Cargas de trabalho do Apache Kafka com armazenamento NetApp NFS

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Índice

Cargas de trabalho do Apache Kafka com armazenamento NetApp NFS

TR-4947: Carga de trabalho do Apache Kafka com armazenamento NetApp NFS - Validação funcional e desempenho

Shantanu Chakole, Karthikeyan Nagalingam e Joe Scott, NetApp

Kafka é um sistema de mensagens de publicação e assinatura distribuído com uma fila robusta que pode aceitar grandes quantidades de dados de mensagens. Com o Kafka, os aplicativos podem gravar e ler dados em tópicos de maneira muito rápida. Devido à sua tolerância a falhas e escalabilidade, o Kafka é frequentemente usado no espaço de big data como uma maneira confiável de ingerir e mover muitos fluxos de dados muito rapidamente. Os casos de uso incluem processamento de fluxo, rastreamento de atividade do site, coleta e monitoramento de métricas, agregação de logs, análises em tempo real e assim por diante.

Embora as operações normais do Kafka no NFS funcionem bem, o problema bobo de renomeação trava o aplicativo durante o redimensionamento ou reparticionamento de um cluster do Kafka em execução no NFS. Este é um problema significativo porque um cluster Kafka precisa ser redimensionado ou reparticionado para fins de balanceamento de carga ou manutenção. Você pode encontrar detalhes adicionais "aqui" .

Este documento descreve os seguintes assuntos:

- O problema da renomeação boba e a validação da solução
- Reduzir a utilização da CPU para reduzir o tempo de espera de E/S
- Tempo de recuperação mais rápido do corretor Kafka
- Desempenho na nuvem e no local

Por que usar armazenamento NFS para cargas de trabalho do Kafka?

As cargas de trabalho do Kafka em aplicativos de produção podem transmitir grandes quantidades de dados entre aplicativos. Esses dados são mantidos e armazenados nos nós do broker Kafka no cluster Kafka. O Kafka também é conhecido por disponibilidade e paralelismo, que ele consegue dividindo tópicos em partições e depois replicando essas partições por todo o cluster. Isso significa que a enorme quantidade de dados que flui por um cluster Kafka geralmente é multiplicada em tamanho. O NFS torna o rebalanceamento de dados muito rápido e fácil, à medida que o número de corretores muda. Para ambientes grandes, o rebalanceamento de dados no DAS quando o número de corretores muda consome muito tempo e, na maioria dos ambientes Kafka, o número de corretores muda com frequência.

Outros benefícios incluem o seguinte:

- **Maturidade.** O NFS é um protocolo maduro, o que significa que a maioria dos aspectos de implementação, proteção e uso são bem compreendidos.
- **Abrir.** O NFS é um protocolo aberto e seu desenvolvimento contínuo está documentado nas especificações da Internet como um protocolo de rede livre e aberto.

- **Custo-benefício.** O NFS é uma solução de baixo custo para compartilhamento de arquivos em rede que é fácil de configurar porque usa a infraestrutura de rede existente.
- **Gerenciado centralmente.** O gerenciamento centralizado do NFS diminui a necessidade de software adicional e espaço em disco em sistemas de usuários individuais.
- **Distribuído.** O NFS pode ser usado como um sistema de arquivos distribuído, reduzindo a necessidade de dispositivos de armazenamento de mídia removíveis.

Por que usar o NetApp para cargas de trabalho do Kafka?

A implementação do NetApp NFS é considerada um padrão ouro para o protocolo e é usada em inúmeros ambientes NAS empresariais. Além da credibilidade da NetApp, ela também oferece os seguintes benefícios:

- Confiabilidade e eficiência
- Escalabilidade e desempenho
- Alta disponibilidade (parceiro de HA em um cluster NetApp ONTAP)
- Proteção de dados
  - **Recuperação de desastres (NetApp SnapMirror).** Seu site sai do ar ou você quer começar em um site diferente e continuar de onde parou.
  - Capacidade de gerenciamento do seu sistema de armazenamento (administração e gerenciamento usando NetApp OnCommand).
  - **Balanceamento de carga.** O cluster permite que você acesse diferentes volumes de LIFs de dados hospedados em diferentes nós.
  - **Operações não disruptivas.** LIFs ou movimentações de volume são transparentes para os clientes NFS.

Solução da NetApp para problema de renomeação de cargas de trabalho NFS para Kafka

O Kafka é criado com a suposição de que o sistema de arquivos subjacente seja compatível com POSIX: por exemplo, XFS ou Ext4. O rebalanceamento de recursos do Kafka remove arquivos enquanto o aplicativo ainda os está utilizando. Um sistema de arquivos compatível com POSIX permite que o unlink prossiga. No entanto, ele só remove o arquivo depois que todas as referências a ele desaparecem. Se o sistema de arquivos subjacente estiver conectado à rede, o cliente NFS interceptará as chamadas de desvinculação e gerenciará o fluxo de trabalho. Como há aberturas pendentes no arquivo que está sendo desvinculado, o cliente NFS envia uma solicitação de renomeação ao servidor NFS e, no último fechamento do arquivo desvinculado, emite uma operação de remoção no arquivo renomeado. Esse comportamento é comumente chamado de renomeação boba do NFS e é orquestrado pelo cliente NFS.

Qualquer corretor Kafka que use armazenamento de um servidor NFSv3 terá problemas por causa desse comportamento. No entanto, o protocolo NFSv4.x tem recursos para resolver esse problema, permitindo que o servidor assuma a responsabilidade pelos arquivos abertos e desvinculados. Os servidores NFS que oferecem suporte a esse recurso opcional comunicam a capacidade de propriedade ao cliente NFS no momento da abertura do arquivo. O cliente NFS então cessa o gerenciamento de desvinculação quando há aberturas pendentes e permite que o servidor gerencie o fluxo. Embora a especificação NFSv4 forneça

diretrizes para implementação, até agora não havia nenhuma implementação de servidor NFS conhecida que suportasse esse recurso opcional.

As seguintes alterações são necessárias para que o servidor NFS e o cliente NFS resolvam o problema bobo de renomeação:

- **Alterações no cliente NFS (Linux).** No momento da abertura do arquivo, o servidor NFS responde com um sinalizador, indicando a capacidade de lidar com a desvinculação de arquivos abertos. As alterações do lado do cliente NFS permitem que o servidor NFS manipule a desvinculação na presença do sinalizador. A NetApp atualizou o cliente Linux NFS de código aberto com essas alterações. O cliente NFS atualizado agora está disponível para o RHEL 8.7 e RHEL 9.1.
- **Alterações no servidor NFS.** O servidor NFS monitora as aberturas. A desvinculação de um arquivo aberto existente agora é gerenciada pelo servidor para corresponder à semântica POSIX. Quando o último arquivo aberto é fechado, o servidor NFS inicia a remoção real do arquivo, evitando assim o processo bobo de renomeação. O servidor ONTAP NFS implementou esse recurso em sua versão mais recente, o ONTAP 9.12.1.

Com as alterações acima no cliente e servidor NFS, o Kafka pode colher com segurança todos os benefícios do armazenamento NFS conectado à rede.

Validação funcional - Correção de renomeação boba

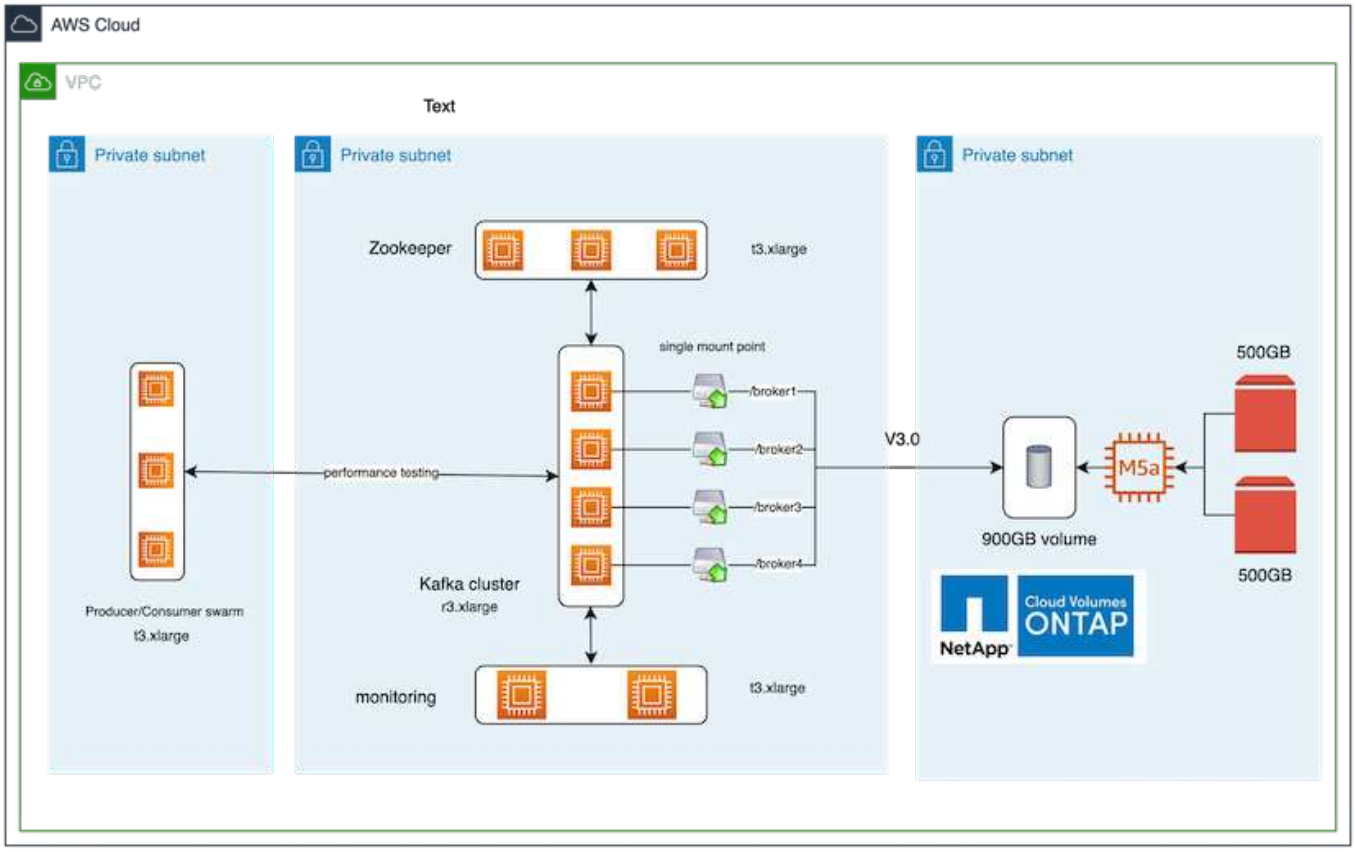
Para a validação funcional, mostramos que um cluster Kafka com uma montagem NFSv3 para armazenamento falha ao executar operações Kafka, como redistribuição de partições, enquanto outro cluster montado em NFSv4 com a correção pode executar as mesmas operações sem interrupções.

Configuração de validação

A configuração é executada na AWS. A tabela a seguir mostra os diferentes componentes da plataforma e a configuração ambiental usados para a validação.

Componente de plataforma	Configuração do ambiente
Plataforma Confluent versão 7.2.1	• 3 x tratadores de zoológico – t3.xlarge • 4 x servidores de corretores – r3.xlarge • 1 x Grafana – t3.xlarge • 1 x centro de controle – t3.xlarge • 3 x Produtor/consumidor
Sistema operacional em todos os nós	RHEL8.7 ou posterior
Instância NetApp Cloud Volumes ONTAP	Instância de nó único – M5.2xLarge

A figura a seguir mostra a configuração arquitetônica desta solução.



Fluxo arquitetônico

- **Calcular.** Usamos um cluster Kafka de quatro nós com um conjunto de zookeepers de três nós em execução em servidores dedicados.
- **Monitoramento.** Usamos dois nós para uma combinação Prometheus-Grafana.
- **Carga de trabalho.** Para gerar cargas de trabalho, usamos um cluster separado de três nós que pode produzir e consumir deste cluster Kafka.
- **Armazenar.** Usamos uma instância ONTAP de volumes NetApp Cloud de nó único com dois volumes GP2 AWS-EBS de 500 GB anexados à instância. Esses volumes foram então expostos ao cluster Kafka como um único volume NFSv4.1 por meio de um LIF.

As propriedades padrões do Kafka foram escolhidas para todos os servidores. O mesmo foi feito para o enxame de tratadores do zoológico.

Metodologia de testes

1. Atualizar `-is-preserve-unlink-enabled true` ao volume de Kafka, como 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. Dois clusters Kafka semelhantes foram criados com a seguinte diferença:
   - **Grupo 1.** O servidor NFS v4.1 de backend executando o ONTAP versão 9.12.1 pronto para produção foi hospedado por uma instância NetApp CVO. RHEL 8.7/RHEL 9.1 foram instalados nos corretores.
   - **Grupo 2.** O servidor NFS de backend era um servidor Linux NFSv3 genérico criado manualmente.
3. Um tópico de demonstração foi criado em ambos os clusters do 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:

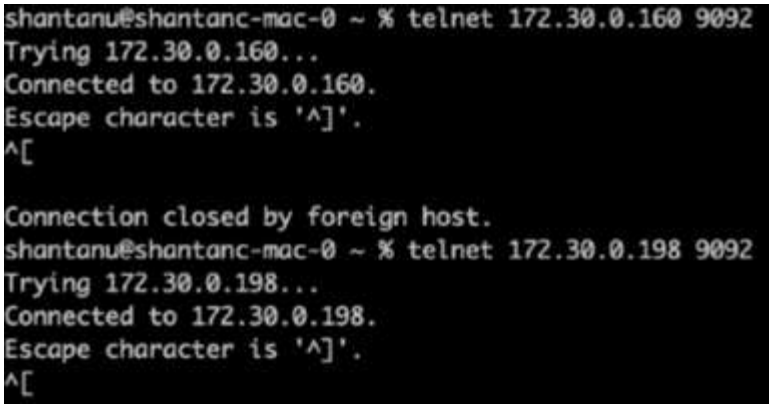
```
[root@ip-172-30-0-198 demo]# kafka-topics --bootstrap-server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:90
92,172.30.0.204:9092 --describe --topic __a_demo_topic
Topic: __a_demo_topic   TopicId: AwQpsZTQShyeMIhaquCG3Q PartitionCount: 4       ReplicationFactor: 2    Configs:
 min.insync.replicas=1,segment.bytes=1073741824
        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. Os dados foram carregados nesses tópicos recém-criados para ambos os clusters. Isso foi feito usando o kit de ferramentas producer-perf-test que vem no pacote padrão do 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. Uma verificação de integridade foi realizada para o broker-1 para cada um dos clusters usando telnet:
   - telnet `172.30.0.160 9092`
   - telnet `172.30.0.198 9092`

     Uma verificação de integridade bem-sucedida para corretores em ambos os clusters é mostrada na próxima captura de tela:



6. Para acionar a condição de falha que faz com que os clusters do Kafka que usam volumes de armazenamento NFSv3 travem, iniciamos o processo de reatribuição de partições em ambos os clusters. A reatribuição da partição foi realizada usando `kafka-reassign-partitions.sh` . O processo detalhado é o seguinte:
   a. Para reatribuir as partições de um tópico em um cluster Kafka, geramos a configuração de reatribuição proposta JSON (isso foi executado para ambos os clusters).

