Rick Huang
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
==========================================================================
====
# The below code was modified from: https://www.kaggle.com/c/rossmann-
store-sales
import argparse
import datetime
import os
import sys
from distutils.version import LooseVersion
import pyspark.sql.types as T
import pyspark.sql.functions as F
from pyspark import SparkConf, Row
from pyspark.sql import SparkSession
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.layers import Input, Embedding, Concatenate, Dense,
Flatten, Reshape, BatchNormalization, Dropout
import horovod.spark.keras as hvd
from horovod.spark.common.backend import SparkBackend
from horovod.spark.common.store import Store
```

```
from horovod.tensorflow.keras.callbacks import BestModelCheckpoint
parser = argparse.ArgumentParser(description='Horovod Keras Spark Rossmann
Estimator Example',

formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--master',
                    help='spark cluster to use for training. If set to
None, uses current default cluster. Cluster'
                         'should be set up to provide a Spark task per
multiple CPU cores, or per GPU, e.g. by'
                         'supplying `-c <NUM_GPUS>` in Spark Standalone
mode')
parser.add_argument('--num-proc', type=int,
                    help='number of worker processes for training,
default: `spark.default.parallelism`')
parser.add_argument('--learning_rate', type=float, default=0.0001,
                    help='initial learning rate')
parser.add_argument('--batch-size', type=int, default=100,
                    help='batch size')
parser.add_argument('--epochs', type=int, default=100,
                    help='number of epochs to train')
parser.add_argument('--sample-rate', type=float,
                    help='desired sampling rate. Useful to set to low
number (e.g. 0.01) to make sure that '
                         'end-to-end process works')
parser.add_argument('--data-dir', default='file://' + os.getcwd(),
                    help='location of data on local filesystem (prefixed
with file://) or on HDFS')
parser.add_argument('--local-submission-csv', default='submission.csv',
                    help='output submission predictions CSV')
parser.add_argument('--local-checkpoint-file', default='checkpoint',
                    help='model checkpoint')
parser.add_argument('--work-dir', default='/tmp',
                    help='temporary working directory to write
intermediate files (prefix with hdfs:// to use HDFS)')
if __name__ == '__main__':
    args = parser.parse_args()
    # ================ #
    # DATA PREPARATION #
    # ================ #
    print('================')
    print('Data preparation')
    print('================')
    # Create Spark session for data preparation.
    conf = SparkConf() \
        .setAppName('Keras Spark Rossmann Estimator Example') \
```

```
        .set('spark.sql.shuffle.partitions', '480') \
        .set("spark.executor.cores", "1") \
        .set('spark.executor.memory', '5gb') \
        .set('spark.executor.memoryOverhead','1000')\
        .set('spark.driver.memoryOverhead','1000')
    if args.master:
        conf.setMaster(args.master)
    elif args.num_proc:
        conf.setMaster('local[{}]'.format(args.num_proc))
    spark = SparkSession.builder.config(conf=conf).getOrCreate()
    train_csv = spark.read.csv('%s/train.csv' % args.data_dir,
header=True)
    test_csv = spark.read.csv('%s/test.csv' % args.data_dir, header=True)
    store_csv = spark.read.csv('%s/store.csv' % args.data_dir,
header=True)
    store_states_csv = spark.read.csv('%s/store_states.csv' %
args.data_dir, header=True)
    state_names_csv = spark.read.csv('%s/state_names.csv' % args.data_dir,
header=True)
    google_trend_csv = spark.read.csv('%s/googletrend.csv' %
args.data_dir, header=True)
    weather_csv = spark.read.csv('%s/weather.csv' % args.data_dir,
header=True)
    def expand_date(df):
        df = df.withColumn('Date', df.Date.cast(T.DateType()))
        return df \
            .withColumn('Year', F.year(df.Date)) \
            .withColumn('Month', F.month(df.Date)) \
            .withColumn('Week', F.weekofyear(df.Date)) \
            .withColumn('Day', F.dayofmonth(df.Date))
    def prepare_google_trend():
        # Extract week start date and state.
        google_trend_all = google_trend_csv \
            .withColumn('Date', F.regexp_extract(google_trend_csv.week,
'(.*?) -', 1)) \
            .withColumn('State', F.regexp_extract(google_trend_csv.file,
'Rossmann_DE_(.*)', 1))
        # Map state NI -> HB,NI to align with other data sources.
        google_trend_all = google_trend_all \
            .withColumn('State', F.when(google_trend_all.State == 'NI',
'HB,NI').otherwise(google_trend_all.State))
        # Expand dates.
        return expand_date(google_trend_all)
    def add_elapsed(df, cols):
        def add_elapsed_column(col, asc):
            def fn(rows):
```

```
                last_store, last_date = None, None
                for r in rows:
                    if last_store != r.Store:
                        last_store = r.Store
                        last_date = r.Date
                    if r[col]:
                        last_date = r.Date
                    fields = r.asDict().copy()
                    fields[('After' if asc else 'Before') + col] = (r.Date
- last_date).days
                    yield Row(**fields)
            return fn
        df = df.repartition(df.Store)
        for asc in [False, True]:
            sort_col = df.Date.asc() if asc else df.Date.desc()
            rdd = df.sortWithinPartitions(df.Store.asc(), sort_col).rdd
            for col in cols:
                rdd = rdd.mapPartitions(add_elapsed_column(col, asc))
            df = rdd.toDF()
        return df
    def prepare_df(df):
        num_rows = df.count()
        # Expand dates.
        df = expand_date(df)
        df = df \
            .withColumn('Open', df.Open != '0') \
            .withColumn('Promo', df.Promo != '0') \
            .withColumn('StateHoliday', df.StateHoliday != '0') \
            .withColumn('SchoolHoliday', df.SchoolHoliday != '0')
        # Merge in store information.
        store = store_csv.join(store_states_csv, 'Store')
        df = df.join(store, 'Store')
        # Merge in Google Trend information.
        google_trend_all = prepare_google_trend()
        df = df.join(google_trend_all, ['State', 'Year',
'Week']).select(df['*'], google_trend_all.trend)
        # Merge in Google Trend for whole Germany.
        google_trend_de = google_trend_all[google_trend_all.file ==
'Rossmann_DE'].withColumnRenamed('trend', 'trend_de')
        df = df.join(google_trend_de, ['Year', 'Week']).select(df['*'],
google_trend_de.trend_de)
        # Merge in weather.
        weather = weather_csv.join(state_names_csv, weather_csv.file ==
state_names_csv.StateName)
        df = df.join(weather, ['State', 'Date'])
        # Fix null values.
```

```
        df = df \
            .withColumn('CompetitionOpenSinceYear',
F.coalesce(df.CompetitionOpenSinceYear, F.lit(1900))) \
            .withColumn('CompetitionOpenSinceMonth',
F.coalesce(df.CompetitionOpenSinceMonth, F.lit(1))) \
            .withColumn('Promo2SinceYear', F.coalesce(df.Promo2SinceYear,
F.lit(1900))) \
            .withColumn('Promo2SinceWeek', F.coalesce(df.Promo2SinceWeek,
F.lit(1)))
        # Days & months competition was open, cap to 2 years.
        df = df.withColumn('CompetitionOpenSince',
                           F.to_date(F.format_string('%s-%s-15',
df.CompetitionOpenSinceYear,

df.CompetitionOpenSinceMonth)))
        df = df.withColumn('CompetitionDaysOpen',
                           F.when(df.CompetitionOpenSinceYear > 1900,
                                  F.greatest(F.lit(0), F.least(F.lit(360 *
2), F.datediff(df.Date, df.CompetitionOpenSince))))
                           .otherwise(0))
        df = df.withColumn('CompetitionMonthsOpen',
(df.CompetitionDaysOpen / 30).cast(T.IntegerType()))
        # Days & weeks of promotion, cap to 25 weeks.
        df = df.withColumn('Promo2Since',
                           F.expr('date_add(format_string("%s-01-01",
Promo2SinceYear), (cast(Promo2SinceWeek as int) - 1) * 7)'))
        df = df.withColumn('Promo2Days',
                           F.when(df.Promo2SinceYear > 1900,
                                  F.greatest(F.lit(0), F.least(F.lit(25 *
7), F.datediff(df.Date, df.Promo2Since))))
                           .otherwise(0))
        df = df.withColumn('Promo2Weeks', (df.Promo2Days /
7).cast(T.IntegerType()))
        # Check that we did not lose any rows through inner joins.
        assert num_rows == df.count(), 'lost rows in joins'
        return df
    def build_vocabulary(df, cols):
        vocab = {}
        for col in cols:
            values = [r[0] for r in df.select(col).distinct().collect()]
            col_type = type([x for x in values if x is not None][0])
            default_value = col_type()
            vocab[col] = sorted(values, key=lambda x: x or default_value)
        return vocab
    def cast_columns(df, cols):
        for col in cols:
```

```
            df = df.withColumn(col,
F.coalesce(df[col].cast(T.FloatType()), F.lit(0.0)))
        return df
    def lookup_columns(df, vocab):
        def lookup(mapping):
            def fn(v):
                return mapping.index(v)
            return F.udf(fn, returnType=T.IntegerType())
        for col, mapping in vocab.items():
            df = df.withColumn(col, lookup(mapping)(df[col]))
        return df
    if args.sample_rate:
        train_csv = train_csv.sample(withReplacement=False,
fraction=args.sample_rate)
        test_csv = test_csv.sample(withReplacement=False,
fraction=args.sample_rate)
    # Prepare data frames from CSV files.
    train_df = prepare_df(train_csv).cache()
    test_df = prepare_df(test_csv).cache()
    # Add elapsed times from holidays & promos, the data spanning training
& test datasets.
    elapsed_cols = ['Promo', 'StateHoliday', 'SchoolHoliday']
    elapsed = add_elapsed(train_df.select('Date', 'Store', *elapsed_cols)
                          .unionAll(test_df.select('Date', 'Store',
*elapsed_cols)),
                          elapsed_cols)
    # Join with elapsed times.
    train_df = train_df \
        .join(elapsed, ['Date', 'Store']) \
        .select(train_df['*'], *[prefix + col for prefix in ['Before',
'After'] for col in elapsed_cols])
    test_df = test_df \
        .join(elapsed, ['Date', 'Store']) \
        .select(test_df['*'], *[prefix + col for prefix in ['Before',
'After'] for col in elapsed_cols])
    # Filter out zero sales.
    train_df = train_df.filter(train_df.Sales > 0)
    print('===================')
    print('Prepared data frame')
    print('===================')
    train_df.show()
    categorical_cols = [
        'Store', 'State', 'DayOfWeek', 'Year', 'Month', 'Day', 'Week',
'CompetitionMonthsOpen', 'Promo2Weeks', 'StoreType',
        'Assortment', 'PromoInterval', 'CompetitionOpenSinceYear',
'Promo2SinceYear', 'Events', 'Promo',
```

```
        'StateHoliday', 'SchoolHoliday'
    ]
    continuous_cols = [
        'CompetitionDistance', 'Max_TemperatureC', 'Mean_TemperatureC',
'Min_TemperatureC', 'Max_Humidity',
        'Mean_Humidity', 'Min_Humidity', 'Max_Wind_SpeedKm_h',
'Mean_Wind_SpeedKm_h', 'CloudCover', 'trend', 'trend_de',
        'BeforePromo', 'AfterPromo', 'AfterStateHoliday',
'BeforeStateHoliday', 'BeforeSchoolHoliday', 'AfterSchoolHoliday'
    ]
    all_cols = categorical_cols + continuous_cols
    # Select features.
    train_df = train_df.select(*(all_cols + ['Sales', 'Date'])).cache()
    test_df = test_df.select(*(all_cols + ['Id', 'Date'])).cache()
    # Build vocabulary of categorical columns.
    vocab = build_vocabulary(train_df.select(*categorical_cols)

.unionAll(test_df.select(*categorical_cols)).cache(),
                              categorical_cols)
    # Cast continuous columns to float & lookup categorical columns.
    train_df = cast_columns(train_df, continuous_cols + ['Sales'])
    train_df = lookup_columns(train_df, vocab)
    test_df = cast_columns(test_df, continuous_cols)
    test_df = lookup_columns(test_df, vocab)
    # Split into training & validation.
    # Test set is in 2015, use the same period in 2014 from the training
set as a validation set.
    test_min_date = test_df.agg(F.min(test_df.Date)).collect()[0][0]
    test_max_date = test_df.agg(F.max(test_df.Date)).collect()[0][0]
    one_year = datetime.timedelta(365)
    train_df = train_df.withColumn('Validation',
                                   (train_df.Date > test_min_date -
one_year) & (train_df.Date <= test_max_date - one_year))
    # Determine max Sales number.
    max_sales = train_df.agg(F.max(train_df.Sales)).collect()[0][0]
    # Convert Sales to log domain
    train_df = train_df.withColumn('Sales', F.log(train_df.Sales))
    print('====================================')
    print('Data frame with transformed columns')
    print('====================================')
    train_df.show()
    print('================')
    print('Data frame sizes')
    print('================')
    train_rows = train_df.filter(~train_df.Validation).count()
    val_rows = train_df.filter(train_df.Validation).count()
```

```
    test_rows = test_df.count()
    print('Training: %d' % train_rows)
    print('Validation: %d' % val_rows)
    print('Test: %d' % test_rows)
    # ============== #
    # MODEL TRAINING #
    # ============== #
    print('==============')
    print('Model training')
    print('==============')
    def exp_rmspe(y_true, y_pred):
        """Competition evaluation metric, expects logarithic inputs."""
        pct = tf.square((tf.exp(y_true) - tf.exp(y_pred)) /
tf.exp(y_true))
        # Compute mean excluding stores with zero denominator.
        x = tf.reduce_sum(tf.where(y_true > 0.001, pct,
tf.zeros_like(pct)))
        y = tf.reduce_sum(tf.where(y_true > 0.001, tf.ones_like(pct),
tf.zeros_like(pct)))
        return tf.sqrt(x / y)
    def act_sigmoid_scaled(x):
        """Sigmoid scaled to logarithm of maximum sales scaled by 20%."""
        return tf.nn.sigmoid(x) * tf.math.log(max_sales) * 1.2
    CUSTOM_OBJECTS = {'exp_rmspe': exp_rmspe,
                      'act_sigmoid_scaled': act_sigmoid_scaled}
    # Disable GPUs when building the model to prevent memory leaks
    if LooseVersion(tf.__version__) >= LooseVersion('2.0.0'):
        # See https://github.com/tensorflow/tensorflow/issues/33168
        os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
    else:

K.set_session(tf.Session(config=tf.ConfigProto(device_count={'GPU': 0})))
    # Build the model.
    inputs = {col: Input(shape=(1,), name=col) for col in all_cols}
    embeddings = [Embedding(len(vocab[col]), 10, input_length=1,
name='emb_' + col)(inputs[col])
                  for col in categorical_cols]
    continuous_bn = Concatenate()([Reshape((1, 1), name='reshape_' +
col)(inputs[col])
                                   for col in continuous_cols])
    continuous_bn = BatchNormalization()(continuous_bn)
    x = Concatenate()(embeddings + [continuous_bn])
    x = Flatten()(x)
    x = Dense(1000, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.00005))(x)
    x = Dense(1000, activation='relu',
```

```
kernel_regularizer=tf.keras.regularizers.l2(0.00005))(x)
    x = Dense(1000, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.00005))(x)
    x = Dense(500, activation='relu',
kernel_regularizer=tf.keras.regularizers.l2(0.00005))(x)
    x = Dropout(0.5)(x)
    output = Dense(1, activation=act_sigmoid_scaled)(x)
    model = tf.keras.Model([inputs[f] for f in all_cols], output)
    model.summary()
    opt = tf.keras.optimizers.Adam(lr=args.learning_rate, epsilon=1e-3)
    # Checkpoint callback to specify options for the returned Keras model
    ckpt_callback = BestModelCheckpoint(monitor='val_loss', mode='auto',
save_freq='epoch')
    # Horovod: run training.
    store = Store.create(args.work_dir)
    backend = SparkBackend(num_proc=args.num_proc,
                           stdout=sys.stdout, stderr=sys.stderr,
                           prefix_output_with_timestamp=True)
    keras_estimator = hvd.KerasEstimator(backend=backend,
                                         store=store,
                                         model=model,
                                         optimizer=opt,
                                         loss='mae',
                                         metrics=[exp_rmspe],
                                         custom_objects=CUSTOM_OBJECTS,
                                         feature_cols=all_cols,
                                         label_cols=['Sales'],
                                         validation='Validation',
                                         batch_size=args.batch_size,
                                         epochs=args.epochs,
                                         verbose=2,

checkpoint_callback=ckpt_callback)
    keras_model =
keras_estimator.fit(train_df).setOutputCols(['Sales_output'])
    history = keras_model.getHistory()
    best_val_rmspe = min(history['val_exp_rmspe'])
    print('Best RMSPE: %f' % best_val_rmspe)
    # Save the trained model.
    keras_model.save(args.local_checkpoint_file)
    print('Written checkpoint to %s' % args.local_checkpoint_file)
    # ================ #
    # FINAL PREDICTION #
    # ================ #
    print('================')
    print('Final prediction')
```

```
    print('================')
    pred_df=keras_model.transform(test_df)
    pred_df.printSchema()
    pred_df.show(5)
    # Convert from log domain to real Sales numbers
    pred_df=pred_df.withColumn('Sales_pred', F.exp(pred_df.Sales_output))
    submission_df = pred_df.select(pred_df.Id.cast(T.IntegerType()),
pred_df.Sales_pred).toPandas()
    submission_df.sort_values(by=['Id']).to_csv(args.local_submission_csv,
index=False)
    print('Saved predictions to %s' % args.local_submission_csv)
    spark.stop()
```

Il terzo script è `run_classification_criteo_spark.py`.

```
import tempfile, string, random, os, uuid
import argparse, datetime, sys, shutil
import csv
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext, Row, DataFrame
from pyspark.mllib import linalg as mllib_linalg
from pyspark.mllib.linalg import SparseVector as mllibSparseVector
from pyspark.mllib.linalg import VectorUDT as mllibVectorUDT
from pyspark.mllib.linalg import Vector as mllibVector, Vectors as
mllibVectors
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.ml import linalg as ml_linalg
from pyspark.ml.linalg import VectorUDT as mlVectorUDT
from pyspark.ml.linalg import SparseVector as mlSparseVector
from pyspark.ml.linalg import Vector as mlVector, Vectors as mlVectors
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import OneHotEncoder
from math import log
from math import exp  # exp(-t) = e^-t
from operator import add
from pyspark.sql.functions import udf, split, lit
from pyspark.sql.functions import size, sum as sqlsum
import pyspark.sql.functions as F
import pyspark.sql.types as T
from pyspark.sql.types import ArrayType, StructType, StructField,
LongType, StringType, IntegerType, FloatType
```

```
from pyspark.sql.functions import explode, col, log, when
from collections import defaultdict
import pandas as pd
import pyspark.pandas as ps
from sklearn.metrics import log_loss, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from deepctr.models import DeepFM
from deepctr.feature_column import SparseFeat, DenseFeat,
get_feature_names
spark = SparkSession.builder \
    .master("yarn") \
    .appName("deep_ctr_classification") \
    .config("spark.jars.packages", "io.github.ravwojdyla:spark-schema-
utils_2.12:0.1.0") \
    .config("spark.executor.cores", "1") \
    .config('spark.executor.memory', '5gb') \
    .config('spark.executor.memoryOverhead', '1500') \
    .config('spark.driver.memoryOverhead', '1500') \
    .config("spark.sql.shuffle.partitions", "480") \
    .config("spark.sql.execution.arrow.enabled", "true") \
    .config("spark.driver.maxResultSize", "50gb") \
    .getOrCreate()
# spark.conf.set("spark.sql.execution.arrow.enabled", "true") # deprecated
print("Apache Spark version:")
print(spark.version)
sc = spark.sparkContext
sqlContext = SQLContext(sc)
parser = argparse.ArgumentParser(description='Spark DCN CTR Prediction
Example',

formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('--data-dir', default='file://' + os.getcwd(),
                    help='location of data on local filesystem (prefixed
with file://) or on HDFS')
def process_input_file(file_name, sparse_feat, dense_feat):
    # Need this preprocessing to turn Criteo raw file into CSV:
    print("START processing input file...")
    # only convert the file ONCE
    # sample = open(file_name)
    # sample = '\n'.join([str(x.replace('\n', '').replace('\t', ',')) for
x in sample])
    # # Add header in data file and save as CSV
    # header = ','.join(str(x) for x in (['label'] + dense_feat +
sparse_feat))
    # with open('/sparkdemo/tr-4570-data/ctr_train.csv', mode='w',
```

```
encoding="utf-8") as f:
    #     f.write(header + '\n' + sample)
    #     f.close()
    # print("Raw training file processed and saved as CSV: ", f.name)
    raw_df = sqlContext.read.option("header", True).csv(file_name)
    raw_df.show(5, False)
    raw_df.printSchema()
    # convert columns I1 to I13 from string to integers
    conv_df = raw_df.select(col('label').cast("double"),
                            *(col(i).cast("float").alias(i) for i in
raw_df.columns if i in dense_feat),
                            *(col(c) for c in raw_df.columns if c in
sparse_feat))
    print("Schema of raw_df with integer columns type changed:")
    conv_df.printSchema()
    # result_pdf = conv_df.select("*").toPandas()
    tmp_df = conv_df.na.fill(0, dense_feat)
    result_df = tmp_df.na.fill('-1', sparse_feat)
    result_df.show()
    return result_df
if __name__ == "__main__":
    args = parser.parse_args()
    # Pandas read CSV
    # data = pd.read_csv('%s/criteo_sample.txt' % args.data_dir)
    # print("Obtained Pandas df.")
    dense_features = ['I' + str(i) for i in range(1, 14)]
    sparse_features = ['C' + str(i) for i in range(1, 27)]
    # Spark read CSV
    # process_input_file('%s/train.txt' % args.data_dir, sparse_features,
dense_features) # run only ONCE
    spark_df = process_input_file('%s/data.txt' % args.data_dir,
sparse_features, dense_features) # sample data
    # spark_df = process_input_file('%s/ctr_train.csv' % args.data_dir,
sparse_features, dense_features)
    print("Obtained Spark df and filled in missing features.")
    data = spark_df
    # Pandas
    #data[sparse_features] = data[sparse_features].fillna('-1', )
    #data[dense_features] = data[dense_features].fillna(0, )
    target = ['label']
    label_npa = data.select("label").toPandas().to_numpy()
    print("label numPy array has length = ", len(label_npa)) # 45,840,617
w/ 11GB dataset
    label_npa.ravel()
    label_npa.reshape(len(label_npa), )
    # 1.Label Encoding for sparse features,and do simple Transformation
```

```
for dense features
    print("Before LabelEncoder():")
    data.printSchema()  # label: float (nullable = true)
    for feat in sparse_features:
        lbe = LabelEncoder()
        tmp_pdf = data.select(feat).toPandas().to_numpy()
        tmp_ndarray = lbe.fit_transform(tmp_pdf)
        print("After LabelEncoder(), tmp_ndarray[0] =", tmp_ndarray[0])
        # print("Data tmp PDF after lbe transformation, the output ndarray
has length = ", len(tmp_ndarray)) # 45,840,617 for 11GB dataset
        tmp_ndarray.ravel()
        tmp_ndarray.reshape(len(tmp_ndarray), )
        out_ndarray = np.column_stack([label_npa, tmp_ndarray])
        pdf = pd.DataFrame(out_ndarray, columns=['label', feat])
        s_df = spark.createDataFrame(pdf)
        s_df.printSchema() # label: double (nullable = true)
        print("Before joining data df with s_df, s_df example rows:")
        s_df.show(1, False)
        data = data.drop(feat).join(s_df, 'label').drop('label')
        print("After LabelEncoder(), data df example rows:")
        data.show(1, False)
        print("Finished processing sparse_features: ", feat)
    print("Data DF after label encoding: ")
    data.show()
    data.printSchema()
    mms = MinMaxScaler(feature_range=(0, 1))
    # data[dense_features] = mms.fit_transform(data[dense_features]) # for
Pandas df
    tmp_pdf = data.select(dense_features).toPandas().to_numpy()
    tmp_ndarray = mms.fit_transform(tmp_pdf)
    tmp_ndarray.ravel()
    tmp_ndarray.reshape(len(tmp_ndarray), len(tmp_ndarray[0]))
    out_ndarray = np.column_stack([label_npa, tmp_ndarray])
    pdf = pd.DataFrame(out_ndarray, columns=['label'] + dense_features)
    s_df = spark.createDataFrame(pdf)
    s_df.printSchema()
    data.drop(*dense_features).join(s_df, 'label').drop('label')
    print("Finished processing dense_features: ", dense_features)
    print("Data DF after MinMaxScaler: ")
    data.show()

    # 2.count #unique features for each sparse field,and record dense
feature field name
    fixlen_feature_columns = [SparseFeat(feat,
vocabulary_size=data.select(feat).distinct().count() + 1, embedding_dim=4)
                              for i, feat in enumerate(sparse_features)] +
```

