핫스팟 교정을 위한 **FlexCache**

ONTAP 9

NetApp
March 13, 2025

목차

핫스팟 교정을 위한 FlexCache

ONTAP로 고성능 컴퓨팅 워크로드의 핫 스팟 해결

애니메이션 렌더링 또는 EDA와 같은 여러 고성능 컴퓨팅 워크로드에서 흔히 발생하는 문제는 주목을 받고 있습니다. 핫스팟팅은 클러스터 또는 네트워크의 특정 부분이 다른 영역에 비해 상당한 부하를 경험하여 해당 위치에 집중된 과도한 데이터 트래픽으로 인해 성능 병목 현상이 발생하고 전반적인 효율성이 감소되는 상황을 말합니다. 예를 들어, 파일 또는 여러 파일이 실행 중인 작업에 대한 수요가 많아 해당 파일에 대한 요청을 처리하는 데 사용되는 CPU에 병목 현상이 발생합니다(볼륨 선호도를 통해). FlexCache는 이 병목 현상을 완화할 수 있지만 올바르게 설정해야 합니다.

이 문서에서는 핫 스팟팅을 개선하기 위해 FlexCache를 설정하는 방법에 대해 설명합니다.

2024년 7월부터 이전에 PDF로 게시된 기술 보고서의 콘텐츠가 ONTAP 제품 문서와 통합되었습니다. 이 ONTAP 핫스팟 개선 기술 보고서 내용은 발행일 현재 완전히 새로 작성되었으며 이전 형식은 아직 작성되지 않았습니다.

주요 개념

핫스팟 개선 계획을 세울 때는 이러한 필수 개념을 이해하는 것이 중요합니다.

- * 고밀도 FlexCache(HDF) *: 캐시 용량 요구 사항이 허용하는 한 몇 개의 노드로 확장되도록 압축된 FlexCache
- * HDF Array(HDFA) *: 클러스터 전체에 분산된 동일한 오리진의 캐시인 HDFS 그룹입니다
- * SVM 간 HDFA *: 서버 가상 머신(SVM)당 HDFA의 HDF 1개
- * 내부 SVM HDFA *: HDFA의 모든 HDFS가 하나의 SVM에 있음
- * 동서 트래픽 *: 클러스터 백엔드 트래픽은 간접 데이터 액세스로부터 생성됩니다

다음 단계

- "고밀도 FlexCache를 사용하여 핫 스팟을 개선하는 방법을 이해합니다"
- "FlexCache 어레이 밀도를 결정합니다"
- "HDFS의 밀도를 결정하고 SVM 간 HDFA 및 SVM 내 HDFA와 함께 NFS를 사용하여 HDFS에 액세스할지 여부를 결정합니다"
- "HDFA 및 데이터 LIF를 구성하여 ONTAP 구성에서 클러스터 내 캐싱을 사용할 경우의 이점을 실현하십시오"
- "클라이언트 구성을 사용하여 ONTAP NAS 연결을 분산하도록 클라이언트를 구성하는 방법에 대해 알아봅니다"

ONTAP FlexCache 핫스팟 개선 솔루션 설계

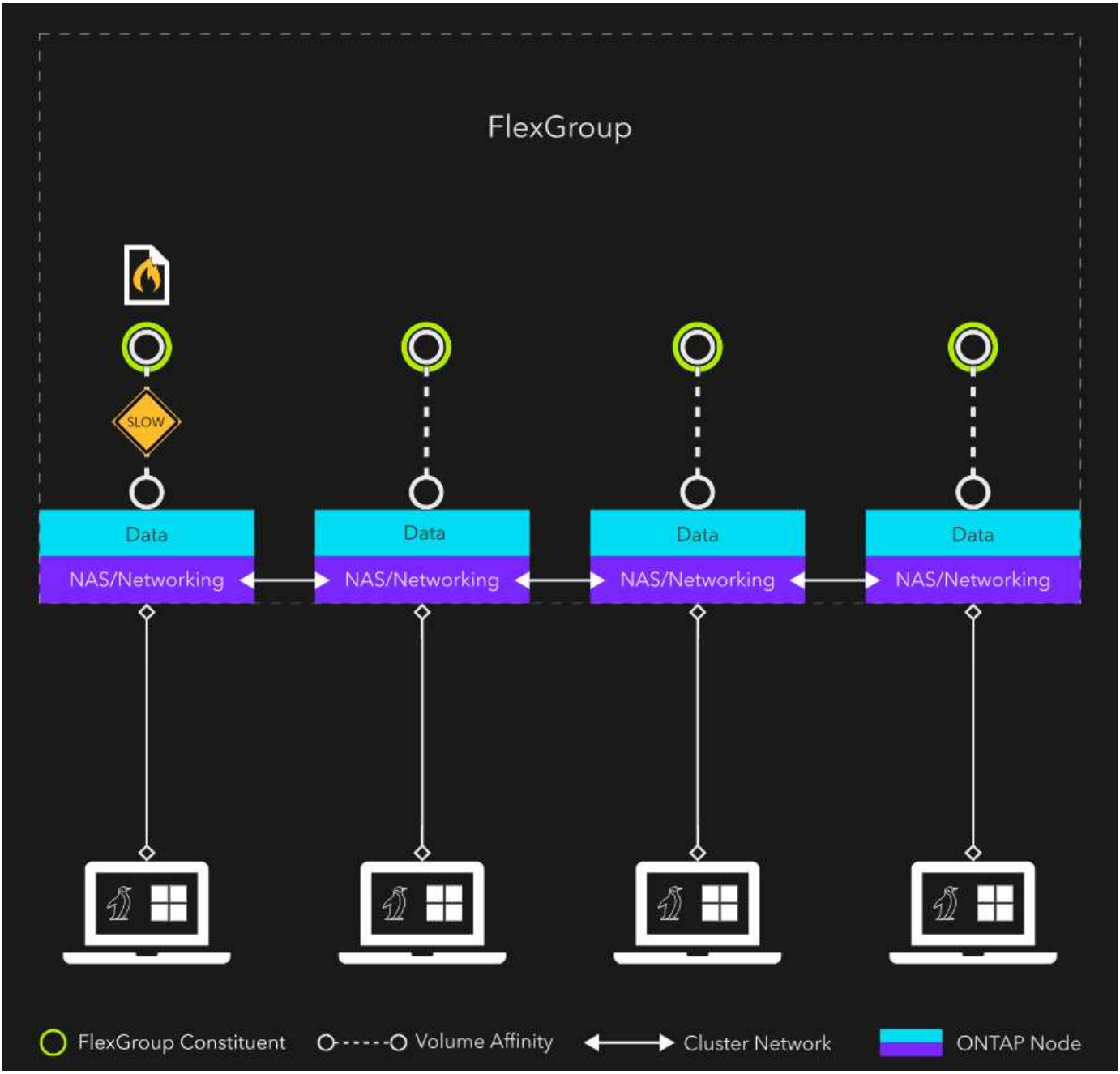
핫스팟을 해결하려면 병목 현상의 근본 원인, 자동 프로비저닝 FlexCache가 충분하지 않은 이유, FlexCache 솔루션을 효과적으로 설계하는 데 필요한 기술 세부 사항을 살펴봅니다. 고밀도 FlexCache 스토리지(HDFA)를 이해하고 구현하면 성능을 최적화하고 수요가 많은 작업 부하에서 병목 현상을 제거할 수 있습니다.