```
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. O JSON de reatribuição gerado foi então salvo em `/tmp/reassignment- file.json` .
   c. O processo real de reatribuição de partição foi acionado pelo seguinte 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. Após alguns minutos, quando a reatribuição foi concluída, outra verificação de integridade nos corretores mostrou que o cluster usando volumes de armazenamento NFSv3 teve um problema de renomeação e travou, enquanto o Cluster 1 usando volumes de armazenamento NetApp ONTAP NFSv4.1 com a correção continuou as operações sem nenhuma interrupção.

```
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
```

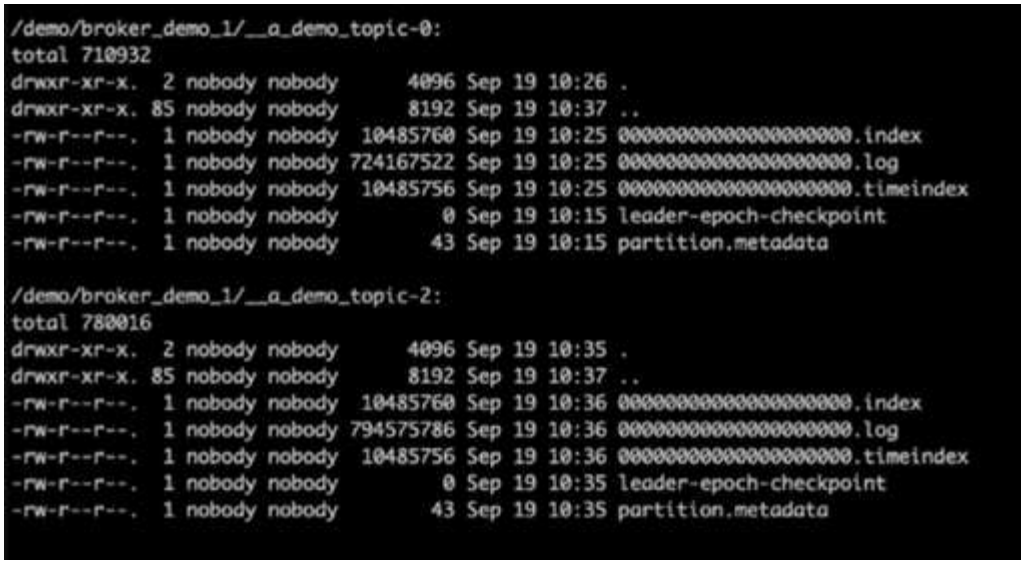
- Cluster1-Broker-1 está ativo.
- Cluster2-broker-1 está morto.

8. Ao verificar os diretórios de log do Kafka, ficou claro que o Cluster 1 usando volumes de armazenamento NetApp ONTAP NFSv4.1 com a correção tinha atribuição de partição limpa, enquanto o Cluster 2 usando armazenamento NFSv3 genérico não tinha devido a problemas de renomeação, o que levou à falha. A imagem a seguir mostra o rebalanceamento de partições do Cluster 2, o que resultou em um problema de renomeação no armazenamento 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
```

A imagem a seguir mostra um rebalanceamento limpo da partição do Cluster 1 usando o armazenamento NetApp NFSv4.1.



Por que usar o NetApp NFS para cargas de trabalho do Kafka?

Agora que há uma solução para o problema bobo de renomeação no armazenamento NFS com o Kafka, você pode criar implantações robustas que aproveitam o armazenamento NetApp ONTAP para sua carga de trabalho do Kafka. Isso não apenas reduz significativamente a sobrecarga operacional, mas também traz os seguintes benefícios aos seus clusters Kafka:

- **Utilização reduzida da CPU em corretores Kafka.** O uso do armazenamento NetApp ONTAP desagregado separa as operações de E/S de disco do broker e, portanto, reduz sua pegada de CPU.
- **Tempo de recuperação mais rápido do corretor.** Como o armazenamento desagregado do NetApp ONTAP é compartilhado entre os nós do broker do Kafka, uma nova instância de computação pode substituir um broker defeituoso a qualquer momento em uma fração do tempo em comparação às implantações convencionais do Kafka, sem reconstruir os dados.
- **Eficiência de armazenamento.** Como a camada de armazenamento do aplicativo agora é provisionada pelo NetApp ONTAP, os clientes podem aproveitar todos os benefícios da eficiência de armazenamento que vem com o ONTAP, como compactação, desduplicação e compactação de dados em linha.

Esses benefícios foram testados e validados em casos de teste que discutimos em detalhes nesta seção.

Utilização reduzida da CPU no broker Kafka

Descobrimos que a utilização geral da CPU é menor do que a do DAS quando executamos cargas de trabalho semelhantes em dois clusters Kafka separados que eram idênticos em suas especificações técnicas, mas diferiam em suas tecnologias de armazenamento. Não apenas a utilização geral da CPU é menor quando o cluster Kafka usa armazenamento ONTAP , mas o aumento na utilização da CPU demonstrou um gradiente mais suave do que em um cluster Kafka baseado em DAS.

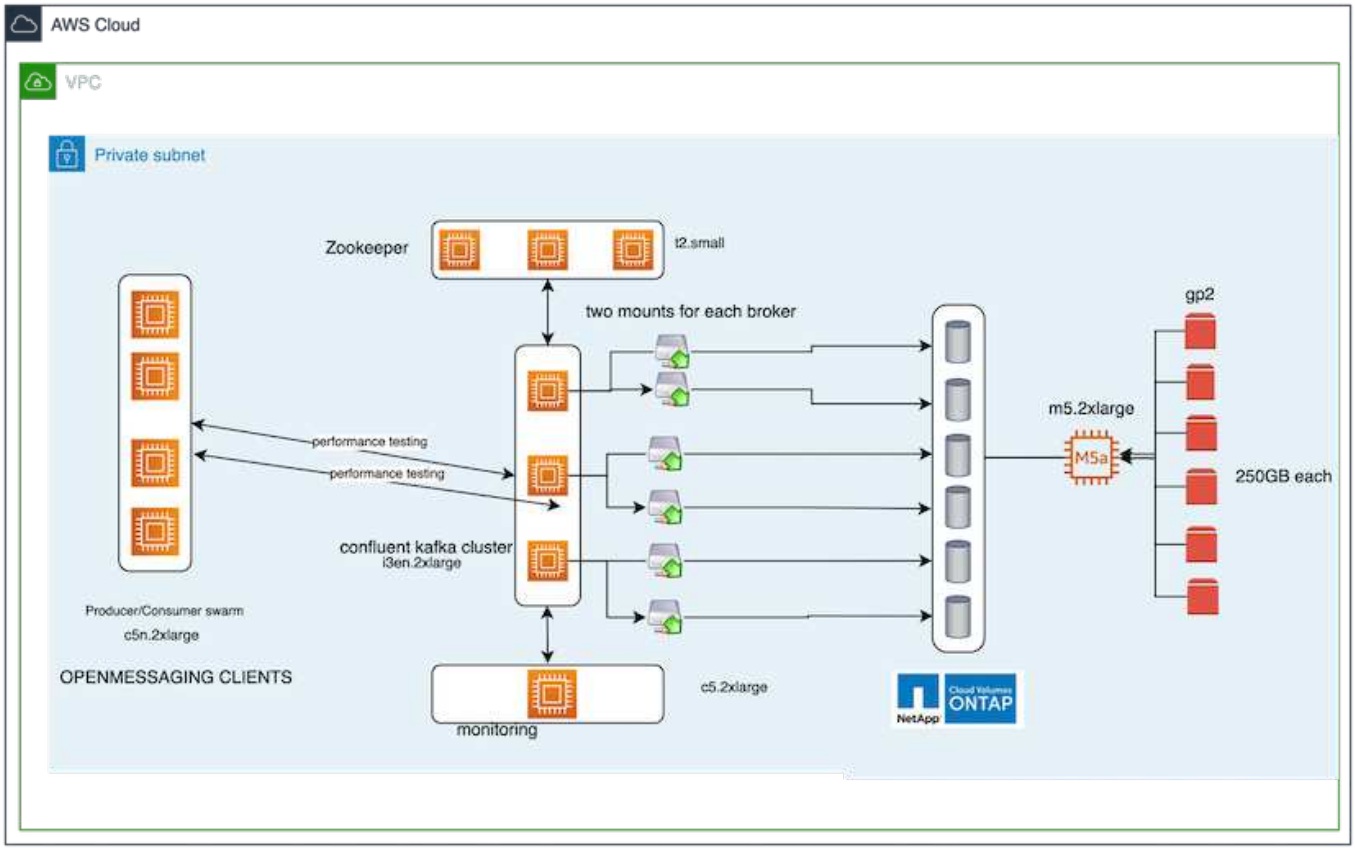
Configuração arquitetônica

A tabela a seguir mostra a configuração ambiental usada para demonstrar a utilização reduzida da CPU.

Componente de plataforma	Configuração do ambiente
Ferramenta de benchmarking do Kafka 3.2.3: OpenMessaging	• 3 tratadores de zoológico – t2.small • 3 servidores de corretor – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x Produtor/Consumidor — c5n.2xlarge
Sistema operacional em todos os nós	RHEL 8.7 ou posterior
Instância NetApp Cloud Volumes ONTAP	Instância de nó único – M5.2xLarge

Ferramenta de benchmarking

A ferramenta de benchmarking usada neste caso de teste é a "Mensagens abertas" estrutura. O OpenMessaging é neutro em relação a fornecedores e independente de linguagem; ele fornece diretrizes do setor para finanças, comércio eletrônico, IoT e big data; e ajuda a desenvolver aplicativos de mensagens e streaming em sistemas e plataformas heterogêneos. A figura a seguir descreve a interação de clientes do OpenMessaging com um cluster Kafka.



- **Calcular.** Usamos um cluster Kafka de três nós com um conjunto zookeeper de três nós em execução em servidores dedicados. Cada broker tinha dois pontos de montagem NFSv4.1 em um único volume na instância NetApp CVO por meio de um LIF dedicado.
- **Monitoramento.** Usamos dois nós para uma combinação Prometheus-Grafana. Para gerar cargas de trabalho, temos um cluster separado de três nós que pode produzir e consumir deste cluster Kafka.
- **Armazenar.** Usamos uma instância ONTAP de volumes NetApp Cloud de nó único com seis volumes GP2 AWS-EBS de 250 GB montados na instância. Esses volumes foram então expostos ao cluster Kafka como

seis volumes NFSv4.1 por meio de LIFs dedicados.

- **Configuração.** Os dois elementos configuráveis neste caso de teste foram os corretores Kafka e as cargas de trabalho do OpenMessaging.
  - **Configuração do corretor** As seguintes especificações foram selecionadas para os corretores Kafka. Utilizamos um fator de replicação de 3 para todas as medições, conforme destacado abaixo.

```
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
```

- **Configuração de carga de trabalho do benchmark OpenMessaging (OMB).** As seguintes especificações foram fornecidas. Especificamos uma taxa de produtor alvo, destacada abaixo.

```
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 de testes

1. Dois clusters semelhantes foram criados, cada um com seu próprio conjunto de enxames de clusters de referência.
   - **Grupo 1.** Cluster Kafka baseado em NFS.
   - **Grupo 2.** Cluster Kafka baseado em DAS.

