Cargas de trabajo de Apache Kafka con almacenamiento NFS de NetApp

NetApp artificial intelligence solutions

NetApp
August 18, 2025

Tabla de contenidos

Cargas de trabajo de Apache Kafka con almacenamiento NFS de NetApp

TR-4947: Carga de trabajo de Apache Kafka con almacenamiento NFS de NetApp : validación funcional y rendimiento

Shantanu Chakole, Karthikeyan Nagalingam y Joe Scott, NetApp

Kafka es un sistema de mensajería distribuida de publicación y suscripción con una cola robusta que puede aceptar grandes cantidades de datos de mensajes. Con Kafka, las aplicaciones pueden escribir y leer datos en temas de forma muy rápida. Debido a su tolerancia a fallas y escalabilidad, Kafka se utiliza a menudo en el espacio de big data como una forma confiable de ingerir y mover muchos flujos de datos muy rápidamente. Los casos de uso incluyen procesamiento de transmisiones, seguimiento de la actividad del sitio web, recopilación y monitoreo de métricas, agregación de registros, análisis en tiempo real, etc.

Si bien las operaciones normales de Kafka en NFS funcionan bien, el problema del cambio de nombre tonto hace que la aplicación se bloquee durante el cambio de tamaño o la repartición de un clúster de Kafka que se ejecuta en NFS. Este es un problema importante porque es necesario redimensionar o reparticionar un clúster de Kafka para equilibrar la carga o realizar mantenimiento. Puede encontrar detalles adicionales "aquí" .

Este documento describe los siguientes temas:

- El problema del cambio de nombre tonto y la validación de la solución
- Reducir la utilización de la CPU para reducir el tiempo de espera de E/S
- Tiempo de recuperación del agente de Kafka más rápido
- Rendimiento en la nube y en las instalaciones

¿Por qué utilizar almacenamiento NFS para cargas de trabajo de Kafka?

Las cargas de trabajo de Kafka en aplicaciones de producción pueden transmitir enormes cantidades de datos entre aplicaciones. Estos datos se guardan y almacenan en los nodos del agente de Kafka en el clúster de Kafka. Kafka también es conocido por su disponibilidad y paralelismo, que logra dividiendo los temas en particiones y luego replicando esas particiones en todo el clúster. Al final, esto significa que la enorme cantidad de datos que fluye a través de un clúster de Kafka generalmente se multiplica en tamaño. NFS permite reequilibrar los datos a medida que cambia el número de corredores de forma muy rápida y sencilla. En el caso de entornos grandes, reequilibrar los datos en DAS cuando cambia la cantidad de intermediarios consume mucho tiempo y, en la mayoría de los entornos de Kafka, la cantidad de intermediarios cambia con frecuencia.

Otros beneficios incluyen los siguientes:

- **Madurez.** NFS es un protocolo maduro, lo que significa que la mayoría de los aspectos de su implementación, protección y uso se comprenden bien.
- **Abierto.** NFS es un protocolo abierto y su desarrollo continuo está documentado en las especificaciones de Internet como un protocolo de red libre y abierto.

- **Rentable.** NFS es una solución de bajo costo para compartir archivos en red que es fácil de configurar porque utiliza la infraestructura de red existente.
- **Gestión centralizada.** La gestión centralizada de NFS reduce la necesidad de software y espacio en disco adicionales en los sistemas de usuarios individuales.
- **Repartido.** NFS se puede utilizar como un sistema de archivos distribuido, lo que reduce la necesidad de dispositivos de almacenamiento de medios extraíbles.

¿Por qué NetApp para las cargas de trabajo de Kafka?

La implementación de NFS de NetApp se considera un estándar de oro para el protocolo y se utiliza en innumerables entornos NAS empresariales. Además de la credibilidad de NetApp, también ofrece los siguientes beneficios:

- Fiabilidad y eficiencia
- Escalabilidad y rendimiento
- Alta disponibilidad (socio de alta disponibilidad en un clúster NetApp ONTAP)
- Protección de datos
  - **Recuperación ante desastres (NetApp SnapMirror).** Su sitio se cae o desea comenzar en un sitio diferente y continuar desde donde lo dejó.
  - Gestionabilidad de su sistema de almacenamiento (administración y gestión mediante NetApp OnCommand).
  - **Equilibrio de carga.** El clúster le permite acceder a diferentes volúmenes de LIF de datos alojados en diferentes nodos.
  - **Operaciones sin interrupciones.** Los LIF o movimientos de volumen son transparentes para los clientes NFS.

Solución de NetApp para el problema del cambio de nombre en cargas de trabajo de NFS a Kafka

Kafka está construido bajo el supuesto de que el sistema de archivos subyacente es compatible con POSIX: por ejemplo, XFS o Ext4. El reequilibrio de recursos de Kafka elimina archivos mientras la aplicación aún los está utilizando. Un sistema de archivos compatible con POSIX permite que la desvinculación continúe. Sin embargo, solo elimina el archivo después de que desaparezcan todas las referencias al mismo. Si el sistema de archivos subyacente está conectado a la red, entonces el cliente NFS intercepta las llamadas de desvinculación y administra el flujo de trabajo. Debido a que hay aperturas pendientes en el archivo que se va a desvincular, el cliente NFS envía una solicitud de cambio de nombre al servidor NFS y, en el último cierre del archivo desvinculado, emite una operación de eliminación en el archivo renombrado. Este comportamiento se conoce comúnmente como cambio de nombre tonto de NFS y está orquestado por el cliente NFS.

Cualquier agente de Kafka que utilice almacenamiento de un servidor NFSv3 tendrá problemas debido a este comportamiento. Sin embargo, el protocolo NFSv4.x tiene características para abordar este problema al permitir que el servidor se haga responsable de los archivos abiertos y no vinculados. Los servidores NFS que admiten esta función opcional comunican la capacidad de propiedad al cliente NFS en el momento de la

apertura del archivo. El cliente NFS luego detiene la gestión de desvinculación cuando hay aperturas pendientes y permite que el servidor administre el flujo. Aunque la especificación NFSv4 proporciona pautas para la implementación, hasta ahora no se conocía ninguna implementación de servidor NFS que admitiera esta característica opcional.

Se requieren los siguientes cambios para el servidor NFS y el cliente NFS para solucionar el problema del cambio de nombre tonto:

- **Cambios en el cliente NFS (Linux).** En el momento de abrir un archivo, el servidor NFS responde con una bandera, indicando la capacidad de manejar la desvinculación de los archivos abiertos. Los cambios del lado del cliente NFS permiten que el servidor NFS maneje la desvinculación en presencia del indicador. NetApp ha actualizado el cliente NFS Linux de código abierto con estos cambios. El cliente NFS actualizado ahora está disponible de forma general en RHEL8.7 y RHEL9.1.
- **Cambios en el servidor NFS.** El servidor NFS realiza un seguimiento de las aperturas. La desvinculación de un archivo abierto existente ahora es administrada por el servidor para que coincida con la semántica POSIX. Cuando se cierra el último archivo abierto, el servidor NFS inicia la eliminación real del archivo y evita así el tonto proceso de cambio de nombre. El servidor NFS de ONTAP ha implementado esta capacidad en su última versión, ONTAP 9.12.1.

Con los cambios mencionados anteriormente en el cliente y el servidor NFS, Kafka puede aprovechar de forma segura todos los beneficios del almacenamiento NFS conectado a la red.

Validación funcional: Solución para un cambio de nombre tonto

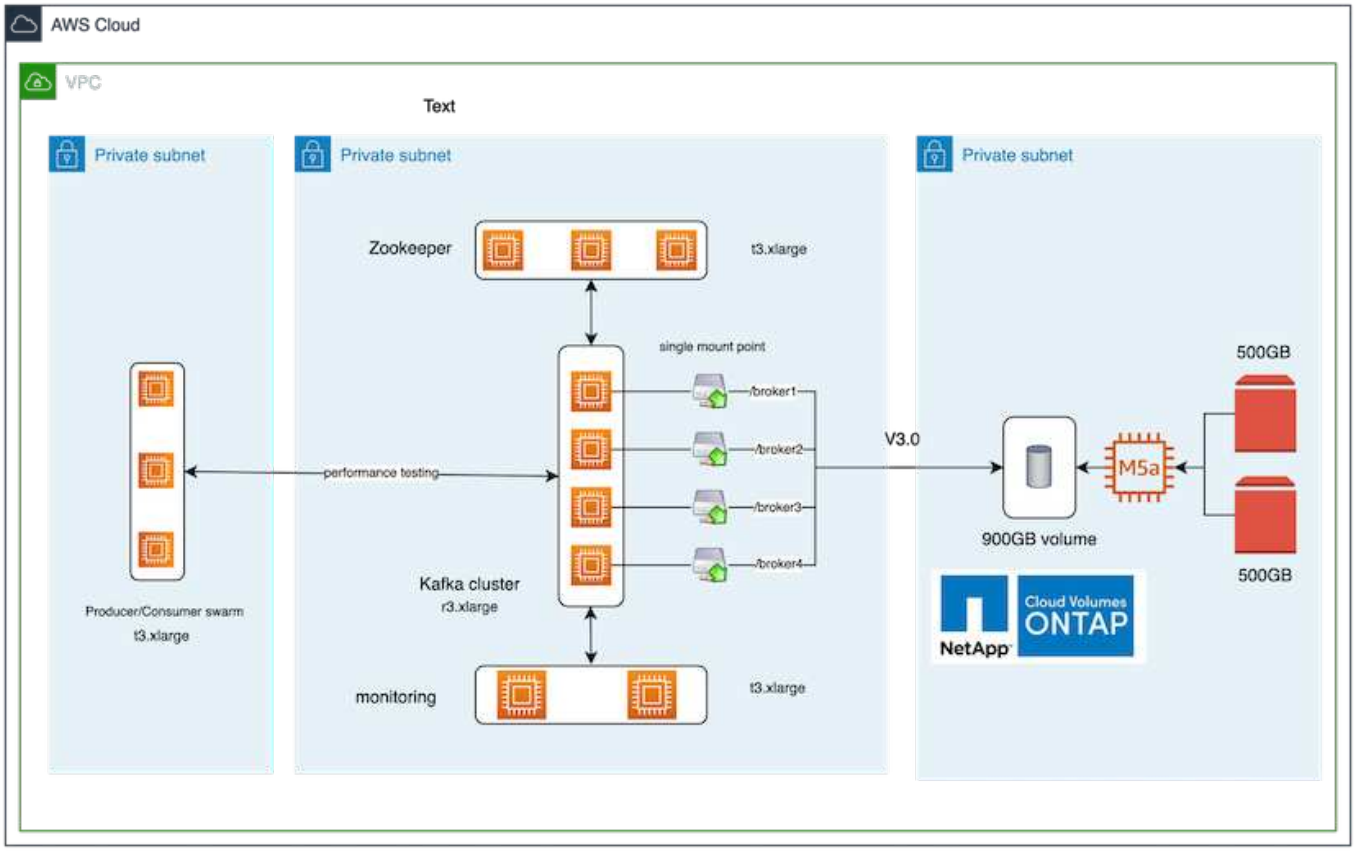
Para la validación funcional, demostramos que un clúster de Kafka con un montaje NFSv3 para almacenamiento no puede realizar operaciones de Kafka como la redistribución de particiones, mientras que otro clúster montado en NFSv4 con la solución puede realizar las mismas operaciones sin interrupciones.

Configuración de validación

La configuración se ejecuta en AWS. La siguiente tabla muestra los diferentes componentes de la plataforma y la configuración ambiental utilizados para la validación.

Componente de plataforma	Configuración del entorno
Plataforma Confluent versión 7.2.1	• 3 x cuidadores del zoológico – t3.xlarge • 4 servidores intermediarios – r3.xlarge • 1 x Grafana – t3.xlarge • 1 x centro de control – t3.xlarge • 3 x Productor/consumidor
Sistema operativo en todos los nodos	RHEL8.7 o posterior
Instancia NetApp Cloud Volumes ONTAP	Instancia de un solo nodo – M5.2xLarge

La siguiente figura muestra la configuración arquitectónica para esta solución.



Flujo arquitectónico

- **Calcular.** Utilizamos un clúster Kafka de cuatro nodos con un conjunto Zookeeper de tres nodos ejecutándose en servidores dedicados.
- **Escucha.** Utilizamos dos nodos para una combinación Prometheus-Grafana.
- **Carga de trabajo.** Para generar cargas de trabajo, utilizamos un clúster separado de tres nodos que puede producir y consumir desde este clúster de Kafka.
- **Almacenamiento.** Utilizamos una instancia de NetApp Cloud Volumes ONTAP de un solo nodo con dos volúmenes AWS-EBS GP2 de 500 GB conectados a la instancia. Luego, estos volúmenes se expusieron al clúster de Kafka como un único volumen NFSv4.1 a través de un LIF.

Se eligieron las propiedades predeterminadas de Kafka para todos los servidores. Lo mismo se hizo con el enjambre de cuidadores del zoológico.

Metodología de pruebas

1. Actualizar `-is-preserve-unlink-enabled true` Al volumen de Kafka, como sigue:

```
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. Se crearon dos clústeres de Kafka similares con la siguiente diferencia:
   - **Grupo 1.** El servidor backend NFS v4.1 que ejecutaba ONTAP versión 9.12.1 lista para producción estaba alojado en una instancia CVO de NetApp . Se instalaron RHEL 8.7/RHEL 9.1 en los brokers.
   - **Grupo 2.** El servidor NFS backend era un servidor NFSv3 Linux genérico creado manualmente.
3. Se creó un tema de demostración en ambos clústeres de Kafka.

   Grupo 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:
```

   Grupo 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. Se cargaron datos en estos temas recién creados para ambos clústeres. Esto se hizo utilizando el kit de herramientas de prueba de rendimiento del productor que viene en el paquete predeterminado de 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. Se realizó una comprobación del estado del broker-1 para cada uno de los clústeres mediante telnet:
   - Telnet `172.30.0.160 9092`
   - Telnet `172.30.0.198 9092`

     En la siguiente captura de pantalla se muestra una verificación de estado exitosa de los corredores en ambos clústeres:

6. Para activar la condición de error que provoca que los clústeres de Kafka que usan volúmenes de almacenamiento NFSv3 se bloqueen, iniciamos el proceso de reasignación de particiones en ambos clústeres. La reasignación de particiones se realizó utilizando `kafka-reassign-partitions.sh` . El proceso detallado es el siguiente:
   a. Para reasignar las particiones de un tema en un clúster de Kafka, generamos el JSON de configuración de reasignación propuesto (esto se realizó para ambos clústeres).

```
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. Luego, el JSON de reasignación generado se guardó en `/tmp/reassignment- file.json` .
   c. El proceso de reasignación de partición real se activó mediante el siguiente 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. Después de unos minutos, cuando se completó la reasignación, otra verificación del estado de los intermediarios mostró que el clúster que usaba volúmenes de almacenamiento NFSv3 se había topado con un problema de cambio de nombre tonto y se había bloqueado, mientras que el clúster 1 que usaba volúmenes de almacenamiento NetApp ONTAP NFSv4.1 con la solución continuó con sus operaciones sin interrupciones.