병목 현상 이해

다음은 이미지일반적인 단일 파일 핫 스팟팅 시나리오를 보여 줍니다. 이 볼륨은 노드당 단일 구성요소가 있는 FlexGroup이며 파일은 노드 1에 상주합니다.

모든 NAS 클라이언트의 네트워크 연결을 클러스터의 서로 다른 노드에 분산시키는 경우 핫 파일이 상주하는 볼륨 선호도를 지원하는 CPU의 병목 현상이 발생합니다. 또한 파일이 상주하는 위치가 아닌 노드에 연결된 클라이언트에서 수신되는 통화에 클러스터 네트워크 트래픽(동부-서부 트래픽)을 도입합니다. 동서부의 트래픽 오버헤드는 일반적으로 작지만 고성능 컴퓨팅 워크로드의 경우에는 비트 수가 거의 없습니다.

그림 **1: FlexGroup** 단일 파일 핫스팟 시나리오



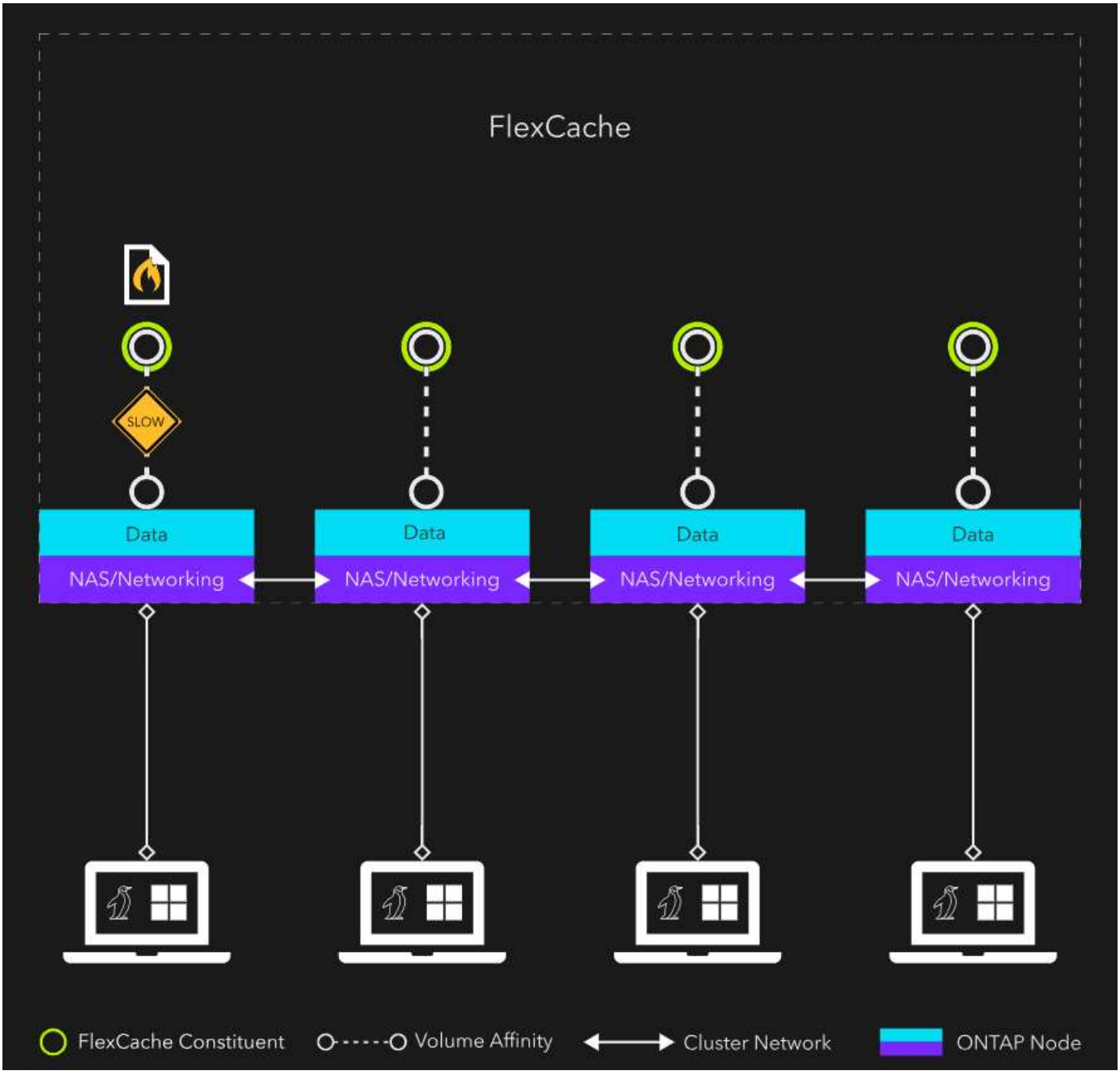
자동 프로비저닝된 **FlexCache**가 해답이 아닌 이유

핫스팟을 해결하려면 CPU 병목 현상을 제거하고 동서 트래픽도 제거합니다. FlexCache가 제대로 설정되어 있으면

도움이 될 수 있습니다.

다음 예에서는 FlexCache가 System Manager, BlueXP 또는 기본 CLI 인수를 사용하여 자동으로 프로비저닝됩니다. 그림 1그림 2 처음에는 같지만, 둘 다 4노드, 단일 구성 NAS 컨테이너입니다. 유일한 차이점은 그림 1의 NAS 컨테이너는 FlexGroup이고 그림 2의 NAS 컨테이너는 FlexCache입니다. 각 그림은 동일한 병목 현상을 보여 줍니다. 즉, 핫 파일에 대한 볼륨 선호도 액세스를 위한 노드 1의 CPU와 지연 시간을 유발하는 동부 서부 트래픽의 병목 현상을 보여 줍니다. 자동 프로비저닝된 FlexCache로 병목 현상이 사라지지는 않았습니다.

그림 **2:** 자동 프로비저닝 **FlexCache** 시나리오



FlexCache의 해부 구조

핫스팟 교정을 위한 FlexCache를 효과적으로 설계하려면 FlexCache에 대한 몇 가지 기술 세부 사항을 이해해야 합니다.

FlexCache는 항상 스파스 FlexGroup입니다. FlexGroup는 여러 개의 FlexVol으로 구성됩니다. 이러한 FlexVol을 FlexGroup 구성요소라고 합니다. 기본 FlexGroup 레이아웃의 경우, 클러스터의 노드당 구성요소가 하나 이상 있습니다. 구성 요소는 추상화 계층 아래에 "함께 결합"되어 단일 대형 NAS 컨테이너로 클라이언트에 제공됩니다. 파일이 FlexGroup에 기록되면 수집 휴리스틱에서 파일이 저장될 구성요소를 결정합니다. 클라이언트의 NAS 연결을 포함하는 구성요소이거나 다른 노드일 수 있습니다. 모든 것이 추상화 계층 아래에서 작동하고 클라이언트에게 보이지 않기 때문에 위치는 관련이 없습니다.

