BeeGFS auf NetApp mit E-Series Storage

BeeGFS on NetApp with E-Series Storage

NetApp
June 09, 2025

Inhalt

BeeGFS auf NetApp mit E-Series Storage

Los geht’s

Was ist in dieser Website enthalten

Auf dieser Website wird dokumentiert, wie BeeGFS auf NetApp mit sowohl NetApp Verified Architectures (NVAs) als auch individuellen Architekturen implementiert und gemanagt wird. NVA-Designs werden ausführlich getestet und bieten Kunden Referenzkonfigurationen und Anleitungen zur Größenbestimmung, um Implementierungsrisiken zu minimieren und die Markteinführungszeit zu beschleunigen. NetApp unterstützt außerdem individuelle BeeGFS-Architekturen, die auf NetApp Hardware ausgeführt werden. So können Kunden und Partner flexibel Filesysteme entwerfen, die einen breiten Anforderungsspektrum erfüllen. Beide Ansätze nutzen Ansible für die Implementierung und bieten einen Appliance-ähnlichen Ansatz, mit dem BeeGFS in jeder Größenordnung über ein flexibles Spektrum an Hardware gemanagt werden kann.

Begriffe und Konzepte

Die folgenden Begriffe und Konzepte gelten für die BeeGFS auf NetApp Lösung.

Im "BeeGFS-Cluster verwalten" Abschnitt finden Sie weitere Details zu Begriffen und Konzepten für die Interaktion mit BeeGFS-Clustern mit hoher Verfügbarkeit (HA).

Laufzeit	Beschreibung
KI	Künstliche Intelligenz.
Ansible-Steuerungsknoten	Eine physische oder virtuelle Maschine, die zum Ausführen der Ansible CLI verwendet wird.
Ansible-Bestandsaufnahme	Verzeichnisstruktur mit YAML-Dateien, die zur Beschreibung des gewünschten BeeGFS HA-Clusters verwendet werden.
BMC	Baseboard Management Controller Und wird manchmal als Service-Prozessor bezeichnet.
Block-Nodes	E-Series Storage-Systemen
Clients	Nodes im HPC-Cluster führen Applikationen aus, die das Filesystem verwenden müssen. Gelegentlich auch als Computing- oder GPU-Nodes bezeichnet
DL	Deep Learning.
Datei-Nodes	BeeGFS-Dateiserver.

Laufzeit	Beschreibung
HOCHVERFÜGBARKEIT	Hochverfügbarkeit,
HIC	Host-Schnittstellenkarte:
HPC	High Performance Computing:
Workloads im HPC-Stil	HPC Workloads werden in der Regel durch mehrere Computing-Nodes oder GPUs gekennzeichnet, die alle parallel auf denselben Datensatz zugreifen müssen, um ein verteiltes Computing- oder Trainingsjob zu ermöglichen. Diese Datensätze bestehen häufig aus großen Dateien, die auf mehrere physische Storage-Nodes verteilt werden sollten, um die herkömmlichen Hardwareengpässe zu beseitigen, die den gleichzeitigen Zugriff auf eine einzelne Datei verhindern würden.
ML	Maschinelles Lernen:
NLP	Natürliche Sprachverarbeitung.
NLU	Verständnis Natürlicher Sprachen.
NVA	Das NetApp Verified Architecture (NVA) Programm enthält Referenzkonfigurationen und Anleitungen zur Dimensionierung für spezifische Workloads und Anwendungsfälle. Diese Lösungen sind sorgfältig getestet und sollen Implementierungsrisiken minimieren und die Markteinführungszeit verkürzen.
Storage-Netzwerk/Client-Netzwerk	Netzwerk, das für Clients zur Kommunikation mit dem BeeGFS-Dateisystem verwendet wird. Dies ist häufig dasselbe Netzwerk, das für MPI (Parallel Message Passing Interface) und andere Anwendungskommunikation zwischen HPC-Clusterknoten verwendet wird.

Einsatz verifizierter Architekturen

Überblick und Anforderungen

Lösungsüberblick

Die BeeGFS auf NetApp Lösung kombiniert das parallele BeeGFS Filesystem mit NetApp EF600 Storage-Systemen und bietet so eine zuverlässige, skalierbare und kostengünstige Infrastruktur, die mit anspruchsvollen Workloads Schritt hält.

NVA-Programm

Die BeeGFS on NetApp Lösung ist Teil des NetApp Verified Architecture (NVA) Programms, das Kunden Referenzkonfigurationen und Orientierungshilfen zur Größenbestimmung für spezifische Workloads und Anwendungsfälle bietet. NVA-Lösungen werden ausführlich getestet und entwickelt, um Implementierungsrisiken zu minimieren und die Markteinführungszeit zu verkürzen.

Design-Übersicht

BeeGFS auf NetApp wurde als skalierbare Bausteinarchitektur konzipiert, die für eine Vielzahl anspruchsvoller Workloads konfigurierbar ist. Das Filesystem kann an diese Anforderungen angepasst werden – unabhängig davon, ob es um zahlreiche kleine Dateien, das Management umfangreicher Dateivorgänge oder um einen Hybrid-Workload geht. Hohe Verfügbarkeit ist in das Design mit der Verwendung einer zweistufigen Hardware-Struktur integriert, die ein unabhängiges Failover auf mehreren Hardware-Schichten ermöglicht und eine konsistente Performance auch bei teilweisen Systemabfällen gewährleistet. Das BeeGFS-Filesystem ermöglicht eine hochperformante und skalierbare Umgebung für verschiedene Linux-Distributionen. Es stellt Clients einen einzelnen, einfach zugänglichen Storage-Namespace zur Verfügung. Erfahren Sie mehr in der "Architekturübersicht".

Anwendungsfälle

Die folgenden Anwendungsfälle gelten für die BeeGFS auf NetApp Lösung:

- NVIDIA DGX SuperPOD-Systeme mit DGX's mit A100, H100, H200 und B200 GPUs
- Künstliche Intelligenz (KI), einschließlich Machine Learning (ML), Deep Learning (DL), großzügiger natürlicher Sprachverarbeitung (NLP) und NLU (Natural Language Understanding) Weitere Informationen finden Sie unter "BeeGFS for AI: Fakt versus Fiction".
- High-Performance Computing (HPC) einschließlich Applikationen, die mit MPI (Message Passing Interface) und anderen Distributed Computing-Techniken beschleunigt werden. Weitere Informationen finden Sie unter "Warum BeeGFS das HPC übertrifft".
- Applikations-Workloads zeichnen sich durch folgende Merkmale aus:
  - Lesen oder Schreiben auf Dateien mit einer Größe von mehr als 1 GB
  - Lesen oder Schreiben in dieselbe Datei durch mehrere Clients (10s, 100s und 1000s)
- Datensätze mit mehreren Terabyte oder mehreren Petabyte.
- Umgebungen, für die ein einziger Storage Namespace benötigt wird, der sich für eine Mischung aus großen und kleinen Dateien optimieren lässt

Vorteile

Die wichtigsten Vorteile von BeeGFS auf NetApp:

- Verfügbarkeit verifizierter Hardware-Designs mit vollständiger Integration von Hardware- und Softwarekomponenten, die zuverlässige Performance und Zuverlässigkeit gewährleisten.
- Implementierung und Management mit Ansible für Einfachheit und Konsistenz nach Maß
- Überwachung und Beobachtbarkeit mithilfe des E-Series Performance Analyzer und BeeGFS Plug-ins. Weitere Informationen finden Sie unter "Framework zur Überwachung von NetApp E-Series Lösungen".
- Hochverfügbarkeit dank einer Shared-Disk-Architektur für Datenaufbewahrungszeit und -Verfügbarkeit
- Unterstützung für modernes Workload-Management und moderne Orchestrierung mithilfe von Containern und Kubernetes Weitere Informationen finden Sie unter "Kubernetes Meet BeeGFS: Eine Geschichte zukunftssichere Investition".

Generationen im Design

Die BeeGFS on NetApp-Lösung befindet sich derzeit in der zweiten Generation.

Sowohl die erste als auch die zweite Generation verfügen über eine Basisarchitektur mit einem BeeGFS Filesystem und einem NVMe EF600 Storage-System. Die zweite Generation baut jedoch auf der ersten auf und beinhaltet folgende zusätzliche Vorteile:

- Verdoppeln Sie die Performance und Kapazität bei gleichzeitiger Ergänzung von nur 2 HE Rack-Stellfläche
- Hochverfügbarkeit (HA) auf Grundlage eines gemeinsam genutzten zwei-Tier-Hardwaredesigns
- Architektur, die für NVIDIA DGX SuperPOD A100, H100, H200 und B200 Systeme entwickelt wurde und zuvor auf einem dedizierten Abnahme-Cluster bei NVIDIA validiert wurde Lesen Sie mehr über NVIDIA DGX SuperPOD mit NetApp in der "Designleitfaden".

Design der zweiten Generation

Die zweite Generation von BeeGFS auf NetApp ist für die Performance-Anforderungen anspruchsvoller Workloads optimiert, darunter High-Performance-Computing (HPC), Machine Learning (ML), Deep Learning (DL) und andere Techniken für künstliche Intelligenz (KI). Durch die Integration einer Shared-Disk-Hochverfügbarkeits-Architektur (HA) sorgt dieses Design für die Langlebigkeit und Verfügbarkeit von Daten und eignet sich daher ideal für Unternehmen und andere Unternehmen, die Ausfallzeiten oder Datenverluste nicht leisten können. Das Design der zweiten Generation umfasst Komponenten wie PCIe gen5-Server und Unterstützung für NVIDIA® Quantum™ QM9700 400 GB/s InfiniBand-Switches. Diese Lösung wurde nicht nur von NetApp verifiziert, sondern hat auch die externe Qualifizierung als Storage-Option für das DGX™ A100 SuperPOD NVIDIA bestanden – mit erweiterter Zertifizierung für DGX SuperPOD H100-, H200- und B200-Systeme.

Design der ersten Generationen

Die erste Generation von BeeGFS auf NetApp wurde für Workloads für Machine Learning (ML) und künstliche Intelligenz (KI) mit NetApp EF600 NVMe-Storage-Systemen, dem parallelen Filesystem BeeGFS, NVIDIA DGX™ A100-Systemen und NVIDIA® Mellanox® Quantum™ QM8700 200 GB/s IB-Switches entwickelt. Dieses Design bietet zudem 200 GB/s InfiniBand (IB) für die Storage- und Computing-Cluster Interconnect Fabric und stellt so eine vollständig IB-basierte Architektur für hochperformante Workloads bereit.

Weitere Informationen zur ersten Generation finden Sie unter "NetApp EF-Series AI mit NVIDIA DGX A100 Systems und BeeGFS".

Überblick über die Architektur

Die BeeGFS on NetApp Lösung beinhaltet Design-Aspekte, die bei der Architekturentwicklung berücksichtigt werden, um die spezifischen Geräte, Kabel und Konfigurationen zu ermitteln, die für validierte Workloads erforderlich sind.

Modulare Architektur

Das BeeGFS-Dateisystem kann je nach Storage-Anforderungen unterschiedlich implementiert und skaliert werden. In Anwendungsfällen, die in erster Linie mehrere kleine Dateien enthalten, profitieren beispielsweise von der zusätzlichen Performance und Kapazität der Metadaten, während in Anwendungsfällen mit weniger großen Dateien mehr Storage-Kapazität und Performance für die tatsächlichen Dateiinhalte erforderlich wären. Diese verschiedenen Überlegungen wirken sich auf die verschiedenen Dimensionen der Implementierung paralleler Dateisysteme aus, was die Entwicklung und Implementierung des Filesystems weiter vereinfacht.

Zur Bewältigung dieser Herausforderungen hat NetApp eine standardmäßige Bausteinarchitektur entwickelt, mit der sich jede dieser Dimensionen skalieren lässt. BeeGFS-Bausteine werden in der Regel in einem von drei Konfigurationsprofilen bereitgestellt:

- Ein einzelner Baustein, einschließlich BeeGFS-Management, Metadaten und Storage-Services
- Ein BeeGFS Metadaten plus Storage-Baustein
- Ein BeeGFS-Lagergebäude

Die einzige Hardware-Änderung zwischen diesen drei Optionen ist die Verwendung kleinerer Laufwerke für BeeGFS-Metadaten. Andernfalls werden alle Konfigurationsänderungen durch die Software übernommen. Und mit Ansible als Implementierungs-Engine gestaltet sich die Einrichtung des gewünschten Profils für einen bestimmten Baustein die Konfigurationsaufgaben unkompliziert.

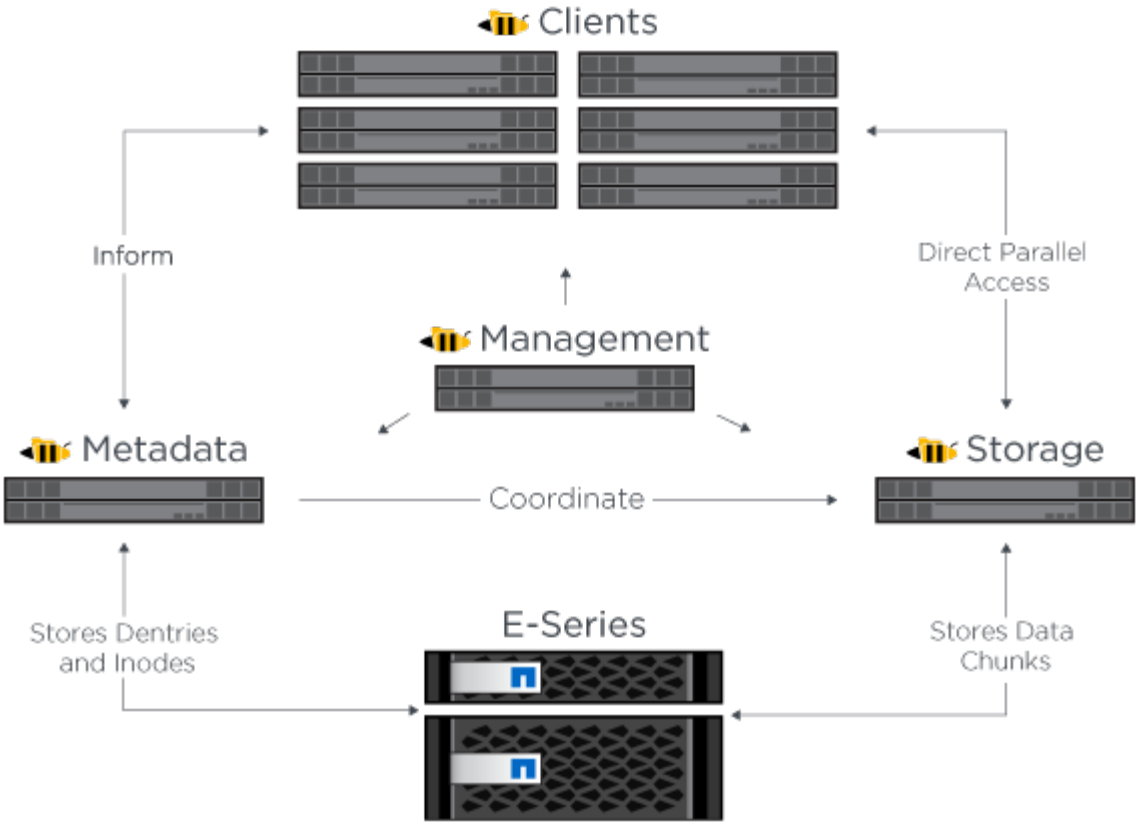
Weitere Informationen finden Sie unter Verifiziertes Hardwaredesign.

File-System-Services

Das BeeGFS-Dateisystem umfasst die folgenden Hauptdienste:

- **Management Service.** registriert und überwacht alle anderen Dienste.
- **Speicherdienst.** speichert den verteilten Inhalt der Benutzerdatei, bekannt als Datenblock-Dateien.
- **Metadatendienst.** verfolgt das Dateisystem-Layout, Verzeichnis, Dateiattribute und so weiter.
- **Client Service.** installiert das Dateisystem, um auf die gespeicherten Daten zuzugreifen.

Die folgende Abbildung zeigt die Komponenten und Beziehungen der BeeGFS-Lösung für NetApp E-Series Systeme.



Als paralleles Dateisystem verteilt BeeGFS seine Dateien auf mehrere Server-Nodes, um die Lese-/Schreib-Performance und Skalierbarkeit zu maximieren. Die Server-Knoten arbeiten zusammen, um ein einziges Dateisystem bereitzustellen, das gleichzeitig von anderen Server-Knoten, allgemein bekannt als *Clients*, gemountet werden kann. Diese Clients können das verteilte Dateisystem auf ähnliche Weise wie ein lokales Dateisystem wie NTFS, XFS oder ext4 sehen und nutzen.

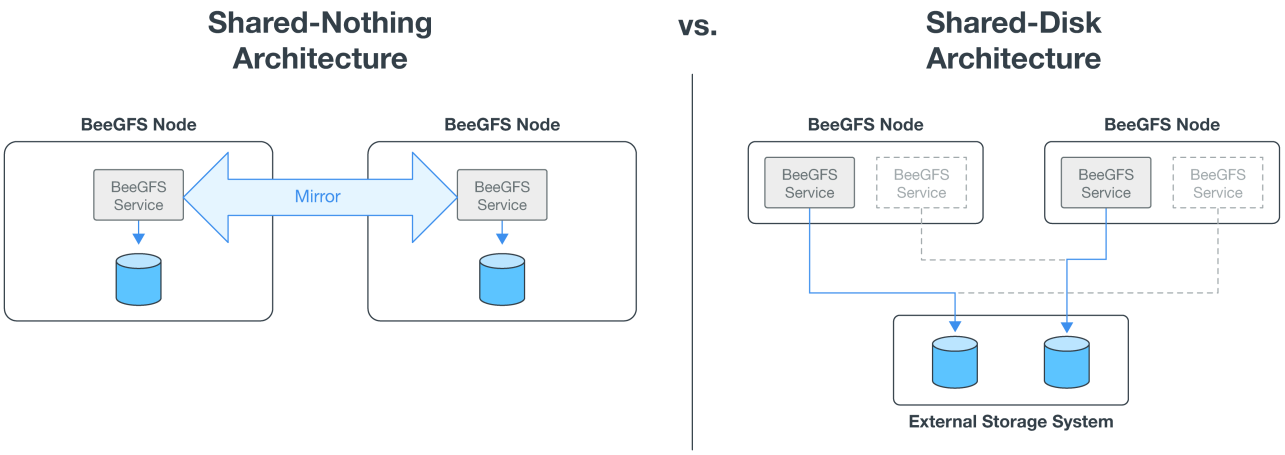
Die vier wichtigsten Services werden in einer Vielzahl von unterstützten Linux Distributionen ausgeführt und kommunizieren über jedes TCP/IP- oder RDMA-fähige Netzwerk, einschließlich InfiniBand (IB), Omni-Path (OPA) und RDMA over Converged Ethernet (RoCE). Die BeeGFS Server Services (Management, Speicherung und Metadaten) sind Benutzerspace-Dämonen, während der Client ein natives Kernel-Modul (patchless) ist. Alle Komponenten können ohne Neustart installiert oder aktualisiert werden. Sie können beliebige Kombinationen von Services auf demselben Node ausführen.