```
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
```

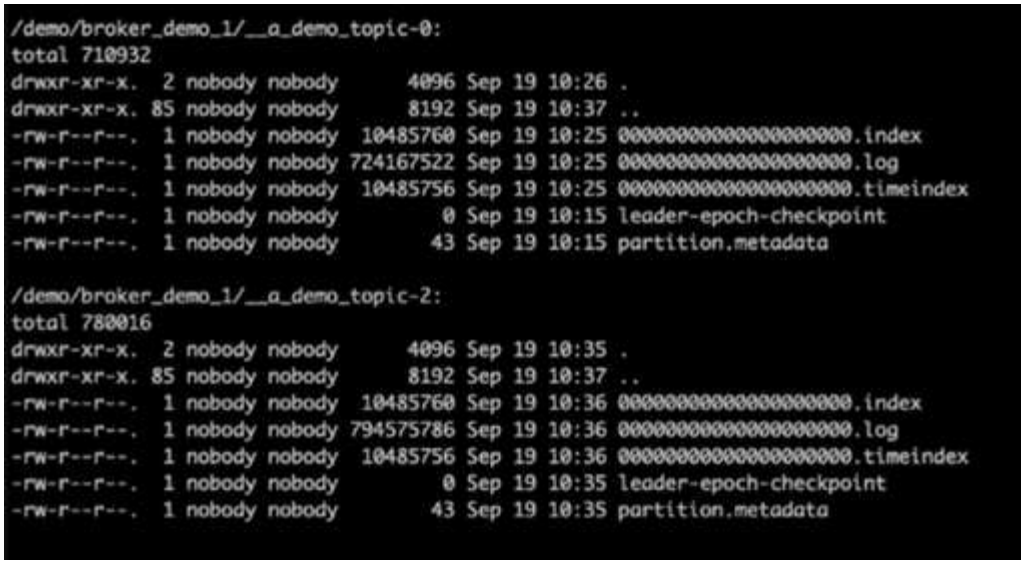
- Cluster1-Broker-1 está activo.
- Cluster2-broker-1 está muerto.

8. Al verificar los directorios de registro de Kafka, quedó claro que el Clúster 1 que usaba volúmenes de almacenamiento NetApp ONTAP NFSv4.1 con la corrección tenía una asignación de partición limpia, mientras que el Clúster 2 que usaba almacenamiento NFSv3 genérico no la tenía debido a problemas de cambio de nombre tontos, que provocaron el bloqueo. La siguiente imagen muestra el reequilibrio de la partición del Clúster 2, lo que provocó un problema de cambio de nombre tonto en el almacenamiento 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 siguiente imagen muestra un reequilibrio de partición limpio del Clúster 1 utilizando almacenamiento NetApp NFSv4.1.



¿Por qué NetApp NFS para cargas de trabajo de Kafka?

Ahora que existe una solución para el problema del cambio de nombre tonto en el almacenamiento NFS con Kafka, puede crear implementaciones sólidas que aprovechen el almacenamiento NetApp ONTAP para su carga de trabajo de Kafka. Esto no solo reduce significativamente los gastos operativos, sino que también aporta los siguientes beneficios a sus clústeres de Kafka:

- **Utilización reducida de CPU en los brókers de Kafka.** El uso de almacenamiento desagregado de NetApp ONTAP separa las operaciones de E/S de disco del agente y, de este modo, reduce su huella de CPU.
- **Tiempo de recuperación del corredor más rápido.** Dado que el almacenamiento desagregado de NetApp ONTAP se comparte entre los nodos del agente de Kafka, una nueva instancia de cómputo puede reemplazar a un agente defectuoso en cualquier momento en una fracción del tiempo en comparación con las implementaciones convencionales de Kafka sin reconstruir los datos.
- **Eficiencia de almacenamiento.** Como la capa de almacenamiento de la aplicación ahora se aprovisiona a través de NetApp ONTAP, los clientes pueden aprovechar todos los beneficios de eficiencia de almacenamiento que viene con ONTAP, como compresión de datos en línea, deduplicación y compactación.

Estos beneficios fueron probados y validados en casos de prueba que discutimos en detalle en esta sección.

Uso reducido de la CPU en el bróker de Kafka

Descubrimos que la utilización general de la CPU es menor que la de su contraparte DAS cuando ejecutamos cargas de trabajo similares en dos clústeres de Kafka separados que eran idénticos en sus especificaciones técnicas pero diferían en sus tecnologías de almacenamiento. No solo la utilización general de la CPU es menor cuando el clúster de Kafka usa almacenamiento ONTAP , sino que el aumento en la utilización de la CPU demostró un gradiente más suave que en un clúster de Kafka basado en DAS.

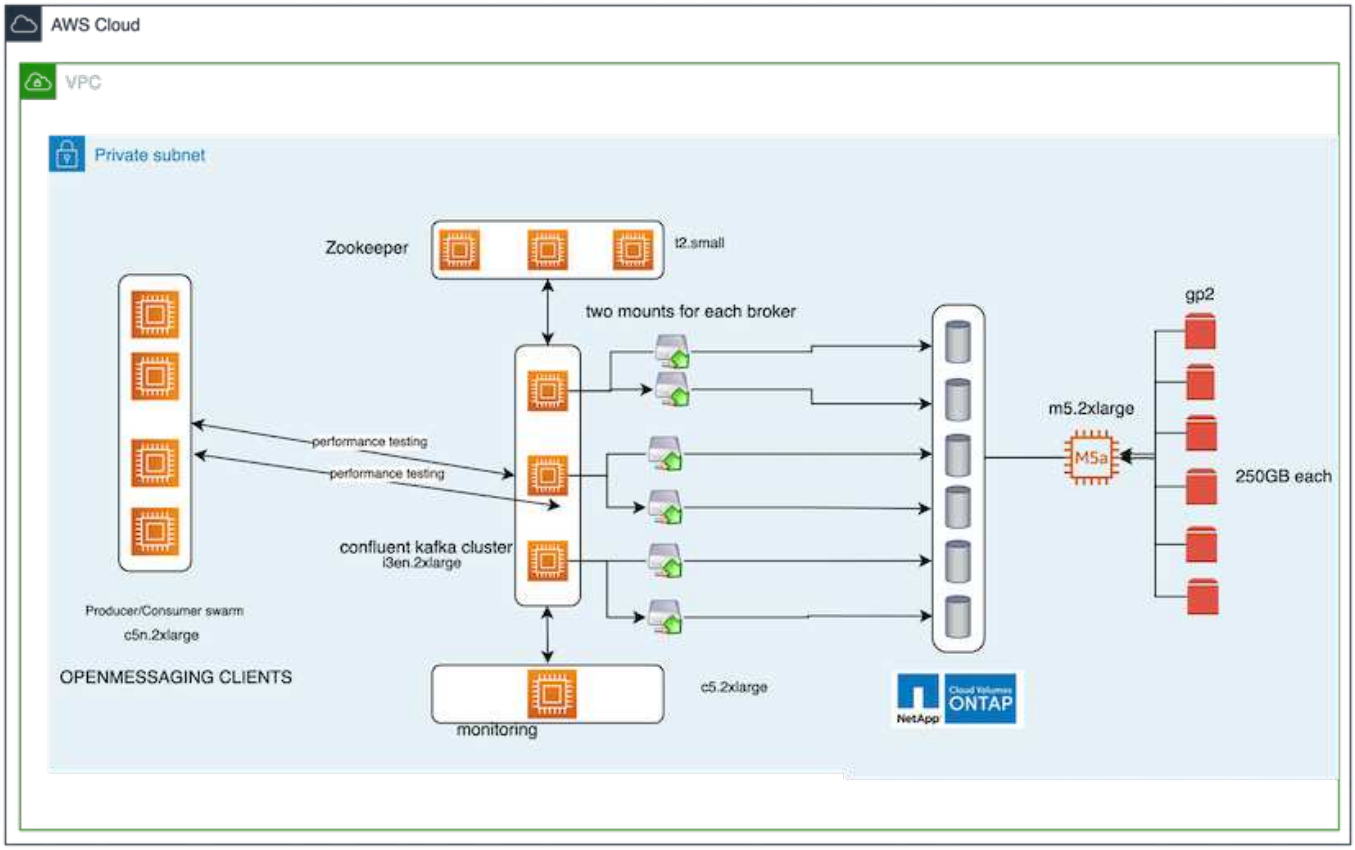
Configuración arquitectónica

La siguiente tabla muestra la configuración ambiental utilizada para demostrar una utilización reducida de la CPU.

Componente de plataforma	Configuración del entorno
Herramienta de evaluación comparativa de Kafka 3.2.3: OpenMessaging	• 3 x cuidadores del zoológico – t2.small • 3 servidores intermediarios – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x Productor/Consumidor — c5n.2xlarge
Sistema operativo en todos los nodos	RHEL 8.7 o posterior
Instancia NetApp Cloud Volumes ONTAP	Instancia de nodo único – M5.2xLarge

Herramienta de evaluación comparativa

La herramienta de evaluación comparativa utilizada en este caso de prueba es la "Mensajería abierta" estructura. OpenMessaging es neutral respecto de proveedores e independiente del lenguaje; proporciona pautas industriales para finanzas, comercio electrónico, IoT y big data; y ayuda a desarrollar aplicaciones de mensajería y transmisión en sistemas y plataformas heterogéneos. La siguiente figura muestra la interacción de los clientes de OpenMessaging con un clúster de Kafka.



- **Calcular.** Utilizamos un clúster Kafka de tres nodos con un conjunto Zookeeper de tres nodos ejecutándose en servidores dedicados. Cada agente tenía dos puntos de montaje NFSv4.1 en un solo volumen en la instancia CVO de NetApp a través de un LIF dedicado.
- **Escucha.** Utilizamos dos nodos para una combinación Prometheus-Grafana. Para generar cargas de trabajo, tenemos un clúster separado de tres nodos que puede producir y consumir desde este clúster de Kafka.

- **Almacenamiento.** Utilizamos una instancia de NetApp Cloud Volumes ONTAP de un solo nodo con seis volúmenes AWS-EBS GP2 de 250 GB montados en la instancia. Luego, estos volúmenes se expusieron al clúster de Kafka como seis volúmenes NFSv4.1 a través de LIF dedicados.
- **Configuración.** Los dos elementos configurables en este caso de prueba fueron los agentes de Kafka y las cargas de trabajo de OpenMessaging.
    - **Configuración del corredor.** Se seleccionaron las siguientes especificaciones para los corredores de Kafka. Utilizamos un factor de replicación de 3 para todas las mediciones, como se destaca a continuación.

```
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
```

- **Configuración de carga de trabajo de referencia de OpenMessaging (OMB).** Se proporcionaron las siguientes especificaciones: Especificamos una tasa de productor objetivo, resaltada a continuación.

```
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
```

Metodología de pruebas

1. Se crearon dos clústeres similares, cada uno con su propio conjunto de enjambres de clústeres de evaluación comparativa.

- **Grupo 1.** Clúster Kafka basado en NFS.
- **Grupo 2.** Clúster Kafka basado en DAS.

2. Usando un comando OpenMessaging, se activaron cargas de trabajo similares en cada clúster.

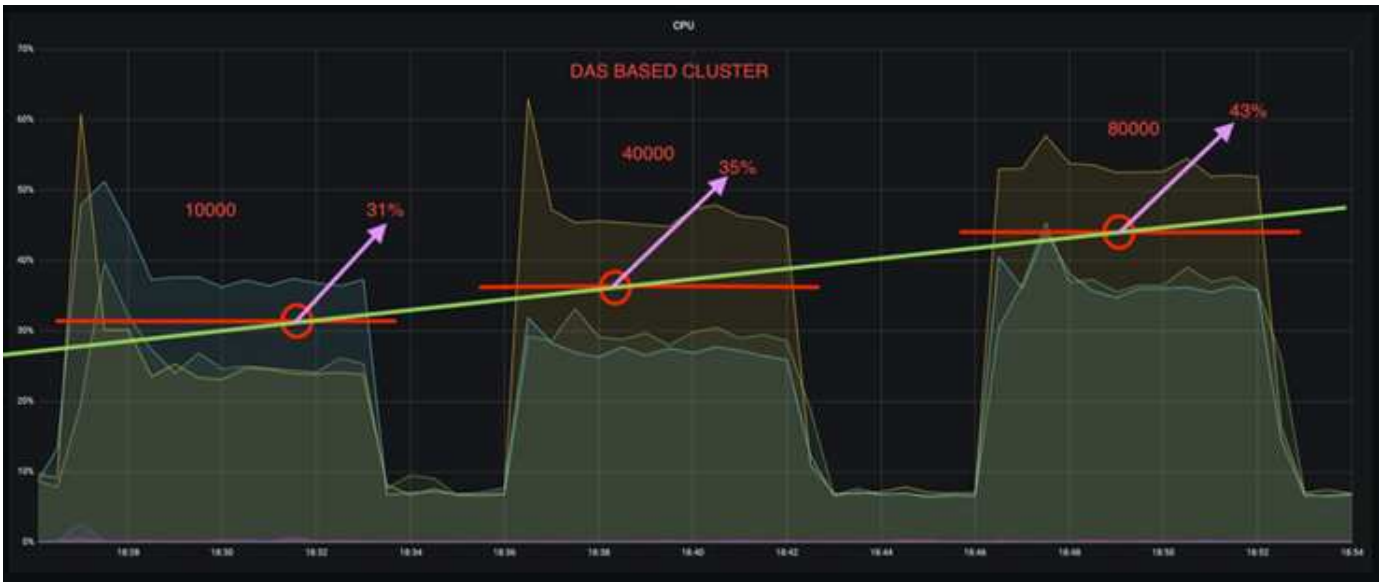
```
sudo bin/benchmark --drivers driver-kafka/kafka-group-all.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

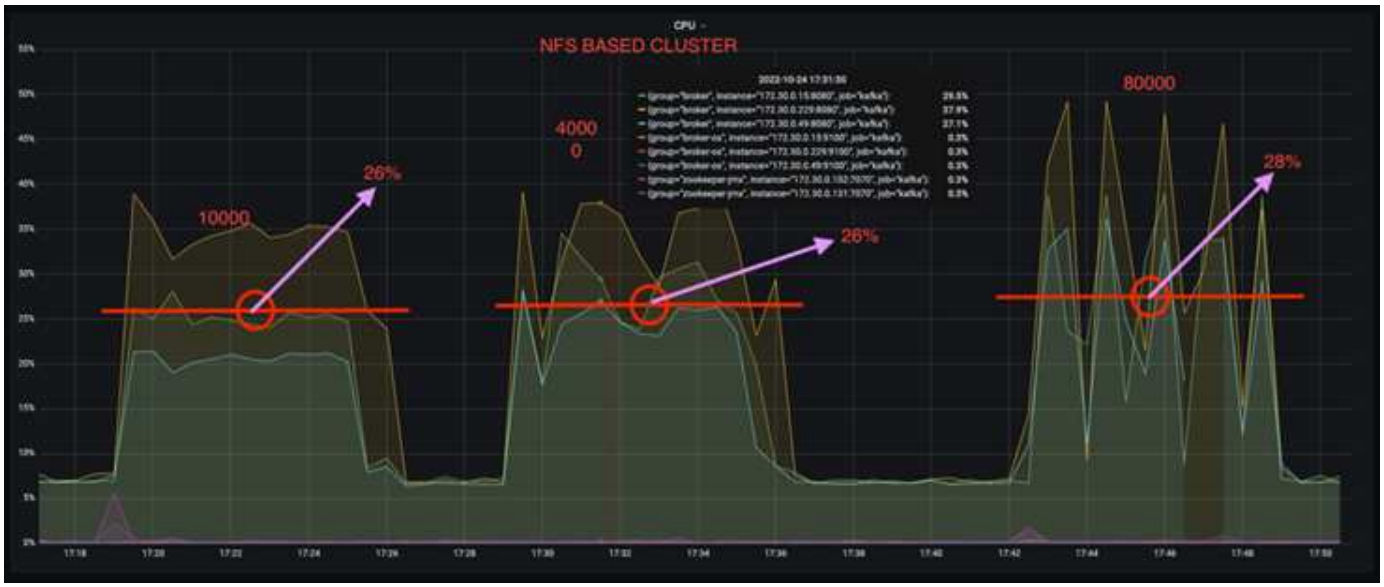
3. La configuración de la tasa de producción se incrementó en cuatro iteraciones y la utilización de la CPU se registró con Grafana. La tasa de producción se fijó en los siguientes niveles:
   - 10.000
   - 40.000
   - 80.000
   - 100.000

