NetApp NFS 스토리지를 사용하는 Apache Kafka 워크로드

NetApp Solutions

NetApp
April 20, 2024

This PDF was generated from https://docs.netapp.com/ko-kr/netapp-solutions/data-analytics/kafka-nfs-introduction.html on April 20, 2024. Always check docs.netapp.com for the latest.

목차

NetApp NFS 스토리지를 사용하는 Apache Kafka 워크로드 . . . 1
TR-4947: NetApp NFS 스토리지의 Apache Kafka 워크로드 - 기능 검증 및 성능 . . . 1
NFS에서 Kafka 워크로드로의 이름 바꾸기 문제를 해결하기 위한 NetApp 솔루션 . . . 2
기능 검증 - 이름 바꾸기 수정 바보 . . . 3
Kafka 워크로드를 위해 NetApp NFS를 선택해야 하는 이유 . . . 7
AWS의 성능 개요 및 검증 . . . 19
NetApp ONTAP용 AWS FSx의 성능 개요 및 검증 . . . 32
AFF A900 온프레미스 성능 개요 및 검증 . . . 40
결론 . . . 46
추가 정보를 찾을 수 있는 위치 . . . 46

NetApp NFS 스토리지를 사용하는 Apache Kafka 워크로드

TR-4947: NetApp NFS 스토리지의 Apache Kafka 워크로드 - 기능 검증 및 성능

Shantanu Chakole, Karthikeyan Nagalingam 및 NetApp의 Joe Scott입니다

Kafka는 많은 양의 메시지 데이터를 수용할 수 있는 강력한 큐가 있는 분산 게시 구독 메시징 시스템입니다. Kafka를 사용하면 응용 프로그램에서 매우 빠른 방법으로 주제에 데이터를 쓰고 읽을 수 있습니다. 내결함성 및 확장성 덕분에 Kafka는 많은 데이터 스트림을 매우 빠르게 수집하고 이동하는 안정적인 방법으로 빅 데이터 공간에서 자주 사용됩니다. 사용 사례에는 스트림 처리, 웹 사이트 활동 추적, 메트릭 수집 및 모니터링, 로그 집계, 실시간 분석 등이 포함됩니다.

NFS에서 Kafka의 정상 작동이 잘 되긴 하지만 "이름 바꾸기 바보 같은 소리" NFS에서 실행 중인 Kafka 클러스터의 크기 조정 또는 재분할 중에 문제가 발생하면 응용 프로그램이 중단됩니다. 로드 밸런싱 또는 유지 관리를 위해 Kafka 클러스터의 크기를 조정하거나 재분할해야 하기 때문에 이는 중요한 문제입니다. 추가 세부 정보를 찾을 수 있습니다 "여기".

이 문서에서는 다음 주제에 대해 설명합니다.

- 이름 바꾸기 문제 및 솔루션 유효성 검사가 어리석은 문제입니다
- CPU 활용률을 줄여 I/O 대기 시간을 줄입니다
- Kafka 브로커 복구 시간 단축
- 클라우드 및 사내 성능

Kafka 워크로드를 위해 NFS 스토리지를 사용하는 이유는 무엇입니까?

운영 애플리케이션의 Kafka 워크로드는 애플리케이션 간에 방대한 양의 데이터를 스트리밍할 수 있습니다. 이 데이터는 Kafka 클러스터의 Kafka 브로커 노드에 저장되어 저장됩니다. Kafka는 또한 가용성 및 병렬 처리도 잘 알려져 있으며, 이를 위해 여러 주제를 파티션으로 분할한 다음 클러스터 전체에 파티션을 복제합니다. 이는 결국 Kafka 클러스터를 통해 흐르는 엄청난 양의 데이터가 일반적으로 크기에 배가된다는 것을 의미합니다. 브로커들의 수가 매우 빠르고 쉽게 변경됨에 따라 NFS에서 데이터의 균형을 재조정합니다. 대규모 환경에서는 브로커 변경 횟수가 매우 많은 경우 DAS 전반에서 데이터 균형을 재조정하는 데 많은 시간이 걸리며, 대부분의 Kafka 환경에서는 브로커 수가 자주 변경됩니다.

그 밖의 이점은 다음과 같습니다.

- 성숙도. * NFS는 완성도 높은 프로토콜입니다. 즉, NFS를 구현, 보안 및 사용하는 대부분의 측면을 충분히 이해할 수 있습니다.
- * Open. * NFS는 개방형 프로토콜이며, 지속적인 개발은 인터넷 사양에 무료 개방형 네트워크 프로토콜로 문서화되어 있습니다.
- 비용 효율성 * NFS는 기존 네트워크 인프라를 사용하기 때문에 설정하기 쉬운 네트워크 파일 공유를 위한 저렴한 솔루션입니다.
- * 중앙 집중식 관리. * NFS의 중앙 집중식 관리로 개별 사용자 시스템에 소프트웨어 및 디스크 공간을 추가할 필요가

없습니다.

- * Distributed. * NFS는 분산 파일 시스템으로 사용할 수 있으므로 이동식 미디어 저장 장치의 필요성을 줄여줍니다.

Kafka 워크로드를 위해 NetApp을 선택해야 하는 이유

NetApp NFS 구현은 프로토콜의 골드 표준으로 간주되며 수많은 엔터프라이즈 NAS 환경에서 사용됩니다. 또한 NetApp의 신용도와 더불어 다음과 같은 이점도 제공합니다.

- 안정성 및 효율성
- 확장성 및 성능
- 고가용성(NetApp ONTAP 클러스터의 HA 파트너)
- 데이터 보호
  - * 재해 복구(NetApp SnapMirror). * 사이트가 다운되거나 다른 사이트에서 바로 시작하여 중단된 부분부터 계속 진행하려고 합니다.
  - 스토리지 시스템의 관리 효율성(NetApp OnCommand를 사용한 관리 및 관리).
  - * 로드 밸런싱. * 이 클러스터를 사용하면 다른 노드에 호스팅된 데이터 LIF에서 서로 다른 볼륨에 액세스할 수 있습니다.
  - * 무중단 운영 * LIF 또는 볼륨 이동은 NFS 클라이언트에 영향을 미치지 않습니다.

NFS에서 Kafka 워크로드로의 이름 바꾸기 문제를 해결하기 위한 NetApp 솔루션

Kafka는 기본 파일 시스템이 POSIX를 준수한다고 가정합니다(예: XFS 또는 ext4). Kafka 리소스 재조정은 애플리케이션이 파일을 계속 사용하는 동안 파일을 제거합니다. POSIX 호환 파일 시스템을 사용하면 연결을 끊을 수 있습니다. 그러나 파일에 대한 모든 참조가 제거된 후에만 파일을 제거합니다. 기본 파일 시스템이 네트워크에 연결되어 있는 경우 NFS 클라이언트는 링크 해제 통화를 차단하고 워크플로우를 관리합니다. 연결 해제 중인 파일에 대해 보류 중인 열기 때문에 NFS 클라이언트는 이름 바꾸기 요청을 NFS 서버로 보내고 연결되지 않은 파일의 마지막 닫기에서는 이름이 변경된 파일에 대해 제거 작업을 실행합니다. 이 동작은 일반적으로 NFS의 이름이 바보 같은 이름 바꾸기라고 하며 NFS 클라이언트가 조율합니다.

NFSv3 서버의 스토리지를 사용하는 Kafka 브로커는 이 동작으로 인해 문제가 발생합니다. 그러나 NFSv4.x 프로토콜에는 열려 있는 연결되지 않은 파일에 대해 서버가 책임을 지도록 하여 이 문제를 해결할 수 있는 기능이 있습니다. 이 선택적 기능을 지원하는 NFS 서버는 파일을 열 때 NFS 클라이언트에 소유권 기능을 전달합니다. 그런 다음 보류 중인 가 열려 있고 서버가 흐름을 관리할 수 있으면 NFS 클라이언트가 연결 해제 관리를 중단합니다. NFSv4 사양에서는 구축 지침을 제공하지만 지금까지 이 선택적 기능을 지원하는 알려진 NFS 서버 구현은 없었습니다.

NFS 서버와 NFS 클라이언트가 이름 바꾸기 문제를 해결하려면 다음 변경 사항이 필요합니다.

- * NFS 클라이언트 변경 사항(Linux). * 파일을 열 때 NFS 서버가 플래그를 응답하여 열린 파일의 연결 해제 처리 기능을 나타냅니다. NFS 클라이언트 측 변경 사항을 사용하면 NFS 서버가 해당 플래그가 있을 때 연결 해제를 처리할 수 있습니다. NetApp은 이러한 변경 사항으로 오픈 소스 Linux NFS 클라이언트를 업데이트했습니다. 업데이트된 NFS 클라이언트는 RHEL8.7 및 RHEL9.1에서 일반적으로 사용할 수 있습니다.
- * NFS 서버에 대한 변경. * NFS 서버가 열린 내용을 추적합니다. 이제 기존 열린 파일의 연결 해제는 POSIX 의미론과 일치하도록 서버에서 관리됩니다. 마지막 열기가 닫히면 NFS 서버가 파일의 실제 제거를 시작하여 이름이

바보스러운 이름 바꾸기 프로세스를 방지합니다. ONTAP NFS 서버는 최신 버전인 ONTAP 9.12.1에서 이 기능을 구현했습니다.

Kafka는 NFS 클라이언트 및 서버의 변경 사항을 통해 네트워크에 연결된 NFS 스토리지의 모든 이점을 안전하게 누릴 수 있습니다.

기능 검증 - 이름 바꾸기 수정 바보

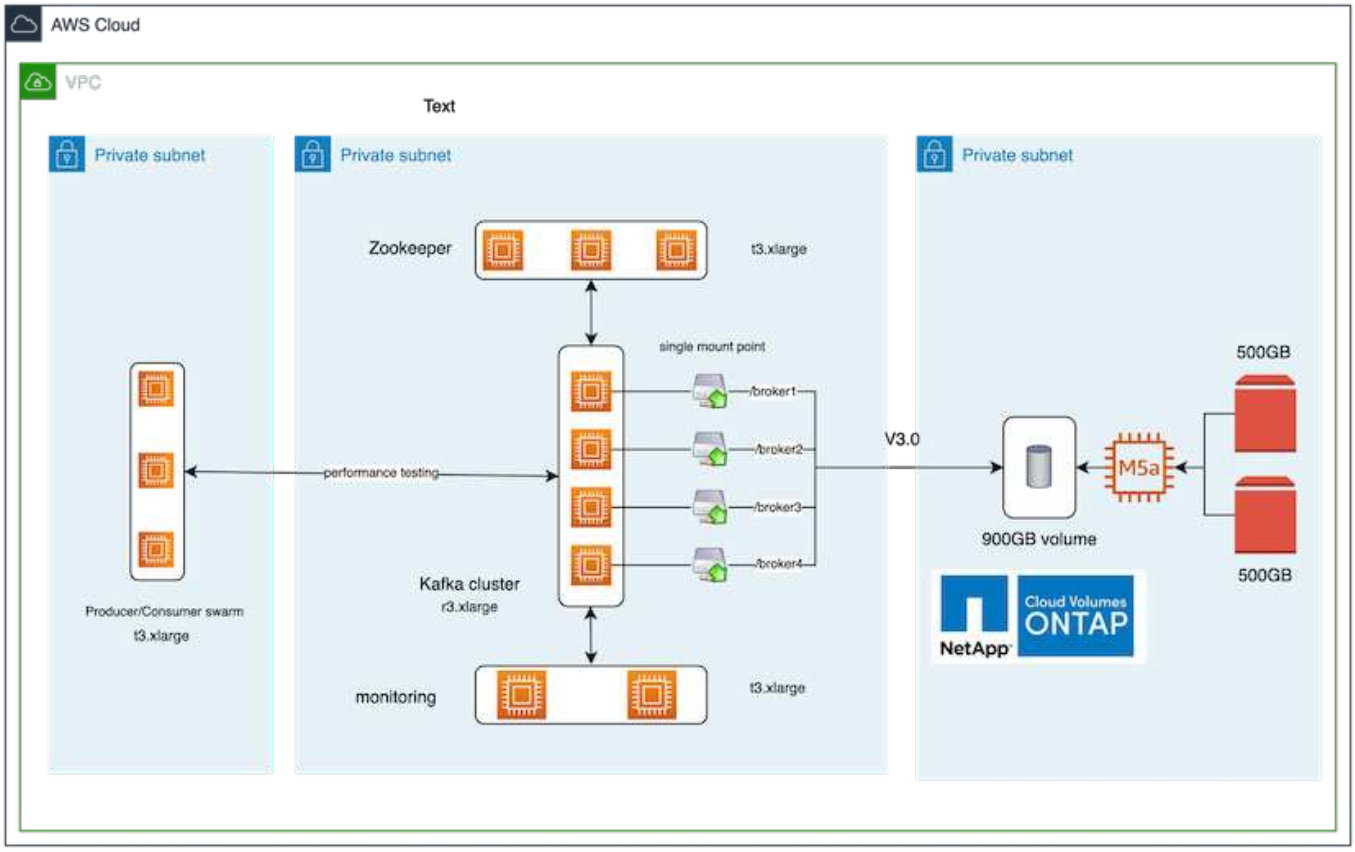
기능 검증을 위해 NFSv3 마운트를 사용하는 Kafka 클러스터가 파티션 재배포와 같은 Kafka 작업을 수행하지 않는 반면, 수정 사항이 있는 NFSv4에 마운트된 또 다른 클러스터는 중단 없이 동일한 작업을 수행할 수 있음을 보여주었습니다.

검증 설정

이 설정은 AWS에서 실행됩니다. 다음 표에는 검증에 사용된 다양한 플랫폼 구성 요소와 환경 구성이 나와 있습니다.

플랫폼 구성 요소	환경 구성
Confluent 플랫폼 버전 7.2.1	• 3 x zookeers – T3.xLarge • 브로커 서버 4대 – R3.xLarge • Grafana 1개 – T3.xLarge • 제어 센터 1개 – T3.xLarge • 프로듀서/소비자 3대
모든 노드의 운영 체제	RHEL8.7 이상
NetApp Cloud Volumes ONTAP 인스턴스	단일 노드 인스턴스 – M5.2xLarge

다음 그림에서는 이 솔루션의 아키텍처 구성을 보여 줍니다.



아키텍처 흐름

- * Compute. * 4노드 Kafka 클러스터와 3노드 zookeeper 앙상블이 전용 서버에서 실행되고 있었습니다.
- * 모니터링. * Prometheus-Grafana 조합에 두 개의 노드를 사용했습니다.
- * 워크로드. * 워크로드를 생성하는 데 별도의 3노드 클러스터를 사용하여 Kafka 클러스터에 생성하고 사용합니다.
- * 스토리지. * 이 인스턴스에는 500GB GP2 AWS-EBS 볼륨 2개가 연결된 단일 노드 NetApp Cloud Volumes ONTAP 인스턴스가 사용되었습니다. 그런 다음 LIF를 통해 이러한 볼륨을 단일 NFSv4.1 볼륨으로 Kafka 클러스터에 노출했습니다.

Kafka의 기본 속성은 모든 서버에 대해 선택되었습니다. 지퍼지기 백팔도 마찬가지였습니다.

테스트 방법

1. 업데이트 `-is-preserve-unlink-enabled true` 다음과 같이 Kafka 볼륨으로 이동합니다.

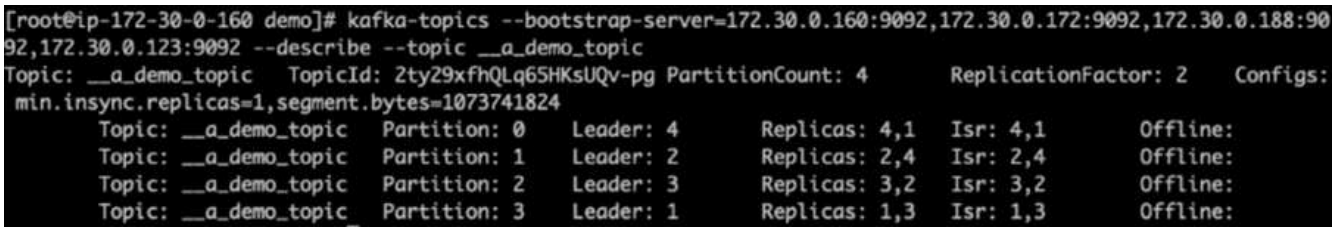
```
aws-shantanclastrecall-aws::*> volume create -vserver kafka_svm -volume
kafka_fg_vol01 -aggregate kafka_aggr -size 3500GB -state online -policy
kafka_policy -security-style unix -unix-permissions 0777 -junction-path
/kafka_fg_vol01 -type RW -is-preserve-unlink-enabled true
[Job 32] Job succeeded: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

2. 두 개의 유사한 Kafka 클러스터가 생성되었으며 다음과 같은 차이점이 있습니다.

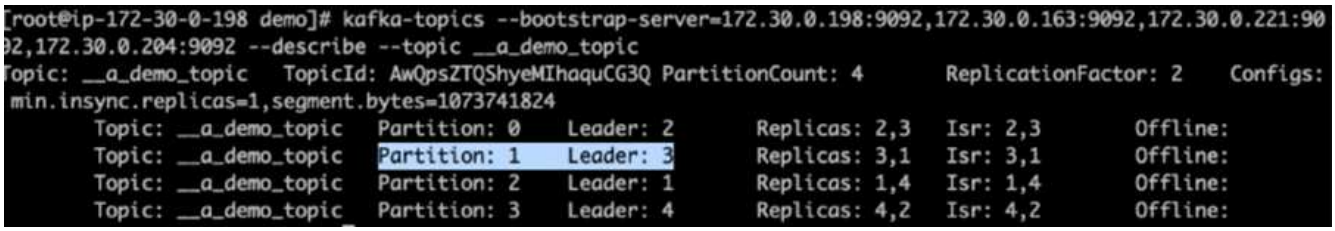
- * 클러스터 1. * 운영 지원 ONTAP 버전 9.12.1을 실행하는 백엔드 NFS v4.1 서버는 NetApp CVO 인스턴스에서 호스팅되었습니다. 브로커에 RHEL 8.7/RHEL 9.1이 설치되었습니다.
- * 클러스터 2. * 백엔드 NFS 서버는 수동으로 생성된 일반 Linux NFSv3 서버였습니다.

3. 두 Kafka 클러스터 모두에서 데모 주제가 생성되었습니다.

   클러스터 1:



   클러스터 2:



4. 두 클러스터에 대해 새로 생성된 이러한 항목에 데이터가 로드되었습니다. 이 작업은 기본 Kafka 패키지에 포함된 프로듀서-perf-test 툴킷을 사용하여 수행되었습니다.