HA-Architektur

BeeGFS auf NetApp erweitert die Funktionalität der BeeGFS Enterprise Edition durch Entwicklung einer vollständig integrierten Lösung mit NetApp Hardware, die eine HA-Architektur (Shared Disk High Availability, Shared-Hochverfügbarkeit) ermöglicht.

> Die BeeGFS Community Edition kann zwar kostenlos genutzt werden, jedoch muss bei der Enterprise Edition ein Professional Support-Abonnementvertrag von einem Partner wie NetApp abgeschlossen werden. Die Enterprise-Version ermöglicht die Nutzung mehrerer zusätzlicher Funktionen wie Ausfallsicherheit, Kontingentzuverfolgung und Storage-Pools.

In der folgenden Abbildung werden die HA-Architekturen ohne Shared-Festplatten verglichen.



Weitere Informationen finden Sie unter "Ankündigung der Hochverfügbarkeit für BeeGFS mit Unterstützung von NetApp".

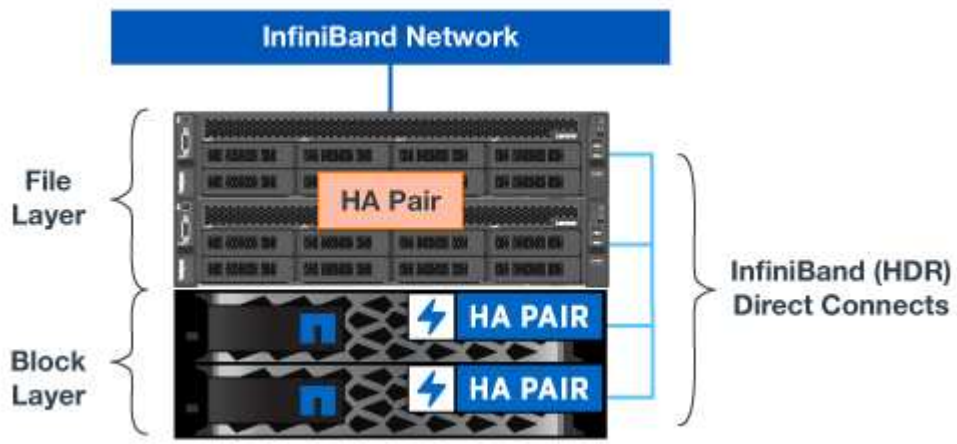
Verifizierte Nodes

Die BeeGFS auf NetApp-Lösung hat die unten aufgeführten Knoten verifiziert.

Knoten	Trennt	Details
Block-Storage	NetApp EF600 Storage-System	Dieses rein NVMe-basierte 2-HE-Storage-Array mit hoher Performance ist für anspruchsvolle Workloads konzipiert.
Datei	Lenovo ThinkSystem SR665 V3-Server	2-Socket-Server mit PCIe 5.0, zwei AMD EPYC 9124 Prozessoren. Weitere Informationen zum Lenovo SR665 V3 finden Sie unter "Lenovo Website".
	Lenovo ThinkSystem SR665 Server	2-Socket-Server mit PCIe 4.0, zwei AMD EPYC 7003 Prozessoren. Weitere Informationen zum Lenovo SR665 finden Sie unter "Lenovo Website".

Verifiziertes Hardwaredesign

Die Bausteine der Lösung (in der folgenden Abbildung dargestellt) verwenden die verifizierten File-Node-Server für die BeeGFS-Dateiebene und zwei EF600-Storage-Systeme als Block-Ebene.

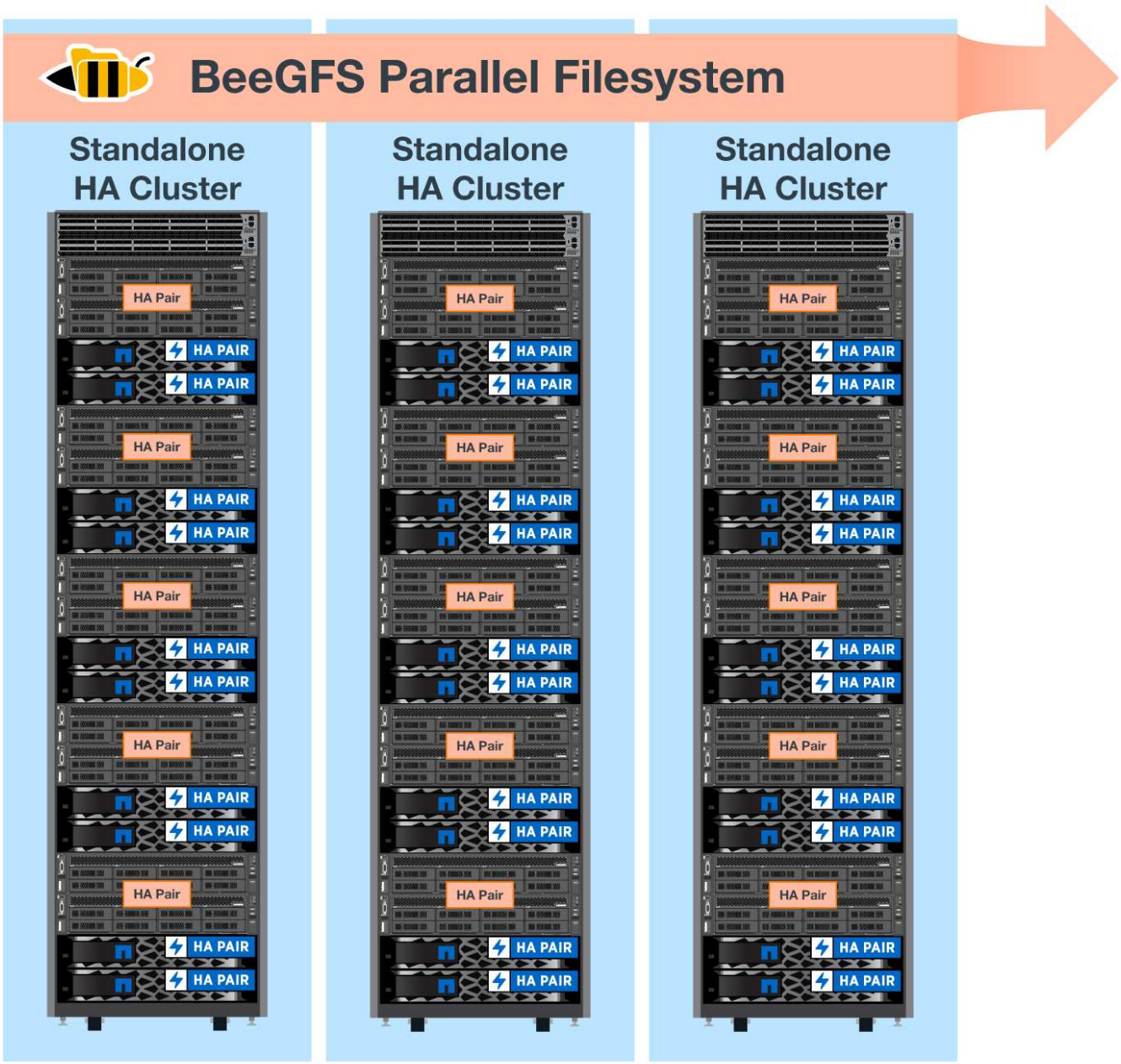


Die BeeGFS on NetApp Lösung läuft über alle Bausteine während der Implementierung hinweg. Auf dem ersten implementierten Baustein müssen BeeGFS-Management-, Metadaten- und Storage-Services (als

Basisbaustein bezeichnet) ausgeführt werden. Alle nachfolgenden Bausteine können über Software konfiguriert werden, um Metadaten und Storage-Services zu erweitern oder ausschließlich Storage-Services bereitzustellen. Mit diesem modularen Ansatz kann das Filesystem an die Anforderungen eines Workloads skaliert werden, während gleichzeitig dieselben zugrunde liegenden Hardware-Plattformen und dasselbe Bausteindesign verwendet werden.

Bis zu fünf Bausteine können als Standalone Linux HA-Cluster implementiert werden. Dies optimiert die Ressourcenverwaltung mit Pacemaker und sorgt für eine effiziente Synchronisierung mit Corosync. Mindestens ein dieser Standalone BeeGFS HA-Cluster wird kombiniert, um ein BeeGFS-Filesystem zu erstellen, das für Clients als einzelner Storage-Namespace zur Verfügung steht. Auf der Hardware-Seite kann ein einzelnes 42-HE-Rack bis zu fünf Bausteine zusammen mit zwei 1-HE-InfiniBand-Switches für das Storage-/Datennetzwerk aufnehmen. Eine visuelle Darstellung finden Sie in der folgenden Grafik.

Zum Herstellen von Quorum im Failover Cluster sind mindestens zwei Bausteine erforderlich. Ein Cluster mit zwei Nodes hat Einschränkungen, die ein erfolgreiches Failover verhindern können. Wenn Sie ein Cluster mit zwei Nodes konfigurieren, wird ein drittes Gerät als Tiebreaker integriert, dieses Design wird jedoch nicht in dieser Dokumentation beschrieben.



Ansible

BeeGFS auf NetApp wird mittels Ansible-Automatisierung bereitgestellt und implementiert. Das Hosting wird auf GitHub und Ansible Galaxy (die BeeGFS-Sammlung ist über verfügbar "Ansible-Galaxie" Und "NetApp E-Series GitHub"). Obwohl Ansible vor allem mit der Hardware getestet wird, die zum Zusammenbauen der BeeGFS-Bausteine verwendet wird, können Sie es so konfigurieren, dass es auf nahezu jedem x86-basierten Server unter Verwendung einer unterstützten Linux-Distribution ausgeführt wird.

Weitere Informationen finden Sie unter "Implementieren von BeeGFS mit E-Series Storage".

Technische Anforderungen

Stellen Sie zur Implementierung der Lösung BeeGFS auf NetApp sicher, dass Ihre Umgebung die in diesem Dokument beschriebenen Technologieanforderungen erfüllt.

Hardwareanforderungen

Stellen Sie zunächst sicher, dass Ihre Hardware die folgenden Spezifikationen für ein einziges Bausteindesign der zweiten Generation der BeeGFS auf NetApp-Lösung erfüllt. Die genauen Komponenten für eine bestimmte Implementierung können je nach den Anforderungen des Kunden variieren.

Menge	Hardwarekomponenten	Anforderungen
2	BeeGFS-Datei-Nodes	Jeder Datei-Node sollte die Spezifikationen der empfohlenen Datei-Nodes erfüllen oder übertreffen, um die erwartete Performance zu erreichen. **Empfohlene Dateiknoten-Optionen:** • * Lenovo ThinkSystem SR665 V3* ◦ **Prozessoren:** 2x AMD EPYC 9124 16C 3.0 GHz (konfiguriert als zwei NUMA Zonen). ◦ **Speicher:** 256 GB (16 x 16 GB TruDDR5 4800 MHz RDIMM-A) ◦ **PCIe-Erweiterung:** vier PCIe Gen5 x16-Steckplätze (zwei pro NUMA-Zone) ◦ **Verschiedenes:** ▪ Zwei Laufwerke in RAID 1 für OS (1 TB 7.200 SATA oder höher) ▪ 1-GbE-Port für in-Band-OS-Management ▪ 1GbE BMC mit Redfish API für Out-of-Band-Server-Management ▪ Zwei Hot-Swap-Netzteile und Lüfter mit hoher Leistung
2	E-Series-Block-Nodes (EF600 Array)	**Speicher:** 256 GB (128 GB pro Controller). **Adapter:** 2-Port 200 GB/HDR (NVMe/IB). **Laufwerke:** entsprechend den gewünschten Metadaten und Speicherkapazität konfiguriert.
8	InfiniBand-Host-Karten-Adapter (für Datei-Nodes)	Hostkartenadapter können je nach Servermodell des Dateiknotens variieren. Zu den Empfehlungen für verifizierte Datei-Nodes gehören: • * Lenovo ThinkSystem SR665 V3 Server:* ◦ MCX755106AS-HEAT ConnectX-7, NDR200, QSFP112, 2 Ports, PCIe Gen5 x16, InfiniBand-Adapter
1	Storage-Netzwerk-Switch	Der Storage-Netzwerk-Switch muss 200 GB/s InfiniBand-Geschwindigkeiten unterstützen. Empfohlene Switch-Modelle: • **NVIDIA QM9700 Quantum 2 NDR InfiniBand Switch** • **NVIDIA MQM8700 Quantum HDR InfiniBand Switch**

Verkabelungsanforderungen

Direkte Verbindungen von Block-Knoten zu File-Knoten.

Menge	Teilenummer	Länge
8	MCP1650-H001E30 (passives NVIDIA-Kupferkabel, QSFP56, 200 GB/s)	1 m

Verbindungen von Dateiknoten zum Speichernetzwerk-Switch. Wählen Sie je nach InfiniBand-Speicherschalter die entsprechende Kabeloption aus der folgenden Tabelle aus. + die empfohlene Kabellänge beträgt 2 m, dies kann jedoch je nach Umgebung des Kunden variieren.

Switch-Modell	Kabeltyp	Menge	Teilenummer
NVIDIA QM9700	Aktive Glasfaser (einschließlich Transceiver)	2	MMA4Z00-NS (Multimode, IB/ETH, 800 GB/s 2x400 GB/s Twin-Port OSFP)
		4	MFP7E20-Nxxx (Multimode, 4-Kanal-zu-zwei 2-Kanal-Splitter-Glasfaserkabel)
		8	MMA1Z00-NS400 (Multimode, IB/ETH, 400 GB/s Single-Port QSFP-112)
	Passives Kupfer	2	MCP7Y40-N002 (passives NVIDIA-Kupferverteilerkabel, InfiniBand 800 GB/s bis 4 x 200 GB/s, OSFP auf 4 x QSFP112)
NVIDIA MQM8700	Aktive Glasfaser	8	MFS1S00-H003E (aktives NVIDIA-Glasfaserkabel, InfiniBand 200 GB/s, QSFP56)
	Passives Kupfer	8	MCP1650-H002E26 (passives NVIDIA-Kupferkabel, InfiniBand 200 GB/s, QSFP56)

Software- und Firmware-Anforderungen zu erfüllen

Um eine vorhersehbare Performance und Zuverlässigkeit zu gewährleisten, werden Versionen der BeeGFS auf NetApp Lösung mit bestimmten Versionen der Software- und Firmware-Komponenten getestet. Diese Versionen sind für die Implementierung der Lösung erforderlich.

Anforderungen an Datei-Nodes

Software	Version
Red hat Enterprise Linux (RHEL)	RHEL 9.4 Server physisch mit hoher Verfügbarkeit (2 Sockel). **Hinweis:** Für Dateiknoten sind ein gültiges Red Hat Enterprise Linux Server-Abonnement und das Red Hat Enterprise Linux High Availability Add-On erforderlich.
Linux-Kernel	5.14.0-427.42.1.el9_4.x86_64
HCA-Firmware	**ConnectX-7 HCA-Firmware** FW: 28.43.1014 + PXE: 3.7.0500 + UEFI: 14.36.0016 **ConnectX-6 HCA-Firmware** FW: 20.43.2566 + PXE: 3.7.0500 + UEFI: 14.37.0013

Anforderungen der EF600 Block-Nodes

Software	Version
SANtricity OS	11.90R1
NVSRAM	N6000-890834-D02.dlp

Software	Version
Festplatten-Firmware	Neueste verfügbar für die verwendeten Antriebsmodelle. Siehe "E-Series Festplatten-Firmware-Website".

Anforderungen an die Softwareimplementierung

In der folgenden Tabelle sind die automatisch bereitgestellten Softwareanforderungen im Rahmen der Ansible-basierten BeeGFS-Implementierung aufgeführt.

Software	Version
BeeGFS	7.4.6
Corosync	3.1.8-1
Schrittmacher	2.1.7-5,2
PCS	0.11.7-2
Zaunmittel (Rotbarsch/apc)	4.10.0-62
InfiniBand-/RDMA-Treiber	MLNX_OFED_LINUX-23.10-3.2.2.1-LTS

Ansible-Control-Node-Anforderungen

Die BeeGFS auf NetApp Lösung wird über einen Ansible-Kontroll-Node implementiert und gemanagt. Weitere Informationen finden Sie im "Ansible-Dokumentation".

Die in den folgenden Tabellen aufgeführten Software-Anforderungen beziehen sich speziell auf die unten aufgeführte Version der NetApp BeeGFS Ansible Sammlung.

Software	Version
Ansible	10.x
Ansible-Core	>= 2.13.0
Python	3,10
Zusätzliche Python-Pakete	Kryptographie-43.0.0, netaddr-1.3.0, ipaddr-2.2.0
NetApp E-Series BeeGFS Ansible Sammlung	3.2.0

Überprüfen des Lösungsdesigns

Designübersicht

Spezifische Geräte, Kabel und Konfigurationen sind erforderlich, um die BeeGFS auf NetApp Lösung zu unterstützen, die das parallele Filesystem BeeGFS mit den NetApp EF600 Storage-Systemen kombiniert.

Weitere Informationen:

- "Hardwarekonfiguration"
- "Softwarekonfiguration"
- "Design-Überprüfung"
- "Richtlinien für die Dimensionierung"
- "Performance-Optimierung"

Derivative Architekturen mit Variationen in Design und Performance:

- "Baustein Mit Hoher Kapazität"

Hardwarekonfiguration

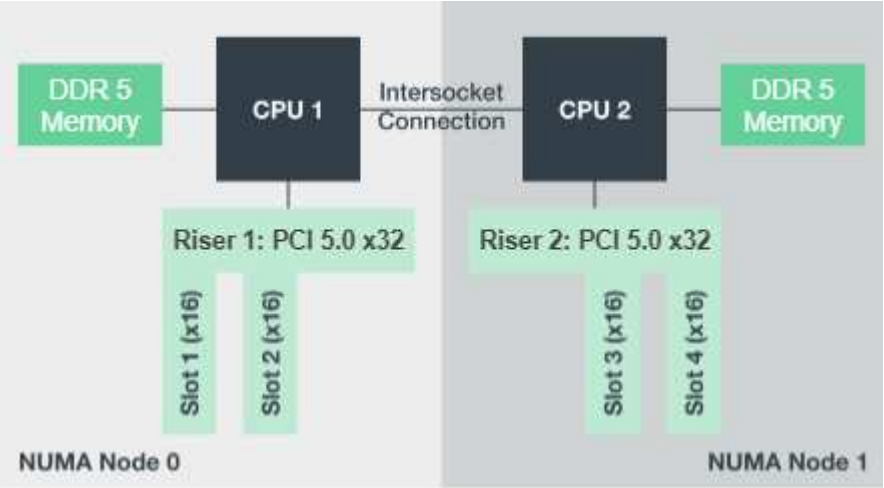
Die Hardware-Konfiguration für BeeGFS auf NetApp umfasst Datei-Nodes und Netzwerkverkabelung.