Observación

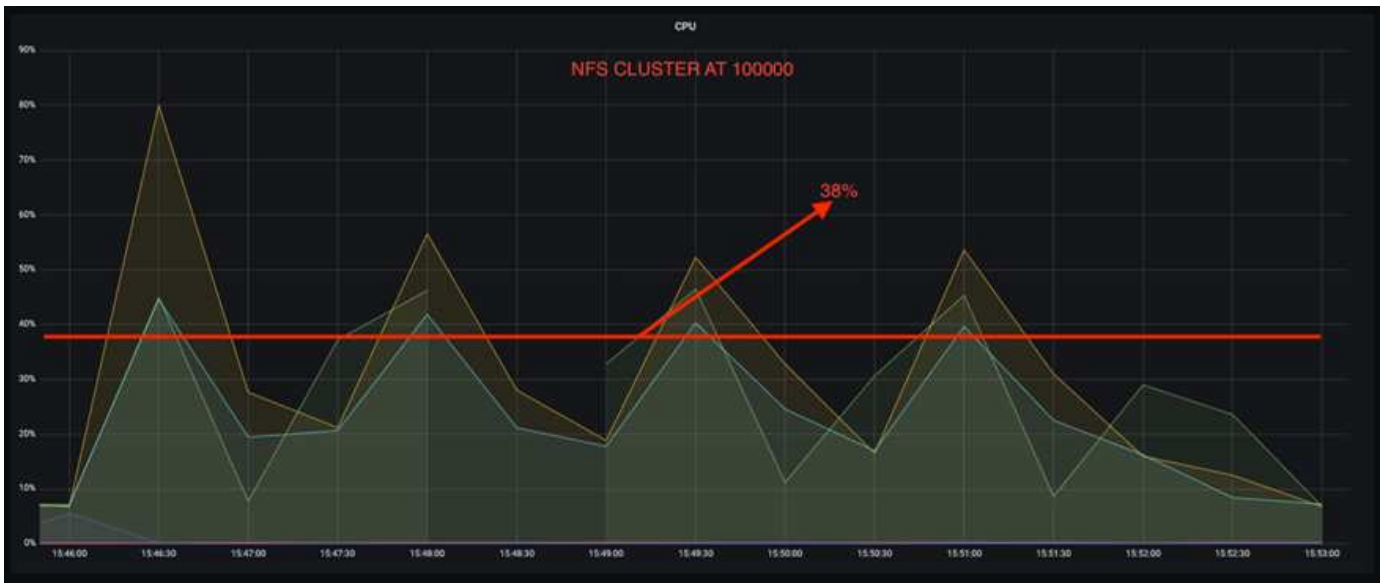
Hay dos beneficios principales de usar almacenamiento NFS de NetApp con Kafka:

- **Puede reducir el uso de la CPU en casi un tercio.** El uso general de la CPU bajo cargas de trabajo similares fue menor para NFS en comparación con los SSD DAS; los ahorros varían del 5 % para tasas de producción más bajas al 32 % para tasas de producción más altas.
- **Una reducción de tres veces en la deriva de utilización de la CPU a tasas de producción más altas.** Como era de esperar, hubo una tendencia ascendente en el aumento de la utilización de la CPU a medida que aumentaron las tasas de producción. Sin embargo, la utilización de la CPU en los brókers de Kafka que usan DAS aumentó del 31 % para la tasa de producción más baja al 70 % para la tasa de producción más alta, un aumento del 39 %. Sin embargo, con un backend de almacenamiento NFS, la utilización de la CPU aumentó del 26% al 38%, un aumento del 12%.





Además, con 100.000 mensajes, DAS muestra una mayor utilización de la CPU que un clúster NFS.



Recuperación más rápida del corredor

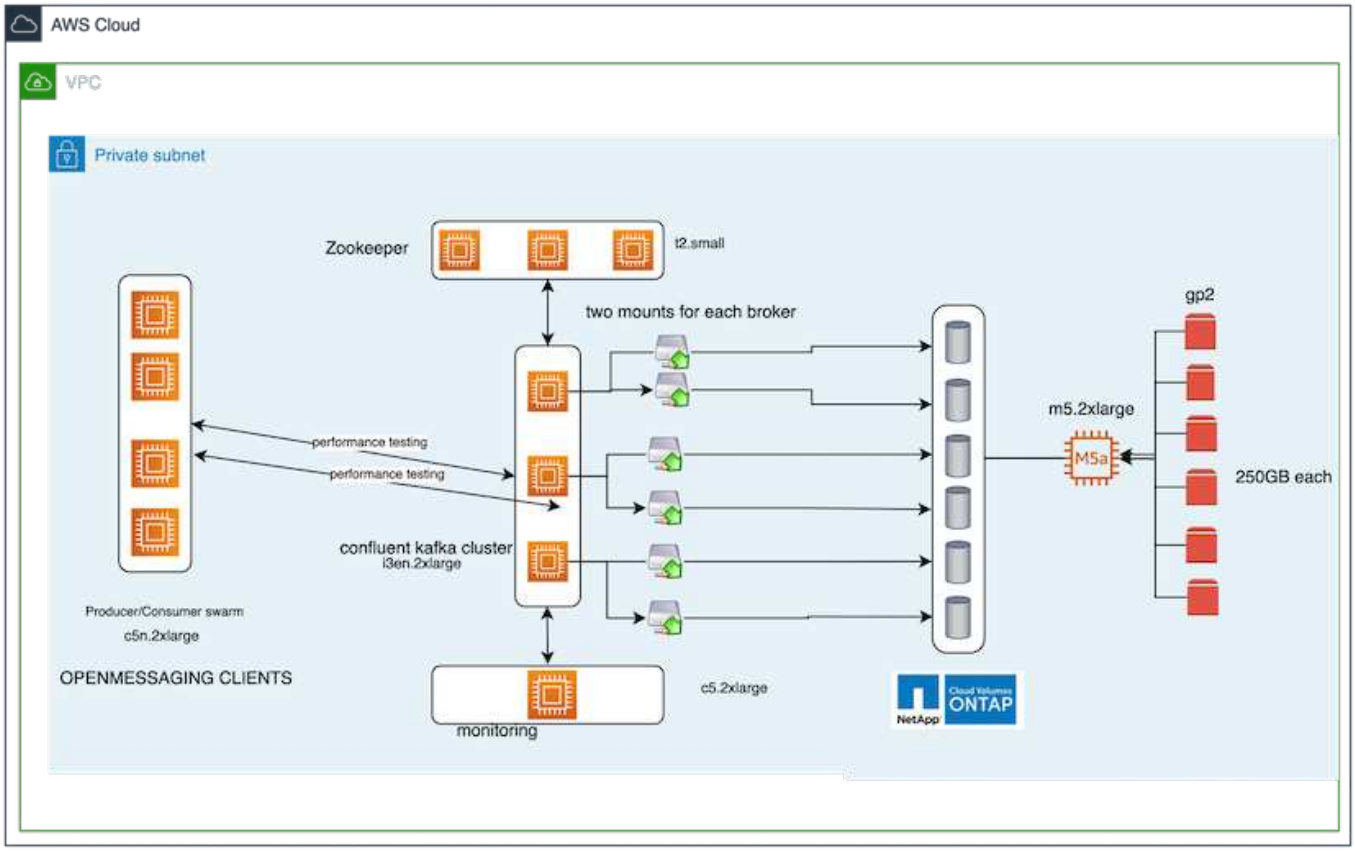
Descubrimos que los agentes de Kafka se recuperan más rápido cuando utilizan almacenamiento NFS compartido de NetApp . Cuando un broker falla en un clúster de Kafka, este broker puede ser reemplazado por un broker en buen estado con el mismo ID de broker. Al realizar este caso de prueba, descubrimos que, en el caso de un clúster de Kafka basado en DAS, el clúster reconstruye los datos en un agente en buen estado recién agregado, lo que consume mucho tiempo. En el caso de un clúster Kafka basado en NFS de NetApp , el agente de reemplazo continúa leyendo datos del directorio de registro anterior y se recupera mucho más rápido.

Configuración arquitectónica

La siguiente tabla muestra la configuración ambiental para un clúster de Kafka que utiliza NAS.

Componente de plataforma	Configuración del entorno
Kafka 3.2.3	• 3 x cuidadores del zoológico – t2.small • 3 servidores intermediarios – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x productor/consumidor — c5n.2xlarge • 1 nodo de respaldo de Kafka – i3en.2xlarge
Sistema operativo en todos los nodos	RHEL8.7 o posterior
Instancia NetApp Cloud Volumes ONTAP	Instancia de un solo nodo – M5.2xLarge

La siguiente figura muestra la arquitectura de un clúster de Kafka basado en NAS.



- **Calcular.** Un clúster de Kafka de tres nodos con un conjunto Zookeeper de tres nodos que se ejecuta en servidores dedicados. Cada agente tiene dos puntos de montaje NFS en un solo volumen en la instancia CVO de NetApp a través de un LIF dedicado.
- **Escucha.** Dos nodos para una combinación Prometheus-Grafana. Para generar cargas de trabajo, utilizamos un clúster separado de tres nodos que puede producir y consumir en este clúster de Kafka.
- **Almacenamiento.** Una instancia de volúmenes NetApp Cloud ONTAP de un solo nodo con seis volúmenes AWS-EBS GP2 de 250 GB montados en la instancia. Luego, estos volúmenes se exponen al clúster de Kafka como seis volúmenes NFS a través de LIF dedicados.
- **Configuración del broker.** El único elemento configurable en este caso de prueba son los brokers de Kafka. Se seleccionaron las siguientes especificaciones para los corredores de Kafka. El `replica.lag.time.mx.ms` se establece en un valor alto porque esto determina qué tan rápido se saca un nodo particular de la lista ISR. Cuando cambia entre nodos defectuosos y saludables, no desea que ese ID de agente se excluya de la lista de 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
```

Metodología de pruebas

1. Se crearon dos clústeres similares:
   - Un clúster confluente basado en EC2.
   - Un clúster confluente basado en NFS de NetApp .
2. Se creó un nodo de Kafka en espera con una configuración idéntica a los nodos del clúster de Kafka original.
3. En cada uno de los clústeres, se creó un tema de muestra y se completaron aproximadamente 110 GB de datos en cada uno de los corredores.
   - **Clúster basado en EC2.** Un directorio de datos del corredor de Kafka está asignado a `/mnt/data-2` (En la siguiente figura, Broker-1 del cluster1 [terminal izquierda]).
   - * Clúster basado en NFS de NetApp .* Un directorio de datos del agente de Kafka está montado en el punto NFS `/mnt/data` (En la siguiente figura, Broker-1 del cluster2 [terminal derecha]).

4. En cada uno de los clústeres, se finalizó Broker-1 para desencadenar un proceso de recuperación de broker fallido.
5. Una vez finalizado el broker, su dirección IP se asignó como IP secundaria al broker en espera. Esto fue necesario porque un bróker en un clúster de Kafka se identifica por lo siguiente:
   - **Dirección IP.** Se asigna reasignando la IP del agente fallido al agente en espera.
   - **Identificación del corredor.** Esto se configuró en el agente en espera `server.properties` .
6. Tras la asignación de IP, se inició el servicio Kafka en el bróker en espera.
7. Después de un tiempo, se extrajeron los registros del servidor para verificar el tiempo que tomó generar datos en el nodo de reemplazo en el clúster.

Observación

La recuperación del corredor de Kafka fue casi nueve veces más rápida. Se descubrió que el tiempo necesario para recuperar un nodo de agente fallido era significativamente más rápido cuando se usaba almacenamiento compartido NFS de NetApp en comparación con el uso de SSD DAS en un clúster de Kafka. Para 1 TB de datos de temas, el tiempo de recuperación de un clúster basado en DAS fue de 48 minutos, en comparación con menos de 5 minutos para un clúster Kafka basado en NetApp-NFS.

Observamos que el clúster basado en EC2 tardó 10 minutos en reconstruir los 110 GB de datos en el nuevo nodo del agente, mientras que el clúster basado en NFS completó la recuperación en 3 minutos. También observamos en los registros que las compensaciones del consumidor para las particiones de EC2 eran 0, mientras que, en el clúster NFS, las compensaciones del consumidor se tomaron del agente anterior.

```
[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$)
[2022-10-31 08:55:55,170] INFO [LogLoader partition=test-topic-qbVsEZg-
0000-8, dir=/mnt/data-1] Loading producer state till offset 0 with message
format version 2 (kafka.log.UnifiedLog$)
```

Clúster basado en DAS

1. El nodo de respaldo se inició a las 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. El proceso de reconstrucción de datos finalizó a las 09:05:24.860. El procesamiento de 110 GB de datos requirió aproximadamente 10 minutos.

```
[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)
```

Clúster basado en NFS

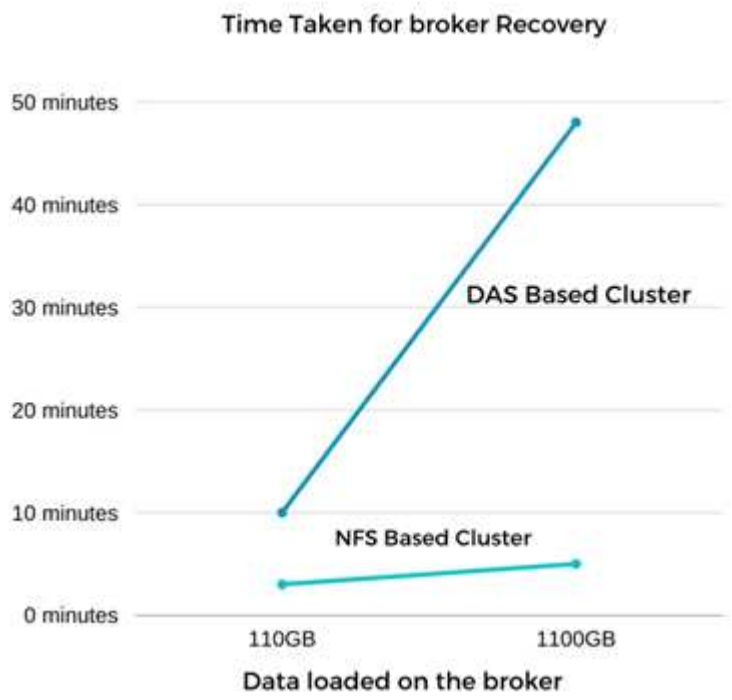
1. El nodo de respaldo se inició a las 09:39:17,213. La entrada del registro de inicio se resalta a continuación.

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

2. El proceso de reconstrucción de datos finalizó a las 09:42:29,115. El procesamiento de 110 GB de datos requirió aproximadamente 3 minutos.