```
./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. telnet을 사용하여 각 클러스터에 대해 broker-1에 대한 상태 점검을 수행했습니다.

   - 텔넷 `172.30.0.160 9092`
   - 텔넷 `172.30.0.198 9092`

     두 클러스터 모두에서 브로커의 상태 점검이 성공적으로 완료된 경우 다음 스크린샷에 나와 있습니다.

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

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

6. NFSv3 스토리지 볼륨을 사용하는 Kafka 클러스터가 중단되는 장애 조건을 트리거하기 위해 두 클러스터에서 파티션 재할당 프로세스를 시작했습니다. 파티션 재할당이 을(를) 사용하여 수행되었습니다 `kafka-reassign-partitions.sh`. 자세한 프로세스는 다음과 같습니다.

   a. Kafka 클러스터의 주제에 대한 파티션을 재할당하기 위해 제안된 재할당 구성 JSON을 생성했습니다(두 클러스터에 대해 수행됨).

   ```
   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. 생성된 재할당 JSON이 에 저장되었습니다 `/tmp/reassignment- file.json`.

   c. 실제 파티션 재할당 프로세스는 다음 명령을 통해 트리거되었습니다.

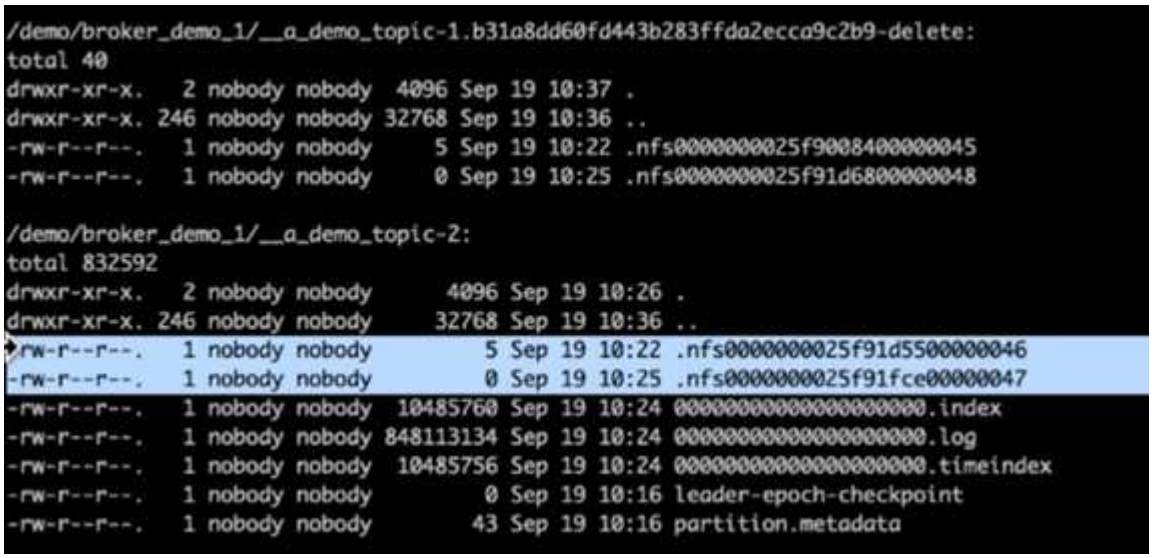
   ```
   kafka-reassign-partitions --bootstrap
   -server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:9092,172.30.
   0.204:9092 --reassignment-json-file /tmp/reassignment-file.json
   -execute
   ```

7. 재할당이 완료되고 몇 분 후에 브로커에 대한 또 다른 상태 점검 결과, NFSv3 스토리지 볼륨을 사용하는 클러스터가 이름이 바보 같은 문제로 실행되고 충돌이 발생했던 반면, NetApp ONTAP NFSv4.1 스토리지 볼륨을 사용하는 클러스터 1은 수정 작업을 중단 없이 계속했습니다.

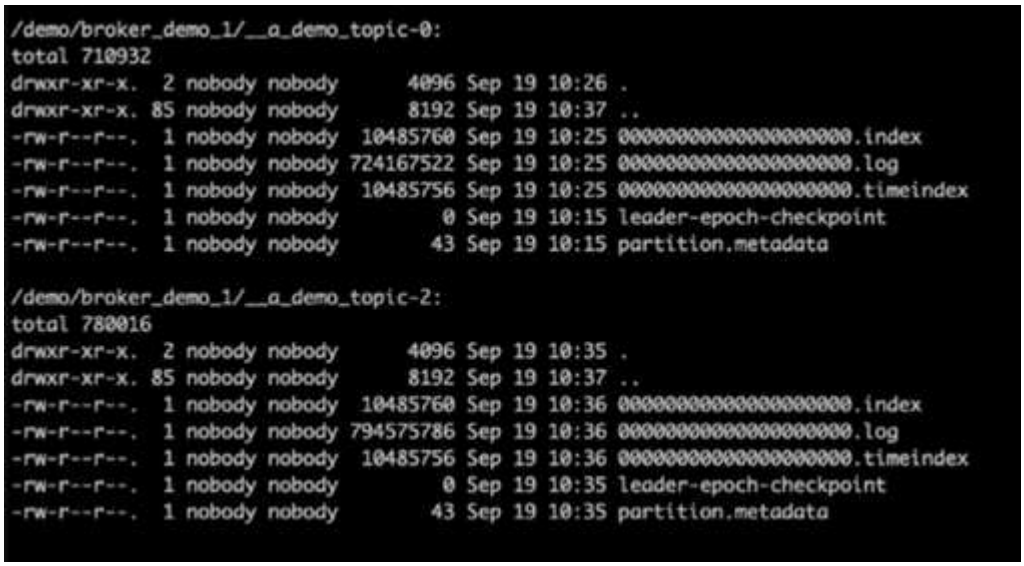
```
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.160 9092
Trying 172.30.0.160...
Connected to 172.30.0.160.
Escape character is '^]'.
^[

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-1이 활성 상태입니다.
- Cluster2-broker-1이 작동하지 않습니다.

8. Kafka 로그 디렉토리를 확인했을 때 수정하여 NetApp ONTAP NFSv4.1 스토리지 볼륨을 사용하는 클러스터 1에서 파티션을 클린 할당했다는 것이 분명했지만, 일반 NFSv3 스토리지를 사용하는 클러스터 2에서는 이름 변경 문제가 우스꽝스러운 이유로 발생하지 않아 충돌이 발생하지 않았습니다. 다음 그림에서는 클러스터 2의 파티션 재조정을 보여 주며, NFSv3 스토리지에서 이름 바꾸기 문제가 우스꽝스러운 것으로 나타났습니다.



다음 그림은 NetApp NFSv4.1 스토리지를 사용하여 클러스터 1의 파티션을 완전히 재조정하는 방법을 보여줍니다.



Kafka 워크로드를 위해 NetApp NFS를 선택해야 하는 이유

이제 Kafka를 사용하는 NFS 스토리지에 이름 바꾸기 관련 우스꽝스러운 문제를 해결하는 솔루션이 존재하므로 Kafka 워크로드를 위해 NetApp ONTAP 스토리지를 활용하는 강력한 구축을 구현할 수 있습니다. 이는 운영 오버헤드를 크게 줄일 뿐만 아니라 Kafka 클러스터에 다음과 같은 이점을 제공합니다.

- * Kafka 브로커의 CPU 사용률 감소 * disaggregated NetApp ONTAP 스토리지를 사용하면 디스크 I/O 작업이 브로커에서 분리되므로 CPU 설치 공간이 감소합니다.
- * 더 빠른 브로커 복구 시간. * 더 빠른 브로커 복구 시간 * 은 Kafka 브로커 노드 간에 분할된 NetApp ONTAP 스토리지가 공유되므로 데이터 재구축이 없는 기존 Kafka 구축에 비해 훨씬 빠른 시간 내에 모든 지점에서 새로운 컴퓨팅 인스턴스가 불량 브로커를 대체할 수 있습니다.
- * 스토리지 효율성. * 이제 애플리케이션의 스토리지 계층이 NetApp ONTAP를 통해 프로비저닝됨에 따라 고객은 인라인 데이터 압축, 중복제거, 컴팩션과 같은 ONTAP의 스토리지 효율성이 제공하는 모든 이점을 활용할 수 있습니다.

이 이점에 대해서는 이 섹션에서 자세히 설명하는 테스트 사례에서 테스트 및 검증을 거쳤습니다.

Kafka 브로커의 CPU 사용률 감소

두 개의 정자로 된 Kafka 클러스터에서 유사한 워크로드를 실행했을 때 전체 CPU 사용률이 DAS보다 낮다는 사실을 발견했습니다. 이러한 워크로드는 기술 사양에서 동일하지만 스토리지 기술에서는 달랐습니다. Kafka 클러스터가 ONTAP 스토리지를 사용하는 경우 전체 CPU 사용률이 낮아질 뿐만 아니라 CPU 활용률이 DAS 기반 Kafka 클러스터보다 더 부드럽게 증가함을 보여 주었습니다.

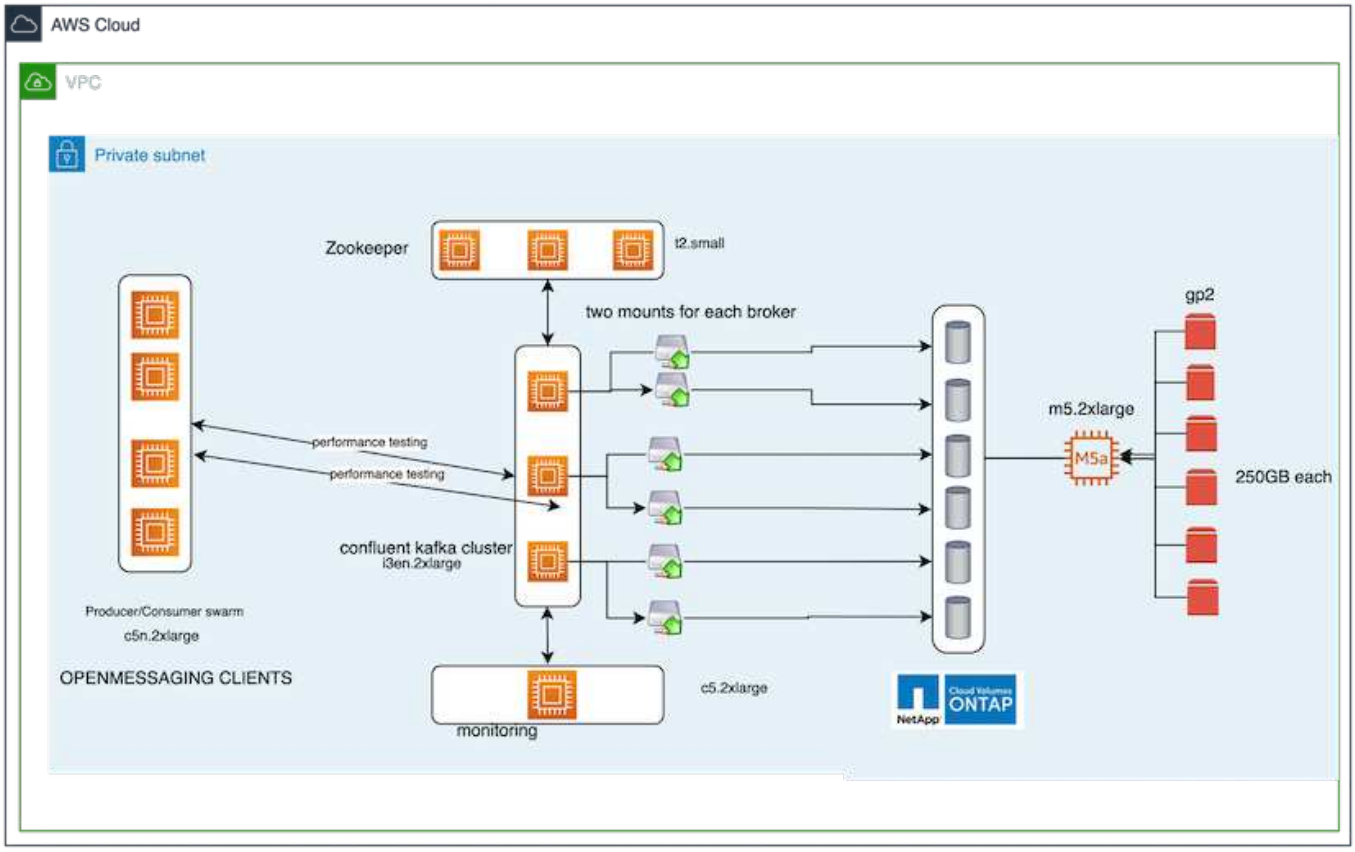
아키텍처 설정

다음 표에는 CPU 사용률 감소를 보여 주는 데 사용되는 환경 구성이 나와 있습니다.

플랫폼 구성 요소	환경 구성
Kafka 3.2.3 벤치마킹 도구: OpenMessaging	• 3 x zookeers – T2.small • 브로커 서버 3대 – i3en.2xLarge • Grafana 1개 – c5n.2xLarge • 프로듀서/소비자 4대 — c5n.2xLarge
모든 노드의 운영 체제	RHEL 8.7 이상
NetApp Cloud Volumes ONTAP 인스턴스	단일 노드 인스턴스 – M5.2xLarge

벤치마킹 툴

이 테스트 사례에 사용된 벤치마킹 툴은 입니다 "OpenMessaging" 프레임워크: OpenMessaging은 공급업체에 구애받지 않고 언어와 독립적입니다. 금융, 전자 상거래, IoT 및 빅 데이터에 대한 업계 지침을 제공하며 이기종 시스템 및 플랫폼에서 메시징 및 스트리밍 애플리케이션을 개발하는 데 도움이 됩니다. 다음 그림에서는 OpenMessaging 클라이언트와 Kafka 클러스터의 상호 작용을 보여 줍니다.



- * Compute. * 3노드 Kafka 클러스터와 전용 서버에서 실행되는 3노드 zookeeper 앙상블이 사용되었습니다. 각 브로커는 전용 LIF를 통해 NetApp CVO 인스턴스의 단일 볼륨에 대한 NFSv4.1 마운트 지점을 2개 가지고 있었습니다.
- * 모니터링. * Prometheus-Grafana 조합에 두 개의 노드를 사용했습니다. 워크로드를 생성하기 위해 Kafka 클러스터에 대해 생성하고 사용할 수 있는 별도의 3노드 클러스터가 있습니다.
- * 스토리지. * 이 인스턴스에 마운트된 250GB GP2 AWS-EBS 볼륨 6개가 포함된 단일 노드 NetApp Cloud Volumes ONTAP 인스턴스를 사용했습니다. 그런 다음 전용 LIF를 통해 이러한 볼륨을 6개의 NFSv4.1 볼륨으로 Kafka 클러스터에 노출했습니다.
- * 구성. * 이 테스트 사례에서 구성 가능한 두 가지 요소는 Kafka 브로커와 OpenMessaging 워크로드였습니다.
  - * 브로커 구성 * Kafka 브로커에 대해 다음 사양이 선택되었습니다. 아래에 강조 표시된 것처럼 모든 측정에 3의 복제 계수를 사용했습니다.

```
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
```

- * OMB(OpenMessaging 벤치마크) 워크로드 구성 * 다음 사양이 제공됩니다. 아래에 강조 표시된 목표 생산자 비율을 지정했습니다.

```
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
```

테스트 방법

1. 서로 유사한 클러스터 두 개가 생성되었으며, 각 클러스터마다 자체 벤치마킹 클러스터 군들이 있습니다.
    - * 클러스터 1. * NFS 기반 Kafka 클러스터
    - * 클러스터 2. * DAS 기반 Kafka 클러스터
2. OpenMessaging 명령을 사용하여 각 클러스터에서 유사한 워크로드가 트리거되었습니다.