Konfiguration der Datei-Nodes

File-Nodes haben zwei CPU-Sockets als separate NUMA-Zonen konfiguriert, die lokalen Zugriff auf eine gleiche Anzahl von PCIe-Steckplätzen und Arbeitsspeicher enthalten.

InfiniBand-Adapter müssen in die entsprechenden PCI-Risers oder Steckplätze gefüllt sein, damit die Workload über die verfügbaren PCIe-Lanes und Speicherkanäle ausgeglichen ist. Sie balancieren den Workload aus, indem einzelne BeeGFS-Services vollständig auf einen bestimmten NUMA-Node isoliert werden. Das Ziel besteht darin, bei jedem Datei-Node eine ähnliche Performance zu erreichen, als ob es sich um zwei unabhängige Single-Socket-Server handelte.

Die folgende Abbildung zeigt die NUMA-Konfiguration des Dateiknotens.



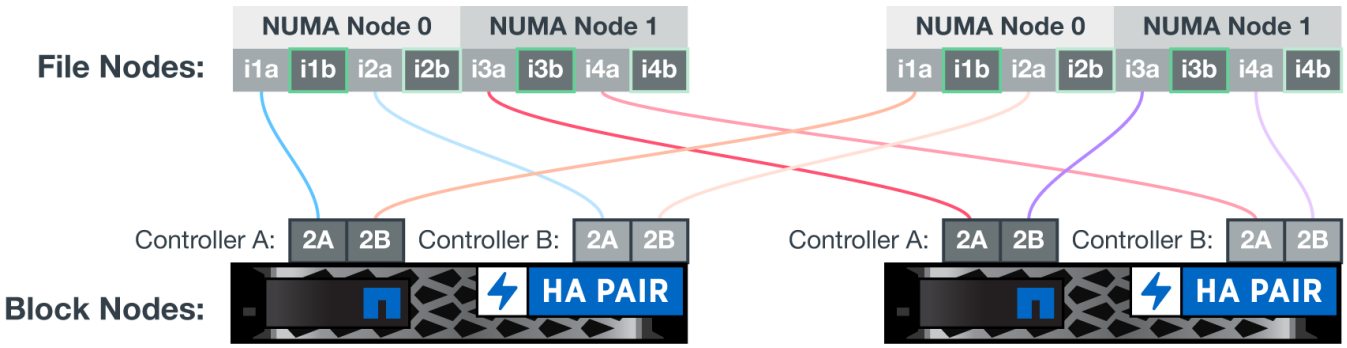
Die BeeGFS-Prozesse sind an eine bestimmte NUMA-Zone gebunden, um sicherzustellen, dass sich die verwendeten Schnittstellen in der gleichen Zone befinden. Diese Konfiguration vermeidet den Remote-Zugriff über die Verbindung zwischen den Sockets. Die Verbindung zwischen den Sockeln wird manchmal als QPI oder GMI2 Link bezeichnet; selbst in modernen Prozessorarchitekturen können sie einen Engpass darstellen, wenn High-Speed-Netzwerke wie HDR InfiniBand eingesetzt werden.

Konfiguration der Netzwerkverkabelung

Jeder Datei-Node ist innerhalb eines Bausteins mit zwei Block-Nodes verbunden. Dabei werden insgesamt vier redundante InfiniBand-Verbindungen verwendet. Zusätzlich verfügt jeder Datei-Node über vier redundante Verbindungen zum InfiniBand-Storage-Netzwerk.

Beachten Sie in der folgenden Abbildung Folgendes:

- Alle grün dargestellten Datei-Node-Ports werden zur Verbindung mit der Storage-Fabric verwendet. Alle anderen Datei-Node-Ports sind die direkten Verbindungen zu den Block-Nodes.
- Zwei InfiniBand-Ports in einer bestimmten NUMA-Zone werden mit den A- und B-Controllern desselben Block-Nodes verbunden.
- Ports im NUMA-Knoten 0 stellen immer eine Verbindung zum ersten Block-Knoten her.
- Die Ports im NUMA-Knoten 1 verbinden sich mit dem zweiten Block-Knoten.



Wenn Sie Splitterkabel verwenden, um den Speicher-Switch mit den Dateiknoten zu verbinden, sollte ein Kabel abzweigen und mit den hellgrünen Ports verbunden werden. Ein anderes Kabel sollte abzweigen und an die dunkelgrünen Ports anschließen. Außerdem sollten bei Speichernetzwerken mit redundanten Switches die hellgrünen Ports mit einem Switch verbunden werden, während dunkelgrüne Ports mit einem anderen Switch verbunden sein sollten.

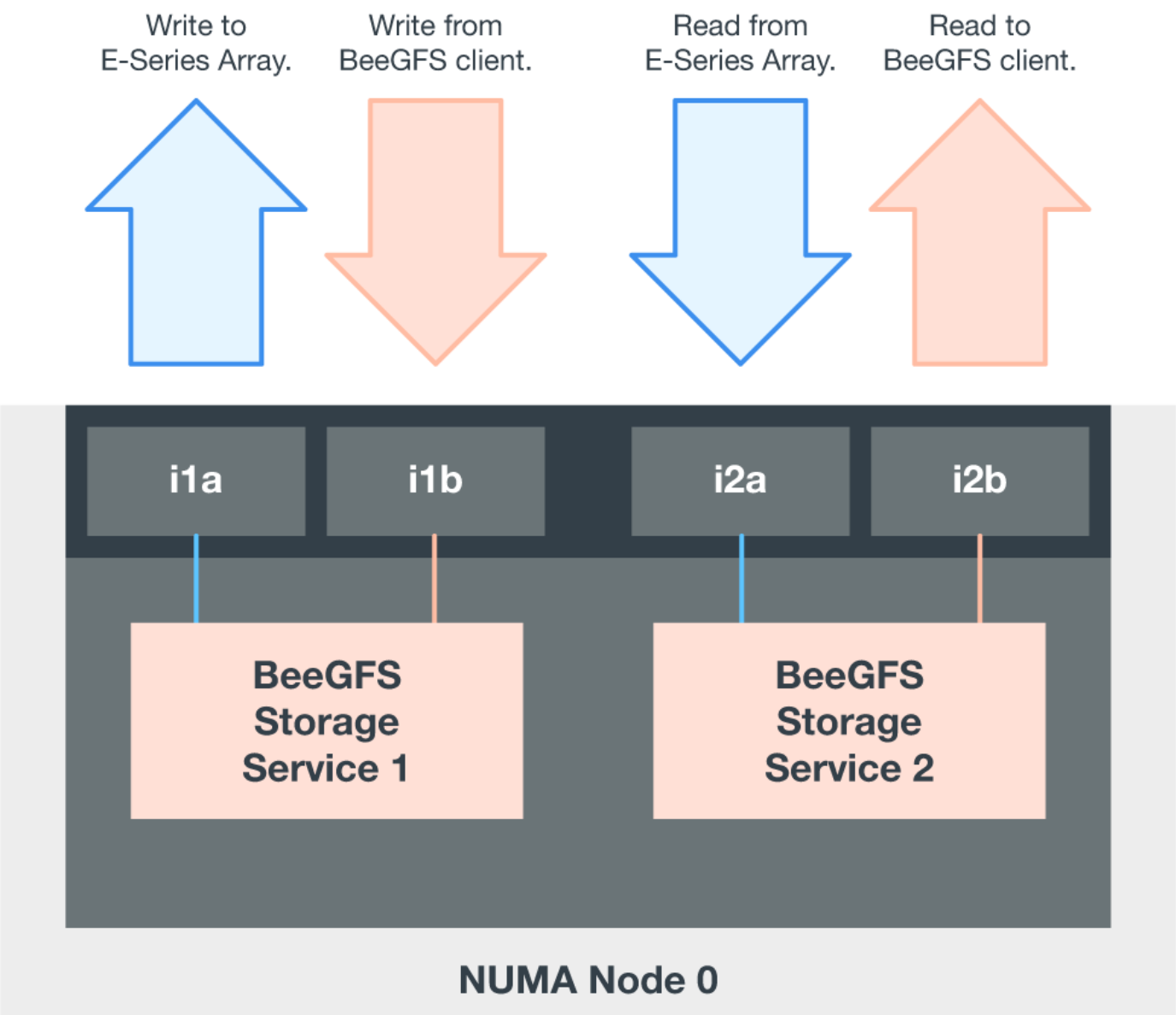
Die in der Abbildung dargestellte Verkabelungskonfiguration ermöglicht jedem BeeGFS-Dienst Folgendes:

- Laufen Sie in derselben NUMA-Zone, unabhängig davon, auf welchem Dateiknoten der BeeGFS-Service ausgeführt wird.
- Sekundäre optimale Pfade zum Front-End-Storage-Netzwerk und zu den Back-End-Block-Nodes sind vorhanden, unabhängig davon, wo ein Ausfall auftritt.
- Minimale Auswirkungen auf die Performance, wenn ein Datei-Node oder Controller in einem Block-Node gewartet werden muss

Verkabelung zur Nutzung der Bandbreite

Um die vollständige bidirektionale PCIe-Bandbreite zu nutzen, stellen Sie sicher, dass ein Port an jedem InfiniBand-Adapter mit der Storage-Fabric verbunden ist, und der andere Port mit einem Block-Node verbunden wird.

Die folgende Abbildung zeigt das Verkabelungsdesign, mit dem die vollständige bidirektionale PCIe-Bandbreite genutzt werden kann.



Verwenden Sie für jeden BeeGFS-Service denselben Adapter, um den bevorzugten Port für Client-Datenverkehr mit dem Pfad zu dem Block-Nodes-Controller zu verbinden, der der primäre Eigentümer dieser Service-Volumes ist. Weitere Informationen finden Sie unter "Softwarekonfiguration".

Softwarekonfiguration

Die Software-Konfiguration für BeeGFS auf NetApp umfasst BeeGFS-Netzwerkkomponenten, EF600 Block-Nodes, BeeGFS-Datei-Nodes, Ressourcengruppen und BeeGFS-Services.

BeeGFS-Netzwerkkonfiguration

Die BeeGFS-Netzwerkkonfiguration besteht aus den folgenden Komponenten.

- **Schwimmende IPs** schwimmende IPs sind eine Art virtueller IP-Adresse, die dynamisch an jeden Server im selben Netzwerk weitergeleitet werden kann. Mehrere Server können dieselbe Floating-IP-Adresse besitzen, sie kann jedoch nur auf einem Server zu einem bestimmten Zeitpunkt aktiv sein.

Jeder BeeGFS-Serverdienst hat eine eigene IP-Adresse, die sich je nach Ausführungsort des BeeGFS-Serverdienstes zwischen Datei-Nodes verschieben kann. Diese fließende IP-Konfiguration ermöglicht jedem Service ein unabhängiges Failover auf den anderen File-Node. Der Client muss einfach die IP-Adresse für einen bestimmten BeeGFS-Service kennen. Es muss nicht wissen, welcher Datei-Node derzeit diesen Service ausführt.

- **BeeGFS-Server Multi-Homing-Konfiguration** um die Dichte der Lösung zu erhöhen, verfügt jeder Dateiknoten über mehrere Speicherschnittstellen mit IPs, die im gleichen IP-Subnetz konfiguriert sind.

  Es ist eine zusätzliche Konfiguration erforderlich, um sicherzustellen, dass diese Konfiguration wie erwartet mit dem Linux-Netzwerk-Stack funktioniert, da standardmäßig Anfragen an eine Schnittstelle auf einer anderen Schnittstelle beantwortet werden können, wenn ihre IPs im selben Subnetz sind. Neben anderen Nachteilen ist es bei diesem Standardverhalten unmöglich, RDMA-Verbindungen ordnungsgemäß einzurichten oder zu pflegen.

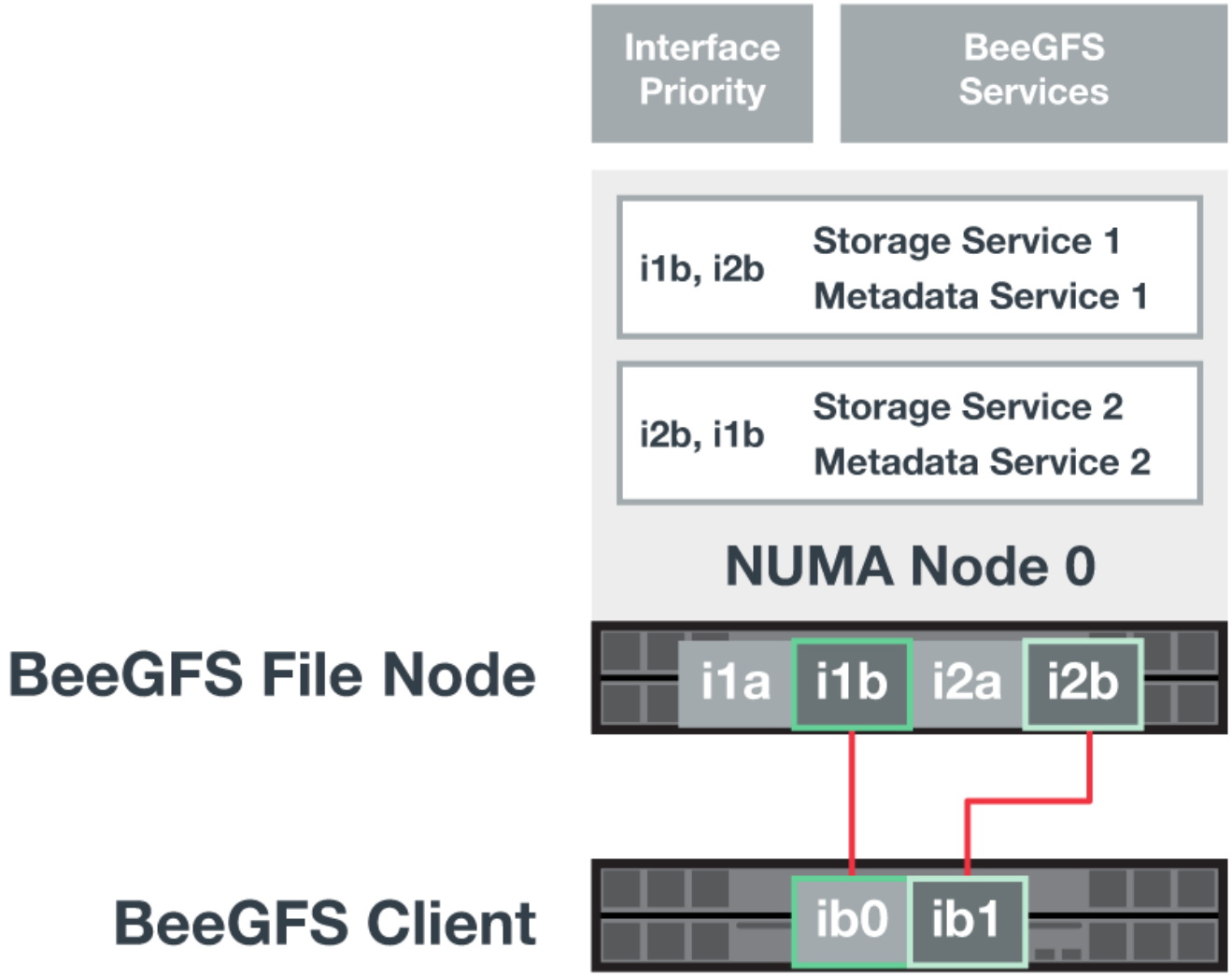
  Die Ansible-basierte Implementierung verarbeitet das Anziehen des Reverse Path (RP) und das ARP-Verhalten (Address Resolution Protocol) und stellt sicher, dass Floating IPs gestartet und gestoppt werden. Die entsprechenden IP-Routen und -Regeln werden dynamisch erstellt, damit die Multihomed-Netzwerkkonfiguration ordnungsgemäß funktioniert.

- **BeeGFS-Client-Multi-Rail-Konfiguration** *Multi-Rail* bezieht sich auf die Fähigkeit einer Anwendung, mehrere unabhängige Netzwerkverbindungen, oder "Schienen", zu verwenden, um die Leistung zu erhöhen.

  BeeGFS implementiert Multi-Rail-Unterstützung, um die Verwendung mehrerer IB-Schnittstellen in einem einzigen IPoIB-Subnetz zu ermöglichen. Diese Funktion ermöglicht Funktionen wie den dynamischen Lastausgleich über RDMA NICs und optimiert damit die Auslastung von Netzwerkressourcen. Die Integration in NVIDIA GPUDirect Storage (GDS) sorgt für eine höhere Systembandbreite sowie geringere Latenz und Auslastung der Client-CPU.

  Diese Dokumentation enthält Anweisungen für einzelne IPoIB-Subnetzkonfigurationen. Duale IPoIB-Subnetzkonfigurationen werden unterstützt, bieten jedoch nicht die gleichen Vorteile wie Konfigurationen mit einem einzigen Subnetz.

  Die folgende Abbildung zeigt die Verteilung des Datenverkehrs auf mehrere BeeGFS-Client-Schnittstellen.



