검증된 아키텍처를 사용합니다

BeeGFS on NetApp with E-Series Storage

NetApp
June 18, 2024

목차

검증된 아키텍처를 사용합니다

개요 및 요구 사항

솔루션 개요

BeeGFS on NetApp 솔루션은 BeeGFS 병렬 파일 시스템을 NetApp EF600 스토리지 시스템과 결합하여 까다로운 워크로드에 대응할 수 있는 안정적이고 확장 가능하며 비용 효율적인 인프라를 제공합니다.

이 설계에서는 최신 엔터프라이즈 서버 및 스토리지 하드웨어와 네트워크 속도를 통해 제공되는 성능 밀도와 듀얼 AMD EPYC 7003 “Milan” 프로세서를 갖춘 파일 노드 및 200GB(HDR) InfiniBand를 사용하는 직접 연결 PCIe 4.0을 지원하는 NVMe/IB 프로토콜을 사용하는 엔드 투 엔드 NVMe 및 NVMeOF를 제공하는 블록 노드 필요.

NVA 프로그램

NetApp 솔루션의 BeeGFS는 NVA(NetApp Verified Architecture) 프로그램에 포함되어 있으며, 특정 워크로드 및 사용 사례에 대한 참조 구성 및 사이징 지침을 고객에게 제공합니다. NVA 솔루션은 구축 위험을 최소화하고 시장 출시 기간을 단축할 수 있도록 철저한 테스트와 설계를 거쳤습니다.

사용 사례

NetApp 기반 BeeGFS 솔루션에는 다음 사용 사례가 적용됩니다.

- 머신 러닝(ML), 딥 러닝(DL), 대규모 자연어 처리(NLP), 자연어 이해(NLU)를 비롯한 인공 지능(AI) 자세한 내용은 을 참조하십시오 "AI용 BeeGFS: 팩션과 픽션 비교".
- MPI(메시지 전달 인터페이스) 및 기타 분산 컴퓨팅 기술에 의해 가속되는 응용 프로그램을 포함한 고성능 컴퓨팅(HPC). 자세한 내용은 을 참조하십시오 "BeeGFS가 HPC를 넘어서는 이유".
- 애플리케이션 워크로드의 특징:
  - 1GB 이상의 파일을 읽거나 쓰는 중입니다
  - 여러 클라이언트(10s, 100s 및 1000)에서 동일한 파일을 읽거나 쓰는 경우
- 테라바이트급 또는 페타바이트급의 데이터 세트
- 크기가 큰 파일과 작은 파일을 혼합하여 사용할 수 있도록 최적화되는 단일 스토리지 네임스페이스가 필요한 환경

이점

NetApp에서 BeeGFS를 사용할 때의 주요 이점은 다음과 같습니다.

- 검증된 하드웨어 설계를 통해 하드웨어 및 소프트웨어 구성요소를 완벽하게 통합하여 예측 가능한 성능과 안정성을 보장합니다.
- Ansible을 사용한 구축 및 관리로 단순성과 일관성 확보
- E-Series Performance Analyzer 및 BeeGFS 플러그인을 사용하여 제공되는 모니터링 및 관찰 가능성 자세한 내용은 을 참조하십시오 "NetApp E-Series 솔루션을 모니터링하는 프레임워크 소개".
- 데이터 내구성과 가용성을 제공하는 공유 디스크 아키텍처를 갖춘 고가용성

- 컨테이너 및 Kubernetes를 사용하여 최신 워크로드 관리 및 오케스트레이션 지원 자세한 내용은 을 참조하십시오 "Kubernetes에서 BeeGFS를 만나 보십시오. 미래 지향형 투자 이야기입니다".

HA 아키텍처

NetApp 기반의 BeeGFS는 NetApp 하드웨어로 완벽하게 통합된 솔루션을 생성하여 공유 디스크 HA(고가용성) 아키텍처를 지원하여 BeeGFS 엔터프라이즈 에디션의 기능을 확장합니다.

> BeeGFS 커뮤니티 에디션은 무료로 사용할 수 있지만, 이 엔터프라이즈 에디션은 NetApp과 같은 파트너로부터 프로페셔널 지원 구독 계약을 구매해야 합니다. Enterprise Edition에서는 복원력, 할당량 적용 및 스토리지 풀을 비롯한 몇 가지 추가 기능을 사용할 수 있습니다.

다음 그림에서는 공유 안 함 및 공유 디스크 HA 아키텍처를 비교하여 보여 줍니다.

[]

자세한 내용은 을 참조하십시오 "NetApp에서 지원하는 BeeGFS에 대한 고가용성 발표".

Ansible

NetApp 기반 BeeGFS는 GitHub 및 Ansible Galaxy(BeeGFS 컬렉션)에서 호스팅되는 Ansible 자동화를 통해 제공 및 구축됩니다 "Ansible 갤럭시" 및 "NetApp의 E-Series GitHub를 참조하십시오")를 클릭합니다. Ansible은 BeeGFS 구성 요소를 조립하는 데 사용되는 하드웨어에서 주로 테스트되지만, 지원되는 Linux 배포를 사용하여 거의 모든 x86 기반 서버에서 실행되도록 구성할 수 있습니다.

자세한 내용은 을 참조하십시오 "E-Series 스토리지를 통해 BeeGFS 구축".

디자인 세대

BeeGFS on NetApp 솔루션은 현재 세대 간 설계의 2세대입니다.

1세대 및 2세대 모두에는 BeeGFS 파일 시스템과 NVMe EF600 스토리지 시스템을 통합한 기본 아키텍처가 포함되어 있습니다. 그러나 2세대 제품에는 다음과 같은 추가적인 이점이 포함되어 있습니다.

- 2U 랙 공간만 추가하여 성능과 용량을 두 배로 향상
- 공유 디스크, 2계층 하드웨어 설계를 기반으로 한 고가용성(HA
- NVIDIA DGX A100 SuperPOD 및 NVIDIA BasePOD 아키텍처에 대한 외부 검증

2세대 설계

NetApp 기반의 2세대 BeeGFS는 HPC(고성능 컴퓨팅) 및 HPC 스타일 머신 러닝(ML), 딥 러닝(DL), 유사한 인공 지능(AI) 기술 등 까다로운 워크로드의 성능 요구사항을 충족하도록 최적화되어 있습니다. BeeGFS on NetApp 솔루션은 공유 디스크 HA(고가용성) 아키텍처를 통합하여 워크로드 및 사용 사례에 맞게 확장 가능한 스토리지를 찾듯이 다운타임 또는 데이터 손실을 허용할 수 없는 기업 및 기타 조직의 데이터 내구성 및 가용성 요구사항을 충족합니다. 이 솔루션은 NetApp에서 검증되었을 뿐만 아니라 NVIDIA DGX SuperPOD 및 DGX BasePOD의 스토리지 옵션으로 외부 인증도 받았습니다.

최초의 세대별 설계

NetApp 기반의 1세대 BeeGFS는 NetApp EF600 NVMe 스토리지 시스템, BeeGFS 병렬 파일 시스템, NVIDIA DGX

™ A100 시스템 및 NVIDIA ® Mellanox ® Quantum ™ QM8700 200Gbps IB 스위치를 사용하는 머신 러닝(ML) 및 인공 지능(AI) 워크로드용으로 설계되었습니다. 또한, 이 설계에서는 스토리지 및 컴퓨팅 클러스터 인터커넥트 패브릭을 위한 200Gbps InfiniBand(IB)를 사용하여 고성능 워크로드를 위한 완벽한 IB 기반 아키텍처를 제공합니다.

1세대 제품에 대한 자세한 내용은 를 참조하십시오 "NVIDIA DGX A100 Systems 및 BeeGFS를 지원하는 NetApp EF-Series AI".

아키텍처 개요

BeeGFS on NetApp 솔루션에는 검증된 워크로드를 지원하는 데 필요한 특정 장비, 케이블링, 구성을 결정하는 데 사용되는 아키텍처 설계 고려사항이 포함되어 있습니다.

빌딩 블록 아키텍처

스토리지 요구 사항에 따라 BeeGFS 파일 시스템을 다양한 방식으로 구축 및 확장할 수 있습니다. 예를 들어, 주로 작은 파일을 많이 사용하는 사용 사례에서는 메타데이터 성능과 용량이 더 큰 반면, 큰 파일이 적은 사용 사례에서는 실제 파일 콘텐츠에 대해 더 많은 스토리지 용량과 성능을 선호할 수 있습니다. 이러한 여러 가지 고려 사항은 병렬 파일 시스템 구축의 여러 가지 차원을 영향을 주므로 파일 시스템을 설계하고 구축하는 작업이 더 복잡해집니다.

이러한 문제를 해결하기 위해 NetApp은 이러한 각 차원을 확장하는 데 사용되는 표준 구성 요소 아키텍처를 설계했습니다. 일반적으로 BeeGFS 빌딩 블록은 다음 세 가지 구성 프로파일 중 하나로 구축됩니다.

- BeeGFS 관리, 메타데이터 및 스토리지 서비스를 포함한 단일 기본 구성 요소입니다
- BeeGFS 메타데이터 및 스토리지 구성 요소입니다
- BeeGFS 스토리지 전용 구성 요소입니다

이 세 가지 옵션 간의 하드웨어 변경 사항은 BeeGFS 메타데이터에 더 작은 드라이브를 사용하는 것입니다. 그렇지 않으면 모든 구성 변경 사항이 소프트웨어를 통해 적용됩니다. 또한 Ansible을 구축 엔진으로 사용하면 특정 구성 요소에 대해 원하는 프로필을 설정하여 구성 작업을 간단하게 수행할 수 있습니다.

자세한 내용은 을 참조하십시오 검증된 하드웨어 설계.

파일 시스템 서비스

BeeGFS 파일 시스템에는 다음과 같은 주요 서비스가 포함됩니다.

- 관리 서비스.* 다른 모든 서비스를 등록하고 모니터링합니다.
- * 스토리지 서비스. * 데이터 청크 파일이라고 하는 분산 사용자 파일 콘텐츠를 저장합니다.
- * 메타데이터 서비스. * 파일 시스템 레이아웃, 디렉토리, 파일 특성 등을 추적합니다.
- * 클라이언트 서비스. * 파일 시스템을 마운트하여 저장된 데이터에 액세스합니다.

다음 그림에서는 NetApp E-Series 시스템에 사용된 BeeGFS 솔루션 구성 요소 및 관계를 보여 줍니다.

[]

BeeGFS는 병렬 파일 시스템으로서 여러 서버 노드에서 파일을 스트라이핑하여 읽기/쓰기 성능과 확장성을 극대화합니다. 서버 노드는 함께 작동하여 일반적으로 _clients_라고 하는 다른 서버 노드에서 동시에 마운트하고 액세스할 수 있는 단일 파일 시스템을 제공합니다. 이러한 클라이언트는 NTFS, XFS 또는 ext4와 같은 로컬 파일 시스템과 유사하게 분산 파일 시스템을 보고 사용할 수 있습니다.

지원되는 다양한 Linux 배포판에서 4개의 기본 서비스를 실행하고 InfiniBand(IB), OPA(Omni-Path), RoCE(RDMA over Converged Ethernet)를 비롯한 TCP/IP 또는 RDMA 지원 네트워크를 통해 통신합니다. BeeGFS 서버 서비스(관리, 스토리지 및 메타데이터)는 사용자 공간 데몬이며, 클라이언트는 네이티브 커널 모듈(패치리스)입니다. 모든 구성요소를 재부팅하지 않고 설치 또는 업데이트할 수 있으며 동일한 노드에서 서비스 조합을 실행할 수 있습니다.

검증된 노드

NetApp 기반 BeeGFS 솔루션에는 NetApp EF600 스토리지 시스템(블록 노드) 및 Lenovo ThinkSystem SR665 서버(파일 노드)의 검증된 노드가 포함됩니다.

블록 노드: **EF600** 스토리지 시스템

NetApp EF600 All-Flash 어레이는 일관된 거의 실시간에 가까운 데이터 액세스를 제공하는 동시에 모든 워크로드를 동시에 지원합니다. AI 및 HPC 애플리케이션에 데이터를 지속적으로 신속하게 제공하기 위해 EF600 스토리지 시스템은 하나의 엔클로저에 최대 2백만 개의 캐시된 읽기 IOPS, 100마이크로초 미만의 응답 시간, 42GBps 순차적 읽기 대역폭을 제공합니다.

파일 노드: **Lenovo ThinkSystem SR665** 서버

SR665는 PCIe 4.0을 지원하는 2소켓 2U 서버입니다. 이 솔루션의 요구사항을 충족하도록 구성하면 직접 연결된 E-Series 노드에서 제공하는 처리량 및 IOP와 잘 어울리도록 구성에서 BeeGFS 파일 서비스를 실행할 수 있는 충분한 성능을 제공합니다.

Lenovo SR665에 대한 자세한 내용은 를 참조하십시오 "Lenovo 웹 사이트".

검증된 하드웨어 설계

다음 그림에 나와 있는 솔루션의 구성 요소는 BeeGFS 파일 계층에 2개의 듀얼 소켓 PCIe 4.0 지원 서버를 사용하고 블록 계층으로 2개의 EF600 스토리지 시스템을 사용합니다.

[]

> 각 구성 요소에 BeeGFS 파일 노드가 2개 포함되어 있으므로 페일오버 클러스터에 쿼럼을 설정하려면 최소 2개의 구성 요소가 필요합니다. 2노드 클러스터를 구성할 수 있지만 이 구성에는 페일오버가 발생하지 않도록 할 수 있는 제한이 있습니다. 2노드 클러스터가 필요한 경우 세 번째 장치를 Tiebreaker로 통합할 수 있습니다(단, 이 설계에서는 다루지 않음).

각 구성 요소는 파일 및 블록 계층에 대한 장애 도메인을 분리하는 2계층 하드웨어 설계를 통해 고가용성을 제공합니다. 각 계층은 독립적으로 페일오버될 수 있으므로 복원력을 높이고 계단식 고장 위험을 줄일 수 있습니다. HDR InfiniBand를 NVMeOF와 함께 사용하면 전체 이중화 및 충분한 링크 초과 할당으로 파일 노드와 블록 노드 간에 높은 처리량과 최소 지연 시간을 제공하여 시스템이 부분적으로 성능 저하 상태일 때도 분리되는 설계를 병목 현상이 발생하지 않습니다.

BeeGFS on NetApp 솔루션은 구축 환경의 모든 구성 요소에서 실행됩니다. 구축된 첫 번째 구성 요소는 BeeGFS 관리, 메타데이터 및 스토리지 서비스(기본 구성 요소라고도 함)를 실행해야 합니다. 이후의 모든 구성 요소는 소프트웨어를 통해 BeeGFS 메타데이터 및 스토리지 서비스를 실행하거나 스토리지 서비스만 실행하도록 구성됩니다. 각 구성 요소에 서로 다른 구성 프로필을 사용할 수 있으므로 동일한 기본 하드웨어 플랫폼 및 구성 요소 설계를 사용하여 파일 시스템 메타데이터 또는 스토리지 용량과 성능을 확장할 수 있습니다.

최대 5개의 빌딩 블록이 독립 실행형 Linux HA 클러스터에 결합되어 클러스터 리소스 관리자(페이스 메이커)당 적절한 리소스 수를 확보하고 클러스터 구성원을 동기화 상태로 유지하는 데 필요한 메시징 오버헤드를 줄입니다(Corosync). 충분한 구성원이 쿼럼을 설정할 수 있도록 클러스터당 최소 2개의 빌딩 블록을 사용하는 것이 좋습니다. 이러한 독립 실행형 BeeGFS HA 클러스터 중 하나 이상이 결합되어 클라이언트가 단일 스토리지 네임스페이스로 액세스할 수 있는

BeeGFS 파일 시스템(다음 그림에 표시)을 생성합니다.

[]

궁극적으로 랙당 구성 요소의 수는 해당 사이트의 전력 및 냉각 요구 사항에 따라 달라지지만, 이 솔루션은 스토리지 /데이터 네트워크에 사용되는 2개의 1U InfiniBand 스위치를 위한 공간을 제공하면서 단일 42U 랙에 최대 5개의 구성 요소를 구축할 수 있도록 설계되었습니다. 각 구성 요소마다 8개의 IB 포트(이중화를 위해 스위치당 4개)가 필요하므로, 5개의 구성 요소는 40포트 HDR InfiniBand 스위치(예: NVIDIA QM8700)에 포트의 절반을 남겨 FAT 트리 또는 이와 유사한 비차단 토폴로지를 구현할 수 있습니다. 이렇게 구성하면 네트워킹 병목 현상 없이 스토리지 또는 컴퓨팅/GPU 랙 수를 확장할 수 있습니다. 필요에 따라 스토리지 패브릭 공급업체의 권장 사항에서 초과 할당된 스토리지 패브릭을 사용할 수 있습니다.

다음 이미지는 80노드 FAT 트리 토폴로지를 보여줍니다.

[]

Ansible을 NetApp 기반의 BeeGFS 구축을 위한 배포 엔진으로 사용하여 관리자는 최신 인프라를 코드 사례로 사용하여 전체 환경을 유지할 수 있습니다. 이렇게 하면 복잡한 시스템이 될 수 있는 일이 크게 단순화되어 관리자가 한 곳에서 구성을 정의 및 조정한 다음, 환경 확장규모에 관계없이 일관되게 적용할 수 있습니다. BeeGFS 컬렉션은 에서 구할 수 있습니다 "Ansible 갤럭시" 및 "NetApp의 E-Series GitHub를 참조하십시오".

기술 요구사항

NetApp 기반의 BeeGFS 솔루션을 구축하려면 해당 환경이 기술 요구사항을 충족하는지 확인하십시오.

하드웨어 요구 사항

다음 표에는 NetApp 기반 BeeGFS 솔루션의 단일 2세대 구성 요소 설계를 구현하는 데 필요한 하드웨어 구성요소가 나와 있습니다.

> 이 솔루션을 구체적으로 구축하는 데 사용되는 하드웨어 구성요소는 고객 요구사항에 따라 다를 수 있습니다.