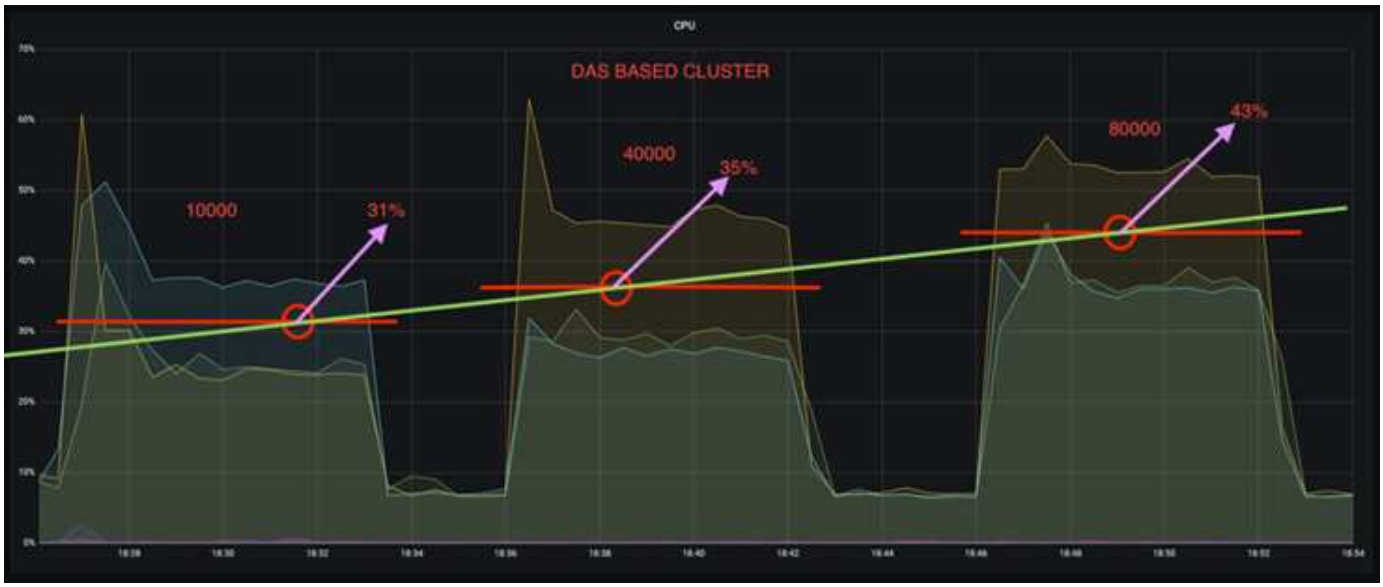
```
sudo bin/benchmark --drivers driver-kafka/kafka-group-all.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

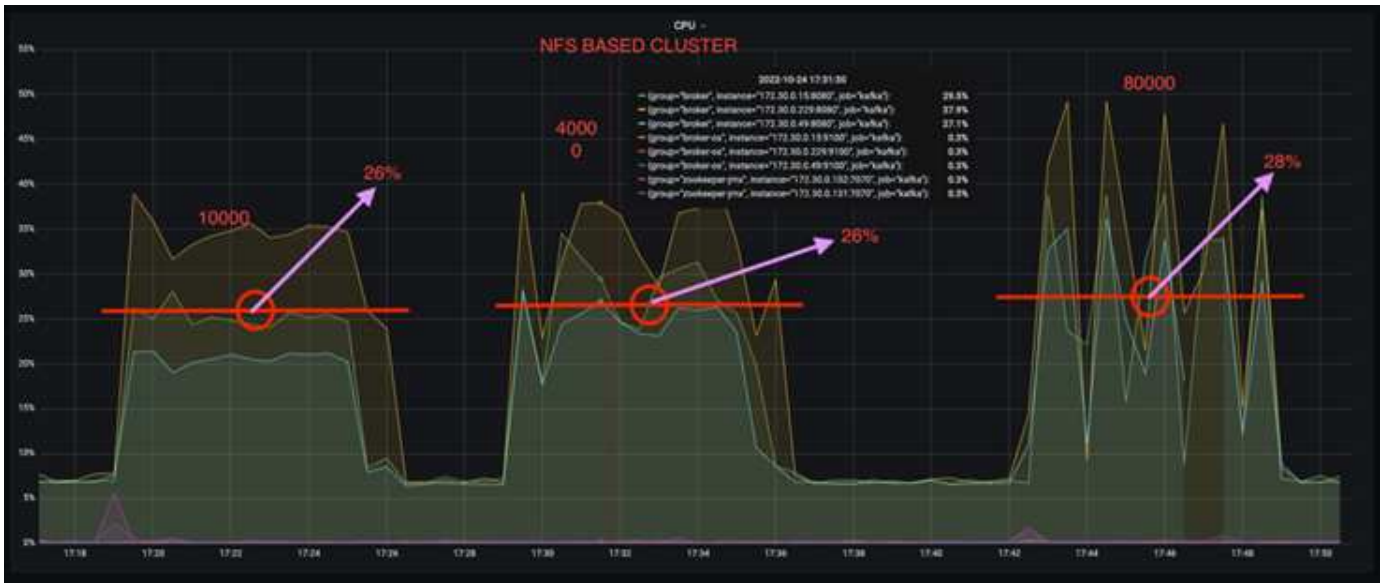
3. 생산율 구성이 4번 반복되어 Grafana로 기록되었으며, CPU 활용률이 기록되었습니다. 생산률이 다음 수준으로 설정되었습니다.
    - 10,000
    - 40,000개
    - 80,000
    - 100,000

관찰

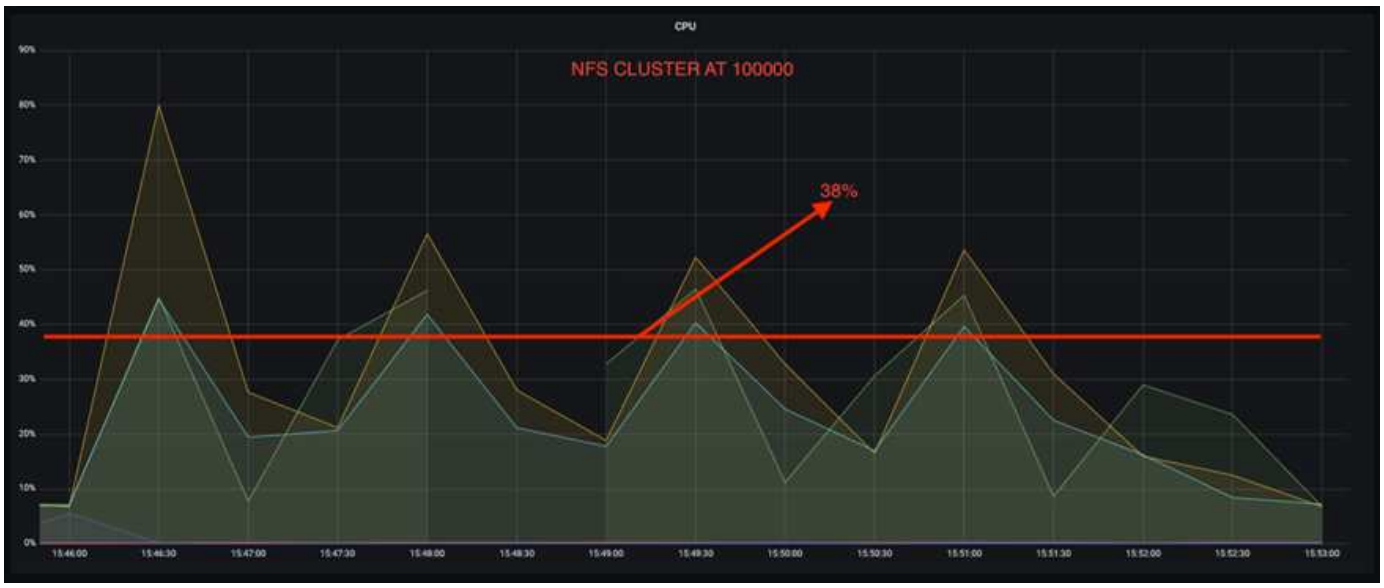
NetApp NFS 스토리지와 Kafka를 함께 사용하면 다음과 같은 두 가지 주요 이점을 얻을 수 있습니다.

- * CPU 사용률을 거의 1/3까지 줄일 수 있습니다. * 유사한 워크로드에서의 전체 CPU 사용은 NFS의 경우 DAS SSD에 비해 더 낮았습니다. 낮은 생산률의 경우 5%에서 더 높은 생산률의 경우 32%까지 절약되었습니다.
- * 더 높은 생산률에서 CPU 사용률 편차가 3배 감소했습니다. * 예상대로 생산률이 증가할수록 CPU 활용률이 증가하기 때문에 상승률이 나타났습니다. 그러나 DAS를 사용하는 Kafka 브로커의 CPU 활용률은 낮은 생산률의 31%에서 높은 생산률의 70%로 39% 증가했습니다. 하지만 NFS 스토리지 백엔드를 사용할 경우 CPU 활용률이 26%에서 38%로 12% 증가했습니다.





또한, 100,000 메시지에서 DAS는 NFS 클러스터보다 CPU 사용률이 더 높습니다.



브로커 복구 시간 단축

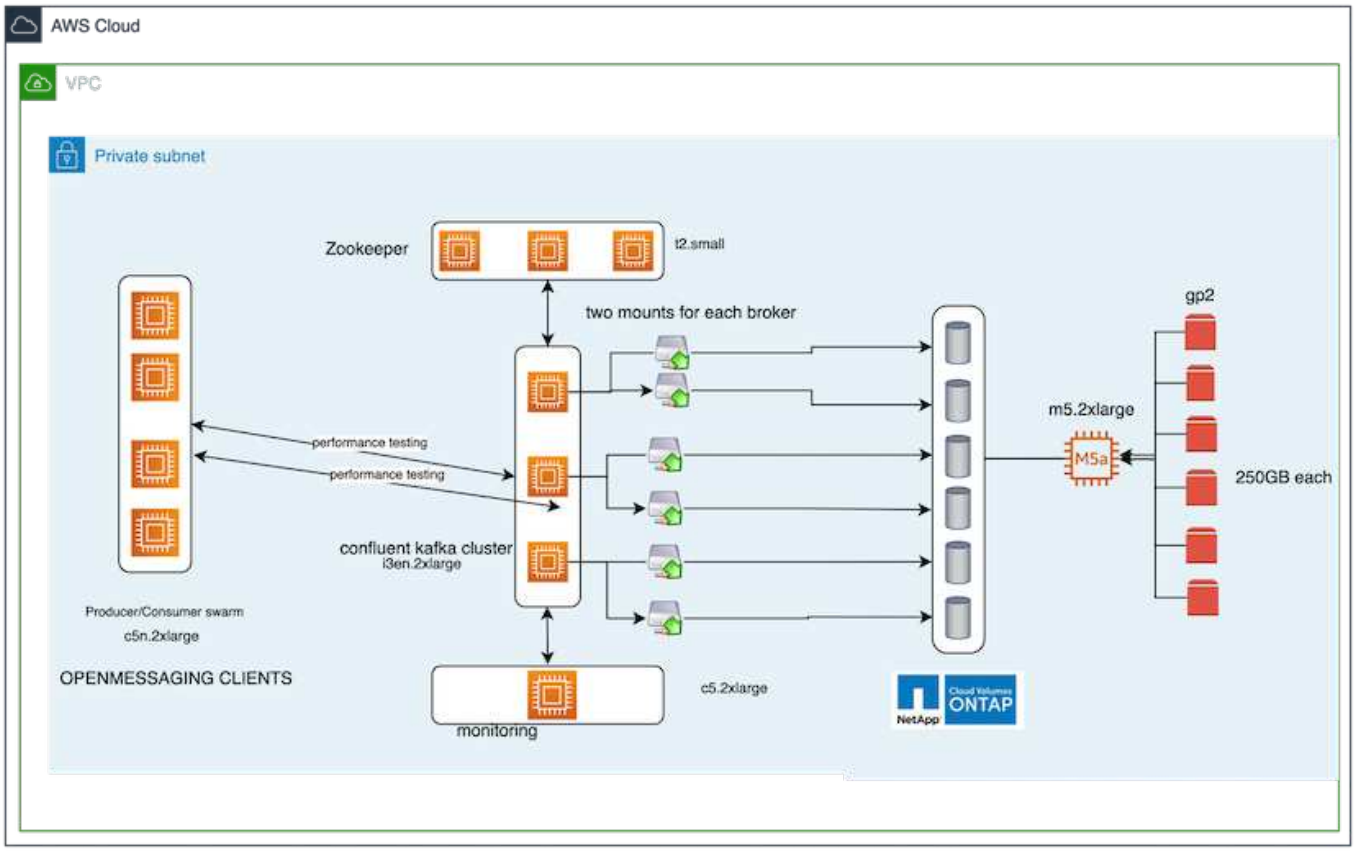
Kafka 브로커가 공유 NetApp NFS 스토리지를 사용할 때 더 빨리 복구한다는 사실을 알게 되었습니다. Kafka 클러스터에서 브로커가 충돌하는 경우 이 브로커는 동일한 브로커 ID를 가진 정상 브로커로 교체될 수 있습니다. 이 테스트 사례를 수행한 결과, DAS 기반 Kafka 클러스터의 경우 클러스터가 새로 추가된 정상적인 브로커로 데이터를 재구축하므로 시간이 오래 걸립니다. NetApp NFS 기반 Kafka 클러스터의 경우 대체 브로커가 이전 로그 디렉토리에서 데이터를 계속 읽고 복구 속도를 훨씬 높일 수 있습니다.

아키텍처 설정

다음 표에는 NAS를 사용하는 Kafka 클러스터의 환경 구성이 나와 있습니다.

플랫폼 구성 요소	환경 구성
Kafka 3.2.3	• 3 x zookeers – T2.small • 브로커 서버 3대 – i3en.2xLarge • Grafana 1개 – c5n.2xLarge • 생산자/소비자 4대 — c5n.2xLarge • 백업 Kafka 노드 1개 – i3en.2xLarge
모든 노드의 운영 체제	RHEL8.7 이상
NetApp Cloud Volumes ONTAP 인스턴스	단일 노드 인스턴스 – M5.2xLarge

다음 그림에서는 NAS 기반 Kafka 클러스터의 아키텍처를 보여 줍니다.



- * Compute. * 전용 서버에서 3노드 zookeeper 앙상블이 실행되는 3노드 Kafka 클러스터. 각 브로커는 전용 LIF를 통해 NetApp CVO 인스턴스의 단일 볼륨에 대한 NFS 마운트 지점을 2개 가집니다.
- * 모니터링. * Prometheus-Grafana 조합에 대한 두 개의 노드. 워크로드를 생성하는 경우 이 Kafka 클러스터를 생성하고 사용할 수 있는 별도의 3노드 클러스터를 사용합니다.
- * 스토리지. * 250GB GP2 AWS-EBS 볼륨 6개가 인스턴스에 마운트된 단일 노드 NetApp Cloud Volumes ONTAP 인스턴스. 그런 다음 전용 LIF를 통해 Kafka 클러스터에 6개의 NFS 볼륨으로 노출됩니다.
- * 브로커 구성. * 이 테스트 사례에서 구성 가능한 한 가지 요소는 Kafka 브로커입니다. Kafka 브로커에 대해 다음과 같은 사양이 선택되었습니다. 를 클릭합니다 `replica.lag.time.mx.ms` ISR 목록에서 특정 노드를 얼마나 빨리 제외할지 결정하므로 값이 높게 설정됩니다. 불량 노드와 정상 노드 간에 전환할 때 ISR 목록에서 브로커 ID를 제외하지 않도록 합니다.

```
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
```

테스트 방법

1. 두 개의 유사한 클러스터가 생성되었습니다.
   - EC2 기반 confluent 클러스터
   - NetApp NFS 기반 confluent 클러스터
2. 하나의 대기 Kafka 노드가 원래 Kafka 클러스터의 노드와 동일한 구성으로 생성되었습니다.
3. 각 클러스터마다 샘플 주제가 생성되었으며 각 브로커에 약 110GB의 데이터가 채워졌습니다.
   - * EC2 기반 클러스터 * Kafka 브로커 데이터 디렉토리가 에 매핑되어 있습니다 `/mnt/data-2` (다음 그림에서 cluster1 의 Broker-1 [왼쪽 터미널]).
   - * NetApp NFS 기반 클러스터 * Kafka 브로커 데이터 디렉토리가 NFS 지점에 마운트됩니다 `/mnt/data` (다음 그림에서 클러스터2의 브로커-1[오른쪽 터미널])

4. 각 클러스터에서 브로커-1이 종료되어 실패한 브로커 복구 프로세스가 트리거되었습니다.

5. 브로커가 종료된 후 브로커 IP 주소가 스탠바이 브로커의 보조 IP로 할당되었습니다. Kafka 클러스터의 브로커가 다음과 같이 식별되기 때문에 이 작업이 필요했습니다.
   - * IP 주소. * 장애가 발생한 브로커 IP를 대기 브로커에 재할당하여 지정합니다.
   - * 브로커 ID. * 이것은 대기 브로커에서 구성되었습니다 `server.properties`.
6. IP 할당 시 Kafka 서비스는 대기 브로커에서 시작되었습니다.
7. 잠시 후 서버 로그를 가져와 클러스터의 교체 노드에 데이터를 구축하는 데 걸린 시간을 확인합니다.

관찰

Kafka 브로커 복구는 거의 9배 빨라졌습니다. NetApp NFS 공유 스토리지를 사용할 경우 실패한 브로커 노드를 복구하는 데 걸리는 시간이 Kafka 클러스터에서 DAS SSD를 사용하는 경우에 비해 훨씬 빠른 것으로 확인되었습니다. 1TB의 항목 데이터에서 DAS 기반 클러스터의 복구 시간은 NetApp-NFS 기반 Kafka 클러스터의 복구 시간은 5분 미만이었습니다.

EC2 기반 클러스터는 새로운 브로커 노드에서 110GB의 데이터를 재구축하는 데 10분이 걸렸지만, NFS 기반 클러스터는 3분 만에 복구를 완료했습니다. 로그에서 EC2의 파티션에 대한 소비자 오프셋이 0인 반면 NFS 클러스터에서는 이전 브로커로부터 소비자 오프셋이 선택되었다는 것을 확인했습니다.

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

DAS 기반 클러스터

1. 백업 노드는 08:55:53,730에 시작되었습니다.