Da jede Datei in BeeGFS normalerweise über mehrere Storage-Services verteilt wird, kann der Client dank der Multi-Rail-Konfiguration einen höheren Durchsatz erzielen als mit einem einzelnen InfiniBand-Port möglich. Die folgende Codeprobe zeigt beispielsweise eine allgemeine File-Striping-Konfiguration, die es dem Client ermöglicht, den Datenverkehr über beide Schnittstellen auszugleichen:

+

```
root@beegfs01:/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]
```

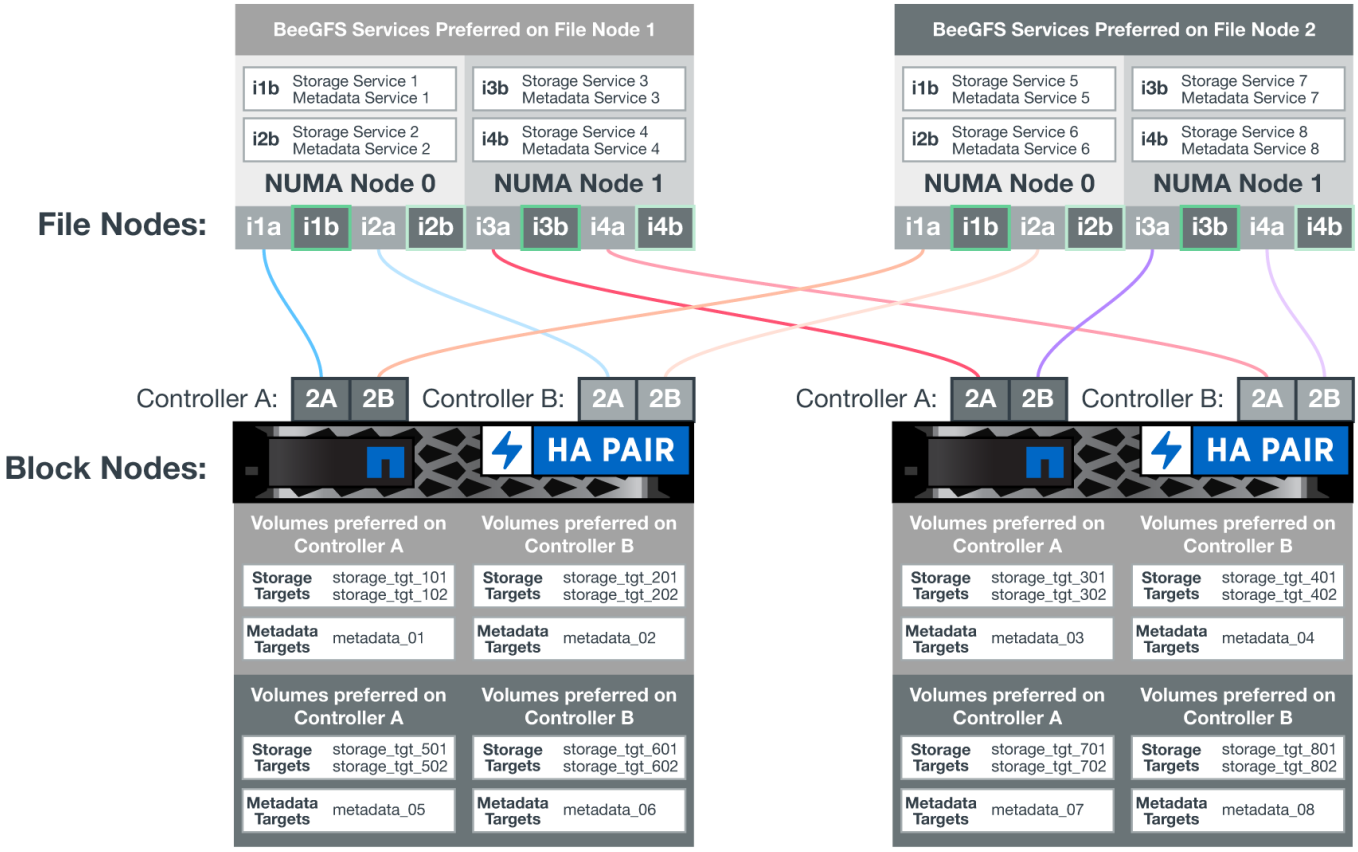
Konfiguration der EF600 Block-Node

Block-Nodes bestehen aus zwei aktiv/aktiv-RAID-Controllern mit gemeinsamem Zugriff auf denselben Satz an Laufwerken. In der Regel besitzt jeder Controller die Hälfte der im System konfigurierten Volumes, jedoch kann für den anderen Controller nach Bedarf die Aufgaben übernehmen.

Multipathing-Software auf den Datei-Knoten bestimmt den aktiven und optimierten Pfad zu jedem Volume und verschiebt sich automatisch nach einem Kabel-, Adapter- oder Controller-Ausfall zum alternativen Pfad.

Das folgende Diagramm zeigt das Controller-Layout in EF600 Block-Nodes.

Um eine Shared-Disk-HA-Lösung zu erleichtern, werden Volumes beiden Datei-Nodes zugeordnet, sodass sie sich bei Bedarf gegenseitig übernehmen können. Das folgende Diagramm zeigt ein Beispiel, wie BeeGFS-Service und die bevorzugte Volumeneigentümer für maximale Performance konfiguriert sind. Die Schnittstelle links von jedem BeeGFS-Dienst gibt die bevorzugte Schnittstelle an, mit der die Clients und andere Dienste Kontakt aufnehmen.



Im vorherigen Beispiel kommunizieren Clients und Server Services lieber mit Storage Service 1 über Schnittstelle i1b. Storage Service 1 verwendet Schnittstelle i1a als bevorzugten Pfad zur Kommunikation mit seinen Volumes (Storage_tgt_101, 102) auf Controller A des ersten Block-Node. Diese Konfiguration nutzt die volle bidirektionale PCIe-Bandbreite, die dem InfiniBand-Adapter zur Verfügung steht, und erreicht mit einem Dual-Port-HDR-InfiniBand-Adapter eine bessere Leistung als bei PCIe 4.0.

BeeGFS-Dateiknoten-Konfiguration

Die BeeGFS Datei-Nodes sind in einem HA-Cluster (High Availability) konfiguriert, um den Failover von BeeGFS-Services zwischen mehreren Datei-Nodes zu ermöglichen.

Das HA Cluster Design basiert auf zwei weit verbreiteten Linux HA Projekten: Corosync für Cluster-Mitgliedschaft und Pacemaker für Cluster Resource Management. Weitere Informationen finden Sie unter "Red hat Training für Add-ons mit hoher Verfügbarkeit".

NetApp hat mehrere Open Cluster Framework (OCF) Resource Agents entwickelt und erweitert, damit das Cluster die BeeGFS Ressourcen intelligent starten und überwachen kann.

BeeGFS HA Cluster

Wenn Sie einen BeeGFS-Dienst starten (mit oder ohne HA), müssen in der Regel einige Ressourcen vorhanden sein:

- IP-Adressen, in denen der Dienst erreichbar ist, werden normalerweise von Network Manager konfiguriert.
- Zugrunde liegende Dateisysteme, die als Ziele für BeeGFS zum Speichern von Daten verwendet werden.

  Diese werden typischerweise in definiert `/etc/fstab` Und montiert von systemd.

- Ein systemd-Service, der für den Start von BeeGFS-Prozessen verantwortlich ist, wenn die anderen Ressourcen bereit sind.

  Ohne zusätzliche Software werden diese Ressourcen nur auf einem einzelnen Datei-Node gestartet. Wenn der Datei-Node offline geschaltet wird, kann auf einen Teil des BeeGFS-Dateisystems zugegriffen werden.

Da mehrere Nodes jeden BeeGFS-Service starten können, muss Pacemaker sicherstellen, dass jeder Service und die abhängigen Ressourcen nur auf jeweils einem Node ausgeführt werden. Wenn beispielsweise zwei Knoten versuchen, denselben BeeGFS-Service zu starten, besteht das Risiko einer Datenbeschädigung, wenn beide versuchen, auf dieselben Dateien auf dem zugrunde liegenden Ziel zu schreiben. Um dieses Szenario zu vermeiden, setzt Pacemaker auf Corosync, um den Zustand des gesamten Clusters zuverlässig über alle Knoten hinweg zu synchronisieren und Quorum zu schaffen.

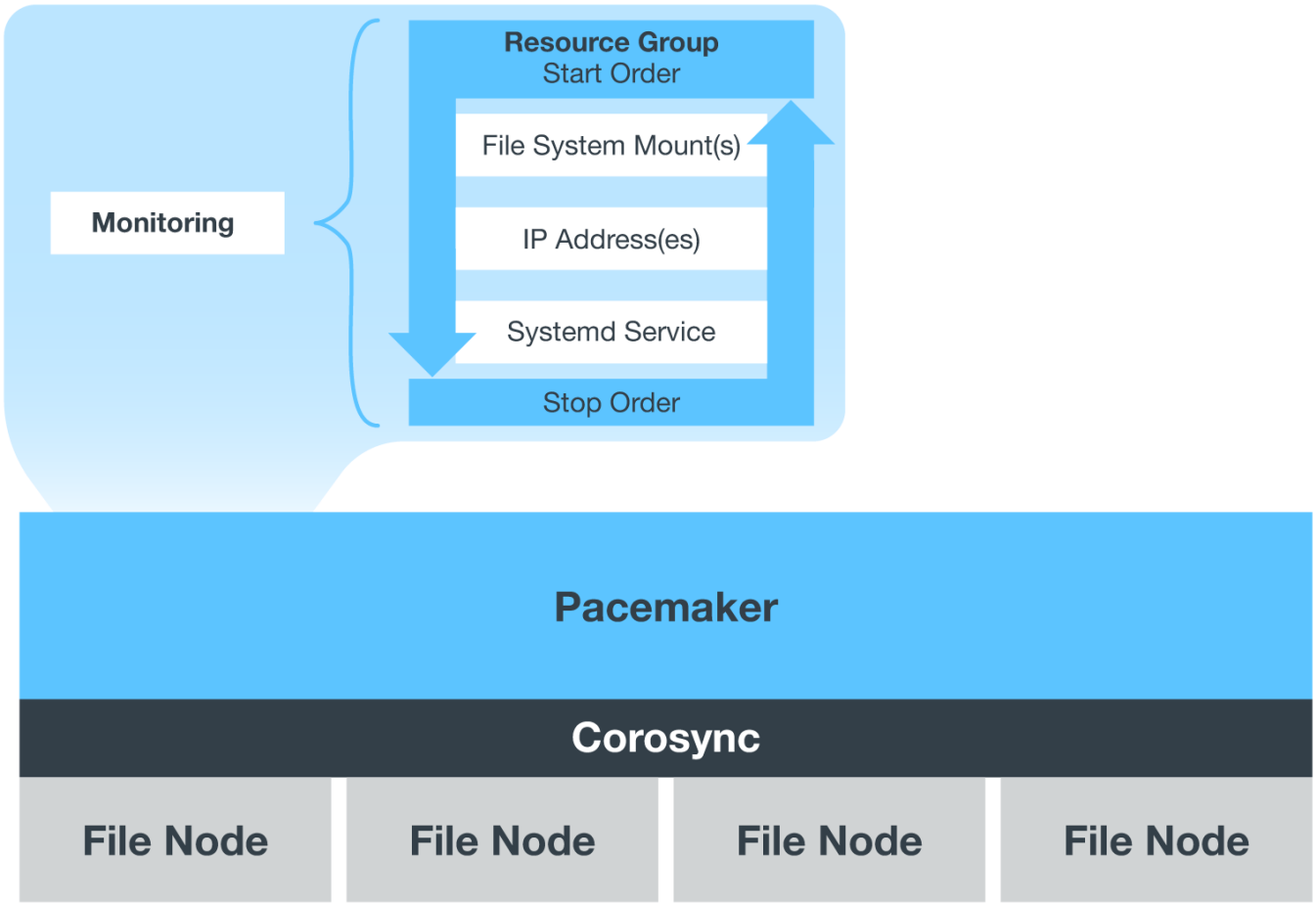
Wenn ein Fehler im Cluster auftritt, reagiert Pacemaker und startet BeeGFS-Ressourcen auf einem anderen Knoten neu. In einigen Fällen kann Pacemaker möglicherweise nicht mit dem ursprünglichen fehlerhaften Knoten kommunizieren, um zu bestätigen, dass die Ressourcen angehalten werden. Um zu überprüfen, ob der Knoten ausgefallen ist, bevor BeeGFS-Ressourcen an anderer Stelle neu gestartet werden, isoliert Pacemaker den fehlerhaften Knoten, idealerweise indem er die Stromversorgung entfernt.

Es stehen zahlreiche Open-Source-Fechten-Agenten zur Verfügung, die es Pacemaker ermöglichen, einen Knoten mit einer Stromverteilungs-Einheit (PDU) oder den Server-Baseboard-Management-Controller (BMC) mit APIs wie Redfish zu Zaun.

Wenn BeeGFS in einem HA-Cluster ausgeführt wird, werden alle BeeGFS-Services und zugrunde liegenden Ressourcen von Pacemaker in Ressourcengruppen gemanagt. Jeder BeeGFS-Service und die Ressourcen, auf die er angewiesen ist, werden in einer Ressourcengruppe konfiguriert, die sicherstellt, dass Ressourcen in der richtigen Reihenfolge gestartet und gestoppt werden und auf demselben Node zusammengelegt werden.

Für jede BeeGFS-Ressourcengruppe führt Pacemaker eine benutzerdefinierte BeeGFS-Überwachungsressource aus, die für die Erkennung von Fehlerbedingungen und die intelligente Auslösung von Failover verantwortlich ist, wenn auf einem bestimmten Knoten kein BeeGFS-Dienst mehr verfügbar ist.

Die folgende Abbildung zeigt die Pacemaker-gesteuerten BeeGFS-Dienste und -Abhängigkeiten.



Damit mehrere BeeGFS-Dienste desselben Typs auf demselben Knoten gestartet werden, wird Pacemaker so konfiguriert, dass BeeGFS-Dienste mit der Multi-Mode-Konfigurationsmethode gestartet werden. Weitere Informationen finden Sie im "BeeGFS-Dokumentation im Multi-Modus".

Da BeeGFS-Dienste auf mehreren Nodes starten können müssen, muss die Konfigurationsdatei für jeden Dienst (normalerweise bei gefunden `/etc/beegfs`) Wird auf einem der E-Series Volumes gespeichert, die als BeeGFS-Ziel für diesen Service verwendet werden. Damit sind die Konfiguration zusammen mit den Daten für einen bestimmten BeeGFS Service für alle Nodes zugänglich, die den Service möglicherweise ausführen müssen.

```
# 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
```

Design-Überprüfung

Das Design der zweiten Generation für die BeeGFS auf NetApp Lösung wurde mithilfe von drei Bausteinkonfigurationsprofilen verifiziert.

Die Konfigurationsprofile umfassen Folgendes:

- Ein einzelner Baustein, einschließlich BeeGFS-Management, Metadaten und Storage-Services
- Ein BeeGFS Metadaten plus ein Storage-Baustein.
- Ein BeeGFS-Speicherbaustein.

Die Bausteine wurden mit zwei NVIDIA Quantum InfiniBand (MQM8700) Switches verbunden. Zehn BeeGFS Clients wurden auch an die InfiniBand Switches angeschlossen und zur Ausführung von Synthetic Benchmark Utilities verwendet.

Die folgende Abbildung zeigt die BeeGFS-Konfiguration, die zur Validierung der BeeGFS auf NetApp Lösung verwendet wird.

BeeGFS-Datei-Striping

Ein Vorteil paralleler Dateisysteme besteht darin, einzelne Dateien über mehrere Storage-Ziele zu verteilen, wodurch Volumes auf demselben oder verschiedenen zugrunde liegenden Storage-Systemen dargestellt werden können.

In BeeGFS können Sie Striping auf Verzeichnisbasis und pro Datei konfigurieren, um die Anzahl der für jede Datei verwendeten Ziele zu steuern und die für jeden Dateistripe verwendete Chunksize (oder Blockgröße) zu steuern. Bei dieser Konfiguration kann das Filesystem verschiedene Workload- und I/O-Profile unterstützen, ohne Services neu konfigurieren oder neu starten zu müssen. Sie können Stripe-Einstellungen mit dem anwenden `beegfs-ctl` Befehlszeilen-Tool oder Anwendungen, die die Striping-API verwenden. Weitere Informationen finden Sie in der BeeGFS-Dokumentation für "Striping" Und "Striping-API".

Um eine optimale Leistung zu erzielen, wurden während des Tests Streifenmuster angepasst und die für jeden Test verwendeten Parameter werden notiert.

IOR-Bandbreitentests: Mehrere Clients

Die IOR-Bandbreitentests verwendeten OpenMPI, um parallele Jobs des synthetischen E/A-Generatorwerkzeugs IOR auszuführen (verfügbar unter "HPC GitHub") Über alle 10 Client-Knoten zu einem oder mehreren BeeGFS-Bausteinen. Sofern nicht anders angegeben:

- Alle Tests verwendeten einen direkten I/O mit einer Übertragungsgröße von 1 MiB.
- BeeGFS-Datei-Striping wurde auf 1 MB Chunksize und ein Ziel pro Datei eingestellt.

Für IOR wurden die folgenden Parameter verwendet, wobei die Segmentanzahl angepasst wurde, um die aggregierte Dateigröße für einen Baustein auf 5 tib und 40 tib für drei Bausteine zu halten.

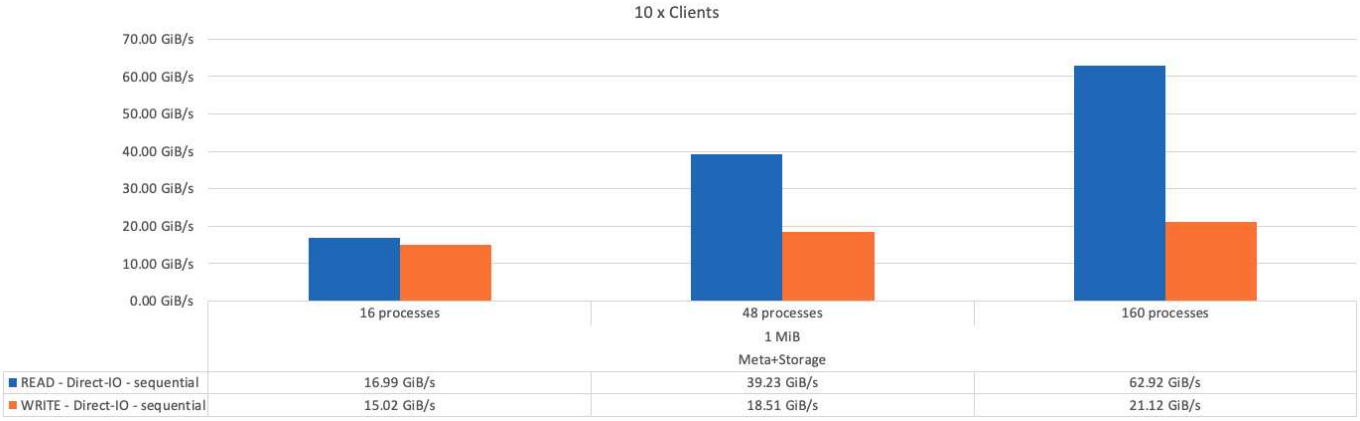
```
mpirun --allow-run-as-root --mca btl tcp -np 48 -map-by node -hostfile
10xnodes ior -b 1024k --posix.odirect -e -t 1024k -s 54613 -z -C -F -E -k
```

Baustein für eine BeeGFS-Basis (Management, Metadaten und Storage

Die folgende Abbildung zeigt die IOR-Testergebnisse mit einem einzelnen BeeGFS-Basisspeicher (Management, Metadaten und Storage).

BeeGFS Metadaten + Storage-Baustein

Die folgende Abbildung zeigt die IOR-Testergebnisse mit einem einzelnen BeeGFS-Metadaten + einem Storage-Baustein.