FlexGroup에 대한 이해도를 FlexCache에 적용해 보겠습니다. FlexCache는 FlexGroup을 기반으로 구축되기 때문에 에 설명된 대로 클러스터의 모든 노드에 대해 단일 FlexCache가 기본적으로 제공됩니다.그림 1 대부분의 경우 이것은 매우 유용합니다. 클러스터의 모든 리소스를 활용하고 있습니다.

하지만 핫 파일을 수정하는 경우에는 단일 파일 및 동서 트래픽의 CPU와 같은 두 가지 병목 현상 때문에 이 방법이 적합하지 않습니다. 핫 파일의 모든 노드에 구성 요소가 있는 FlexCache를 생성하는 경우 해당 파일은 여전히 구성요소 중 하나에 상주합니다. 즉, 핫 파일에 대한 모든 액세스를 서비스하는 CPU가 하나 있습니다. 또한 핫 파일에 도달하는 데 필요한 동서 트래픽의 양을 제한해야 합니다.

이 솔루션은 고밀도 FlexCaches의 어레이입니다.

고밀도 **FlexCache**의 해부 구조

고집적 HDF(FlexCache)는 캐시된 데이터의 용량 요구 사항이 허용하는 만큼 소수의 노드에 구성 요소가 있습니다. 목표는 단일 노드에서 캐시를 활성화하는 것입니다. 용량 요구사항에 따라 그렇게 할 수 없는 경우 몇 개의 노드에만 구성요소를 사용할 수 있습니다.

예를 들어, 24노드 클러스터에는 3개의 고밀도 FlexCh가 있을 수 있습니다.

- 노드 1에서 8까지 확장되는 경로입니다
- 노드 9에서 16까지 확장되는 초입니다
- 노드 17-24에 걸쳐 3분의 1을 제공합니다

이 세 개의 HDFS는 하나의 HDFA(High-Density FlexCache Array)를 구성합니다. 각 HDF 내에 파일이 균등하게 배포되면 클라이언트가 요청한 파일이 프런트엔드 NAS 접속에 로컬로 상주할 확률은 8%입니다. 각각 2개의 노드만 사용하는 12개의 HDFS를 사용하는 경우 파일이 로컬일 가능성이 50% 높습니다. HDF를 단일 노드로 축소하여 24개 노드로 만들 수 있다면 파일이 로컬임을 보증할 수 있습니다.

이 구성은 모든 동서 트래픽을 제거하며 가장 중요한 것은 핫 파일에 액세스하기 위한 24개의 CPU/볼륨 선호도를 제공합니다.

다음 단계

"FlexCache 어레이 밀도를 결정합니다"

관련 정보

"FlexGroup 및 TR에 대한 설명서"

ONTAP FlexCache 밀도 결정

첫 번째 핫스팟 개선 설계 결정은 FlexCache 밀도를 파악하는 것입니다. 다음 예는 4노드 클러스터입니다. 파일 수가 각 HDF의 모든 구성 요소에 균등하게 분포되어 있다고 가정합니다. 또한 모든 노드에 걸쳐 프런트엔드 NAS 연결이 고르게 분포되어 있다고 가정합니다.

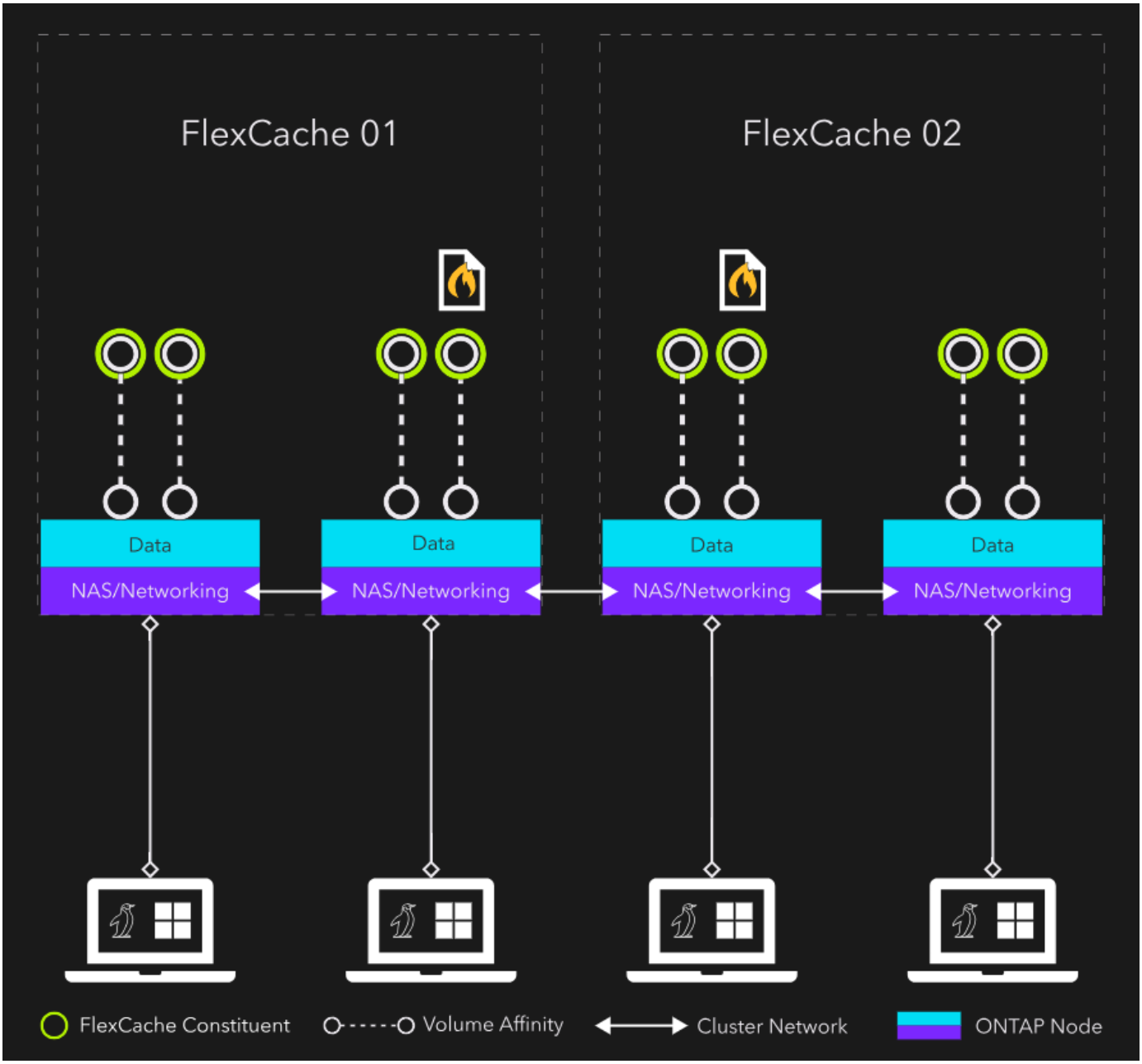
이러한 예만 사용할 수 있는 구성은 아니지만 공간 요구 사항과 사용 가능한 리소스가 허용하는 한 많은 HDFS를 만드는 기본 설계 원칙을 이해해야 합니다.