```
[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. 데이터 리빌딩 프로세스는 09:05:24,860으로 끝났습니다. 110GB의 데이터를 처리하는 데 약 10분이 소요됩니다.

```
[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)
```

NFS 기반 클러스터

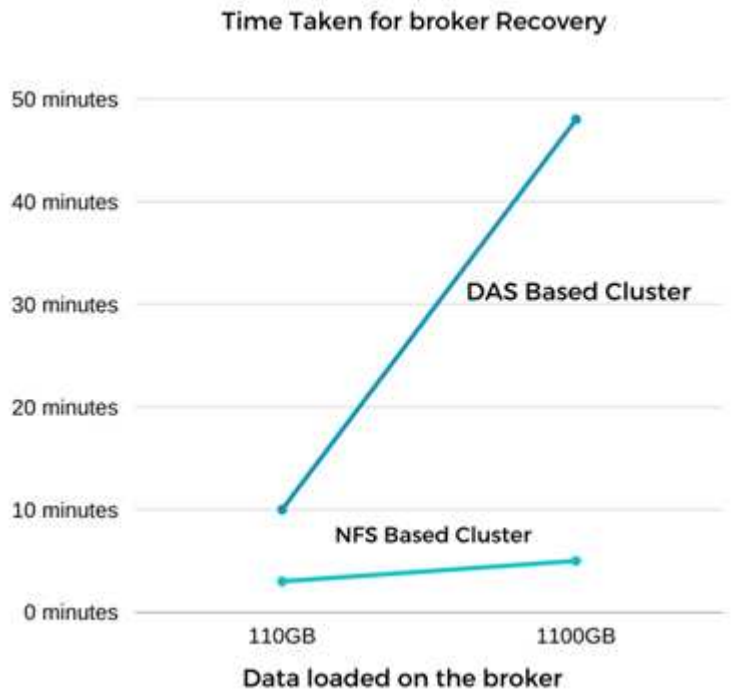
1. 백업 노드는 09:39:17,213에 시작되었습니다. 시작 로그 항목이 아래에 강조 표시됩니다.

```
[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. 데이터 리빌드 프로세스는 09:42:29,115로 종료되었습니다. 110GB의 데이터를 처리하는 데 약 3분이 소요됩니다.

```
[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)
```

이 테스트는 약 1TB 데이터를 포함하는 브로커에 대해 반복되었으며 DAS의 경우 약 48분, NFS의 경우 약 3분이 소요됩니다. 결과는 다음 그래프에 나와 있습니다.



스토리지 효율성

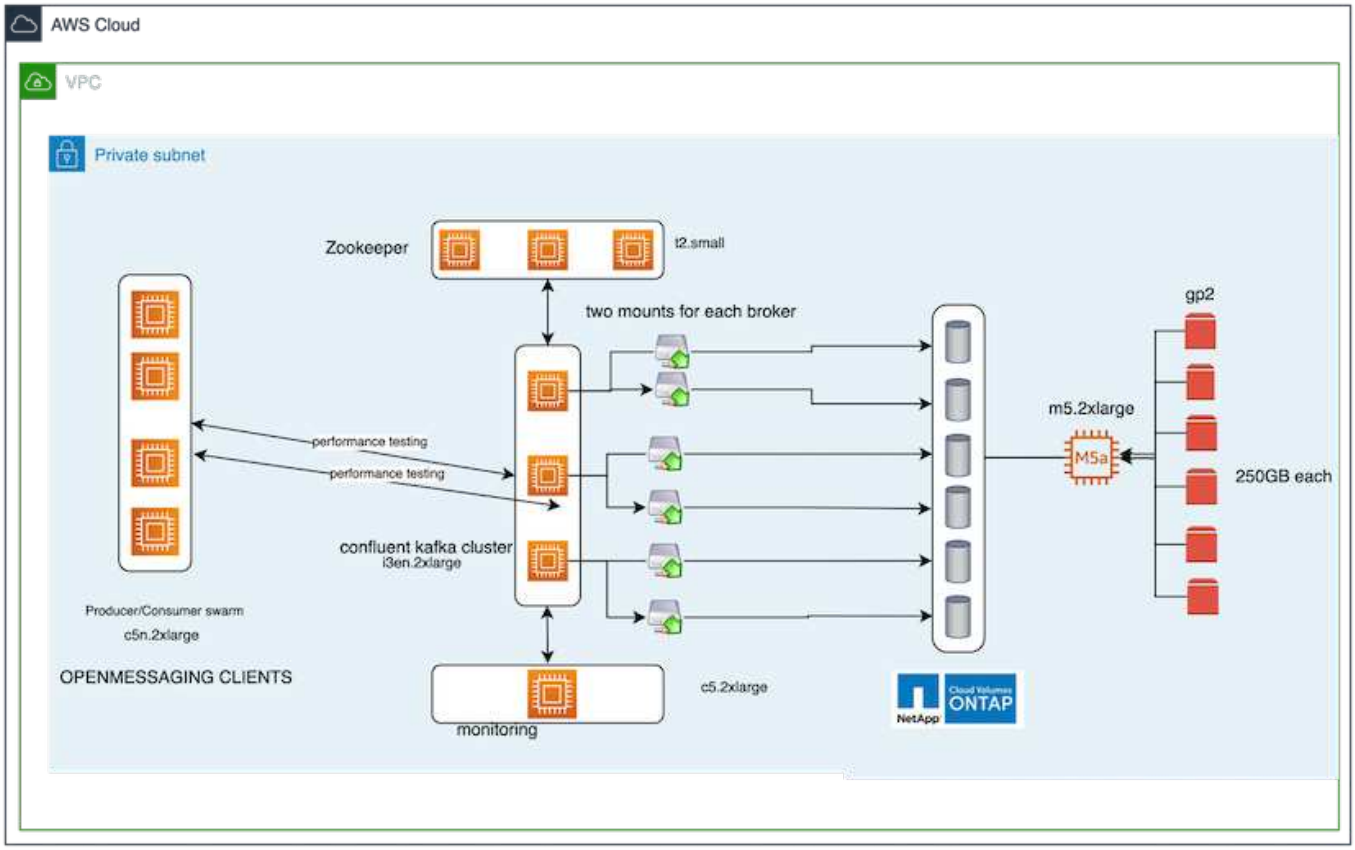
Kafka 클러스터의 스토리지 계층은 NetApp ONTAP를 통해 프로비저닝되었기 때문에 ONTAP의 모든 스토리지 효율성 기능을 얻었습니다. 이 테스트는 Cloud Volumes ONTAP에서 프로비저닝된 NFS 스토리지가 있는 Kafka 클러스터에 상당한 양의 데이터를 생성하여 테스트했습니다. ONTAP 기능 덕분에 공간을 상당히 줄일 수 있다는 것을 알 수 있었습니다.

아키텍처 설정

다음 표에는 NAS를 사용하는 Kafka 클러스터의 환경 구성이 나와 있습니다.

플랫폼 구성 요소	환경 구성
Kafka 3.2.3	• 3 x zookeers – T2.small • 브로커 서버 3대 – i3en.2xLarge • Grafana 1개 – c5n.2xLarge • 생산자/소비자 4대 — c5n.2xLarge *
모든 노드의 운영 체제	RHEL8.7 이상
NetApp Cloud Volumes ONTAP 인스턴스	단일 노드 인스턴스 – M5.2xLarge

다음 그림에서는 NAS 기반 Kafka 클러스터의 아키텍처를 보여 줍니다.



- * Compute. * 3노드 Kafka 클러스터와 전용 서버에서 실행되는 3노드 zookeeper 앙상블이 사용되었습니다. 각 브로커는 전용 LIF를 통해 NetApp CVO 인스턴스의 단일 볼륨에 대한 NFS 마운트 지점을 2개 가지고 있었습니다.
- * 모니터링. * Prometheus-Grafana 조합에 두 개의 노드를 사용했습니다. 워크로드를 생성하는데 이 Kafka 클러스터를 생성하고 사용할 수 있는 별도의 3노드 클러스터를 사용했습니다.
- * 스토리지. * 이 인스턴스에 마운트된 250GB GP2 AWS-EBS 볼륨 6개가 포함된 단일 노드 NetApp Cloud Volumes ONTAP 인스턴스를 사용했습니다. 그런 다음 전용 LIF를 통해 이 볼륨을 6개의 NFS 볼륨으로 Kafka 클러스터에 노출했습니다.
- * 구성. * 이 테스트 사례에서 구성 가능한 요소는 Kafka 브로커였습니다.

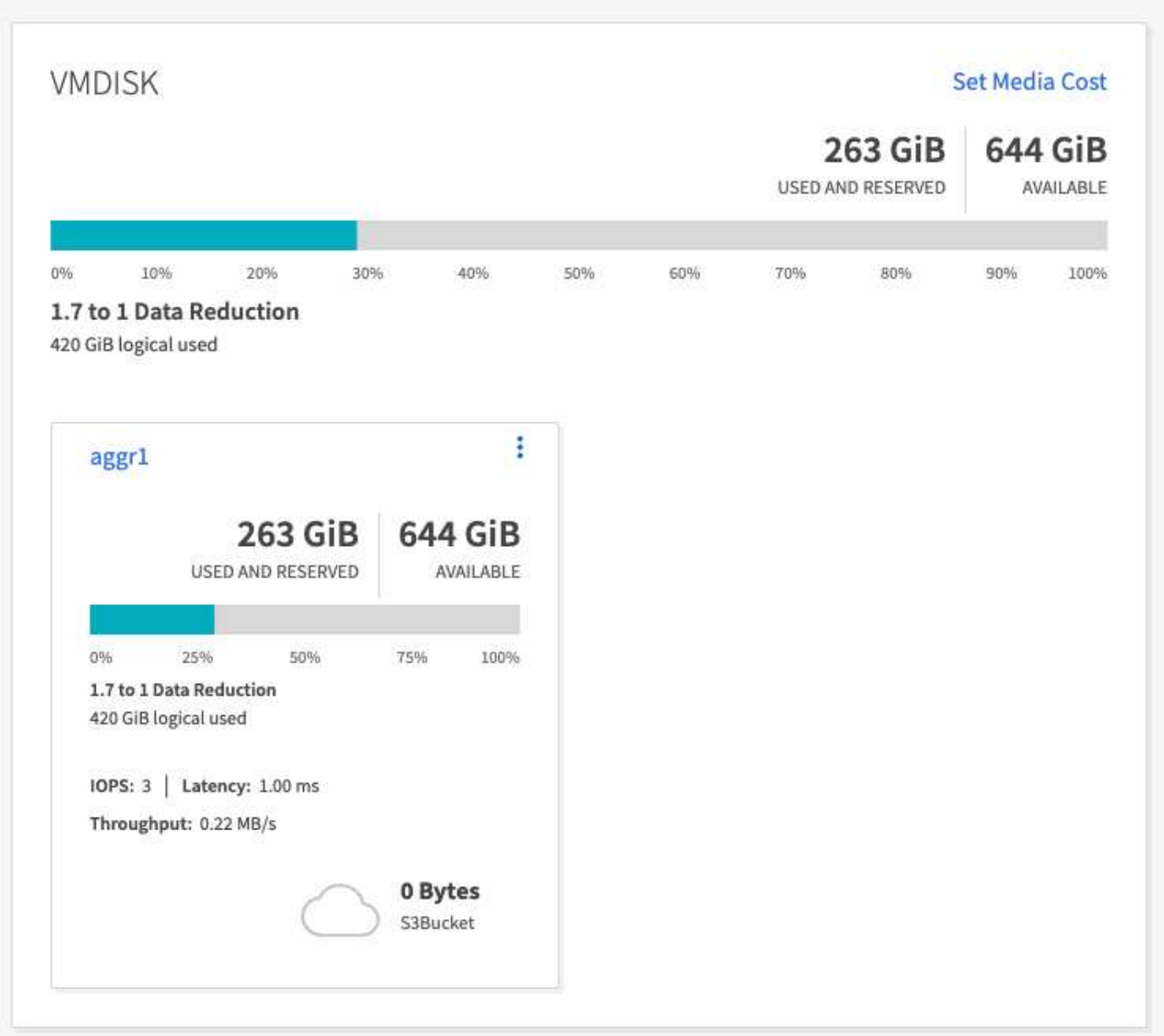
생산자 측의 압축 기능이 꺼지므로 생산자가 높은 처리량을 생성할 수 있습니다. 대신 컴퓨팅 계층에서 스토리지 효율성을 처리했습니다.

테스트 방법

1. Kafka 클러스터는 위에서 언급한 사양을 바탕으로 프로비저닝되었습니다.
2. 클러스터에서는 OpenMessaging 벤치마킹 도구를 사용하여 약 350GB의 데이터가 생성되었습니다.
3. 워크로드가 완료된 후 ONTAP System Manager 및 CLI를 사용하여 스토리지 효율성 통계가 수집됩니다.

관찰

OMB 툴을 사용하여 생성한 데이터의 경우, 스토리지 효율성 비율이 1.70:1인 33%까지 절약되었습니다. 다음 그림에서 볼 수 있듯이, 생성된 데이터에 사용된 논리적 공간은 420.3GB였으며 데이터를 저장하는 데 사용된 물리적 공간은 281.7GB였습니다.