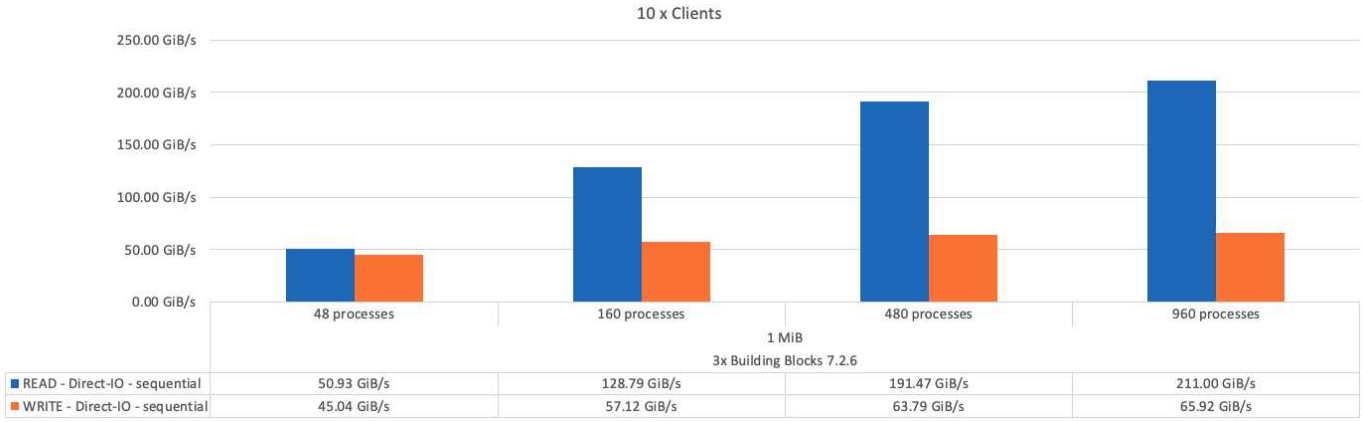


BeeGFS-Speicherbaustein

Die folgende Abbildung zeigt die IOR-Testergebnisse mit einem einzelnen BeeGFS-Storage-only-Baustein.

Drei BeeGFS-Bausteine

Die folgende Abbildung zeigt die IOR-Testergebnisse mit drei BeeGFS-Bausteinen.



Wie erwartet, ist der Performance-Unterschied zwischen dem Basis-Baustein und den nachfolgenden Metadaten + dem Storage-Baustein vernachlässigbar. Ein Vergleich zwischen Metadaten und Storage-Bausteinen und einem ausschließlich Storage-Baustein zeigt einen leichten Anstieg der Lese-Performance aufgrund der zusätzlichen Laufwerke, die als Storage-Ziele verwendet werden. Allerdings gibt es keinen wesentlichen Unterschied in der Schreib-Performance. Um eine höhere Performance zu erzielen, können Sie mehrere Bausteine gleichzeitig hinzufügen, um die Performance linear zu skalieren.

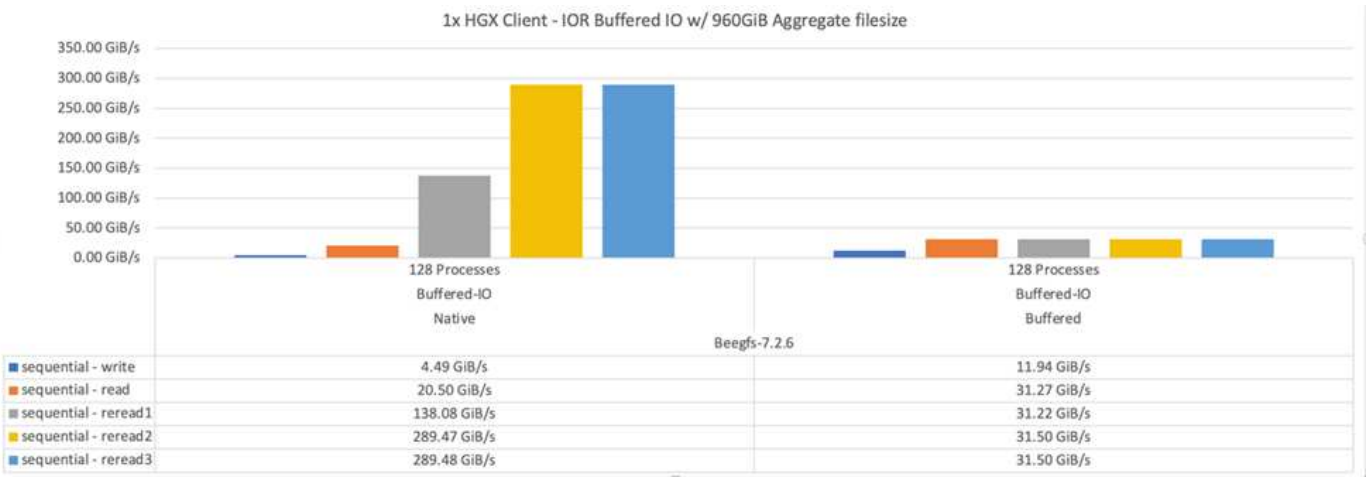
IOR-Bandbreitentests: Einzelner Client

Der IOR-Bandbreitentest nutzte OpenMPI, um mehrere IOR-Prozesse mithilfe eines einzigen leistungsstarken GPU-Servers auszuführen, um die Performance zu untersuchen, die für einen einzelnen Client erreichbar ist.

Dieser Test vergleicht auch das Verhalten und die Leistung von BeeGFS, wenn der Client so konfiguriert ist, dass er den Linux-Kernel-Page-Cache verwendet (`tuneFileCacheType = native`) Im Vergleich zum Standard `buffered` Einstellung.

Der native Caching-Modus verwendet den Linux-Kernel-Page-Cache auf dem Client, sodass die Readervorgänge nicht über das Netzwerk übertragen werden, sondern aus dem lokalen Speicher stammen.

Das folgende Diagramm zeigt die IOR-Testergebnisse mit drei BeeGFS-Bausteinen und einem einzelnen Client.



BeeGFS Striping für diese Tests wurde auf 1 MB Chunksize mit acht Zielen pro Datei eingestellt.

Obwohl die Performance bei den Schreibzugriffen und beim ersten Lesen im standardmäßigen gepufferten Modus höher ist, werden bei Workloads, die dieselben Daten mehrmals lesen, eine deutliche Performance-

Steigerung im nativen Caching-Modus erzielt. Diese verbesserte Performance bei erneuten Lesevorgängen ist für Workloads wie Deep Learning wichtig, die denselben Datensatz mehrmals in vielen Epoch-Durchläufen lesen.

Metadaten-Performance-Test

Bei den Metadaten-Performance-Tests wurde das MDTest-Tool (im Rahmen von IOR enthalten) verwendet, um die Metadaten-Performance von BeeGFS zu messen. Die Tests verwendeten OpenMPI, um parallele Jobs auf allen zehn Client-Knoten auszuführen.

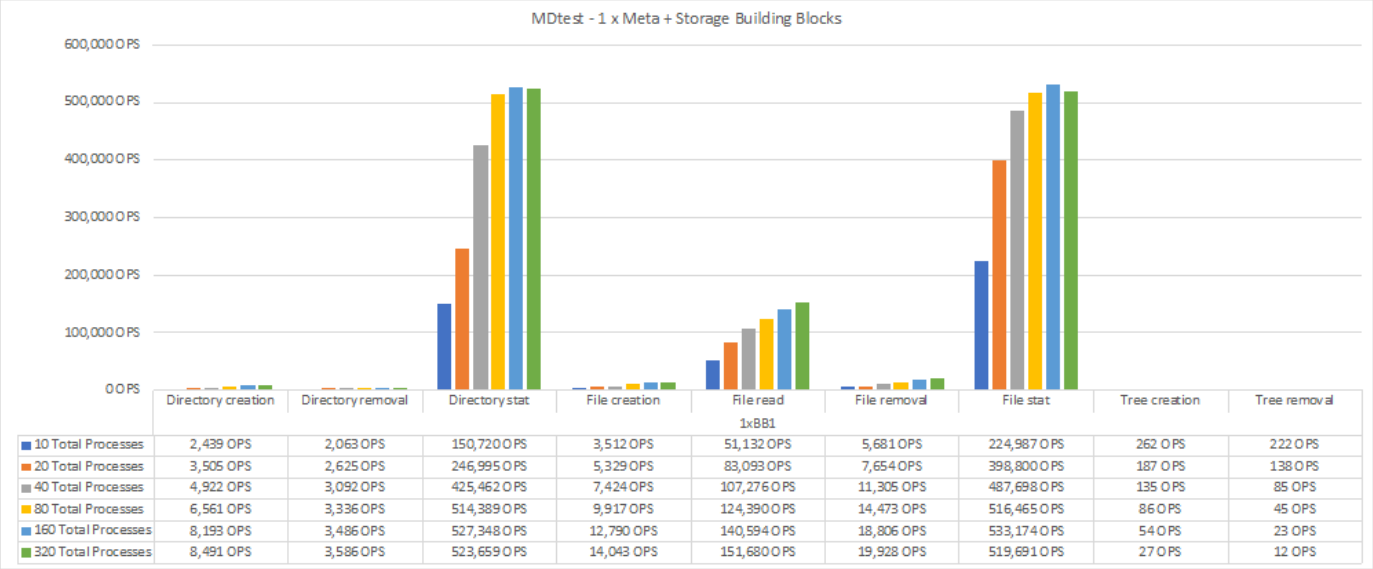
Die folgenden Parameter wurden verwendet, um den Benchmark-Test mit der Gesamtzahl der Prozesse von 10 auf 320 in Schritt 2 x und mit einer Dateigröße von 4k durchgeführt.

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

Die Metadaten-Performance wurde zuerst mit ein bis zwei Metadaten + Storage-Bausteinen gemessen, um die Performance durch das Hinzufügen weiterer Bausteine zu verdeutlichen.

1 BeeGFS Metadaten + Storage-Baustein

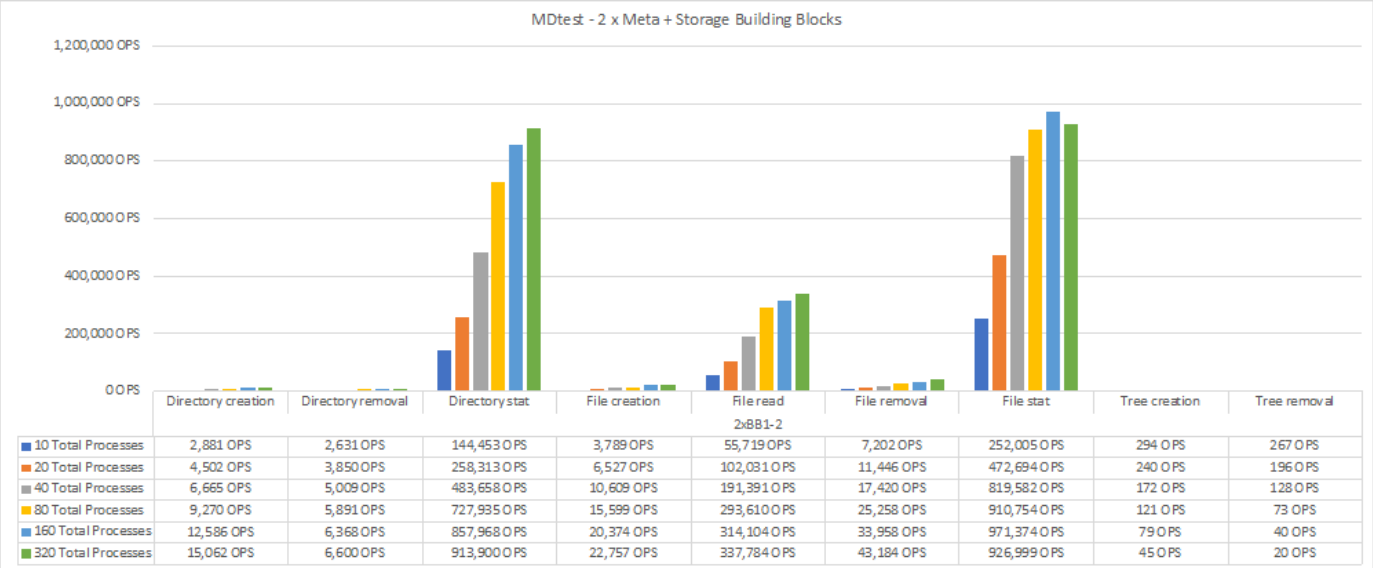
Das folgende Diagramm zeigt die MDTest-Ergebnisse mit einer BeeGFS-Metadaten + Speicherbausteinen.



1xBB1	Directory creation	Directory removal	Directory stat	File creation	File read	File removal	File stat	Tree creation	Tree removal
10 Total Processes	2,439 OPS	2,063 OPS	150,720 OPS	3,512 OPS	51,132 OPS	5,681 OPS	224,987 OPS	262 OPS	222 OPS
20 Total Processes	3,505 OPS	2,625 OPS	246,995 OPS	5,329 OPS	83,093 OPS	7,654 OPS	398,800 OPS	187 OPS	138 OPS
40 Total Processes	4,922 OPS	3,092 OPS	425,462 OPS	7,424 OPS	107,276 OPS	11,305 OPS	487,698 OPS	135 OPS	85 OPS
80 Total Processes	6,561 OPS	3,336 OPS	514,389 OPS	9,917 OPS	124,390 OPS	14,473 OPS	516,465 OPS	86 OPS	45 OPS
160 Total Processes	8,193 OPS	3,486 OPS	527,348 OPS	12,790 OPS	140,594 OPS	18,806 OPS	533,174 OPS	54 OPS	23 OPS
320 Total Processes	8,491 OPS	3,586 OPS	523,659 OPS	14,043 OPS	151,680 OPS	19,928 OPS	519,691 OPS	27 OPS	12 OPS

Zwei BeeGFS Metadaten + Storage-Bausteine

Das folgende Diagramm zeigt die MDTest-Ergebnisse mit zwei BeeGFS-Metadaten + Speicherbausteinen.



	Directory creation	Directory removal	Directory stat	File creation	File read	File removal	File stat	Tree creation	Tree removal
					2xBB1-2				
10 Total Processes	2,881 OPS	2,631 OPS	144,453 OPS	3,789 OPS	55,719 OPS	7,202 OPS	252,005 OPS	294 OPS	267 OPS
20 Total Processes	4,502 OPS	3,850 OPS	258,313 OPS	6,527 OPS	102,031 OPS	11,446 OPS	472,694 OPS	240 OPS	196 OPS
40 Total Processes	6,665 OPS	5,009 OPS	483,658 OPS	10,609 OPS	191,391 OPS	17,420 OPS	819,582 OPS	172 OPS	128 OPS
80 Total Processes	9,270 OPS	5,891 OPS	727,935 OPS	15,599 OPS	293,610 OPS	25,258 OPS	910,754 OPS	121 OPS	73 OPS
160 Total Processes	12,586 OPS	6,368 OPS	857,968 OPS	20,374 OPS	314,104 OPS	33,958 OPS	971,374 OPS	79 OPS	40 OPS
320 Total Processes	15,062 OPS	6,600 OPS	913,900 OPS	22,757 OPS	337,784 OPS	43,184 OPS	926,999 OPS	45 OPS	20 OPS

Funktionsprüfung

Im Rahmen der Validierung dieser Architektur führte NetApp mehrere Funktionstests durch, darunter:

- Ausfall eines einzelnen InfiniBand-Ports des Clients durch Deaktivieren des Switch-Ports
- Ausfall eines InfiniBand-Ports mit einem einzelnen Server durch Deaktivieren des Switch-Ports
- Sofortige Abschaltung des Servers mithilfe des BMC.
- Anmutig Platzierung eines Node im Standby-Modus und Failover-Betrieb zu einem anderen Node
- Anmutig Setzen eines Node wieder online und Failback-Services auf den ursprünglichen Node.
- Schalten Sie einen der InfiniBand-Switches mithilfe der PDU aus. Alle Tests wurden durchgeführt, während Belastungstests mit dem durchgeführt wurden `sysSessionChecksEnabled: false` Parameter auf BeeGFS-Clients gesetzt. Es wurden keine Fehler oder Störungen bei I/O festgestellt.

> Es ist ein bekanntes Problem aufgetreten (siehe "Changelog") Wenn BeeGFS-Client/Server-RDMA-Verbindungen unerwartet unterbrochen werden, entweder durch Ausfall der primären Schnittstelle (wie in definiert `connInterfacesFile`) Oder ein BeeGFS-Server fällt aus. Aktive Client-I/O kann bis zu zehn Minuten lang aufhängen, bevor der Vorgang fortgesetzt wird. Dieses Problem tritt nicht auf, wenn BeeGFS-Knoten ordnungsgemäß für geplante Wartung in den Standby-Modus versetzt oder TCP verwendet wird.

Validierung von NVIDIA DGX SuperPOD und BasePOD

NetApp validierte eine Storage-Lösung für IANVIDDGX A100 SuperPOD unter Verwendung eines ähnlichen BeeGFS Filesystem, das aus drei Bausteinen mit den Metadaten und angewandtem Storage-Konfigurationsprofil besteht. Die Qualifizierung bestand darin, die von dieser NVA beschriebene Lösung mit zwanzig DGX A100 GPU-Servern zu testen, auf denen eine Vielzahl von Storage-, Machine-Learning- und Deep-Learning-Benchmarks ausgeführt wurden. Aufbauend auf der Validierung mit DGX A100 SuperPOD von NVIDIA wurde die BeeGFS auf NetApp Lösung für DGX SuperPOD H100-, H200- und B200-Systeme genehmigt. Diese Erweiterung basiert auf der Erfüllung der zuvor mit dem NVIDIA DGX A100 validierten Benchmarks und Systemanforderungen.

Weitere Informationen finden Sie unter "NVIDIA DGX SuperPOD mit NetApp" Und "NVIDIA DGX BasePOD".

Richtlinien für die Dimensionierung

Die BeeGFS Lösung enthält Empfehlungen für das Performance- und Kapazitäts-Sizing, die auf Verifizierungstests basieren.

Die Bausteinarchitektur verfolgt das Ziel, eine Lösung zu entwickeln, die sich einfach dimensionieren lässt. Dazu werden mehrere Bausteine hinzugefügt, die die Anforderungen für ein bestimmtes BeeGFS-System erfüllen. Mithilfe der nachstehenden Richtlinien können Sie die Anzahl und die Arten von BeeGFS-Bausteinen schätzen, die für die Anforderungen Ihrer Umgebung benötigt werden.

Beachten Sie, dass es sich bei diesen Schätzungen um die beste Performance handelt. Benchmark-Applikationen werden geschrieben und verwendet, um die Nutzung der zugrunde liegenden Filesysteme auf eine Weise zu optimieren, die reale Applikationen vielleicht nicht erreichen.

Performance-Dimensionierung

Die folgende Tabelle enthält ein empfohlene Performance-Sizing.