```
\
                            [DenseFeat(feat, 1, ) for feat in
dense_features]
    dnn_feature_columns = fixlen_feature_columns
    linear_feature_columns = fixlen_feature_columns
    feature_names = get_feature_names(linear_feature_columns +
dnn_feature_columns)
    # 3.generate input data for model
    # train, test = train_test_split(data.toPandas(), test_size=0.2,
random_state=2020) # Pandas; might hang for 11GB data
    train, test = data.randomSplit(weights=[0.8, 0.2], seed=200)
    print("Training dataset size = ", train.count())
    print("Testing dataset size = ", test.count())
    # Pandas:
    # train_model_input = {name: train[name] for name in feature_names}
    # test_model_input = {name: test[name] for name in feature_names}
    # Spark DF:
    train_model_input = {}
    test_model_input = {}
    for name in feature_names:
        if name.startswith('I'):
            tr_pdf = train.select(name).toPandas()
            train_model_input[name] = pd.to_numeric(tr_pdf[name])
            ts_pdf = test.select(name).toPandas()
            test_model_input[name] = pd.to_numeric(ts_pdf[name])
    # 4.Define Model,train,predict and evaluate
    model = DeepFM(linear_feature_columns, dnn_feature_columns,
task='binary')
    model.compile("adam", "binary_crossentropy",
                  metrics=['binary_crossentropy'], )
    lb_pdf = train.select(target).toPandas()
    history = model.fit(train_model_input,
pd.to_numeric(lb_pdf['label']).values,
                        batch_size=256, epochs=10, verbose=2,
validation_split=0.2, )
    pred_ans = model.predict(test_model_input, batch_size=256)
    print("test LogLoss",
round(log_loss(pd.to_numeric(test.select(target).toPandas()).values,
pred_ans), 4))
    print("test AUC",
round(roc_auc_score(pd.to_numeric(test.select(target).toPandas()).values,
pred_ans), 4))
```

Conclusione

In questo documento, discutiamo dell'architettura di Apache Spark, dei casi di utilizzo dei clienti e del portfolio di storage NetApp in relazione a big data, analytics moderni e ai, ML e DL. Nei nostri test di convalida delle performance basati su strumenti di benchmarking standard di settore e sulla domanda dei clienti, le soluzioni NetApp Spark hanno dimostrato performance superiori rispetto ai sistemi Hadoop nativi. Una combinazione dei casi di utilizzo dei clienti e dei risultati delle performance presentati in questo report può aiutarti a scegliere una soluzione Spark appropriata per la tua implementazione.

Dove trovare ulteriori informazioni

In questo TR sono stati utilizzati i seguenti riferimenti:

- Architettura e componenti di Apache Spark

  "http://spark.apache.org/docs/latest/cluster-overview.html"

- Casi di utilizzo di Apache Spark

  "https://www.qubole.com/blog/big-data/apache-spark-use-cases/"

- Le sfide di Apache

  "http://www.infoworld.com/article/2897287/big-data/5-reasons-to-turn-to-spark-for-big-data-analytics.html"

- NLP. Scintilla

  "https://www.johnsnowlabs.com/spark-nlp/"

- BERT

  "https://arxiv.org/abs/1810.04805"

- Deep and Cross Network for ad Click Predictions

  "https://arxiv.org/abs/1708.05123"

- FlexGroup

  "http://www.netapp.com/us/media/tr-4557.pdf"

- ETL streaming

  "https://www.infoq.com/articles/apache-spark-streaming"

- Soluzioni NetApp e-Series per Hadoop

  "https://www.netapp.com/media/16420-tr-3969.pdf"

- Analisi del sentimento da Customer Communications con NetApp ai

  "https://docs.netapp.com/us-en/netapp-solutions/pdfs/sidebar/Sentiment_analysis_with_NetApp_AI.pdf"

- Soluzioni NetApp per l'analisi dei dati moderna

  "https://docs.netapp.com/us-en/netapp-solutions/data-analytics/index.html"

- SnapMirror

  "https://docs.netapp.com/us-en/ontap/data-protection/snapmirror-replication-concept.html"

- XCP

  https://mysupport.netapp.com/documentation/docweb/index.html?productID=63942&language=en-US

- Copia e sincronizzazione di BlueXP

  "https://cloud.netapp.com/cloud-sync-service"

- Toolkit DataOps

  "https://github.com/NetApp/netapp-dataops-toolkit"

Analisi dei big data dati per l'intelligenza artificiale

TR-4732: Dai dati di analisi dei big data all'intelligenza artificiale

Karthikeyan Nagalingam, NetApp

Questo documento descrive come spostare i dati di analisi dei big data e i dati HPC nell'ai. L'ai elabora i dati NFS attraverso le esportazioni NFS, mentre i clienti spesso dispongono dei propri dati ai in una piattaforma di analisi dei big data, come lo storage HDFS, Blob o S3, oltre a piattaforme HPC come GPFS. Questo documento fornisce linee guida per lo spostamento dei dati di analisi dei big data e dei dati HPC nell'ai utilizzando NetApp XCP e NIPAM. Discutiamo inoltre dei vantaggi per il business derivanti dal passaggio dei dati da big data e HPC all'ai.

Concetti e componenti

Storage per l'analisi dei big data

L'analisi dei big data è il principale provider di storage per HDFS. Un cliente utilizza spesso un file system compatibile con Hadoop (HCFS) come Windows Azure Blob Storage, MapR file System (MapR-FS) e lo storage a oggetti S3.

File system parallelo generale

Il GPFS di IBM è un file system aziendale che offre un'alternativa a HDFS. LE GPF offrono alle applicazioni la flessibilità necessaria per decidere le dimensioni dei blocchi e il layout di replica, garantendo buone performance ed efficienza.

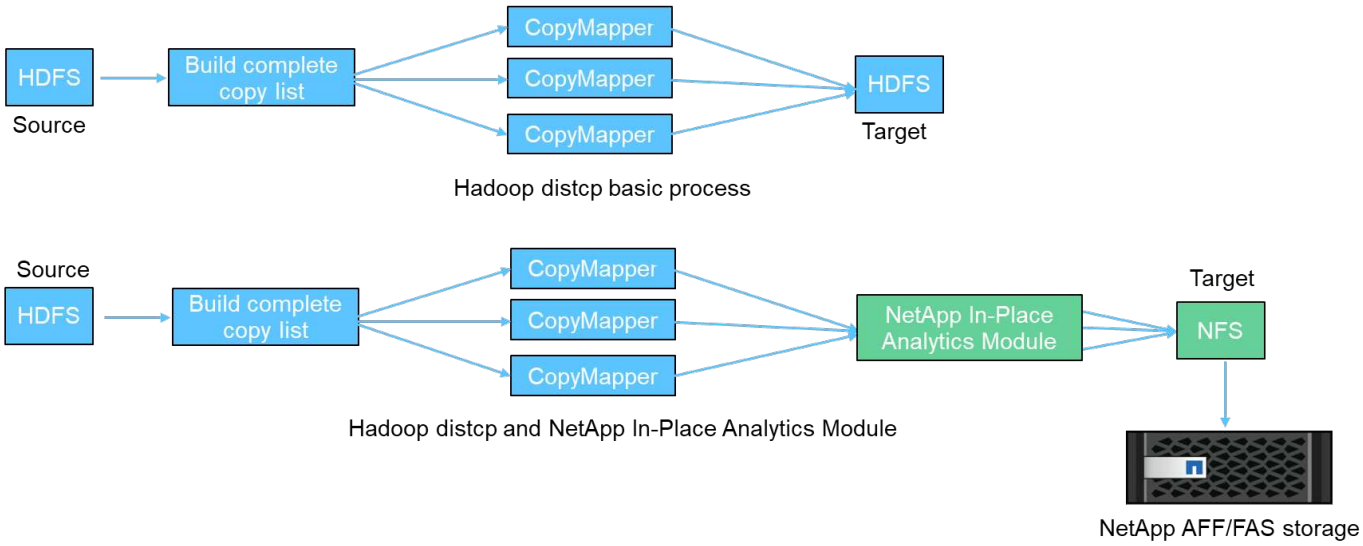
Modulo NetApp in-place Analytics

Il NetApp in-place Analytics Module (NIPAM) funge da driver per i cluster Hadoop per accedere ai dati NFS. Ha quattro componenti: Un pool di connessioni, un NFS InputStream, una cache di handle di file e un NFS

OutputStream. Per ulteriori informazioni, vedere "TR-4382: Modulo NetApp in-place Analytics."

Copia distribuita Hadoop

La copia distribuita di Hadoop (DistCp) è uno strumento di copia distribuita utilizzato per attività di coping tra cluster e intra-cluster di grandi dimensioni. Questo strumento utilizza MapReduce per la distribuzione dei dati, la gestione degli errori e il reporting. Espande l'elenco di file e directory e li inserisce per mappare le attività per copiare i dati dall'elenco di origine. L'immagine seguente mostra l'operazione DistCp in HDFS e non in HDFS.



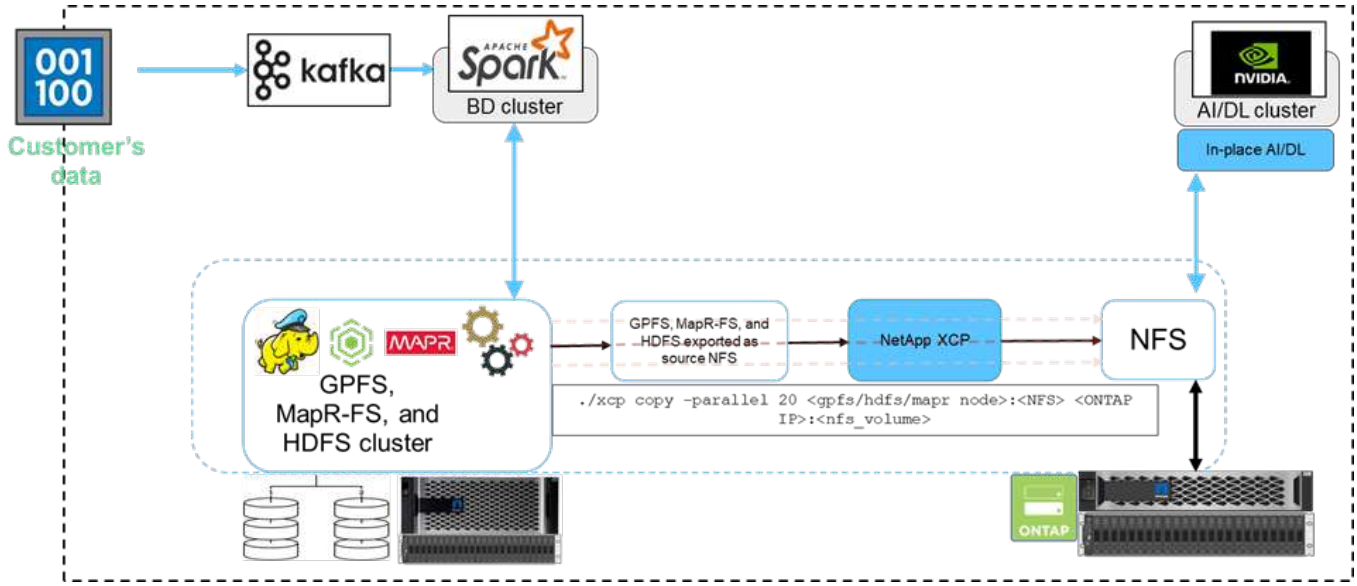
Hadoop DistCp sposta i dati tra i due sistemi HDFS senza utilizzare un driver aggiuntivo. NetApp fornisce il driver per i sistemi non HDFS. Per una destinazione NFS, NIPAM fornisce il driver per copiare i dati utilizzati da Hadoop DistCp per comunicare con le destinazioni NFS durante la copia dei dati.

NetApp Cloud Volumes Service

NetApp Cloud Volumes Service è un file service nativo del cloud con performance estreme. Questo servizio aiuta i clienti ad accelerare il time-to-market, aumentando e diminuendo rapidamente le risorse e utilizzando le funzionalità NetApp per migliorare la produttività e ridurre i tempi di inattività del personale. Cloud Volumes Service è la giusta alternativa per il disaster recovery e il backup nel cloud, in quanto riduce l'impatto complessivo del data center e consuma meno storage di cloud pubblico nativo.

XCP di NetApp

NetApp XCP è un software client che consente una migrazione dei dati rapida e affidabile da qualsiasi a NetApp e da NetApp a NetApp. Questo tool è progettato per copiare una grande quantità di dati NAS non strutturati da qualsiasi sistema NAS a un controller di storage NetApp. XCP Migration Tool utilizza un motore di streaming i/o multicore e multicanale in grado di elaborare molte richieste in parallelo, ad esempio migrazione dei dati, elenchi di file o directory e report di spazio. Questo è il tool di migrazione dei dati NetApp predefinito. È possibile utilizzare XCP per copiare i dati da un cluster Hadoop e HPC allo storage NetApp NFS. Il diagramma seguente mostra il trasferimento dei dati da un cluster Hadoop e HPC a un volume NetApp NFS utilizzando XCP.



Copia e sincronizzazione di NetApp BlueXP

NetApp BlueXP Copy and Sync è un software-as-a-service di replica dei dati ibridi che trasferisce e sincronizza i dati NFS, S3 e CIFS in modo perfetto e sicuro tra storage on-premise e cloud storage. Questo software viene utilizzato per la migrazione dei dati, l'archiviazione, la collaborazione, l'analisi e altro ancora. Una volta trasferiti i dati, BlueXP Copy e Sync sincronizza costantemente i dati tra origine e destinazione. In futuro, trasferisce il delta. Inoltre, protegge i dati all'interno della tua rete, nel cloud o on-premise. Questo software si basa su un modello pay-as-you-go, che fornisce una soluzione conveniente e offre funzionalità di monitoraggio e reporting per il trasferimento dei dati.

Sfide per i clienti

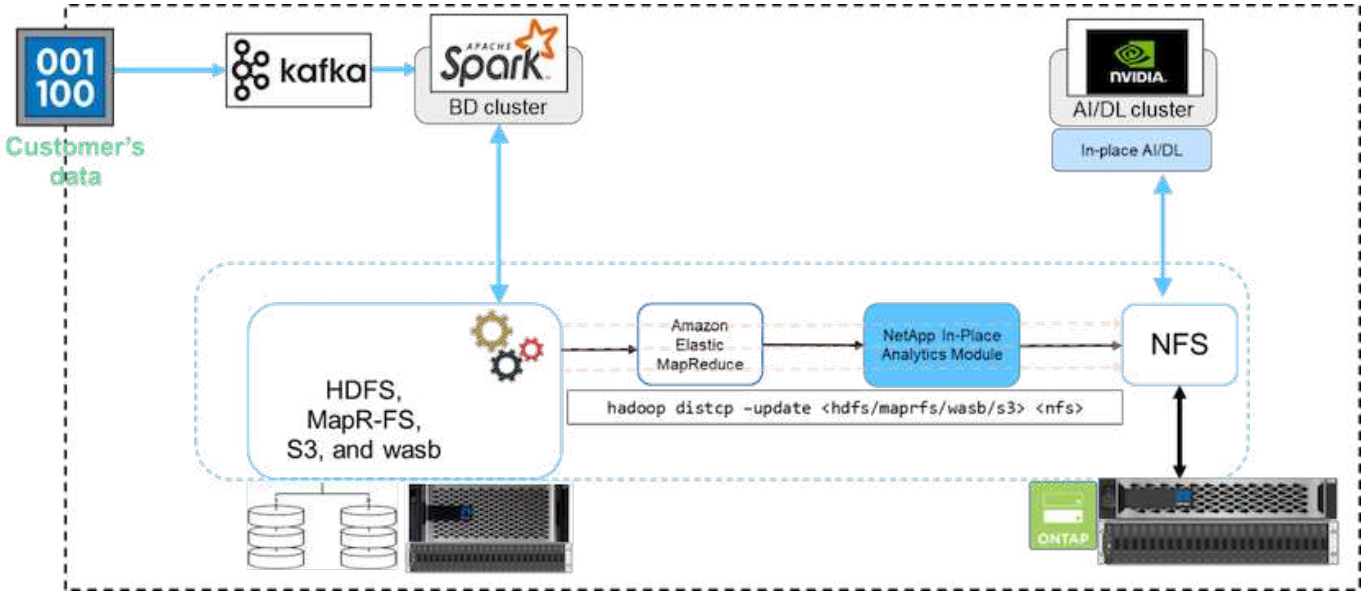
I clienti potrebbero affrontare le seguenti sfide quando tentano di accedere ai dati dalle analisi dei big data per le operazioni di ai:

- I dati dei clienti si trovano in un repository di data Lake. Il data Lake può contenere diversi tipi di dati, ad esempio dati strutturati, non strutturati, semistrutturati, log e dati machine-to-machine. Tutti questi tipi di dati devono essere elaborati nei sistemi ai.
- Ai non è compatibile con i file system Hadoop. Un'architettura ai tipica non è in grado di accedere direttamente ai dati HDFS e HCFS, che devono essere spostati in un file system ai-understandable (NFS).
- Il trasferimento dei dati del data Lake all'ai richiede in genere processi specializzati. La quantità di dati nel data Lake può essere molto grande. Un cliente deve disporre di un modo efficiente, ad alto throughput e conveniente per trasferire i dati nei sistemi ai.
- Sincronizzazione dei dati. Se un cliente desidera sincronizzare i dati tra la piattaforma per big data e l'ai, a volte i dati elaborati tramite l'ai possono essere utilizzati con i big data per l'elaborazione analitica.

Soluzione per il data mover

In un cluster di big data, i dati vengono memorizzati in HDFS o HCFS, come MapR-FS, Windows Azure Storage Blob, S3 o il file system Google. Abbiamo eseguito test con HDFS, MapR-FS e S3 come origine per copiare i dati nell'esportazione NFS di NetApp ONTAP con l'aiuto di NIPAM utilizzando `hadoop distcp` comando dall'origine.

Il seguente diagramma illustra il tipico spostamento dei dati da un cluster Spark in esecuzione con storage HDFS a un volume NFS NetApp ONTAP in modo che NVIDIA possa elaborare le operazioni ai.



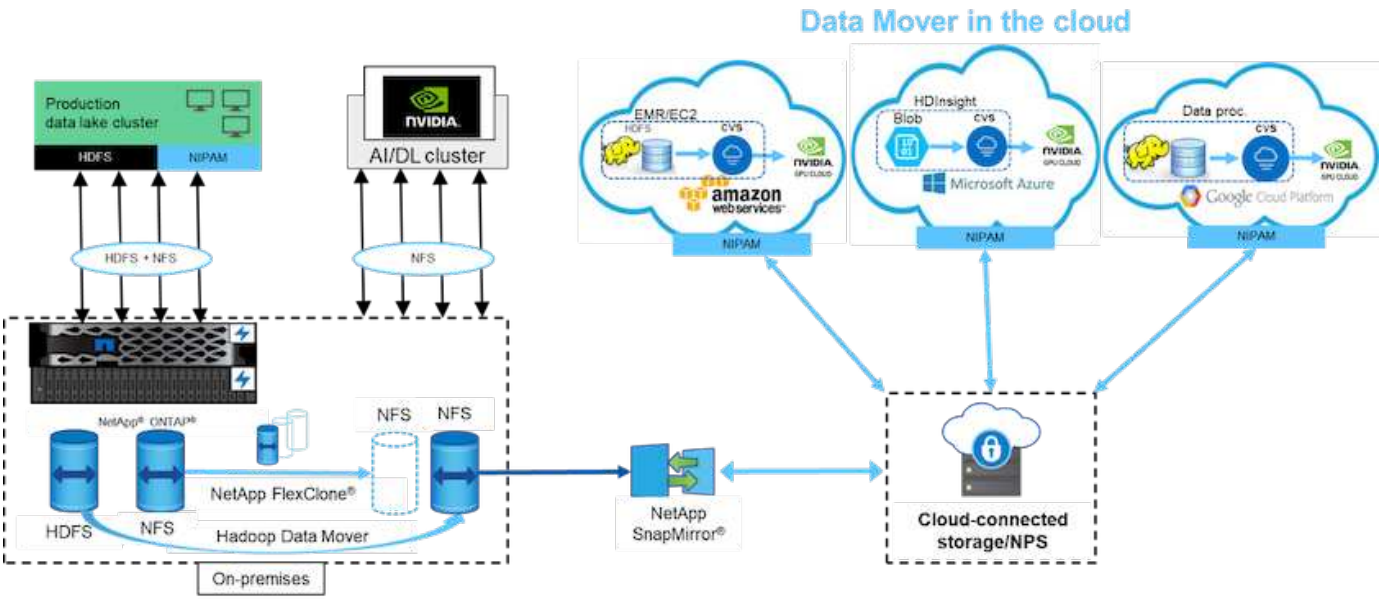
Il `hadoop distcp` Il comando utilizza il programma MapReduce per copiare i dati. NIPAM lavora con MapReduce per fungere da driver per il cluster Hadoop durante la copia dei dati. NIPAM può distribuire un carico su più interfacce di rete per una singola esportazione. Questo processo massimizza il throughput di rete distribuendo i dati tra più interfacce di rete quando si copiano i dati da HDFS o HCFS a NFS.

NIPAM non è supportato o certificato con MapR.

Soluzione di data mover per l'ai

La soluzione data mover per l'ai si basa sulle esigenze dei clienti di elaborare i dati Hadoop dalle operazioni ai. NetApp trasferisce i dati da HDFS a NFS utilizzando NIPAM. In un caso di utilizzo, il cliente doveva spostare i dati su NFS on-premise e un altro cliente doveva spostare i dati da Windows Azure Storage Blob a Cloud Volumes Service per elaborare i dati dalle istanze cloud della GPU nel cloud.

Il seguente diagramma illustra i dettagli della soluzione data mover.



Per creare la soluzione di data mover sono necessari i seguenti passaggi:

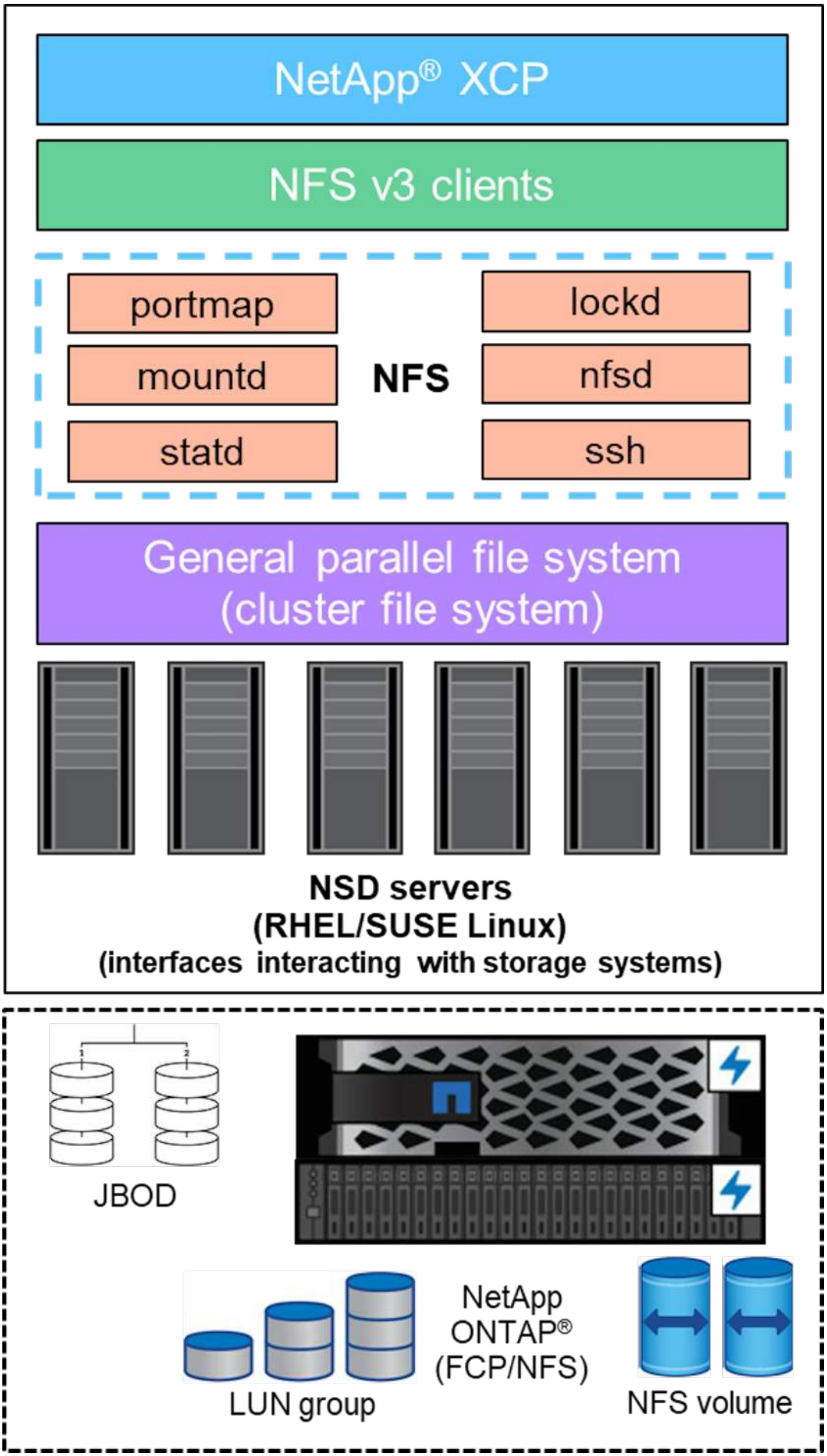
1. LA SAN ONTAP fornisce HDFS e il NAS fornisce il volume NFS tramite NIPAM al cluster di data Lake di produzione.
2. I dati del cliente sono in HDFS e NFS. I dati NFS possono essere dati di produzione di altre applicazioni utilizzate per l'analisi dei big data e le operazioni ai.
3. La tecnologia NetApp FlexClone crea un clone del volume NFS di produzione e lo fornisce al cluster ai on-premise.
4. I dati di un LUN SAN HDFS vengono copiati in un volume NFS con NIPAM e il `hadoop distcp` comando. NIPAM utilizza la larghezza di banda di più interfacce di rete per trasferire i dati. Questo processo riduce i tempi di copia dei dati in modo che sia possibile trasferire più dati.
5. Entrambi i volumi NFS vengono forniti al cluster ai per le operazioni ai.
6. Per elaborare i dati NFS on-the-premise con GPU nel cloud, i volumi NFS vengono mirrorati su NetApp Private Storage (NPS) con la tecnologia NetApp SnapMirror e montati sui cloud service provider per GPU.
7. Il cliente desidera elaborare i dati nei servizi EC2/EMR, HDInsight o DataProc nelle GPU dei provider di servizi cloud. Il data mover di Hadoop sposta i dati dai servizi Hadoop ai Cloud Volumes Services con NIPAM e a. `hadoop distcp` comando.
8. I dati Cloud Volumes Service vengono forniti all'ai tramite il protocollo NFS.i dati elaborati tramite l'ai possono essere inviati in una posizione on-premise per l'analisi dei big data oltre al cluster NVIDIA tramite NIPAM, SnapMirror e NPS.

In questo scenario, il cliente dispone di dati con un elevato numero di file nel sistema NAS in una posizione remota richiesta per l'elaborazione dell'ai sul controller di storage NetApp on-premise. In questo scenario, è meglio utilizzare XCP Migration Tool per migrare i dati a una velocità superiore.

Il cliente con caso d'utilizzo ibrido può utilizzare BlueXP Copy e Sync per migrare i dati on-premise dai dati NFS, CIFS e S3 nel cloud e viceversa per l'elaborazione ai utilizzando GPU come quelle in un cluster NVIDIA. Sia BlueXP Copy che Sync e lo strumento di migrazione XCP sono utilizzati per la migrazione dei dati NFS in NetApp ONTAP NFS.

GPF per NetApp ONTAP NFS

In questa convalida, abbiamo utilizzato quattro server come server NSD (Network Shared Disk) per fornire dischi fisici per GPFS. LE GPF vengono create sui dischi NSD per esportarle come esportazioni NFS in modo che i client NFS possano accedervi, come mostrato nella figura seguente. Abbiamo utilizzato XCP per copiare i dati da NFS esportati da GPFS in un volume NetApp NFS.


NetApp® XCP
NFS v3 clients
portmap
mountd
statd
NFS
lockd
nfsd
ssh
General parallel file system
(cluster file system)
NSD servers
(RHEL/SUSE Linux)
(interfaces interacting with storage systems)
JBOD
LUN group
NetApp
ONTAP®
(FCP/NFS)
NFS volume

Elementi essenziali DELLA GPF

In GPFS vengono utilizzati i seguenti tipi di nodo:

- **Admin node.** specifica un campo opzionale contenente un nome di nodo utilizzato dai comandi di amministrazione per comunicare tra i nodi. Ad esempio, il nodo admin `mastr-51.netapp.com` può passare un controllo di rete a tutti gli altri nodi del cluster.
- Nodo **Quorum.** determina se un nodo è incluso nel pool di nodi da cui è derivato il quorum. È necessario almeno un nodo come nodo di quorum.
- **Manager Node.** indica se un nodo fa parte del pool di nodi da cui è possibile selezionare i gestori del file system e i gestori dei token. È consigliabile definire più di un nodo come nodo manager. Il numero di nodi da assegnare come manager dipende dal carico di lavoro e dal numero di licenze del server GPFS di cui si dispone. Se si eseguono lavori paralleli di grandi dimensioni, potrebbero essere necessari più nodi di gestione rispetto a un cluster a quattro nodi che supporta un'applicazione Web.
- **Server NSD.** il server che prepara ogni disco fisico per l'utilizzo con GPFS.
- **Protocol node.** nodo che condivide i dati GPFS direttamente tramite qualsiasi protocollo SSH (Secure Shell) con NFS. Questo nodo richiede una licenza server GPFS.

Elenco delle operazioni per GPFS, NFS e XCP

Questa sezione fornisce l'elenco delle operazioni che creano GPFS, esportano GPFS come esportazione NFS e trasferiscono i dati utilizzando XCP.

Creare GPFS

Per creare GPFS, attenersi alla seguente procedura:

1. Scaricare e installare l'accesso ai dati in scala di spettro per la versione Linux su uno dei server.
2. Installare il pacchetto prerequisito (ad esempio Chef) in tutti i nodi e disattivare Security-Enhanced Linux (SELinux) in tutti i nodi.
3. Impostare il nodo di installazione e aggiungere il nodo admin e il nodo GPFS al file di definizione del cluster.
4. Aggiungere il nodo manager, il nodo quorum, i server NSD e il nodo GPFS.
5. Aggiungere i nodi GUI, admin e GPFS e, se necessario, aggiungere un server GUI aggiuntivo.
6. Aggiungere un altro nodo GPFS e controllare l'elenco di tutti i nodi.
7. Specificare un nome cluster, un profilo, un binario shell remoto, un binario copia file remoto e un intervallo di porte da impostare su tutti i nodi GPFS nel file di definizione del cluster.
8. Visualizzare le impostazioni di configurazione GPFS e aggiungere un nodo admin aggiuntivo.
9. Disattivare la raccolta di dati e caricare il pacchetto di dati su IBM Support Center.
10. Abilitare NTP e controllare le configurazioni prima dell'installazione.
11. Configurare, creare e controllare i dischi NSD.
12. Creare il GPFS.
13. Montare il GPFS.
14. Verificare e fornire le autorizzazioni necessarie per GPFS.
15. Verificare la lettura e la scrittura del GPFS eseguendo `dd` comando.

Esportare GPFS in NFS

Per esportare GPFS in NFS, attenersi alla seguente procedura:

1. Esportare GPFS come NFS tramite `/etc/exports` file.
2. Installare i pacchetti server NFS richiesti.
3. Avviare il servizio NFS.
4. Elencare i file nella GPFS per convalidare il client NFS.

Configurare il client NFS

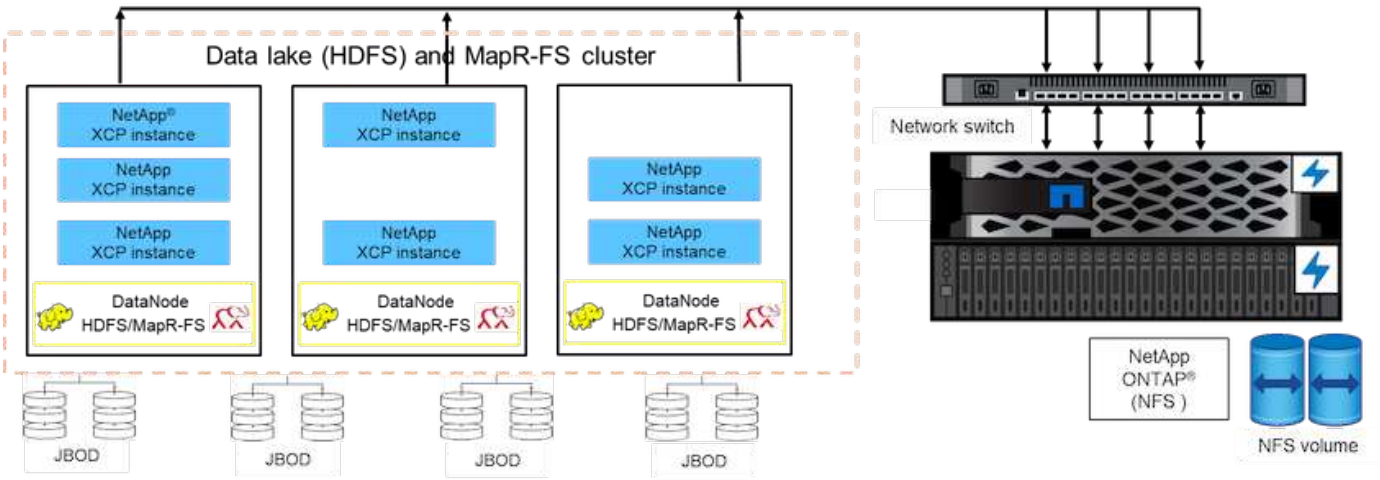
Per configurare il client NFS, attenersi alla seguente procedura:

1. Esportare il GPFS come NFS attraverso `/etc/exports` file.
2. Avviare i servizi client NFS.
3. Montare il GPFS tramite il protocollo NFS sul client NFS.
4. Convalidare l'elenco dei file GPFS nella cartella NFS Mounted.
5. Spostare i dati da NFS esportati da GPFS a NetApp NFS utilizzando XCP.
6. Convalidare i file GPFS sul client NFS.

HDFS e MapR-FS su NFS ONTAP

Per questa soluzione, NetApp ha validato la migrazione dei dati da dati Lake (HDFS) e dati del cluster MapR a NFS ONTAP. I dati risiedevano in MapR-FS e HDFS. NetApp XCP ha introdotto una nuova funzionalità che consente la migrazione diretta dei dati da un file system distribuito come HDFS e MapR-FS a ONTAP NFS. XCP utilizza thread asincroni e chiamate API HDFS C per comunicare e trasferire dati da MapR- FS e HDFS.

La figura seguente mostra la migrazione dei dati da un data Lake (HDFS) e MapR-FS a un NFS ONTAP. Con questa nuova funzionalità, non è necessario esportare l'origine come condivisione NFS.



Perché i clienti stanno passando da HDFS e MapR-FS a NFS?

La maggior parte delle distribuzioni Hadoop, come Cloudera e Hortonworks, utilizza le distribuzioni HDFS e

MapR per memorizzare i dati utilizzando il proprio file system chiamato MapR-FS. I dati HDFS e MapR-FS forniscono informazioni preziose ai data scientist che possono essere sfruttate nell'apprendimento automatico (ML) e nel deep learning (DL). I dati in HDFS e MapR-FS non sono condivisi, il che significa che non possono essere utilizzati da altre applicazioni. I clienti cercano dati condivisi, in particolare nel settore bancario in cui i dati sensibili dei clienti vengono utilizzati da più applicazioni. L'ultima versione di Hadoop (3.x o successiva) supporta l'origine dati NFS, a cui è possibile accedere senza software aggiuntivo di terze parti. Con la nuova funzionalità XCP di NetApp, è possibile spostare i dati direttamente da HDFS e MapR-FS a NetApp NFS per fornire l'accesso a più applicazioni

I test sono stati eseguiti in Amazon Web Services (AWS) per trasferire i dati da MapR-FS a NFS per il test iniziale delle performance con 12 nodi MAPR e 4 server NFS.

	Quantità	Dimensione	VCPU	Memoria	Storage	Rete
Server NFS	4	i3en.24xlarge	96	488 GiB	8 SSD NVMe 7500	100
Nodi MapR	12	I3en.12xLarge	48	384GiB	4 SSD NVMe 7500	50

In base ai test iniziali, abbiamo ottenuto un throughput di 20 Gbps e siamo stati in grado di trasferire 2 PB al giorno di dati.

Per ulteriori informazioni sulla migrazione dei dati HDFS senza esportare HDFS in NFS, vedere la sezione "fasi di implementazione - NAS" in "TR-4863: TR-4863: Linee guida sulle Best practice per NetApp XCP - Data Mover, migrazione dei file e analisi".

Benefici per il business

Il trasferimento dei dati dall'analisi dei big data all'ai offre i seguenti vantaggi:

- Capacità di estrarre dati da diversi file system Hadoop e GPFS in un sistema storage NFS unificato
- Un metodo automatizzato e integrato con Hadoop per trasferire i dati
- Riduzione del costo dello sviluppo delle librerie per lo spostamento dei dati dai file system Hadoop
- Prestazioni massime grazie al throughput aggregato di più interfacce di rete da una singola origine di dati utilizzando NIPAM
- Metodi pianificati e on-demand per il trasferimento dei dati
- Efficienza dello storage e funzionalità di gestione aziendale per dati NFS unificati utilizzando il software di gestione dei dati ONTAP
- Nessun costo per lo spostamento dei dati con il metodo Hadoop per il trasferimento dei dati

PASSAGGI dettagliati DA GPF a NFS

In questa sezione vengono fornite le procedure dettagliate necessarie per configurare GPFS e spostare i dati in NFS utilizzando NetApp XCP.

Configurare GPFS

1. Scaricare e installare Spectrum Scale Data Access per Linux su uno dei server.

```
[root@mastr-51 Spectrum_Scale_Data_Access-5.0.3.1-x86_64-Linux-
install_folder]# ls
Spectrum_Scale_Data_Access-5.0.3.1-x86_64-Linux-install
[root@mastr-51 Spectrum_Scale_Data_Access-5.0.3.1-x86_64-Linux-
install_folder]# chmod +x Spectrum_Scale_Data_Access-5.0.3.1-x86_64-
Linux-install
[root@mastr-51 Spectrum_Scale_Data_Access-5.0.3.1-x86_64-Linux-
install_folder]# ./Spectrum_Scale_Data_Access-5.0.3.1-x86_64-Linux-
install --manifest
manifest
…
<contents removes to save page space>
…
```

2. Installare il pacchetto prerequisito (inclusi chef e kernel header) su tutti i nodi.

```
[root@mastr-51 5.0.3.1]# for i in 51 53 136 138 140  ; do ssh
10.63.150.$i "hostname; rpm -ivh  /gpfs_install/chef* "; done
mastr-51.netapp.com
warning: /gpfs_install/chef-13.6.4-1.el7.x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 83ef826a: NOKEY
Preparing...
########################################
package chef-13.6.4-1.el7.x86_64 is already installed
mastr-53.netapp.com
warning: /gpfs_install/chef-13.6.4-1.el7.x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 83ef826a: NOKEY
Preparing...
########################################
Updating / installing...
chef-13.6.4-1.el7
########################################
Thank you for installing Chef!
workr-136.netapp.com
warning: /gpfs_install/chef-13.6.4-1.el7.x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 83ef826a: NOKEY
Preparing...
########################################
Updating / installing...
chef-13.6.4-1.el7
########################################
Thank you for installing Chef!
workr-138.netapp.com
warning: /gpfs_install/chef-13.6.4-1.el7.x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 83ef826a: NOKEY
```

```
Preparing...
########################################
Updating / installing...
chef-13.6.4-1.el7
########################################
Thank you for installing Chef!
workr-140.netapp.com
warning: /gpfs_install/chef-13.6.4-1.el7.x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 83ef826a: NOKEY
Preparing...
########################################
Updating / installing...
chef-13.6.4-1.el7
########################################
Thank you for installing Chef!
[root@mastr-51 5.0.3.1]#
[root@mastr-51 installer]# for i in 51 53 136 138 140  ; do ssh
10.63.150.$i "hostname; yumdownloader kernel-headers-3.10.0-
862.3.2.el7.x86_64 ; rpm -Uvh --oldpackage kernel-headers-3.10.0-
862.3.2.el7.x86_64.rpm"; done
mastr-51.netapp.com
Loaded plugins: priorities, product-id, subscription-manager
Preparing...
########################################
Updating / installing...
kernel-headers-3.10.0-862.3.2.el7
########################################
Cleaning up / removing...
kernel-headers-3.10.0-957.21.2.el7
########################################
mastr-53.netapp.com
Loaded plugins: product-id, subscription-manager
Preparing...
########################################
Updating / installing...
kernel-headers-3.10.0-862.3.2.el7
########################################
Cleaning up / removing...
kernel-headers-3.10.0-862.11.6.el7
########################################
workr-136.netapp.com
Loaded plugins: product-id, subscription-manager
Repository ambari-2.7.3.0 is listed more than once in the configuration
Preparing...
########################################
Updating / installing...
```

```
kernel-headers-3.10.0-862.3.2.el7
##########################################
Cleaning up / removing...
kernel-headers-3.10.0-862.11.6.el7
##########################################
workr-138.netapp.com
Loaded plugins: product-id, subscription-manager
Preparing...
##########################################
package kernel-headers-3.10.0-862.3.2.el7.x86_64 is already installed
workr-140.netapp.com
Loaded plugins: product-id, subscription-manager
Preparing...
##########################################
Updating / installing...
kernel-headers-3.10.0-862.3.2.el7
##########################################
Cleaning up / removing...
kernel-headers-3.10.0-862.11.6.el7
##########################################
[root@mastr-51 installer]#
```

3. Disattivare SELinux in tutti i nodi.

```
[root@mastr-51 5.0.3.1]# for i in 51 53 136 138 140  ; do ssh
10.63.150.$i "hostname; sudo setenforce 0"; done
mastr-51.netapp.com
setenforce: SELinux is disabled
mastr-53.netapp.com
setenforce: SELinux is disabled
workr-136.netapp.com
setenforce: SELinux is disabled
workr-138.netapp.com
setenforce: SELinux is disabled
workr-140.netapp.com
setenforce: SELinux is disabled
[root@mastr-51 5.0.3.1]#
```

4. Configurare il nodo di installazione.

```
[root@mastr-51 installer]# ./spectrumscale setup -s 10.63.150.51
[ INFO  ] Installing prerequisites for install node
[ INFO  ] Existing Chef installation detected. Ensure the PATH is
configured so that chef-client and knife commands can be run.
[ INFO  ] Your control node has been configured to use the IP
10.63.150.51 to communicate with other nodes.
[ INFO  ] Port 8889 will be used for chef communication.
[ INFO  ] Port 10080 will be used for package distribution.
[ INFO  ] Install Toolkit setup type is set to Spectrum Scale (default).
If an ESS is in the cluster, run this command to set ESS mode:
./spectrumscale setup -s server_ip -st ess
[ INFO  ] SUCCESS
[ INFO  ] Tip : Designate protocol, nsd and admin nodes in your
environment to use during install:./spectrumscale -v node add <node> -p
-a -n
[root@mastr-51 installer]#
```

5. Aggiungere il nodo admin e il nodo GPFS al file di definizione del cluster.

```
[root@mastr-51 installer]# ./spectrumscale node add mastr-51 -a
[ INFO  ] Adding node mastr-51.netapp.com as a GPFS node.
[ INFO  ] Setting mastr-51.netapp.com as an admin node.
[ INFO  ] Configuration updated.
[ INFO  ] Tip : Designate protocol or nsd nodes in your environment to
use during install:./spectrumscale node add <node> -p -n
[root@mastr-51 installer]#
```

6. Aggiungere il nodo manager e il nodo GPFS.

```
[root@mastr-51 installer]# ./spectrumscale node add mastr-53 -m
[ INFO  ] Adding node mastr-53.netapp.com as a GPFS node.
[ INFO  ] Adding node mastr-53.netapp.com as a manager node.
[root@mastr-51 installer]#
```

7. Aggiungere il nodo quorum e il nodo GPFS.

```
[root@mastr-51 installer]# ./spectrumscale node add workr-136 -q
[ INFO  ] Adding node workr-136.netapp.com as a GPFS node.
[ INFO  ] Adding node workr-136.netapp.com as a quorum node.
[root@mastr-51 installer]#
```

8. Aggiungere i server NSD e il nodo GPFS.

```
[root@mastr-51 installer]# ./spectrumscale node add workr-138 -n
[ INFO  ] Adding node workr-138.netapp.com as a GPFS node.
[ INFO  ] Adding node workr-138.netapp.com as an NSD server.
[ INFO  ] Configuration updated.
[ INFO  ] Tip :If all node designations are complete, add NSDs to your
cluster definition and define required filessytems:./spectrumscale nsd
add <device> -p <primary node> -s <secondary node> -fs <file system>
[root@mastr-51 installer]#
```

9. Aggiungere i nodi GUI, admin e GPFS.

```
[root@mastr-51 installer]# ./spectrumscale node add workr-136 -g
[ INFO  ] Setting workr-136.netapp.com as a GUI server.
[root@mastr-51 installer]# ./spectrumscale node add workr-136 -a
[ INFO  ] Setting workr-136.netapp.com as an admin node.
[ INFO  ] Configuration updated.
[ INFO  ] Tip : Designate protocol or nsd nodes in your environment to
use during install:./spectrumscale node add <node> -p -n
[root@mastr-51 installer]#
```

10. Aggiungere un altro server GUI.

```
[root@mastr-51 installer]# ./spectrumscale node add mastr-53 -g
[ INFO  ] Setting mastr-53.netapp.com as a GUI server.
[root@mastr-51 installer]#
```

11. Aggiungere un altro nodo GPFS.

```
[root@mastr-51 installer]# ./spectrumscale node add workr-140
[ INFO  ] Adding node workr-140.netapp.com as a GPFS node.
[root@mastr-51 installer]#
```

12. Verificare ed elencare tutti i nodi.