```
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
```

AWS의 성능 개요 및 검증

NetApp NFS에 스토리지 계층이 마운트된 Kafka 클러스터가 AWS 클라우드의 성능을 벤치마킹했습니다. 벤치마킹 예는 다음 섹션에 설명되어 있습니다.

AWS 클라우드의 Kafka 및 NetApp Cloud Volumes ONTAP(고가용성 쌍 및 단일 노드)

NetApp Cloud Volumes ONTAP(HA 쌍)를 사용한 Kafka 클러스터는 AWS 클라우드의 성능을 벤치마킹했습니다. 이 벤치마킹은 다음 섹션에서 설명합니다.

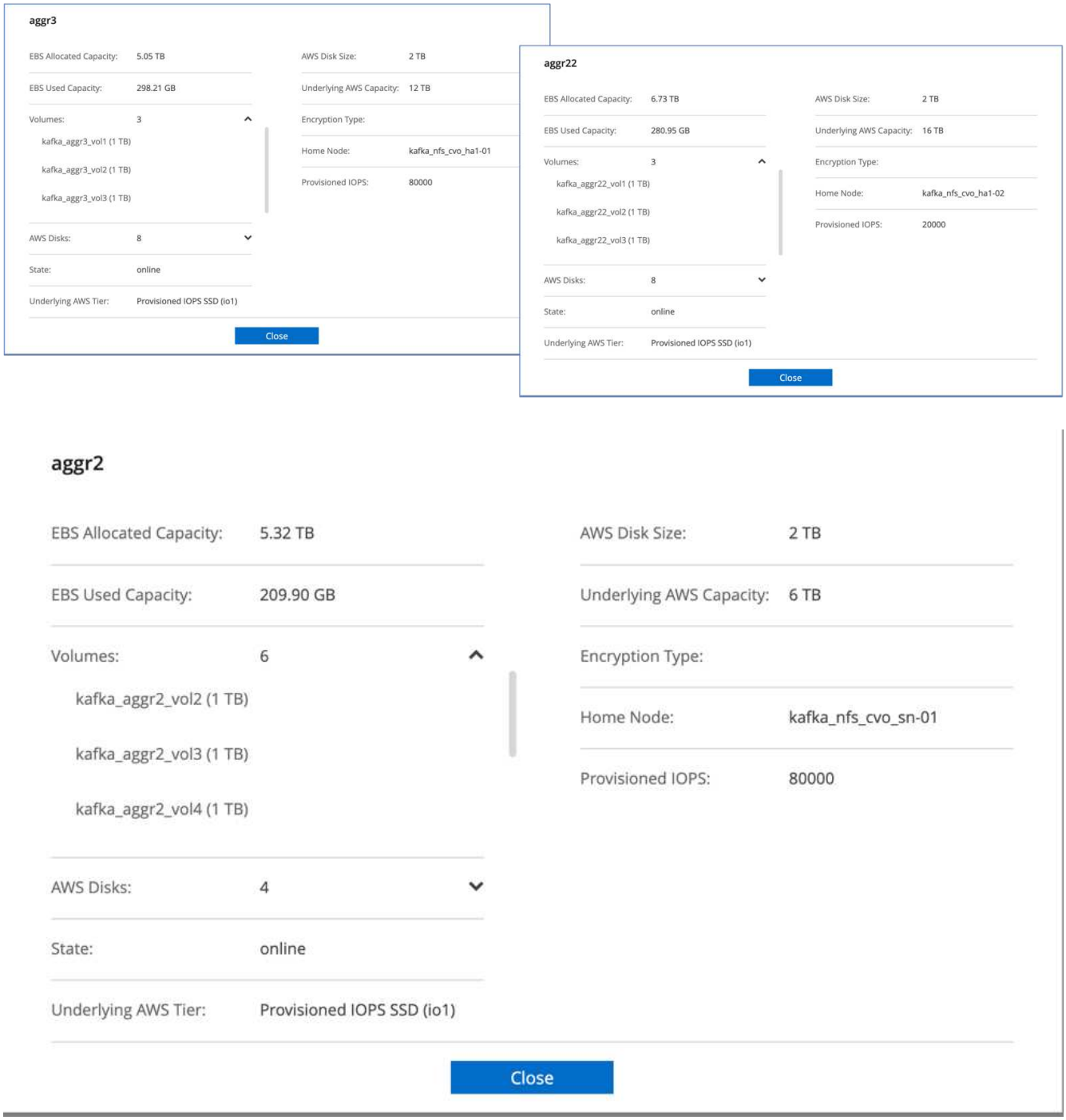
아키텍처 설정

다음 표에는 NAS를 사용하는 Kafka 클러스터의 환경 구성이 나와 있습니다.

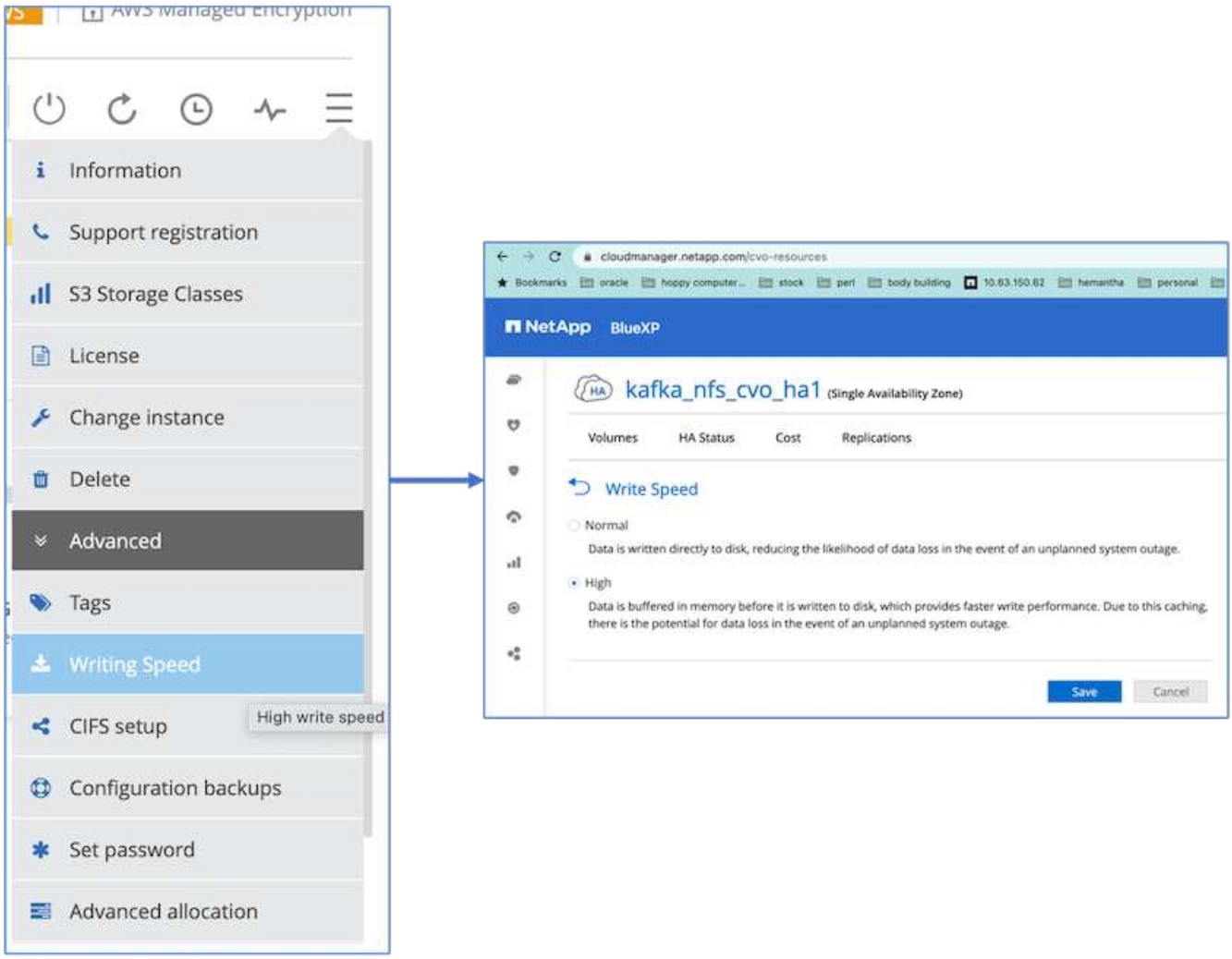
플랫폼 구성 요소	환경 구성
Kafka 3.2.3	• 3 x zookeers – T2.small • 브로커 서버 3대 – i3en.2xLarge • Grafana 1개 – c5n.2xLarge • 생산자/소비자 4대 — c5n.2xLarge *
모든 노드의 운영 체제	RHEL8.6
NetApp Cloud Volumes ONTAP 인스턴스	HA 쌍 인스턴스 – m5dn.12xLargge x 2노드 단일 노드 인스턴스 - m5dn.12xLargge x 1개 노드

NetApp 클러스터 볼륨 **ONTAP** 설정

1. Cloud Volumes ONTAP HA 쌍의 경우, 각 스토리지 컨트롤러에서 각 애그리게이트의 볼륨 3개가 포함된 애그리게이트 2개를 생성했습니다. 단일 Cloud Volumes ONTAP 노드의 경우 Aggregate에 6개의 볼륨을 생성합니다.

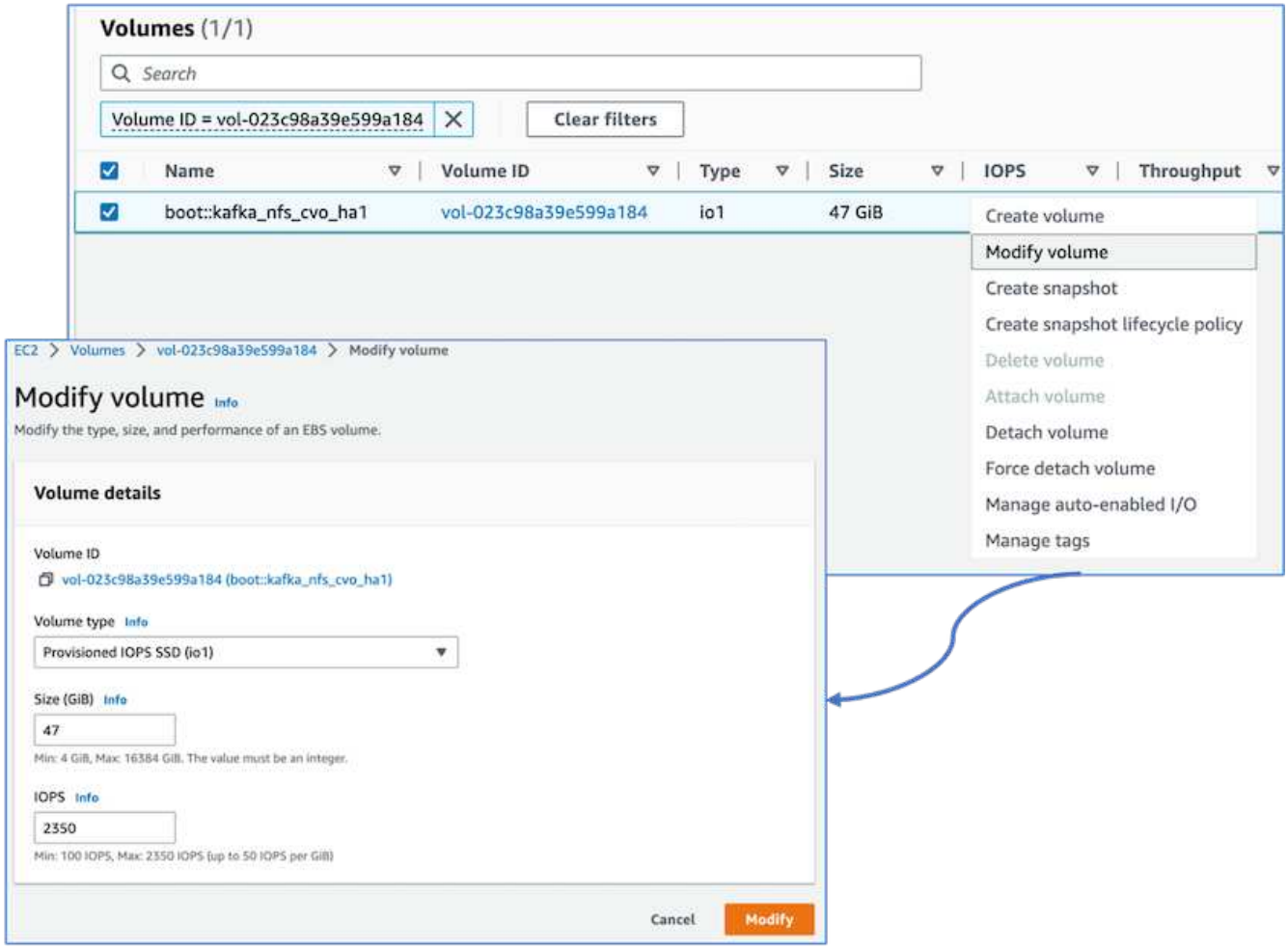


2. 네트워크 성능을 향상시키기 위해 HA 쌍과 단일 노드 모두에 고속 네트워킹을 활성화했습니다.



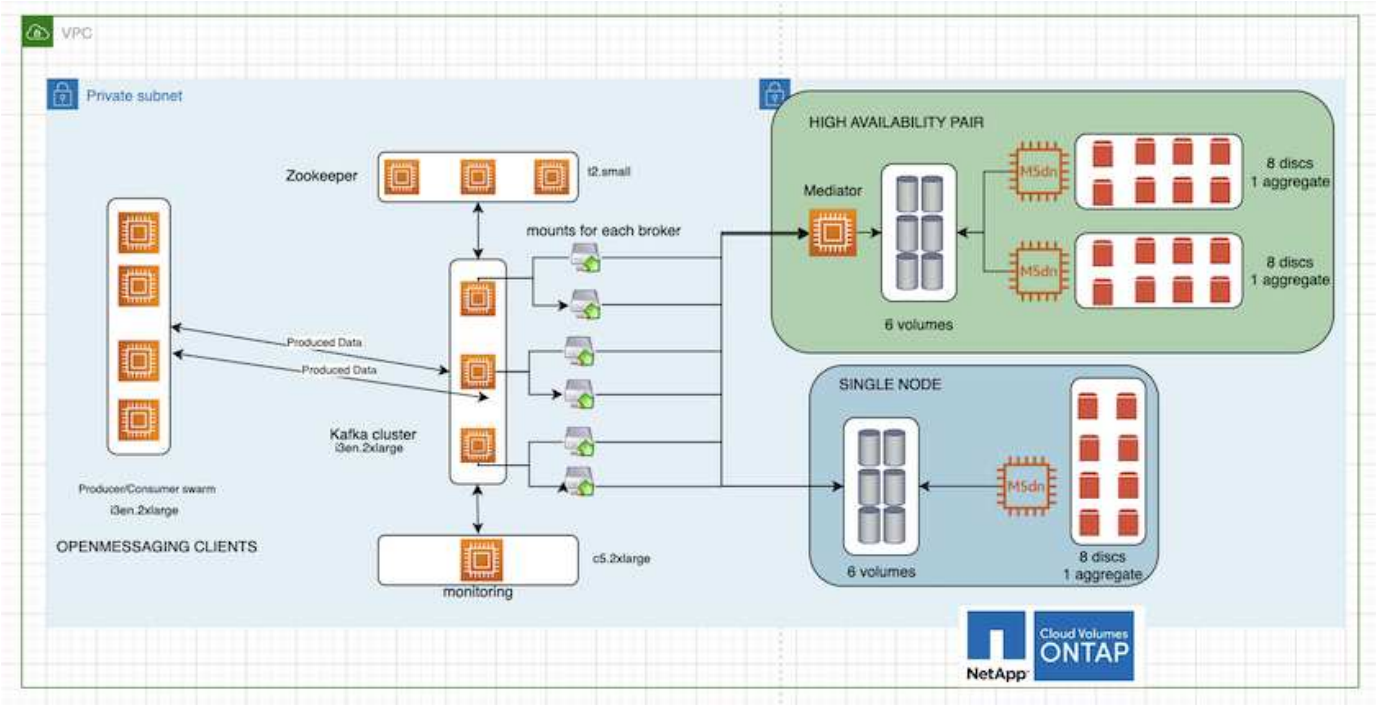
3. ONTAP NVRAM의 IOPS가 더 많으므로 Cloud Volumes ONTAP 루트 볼륨의 IOPS를 2350으로 변경했습니다. Cloud Volumes ONTAP의 루트 볼륨 디스크 크기는 47GB였습니다. 다음 ONTAP 명령은 HA 쌍용이며 단일 노드에도 동일한 단계를 적용할 수 있습니다.

```
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::*>
```



다음 그림에서는 NAS 기반 Kafka 클러스터의 아키텍처를 보여 줍니다.

- * Compute. * 3노드 Kafka 클러스터와 전용 서버에서 실행되는 3노드 zookeeper 앙상블이 사용되었습니다. 각 브로커에는 전용 LIF를 통해 Cloud Volumes ONTAP 인스턴스의 단일 볼륨에 대한 NFS 마운트 지점이 2개 있었습니다.
- * 모니터링. * Prometheus-Grafana 조합에 두 개의 노드를 사용했습니다. 워크로드를 생성하는데 이 Kafka 클러스터를 생성하고 사용할 수 있는 별도의 3노드 클러스터를 사용했습니다.
- * 스토리지. * 인스턴스에 하나의 6TB GP3 AWS-EBS 볼륨이 마운트된 HA 쌍 Cloud Volumes ONTAP 인스턴스를 사용했습니다. 그런 다음 NFS 마운트를 사용하여 Kafka 브로커로 볼륨을 내보냅니다.



Openmessage 벤치마킹 구성

1. NFS 성능을 개선하려면 nconnect를 사용하여 생성할 수 있는 NFS 서버와 NFS 클라이언트 간에 더 많은 네트워크 연결이 필요합니다. 다음 명령을 실행하여 브로커 노드에서 nconnect 옵션으로 NFS 볼륨을 마운트합니다.

```
[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. Cloud Volumes ONTAP에서 네트워크 연결을 확인합니다. 다음 ONTAP 명령은 단일 Cloud Volumes ONTAP 노드에서 사용됩니다. Cloud Volumes ONTAP HA 쌍에도 동일한 단계가 적용됩니다.

```
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. 다음 Kafka를 사용합니다 `server.properties` 모든 Kafka 브로커는 Cloud Volumes ONTAP HA 페어를 지원합니다. 를 클릭합니다 `log.dirs` 각 브로커에 따라 속성이 다르며 나머지 속성은 브로커에 공통입니다. broker1의 경우, 를 참조하십시오 `log.dirs` 값은 다음과 같습니다.

```
[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 ~]#
```

◦ broker2의 경우, 를 참조하십시오 `log.dirs` 속성 값은 다음과 같습니다.

```
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
```

◦ broker3의 경우, 를 참조하십시오 `log.dirs` 속성 값은 다음과 같습니다.

```
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. 단일 Cloud Volumes ONTAP 노드의 경우 Kafka입니다 `servers.properties` 은(는) 를 제외하고 Cloud Volumes ONTAP HA 쌍과 동일합니다 `log.dirs` 속성.
   - broker1의 경우, 를 참조하십시오 `log.dirs` 값은 다음과 같습니다.

```
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
```

   - broker2의 경우, 를 참조하십시오 `log.dirs` 값은 다음과 같습니다.

```
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
```

   - broker3의 경우, 를 참조하십시오 `log.dirs` 속성 값은 다음과 같습니다.

```
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. OMB의 워크로드는 다음과 같은 속성으로 구성됩니다. `(/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
```

를 클릭합니다 `messageSize` 사용 사례마다 다를 수 있습니다. 성능 테스트에서는 3K를 사용했습니다.