카운트	하드웨어 구성 요소	요구 사항
2	BeeGFS 파일 노드.	각 파일 노드는 예상 성능을 달성하기 위해 다음 구성을 충족하거나 초과해야 합니다. 프로세서: * • 2x AMD EPYC 7343 16C 3.2GHz. • 2개의 NUMA 존으로 구성됩니다. • 메모리: * • 256GB • 16x 16GB TruDDR4 3200MHz(2Rx8 1.2V) RDIMM-A(더 적은 수의 큰 DIMM에 비해 더 작은 DIMM 선호). • 메모리 대역폭을 최대화하도록 채워집니다. • PCIe 확장: PCE Gen4 x16 슬롯 4개: * • NUMA 존당 2개의 슬롯 • 각 슬롯은 Mellanox MCX653106A-HDAT 어댑터에 충분한 전력/냉각을 제공해야 합니다. • 기타: * • OS용 RAID 1에 구성된 1TB 7.2K SATA 드라이브 2개(또는 동급 이상) • 대역 내 OS 관리를 위한 1GbE(또는 이상) 포트 • Out-of-Band Server Management용 Redfish API가 포함된 1GbE BMC • 이중 핫 스왑 전원 공급 장치 및 성능 팬 • 스토리지 InfiniBand 스위치에 연결하는 데 필요한 경우 Mellanox 광 InfiniBand 케이블을 지원해야 합니다. • Lenovo SR665: * • 사용자 지정 NetApp 모델에는 이중 포트 Mellanox ConnectX-6 어댑터를 지원하는 데 필요한 XClarity 컨트롤러 펌웨어의 필수 버전이 포함되어 있습니다. 주문 정보는 NetApp에 문의하십시오.
8	Mellanox ConnectX-6 HCA(파일 노드용)	• MCX653106A-HDAT 호스트 채널 어댑터(HDR IB 200GB, 이중 포트 QSFP 56, PCIe4.0 x16)
8	1m의 HDR InfiniBand 케이블(파일/블록 노드 직접 연결용)	• MCP1650-H001E30(1m Mellanox Passive Copper 케이블, IB HDR, 최대 200Gbps, QSFP 56, 30AWG). 필요한 경우 파일 노드와 블록 노드 간의 더 긴 거리를 고려하여 길이를 조정할 수 있습니다.

카운트	하드웨어 구성 요소	요구 사항
8	HDR InfiniBand 케이블(파일 노드 /스토리지 스위치 연결용)	파일 노드를 스토리지 리프 스위치에 연결하려면 적절한 길이의 InfiniBand HDR 케이블(QSFP 56 트랜시버)이 필요합니다. 가능한 옵션은 다음과 같습니다. • MCP1650-H002E26(2m Mellanox Passive Copper 케이블, IB HDR, 최대 200GB/s, QSFP 56, 30AWG). • MFS1S00-H003E(3m Mellanox 활성 파이버 케이블, IB HDR, 최대 200GB/s, QSFP 56).
2	E-Series 블록 노드	EF600 컨트롤러 2개는 다음과 같이 구성됩니다. • 메모리: 256GB(컨트롤러당 128GB) • 어댑터: 2포트 200GB/HDR(NVMe/IB) • 드라이브: 원하는 용량과 일치하도록 구성됨

소프트웨어 요구 사항

예측 가능한 성능 및 안정성을 위해 NetApp 기반 BeeGFS 솔루션의 릴리즈는 솔루션 구축에 필요한 소프트웨어 구성 요소의 특정 버전을 사용하여 테스트됩니다.

소프트웨어 배포 요구 사항

다음 표에는 Ansible 기반 BeeGFS 구축의 일부로 자동 구축되는 소프트웨어 요구사항이 나와 있습니다.

소프트웨어	버전
BeeGFS	7.2.6
Corosync 를 참조하십시오	3.1.5-1
심장박동기	2.1.0-8
OpenSM을 참조하십시오	OpenSM-5.9.0(mlnx_OFED 5.4-1.0.3.0부터) 가상화를 활성화하기 위해 직접 연결에만 필요합니다.

Ansible 제어 노드 요구사항

NetApp 기반 BeeGFS 솔루션은 Ansible 제어 노드에서 구축 및 관리됩니다. 자세한 내용은 를 참조하십시오 "Ansible 설명서".

다음 표에 나와 있는 소프트웨어 요구사항은 아래 나열된 NetApp BeeGFS Ansible 컬렉션 버전과 관련이 있습니다.

소프트웨어	버전
Ansible	2.11 PIP를 통해 설치된 경우: Ansible-4.7.0 및 Ansible-Core<2.12,>=2.11.6
파이썬	3.9

소프트웨어	버전
추가 Python 패키지	암호화 - 35.0.0, netaddr-0.8.0
BeeGFS Ansible 컬렉션	3.0.0

파일 노드 요구 사항

소프트웨어	버전
RedHat Enterprise Linux	RedHat 8.4 서버의 물리적 및 고가용성(2 소켓). 파일 노드에는 유효한 RedHat Enterprise Linux Server 서브스크립션과 Red Hat Enterprise Linux 고가용성 애드온이 필요합니다.
Linux 커널	4.18.0-305.25.1.el8_4.x86_64
InfiniBand/RDMA 드라이버	받은 편지함
ConnectX-6 HCA 펌웨어	FW: 20.31.1014
PXE: 3.6.0403	UEFI: 14.24.0013

EF600 블록 노드 요구사항

소프트웨어	버전
SANtricity OS를 참조하십시오	11.70.2
NVSRAM	N6000-872834-D06.DLP
드라이브 펌웨어	사용 중인 드라이브 모델에 대한 최신 버전입니다.

추가 요구 사항

다음 표에 나열된 장비가 검증에 사용되었지만 필요에 따라 적절한 대안을 사용할 수 있습니다. 일반적으로 예기치 않은 문제를 방지하려면 최신 소프트웨어 버전을 실행하는 것이 좋습니다.

하드웨어 구성 요소	설치된 소프트웨어
• 2x Mellanox MQM8700 200GB InfiniBand 스위치	• 펌웨어 3.9.2110
• 1x Ansible 제어 노드(가상화): * • 프로세서: 인텔® 제온® 골드 6146 CPU @ 3.20GHz • 메모리: 8GB • 로컬 스토리지: 24GB	• CentOS Linux 8.4.2105 • 커널 4.18.0-305.3.1.el8.x86_64 설치된 Ansible 및 Python 버전이 위 표의 버전과 일치합니다.

하드웨어 구성 요소	설치된 소프트웨어
• 10x BeeGFS 클라이언트(CPU 노드): * • 프로세서: 1x AMD EPYC 7302 16코어 CPU, 3.0GHz • 메모리: 128GB • 네트워크: 2x Mellanox MCX653106A-HDAT(어댑터당 하나의 포트 연결).	• Ubuntu 20.04 • 커널: 5.4.0-100 - 일반 • InfiniBand 드라이버: Mellanox OFED 5.4-1.0.3.0
• 1x BeeGFS 클라이언트(GPU 노드): * • 프로세서: 2.25GHz에서 AMD EPYC 7742 64코어 CPU 2개 • 메모리: 1TB • 네트워크: 2x Mellanox MCX653106A-HDAT(어댑터당 하나의 포트 연결). 이 시스템은 NVIDIAs HGX A100 플랫폼을 기반으로 하며 4개의 A100 GPU를 포함합니다.	• Ubuntu 20.04 • 커널: 5.4.0-100 - 일반 • InfiniBand 드라이버: Mellanox OFED 5.4-1.0.3.0

솔루션 설계를 검토합니다

설계 개요

BeeGFS 병렬 파일 시스템을 NetApp EF600 스토리지 시스템과 결합하는 NetApp 솔루션 기반 BeeGFS를 지원하려면 특정 장비, 케이블링, 구성이 필요합니다.

자세한 내용:

- "하드웨어 구성"
- "소프트웨어 구성"
- "설계 검증"
- "사이징 지침"
- "성능 튜닝"

설계 및 성능 면에서 다양한 파생 아키텍처:

- "고용량 빌딩 블록"

하드웨어 구성

NetApp의 BeeGFS에 대한 하드웨어 구성에는 파일 노드 및 네트워크 케이블 연결이 포함됩니다.

파일 노드 구성

파일 노드에는 동일한 수의 PCIe 슬롯 및 메모리에 대한 로컬 액세스를 포함하는 별도의 NUMA 존으로 구성된 2개의 CPU 소켓이 있습니다.

InfiniBand 어댑터는 적절한 PCI 라이저 또는 슬롯에 설치되어야 사용 가능한 PCIe 레인 및 메모리 채널에 걸쳐 작업 부하가 분산됩니다. 개별 BeeGFS 서비스에 대한 작업을 특정 NUMA 노드에 완전히 격리하여 워크로드의 균형을 조정합니다. 목표는 두 개의 독립적인 단일 소켓 서버처럼 각 파일 노드에서 유사한 성능을 얻는 것입니다.

다음 그림에서는 파일 노드 NUMA 구성을 보여 줍니다.

[]

BeeGFS 프로세스는 사용된 인터페이스가 동일한 존에 있도록 특정 NUMA 존에 고정됩니다. 이렇게 구성하면 소켓 간 연결을 통한 원격 액세스가 필요하지 않습니다. 소켓 간 연결은 QPI 또는 GMI2 링크라고도 합니다. 최신 프로세서 아키텍처에서도 HDR InfiniBand와 같은 고속 네트워킹을 사용할 때 병목 현상이 발생할 수 있습니다.

네트워크 케이블 연결 구성

구성 요소 내에서 각 파일 노드는 총 4개의 중복 InfiniBand 연결을 사용하여 2개의 블록 노드에 연결됩니다. 또한 각 파일 노드에는 InfiniBand 스토리지 네트워크에 대한 4개의 이중화된 접속이 있습니다.

다음 그림에서 주목하십시오.

- 녹색으로 표시된 모든 파일 노드 포트는 스토리지 패브릭에 접속하는 데 사용되며, 다른 모든 파일 노드 포트는 블록 노드에 직접 연결됩니다.
- 특정 NUMA 존에 있는 2개의 InfiniBand 포트는 동일한 블록 노드의 A 및 B 컨트롤러에 연결됩니다.
- NUMA 노드 0의 포트는 항상 첫 번째 블록 노드에 연결됩니다.
- NUMA 노드 1의 포트는 두 번째 블록 노드에 연결됩니다.

[]

이중화 스위치가 있는 스토리지 네트워크의 경우 녹색으로 표시된 포트가 한 스위치에 연결되고 어두운 녹색으로 표시된 포트는 다른 스위치에 연결해야 합니다.

그림에 표시된 케이블 연결 구성을 통해 각 BeeGFS 서비스는 다음을 수행할 수 있습니다.

- BeeGFS 서비스를 실행 중인 파일 노드에 관계없이 동일한 NUMA 존에서 실행합니다.
- 장애 발생 위치에 관계없이 프런트엔드 스토리지 네트워크와 백엔드 블록 노드에 대한 보조 최적 경로 제공
- 블록 노드의 파일 노드 또는 컨트롤러에 유지 관리가 필요한 경우 성능 영향을 최소화합니다.

대역폭을 활용하기 위한 케이블 연결

PCIe 양방향 대역폭을 최대한 활용하려면 각 InfiniBand 어댑터의 포트 하나를 스토리지 패브릭에 연결하고 다른 포트는 블록 노드에 연결해야 합니다. HDR InfiniBand 포트의 이론적인 최대 속도는 25GBps입니다(신호 및 기타 오버헤드는 고려하지 않음). PCIe 4.0 x16 슬롯의 최대 단일 방향 대역폭은 32GBps이며 이론적으로 50GBps 대역폭을 처리할 수 있는 이중 포트 InfiniBand 어댑터가 통합된 파일 노드를 구현할 때 잠재적인 병목 현상이 발생합니다.

다음 그림은 전체 PCIe 양방향 대역폭을 활용하는 데 사용되는 케이블링 설계를 보여줍니다.

[]

각 BeeGFS 서비스에 대해 동일한 어댑터를 사용하여 클라이언트 트래픽에 사용되는 기본 포트를 해당 서비스 볼륨의 기본 소유자인 블록 노드 컨트롤러의 경로와 연결합니다. 자세한 내용은 을 참조하십시오 "소프트웨어 구성".

소프트웨어 구성

NetApp 기반 BeeGFS에 대한 소프트웨어 구성에는 BeeGFS 네트워크 구성 요소, EF600 블록 노드, BeeGFS 파일 노드, 리소스 그룹, BeeGFS 서비스가 포함됩니다.

BeeGFS 네트워크 구성

BeeGFS 네트워크 구성은 다음과 같은 구성 요소로 이루어집니다.

- * 부동 IP * 부동 IP는 동일한 네트워크의 모든 서버로 동적으로 라우팅될 수 있는 일종의 가상 IP 주소입니다. 여러 서버가 동일한 부동 IP 주소를 소유할 수 있지만, 특정 시간에 한 서버에서만 활성화될 수 있습니다.

  각 BeeGFS 서버 서비스에는 BeeGFS 서버 서비스의 실행 위치에 따라 파일 노드 간에 이동할 수 있는 고유한 IP 주소가 있습니다. 이러한 부동 IP 구성을 통해 각 서비스가 다른 파일 노드로 독립적으로 페일오버할 수 있습니다. 클라이언트는 특정 BeeGFS 서비스의 IP 주소를 알고 있으면 됩니다. 이 경우 현재 해당 서비스를 실행 중인 파일 노드를 알 필요가 없습니다.

- * BeeGFS 서버 다중 홈 구성 * 솔루션의 밀도를 높이기 위해 각 파일 노드에는 동일한 IP 서브넷에 구성된 IP를 가진 여러 스토리지 인터페이스가 있습니다.

  기본적으로 하나의 인터페이스에 대한 요청은 동일한 서브넷에 있는 경우 다른 인터페이스에서 응답할 수 있기 때문에 Linux 네트워킹 스택에서 이 구성이 예상대로 작동하는지 확인하기 위해 추가 구성이 필요합니다. 다른 단점 외에도 이 기본 동작으로 인해 RDMA 연결을 적절하게 설정하거나 유지할 수 없습니다.

  Ansible 기반 배포에서는 부동 IP가 시작 및 중지되는 시기와 함께 RP(역방향 경로) 및 ARP(주소 해상도 프로토콜) 동작 조임을 처리하고, 다중 홈 네트워크 구성이 제대로 작동하도록 해당 IP 경로 및 규칙을 동적으로 생성합니다.

- * BeeGFS 클라이언트 다중 레일 구성 * _Multi-RAIL_은 애플리케이션이 여러 개의 독립적인 네트워크 "레일"을 사용하여 성능을 향상시키는 기능을 의미합니다.

  BeeGFS는 RDMA 연결을 위해 RDMA를 사용할 수 있지만, BeeGFS는 IPoIB를 사용하여 RDMA 연결 검색 및 설정을 간소화합니다. BeeGFS 클라이언트가 여러 InfiniBand 인터페이스를 사용할 수 있도록 하려면 각 클라이언트를 다른 서브넷에 있는 IP 주소로 구성한 다음 각 서브넷에 있는 BeeGFS 서버 서비스의 절반에 대해 기본 인터페이스를 구성할 수 있습니다.

  다음 다이어그램에서는 밝은 녹색으로 강조 표시된 인터페이스가 하나의 IP 서브넷(예: 100.127.0.0/16)에 있고 어두운 녹색 인터페이스는 다른 서브넷(예: 100.128.0.0/16)에 있습니다.

  다음 그림에서는 여러 BeeGFS 클라이언트 인터페이스에서 트래픽의 균형을 조정하는 방법을 보여 줍니다.

  []

  BeeGFS의 각 파일은 일반적으로 여러 스토리지 서비스에 걸쳐 스트라이핑되기 때문에 다중 레일 구성을 통해 클라이언트는 단일 InfiniBand 포트보다 더 많은 처리량을 달성할 수 있습니다. 예를 들어, 다음 코드 샘플은 클라이언트가 두 인터페이스 간에 트래픽의 균형을 조정할 수 있도록 하는 일반적인 파일 스트라이핑 구성을 보여 줍니다.

```
root@ictad21h01:/mnt/beegfs# beegfs-ctl --getentryinfo myfile
Entry type: file
EntryID: 11D-624759A9-65
Metadata node: meta_01_tgt_0101 [ID: 101]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 1M
+ Number of storage targets: desired: 4; actual: 4
+ Storage targets:
  + 101 @ stor_01_tgt_0101 [ID: 101]
  + 102 @ stor_01_tgt_0101 [ID: 101]
  + 201 @ stor_02_tgt_0201 [ID: 201]
  + 202 @ stor_02_tgt_0201 [ID: 201]
```

두 개의 IPoIB 서브넷을 사용하는 것은 논리적인 구분입니다. 필요한 경우 단일 물리적 InfiniBand 서브넷(스토리지 네트워크)을 사용할 수 있습니다.

> 단일 IPoIB 서브넷에서 여러 IB 인터페이스를 사용할 수 있도록 BeeGFS 7.3.0에 멀티 레일 지원이 추가되었습니다. BeeGFS 7.3.0을 GA 전에 NetApp 기반 BeeGFS 솔루션의 설계가 개발되었으며, BeeGFS 클라이언트에서 두 개의 IB 인터페이스를 사용하는 두 개의 IP 서브넷을 보여 줍니다. 다중 IP 서브넷 접근 방식의 한 가지 장점은 BeeGFS 클라이언트 노드에서 멀티호밍을 구성할 필요가 없기 때문입니다(자세한 내용은 참조) "BeeGFS RDMA 지원")를 클릭합니다.

EF600 블록 노드 구성

블록 노드는 동일한 드라이브 세트에 대한 공유 액세스를 가진 2개의 액티브/액티브 RAID 컨트롤러로 구성됩니다. 일반적으로 각 컨트롤러는 시스템에 구성된 볼륨의 절반을 소유하지만 필요에 따라 다른 컨트롤러를 인수할 수 있습니다.

파일 노드의 다중 경로 소프트웨어는 각 볼륨에 대한 최적화된 활성 경로를 결정하고 케이블, 어댑터 또는 컨트롤러에 장애가 발생할 경우 대체 경로로 자동으로 이동합니다.

다음 다이어그램은 EF600 블록 노드의 컨트롤러 레이아웃을 보여 줍니다.

[]

공유 디스크 HA 솔루션을 지원하기 위해 볼륨은 두 파일 노드에 매핑되므로 필요에 따라 서로 테이크오버할 수 있습니다. 다음 다이어그램은 BeeGFS 서비스 및 기본 볼륨 소유권이 최대 성능을 위해 구성되는 방법의 예를 보여 줍니다. 각 BeeGFS 서비스 왼쪽에 있는 인터페이스는 클라이언트 및 기타 서비스가 연락하는 데 사용하는 기본 인터페이스를 나타냅니다.