```
[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)
```

La prueba se repitió para los corredores que contenían alrededor de 1 TB de datos, lo que tomó aproximadamente 48 minutos para DAS y 3 minutos para NFS. Los resultados se muestran en el siguiente gráfico.



Eficiencia de almacenamiento

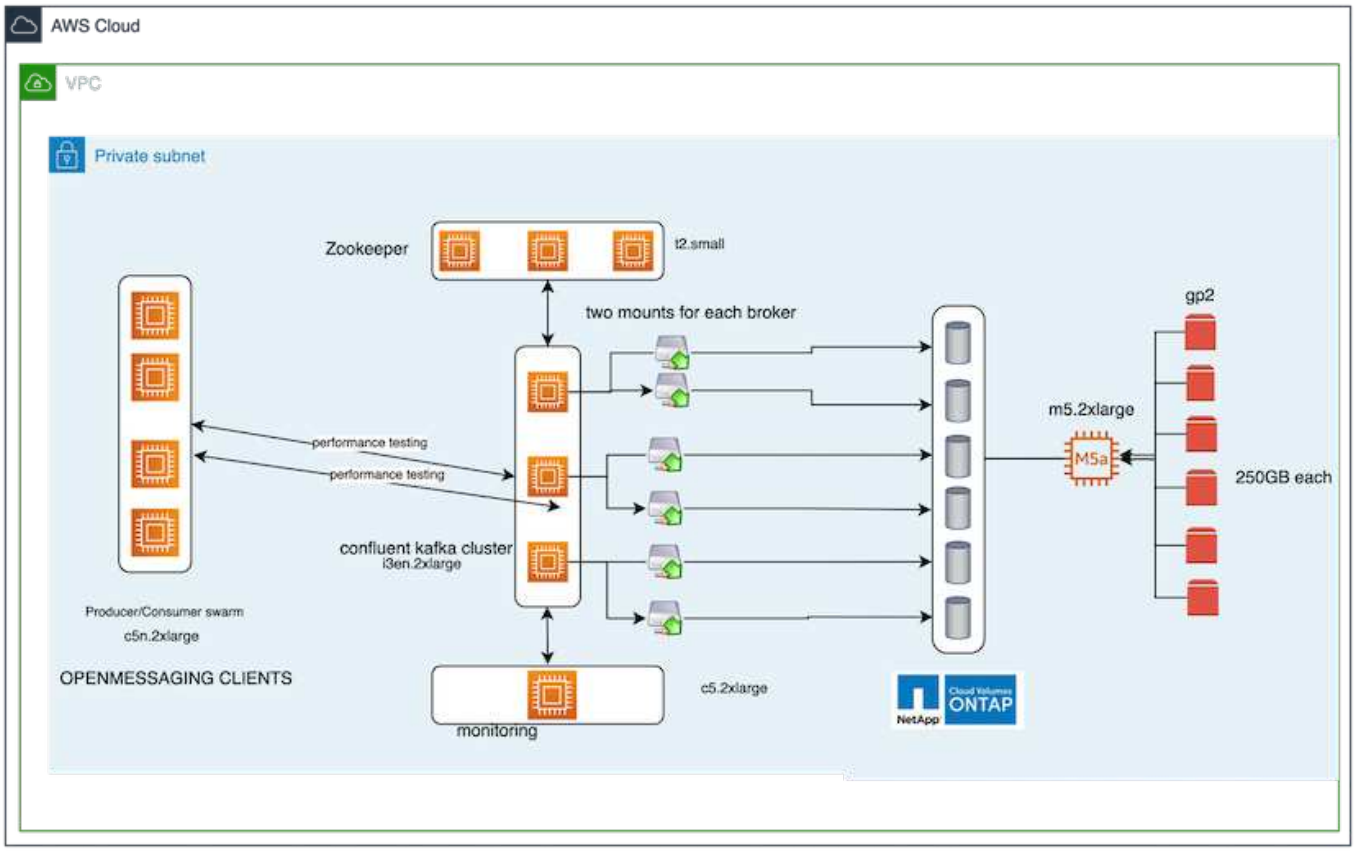
Debido a que la capa de almacenamiento del clúster Kafka se aprovisionó a través de NetApp ONTAP, obtuvimos todas las capacidades de eficiencia de almacenamiento de ONTAP. Esto se probó generando una cantidad significativa de datos en un clúster de Kafka con almacenamiento NFS aprovisionado en Cloud Volumes ONTAP. Pudimos ver que hubo una reducción de espacio significativa debido a las capacidades de ONTAP .

Configuración arquitectónica

La siguiente tabla muestra la configuración ambiental para un clúster de Kafka que utiliza NAS.

Componente de plataforma	Configuración del entorno
Kafka 3.2.3	• 3 x cuidadores del zoológico – t2.small • 3 servidores intermediarios – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x productor/consumidor — c5n.2xlarge *
Sistema operativo en todos los nodos	RHEL8.7 o posterior
Instancia NetApp Cloud Volumes ONTAP	Instancia de nodo único – M5.2xLarge

La siguiente figura muestra la arquitectura de un clúster de Kafka basado en NAS.



- **Calcular.** Utilizamos un clúster Kafka de tres nodos con un conjunto Zookeeper de tres nodos ejecutándose en servidores dedicados. Cada agente tenía dos puntos de montaje NFS en un solo volumen en la instancia CVO de NetApp a través de un LIF dedicado.
- **Escucha.** Utilizamos dos nodos para una combinación Prometheus-Grafana. Para generar cargas de trabajo, utilizamos un clúster separado de tres nodos que podía producir y consumir en este clúster de Kafka.
- **Almacenamiento.** Utilizamos una instancia de NetApp Cloud Volumes ONTAP de un solo nodo con seis volúmenes AWS-EBS GP2 de 250 GB montados en la instancia. Luego, estos volúmenes se expusieron al clúster de Kafka como seis volúmenes NFS a través de LIF dedicados.

- **Configuración.** Los elementos configurables en este caso de prueba fueron los brokers de Kafka.

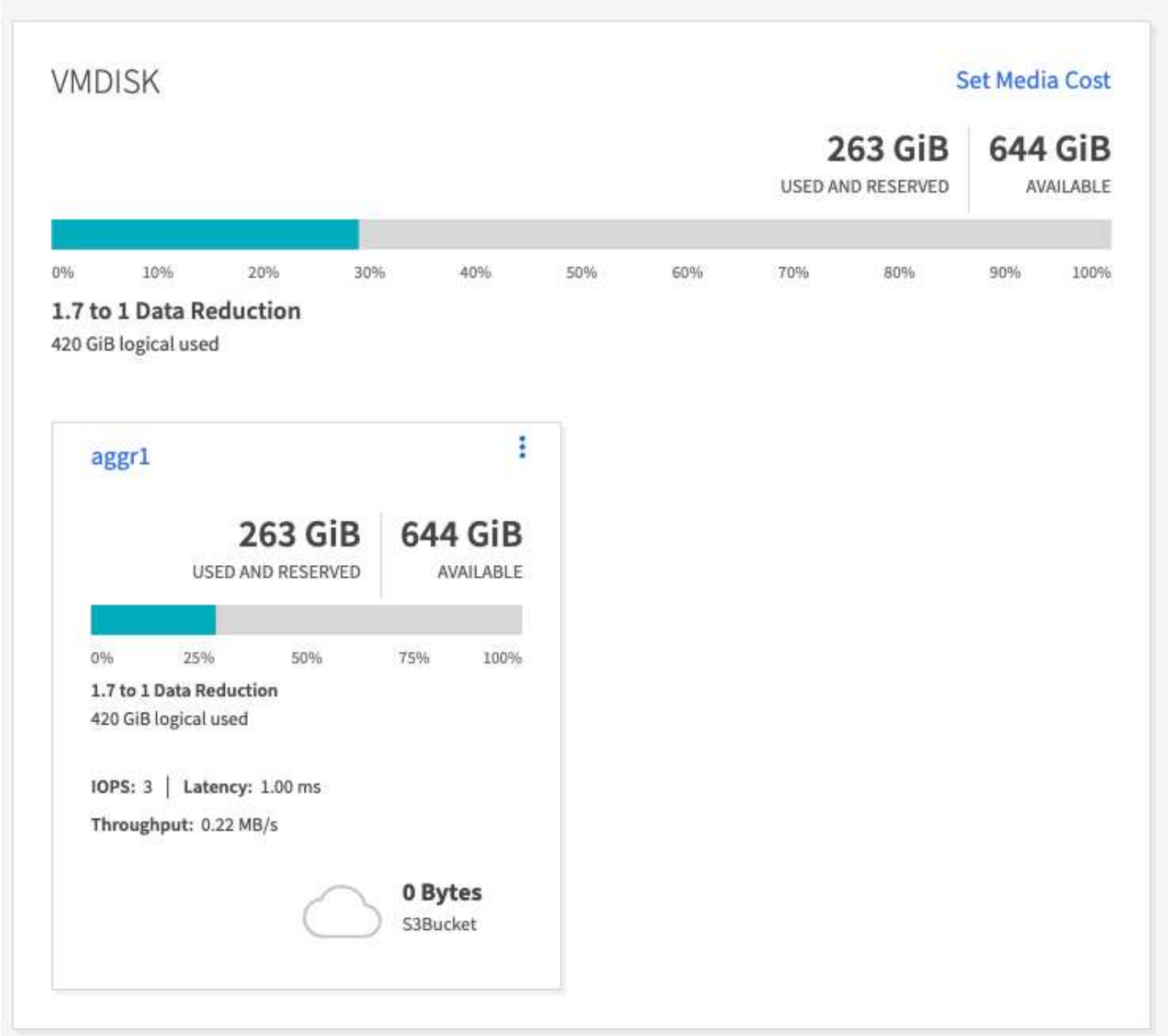
La compresión se desactivó en el extremo del productor, lo que permitió a los productores generar un alto rendimiento. La eficiencia del almacenamiento, en cambio, quedó a cargo de la capa de cómputo.

Metodología de pruebas

1. Se aprovisionó un clúster de Kafka con las especificaciones mencionadas anteriormente.
2. En el clúster, se produjeron alrededor de 350 GB de datos utilizando la herramienta OpenMessaging Benchmarking.
3. Una vez completada la carga de trabajo, se recopilaron las estadísticas de eficiencia de almacenamiento utilizando ONTAP System Manager y la CLI.

Observación

En el caso de los datos generados con la herramienta OMB, observamos un ahorro de espacio de aproximadamente un 33 % con una relación de eficiencia de almacenamiento de 1,70:1. Como se ve en las siguientes figuras, el espacio lógico utilizado por los datos producidos fue de 420,3 GB y el espacio físico utilizado para almacenar los datos fue de 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
```

Descripción general del rendimiento y validación en AWS

Se evaluó el rendimiento de un clúster de Kafka con la capa de almacenamiento montada en NetApp NFS en la nube de AWS. Los ejemplos de evaluación comparativa se describen en las siguientes secciones.

Kafka en la nube de AWS con NetApp Cloud Volumes ONTAP (par de alta disponibilidad y nodo único)

Se evaluó el rendimiento de un clúster de Kafka con NetApp Cloud Volumes ONTAP (par HA) en la nube de AWS. Esta evaluación comparativa se describe en las siguientes secciones.

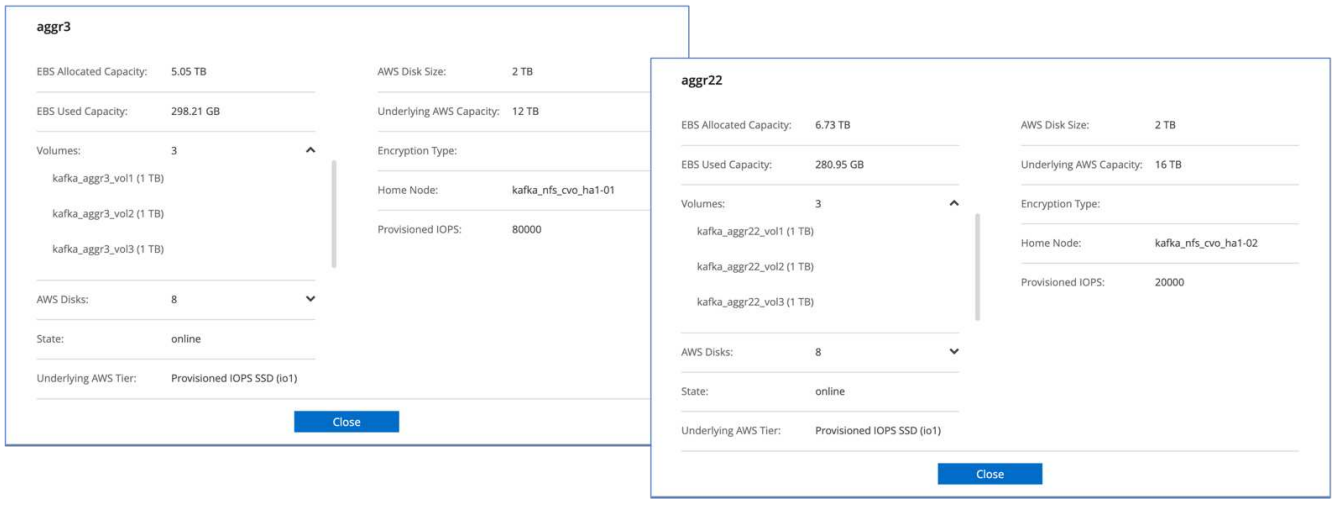
Configuración arquitectónica

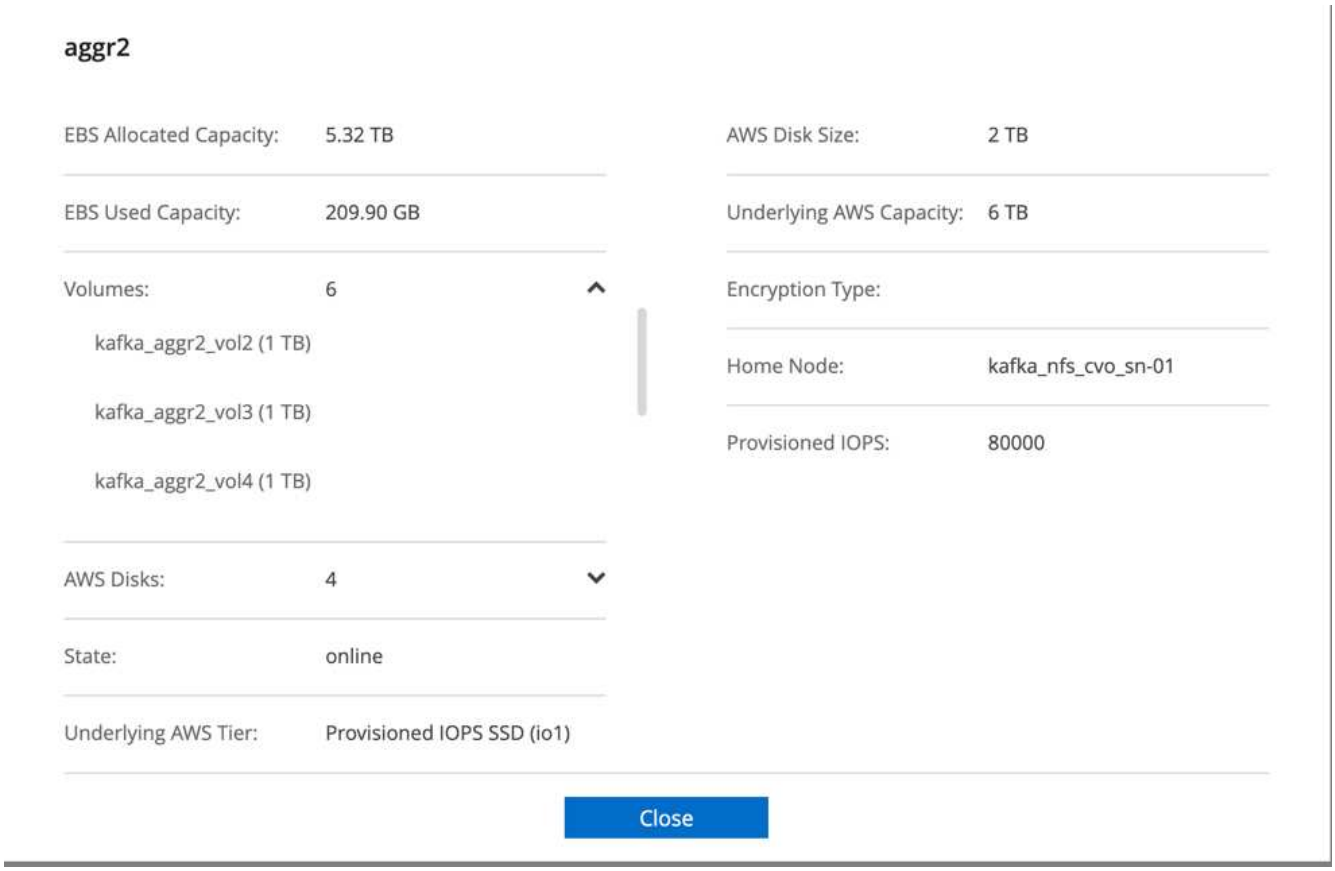
La siguiente tabla muestra la configuración ambiental para un clúster de Kafka que utiliza NAS.