2. Usando um comando OpenMessaging, cargas de trabalho semelhantes foram acionadas em cada cluster.

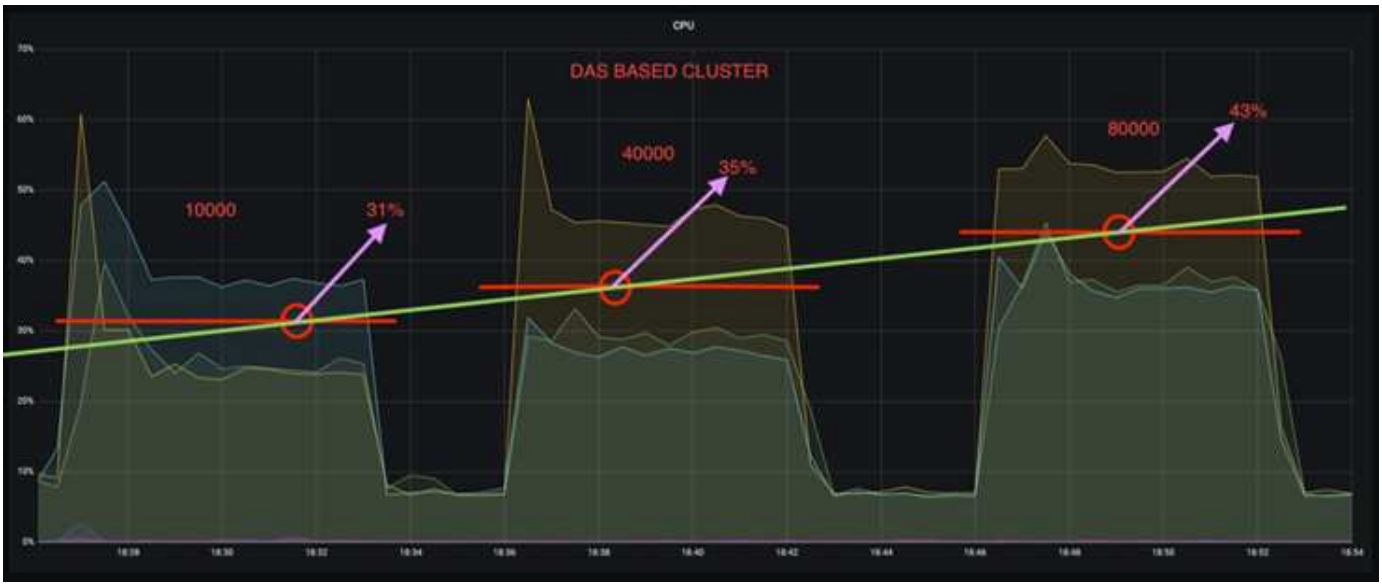
```
sudo bin/benchmark --drivers driver-kafka/kafka-group-all.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

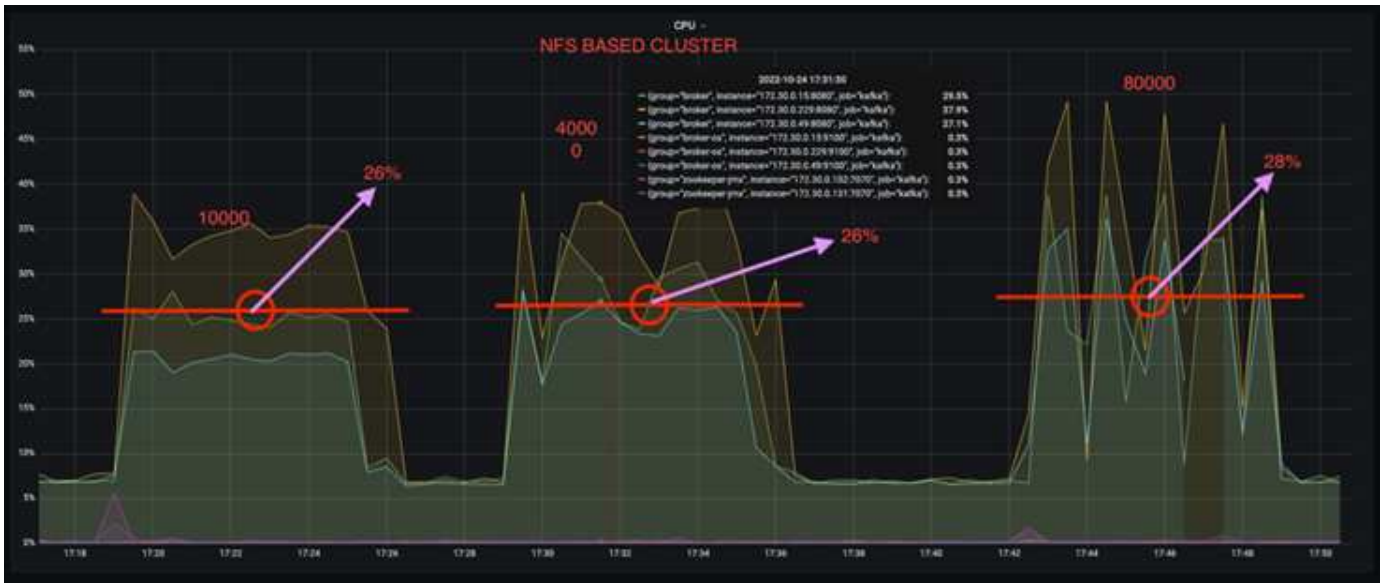
3. A configuração da taxa de produção foi aumentada em quatro iterações, e a utilização da CPU foi registrada com o Grafana. A taxa de produção foi definida nos seguintes níveis:
    - 10.000
    - 40.000
    - 80.000
    - 100.000

Observação

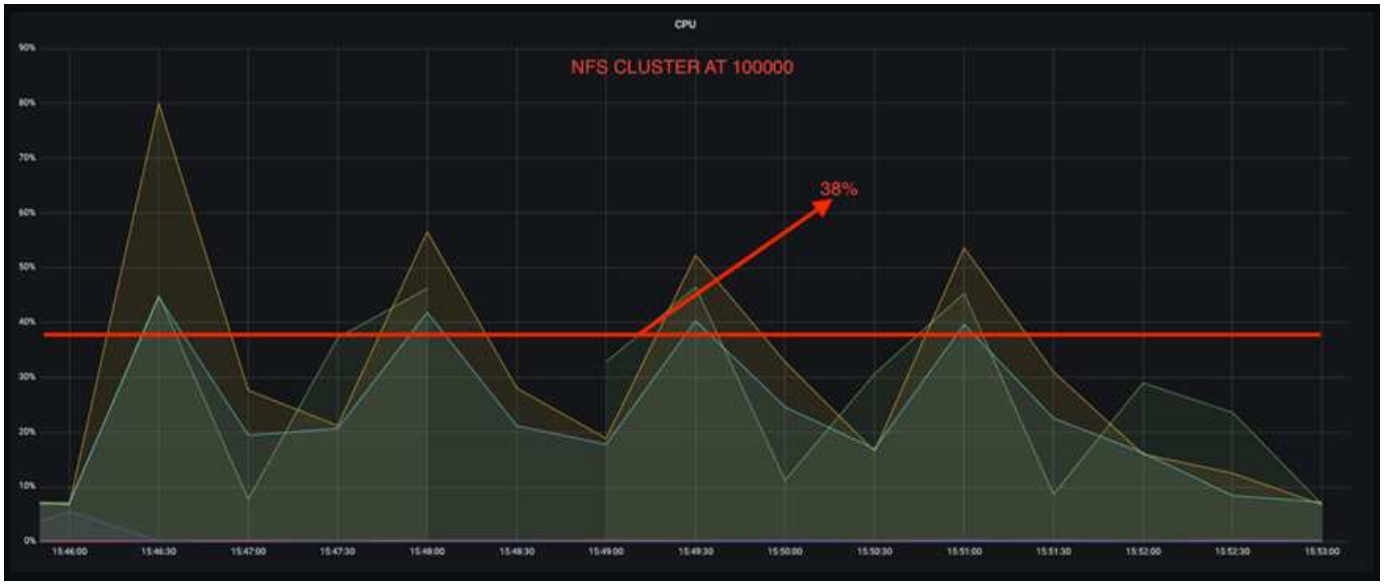
Há dois benefícios principais em usar o armazenamento NetApp NFS com o Kafka:

- **Você pode reduzir o uso da CPU em quase um terço.** O uso geral da CPU em cargas de trabalho semelhantes foi menor para NFS em comparação aos SSDs DAS; a economia variou de 5% para taxas de produção mais baixas a 32% para taxas de produção mais altas.
- **Uma redução de três vezes no desvio de utilização da CPU em taxas de produção mais altas.** Como esperado, houve um aumento na utilização da CPU à medida que as taxas de produção foram aumentadas. No entanto, a utilização da CPU em corretores Kafka usando DAS aumentou de 31% para a menor taxa de produção para 70% para a maior taxa de produção, um aumento de 39%. No entanto, com um backend de armazenamento NFS, a utilização da CPU aumentou de 26% para 38%, um aumento de 12%.





Além disso, com 100.000 mensagens, o DAS mostra mais utilização de CPU do que um cluster NFS.



Recuperação mais rápida do corretor

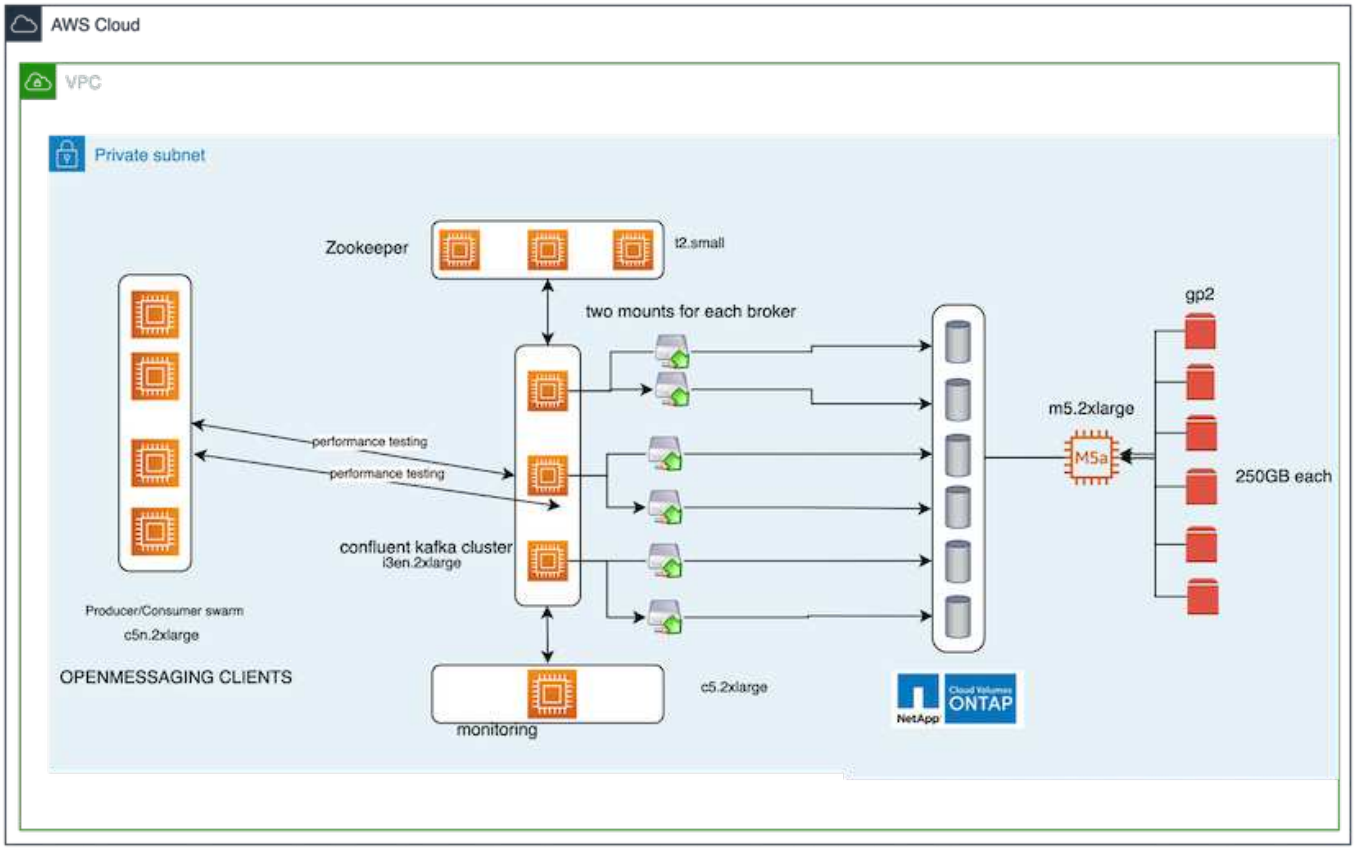
Descobrimos que os corretores do Kafka se recuperam mais rápido quando usam armazenamento NFS compartilhado da NetApp . Quando um broker falha em um cluster do Kafka, esse broker pode ser substituído por um broker íntegro com o mesmo ID de broker. Ao executar este caso de teste, descobrimos que, no caso de um cluster Kafka baseado em DAS, o cluster reconstrói os dados em um broker saudável recém-adicionado, o que consome tempo. No caso de um cluster Kafka baseado em NetApp NFS, o broker substituto continua lendo dados do diretório de log anterior e se recupera muito mais rápido.

Configuração arquitetônica

A tabela a seguir mostra a configuração ambiental para um cluster Kafka usando NAS.

Componente de plataforma	Configuração do ambiente
Kafka 3.2.3	• 3 tratadores de zoológico – t2.small • 3 servidores de corretor – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x produtor/consumidor — c5n.2xlarge • 1 x nó Kafka de backup – i3en.2xlarge
Sistema operacional em todos os nós	RHEL8.7 ou posterior
Instância NetApp Cloud Volumes ONTAP	Instância de nó único – M5.2xLarge

A figura a seguir descreve a arquitetura de um cluster Kafka baseado em NAS.



- **Calcular.** Um cluster Kafka de três nós com um conjunto de zookeepers de três nós em execução em servidores dedicados. Cada broker tem dois pontos de montagem NFS em um único volume na instância NetApp CVO por meio de um LIF dedicado.
- **Monitoramento.** Dois nós para uma combinação Prometheus-Grafana. Para gerar cargas de trabalho, usamos um cluster separado de três nós que pode produzir e consumir neste cluster Kafka.
- **Armazenar.** Uma instância ONTAP de volumes NetApp Cloud de nó único com seis volumes GP2 AWS-EBS de 250 GB montados na instância. Esses volumes são então expostos ao cluster Kafka como seis volumes NFS por meio de LIFs dedicados.
- **Configuração do corretor.** O único elemento configurável neste caso de teste são os corretores Kafka. As seguintes especificações foram selecionadas para os corretores Kafka. O `replica.lag.time.mx.ms` é definido como um valor alto porque isso determina a rapidez com que um nó específico é retirado da lista ISR. Ao alternar entre nós ruins e saudáveis, você não quer que o ID do broker seja excluído da 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
```

Metodologia de testes

1. Dois clusters semelhantes foram criados:
   - Um cluster confluente baseado em EC2.
   - Um cluster confluente baseado em NetApp NFS.
2. Um nó Kafka de espera foi criado com uma configuração idêntica aos nós do cluster Kafka original.
3. Em cada um dos clusters, um tópico de amostra foi criado e aproximadamente 110 GB de dados foram preenchidos em cada um dos corretores.
   - **Cluster baseado em EC2.** Um diretório de dados do corretor Kafka é mapeado em `/mnt/data-2` (Na figura a seguir, Broker-1 do cluster1 [terminal esquerdo]).
   - * Cluster baseado em NetApp NFS.* Um diretório de dados do broker Kafka é montado no ponto NFS `/mnt/data` (Na figura a seguir, Broker-1 do cluster2 [terminal direito]).

4. Em cada um dos clusters, o Broker-1 foi encerrado para acionar um processo de recuperação do broker com falha.
5. Após o encerramento do broker, o endereço IP do broker foi atribuído como um IP secundário ao broker em espera. Isso foi necessário porque um corretor em um cluster Kafka é identificado pelo seguinte:
   - **Endereço IP.** Atribuído pela reatribuição do IP do broker com falha ao broker em espera.
   - **ID do corretor.** Isso foi configurado no corretor standby `server.properties` .
6. Após a atribuição de IP, o serviço Kafka foi iniciado no broker em espera.
7. Depois de um tempo, os logs do servidor foram extraídos para verificar o tempo necessário para construir dados no nó de substituição no cluster.

Observação

A recuperação do corretor Kafka foi quase nove vezes mais rápida. O tempo necessário para recuperar um nó de broker com falha foi significativamente mais rápido ao usar o armazenamento compartilhado NetApp NFS em comparação ao uso de SSDs DAS em um cluster Kafka. Para 1 TB de dados de tópicos, o tempo de recuperação para um cluster baseado em DAS foi de 48 minutos, em comparação com menos de 5 minutos para um cluster Kafka baseado em NetApp-NFS.

Observamos que o cluster baseado em EC2 levou 10 minutos para reconstruir os 110 GB de dados no novo nó do broker, enquanto o cluster baseado em NFS concluiu a recuperação em 3 minutos. Também observamos nos logs que os deslocamentos do consumidor para as partições do EC2 eram 0, enquanto, no cluster NFS, os deslocamentos do consumidor eram coletados do broker 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$)
[2022-10-31 08:55:55,170] INFO [LogLoader partition=test-topic-qbVsEZg-
0000-8, dir=/mnt/data-1] Loading producer state till offset 0 with message
format version 2 (kafka.log.UnifiedLog$)
```

Cluster baseado em DAS

1. O nó de backup foi iniciado em 08:55:53.730.

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

2. O processo de reconstrução de dados terminou em 09:05:24.860. O processamento de 110 GB de dados levou 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)
```

Cluster baseado em NFS

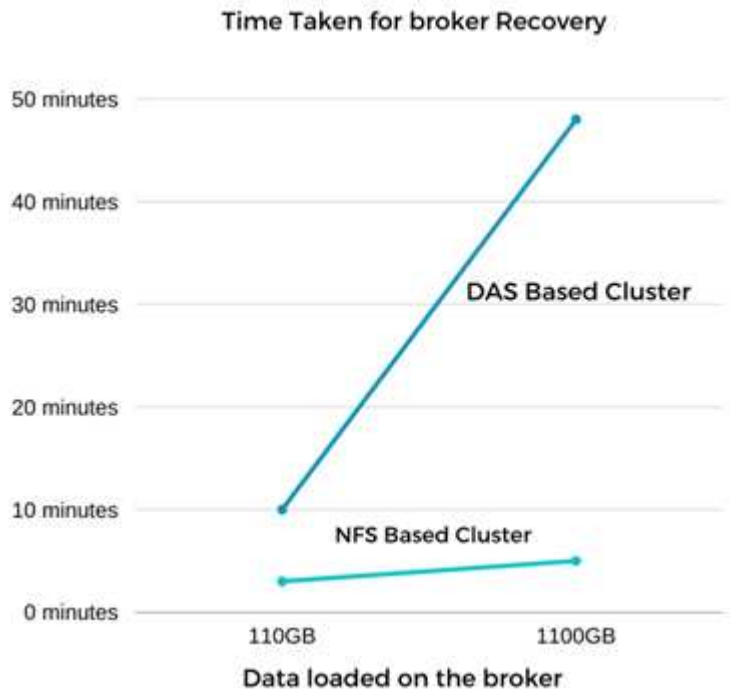
1. O nó de backup foi iniciado em 09:39:17.213. A entrada de log inicial é destacada abaixo.

```
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
```