HDFA는 다음 구문을 사용하여 표현됩니다. `HDFs per HDFA x nodes per HDF x constituents per node per HDF`

2x2x2 HDFA 구성

그림 1 은 2x2x2x2 HDFA 구성의 예입니다. 두 개의 HDFS, 각각 두 개의 노드를 스패닝하고 각 노드는 두 개의 구성 볼륨을 포함합니다. 이 예에서 각 클라이언트는 핫 파일에 직접 액세스할 확률이 50%입니다. 네 고객 중 두 명은 동서 트래픽을 가지고 있습니다. 중요한 것은 이제 두 개의 HDFS가 존재한다는 것입니다. 즉, 핫 파일의 두 가지 캐시를 의미합니다. 현재 핫 파일에 대한 액세스를 처리하는 CPU/볼륨 선호도가 2개 있습니다.

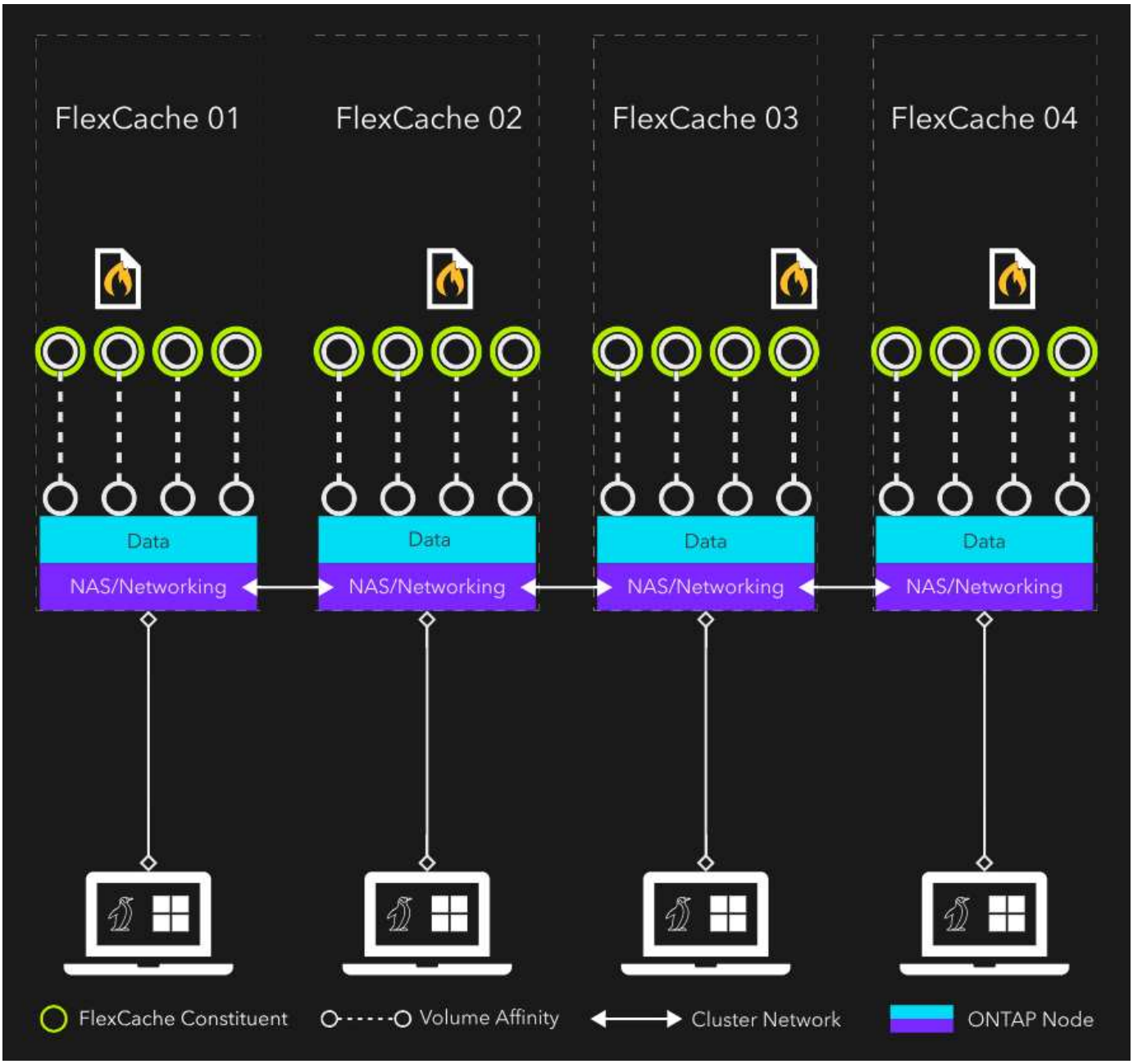
그림 **1: 2x2x2 HDFA** 구성



4x1x4 HDFA 구성

그림 2 최적의 구성을 나타냅니다. 4x1x4 HDFA 구성의 한 예로 4개의 HDFS가 단일 노드에 포함되고 각 노드는 4개의 구성 요소를 포함합니다. 이 예에서는 각 클라이언트가 핫 파일의 캐시에 직접 액세스할 수 있도록 보장합니다. 4개의 다른 노드에 4개의 캐시된 파일이 있으므로 4개의 서로 다른 CPU/볼륨 선호도가 핫 파일에 대한 서비스 액세스를 지원합니다. 또한 동서 트래픽이 생성되지 않습니다.

그림 **2: 4x1x4 HDFA** 구성



다음 단계

HDFS의 집적도를 결정한 후 NFS를 사용하여 HDFS에 액세스할 경우 다른 설계 결정을 내려야 "SVM 간 HDFA 및 SVM 내 HDFA"합니다.

ONTAP SVM 간 또는 SVM 내 HDFA 옵션을 결정합니다

HDFS의 밀도를 확인한 후 NFS를 사용하여 HDFS에 액세스할지 여부를 결정하고 SVM 간 HDFA 및 SVM 내 HDFA 옵션에 대해 알아봅니다.