OMB에서 서로 다른 두 드라이버, 즉 Sync 또는 Throughput을 사용하여 Kafka 클러스터에서 워크로드를 생성했습니다.

- Sync 드라이버 속성에 사용되는 YAML 파일은 다음과 같습니다 (`/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
```

- 처리량 운전자 속성에 사용되는 YAML 파일은 다음과 같습니다 (`/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
```

테스트 방법

1. Kafka 클러스터는 Terraform 및 Ansible을 사용하여 위에서 설명한 사양에 따라 프로비저닝되었습니다. Terraform은 Kafka 클러스터용 AWS 인스턴스를 사용하여 인프라를 구축하는 데 사용되며, Ansible은 Kafka 클러스터를 기반으로 합니다.
2. 위에 설명된 워크로드 구성과 동기화 드라이버로 OMB 워크로드가 트리거되었습니다.

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

3. 동일한 워크로드 구성의 처리량 드라이버에서 또 다른 워크로드가 트리거되었습니다.

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

관찰

NFS에서 실행되는 Kafka 인스턴스의 성능을 벤치마크하는 워크로드를 생성하는 데 두 가지 유형의 드라이버가 사용되었습니다. 드라이버의 차이점은 로그 플러시 속성입니다.

Cloud Volumes ONTAP HA 쌍:

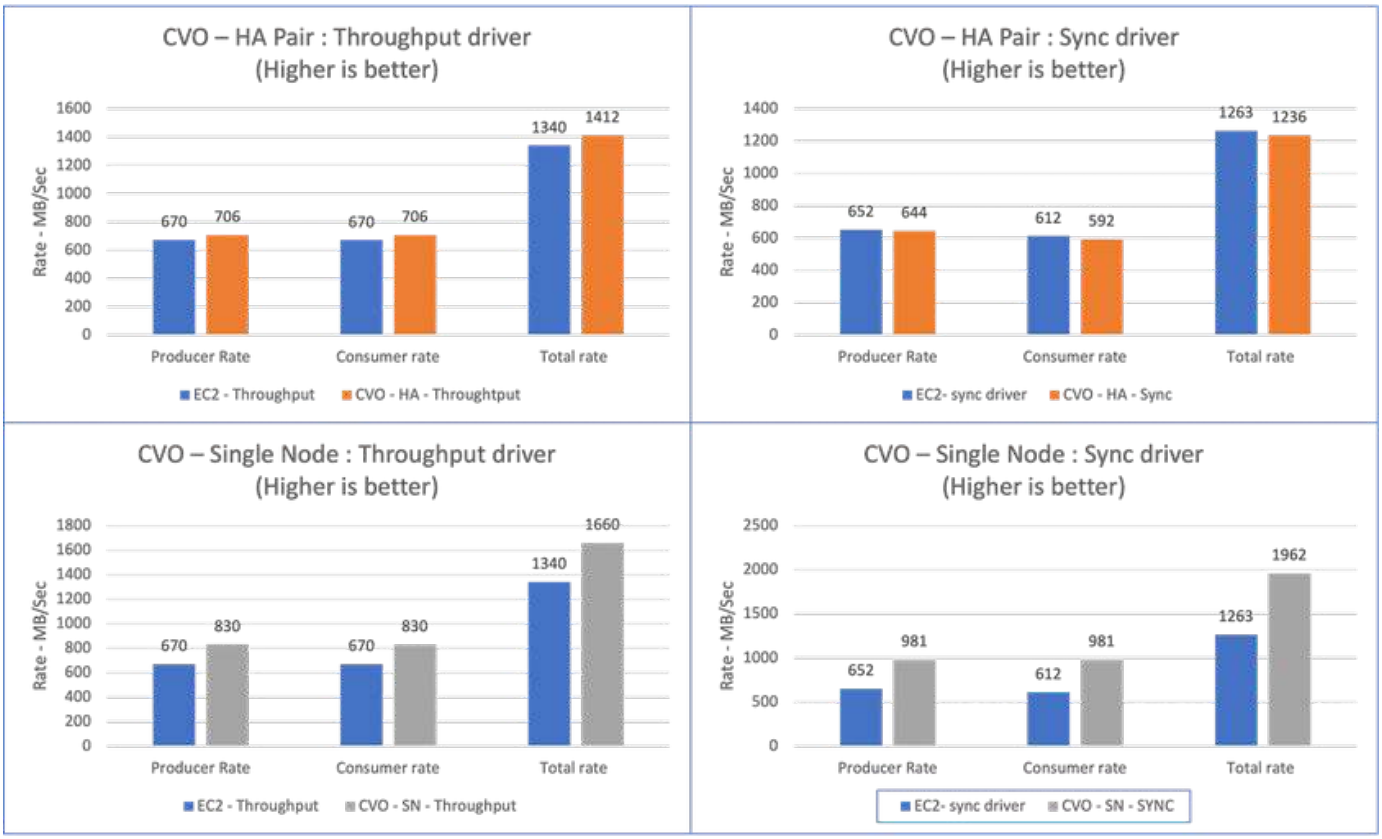
- Sync 드라이버에서 일관되게 생성된 총 처리량: ~1236Mbps.
- 처리량 드라이버에 대해 생성된 총 처리량: 최대 1412Mbps.

단일 Cloud Volumes ONTAP 노드의 경우:

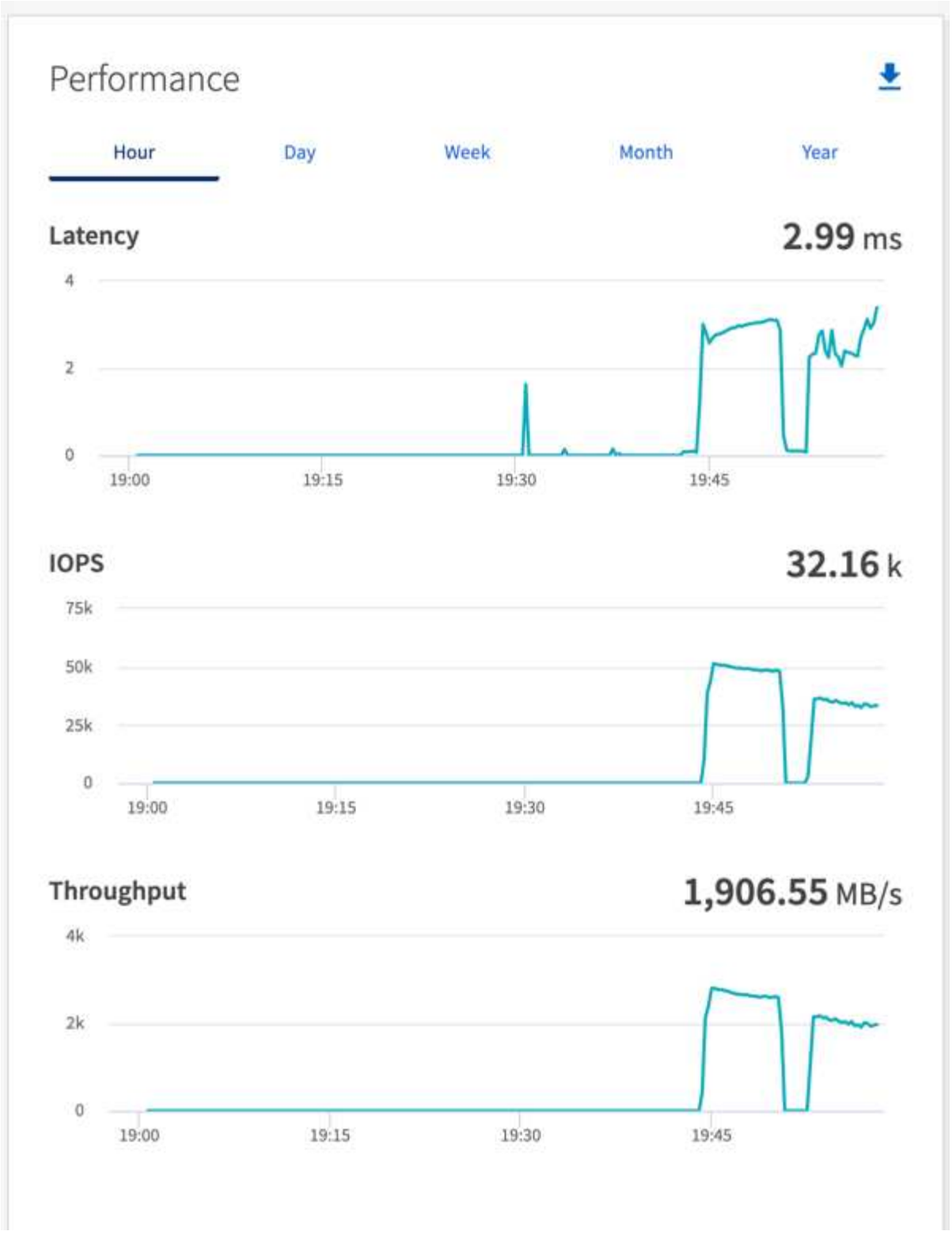
- Sync 드라이버에서 일관되게 생성된 총 처리량: ~1962MBps
- 처리량 드라이버에서 생성된 총 처리량: 최대 1660MBps

Sync 드라이버는 로그가 디스크에 즉시 플러시될 때 일관된 처리량을 생성할 수 있는 반면, 처리량 드라이버는 로그가 대량으로 디스크에 커밋될 때 처리량 버스트를 생성합니다.

이러한 처리량 수치는 지정된 AWS 구성에 대해 생성됩니다. 더 높은 성능 요구 사항을 위해 더 나은 처리량 수치를 위해 인스턴스 유형을 확장하고 조정할 수 있습니다. 총 처리량 또는 총 속도는 생산자와 소비자 속도의 조합입니다.



처리량 또는 동기화 드라이버 벤치마킹 수행 시 스토리지 처리량을 확인하십시오.



NetApp ONTAP용 AWS FSx의 성능 개요 및 검증

NetApp NFS에 스토리지 계층이 마운트된 Kafka 클러스터가 NetApp ONTAP용 AWS FSx의 성능을 벤치마킹했습니다. 벤치마킹 예는 다음 섹션에 설명되어 있습니다.

NetApp ONTAP용 AWS FSx의 Apache Kafka

NFS(Network File System)는 대량의 데이터를 저장하는 데 널리 사용되는 네트워크 파일 시스템입니다. 대부분의 조직에서 데이터는 Apache Kafka와 같은 스트리밍 애플리케이션에 의해 점점 더 많이 생성되고 있습니다. 이러한 워크로드는 최신 스토리지 기능을 갖춘 확장성, 짧은 지연 시간, 강력한 데이터 수집 아키텍처를 필요로 합니다. 실시간 분석을 지원하고 실행 가능한 통찰력을 제공하기 위해 우수한 설계 및 고성능 인프라가 필요합니다.

Kafka는 POSIX 호환 파일 시스템과 호환되며 파일 시스템에 의존하여 파일 작업을 처리하지만 NFSv3 파일 시스템에 데이터를 저장할 때 Kafka 브로커 NFS 클라이언트는 XFS 또는 ext4 같은 로컬 파일 시스템과 다르게 파일 작업을 해석할 수 있습니다. 일반적인 예로는 클러스터를 확장하고 파티션을 다시 할당할 때 Kafka 브로커가 실패하는 NFS의 이름이 있습니다. 이러한 문제를 해결하기 위해 NetApp은 RHEL8.7, RHEL9.1에서 일반적으로 사용할 수 있는 변경 사항으로 오픈 소스 Linux NFS 클라이언트를 업데이트했으며 현재 NetApp ONTAP용 FSx 릴리즈 ONTAP 9.12.1에서 지원합니다.

NetApp ONTAP용 Amazon FSx는 클라우드에서 완벽하게 관리되고 확장 가능하며 뛰어난 성능의 NFS 파일 시스템을 제공합니다. NetApp ONTAP용 FSx의 Kafka 데이터는 대량의 데이터를 처리하고 내결함성을 보장하기 위해 확장할 수 있습니다. NFS는 중요하거나 민감한 데이터 세트에 대해 중앙 집중식 스토리지 관리 및 데이터 보호를 제공합니다.

이러한 향상된 기능을 통해 AWS 고객은 AWS 컴퓨팅 서비스에서 Kafka 워크로드를 실행할 때 NetApp ONTAP용 FSx를 활용할 수 있습니다. 다음과 같은 이점을 제공합니다.
* CPU 활용률을 줄여 I/O 대기 시간을 줄입니다
* 더 빠른 Kafka 브로커 복구 시간.
* 안정성 및 효율성.
* 확장성 및 성능.
* 다중 가용성 영역 가용성.
* 데이터 보호.

NetApp ONTAP용 AWS FSx의 성능 개요 및 검증

NetApp NFS에 스토리지 계층이 마운트된 Kafka 클러스터가 AWS 클라우드의 성능을 벤치마킹했습니다. 벤치마킹 예는 다음 섹션에 설명되어 있습니다.

NetApp ONTAP용 AWS FSx의 Kafka

NetApp ONTAP용 AWS FSx가 있는 Kafka 클러스터는 AWS 클라우드에서 성능을 벤치마킹했습니다. 이 벤치마킹은 다음 섹션에서 설명합니다.

아키텍처 설정

다음 표에는 NetApp ONTAP용 AWS FSx를 사용하는 Kafka 클러스터의 환경 구성이 나와 있습니다.

플랫폼 구성 요소	환경 구성
Kafka 3.2.3	• 3 x zookeers – T2.small • 브로커 서버 3대 – i3en.2xLarge • Grafana 1개 – c5n.2xLarge • 생산자/소비자 4대 — c5n.2xLarge *
모든 노드의 운영 체제	RHEL8.6
NetApp ONTAP용 AWS FSx	4GB/sec의 처리량과 160000 IOPS의 Multi-AZ

NetApp ONTAP 설정용 NetApp FSx

1. 초기 테스트를 위해 2TB 용량과 40000 IOPS로 2GB/sec 처리량의 NetApp ONTAP 파일 시스템에 대한 FSx를 만들었습니다.

```
[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"}
```

   이 예에서는 AWS CLI를 통해 NetApp ONTAP용 FSx를 구축하고 있습니다. 필요에 따라 사용자 환경에서 명령을 추가로 사용자 지정해야 합니다. NetApp ONTAP용 FSX는 AWS 콘솔을 통해 추가로 구축 및 관리할 수 있어 명령줄 입력 작업을 줄이고 더욱 쉽고 능률적인 배포 환경을 제공합니다.

   FSx for NetApp ONTAP에 대한 설명서이며, 테스트 영역(US-East-1)의 2GB/sec 처리 파일 시스템에 대해 달성 가능한 최대 IOPS는 80,000 IOPS입니다. NetApp ONTAP 파일 시스템용 FSx의 총 최대 IOPS는 160,000 IOPS이며, 이를 위해서는 4GB/sec의 처리량 구축이 필요하며, 이 문서의 후반부에 이 내용이 설명되어 있습니다.

   NetApp ONTAP 성능 사양에 대한 FSx에 대한 자세한 내용은 여기 에서 AWS FSx for NetApp ONTAP 설명서를 참조하십시오. https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/performance.html .

   FSx "create-file-system"에 대한 자세한 명령줄 구문은 여기에서 찾을 수 있습니다. https://docs.aws.amazon.com/cli/latest/reference/fsx/create-file-system.html

   예를 들어 KMS 키를 지정하지 않을 때 사용되는 기본 AWS FSx 마스터 키가 아니라 특정 KMS 키를 지정할 수 있습니다.

2. NetApp ONTAP 파일 시스템에 대한 FSx를 생성하는 동안 다음과 같이 파일 시스템을 설명한 후 JSON 반환에서 "lifeCycle" 상태가 "Available"으로 변경될 때까지 기다립니다.

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

3. fsxadmin 사용자로 NetApp ONTAP SSH용 FSX에 로그인하여 자격 증명을 검증합니다. Fsxadmin은 생성 시 NetApp ONTAP 파일 시스템용 FSx의 기본 관리자 계정입니다. fsxadmin의 암호는 1단계에서 완료한 대로 AWS 콘솔 또는 AWS CLI에서 파일 시스템을 처음 생성할 때 구성된 암호입니다.

```
[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. 자격 증명이 확인되면 NetApp ONTAP 파일 시스템용 FSx에 스토리지 가상 머신을 생성합니다

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

SVM(스토리지 가상 시스템)은 NetApp ONTAP 볼륨용 FSx의 데이터를 관리 및 액세스하기 위한 자체 관리 자격 증명과 엔드포인트를 갖춘 격리된 파일 서버이며 NetApp ONTAP 멀티 테넌시를 위한 FSx를 제공합니다.

5. 기본 스토리지 가상 머신을 구성한 후에는 새로 생성된 NetApp ONTAP 파일 시스템용 FSx에 SSH를 사용하여 스토리지 가상 머신에 볼륨을 생성하고, 이와 유사하게 이 검증을 위해 6개의 볼륨을 생성합니다. 당사의 검증을 기반으로 Kafka에 더 나은 성능을 제공할 수 있는 기본 구성 요소(8) 이하를 유지합니다.