2. O processo de reconstrução de dados terminou em 09:42:29.115. O processamento de 110 GB de dados levou 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)
```

O teste foi repetido para corretores contendo cerca de 1 TB de dados, o que levou aproximadamente 48 minutos para o DAS e 3 minutos para o NFS. Os resultados são mostrados no gráfico a seguir.



Eficiência de armazenamento

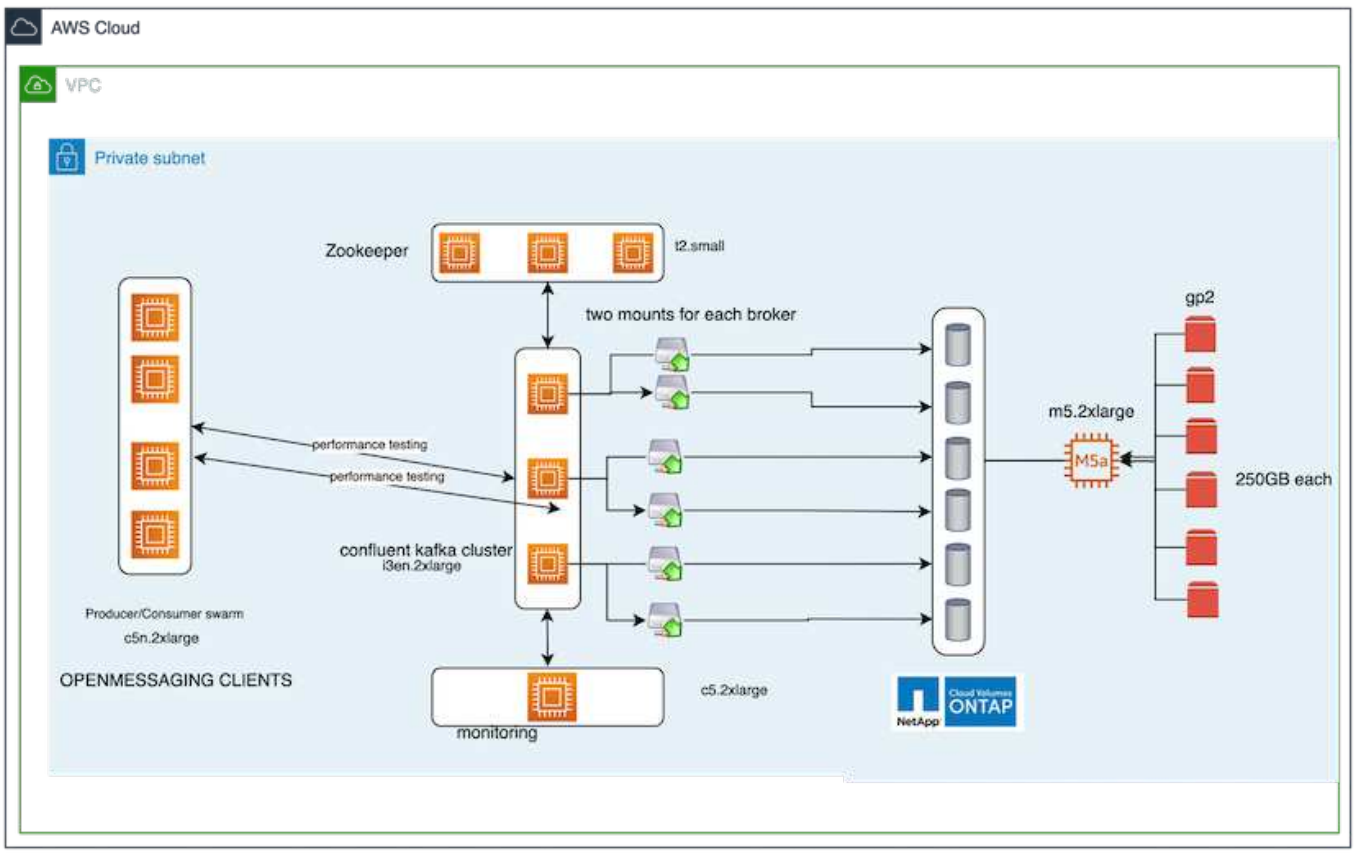
Como a camada de armazenamento do cluster Kafka foi provisionada pelo NetApp ONTAP, obtivemos todos os recursos de eficiência de armazenamento do ONTAP. Isso foi testado gerando uma quantidade significativa de dados em um cluster Kafka com armazenamento NFS provisionado no Cloud Volumes ONTAP. Pudemos ver que houve uma redução significativa de espaço devido aos recursos do ONTAP .

Configuração arquitetônica

A tabela a seguir mostra a configuração ambiental para um cluster Kafka usando NAS.

Componente de plataforma	Configuração do ambiente
Kafka 3.2.3	• 3 tratadores de zoológico – t2.small • 3 servidores de corretor – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x produtor/consumidor — c5n.2xlarge *
Sistema operacional em todos os nós	RHEL8.7 ou posterior
Instância NetApp Cloud Volumes ONTAP	Instância de nó único – M5.2xLarge

A figura a seguir descreve a arquitetura de um cluster Kafka baseado em NAS.



- **Calcular.** Usamos um cluster Kafka de três nós com um conjunto zookeeper de três nós em execução em servidores dedicados. Cada corretor tinha dois pontos de montagem NFS em um único volume na instância NetApp CVO por meio de um LIF dedicado.

- **Monitoramento.** Usamos dois nós para uma combinação Prometheus-Grafana. Para gerar cargas de trabalho, usamos um cluster separado de três nós que poderia produzir e consumir neste cluster Kafka.
- **Armazenar.** Usamos uma instância NetApp Cloud Volumes ONTAP de nó único com seis volumes GP2 AWS-EBS de 250 GB montados na instância. Esses volumes foram então expostos ao cluster Kafka como seis volumes NFS por meio de LIFs dedicados.
- **Configuração.** Os elementos configuráveis neste caso de teste foram os corretores Kafka.

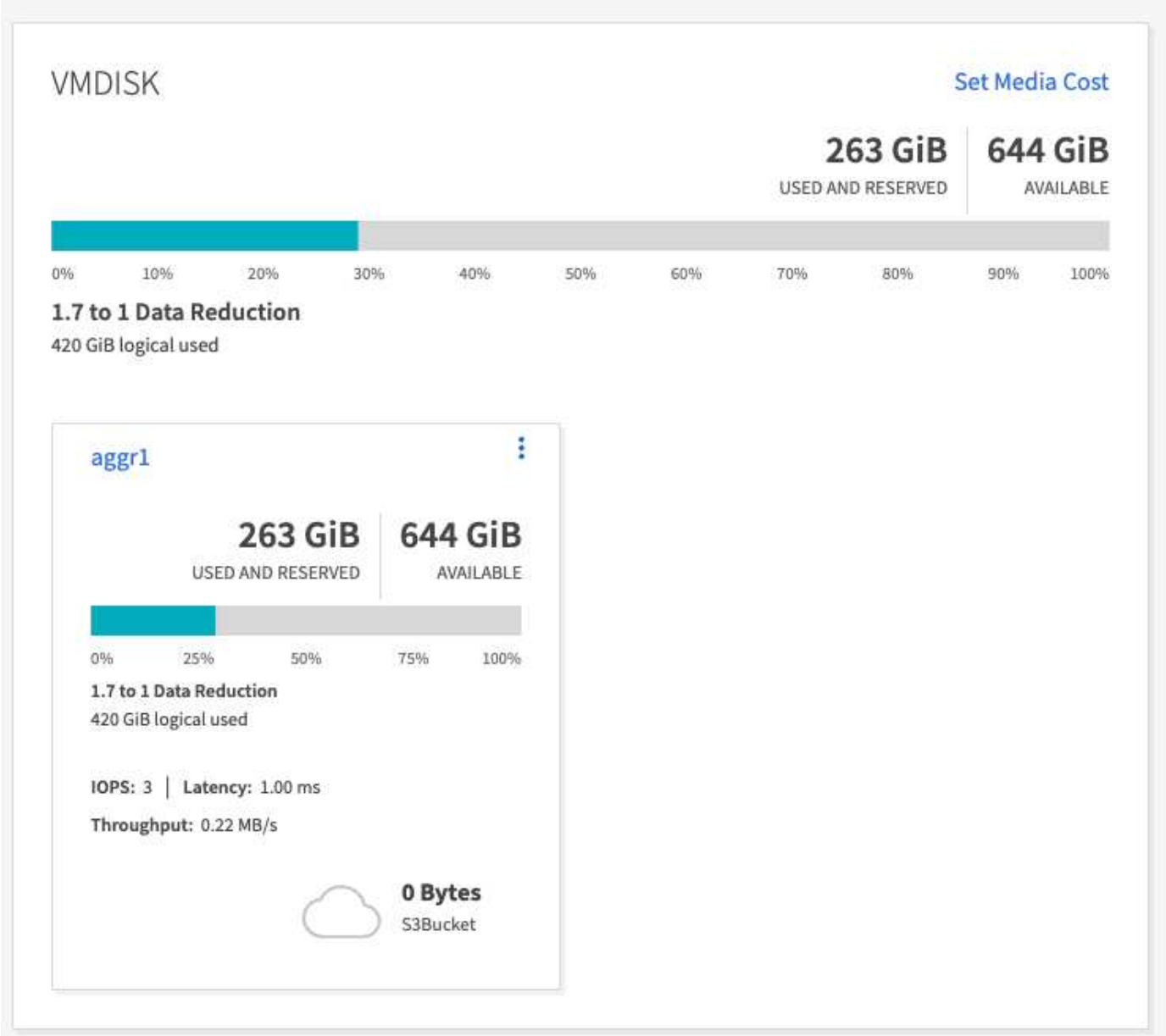
A compressão foi desativada pelo produtor, permitindo assim que ele gerasse alto rendimento. A eficiência do armazenamento era gerenciada pela camada de computação.

Metodologia de testes

1. Um cluster Kafka foi provisionado com as especificações mencionadas acima.
2. No cluster, cerca de 350 GB de dados foram produzidos usando a ferramenta OpenMessaging Benchmarking.
3. Após a conclusão da carga de trabalho, as estatísticas de eficiência de armazenamento foram coletadas usando o ONTAP System Manager e a CLI.

Observação

Para dados gerados usando a ferramenta OMB, observamos uma economia de espaço de ~33%, com uma taxa de eficiência de armazenamento de 1,70:1. Como visto nas figuras a seguir, o espaço lógico utilizado pelos dados produzidos foi de 420,3 GB e o espaço físico utilizado para armazenar os dados foi 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
```

Visão geral de desempenho e validação na AWS

Um cluster Kafka com a camada de armazenamento montada no NetApp NFS foi avaliado quanto ao desempenho na nuvem AWS. Os exemplos de benchmarking são descritos nas seções a seguir.

Kafka na nuvem AWS com NetApp Cloud Volumes ONTAP (par de alta disponibilidade e nó único)

Um cluster Kafka com NetApp Cloud Volumes ONTAP (par HA) foi avaliado quanto ao desempenho na nuvem AWS. Esse benchmarking é descrito nas seções a seguir.

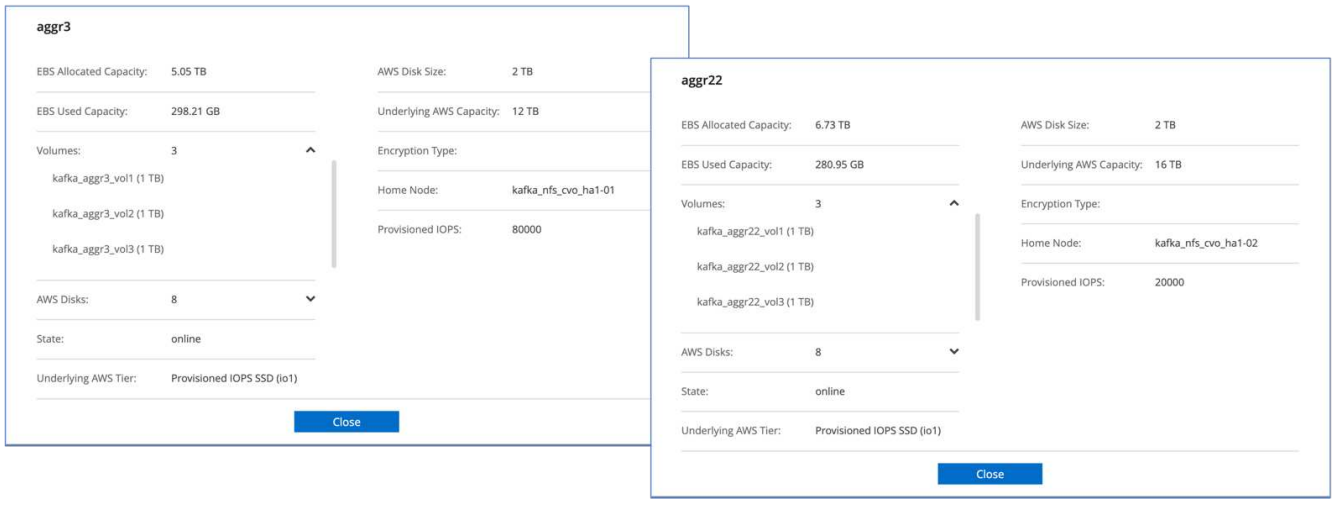
Configuração arquitetônica

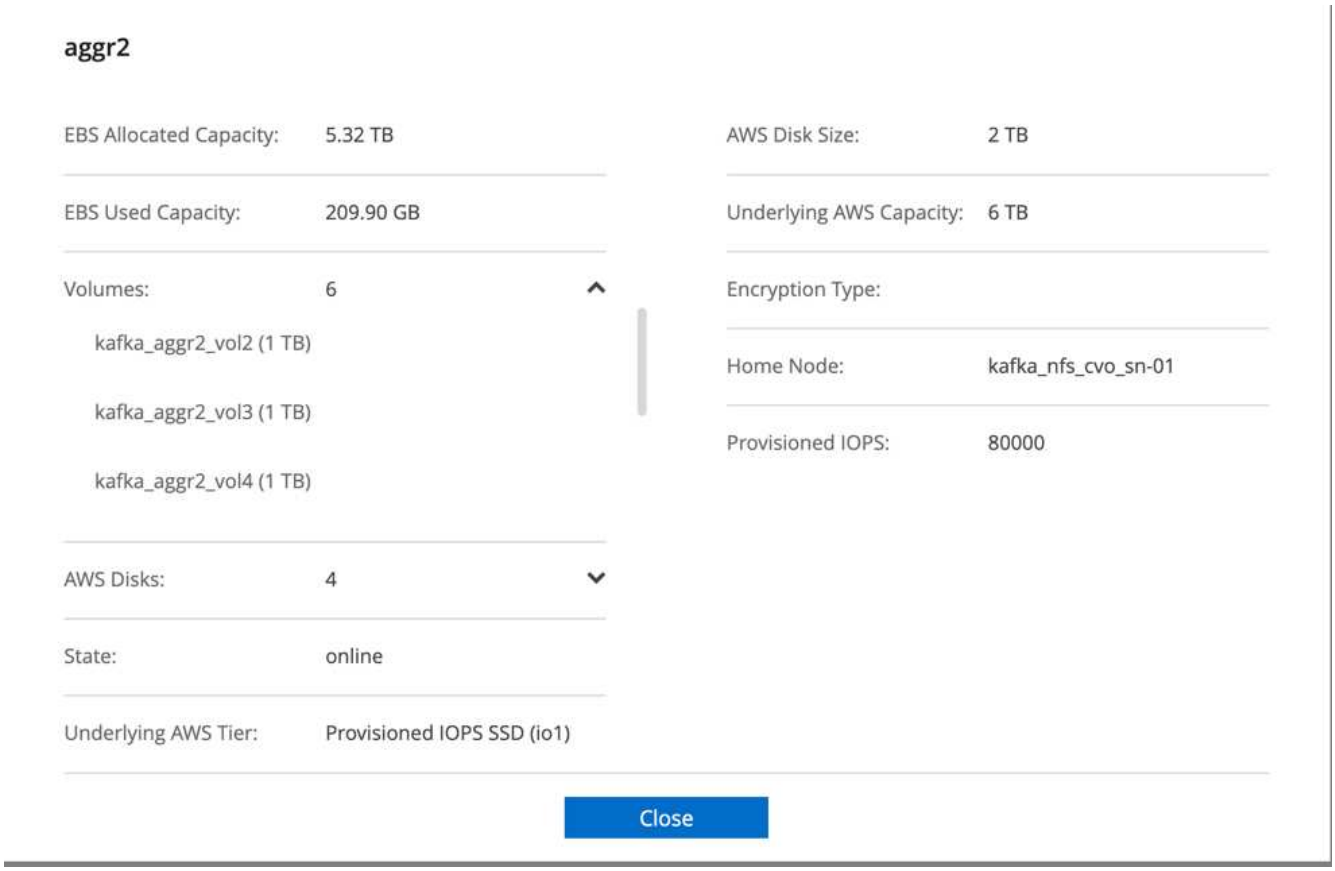
A tabela a seguir mostra a configuração ambiental para um cluster Kafka usando NAS.