```
[root@mastr-51 installer]# ./spectrumscale node list
[ INFO  ] List of nodes in current configuration:
[ INFO  ] [Installer Node]
[ INFO  ] 10.63.150.51
[ INFO  ]
[ INFO  ] [Cluster Details]
[ INFO  ] No cluster name configured
[ INFO  ] Setup Type: Spectrum Scale
[ INFO  ]
[ INFO  ] [Extended Features]
[ INFO  ] File Audit logging     : Disabled
[ INFO  ] Watch folder           : Disabled
[ INFO  ] Management GUI         : Enabled
[ INFO  ] Performance Monitoring : Disabled
[ INFO  ] Callhome               : Enabled
[ INFO  ]
[ INFO  ] GPFS                 Admin  Quorum  Manager   NSD   Protocol
GUI   Callhome   OS   Arch
[ INFO  ] Node                  Node   Node     Node   Server   Node
Server  Server
[ INFO  ] mastr-51.netapp.com    X
rhel7  x86_64
[ INFO  ] mastr-53.netapp.com                    X
X             rhel7  x86_64
[ INFO  ] workr-136.netapp.com   X       X
X             rhel7  x86_64
[ INFO  ] workr-138.netapp.com                          X
rhel7  x86_64
[ INFO  ] workr-140.netapp.com
rhel7  x86_64
[ INFO  ]
[ INFO  ] [Export IP address]
[ INFO  ] No export IP addresses configured
[root@mastr-51 installer]#
```

13. Specificare un nome del cluster nel file di definizione del cluster.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs -c mastr-
51.netapp.com
[ INFO  ] Setting GPFS cluster name to mastr-51.netapp.com
[root@mastr-51 installer]#
```

14. Specificare il profilo.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs -p default
[ INFO  ] Setting GPFS profile to default
[root@mastr-51 installer]#
Profiles options: default [gpfsProtocolDefaults], random I/O
[gpfsProtocolsRandomIO], sequential I/O [gpfsProtocolDefaults], random
I/O [gpfsProtocolRandomIO]
```

15. Specificare il binario della shell remota che deve essere utilizzato da GPFS; use `-r argument`.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs -r /usr/bin/ssh
[ INFO  ] Setting Remote shell command to /usr/bin/ssh
[root@mastr-51 installer]#
```

16. Specificare il binario di copia del file remoto da utilizzare da GPFS; Use `-rc argument`.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs -rc /usr/bin/scp
[ INFO  ] Setting Remote file copy command to /usr/bin/scp
[root@mastr-51 installer]#
```

17. Specificare l'intervallo di porte da impostare su tutti i nodi GPFS; utilizzare `-e argument`.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs -e 60000-65000
[ INFO  ] Setting GPFS Daemon communication port range to 60000-65000
[root@mastr-51 installer]#
```

18. Visualizzare le impostazioni di configurazione di GPFS.

```
[root@mastr-51 installer]# ./spectrumscale config gpfs --list
[ INFO  ] Current settings are as follows:
[ INFO  ] GPFS cluster name is mastr-51.netapp.com.
[ INFO  ] GPFS profile is default.
[ INFO  ] Remote shell command is /usr/bin/ssh.
[ INFO  ] Remote file copy command is /usr/bin/scp.
[ INFO  ] GPFS Daemon communication port range is 60000-65000.
[root@mastr-51 installer]#
```

19. Aggiungere un nodo admin.

```
[root@mastr-51 installer]# ./spectrumscale node add 10.63.150.53 -a
[ INFO  ] Setting mastr-53.netapp.com as an admin node.
[ INFO  ] Configuration updated.
[ INFO  ] Tip : Designate protocol or nsd nodes in your environment to
use during install:./spectrumscale node add <node> -p -n
[root@mastr-51 installer]#
```

20. Disattivare la raccolta di dati e caricare il pacchetto di dati su IBM Support Center.

```
[root@mastr-51 installer]# ./spectrumscale callhome disable
[ INFO  ] Disabling the callhome.
[ INFO  ] Configuration updated.
[root@mastr-51 installer]#
```

21. Abilitare NTP.

```
[root@mastr-51 installer]# ./spectrumscale config ntp -e on
[root@mastr-51 installer]# ./spectrumscale config ntp -l
[ INFO  ] Current settings are as follows:
[ WARN  ] No value for Upstream NTP Servers(comma separated IP's with NO
space between multiple IPs) in clusterdefinition file.
[root@mastr-51 installer]# ./spectrumscale config ntp -s 10.63.150.51
[ WARN  ] The NTP package must already be installed and full
bidirectional access to the UDP port 123 must be allowed.
[ WARN  ] If NTP is already running on any of your nodes, NTP setup will
be skipped. To stop NTP run 'service ntpd stop'.
[ WARN  ] NTP is already on
[ INFO  ] Setting Upstream NTP Servers(comma separated IP's with NO
space between multiple IPs) to 10.63.150.51
[root@mastr-51 installer]# ./spectrumscale config ntp -e on
[ WARN  ] NTP is already on
[root@mastr-51 installer]# ./spectrumscale config ntp -l
[ INFO  ] Current settings are as follows:
[ INFO  ] Upstream NTP Servers(comma separated IP's with NO space
between multiple IPs) is 10.63.150.51.
[root@mastr-51 installer]#

[root@mastr-51 installer]# service ntpd start
Redirecting to /bin/systemctl start ntpd.service
[root@mastr-51 installer]# service ntpd status
Redirecting to /bin/systemctl status ntpd.service
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor
preset: disabled)
```

```
   Active: active (running) since Tue 2019-09-10 14:20:34 UTC; 1s ago
  Process: 2964 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS
(code=exited, status=0/SUCCESS)
 Main PID: 2965 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─2965 /usr/sbin/ntpd -u ntp:ntp -g

Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: ntp_io: estimated max
descriptors: 1024, initial socket boundary: 16
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen and drop on 0
v4wildcard 0.0.0.0 UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen and drop on 1
v6wildcard :: UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen normally on 2 lo
127.0.0.1 UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen normally on 3
enp4s0f0 10.63.150.51 UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen normally on 4 lo
::1 UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listen normally on 5
enp4s0f0 fe80::219:99ff:feef:99fa UDP 123
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: Listening on routing
socket on fd #22 for interface updates
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: 0.0.0.0 c016 06 restart
Sep 10 14:20:34 mastr-51.netapp.com ntpd[2965]: 0.0.0.0 c012 02 freq_set
kernel 11.890 PPM
[root@mastr-51 installer]#
```

22. Controllare le configurazioni prima dell'installazione.

```
[root@mastr-51 installer]# ./spectrumscale install -pr
[ INFO  ] Logging to file: /usr/lpp/mmfs/5.0.3.1/installer/logs/INSTALL-
PRECHECK-10-09-2019_14:51:43.log
[ INFO  ] Validating configuration
[ INFO  ] Performing Chef (deploy tool) checks.
[ WARN  ] NTP is already running on: mastr-51.netapp.com. The install
toolkit will no longer setup NTP.
[ INFO  ] Node(s): ['workr-138.netapp.com'] were defined as NSD node(s)
but the toolkit has not been told about any NSDs served by these node(s)
nor has the toolkit been told to create new NSDs on these node(s). The
install will continue and these nodes will be assigned server licenses.
If NSDs are desired, either add them to the toolkit with
<./spectrumscale nsd add> followed by a <./spectrumscale install> or add
them manually afterwards using mmcrnsd.
[ INFO  ] Install toolkit will not configure file audit logging as it
has been disabled.
[ INFO  ] Install toolkit will not configure watch folder as it has been
disabled.
[ INFO  ] Checking for knife bootstrap configuration...
[ INFO  ] Performing GPFS checks.
[ INFO  ] Running environment checks
[ INFO  ] Skipping license validation as no existing GPFS cluster
detected.
[ INFO  ] Checking pre-requisites for portability layer.
[ INFO  ] GPFS precheck OK
[ INFO  ] Performing Performance Monitoring checks.
[ INFO  ] Running environment checks for Performance Monitoring
[ INFO  ] Performing GUI checks.
[ INFO  ] Performing FILE AUDIT LOGGING checks.
[ INFO  ] Running environment checks for file  Audit logging
[ INFO  ] Network check from admin node workr-136.netapp.com to all
other nodes in the cluster passed
[ INFO  ] Network check from admin node mastr-51.netapp.com to all other
nodes in the cluster passed
[ INFO  ] Network check from admin node mastr-53.netapp.com to all other
nodes in the cluster passed
[ INFO  ] The install toolkit will not configure call home as it is
disabled. To enable call home, use the following CLI command:
./spectrumscale callhome enable
[ INFO  ] Pre-check successful for install.
[ INFO  ] Tip : ./spectrumscale install
[root@mastr-51 installer]#
```

23. Configurare i dischi NSD.

```
[root@mastr-51 cluster-test]# cat disk.1st
%nsd: device=/dev/sdf
nsd=nsd1
servers=workr-136
usage=dataAndMetadata
failureGroup=1

%nsd: device=/dev/sdf
nsd=nsd2
servers=workr-138
usage=dataAndMetadata
failureGroup=1
```

24. Creare i dischi NSD.

```
[root@mastr-51 cluster-test]# mmcrnsd -F disk.1st -v no
mmcrnsd: Processing disk sdf
mmcrnsd: Processing disk sdf
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root@mastr-51 cluster-test]#
```

25. Controllare lo stato del disco NSD.

```
[root@mastr-51 cluster-test]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
---
 (free disk)   nsd1         workr-136.netapp.com
 (free disk)   nsd2         workr-138.netapp.com

[root@mastr-51 cluster-test]#
```

26. Creare il GPFS.

```
[root@mastr-51 cluster-test]# mmcrfs gpfs1 -F disk.1st -B 1M -T /gpfs1

The following disks of gpfs1 will be formatted on node workr-
136.netapp.com:
    nsd1: size 3814912 MB
    nsd2: size 3814912 MB
Formatting file system ...
Disks up to size 33.12 TB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root@mastr-51 cluster-test]#
```

27. Montare il GPFS.

```
[root@mastr-51 cluster-test]# mmmount all -a
Tue Oct  8 18:05:34 UTC 2019: mmmount: Mounting file systems ...
[root@mastr-51 cluster-test]#
```

28. Controllare e fornire le autorizzazioni necessarie per GPFS.

```
[root@mastr-51 cluster-test]# mmlsdisk gpfs1
disk         driver   sector     failure holds    holds
storage
name         type       size       group metadata data  status
availability pool
------------ -------- ------ ----------- -------- ----- -------------
------------ ------------
nsd1         nsd         512           1 Yes      Yes   ready         up
system
nsd2         nsd         512           1 Yes      Yes   ready         up
system
[root@mastr-51 cluster-test]#

[root@mastr-51 cluster-test]# for i in 51 53 136 138  ; do ssh
10.63.150.$i "hostname; chmod 777 /gpfs1" ; done;
mastr-51.netapp.com
mastr-53.netapp.com
workr-136.netapp.com
workr-138.netapp.com
[root@mastr-51 cluster-test]#
```

29. Controllare la lettura e la scrittura della GPFS eseguendo `dd` comando.

```
[root@mastr-51 cluster-test]# dd if=/dev/zero of=/gpfs1/testfile
bs=1024M count=5
5+0 records in
5+0 records out
5368709120 bytes (5.4 GB) copied, 8.3981 s, 639 MB/s
[root@mastr-51 cluster-test]# for i in 51 53 136 138  ; do ssh
10.63.150.$i "hostname; ls -ltrh /gpfs1" ; done;
mastr-51.netapp.com
total 5.0G
-rw-r--r-- 1 root root 5.0G Oct  8 18:10 testfile
mastr-53.netapp.com
total 5.0G
-rw-r--r-- 1 root root 5.0G Oct  8 18:10 testfile
workr-136.netapp.com
total 5.0G
-rw-r--r-- 1 root root 5.0G Oct  8 18:10 testfile
workr-138.netapp.com
total 5.0G
-rw-r--r-- 1 root root 5.0G Oct  8 18:10 testfile
[root@mastr-51 cluster-test]#
```

Esportare GPFS in NFS

Per esportare GPFS in NFS, attenersi alla seguente procedura:

1. Esportare il GPFS come NFS attraverso `/etc/exports` file.

```
[root@mastr-51 gpfs1]# cat /etc/exports
/gpfs1        *(rw,fsid=745)
[root@mastr-51 gpfs1]
```

2. Installare i pacchetti server NFS richiesti.

```
[root@mastr-51 ~]# yum install rpcbind
Loaded plugins: priorities, product-id, search-disabled-repos,
subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package rpcbind.x86_64 0:0.2.0-47.el7 will be updated
---> Package rpcbind.x86_64 0:0.2.0-48.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

=================================================================================
=================================================================================
=================================================================================
======================
 Package                                                Arch
Version                                                     Repository
Size
=================================================================================
=================================================================================
=================================================================================
======================
Updating:
 rpcbind                                                x86_64
0.2.0-48.el7                                                rhel-7-
server-rpms                                                 60 k

Transaction Summary
=================================================================================
=================================================================================
=================================================================================
======================
Upgrade  1 Package
```

```
Total download size: 60 k
Is this ok [y/d/N]: y
Downloading packages:
No Presto metadata available for rhel-7-server-rpms
rpcbind-0.2.0-48.el7.x86_64.rpm
|  60 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : rpcbind-0.2.0-48.el7.x86_64
1/2
  Cleanup    : rpcbind-0.2.0-47.el7.x86_64
2/2
  Verifying  : rpcbind-0.2.0-48.el7.x86_64
1/2
  Verifying  : rpcbind-0.2.0-47.el7.x86_64
2/2

Updated:
  rpcbind.x86_64 0:0.2.0-48.el7

Complete!
[root@mastr-51 ~]#
```

3. Avviare il servizio NFS.

```
[root@mastr-51 ~]# service nfs status
Redirecting to /bin/systemctl status nfs.service
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled;
vendor preset: disabled)
  Drop-In: /run/systemd/generator/nfs-server.service.d
           └─order-with-mounts.conf
   Active: inactive (dead)
[root@mastr-51 ~]# service rpcbind start
Redirecting to /bin/systemctl start rpcbind.service
[root@mastr-51 ~]# service nfs start
Redirecting to /bin/systemctl start nfs.service
[root@mastr-51 ~]# service nfs status
Redirecting to /bin/systemctl status nfs.service
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled;
vendor preset: disabled)
  Drop-In: /run/systemd/generator/nfs-server.service.d
           └─order-with-mounts.conf
   Active: active (exited) since Wed 2019-11-06 16:34:50 UTC; 2s ago
  Process: 24402 ExecStartPost=/bin/sh -c if systemctl -q is-active
gssproxy; then systemctl reload gssproxy ; fi (code=exited,
status=0/SUCCESS)
  Process: 24383 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited,
status=0/SUCCESS)
  Process: 24379 ExecStartPre=/usr/sbin/exportfs -r (code=exited,
status=0/SUCCESS)
 Main PID: 24383 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service

Nov 06 16:34:50 mastr-51.netapp.com systemd[1]: Starting NFS server and
services...
Nov 06 16:34:50 mastr-51.netapp.com systemd[1]: Started NFS server and
services.
[root@mastr-51 ~]#
```

4. Elencare i file in GPFS per convalidare il client NFS.

```
[root@mastr-51 gpfs1]# df -Th
Filesystem                                  Type      Size  Used Avail
Use% Mounted on
/dev/mapper/rhel_stlrx300s6--22--irmc-root xfs        94G   55G   39G
59% /
devtmpfs                                    devtmpfs   32G     0   32G
0% /dev
tmpfs                                       tmpfs      32G     0   32G
0% /dev/shm
tmpfs                                       tmpfs      32G  3.3G   29G
11% /run
tmpfs                                       tmpfs      32G     0   32G
0% /sys/fs/cgroup
/dev/sda7                                   xfs       9.4G  210M  9.1G
3% /boot
tmpfs                                       tmpfs     6.3G     0  6.3G
0% /run/user/10065
tmpfs                                       tmpfs     6.3G     0  6.3G
0% /run/user/10068
tmpfs                                       tmpfs     6.3G     0  6.3G
0% /run/user/10069
10.63.150.213:/nc_volume3                   nfs4      380G  8.0M  380G
1% /mnt
tmpfs                                       tmpfs     6.3G     0  6.3G
0% /run/user/0
gpfs1                                       gpfs      7.3T  9.1G  7.3T
1% /gpfs1
[root@mastr-51 gpfs1]#
[root@mastr-51 ~]# cd /gpfs1
[root@mastr-51 gpfs1]# ls
catalog  ces  gpfs-ces  ha  testfile
[root@mastr-51 gpfs1]#
[root@mastr-51 ~]# cd /gpfs1
[root@mastr-51 gpfs1]# ls
ces  gpfs-ces  ha  testfile
[root@mastr-51 gpfs1]# ls -ltrha
total 5.1G
dr-xr-xr-x   2 root root 8.0K Jan  1  1970 .snapshots
-rw-r--r--   1 root root 5.0G Oct  8 18:10 testfile
dr-xr-xr-x. 30 root root 4.0K Oct  8 18:19 ..
drwxr-xr-x   2 root root 4.0K Nov  5 20:02 gpfs-ces
drwxr-xr-x   2 root root 4.0K Nov  5 20:04 ha
drwxrwxrwx   5 root root 256K Nov  5 20:04 .
drwxr-xr-x   4 root root 4.0K Nov  5 20:35 ces
[root@mastr-51 gpfs1]#
```

Configurare il client NFS

Per configurare il client NFS, attenersi alla seguente procedura:

1. Installare i pacchetti nel client NFS.

```
[root@hdp2 ~]# yum install nfs-utils rpcbind
Loaded plugins: product-id, search-disabled-repos, subscription-manager
HDP-2.6-GPL-repo-4
| 2.9 kB  00:00:00
HDP-2.6-repo-4
| 2.9 kB  00:00:00
HDP-3.0-GPL-repo-2
| 2.9 kB  00:00:00
HDP-3.0-repo-2
| 2.9 kB  00:00:00
HDP-3.0-repo-3
| 2.9 kB  00:00:00
HDP-3.1-repo-1
| 2.9 kB  00:00:00
HDP-3.1-repo-51
| 2.9 kB  00:00:00
HDP-UTILS-1.1.0.22-repo-1
| 2.9 kB  00:00:00
HDP-UTILS-1.1.0.22-repo-2
| 2.9 kB  00:00:00
HDP-UTILS-1.1.0.22-repo-3
| 2.9 kB  00:00:00
HDP-UTILS-1.1.0.22-repo-4
| 2.9 kB  00:00:00
HDP-UTILS-1.1.0.22-repo-51
| 2.9 kB  00:00:00
ambari-2.7.3.0
| 2.9 kB  00:00:00
epel/x86_64/metalink
|  13 kB  00:00:00
epel
| 5.3 kB  00:00:00
mysql-connectors-community
| 2.5 kB  00:00:00
mysql-tools-community
| 2.5 kB  00:00:00
mysql56-community
| 2.5 kB  00:00:00
rhel-7-server-optional-rpms
| 3.2 kB  00:00:00
rhel-7-server-rpms
```

```
| 3.5 kB  00:00:00
(1/10): mysql-connectors-community/x86_64/primary_db
|  49 kB  00:00:00
(2/10): mysql-tools-community/x86_64/primary_db
|  66 kB  00:00:00
(3/10): epel/x86_64/group_gz
|  90 kB  00:00:00
(4/10): mysql56-community/x86_64/primary_db
| 241 kB  00:00:00
(5/10): rhel-7-server-optional-rpms/7Server/x86_64/updateinfo
| 2.5 MB  00:00:00
(6/10): rhel-7-server-rpms/7Server/x86_64/updateinfo
| 3.4 MB  00:00:00
(7/10): rhel-7-server-optional-rpms/7Server/x86_64/primary_db
| 8.3 MB  00:00:00
(8/10): rhel-7-server-rpms/7Server/x86_64/primary_db
|  62 MB  00:00:01
(9/10): epel/x86_64/primary_db
| 6.9 MB  00:00:08
(10/10): epel/x86_64/updateinfo
| 1.0 MB  00:00:13
Resolving Dependencies
--> Running transaction check
---> Package nfs-utils.x86_64 1:1.3.0-0.61.el7 will be updated
---> Package nfs-utils.x86_64 1:1.3.0-0.65.el7 will be an update
---> Package rpcbind.x86_64 0:0.2.0-47.el7 will be updated
---> Package rpcbind.x86_64 0:0.2.0-48.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

=========================================================================
===============================================
 Package                 Arch                 Version
Repository                         Size
=========================================================================
===============================================
Updating:
 nfs-utils               x86_64               1:1.3.0-0.65.el7
rhel-7-server-rpms               412 k
 rpcbind                 x86_64               0.2.0-48.el7
rhel-7-server-rpms                60 k

Transaction Summary
=========================================================================
===============================================
```

```
Upgrade  2 Packages

Total download size: 472 k
Is this ok [y/d/N]: y
Downloading packages:
No Presto metadata available for rhel-7-server-rpms
(1/2): rpcbind-0.2.0-48.el7.x86_64.rpm
|  60 kB  00:00:00
(2/2): nfs-utils-1.3.0-0.65.el7.x86_64.rpm
| 412 kB  00:00:00
--------------------------------------------------------------------------
----------------------------------------------
Total
1.2 MB/s | 472 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : rpcbind-0.2.0-48.el7.x86_64
1/4
service rpcbind start

  Updating   : 1:nfs-utils-1.3.0-0.65.el7.x86_64
2/4
  Cleanup    : 1:nfs-utils-1.3.0-0.61.el7.x86_64
3/4
  Cleanup    : rpcbind-0.2.0-47.el7.x86_64
4/4
  Verifying  : 1:nfs-utils-1.3.0-0.65.el7.x86_64
1/4
  Verifying  : rpcbind-0.2.0-48.el7.x86_64
2/4
  Verifying  : rpcbind-0.2.0-47.el7.x86_64
3/4
  Verifying  : 1:nfs-utils-1.3.0-0.61.el7.x86_64
4/4

Updated:
  nfs-utils.x86_64 1:1.3.0-0.65.el7
rpcbind.x86_64 0:0.2.0-48.el7

Complete!
[root@hdp2 ~]#
```

2. Avviare i servizi client NFS.

```
[root@hdp2 ~]# service rpcbind start
Redirecting to /bin/systemctl start rpcbind.service
 [root@hdp2 ~]#
```

3. Montare il GPFS tramite il protocollo NFS sul client NFS.

```
[root@hdp2 ~]# mkdir /gpfstest
[root@hdp2 ~]# mount 10.63.150.51:/gpfs1 /gpfstest
[root@hdp2 ~]# df -h
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/rhel_stlrx300s6--22-root  1.1T  113G  981G  11% /
devtmpfs                                126G     0  126G   0% /dev
tmpfs                                   126G   16K  126G   1% /dev/shm
tmpfs                                   126G  510M  126G   1% /run
tmpfs                                   126G     0  126G   0%
/sys/fs/cgroup
/dev/sdd2                               197M  191M  6.6M  97% /boot
tmpfs                                    26G     0   26G   0% /run/user/0
10.63.150.213:/nc_volume2                95G  5.4G   90G   6% /mnt
10.63.150.51:/gpfs1                     7.3T  9.1G  7.3T   1% /gpfstest
[root@hdp2 ~]#
```

4. Convalidare l’elenco dei file GPFS nella cartella montata su NFS.

```
[root@hdp2 ~]# cd /gpfstest/
[root@hdp2 gpfstest]# ls
ces  gpfs-ces  ha  testfile
[root@hdp2 gpfstest]# ls -l
total 5242882
drwxr-xr-x 4 root root       4096 Nov  5 15:35 ces
drwxr-xr-x 2 root root       4096 Nov  5 15:02 gpfs-ces
drwxr-xr-x 2 root root       4096 Nov  5 15:04 ha
-rw-r--r-- 1 root root 5368709120 Oct  8 14:10 testfile
[root@hdp2 gpfstest]#
```

5. Spostare i dati dal file NFS esportato con GPFS al NetApp NFS utilizzando XCP.

```
[root@hdp2 linux]# ./xcp copy -parallel 20 10.63.150.51:/gpfs1
10.63.150.213:/nc_volume2/
XCP 1.4-17914d6; (c) 2019 NetApp, Inc.; Licensed to Karthikeyan
Nagalingam [NetApp Inc] until Tue Nov  5 12:39:36 2019

xcp: WARNING: your license will expire in less than one week! You can
renew your license at https://xcp.netapp.com
xcp: open or create catalog 'xcp': Creating new catalog in
'10.63.150.51:/gpfs1/catalog'
xcp: WARNING: No index name has been specified, creating one with name:
autoname_copy_2019-11-11_12.14.07.805223
xcp: mount '10.63.150.51:/gpfs1': WARNING: This NFS server only supports
1-second timestamp granularity. This may cause sync to fail because
changes will often be undetectable.
 34 scanned, 32 copied, 32 indexed, 1 giant, 301 MiB in (59.5 MiB/s),
784 KiB out (155 KiB/s), 6s
 34 scanned, 32 copied, 32 indexed, 1 giant, 725 MiB in (84.6 MiB/s),
1.77 MiB out (206 KiB/s), 11s
 34 scanned, 32 copied, 32 indexed, 1 giant, 1.17 GiB in (94.2 MiB/s),
2.90 MiB out (229 KiB/s), 16s
 34 scanned, 32 copied, 32 indexed, 1 giant, 1.56 GiB in (79.8 MiB/s),
3.85 MiB out (194 KiB/s), 21s
 34 scanned, 32 copied, 32 indexed, 1 giant, 1.95 GiB in (78.4 MiB/s),
4.80 MiB out (191 KiB/s), 26s
 34 scanned, 32 copied, 32 indexed, 1 giant, 2.35 GiB in (80.4 MiB/s),
5.77 MiB out (196 KiB/s), 31s
 34 scanned, 32 copied, 32 indexed, 1 giant, 2.79 GiB in (89.6 MiB/s),
6.84 MiB out (218 KiB/s), 36s
 34 scanned, 32 copied, 32 indexed, 1 giant, 3.16 GiB in (75.3 MiB/s),
7.73 MiB out (183 KiB/s), 41s
 34 scanned, 32 copied, 32 indexed, 1 giant, 3.53 GiB in (75.4 MiB/s),
8.64 MiB out (183 KiB/s), 46s
 34 scanned, 32 copied, 32 indexed, 1 giant, 4.00 GiB in (94.4 MiB/s),
9.77 MiB out (230 KiB/s), 51s
 34 scanned, 32 copied, 32 indexed, 1 giant, 4.46 GiB in (94.3 MiB/s),
10.9 MiB out (229 KiB/s), 56s
 34 scanned, 32 copied, 32 indexed, 1 giant, 4.86 GiB in (80.2 MiB/s),
11.9 MiB out (195 KiB/s), 1m1s
Sending statistics...
34 scanned, 33 copied, 34 indexed, 1 giant, 5.01 GiB in (81.8 MiB/s),
12.3 MiB out (201 KiB/s), 1m2s.
[root@hdp2 linux]#
```

6. Convalidare i file GPFS sul client NFS.

```
[root@hdp2 mnt]# df -Th
Filesystem                          Type      Size  Used Avail Use%
Mounted on
/dev/mapper/rhel_stlrx300s6--22-root xfs       1.1T  113G  981G  11% /
devtmpfs                            devtmpfs  126G     0  126G   0%
/dev
tmpfs                               tmpfs     126G   16K  126G   1%
/dev/shm
tmpfs                               tmpfs     126G  518M  126G   1%
/run
tmpfs                               tmpfs     126G     0  126G   0%
/sys/fs/cgroup
/dev/sdd2                           xfs       197M  191M  6.6M  97%
/boot
tmpfs                               tmpfs      26G     0   26G   0%
/run/user/0
10.63.150.213:/nc_volume2           nfs4       95G  5.4G   90G   6%
/mnt
10.63.150.51:/gpfs1                 nfs4      7.3T  9.1G  7.3T   1%
/gpfstest
[root@hdp2 mnt]#
[root@hdp2 mnt]# ls -ltrha
total 128K
dr-xr-xr-x   2 root        root               4.0K Dec 31  1969
.snapshots
drwxrwxrwx   2 root        root               4.0K Feb 14  2018 data
drwxrwxrwx   3 root        root               4.0K Feb 14  2018
wcresult
drwxrwxrwx   3 root        root               4.0K Feb 14  2018
wcresult1
drwxrwxrwx   2 root        root               4.0K Feb 14  2018
wcresult2
drwxrwxrwx   2 root        root               4.0K Feb 16  2018
wcresult3
-rw-r--r--   1 root        root               2.8K Feb 20  2018
README demo
drwxrwxrwx   3 root        root               4.0K Jun 28 13:38 scantg
drwxrwxrwx   3 root        root               4.0K Jun 28 13:39
scancopyFromLocal
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:28 f3
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:28 README
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:28 f9
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:28 f6
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:28 f5
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:30 f4
-rw-r--r--   1 hdfs        hadoop             1.2K Jul  3 19:30 f8
```

```
-rw-r--r--   1 hdfs        hadoop              1.2K Jul  3 19:30 f2
-rw-r--r--   1 hdfs        hadoop              1.2K Jul  3 19:30 f7
drwxrwxrwx   2 root        root                4.0K Jul  9 11:14 test
drwxrwxrwx   3 root        root                4.0K Jul 10 16:35
warehouse
drwxr-xr-x   3       10061 tester1             4.0K Jul 15 14:40 sdd1
drwxrwxrwx   3 testeruser1 hadoopkerberosgroup 4.0K Aug 20 17:00
kermkdir
-rw-r--r--   1 testeruser1 hadoopkerberosgroup    0 Aug 21 14:20 newfile
drwxrwxrwx   2 testeruser1 hadoopkerberosgroup 4.0K Aug 22 10:13
teragen1copy_3
drwxrwxrwx   2 testeruser1 hadoopkerberosgroup 4.0K Aug 22 10:33
teragen2copy_1
-rw-rwxr--   1 root        hdfs                1.2K Sep 19 16:38 R1
drwx------   3 root        root                4.0K Sep 20 17:28 user
-rw-r--r--   1 root        root                5.0G Oct  8 14:10
testfile
drwxr-xr-x   2 root        root                4.0K Nov  5 15:02 gpfs-
ces
drwxr-xr-x   2 root        root                4.0K Nov  5 15:04 ha
drwxr-xr-x   4 root        root                4.0K Nov  5 15:35 ces
dr-xr-xr-x. 26 root        root                4.0K Nov  6 11:40 ..
drwxrwxrwx  21 root        root                4.0K Nov 11 12:14 .
drwxrwxrwx   7 nobody      nobody              4.0K Nov 11 12:14 catalog
[root@hdp2 mnt]#
```

Da MapR-FS a NFS ONTAP

In questa sezione vengono fornite le procedure dettagliate necessarie per spostare i dati MapR-FS in NFS ONTAP utilizzando NetApp XCP.

1. Eseguire il provisioning di tre LUN per ciascun nodo MapR e assegnare alle LUN la proprietà di tutti i nodi MapR.
2. Durante l'installazione, scegliere i LUN aggiunti di recente per i dischi del cluster MapR utilizzati per MapR-FS.
3. Installare un cluster MapR in base a. "Documentazione di MapR 6.1".
4. Controllare le operazioni di base di Hadoop utilizzando i comandi MapReduce, ad esempio `hadoop jar xxx`.
5. Conservare i dati dei clienti in MapR-FS. Ad esempio, utilizzando Teragen, abbiamo generato circa un terabyte di dati campione in MapR-FS.
6. Configurare MapR-FS come esportazione NFS.
   a. Disattivare il servizio nlockmgr su tutti i nodi MapR.

```
root@workr-138: ~$ rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100003    4   tcp   2049  nfs
    100227    3   tcp   2049  nfs_acl
    100003    4   udp   2049  nfs
    100227    3   udp   2049  nfs_acl
    100021    3   udp  55270  nlockmgr
    100021    4   udp  55270  nlockmgr
    100021    3   tcp  35025  nlockmgr
    100021    4   tcp  35025  nlockmgr
    100003    3   tcp   2049  nfs
    100005    3   tcp   2049  mountd
    100005    1   tcp   2049  mountd
    100005    3   udp   2049  mountd
    100005    1   udp   2049  mountd
root@workr-138: ~$

root@workr-138: ~$ rpcinfo -d 100021 3
root@workr-138: ~$ rpcinfo -d 100021 4
```

b. Esportare cartelle specifiche da MapR-FS su tutti i nodi MapR in `/opt/mapr/conf/exports` file. Non esportare la cartella padre con permessi diversi quando si esportano le sottocartelle.

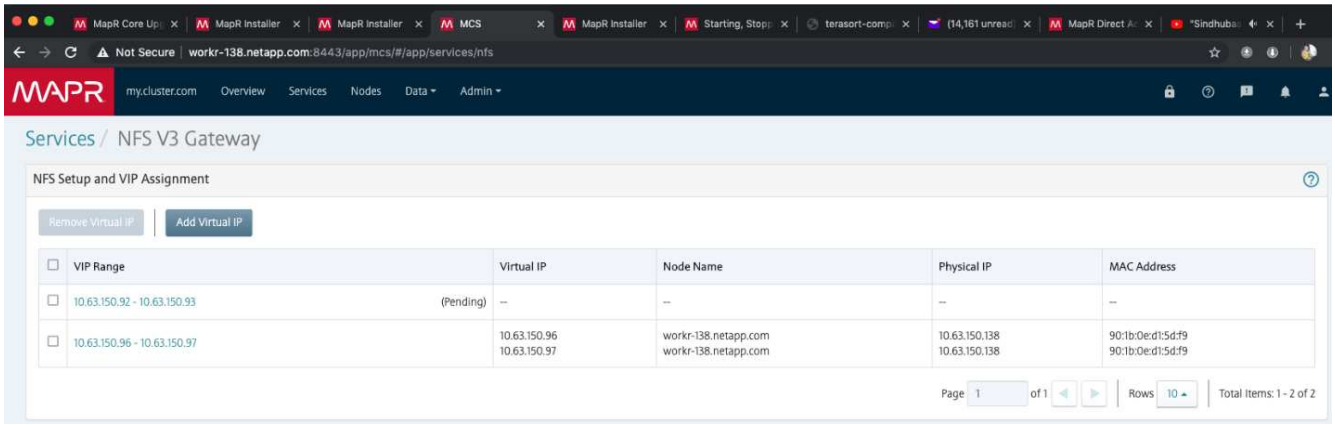
```
[mapr@workr-138 ~]$ cat /opt/mapr/conf/exports
# Sample Exports file
# for /mapr exports
# <Path> <exports_control>
#access_control -> order is specific to default
# list the hosts before specifying a default for all
#  a.b.c.d,1.2.3.4(ro) d.e.f.g(ro) (rw)
#  enforces ro for a.b.c.d & 1.2.3.4 and everybody else is rw
# special path to export clusters in mapr-clusters.conf. To disable
exporting,
# comment it out. to restrict access use the exports_control
#
#/mapr (rw)
#karthik
/mapr/my.cluster.com/tmp/testnfs /maprnfs3 (rw)
#to export only certain clusters, comment out the /mapr & uncomment.
#/mapr/clustername (rw)
#to export /mapr only to certain hosts (using exports_control)
#/mapr a.b.c.d(rw),e.f.g.h(ro)
# export /mapr/cluster1 rw to a.b.c.d & ro to e.f.g.h (denied for
others)
#/mapr/cluster1 a.b.c.d(rw),e.f.g.h(ro)
# export /mapr/cluster2 only to e.f.g.h (denied for others)
#/mapr/cluster2 e.f.g.h(rw)
# export /mapr/cluster3 rw to e.f.g.h & ro to others
#/mapr/cluster2 e.f.g.h(rw) (ro)
#to export a certain cluster, volume or a subdirectory as an alias,
#comment out  /mapr & uncomment
#/mapr/clustername         /alias1 (rw)
#/mapr/clustername/vol     /alias2 (rw)
#/mapr/clustername/vol/dir /alias3 (rw)
#only the alias will be visible/exposed to the nfs client not the
mapr path, host options as before
[mapr@workr-138 ~]$
```

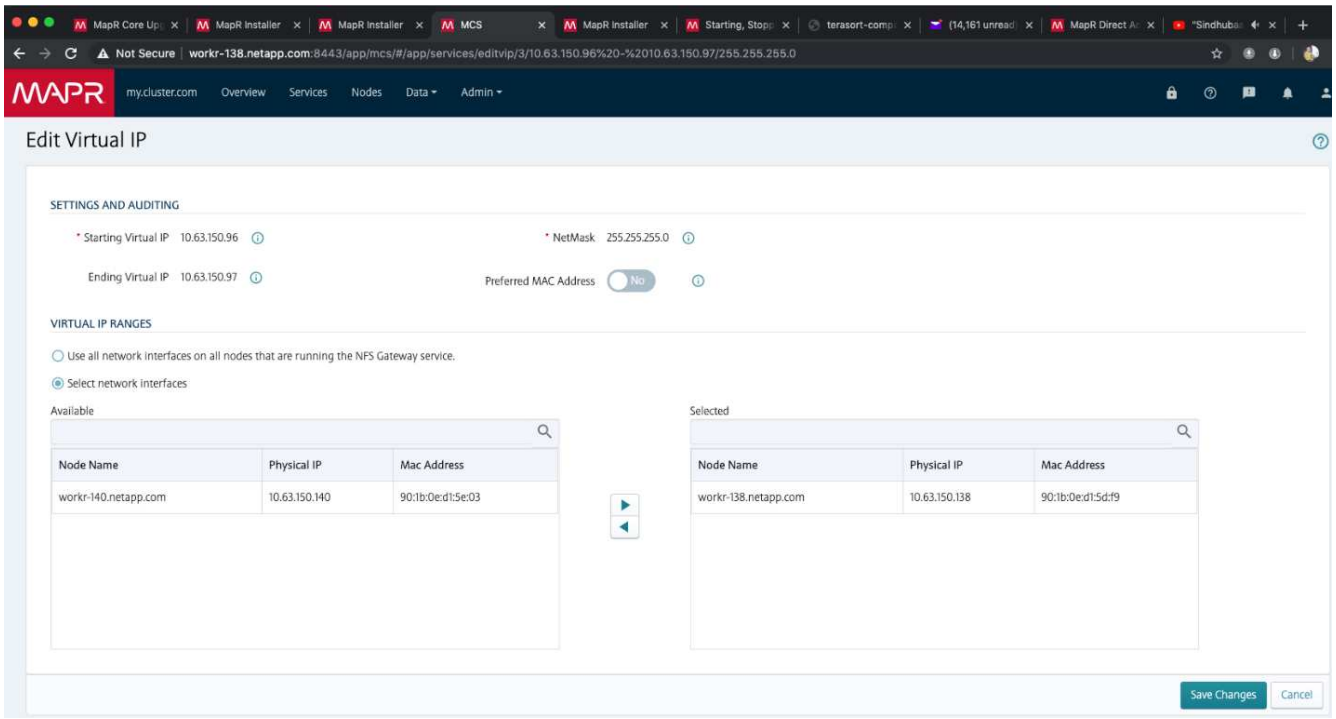
7. Aggiornare il servizio NFS MapR-FS.