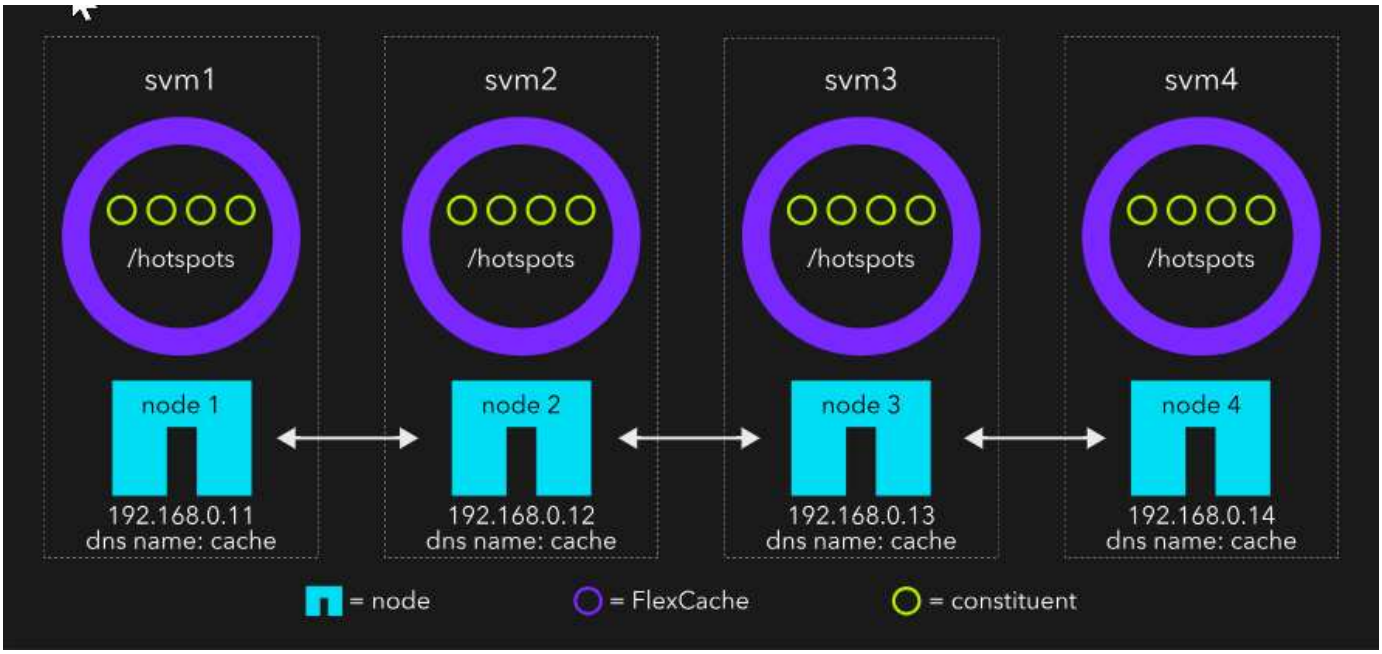
SMB 클라이언트만 HDFS에 액세스하는 경우 모든 HDFS를 단일 SVM에 생성해야 합니다. DFS 대상을 로드 밸런싱에 사용하는 방법은 Windows 클라이언트 구성을 참조하십시오.

SVM 간 HDFA 구축

SVM 간 HDFA는 HDFA의 각 HDF에 대해 SVM을 생성해야 합니다. 따라서 HDFA 내의 모든 HDFS가 동일한 접합 경로를 갖게 되므로 클라이언트 측에서 보다 쉽게 구성할 수 있습니다.

이 그림 1 예에서 각 HDF는 자체 SVM에 있습니다. 이는 SVM 간 HDFA 구축입니다. 각 HDF에는 /핫스팟의 접합 경로가 있습니다. 또한 모든 IP에는 호스트 이름 캐시의 DNS A 레코드가 있습니다. 이 구성은 DNS 라운드 로빈을 활용하여 다양한 HDFS에 대한 로드 밸런싱 마운트를 수행합니다.

그림 **1: 4x1x4 SVM** 간 **HDFA** 구성

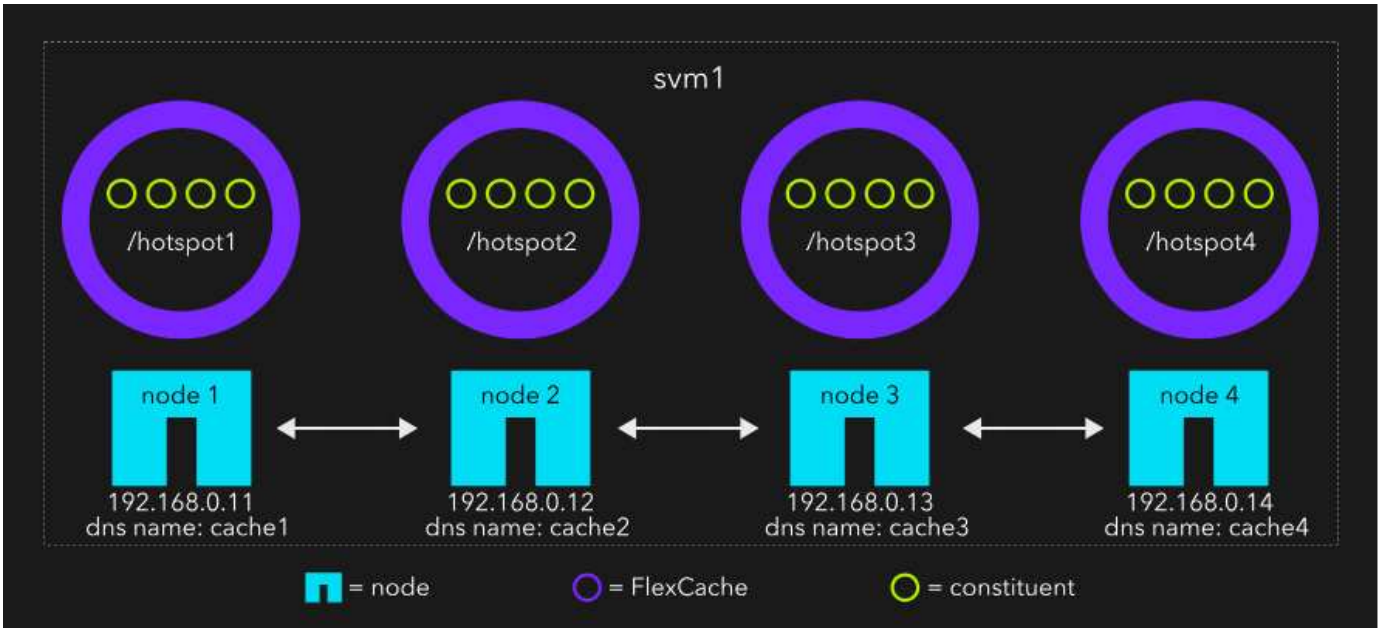


SVM HDFA 내부 구축

SVM 내부의 경우 각 HDF에 고유한 접합 경로가 필요하지만 모든 HDFS는 하나의 SVM에 있습니다. ONTAP에서는 하나의 SVM만 필요하기 때문에 이 설정은 더 쉬울 수 있지만 Linux 측면에서는 고급 구성이 필요하며 ONTAP에 데이터 LIF가 배치되어야 `autofs` 합니다.

이 그림 2 예에서 모든 HDF는 동일한 SVM에 있습니다. 이는 SVM 내 HDFA 구축이며 접합 경로가 고유해야 합니다. 로드 밸런싱이 제대로 작동하려면 각 IP에 대해 고유한 DNS 이름을 생성하고 호스트 이름이 확인되는 데이터 LIF를 HDF가 상주하는 노드에만 배치해야 합니다. 또한 에서 설명한 대로 여러 항목으로 를 구성해야 `autofs`"Linux 클라이언트 구성" 합니다.