Componente de plataforma	Configuração do ambiente
Kafka 3.2.3	• 3 tratadores de zoológico – t2.small • 3 servidores de corretor – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x produtor/consumidor — c5n.2xlarge *
Sistema operacional em todos os nós	RHEL8.6
Instância NetApp Cloud Volumes ONTAP	Instância de par HA – m5dn.12xLarge x 2node Instância de nó único – m5dn.12xLarge x 1 nó

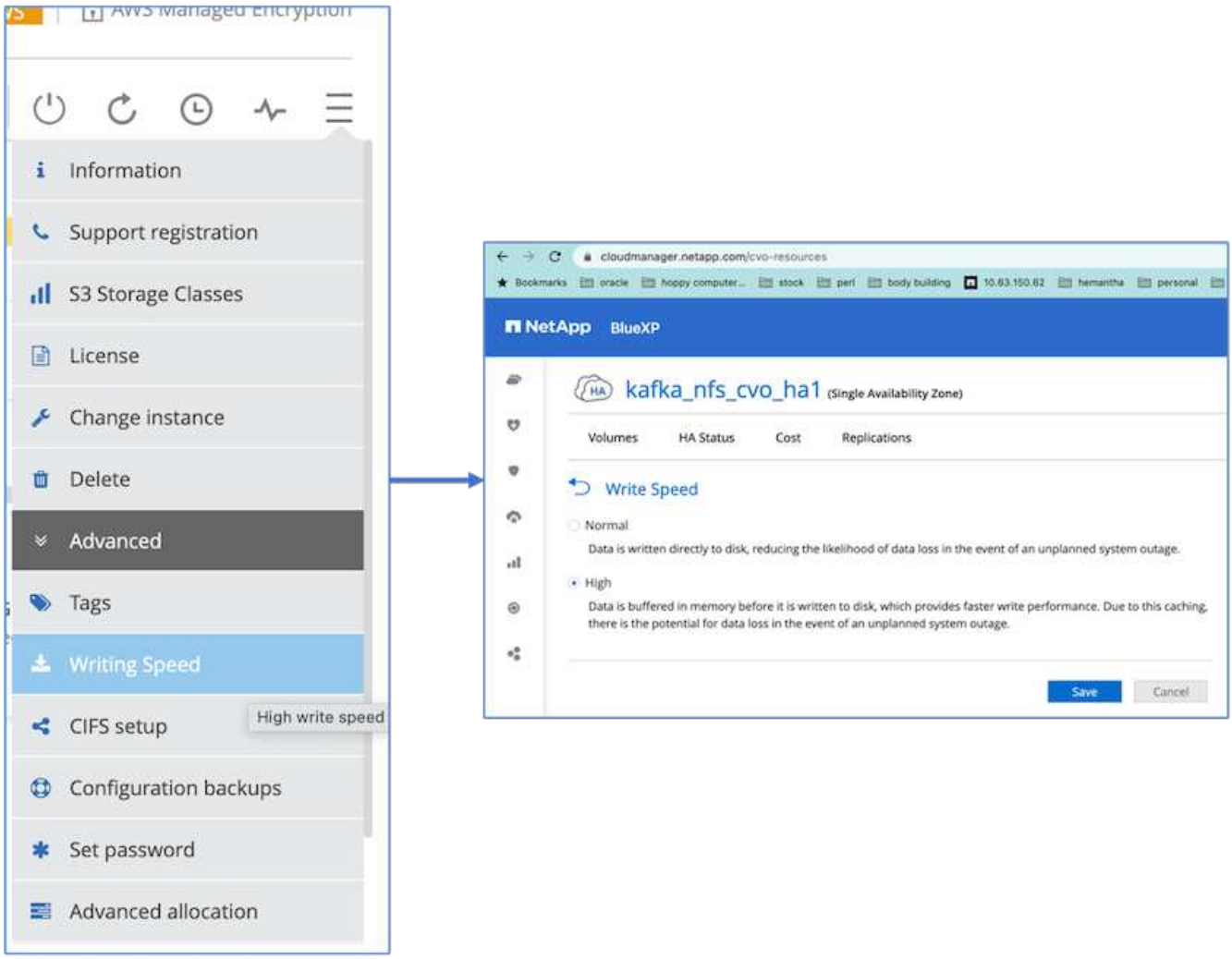
Configuração do volume do cluster NetApp ONTAP

1. Para o par Cloud Volumes ONTAP HA, criamos dois agregados com três volumes em cada agregado em cada controlador de armazenamento. Para o nó único do Cloud Volumes ONTAP , criamos seis volumes em um agregado.



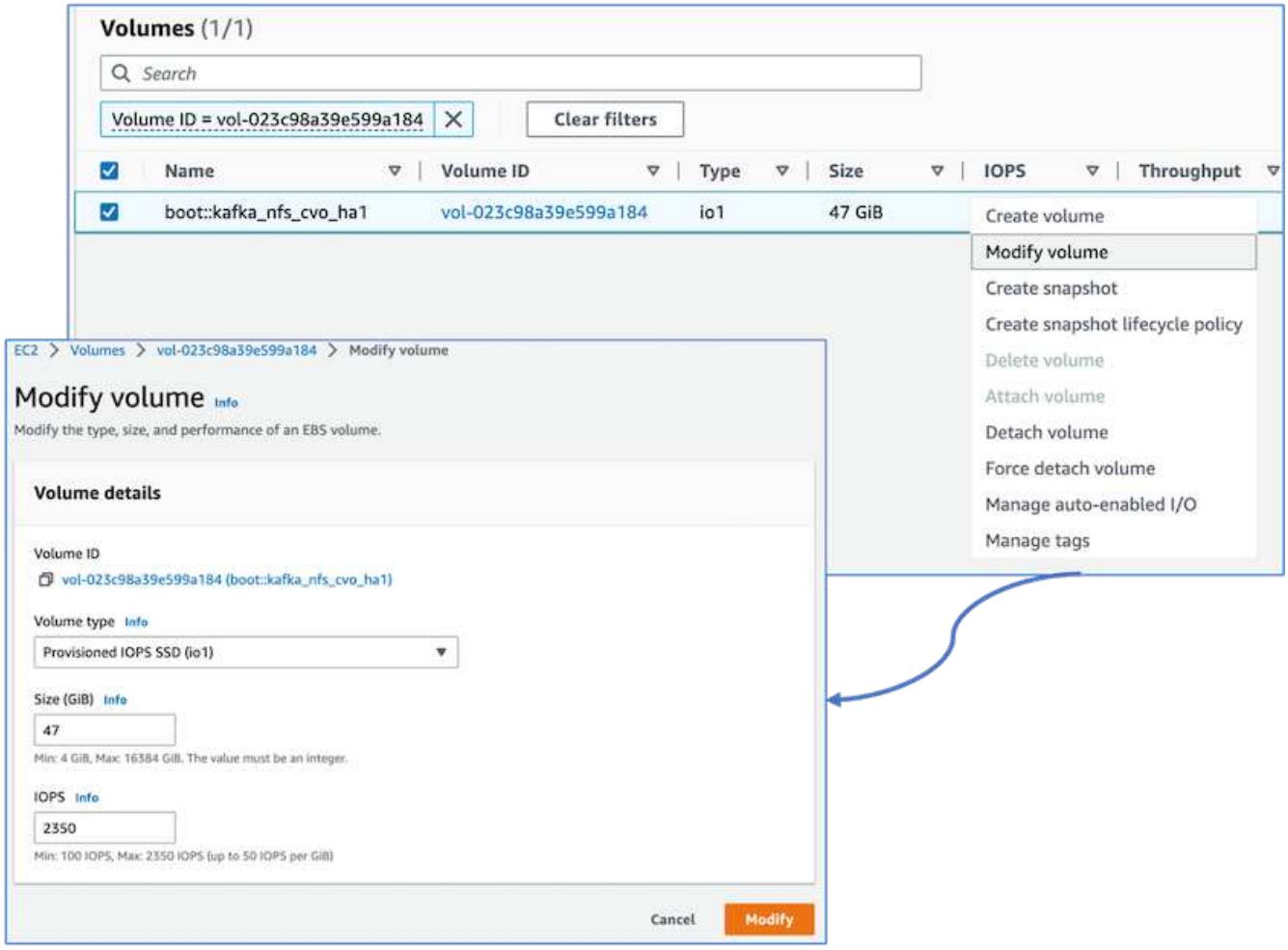


2. Para obter melhor desempenho de rede, habilitamos a rede de alta velocidade para o par HA e o nó único.



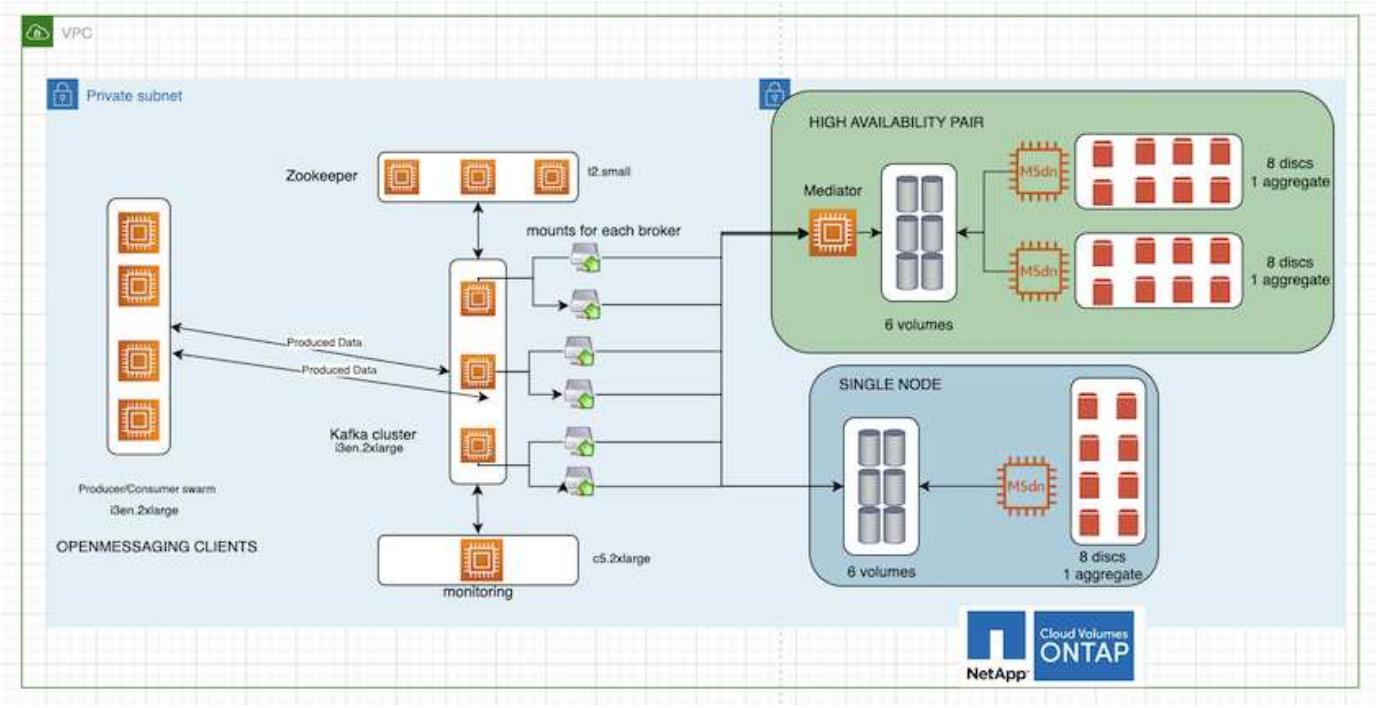
3. Percebemos que a NVRAM ONTAP tinha mais IOPS, então alteramos o IOPS para 2350 para o volume raiz Cloud Volumes ONTAP . O disco de volume raiz no Cloud Volumes ONTAP tinha 47 GB de tamanho. O seguinte comando ONTAP é para o par HA, e a mesma etapa é aplicável para o nó único.

```
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::*>
```



A figura a seguir descreve a arquitetura de um cluster Kafka baseado em NAS.

- **Calcular.** Usamos um cluster Kafka de três nós com um conjunto zookeeper de três nós em execução em servidores dedicados. Cada broker tinha dois pontos de montagem NFS para um único volume na instância do Cloud Volumes ONTAP por meio de um LIF dedicado.
- **Monitoramento.** Usamos dois nós para uma combinação Prometheus-Grafana. Para gerar cargas de trabalho, usamos um cluster separado de três nós que poderia produzir e consumir neste cluster Kafka.
- **Armazenar.** Usamos uma instância ONTAP de volumes em nuvem de par HA com um volume GP3 AWS-EBS de 6 TB montado na instância. O volume foi então exportado para o broker Kafka com uma montagem NFS.



Configurações de benchmarking do OpenMessage

1. Para melhor desempenho do NFS, precisamos de mais conexões de rede entre o servidor NFS e o cliente NFS, que podem ser criadas usando nconnect. Monte os volumes NFS nos nós do broker com a opção nconnect executando o seguinte 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 as conexões de rede no Cloud Volumes ONTAP. O seguinte comando ONTAP é usado a partir do nó Cloud Volumes ONTAP . A mesma etapa é aplicável ao 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 o seguinte Kafka `server.properties` em todos os corretores Kafka para o par Cloud Volumes ONTAP HA. O `log.dirs` a propriedade é diferente para cada corretor, e as propriedades restantes são comuns para os corretores. Para o broker1, o `log.dirs` o valor é o seguinte:

```
[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 o broker2, o `log.dirs` o valor do imóvel é o seguinte:

```
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 o broker3, o `log.dirs` o valor do imóvel é o seguinte:

```
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 o nó único do Cloud Volumes ONTAP , o Kafka `servers.properties` é o mesmo que para o par Cloud Volumes ONTAP HA, exceto para `log.dirs` propriedade.
   - Para o broker1, o `log.dirs` o valor é o seguinte:

```
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 o broker2, o `log.dirs` o valor é o seguinte:

```
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 o broker3, o `log.dirs` o valor do imóvel é o seguinte:

```
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. A carga de trabalho no OMB é configurada com as seguintes propriedades: `(/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
```

   O `messageSize` pode variar para cada caso de uso. Em nosso teste de desempenho, usamos 3K.

Usamos dois drivers diferentes, Sync ou Throughput, do OMB para gerar a carga de trabalho no cluster Kafka.

- O arquivo yaml usado para propriedades do driver de sincronização é o seguinte (`/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
```

- O arquivo yaml usado para as propriedades do driver Throughput é o seguinte (`/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 de testes

1. Um cluster Kafka foi provisionado conforme a especificação descrita acima usando Terraform e Ansible. O Terraform é usado para construir a infraestrutura usando instâncias da AWS para o cluster Kafka e o Ansible constrói o cluster Kafka nelas.
2. Uma carga de trabalho OMB foi acionada com a configuração de carga de trabalho descrita acima e o driver Sync.

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

3. Outra carga de trabalho foi acionada com o driver Throughput com a mesma configuração de carga de trabalho.

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

Observação

Dois tipos diferentes de drivers foram usados para gerar cargas de trabalho para comparar o desempenho de uma instância do Kafka em execução no NFS. A diferença entre os drivers é a propriedade log flush.