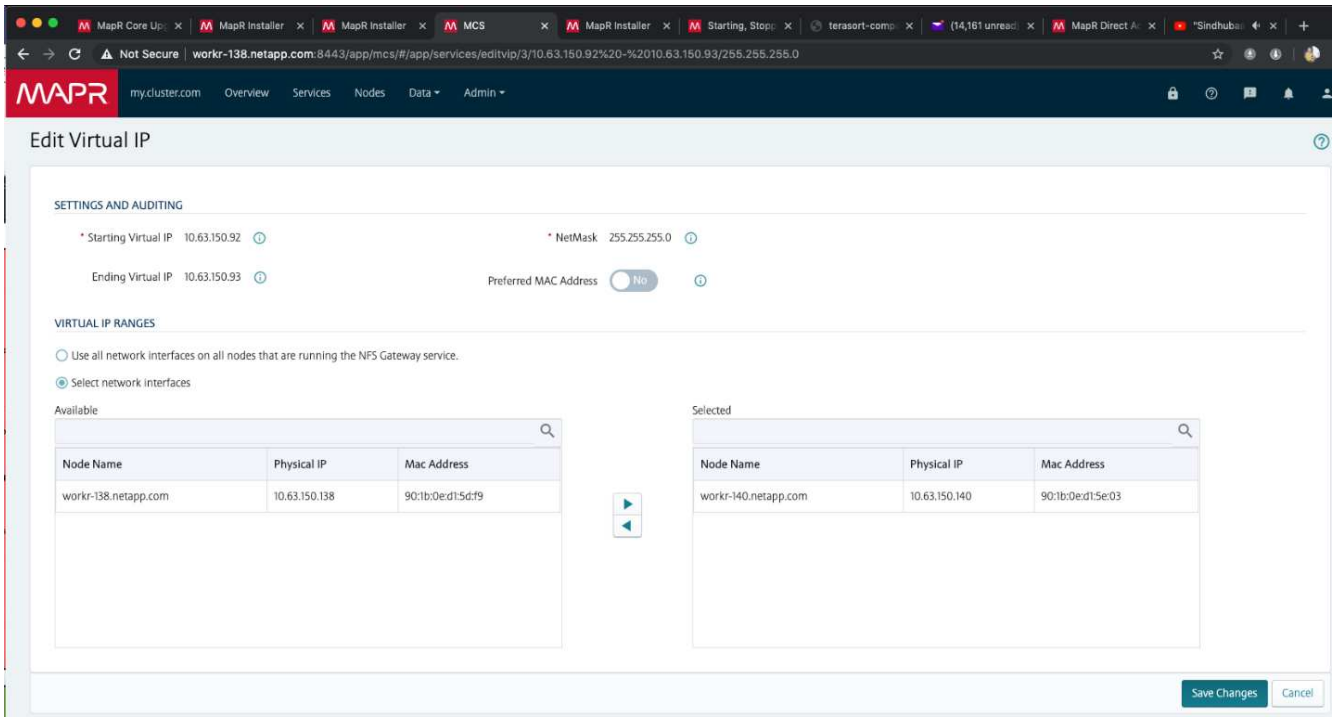
```
root@workr-138: tmp$ maprcli nfsmgmt refreshexports
ERROR (22) -  You do not have a ticket to communicate with
127.0.0.1:9998. Retry after obtaining a new ticket using maprlogin
root@workr-138: tmp$ su - mapr
[mapr@workr-138 ~]$ maprlogin password -cluster my.cluster.com
[Password for user 'mapr' at cluster 'my.cluster.com': ]
MapR credentials of user 'mapr' for cluster 'my.cluster.com' are written
to '/tmp/maprticket_5000'
[mapr@workr-138 ~]$ maprcli nfsmgmt refreshexports
```

8. Assegnare un intervallo IP virtuale a un server specifico o a un gruppo di server nel cluster MapR. Il cluster MapR assegna quindi un IP a un server specifico per l'accesso ai dati NFS. Gli IP abilitano la disponibilità elevata, il che significa che, in caso di guasto di un server o di una rete con un determinato IP, è possibile utilizzare l'IP successivo dell'intervallo di IP per l'accesso NFS.

> Se si desidera fornire l'accesso NFS da tutti i nodi MapR, è possibile assegnare un set di IP virtuali a ciascun server e utilizzare le risorse di ciascun nodo MapR per l'accesso ai dati NFS.







9. Controllare gli IP virtuali assegnati a ciascun nodo MapR e utilizzarli per l'accesso ai dati NFS.

```
root@workr-138: ~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: ens3f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP
group default qlen 1000
    link/ether 90:1b:0e:d1:5d:f9 brd ff:ff:ff:ff:ff:ff
    inet 10.63.150.138/24 brd 10.63.150.255 scope global noprefixroute
ens3f0
       valid_lft forever preferred_lft forever
    inet 10.63.150.96/24 scope global secondary ens3f0:~m0
       valid_lft forever preferred_lft forever
    inet 10.63.150.97/24 scope global secondary ens3f0:~m1
       valid_lft forever preferred_lft forever
    inet6 fe80::921b:eff:fed1:5df9/64 scope link
       valid_lft forever preferred_lft forever
3: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 90:1b:0e:d1:af:b4 brd ff:ff:ff:ff:ff:ff
4: ens3f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 90:1b:0e:d1:5d:fa brd ff:ff:ff:ff:ff:ff
5: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state
DOWN group default qlen 1000
    link/ether 90:1b:0e:d1:af:b5 brd ff:ff:ff:ff:ff:ff
[root@workr-138: ~$
[root@workr-140 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP
group default qlen 1000
    link/ether 90:1b:0e:d1:5e:03 brd ff:ff:ff:ff:ff:ff
    inet 10.63.150.140/24 brd 10.63.150.255 scope global noprefixroute
ens3f0
       valid_lft forever preferred_lft forever
    inet 10.63.150.92/24 scope global secondary ens3f0:~m0
       valid_lft forever preferred_lft forever
    inet6 fe80::921b:eff:fed1:5e03/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 90:1b:0e:d1:af:9a brd ff:ff:ff:ff:ff:ff
4: ens3f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
```

```
    link/ether 90:1b:0e:d1:5e:04 brd ff:ff:ff:ff:ff:ff
5: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state
DOWN group default qlen 1000
    link/ether 90:1b:0e:d1:af:9b brd ff:ff:ff:ff:ff:ff
[root@workr-140 ~]#
```

10. Montare il file MapR-FS esportato con NFS utilizzando l'IP virtuale assegnato per controllare il funzionamento di NFS. Tuttavia, questo passaggio non è necessario per il trasferimento dei dati utilizzando NetApp XCP.

```
root@workr-138: tmp$ mount -v -t nfs 10.63.150.92:/maprnfs3
/tmp/testmount/
mount.nfs: timeout set for Thu Dec  5 15:31:32 2019
mount.nfs: trying text-based options
'vers=4.1,addr=10.63.150.92,clientaddr=10.63.150.138'
mount.nfs: mount(2): Protocol not supported
mount.nfs: trying text-based options
'vers=4.0,addr=10.63.150.92,clientaddr=10.63.150.138'
mount.nfs: mount(2): Protocol not supported
mount.nfs: trying text-based options 'addr=10.63.150.92'
mount.nfs: prog 100003, trying vers=3, prot=6
mount.nfs: trying 10.63.150.92 prog 100003 vers 3 prot TCP port 2049
mount.nfs: prog 100005, trying vers=3, prot=17
mount.nfs: trying 10.63.150.92 prog 100005 vers 3 prot UDP port 2049
mount.nfs: portmap query retrying: RPC: Timed out
mount.nfs: prog 100005, trying vers=3, prot=6
mount.nfs: trying 10.63.150.92 prog 100005 vers 3 prot TCP port 2049
root@workr-138: tmp$ df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/sda7                84G   48G   37G  57% /
devtmpfs                126G     0  126G   0% /dev
tmpfs                   126G     0  126G   0% /dev/shm
tmpfs                   126G   19M  126G   1% /run
tmpfs                   126G     0  126G   0% /sys/fs/cgroup
/dev/sdd1               3.7T  201G  3.5T   6% /mnt/sdd1
/dev/sda6               946M  220M  726M  24% /boot
tmpfs                    26G     0   26G   0% /run/user/5000
gpfs1                   7.3T  9.1G  7.3T   1% /gpfs1
tmpfs                    26G     0   26G   0% /run/user/0
localhost:/mapr         100G     0  100G   0% /mapr
10.63.150.92:/maprnfs3   53T  8.4G   53T   1% /tmp/testmount
root@workr-138: tmp$
```

11. Configurare NetApp XCP per il trasferimento dei dati dal gateway NFS MapR-FS a NFS ONTAP.
    a. Configurare la posizione del catalogo per XCP.

```
[root@hdp2 linux]# cat /opt/NetApp/xFiles/xcp/xcp.ini
# Sample xcp config
[xcp]
#catalog =  10.63.150.51:/gpfs1
catalog =  10.63.150.213:/nc_volume1
```

b. Copiare il file di licenza in `/opt/NetApp/xFiles/xcp/`.

```
root@workr-138: src$ cd /opt/NetApp/xFiles/xcp/
root@workr-138: xcp$ ls -ltrha
total 252K
drwxr-xr-x 3 root   root     16 Apr  4  2019 ..
-rw-r--r-- 1 root   root    105 Dec  5 19:04 xcp.ini
drwxr-xr-x 2 root   root     59 Dec  5 19:04 .
-rw-r--r-- 1 faiz89 faiz89  336 Dec  6 21:12 license
-rw-r--r-- 1 root   root    192 Dec  6 21:13 host
-rw-r--r-- 1 root   root   236K Dec 17 14:12 xcp.log
root@workr-138: xcp$
```

c. Attivare XCP utilizzando `xcp activate` comando.

d. Controllare l’origine per l’esportazione NFS.

```
[root@hdp2 linux]# ./xcp show 10.63.150.92
XCP 1.4-17914d6; (c) 2019 NetApp, Inc.; Licensed to Karthikeyan
Nagalingam [NetApp Inc] until Wed Feb  5 11:07:27 2020
getting pmap dump from 10.63.150.92 port 111...
getting export list from 10.63.150.92...
sending 1 mount and 4 nfs requests to 10.63.150.92...
== RPC Services ==
'10.63.150.92': TCP rpc services: MNT v1/3, NFS v3/4, NFSACL v3, NLM
v1/3/4, PMAP v2/3/4, STATUS v1
'10.63.150.92': UDP rpc services: MNT v1/3, NFS v4, NFSACL v3, NLM
v1/3/4, PMAP v2/3/4, STATUS v1
== NFS Exports ==
 Mounts  Errors  Server
      1       0  10.63.150.92
     Space    Files      Space    Files
      Free     Free       Used     Used Export
  52.3 TiB    53.7B   8.36 GiB    53.7B 10.63.150.92:/maprnfs3
== Attributes of NFS Exports ==
drwxr-xr-x --- root root 2 2 10m51s 10.63.150.92:/maprnfs3
1.77 KiB in (8.68 KiB/s), 3.16 KiB out (15.5 KiB/s), 0s.
[root@hdp2 linux]#
```

e. Trasferire i dati utilizzando XCP da più nodi MapR da IP di origine multipli e IP di destinazione multipli (LIF ONTAP).

```
root@workr-138: linux$ ./xcp_yatin copy --parallel 20
10.63.150.96,10.63.150.97:/maprnfs3/tg4
10.63.150.85,10.63.150.86:/datapipeline_dataset/tg4_dest
XCP 1.6-dev; (c) 2019 NetApp, Inc.; Licensed to Karthikeyan
Nagalingam [NetApp Inc] until Wed Feb  5 11:07:27 2020
xcp: WARNING: No index name has been specified, creating one with
name: autoname_copy_2019-12-06_21.14.38.652652
xcp: mount '10.63.150.96,10.63.150.97:/maprnfs3/tg4': WARNING: This
NFS server only supports 1-second timestamp granularity. This may
cause sync to fail because changes will often be undetectable.
 130 scanned, 128 giants, 3.59 GiB in (723 MiB/s), 3.60 GiB out (724
MiB/s), 5s
 130 scanned, 128 giants, 8.01 GiB in (889 MiB/s), 8.02 GiB out (890
MiB/s), 11s
 130 scanned, 128 giants, 12.6 GiB in (933 MiB/s), 12.6 GiB out (934
MiB/s), 16s
 130 scanned, 128 giants, 16.7 GiB in (830 MiB/s), 16.7 GiB out (831
MiB/s), 21s
 130 scanned, 128 giants, 21.1 GiB in (907 MiB/s), 21.1 GiB out (908
MiB/s), 26s
```

```
 130 scanned, 128 giants, 25.5 GiB in (893 MiB/s), 25.5 GiB out (894
MiB/s), 31s
 130 scanned, 128 giants, 29.6 GiB in (842 MiB/s), 29.6 GiB out (843
MiB/s), 36s
….
[root@workr-140 linux]# ./xcp_yatin copy  --parallel 20
10.63.150.92:/maprnfs3/tg4_2
10.63.150.85,10.63.150.86:/datapipeline_dataset/tg4_2_dest
XCP 1.6-dev; (c) 2019 NetApp, Inc.; Licensed to Karthikeyan
Nagalingam [NetApp Inc] until Wed Feb  5 11:07:27 2020
xcp: WARNING: No index name has been specified, creating one with
name: autoname_copy_2019-12-06_21.14.24.637773
xcp: mount '10.63.150.92:/maprnfs3/tg4_2': WARNING: This NFS server
only supports 1-second timestamp granularity. This may cause sync to
fail because changes will often be undetectable.
 130 scanned, 128 giants, 4.39 GiB in (896 MiB/s), 4.39 GiB out (897
MiB/s), 5s
 130 scanned, 128 giants, 9.94 GiB in (1.10 GiB/s), 9.96 GiB out
(1.10 GiB/s), 10s
 130 scanned, 128 giants, 15.4 GiB in (1.09 GiB/s), 15.4 GiB out
(1.09 GiB/s), 15s
 130 scanned, 128 giants, 20.1 GiB in (953 MiB/s), 20.1 GiB out (954
MiB/s), 20s
 130 scanned, 128 giants, 24.6 GiB in (928 MiB/s), 24.7 GiB out (929
MiB/s), 25s
 130 scanned, 128 giants, 29.0 GiB in (877 MiB/s), 29.0 GiB out (878
MiB/s), 31s
 130 scanned, 128 giants, 33.2 GiB in (852 MiB/s), 33.2 GiB out (853
MiB/s), 36s
 130 scanned, 128 giants, 37.8 GiB in (941 MiB/s), 37.8 GiB out (942
MiB/s), 41s
 130 scanned, 128 giants, 42.0 GiB in (860 MiB/s), 42.0 GiB out (861
MiB/s), 46s
 130 scanned, 128 giants, 46.1 GiB in (852 MiB/s), 46.2 GiB out (853
MiB/s), 51s
 130 scanned, 128 giants, 50.1 GiB in (816 MiB/s), 50.2 GiB out (817
MiB/s), 56s
 130 scanned, 128 giants, 54.1 GiB in (819 MiB/s), 54.2 GiB out (820
MiB/s), 1m1s
 130 scanned, 128 giants, 58.5 GiB in (897 MiB/s), 58.6 GiB out (898
MiB/s), 1m6s
 130 scanned, 128 giants, 62.9 GiB in (900 MiB/s), 63.0 GiB out (901
MiB/s), 1m11s
 130 scanned, 128 giants, 67.2 GiB in (876 MiB/s), 67.2 GiB out (877
MiB/s), 1m16s
```

f. Controllare la distribuzione del carico sul controller di storage.

```
Hadoop-AFF8080::*> statistics show-periodic -interval 2 -iterations 0
-summary true -object nic_common -counter rx_bytes|tx_bytes -node
Hadoop-AFF8080-01 -instance e3b
Hadoop-AFF8080: nic_common.e3b: 12/6/2019 15:55:04
 rx_bytes tx_bytes
 -------- --------
    879MB   4.67MB
    856MB   4.46MB
    973MB   5.66MB
    986MB   5.88MB
    945MB   5.30MB
    920MB   4.92MB
    894MB   4.76MB
    902MB   4.79MB
    886MB   4.68MB
    892MB   4.78MB
    908MB   4.96MB
    905MB   4.85MB
    899MB   4.83MB
Hadoop-AFF8080::*> statistics show-periodic -interval 2 -iterations 0
-summary true -object nic_common -counter rx_bytes|tx_bytes -node
Hadoop-AFF8080-01 -instance e9b
Hadoop-AFF8080: nic_common.e9b: 12/6/2019 15:55:07
 rx_bytes tx_bytes
 -------- --------
    950MB   4.93MB
    991MB   5.84MB
    959MB   5.63MB
    914MB   5.06MB
    903MB   4.81MB
    899MB   4.73MB
    892MB   4.71MB
    890MB   4.72MB
    905MB   4.86MB
    902MB   4.90MB
```

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Best practice per il modulo NetApp in-place Analytics

"https://www.netapp.com/us/media/tr-4382.pdf"

- Guida all'implementazione e alle Best practice per i volumi NetApp FlexGroup

  "https://www.netapp.com/us/media/tr-4571.pdf"

- Documentazione sui prodotti NetApp

  https://www.netapp.com/us/documentation/index.aspx

Best practice per Confluent Kafka

TR-4912: Linee guida sulle Best practice per lo storage a più livelli Confluent Kafka con NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluent

Apache Kafka è una piattaforma per lo streaming di eventi distribuita dalla community in grado di gestire migliaia di miliardi di eventi al giorno. Inizialmente concepito come coda di messaggistica, Kafka si basa su un'astrazione di un log di commit distribuito. Da quando è stata creata e open-source da LinkedIn nel 2011, Kafka si è evoluta da una coda di messaggi a una piattaforma completa per lo streaming di eventi. Confluent offre la distribuzione di Apache Kafka con la Confluent Platform. La piattaforma Confluent integra Kafka con funzionalità commerciali e di community aggiuntive progettate per migliorare l'esperienza di streaming di operatori e sviluppatori in produzione su vasta scala.

Questo documento descrive le linee guida sulle Best practice per l'utilizzo dello storage a livelli confluente su un'offerta di storage a oggetti NetApp, fornendo i seguenti contenuti:

- Verifica confluente con lo storage a oggetti NetApp: NetApp StorageGRID
- Tiered storage performance test
- Linee guida sulle Best practice per Confluent sui sistemi storage NetApp

Perché lo storage a livelli confluente?

Confluent è diventata la piattaforma di streaming real-time predefinita per molte applicazioni, in particolare per i big data, gli analytics e i carichi di lavoro di streaming. Tiered Storage consente agli utenti di separare il calcolo dallo storage nella piattaforma Confluent. L'archiviazione dei dati risulta più conveniente, consente di memorizzare quantità virtualmente infinite di dati e di scalare i carichi di lavoro on-demand in su (o in giù) e semplifica le attività amministrative come il ribilanciamento dei dati e dei tenant. I sistemi storage compatibili con S3 possono sfruttare tutte queste funzionalità per democratizzare i dati con tutti gli eventi in un unico posto, eliminando la necessità di un'ingegneria dei dati complessa. Per ulteriori informazioni sul motivo per cui dovresti utilizzare lo storage a più livelli per Kafka, consulta "Questo articolo di Confluent".

Perché scegliere NetApp StorageGRID per lo storage su più livelli?

StorageGRID è una piattaforma di storage a oggetti leader del settore di NetApp. StorageGRID è una soluzione di storage a oggetti, software-defined, che supporta API a oggetti standard di settore, inclusa l'API S3 (Simple Storage Service) di Amazon. StorageGRID archivia e gestisce i dati non strutturati su larga scala

per fornire uno storage a oggetti sicuro e durevole. I contenuti vengono posizionati nella giusta posizione, al momento giusto e nel giusto Tier di storage, ottimizzando i flussi di lavoro e riducendo i costi per i contenuti multimediali distribuiti a livello globale.

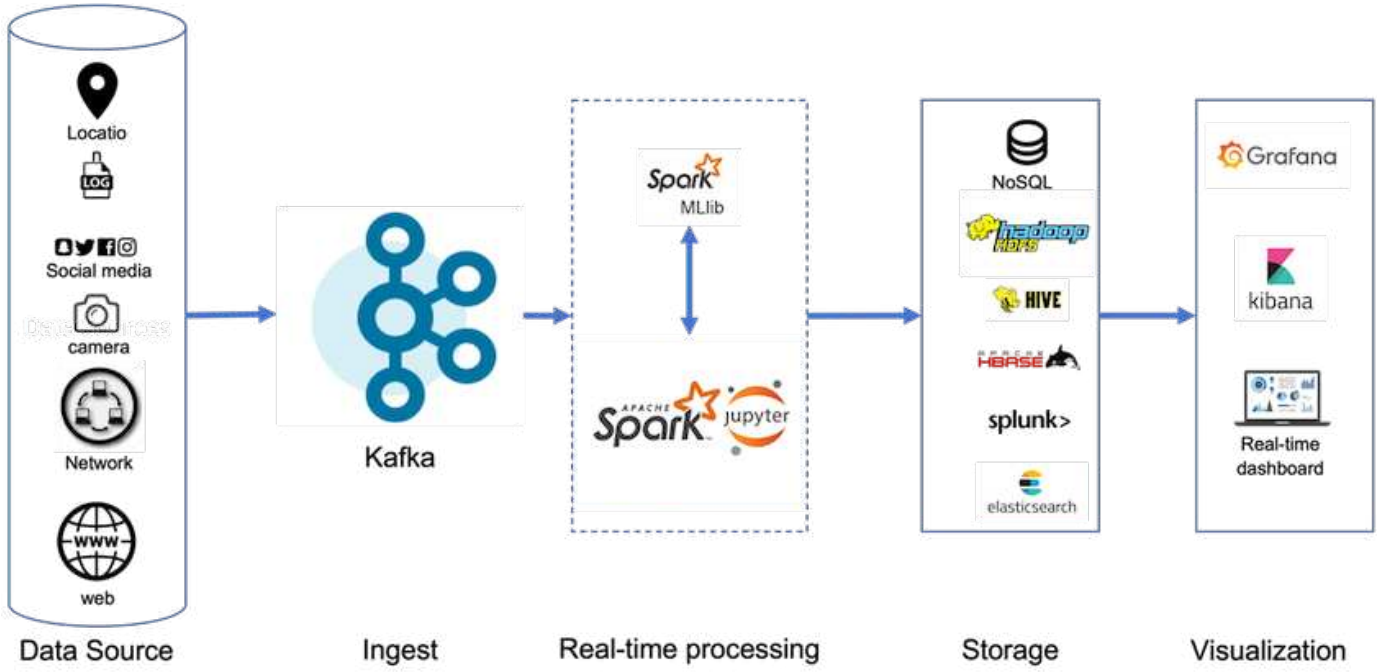
Il principale elemento di differenziazione per StorageGRID è il motore di policy per la gestione del ciclo di vita delle informazioni (ILM) che consente la gestione del ciclo di vita dei dati basata su policy. Il motore delle policy può utilizzare i metadati per gestire il modo in cui i dati vengono memorizzati per tutta la vita utile, per ottimizzare inizialmente le performance e ottimizzare automaticamente i costi e la durata con l'invecchiamento dei dati.

Abilitare lo storage a livelli confluente

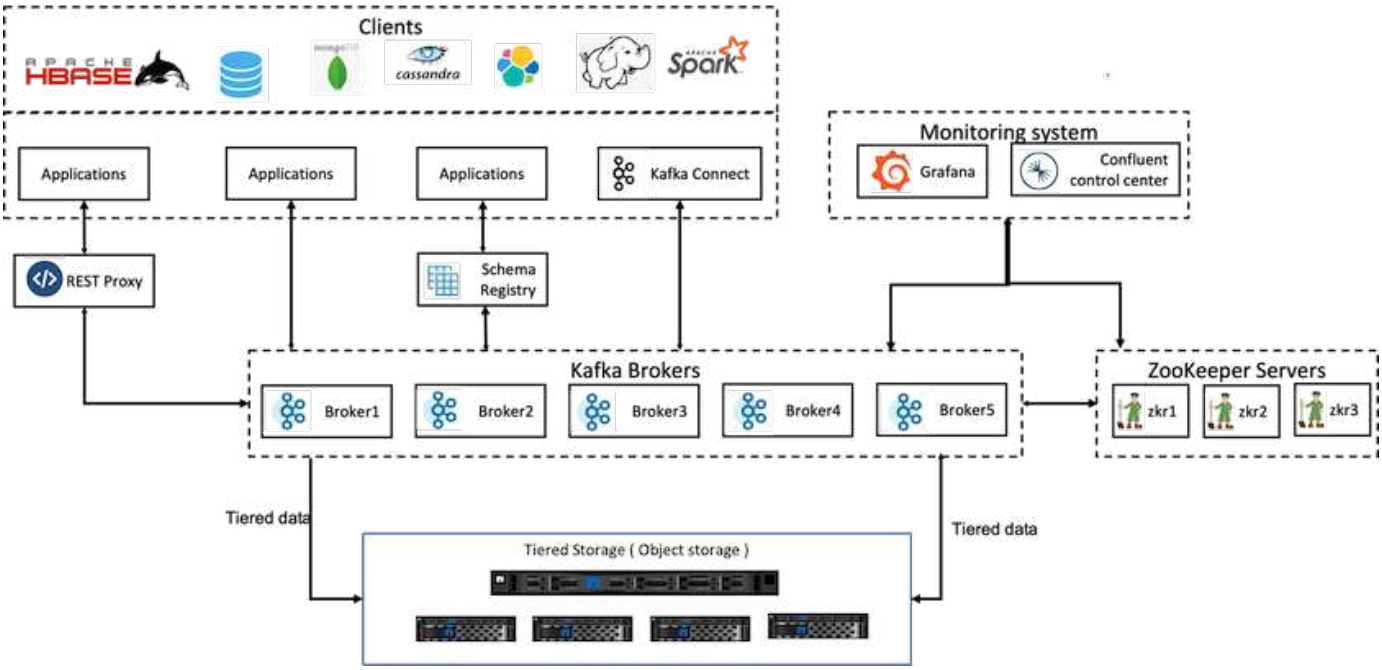
L'idea di base dello storage su più livelli è separare le attività dello storage dei dati dall'elaborazione dei dati. Grazie a questa separazione, la scalabilità indipendente del Tier di storage e del Tier di elaborazione dei dati diventa molto più semplice.

Una soluzione storage a più livelli per Confluent deve affrontare due fattori. Innanzitutto, deve aggirare o evitare proprietà comuni di coerenza e disponibilità degli archivi di oggetti, come incoerenze nelle operazioni DI ELENCO e occasionali indisponibilità degli oggetti. In secondo luogo, deve gestire correttamente l'interazione tra lo storage su più livelli e il modello di tolleranza agli errori e replica di Kafka, inclusa la possibilità che i leader zombie continuino a tierare gli intervalli di offset. Lo storage a oggetti NetApp offre la disponibilità costante degli oggetti e il modello ha che rendono lo storage stanco disponibile per gli intervalli di offset del Tier. Lo storage a oggetti NetApp offre una disponibilità degli oggetti coerente e un modello ha per rendere lo storage stanco disponibile per gli intervalli di offset del Tier.

Con lo storage a più livelli, puoi utilizzare piattaforme dalle performance elevate per letture e scritture a bassa latenza in prossimità della coda dei dati in streaming e puoi anche utilizzare archivi di oggetti scalabili e più economici come NetApp StorageGRID per letture storiche ad alto throughput. Disponiamo anche di una soluzione tecnica per Spark con controller di storage netapp e i dettagli sono qui. La figura seguente mostra come Kafka si inserisce in una pipeline di analytics in tempo reale.



La figura seguente mostra come NetApp StorageGRID si inserisce nel livello di storage a oggetti di Confluent Kafka.



Dettagli sull'architettura della soluzione

Questa sezione descrive l'hardware e il software utilizzati per la verifica confluente. Queste informazioni sono applicabili all'implementazione della piattaforma confluente con lo storage NetApp. La seguente tabella illustra l'architettura della soluzione testata e i componenti di base.

Componenti della soluzione	Dettagli
Confluent Kafka versione 6.2	• Tre zookeeper • Cinque server di broker • Cinque server di strumenti • Una Grafana • Un centro di controllo
Linux (Ubuntu 18.04)	Tutti i server
NetApp StorageGRID per lo storage su più livelli	• Software StorageGRID • 1 SG1000 (bilanciamento del carico) • 4 x SGF6024 • 4 SSD 24 x 800 • Protocollo S3 • 4 x 100 GbE (connettività di rete tra istanze di broker e StorageGRID)
15 server Fujitsu PRIMERGY RX2540	Ciascuno dotato di: * 2 CPU, 16 core fisici totali * Intel Xeon * 256 GB di memoria fisica * 100 GbE a doppia porta

Panoramica della tecnologia

Questa sezione descrive la tecnologia utilizzata in questa soluzione.

NetApp StorageGRID

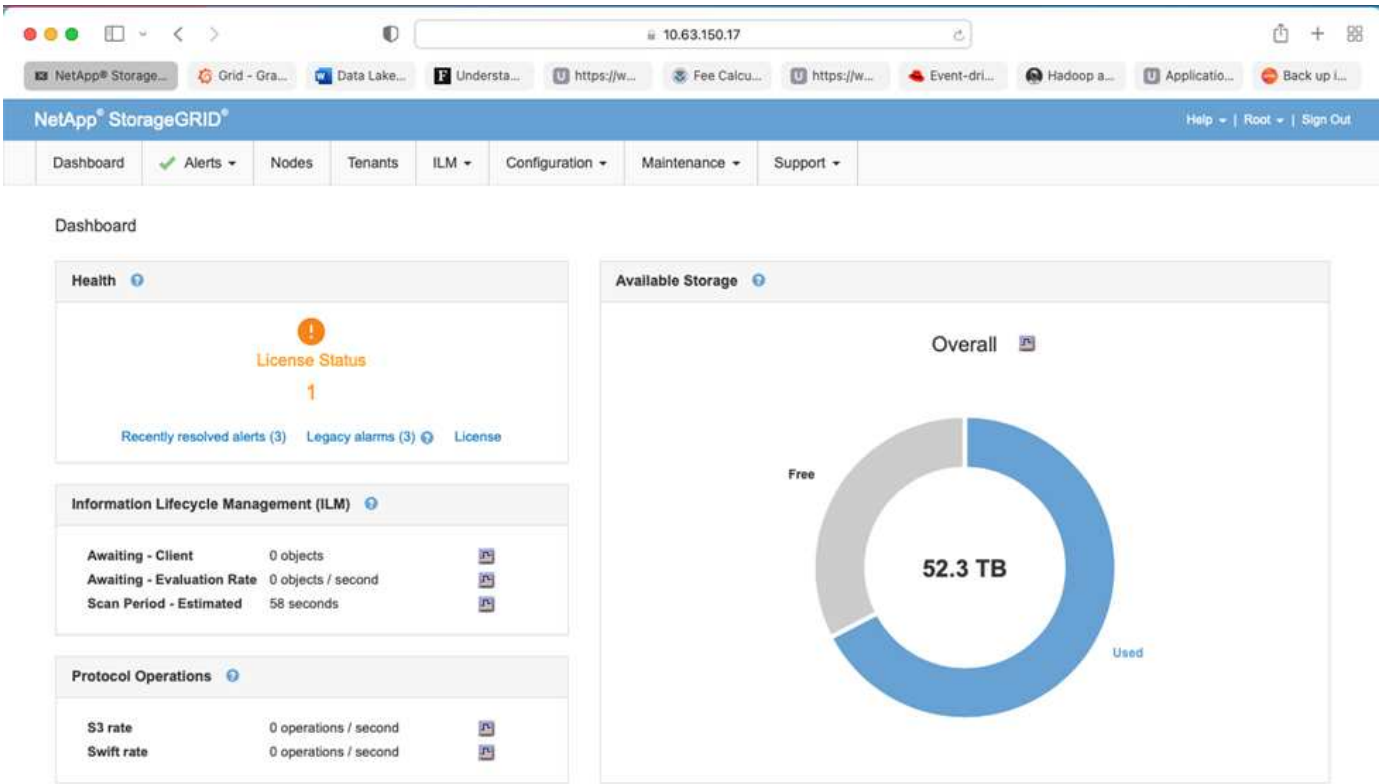
NetApp StorageGRID è una piattaforma di storage a oggetti dalle performance elevate e conveniente. Utilizzando lo storage a più livelli, la maggior parte dei dati su Confluent Kafka, che sono memorizzati nello storage locale o NELLO storage SAN del broker, viene scaricata nell'archivio a oggetti remoto. Questa configurazione comporta significativi miglioramenti operativi riducendo i tempi e i costi per ribilanciare, espandere o ridurre i cluster o sostituire un broker guasto. Lo storage a oggetti svolge un ruolo importante nella gestione dei dati che risiedono nel Tier dell'archivio di oggetti, motivo per cui è importante scegliere lo storage a oggetti giusto.

StorageGRID offre una gestione dei dati globale intelligente e basata su policy utilizzando un'architettura grid distribuita e basata su nodi. Semplifica la gestione di petabyte di dati non strutturati e miliardi di oggetti attraverso il suo onnipresente namespace globale a oggetti combinato con sofisticate funzionalità di gestione dei dati. L'accesso a oggetti a chiamata singola si estende tra i siti e semplifica le architetture ad alta disponibilità garantendo al contempo un accesso continuo agli oggetti, indipendentemente dalle interruzioni del sito o dell'infrastruttura.