Konfigurationsprofil	1 MiB Lesevorgänge	1 MiB Schreibvorgänge
Metadaten + Storage	62Gibps	21Gibps
Nur Storage	64Gibps	21Gibps

Die Schätzungen zur Größe der Metadaten-Kapazität basieren auf der „Faustregel“, dass eine Kapazität von 500 GB für etwa 150 Millionen Dateien in BeeGFS ausreichend ist. (Weitere Informationen finden Sie in der BeeGFS-Dokumentation für "Systemanforderungen".)

Die Verwendung von Funktionen wie Zugriffssteuerungslisten und die Anzahl der Verzeichnisse und Dateien pro Verzeichnis wirkt sich auch darauf aus, wie schnell der Metadatenspeicherplatz verbraucht wird. Schätzungen zur Storage-Kapazität berücksichtigen neben RAID 6 und XFS-Overhead die nutzbare Laufwerkskapazität.

Kapazitätsdimensionierung für Metadaten + Storage-Bausteine

Die folgende Tabelle enthält eine empfohlene Kapazitätsdimensionierung für Metadaten und Storage-Bausteine.

Laufwerksgröße (2+2 RAID 1) Metadaten-Volume-Gruppen	Metadaten-Kapazität (Anzahl der Dateien)	Laufwerksgröße (8+2 RAID 6) Storage-Volume-Gruppen	Speicherkapazität (File-Inhalte)
1,92 TB	1,938,577,200	1,92 TB	51,77 TB
3,84 TB	3,880,388,400	3,84 TB	103,5 TB
7,68 TB	8,125,278,000	7,68 TB	216,74 TB
15,3 TB	17,269,854,000	15,3 TB	460 TB

Bei der Größenbestimmung von Metadaten und Storage-Bausteinen können die Kosten reduziert werden, da weniger Laufwerke für Metadaten-Volume-Gruppen anstelle von Storage-Volume-Gruppen verwendet werden.

Kapazitätsdimensionierung für reine Storage-Bausteine

Die folgende Tabelle zeigt die Größenanpassung der Kapazität für reine Storage-Bausteine.

Laufwerksgröße (10+2 RAID 6) Storage-Volume-Gruppen	Speicherkapazität (File-Inhalte)
1,92 TB	59,89 TB
3,84 TB	119,80 TB
7,68 TB	251,89 TB
15,3 TB	538,5 TB

Die Performance- und der Kapazitäts-Overhead, die durch das Einbinden des Management-Service in den Basis-Baustein (erster) verursacht werden, sind minimal, es sei denn, die globale Dateisperrung ist aktiviert.

Performance-Optimierung

Die BeeGFS-Lösung enthält Empfehlungen für Performance-Tuning, die auf Verifikationstests basieren.

Obwohl BeeGFS Out-of-the-Box-Performance bietet, hat NetApp eine Reihe von empfohlenen Tuning-Parametern entwickelt, um die Performance zu maximieren. Diese Parameter berücksichtigen die Funktionen der zugrunde liegenden Block-Nodes der E-Series und alle speziellen Anforderungen, die für die Ausführung von BeeGFS in einer HA-Architektur mit Shared-Festplatten erforderlich sind.

Performance-Tuning für Datei-Nodes

Zu den verfügbaren Tuning-Parametern, die Sie konfigurieren können, gehören folgende:

1. **Systemeinstellungen im UEFI/BIOS von Datei-Nodes.** um die Leistung zu maximieren, empfehlen wir die Konfiguration der Systemeinstellungen auf dem Server-Modell, das Sie als Datei-Knoten verwenden. Sie konfigurieren die Systemeinstellungen, wenn Sie die Datei-Nodes einrichten, indem Sie entweder das System-Setup (UEFI/BIOS) oder die Redfish-APIs verwenden, die vom Baseboard-Management-Controller (BMC) bereitgestellt werden.

   Die Systemeinstellungen variieren je nach Servermodell, das Sie als Dateiknoten verwenden. Die Einstellungen müssen manuell auf der Grundlage des verwendeten Servermodells konfiguriert werden. Informationen zum Konfigurieren der Systemeinstellungen für die validierten Lenovo SR665 V3-Dateiknoten finden Sie unter "Optimieren Sie die System-Einstellungen des File Node für die Performance" .

2. **Standardeinstellungen für erforderliche Konfigurationsparameter.** die erforderlichen Konfigurationsparameter beeinflussen die Konfiguration von BeeGFS-Diensten und die Formatierung und Bereitstellung von E-Series-Volumes (Blockgeräte) durch Pacemaker. Die folgenden Konfigurationsparameter sind erforderlich:
   - BeeGFS Service-Konfigurationsparameter

     Sie können die Standardeinstellungen für die Konfigurationsparameter bei Bedarf überschreiben. Informationen zu den Parametern, die Sie an Ihre spezifischen Workloads oder Anwendungsfälle anpassen können, finden Sie im "Parameter für BeeGFS-Service-Konfiguration".

- Für die Volume-Formatierung und die Montage-Parameter gelten die empfohlenen Standardeinstellungen, die nur für erweiterte Anwendungsfälle angepasst werden sollten. Die Standardwerte führen folgende Schritte aus:
  - Optimieren Sie die ursprüngliche Volume-Formatierung auf Basis des Zieltyps (beispielsweise Management, Metadaten oder Storage), zusammen mit der RAID-Konfiguration und Segmentgröße des zugrunde liegenden Volumes.
  - Passen Sie an, wie Pacemaker die einzelnen Lautstärkerahmen einstellt, um sicherzustellen, dass Änderungen sofort an Block-Nodes der E-Series gespült werden. So wird Datenverlust verhindert, wenn bei aktiven Schreibvorgängen Datei-Nodes ausfallen.

    Informationen zu den Parametern, die Sie an Ihre spezifischen Workloads oder Anwendungsfälle anpassen können, finden Sie im "Volume-Formatierung und Montage der Konfigurationsparameter".

3. **Systemeinstellungen im Linux-Betriebssystem, das auf den Dateiknoten installiert ist.** Sie können die standardmäßigen Linux OS-Systemeinstellungen überschreiben, wenn Sie den Ansible-Bestand in Schritt 4 von erstellen "Erstellen des Ansible-Inventars".

   Die Standardeinstellungen wurden zur Validierung der BeeGFS auf NetApp Lösung verwendet. Sie können sie jedoch an Ihre spezifischen Workloads oder Anwendungsfälle anpassen. Einige Beispiele für die Linux-Betriebssystemeinstellungen, die Sie ändern können, sind:

   - I/O-Warteschlangen an Block-Geräten der E-Series.

     Auf den als BeeGFS-Ziele verwendeten E-Series-Block-Geräten können Sie I/O-Warteschlangen folgendermaßen konfigurieren:

     - Passen Sie den Planungsalgorithmus basierend auf dem Gerätetyp (NVMe, HDD usw.) an.
     - Erhöhen Sie die Anzahl der ausstehenden Anfragen.
     - Passen Sie Anfragegrößen an.
     - Optimierung des Verhaltens beim Lesen.
   - Einstellungen für virtuellen Speicher.

     Sie können die Einstellungen für den virtuellen Speicher für eine optimale Streaming-Leistung anpassen.

   - CPU-Einstellungen.

     Sie können den CPU-Frequenzregler und andere CPU-Konfigurationen für maximale Leistung anpassen.

   - Anfragegröße lesen

     Sie können die maximale Größe für Leseanforderungen für NVIDIA-HCAs erhöhen.

Performance-Tuning für Block-Nodes

Basierend auf den Konfigurationsprofilen, die auf einen bestimmten BeeGFS-Baustein angewendet werden, ändern sich die auf den Block-Nodes konfigurierten Volume-Gruppen leicht. Beispiel mit einem EF600 Block-Node mit 24 Laufwerken:

- Für den einzelnen Basis-Block, einschließlich BeeGFS-Management, Metadaten und Storage-Services:

- 1x 2+2 RAID 10 Volume-Gruppe für BeeGFS Management und Metadaten-Services
- 2 x 8+2 RAID 6-Volume-Gruppen für BeeGFS-Storage-Services

- Für BeeGFS Metadaten + Storage-Baustein:
  - 1x 2+2 RAID 10 Volume-Gruppe für BeeGFS Metadaten-Services
  - 2 x 8+2 RAID 6-Volume-Gruppen für BeeGFS-Storage-Services
- Nur für BeeGFS-Lagergebäude:
  - 2 x 10+2 RAID 6-Volume-Gruppen für BeeGFS-Storage-Services

> Da BeeGFS für Management und Metadaten im Gegensatz zu Storage erheblich weniger Speicherplatz benötigt, besteht die Möglichkeit, für die RAID 10-Volume-Gruppen kleinere Laufwerke zu verwenden. Kleinere Laufwerke sollten in den äußeren Laufwerksteckplätzen befüllt werden. Weitere Informationen finden Sie im "Implementierungsanleitungen".

Diese werden alle durch die Ansible-basierte Implementierung konfiguriert, und verschiedene andere allgemein empfohlene Einstellungen für Performance-/Verhaltensoptimierung:

- Anpassung der globalen Cache-Blockgröße auf 32 KiB und Anpassung der bedarfsorientierten Cache-Flush auf 80 %.
- Deaktivieren des automatischen Load-Balancing (sicherstellen, dass die Controller-Volume-Zuweisungen wie vorgesehen bleiben).
- Aktivieren von Lese-Caching und Deaktivieren des Read-Ahead-Caching
- Aktivieren des Schreib-Caches mit Spiegelung und Bedarf an Akku-Backups, sodass Caches während des Ausfalls eines Block-Node-Controllers bestehen.
- Festlegen der Reihenfolge, in der Laufwerke Volume-Gruppen zugewiesen werden, und Ausgleich der I/O-Vorgänge über verfügbare Laufwerkskanäle

Baustein mit hoher Kapazität

Bei der Entwicklung der BeeGFS-Standardlösung wurde auf Workloads mit hoher Performance gedenkt. Kunden, die nach Anwendungsfällen mit hoher Kapazität suchen, sollten die hier beschriebenen Design- und Performance-Unterschiede beobachten.

Hardware- und Softwarekonfiguration

Hardware- und Softwarekonfiguration für den Baustein mit hoher Kapazität ist Standard. Es sei denn, die EF600 Controller sollten durch einen EF300 Controller ersetzt werden. Sie können zwischen 1 und 7 IOM-Erweiterungsfächern mit jeweils 60 Laufwerken pro Storage-Array anschließen, Insgesamt 2 bis 14 Erweiterungsfächer pro Baustein.

Unternehmen, die ein High-Capacity-Baustein-Design implementieren, verwenden wahrscheinlich nur die Basis-Bausteinkonfiguration, die aus BeeGFS-Management, Metadaten und Storage-Services für jeden Node besteht. Um die Kosteneffizienz zu steigern, sollten Storage-Nodes mit hoher Kapazität Metadaten-Volumes auf den NVMe-Laufwerken im EF300-Controller-Gehäuse bereitstellen und Storage-Volumes für die NL-SAS-Laufwerke in den Erweiterungsfächern bereitstellen.

[]

Richtlinien für die Dimensionierung

Bei diesen Richtlinien zur Dimensionierung ist davon auszugehen, dass Bausteine mit hoher Kapazität mit einer NVMe-SSD-Volume-Gruppe von 2+2 für Metadaten im Basis-EF300-Gehäuse und 6 x 8+2 NL-SAS-Volume-Gruppen pro IOM-Erweiterungsfach für Storage konfiguriert sind.

Laufwerkgröße (Kapazitäts-HDDs)	Kapazität pro BB (1 Fach)	Kapazität pro BB (2 Einschübe)	Kapazität pro BB (3 Einschübe)	Kapazität pro BB (4 Einschübe)
4 TB	439 TB	878 TB	1317 TB	1756 TB
8 TB	878 TB	1756 TB	2634 TB	3512 TB
10 TB	1097 TB	2195 TB	3292 TB	4390 TB
12 TB	1317 TB	2634 TB	3951 TB	5268 TB
16 TB	1756 TB	3512 TB	5268 TB	7024 TB
18 TB	1975 TB	3951 TB	5927 TB	7902 TB

Implementieren der Lösung

Implementierungsübersicht

BeeGFS auf NetApp kann auf validierte Datei- und Block-Nodes mithilfe von Ansible mit dem BeeGFS-Baustein-Design von NetApp implementiert werden.

Ansible-Sammlungen und -Funktionen

Die BeeGFS auf NetApp-Lösung wird mithilfe von Ansible implementiert, einer beliebten IT-Automatisierungs-Engine, die Applikationsimplementierungen automatisiert. Ansible verwendet eine Reihe von Dateien, die gemeinsam als Inventar bezeichnet werden. Hierbei wird das BeeGFS-Filesystem modelliert, das Sie implementieren möchten.

Ansible ermöglicht Unternehmen wie NetApp die Erweiterung auf integrierte Funktionen mithilfe von Sammlungen, die auf Ansible Galaxy verfügbar sind (siehe "NetApp E-Series BeeGFS Sammlung"). Sammlungen umfassen Module, die bestimmte Funktionen oder Aufgaben (wie das Erstellen eines E-Series Volumes) ausführen, sowie Rollen, die mehrere Module und andere Rollen aufrufen können. Dieser automatisierte Ansatz reduziert die Zeit für die Implementierung des BeeGFS-Filesystems und des zugrunde liegenden HA-Clusters. Darüber hinaus vereinfacht es die Wartung und Erweiterung des Clusters und des BeeGFS-Dateisystems.

Weitere Informationen finden Sie unter "Weitere Informationen zum Ansible Inventar".

> Da zahlreiche Schritte an der Implementierung der BeeGFS auf NetApp-Lösung beteiligt sind, unterstützt NetApp die manuelle Bereitstellung der Lösung nicht.

Konfigurationsprofile für BeeGFS-Bausteine

Die Implementierungsverfahren umfassen die folgenden Konfigurationsprofile:

- Ein einziger Baustein, der Management-, Metadaten- und Storage-Services umfasst
- Ein zweiter Baustein, der Metadaten und Storage-Services umfasst.

- Ein dritter Baustein, der nur Storage-Services umfasst.

Diese Profile veranschaulichen die gesamte Palette der empfohlenen Konfigurationsprofile für die NetApp BeeGFS-Bausteine. Bei jeder Implementierung kann die Anzahl der Metadaten und Storage-Bausteine oder nur-Storage-Services-Bausteine je nach Kapazitäts- und Performance-Anforderungen variieren.

Übersicht über die einzelnen Implementierungsschritte

Die Bereitstellung umfasst folgende allgemeine Aufgaben:

Hardwarebereitstellung

1. Stellen Sie jeden Baustein physisch zusammen.
2. Rack-und Kabelhardware: Ausführliche Verfahren finden Sie unter "Implementierung von Hardware".

Softwareimplementierung

1. "Richten Sie Datei- und Block-Nodes ein".
   - Konfigurieren Sie BMC-IPs auf Datei-Knoten
   - Installieren Sie ein unterstütztes Betriebssystem und konfigurieren Sie Managementnetzwerk auf Datei-Knoten
   - Konfiguration der Management-IPs auf Block-Nodes
2. "Richten Sie einen Ansible-Steuerungsknoten ein".
3. "Passen Sie die Systemeinstellungen für die Performance an".
4. "Erstellen des Ansible-Inventars".
5. "Definieren Sie den Ansible-Bestand für BeeGFS-Bausteine".
6. "Implementieren Sie BeeGFS mit Ansible".
7. "Konfigurieren Sie BeeGFS-Clients".

> Die Bereitstellungsverfahren umfassen mehrere Beispiele, in denen Text in eine Datei kopiert werden muss. Achten Sie besonders auf Inline-Kommentare, die durch die Zeichen „#“ oder „//“ gekennzeichnet sind und auf alles hinweisen, was für eine bestimmte Bereitstellung geändert werden sollte oder kann. Beispiel:
>
> ```
> `beegfs_ha_ntp_server_pools:  # THIS IS AN EXAMPLE OF A COMMENT!
>   - "pool 0.pool.ntp.org iburst maxsources 3"
>   - "pool 1.pool.ntp.org iburst maxsources 3"`
> ```

Derivative Architekturen mit Variationen bei Implementierungsempfehlungen:

- "Baustein Mit Hoher Kapazität"

Weitere Informationen zum Ansible Inventar

Machen Sie sich vor der Implementierung mit der Konfiguration von Ansible vertraut und werden Sie zur Implementierung der BeeGFS auf NetApp Lösung verwendet.

Der Ansible-Bestand ist eine Verzeichnisstruktur mit den Datei- und Block-Nodes für das zu implementierende

BeeGFS-Filesystem. Es enthält Hosts, Gruppen und Variablen, die das gewünschte BeeGFS-Dateisystem beschreiben. Die Ansible-Bestandsaufnahme muss auf dem Ansible-Steuerungsknoten gespeichert werden, bei dem es sich um jeden Computer mit Zugriff auf die Datei- und Block-Nodes handelt, mit denen das Ansible-Playbook ausgeführt wird. Probenbestände können von der heruntergeladen werden "NetApp E-Series BeeGFS GitHub".

Ansible-Module und -Rollen

Um die im Ansible Inventar beschriebene Konfiguration anzuwenden, verwenden Sie die verschiedenen Ansible Module und Rollen aus der NetApp E-Series Ansible Sammlung (verfügbar über "NetApp E-Series BeeGFS GitHub"), die die End-to-End-Lösung implementieren.

Jede Rolle der NetApp E-Series Ansible Sammlung ist eine vollständige End-to-End-Implementierung der BeeGFS auf NetApp Lösung. Die Rollen verwenden die Sammlungen NetApp E-Series SANtricity, Host und BeeGFS, mit denen Sie das BeeGFS Filesystem mit HA (High Availability, Hochverfügbarkeit) konfigurieren können. Anschließend können Sie Storage bereitstellen und zuordnen und den Cluster-Storage betriebsbereit machen.

Die Rollen verfügen über eine ausführliche Dokumentation. In den Implementierungsverfahren wird beschrieben, wie die Rolle bei der Implementierung einer NetApp Verified Architecture mit dem BeeGFS-Bausteindesign der zweiten Generation eingesetzt wird.

> Obwohl die Implementierungsschritte versucht, genügend Details bereitzustellen, sodass frühere Erfahrungen mit Ansible nicht erforderlich sind, sollten Sie mit Ansible und der zugehörigen Terminologie vertraut sein.

Bestandslayout für BeeGFS HA-Cluster

Definieren Sie ein BeeGFS HA-Cluster mit der Ansible-Bestandsstruktur.

Jeder mit früheren Ansible-Erfahrungen sollte sich bewusst sein, dass die BeeGFS-HA-Rolle eine benutzerdefinierte Methode implementiert, um zu ermitteln, welche Variablen (oder Fakten) für jeden Host gelten. Dieses Design vereinfacht die Strukturierung des Ansible-Bestands, um Ressourcen zu beschreiben, die auf mehreren Servern ausgeführt werden können.

Ein Ansible-Inventar besteht in der Regel aus den Dateien in `host_vars` und `group_vars` sowie einer `inventory.yml` Datei, die Hosts spezifischen Gruppen (und potenziell Gruppen anderen Gruppen) zuweist.