Para um par de Cloud Volumes ONTAP HA:

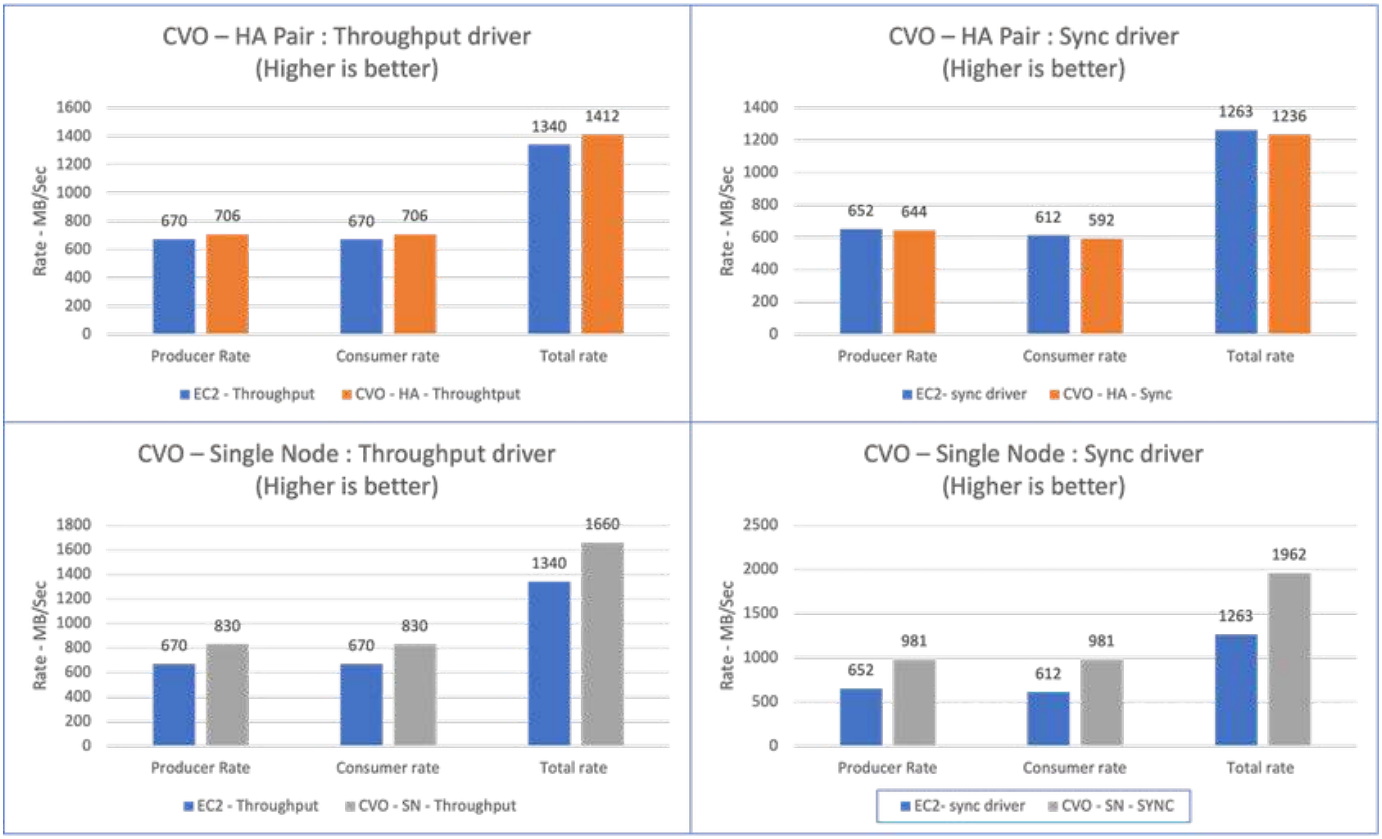
- Taxa de transferência total gerada consistentemente pelo driver de sincronização: ~1236 MBps.
- Taxa de transferência total gerada para o driver de taxa de transferência: pico de ~1412 MBps.

Para um único nó Cloud Volumes ONTAP :

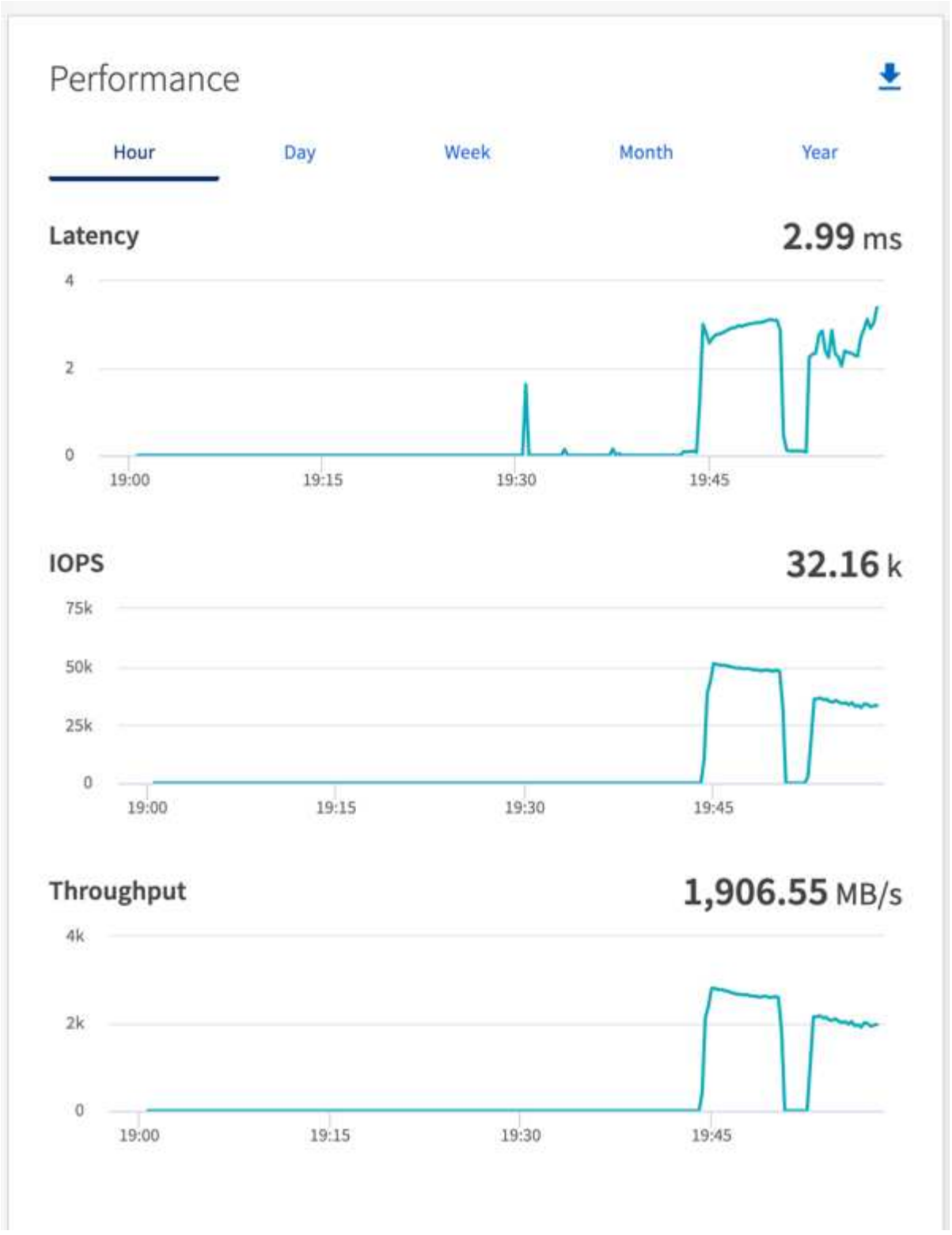
- Taxa de transferência total gerada consistentemente pelo driver de sincronização: ~ 1962 MBps.
- Taxa de transferência total gerada pelo driver de taxa de transferência: pico ~1660 MBps

O driver Sync pode gerar uma taxa de transferência consistente, pois os logs são liberados no disco instantaneamente, enquanto o driver Throughput gera picos de taxa de transferência, pois os logs são confirmados no disco em massa.

Esses números de taxa de transferência são gerados para a configuração da AWS fornecida. Para requisitos de desempenho mais altos, os tipos de instância podem ser ampliados e ajustados ainda mais para obter melhores números de taxa de transferência. A produção total ou taxa total é a combinação das taxas do produtor e do consumidor.



Certifique-se de verificar a taxa de transferência de armazenamento ao executar o benchmark de taxa de transferência ou driver de sincronização.



Visão geral de desempenho e validação no AWS FSx ONTAP

Um cluster Kafka com a camada de armazenamento montada no NetApp NFS foi avaliado quanto ao desempenho no AWS FSx ONTAP. Os exemplos de benchmarking são descritos nas seções a seguir.

Apache Kafka no AWS FSx ONTAP

O Network File System (NFS) é um sistema de arquivos de rede amplamente utilizado para armazenar grandes quantidades de dados. Na maioria das organizações, os dados estão sendo cada vez mais gerados por aplicativos de streaming como o Apache Kafka. Essas cargas de trabalho exigem escalabilidade, baixa latência e uma arquitetura de ingestão de dados robusta com recursos de armazenamento modernos. Para permitir análises em tempo real e fornecer insights acionáveis, é necessária uma infraestrutura bem projetada e de alto desempenho.

O Kafka funciona, por padrão, com um sistema de arquivos compatível com POSIX e depende do sistema de arquivos para manipular operações de arquivo, mas ao armazenar dados em um sistema de arquivos NFSv3, o cliente NFS do broker Kafka pode interpretar as operações de arquivo de forma diferente de um sistema de arquivos local, como XFS ou Ext4. Um exemplo comum é a renomeação do NFS Silly, que fazia com que os corretores do Kafka falhassem ao expandir clusters e realocar partições. Para lidar com esse desafio, a NetApp atualizou o cliente Linux NFS de código aberto com alterações agora disponíveis no RHEL8.7, RHEL9.1 e com suporte da versão atual do FSx ONTAP , ONTAP 9.12.1.

O Amazon FSx ONTAP fornece um sistema de arquivos NFS totalmente gerenciado, escalável e de alto desempenho na nuvem. Os dados do Kafka no FSx ONTAP podem ser dimensionados para lidar com grandes quantidades de dados e garantir tolerância a falhas. O NFS fornece gerenciamento de armazenamento centralizado e proteção de dados para conjuntos de dados críticos e sensíveis.

Esses aprimoramentos possibilitam que os clientes da AWS aproveitem o FSx ONTAP ao executar cargas de trabalho do Kafka nos serviços de computação da AWS. Esses benefícios são: * Redução da utilização da CPU para reduzir o tempo de espera de E/S * Tempo de recuperação mais rápido do broker Kafka. * Confiabilidade e eficiência. * Escalabilidade e desempenho. * Disponibilidade de Zona de Multidisponibilidade. * Proteção de dados.

Visão geral de desempenho e validação no AWS FSx ONTAP

Um cluster Kafka com a camada de armazenamento montada no NetApp NFS foi avaliado quanto ao desempenho na nuvem AWS. Os exemplos de benchmarking são descritos nas seções a seguir.

Kafka no AWS FSx ONTAP

Um cluster Kafka com AWS FSx ONTAP foi avaliado quanto ao desempenho na nuvem AWS. Esse benchmarking é descrito nas seções a seguir.

Configuração arquitetônica

A tabela a seguir mostra a configuração ambiental para um cluster Kafka usando o AWS FSx ONTAP.

Componente de plataforma	Configuração do ambiente
Kafka 3.2.3	• 3 tratadores de zoológico – t2.small • 3 servidores de corretor – i3en.2xlarge • 1 x Grafana – c5n.2xgrande • 4 x produtor/consumidor — c5n.2xlarge *
Sistema operacional em todos os nós	RHEL8.6
AWS FSx ONTAP	Multi-AZ com taxa de transferência de 4 GB/s e 160.000 IOPS

Configuração do NetApp FSx ONTAP

1. Para nossos testes iniciais, criamos um sistema de arquivos FSx ONTAP com 2 TB de capacidade e 40.000 IOPs para uma taxa de transferência de 2 GB/s.

```
[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"}
```

   Em nosso exemplo, estamos implantando o FSx ONTAP por meio do AWS CLI. Você precisará personalizar ainda mais o comando em seu ambiente, conforme necessário. O FSx ONTAP também pode ser implantado e gerenciado por meio do AWS Console para uma experiência de implantação mais fácil e otimizada, com menos entrada de linha de comando.

   Documentação No FSx ONTAP, o IOPS máximo atingível para um sistema de arquivos com taxa de transferência de 2 GB/s em nossa região de teste (US-East-1) é de 80.000 iops. O total máximo de iops para um sistema de arquivos FSx ONTAP é de 160.000 iops, o que requer uma implantação de taxa de transferência de 4 GB/s para ser alcançado, o que demonstraremos mais adiante neste documento.

   Para obter mais informações sobre as especificações de desempenho do FSx ONTAP , visite a documentação do AWS FSx ONTAP aqui: https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/performance.html .

   A sintaxe detalhada da linha de comando para FSx "create-file-system" pode ser encontrada aqui: https://docs.aws.amazon.com/cli/latest/reference/fsx/create-file-system.html

   Por exemplo, você pode especificar uma chave KMS específica em vez da chave mestra padrão do AWS FSx que é usada quando nenhuma chave KMS é especificada.

2. Ao criar o sistema de arquivos FSx ONTAP , aguarde até que o status "LifeCycle" mude para "AVAILABLE" no seu retorno JSON após descrever seu sistema de arquivos da seguinte maneira:

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

3. Valide as credenciais fazendo login no FSx ONTAP SSH com o usuário fsxadmin: Fsxadmin é a conta de administrador padrão para sistemas de arquivos FSx ONTAP na criação. A senha para fsxadmin é a senha que foi configurada ao criar o sistema de arquivos pela primeira vez no Console da AWS ou com a CLI da AWS, conforme concluímos na Etapa 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. Depois que suas credenciais forem validadas, crie a máquina virtual de armazenamento no sistema de arquivos 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
```

Uma Máquina Virtual de Armazenamento (SVM) é um servidor de arquivos isolado com suas próprias credenciais administrativas e pontos de extremidade para administrar e acessar dados em volumes FSx ONTAP e fornece multilocação do FSx ONTAP .

5. Depois de configurar sua máquina virtual de armazenamento primária, faça SSH no sistema de arquivos FSx ONTAP recém-criado e crie volumes na máquina virtual de armazenamento usando o comando de exemplo abaixo. Da mesma forma, criamos 6 volumes para essa validação. Com base em nossa validação, mantenha o constituinte padrão (8) ou menos constituintes, o que proporcionará melhor desempenho ao 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. Precisaremos de capacidade adicional em nossos volumes para nossos testes. Aumente o tamanho do volume para 2 TB e monte no caminho de junção.