Componente de plataforma	Configuración del entorno
Kafka 3.2.3	• 3 x cuidadores del zoológico – t2.small • 3 servidores intermediarios – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x productor/consumidor — c5n.2xlarge *
Sistema operativo en todos los nodos	RHEL8.6
Instancia NetApp Cloud Volumes ONTAP	Instancia de par HA: m5dn.12xLarge x 2 nodos Instancia de nodo único: m5dn.12xLarge x 1 nodo

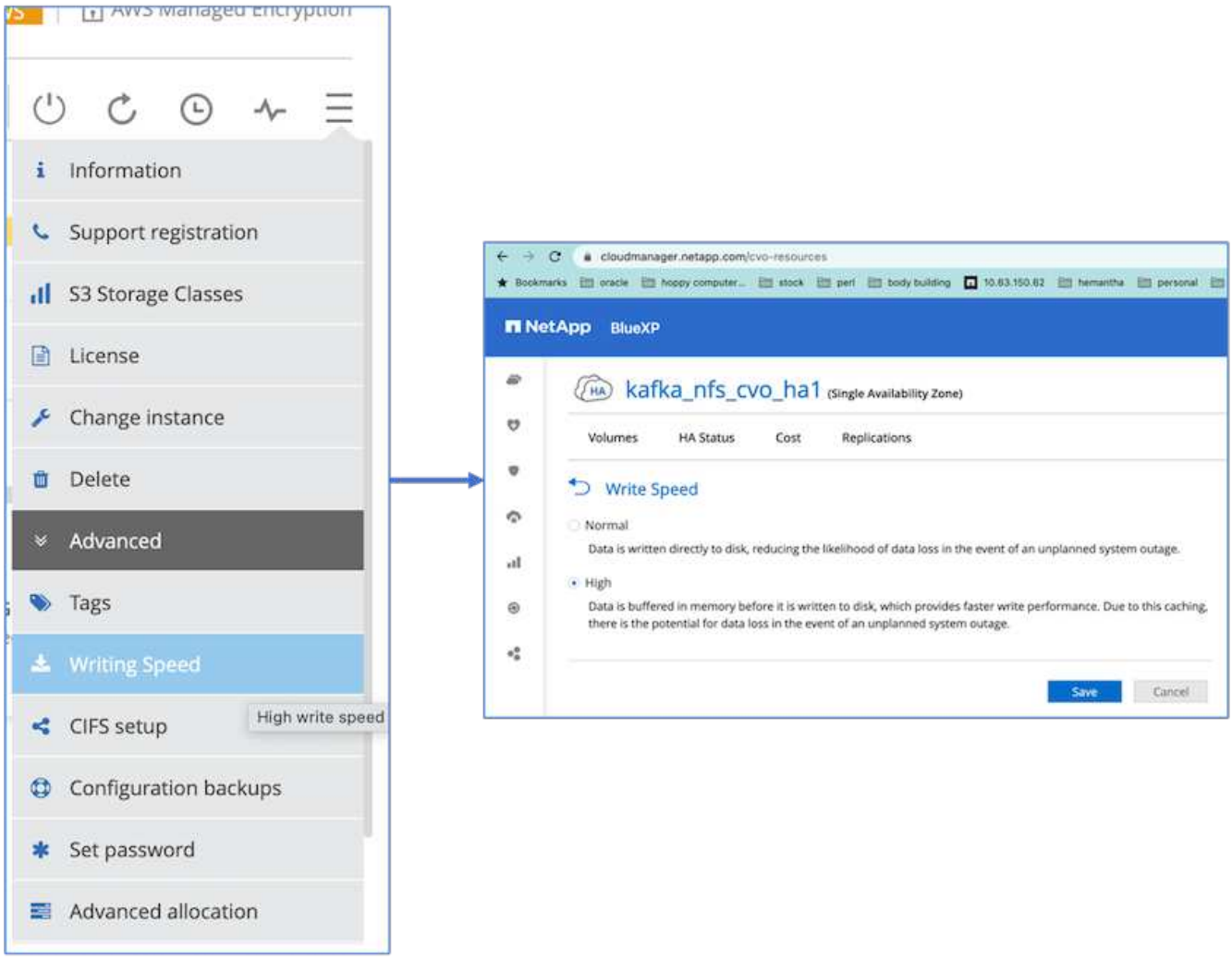
Configuración de ONTAP del volumen del clúster de NetApp

1. Para el par Cloud Volumes ONTAP HA, creamos dos agregados con tres volúmenes en cada agregado en cada controlador de almacenamiento. Para cada nodo de Cloud Volumes ONTAP , creamos seis volúmenes en total.



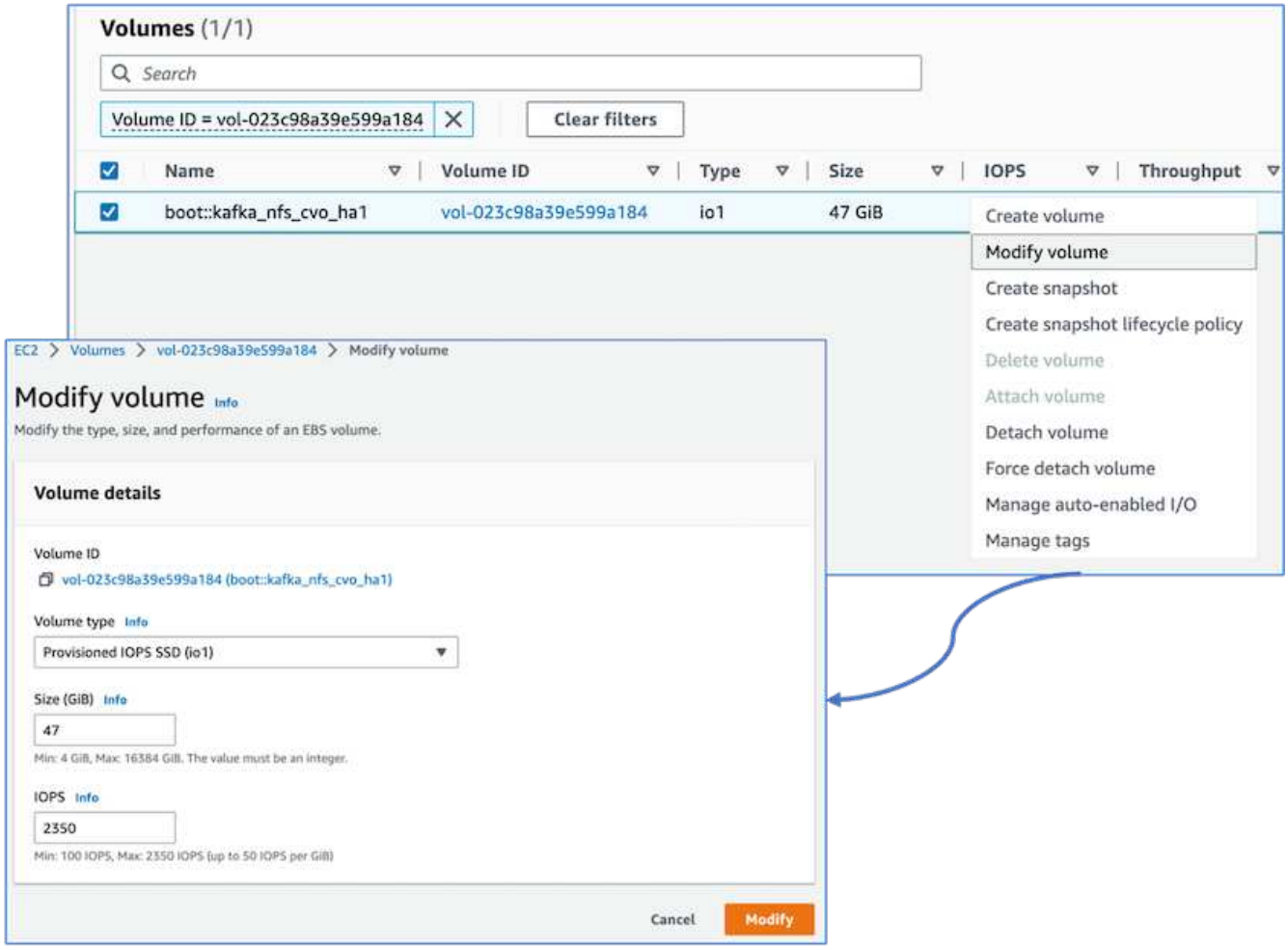


2. Para lograr un mejor rendimiento de la red, habilitamos redes de alta velocidad tanto para el par HA como para el nodo único.



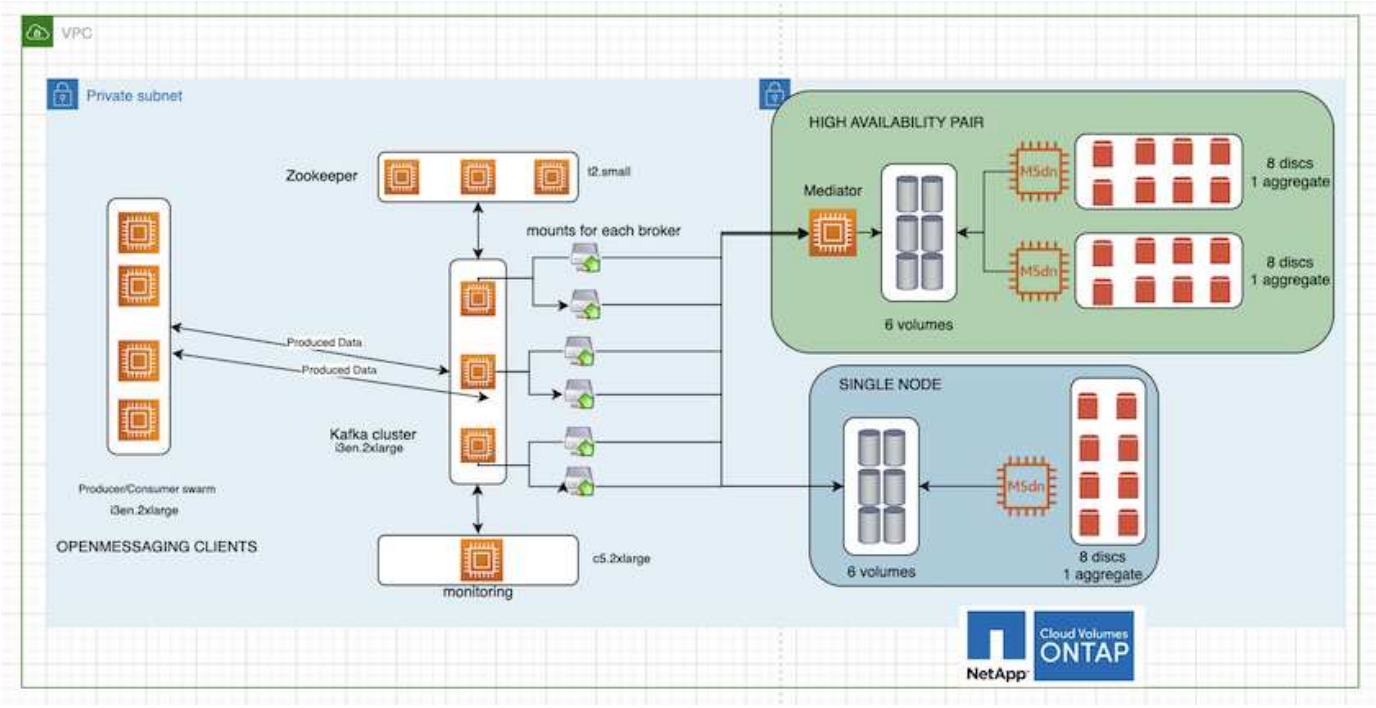
3. Notamos que la NVRAM de ONTAP tenía más IOPS, por lo que cambiamos las IOPS a 2350 para el volumen raíz de Cloud Volumes ONTAP . El disco de volumen raíz en Cloud Volumes ONTAP tenía un tamaño de 47 GB. El siguiente comando ONTAP es para el par HA y el mismo paso se aplica para el nodo único.

```
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 siguiente figura muestra la arquitectura de un clúster de Kafka basado en NAS.

- **Calcular.** Utilizamos un clúster Kafka de tres nodos con un conjunto Zookeeper de tres nodos ejecutándose en servidores dedicados. Cada agente tenía dos puntos de montaje NFS en un solo volumen en la instancia de Cloud Volumes ONTAP a través de un LIF dedicado.
- **Escucha.** Utilizamos dos nodos para una combinación Prometheus-Grafana. Para generar cargas de trabajo, utilizamos un clúster separado de tres nodos que podía producir y consumir en este clúster de Kafka.
- **Almacenamiento.** Utilizamos una instancia de Cloud Volumes ONTAP de par HA con un volumen AWS-EBS GP3 de 6 TB montado en la instancia. Luego, el volumen se exportó al broker de Kafka con un montaje NFS.



Configuraciones de evaluación comparativa de OpenMessage

1. Para un mejor rendimiento de NFS, necesitamos más conexiones de red entre el servidor NFS y el cliente NFS, que se pueden crear utilizando nconnect. Monte los volúmenes NFS en los nodos del agente con la opción nconnect ejecutando el siguiente 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. Verifique las conexiones de red en Cloud Volumes ONTAP. El siguiente comando ONTAP se utiliza desde el único nodo Cloud Volumes ONTAP . El mismo paso se aplica al par 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. Usamos el siguiente Kafka `server.properties` en todos los brokers de Kafka para el par Cloud Volumes ONTAP HA. El `log.dirs` La propiedad es diferente para cada corredor y las propiedades restantes son comunes para los corredores. Para el broker1, el `log.dirs` El valor es el siguiente:

```
[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 ~]#
```

   - Para el broker2, el `log.dirs` El valor de la propiedad es el siguiente:

```
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
```

   - Para el broker3, el `log.dirs` El valor de la propiedad es el siguiente:

```
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. Para el nodo único de Cloud Volumes ONTAP , Kafka `servers.properties` es lo mismo que para el par Cloud Volumes ONTAP HA excepto por el `log.dirs` propiedad.
   - Para el broker1, el `log.dirs` El valor es el siguiente:

```
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
```

   - Para el broker2, el `log.dirs` El valor es el siguiente:

```
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
```

   - Para el broker3, el `log.dirs` El valor de la propiedad es el siguiente:

```
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. La carga de trabajo en la OMB está configurada con las siguientes propiedades: `(/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
```

El `messageSize` Puede variar para cada caso de uso. En nuestra prueba de rendimiento, utilizamos 3K.

Utilizamos dos controladores diferentes, Sync o Throughput, de OMB para generar la carga de trabajo en el clúster de Kafka.

- El archivo yaml utilizado para las propiedades del controlador de sincronización es el siguiente (`/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
```

- El archivo yaml utilizado para las propiedades del controlador de rendimiento es el siguiente (`/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
```

Metodología de pruebas

1. Se aprovisionó un clúster de Kafka según la especificación descrita anteriormente utilizando Terraform y Ansible. Terraform se utiliza para construir la infraestructura utilizando instancias de AWS para el clúster de Kafka y Ansible construye el clúster de Kafka en ellas.
2. Se activó una carga de trabajo OMB con la configuración de carga de trabajo descrita anteriormente y el controlador de sincronización.

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

3. Se activó otra carga de trabajo con el controlador de rendimiento con la misma configuración de carga de trabajo.

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

Observación

Se utilizaron dos tipos diferentes de controladores para generar cargas de trabajo para evaluar el rendimiento de una instancia de Kafka que se ejecuta en NFS. La diferencia entre los controladores es la propiedad de

vaciado del registro.