[]

앞의 예에서 클라이언트 및 서버 서비스는 인터페이스 i1b를 사용하여 스토리지 서비스 1과 통신하는 것을 선호합니다. 스토리지 서비스 1은 인터페이스 i1a를 기본 경로로 사용하여 첫 번째 블록 노드의 컨트롤러 A에 있는 해당 볼륨(storage_tgt_101, 102)과 통신합니다. 이 구성은 InfiniBand 어댑터에서 사용할 수 있는 양방향 PCIe 대역폭을 완벽하게 사용하고 PCIe 4.0에서 사용할 수 있는 것보다 듀얼 포트 HDR InfiniBand 어댑터에서 더 나은 성능을 제공합니다.

BeeGFS 파일 노드 구성

BeeGFS 파일 노드는 HA(High-Availability) 클러스터로 구성되어 여러 파일 노드 간에 BeeGFS 서비스의 페일오버를 지원합니다.

HA 클러스터 설계는 널리 사용되는 두 가지 Linux HA 프로젝트, 즉 클러스터 멤버십을 위한 Corosync 및 클러스터 리소스 관리를 위한 Pacemaker를 기반으로 합니다. 자세한 내용은 을 참조하십시오 "고가용성 애드온을 위한 Red Hat 교육".

NetApp은 클러스터가 지능적으로 BeeGFS 리소스를 시작하고 모니터링할 수 있도록 여러 OCF(Open Cluster Framework) 리소스 에이전트를 저술하고 확장했습니다.

BeeGFS HA 클러스터

일반적으로, HA를 사용하거나 사용하지 않고 BeeGFS 서비스를 시작할 때 다음과 같은 몇 가지 리소스를 사용해야 합니다.

- 서비스에 연결할 수 있는 IP 주소이며 일반적으로 Network Manager에서 구성합니다.
- BeeGFS에서 데이터를 저장하기 위한 타겟으로 사용되는 기본 파일 시스템입니다.

  일반적으로 이러한 항목은 '/etc/fstab'에 정의되어 있으며 systemd에 의해 마운트됩니다.

- 다른 리소스가 준비되면 BeeGFS 프로세스를 시작하는 시스템 서비스입니다.

  추가 소프트웨어가 없으면 이러한 리소스는 단일 파일 노드에서만 시작됩니다. 따라서 파일 노드가 오프라인이 되면 BeeGFS 파일 시스템의 일부를 액세스할 수 없습니다.

여러 노드에서 각 BeeGFS 서비스를 시작할 수 있으므로, Pacemaker는 각 서비스와 종속 리소스가 한 번에 하나의 노드에서만 실행되도록 해야 합니다. 예를 들어, 두 노드가 동일한 BeeGFS 서비스를 시작하려고 하면 둘 다 기본 타겟의 동일한 파일에 쓰려고 하면 데이터 손상이 발생할 위험이 있습니다. 이러한 시나리오를 피하기 위해, 페이스메이커의 Corosync를 사용하여 전체 클러스터의 상태를 모든 노드에 걸쳐 안정적으로 유지하고 쿼럼을 설정합니다.

클러스터에서 장애가 발생하면 심장박동기가 반응하여 다른 노드에서 BeeGFS 리소스를 다시 시작합니다. 일부 시나리오에서는 심박조율기가 장애가 발생한 원래 노드와 통신하지 못하여 리소스가 중지되었는지 확인할 수 없습니다. 다른 곳에서 BeeGFS 리소스를 다시 시작하기 전에 노드가 다운되었는지 확인하려면 심장박동기가 장애가 있는 노드를 분리합니다. 즉, 전원을 제거하는 것이 좋습니다.

심박조율기가 PDU(Power Distribution Unit)를 사용하여 노드를 펜싱하거나 서버 BMC(Baseboard Management Controller)를 Redfish와 같은 API와 함께 사용하여 오픈 소스 펜싱 에이전트를 많이 사용할 수 있습니다.

BeeGFS가 HA 클러스터에서 실행 중인 경우 모든 BeeGFS 서비스 및 기본 리소스는 리소스 그룹의 페이스 메이커를 통해 관리됩니다. 각 BeeGFS 서비스 및 해당 서비스가 의존하는 리소스가 리소스 그룹으로 구성되어 리소스가 올바른 순서로 시작 및 중지되어 동일한 노드에 배치됩니다.

각 BeeGFS 리소스 그룹에 대해 심장박동기는 특정 노드에서 BeeGFS 서비스에 더 이상 액세스할 수 없을 때 장애 조건을 감지하고 페일오버를 지능적으로 트리거하는 사용자 지정 BeeGFS 모니터링 리소스를 실행합니다.

다음 그림에서는 심장박동기 제어 BeeGFS 서비스 및 종속성을 보여 줍니다.

[]

동일한 유형의 여러 BeeGFS 서비스를 동일한 노드에서 시작할 수 있도록 다중 모드 구성 방법을 사용하여 BeeGFS 서비스를 시작하도록 페이스 메이커를 구성합니다. 자세한 내용은 를 참조하십시오 "멀티 모드에 대한 BeeGFS 문서".

BeeGFS 서비스는 여러 노드에서 시작할 수 있어야 하므로 각 서비스의 구성 파일('/etc/beegfs'에 있음)은 해당 서비스의 BeeGFS 타겟으로 사용되는 E-Series 볼륨 중 하나에 저장됩니다. 따라서 특정 BeeGFS 서비스에 대한 데이터와 함께 서비스를 실행해야 하는 모든 노드에서 해당 구성을 액세스할 수 있습니다.

```
# tree stor_01_tgt_0101/ -L 2
stor_01_tgt_0101/
├── data
│   ├── benchmark
│   ├── buddymir
│   ├── chunks
│   ├── format.conf
│   ├── lock.pid
│   ├── nodeID
│   ├── nodeNumID
│   ├── originalNodeID
│   ├── targetID
│   └── targetNumID
└── storage_config
    ├── beegfs-storage.conf
    ├── connInterfacesFile.conf
    └── connNetFilterFile.conf
```

설계 검증

BeeGFS on NetApp 솔루션의 2세대 설계는 3가지 구성 요소 구성 프로필을 사용하여 검증되었습니다.

구성 프로파일에는 다음이 포함됩니다.

- BeeGFS 관리, 메타데이터 및 스토리지 서비스를 포함한 단일 기본 구성 요소입니다.
- BeeGFS 메타데이터와 스토리지 구성 요소
- BeeGFS 스토리지 전용 구성 요소입니다.

빌딩 블록은 2개의 Mellanox Quantum InfiniBand(MQM8700) 스위치에 연결되었습니다. 10개의 BeeGFS 클라이언트도 InfiniBand 스위치에 연결되었으며 통합 벤치마크 유틸리티를 실행하는 데 사용되었습니다.

다음 그림에서는 NetApp 솔루션의 BeeGFS 검증을 위해 사용되는 BeeGFS 구성을 보여 줍니다.

[]

BeeGFS 파일 스트라이핑

병렬 파일 시스템의 이점은 여러 스토리지 대상 간에 개별 파일을 스트라이핑하는 기능입니다. 이 기능은 동일하거나 다른 기본 스토리지 시스템의 볼륨을 나타낼 수 있습니다.

BeeGFS에서는 디렉토리 및 파일별로 스트라이핑을 구성하여 각 파일에 사용되는 타겟 수를 제어하고 각 파일 스트라이프에 사용되는 청크 크기(또는 블록 크기)를 제어할 수 있습니다. 따라서 서비스를 재구성하거나 다시 시작할 필요 없이 파일 시스템에서 다양한 유형의 워크로드와 I/O 프로필을 지원할 수 있습니다. "begfs -ctl" 명령줄 도구 또는 스트라이핑 API를 사용하는 응용 프로그램을 사용하여 스트라이프 설정을 적용할 수 있습니다. 자세한 내용은 의 BeeGFS 설명서를 참조하십시오 "스트라이핑" 및 "스트라이핑 API".

최상의 성능을 얻기 위해 테스트 중에 스트라이프 패턴을 조정하고 각 테스트에 사용된 매개 변수를 기록하였습니다.

IOR 대역폭 테스트**:** 여러 클라이언트

IOR 대역폭 테스트는 OpenMPI를 사용하여 합성 I/O 생성기 툴 IOR(에서 제공)의 병렬 작업을 실행했습니다 "HPC GitHub를 참조하십시오") 10개 클라이언트 노드 전체에서 하나 이상의 BeeGFS 구성 요소로 이동합니다. 달리 명시되지 않은 한:

- 모든 테스트는 1MiB 전송 크기의 직접 I/O를 사용했습니다.
- BeeGFS 파일 스트라이핑은 1MB 청크 크기 및 파일당 하나의 타겟으로 설정되었습니다.

다음 매개 변수는 IOR에 사용되었습니다. 세그먼트 수는 구성 요소 1개의 경우 애그리게이트 파일 크기를 5TiB로, 3개의 구성 요소는 40TiB로 유지하도록 조정되었습니다.

```
mpirun --allow-run-as-root --mca btl tcp -np 48 -map-by node -hostfile
10xnodes ior -b 1024k --posix.odirect -e -t 1024k -s 54613 -z -C -F -E -k
```

하나의 **BeeGFS** 기본**(**관리**,** 메타데이터 및 스토리지**)** 구성 요소입니다

다음 그림에서는 단일 BeeGFS 기반(관리, 메타데이터 및 스토리지) 구성 요소를 사용한 IOR 테스트 결과를 보여 줍니다.

[]

BeeGFS 메타데이터 **+** 스토리지 구성 요소입니다

다음 그림에서는 단일 BeeGFS 메타데이터 + 스토리지 구성 요소를 사용한 IOR 테스트 결과를 보여 줍니다.

[]

BeeGFS 스토리지 전용 구성 요소입니다

다음 그림에서는 단일 BeeGFS 스토리지 전용 구성 요소를 사용한 IOR 테스트 결과를 보여 줍니다.

[]

BeeGFS 빌딩 블록 **3**개

다음 그림에서는 세 개의 BeeGFS 구성 요소가 포함된 IOR 테스트 결과를 보여 줍니다.

[]

예상한 대로 기본 구성 요소과 후속 메타데이터 + 스토리지 구성 요소 간의 성능 차이는 무시할 수 있습니다. 메타데이터

+ 스토리지 구성 요소 및 스토리지 전용 구성 요소를 비교하면 스토리지 대상으로 사용되는 추가 드라이브로 인해 읽기 성능이 약간 향상됩니다. 그러나 쓰기 성능에는 큰 차이가 없습니다. 더 높은 성능을 얻기 위해 여러 구성 요소를 함께 추가하여 성능을 선형 방식으로 확장할 수 있습니다.

IOR 대역폭 테스트**:** 단일 클라이언트

IOR 대역폭 테스트는 OpenMPI를 사용하여 단일 고성능 GPU 서버를 사용하여 여러 IOR 프로세스를 실행하여 단일 클라이언트에서 얻을 수 있는 성능을 탐색했습니다.

이 테스트는 클라이언트가 Linux 커널 페이지 캐시('tuneFileCacheType=NATIVE')를 사용하도록 구성된 경우 BeeGFS의 다시 읽기 동작 및 성능을 기본 '버퍼링' 설정과 비교합니다.

네이티브 캐싱 모드는 클라이언트의 Linux 커널 페이지 캐시를 사용하므로 네트워크를 통해 다시 전송되는 것이 아니라 로컬 메모리에서 다시 읽기 작업을 수행할 수 있습니다.

다음 다이어그램은 BeeGFS 빌딩 블록 3개와 단일 클라이언트를 사용한 IOR 테스트 결과를 보여 줍니다.

[]

이러한 테스트를 위한 BeeGFS 스트라이핑은 파일당 타겟 8개가 포함된 1MB 청크 크기로 설정되었습니다.

기본 버퍼링 모드에서 쓰기 및 초기 읽기 성능이 향상되지만, 동일한 데이터를 여러 번 다시 읽는 워크로드의 경우 네이티브 캐싱 모드에서 성능이 크게 향상됩니다. 이렇게 향상된 다시 읽기 성능은 여러 번의 Epoch에서 동일한 데이터 세트를 여러 번 다시 읽는 딥 러닝과 같은 워크로드에 중요합니다.

메타데이터 성능 테스트

Metadata 성능 테스트는 IOR의 일부로 포함된 MDTest 도구를 사용하여 BeeGFS의 메타데이터 성능을 측정했습니다. 이 테스트에서는 OpenMPI를 사용하여 10개의 클라이언트 노드 모두에서 병렬 작업을 실행했습니다.

다음 매개 변수는 총 프로세스 수가 2배속 단계에서 10개에서 320으로 조정되고 파일 크기가 4K인 벤치마크 테스트를 실행하는 데 사용되었습니다.

```
mpirun -h 10xnodes –map-by node np $processes mdtest -e 4k -w 4k -i 3 -I
16 -z 3 -b 8 -u
```

메타데이터 성능은 먼저 메타데이터 + 스토리지 구성 요소 하나를 측정한 후 추가 구성 요소를 추가하여 성능이 얼마나 향상되는지를 보여 줍니다.

하나의 **BeeGFS** 메타데이터 **+** 스토리지 구성 요소입니다

다음 다이어그램은 하나의 BeeGFS 메타데이터 + 스토리지 구성 요소가 포함된 MDTest 결과를 보여 줍니다.

[]

BeeGFS 메타데이터 **+** 스토리지 구성 요소 **2**개

다음 다이어그램은 BeeGFS 메타데이터 + 스토리지 구성 요소 두 개가 포함된 MDTest 결과를 보여 줍니다.

[]

기능 검증

이 아키텍처의 검증 과정에서 NetApp은 다음을 비롯한 여러 기능 테스트를 수행했습니다.

- 스위치 포트를 비활성화하여 단일 클라이언트 InfiniBand 포트에 장애 발생
- 스위치 포트를 비활성화하여 단일 서버 InfiniBand 포트에 장애 발생
- BMC를 사용하여 즉시 서버 전원을 끕니다.
- 노드를 대기 노드에 배치하고 다른 노드에 대한 서비스 장애 조치를 원활히 합니다.
- 노드를 다시 온라인 상태로 전환하고 원래 노드에 서비스를 페일백합니다.
- PDU를 사용하여 InfiniBand 스위치 중 하나의 전원을 끕니다. BeeGFS 클라이언트에 설정된 'sysSessionChecksEnabled:false' 매개 변수를 사용하여 스트레스 테스트가 진행되는 동안 모든 테스트가 수행되었습니다. I/O에 대한 오류나 운영 중단이 관찰되지 않았습니다.

> 알려진 문제가 있습니다(참조) "변경 로그") 기본 인터페이스('connInterfacesFile'에 정의된 대로) 손실 또는 BeeGFS 서버 장애로 인해 BeeGFS 클라이언트/서버 RDMA 연결이 예기치 않게 중단되거나 활성 클라이언트 I/O가 최대 10분 동안 중단되어 다시 시작할 수 있습니다. 이 문제는 계획된 유지 관리를 위해 BeeGFS 노드가 정상적으로 대기 상태가 되거나 TCP가 사용 중인 경우 발생하지 않습니다.

NVIDIA DGX A100 SuperPOD 및 **BasePOD** 검증

NetApp은 메타데이터와 스토리지 구성 프로필이 적용된 3개의 구성 블록으로 구성된 유사한 BeeGFS 파일 시스템을 사용하여 NVIDIAs DGX A100 SuperPOD에 대한 스토리지 솔루션을 검증했습니다. 검증 노력에는 다양한 스토리지, 머신 러닝 및 딥 러닝 벤치마크를 실행하는 20개의 DGX A100 GPU 서버를 통해 이 NVA에 의해 설명된 솔루션을 테스트하는 작업이 포함되었습니다. NVIDIA DGX A100 SuperPOD에서 사용하도록 인증된 모든 스토리지는 NVIDIA BasePOD 아키텍처에도 자동으로 사용하도록 인증되었습니다.

자세한 내용은 을 참조하십시오 "NetApp을 포함한 NVIDIA DGX SuperPOD" 및 "NVIDIA DGX 베이스POD".

사이징 지침

BeeGFS 솔루션에는 검증 테스트를 기반으로 한 성능 및 용량 사이징에 대한 권장 사항이 포함되어 있습니다.

빌딩 블록 아키텍처의 목표는 특정 BeeGFS 시스템의 요구 사항을 충족하기 위해 여러 빌딩 블록을 추가하여 간편하게 사이징할 수 있는 솔루션을 구축하는 것입니다. 아래 지침에 따라 환경 요구 사항을 충족하는 데 필요한 BeeGFS 빌딩 블록의 양과 유형을 예측할 수 있습니다.

이러한 추정치는 최상의 성능을 제공하는 것으로, 가상 벤치마킹 애플리케이션은 실제 애플리케이션이 사용할 수 없는 방식으로 기본 파일 시스템의 사용을 최적화하기 위해 작성 및 사용됩니다.

성능 사이징

다음 표에는 권장되는 성능 사이징이 나와 있습니다.

구성 프로파일	**1MiB** 읽기	**1MiB**의 쓰기입니다
메타데이터 + 스토리지	62GiBps	21GiBps

구성 프로파일	**1MiB** 읽기	**1MiB**의 쓰기입니다
스토리지만 해당	64GiBps	21GiBps

메타데이터 용량 사이징 예상치는 "경험 규칙"을 기반으로 하며, 이 경우 500GB의 용량으로 BeeGFS에서 약 1억 5천만 개의 파일을 저장할 수 있습니다. (자세한 내용은 BeeGFS 설명서를 참조하십시오 "시스템 요구 사항"참조)

액세스 제어 목록, 디렉터리별 디렉토리 및 파일 수와 같은 기능을 사용하면 메타데이터 공간이 얼마나 빨리 소비되는지를 알 수 있습니다. 스토리지 용량 추정치는 RAID 6 및 XFS 오버헤드와 함께 사용 가능한 드라이브 용량을 고려합니다.

메타데이터 **+** 스토리지 구성 요소에 대한 용량 사이징

다음 표에는 메타데이터와 스토리지 구성 요소에 권장되는 용량 사이징이 나와 있습니다.

드라이브 크기**(2+2 RAID 1)** 메타데이터 볼륨 그룹	메타데이터 용량(파일 수)	드라이브 크기**(8 + 2 RAID 6)** 스토리지 볼륨 그룹	스토리지 용량(파일 콘텐츠)
1.92TB	1,938,577,200	1.92TB	51.77TB
3.84TB	3,880,388,400	3.84TB	103.55TB
7.68TB	8,125,278,000입니다	7.68TB	216.74TB
15.3TB	17,269,854,000	15.3TB	460.60TB

메타데이터 및 스토리지 구성 요소의 크기를 조정할 때 더 작은 드라이브를 메타데이터 볼륨 그룹에 사용하는 것과 스토리지 볼륨 그룹을 사용하는 것을 통해 비용을 절감할 수 있습니다.