> Erstellen Sie keine Dateien mit dem Inhalt in diesem Unterabschnitt, der nur als Beispiel gedacht ist.

Obwohl diese Konfiguration anhand des Konfigurationsprofils vorab festgelegt ist, sollten Sie wie folgt allgemeine Kenntnisse darüber haben, wie alles als Ansible-Inventar ausgelegt ist:

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp01:
        netapp02:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
        meta_01:  # Group representing a metadata service with ID 01.
          hosts:
            beegfs_01:  # This service is preferred on the first file
node.
            beegfs_02:  # And can failover to the second file node.
        meta_02:  # Group representing a metadata service with ID 02.
          hosts:
            beegfs_02:  # This service is preferred on the second file
node.
            beegfs_01: # And can failover to the first file node.
```

Für jeden Dienst wird unter eine zusätzliche Datei erstellt `group_vars` Beschreibung der Konfiguration:

```
# 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>
  - i2b: <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:
  netapp01:
    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: 21.25
            owning_controller: A
```

Mit diesem Layout können BeeGFS-Service-, Netzwerk- und Storage-Konfigurationen für jede Ressource an einem Ort definiert werden. Hinter den Kulissen aggregiert die Rolle BeeGFS basierend auf dieser Bestandsstruktur die erforderliche Konfiguration für jede Datei und jeden Block-Node.

> Die numerische BeeGFS- und String-Node-ID für jeden Dienst wird automatisch auf Basis des Gruppennamens konfiguriert. Zusätzlich zur allgemeinen Ansible-Anforderung, dass Gruppennamen eindeutig sein sollen, müssen Gruppen, die einen BeeGFS-Service darstellen, in einer Zahl enden, die für den BeeGFS-Service eindeutig ist, für den diese Gruppe repräsentiert. Zum Beispiel sind meta_01 und stor_01 zulässig, aber Metadaten_01 und meta_01 sind nicht.

Besprechen der Best Practices

Beachten Sie bei der Implementierung der BeeGFS auf NetApp-Lösung die Best Practice-Richtlinien.

Standardkonventionen

Folgen Sie beim physischen Zusammenbau und Erstellen der Ansible-Bestandsdatei diesen Standardkonventionen (weitere Informationen finden Sie unter "Erstellen des Ansible-Inventars").

- Host-Namen der Dateiknoten werden nacheinander nummeriert (h01-HN) mit niedrigeren Zahlen oben im Rack und höheren Zahlen unten.

  Die Namenskonvention sieht beispielsweise `[location][row][rack]hN` wie folgt aus: `beegfs_01`.

- Jeder Block-Node besteht aus zwei Storage-Controllern, die jeweils über einen eigenen Host-Namen verfügen.

  Mit einem Storage-Array-Namen wird im Rahmen eines Ansible-Inventars das gesamte Block-Storage-System bezeichnet. Die Namen des Speicher-Arrays sollten nacheinander nummeriert sein (a01 - an), und die Hostnamen für einzelne Controller werden aus dieser Namenskonvention abgeleitet.

  Beispielsweise kann bei einem Block-Node mit dem Namen `ictad22a01` normalerweise Hostnamen für jeden Controller wie und konfiguriert `ictad22a01-a ictad22a01-b`sein, in einem Ansible-Inventar jedoch als bezeichnet werden `netapp_01`.

- File- und Block-Nodes innerhalb desselben Bausteins teilen sich das gleiche Nummerierungsschema und sind im Rack nebeneinander, wobei beide Datei-Nodes oben und beide Block-Nodes direkt darunter liegen.

  Im ersten Baustein sind beispielsweise die Datei-Nodes h01 und h02 direkt mit den Block-Nodes a01 und a02 verbunden. Von oben nach unten sind die Hostnamen h01, h02, a01 und a02.

- Bausteine werden in sequenzieller Reihenfolge auf der Grundlage ihrer Hostnamen installiert, sodass sich die niedrigeren Host-Namen oben im Rack befinden und die höheren nummerierten Host-Namen sich unten befinden.

  Ziel ist es, die Länge des Kabels zu minimieren, das oben auf den Rack Switches läuft, und eine standardisierte Implementierungspraxis zu definieren, um die Fehlerbehebung zu vereinfachen. Für Rechenzentren, in denen dies nicht erlaubt ist, aufgrund von Bedenken um die Rack-Stabilität, ist die umgekehrte sicherlich erlaubt, das Befüllen des Racks von unten nach oben.

InfiniBand-Storage-Netzwerkkonfiguration

Die Hälfte der InfiniBand-Ports an jedem Datei-Node werden für eine direkte Verbindung mit Block-Nodes verwendet. Die andere Hälfte ist mit den InfiniBand-Switches verbunden und wird für die BeeGFS-Client-Server-Konnektivität verwendet. Beim Bestimmen der Größe der IPoIB-Subnetze, die für BeeGFS-Clients und -Server verwendet werden, müssen Sie das erwartete Wachstum Ihres Compute/GPU-Clusters und BeeGFS-Dateisystems berücksichtigen. Wenn Sie von den empfohlenen IP-Bereichen abweichen müssen, beachten Sie, dass jede direkte Verbindung in einem einzelnen Baustein ein eigenes Subnetz hat und es keine Überschneidung mit Subnetzen gibt, die für die Client-Server-Konnektivität verwendet werden.

Direkte Verbindungen

Datei- und Block-Nodes innerhalb jedes Bausteins verwenden für ihre direkten Verbindungen immer die IPs in der folgenden Tabelle.

Dieses Adressprogramm entspricht der folgenden Regel: Das dritte Oktett ist immer ungerade oder gerade, was davon abhängt, ob der Datei-Node ungerade oder gerade ist.

Datei-Node	IB-Port	IP-Adresse	Block-Node	IB-Port	Physische IP-Adresse	Virtuelle IP
ODD (h1)	i1a	192.168.1.10	Ungerade (c1)	2 a	192.168.1.100	192.168.1.101
ODD (h1)	i2a	192.168.3.10	Ungerade (c1)	2 a	192.168.3.100	192.168.3.101
ODD (h1)	i3a	192.168.5.10	Gleichmäßig (c2)	2 a	192.168.5.100	192.168.5.101

Datei-Node	IB-Port	IP-Adresse	Block-Node	IB-Port	Physische IP-Adresse	Virtuelle IP
ODD (h1)	I4a	192.168.7.10	Gleichmäßig (c2)	2 a	192.168.7.100	192.168.7.101
Gleichmäßig (h2)	i1a	192.168.2.10	Ungerade (c1)	2b	192.168.2.100	192.168.2.101
Gleichmäßig (h2)	i2a	192.168.4.10	Ungerade (c1)	2b	192.168.4.100	192.168.4.101
Gleichmäßig (h2)	i3a	192.168.6.10	Gleichmäßig (c2)	2b	192.168.6.100	192.168.6.101
Gleichmäßig (h2)	I4a	192.168.8.10	Gleichmäßig (c2)	2b	192.168.8.100	192.168.8.101

IPoIB-Adressierungsschemata für BeeGFS-Client-Server

Auf jedem Datei-Node werden mehrere BeeGFS-Serverservices ausgeführt (Management, Metadaten oder Storage). Damit jeder Service unabhängig vom anderen Datei-Node ein Failover durchführen kann, verfügt jeder über eindeutige IP-Adressen, die zwischen beiden Nodes schweben können (auch als logische Schnittstelle oder LIF bezeichnet).

Diese Bereitstellung setzt zwar nicht zwingend voraus, dass für diese Verbindungen folgende IPoIB-Subnetzbereiche verwendet werden und definiert ein Standard-Adressierungsschema, das folgende Regeln anwendet:

- Das zweite Oktett ist immer ungerade oder sogar, basierend darauf, ob der InfiniBand-Port des Datei-Nodes ungerade oder sogar ungerade ist.
- BeeGFS Cluster-IPs sind immer `xxx. 127.100.yyy` Oder `xxx.128.100.yyy`.

> Zusätzlich zur Schnittstelle, die für die bandinterne Betriebssystemverwaltung verwendet wird, können zusätzliche Schnittstellen von Corosync für Cluster Heart-Schläge und Synchronisation verwendet werden. So wird sichergestellt, dass der Verlust einer einzelnen Schnittstelle das gesamte Cluster nicht in den Fall bringt.

- Der BeeGFS Management Service ist immer im Betrieb `xxx.yyy.101.0` Oder `xxx.yyy.102.0`.
- BeeGFS Metadatendienste sind immer dabei `xxx.yyy.101.zzz` Oder `xxx.yyy.102.zzz`.
- BeeGFS Storage-Services finden sich immer bei `xxx.yyy.103.zzz` oder `xxx.yyy.104.zzz`.
- Adressen im Bereich `100.xxx.1.1` Bis `100.xxx.99.255` Sind für Kunden reserviert.

IPoIB-Adressierungsschema für ein einzelnes Subnetz

In diesem Bereitstellungshandbuch wird ein einziges Subnetz-Schema verwendet, da die in aufgeführten Vorteile im aufgeführt "Softwarearchitektur"sind.

Subnetz: 100.127.0.0/16

Die folgende Tabelle enthält den Bereich für ein einzelnes Subnetz: 100.127.0.0/16.

Zweck	InfiniBand-Port	IP-Adresse oder Bereich
BeeGFS Cluster-IP	i1b oder i4b	100.127.100.1 - 100.127.100.255
BeeGFS Management	i1b	100.127.101.0
	i2b	100.127.102.0
BeeGFS-Metadaten	i1b oder i3b	100.127.101.1 - 100.127.101.255
	i2b oder i4b	100.127.102.1 - 100.127.102.255
BeeGFS-Speicherung	i1b oder i3b	100.127.103.1 - 100.127.103.255
	i2b oder i4b	100.127.104.1 - 100.127.104.255
BeeGFS-Clients	(Je nach Kunde)	100.127.1.1 - 100.127.99.255

IPoIB zwei Subnetz-Adressierungsschema

Ein zwei-Subnetz-Adressierungsschema wird nicht mehr empfohlen, kann aber trotzdem implementiert werden. In den folgenden Tabellen finden Sie ein empfohlenes zwei-Subnetz-Schema.

Subnetz A: 100.127.0.0/16

In der folgenden Tabelle ist der Bereich für Subnetz A angegeben: 100.127.0.0/16.

Zweck	InfiniBand-Port	IP-Adresse oder Bereich
BeeGFS Cluster-IP	i1b	100.127.100.1 - 100.127.100.255
BeeGFS Management	i1b	100.127.101.0
BeeGFS-Metadaten	i1b oder i3b	100.127.101.1 - 100.127.101.255
BeeGFS-Speicherung	i1b oder i3b	100.127.103.1 - 100.127.103.255
BeeGFS-Clients	(Je nach Kunde)	100.127.1.1 - 100.127.99.255

Subnetz B: 100.128.0.0/16

In der folgenden Tabelle ist der Bereich für Subnetz B angegeben: 100.128.0.0/16.

Zweck	InfiniBand-Port	IP-Adresse oder Bereich
BeeGFS Cluster-IP	I4b	100.128.100.1 - 100.128.100.255
BeeGFS Management	i2b	100.128.102.0
BeeGFS-Metadaten	i2b oder i4b	100.128.102.1 - 100.128.102.255
BeeGFS-Speicherung	i2b oder i4b	100.128.104.1 - 100.128.104.255
BeeGFS-Clients	(Je nach Kunde)	100.128.1.1 - 100.128.99.255

In dieser NetApp Verified Architecture werden nicht alle IPs in den oben genannten Bereichen verwendet. Sie zeigen, wie IP-Adressen vorzugewiesen werden können, um eine einfache Erweiterung des Dateisystems mit einem konsistenten IP-Adressierungschema zu ermöglichen. In diesem Schema entsprechen BeeGFS-Datei-Knoten und Service-IDs dem vierten Oktett eines bekannten IP-Bereichs. Das Filesystem konnte bei Bedarf auf jeden Fall über 255 Nodes oder Services skaliert werden.

Implementierung von Hardware

Jeder Baustein besteht aus zwei validierten x86-Datei-Nodes, die mithilfe von HDR-Kabeln (200 GB) direkt mit zwei Block-Nodes verbunden sind.

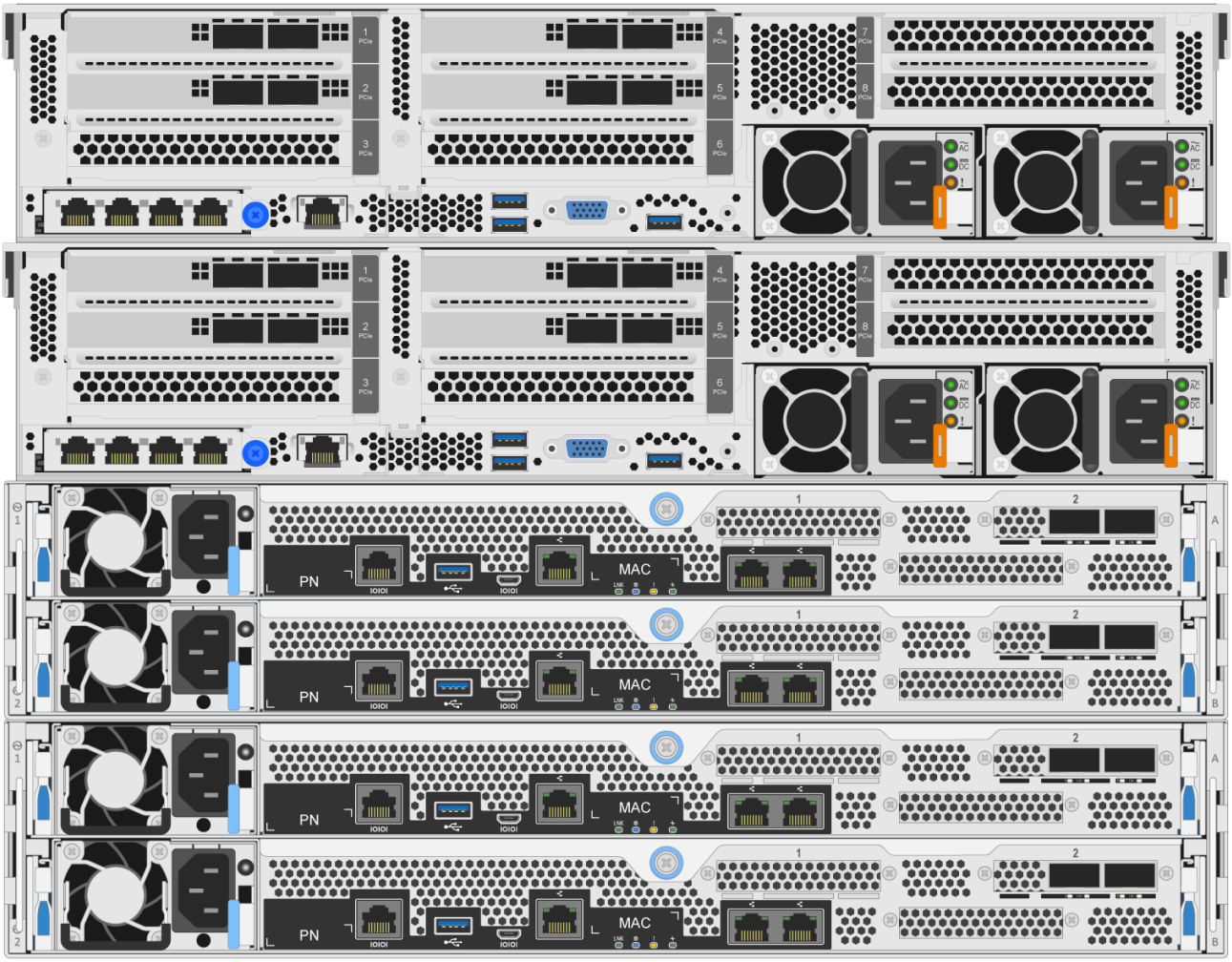
> Zum Herstellen von Quorum im Failover Cluster sind mindestens zwei Bausteine erforderlich. Ein Cluster mit zwei Nodes hat Einschränkungen, die ein erfolgreiches Failover verhindern können. Wenn Sie ein Cluster mit zwei Nodes konfigurieren, wird ein drittes Gerät als Tiebreaker integriert, dieses Design wird jedoch nicht in dieser Dokumentation beschrieben.

Die folgenden Schritte sind für jeden Baustein im Cluster identisch, unabhängig davon, ob er sowohl für die Ausführung von BeeGFS-Metadaten- und Storage-Services als auch nur für Storage-Services eingesetzt wird, sofern nicht anders angegeben.

Schritte

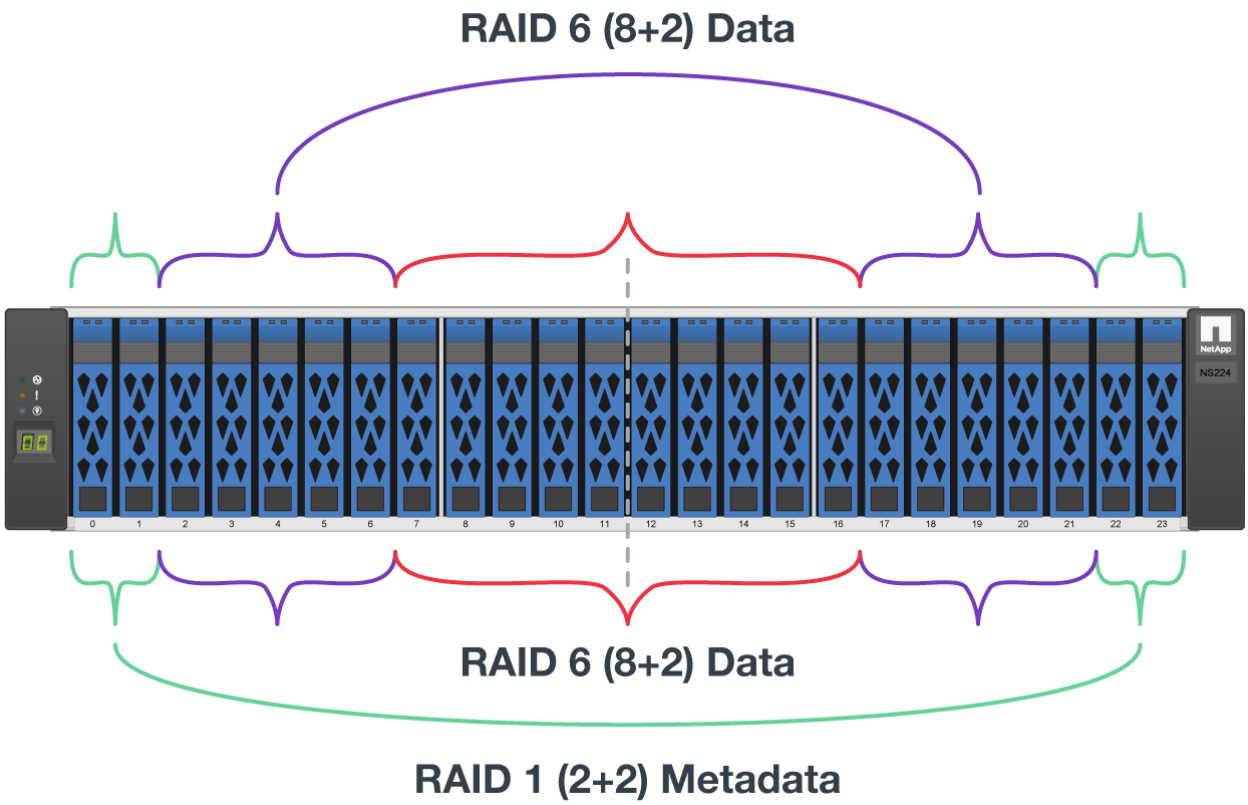
1. Richten Sie jeden BeeGFS-Dateiknoten mit vier Host-Channel-Adaptern (HCAs) mithilfe der in der angegebenen Modelle "Technische Anforderungen"ein. Legen Sie die HCAs gemäß den folgenden Spezifikationen in die PCIe-Steckplätze des Dateiknotens ein:
   - * Lenovo ThinkSystem SR665 V3 Server:* Verwenden Sie die PCIe-Steckplätze 1, 2, 4 und 5.
   - * Lenovo ThinkSystem SR665 Server:* Verwenden Sie die PCIe-Steckplätze 2, 3, 5 und 6.
2. Konfigurieren Sie jeden BeeGFS-Block-Node mit einer 200-GB-Host-Schnittstellenkarte (HIC) mit zwei Ports, und installieren Sie die HIC in jedem ihrer beiden Storage Controller.

   Stellen Sie die Bausteine so ein, dass die beiden BeeGFS-Datei-Nodes über den BeeGFS-Block-Nodes liegen. Die folgende Abbildung zeigt die richtige Hardwarekonfiguration für den BeeGFS-Baustein, der Lenovo ThinkSystem SR665 V3-Server als Dateiknoten verwendet (Rückansicht).

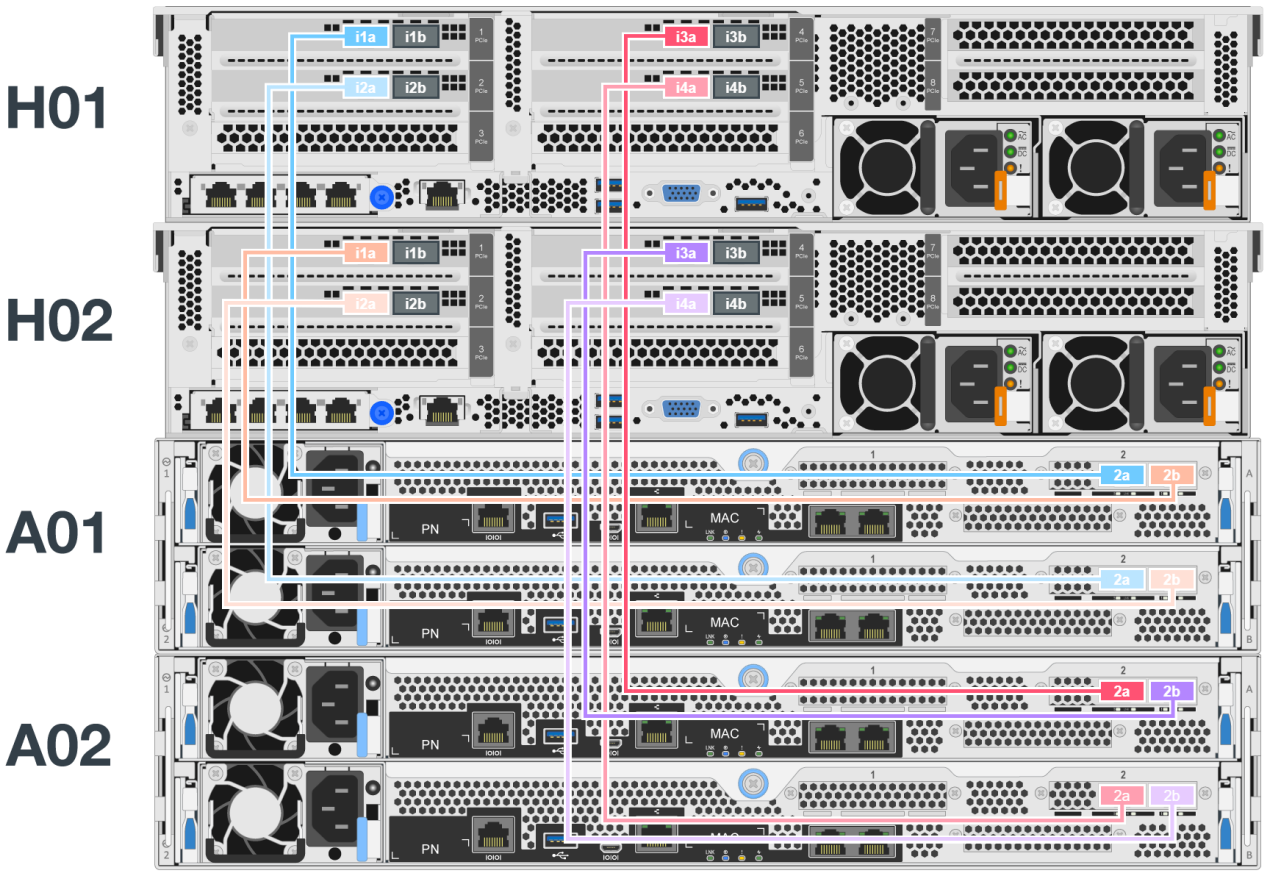


Die Konfiguration der Stromversorgung für Produktionsanwendungsfälle sollte in der Regel redundante Netzteile verwenden.

3. Installieren Sie bei Bedarf die Laufwerke in jedem BeeGFS-Block-Knoten.
   a. Wenn der Baustein zur Ausführung von BeeGFS-Metadaten und Speicherdiensten verwendet wird und kleinere Laufwerke für Metadaten-Volumes verwendet werden, vergewissern Sie sich, dass diese in den äußeren Laufwerksschächten gefüllt sind, wie in der Abbildung unten gezeigt.
   b. Wenn ein Laufwerkgehäuse nicht vollständig bestückt ist, stellen Sie bei allen Bausteinkonfigurationen sicher, dass eine gleiche Anzahl an Laufwerken in den Steckplätzen 0–11 und 12–23 gefüllt ist, um eine optimale Performance zu erzielen.



4. Verbinden Sie die Block- und File-Knoten mit dem "1 m InfiniBand HDR 200 GB Direct-Attach-Kupferkabel", so dass sie mit der in der folgenden Abbildung gezeigten Topologie übereinstimmen.

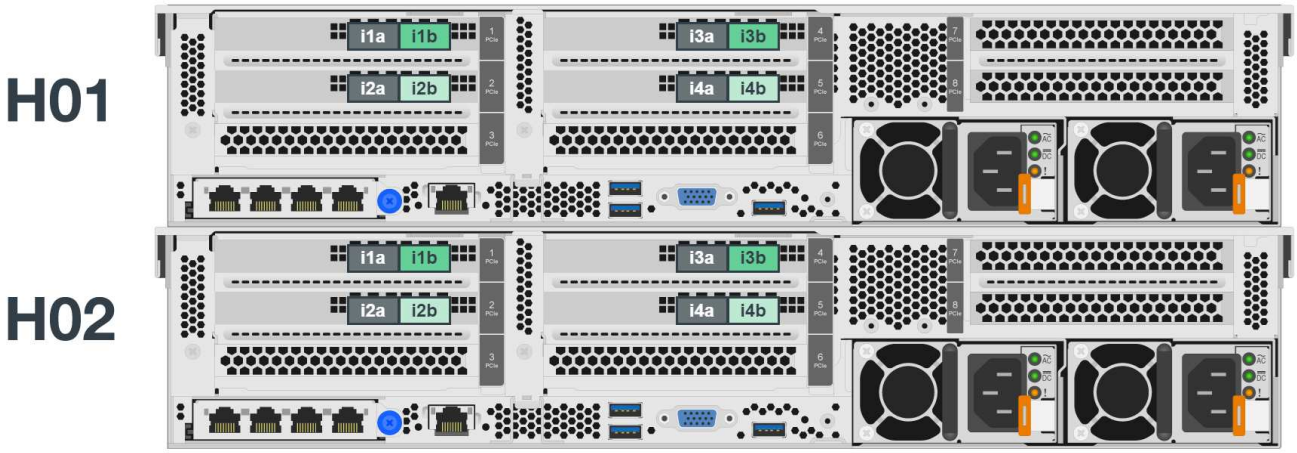


Die Nodes sind über mehrere Bausteine hinweg nie direkt miteinander verbunden. Jeder Baustein sollte als eigenständige Einheit behandelt werden und alle Kommunikation zwischen Bausteinen erfolgt über Netzwerk-Switches.

5. Verbinden Sie die übrigen InfiniBand-Ports auf dem Datei-Node mithilfe des spezifischen InfiniBand-Speicherswitters des Speichernetzwerks mit dem "2 m InfiniBand-Kabel" InfiniBand-Switch.

   Wenn Sie Splitterkabel verwenden, um den Speicher-Switch mit Dateiknoten zu verbinden, sollte ein Kabel vom Switch abzweigen und mit den hellgrünen Ports verbunden werden. Ein anderes Splitterkabel sollte sich vom Switch abzweigen und an die dunkelgrünen Ports anschließen.

   Außerdem sollten bei Speichernetzwerken mit redundanten Switches die hellgrünen Ports mit einem Switch verbunden werden, während dunkelgrüne Ports mit einem anderen Switch verbunden sein sollten.



6. Montieren Sie bei Bedarf weitere Bausteine gemäß den gleichen Verkabelungsrichtlinien.

Die Gesamtzahl der Bausteine, die in einem einzigen Rack implementiert werden können, hängt von der verfügbaren Stromversorgung und Kühlung an jedem Standort ab.

Implementierung von Software

Einrichten von Datei-Nodes und Block-Nodes

Während die meisten Software-Konfigurationsaufgaben mithilfe der von NetApp zur Verfügung gestellten Ansible Sammlungen automatisiert werden, müssen Sie das Netzwerk auf dem Baseboard Management Controller (BMC) jedes Servers konfigurieren und den Management-Port auf jedem Controller konfigurieren.

Richten Sie die Datei-Nodes ein

1. Konfigurieren Sie das Netzwerk auf dem Baseboard Management Controller (BMC) jedes Servers.

   Informationen zum Konfigurieren der Netzwerkkonfiguration für die validierten Lenovo SR665 V3-Dateiknoten finden Sie unter "Lenovo ThinkSystem Dokumentation".

Ein Baseboard Management Controller (BMC), der manchmal als Service-Prozessor bezeichnet wird, ist der generische Name für die Out-of-Band-Management-Funktion, die in verschiedenen Server-Plattformen integriert ist, die Remote-Zugriff bieten können, selbst wenn das Betriebssystem nicht installiert ist oder nicht zugänglich ist. Anbieter vermarkten diese Funktionalität in der Regel mit ihrem eigenen Branding. Auf dem Lenovo SR665 wird beispielsweise der BMC als *Lenovo XClarity Controller (XCC)* bezeichnet.

2. Konfigurieren Sie die Systemeinstellungen für maximale Performance.

   Sie konfigurieren die Systemeinstellungen über das UEFI-Setup (früher BIOS) oder über die Redfish APIs, die von vielen BMCs bereitgestellt werden. Die Systemeinstellungen variieren je nach Servermodell, das als Dateiknoten verwendet wird.

   Informationen zum Konfigurieren der Systemeinstellungen für die validierten Lenovo SR665 V3-Dateiknoten finden Sie unter "Passen Sie die Systemeinstellungen für die Performance an" .

3. Installieren Sie Red Hat Enterprise Linux (RHEL) 9.4 und konfigurieren Sie den Hostnamen und den Netzwerkport, die zur Verwaltung des Betriebssystems verwendet werden, einschließlich der SSH-Konnektivität vom Ansible-Steuerknoten.

   Konfigurieren Sie derzeit keine IPs auf einem der InfiniBand-Ports.

   Die nachfolgenden Abschnitte gehen davon aus, dass die Hostnamen sequenziell nummeriert sind (z. B. h1-HN), und beziehen sich auf Aufgaben, die auf ungeraden oder gar nummerierten Hosts ausgeführt werden sollten.

4. Verwenden Sie den Red Hat Subscription Manager, um das System zu registrieren und zu abonnieren, damit die erforderlichen Pakete aus den offiziellen Red Hat-Repositorys installiert werden können und um Updates auf die unterstützte Version von Red Hat zu beschränken: `subscription-manager release --set=9.4` . Anweisungen hierzu finden Sie unter "Registrieren und Abonnieren eines RHEL Systems" und "Einschränken von Aktualisierungen".
5. Aktivieren Sie das Red hat Repository mit den für hohe Verfügbarkeit erforderlichen Paketen.

   ```
   subscription-manager repo-override --repo=rhel-9-for-x86_64
   -highavailability-rpms --add=enabled:1
   ```

6. Aktualisieren Sie alle HCA-Firmware auf die in Verwendung des "Aktualisiert die Datei-Node-Adapter-Firmware"Handbuchs empfohlene Version"Technologieanforderungen erfüllt".

Richten Sie Block-Nodes ein

Richten Sie die EF600 Block-Nodes ein, indem Sie den Managementport pro Controller konfigurieren.

1. Konfigurieren Sie den Managementport an jedem EF600 Controller.

   Anweisungen zum Konfigurieren von Ports finden Sie im "E-Series Documentation Center".

2. Legen Sie optional den Speicher-Array-Namen für jedes System fest.

   Durch das Festlegen eines Namens kann es einfacher sein, in den nachfolgenden Abschnitten auf jedes System zu verweisen. Anweisungen zum Festlegen des Arraynamens finden Sie im "E-Series Documentation Center".

Folgende Themen setzen voraus, dass die Namen der Speicherarrays nacheinander nummeriert sind (z. B. c1 - CN), und beziehen sich auf die Schritte, die auf ungeraden und nicht gerade nummerierten Systemen ausgeführt werden sollten.

Optimieren Sie die System-Einstellungen des File Node für die Performance

Um die Leistung zu maximieren, empfehlen wir, die Systemeinstellungen auf dem Servermodell zu konfigurieren, das Sie als Dateiknoten verwenden.

Die Systemeinstellungen variieren je nach Servermodell, das Sie als Dateiknoten verwenden. In diesem Thema wird beschrieben, wie die Systemeinstellungen für die validierten Lenovo ThinkSystem SR665-Serverdateiknoten konfiguriert werden.

Über die UEFI-Schnittstelle können Sie die Systemeinstellungen anpassen

Die System-Firmware des Lenovo SR665 V3-Servers enthält zahlreiche Tuning-Parameter, die über die UEFI-Schnittstelle eingestellt werden können. Diese Optimierungsparameter können sich auf alle Aspekte der Serverfunktionen und die Leistung des Servers auswirken.

Passen Sie unter **UEFI Setup > Systemeinstellungen** die folgenden Systemeinstellungen an:

Menü „Betriebsmodus“

Systemeinstellung	Wechseln Sie zu
Betriebsmodus	Individuell
CTDP	Manuell
CTDP-Handbuch	350
Maximale Leistung Des Pakets	Manuell
Effizienzmodus	Deaktivieren
Global-Cstate-Control	Deaktivieren
SOC P-Staaten	P0
DF-C-Staaten	Deaktivieren
P-Zustand	Deaktivieren
Speicherabschaltstrom Aktivieren	Deaktivieren
NUMA-Knoten pro Socket	NPS1

Menü „Geräte“ und „E/A-Anschlüsse“

Systemeinstellung	Wechseln Sie zu
IOMMU	Deaktivieren

Ein-/aus-Menü

Systemeinstellung	Wechseln Sie zu
PCIe-Power-Brake	Deaktivieren

Menü „Prozessoren“

Systemeinstellung	Wechseln Sie zu
Global C-State Control	Deaktivieren
DF-C-Staaten	Deaktivieren
SMT-Modus	Deaktivieren
CPPC	Deaktivieren

Verwenden Sie die Redfish-API, um die Systemeinstellungen anzupassen

Zusätzlich zur Verwendung von UEFI Setup können Sie die Redfish API verwenden, um Systemeinstellungen zu ändern.