```
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. 테스트를 위해 볼륨에 추가 용량이 필요합니다. 볼륨의 크기를 2TB로 확장하고 접합 경로에 마운트합니다.

```
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
```

NetApp ONTAP용 FSx에서는 볼륨을 씬 프로비저닝할 수 있습니다. 이 예에서는 총 확장 볼륨 용량이 총 파일 시스템 용량을 초과하므로 다음 단계에서 보여 드릴 추가 프로비저닝 볼륨 용량을 잠금 해제하려면 전체 파일 시스템 용량을 확장해야 합니다.

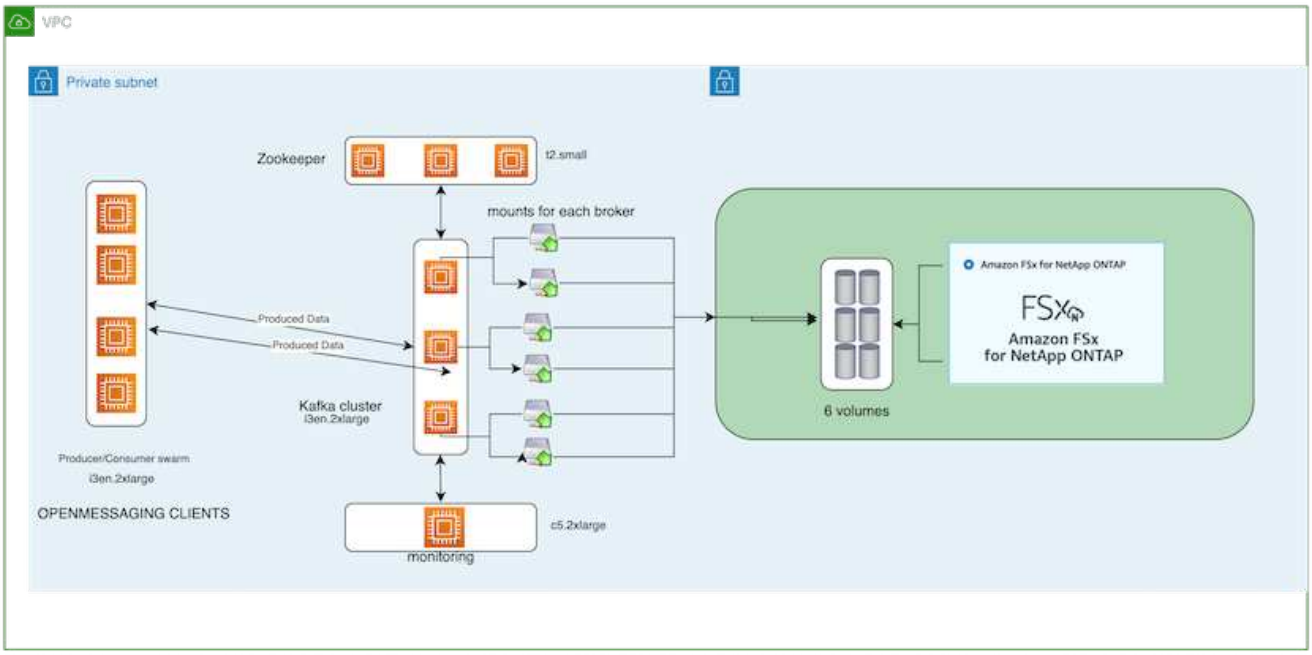
7. 다음으로 성능 및 용량을 늘리기 위해 NetApp ONTAP용 FSx 처리량 용량을 2GB/sec에서 4GB/sec로, IOPS는 160000으로, 용량은 5TB로 확장합니다

```
[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
```

FSx "update-file-system"에 대한 자세한 명령줄 구문은 여기에서 찾을 수 있습니다. https://docs.aws.amazon.com/cli/latest/reference/fsx/update-file-system.html

8. NetApp ONTAP 볼륨용 FSx는 Kafka 브로커에서 nconnect 및 기본 opions로 마운트됩니다

다음 그림은 NetApp ONTAP 기반 Kafka 클러스터용 FSx의 최종 아키텍처를 보여줍니다.



◦ 컴퓨팅. 3노드 Kafka 클러스터를 전용 서버에서 실행되는 3노드 zookeeper 앙상블과 함께 사용했습니다. 각

브로커는 NetApp ONTAP 인스턴스의 FSx에 있는 6개의 볼륨에 6개의 NFS 마운트 지점을 가지고 있었습니다.

- 모니터링. 두 개의 노드를 사용하여 Prometheus-Grafana 조합을 사용했습니다. 워크로드를 생성하는데 이 Kafka 클러스터를 생성하고 사용할 수 있는 별도의 3노드 클러스터를 사용했습니다.
- 스토리지. 2TB 볼륨 6개가 마운트된 NetApp ONTAP용 FSx를 사용했습니다. 그런 다음 NFS 마운트를 사용하여 Kafka 브로커로 볼륨을 내보냅니다. NetApp ONTAP 볼륨의 FSx는 Kafka 브로커에서 16개의 nconnect 세션과 기본 옵션으로 마운트됩니다.

Openmessage 벤치마킹 구성.

NetApp Cloud Volumes ONTAP에 사용된 것과 동일한 구성을 사용했으며 세부 정보는 다음과 같습니다. https://docs.netapp.com/us-en/netapp-solutions/data-analytics/kafka-nfs-performance-overview-and-validation-in-aws.html#architectural-setup

테스트 방법

1. Kafka 클러스터는 Terraform 및 Ansible을 사용하여 위에서 설명한 사양에 따라 프로비저닝되었습니다. Terraform은 Kafka 클러스터용 AWS 인스턴스를 사용하여 인프라를 구축하는 데 사용되며, Ansible은 Kafka 클러스터를 기반으로 합니다.
2. 위에 설명된 워크로드 구성과 동기화 드라이버로 OMB 워크로드가 트리거되었습니다.

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

3. 동일한 워크로드 구성의 처리량 드라이버에서 또 다른 워크로드가 트리거되었습니다.

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

관찰

NFS에서 실행되는 Kafka 인스턴스의 성능을 벤치마크하는 워크로드를 생성하는 데 두 가지 유형의 드라이버가 사용되었습니다. 드라이버의 차이점은 로그 플러시 속성입니다.

Kafka 복제 계수 1 및 NetApp ONTAP용 FSx의 경우:

- Sync 드라이버에서 일관되게 생성된 총 처리량: ~3218Mbps 및 최대 성능: ~3652Mbps.
- 처리량 드라이버에서 일관되게 생성된 총 처리량: ~3679Mbps 및 최대 성능: ~3908Mbps.

복제 계수가 3인 Kafka 및 NetApp ONTAP용 FSx의 경우:

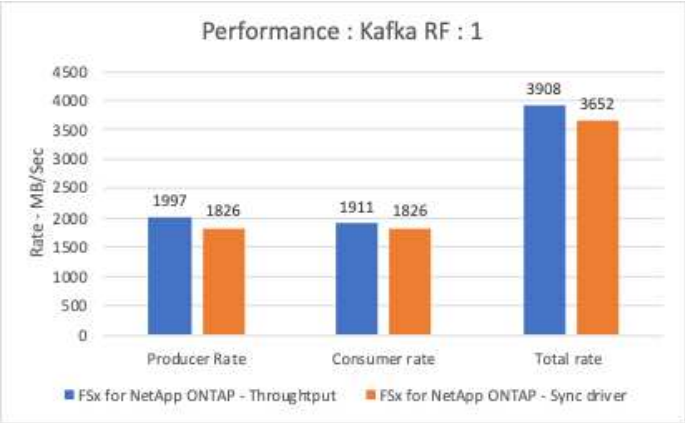
- Sync 드라이버에서 일관되게 생성된 총 처리량: ~1252Mbps 및 최대 성능: ~1382Mbps.
- 처리량 드라이버에서 일관되게 생성된 총 처리량: ~1218Mbps 및 최대 성능: ~1328Mbps.

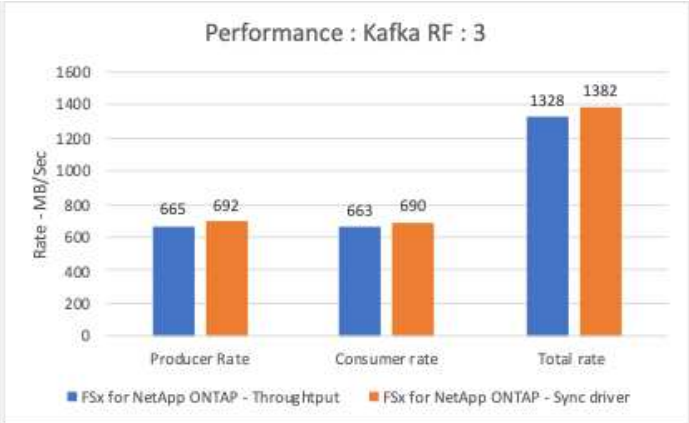
Kafka 복제 계수 3에서 읽기 및 쓰기 작업은 FSx for NetApp ONTAP에서 세 번, Kafka 복제 계수 1에서 NetApp ONTAP용 FSx에서 읽기 및 쓰기 작업이 한 번 수행되므로 두 검증 모두에서 최대 4GB/sec 처리량에 도달할 수

있습니다.

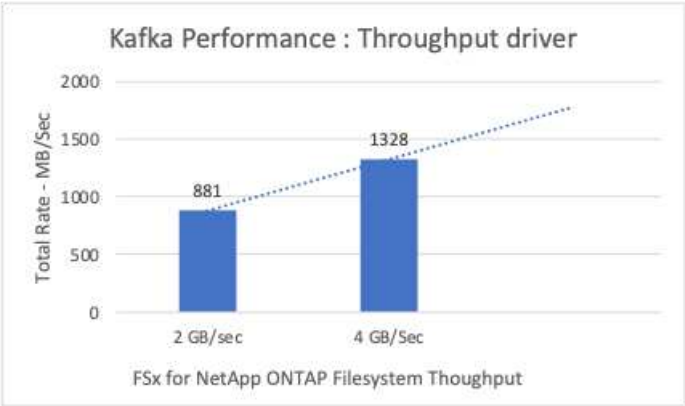
Sync 드라이버는 로그가 디스크에 즉시 플러시될 때 일관된 처리량을 생성할 수 있는 반면, 처리량 드라이버는 로그가 대량으로 디스크에 커밋될 때 처리량 버스트를 생성합니다.

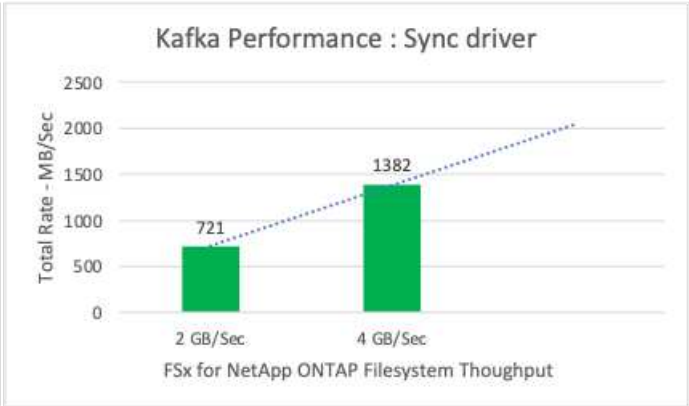
이러한 처리량 수치는 지정된 AWS 구성에 대해 생성됩니다. 더 높은 성능 요구 사항을 위해 더 나은 처리량 수치를 위해 인스턴스 유형을 확장하고 조정할 수 있습니다. 총 처리량 또는 총 속도는 생산자와 소비자 속도의 조합입니다.



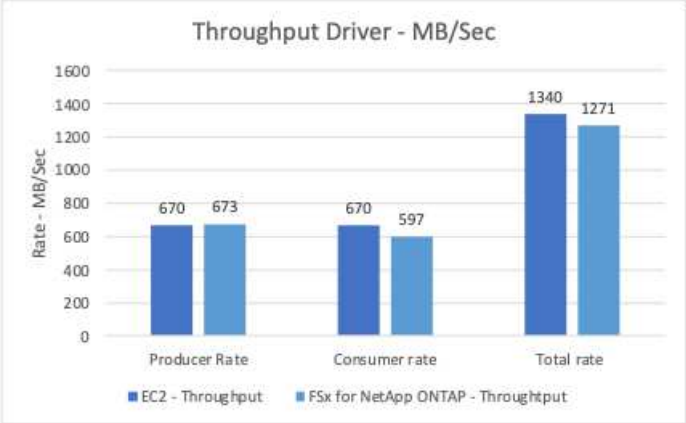


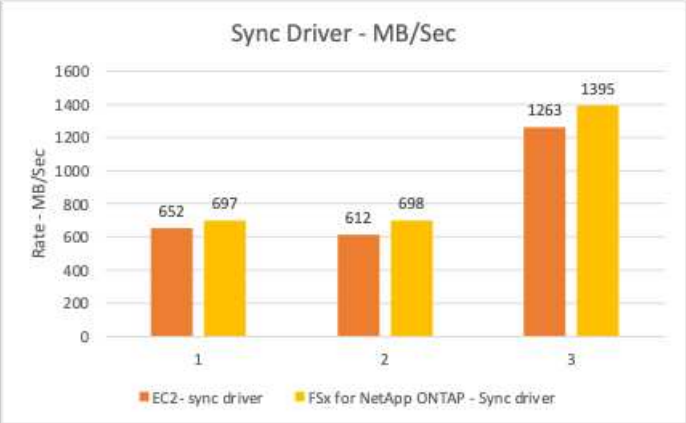
아래 차트는 NetApp ONTAP의 경우 2GB/sec FSx, Kafka 복제 계수 3의 경우 4GB/sec 성능을 보여 줍니다. 복제 계수 3은 NetApp ONTAP 스토리지용 FSx에서 읽기 및 쓰기 작업을 세 번 수행합니다. 처리량 드라이버의 총 속도는 881MB/sec이며, Kafka는 NetApp ONTAP 파일 시스템용 2GB/sec FSx에서 약 2.64GB/sec의 읽기 및 쓰기 작업을 수행하고 처리량 드라이버의 총 속도는 1328MB/sec이며, Kafka 작업은 약 3.98GB/sec입니다. 또한 Kafka 성능은 NetApp ONTAP 처리량을 위한 FSx를 기반으로 하는 선형적이고 확장성이 뛰어납니다.





아래 차트는 NetApp ONTAP용 EC2 인스턴스와 FSx 간의 성능을 보여줍니다(Kafka 복제 계수: 3).

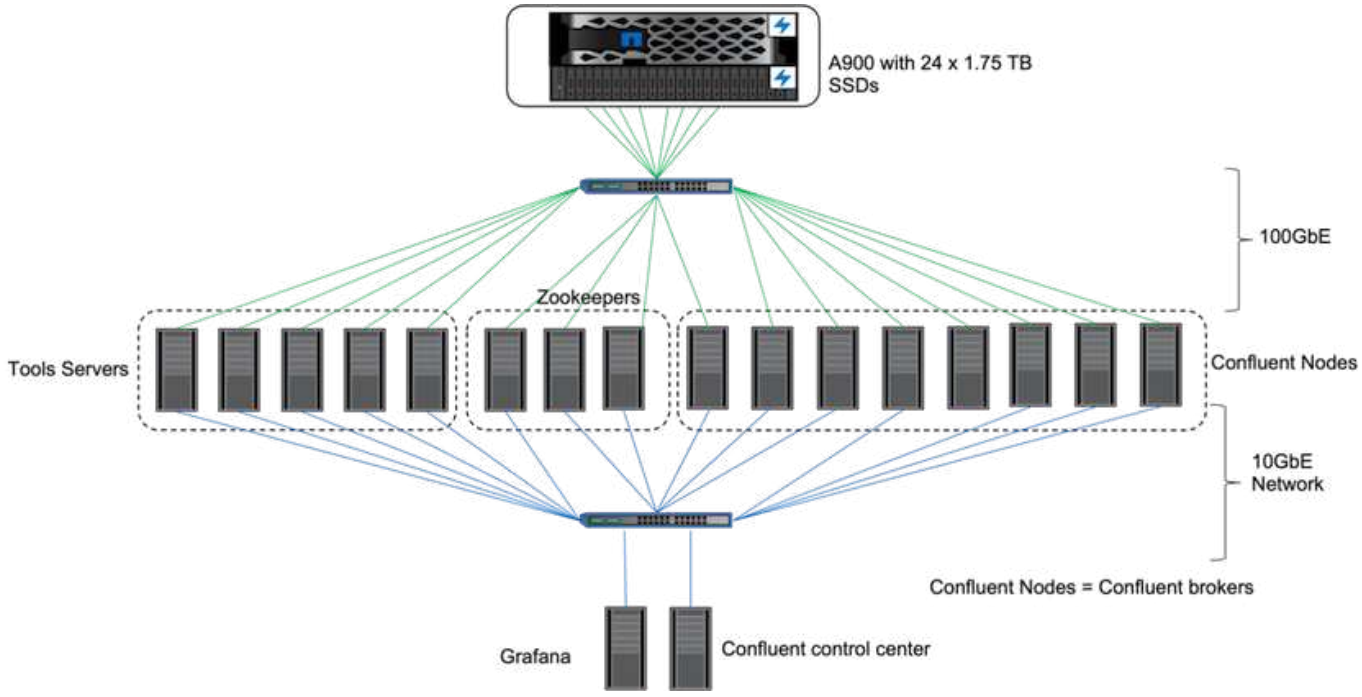




AFF A900 온프레미스 성능 개요 및 검증

사내에서 NetApp AFF A900 스토리지 컨트롤러와 ONTAP 9.12.1 RC1을 사용하여 Kafka 클러스터의 성능 및 확장성을 검증했습니다. ONTAP 및 AFF의 이전 계층형 스토리지 모범 사례와 동일한 테스트 베드를 사용했습니다.