스토리지 전용 구성 요소에 대한 용량 사이징

다음 표에는 스토리지 전용 구성 요소에 대한 경험 많은 용량 사이징이 나와 있습니다.

드라이브 크기**(10 + 2 RAID 6)** 스토리지 볼륨 그룹	스토리지 용량(파일 콘텐츠)
1.92TB	59.89TB
3.84TB	119.80TB
7.68TB	251.89TB
15.3TB	538.55TB

글로벌 파일 잠금이 활성화되지 않은 경우 기본(첫 번째) 구성 요소에서 관리 서비스를 포함할 때 발생하는 성능 및 용량 오버헤드가 최소화됩니다.

성능 튜닝

BeeGFS 솔루션에는 검증 테스트를 기반으로 한 성능 조정을 위한 권장 사항이 포함되어 있습니다.

BeeGFS가 즉시 사용 가능한 우수한 성능을 제공하기는 하지만, NetApp은 성능을 극대화하기 위해 일련의 권장 튜닝 매개 변수를 개발했습니다. 이 매개 변수는 기본 E-Series 블록 노드의 기능과 공유 디스크 HA 아키텍처에서

BeeGFS를 실행하는 데 필요한 특수 요구사항을 모두 고려합니다.

파일 노드의 성능 튜닝

구성할 수 있는 조정 매개변수는 다음과 같습니다.

1. * 파일 노드의 UEFI/BIOS에서 시스템 설정. * 성능을 극대화하려면 파일 노드로 사용하는 서버 모델에서 시스템 설정을 구성하는 것이 좋습니다. 시스템 설정(UEFI/BIOS) 또는 베이스보드 관리 컨트롤러(BMC)에서 제공하는 Redfish API를 사용하여 파일 노드를 설정할 때 시스템 설정을 구성합니다.

   시스템 설정은 파일 노드로 사용하는 서버 모델에 따라 달라집니다. 사용 중인 서버 모델에 따라 설정을 수동으로 구성해야 합니다. 검증된 Lenovo SR665 파일 노드에 대한 시스템 설정을 구성하는 방법은 을 참조하십시오 "성능을 위해 파일 노드 시스템 설정을 조정합니다".

2. * 필수 구성 매개 변수에 대한 기본 설정 * 필수 구성 매개 변수는 BeeGFS 서비스의 구성 방법과 E-Series 볼륨(블록 장치)의 포맷 및 마운트에 영향을 미칩니다. 이러한 필수 구성 매개 변수는 다음과 같습니다.
   - BeeGFS 서비스 구성 매개 변수입니다

     필요에 따라 구성 매개 변수의 기본 설정을 재정의할 수 있습니다. 특정 워크로드 또는 사용 사례에 맞게 조정할 수 있는 매개 변수는 를 참조하십시오 "BeeGFS 서비스 구성 매개 변수입니다".

   - 볼륨 포맷팅 및 마운팅 매개 변수는 권장 기본값으로 설정되며 고급 사용 사례에만 조정해야 합니다. 기본값은 다음을 수행합니다.
     - 기본 볼륨의 RAID 구성 및 세그먼트 크기와 함께 타겟 유형(예: 관리, 메타데이터 또는 스토리지)을 기준으로 초기 볼륨 포맷을 최적화합니다.
     - 심박동조율기가 각 볼륨을 마운트하는 방법을 조정하여 변경 사항이 즉시 E-시리즈 블록 노드로 플러시되도록 합니다. 이렇게 하면 활성 쓰기가 진행 중일 때 파일 노드가 실패할 때 데이터 손실을 방지할 수 있습니다.

       특정 워크로드 또는 사용 사례에 맞게 조정할 수 있는 매개 변수는 를 참조하십시오 "볼륨 포맷 및 마운팅 구성 매개 변수".

3. * 파일 노드에 설치된 Linux OS의 시스템 설정. * 의 4단계에서 Ansible 인벤토리를 작성할 때 기본 Linux OS 시스템 설정을 재정의할 수 있습니다 "Ansible 인벤토리를 작성합니다".

   기본 설정을 사용하여 NetApp 기반 BeeGFS 솔루션을 검증했지만 특정 워크로드 또는 사용 사례에 맞게 수정할 수 있습니다. 변경할 수 있는 Linux OS 시스템 설정의 예는 다음과 같습니다.

   - E-Series 블록 장치의 I/O 큐

     BeeGFS 타겟으로 사용되는 E-Series 블록 디바이스에서 I/O 큐를 구성하여 다음을 수행할 수 있습니다.

     - 장치 유형(NVMe, HDD 등)에 따라 스케줄링 알고리즘을 조정합니다.
     - 미결 요청 수를 늘립니다.
     - 요청 크기를 조정합니다.
     - 미리 읽기 동작 최적화

   - 가상 메모리 설정.

     최적의 스트리밍 성능을 위해 가상 메모리 설정을 조정할 수 있습니다.

- CPU 설정

  최대 성능을 위해 CPU 주파수 조절기 및 기타 CPU 구성을 조정할 수 있습니다.

- 읽기 요청 크기입니다.

  Mellanox HCA의 최대 읽기 요청 크기를 늘릴 수 있습니다.

블록 노드의 성능 튜닝

특정 BeeGFS 빌딩 블록에 적용되는 구성 프로필을 기준으로 블록 노드에 구성된 볼륨 그룹이 약간 변경됩니다. 예를 들어, 24-드라이브 EF600 블록 노드는 다음과 같습니다.

- BeeGFS 관리, 메타데이터 및 스토리지 서비스를 비롯한 단일 기본 구성 요소:
  - BeeGFS 관리 및 메타데이터 서비스를 위한 1 x 2 + 2 RAID 10 볼륨 그룹
  - BeeGFS 스토리지 서비스용 2x 8+2 RAID 6 볼륨 그룹
- BeeGFS 메타데이터 + 스토리지 구성 요소:
  - BeeGFS 메타데이터 서비스용 2 + 2 RAID 10 볼륨 그룹 1개
  - BeeGFS 스토리지 서비스용 2x 8+2 RAID 6 볼륨 그룹
- BeeGFS 스토리지 전용 구성 요소:
  - BeeGFS 스토리지 서비스용 10+2 RAID 6 볼륨 그룹 2개

> BeeGFS는 스토리지 대비 관리 및 메타데이터를 위해 스토리지 공간이 훨씬 적게 요구되므로 RAID 10 볼륨 그룹에 더 작은 드라이브를 사용하는 옵션이 있습니다. 작은 드라이브는 가장 바깥쪽 드라이브 슬롯에 장착해야 합니다. 자세한 내용은 를 참조하십시오 "배포 지침".

이러한 모든 설정은 Ansible 기반 배포를 통해 구성되며, 다음은 성능/동작을 최적화하기 위해 일반적으로 권장되는 몇 가지 다른 설정과 함께 제공됩니다.

- 글로벌 캐시 블록 크기를 32KiB로 조정하고 요구 기반 캐시 플러시를 80%로 조정합니다.
- 자동 로드 밸런싱 비활성화(컨트롤러 볼륨 할당이 의도한 대로 유지되는지 확인)
- 읽기 캐싱 설정 및 미리 읽기 캐싱 해제
- 미러링으로 쓰기 캐시를 설정하고 배터리 백업이 필요하므로 블록 노드 컨트롤러의 장애가 발생해도 캐시가 유지됩니다.
- 드라이브가 볼륨 그룹에 할당되는 순서를 지정하여 사용 가능한 드라이브 채널 간에 I/O의 균형을 조정합니다.

고용량 구성 요소입니다

표준 BeeGFS 솔루션 설계는 고성능 워크로드를 염두에 두고 설계되었습니다. 고용량 사용 사례를 찾는 고객은 여기에 설명된 설계 및 성능 특성의 변화를 준수해야 합니다.

하드웨어 및 소프트웨어 구성

고용량 구성 요소에 대한 하드웨어 및 소프트웨어 구성은 EF300 컨트롤러를 각 스토리지 어레이당 60개의 드라이브로 1~7개의 IOM 확장 트레이를 연결하는 옵션과 함께 EF300 컨트롤러로 교체해야 한다는 점을 제외하고 표준입니다.

빌딩 블록당 총 2-14개의 확장 트레이.

대용량 구성 요소 설계를 구축하는 고객은 각 노드에 대해 BeeGFS 관리, 메타데이터 및 스토리지 서비스로 구성된 기본 구성 요소 스타일 구성만 사용할 수 있습니다. 비용 효율성을 위해 대용량 스토리지 노드는 EF300 컨트롤러 엔클로저의 NVMe 드라이브에 메타데이터 볼륨을 프로비저닝하고 확장 트레이의 NL-SAS 드라이브에 스토리지 볼륨을 프로비저닝해야 합니다.

[]

사이징 지침

이 사이징 지침은 대용량 구성 요소가 기본 EF300 엔클로저의 메타데이터용 2+2 NVMe SSD 볼륨 그룹 1개와 스토리지용 IOM 확장 트레이당 8개+2 NL-SAS 볼륨 그룹 6개로 구성되어 있다고 가정합니다.

드라이브 크기(용량 **HDD)**	**BB**당 용량(**1**트레이)	**BB**당 용량(**2**개의 트레이)	**BB**당 용량(**3**개의 트레이)	**BB**당 용량(**4**개의 트레이)
4TB	439TB	878TB	1,317TB	1756TB
8TB	878TB	1756TB	2,634TB	3512TB
10TB	1097TB	2195TB	3292TB	4390TB
12TB	1,317TB	2,634TB	3,951TB	5268TB
16TB	1756TB	3512TB	5268TB	7024TB
18TB	1975TB	3,951TB	5,927TB	7902TB

솔루션 구축

구축 개요

2세대 NetApp BeeGFS 빌딩 블록 설계를 사용하여 NetApp에서 검증된 파일 및 블록 노드에 BeeGFS를 구축할 수 있습니다.

Ansible 컬렉션 및 역할

Ansible을 사용하여 NetApp 솔루션에 BeeGFS를 구축할 수 있습니다. 이는 애플리케이션 구축을 자동화하는 데 사용되는 일반적인 IT 자동화 엔진입니다. Ansible은 구축할 BeeGFS 파일 시스템을 모델링하는 인벤토리로 통칭되는 일련의 파일을 사용합니다.

Ansible을 사용하면 NetApp과 같은 기업에서 Ansible Galaxy의 컬렉션을 사용하여 기본 제공 기능을 확장할 수 있습니다(참조) "NetApp E-Series BeeGFS 컬렉션")를 클릭합니다. 컬렉션에는 E-Series 볼륨 만들기와 같이 특정 기능 또는 작업을 수행하는 모듈과 여러 모듈 및 기타 역할을 호출할 수 있는 역할이 포함되어 있습니다. 이 자동화된 방식을 통해 BeeGFS 파일 시스템 및 기본 HA 클러스터를 구축하는 데 필요한 시간을 줄일 수 있습니다. 또한, 기존 파일 시스템을 확장하기 위해 구성 요소를 간단하게 추가할 수 있습니다.

자세한 내용은 을 참조하십시오 "Ansible 인벤토리에 대해 알아보십시오".

NetApp 솔루션에 BeeGFS를 구축하는 데 다양한 단계가 포함되므로 NetApp에서는 수동으로 솔루션 구축을 지원하지 않습니다.

BeeGFS 구성 요소에 대한 구성 프로필입니다

배포 절차는 다음과 같은 구성 프로파일을 다룹니다.

- 관리, 메타데이터 및 스토리지 서비스를 포함하는 하나의 기본 구성 요소입니다.
- 메타데이터와 스토리지 서비스가 포함된 두 번째 구성 요소입니다.
- 스토리지 서비스만 포함하는 세 번째 구성 요소입니다.

이러한 프로필을 통해 NetApp BeeGFS 구성 요소에 대한 권장 구성 프로필 전체 범위를 볼 수 있습니다. 각 구축에 필요한 메타데이터 및 스토리지 구성 요소 수 또는 스토리지 서비스 전용 구성 요소는 용량 및 성능 요구사항에 따라 절차에 따라 달라질 수 있습니다.

배포 단계 개요

배포에는 다음과 같은 고급 작업이 포함됩니다.

하드웨어 구축

1. 각 구성 요소를 물리적으로 조립합니다.
2. 랙 및 케이블 하드웨어. 자세한 절차는 를 참조하십시오 "하드웨어 구축".

소프트웨어 구축

1. "파일 및 블록 노드 설정".
    - 파일 노드에서 BMC IP를 구성합니다
    - 지원되는 운영 체제를 설치하고 파일 노드에서 관리 네트워킹을 구성합니다
    - 블록 노드에서 관리 IP를 구성합니다
2. "Ansible 제어 노드를 설정합니다".
3. "성능을 위해 시스템 설정을 조정합니다".
4. "Ansible 인벤토리를 작성합니다".
5. "BeeGFS 구성 요소에 대한 Ansible 인벤토리를 정의합니다".
6. "Ansible을 사용하여 BeeGFS 구축".
7. "BeeGFS 클라이언트를 구성합니다".

> 배포 절차에는 텍스트를 파일로 복사해야 하는 몇 가지 예제가 포함되어 있습니다. 특정 배포에 대해 수정하거나 수정할 수 있는 모든 사항에 대해 "" 또는 "//" 문자로 표시된 인라인 코멘트에 세심한 주의를 기울이십시오. 예를 들어, ``begfs_ha_ntp_server_pool: 이것은 설명의 예입니다! "pool 0.pool.ntp.org iburst maxsources 3" - "pool 1.pool.ntp.org iburst maxsources 3"

배포 권장 사항의 차이가 있는 파생 아키텍처:

- "고용량 빌딩 블록"

Ansible 인벤토리에 대해 알아보십시오

구축을 시작하기 전에, Ansible을 사용하여 2세대 BeeGFS 구성 요소 설계를 사용하여 NetApp

솔루션에서 BeeGFS를 구성 및 구축하는 방법을 이해해야 합니다.

Ansible 인벤토리는 파일 및 블록 노드의 구성을 정의하며 구축할 BeeGFS 파일 시스템을 나타냅니다. 인벤토리는 원하는 BeeGFS 파일 시스템을 설명하는 호스트, 그룹 및 변수를 포함합니다. 샘플 재고는 에서 다운로드할 수 있습니다 "NetApp E-Series BeeGFS GitHub를 참조하십시오".

Ansible 모듈 및 역할

Ansible 인벤토리에서 설명한 구성을 적용하려면 NetApp E-Series Ansible 컬렉션에서 제공하는 다양한 Ansible 모듈 및 역할, 특히 BeeGFS HA 7.2 역할(에서 제공)을 사용하십시오 "NetApp E-Series BeeGFS GitHub를 참조하십시오")를 구축하는 것이 좋습니다.

NetApp E-Series Ansible 컬렉션에서 각 역할은 NetApp 솔루션 기반의 BeeGFS를 완벽하게 구축하는 데 있습니다. 이 역할은 NetApp E-Series SANtricity, 호스트 및 BeeGFS 컬렉션을 사용하여 HA(High Availability)를 통해 BeeGFS 파일 시스템을 구성할 수 있습니다. 그런 다음 스토리지를 프로비저닝하고 매핑하고 클러스터 스토리지를 사용할 준비가 되었는지 확인할 수 있습니다.

역할에 맞는 심층적인 문서가 제공되지만, 구축 절차에서는 제2세대 BeeGFS 구성 요소 설계를 사용하여 NetApp 검증 아키텍처를 구축하는 데 역할을 사용하는 방법에 대해 설명합니다.

> Ansible에 대한 사전 경험이 사전 필수 요소가 될 수 있도록 구축 단계에서 자세한 정보를 제공하려고 하지만, Ansible 및 관련 용어에 친숙해야 합니다.

BeeGFS HA 클러스터의 인벤토리 레이아웃

Ansible 인벤토리 구조를 사용하여 BeeGFS HA 클러스터를 정의합니다.

이전 Ansible 경험을 가진 사람이라면 누구나 BeeGFS HA 역할이 각 호스트에 적용되는 변수(또는 팩트)를 검색하는 맞춤형 방법을 구현한다는 점을 알고 있어야 합니다. 이는 여러 서버에서 실행될 수 있는 리소스를 설명하는 Ansible 인벤토리를 작성하는 작업을 단순화하기 위해 필요합니다.

Ansible 재고는 일반적으로 'host_vars'와 'group_vars'에 있는 파일과 특정 그룹(또는 다른 그룹에 잠재적으로 그룹)에 호스트를 할당하는 재고 .yml 파일로 구성됩니다.

> 본 하위 섹션의 내용을 포함하는 파일을 만들지 마십시오. 이 내용은 예제로만 제공됩니다.

이 구성은 구성 프로필을 기반으로 사전 결정됩니다. 하지만 다음과 같이 Ansible 인벤토리로 모든 내용을 레이아웃하는 방법을 전반적으로 이해해야 합니다.

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        ictad22a01:
        ictad22a02:
        ictad22a03:
        ictad22a04:
        ictad22a05:
        ictad22a06:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
        meta_01:  # Group representing a metadata service with ID 01.
          hosts:
            file_node_01:  # This service is preferred on the first file
node.
            file_node_02:  # And can failover to the second file node.
        meta_02:  # Group representing a metadata service with ID 02.
          hosts:
            file_node_02:  # This service is preferred on the second file
node.
            file_node_01: # And can failover to the first file node.
```

각 서비스에 대해 해당 구성을 설명하는 group_vars 아래에 추가 파일이 생성됩니다.