```
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
```

No FSx ONTAP, os volumes podem ser provisionados de forma fina. Em nosso exemplo, a capacidade total do volume estendido excede a capacidade total do sistema de arquivos, então precisaremos estender a capacidade total do sistema de arquivos para desbloquear capacidade adicional do volume provisionado, o que demonstraremos na próxima etapa.

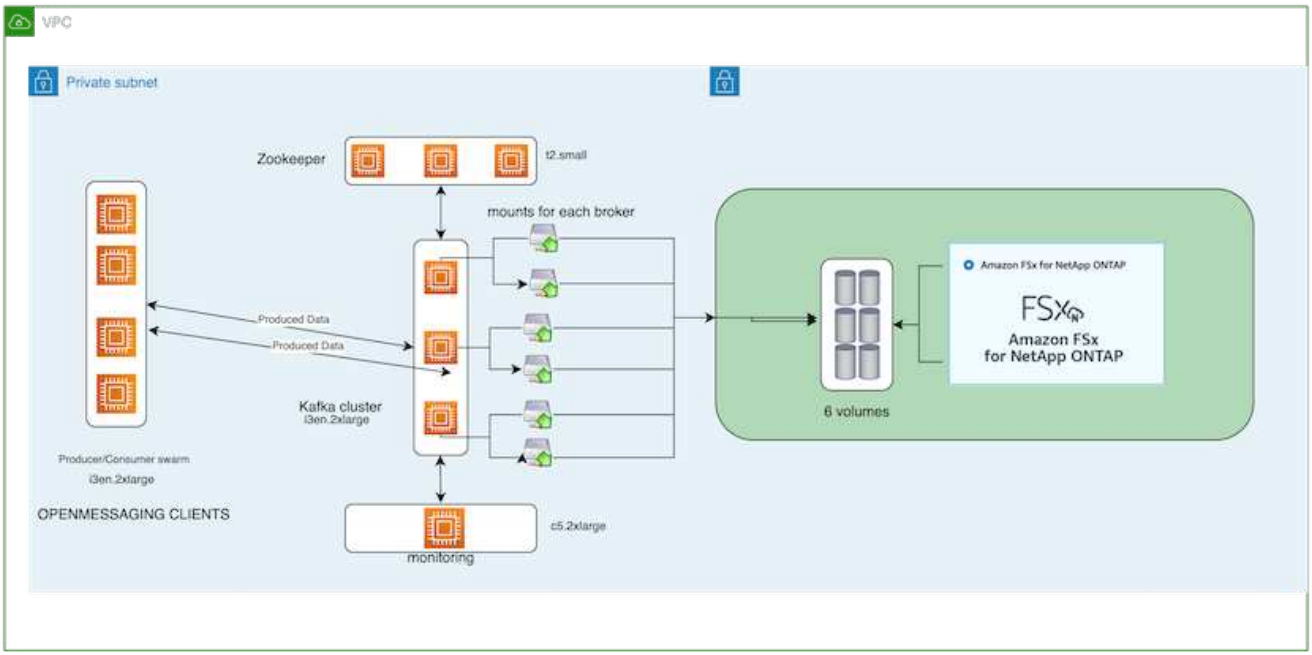
7. Em seguida, para desempenho e capacidade adicionais, ampliamos a capacidade de transferência do FSx ONTAP de 2 GB/seg para 4 GB/seg e IOPS para 160.000, e a capacidade para 5 TB

```
[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
```

A sintaxe detalhada da linha de comando para FSx "update-file-system" pode ser encontrada aqui:https://docs.aws.amazon.com/cli/latest/reference/fsx/update-file-system.html[]

8. Os volumes FSx ONTAP são montados com nconnect e opções padrão em corretores Kafka

A imagem a seguir mostra nossa arquitetura final do cluster Kafka baseado no FSx ONTAP :



- Calcular. Usamos um cluster Kafka de três nós com um conjunto zookeeper de três nós em execução em servidores dedicados. Cada corretor tinha seis pontos de montagem NFS para seis volumes na instância FSx ONTAP .
- Monitoramento. Usamos dois nós para uma combinação Prometheus-Grafana. Para gerar cargas de trabalho, usamos um cluster separado de três nós que poderia produzir e consumir neste cluster Kafka.
- Armazenar. Usamos um FSx ONTAP com seis volumes de 2 TB montados. O volume foi então exportado para o broker Kafka com uma montagem NFS. Os volumes FSx ONTAP são montados com 16 sessões nconnect e opções padrão nos brokers Kafka.

Configurações de benchmarking do OpenMessage.

Usamos a mesma configuração usada para os volumes ONTAP do NetApp Cloud e seus detalhes estão aqui - xref:./data-analytics/kafka-nfs-performance-overview-and-validation-in-aws.html#architectural-setup

Metodologia de testes

1. Um cluster Kafka foi provisionado conforme a especificação descrita acima usando Terraform e Ansible. O Terraform é usado para construir a infraestrutura usando instâncias da AWS para o cluster Kafka e o Ansible constrói o cluster Kafka nelas.
2. Uma carga de trabalho OMB foi acionada com a configuração de carga de trabalho descrita acima e o driver Sync.

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

3. Outra carga de trabalho foi acionada com o driver Throughput com a mesma configuração de carga de trabalho.

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

Observação

Dois tipos diferentes de drivers foram usados para gerar cargas de trabalho para comparar o desempenho de uma instância do Kafka em execução no NFS. A diferença entre os drivers é a propriedade log flush.

Para um fator de replicação Kafka 1 e o FSx ONTAP:

- Taxa de transferência total gerada consistentemente pelo driver Sync: ~ 3218 MBps e desempenho máximo em ~ 3652 MBps.
- Taxa de transferência total gerada consistentemente pelo driver Throughput: ~ 3679 MBps e desempenho máximo em ~ 3908 MBps.

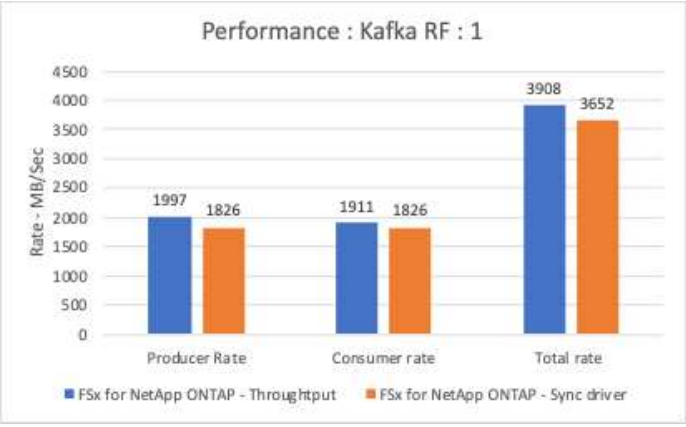
Para Kafka com fator de replicação 3 e FSx ONTAP :

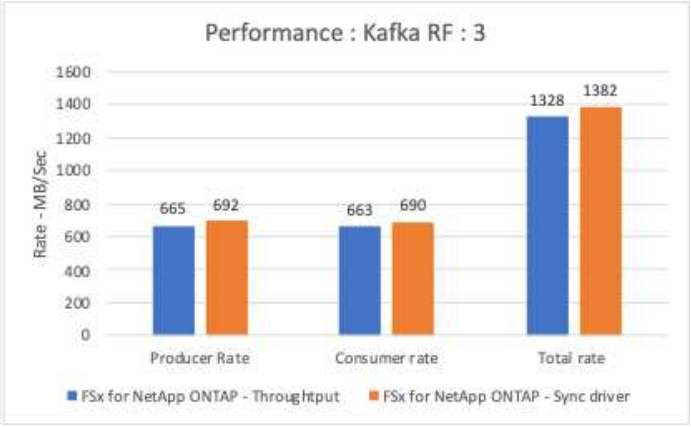
- Taxa de transferência total gerada consistentemente pelo driver Sync: ~ 1252 MBps e desempenho máximo em ~ 1382 MBps.
- Taxa de transferência total gerada consistentemente pelo driver Throughput: ~ 1218 MBps e desempenho máximo em ~ 1328 MBps.

No fator 3 de replicação do Kafka, a operação de leitura e gravação ocorreu três vezes no FSx ONTAP. No fator 1 de replicação do Kafka, a operação de leitura e gravação ocorreu uma vez no FSx ONTAP, portanto, em ambas as validações, conseguimos atingir a taxa de transferência máxima de 4 GB/s.

O driver Sync pode gerar uma taxa de transferência consistente, pois os logs são liberados no disco instantaneamente, enquanto o driver Throughput gera picos de taxa de transferência, pois os logs são confirmados no disco em massa.

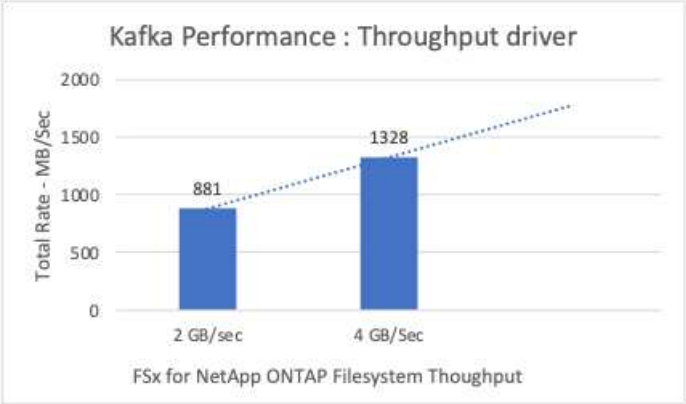
Esses números de taxa de transferência são gerados para a configuração da AWS fornecida. Para requisitos de desempenho mais altos, os tipos de instância podem ser ampliados e ajustados ainda mais para obter melhores números de taxa de transferência. A produção total ou taxa total é a combinação das taxas do produtor e do consumidor.

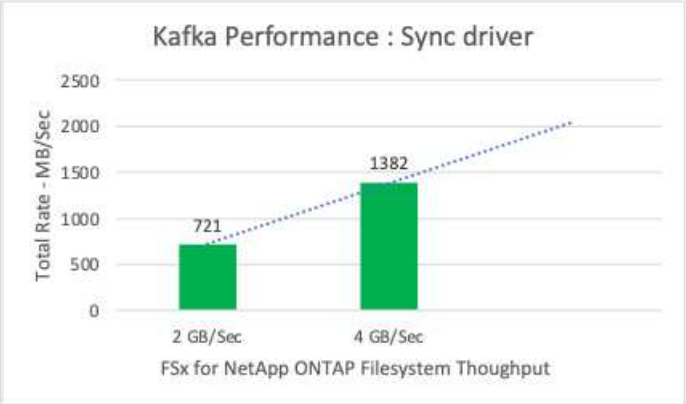




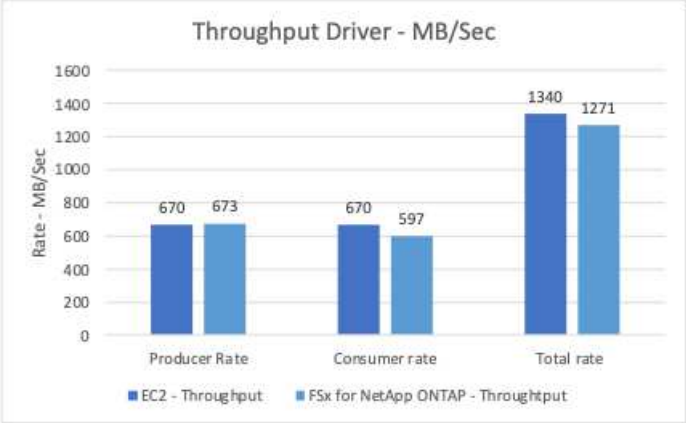
O gráfico abaixo mostra o desempenho de 2 GB/s do FSx ONTAP e de 4 GB/s para o fator de replicação 3 do Kafka. O fator de replicação 3 realiza a operação de leitura e gravação três vezes no armazenamento FSx ONTAP . A taxa total para o driver de transferência é de 881 MB/s, o que faz com que a operação de leitura e

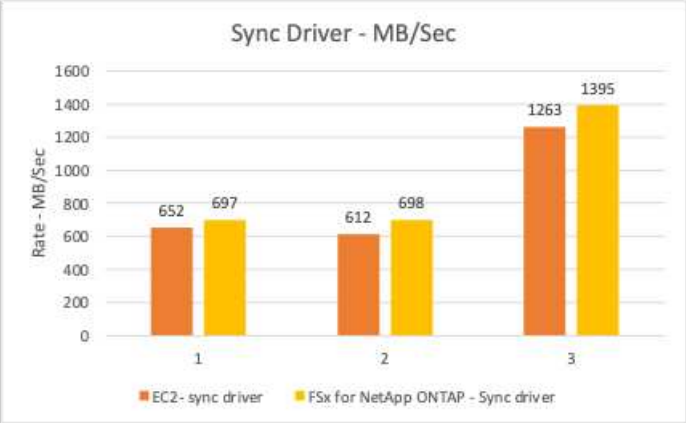
gravação do Kafka seja de aproximadamente 2,64 GB/s no sistema de arquivos FSx ONTAP de 2 GB/s, e a taxa total para o driver de transferência é de 1328 MB/s, o que faz com que a operação de leitura e gravação do Kafka seja de aproximadamente 3,98 GB/s. O desempenho do Kafka é linear e escalável com base na taxa de transferência do FSx ONTAP .





O gráfico abaixo mostra o desempenho entre a instância EC2 vs FSx ONTAP (Fator de Replicação Kafka: 3)

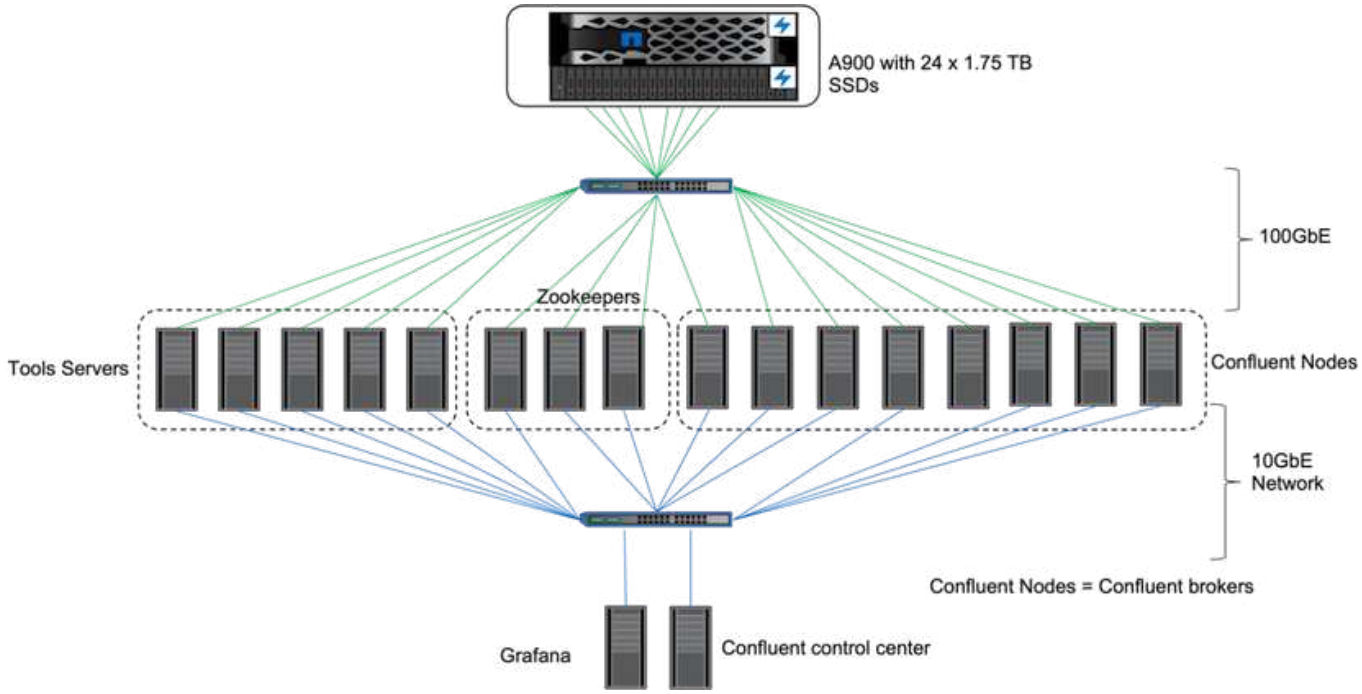




Visão geral de desempenho e validação com AFF A900 local

No local, usamos o controlador de armazenamento NetApp AFF A900 com ONTAP 9.12.1RC1 para validar o desempenho e o dimensionamento de um cluster Kafka. Usamos o mesmo ambiente de teste de nossas práticas recomendadas de armazenamento em camadas anteriores com ONTAP e AFF.

Usamos o Confluent Kafka 6.2.0 para avaliar o AFF A900. O cluster conta com oito nós de corretores e três nós de tratadores de zoológicos. Para testes de desempenho, usamos cinco nós de trabalho OMB.



Configuração de armazenamento

Usamos instâncias do NetApp FlexGroups para fornecer um único namespace para diretórios de log, simplificando a recuperação e a configuração. Usamos NFSv4.1 e pNFS para fornecer acesso direto ao caminho para dados de segmento de log.

Ajuste de cliente

Cada cliente montou a instância do FlexGroup com o seguinte comando.

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

Além disso, aumentamos a `max_session_slots` do padrão `64` para `180` . Isso corresponde ao limite de slot de sessão padrão no ONTAP.

Ajuste do corretor Kafka

Para maximizar o rendimento no sistema em teste, aumentamos significativamente os parâmetros padrão para determinados pools de threads principais. Recomendamos seguir as práticas recomendadas do Confluent Kafka para a maioria das configurações. Esse ajuste foi usado para maximizar a simultaneidade de E/S pendentes para armazenamento. Esses parâmetros podem ser ajustados para corresponder aos recursos de computação e atributos de armazenamento do seu corretor.