그림 **2: 4x1x4** 내부 **SVM HDFA** 구성



다음 단계

이제 HDFA를 구축하는 방법에 대해 알아보았습니다."HDFA를 배포하고 클라이언트가 분산 방식으로 액세스할 수 있도록 구성합니다"

ONTAP에서 HDFA 및 데이터 LIF를 구성합니다

이 핫스팟 교정 솔루션의 이점을 실현하려면 HDFA 및 데이터 LIF를 적절히 구성해야 합니다. 이 솔루션은 동일한 클러스터에서 오리진 및 HDFA와 함께 클러스터 내 캐싱을 사용합니다.

다음은 두 가지 HDFA 샘플 구성입니다.

- 2x2x2 SVM 간 HDFA
- 4x1x4 내부 SVM HDFA

이 작업에 대해

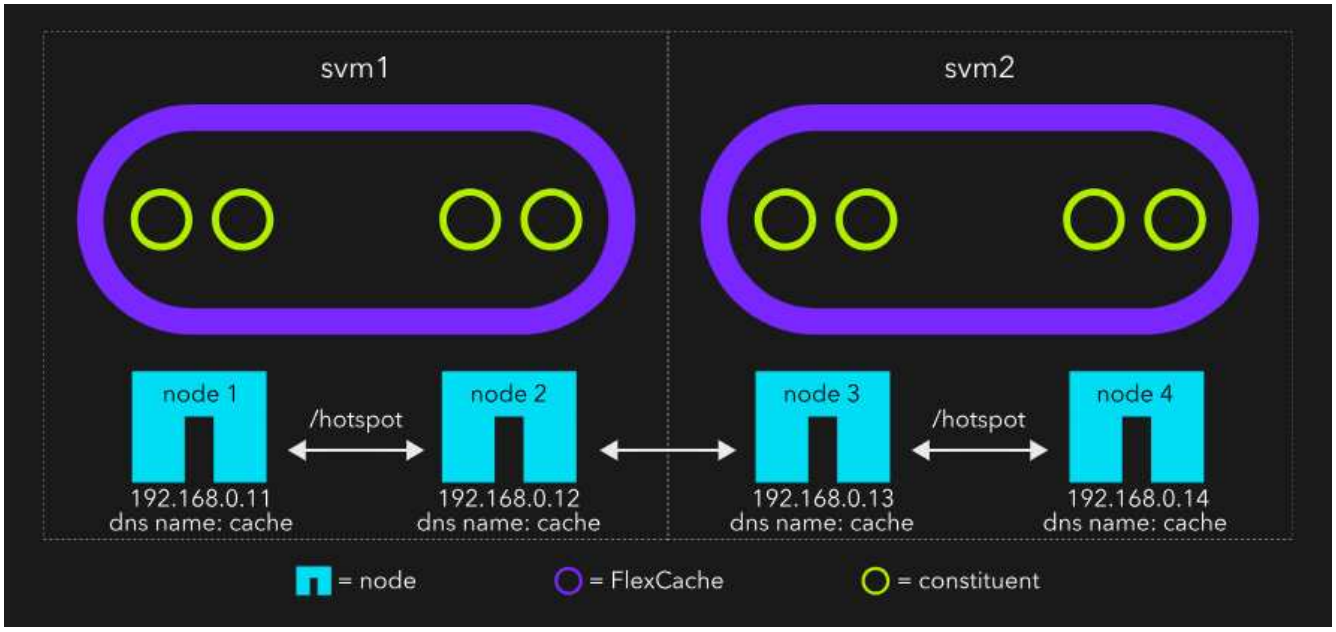
ONTAP CLI를 사용하여 이 고급 구성을 수행합니다. 명령에서 두 가지 구성을 사용해야 `flexcache create` 하며 한 가지 구성을 구성하지 않도록 해야 합니다.

- `-aggr-list`: HDF를 제한하려는 노드의 노드 또는 하위 집합에 상주하는 애그리게이트 또는 애그리게이트 목록을 제공합니다.
- `-aggr-list-multiplier`: 옵션에 나열된 애그리게이트당 생성할 구성 요소의 수를 `aggr-list` 결정합니다. 두 개의 애그리게이트가 나열되고 이 값을 로 설정하면 `2` 4개의 구성요소가 됩니다. NetApp에서는 애그리게이트당 최대 8개의 구성요소를 사용할 것을 권장하지만 16개로도 충분합니다.
- `-auto-provision-as`: 탭 아웃 경우, CLI는 자동 채우기를 시도하고 값을 로 `flexgroup`설정합니다. 이 설정이 구성되어 있지 않은지 확인합니다. 이 메시지가 나타나면 삭제합니다.

2x2x2 Inter-SVM HDFA 구성을 생성합니다

1. 그림 1과 같이 2x2x2x2 SVM 간 HDFA 구성을 지원하려면 준비 시트를 작성합니다.

그림 **1: 2x2x2 Inter-SVM HDFA** 레이아웃



SVM	**HDF**당 노드 수	애그리게이트	노드당 구성 요소	접합 경로	데이터 **LIF IP**
svm1를 참조하십시오	node1, node2	aggr1, aggr2	2	/핫스팟	192.168.0.11,192.168.0.12
svm2를 참조하십시오	node3, node4	aggr3, aggr4	2	/핫스팟	192.168.0.13,192.168.0.14

2. HDFS를 생성합니다. 준비 시트의 각 행에 대해 다음 명령을 두 번 실행합니다. 두 번째 반복의 및 `aggr-list` 값을 조정해야 `vserver` 합니다.

```
cache::> flexcache create -vserver svm1 -volume hotspot -aggr-list
aggr1,aggr2 -aggr-list-multiplier 2 -origin-volume <origin_vol> -origin
-vserver <origin_svm> -size <size> -junction-path /hotspot
```

3. 데이터 LIF를 생성합니다. 명령을 4회 실행하여 준비 시트에 나열된 노드에서 SVM당 데이터 LIF 두 개를 생성합니다. 각 반복에 맞게 값을 적절히 조정해야 합니다.

```
cache::> net int create -vserver svm1 -home-port e0a -home-node node1
-address 192.168.0.11 -netmask-length 24
```