La multi-tenancy consente di gestire in modo sicuro più cloud non strutturati e applicazioni dati aziendali all'interno dello stesso grid, aumentando il ROI e i casi di utilizzo per NetApp StorageGRID. Puoi creare diversi livelli di servizio con policy sul ciclo di vita degli oggetti basate sui metadati, ottimizzando durata, protezione, performance e località in più aree geografiche. Gli utenti possono regolare le policy di gestione dei dati, monitorare e applicare i limiti di traffico per riallinearsi con il panorama dei dati senza interruzioni in base al cambiamento dei requisiti in ambienti IT in continua evoluzione.

Gestione semplice con Grid Manager

StorageGRID Grid Manager è un'interfaccia grafica basata su browser che consente di configurare, gestire e monitorare il sistema StorageGRID in ubicazioni distribuite a livello globale in un unico pannello di controllo.



Con l'interfaccia Gestore griglia di StorageGRID è possibile eseguire le seguenti attività:

- Gestisci repository distribuiti a livello globale e scalabili in petabyte di oggetti come immagini, video e record.
- Monitorare i nodi e i servizi di grid per garantire la disponibilità degli oggetti.
- Gestire il posizionamento dei dati a oggetti nel tempo utilizzando le regole ILM (Information Lifecycle Management). Queste regole regolano ciò che accade ai dati di un oggetto dopo l'acquisizione, il modo in cui sono protetti dalla perdita, dove sono memorizzati i dati dell'oggetto e per quanto tempo.
- Monitorare transazioni, performance e operazioni all'interno del sistema.

Policy di Information Lifecycle Management

StorageGRID offre policy di gestione dei dati flessibili che includono la conservazione delle copie di replica degli oggetti e l'utilizzo di schemi EC (erasure coding) come 2+1 e 4+2 (tra gli altri) per memorizzare gli oggetti, a seconda dei requisiti specifici di performance e protezione dei dati. Con il variare dei carichi di lavoro e dei requisiti nel tempo, è comune che anche le policy ILM debbano cambiare nel tempo. La modifica delle policy ILM è una funzionalità fondamentale che consente ai clienti StorageGRID di adattarsi al loro ambiente in continua evoluzione in modo rapido e semplice. Controllare "Policy ILM" e. "Regole ILM" configurazione in StorageGRID.

Performance

StorageGRID scala le performance aggiungendo più nodi di storage, che possono essere macchine virtuali, bare metal o appliance appositamente costruite come "SG5712, SG5760, SG6060 O SGF6024". Nei nostri test, abbiamo superato i requisiti di performance chiave di Apache Kafka con un grid a tre nodi di dimensioni minime utilizzando l'appliance SGF6024. Man mano che i clienti scalano il cluster Kafka con broker aggiuntivi, possono aggiungere più nodi di storage per aumentare performance e capacità.

Bilanciamento del carico e configurazione degli endpoint

I nodi di amministrazione in StorageGRID forniscono l'interfaccia utente (interfaccia utente) e l'endpoint REST API per visualizzare, configurare e gestire il sistema StorageGRID, nonché registri di controllo per tenere traccia dell'attività del sistema. Per fornire un endpoint S3 altamente disponibile per lo storage a più livelli Confluent Kafka, abbiamo implementato il bilanciamento del carico StorageGRID, che viene eseguito come servizio su nodi di amministrazione e nodi gateway. Inoltre, il bilanciamento del carico gestisce anche il traffico locale e comunica con GSLB (Global Server Load Balancing) per agevolare il disaster recovery.

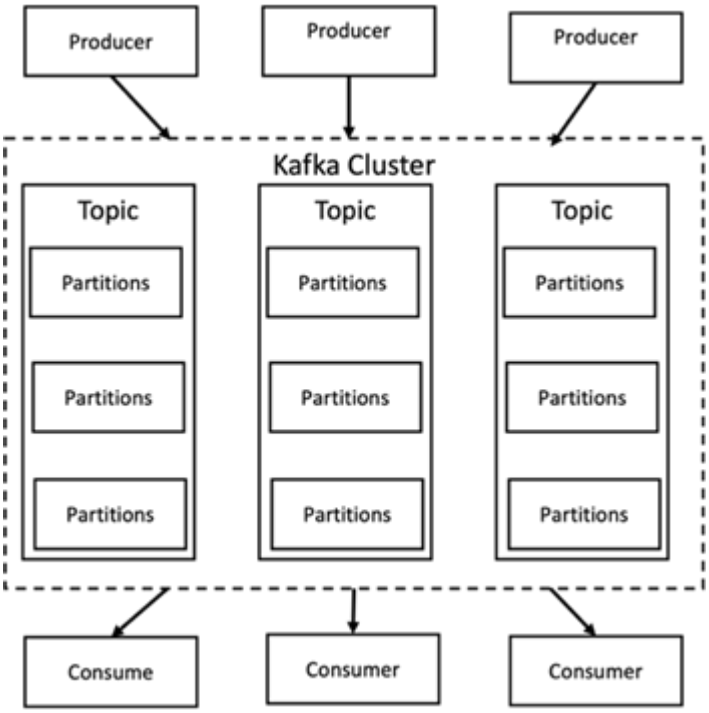
Per migliorare ulteriormente la configurazione degli endpoint, StorageGRID fornisce policy di classificazione del traffico integrate nel nodo di amministrazione, consente di monitorare il traffico dei workload e applica vari limiti di qualità del servizio (QoS) ai carichi di lavoro. I criteri di classificazione del traffico vengono applicati agli endpoint del servizio bilanciamento del carico StorageGRID per i nodi gateway e i nodi di amministrazione. Queste policy possono essere utili per la definizione e il monitoraggio del traffico.

Classificazione del traffico in StorageGRID

StorageGRID dispone di funzionalità QoS integrate. I criteri di classificazione del traffico possono aiutare a monitorare diversi tipi di traffico S3 provenienti da un'applicazione client. È quindi possibile creare e applicare policy per limitare il traffico in base alla larghezza di banda in/out, al numero di richieste simultanee in lettura/scrittura o alla velocità di richiesta in lettura/scrittura.

Apache Kafka

Apache Kafka è un'implementazione framework di un bus software che utilizza l'elaborazione del flusso scritta in Java e Scala. Il suo scopo è fornire una piattaforma unificata, ad alto throughput e a bassa latenza per la gestione dei feed di dati in tempo reale. Kafka può connettersi a un sistema esterno per l'esportazione e l'importazione dei dati tramite Kafka Connect e fornisce Kafka Streams, una libreria di elaborazione del flusso Java. Kafka utilizza un protocollo binario basato su TCP ottimizzato per l'efficienza e basato su un'astrazione "message set" che raggruppa naturalmente i messaggi per ridurre l'overhead del roundtrip di rete. Ciò consente operazioni su disco sequenziali più grandi, pacchetti di rete più grandi e blocchi di memoria contigui, consentendo a Kafka di trasformare un flusso bursty di scritture casuali di messaggi in scritture lineari. La figura seguente mostra il flusso di dati di base di Apache Kafka.



Kafka memorizza i messaggi chiave-valore che provengono da un numero arbitrario di processi chiamati produttori. I dati possono essere suddivisi in partizioni diverse all'interno di diversi argomenti. All'interno di una partizione, i messaggi vengono ordinati in base agli offset (la posizione di un messaggio all'interno di una partizione) e indicizzati e memorizzati insieme a un indicatore data e ora. Altri processi denominati consumer possono leggere i messaggi dalle partizioni. Per l'elaborazione dei flussi, Kafka offre l'API Streams che consente di scrivere applicazioni Java che consumano dati da Kafka e di riscrivere i risultati a Kafka. Apache Kafka funziona anche con sistemi di elaborazione del flusso esterni come Apache Apex, Apache Flink, Apache Spark, Apache Storm e Apache NiFi.

Kafka viene eseguito su un cluster di uno o più server (chiamati broker) e le partizioni di tutti gli argomenti vengono distribuite tra i nodi del cluster. Inoltre, le partizioni vengono replicate su più broker. Questa architettura consente a Kafka di inviare flussi di messaggi enormi in modo fault-tolerant e ha consentito al reparto IT di sostituire alcuni dei sistemi di messaggistica convenzionali come Java message Service (JMS), Advanced message Queuing Protocol (AMQP) e così via. A partire dalla release 0.11.0.0, Kafka offre scritture transazionali, che forniscono un'elaborazione del flusso esattamente una volta usando l'API Streams.

Kafka supporta due tipi di argomenti: Regolare e compatto. Gli argomenti regolari possono essere configurati con un tempo di conservazione o un limite di spazio. Se ci sono record più vecchi del tempo di conservazione specificato o se viene superato il limite di spazio per una partizione, Kafka può eliminare i vecchi dati per liberare spazio di storage. Per impostazione predefinita, gli argomenti sono configurati con un tempo di conservazione di 7 giorni, ma è anche possibile memorizzare i dati a tempo indeterminato. Per gli argomenti compattati, i record non scadono in base ai limiti di tempo o spazio. Kafka considera invece i messaggi successivi come aggiornamenti dei messaggi meno recenti con la stessa chiave e garantisce di non eliminare mai l'ultimo messaggio per chiave. Gli utenti possono eliminare completamente i messaggi scrivendo un cosiddetto messaggio tombstone con il valore null per una chiave specifica.

Kafka offre cinque API principali:

- **Producer API.** consente a un'applicazione di pubblicare flussi di record.
- **API Consumer.** consente a un'applicazione di iscriversi ad argomenti ed elaborare flussi di record.
- **Connector API.** esegue le API riutilizzabili di produttori e consumatori che possono collegare gli argomenti alle applicazioni esistenti.
- **Streams API.** questa API converte i flussi di input in output e produce il risultato.
- **Admin API.** utilizzato per gestire argomenti Kafka, broker e altri oggetti Kafka.

Le API consumer e Producer si basano sul protocollo di messaggistica Kafka e offrono un'implementazione di riferimento per i clienti consumer e Producer Kafka in Java. Il protocollo di messaging sottostante è un protocollo binario che gli sviluppatori possono utilizzare per scrivere i propri client consumer o Producer in qualsiasi linguaggio di programmazione. In questo modo, Kafka viene sbloccato dall'ecosistema JVM (Java Virtual Machine). Un elenco di client non Java disponibili viene mantenuto nel wiki Apache Kafka.

Casi di utilizzo di Apache Kafka

Apache Kafka è più popolare per la messaggistica, il monitoraggio delle attività dei siti Web, le metriche, l'aggregazione dei log, l'elaborazione dei flussi, sourcing degli eventi e registrazione del commit.

- Kafka ha migliorato il throughput, il partizionamento integrato, la replica e la tolleranza agli errori, il che lo rende una buona soluzione per le applicazioni di elaborazione dei messaggi su larga scala.
- Kafka può ricostruire le attività di un utente (visualizzazioni di pagine, ricerche) in una pipeline di monitoraggio come un insieme di feed di iscrizione alla pubblicazione in tempo reale.
- Kafka viene spesso utilizzato per il monitoraggio dei dati operativi. Ciò comporta l'aggregazione di statistiche da applicazioni distribuite per produrre feed centralizzati di dati operativi.

- Molte persone utilizzano Kafka come sostituto di una soluzione di aggregazione dei log. L'aggregazione dei log generalmente raccoglie i file di log fisici dai server e li colloca in una posizione centrale (ad esempio, un file server o HDFS) per l'elaborazione. Kafka astratta i dettagli dei file e fornisce un'astrazione più pulita dei dati di log o degli eventi come flusso di messaggi. Ciò consente un'elaborazione a latenza ridotta e un supporto più semplice per più origini dati e un consumo di dati distribuito.
- Molti utenti di Kafka elaborano i dati in pipeline di elaborazione costituite da più fasi, in cui i dati di input raw vengono utilizzati da argomenti di Kafka e quindi aggregati, arricchiti o altrimenti trasformati in nuovi argomenti per un ulteriore consumo o un'elaborazione di follow-up. Ad esempio, una pipeline di elaborazione per consigliare articoli di notizie potrebbe strisciare il contenuto degli articoli dai feed RSS e pubblicarlo in un argomento "articoli". Un'ulteriore elaborazione potrebbe normalizzare o deduplicare questo contenuto e pubblicare il contenuto pulito dell'articolo su un nuovo argomento, mentre una fase finale di elaborazione potrebbe tentare di consigliare questo contenuto agli utenti. Tali pipeline di elaborazione creano grafici dei flussi di dati in tempo reale in base ai singoli argomenti.
- L'origine degli eventi è uno stile di progettazione dell'applicazione per cui le modifiche di stato vengono registrate come una sequenza di record ordinata in base al tempo. Il supporto di Kafka per i dati di log memorizzati di grandi dimensioni lo rende un eccellente backend per un'applicazione costruita in questo stile.
- Kafka può fungere da commit-log esterno per un sistema distribuito. Il log consente di replicare i dati tra i nodi e funge da meccanismo di risyncing per i nodi guasti per il ripristino dei dati. La funzione di compattazione del log di Kafka aiuta a supportare questo caso d'utilizzo.

Confluente

Confluent Platform è una piattaforma Enterprise-ready che completa Kafka con funzionalità avanzate progettate per accelerare lo sviluppo e la connettività delle applicazioni, consentire trasformazioni attraverso l'elaborazione del flusso, semplificare le operazioni aziendali su larga scala e soddisfare rigorosi requisiti architetturali. Creato dai creatori originali di Apache Kafka, Confluent amplia i vantaggi di Kafka con funzionalità di livello Enterprise, eliminando al contempo il peso della gestione o del monitoraggio di Kafka. Oggi, oltre il 80% delle aziende Fortune 100 è basato su tecnologia di streaming dei dati, e la maggior parte di esse utilizza Confluent.

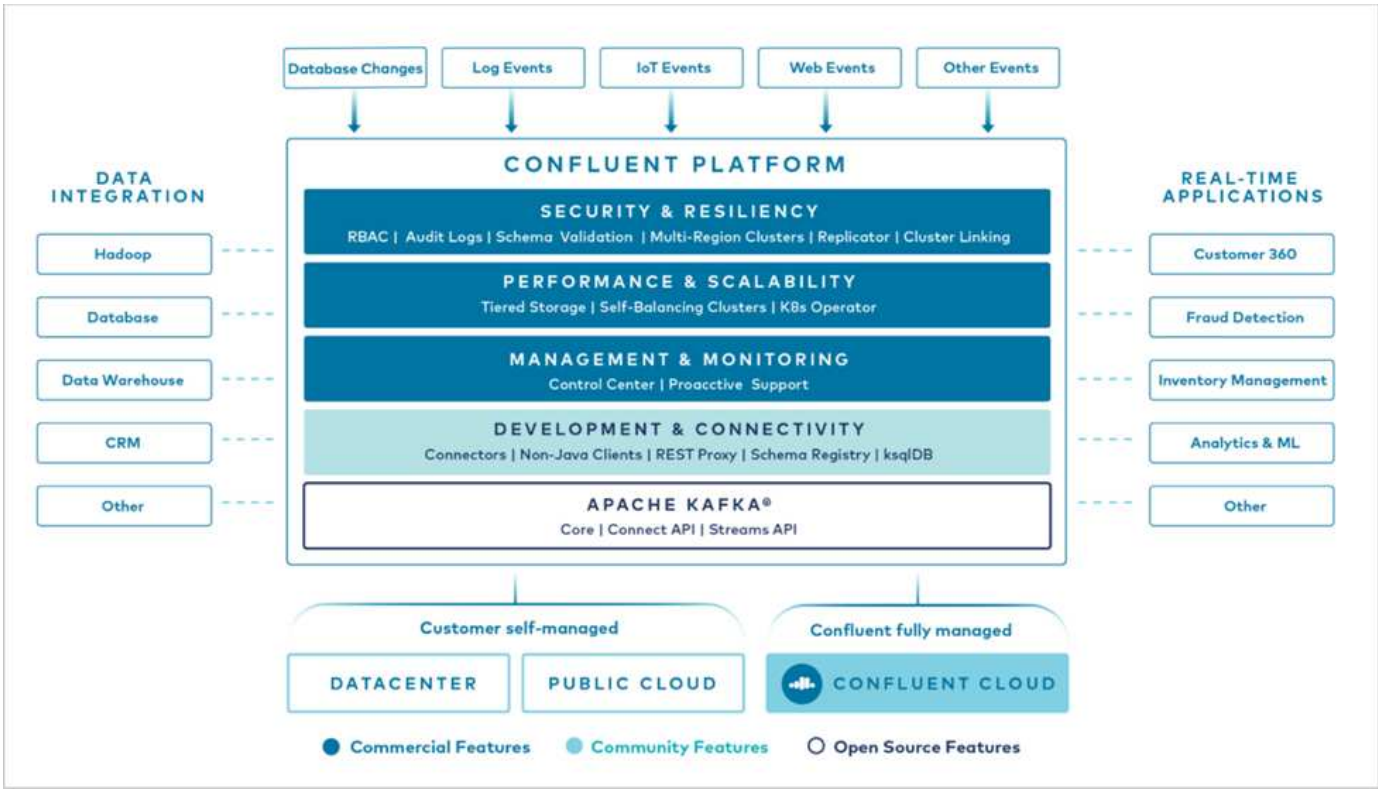
Perché confluente?

Integrando dati storici e in tempo reale in un'unica fonte di verità centrale, Confluent semplifica la creazione di una categoria completamente nuova di applicazioni moderne e basate sugli eventi, l'acquisizione di una pipeline universale di dati e lo sblocco di nuovi casi di utilizzo potenti con scalabilità, performance e affidabilità complete.

A cosa serve Confluent?

Confluent Platform ti consente di concentrarti su come ricavare il valore di business dai tuoi dati piuttosto che preoccuparsi delle meccaniche sottostanti, come ad esempio il modo in cui i dati vengono trasportati o integrati tra sistemi diversi. In particolare, Confluent Platform semplifica la connessione delle origini dati a Kafka, la creazione di applicazioni di streaming e la protezione, il monitoraggio e la gestione dell'infrastruttura Kafka. Attualmente, Confluent Platform viene utilizzata per un'ampia gamma di casi di utilizzo in numerosi settori, dai servizi finanziari alla vendita al dettaglio, alle auto autonome, al rilevamento delle frodi, Microservizi e IoT.

La figura seguente mostra i componenti della piattaforma Confluent Kafka.



Panoramica della tecnologia di streaming degli eventi di Confluent

Il fulcro della piattaforma confluente è "Apache Kafka", la piattaforma di streaming distribuito open-source più diffusa. Le principali funzionalità di Kafka sono le seguenti:

- Pubblicare e sottoscrivere flussi di record.
- Memorizzare i flussi di record in modo tollerante agli errori.
- Elaborazione di flussi di record.

Confluent Platform include anche il Registro di sistema dello schema, il proxy REST, oltre 100 connettori Kafka preintegrati e ksqlDB.

Panoramica delle funzionalità aziendali della piattaforma Confluent

- **Confluent Control Center.** sistema basato su GUI per la gestione e il monitoraggio di Kafka. Consente di gestire facilmente Kafka Connect e creare, modificare e gestire le connessioni ad altri sistemi.
- **Confluent per Kubernetes.** Confluent per Kubernetes è un operatore di Kubernetes. Gli operatori di Kubernetes estendono le funzionalità di orchestrazione di Kubernetes fornendo funzionalità e requisiti unici per una specifica applicazione della piattaforma. Per Confluent Platform, ciò include una notevole semplificazione del processo di implementazione di Kafka su Kubernetes e l'automazione delle attività tipiche del ciclo di vita dell'infrastruttura.
- **Connettori confluenti verso Kafka.** i connettori utilizzano l'API Kafka Connect per connettere Kafka ad altri sistemi come database, archivi di valori chiave, indici di ricerca e file system. Confluent Hub dispone di connettori scaricabili per le fonti di dati e i sink più diffusi, incluse le versioni completamente testate e supportate di questi connettori con Confluent Platform. Ulteriori dettagli sono disponibili "qui".
- **Cluster con bilanciamento automatico.** offre bilanciamento del carico automatico, rilevamento degli errori e riparazione automatica. Fornisce supporto per l'aggiunta o la disattivazione di broker in base alle necessità, senza tuning manuale.

- **Collegamento di cluster confluente.** collega direttamente i cluster e esegue il mirroring degli argomenti da un cluster all'altro tramite un bridge di collegamento. Il collegamento dei cluster semplifica la configurazione di implementazioni di cloud ibrido, multi-cluster e multi-data center.
- **Confluent auto data balancer.** monitora il cluster per il numero di broker, la dimensione delle partizioni, il numero di partizioni e il numero di leader all'interno del cluster. Consente di spostare i dati per creare un carico di lavoro uniforme nel cluster, riducendo al contempo il ribilanciamento del traffico per ridurre al minimo l'effetto sui carichi di lavoro di produzione durante il ribilanciamento.
- **Confluent Replicator.** semplifica la gestione di più cluster Kafka in più data center.
- **Tiered storage.** offre opzioni per l'archiviazione di grandi volumi di dati Kafka utilizzando il tuo cloud provider preferito, riducendo così il carico operativo e i costi. Con lo storage a più livelli, puoi mantenere i dati su uno storage a oggetti conveniente e scalare i broker solo quando hai bisogno di più risorse di calcolo.
- **Confluent JMS client.** Confluent Platform include un client compatibile con JMS per Kafka. Questo client Kafka implementa l'API standard JMS 1.1, utilizzando i broker Kafka come backend. Questo è utile se si utilizzano applicazioni legacy con JMS e si desidera sostituire il message broker JMS esistente con Kafka.
- **Il proxy MQTT confluente.** offre un modo per pubblicare i dati direttamente su Kafka da dispositivi e gateway MQTT senza la necessità di un broker MQTT al centro.
- **I plug-in di sicurezza confluenti.** i plug-in di sicurezza confluenti vengono utilizzati per aggiungere funzionalità di sicurezza a vari strumenti e prodotti della piattaforma confluente. Attualmente, è disponibile un plug-in per il proxy REST confluente che consente di autenticare le richieste in entrata e propagare l'identità autenticata alle richieste a Kafka. Ciò consente ai client proxy REST confluenti di utilizzare le funzionalità di sicurezza multi-tenant del broker Kafka.

Verifica confluente

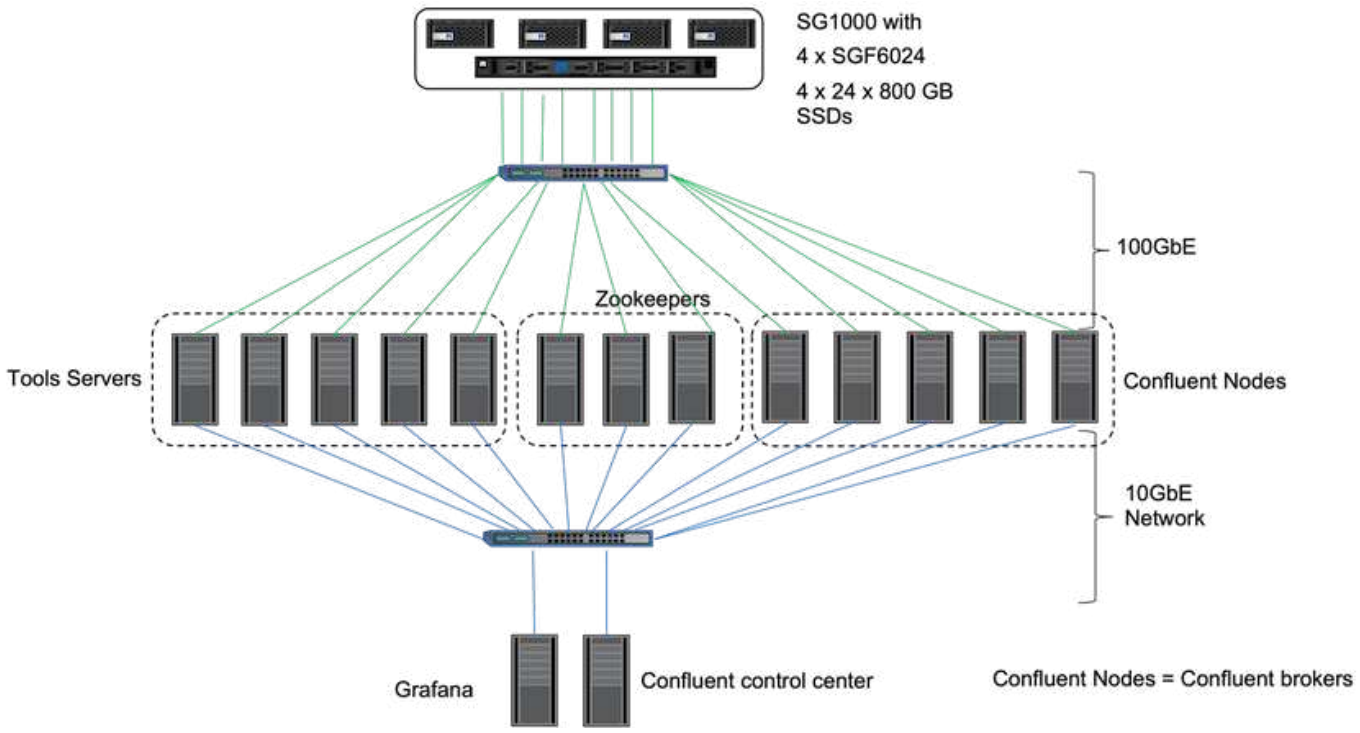
Abbiamo eseguito la verifica con la piattaforma confluente 6.2 Tiered Storage in NetApp StorageGRID. I team NetApp e Confluent hanno lavorato insieme a questa verifica ed hanno eseguito i casi di test richiesti per la verifica.

Configurazione della piattaforma confluente

Per la verifica abbiamo utilizzato la seguente configurazione.

Per la verifica, abbiamo utilizzato tre zookeeper, cinque broker, cinque server di esecuzione script di test, server named tools con 256 GB di RAM e 16 CPU. Per lo storage NetApp, abbiamo utilizzato StorageGRID con un bilanciamento del carico SG1000 con quattro SGF6024. Lo storage e i broker erano connessi tramite connessioni 100GbE.

La figura seguente mostra la topologia di rete della configurazione utilizzata per la verifica confluente.



I server degli strumenti fungono da client applicativi che inviano richieste ai nodi confluenti.

Configurazione dello storage a più livelli confluente

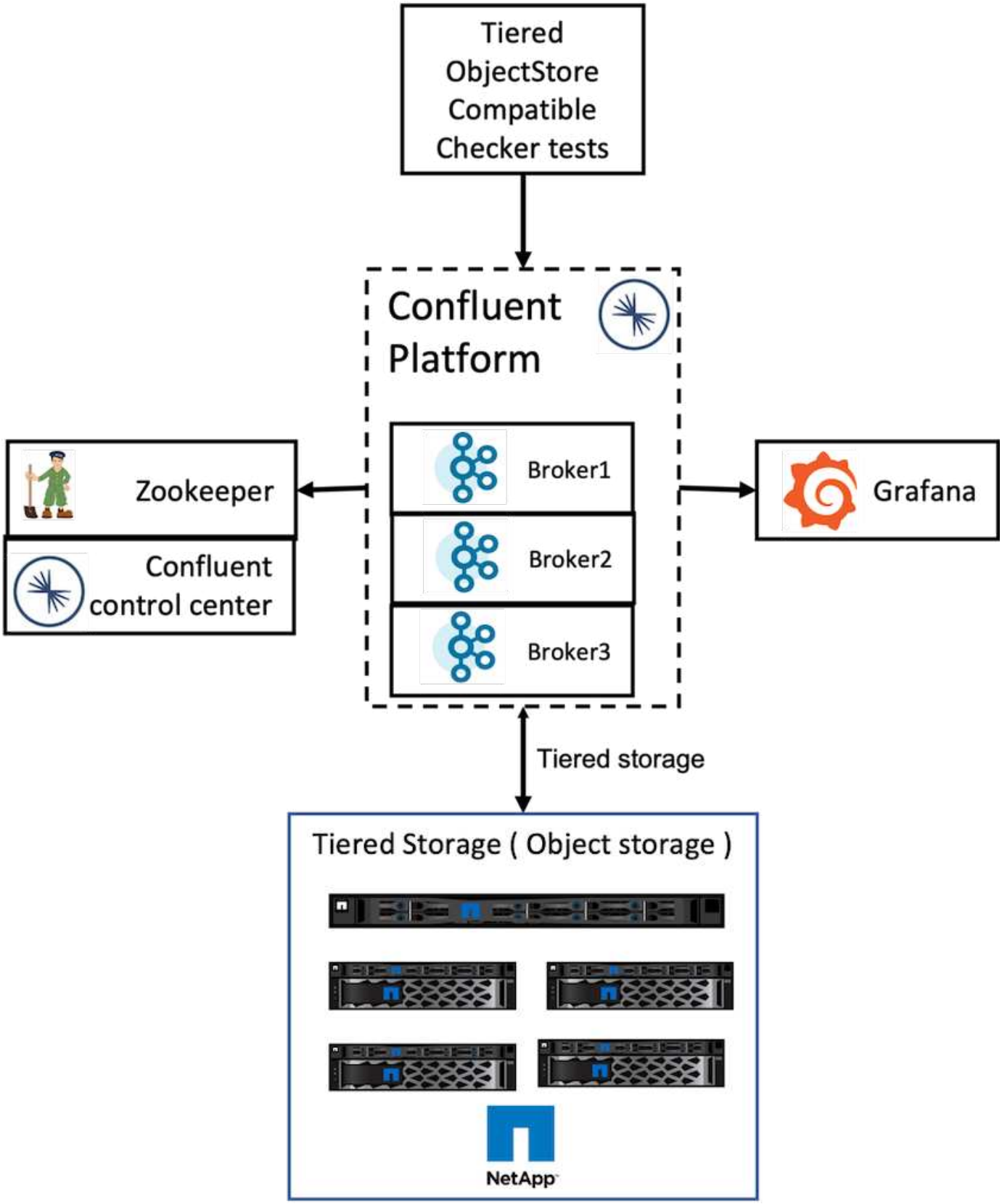
La configurazione dello storage a più livelli richiede i seguenti parametri in Kafka:

```
Confluent.tier.archiver.num.threads=16
confluent.tier.fetcher.num.threads=32
confluent.tier.enable=true
confluent.tier.feature=true
confluent.tier.backend=S3
confluent.tier.s3.bucket=kafkasgdbucket1-2
confluent.tier.s3.region=us-west-2
confluent.tier.s3.cred.file.path=/data/kafka/.ssh/credentials
confluent.tier.s3.aws.endpoint.override=http://kafkasgd.rtpppe.netapp.com:
10444/
confluent.tier.s3.force.path.style.access=true
```

Per la verifica, abbiamo utilizzato StorageGRID con il protocollo HTTP, ma funziona anche HTTPS. La chiave di accesso e la chiave segreta vengono memorizzate nel nome file fornito in `confluent.tier.s3.cred.file.path` parametro.

Storage a oggetti NetApp - StorageGRID

Abbiamo configurato la configurazione a sito singolo in StorageGRID per la verificazione.



Test di verifica

Abbiamo completato i seguenti cinque casi di test per la verifica. Questi test vengono eseguiti sul framework Trogdor. I primi due erano test di funzionalità e i restanti tre erano test di performance.

Test di correttezza dell'archivio di oggetti

Questo test determina se tutte le operazioni di base (ad esempio, GET/put/delete) sull'API dell'archivio di oggetti funzionano correttamente in base alle esigenze dello storage su più livelli. Si tratta di un test di base che ogni servizio dell'archivio di oggetti dovrebbe superare prima dei seguenti test. Si tratta di un test assertivo che può essere superato o non superato.

Test di correttezza delle funzionalità di tiering

Questo test determina se la funzionalità di storage a più livelli end-to-end funziona correttamente con un test assertivo che supera o non supera. Il test crea un argomento di test che per impostazione predefinita è configurato con il tiering attivato e una dimensione hotset altamente ridotta. Produce un flusso di eventi per l'argomento di test appena creato, attende che i broker archivino i segmenti nell'archivio di oggetti, quindi consuma il flusso di eventi e convalida che il flusso consumato corrisponda al flusso prodotto. Il numero di messaggi prodotti per il flusso di eventi è configurabile, il che consente all'utente di generare un carico di lavoro sufficientemente grande in base alle esigenze di test. La dimensione ridotta degli hotset garantisce che i fetch consumer al di fuori del segmento attivo vengano serviti solo dall'archivio di oggetti; questo aiuta a verificare la correttezza dell'archivio di oggetti per le letture. Questo test è stato eseguito con e senza un'iniezione di errori dello store di oggetti. Abbiamo simulato il guasto del nodo arrestando il servizio di gestione dei servizi in uno dei nodi in StorageGRID e convalidando che la funzionalità end-to-end funziona con lo storage a oggetti.

Benchmark Tier fetch

Questo test ha validato le prestazioni di lettura dello storage a più livelli e verificato l'intervallo di richieste di lettura di recupero sotto carico pesante dai segmenti generati dal benchmark. In questo benchmark, Confluent ha sviluppato client personalizzati per soddisfare le richieste di recupero del Tier.

Benchmark sui carichi di lavoro producete-consumate

Questo test ha generato indirettamente il carico di lavoro di scrittura nell'archivio di oggetti attraverso l'archiviazione dei segmenti. Il carico di lavoro di lettura (segmenti letti) è stato generato dallo storage a oggetti quando i gruppi di consumatori hanno recuperato i segmenti. Questo carico di lavoro è stato generato dallo script di test. Questo test ha verificato le prestazioni di lettura e scrittura sullo storage a oggetti in thread paralleli. Abbiamo eseguito test con e senza l'iniezione di errori del negozio di oggetti come abbiamo fatto per il test di correttezza della funzionalità di tiering.

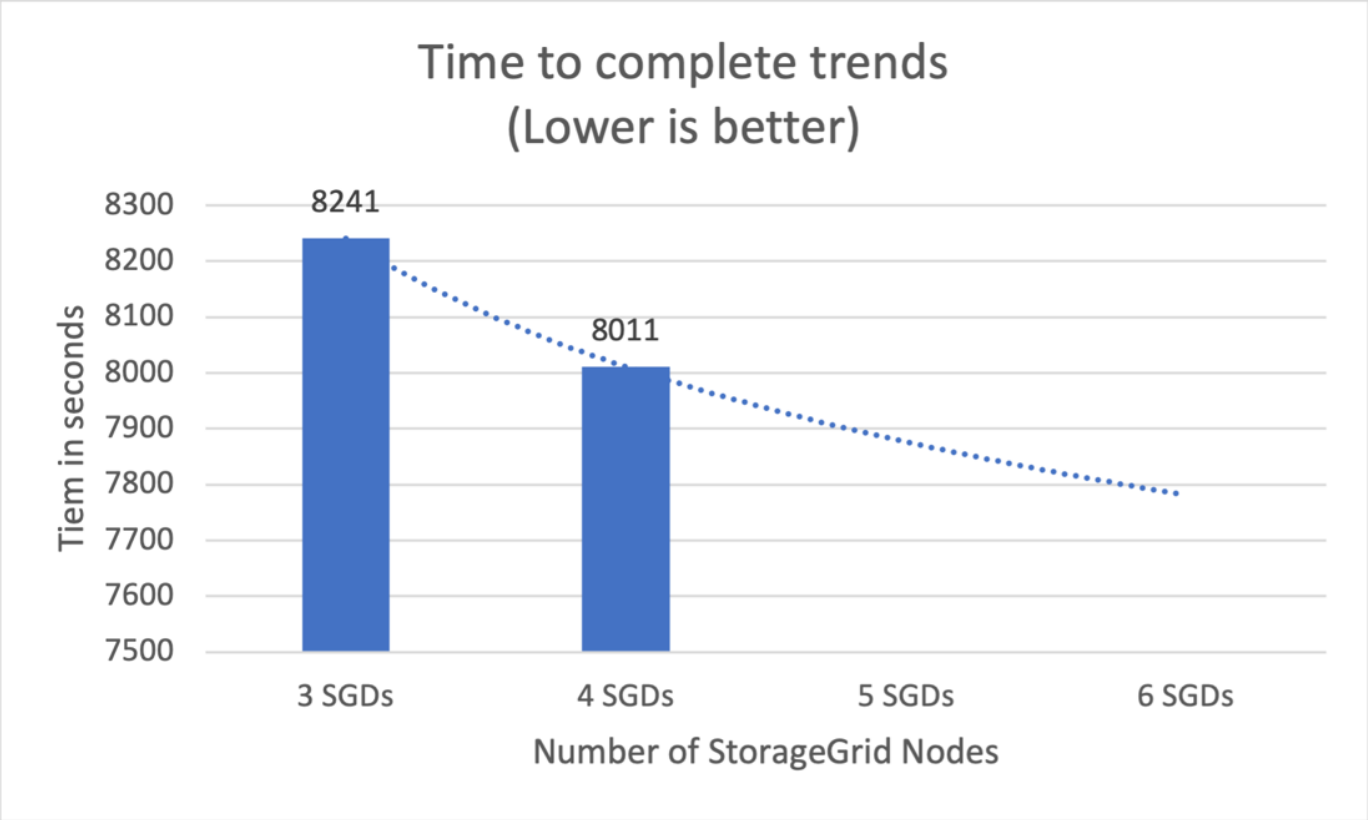
Benchmark del carico di lavoro di conservazione

Questo test ha controllato le prestazioni di eliminazione di un archivio di oggetti con un carico di lavoro pesante di conservazione degli argomenti. Il carico di lavoro di conservazione è stato generato utilizzando uno script di test che produce molti messaggi in parallelo a un argomento di test. L'argomento del test è stato configurato con un'impostazione di conservazione aggressiva basata sulle dimensioni e sul tempo che ha causato la rimozione continua del flusso di eventi dall'archivio di oggetti. I segmenti sono stati quindi archiviati. Ciò ha portato a un gran numero di eliminazioni nello storage a oggetti da parte del broker e alla raccolta delle performance delle operazioni di eliminazione degli archivi di oggetti.