```
# meta_01 - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: 8015
  connMetaPortUDP: 8015
  tuneBindToNumaZone: 0
floating_ips:
  - i1b: <IP>/<SUBNET_MASK>
  - i4b: <IP>/<SUBNET_MASK>
# Type of BeeGFS service the HA resource group will manage.
beegfs_service: metadata # Choices: management, metadata, storage.
# What block node should be used to create a volume for this service:
beegfs_targets:
  ictad22a01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        owning_controller: A
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.25
```

이 레이아웃을 통해 각 리소스에 대한 BeeGFS 서비스, 네트워크 및 스토리지 구성을 단일 위치에서 정의할 수 있습니다. BeeGFS 역할은 이러한 인벤토리 구조를 기반으로 각 파일 및 블록 노드에 필요한 구성을 집계합니다. 자세한 내용은 다음 블로그 게시물을 참조하십시오. "NetApp은 Ansible을 사용하여 BeeGFS에 대한 HA 구축을 가속화합니다".

> 각 서비스의 BeeGFS 숫자 및 문자열 노드 ID는 그룹 이름을 기준으로 자동으로 구성됩니다. 따라서 그룹 이름이 고유해야 하는 일반적인 Ansible 요구 사항 외에도 BeeGFS 서비스를 나타내는 그룹은 해당 그룹이 나타내는 BeeGFS 서비스 유형에 고유한 번호로 끝나야 합니다. 예를 들어, meta_01 및 stor_01은 허용되지만 metadata_01 및 meta_01은 허용되지 않습니다.

모범 사례를 검토합니다

NetApp 솔루션에 BeeGFS를 구축할 때는 모범 사례 지침을 따르십시오.

표준 규약

Ansible 인벤토리 파일을 물리적으로 조립하고 생성할 때는 다음 표준 규칙을 따르십시오(자세한 내용은 을 참조하십시오 "Ansible 인벤토리를 작성합니다")를 클릭합니다.

- 파일 노드 호스트 이름은 랙 상단에 더 적은 숫자가 있고 하단에 더 높은 숫자가 있는 순서대로 번호가 지정됩니다(H01-HN).

  예를 들어, 이름 지정 규칙 '[location][row][rack]hN'은 'ictad22h01'과 같습니다.

- 각 블록 노드는 각각 고유한 호스트 이름을 가진 두 개의 스토리지 컨트롤러로 구성됩니다.

  스토리지 어레이 이름은 Ansible 인벤토리의 일부로 전체 블록 스토리지 시스템을 나타내는 데 사용됩니다. 스토리지 배열 이름은 순서대로 번호(A01-AN)로 지정되어야 하며, 개별 컨트롤러의 호스트 이름은 해당 명명 규칙에서 파생됩니다.

  예를 들어, "ictad22a01"이라는 블록 노드는 일반적으로 "ictad22a01-a"와 "ictad22a01-b"와 같이 각 컨트롤러에 대해 구성된 호스트 이름을 가질 수 있지만, Ansible 재고에서는 "ictad22a01"이라고 합니다.

- 동일한 빌딩 블록 내의 파일 및 블록 노드는 동일한 번호 지정 체계를 공유하며, 랙의 서로 인접해 있으며 두 파일 노드 모두 위에 있고 두 블록 노드 바로 아래에 있습니다.

  예를 들어 첫 번째 빌딩 블록에서 파일 노드 H01 및 H02는 모두 블록 노드 A01 및 A02에 직접 연결됩니다. 위에서 아래로 호스트 이름은 H01, H02, A01 및 A02입니다.

- 빌딩 블록은 호스트 이름을 기준으로 순차적으로 설치되므로 번호가 낮은 호스트 이름은 랙 상단에, 번호가 높은 호스트 이름은 하단에 표시됩니다.

  이는 랙 스위치 상단으로 연결되는 케이블의 길이를 최소화하고 문제 해결을 단순화하기 위한 표준 배포 방법을 정의하는 것입니다. 랙 안정성 문제로 인해 이것이 허용되지 않는 데이터 센터의 경우, 맨 아래부터 랙을 채우는 역작업이 허용됩니다.

InfiniBand 스토리지 네트워크 구성

각 파일 노드의 InfiniBand 포트 중 절반은 블록 노드에 직접 연결하는 데 사용됩니다. 나머지 절반은 InfiniBand 스위치에 연결되며 BeeGFS 클라이언트-서버 연결에 사용됩니다. BeeGFS 클라이언트 및 서버에 사용되는 IPoIB 서브넷의 크기를 결정할 때 예상되는 컴퓨팅/GPU 클러스터 및 BeeGFS 파일 시스템 확장을 고려해야 합니다. 권장 IP 범위를 벗어나야 하는 경우, 단일 빌딩 블록의 각 직접 접속은 고유한 서브넷을 가지며 클라이언트-서버 접속에 사용되는 서브넷과 중복되지 않는다는 점에 유의하십시오.

직접 연결

각 빌딩 블록 내의 파일 및 블록 노드는 항상 직접 연결에 다음 표의 IP를 사용합니다.

> ⓘ 이 주소 지정 체계는 다음 규칙을 따릅니다. 세 번째 옥텟은 항상 홀수이거나 짝수이며, 이는 파일 노드가 홀수인지 아니면 짝수인지에 따라 다릅니다.

파일 노드	**IB** 포트	**IP** 주소입니다	블록 노드	**IB** 포트	물리적 **IP**	가상 **IP**
홀수(h1)	i1a	192.168.1.10	홀수(C1)	2A	192.168.1.100	192.168.1.101
홀수(h1)	i2a	192.168.3.10	홀수(C1)	2A	192.168.3.100	192.168.3.101
홀수(h1)	i3a	192.168.5.10	짝수(C2)	2A	192.168.5.100	192.168.5.101
홀수(h1)	i4a	192.168.7.10	짝수(C2)	2A	192.168.7.100	192.168.7.101
짝수(H2)	i1a	192.168.2.10	홀수(C1)	2B	192.168.2.100	192.168.2.101
짝수(H2)	i2a	192.168.4.10	홀수(C1)	2B	192.168.4.100	192.168.4.101
짝수(H2)	i3a	192.168.6.10	짝수(C2)	2B	192.168.6.100	192.168.6.101
짝수(H2)	i4a	192.168.8.10	짝수(C2)	2B	192.168.8.100	192.168.8.101

BeeGFS 클라이언트-서버 **IPoIB** 주소 지정 체계(서브넷 **2**개)

BeeGFS 클라이언트가 두 개의 InfiniBand 포트를 사용할 수 있도록 하려면 각 서브넷에서 기본 IP로 구성된 BeeGFS 서버 서비스의 절반을 사용하여 IPoIB 서브넷 두 개가 필요합니다. 이를 통해 클라이언트가 두 개의 InfiniBand 포트를 사용하여 파일 시스템에 대한 이중화 및 처리량을 극대화할 수 있습니다.

각 파일 노드에서 여러 BeeGFS 서버 서비스(관리, 메타데이터 또는 스토리지)를 실행합니다. 각 서비스가 다른 파일 노드로 독립적으로 페일오버할 수 있도록 각 서비스마다 고유한 IP 주소가 구성되며 이 주소는 두 노드 간에 자유롭게 움직일 수 있습니다(LIF라고도 함).

필수 사항은 아니지만 이 구축 환경에서 이러한 연결에 다음 IPoIB 서브넷 범위가 사용 중인 것으로 가정하며 다음 규칙을 적용하는 표준 주소 지정 체계를 정의합니다.

- 두 번째 옥텟은 파일 노드 InfiniBand 포트가 홀수인지 또는 짝수인지에 따라 항상 홀수이거나 짝수 입니다.
- BeeGFS 클러스터 IP는 항상 xxx입니다. 127.100.yyy 또는 xxx.128.100.yyy.

> 대역 내 OS 관리에 사용되는 인터페이스 외에도 클러스터 심장 박동 및 동기화를 위한 Corosync에서 추가 인터페이스를 사용할 수 있습니다. 따라서 단일 인터페이스가 손실되어도 전체 클러스터가 다운되지 않습니다.

- BeeGFS Management 서비스는 항상 xxx.yyy.101.0 또는 xxx.yyy.102.0 중 입니다.
- BeeGFS 메타데이터 서비스는 항상 xxx.yyy.101.zzz 또는 xxx.yyy.102.zzz입니다.
- BeeGFS 스토리지 서비스는 항상 xxx.yyy.103.zzz 또는 xxx.yyy.103.zzz로 제공됩니다.
- 100.xxx.1.1 ~ 100.xxx.99.255 범위의 주소는 고객용으로 예약되어 있습니다.

서브넷 **A: 100.127.0.0/16**

다음 표에는 서브넷 A:100.127.0.0/16의 범위가 나와 있습니다.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP입니다	i1b	100.127.100.1-100.127.100.255
BeeGFS 관리	i1b	100.127.101.0
BeeGFS 메타데이터	i1b 또는 i3b	100.127.101.1 - 100.127.101.255
BeeGFS 스토리지	i1b 또는 i3b	100.127.103.1 - 100.127.103.255
BeeGFS 클라이언트	(클라이언트에 따라 다름)	100.127.1.1 - 100.127.99.255

서브넷 **B: 100.128.0.0/16**

다음 표에는 서브넷 B:100.128.0.0/16의 범위가 나와 있습니다.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP입니다	i4b	100.128.100.1-100.128.100.255
BeeGFS 관리	i2b	100.128.102.0
BeeGFS 메타데이터	i2b 또는 i4b	100.128.102.1-100.128.102.255
BeeGFS 스토리지	i2b 또는 i4b	100.128.104.1 - 100.128.104.255

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클라이언트	(클라이언트에 따라 다름)	100.128.1.1-100.128.99.255

위 범위에 있는 모든 IP가 이 NetApp 검증 아키텍처에 사용되는 것은 아닙니다. 또한 IP 주소를 사전 할당하여 일관된 IP 주소 지정 체계를 사용하여 파일 시스템을 쉽게 확장할 수 있는 방법을 보여 줍니다. 이 스키마에서는 BeeGFS 파일 노드 및 서비스 ID가 잘 알려진 IP 범위의 네 번째 옥텟과 일치합니다. 필요한 경우 파일 시스템을 255개 노드 또는 서비스 이상으로 확장할 수 있습니다.

하드웨어 구축

각 구성 요소는 HDR(200GB) InfiniBand 케이블을 사용하여 두 블록 노드에 직접 연결된 검증된 x86 파일 노드 2개로 구성됩니다.

각 구성 요소에 BeeGFS 파일 노드가 2개 포함되어 있으므로 페일오버 클러스터에 쿼럼을 설정하려면 최소 2개의 구성 요소가 필요합니다. 2노드 클러스터를 구성할 수 있지만 일부 시나리오에서는 성공적인 페일오버가 발생하지 않도록 이 구성에 제한이 있습니다. 2노드 클러스터가 필요한 경우 이 구축 절차에 포함되지 않지만 3차 디바이스를 Tiebreaker로 통합할 수도 있습니다.

별도로 언급하지 않는 한, 다음 단계는 BeeGFS 메타데이터 및 스토리지 서비스 또는 스토리지 서비스만 실행하는 데 사용되는지 여부와 관계없이 클러스터의 각 구성 요소에 대해 동일합니다.

단계

1. InfiniBand 모드에서 PCIe 4.0 ConnectX-6 이중 포트 호스트 채널 어댑터(HCA) 4개로 각 BeeGFS 파일 노드를 구성하고 PCIe 슬롯 2, 3, 5 및 6에 설치합니다.
2. 이중 포트 200GB 호스트 인터페이스 카드(HIC)로 각 BeeGFS 블록 노드를 구성하고 두 스토리지 컨트롤러 각각에 HIC를 설치합니다.

   두 개의 BeeGFS 파일 노드가 BeeGFS 블록 노드 위에 있도록 구성 요소를 랙에 설치하십시오. 다음 그림은 BeeGFS 빌딩 블록에 대한 올바른 하드웨어 구성을 보여 줍니다(후면).

   []

   운영 활용 사례에 대한 전원 공급 장치 구성은 일반적으로 중복 PSU를 사용해야 합니다.

3. 필요한 경우 각 BeeGFS 블록 노드에 드라이브를 설치합니다.
   a. 빌딩 블록을 사용하여 BeeGFS 메타데이터 및 스토리지 서비스를 실행하고 작은 드라이브를 메타데이터 볼륨에 사용하는 경우 아래 그림과 같이 가장 바깥쪽 드라이브 슬롯에 채워졌는지 확인합니다.
   b. 모든 구성 요소 구성에서 드라이브 엔클로저가 완전히 채워지지 않은 경우 최적의 성능을 위해 슬롯 0–11 및 12–23에 동일한 수의 드라이브가 채워졌는지 확인하십시오.

      []

4. 파일 및 블록 노드에 케이블을 연결하려면 1m InfiniBand HDR 200GB 직접 연결 구리 케이블을 사용하여 아래 그림에 표시된 토폴로지와 일치시킵니다.

   []

여러 빌딩 블록의 노드는 직접 연결되지 않습니다. 각 구성 요소는 독립형 장치로 취급해야 하며 구성 요소 간의 모든 통신은 네트워크 스위치를 통해 이루어집니다.

5. 2m(또는 적절한 길이) InfiniBand HDR 200GB 직접 연결 구리 케이블을 사용하여 각 파일 노드의 나머지 InfiniBand 포트를 스토리지 네트워크에 사용할 InfiniBand 스위치에 케이블로 연결합니다.

   사용 중인 중복 InfiniBand 스위치가 있는 경우 다음 그림에서 녹색으로 강조 표시된 포트를 서로 다른 스위치에 케이블로 연결합니다.

   []

6. 필요한 경우 동일한 케이블 연결 지침에 따라 추가 구성 요소를 조립합니다.

   단일 랙에 구축할 수 있는 총 구성 요소 수는 각 사이트의 사용 가능한 전력 및 냉각에 따라 다릅니다.

소프트웨어 배포

파일 노드 및 블록 노드 설정

대부분의 소프트웨어 구성 작업은 NetApp에서 제공하는 Ansible 컬렉션을 사용하여 자동화되지만, 각 서버의 BMC(베이스보드 관리 컨트롤러)에서 네트워킹을 구성하고 각 컨트롤러의 관리 포트를 구성해야 합니다.

파일 노드 설정

1. 각 서버의 BMC(베이스보드 관리 컨트롤러)에서 네트워킹을 구성합니다.

   검증된 Lenovo SR665 파일 노드의 네트워킹을 구성하는 방법은 를 참조하십시오 "Lenovo ThinkSystem 설명서".

   서비스 프로세서라고도 하는 베이스보드 관리 컨트롤러(BMC)는 다양한 서버 플랫폼에 내장되어 운영 체제가 설치되어 있지 않거나 액세스할 수 없는 경우에도 원격 액세스를 제공할 수 있는 대역외 관리 기능의 일반 이름입니다. 공급업체는 일반적으로 고유한 브랜딩으로 이 기능을 마케팅합니다. 예를 들어, Lenovo SR665에서 BMC는 _Lenovo XClarity Controller(XCC)_라고 합니다.

2. 최대 성능을 위해 시스템 설정을 구성합니다.

   UEFI 설정(이전의 BIOS)을 사용하거나 많은 BMC에서 제공하는 Redfish API를 사용하여 시스템 설정을 구성합니다. 시스템 설정은 파일 노드로 사용되는 서버 모델에 따라 달라집니다.

   검증된 Lenovo SR665 파일 노드에 대한 시스템 설정을 구성하는 방법은 을 참조하십시오 "성능을 위해 시스템 설정을 조정합니다".

3. Red Hat 8.4를 설치하고 Ansible 제어 노드의 SSH 연결을 포함하여 운영 체제를 관리하는 데 사용되는 호스트 이름과 네트워크 포트를 구성합니다.

   지금은 InfiniBand 포트에 IP를 구성하지 마십시오.

   엄밀히 요구되지는 않지만, 이후의 섹션에서는 호스트 이름이 순차적으로 번호가 매겨진 것으로 간주하고(예: h1-hn) 홀수 호스트와 짝수 번호의 호스트에서 완료해야 하는 작업을 참조합니다.

4. RedHat 서브스크립션 관리자를 사용하여 공식 Red Hat 리포지토리에서 필수 패키지 설치를 허용하고 지원되는 Red Hat 버전('Ssubscription-manager release-set=8.4')으로 업데이트를 제한하려면 시스템을 등록하고 가입합니다. 자세한 내용은 을 참조하십시오 "RHEL 시스템을 등록하고 가입하는 방법" 및 "업데이트 제한 방법".
5. 고가용성을 위해 필요한 패키지가 포함된 Red Hat 리포지토리를 활성화합니다.

```
subscription-manager repo-override --repo=rhel-8-for-x86_64
-highavailability-rpms --add=enabled:1
```

6. 모든 ConnectX-6 HCA 펌웨어를 에서 권장하는 버전으로 업데이트합니다 "기술 요구 사항".

   이 업데이트는 권장 펌웨어를 번들로 제공하는 mlxup 도구 버전을 다운로드하여 실행하면 됩니다. 이 도구는 에서 다운로드할 수 있습니다 "mlxup - 업데이트 및 쿼리 유틸리티" ("사용 설명서")를 클릭합니다.

블록 노드 설정

각 컨트롤러의 관리 포트를 구성하여 EF600 블록 노드를 설정합니다.

1. 각 EF600 컨트롤러의 관리 포트를 구성합니다.

   포트 구성에 대한 지침은 로 이동하십시오 "E-Series 문서 센터 를 참조하십시오".

2. 필요에 따라 각 시스템의 스토리지 어레이 이름을 설정합니다.

   이름을 설정하면 이후 섹션에서 각 시스템을 쉽게 참조할 수 있습니다. 어레이 이름 설정에 대한 지침은 로 이동하십시오 "E-Series 문서 센터 를 참조하십시오".

> ⓘ 엄밀히 요구되지는 않지만, 후속 주제는 스토리지 배열 이름이 순차적으로 번호가 매겨진 것으로 간주하고(예: C1-CN) 홀수 대 짝수 번호의 시스템에서 완료해야 하는 단계를 참조합니다.

Ansible 제어 노드를 설정합니다

Ansible 제어 노드를 설정하려면 솔루션을 구성하는 데 사용할 수 있는 모든 파일 및 블록 노드의 관리 포트에 대한 네트워크 액세스를 갖춘 가상 머신 또는 물리적 머신을 식별해야 합니다.

다음 단계는 CentOS 8.4에서 테스트되었습니다. 선호하는 Linux 배포판에 대한 단계는 를 참조하십시오 "Ansible 설명서".

1. Python 3.9를 설치하고 올바른 버전의 PIP가 설치되어 있는지 확인합니다.

```
sudo dnf install python3.9 -y
sudo dnf install python39-pip
sudo dnf install sshpass
```

2. 심볼 링크를 생성하여 'python3' 또는 'python'이 호출될 때마다 Python 3.9 바이너리가 사용되도록 합니다.

```
sudo ln -sf /usr/bin/python3.9 /usr/bin/python3
sudo ln -sf /usr/bin/python3 /usr/bin/python
```

3. NetApp BeeGFS 컬렉션에 필요한 Python 패키지를 설치합니다.

```
python3 -m pip install ansible cryptography netaddr
```

지원되는 Ansible 버전과 필요한 모든 Python 패키지를 설치하려면 BeeGFS 컬렉션의 Readme 파일을 참조하십시오. 지원되는 버전은 에도 나와 있습니다 "기술 요구사항".

4. 올바른 버전의 Ansible 및 Python이 설치되어 있는지 확인하십시오.