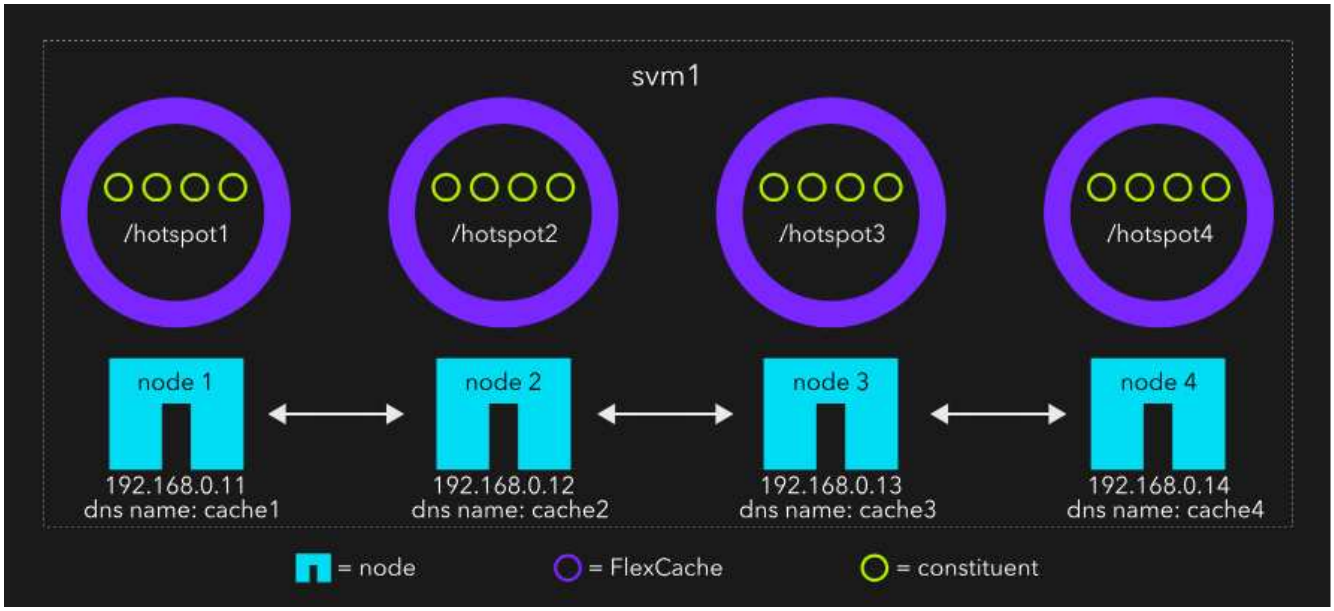
다음 단계

이제 HDFA를 적절히 활용할 수 있도록 클라이언트를 구성해야 합니다. 을 "클라이언트 구성"참조하십시오.

4x1x4 내부 SVM HDFA를 생성합니다

1. 그림 2와 같이 4x1x4 SVM 간 HDFA 구성을 지원하려면 준비 시트를 작성합니다.

그림 **2: 4x1x4** 내부 **SVM HDFA** 레이아웃



SVM	**HDF**당 노드 수	애그리게이트	노드당 구성 요소	접합 경로	데이터 **LIF IP**
svm1를 참조하십시오	노드1	aggr1를 참조하십시오	4	/hotspot1 을 참조하십시오	192.168.0.11
svm1를 참조하십시오	노드2	aggr2를 참조하십시오	4	/hotspot2 를 참조하십시오	192.168.0.12
svm1를 참조하십시오	node3를 참조하십시오	aggr3를 참조하십시오	4	/hotspot3 을 참조하십시오	192.168.0.13
svm1를 참조하십시오	node4를 참조하십시오	aggr4를 참조하십시오	4	/hotspot4 를 참조하십시오	192.168.0.14

2. HDFS를 생성합니다. 준비 시트의 각 행에 대해 다음 명령을 네 번 실행합니다. 각 반복에 대한 및 `junction-path` 값을 조정해야 `aggr-list` 합니다.

```
cache::> flexcache create -vserver svm1 -volume hotspot1 -aggr-list
aggr1 -aggr-list-multiplier 4 -origin-volume <origin_vol> -origin
-vserver <origin_svm> -size <size> -junction-path /hotspot1
```

3. 데이터 LIF를 생성합니다. 명령을 4회 실행하여 SVM에 총 4개의 데이터 LIF를 생성합니다. 노드당 데이터 LIF는 하나만 있어야 합니다. 각 반복에 맞게 값을 적절히 조정해야 합니다.

```
cache::> net int create -vserver svm1 -home-port e0a -home-node node1
-address 192.168.0.11 -netmask-length 24
```

다음 단계

이제 HDFA를 적절히 활용할 수 있도록 클라이언트를 구성해야 합니다. 을 "클라이언트 구성"참조하십시오.

ONTAP NAS 연결을 분산하도록 클라이언트를 구성합니다

핫스팟을 해결하려면 CPU 병목 현상을 방지하기 위해 클라이언트를 적절히 구성합니다.

Linux 클라이언트 구성

SVM 내 또는 SVM 간 HDFA 구축을 선택하든 Linux에서 를 사용하여 클라이언트가 서로 다른 HDFS 간에 로드 밸런싱을 수행할 수 있어야 `autofs` 합니다. `autofs`구성은 SVM 간 및 내부 SVM에 따라 다릅니다.

시작하기 전에

적절한 종속성이 설치되어 있어야 `autofs` 합니다. 이에 대한 자세한 내용은 Linux 설명서를 참조하십시오.

이 작업에 대해

설명된 단계에서는 다음 항목이 포함된 예제 `/etc/auto_master` 파일을 사용합니다.

```
/flexcache auto_hotspot
```

`autofs` `/etc`프로세스가 디렉터리에 액세스하려고 할 때마다 디렉터리에서 `/flexcache` 호출된 파일을 찾도록 `auto_hotspot` 구성합니다. 파일의 내용은 `auto_hotspot` 디렉토리 내에 마운트할 `NFS` 서버 및 접속 경로를 `/flexcache` 지정합니다. 설명된 예는 파일에 대한 서로 다른 `auto_hotspot` 설정입니다.