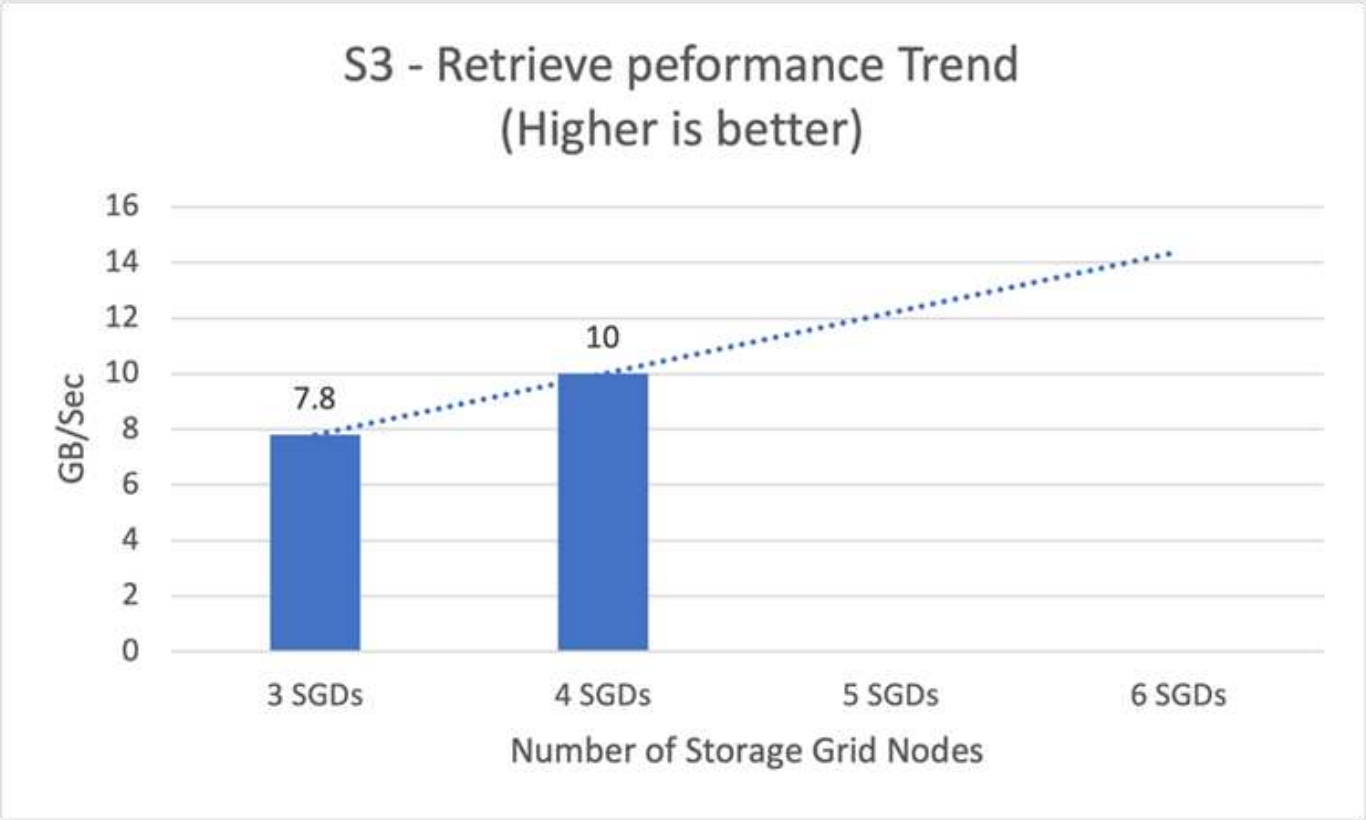
Test delle performance con scalabilità

Con la configurazione di NetApp StorageGRID, abbiamo eseguito il test dello storage su più livelli con da tre a quattro nodi per carichi di lavoro consumer e di produttori. Secondo i nostri test, il tempo di completamento e i risultati delle performance erano direttamente proporzionali al numero di nodi StorageGRID. L'installazione di StorageGRID richiedeva almeno tre nodi.

- Il tempo necessario per completare le operazioni di produzione e di consumo è diminuito in modo lineare con l'aumento del numero di nodi di storage.



- Le performance per l'operazione di recupero s3 sono aumentate linearmente in base al numero di nodi StorageGRID. StorageGRID supporta fino a 200 nodi StorgeGRID.



Connettore s3 confluente

Amazon S3 Sink Connector esporta i dati dagli argomenti di Apache Kafka in oggetti S3 nei formati Avro, JSON o Bytes. Amazon S3 sink Connector esegue periodicamente il polling dei dati da Kafka e a sua volta li carica in S3. Un partitioner viene utilizzato per suddividere i dati di ogni partizione Kafka in blocchi. Ogni blocco di dati viene rappresentato come un oggetto S3. Il nome della chiave codifica l'argomento, la partizione Kafka e l'offset iniziale di questo blocco di dati.

In questa configurazione, viene illustrato come leggere e scrivere argomenti nello storage a oggetti da Kafka direttamente utilizzando il connettore del sink Kafka s3. Per questo test, abbiamo utilizzato un cluster Confluent autonomo, ma questa configurazione è applicabile a un cluster distribuito.

1. Scarica Confluent Kafka dal sito Web di Confluent.
2. Disimballare il pacchetto in una cartella sul server.
3. Esportare due variabili.

```
Export CONFLUENT_HOME=/data/confluent/confluent-6.2.0
export PATH=$PATH:/data/confluent/confluent-6.2.0/bin
```

4. Per un'installazione autonoma di Confluent Kafka, il cluster crea una cartella root temporanea in `/tmp`. Inoltre, crea Zoosekeeper, Kafka, un registro dello schema, Connect, un server ksql, e control center da cui copiare i rispettivi file di configurazione `$CONFLUENT_HOME`. Vedere il seguente esempio:

```
root@stlrx2540m1-108:~# ls -ltr /tmp/confluent.406980/
total 28
drwxr-xr-x 4 root root 4096 Oct 29 19:01 zookeeper
drwxr-xr-x 4 root root 4096 Oct 29 19:37 kafka
drwxr-xr-x 4 root root 4096 Oct 29 19:40 schema-registry
drwxr-xr-x 4 root root 4096 Oct 29 19:45 kafka-rest
drwxr-xr-x 4 root root 4096 Oct 29 19:47 connect
drwxr-xr-x 4 root root 4096 Oct 29 19:48 ksql-server
drwxr-xr-x 4 root root 4096 Oct 29 19:53 control-center
root@stlrx2540m1-108:~#
```

5. Configurare Zoosekeeper. Non è necessario modificare nulla se si utilizzano i parametri predefiniti.

```
root@stlrx2540m1-108:~# cat
/tmp/confluent.406980/zookeeper/zookeeper.properties  | grep -iv ^#
dataDir=/tmp/confluent.406980/zookeeper/data
clientPort=2181
maxClientCnxns=0
admin.enableServer=false
tickTime=2000
initLimit=5
syncLimit=2
server.179=controlcenter:2888:3888
root@stlrx2540m1-108:~#
```

Nella configurazione precedente, è stato aggiornato il `server. xxx` proprietà. Per impostazione predefinita, sono necessari tre Zooseeper per la selezione dei leader Kafka.

6. Abbiamo creato un file myid in `/tmp/confluent.406980/zookeeper/data` Con un ID univoco:

```
root@stlrx2540m1-108:~# cat /tmp/confluent.406980/zookeeper/data/myid
179
root@stlrx2540m1-108:~#
```

Abbiamo utilizzato l'ultimo numero di indirizzi IP per il file myid. Abbiamo utilizzato i valori predefiniti per Kafka, CONNECT, Control-Center, Kafka, Kafka-REST, configurazioni del server ksql e del registro di sistema dello schema.

7. Avviare i servizi Kafka.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  start
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
ZooKeeper is [UP]
Kafka is [UP]
Schema Registry is [UP]
Kafka REST is [UP]
Connect is [UP]
ksqlDB Server is [UP]
Control Center is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

Per ciascuna configurazione è disponibile una cartella di log che consente di risolvere i problemi. In alcuni casi, l'avvio dei servizi richiede più tempo. Assicurarsi che tutti i servizi siano attivi e in esecuzione.

8. Installare Kafka Connect utilizzando `confluent-hub`.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# ./confluent-
hub install confluentinc/kafka-connect-s3:latest
The component can be installed in any of the following Confluent
Platform installations:
  1. /data/confluent/confluent-6.2.0 (based on $CONFLUENT_HOME)
  2. /data/confluent/confluent-6.2.0 (where this tool is installed)
Choose one of these to continue the installation (1-2): 1
Do you want to install this into /data/confluent/confluent-
6.2.0/share/confluent-hub-components? (yN) y

Component's license:
Confluent Community License
http://www.confluent.io/confluent-community-license
I agree to the software license agreement (yN) y
Downloading component Kafka Connect S3 10.0.3, provided by Confluent,
Inc. from Confluent Hub and installing into /data/confluent/confluent-
6.2.0/share/confluent-hub-components
Do you want to uninstall existing version 10.0.3? (yN) y
Detected Worker's configs:
  1. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  2. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  3. Standard: /data/confluent/confluent-6.2.0/etc/schema-
```

```
registry/connect-avro-distributed.properties
  4. Standard: /data/confluent/confluent-6.2.0/etc/schema-
registry/connect-avro-standalone.properties
  5. Based on CONFLUENT_CURRENT:
/tmp/confluent.406980/connect/connect.properties
  6. Used by Connect process with PID 15904:
/tmp/confluent.406980/connect/connect.properties
Do you want to update all detected configs? (yN) y
Adding installation directory to plugin path in the following files:
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
standalone.properties
  /tmp/confluent.406980/connect/connect.properties
  /tmp/confluent.406980/connect/connect.properties

Completed
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

È inoltre possibile installare una versione specifica utilizzando `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Per impostazione predefinita, `confluentinc-kafka-connect-s3` è installato in `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Aggiornare il percorso del plug-in con il nuovo `confluentinc-kafka-connect-s3`.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/etc/kafka/connect-distributed.properties | grep plugin.path
#
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/co
nnectors,
plugin.path=/usr/share/java,/data/zookeeper/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3
root@stlrx2540m1-108:~#
```

11. Arrestare e riavviare i servizi confluenti.

```
confluent local services  stop
confluent local services  start
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  status
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
Connect is [UP]
Control Center is [UP]
Kafka is [UP]
Kafka REST is [UP]
ksqlDB Server is [UP]
Schema Registry is [UP]
ZooKeeper is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

12. Configurare l'ID di accesso e la chiave segreta in `/root/.aws/credentials` file.

```
root@stlrx2540m1-108:~# cat /root/.aws/credentials
[default]
aws_access_key_id = xxxxxxxxxxxx
aws_secret_access_key = xxxxxxxxxxxxxxxxxxxxxxxxx
root@stlrx2540m1-108:~#
```

13. Verificare che il bucket sia raggiungibile.

```
root@stlrx2540m4-01:~# aws s3 –endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls kafkasgdbucket1-2
2021-10-29 21:04:18       1388 1
2021-10-29 21:04:20       1388 2
2021-10-29 21:04:22       1388 3
root@stlrx2540m4-01:~#
```

14. Configurare il file di proprietà s3-sink per la configurazione s3 e bucket.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-
s3/etc/quickstart-s3.properties | grep -v ^#
name=s3-sink
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=1
topics=s3_testtopic
s3.region=us-west-2
s3.bucket.name=kafkasgdbucket1-2
store.url=http://kafkasgd.rtpppe.netapp.com:10444/
s3.part.size=5242880
flush.size=3
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.avro.AvroFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartit
ioner
schema.compatibility=NONE
root@stlrx2540m1-108:~#
```

15. Importare alcuni record nel bucket s3.

```
kafka-avro-console-producer --broker-list localhost:9092 --topic
s3_topic \
--property
value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1",
"type":"string"}]}'
{"f1": "value1"}
{"f1": "value2"}
{"f1": "value3"}
{"f1": "value4"}
{"f1": "value5"}
{"f1": "value6"}
{"f1": "value7"}
{"f1": "value8"}
{"f1": "value9"}
```

16. Caricare il connettore s3-sink.

```
root@stlrx2540m1-108:~# confluent local services connect connector load
s3-sink  --config /data/confluent/confluent-6.2.0/share/confluent-hub-
components/confluentinc-kafka-connect-s3/etc/quickstart-s3.properties
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "config": {
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "flush.size": "3",
    "format.class": "io.confluent.connect.s3.format.avro.AvroFormat",
    "partitioner.class":
"io.confluent.connect.storage.partitioner.DefaultPartitioner",
    "s3.bucket.name": "kafkasgdbucket1-2",
    "s3.part.size": "5242880",
    "s3.region": "us-west-2",
    "schema.compatibility": "NONE",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "store.url": "http://kafkasgd.rtpppe.netapp.com:10444/",
    "tasks.max": "1",
    "topics": "s3_testtopic",
    "name": "s3-sink"
  },
  "tasks": [],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

17. Controllare lo stato del sink s3.

```
root@stlrx2540m1-108:~# confluent local services connect connector
status s3-sink
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "connector": {
    "state": "RUNNING",
    "worker_id": "10.63.150.185:8083"
  },
  "tasks": [
    {
      "id": 0,
      "state": "RUNNING",
      "worker_id": "10.63.150.185:8083"
    }
  ],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

18. Controllare il registro per assicurarsi che s3-sink sia pronto ad accettare gli argomenti.

```
root@stlrx2540m1-108:~# confluent local services connect log
```

19. Consulta gli argomenti di Kafka.

```
kafka-topics --list --bootstrap-server localhost:9092
…
connect-configs
connect-offsets
connect-statuses
default_ksql_processing_log
s3_testtopic
s3_topic
s3_topic_new
root@stlrx2540m1-108:~#
```

20. Controllare gli oggetti nel bucket s3.

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls --recursive kafkasgdbucket1-
2/topics/
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000003.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000006.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000009.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000012.avro
2021-10-29 21:24:09        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000015.avro
root@stlrx2540m1-108:~#
```

21. Per verificare il contenuto, copiare ciascun file da S3 al file system locale eseguendo il seguente comando:

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 cp s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
tes.avro
download: s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro to
./tes.avro
root@stlrx2540m1-108:~#
```

22. Per stampare i record, utilizzare avro-tools-1.11.0.1.jar (disponibile in "Archivi Apache").

```
root@stlrx2540m1-108:~# java -jar /usr/src/avro-tools-1.11.0.1.jar
tojson tes.avro
21/10/30 00:20:24 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
{"f1":"value1"}
{"f1":"value2"}
{"f1":"value3"}
root@stlrx2540m1-108:~#
```

Clusters a bilanciamento automatico confluente

Se hai già gestito un cluster Kafka in precedenza, probabilmente conosci le sfide legate

alla riassegnazione manuale delle partizioni a diversi broker per garantire che il carico di lavoro sia bilanciato in tutto il cluster. Per le organizzazioni con implementazioni Kafka di grandi dimensioni, rimescolare grandi quantità di dati può essere scoraggiante, noioso e rischioso, soprattutto se le applicazioni mission-critical sono costruite sul cluster. Tuttavia, anche per i più piccoli casi di utilizzo di Kafka, il processo richiede tempo e può essere soggetto a errori umani.

Nel nostro laboratorio, abbiamo testato la funzionalità dei cluster di bilanciamento automatico Confluent, che automatizza il ribilanciamento in base alle modifiche della topologia dei cluster o al carico non uniforme. Il test di ribilanciamento confluente consente di misurare il tempo necessario per aggiungere un nuovo broker quando un guasto al nodo o il nodo di scalabilità richiede il ribilanciamento dei dati tra broker. Nelle configurazioni Kafka classiche, la quantità di dati da ribilanciare aumenta con la crescita del cluster, ma, nello storage a più livelli, il ribilanciamento è limitato a una piccola quantità di dati. In base alla nostra convalida, il ribilanciamento dello storage a più livelli richiede secondi o minuti in una classica architettura Kafka e cresce linearmente con la crescita del cluster.

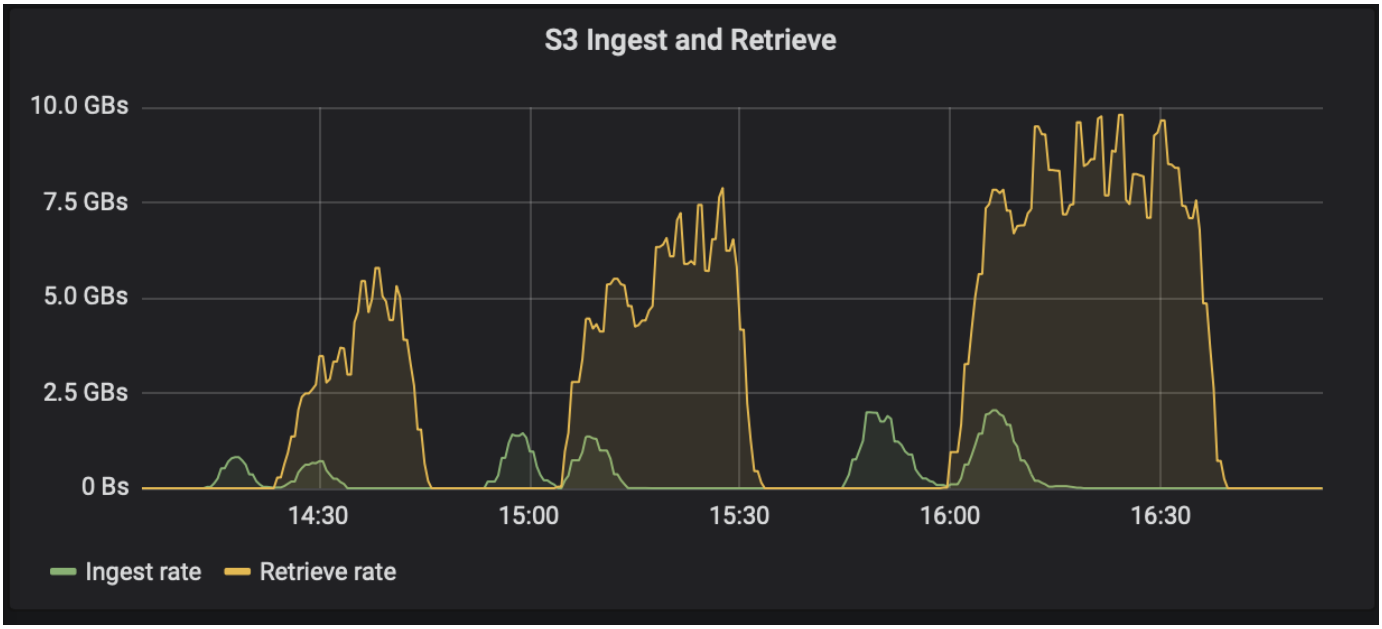
Nei cluster con bilanciamento automatico, i ribilanciamenti delle partizioni sono completamente automatizzati per ottimizzare il throughput di Kafka, accelerare la scalabilità dei broker e ridurre il carico operativo di un cluster di grandi dimensioni. A stato stazionario, i cluster con bilanciamento automatico monitorano l'inclinazione dei dati tra i broker e riassegnano continuamente le partizioni per ottimizzare le performance del cluster. Quando la piattaforma viene scalata verso l'alto o verso il basso, i cluster a bilanciamento automatico riconoscono automaticamente la presenza di nuovi broker o la rimozione di vecchi broker e attivano una successiva riassegnazione delle partizioni. In questo modo potrai aggiungere e decommissionare facilmente i broker, rendendo i cluster Kafka fondamentalmente più elastici. Questi vantaggi non richiedono interventi manuali, calcoli complessi o il rischio di errori umani che le riassegnazioni delle partizioni comportano in genere. Di conseguenza, i ribilanciamenti dei dati vengono completati in meno tempo e puoi concentrarti su progetti di streaming di eventi di valore superiore, invece di dover monitorare costantemente i tuoi cluster.

Linee guida sulle Best practice

Questa sezione presenta le lezioni apprese da questa certificazione.

- In base alla nostra convalida, lo storage a oggetti S3 è la soluzione migliore per Confluent per conservare i dati.
- Possiamo utilizzare SAN ad alto throughput (in particolare FC) per mantenere i dati hot del broker o il disco locale, perché, nella configurazione dello storage a più livelli Confluent, la dimensione dei dati contenuti nella directory dei dati broker si basa sulle dimensioni del segmento e sul tempo di conservazione quando i dati vengono spostati nello storage a oggetti.
- Gli archivi di oggetti offrono performance migliori quando segment.bytes è più elevato; abbiamo testato 512 MB.
- In Kafka, la lunghezza della chiave o del valore (in byte) per ciascun record prodotto per l'argomento è controllata da `length.key.value` parametro. Per StorageGRID, le performance di acquisizione e recupero degli oggetti S3 sono aumentate a valori più elevati. Ad esempio, 512 byte hanno fornito un recupero da 5,8 Gbps, 1024 byte hanno fornito un recupero s3 da 7,5 Gbps e 2048 byte sono forniti quasi a 10 Gbps.

La figura seguente illustra l'acquisizione e il recupero dell'oggetto S3 in base a. `length.key.value`.



- **Tuning di Kafka.** per migliorare le performance dello storage a più livelli, è possibile aumentare TierFetcherNumThreads e TierArchiverNumThreads. Come linea guida generale, si desidera aumentare TierFetcherNumThreads in modo che corrisponda al numero di core della CPU fisica e aumentare TierArchiverNumThreads fino alla metà del numero di core della CPU. Ad esempio, nelle proprietà del server, se si dispone di un computer con otto core fisici, impostare confluent.Tier.fetcher.num.threads = 8 e confluent.Tier.archiver.num.threads = 4.
- **Intervallo di tempo per l'eliminazione dell'argomento.** quando un argomento viene cancellato, l'eliminazione dei file di segmenti di registro nella memoria a oggetti non inizia immediatamente. Piuttosto, esiste un intervallo di tempo con un valore predefinito di 3 ore prima che venga eseguita l'eliminazione di tali file. È possibile modificare la configurazione, confluent.tier.topic.delete.check.interval.ms, per modificare il valore di questo intervallo. Se si elimina un argomento o un cluster, è anche possibile eliminare manualmente gli oggetti nel rispettivo bucket.
- **ACL su argomenti interni dello storage a più livelli.** Una Best practice consigliata per le implementazioni on-premise è abilitare un autorizzatore ACL sugli argomenti interni utilizzati per lo storage a più livelli. Impostare le regole ACL per limitare l'accesso a questi dati solo all'utente del broker. In questo modo si proteggono gli argomenti interni e si impedisce l'accesso non autorizzato ai dati e ai metadati dello storage a più livelli.