```
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 de teste do gerador de carga de trabalho

Usamos as mesmas configurações de OMB usadas nos testes de nuvem para o driver de throughput e a configuração do tópico.

1. Uma instância do FlexGroup foi provisionada usando o Ansible em um 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. O pNFS foi habilitado no ONTAP SVM.

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

3. A carga de trabalho foi acionada com o driver Throughput usando a mesma configuração de carga de trabalho do Cloud Volumes ONTAP. Veja a seção "Desempenho em estado estacionário " abaixo. A carga de trabalho usou um fator de replicação de 3, o que significa que três cópias de segmentos de log foram mantidas no NFS.

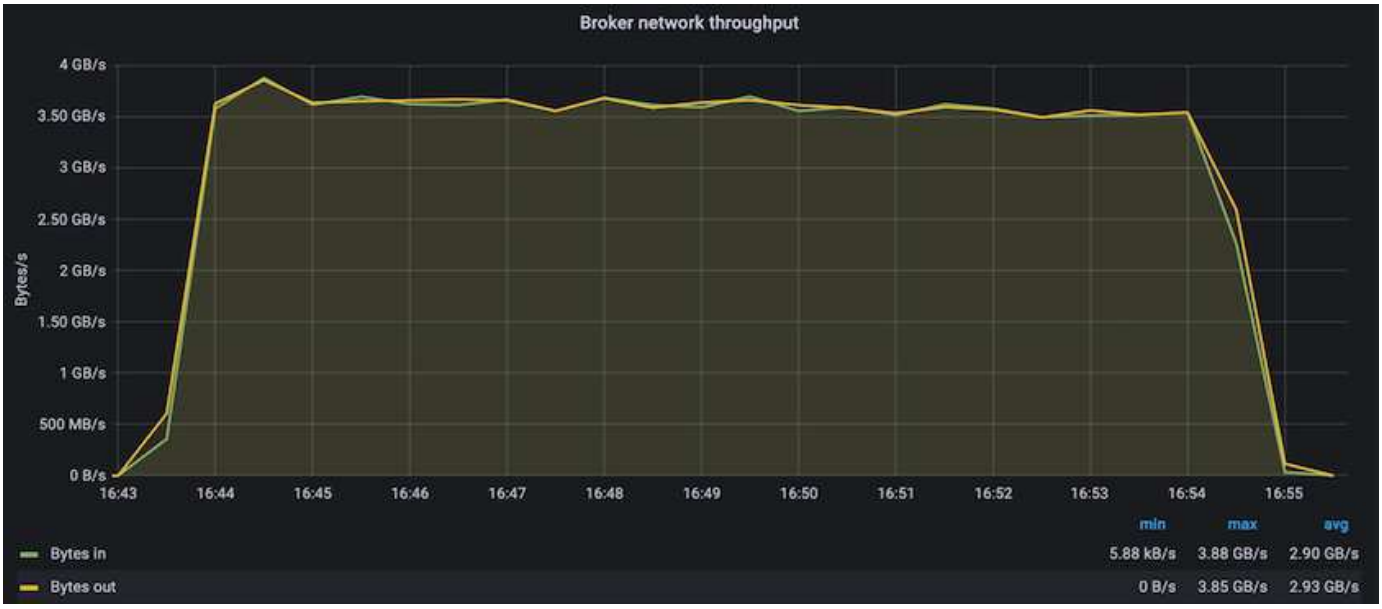
```
sudo bin/benchmark --drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

4. Por fim, concluímos medições usando um backlog para medir a capacidade dos consumidores de acompanhar as mensagens mais recentes. O OMB cria um backlog pausando os consumidores durante o início de uma medição. Isso produz três fases distintas: criação de backlog (tráfego somente do produtor), redução de backlog (uma fase com grande demanda do consumidor, na qual os consumidores recuperam os eventos perdidos em um tópico) e o estado estável. Veja a seção "Desempenho extremo e exploração de limites de armazenamento " para mais informações.

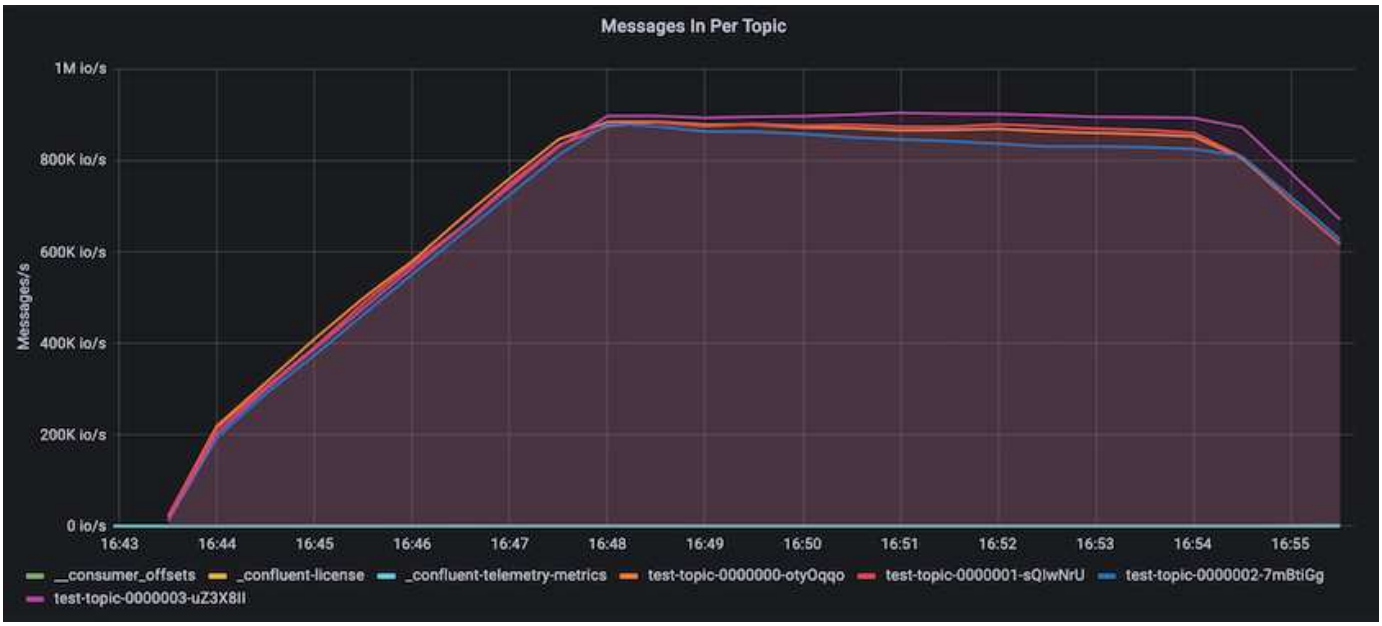
Desempenho em estado estacionário

Avaliamos o AFF A900 usando o OpenMessaging Benchmark para fornecer uma comparação semelhante à do Cloud Volumes ONTAP na AWS e do DAS na AWS. Todos os valores de desempenho representam a taxa de transferência do cluster Kafka no nível do produtor e do consumidor.

O desempenho em estado estável com o Confluent Kafka e o AFF A900 atingiu uma taxa de transferência média de mais de 3,4 GBps para produtores e consumidores. Isso representa mais de 3,4 milhões de mensagens no cluster Kafka. Ao visualizar a taxa de transferência sustentada em bytes por segundo para BrokerTopicMetrics, vemos o excelente desempenho de estado estável e o tráfego suportado pelo AFF A900.



Isso se alinha bem com a visão de mensagens entregues por tópico. O gráfico a seguir fornece uma análise por tópico. Na configuração testada, vimos quase 900 mil mensagens por tópico em quatro tópicos.

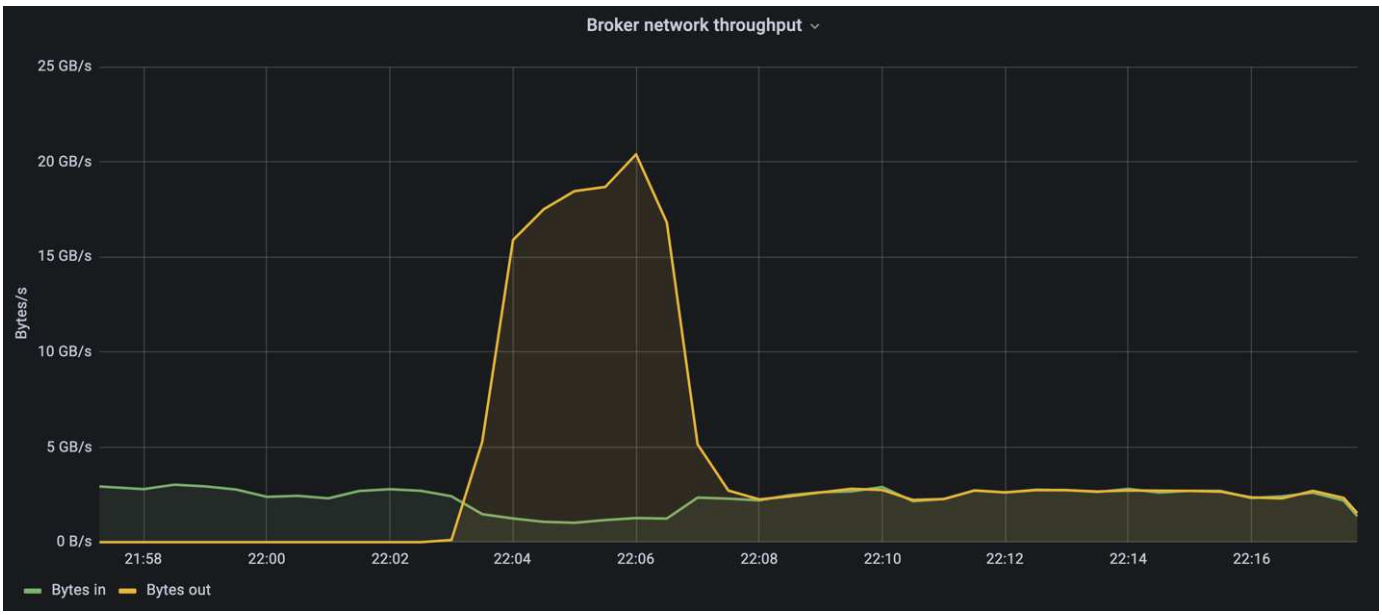


Desempenho extremo e exploração de limites de armazenamento

Para o AFF, também testamos com o OMB usando o recurso de backlog. O recurso de backlog pausa as assinaturas do consumidor enquanto um backlog de eventos é criado no cluster do Kafka. Durante esta fase, ocorre apenas o tráfego do produtor, o que gera eventos que são confirmados em logs. Isso emula mais de perto o processamento em lote ou os fluxos de trabalho de análise offline; nesses fluxos de trabalho, as assinaturas do consumidor são iniciadas e devem ler dados históricos que já foram removidos do cache do corretor.

Para entender as limitações de armazenamento na taxa de transferência do consumidor nesta configuração, medimos a fase somente do produtor para entender quanto tráfego de gravação o A900 poderia absorver. Veja a próxima seção "Orientação de dimensionamento " para entender como aproveitar esses dados.

Durante a parte exclusiva do produtor dessa medição, observamos um alto pico de rendimento que ultrapassou os limites de desempenho do A900 (quando outros recursos do corretor não estavam saturados, atendendo ao tráfego de produtores e consumidores).



Aumentamos o tamanho da mensagem para 16k para esta medição para limitar as sobrecargas por mensagem e maximizar a taxa de transferência de armazenamento para pontos de montagem NFS.

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

O cluster Confluent Kafka atingiu um pico de produtividade do produtor 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 …
```

Depois que o OMB concluiu o preenchimento do eventbacklog, o tráfego do consumidor foi reiniciado. Durante as medições com drenagem de backlog, observamos um pico de rendimento do consumidor de mais de 20 GBps em todos os tópicos. A taxa de transferência combinada para o volume NFS que armazena os dados de log OMB se aproximou de ~30 GBps.

Orientação de dimensionamento

A Amazon Web Services oferece uma "guia de tamanhos" para dimensionamento e escalonamento de clusters do Kafka.

Esse dimensionamento fornece uma fórmula útil para determinar os requisitos de taxa de transferência de armazenamento para seu cluster Kafka:

Para uma taxa de transferência agregada produzida no cluster de tcluster com um fator de replicação de r, a taxa de transferência recebida pelo armazenamento do broker é a seguinte:

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

Isso pode ser simplificado ainda mais:

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

Usar esta fórmula permite que você selecione a plataforma ONTAP apropriada para suas necessidades de nível ativo do Kafka.

A tabela a seguir explica a produtividade esperada do produtor para o A900 com diferentes fatores de replicação:

Fator de replicação	Taxa de transferência do produtor (GPps)
3 (medido)	3,4
2	5,1

Fator de replicação	Taxa de transferência do produtor (GPps)
1	10,2

Conclusão

A solução da NetApp para o problema bobo de renomeação fornece uma forma de armazenamento simples, barata e gerenciada centralmente para cargas de trabalho que antes eram incompatíveis com o NFS.

Esse novo paradigma permite que os clientes criem clusters Kafka mais gerenciáveis, mais fáceis de migrar e espelhar para fins de recuperação de desastres e proteção de dados. Também vimos que o NFS oferece benefícios adicionais, como menor utilização da CPU e um tempo de recuperação mais rápido, eficiência de armazenamento significativamente melhorada e melhor desempenho por meio do NetApp ONTAP.

Onde encontrar informações adicionais

Para saber mais sobre as informações descritas neste documento, revise os seguintes documentos e/ou sites:

- O que é Apache Kafka?

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

- O que é renomeação boba?

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

- ONATP é lido para aplicações de streaming.

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

- Documentação do produto NetApp

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

- O que é NFS?

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

- O que é reatribuição de partição do Kafka?

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

- O que é o OpenMessaging Benchmark?

  "https://openmessaging.cloud/"

- Como você migra um corretor Kafka?

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

- Como monitorar o corretor Kafka com o Prometheus?

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

- Plataforma gerenciada para Apache Kafka

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

- Suporte para Apache Kafka

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

- Serviços de consultoria para Apache Kafka

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

Informações sobre direitos autorais

Informações sobre marcas comerciais