Para un par de Cloud Volumes ONTAP HA:

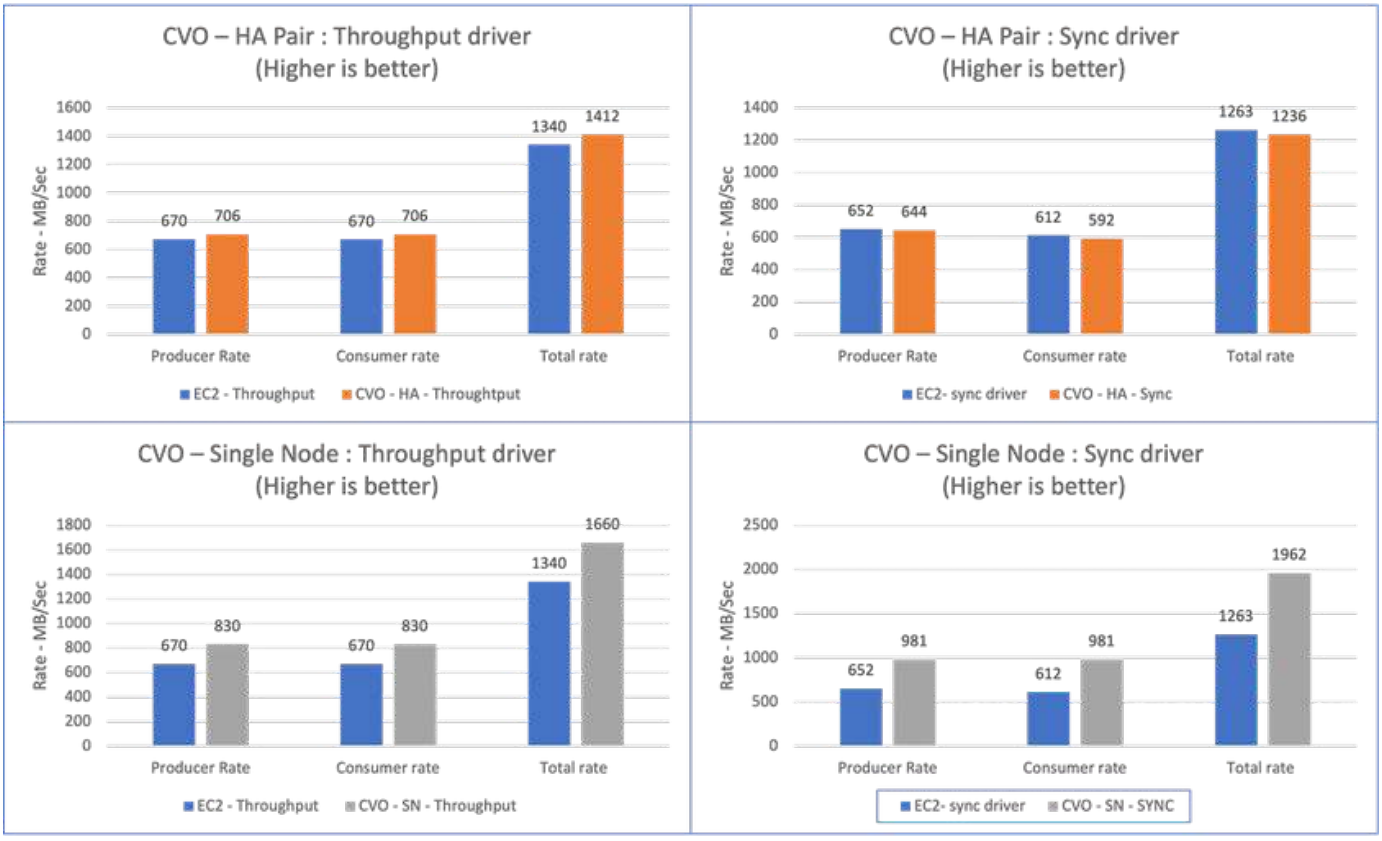
- Rendimiento total generado consistentemente por el controlador de sincronización: ~1236 MBps.
- Rendimiento total generado para el controlador de rendimiento: pico ~1412 MBps.

Para un solo nodo de Cloud Volumes ONTAP :

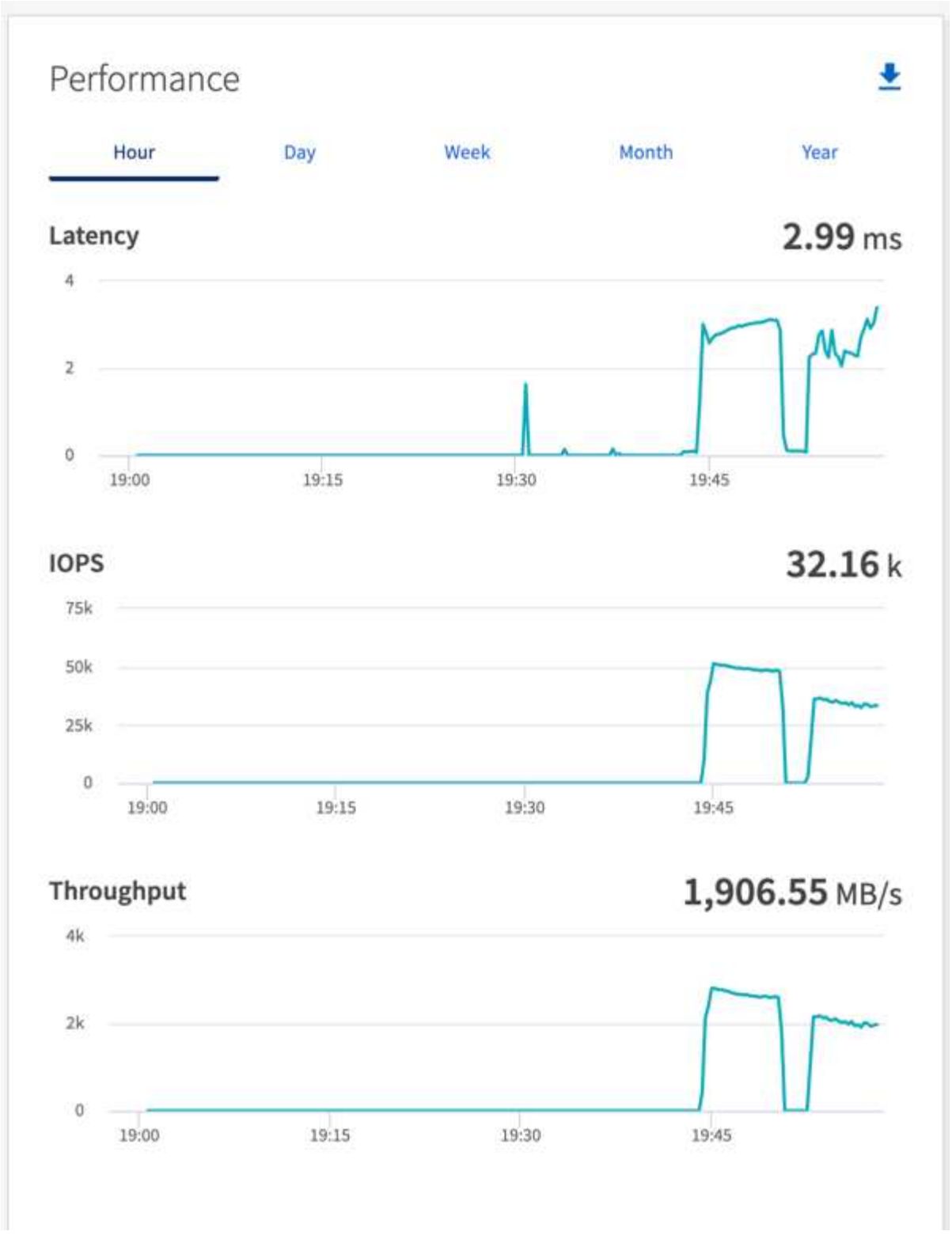
- Rendimiento total generado consistentemente por el controlador de sincronización: ~ 1962 MBps.
- Rendimiento total generado por el controlador de rendimiento: pico ~1660 MBps

El controlador de sincronización puede generar un rendimiento constante a medida que los registros se vacían en el disco instantáneamente, mientras que el controlador de rendimiento genera ráfagas de rendimiento a medida que los registros se envían al disco en forma masiva.

Estos números de rendimiento se generan para la configuración de AWS dada. Para requisitos de mayor rendimiento, los tipos de instancias se pueden escalar y ajustar aún más para obtener mejores números de rendimiento. El rendimiento total o tasa total es la combinación de la tasa del productor y del consumidor.



Asegúrese de verificar el rendimiento del almacenamiento al realizar evaluaciones comparativas del controlador de sincronización o del rendimiento.



Descripción general del rendimiento y validación en AWS FSx ONTAP

Se evaluó el rendimiento de un clúster de Kafka con la capa de almacenamiento montada en NetApp NFS en AWS FSx ONTAP. Los ejemplos de evaluación comparativa se describen en las siguientes secciones.

Apache Kafka en AWS FSx ONTAP

El sistema de archivos de red (NFS) es un sistema de archivos de red ampliamente utilizado para almacenar grandes cantidades de datos. En la mayoría de las organizaciones, los datos se generan cada vez más mediante aplicaciones de transmisión como Apache Kafka. Estas cargas de trabajo requieren escalabilidad, baja latencia y una arquitectura de ingesta de datos sólida con capacidades de almacenamiento modernas. Para permitir análisis en tiempo real y proporcionar información útil, se requiere una infraestructura bien diseñada y de alto rendimiento.

Kafka, por diseño, funciona con un sistema de archivos compatible con POSIX y se basa en el sistema de archivos para manejar las operaciones de archivos, pero al almacenar datos en un sistema de archivos NFSv3, el cliente NFS del agente de Kafka puede interpretar las operaciones de archivos de manera diferente a un sistema de archivos local como XFS o Ext4. Un ejemplo común es el cambio de nombre tonto de NFS que provocó que los agentes de Kafka fallaran al expandir clústeres y reasignar particiones. Para enfrentar este desafío, NetApp ha actualizado el cliente NFS de Linux de código abierto con cambios que ahora están generalmente disponibles en RHEL8.7, RHEL9.1 y son compatibles con la versión actual de FSx ONTAP , ONTAP 9.12.1.

Amazon FSx ONTAP proporciona un sistema de archivos NFS totalmente administrado, escalable y de alto rendimiento en la nube. Los datos de Kafka en FSx ONTAP pueden escalar para manejar grandes cantidades de datos y garantizar la tolerancia a fallas. NFS proporciona gestión de almacenamiento centralizada y protección de datos para conjuntos de datos críticos y confidenciales.

Estas mejoras permiten que los clientes de AWS aprovechen FSx ONTAP al ejecutar cargas de trabajo de Kafka en los servicios informáticos de AWS. Estos beneficios son: * Reducir la utilización de la CPU para reducir el tiempo de espera de E/S * Tiempo de recuperación del agente de Kafka más rápido. * Confiabilidad y eficiencia. * Escalabilidad y rendimiento. * Disponibilidad de zonas multidisponibilidad. * Protección de datos.

Descripción general del rendimiento y validación en AWS FSx ONTAP

Se evaluó el rendimiento de un clúster de Kafka con la capa de almacenamiento montada en NetApp NFS en la nube de AWS. Los ejemplos de evaluación comparativa se describen en las siguientes secciones.

Kafka en AWS FSx ONTAP

Se evaluó el rendimiento de un clúster de Kafka con AWS FSx ONTAP en la nube de AWS. Esta evaluación comparativa se describe en las siguientes secciones.

Configuración arquitectónica

La siguiente tabla muestra la configuración ambiental para un clúster de Kafka que utiliza AWS FSx ONTAP.

Componente de plataforma	Configuración del entorno
Kafka 3.2.3	• 3 x cuidadores del zoológico – t2.small • 3 servidores intermediarios – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x productor/consumidor — c5n.2xlarge *
Sistema operativo en todos los nodos	RHEL8.6
AWS FSx ONTAP	Multi-AZ con un rendimiento de 4 GB/seg y 160 000 IOPS

Configuración de NetApp FSx ONTAP

1. Para nuestras pruebas iniciales, hemos creado un sistema de archivos FSx ONTAP con 2 TB de capacidad y 40 000 IOP para un rendimiento de 2 GB/seg.

```
[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"}
```

   En nuestro ejemplo, estamos implementando FSx ONTAP a través de AWS CLI. Necesitará personalizar aún más el comando en su entorno según sea necesario. FSx ONTAP también se puede implementar y administrar a través de la consola de AWS para una experiencia de implementación más sencilla y optimizada con menos entrada de línea de comandos.

   Documentación En FSx ONTAP, las IOPS máximas alcanzables para un sistema de archivos con un rendimiento de 2 GB/seg en nuestra región de prueba (US-East-1) son 80 000 IOPS. El total máximo de iops para un sistema de archivos FSx ONTAP es 160 000 iops, lo que requiere una implementación con un rendimiento de 4 GB/seg para lograrlo, lo cual demostraremos más adelante en este documento.

   Para obtener más información sobre las especificaciones de rendimiento de FSx ONTAP , no dude en visitar la documentación de AWS FSx ONTAP aquí: https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/performance.html .

   La sintaxis detallada de la línea de comandos para "create-file-system" de FSx se puede encontrar aquí: https://docs.aws.amazon.com/cli/latest/reference/fsx/create-file-system.html

   Por ejemplo, puede especificar una clave KMS específica en lugar de la clave maestra de AWS FSx predeterminada que se utiliza cuando no se especifica ninguna clave KMS.

2. Al crear el sistema de archivos FSx ONTAP , espere hasta que el estado de "Ciclo de vida" cambie a "DISPONIBLE" en su devolución JSON después de describir su sistema de archivos de la siguiente manera:

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

3. Valide las credenciales iniciando sesión en FSx ONTAP SSH con el usuario fsxadmin: Fsxadmin es la cuenta de administrador predeterminada para los sistemas de archivos FSx ONTAP en el momento de la creación. La contraseña para fsxadmin es la contraseña que se configuró cuando se creó por primera vez el sistema de archivos, ya sea en la consola de AWS o con la CLI de AWS, como completamos en el Paso 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 vez validadas sus credenciales, cree la máquina virtual de almacenamiento en el sistema de archivos FSx 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 máquina virtual de almacenamiento (SVM) es un servidor de archivos aislado con sus propias credenciales administrativas y puntos finales para administrar y acceder a datos en volúmenes FSx ONTAP y proporciona multitenencia FSx ONTAP .

5. Una vez que haya configurado su máquina virtual de almacenamiento principal, acceda por SSH al sistema de archivos FSx ONTAP recién creado y cree volúmenes en la máquina virtual de almacenamiento utilizando el siguiente comando de muestra y, de manera similar, creamos 6 volúmenes para esta validación. Según nuestra validación, mantenga el constituyente predeterminado (8) o menos constituyentes, lo que proporcionará un mejor rendimiento 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. Necesitaremos capacidad adicional en nuestros volúmenes para nuestras pruebas. Amplíe el tamaño del volumen a 2 TB y móntelo en la ruta de unión.

```
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
```

En FSx ONTAP, los volúmenes pueden aprovisionarse de manera fina. En nuestro ejemplo, la capacidad total del volumen extendido excede la capacidad total del sistema de archivos, por lo que necesitaremos ampliar la capacidad total del sistema de archivos para desbloquear capacidad de volumen aprovisionada adicional, lo que demostraremos en el próximo paso.