```
ansible --version
ansible [core 2.11.6]
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.9/site-
packages/ansible
  ansible collection location =
/root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.9.2 (default, Mar 10 2021, 17:29:56) [GCC 8.4.1
20200928 (Red Hat 8.4.1-1)]
  jinja version = 3.0.2
  libyaml = True
```

5. Git 또는 BitBucket과 같은 소스 제어 시스템에 BeeGFS 구축을 설명하는 데 사용되는 Ansible 인벤토리를 저장한 다음 Git를 설치하여 이러한 시스템과 상호 작용합니다.

```
sudo dnf install git -y
```

6. 암호 없는 SSH를 설정합니다. 이는 Ansible에서 Ansible 제어 노드의 원격 BeeGFS 파일 노드에 액세스할 수 있는 가장 쉬운 방법입니다.
   a. 필요한 경우 Ansible 컨트롤 노드에서 ssh-keygen을 사용하여 공개 키 쌍을 생성합니다
   b. ssh-copy-id <ip_or_hostname>'을 사용하여 각 파일 노드에 대해 암호 없는 SSH를 설정합니다

      블록 노드에 암호 없는 SSH를 * 설정하지 마십시오. 이 작업은 지원되거나 필요하지 않습니다.

7. Ansible Galaxy를 사용하여 에 나열된 BeeGFS 컬렉션 버전을 설치합니다 "기술 요구사항".

   이 설치에는 NetApp SANtricity 소프트웨어 및 호스트 컬렉션과 같은 추가 Ansible 종속성이 포함됩니다.

```
ansible-galaxy collection install netapp_eseries.beegfs:==3.0.1
```

Ansible 인벤토리를 작성합니다

파일 및 블록 노드의 구성을 정의하려면 구축할 BeeGFS 파일 시스템을 나타내는 Ansible 인벤토리를 생성합니다. 인벤토리는 원하는 BeeGFS 파일 시스템을 설명하는 호스트, 그룹 및 변수를 포함합니다.

1단계**:** 모든 빌딩 블록에 대한 설정을 정의합니다

개별적으로 적용할 수 있는 구성 프로파일에 관계없이 모든 구성 요소에 적용되는 구성을 정의합니다.

시작하기 전에

- BitBucket 또는 Git와 같은 소스 제어 시스템을 사용하여 Ansible 인벤토리 및 플레이북 파일이 포함된 디렉토리의 콘텐츠를 저장합니다.
- Git가 무시해야 하는 파일을 지정하는 '.gignore' 파일을 만듭니다. 이렇게 하면 Git에 큰 파일을 저장하지 않아도 됩니다.

단계

1. Ansible 제어 노드에서 Ansible 인벤토리 및 플레이북 파일을 저장하는 데 사용할 디렉토리를 식별하십시오.

   별도로 언급하지 않는 한, 이 단계에서 만든 모든 파일과 디렉터리와 다음 단계는 이 디렉터리를 기준으로 생성됩니다.

2. 다음 하위 디렉터리를 만듭니다.

   HOST_VAR'입니다

   group_vars입니다

   '패키지'

2단계**:** 개별 파일 및 블록 노드에 대한 설정을 정의합니다

개별 파일 노드 및 개별 구성 요소 노드에 적용되는 구성을 정의합니다.

1. 'host_vars/'에서 다음 내용으로 이름이 '<HOSTNAME>.yml'인 각 BeeGFS 파일 노드에 대한 파일을 만듭니다. BeeGFS 클러스터 IP 및 호스트 이름에 대해 채울 콘텐츠에 대한 메모는 홀수와 짝수로 끝나는 것이 좋습니다.

   처음에는 파일 노드 인터페이스 이름이 여기에 나열된 것과 일치합니다(예: ib0 또는 ibs1f0). 이러한 사용자 정의 이름은 에 구성되어 있습니다 4단계: 모든 파일 노드에 적용할 구성을 정의합니다.

```
ansible_host: "<MANAGEMENT_IP>"
eseries_ipoib_interfaces:  # Used to configure BeeGFS cluster IP
addresses.
  - name: i1b
    address: 100.127.100. <NUMBER_FROM_HOSTNAME>/16
  - name: i4b
    address: 100.128.100. <NUMBER_FROM_HOSTNAME>/16
beegfs_ha_cluster_node_ips:
  - <MANAGEMENT_IP>
  - <i1b_BEEGFS_CLUSTER_IP>
  - <i4b_BEEGFS_CLUSTER_IP>
# NVMe over InfiniBand storage communication protocol information
# For odd numbered file nodes (i.e., h01, h03, ..):
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.1.10/24
    configure: true
  - name: i2a
    address: 192.168.3.10/24
    configure: true
  - name: i3a
    address: 192.168.5.10/24
    configure: true
  - name: i4a
    address: 192.168.7.10/24
    configure: true
# For even numbered file nodes (i.e., h02, h04, ..):
# NVMe over InfiniBand storage communication protocol information
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.2.10/24
    configure: true
  - name: i2a
    address: 192.168.4.10/24
    configure: true
  - name: i3a
    address: 192.168.6.10/24
    configure: true
  - name: i4a
    address: 192.168.8.10/24
    configure: true
```

BeeGFS 클러스터를 이미 구축한 경우, NVMe/IB에 사용되는 클러스터 IP 및 IP를 포함하여 정적으로 구성된 IP 주소를 추가하거나 변경하기 전에 클러스터를 중지해야 합니다. 이러한 변경 사항이 적절히 적용되고 클러스터 작업을 방해하지 않도록 이 작업이 필요합니다.

2. 'host_vars/'에서 '<HOSTNAME>.yml'이라는 이름의 각 BeeGFS 블록 노드에 대한 파일을 생성하고 다음 내용으로 채웁니다.

   홀수와 짝수로 끝나는 스토리지 배열 이름에 대한 내용을 입력할 때 특히 주의해야 합니다.

   각 블록 노드에 대해 하나의 파일을 생성하고 두 컨트롤러 중 하나의 "<management_ip>"를 지정합니다 (일반적으로 A).

```
eseries_system_name: <STORAGE_ARRAY_NAME>
eseries_system_api_url: https://<MANAGEMENT_IP>:8443/devmgr/v2/
eseries_initiator_protocol: nvme_ib
# For odd numbered block nodes (i.e., a01, a03, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101
    - 192.168.2.101
    - 192.168.1.100
    - 192.168.2.100
  controller_b:
    - 192.168.3.101
    - 192.168.4.101
    - 192.168.3.100
    - 192.168.4.100
# For even numbered block nodes (i.e., a02, a04, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.5.101
    - 192.168.6.101
    - 192.168.5.100
    - 192.168.6.100
  controller_b:
    - 192.168.7.101
    - 192.168.8.101
    - 192.168.7.100
    - 192.168.8.100
```

3단계**:** 모든 파일 및 블록 노드에 적용되어야 하는 설정을 정의합니다

그룹에 해당하는 파일 이름으로 group_vars 아래에 있는 호스트 그룹에 공통된 구성을 정의할 수 있습니다. 이렇게 하면 여러 위치에서 공유 구성이 반복되지 않습니다.

이 작업에 대해

호스트는 둘 이상의 그룹에 있을 수 있으며 런타임 시 Ansible은 변수 우선 순위 규칙에 따라 특정 호스트에 적용되는 변수를 선택합니다. (이 규칙에 대한 자세한 내용은 용 Ansible 설명서를 참조하십시오 "변수 사용"참조)

호스트 대 그룹 지정은 이 절차의 마지막을 위해 생성되는 실제 Ansible 인벤토리 파일에 정의됩니다.

단계

Ansible에서는 모든 호스트에 적용할 구성을 '모두'라는 그룹으로 정의할 수 있습니다. 다음 내용으로 group_vars/all.yml 파일을 만듭니다.

```
ansible_python_interpreter: /usr/bin/python3
beegfs_ha_ntp_server_pools:  # Modify the NTP server addressess if
desired.
  - "pool 0.pool.ntp.org iburst maxsources 3"
  - "pool 1.pool.ntp.org iburst maxsources 3"
```

4단계**:** 모든 파일 노드에 적용할 구성을 정의합니다

파일 노드의 공유 구성은 ha_cluster라는 그룹에 정의됩니다. 이 섹션의 단계에서는 group_vars/ha_cluster.yml 파일에 포함되어야 하는 구성을 작성합니다.

단계

1. 파일 맨 위에서 파일 노드의 'SUDO' 사용자로 사용할 암호를 포함하여 기본값을 정의합니다.

```
### ha_cluster Ansible group inventory file.
# Place all default/common variables for BeeGFS HA cluster resources
below.
### Cluster node defaults
ansible_ssh_user: root
ansible_become_password: <PASSWORD>
eseries_ipoib_default_hook_templates:
  - 99-multihoming.j2 # This is required when configuring additional
static IPs (for example cluster IPs) when multiple IB ports are in the
same IPoIB subnet.
# If the following options are specified, then Ansible will
automatically reboot nodes when necessary for changes to take effect:
eseries_common_allow_host_reboot: true
eseries_common_reboot_test_command: "systemctl --state=active,exited |
grep eseries_nvme_ib.service"
```

특히 프로덕션 환경에서는 암호를 일반 텍스트로 저장하지 마십시오. 대신 Ansible Vault를 사용하십시오(참조 "Ansible Vault로 콘텐츠 암호화") 또는 '--Ask-when-pass' 옵션을 선택합니다. 'Ansible_ssh_user'가 이미 'root'인 경우 Anabilities_BAREY_PASSWORD를 선택적으로 생략할 수 있습니다.

2. 필요에 따라 고가용성(HA) 클러스터의 이름을 구성하고 클러스터 내 통신을 위한 사용자를 지정합니다.

   전용 IP 주소 지정 체계를 수정하는 경우 기본 "begfs_ha_mgmtd_floating_ip"도 업데이트해야 합니다. 나중에 BeeGFS 관리 리소스 그룹에 대해 구성한 것과 일치해야 합니다.

   "begfs_ha_alert_email_list"를 사용하여 클러스터 이벤트에 대한 경고를 수신할 e-메일을 하나 이상 지정합니다.

```
### Cluster information
beegfs_ha_firewall_configure: True
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
# The following variables should be adjusted depending on the desired
configuration:
beegfs_ha_cluster_name: hacluster                    # BeeGFS HA cluster
name.
beegfs_ha_cluster_username: hacluster                # BeeGFS HA cluster
username.
beegfs_ha_cluster_password: hapassword               # BeeGFS HA cluster
username's password.
beegfs_ha_cluster_password_sha512_salt: randomSalt # BeeGFS HA cluster
username's password salt.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0           # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
      mydomain: “example.com”
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
beegfs_ha_alert_verbosity: 3
#  1) high-level node activity
#  3) high-level node activity + fencing action information + resources
(filter on X-monitor)
#  5) high-level node activity + fencing action information + resources
```

중복된 것처럼 보이지만 BeeGFS 파일 시스템을 단일 HA 클러스터 이상으로 확장하는 경우 "begfs_ha_mgmtd_floating_ip"가 중요합니다. 이후 HA 클러스터는 추가 BeeGFS 관리 서비스 없이 구축되고 첫 번째 클러스터에서 제공하는 관리 서비스를 가리키도록 구축됩니다.

3. 펜싱 에이전트를 구성합니다. (자세한 내용은 을 참조하십시오 "Red Hat High Availability 클러스터에서 펜싱을 구성합니다"참조) 다음 출력에서는 일반적인 펜싱 에이전트를 구성하는 예를 보여 줍니다. 다음 옵션 중 하나를 선택합니다.

   이 단계에서는 다음 사항에 유의하십시오.

   ◦ 기본적으로 펜싱은 활성화되어 있지만 fencing_agent_를 구성해야 합니다.

- pcmk_host_map 또는 pcmk_host_list에 지정된 '<HOSTNAME>'은(는) Ansible 인벤토리의 호스트 이름과 일치해야 합니다.
- 특히 운영 환경에서는 펜싱 없이 BeeGFS 클러스터를 실행할 수 없습니다. 이는 주로 블록 디바이스와 같은 리소스 종속성이 포함된 BeeGFS 서비스가 문제로 인해 페일오버될 때 파일 시스템 손상 또는 기타 바람직하지 않거나 예기치 않은 동작으로 이어질 수 있는 여러 노드에 의한 동시 액세스 위험이 발생하지 않도록 하기 위한 것입니다. 펜싱을 비활성화해야 하는 경우 BeeGFS HA 역할의 시작 가이드의 일반 참고를 참조하여 ha_cluster_crm_config_options ["STONITH -enabled"]"를 false 로 설정합니다.
- 사용 가능한 노드 레벨 펜싱 장치가 여러 개 있으며 BeeGFS HA 역할은 Red Hat HA 패키지 리포지토리에서 사용 가능한 펜싱 에이전트를 구성할 수 있습니다. 가능한 경우 무정전 전원 공급 장치(UPS) 또는 랙 배전 장치(rPDU)를 통해 작동하는 펜싱 에이전트를 사용합니다. BMC(베이스보드 관리 컨트롤러) 또는 서버에 내장된 기타 표시등 출력 장치와 같은 일부 펜싱 에이전트가 특정 장애 시나리오에서 Fence 요청에 응답하지 않을 수 있기 때문입니다.

```
### Fencing configuration:
# OPTION 1: To enable fencing using APC Power Distribution Units
(PDUs):
beegfs_ha_fencing_agents:
 fence_apc:
   - ipaddr: <PDU_IP_ADDRESS>
     login: <PDU_USERNAME>
     passwd: <PDU_PASSWORD>
     pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>"
# OPTION 2: To enable fencing using the Redfish APIs provided by the
Lenovo XCC (and other BMCs):
redfish: &redfish
  username: <BMC_USERNAME>
  password: <BMC_PASSWORD>
  ssl_insecure: 1 # If a valid SSL certificate is not available
specify “1”.
beegfs_ha_fencing_agents:
  fence_redfish:
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
# For details on configuring other fencing agents see
https://access.redhat.com/documentation/en-
us/red_hat_enterprise_linux/8/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Linux OS에서 권장되는 성능 조정을 활성화합니다.

일반적으로 성능 매개 변수에 대한 기본 설정은 대부분의 사용자가 찾지만 선택적으로 특정 작업 부하에 대한 기본 설정을 변경할 수 있습니다. 따라서 이러한 권장 사항은 BeeGFS 역할에 포함되지만 기본적으로 설정되어 있지 않으므로 사용자가 파일 시스템에 적용된 튜닝에 대해 알 수 있습니다.

성능 조정을 활성화하려면 다음을 지정하십시오.

```
### Performance Configuration:
beegfs_ha_enable_performance_tuning: True
```

5. (선택 사항) 필요에 따라 Linux OS에서 성능 조정 매개 변수를 조정할 수 있습니다.

   조정할 수 있는 사용 가능한 튜닝 매개 변수의 전체 목록은 에서 BeeGFS HA 역할의 성능 조정 기본값 섹션을 참조하십시오 "E-Series BeeGFS GitHub 사이트". 이 파일의 클러스터에 있는 모든 노드 또는 개별 노드에 대한 'host_vars' 파일에 대해 기본값을 재정의할 수 있습니다.

6. 블록과 파일 노드 간에 전체 200GB/HDR 연결을 허용하려면 Mellanox Open Fabrics Enterprise Distribution(MLNX_OFED)의 OpenSM(Open Subnet Manager) 패키지를 사용하십시오. (받은 편지함인 OpenSM 패키지는 필요한 가상화 기능을 지원하지 않습니다.) Ansible을 사용하여 구축할 수도 있지만, 먼저 BeeGFS 역할을 실행하는 데 사용되는 Ansible 제어 노드에 원하는 패키지를 다운로드해야 합니다.

   a. 컬링이나 원하는 도구를 사용하여 Mellanox 웹 사이트의 기술 요구 사항 섹션에 나열된 OpenSM 버전의 패키지를 "packages/" 디렉토리로 다운로드합니다. 예를 들면 다음과 같습니다.

      ```
      curl -o packages/opensm-libs-5.9.0.MLNX20210617.c9f2ade-
      0.1.54103.x86_64.rpm
      https://linux.mellanox.com/public/repo/mlnx_ofed/5.4-
      1.0.3.0/rhel8.4/x86_64/opensm-libs-5.9.0.MLNX20210617.c9f2ade-
      0.1.54103.x86_64.rpm

      curl -o packages/opensm-5.9.0.MLNX20210617.c9f2ade-
      0.1.54103.x86_64.rpm
      https://linux.mellanox.com/public/repo/mlnx_ofed/5.4-
      1.0.3.0/rhel8.4/x86_64/opensm-5.9.0.MLNX20210617.c9f2ade-
      0.1.54103.x86_64.rpm
      ```

   b. group_vars/ha_cluster.yml에 다음 파라미터를 입력합니다(필요에 따라 패키지 조정).

```
### OpenSM package and configuration information
eseries_ib_opensm_allow_upgrades: true
eseries_ib_opensm_skip_package_validation: true
eseries_ib_opensm_rhel_packages: []
eseries_ib_opensm_custom_packages:
  install:
    - files:
        add:
          "packages/opensm-libs-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm": "/tmp/"
          "packages/opensm-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm": "/tmp/"
    - packages:
        add:
          - /tmp/opensm-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm
          - /tmp/opensm-libs-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm
  uninstall:
    - packages:
        remove:
          - opensm
          - opensm-libs
      files:
        remove:
          - /tmp/opensm-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm
          - /tmp/opensm-libs-5.9.0.MLNX20210617.c9f2ade-
0.1.54103.x86_64.rpm
eseries_ib_opensm_options:
  virt_enabled: "2"
```

7. 논리적 InfiniBand 포트 식별자를 기본 PCIe 디바이스에 일관되게 매핑하도록 'udev' 규칙을 구성합니다.

   udev 규칙은 BeeGFS 파일 노드로 사용되는 각 서버 플랫폼의 PCIe 토폴로지에 고유해야 합니다.

   검증된 파일 노드에 대해 다음 값을 사용합니다.

```
### Ensure Consistent Logical IB Port Numbering
# OPTION 1: Lenovo SR665 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:41:00.0": i1a
  "0000:41:00.1": i1b
  "0000:01:00.0": i2a
  "0000:01:00.1": i2b
  "0000:a1:00.0": i3a
  "0000:a1:00.1": i3b
  "0000:81:00.0": i4a
  "0000:81:00.1": i4b

# Note: At this time no other x86 servers have been qualified.
Configuration for future qualified file nodes will be added here.
```

8. (선택 사항) 메타데이터 대상 선택 알고리즘을 업데이트합니다.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

검증 테스트에서는 일반적으로 성능 벤치마킹 중에 테스트 파일이 모든 BeeGFS 스토리지 대상에 고르게 분산되도록 하기 위해 "랜덤 로빈"이 사용되었습니다(벤치마킹을 위한 자세한 내용은 BeeGFS 사이트 참조) "BeeGFS 시스템을 벤치마킹합니다")를 클릭합니다. 실제 환경에서 사용하면 낮은 번호의 대상이 높은 번호의 목표보다 빠르게 채워질 수 있습니다. 기본 '무작위 배정' 값을 사용하기만 하면 사용 가능한 모든 대상을 활용하는 동시에 우수한 성능을 제공하는 것으로 나타났습니다.