AFF A900을 평가하기 위해 Confluent Kafka 6.2.0을 사용했습니다. 클러스터에는 8개의 브로커 노드와 3개의 Zookeeper 노드가 있습니다. 성능 테스트를 위해 OMB 작업자 노드 5개를 사용했습니다.



스토리지 구성

NetApp FlexGroups 인스턴스를 사용하여 로그 디렉토리에 단일 네임스페이스를 제공하여 복구 및 구성을 간소화했습니다. NFSv4.1 및 pNFS를 사용하여 로그 세그먼트 데이터에 직접 액세스할 수 있습니다.

클라이언트 튜닝

각 클라이언트는 다음 명령을 사용하여 FlexGroup 인스턴스를 마운트했습니다.

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

또한, 을 늘렸습니다 `max_session_slots` 를 선택합니다 `64` 를 선택합니다 `180`. 이는 ONTAP의 기본 세션 슬롯 제한과 일치합니다.

Kafka 브로커 튜닝

테스트 중인 시스템의 처리량을 극대화하기 위해 특정 주요 스레드 풀에 대한 기본 매개 변수를 크게 늘렸습니다. 대부분의 구성에 대해 Confluent Kafka 모범 사례를 따르는 것이 좋습니다. 이 튜닝은 스토리지에 대한 미해결 입출력의 동시성을 극대화하는 데 사용되었습니다. 이러한 매개 변수는 브로커의 컴퓨팅 리소스 및 스토리지 속성에 맞게 조정할 수 있습니다.

```
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
```

워크로드 생성기 테스트 방법

처리량 드라이버 및 주제 구성에 대한 클라우드 테스트와 동일한 OMB 구성을 사용했습니다.

1. AFF 클러스터에서 Ansible을 사용하여 FlexGroup 인스턴스를 프로비저닝했습니다.

```
---
- 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. ONTAP SVM에서 pNFS를 사용할 수 있게 되었습니다.

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

3. 워크로드는 Cloud Volumes ONTAP와 동일한 워크로드 구성을 사용하는 처리량 드라이버에서 트리거되었습니다. 자세한 내용은 " 단원을 참조하십시오안정적인 성능"아래에 있습니다. 워크로드는 복제 계수 3을 사용했습니다. 즉 NFS에서 로그 세그먼트의 복제본 3개가 유지되었습니다.

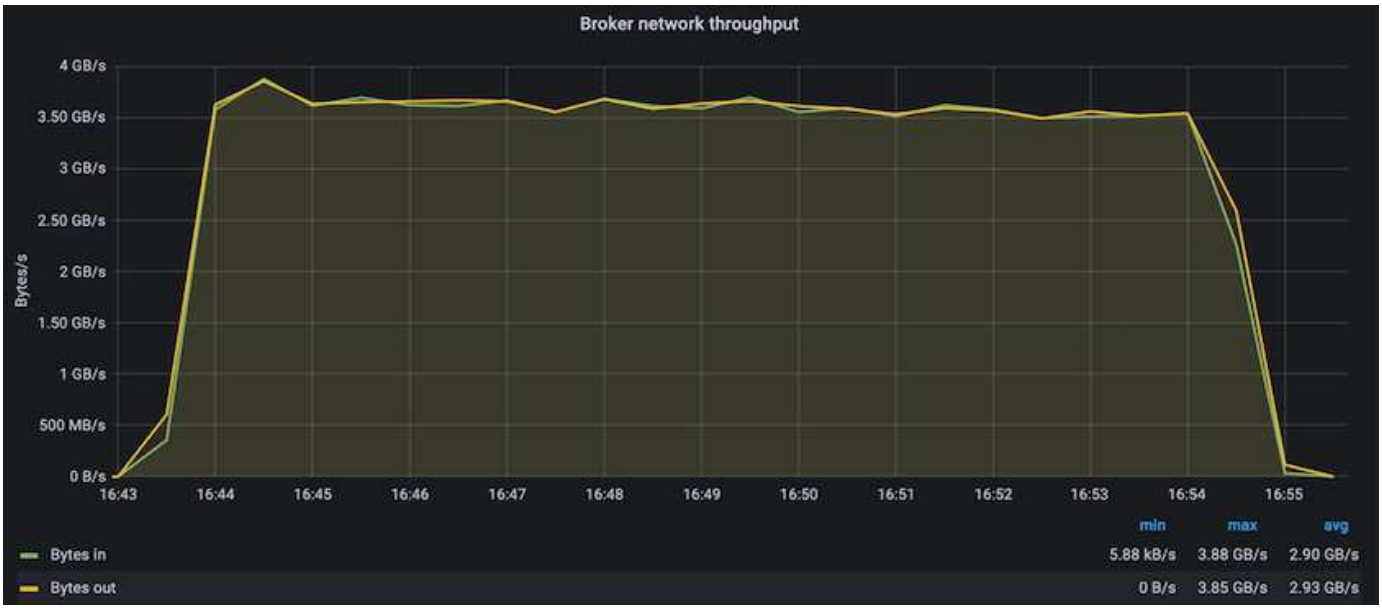
```
sudo bin/benchmark --drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

4. 마지막으로, 백로그를 사용하여 소비자가 최신 메시지를 따라잡을 수 있는 능력을 측정하는 측정을 완료했습니다. OMB는 측정을 시작하는 동안 소비자를 일시 중지하여 백로그를 생성합니다. 이 경우 백로그 생성(생산자 전용 트래픽), 백로그 드레이닝(소비자가 주제에서 놓친 이벤트를 포착하는 소비자 중심 단계), 안정적 상태 등 세 가지 단계가 발생합니다. 자세한 내용은 " 단원을 참조하십시오스토리지 제한값 탐색"를 참조하십시오.

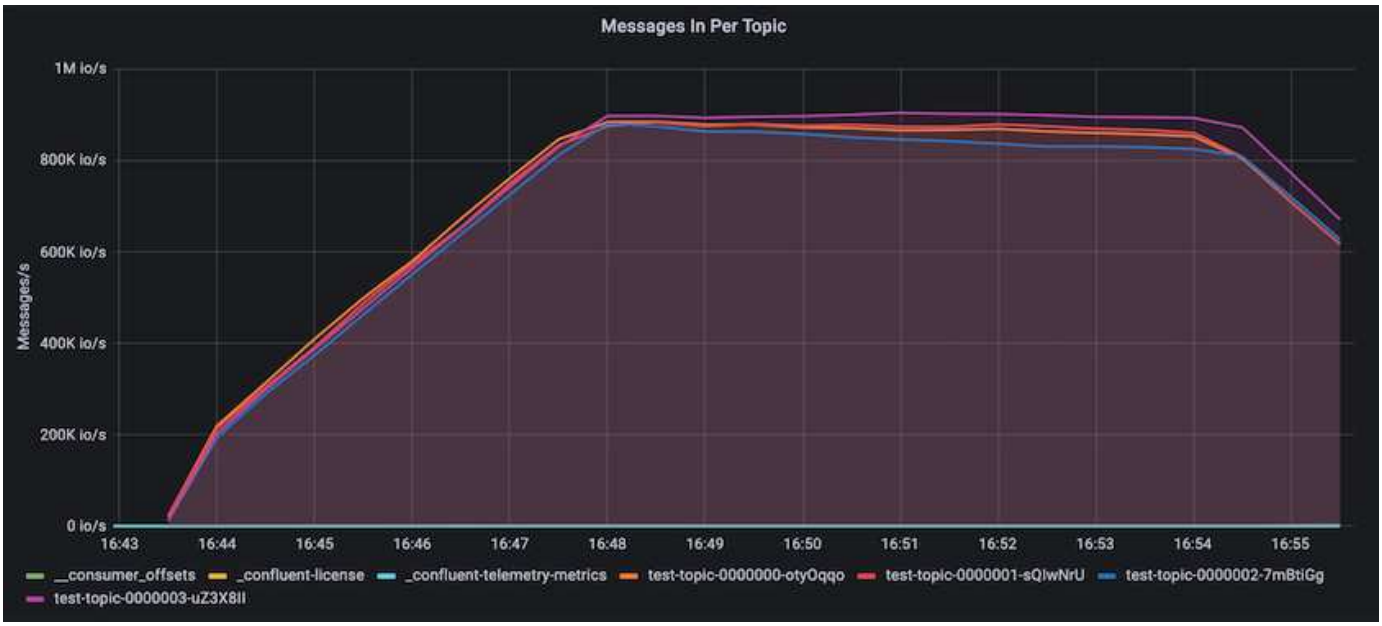
안정적인 성능

NetApp은 OpenMessaging 벤치마크를 사용하여 AFF A900을 평가하여 AWS의 Cloud Volumes ONTAP 및 AWS의 DAS와 유사한 비교를 제공했습니다. 모든 성능 값은 생산자 및 소비자 수준에서 Kafka 클러스터 처리량을 나타냅니다.

Confluent Kafka 및 AFF A900의 안정적인 상태 성능은 생산자와 소비자 모두의 평균 3.4GBps 처리량을 달성했습니다. 이는 Kafka 클러스터에서 340만 개 이상의 메시지입니다. BrokerTopicMetrics의 지속적인 처리량을 초당 바이트 단위로 시각화함으로써 AFF A900이 지원하는 뛰어난 안정적인 상태 성능과 트래픽을 확인할 수 있습니다.



이것은 주제별로 전달되는 메시지의 보기에 잘 부합됩니다. 다음 그래프는 주제별 분석을 제공합니다. 테스트를 거친 구성에서 4개 주제에 대해 주제당 약 900,000개의 메시지가 나타났습니다.

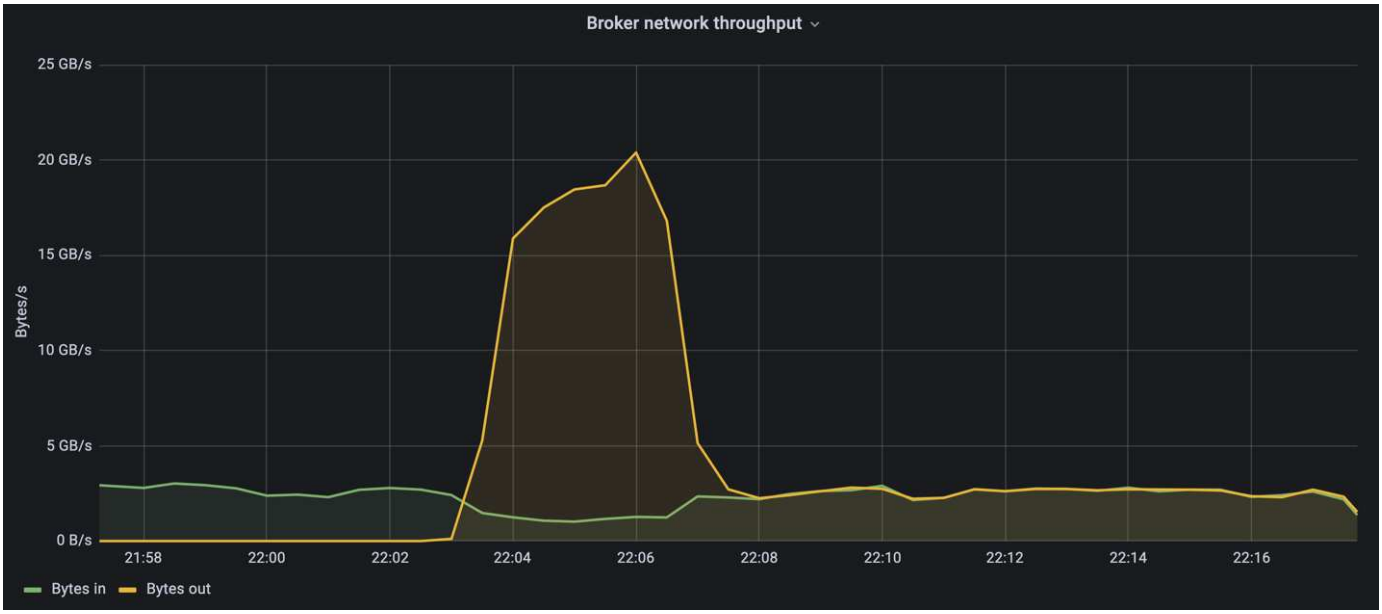


성능을 극대화 및 스토리지 제한 탐색

AFF의 경우 백로그 기능을 사용하여 OMB를 테스트했습니다. 백로그 기능은 Kafka 클러스터에 이벤트 백로그가 구축된 동안 소비자 가입을 일시 중지합니다. 이 단계에서는 프로듀서 트래픽만 발생하므로 로그에 커밋되는 이벤트가 생성됩니다. 이렇게 하면 일괄 처리 또는 오프라인 분석 워크플로를 가장 긴밀하게 에뮬레이트할 수 있습니다. 이러한 워크플로에서는 소비자 구독이 시작되고 브로커 캐시에서 이미 제거된 내역 데이터를 읽어야 합니다.

이 구성에서 소비자 처리량에 대한 저장소 제한을 이해하기 위해 A900이 흡수할 수 있는 쓰기 트래픽의 양을 파악하기 위해 생산자 전용 단계를 측정했습니다. 다음 섹션 " 을 참조하십시오사이징 지침"이 데이터를 활용하는 방법을 이해합니다.

생산자 전용 측정 과정에서 A900 성능의 한계를 넘어선 높은 피크 처리량을 볼 수 있었습니다(다른 브로커 리소스가 생산자 및 소비자 트래픽을 지원하는 데 포화 상태가 되지 않았을 때).



이 측정을 위해 메시지당 오버헤드를 제한하고 NFS 마운트 지점에 대한 스토리지 처리량을 최대화하기 위해 메시지 크기를 16K로 늘렸습니다.

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

Confluent Kafka 클러스터는 4.03GBps의 피크 프로듀서 처리량을 달성했습니다.

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

OMB가 eventbacklog를 채운 후 소비자 트래픽이 다시 시작되었습니다. 백로그 드레이닝으로 측정하는 동안 모든 항목에서 20Gbps 이상의 최고 소비자 처리량을 관찰했습니다. OMB 로그 데이터를 저장하는 NFS 볼륨에 대한 총 처리량이 최대 30Gbps에 근접했습니다.

사이징 지침

Amazon Web Services에서 제공합니다 "사이징 가이드" Kafka 클러스터의 사이징 및 확장에 사용됩니다.

이 사이징은 Kafka 클러스터의 스토리지 처리량 요구 사항을 결정하는 데 유용한 공식을 제공합니다.

복제 계수가 r인 tcluster 클러스터로 생성되는 총 처리량의 경우 브로커 스토리지에서 수신한 처리량은 다음과 같습니다.

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

이 작업은 더욱 단순화될 수 있습니다.

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

이 공식을 사용하여 Kafka 핫 티어 요구에 적합한 ONTAP 플랫폼을 선택할 수 있습니다.

다음 표에는 여러 복제 요소를 사용하는 A900의 예상 생산자 처리량이 설명되어 있습니다.

복제 계수	생산자 처리량**(GPPS)**
3(측정)	3.4
2	5.1
1	10.2

결론

이름 바꾸기 비용이 저렴하다는 어리석은 문제를 위한 NetApp 솔루션은 이전에 NFS와 호환되지 않았던 워크로드를 위해 중앙에서 관리하는 단순한 형태의 스토리지를 제공합니다.

이 새로운 패러다임은 고객이 재해 복구 및 데이터 보호를 위해 마이그레이션 및 미러링이 용이한 보다 관리가 용이한 Kafka 클러스터를 생성할 수 있도록 해 줍니다.
또한 NFS는 CPU 활용률 감소, 복구 시간 단축, 스토리지 효율성 대폭 향상, NetApp ONTAP를 통한 성능 향상과 같은 추가 이점을 제공합니다.

추가 정보를 찾을 수 있는 위치

이 문서에 설명된 정보에 대해 자세히 알아보려면 다음 문서 및/또는 웹 사이트를 검토하십시오.

- 아파치 카프카란?

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

- 이름이 바보 같습니까?

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

- ONATP는 스트리밍 애플리케이션에 대해 읽혀집니다.

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

- Kafka의 이름 변경 문제.

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

- NetApp 제품 설명서

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

- NFS란 무엇입니까?

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

- Kafka 파티션 재할당이란 무엇입니까?

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

- OpenMessaging 벤치마크란 무엇입니까?

  "https://openmessaging.cloud/"

- Kafka 브로커를 어떻게 마이그레이션합니까?

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

- Prometheus로 Kafka 브로커를 어떻게 모니터링하십니까?

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

- Apache Kafka용 관리 플랫폼

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

- 아파치 카프카 지원

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

- Apache Kafka용 컨설팅 서비스

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

저작권 정보

Copyright © 2024 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.