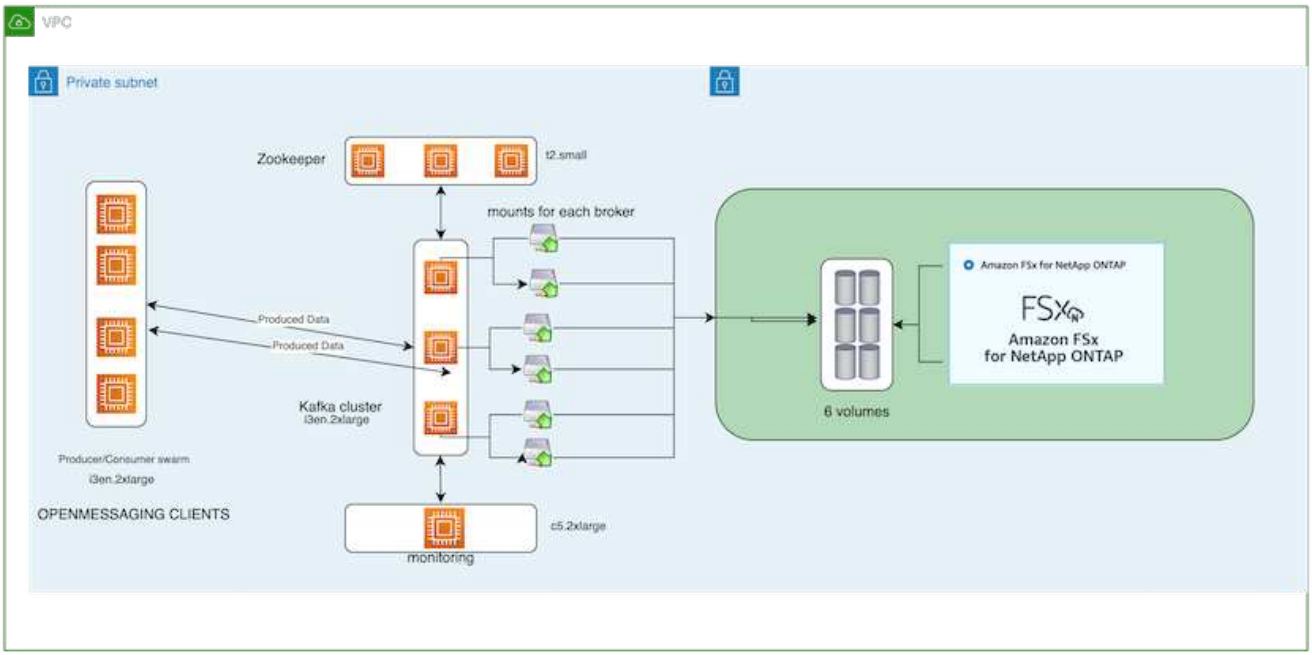
7. A continuación, para obtener mayor rendimiento y capacidad, ampliamos la capacidad de rendimiento de FSx ONTAP de 2 GB/seg a 4 GB/seg y las IOPS a 160 000, y la capacidad a 5 TB.

```
[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 sintaxis detallada de la línea de comandos para "update-file-system" de FSx se puede encontrar aquí:https://docs.aws.amazon.com/cli/latest/reference/fsx/update-file-system.html[]

8. Los volúmenes de FSx ONTAP se montan con nconnect y opciones predeterminadas en los brokers de Kafka

La siguiente imagen muestra nuestra arquitectura final de nuestro clúster Kafka basado en FSx ONTAP :



- Calcular. Utilizamos un clúster Kafka de tres nodos con un conjunto Zookeeper de tres nodos ejecutándose en servidores dedicados. Cada agente tenía seis puntos de montaje NFS en seis volúmenes en la instancia FSx ONTAP .
- Escucha. Utilizamos dos nodos para una combinación Prometheus-Grafana. Para generar cargas de trabajo, utilizamos un clúster separado de tres nodos que podía producir y consumir en este clúster de Kafka.
- Almacenamiento. Utilizamos un FSx ONTAP con seis volúmenes de 2 TB montados. Luego, el volumen se exportó al agente de Kafka con un montaje NFS. Los volúmenes de FSx ONTAP se montan con 16 sesiones nconnect y opciones predeterminadas en los agentes de Kafka.

Configuraciones de evaluación comparativa de OpenMessage.

Usamos la misma configuración utilizada para los volúmenes NetApp Cloud ONTAP y sus detalles están aquí: enlace:kafka-nfs-performance-overview-and-validation-in-aws.html#architectural-setup

Metodología de pruebas

1. Se aprovisionó un clúster de Kafka según la especificación descrita anteriormente utilizando Terraform y Ansible. Terraform se utiliza para construir la infraestructura utilizando instancias de AWS para el clúster de Kafka y Ansible construye el clúster de Kafka en ellas.
2. Se activó una carga de trabajo OMB con la configuración de carga de trabajo descrita anteriormente y el controlador de sincronización.

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

3. Se activó otra carga de trabajo con el controlador de rendimiento con la misma configuración de carga de trabajo.

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

Observación

Se utilizaron dos tipos diferentes de controladores para generar cargas de trabajo para evaluar el rendimiento de una instancia de Kafka que se ejecuta en NFS. La diferencia entre los controladores es la propiedad de vaciado del registro.

Para un factor de replicación de Kafka 1 y FSx ONTAP:

- Rendimiento total generado consistentemente por el controlador de sincronización: ~ 3218 MBps y rendimiento máximo en ~ 3652 MBps.
- Rendimiento total generado consistentemente por el controlador de rendimiento: ~ 3679 MBps y rendimiento máximo en ~ 3908 MBps.

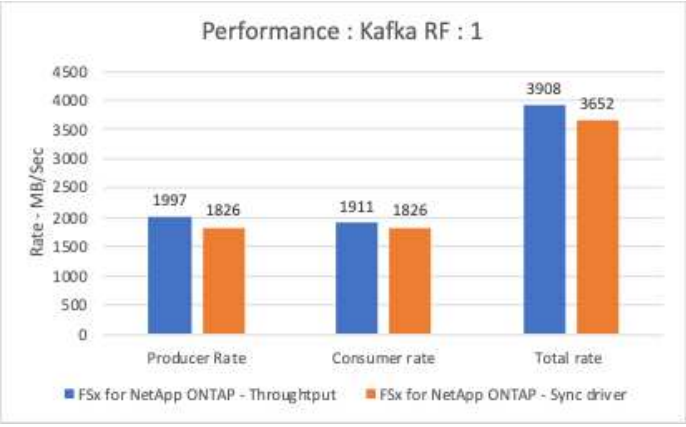
Para Kafka con factor de replicación 3 y FSx ONTAP :

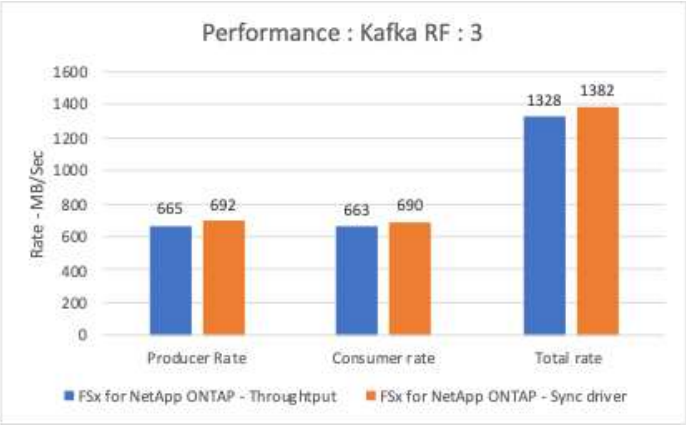
- Rendimiento total generado consistentemente por el controlador de sincronización: ~ 1252 MBps y rendimiento máximo en ~ 1382 MBps.
- Rendimiento total generado consistentemente por el controlador de rendimiento: ~ 1218 MBps y rendimiento máximo en ~ 1328 MBps.

En el factor de replicación 3 de Kafka, la operación de lectura y escritura ocurrió tres veces en FSx ONTAP. En el factor de replicación 1 de Kafka, la operación de lectura y escritura ocurrió una vez en FSx ONTAP, por lo que en ambas validaciones pudimos alcanzar el rendimiento máximo de 4 GB/seg.

El controlador de sincronización puede generar un rendimiento constante a medida que los registros se vacían en el disco instantáneamente, mientras que el controlador de rendimiento genera ráfagas de rendimiento a medida que los registros se envían al disco en forma masiva.

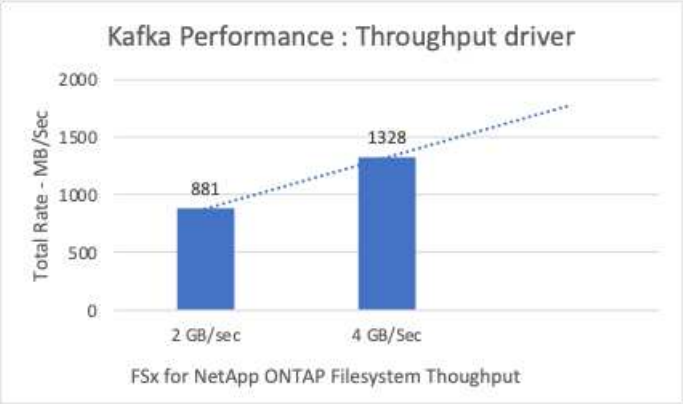
Estos números de rendimiento se generan para la configuración de AWS dada. Para requisitos de mayor rendimiento, los tipos de instancias se pueden escalar y ajustar aún más para obtener mejores números de rendimiento. El rendimiento total o tasa total es la combinación de la tasa del productor y del consumidor.

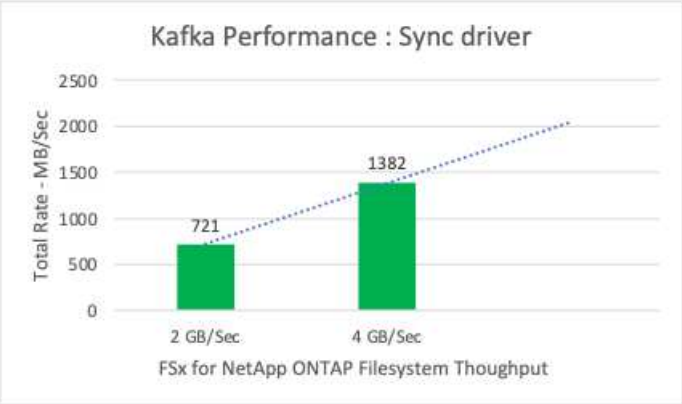




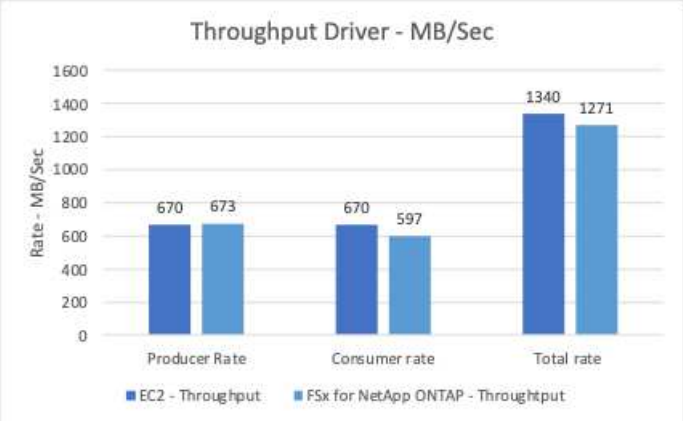
La siguiente gráfica muestra el rendimiento de FSx ONTAP a 2 GB/seg y de 4 GB/seg para el factor de replicación 3 de Kafka. El factor de replicación 3 realiza la operación de lectura y escritura tres veces en el almacenamiento FSx ONTAP . La tasa total del controlador de rendimiento es de 881 MB/seg, que realiza

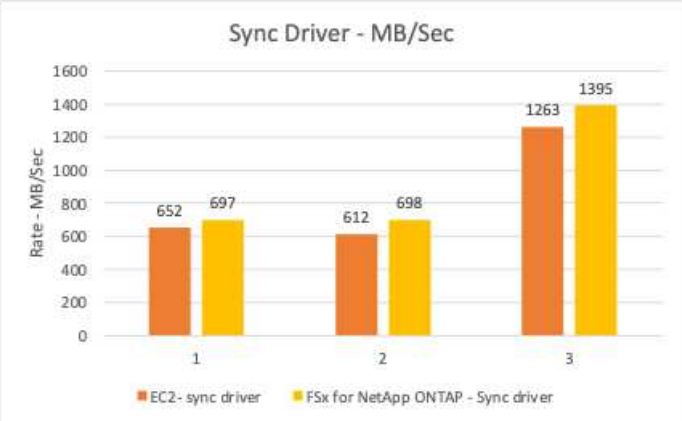
operaciones de lectura y escritura de Kafka a aproximadamente 2,64 GB/seg en el sistema de archivos FSx ONTAP de 2 GB/seg, y la tasa total del controlador de rendimiento es de 1328 MB/seg, que realiza operaciones de lectura y escritura de Kafka a aproximadamente 3,98 GB/seg. El rendimiento de Kafka es lineal y escalable según el rendimiento de FSx ONTAP .





El siguiente gráfico muestra el rendimiento entre la instancia EC2 y FSx ONTAP (factor de replicación de Kafka: 3)

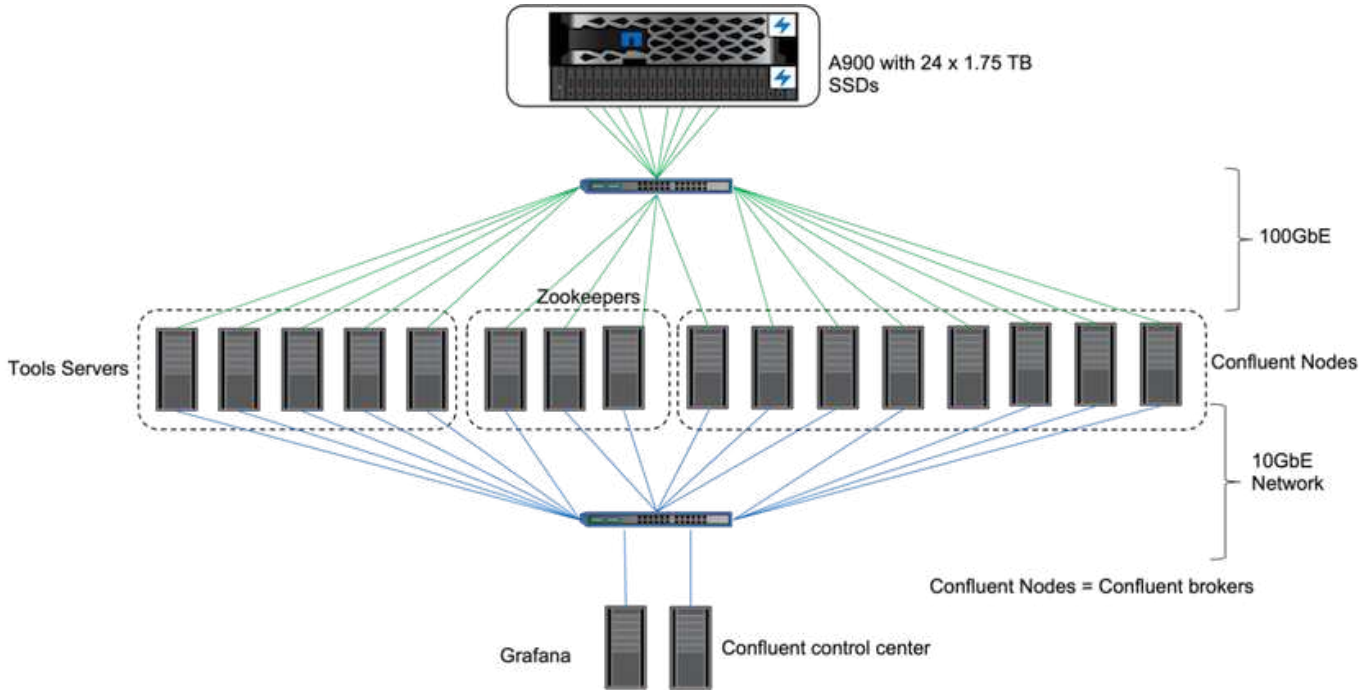




Descripción general del rendimiento y validación con AFF A900 en las instalaciones

En las instalaciones, utilizamos el controlador de almacenamiento NetApp AFF A900 con ONTAP 9.12.1RC1 para validar el rendimiento y la escalabilidad de un clúster de Kafka. Utilizamos el mismo banco de pruebas que en nuestras prácticas recomendadas de almacenamiento en niveles anteriores con ONTAP y AFF.

Utilizamos Confluent Kafka 6.2.0 para evaluar el AFF A900. El clúster cuenta con ocho nodos intermediarios y tres nodos guardianes del zoológico. Para probar el rendimiento, utilizamos cinco nodos de trabajo OMB.



Configuración de almacenamiento

Utilizamos instancias de NetApp FlexGroups para proporcionar un único espacio de nombres para los directorios de registro, lo que simplifica la recuperación y la configuración. Utilizamos NFSv4.1 y pNFS para proporcionar acceso directo a los datos del segmento de registro.

Ajuste del cliente

Cada cliente montó la instancia de FlexGroup con el siguiente comando.

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

Además, aumentamos la `max_session_slots` del valor predeterminado `64` a `180` . Esto coincide con el límite de espacio de sesión predeterminado en ONTAP.

Ajuste del bróker de Kafka

Para maximizar el rendimiento en el sistema bajo prueba, aumentamos significativamente los parámetros predeterminados para ciertos grupos de subprocesos clave. Recomendamos seguir las mejores prácticas de Confluent Kafka para la mayoría de las configuraciones. Este ajuste se utilizó para maximizar la concurrencia de E/S pendientes al almacenamiento. Estos parámetros se pueden ajustar para que coincidan con los recursos computacionales y los atributos de almacenamiento de su corredor.