5단계**:** 공통 블록 노드에 대한 구성을 정의합니다

블록 노드의 공유 구성은 eseries_storage_systems라는 그룹에 정의되어 있습니다. 이 섹션의 단계에서는 group_vars/eseries_storage_systems.yml 파일에 포함되어야 하는 구성을 작성합니다.

단계

1. Ansible 연결을 로컬로 설정하고 시스템 암호를 제공하며 SSL 인증서를 확인해야 하는지 여부를 지정합니다. (일반적으로 Ansible은 SSH를 사용하여 관리 호스트에 연결하지만, 블록 노드로 사용되는 NetApp E-Series 스토리지 시스템의 경우 모듈은 통신에 REST API를 사용합니다.) 파일 맨 위에 다음을 추가합니다.

```
### eseries_storage_systems Ansible group inventory file.
# Place all default/common variables for NetApp E-Series Storage Systems
here:
ansible_connection: local
eseries_system_password: <PASSWORD>
eseries_validate_certs: false
```

암호를 일반 텍스트로 나열하는 것은 권장되지 않습니다. Ansible 볼트를 사용하거나 '- Extra-VAR'을 사용하여 Ansible을 실행할 때 'eseries_system_password'를 제공하십시오.

2. 최적의 성능을 보장하기 위해 에 블록 노드에 대해 나열된 버전을 설치합니다 "기술 요구사항".

   에서 해당 파일을 다운로드합니다 "NetApp Support 사이트". 수동으로 업그레이드하거나 Ansible 제어 노드의 'packages/' 디렉토리에 추가한 다음, Ansible을 사용하여 업그레이드하려면 "eseries_storage_systems.yml"에 다음 매개 변수를 입력합니다.

   ```
   # Firmware, NVSRAM, and Drive Firmware (modify the filenames as needed):
   eseries_firmware_firmware: "packages/RCB_11.70.2_6000_61b1131d.dlp"
   eseries_firmware_nvsram: "packages/N6000-872834-D06.dlp"
   ```

3. 에서 Block 노드에 설치된 드라이브에 사용할 수 있는 최신 드라이브 펌웨어를 다운로드하여 설치합니다 "NetApp Support 사이트". 수동으로 업그레이드하거나 Ansible 제어 노드의 'packages/' 디렉토리에 추가한 다음, Ansible을 사용하여 업그레이드하려면 "eseries_storage_systems.yml"에 다음 매개 변수를 입력합니다.

   ```
   eseries_drive_firmware_firmware_list:
     - "packages/<FILENAME>.dlp"
   eseries_drive_firmware_upgrade_drives_online: true
   ```

   eseries_drive_firmware_upgrade_drives_online을 "false"로 설정하면 업그레이드 속도가 빨라지지만 BeeGFS가 구축되기 전에는 수행할 수 없습니다. 이 설정은 응용 프로그램 오류를 방지하기 위해 업그레이드 전에 드라이브에 대한 모든 I/O를 중지하도록 하기 때문입니다. 볼륨을 구성하기 전에 온라인 드라이브 펌웨어 업그레이드를 수행하는 것이 여전히 빠르지만 나중에 문제가 발생하지 않도록 항상 이 값을 "참"으로 설정하는 것이 좋습니다.

4. 성능을 최적화하려면 글로벌 구성을 다음과 같이 변경합니다.

   ```
   # Global Configuration Defaults
   eseries_system_cache_block_size: 32768
   eseries_system_cache_flush_threshold: 80
   eseries_system_default_host_type: linux dm-mp
   eseries_system_autoload_balance: disabled
   eseries_system_host_connectivity_reporting: disabled
   eseries_system_controller_shelf_id: 99 # Required.
   ```

5. 최적의 볼륨 프로비저닝 및 동작을 위해 다음 매개 변수를 지정합니다.

```
# Storage Provisioning Defaults
eseries_volume_size_unit: pct
eseries_volume_read_cache_enable: true
eseries_volume_read_ahead_enable: false
eseries_volume_write_cache_enable: true
eseries_volume_write_cache_mirror_enable: true
eseries_volume_cache_without_batteries: false
eseries_storage_pool_usable_drives:
"99:0,99:23,99:1,99:22,99:2,99:21,99:3,99:20,99:4,99:19,99:5,99:18,99:6,
99:17,99:7,99:16,99:8,99:15,99:9,99:14,99:10,99:13,99:11,99:12"
```

'eseries_storage_pool_usable_drives'에 지정된 값은 NetApp EF600 블록 노드에만 해당되며 드라이브가 새 볼륨 그룹에 할당되는 순서를 제어합니다. 이 주문을 통해 각 그룹에 대한 입출력이 백엔드 드라이브 채널에 균등하게 분산됩니다.

BeeGFS 구성 요소에 대한 **Ansible** 인벤토리를 정의합니다

일반 Ansible 인벤토리 구조를 정의한 후 BeeGFS 파일 시스템의 각 구성 요소에 대한 구성을 정의합니다.

이러한 구축 지침은 관리, 메타데이터 및 스토리지 서비스를 포함한 기본 구성 요소로 구성된 파일 시스템, 메타데이터 및 스토리지 서비스를 포함하는 두 번째 구성 요소, 세 번째 스토리지 전용 구성 요소로 이루어진 파일 시스템을 구축하는 방법을 보여 줍니다.

이 단계에서는 전체 BeeGFS 파일 시스템의 요구 사항을 충족하도록 NetApp BeeGFS 구성 요소를 구성하는 데 사용할 수 있는 모든 일반 구성 프로필을 보여 주기 위한 것입니다.

이 섹션과 후속 섹션에서 필요에 따라 조정하여 구축할 BeeGFS 파일 시스템을 나타내는 인벤토리를 생성합니다. 특히, 스토리지 네트워크에 대해 각 블록 또는 파일 노드를 나타내는 Ansible 호스트 이름과 원하는 IP 주소 지정 스키마를 사용하여 BeeGFS 파일 노드 및 클라이언트의 수에 맞게 확장할 수 있도록 합니다.

1단계**: Ansible** 인벤토리 파일을 만듭니다

단계

1. 새 'inventory.yml' 파일을 만든 다음, 필요에 따라 'eseries_storage_systems' 아래에 있는 호스트를 대체하여 구축의 블록 노드를 나타냅니다. 이름은 host_VAR/<filename>.yml에 사용되는 이름과 일치해야 합니다.

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        ictad22a01:
        ictad22a02:
        ictad22a03:
        ictad22a04:
        ictad22a05:
        ictad22a06:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
```

다음 섹션에서는 클러스터에서 실행할 BeeGFS 서비스를 나타내는 "ha_cluster" 아래에 Ansible 그룹을 추가로 생성합니다.

2단계**:** 관리**,** 메타데이터 및 스토리지 구성 요소에 대한 인벤토리를 구성합니다

클러스터 또는 기본 구성 요소에서 첫 번째 구성 요소는 메타데이터 및 스토리지 서비스와 함께 BeeGFS 관리 서비스를 포함해야 합니다.

단계

1. inventory.yml에서 ha_cluster:Children 아래에 다음 매개 변수를 입력합니다.

```
      # ictad22h01/ictad22h02 HA Pair (mgmt/meta/storage building
block):
        mgmt:
          hosts:
            ictad22h01:
            ictad22h02:
        meta_01:
          hosts:
            ictad22h01:
            ictad22h02:
        stor_01:
          hosts:
            ictad22h01:
            ictad22h02:
        meta_02:
          hosts:
            ictad22h01:
            ictad22h02:
```

```
        stor_02:
          hosts:
            ictad22h01:
            ictad22h02:
        meta_03:
          hosts:
            ictad22h01:
            ictad22h02:
        stor_03:
          hosts:
            ictad22h01:
            ictad22h02:
        meta_04:
          hosts:
            ictad22h01:
            ictad22h02:
        stor_04:
          hosts:
            ictad22h01:
            ictad22h02:
        meta_05:
          hosts:
            ictad22h02:
            ictad22h01:
        stor_05:
          hosts:
            ictad22h02:
            ictad22h01:
        meta_06:
          hosts:
            ictad22h02:
            ictad22h01:
        stor_06:
          hosts:
            ictad22h02:
            ictad22h01:
        meta_07:
          hosts:
            ictad22h02:
            ictad22h01:
        stor_07:
          hosts:
            ictad22h02:
            ictad22h01:
        meta_08:
          hosts:
```

```
            ictad22h02:
            ictad22h01:
        stor_08:
          hosts:
            ictad22h02:
            ictad22h01:
```

2. 'group_vars/mgmt.yml' 파일을 생성하고 다음을 포함합니다.

```
# mgmt - BeeGFS HA Management Resource Group
# OPTIONAL: Override default BeeGFS management configuration:
# beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
#  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
floating_ips:
  - i1b: 100.127.101.0/16
  - i2b: 100.128.102.0/16
beegfs_service: management
beegfs_targets:
  ictad22a01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 1
            owning_controller: A
```

3. group_vars/ 아래에서 다음 템플릿을 사용하여 META_08을 통해 자원 그룹 META_01에 대한 파일을 만든 다음 아래 표를 참조하여 각 서비스에 대한 자리 표시자 값을 입력합니다.

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET> # Example: i1b:192.168.120.1/16
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 21.25 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

볼륨 크기는 전체 스토리지 풀(볼륨 그룹이라고도 함)의 백분율로 지정됩니다. SSD 오버 프로비저닝을 위해 각 풀에 여유 용량을 남겨 두는 것이 좋습니다(자세한 내용은 참조) "NetApp EF600 어레이 소개")를 클릭합니다. 스토리지 풀 'begfs_m1_m2_m5_m6'도 관리 서비스를 위해 풀 용량의 1%를 할당합니다. 따라서 스토리지 풀의 메타데이터 볼륨에 대해 1.92TB 또는 3.84TB 드라이브를 사용할 때 Beegfs_M1_m2_M5_M6의 경우 이 값을 21.25로 설정하고, 7.65TB 드라이브의 경우 이 값을 22.25로 설정하고, 15.3TB 드라이브의 경우 이 값을 23.75로 설정합니다.

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
meta_01.yml	8015	i1b: 100.127.101. 1 / 16 i2b: 100.128.102. 1 / 16	0	ictad22a01	Beegfs_m1_ m2_m5_m6	A
meta_02.yml	8025	i2b:100.128.1 02.2/16 i1b:100.127.1 01.2/16	0	ictad22a01	Beegfs_m1_ m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.1 01.3/16 i4b:100.128.1 02.3/16	1	ictad22a02	Beegfs_m3_ M4_M7_M8	A

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
meta_04.yml	8045	i4b:100.128.102.4/16 i3b:100.127.101.4/16	1	ictad22a02	Beegfs_m3_M4_M7_M8	B
meta_05.yml	8055	i1b: 100.127.101.5 / 16 i2b: 100.128.102.5 / 16	0	ictad22a01	Beegfs_m1_m2_m5_m6	A
meta_06.yml	8065	i2b:100.128.102.6/16 i1b:100.127.101.6/16	0	ictad22a01	Beegfs_m1_m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.101.7 / 16 i4b:100.128.102.7 / 16	1	ictad22a02	Beegfs_m3_M4_M7_M8	A
META_08.월	8085	i4b:100.128.102.8/16 i3b:100.127.101.8/16	1	ictad22a02	Beegfs_m3_M4_M7_M8	B

4. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_08'을 통해 리소스 그룹 tor_01에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource
Groupbeegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!             owning_controller:
<OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율".

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_01.대칭	8013	i1b: 100.127.103. 1 / 16 i2b: 100.128.104. 1 / 16	0	ictad22a01	Beegfs_s1_s 2	A
STOR_02.월	8023	i2b:100.128.1 04.2 / 16 i1b:100.127.1 03.2 / 16	0	ictad22a01	Beegfs_s1_s 2	B
STOR_03.월	8033	i3b:100.127.1 03.3 / 16 i4b:100.128.1 04.3 / 16	1	ictad22a02	Beegfs_S3_S 4	A
STOR_04.yml	8043	i4b:100.128.1 04.4/16 i3b:100.127.1 03.4/16	1	ictad22a02	Beegfs_S3_S 4	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_05.월	8053	i1b: 100.127.103. 5 / 16 i2b: 100.128.104. 5 / 16	0	ictad22a01	Beegfs_S5_S 6	A
STOR_06.대칭	8063	i2b:100.128.1 04.6/16 i1b:100.127.1 03.6/16	0	ictad22a01	Beegfs_S5_S 6	B
STOR_07.월	8073	i3b:100.127.1 03.7 / 16 i4b:100.128.1 04.7 / 16	1	ictad22a02	Beegfs_S7_s 8	A
STOR_08.월	8083	i4b:100.128.1 04.8 / 16 i3b:100.127.1 03.8/16	1	ictad22a02	Beegfs_S7_s 8	B

3단계**:** 메타데이터 **+** 스토리지 구성 요소에 대한 인벤토리를 구성합니다

다음 단계에서는 BeeGFS 메타데이터 + 스토리지 구성 요소에 대한 Ansible 인벤토리를 설정하는 방법을 설명합니다.

단계

1. 'inventory.yml'에서 기존 설정 아래에 다음 파라미터를 입력합니다.

```
        meta_09:
          hosts:
            ictad22h03:
            ictad22h04:
        stor_09:
          hosts:
            ictad22h03:
            ictad22h04:
        meta_10:
          hosts:
            ictad22h03:
            ictad22h04:
        stor_10:
          hosts:
            ictad22h03:
            ictad22h04:
        meta_11:
          hosts:
            ictad22h03:
```

```
        ictad22h04:
    stor_11:
      hosts:
        ictad22h03:
        ictad22h04:
    meta_12:
      hosts:
        ictad22h03:
        ictad22h04:
    stor_12:
      hosts:
        ictad22h03:
        ictad22h04:
    meta_13:
      hosts:
        ictad22h04:
        ictad22h03:
    stor_13:
      hosts:
        ictad22h04:
        ictad22h03:
    meta_14:
      hosts:
        ictad22h04:
        ictad22h03:
    stor_14:
      hosts:
        ictad22h04:
        ictad22h03:
    meta_15:
      hosts:
        ictad22h04:
        ictad22h03:
    stor_15:
      hosts:
        ictad22h04:
        ictad22h03:
    meta_16:
      hosts:
        ictad22h04:
        ictad22h03:
    stor_16:
      hosts:
        ictad22h04:
        ictad22h03:
```

2. group_vars/ 아래에서 다음 템플릿을 사용하여 META_16을 통해 자원 그룹 META_09 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.5 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율".

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
META_09.대칭	8015	i1b: 100.127.101.9 / 16 i2b: 100.128.102.9 / 16	0	ictad22a03	Beegfs_m9_M10_M13_M14	A
META_10.월	8025	i2b:100.128.102.10/16 i1b:100.127.101.10/16	0	ictad22a03	Beegfs_m9_M10_M13_M14	B
meta_11.yml	8035	i3b:100.127.101.11 / 16 i4b:100.128.102.11 / 16	1	ictad22a04	Beegfs_M11_M12_M15_M16	A
META_12.월	8045	i4b:100.128.102.12/16 i3b:100.127.101.12/16	1	ictad22a04	Beegfs_M11_M12_M15_M16	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
META_13.월	8055	i1b:100.127.101.13/16 i2b:100.128.102.13/16	0	ictad22a03	Beegfs_m9_M10_M13_M14	A
meta_14.yml	8065	i2b:100.128.102.14 / 16 i1b:100.127.101.14 / 16	0	ictad22a03	Beegfs_m9_M10_M13_M14	B
META_15.월	8075	i3b:100.127.101.15/16 i4b:100.128.102.15/16	1	ictad22a04	Beegfs_M11_M12_M15_M16	A
meta_16.yml	8085	i4b:100.128.102.16/16 i3b:100.127.101.16/16	1	ictad22a04	Beegfs_M11_M12_M15_M16	B

3. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_16'을 통해 리소스 그룹 tor_09에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율"...

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_09.대칭	8013	i1b: 100.127.103. 9 / 16 i2b: 100.128.104. 9 / 16	0	ictad22a03	Beegfs_S9_S 10	A
STOR_10.월	8023	i2b:100.128.1 04.10/16 i1b:100.127.1 03.10/16	0	ictad22a03	Beegfs_S9_S 10	B
STOR_11.월	8033	i3b:100.127.1 03.11 / 16 i4b:100.128.1 04.11 / 16	1	ictad22a04	Beegfs_s11_ s12	A
STOR_12.월	8043	i4b:100.128.1 04.12/16 i3b:100.127.1 03.12/16	1	ictad22a04	Beegfs_s11_ s12	B
STOR_13.월	8053	i1b: 100.127.103. 13 / 16 i2b: 100.128.104. 13 / 16	0	ictad22a03	Beegfs_S13_ s14	A
STOR_14.월	8063	i2b:100.128.1 04.14 / 16 i1b:100.127.1 03.14 / 16	0	ictad22a03	Beegfs_S13_ s14	B
STOR_15.월	8073	i3b:100.127.1 03.15 / 16 i4b:100.128.1 04.15 / 16	1	ictad22a04	Beegfs_s15_ S16	A
STOR_16.월	8083	i4b:100.128.1 04.16/16 i3b:100.127.1 03.16/16	1	ictad22a04	Beegfs_s15_ S16	B

4단계**:** 스토리지 전용 구성 요소에 대한 인벤토리를 구성합니다

다음 단계에서는 BeeGFS 스토리지 전용 구성 요소에 대한 Ansible 인벤토리를 설정하는 방법을 설명합니다. 메타데이터 + 스토리지에 대한 구성을 설정하는 것과 스토리지 전용 구성 블록에 대한 구성을 설정하는 것의 주된 차이점은 모든 메타데이터 리소스 그룹이 생략되고 각 스토리지 풀에 대해 "criteria_drive_count"를 10에서 12로 변경하는 것입니다.

단계

1. 'inventory.yml'에서 기존 설정 아래에 다음 파라미터를 입력합니다.