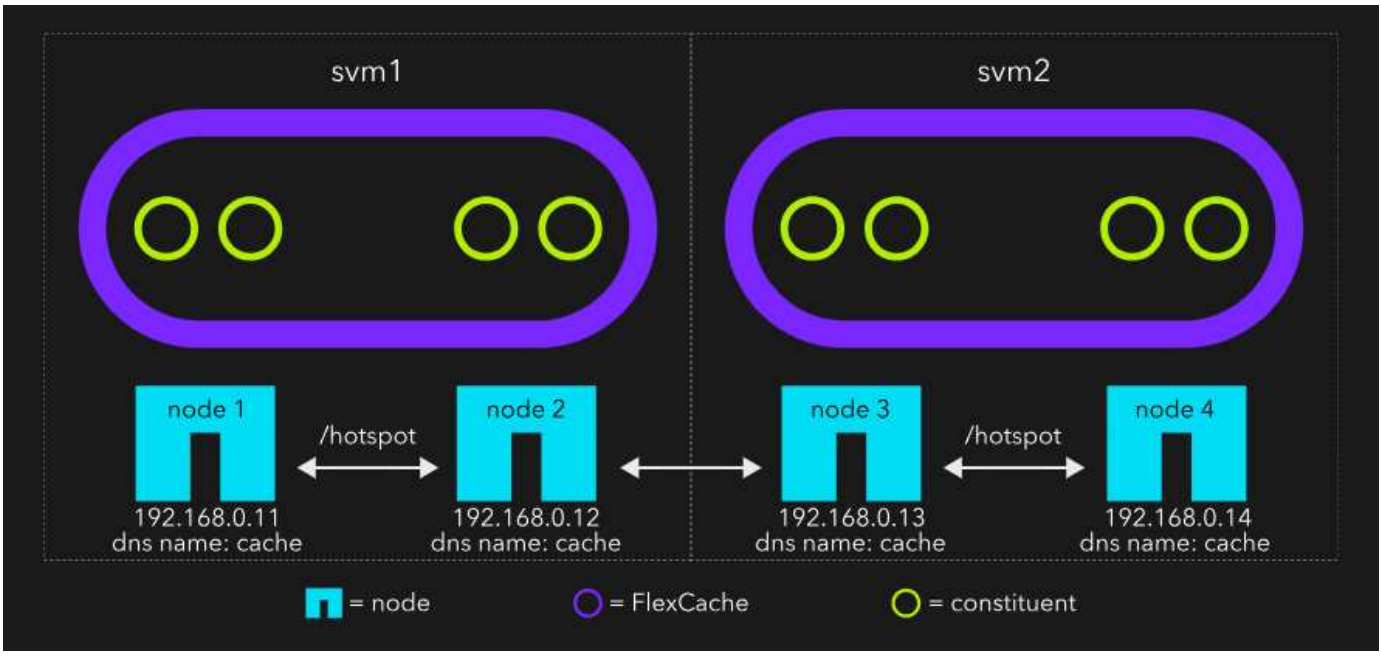
내부 **SVM HDFA** 자동 전송 구성

다음 예제에서는 의 다이어그램에 대한 지도를 만듭니다 `autofs`그림 1. 각 캐시에는 동일한 접합 경로가 있고 호스트 이름에 4개의 DNS A 레코드가 있으므로 `cache` 한 줄만 있으면 됩니다.

```
hotspot cache:/hotspot
```

이 간단한 한 줄은 NFS 클라이언트가 호스트 이름에 대한 DNS 조회를 `cache` 수행하도록 합니다. DNS가 라운드 로빈 방식으로 IP를 반환하도록 설정됩니다. 이렇게 하면 프런트엔드 NAS 연결이 고르게 분산됩니다. 클라이언트는 IP를 수신한 후 에서 `/flexcache/hotspot` 접합 경로를 마운트합니다 `/hotspot`. SVM1, SVM2, SVM3 또는 SVM4에 연결할 수 있지만 특정 SVM은 중요하지 않습니다.

그림 **1: 2x2x2 SVM** 간 **HDFA**



내부 **SVM HDFA** 자동 전송 구성

다음 예제에서는 의 다이어그램에 대한 지도를 만듭니다 `autofs`그림 2. NFS 클라이언트가 HDF 접합 경로 배포의 일부인 IP를 마운트해야 합니다. 즉, IP 192.168.0.11이 아닌 다른 것을 탑재하고 싶지 않습니다. `/hotspot1` 이를 위해 맵에서 하나의 로컬 마운트 위치에 대한 IP/접합 경로 쌍 4개를 모두 나열할 수 `auto_hotspot` 있습니다.

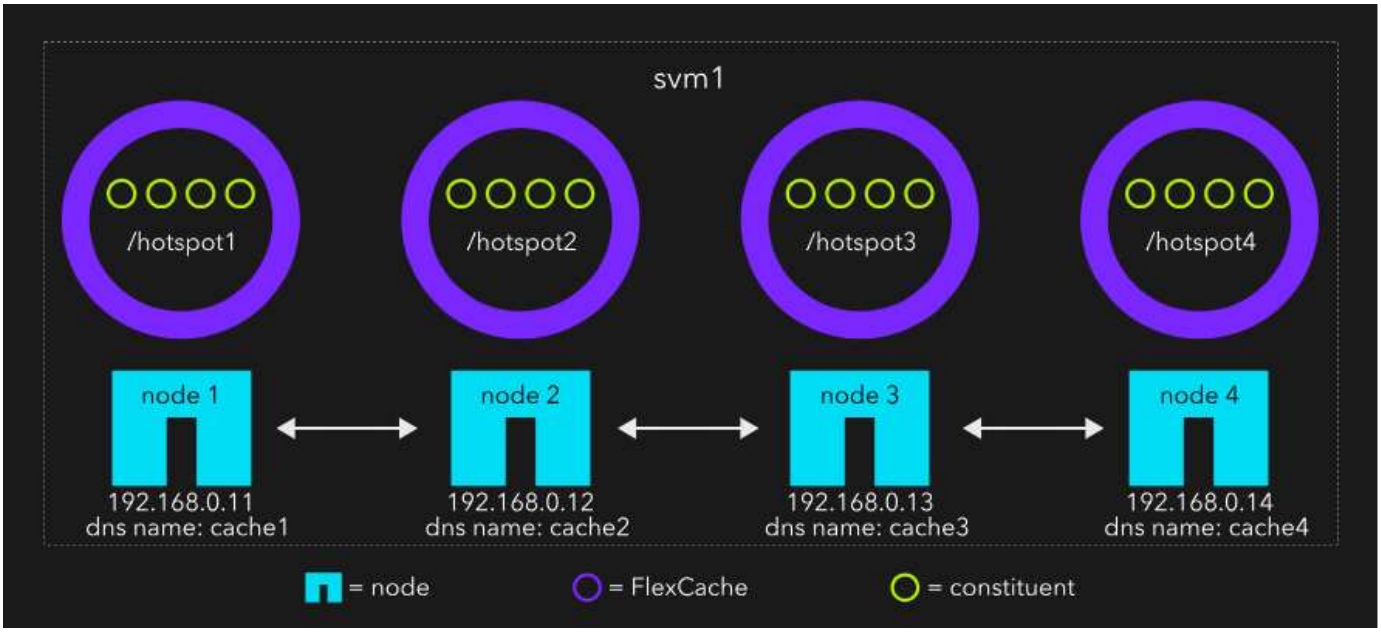
(`\`다음 예제의 백슬래시)는 항목을 다음 줄로 이어가며 읽기 쉽도록 합니다.

```
hotspot     cache1:/hostspot1 \
            cache2:/hostspot2 \
            cache3:/hostspot3 \
            cache4:/hostspot4
```

클라이언트가 액세스를 시도할 때 `/flexcache/hotspot` 은 `autofs` 네 개의 호스트 이름 모두에 대해 정방향 조회를 수행합니다. 4개의 IP가 모두 클라이언트와 동일한 서브넷에 있거나 다른 서브넷에 있다고 가정하면 는 `autofs` 각 IP에 대해 NFS NULL ping을 실행합니다.

이 NULL ping을 수행하려면 ONTAP의 NFS 서비스에서 패킷을 처리해야 하지만 디스크 액세스가 필요하지 않습니다. 반환할 첫 번째 ping은 마운트하도록 선택한 IP 및 접합 경로입니다. `autofs`

그림 **2: 4x1x4** 내부 **SVM HDFA**



Windows 클라이언트 구성

Windows 클라이언트에서는 내부 SVM HDFA를 사용해야 합니다. SVM의 여러 HDFS 간에 로드 밸런싱을 수행하려면 각 HDF에 고유한 공유 이름을 추가해야 합니다. 그런 다음 의 단계를 따라 "Microsoft 설명서"동일한 폴더에 대해 여러 DFS 대상을 구현합니다.

저작권 정보

상표 정보