```
kafka-acls --bootstrap-server localhost:9092 --command-config adminclient-
configs.conf \
--add --allow-principal User:<kafka> --operation All --topic "_confluent-
tier-state"
```

Sostituire l'utente `<kafka>` con l'effettivo broker principal nella tua implementazione.

Ad esempio, il comando `confluent-tier-state` Imposta gli ACL sull'argomento interno per lo storage a più livelli. Attualmente, esiste solo un singolo argomento interno relativo allo storage a più livelli. Nell'esempio viene creato un ACL che fornisce l'autorizzazione principale Kafka per tutte le operazioni sull'argomento interno.

Dimensionamento

Il dimensionamento di Kafka può essere eseguito con quattro modalità di configurazione: Semplice, granulare, inversa e partizioni.

Semplice

La modalità Simple è adatta per gli utenti Apache Kafka per la prima volta o per i primi casi di utilizzo. Per questa modalità, vengono forniti requisiti come throughput Mbps, fanout di lettura, conservazione e percentuale di utilizzo delle risorse (il valore predefinito è 60%). Si accede anche all'ambiente, ad esempio on-premise (bare-metal, VMware, Kubernetes o OpenStack) o al cloud. In base a queste informazioni, il dimensionamento di un cluster Kafka fornisce il numero di server richiesti per il broker, lo zookeeper, gli impiegati di connessione Apache Kafka, il registro dello schema, un proxy REST, ksqlDB e il centro di controllo Confluent.

Per lo storage su più livelli, prendere in considerazione la modalità di configurazione granulare per il dimensionamento di un cluster Kafka. La modalità granulare è adatta agli utenti esperti di Apache Kafka o a casi di utilizzo ben definiti. Questa sezione descrive il dimensionamento per produttori, processori di streaming e consumatori.

Produttori

Per descrivere i produttori di Apache Kafka (ad esempio un client nativo, un proxy REST o un connettore Kafka), fornire le seguenti informazioni:

- **Nome.** Spark.
- **Tipo produttore.** applicazione o servizio, proxy (REST, MQTT, Altro) e database esistente (RDBMS, NOSQL, Altro). È anche possibile selezionare "non so".
- **Throughput medio.** in eventi al secondo (ad esempio 1,000,000).
- **Throughput massimo.** in eventi al secondo (ad esempio 4,000,000).
- **Dimensione media dei messaggi.** in byte, non compressi (massimo 1 MB; 1000 ad esempio).
- **Message format.** le opzioni includono Avro, JSON, buffer di protocollo, binario, testo, "Non lo so" e altri.
- **Fattore di replica.** le opzioni sono 1, 2, 3 (raccomandazione confluente), 4, 5, oppure 6.
- **Tempo di conservazione.** un giorno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati in Apache Kafka? Inserire -1 con qualsiasi unità per un tempo infinito. Il calcolatore presuppone un tempo di conservazione di 10 anni per una conservazione infinita.
- Selezionare la casella di controllo "Enable Tiered Storage to Decrease Broker Count and Allow for Infinite Storage?" (attiva lo storage a livelli per ridurre il numero di broker e consentire lo storage
- Quando lo storage su più livelli è attivato, i campi di conservazione controllano l'hot set di dati memorizzati localmente sul broker. I campi di conservazione dell'archivio controllano la durata della memorizzazione dei dati nello storage a oggetti di archiviazione.
- **Archival Storage Retention.** un anno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati nello storage di archiviazione? Inserire -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una conservazione di 10 anni per una conservazione infinita.
- **Growth Multiplier.** 1 (ad esempio). Se il valore di questo parametro si basa sul throughput corrente, impostarlo su 1. Per dimensionare in base alla crescita aggiuntiva, impostare questo parametro su un moltiplicatore di crescita.
- **Numero di istanze produttore.** 10 (ad esempio). Quante istanze di produttori saranno in esecuzione? Questo input è necessario per incorporare il carico della CPU nel calcolo del dimensionamento. Un valore

vuoto indica che il carico della CPU non è incorporato nel calcolo.

Sulla base di questo esempio di input, il dimensionamento ha il seguente effetto sui produttori:

- Throughput medio in byte non compressi: 1 Gbps. Throughput massimo in byte non compressi: 4 Gbps. Throughput medio in byte compressi: 400 Mbps. Throughput massimo in byte compressi: 1,6 Gbps. Si basa su un tasso di compressione predefinito del 60% (è possibile modificare questo valore).
  - Storage hotset on-broker totale richiesto: 31,104 TB, inclusa la replica, compressa. Storage di archiviazione off-broker totale richiesto: 378,432 TB, compresso. Utilizzare "https://fusion.netapp.com" Per il dimensionamento StorageGRID.

I processori stream devono descrivere le proprie applicazioni o servizi che consumano dati da Apache Kafka e riproducono in Apache Kafka. Nella maggior parte dei casi, questi sono integrati in flussi ksqlDB o Kafka.

- **Nome.** Spark streamer.
- **Tempo di elaborazione.** quanto tempo impiega questo processore per elaborare un singolo messaggio?
  - 1 ms (semplice, stateless transformation) [esempio], 10 ms (stateful in-memory operation).
  - 100 ms (funzionamento su disco o rete stateful), 1000 ms (chiamata DI PAUSA di terze parti).
  - Ho eseguito un benchmark di questo parametro e so esattamente quanto tempo occorre.
- **Conservazione dell'output.** 1 giorno (esempio). Un stream processor produce l'output di Apache Kafka. Per quanto tempo vuoi che questi dati di output siano memorizzati in Apache Kafka? Inserire -1 con qualsiasi unità per una durata infinita.
- Selezionare la casella di controllo "Enable Tiered Storage to Decrease Broker Count and Allow for Infinite Storage?" (attiva lo storage a più livelli per ridurre il numero di broker e
- **Archival Storage Retention.** 1 anno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati nello storage di archiviazione? Inserire -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una conservazione di 10 anni per una conservazione infinita.
- **Output Passthrough percent.** 100 (ad esempio). Un stream processor produce l'output di Apache Kafka. Quale percentuale di throughput in entrata verrà restituita ad Apache Kafka? Ad esempio, se il throughput in entrata è 20 Mbps e questo valore è 10, il throughput in uscita sarà 2 Mbps.
- Da quali applicazioni viene letto? Selezionare "Spark" (Spark), il nome utilizzato nel dimensionamento basato sul tipo di produttore. In base all'input di cui sopra, è possibile prevedere i seguenti effetti del dimensionamento sulle istanze del processo di flusso e sulle stime delle partizioni degli argomenti:
- Questa applicazione di elaborazione del flusso richiede il seguente numero di istanze. Gli argomenti in entrata probabilmente richiedono anche queste partizioni. Contattare Confluent per confermare questo parametro.
  - 1,000 per throughput medio senza moltiplicatore di crescita
  - 4,000 per throughput di picco senza moltiplicatore di crescita
  - 1,000 per un throughput medio con un moltiplicatore di crescita
  - 4,000 per throughput di picco con un moltiplicatore di crescita

Consumatori

Descrivi le tue applicazioni o servizi che consumano dati da Apache Kafka e non vengono prodotti in Apache Kafka, ad esempio un client nativo o un connettore Kafka.

- **Nome.** Spark consumer.

- **Tempo di elaborazione.** quanto tempo impiega questo cliente per elaborare un singolo messaggio?
  - 1 ms (ad esempio, un'attività semplice e senza stato come la registrazione)
  - 10 ms (scritture rapide in un datastore)
  - 100 ms (scritture lente in un datastore)
  - 1000 ms (chiamata DI RIPOSO di terze parti)
  - Alcuni altri processi di riferimento di durata nota.
- **Consumer type.** Application, proxy, or sink to a existing datastore (RDBMS, NoSQL, other).
- Da quali applicazioni viene letto? Collegare questo parametro al produttore e al dimensionamento del flusso determinati in precedenza.

In base all'input di cui sopra, è necessario determinare il dimensionamento per le istanze consumer e le stime delle partizioni degli argomenti. Un'applicazione consumer richiede il seguente numero di istanze.

- 2,000 per throughput medio, nessun moltiplicatore di crescita
- 8,000 per throughput di picco, nessun moltiplicatore di crescita
- 2,000 per throughput medio, incluso il moltiplicatore di crescita
- 8,000 per throughput di picco, incluso il moltiplicatore di crescita

Gli argomenti in entrata probabilmente necessitano anche di questo numero di partizioni. Contatta Confluent per confermare.

Oltre ai requisiti per i produttori, i processori di streaming e i consumatori, devi fornire i seguenti requisiti aggiuntivi:

- **Tempo di ricostruzione.** ad esempio, 4 ore. Se un host del broker Apache Kafka si guasta, i dati vengono persi e viene eseguito il provisioning di un nuovo host per sostituire l'host guasto, quanto velocemente deve essere ricostruito questo nuovo host? Lasciare vuoto questo parametro se il valore non è noto.
- **Obiettivo di utilizzo delle risorse (percentuale).** ad esempio, 60. In che modo desiderate che i vostri host siano utilizzati durante il throughput medio? Confluent consiglia un utilizzo del 60% a meno che non si utilizzino cluster di bilanciamento automatico Confluent, nel qual caso l'utilizzo può essere maggiore.

Descrivi il tuo ambiente

- **In quale ambiente verrà eseguito il tuo cluster?** Amazon Web Services, Microsoft Azure, piattaforma cloud Google, bare-metal on-premise, VMware on premise, OpenStack on premise o Kubernates on premise?
- **Dettagli host.** numero di core: 48 (ad esempio), tipo di scheda di rete (10GbE, 40GbE, 16GbE, 1GbE o altro tipo).
- **Storage Volumes.** host: 12 (ad esempio). Quanti dischi rigidi o SSD sono supportati per host? Confluent consiglia 12 dischi rigidi per host.
- **Capacità/volume di storage (in GB).** 1000 (ad esempio). Quanto storage può memorizzare un singolo volume in gigabyte? Confluent consiglia dischi da 1 TB.
- **Configurazione dello storage.** come vengono configurati i volumi dello storage? Confluent consiglia RAID10 per sfruttare tutte le funzionalità confluenti. JBOD, SAN, RAID 1, RAID 0, RAID 5, e altri tipi sono supportati.
- **Throughput di un volume singolo (Mbps).** 125 (ad esempio). Quanto velocemente un singolo volume di storage può leggere o scrivere in megabyte al secondo? Confluent consiglia dischi rigidi standard, che in

genere hanno un throughput di 125 MBps.

- **Capacità di memoria (GB).** 64 (ad esempio).

Dopo aver determinato le variabili ambientali, selezionare Size my Cluster (dimensione cluster). In base ai parametri di esempio sopra indicati, abbiamo determinato il seguente dimensionamento per Confluent Kafka:

- * Apache Kafka.* numero di broker: 22. Il cluster è legato allo storage. Considerare la possibilità di abilitare lo storage a più livelli per ridurre il numero di host e consentire lo storage infinito.
- **Apache Zooseeper.** Conteggio: 5; Apache Kafka Connect Workers: Conteggio: 2; Registro dello schema: Conteggio: 2; REST Proxy: Conteggio: 2; ksqlDB: Conteggio: 2; Centro di controllo confluente: Conteggio: 1.

Utilizza la modalità inversa per i team di piattaforme senza un caso d'utilizzo. Utilizzare la modalità Partitions per calcolare il numero di partizioni richiesto da un singolo argomento. Vedere https://eventsizer.io per il dimensionamento in base alle modalità di reverse e partitions.

Conclusione

Questo documento fornisce le linee guida sulle Best practice per l'utilizzo dello storage a livelli confluente con lo storage NetApp, inclusi test di verifica, risultati delle performance dello storage a livelli, tuning, connettori Confluent S3 e funzionalità di bilanciamento automatico. Considerando le policy ILM, le performance costanti con test delle performance multipli per la verifica e le API S3 standard di settore, lo storage a oggetti NetApp StorageGRID è la scelta ottimale per lo storage a livelli confluente.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Che cos'è Apache Kafka

  "https://www.confluent.io/what-is-apache-kafka/"

- Documentazione sui prodotti NetApp

  "https://www.netapp.com/support-and-training/documentation/"

- Dettagli del parametro S3-sink

  "https://docs.confluent.io/kafka-connect-s3-sink/current/configuration_options.html#s3-configuration-options"

- Apache Kafka

  "https://en.wikipedia.org/wiki/Apache_Kafka"

- Storage infinito nella piattaforma confluente

  "https://www.confluent.io/blog/infinite-kafka-storage-in-confluent-platform/"

- Confluent Tiered Storage - Best practice e dimensionamento

"https://docs.confluent.io/platform/current/kafka/tiered-storage.html#best-practices-and-recommendations"

- Connettore Amazon S3 sink per Confluent Platform

  "https://docs.confluent.io/kafka-connect-s3-sink/current/overview.html"

- Dimensionamento di Kafka

  "https://eventsizer.io"

- Dimensionamento StorageGRID

  "https://fusion.netapp.com/"

- Casi di utilizzo di Kafka

  "https://kafka.apache.org/uses"

- Cluster Kafka con bilanciamento automatico nella piattaforma confluente 6.0

  "https://www.confluent.io/blog/self-balancing-kafka-clusters-in-confluent-platform-6-0/"

  "https://www.confluent.io/blog/confluent-platform-6-0-delivers-the-most-powerful-event-streaming-platform-to-date/"

Soluzioni dati di cloud ibrido NetApp: Spark e Hadoop in base ai casi di utilizzo dei clienti

TR-4657: Soluzioni dati di cloud ibrido NetApp - Spark e Hadoop in base ai casi di utilizzo dei clienti

Karthikeyan Nagalingam e Sathish Thyagarajan, NetApp

Questo documento descrive le soluzioni dati del cloud ibrido che utilizzano i sistemi storage NetApp AFF e FAS, NetApp Cloud Volumes ONTAP, NetApp Connected Storage e la tecnologia FlexClone per Spark e Hadoop. Queste architetture di soluzioni consentono ai clienti di scegliere una soluzione di protezione dei dati appropriata per il proprio ambiente. NetApp ha progettato queste soluzioni in base all'interazione con i clienti e ai casi di utilizzo aziendali. Questo documento fornisce le seguenti informazioni dettagliate:

- Perché abbiamo bisogno della protezione dei dati per gli ambienti Spark e Hadoop e per le sfide dei clienti.
- Il data fabric basato sulla visione di NetApp e i suoi elementi di base e servizi.
- Come questi building block possono essere utilizzati per progettare flussi di lavoro flessibili per la protezione dei dati.
- I pro e i contro di diverse architetture basate su casi di utilizzo reali dei clienti. Ogni caso d'utilizzo fornisce i seguenti componenti:
  - Scenari dei clienti
  - Requisiti e sfide

- Soluzioni
- Riepilogo delle soluzioni

Perché la protezione dei dati Hadoop?

In un ambiente Hadoop e Spark, è necessario risolvere i seguenti problemi:

- **Errori software o umani.** un errore umano negli aggiornamenti software durante l'esecuzione delle operazioni dei dati Hadoop può portare a comportamenti errati che possono causare risultati imprevisti dal lavoro. In tal caso, dobbiamo proteggere i dati per evitare errori o risultati irragionevoli. Ad esempio, a causa di un aggiornamento software eseguito in modo non corretto per un'applicazione di analisi del segnale di traffico, una nuova funzionalità che non riesce ad analizzare correttamente i dati del segnale di traffico sotto forma di testo normale. Il software continua ad analizzare JSON e altri formati di file non di testo, con il risultato che il sistema di analisi del controllo del traffico in tempo reale produce risultati di previsione che sono punti dati mancanti. Questa situazione può causare uscite guaste che potrebbero causare incidenti ai segnali stradali. La protezione dei dati può risolvere questo problema fornendo la possibilità di eseguire rapidamente il rollback alla versione dell'applicazione precedente.
- **Dimensione e scalabilità.** la dimensione dei dati di analisi cresce giorno per giorno a causa del numero sempre crescente di origini dati e volumi. I social media, le app mobili, l'analisi dei dati e le piattaforme di cloud computing sono le principali fonti di dati nell'attuale mercato dei big data, che sta crescendo molto rapidamente, e quindi i dati devono essere protetti per garantire operazioni accurate dei dati.
- **Protezione dei dati nativa di Hadoop.** Hadoop ha un comando nativo per proteggere i dati, ma questo comando non fornisce coerenza dei dati durante il backup. Supporta solo il backup a livello di directory. Le snapshot create da Hadoop sono di sola lettura e non possono essere utilizzate per riutilizzare direttamente i dati di backup.

Problemi di protezione dei dati per i clienti Hadoop e Spark

Una sfida comune per i clienti di Hadoop e Spark consiste nel ridurre i tempi di backup e aumentare l'affidabilità del backup senza influire negativamente sulle performance del cluster di produzione durante la protezione dei dati.

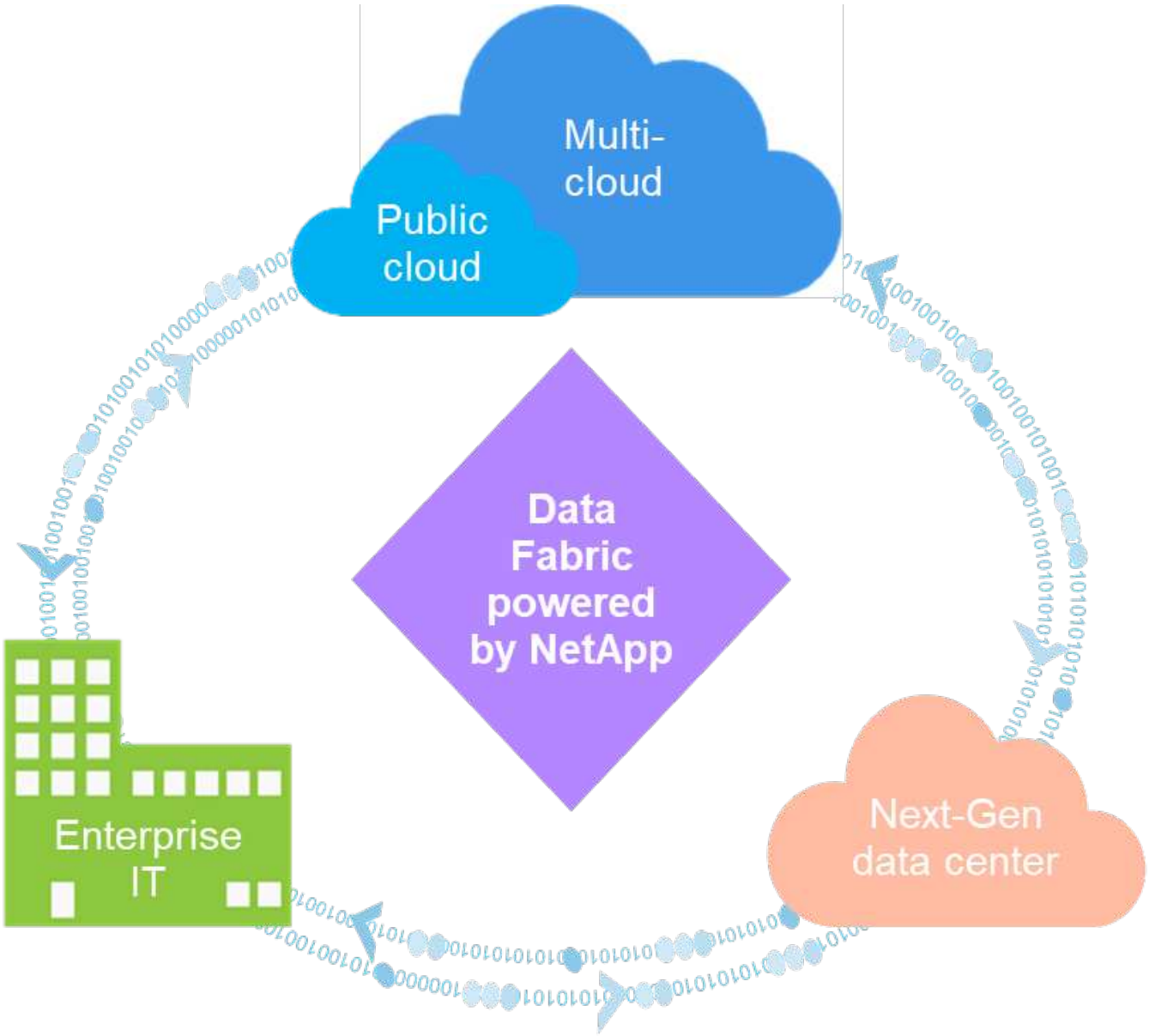
I clienti devono anche ridurre al minimo i tempi di inattività dell'obiettivo del punto di ripristino (RPO) e dell'obiettivo del tempo di ripristino (RTO) e controllare i siti di disaster recovery on-premise e basati sul cloud per una business continuity ottimale. Questo controllo deriva in genere dalla disponibilità di strumenti di gestione di livello Enterprise.

Gli ambienti Hadoop e Spark sono complicati perché non solo il volume di dati è enorme e in crescita, ma la velocità di arrivo dei dati è in aumento. Questo scenario rende difficile creare rapidamente ambienti DevTest e QA efficienti e aggiornati dai dati di origine. NetApp riconosce queste sfide e offre le soluzioni presentate in questo documento.

Data fabric basato su NetApp per l'architettura dei big data

Il data fabric basato su NetApp semplifica e integra la gestione dei dati in ambienti cloud e on-premise per accelerare la trasformazione digitale.

Il data fabric basato su NetApp offre servizi e applicazioni per la gestione dei dati coerenti e integrati (building block) per visibilità e approfondimenti dei dati, accesso e controllo dei dati, protezione e sicurezza dei dati, come mostrato nella figura seguente.



Casi di utilizzo comprovati per i clienti del data fabric

Il data fabric basato su NetApp offre ai clienti i seguenti nove casi di utilizzo comprovati:

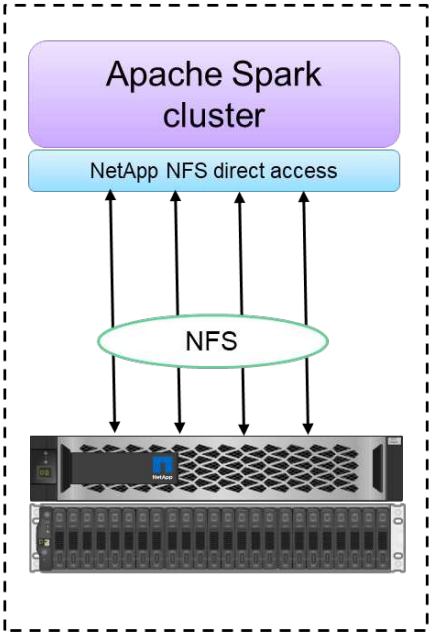
- Accelera i carichi di lavoro di analisi
- Accelera la trasformazione DevOps
- Costruire un'infrastruttura di cloud hosting
- Integra i servizi dati cloud
- Proteggere e proteggere i dati
- Ottimizza i dati non strutturati
- Ottieni efficienze nel data center
- Offri informazioni e controllo sui dati
- Semplifica e automatizza

Questo documento tratta due dei nove casi di utilizzo (insieme alle relative soluzioni):

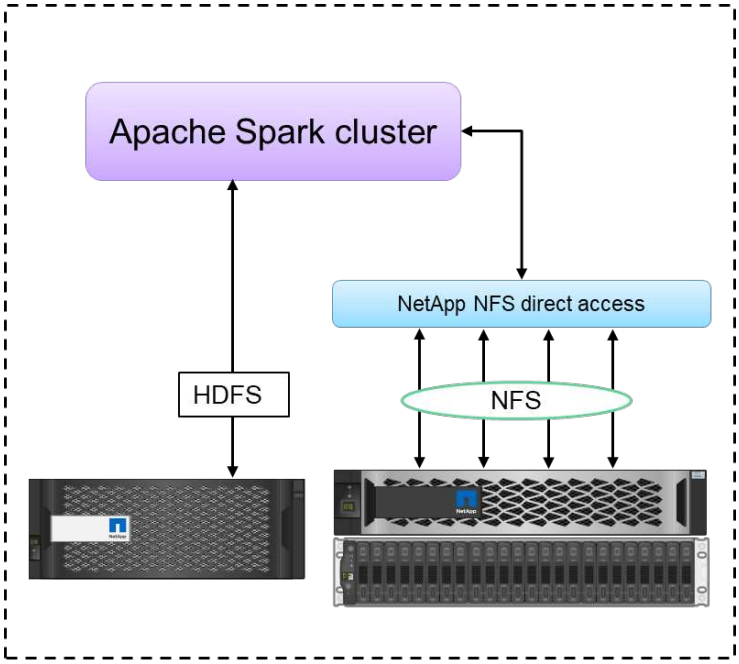
- Accelera i carichi di lavoro di analisi
- Proteggere e proteggere i dati

Accesso diretto NetApp NFS

NFS NetApp consente ai clienti di eseguire job di analytics dei big data sui propri NFSv4 dati NFSv3 o nuovi, senza spostare o copiare i dati. Impedisce più copie di dati ed elimina la necessità di sincronizzare i dati con un'origine. Ad esempio, nel settore finanziario, lo spostamento dei dati da un luogo all'altro deve soddisfare obblighi legali, il che non è un compito facile. In questo scenario, l'accesso diretto NetApp NFS analizza i dati finanziari dalla posizione originale. Un altro vantaggio chiave è che l'utilizzo dell'accesso diretto NetApp NFS semplifica la protezione dei dati Hadoop utilizzando comandi Hadoop nativi e abilita i flussi di lavoro per la protezione dei dati sfruttando il ricco portfolio di gestione dei dati di NetApp.



Configuration 1: NFS as primary storage



Configuration 2: HDFS and NFS in single Spark cluster

L'accesso diretto NetApp NFS offre due tipi di opzioni di implementazione per i cluster Hadoop/Spark:

- Per impostazione predefinita, i cluster Hadoop/Spark utilizzano Hadoop Distributed file System (HDFS) per lo storage dei dati e il file system predefinito. L'accesso diretto NetApp NFS può sostituire l'HDFS predefinito con lo storage NFS come file system predefinito, consentendo operazioni di analisi dirette sui dati NFS.
- In un'altra opzione di implementazione, l'accesso diretto NetApp NFS supporta la configurazione di NFS come storage aggiuntivo insieme a HDFS in un singolo cluster Hadoop/Spark. In questo caso, il cliente può condividere i dati attraverso le esportazioni NFS e accedervi dallo stesso cluster insieme ai dati HDFS.

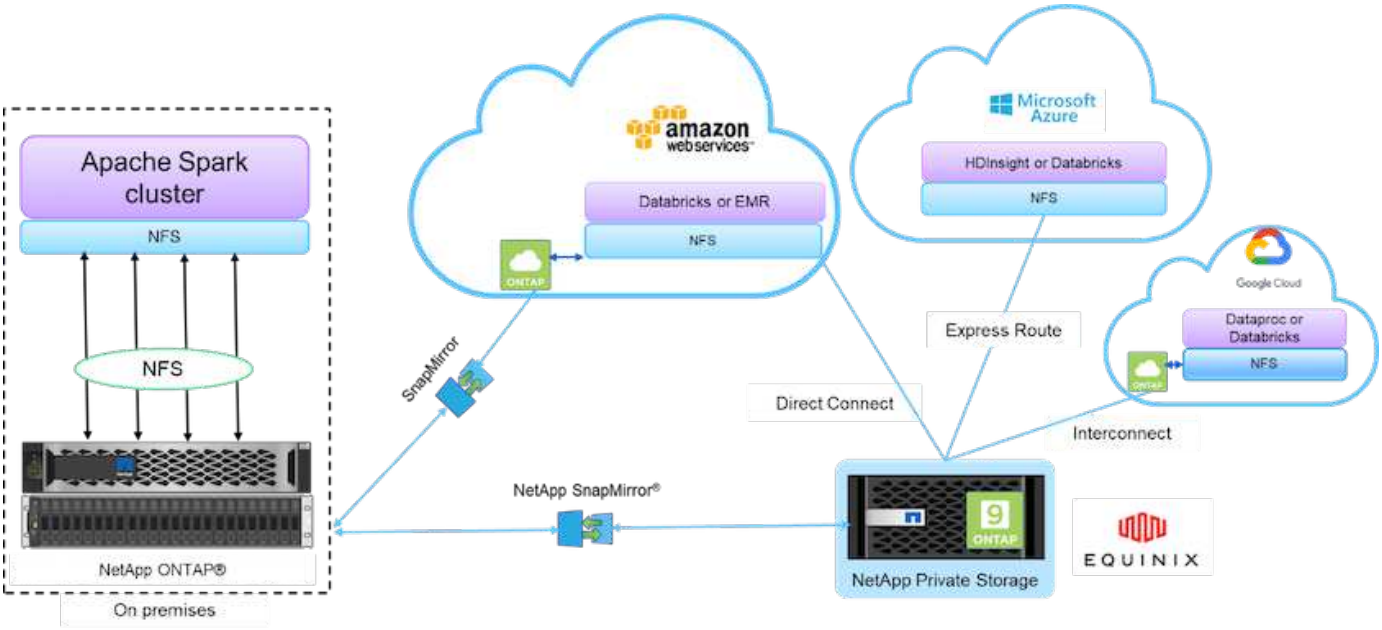
I vantaggi principali dell'utilizzo dell'accesso diretto NetApp NFS includono:

- Analizza i dati dalla posizione corrente, impedendo il dispendio di tempo e performance dello spostamento dei dati di analisi in un'infrastruttura Hadoop come HDFS.
- Riduce il numero di repliche da tre a uno.
- Consente agli utenti di separare il calcolo e lo storage per scalare in modo indipendente.
- Offre protezione dei dati aziendali sfruttando le funzionalità di gestione dei dati avanzate di ONTAP.

- È certificato con la piattaforma dati Hortonworks.
- Consente implementazioni di analisi dei dati ibride.
- Riduce i tempi di backup sfruttando la funzionalità multithread dinamica.

Elementi di base per i big data

Il data fabric basato su NetApp integra i servizi e le applicazioni di gestione dei dati (building block) per l'accesso, il controllo, la protezione e la sicurezza dei dati, come mostrato nella figura seguente.



Gli elementi di base della figura precedente includono:

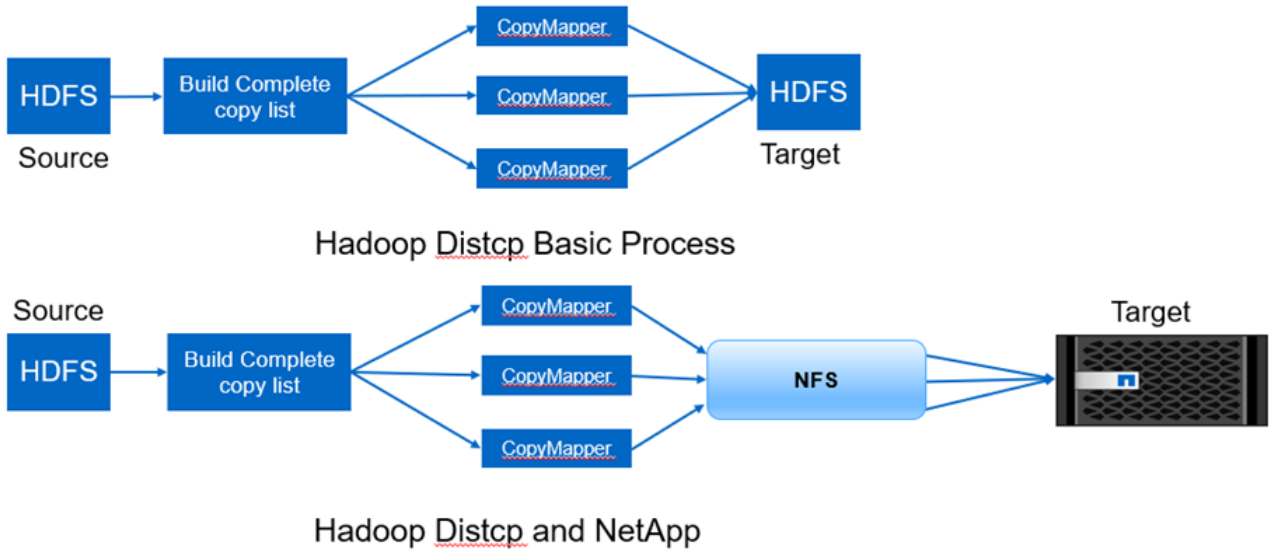
- **Accesso diretto NetApp NFS.** offre i più recenti cluster Hadoop e Spark con accesso diretto ai volumi NetApp NFS senza requisiti aggiuntivi di software o driver.
- **NetApp Cloud Volumes ONTAP e servizi di volume cloud.** storage connesso definito tramite software basato su ONTAP eseguito in AWS (Amazon Web Services) o Azure NetApp Files (ANF) nei servizi cloud Microsoft Azure.
- **Tecnologia NetApp SnapMirror**. Offre funzionalità di protezione dei dati tra istanze cloud on-premise e ONTAP o NPS.
- **Cloud service provider.** questi provider includono AWS, Microsoft Azure, Google Cloud e IBM Cloud.
- **PaaS.** servizi di analisi basati sul cloud come Amazon Elastic MapReduce (EMR) e Databricks in AWS, Microsoft Azure HDInsight e Azure Databricks.

Protezione dei dati Hadoop e NetApp

Hadoop DistCp è uno strumento nativo utilizzato per la copia di intercluster e intracluster di grandi dimensioni. Il processo di base di Hadoop DistCp illustrato nella figura seguente è un tipico workflow di backup che utilizza strumenti nativi di Hadoop come MapReduce per copiare i dati di Hadoop da un'origine HDFS a una destinazione corrispondente.

L'accesso diretto NetApp NFS consente ai clienti di impostare NFS come destinazione per lo strumento Hadoop DistCp per copiare i dati dall'origine HDFS in una condivisione NFS tramite MapReduce. L'accesso

diretto NetApp NFS funge da driver NFS per lo strumento DistCp.



Panoramica dei casi di utilizzo della protezione dei dati Hadoop