```
    # ictad22h05/ictad22h06 HA Pair (storage only building block):
      stor_17:
        hosts:
          ictad22h05:
          ictad22h06:
      stor_18:
        hosts:
          ictad22h05:
          ictad22h06:
      stor_19:
        hosts:
          ictad22h05:
          ictad22h06:
      stor_20:
        hosts:
          ictad22h05:
          ictad22h06:
      stor_21:
        hosts:
          ictad22h06:
          ictad22h05:
      stor_22:
        hosts:
          ictad22h06:
          ictad22h05:
      stor_23:
        hosts:
          ictad22h06:
          ictad22h05:
      stor_24:
        hosts:
          ictad22h06:
          ictad22h05:
```

2. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_24'를 통해 리소스 그룹 tor_17에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 12
        common_volume_configuration:
          segment_size_kb: 512
        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50
            owning_controller: <OWNING CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율".

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_17.월	8013	i1b: 100.127.103. 17 / 16 i2b: 100.128.104. 17 / 16	0	ictad22a05	Beegfs_S17_ s18	A
STOR_18.월	8023	i2b:100.128.1 04.18 / 16 i1b:100.127.1 03.18 / 16	0	ictad22a05	Beegfs_S17_ s18	B
STOR_19.대칭	8033	i3b:100.127.1 03.19 / 16 i4b:100.128.1 04.19 / 16	1	ictad22a06	Beegfs_S19_ S20	A
STOR_20.월	8043	i4b:100.128.1 04.20 / 16 i3b:100.127.1 03.20 / 16	1	ictad22a06	Beegfs_S19_ S20	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_21.대칭	8053	i1b: 100.127.103. 21 / 16 i2b: 100.128.104. 21 / 16	0	ictad22a05	Beegfs_s21_ S22	A
STOR_22.월	8063	i2b:100.128.1 04.22/16 i1b:100.127.1 03.22/16	0	ictad22a05	Beegfs_s21_ S22	B
STOR_23.월	8073	i3b:100.127.1 03.23 / 16 i4b:100.128.1 04.23 / 16	1	ictad22a06	Beegfs_S23_ S24	A
STOR_24.월	8083	i4b:100.128.1 04.24 / 16 i3b:100.127.1 03.24 / 16	1	ictad22a06	Beegfs_S23_ S24	B

BeeGFS 구축

구성 배포 및 관리에는 Ansible이 실행해야 하는 작업이 포함된 하나 이상의 플레이북을 실행해야 전체 시스템을 원하는 상태로 되돌릴 수 있습니다.

모든 작업이 단일 Playbook에 포함될 수 있지만 복잡한 시스템의 경우 이를 관리하기가 매우 복잡해집니다. Ansible을 사용하면 재사용 가능한 플레이북 및 관련 콘텐츠(예: 기본 변수, 작업, 처리기)를 패키지하는 방법으로 역할을 만들고 배포할 수 있습니다. 자세한 내용은 용 Ansible 설명서를 참조하십시오 "역할".

역할은 종종 관련 역할 및 모듈이 포함된 Ansible 컬렉션의 일부로 배포됩니다. 따라서, 이러한 플레이북은 다양한 NetApp E-Series Ansible 컬렉션에서 주로 다양한 역할을 하지만

ⓘ 2노드 클러스터를 사용하여 쿼럼을 설정할 때 문제를 완화하기 위해 별도의 쿼럼 장치를 Tiebreaker로 구성하지 않는 한 현재 BeeGFS를 구축하려면 최소 2개의 구성 요소(4개의 파일 노드)가 필요합니다.

단계

1. 새로운 '플레이북.yml' 파일을 생성하고 다음을 포함합니다.

```
# BeeGFS HA (High Availability) cluster playbook.
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

```
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
  pre_tasks:
    - name: Ensure a supported version of Python is available on all
file nodes.
      block:
        - name: Check if python is installed.
          failed_when: false
          changed_when: false
          raw: python --version
          register: python_version
        - name: Check if python3 is installed.
          raw: python3 --version
          failed_when: false
          changed_when: false
          register: python3_version
          when: 'python_version["rc"] != 0 or (python_version["stdout"]
| regex_replace("Python ", "")) is not version("3.0", ">=")'
        - name: Install python3 if needed.
          raw: |
            id=$(grep "^ID=" /etc/*release* | cut -d= -f 2 | tr -d '"')
            case $id in
              ubuntu) sudo apt install python3 ;;
              rhel|centos) sudo yum -y install python3 ;;
              sles) sudo zypper install python3 ;;
            esac
          args:
            executable: /bin/bash
          register: python3_install
          when: python_version['rc'] != 0 and python3_version['rc'] != 0
          become: true
        - name: Create a symbolic link to python from python3.
          raw: ln -s /usr/bin/python3 /usr/bin/python
          become: true
          when: python_version['rc'] != 0
      when: inventory_hostname not in
groups[beegfs_ha_ansible_storage_group]
    - name: Verify any provided tags are supported.
      fail:
        msg: "{{ item }} tag is not a supported BeeGFS HA tag. Rerun
your playbook command with --list-tags to see all valid playbook tags."
      when: 'item not in ["all", "storage", "beegfs_ha",
"beegfs_ha_package", "beegfs_ha_configure",
```

```
"beegfs_ha_configure_resource", "beegfs_ha_performance_tuning",
"beegfs_ha_backup", "beegfs_ha_client"]'
      loop: "{{ ansible_run_tags }}"
  tasks:
    - name: Verify before proceeding.
      pause:
        prompt: "Are you ready to proceed with running the BeeGFS HA
role? Depending on the size of the deployment and network performance
between the Ansible control node and BeeGFS file and block nodes this
can take awhile (10+ minutes) to complete."
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_2
```

ⓘ 이 플레이북에서는 파일 노드에 Python 3이 설치되어 있는지 확인하는 몇 가지 "pre_tasks"를 실행하고 제공되는 Ansible 태그가 지원되는지 확인합니다.

2. BeeGFS를 배포할 준비가 되면 재고 및 플레이북 파일과 함께 'Ansible-Playbook' 명령을 사용하십시오.

   구축 과정에서 모든 "pre_tasks"를 실행한 다음 실제 BeeGFS 구축을 진행하기 전에 사용자 확인을 묻는 메시지가 표시됩니다.

   다음 명령을 실행하여 필요에 따라 포크 수를 조정합니다(아래 참고 참조).

   ```
   ansible-playbook -i inventory.yml playbook.yml --forks 20
   ```

   ⓘ 특히 대규모 배포의 경우 Ansible에서 병렬로 구성하는 호스트의 수를 늘리려면 '포크' 매개 변수를 사용하여 기본 포크 수(5)를 재정의하는 것이 좋습니다. (자세한 내용은 을 참조하십시오 "Ansible 성능 조정" 및 "플레이북 실행 제어"참조) 최대 값 설정은 Ansible 컨트롤 노드에서 사용할 수 있는 처리 능력에 따라 달라집니다. 위의 20개 예는 CPU 4개(인텔® 제온® 골드 6146 CPU @ 3.20GHz)가 장착된 가상 Ansible 컨트롤 노드에서 실행되었습니다.

   Ansible 제어 노드와 BeeGFS 파일 및 블록 노드 간의 구축 및 네트워크 성능 크기에 따라 구축 시간이 달라질 수 있습니다.

BeeGFS 클라이언트를 구성합니다

컴퓨팅 또는 GPU 노드와 같이 BeeGFS 파일 시스템에 액세스해야 하는 모든 호스트에 BeeGFS 클라이언트를 설치하고 구성해야 합니다. 이 작업에서는 Ansible 및 BeeGFS 컬렉션을 사용할 수 있습니다.

단계

1. 필요한 경우, Ansible 제어 노드에서 BeeGFS 클라이언트로 구성하려는 각 호스트에 대해 암호 없는 SSH를 설정합니다.

   'ssh-copy-id <user>@<HOSTNAME_OR_IP>'를 참조하십시오

2. 'host_vars/'에서 다음 내용으로 이름이 '<HOSTNAME>.yml'인 각 BeeGFS 클라이언트에 대한 파일을 만들어 사용자 환경에 맞는 올바른 정보로 자리 표시자 텍스트를 채웁니다.

```
# BeeGFS Client
ansible_host: <MANAGEMENT_IP>
# OPTIONAL: If you want to use the NetApp E-Series Host Collection's
IPoIB role to configure InfiniBand interfaces for clients to connect to
BeeGFS file systems:
eseries_ipoib_interfaces:
  - name: <INTERFACE>
    address: <IP>/<SUBNET_MASK> # Example: 100.127.1. 1/16
  - name: <INTERFACE>0
    address: <IP>/<SUBNET_MASK>
```

현재 각 클라이언트에서 두 개의 InfiniBand 인터페이스를 구성해야 하며, 각 인터페이스는 두 스토리지 IPoIB 서브넷에 하나씩 있어야 합니다. 여기에 나열된 각 BeeGFS 서비스에 대해 서브넷 및 권장 범위의 예를 사용하는 경우 클라이언트는 "100.127.1" 범위에서 하나의 인터페이스를 구성해야 합니다. 100.127.99.255까지, 100.128.1로. 0에서 100.128까지. 99.255".

3. 새 파일 'client_inventory.yml'를 만든 다음 맨 위에 다음 매개 변수를 입력합니다.

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER> # This is the user Ansible should use to
connect to each client.
    ansible_become_password: <PASSWORD> # This is the password Ansible
will use for privilege escalation, and requires the ansible_ssh_user be
root, or have sudo privileges.
The defaults set by the BeeGFS HA role are based on the testing
performed as part of this NetApp Verified Architecture and differ from
the typical BeeGFS client defaults.
```

암호를 일반 텍스트로 저장하지 마십시오. 대신 Ansible Vault를 사용하십시오(의 Ansible 설명서 참조) "Ansible Vault로 콘텐츠 암호화") 또는 플레이북이 실행될 때 '--Ask-when-pass' 옵션을 사용합니다.

4. 'client_inventory.yml' 파일에서 Beegfs_clients' 그룹 아래에 BeeGFS 클라이언트로 구성해야 하는 모든 호스트를 나열한 다음 BeeGFS 클라이언트 커널 모듈을 구축하는 데 필요한 추가 구성을 지정합니다.

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        ictad21h01:
        ictad21h02:
        ictad21h03:
        ictad21h04:
        ictad21h05:
        ictad21h06:
        ictad21h07:
        ictad21h08:
        ictad21h09:
        ictad21h10:
      vars:
        # OPTION 1: If you're using the Mellanox OFED drivers and they
are already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        beegfs_client_ofed_enable: True
        beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"
        # OPTION 2: If you're using inbox IB/RDMA drivers and they are
already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        # OPTION 3: If you want to use inbox IB/RDMA drivers and need
them installed/configured.
        eseries_ib_skip: False # Default value.
        beegfs_client_ofed_enable: False # Default value.
```

Mellanox OFED 드라이버를 사용하는 경우, "begfs_client_OFED_include_path"가 Linux 설치를 위한 올바른 "헤더 포함 경로"를 가리키는지 확인하십시오. 자세한 내용은 의 BeeGFS 설명서를 참조하십시오 "RDMA 지원".

5. client_inventory.yml 파일에 미리 정의된 VAR의 하단에 마운트할 BeeGFS 파일 시스템을 나열합니다.

```
        beegfs_client_mounts:
          - sysMgmtdHost: 100.127.101.0 # Primary IP of the BeeGFS
management service.
            mount_point: /mnt/beegfs    # Path to mount BeeGFS on the
client.
            connInterfaces:
              - <INTERFACE> # Example: ibs4f1
              - <INTERFACE>
            beegfs_client_config:
              # Maximum number of simultaneous connections to the same
node.
              connMaxInternodeNum: 128 # BeeGFS Client Default: 12
              # Allocates the number of buffers for transferring IO.
              connRDMABufNum: 36 # BeeGFS Client Default: 70
              # Size of each allocated RDMA buffer
              connRDMABufSize: 65536 # BeeGFS Client Default: 8192
              # Required when using the BeeGFS client with the shared-
disk HA solution.
              # This does require BeeGFS targets be mounted in the
default "sync" mode.
              # See the documentation included with the BeeGFS client
role for full details.
              sysSessionChecksEnabled: false
```

Beegfs_client_config는 테스트된 설정을 나타냅니다. 모든 옵션에 대한 종합적인 개요는 netapp_eseries.beegfs` 컬렉션의 "begfs_client" 역할에 포함된 설명서를 참조하십시오. 여기에는 여러 개의 BeeGFS 파일 시스템을 마운트하거나 동일한 BeeGFS 파일 시스템을 여러 번 마운트하는 방법에 대한 세부 정보가 포함됩니다.

6. 새 'client_Playbook.yml' 파일을 만든 후 다음 매개 변수를 입력합니다.

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

필요한 IB/RDMA 드라이버와 IP를 해당 IPoIB 인터페이스에 이미 설치한 경우 'NetApp_eseries.host' 수집 및 'IPoIB' 역할을 가져오지 마십시오.

7. 클라이언트를 설치 및 구축하고 BeeGFS를 마운트하려면 다음 명령을 실행합니다.

```
ansible-playbook -i client_inventory.yml client_playbook.yml
```

8. BeeGFS 파일 시스템을 운영 환경에 배치하기 전에 모든 클라이언트에 로그인하고 "begfs-fsck—checkfs"를 실행하여 모든 노드에 연결할 수 있고 보고된 문제가 없는지 확인하는 것이 좋습니다.

5가지 구성 요소 이상으로 확장

5개의 빌딩 블록(10개의 파일 노드) 이상으로 확장하기 위해 페이스 메이커 및 Corosync를 구성할 수 있습니다. 그러나 더 큰 클러스터에는 결점이 있으며, 결국 심장박동기 및 Corosync로 인해 최대 32개의 노드가 필요합니다.

NetApp은 최대 10개 노드까지 BeeGFS HA 클러스터를 테스트했습니다. 따라서 개별 클러스터를 이 제한을 초과하여 확장하는 것은 권장되거나 지원되지 않습니다. 하지만 BeeGFS 파일 시스템은 여전히 10개 노드 이상으로 확장되어야 하며 NetApp은 BeeGFS on NetApp 솔루션에서 이 점을 고려했습니다.

각 파일 시스템에서 구성 요소의 하위 집합이 포함된 여러 HA 클러스터를 구축함으로써 기본 HA 클러스터링 메커니즘에 대한 권장 제한 또는 하드 제한과는 별개로 전체 BeeGFS 파일 시스템을 확장할 수 있습니다. 이 시나리오에서는 다음을 수행합니다.

- 추가 HA 클러스터를 나타내는 새 Ansible 인벤토리를 생성한 다음 다른 관리 서비스 구성 생략합니다. 대신 각 추가 클러스터 ha_cluster.yml의 "begfs_ha_mgmtd_floating_ip" 변수를 첫 번째 BeeGFS 관리 서비스의 IP에 지정합니다.
- 동일한 파일 시스템에 HA 클러스터를 추가할 때는 다음 사항을 확인하십시오.
  - BeeGFS 노드 ID는 고유합니다.

- group_vars의 각 서비스에 해당하는 파일 이름은 모든 클러스터에서 고유합니다.
- BeeGFS 클라이언트 및 서버 IP 주소는 모든 클러스터에서 고유합니다.
- 추가 클러스터를 구축 또는 업데이트하기 전에 BeeGFS 관리 서비스가 포함된 첫 번째 HA 클러스터가 실행되고 있습니다.

- 각 HA 클러스터에 대한 인벤토리를 각각 자체 디렉토리 트리에서 유지 관리합니다.

  하나의 디렉토리 트리에서 여러 클러스터의 인벤토리 파일을 혼합하려고 하면 BeeGFS HA 역할이 특정 클러스터에 적용된 구성을 집계하는 방식에 문제가 발생할 수 있습니다.

> 새 HA 클러스터를 생성하기 전에 각 HA 클러스터를 5개의 구성 요소로 확장할 필요가 없습니다. 대부분의 경우 클러스터당 더 적은 수의 구성 요소를 사용하는 것이 더 쉽게 관리할 수 있습니다. 한 가지 접근 방식은 각 단일 랙의 구성 요소를 HA 클러스터로 구성하는 것입니다.

권장되는 스토리지 풀 오버 프로비저닝 비율

2세대 구성 요소에 대해 스토리지 풀당 표준 볼륨 4개를 따르는 경우 다음 표를 참조하십시오.

이 표에는 각 BeeGFS 메타데이터 또는 스토리지 타겟에 대한 "eseries_storage_pool_configuration"의 볼륨 크기로 사용할 권장 비율이 나와 있습니다.

드라이브 크기	크기
1.92TB	18
3.84TB	21.5
7.68TB	22.5
15.3TB	24

> 위의 지침은 관리 서비스가 포함된 스토리지 풀에 적용되지 않으며, 관리 데이터에 대해 스토리지 풀의 1%를 할당하기 위해 위의 크기를 .25%까지 줄여야 합니다.

이러한 값이 어떻게 결정되었는지 확인하려면 을 참조하십시오 "TR-4800: 부록 A: SSD 내구성 및 오버 프로비저닝 이해".

고용량 구성 요소입니다

표준 BeeGFS 솔루션 구축 가이드에서는 고성능 워크로드 요구 사항에 대한 절차 및 권장 사항을 간략하게 설명합니다. 고용량 요구 사항을 충족하려는 고객은 여기에 설명된 구축 및 권장 사항의 변화를 관찰해야 합니다.

[]

컨트롤러

고용량 구성 요소의 경우 EF600 컨트롤러를 EF300 컨트롤러로 교체해야 하며, 각 컨트롤러는 SAS 확장을 위해 Cascade HIC를 설치합니다. 각 블록 노드는 스토리지 엔클로저에 BeeGFS 메타데이터 스토리지를 위한 최소한의 NVMe SSD를 포함하고 BeeGFS 스토리지 볼륨용 NL-SAS HDD로 채워진 확장 셸프에 연결됩니다.

File Node to Block 노드 구성은 동일하게 유지됩니다.

드라이브 배치

BeeGFS 메타데이터 스토리지를 위해 각 블록 노드에 최소 4개의 NVMe SSD가 필요합니다. 이러한 드라이브는 인클로저의 가장 바깥쪽 슬롯에 위치해야 합니다.

[]

확장 트레이

스토리지 어레이당 1-7, 60 드라이브 확장 트레이를 사용하여 대용량 구성 요소를 사이징할 수 있습니다.

각 확장 트레이 케이블 연결 지침은 "드라이브 쉘프의 EF300 케이블 연결을 참조하십시오".