```
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
```

Metodología de prueba del generador de carga de trabajo

Utilizamos las mismas configuraciones OMB que para las pruebas en la nube para el controlador de rendimiento y la configuración del tema.

1. Se aprovisionó una instancia de FlexGroup mediante Ansible en un clúster 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. Se habilitó pNFS en ONTAP SVM.

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

3. La carga de trabajo se activó con el controlador de rendimiento utilizando la misma configuración de carga de trabajo que para Cloud Volumes ONTAP. Ver la sección "Rendimiento en estado estacionario " abajo. La carga de trabajo utilizó un factor de replicación de 3, lo que significa que se mantuvieron tres copias de segmentos de registro en NFS.

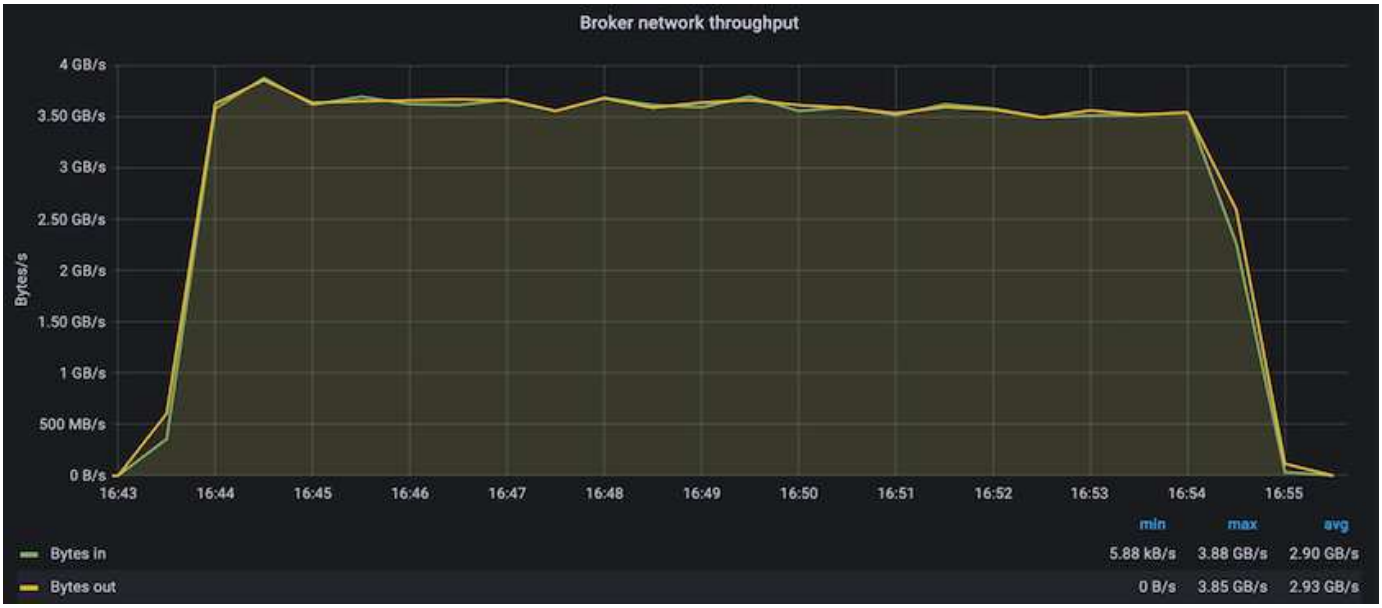
```
sudo bin/benchmark --drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

4. Por último, completamos mediciones utilizando un backlog para medir la capacidad de los consumidores para ponerse al día con los últimos mensajes. La OMB crea un backlog pausando a los consumidores durante el comienzo de una medición. Esto produce tres fases distintas: creación de cartera de pedidos (tráfico exclusivo del productor), vaciado de cartera de pedidos (una fase de gran actividad del consumidor en la que los consumidores se ponen al día con los eventos perdidos en un tema) y el estado estable. Ver la sección "Rendimiento extremo y exploración de los límites del almacenamiento " para obtener más información.

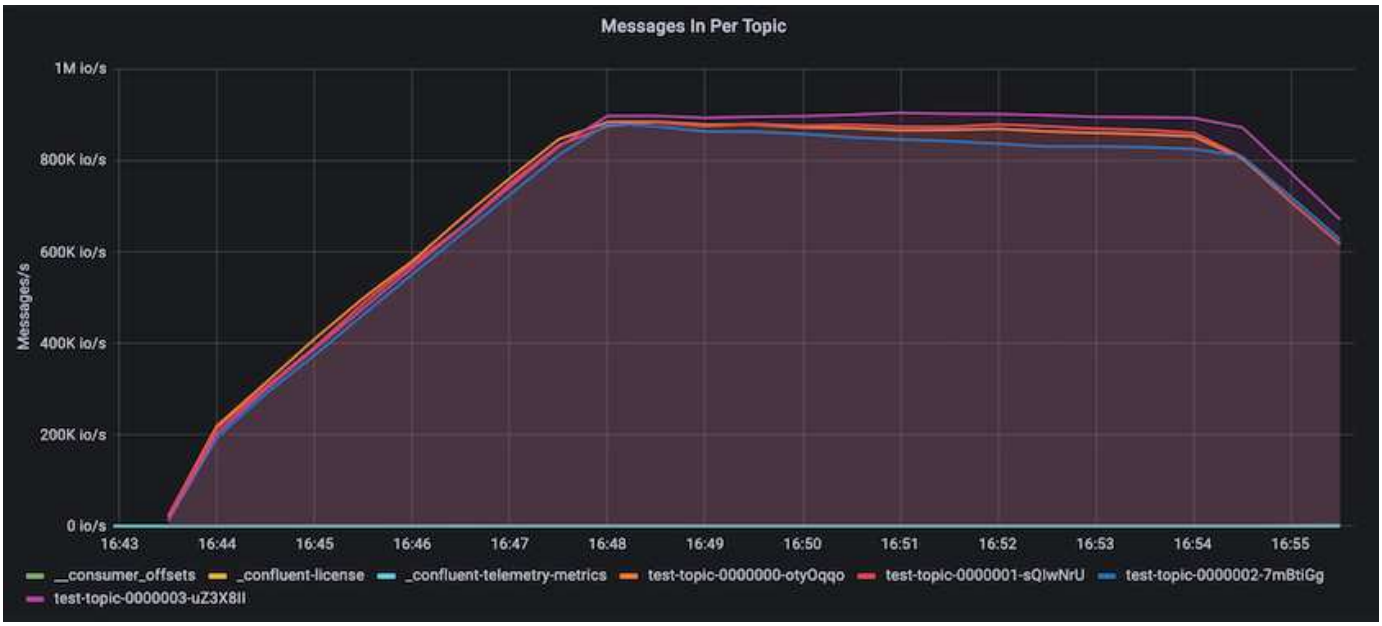
Rendimiento en estado estacionario

Evaluamos el AFF A900 utilizando OpenMessaging Benchmark para proporcionar una comparación similar a la de Cloud Volumes ONTAP en AWS y DAS en AWS. Todos los valores de rendimiento representan el rendimiento del clúster de Kafka a nivel de productor y consumidor.

El rendimiento en estado estable con Confluent Kafka y AFF A900 logró un rendimiento promedio de más de 3,4 GBps tanto para el productor como para los consumidores. Esto supone más de 3,4 millones de mensajes en todo el clúster de Kafka. Al visualizar el rendimiento sostenido en bytes por segundo para BrokerTopicMetrics, vemos el excelente rendimiento en estado estable y el tráfico admitido por el AFF A900.



Esto se alinea bien con la vista de los mensajes entregados por tema. El siguiente gráfico proporciona un desglose por tema. En la configuración probada, vimos casi 900 000 mensajes por tema en cuatro temas.

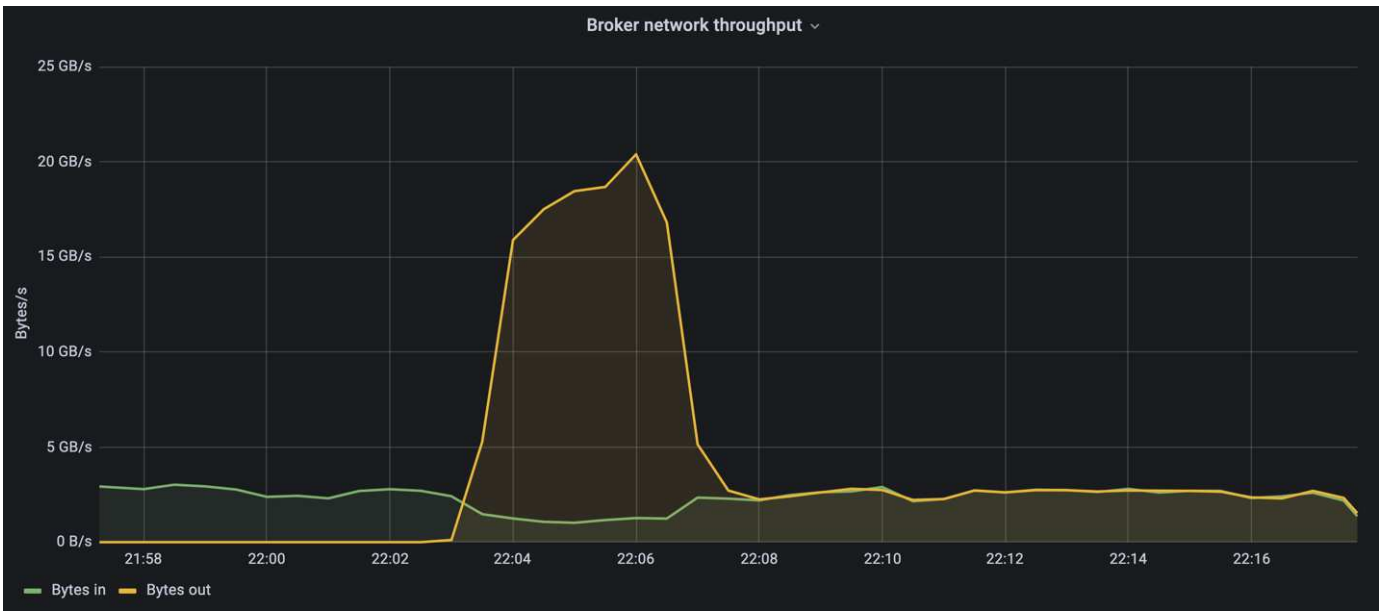


Rendimiento extremo y exploración de los límites del almacenamiento

Para AFF, también realizamos pruebas con OMB utilizando la función de backlog. La función de acumulación de eventos pausa las suscripciones de los consumidores mientras se crea una acumulación de eventos en el clúster de Kafka. Durante esta fase, solo se produce tráfico de productor, lo que genera eventos que se confirman en los registros. Esto emula de manera más cercana el procesamiento por lotes o los flujos de trabajo de análisis fuera de línea; en estos flujos de trabajo, se inician las suscripciones de los consumidores y deben leer datos históricos que ya se han eliminado de la memoria caché del agente.

Para comprender las limitaciones de almacenamiento en el rendimiento del consumidor en esta configuración, medimos solo la fase de productor para comprender cuánto tráfico de escritura podía absorber el A900. Vea la siguiente sección "Guía de tallas "para entender cómo aprovechar estos datos.

Durante la parte de esta medición solo para productores, vimos un alto pico de rendimiento que empujó los límites del rendimiento del A900 (cuando otros recursos del broker no estaban saturados y atendían el tráfico de productores y consumidores).



Aumentamos el tamaño del mensaje a 16k para esta medición para limitar los costos generales por mensaje y maximizar el rendimiento del almacenamiento en los puntos de montaje NFS.

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

El clúster Confluent Kafka alcanzó un rendimiento máximo de producción de 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 …
```

Después de que OMB terminó de completar el registro de eventos, se reinició el tráfico del consumidor. Durante las mediciones con drenaje de cartera, observamos un rendimiento máximo del consumidor de más de 20 GBps en todos los temas. El rendimiento combinado del volumen NFS que almacena los datos del registro OMB se acercó a ~30 GBps.

Guía de tallas

Amazon Web Services ofrece una "guía de tallas" para dimensionar y escalar clústeres de Kafka.

Este dimensionamiento proporciona una fórmula útil para determinar los requisitos de rendimiento de almacenamiento para su clúster de Kafka:

Para un rendimiento agregado producido en el clúster tcluster con un factor de replicación de r, el rendimiento recibido por el almacenamiento del intermediario es el siguiente:

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

Esto se puede simplificar aún más:

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

El uso de esta fórmula le permite seleccionar la plataforma ONTAP adecuada para sus necesidades de nivel activo de Kafka.

La siguiente tabla explica el rendimiento del productor previsto para el A900 con diferentes factores de replicación:

Factor de replicación	Rendimiento del productor (GPps)
3 (medido)	3,4
2	5,1
1	10,2

Conclusión

La solución de NetApp para el tonto problema del cambio de nombre proporciona una forma de almacenamiento simple, económica y administrada de manera centralizada para cargas de trabajo que antes eran incompatibles con NFS.

Este nuevo paradigma permite a los clientes crear clústeres de Kafka más manejables que son más fáciles de migrar y reflejar para fines de recuperación ante desastres y protección de datos. También hemos visto que NFS proporciona beneficios adicionales, como una menor utilización de la CPU y un tiempo de recuperación más rápido, una eficiencia de almacenamiento drásticamente mejorada y un mejor rendimiento a través de NetApp ONTAP.

Dónde encontrar información adicional

Para obtener más información sobre la información que se describe en este documento, revise los siguientes documentos y/o sitios web:

- ¿Qué es Apache Kafka?

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

- ¿Qué es un cambio de nombre tonto?

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

- ONATP se lee para aplicaciones de streaming.

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

- Documentación de productos de NetApp

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

- ¿Qué es NFS?

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

- ¿Qué es la reasignación de particiones de Kafka?

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

- ¿Qué es el OpenMessaging Benchmark?

  "https://openmessaging.cloud/"

- ¿Cómo migrar un broker de Kafka?

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

- ¿Cómo monitorear el broker Kafka con Prometheus?

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

- Plataforma administrada para Apache Kafka

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

- Compatibilidad con Apache Kafka

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

- Servicios de consultoría para Apache Kafka

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