In questa sezione viene fornita una descrizione dettagliata dei casi di utilizzo della protezione dei dati, che costituiscono l'oggetto del presente documento. Le restanti sezioni forniscono ulteriori dettagli per ciascun caso di utilizzo, ad esempio il problema del cliente (scenario), i requisiti, le sfide e le soluzioni.

Caso d'utilizzo 1: Backup dei dati Hadoop

Per questo caso d'utilizzo, il volume NFS di NetApp ha aiutato un grande istituto finanziario a ridurre l'estesa finestra di backup da più di 24 ore a poco meno di poche ore.

Caso d'utilizzo 2: Backup e disaster recovery dal cloud all'on-premise

Utilizzando il data fabric basato su NetApp come elementi di base, una grande azienda di radiodiffusione è stata in grado di soddisfare il proprio requisito di backup dei dati cloud nel proprio data center on-premise, a seconda delle diverse modalità di trasferimento dei dati, come on-demand, istantaneo, O in base al carico del cluster Hadoop/Spark.

Caso d'utilizzo 3: Abilitare DevTest sui dati Hadoop esistenti

Le soluzioni NetApp hanno aiutato un distributore di musica online a creare rapidamente più cluster Hadoop efficienti in termini di spazio in diverse filiali per creare report ed eseguire attività DevTest quotidiane utilizzando policy pianificate.

Caso d'utilizzo 4: Protezione dei dati e connettività multicloud

Un grande provider di servizi ha utilizzato il data fabric basato su NetApp per fornire ai propri clienti analytics multicloud da diverse istanze di cloud.

Caso d'utilizzo 5: Accelerare i carichi di lavoro di analisi

Una delle più grandi banche di investimento e servizi finanziari ha utilizzato la soluzione di storage NAS di NetApp per ridurre i tempi di attesa i/o e accelerare la piattaforma di analisi finanziaria quantitativa.

Caso d'utilizzo 1: Backup dei dati Hadoop

Scenario

In questo scenario, il cliente dispone di un grande repository Hadoop on-premise e desidera eseguirne il backup a scopo di disaster recovery. Tuttavia, l'attuale soluzione di backup del cliente è costosa e presenta una lunga finestra di backup di oltre 24 ore.

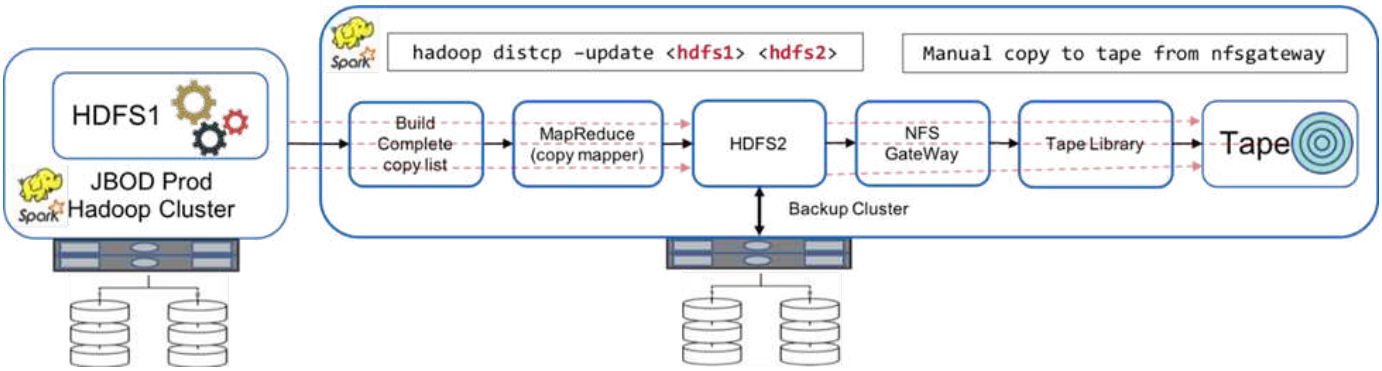
Requisiti e sfide

I requisiti e le sfide principali per questo caso di utilizzo includono:

- Compatibilità con le versioni precedenti del software:
  - La soluzione di backup alternativa proposta deve essere compatibile con le versioni software attualmente in esecuzione utilizzate nel cluster Hadoop di produzione.
- Per soddisfare gli SLA impegnati, la soluzione alternativa proposta dovrebbe raggiungere RPO e RTO molto bassi.
- Il backup creato dalla soluzione di backup NetApp può essere utilizzato nel cluster Hadoop costruito localmente nel data center e nel cluster Hadoop in esecuzione nella posizione di disaster recovery presso il sito remoto.
- La soluzione proposta deve essere conveniente.
- La soluzione proposta deve ridurre l'effetto delle performance sui processi di analisi in produzione attualmente in esecuzione durante i tempi di backup.

La soluzione di backup esistente del cliente x

La figura seguente mostra la soluzione di backup nativa Hadoop originale.



I dati di produzione sono protetti su nastro attraverso il cluster di backup intermedio:

- I dati HDFS1 vengono copiati in HDFS2 eseguendo il `hadoop distcp -update <hdfs1> <hdfs2>` comando.
- Il cluster di backup funge da gateway NFS e i dati vengono copiati manualmente su nastro attraverso Linux `cp` tramite la libreria di nastri.

I vantaggi della soluzione di backup nativa Hadoop originale includono:

- La soluzione si basa sui comandi nativi di Hadoop, che consentono all'utente di non dover apprendere nuove procedure.
- La soluzione sfrutta l'architettura e l'hardware standard di settore.

Gli svantaggi della soluzione di backup nativa Hadoop originale includono:

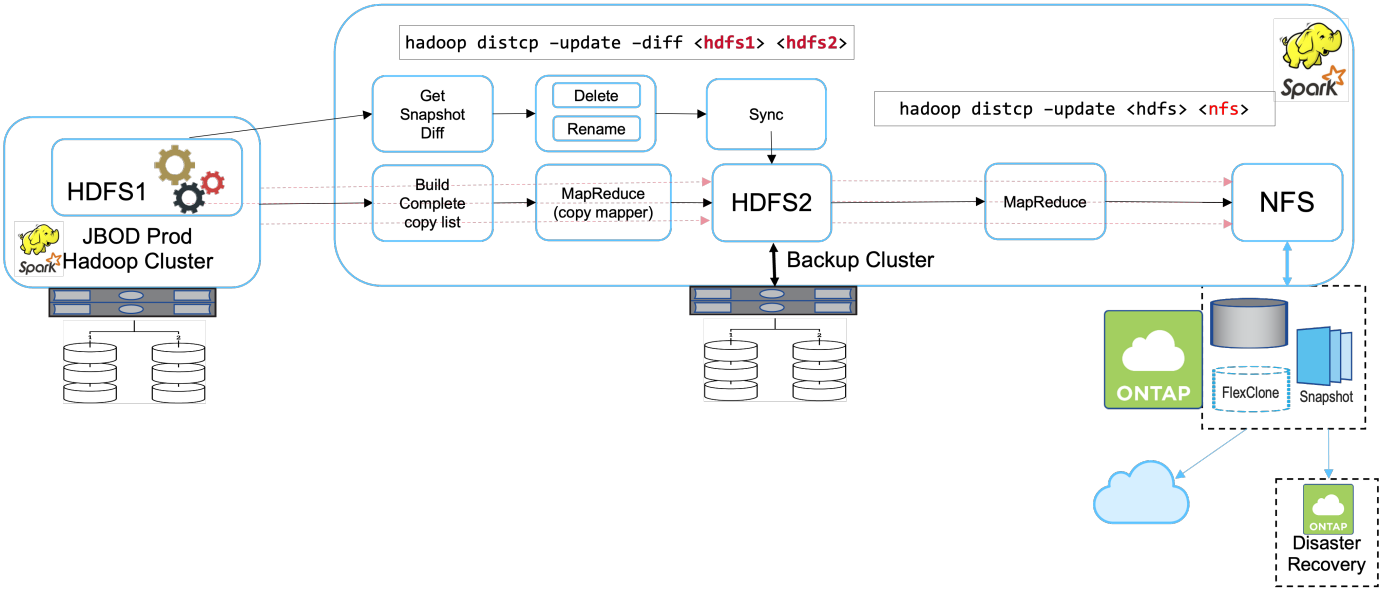
- La lunga finestra di backup supera le 24 ore, rendendo vulnerabili i dati di produzione.
- Peggioramento significativo delle performance del cluster durante i tempi di backup.
- La copia su nastro è un processo manuale.
- La soluzione di backup è costosa in termini di hardware richiesto e di ore umane richieste per i processi manuali.

Soluzioni di backup

In base a queste sfide e requisiti e tenendo conto del sistema di backup esistente, sono state suggerite tre possibili soluzioni di backup. Le seguenti sottosezioni descrivono ciascuna di queste tre diverse soluzioni di backup, etichettate dalla soluzione A alla soluzione C.

Soluzione A

Nella soluzione A, il cluster Hadoop di backup invia i backup secondari ai sistemi storage NFS NetApp, eliminando i requisiti su nastro, come mostrato nella figura sotto.



Le attività dettagliate per la soluzione A includono:

- Il cluster di produzione Hadoop contiene i dati di analisi del cliente nell'HDFS che richiede protezione.
- Il cluster di backup Hadoop con HDFS funge da posizione intermedia per i dati. Solo un gruppo di dischi (JBOD) fornisce lo storage per HDFS sia nei cluster Hadoop di produzione che di backup.
- Proteggere i dati di produzione di Hadoop dal cluster di produzione HDFS al cluster di backup HDFS mediante l'esecuzione di `Hadoop distcp -update -diff <hdfs1> <hdfs2>` comando.

> Lo snapshot Hadoop viene utilizzato per proteggere i dati dalla produzione al cluster di backup Hadoop.

- Il controller di storage NetApp ONTAP fornisce un volume esportato NFS, che viene fornito al cluster di backup Hadoop.
- Eseguendo il `Hadoop distcp` Comando che sfrutta MapReduce e mappatori multipli, i dati degli analytics sono protetti dal cluster Hadoop di backup su NFS.

  Una volta archiviati i dati in NFS sul sistema storage NetApp, le tecnologie NetApp Snapshot, SnapRestore e FlexClone vengono utilizzate per eseguire il backup, il ripristino e la duplicazione dei dati Hadoop in base alle necessità.

> I dati Hadoop possono essere protetti nel cloud e nelle posizioni di disaster recovery utilizzando la tecnologia SnapMirror.

I vantaggi della soluzione A includono:

- I dati di produzione di Hadoop sono protetti dal cluster di backup.
- I dati HDFS sono protetti tramite NFS, consentendo la protezione in ambienti cloud e di disaster recovery.
- Migliora le performance trasferendo le operazioni di backup nel cluster di backup.
- Elimina le operazioni manuali su nastro
- Consente funzioni di gestione aziendale tramite gli strumenti NetApp.
- Richiede modifiche minime all'ambiente esistente.
- È una soluzione conveniente.

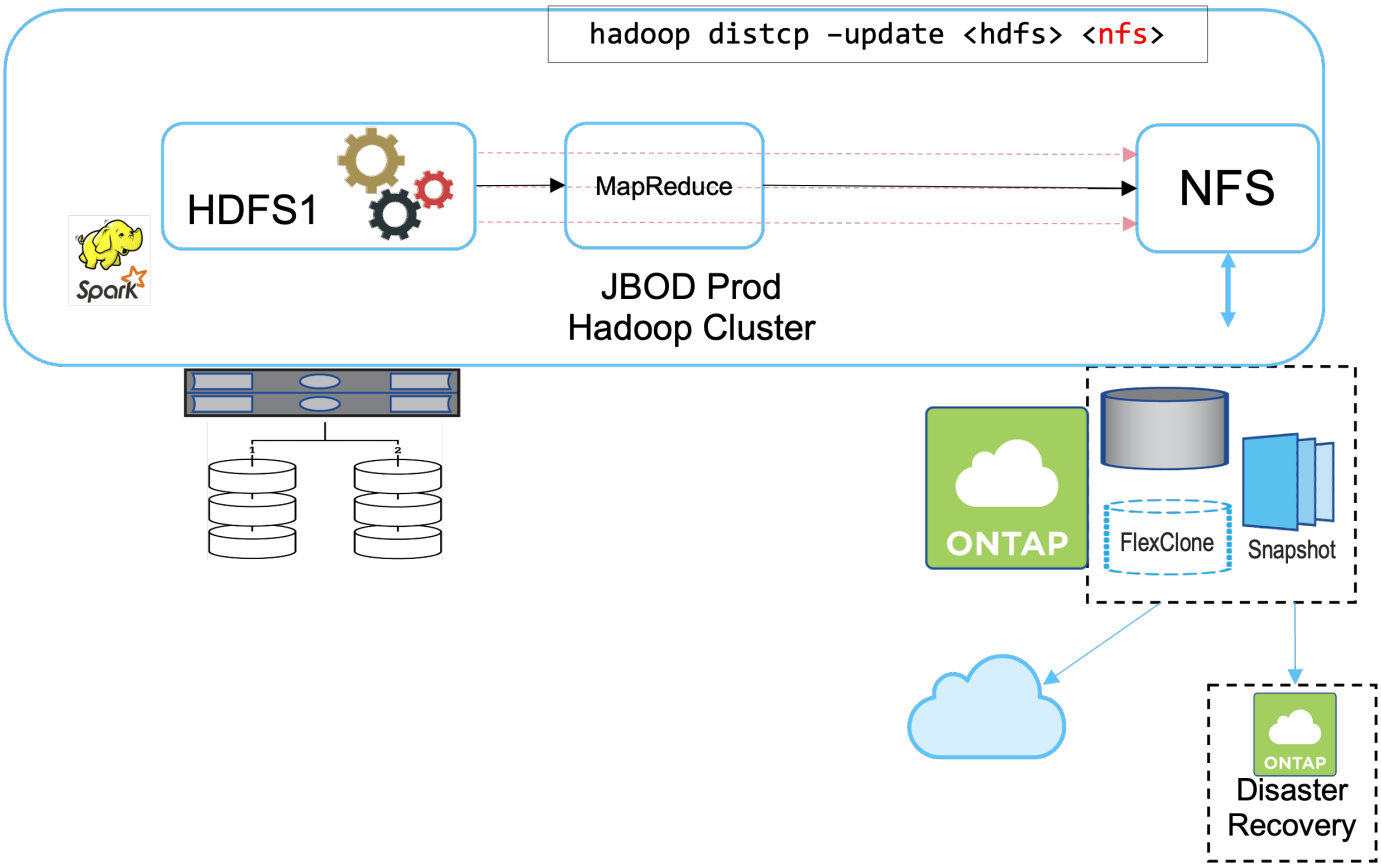
Lo svantaggio di questa soluzione è che richiede un cluster di backup e mappatori aggiuntivi per migliorare le performance.

Il cliente ha recentemente implementato la soluzione A per la sua semplicità, i costi e le performance complessive.

In questa soluzione, è possibile utilizzare i dischi SAN di ONTAP invece di JBOD. Questa opzione trasferisce il carico dello storage del cluster di backup su ONTAP; tuttavia, il downside è che sono richiesti switch fabric SAN.

Soluzione B

La soluzione B aggiunge un volume NFS al cluster Hadoop di produzione, che elimina la necessità di un cluster Hadoop di backup, come mostrato nella figura sotto.



Le attività dettagliate per la soluzione B includono:

- Lo storage controller NetApp ONTAP fornisce l'esportazione NFS al cluster Hadoop di produzione.

  Nativo di Hadoop `hadoop distcp` Command protegge i dati Hadoop dal cluster di produzione HDFS a NFS.

- Una volta archiviati i dati in NFS sul sistema storage NetApp, le tecnologie Snapshot, SnapRestore e FlexClone vengono utilizzate per eseguire il backup, il ripristino e la duplicazione dei dati Hadoop in base alle necessità.

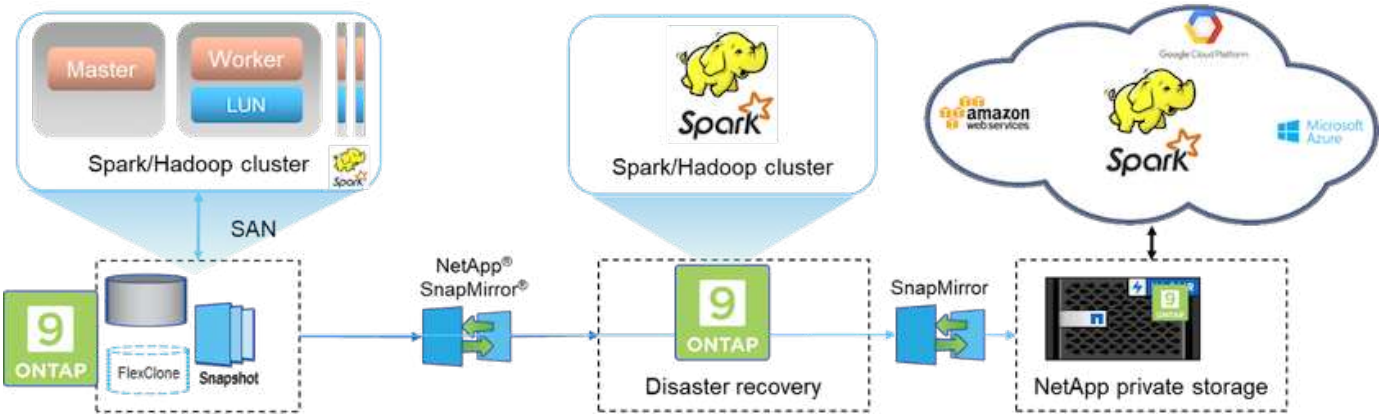
I vantaggi della soluzione B includono:

- Il cluster di produzione viene leggermente modificato per la soluzione di backup, semplificando l'implementazione e riducendo i costi aggiuntivi dell'infrastruttura.
- Non è necessario un cluster di backup per l'operazione di backup.
- I dati di produzione HDFS sono protetti nella conversione in dati NFS.
- La soluzione consente funzioni di gestione aziendale tramite gli strumenti NetApp.

Lo svantaggio di questa soluzione è che è implementata nel cluster di produzione, che può aggiungere ulteriori attività di amministratore nel cluster di produzione.

Soluzione C

Nella soluzione C, il provisioning dei volumi SAN NetApp viene eseguito direttamente nel cluster di produzione Hadoop per lo storage HDFS, come illustrato nella figura seguente.



I passaggi dettagliati per la soluzione C includono:

- Lo storage SAN NetApp ONTAP viene fornito nel cluster di produzione Hadoop per lo storage dei dati HDFS.
- Le tecnologie NetApp Snapshot e SnapMirror vengono utilizzate per eseguire il backup dei dati HDFS dal cluster Hadoop di produzione.
- Durante il processo di backup della copia Snapshot non si verificano effetti sulle performance per il cluster Hadoop/Spark, poiché il backup si trova a livello di storage.

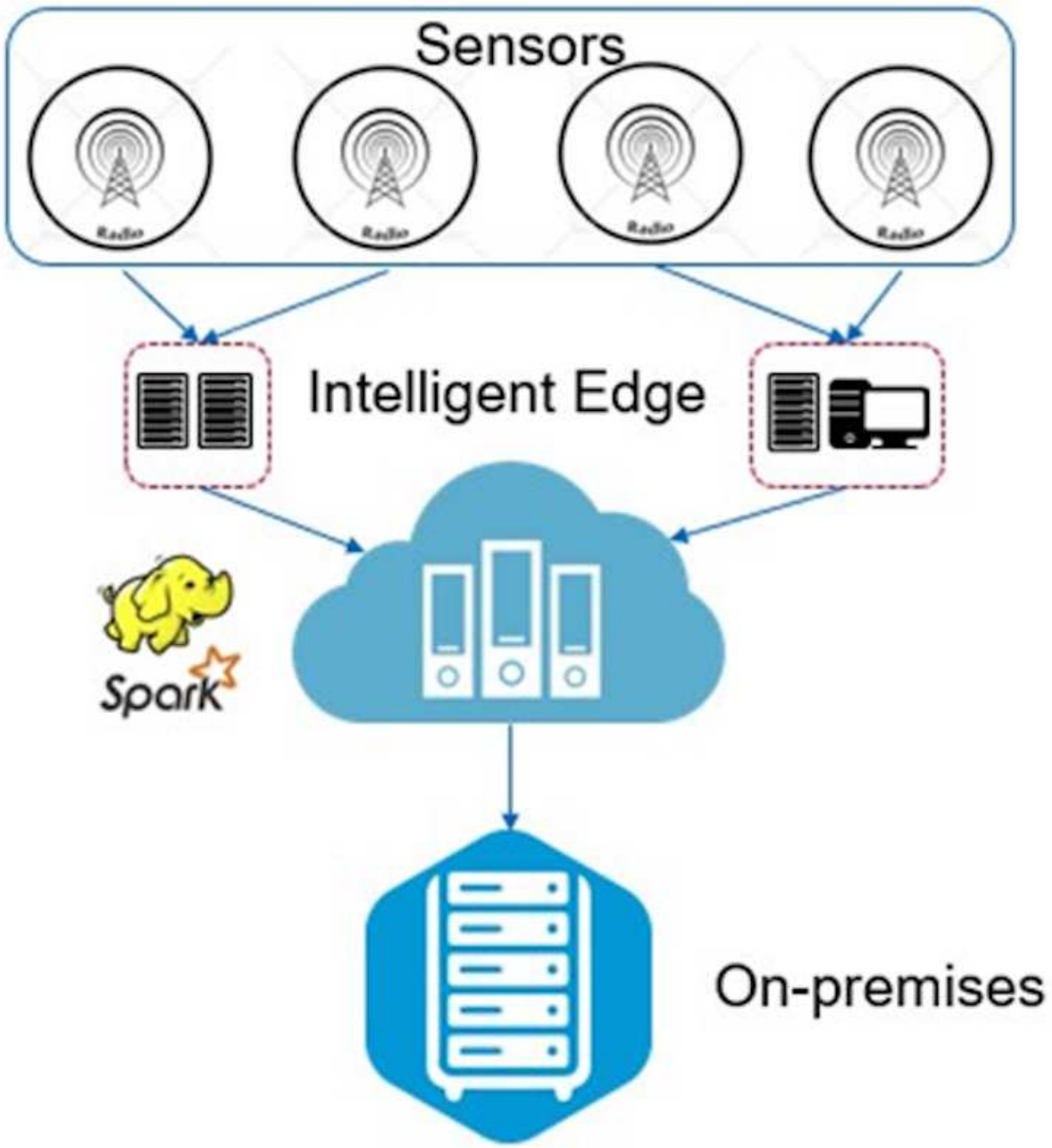
> La tecnologia Snapshot offre backup completi in pochi secondi, indipendentemente dalle dimensioni dei dati.

I vantaggi della soluzione C includono:

- È possibile creare backup efficienti in termini di spazio utilizzando la tecnologia Snapshot.
- Consente funzioni di gestione aziendale tramite gli strumenti NetApp.

Caso d'utilizzo 2: Backup e disaster recovery dal cloud all'on-premise

Questo caso di utilizzo si basa su un cliente che trasmette dati che deve eseguire il backup dei dati di analisi basati sul cloud nel proprio data center on-premise, come illustrato nella figura seguente.



Scenario

In questo scenario, i dati dei sensori IoT vengono acquisiti nel cloud e analizzati utilizzando un cluster open source Apache Spark all'interno di AWS. Il requisito è di eseguire il backup dei dati elaborati dal cloud a on-premise.

Requisiti e sfide

I requisiti e le sfide principali per questo caso di utilizzo includono:

- L'attivazione della protezione dei dati non dovrebbe causare alcun effetto sulle performance del cluster Spark/Hadoop di produzione nel cloud.
- I dati dei sensori cloud devono essere spostati e protetti on-premise in modo efficiente e sicuro.
- Flessibilità per il trasferimento dei dati dal cloud all'on-premise in diverse condizioni, ad esempio on-

demand, istantaneo e durante i bassi tempi di carico del cluster.

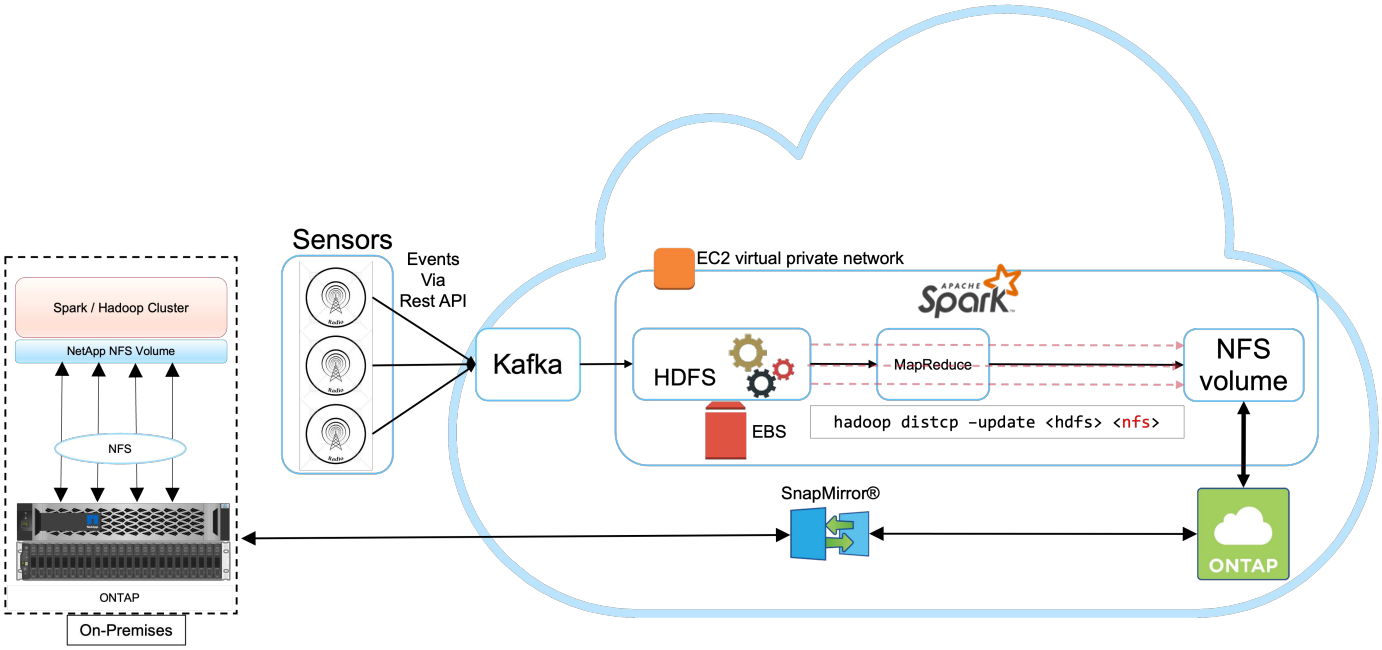
Soluzione

Il cliente utilizza AWS Elastic Block Store (EBS) per lo storage HDFS del cluster Spark per ricevere e acquisire dati dai sensori remoti tramite Kafka. Di conseguenza, lo storage HDFS funge da origine per i dati di backup.

Per soddisfare questi requisiti, il cloud NetApp ONTAP viene implementato in AWS e viene creata una condivisione NFS per fungere da destinazione di backup per il cluster Spark/Hadoop.

Una volta creata la share NFS, copia i dati dallo storage HDFS EBS nella share NFS ONTAP. Una volta che i dati risiedono in NFS nel cloud ONTAP, la tecnologia SnapMirror può essere utilizzata per eseguire il mirroring dei dati dal cloud allo storage on-premise in modo sicuro ed efficiente.

Questa immagine mostra il backup e il disaster recovery dal cloud alla soluzione on-premise.



Caso d'utilizzo 3: Abilitare DevTest sui dati Hadoop esistenti

In questo caso di utilizzo, il requisito del cliente è quello di creare rapidamente ed efficientemente nuovi cluster Hadoop/Spark basati su un cluster Hadoop esistente contenente una grande quantità di dati di analisi per DevTest e scopi di reporting nello stesso data center e in sedi remote.

Scenario

In questo scenario, i cluster Spark/Hadoop multipli sono costruiti da un'implementazione di data Lake Hadoop di grandi dimensioni on-premise e in ubicazioni di disaster recovery.

Requisiti e sfide

I requisiti e le sfide principali per questo caso di utilizzo includono:

- Creare più cluster Hadoop per DevTest, QA o qualsiasi altro scopo che richieda l'accesso agli stessi dati di produzione. La sfida consiste nel clonare un cluster Hadoop molto grande più volte istantaneamente e in

modo molto efficiente in termini di spazio.

- Sincronizza i dati di Hadoop con DevTest e i team di reporting per l'efficienza operativa.
- Distribuire i dati Hadoop utilizzando le stesse credenziali in produzione e nei nuovi cluster.
- Utilizza policy pianificate per creare in modo efficiente cluster di QA senza influire sul cluster di produzione.

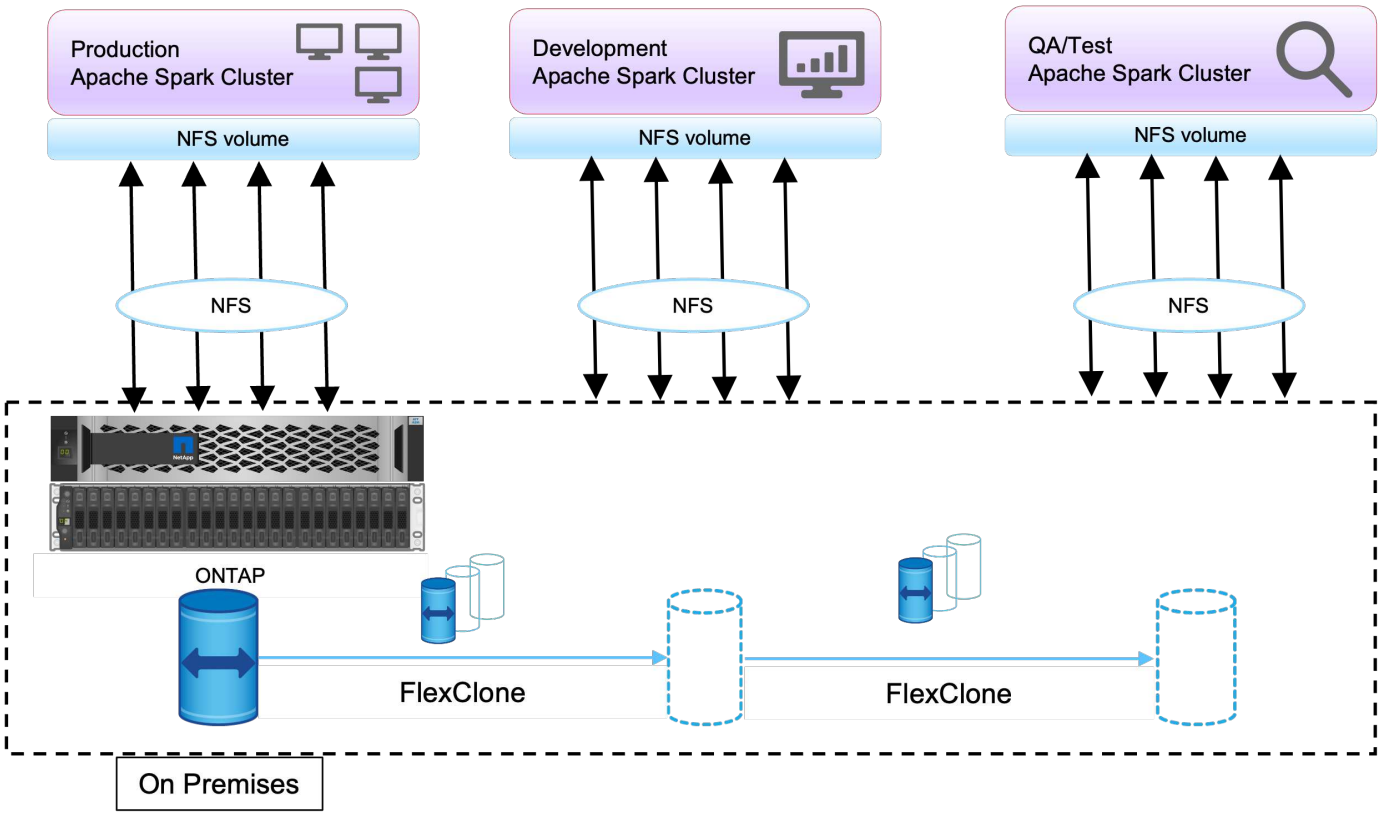
Soluzione

La tecnologia FlexClone viene utilizzata per rispondere ai requisiti appena descritti. La tecnologia FlexClone è la copia in lettura/scrittura di una copia Snapshot. Legge i dati dai dati di copia Snapshot padre e consuma solo spazio aggiuntivo per i blocchi nuovi/modificati. È veloce ed efficiente in termini di spazio.

Innanzitutto, è stata creata una copia Snapshot del cluster esistente utilizzando un gruppo di coerenza NetApp.

Copie Snapshot all'interno di NetApp System Manager o al prompt di amministrazione dello storage. Il gruppo di coerenza copie Snapshot sono copie Snapshot di gruppo coerenti con l'applicazione e il volume FlexClone viene creato in base alle copie Snapshot del gruppo di coerenza. Vale la pena ricordare che un volume FlexClone eredita la policy di esportazione NFS del volume padre. Una volta creata la copia Snapshot, è necessario installare un nuovo cluster Hadoop per DevTest e per la creazione di report, come illustrato nella figura seguente. Il volume NFS clonato dal nuovo cluster Hadoop accede ai dati NFS.

Questa immagine mostra il cluster Hadoop per DevTest.



Caso d'utilizzo 4: Protezione dei dati e connettività multicloud

Questo caso di utilizzo è importante per un partner di servizi cloud che ha il compito di fornire connettività multi-cloud per i dati di analisi dei big data dei clienti.

Scenario

In questo scenario, i dati IoT ricevuti in AWS da diverse origini vengono memorizzati in una posizione centrale in NPS. Lo storage NPS è connesso ai cluster Spark/Hadoop situati in AWS e Azure, consentendo l'esecuzione di applicazioni di analisi dei big data in più cloud che accedono agli stessi dati.

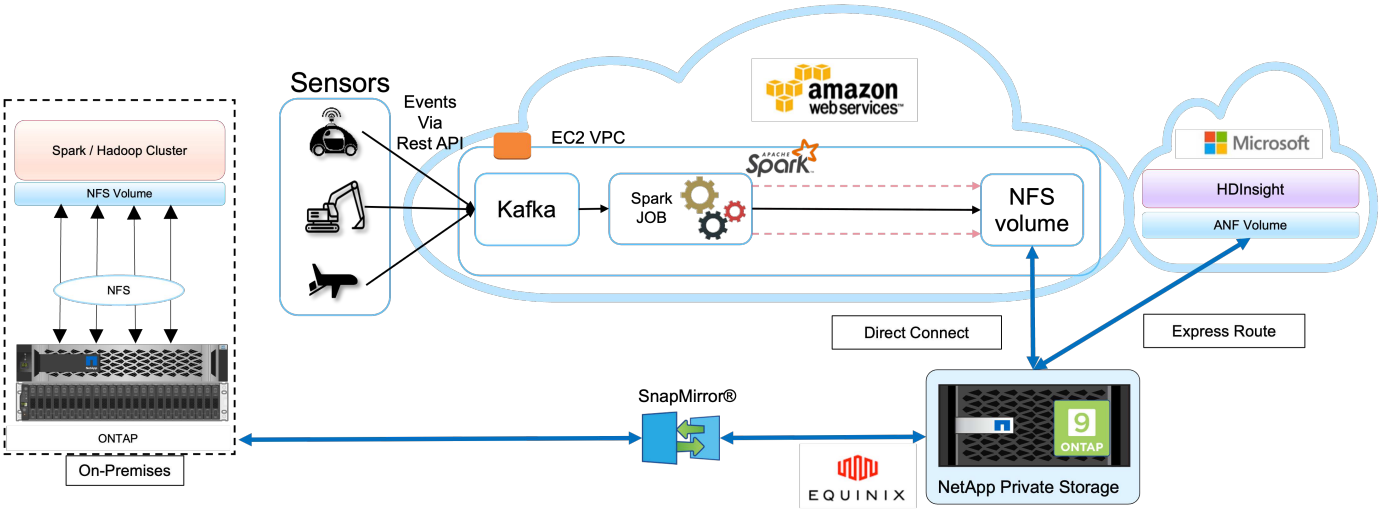
Requisiti e sfide

I requisiti e le sfide principali per questo caso di utilizzo includono:

- I clienti vogliono eseguire lavori di analisi sugli stessi dati utilizzando più cloud.
- I dati devono essere ricevuti da fonti diverse, ad esempio on-premise e cloud, attraverso diversi sensori e hub.
- La soluzione deve essere efficiente e conveniente.
- La sfida principale è quella di creare una soluzione conveniente ed efficiente in grado di offrire servizi di analisi ibridi tra cloud diversi e on-premise.

Soluzione

Questa immagine illustra la soluzione per la protezione dei dati e la connettività multicloud.



Come mostrato nella figura precedente, i dati provenienti dai sensori vengono trasmessi e acquisiti nel cluster AWS Spark tramite Kafka. I dati vengono memorizzati in una condivisione NFS residente in NPS, che si trova all'esterno del cloud provider all'interno di un data center Equinix. Poiché NetApp NetApp Private Storage è connesso ad Amazon AWS e Microsoft Azure tramite connessioni Direct Connect ed Express Route, i clienti possono accedere ai dati NFS dai cluster di analytics di Amazon e AWS. Questo approccio risolve l'utilizzo di analytics nel cloud in più hyperscaler.

Di conseguenza, poiché sia lo storage on-premise che NPS esegue il software ONTAP, SnapMirror può eseguire il mirroring dei dati NPS nel cluster on-premise, fornendo analisi del cloud ibrido tra cloud on-premise e multipli.

Per ottenere le migliori performance, NetApp consiglia di utilizzare interfacce di rete multiple e percorsi di connessione diretta/express per accedere ai dati dalle istanze cloud.

Caso d’utilizzo 5: Accelerare i carichi di lavoro di analisi

In questo scenario, è stata modernizzata una grande piattaforma di analytics per i servizi finanziari e le banche di investimento utilizzando la soluzione di storage NetApp NFS per ottenere un miglioramento significativo nell’analisi dei rischi di investimento e dei derivati per la sua business unit quantitativa e di gestione delle risorse.

Scenario

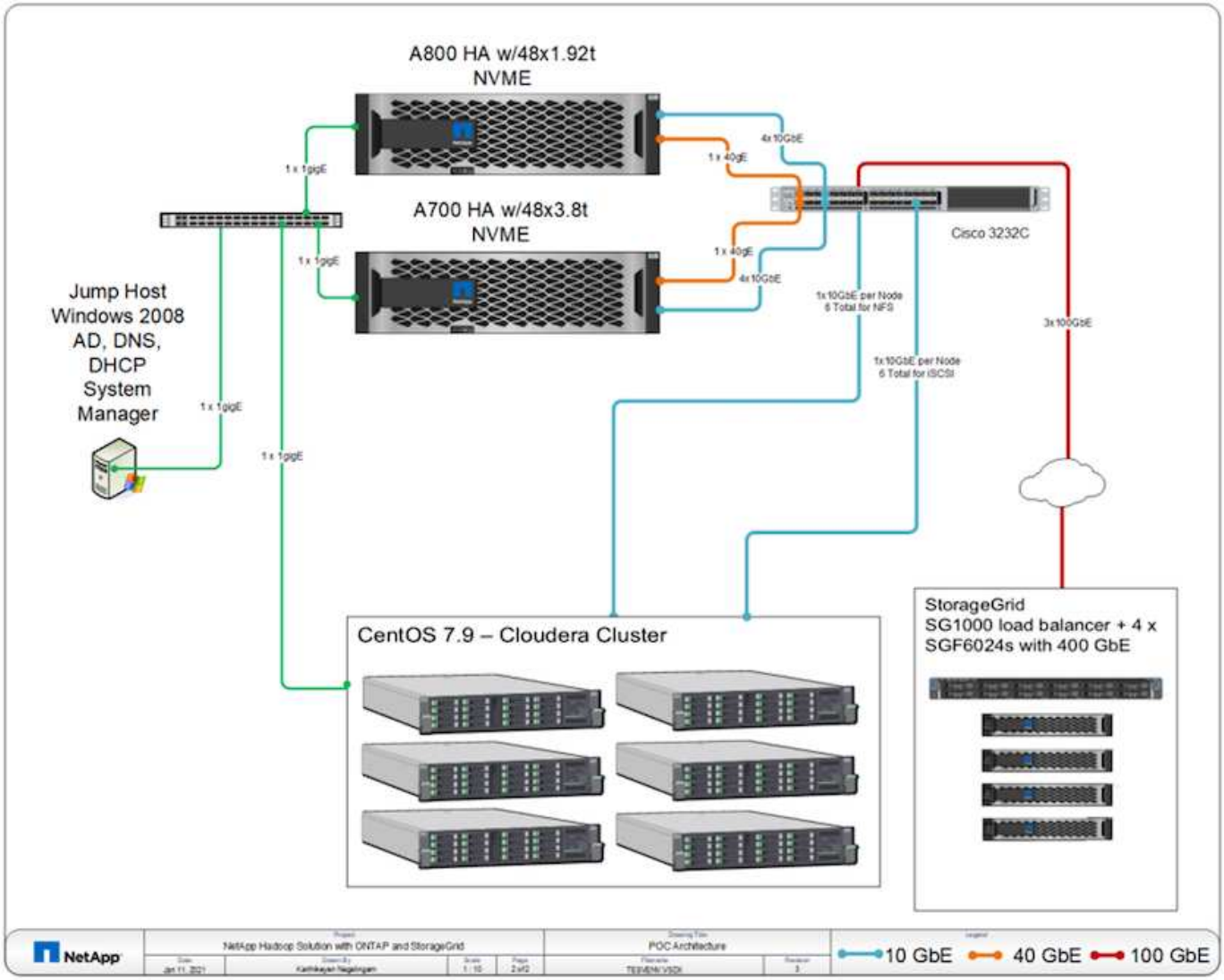
Nell’ambiente esistente del cliente, l’infrastruttura Hadoop utilizzata per la piattaforma di analisi ha sfruttato lo storage interno dei server Hadoop. A causa della natura proprietaria dell’ambiente JBOD, molti clienti interni all’interno dell’organizzazione non sono stati in grado di sfruttare il proprio modello quantitativo Monte Carlo, una simulazione che si basa sui campioni ricorrenti di dati in tempo reale. La capacità non ottimale di comprendere gli effetti dell’incertezza nei movimenti di mercato serviva in modo sfavorevole per la business unit di gestione quantitativa delle risorse.

Requisiti e sfide

La business unit quantitativa della banca desiderava un metodo di previsione efficiente per ottenere previsioni precise e tempestive. A tale scopo, il team ha riconosciuto la necessità di modernizzare l’infrastruttura, ridurre i tempi di attesa i/o esistenti e migliorare le performance delle applicazioni di analisi come Hadoop e Spark per simulare in modo efficiente i modelli di investimento, misurare i potenziali guadagni e analizzare i rischi.

Soluzione

Il cliente disponeva di JBOD per la soluzione Spark esistente. NetApp ONTAP, NetApp StorageGRID e MinIO Gateway to NFS sono stati quindi sfruttati per ridurre i tempi di attesa i/o per il gruppo finanziario quantitativo della banca che esegue simulazioni e analisi su modelli di investimento che valutano potenziali guadagni e rischi. Questa immagine mostra la soluzione Spark con lo storage NetApp.



Come mostrato nella figura precedente, i sistemi AFF A800, A700 e StorageGRID sono stati implementati per accedere ai file di parquet attraverso i protocolli NFS e S3 in un cluster Hadoop a sei nodi con Spark e i servizi di metadati YARN e Hive per le operazioni di analisi dei dati.

Una soluzione DAS (Direct-Attached Storage) nel vecchio ambiente del cliente ha avuto lo svantaggio di scalare calcolo e storage in modo indipendente. Con la soluzione NetApp ONTAP per Spark, la business unit di analisi finanziaria della banca è stata in grado di separare lo storage dal calcolo e di portare le risorse dell'infrastruttura in modo più efficace secondo necessità.

Utilizzando ONTAP con NFS, le CPU dei server di calcolo sono state quasi completamente utilizzate per i job SQL di Spark e il tempo di attesa i/o è stato ridotto di quasi il 70%, fornendo quindi una potenza di calcolo e un incremento delle performance migliori per i carichi di lavoro di Spark. In seguito, l'aumento dell'utilizzo della CPU ha anche consentito al cliente di sfruttare GPU, come GPUDirect, per un'ulteriore modernizzazione della piattaforma. Inoltre, StorageGRID offre un'opzione di storage a basso costo per i carichi di lavoro Spark e il gateway MinIO fornisce un accesso sicuro ai dati NFS attraverso il protocollo S3. Per i dati nel cloud, NetApp consiglia Cloud Volumes ONTAP, Azure NetApp Files e NetApp Cloud Volumes Service.

Conclusione

Questa sezione fornisce un riepilogo dei casi di utilizzo e delle soluzioni fornite da NetApp per soddisfare i vari requisiti di protezione dei dati Hadoop. Utilizzando il data

fabric basato su NetApp, i clienti possono:

- Avere la flessibilità di scegliere le giuste soluzioni per la protezione dei dati sfruttando le ricche funzionalità di gestione dei dati di NetApp e l'integrazione con i flussi di lavoro nativi di Hadoop.
- Ridurre i tempi di backup del cluster Hadoop di quasi il 70%.
- Elimina qualsiasi effetto sulle performance derivante dai backup del cluster Hadoop.
- Protezione dei dati multicloud e accesso ai dati da diversi cloud provider contemporaneamente a una singola fonte di dati di analisi.
- Crea copie cluster Hadoop veloci ed efficienti in termini di spazio utilizzando la tecnologia FlexClone.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Soluzioni NetApp per l'analisi dei big data

  "https://www.netapp.com/us/solutions/applications/big-data-analytics/index.aspx"

- Apache Spark workload con lo storage NetApp

  https://www.netapp.com/pdf.html?item=/media/26877-nva-1157-deploy.pdf

- Soluzioni storage NetApp per Apache Spark

  "https://www.netapp.com/media/16864-tr-4570.pdf"

- Apache Hadoop su data fabric abilitato da NetApp

  "https://www.netapp.com/media/16877-tr-4529.pdf"

Ringraziamenti

- Paul Burland, Sales Rep, ANZ Victoria District Sales, NetApp
- Hoseb Dermanilian, Business Development Manager, NetApp
- Lee Dorrier, Director MPSG, NetApp
- David Thiessen, Systems Engineer, ANZ Victoria District se, NetApp

Cronologia delle versioni

Versione	Data	Cronologia delle versioni del documento
Versione 1.0	Gennaio 2018	Release iniziale
Versione 2.0	Ottobre 2021	Aggiornato con il caso di utilizzo n. 5: Accelerare il carico di lavoro di analisi
Versione 3.0	Novembre 2023	Dettagli NIPAM rimossi

Analisi dei dati moderna: Soluzioni diverse per diverse strategie di analisi

Questo white paper descrive le moderne strategie delle soluzioni di analisi dei dati di NetApp. Include dettagli sui risultati di business, le sfide dei clienti, le tendenze tecnologiche, l'architettura legacy della concorrenza, i flussi di lavoro moderni, casi di utilizzo, settori, cloud, partner tecnologici, data mover, NetApp Active IQ, NetApp DataOps Toolkit, Hadoop to Spark, storage software-defined con NetApp Astra Control, container, gestione dei dati aziendali, archiviazione e tiering per raggiungere gli obiettivi di ai e analisi e come NetApp e i clienti stanno modernizzando insieme la propria architettura dei dati.

"Analisi dei dati moderna: Soluzioni diverse per diverse strategie di analisi"

TR-4623: NetApp e-Series E5700 e Splunk Enterprise

Mitch Blackburn, NetApp

TR-4623 descrive l'architettura integrata del design NetApp e-Series e Splunk. Ottimizzato per bilanciamento dello storage dei nodi, affidabilità, performance, capacità dello storage e densità, Questa progettazione utilizza il modello di nodo Splunk Clustered Index, con una maggiore scalabilità e un TCO inferiore. Il disaccoppiamento dello storage dal calcolo offre la possibilità di scalare ciascuno separatamente, risparmiando il costo dell'overprovisioning uno o l'altro. Inoltre, questo documento riassume i risultati dei test delle performance ottenuti da uno strumento di simulazione degli eventi del log della macchina Splunk.

"TR-4623: NetApp e-Series E5700 e Splunk Enterprise"

NVA-1157-DEPLOY: Workload Apache Spark con soluzione storage NetApp

Karthikeyan Nagalingam, NetApp

NVA-1157-DEPLOY descrive le performance e la convalida delle funzionalità di Apache Spark SQL su sistemi storage NetApp NFS AFF. Esamina la configurazione, l'architettura e i test delle performance in base a diversi scenari, oltre a consigli per l'utilizzo di Spark con il software di gestione dei dati NetApp ONTAP. Vengono inoltre illustrati i risultati dei test basati su un gruppo di dischi (JBOD) rispetto al controller di storage NetApp AFF A800.

"NVA-1157-DEPLOY: Workload Apache Spark con soluzione storage NetApp"