```
curl --request PATCH \
  --url https://<BMC_IP_ADDRESS>/redfish/v1/Systems/1/Bios/Pending \
  --user <BMC_USER>:<BMC- PASSWORD> \
  --header 'Content-Type: application/json' \
  --data '{
"Attributes": {
"OperatingModes_ChooseOperatingMode": "CustomMode",
"Processors_cTDP": "Manual",
"Processors_PackagePowerLimit": "Manual",
"Power_EfficiencyMode": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_SOCP_states": "P0",
"Processors_DFC_States": "Disable",
"Processors_P_State": "Disable",
"Memory_MemoryPowerDownEnable": "Disable",
"DevicesandIOPorts_IOMMU": "Disable",
"Power_PCIePowerBrake": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_DFC_States": "Disable",
"Processors_SMTMode": "Disable",
"Processors_CPPC": "Disable",
"Memory_NUMANodesperSocket":"NPS1"
}
}
'
```

Ausführliche Informationen zum Schema Redfish finden Sie im "DMTF-Website".

Richten Sie einen Ansible-Steuerungsknoten ein

Zum Einrichten eines Ansible-Steuerknotens müssen Sie eine virtuelle oder physische Maschine mit Netzwerkzugriff auf alle Datei- und Block-Nodes zuweisen, die für die BeeGFS auf NetApp Lösung implementiert werden.

Eine Liste der empfohlenen Paketversionen finden Sie im"Technische Anforderungen". Die folgenden Schritte wurden auf Ubuntu 22.04 getestet. Für spezifische Schritte zu Ihrer bevorzugten Linux-Distribution, siehe "Ansible-Dokumentation".

1. Installieren Sie über Ihren Ansible Control Node die folgenden Pakete für Python und Python Virtual Environment.

   ```
   sudo apt-get install python3 python3-pip python3-setuptools python3.10-venv
   ```

2. Erstellen Sie eine virtuelle Python-Umgebung.

```
python3 -m venv ~/pyenv
```

3. Aktivieren Sie die virtuelle Umgebung.

```
source ~/pyenv/bin/activate
```

4. Installieren Sie die erforderlichen Python-Pakete in der aktivierten virtuellen Umgebung.

```
pip install ansible netaddr cryptography passlib
```

5. Installieren Sie die BeeGFS-Sammlung mit Ansible Galaxy.

```
ansible-galaxy collection install netapp_eseries.beegfs
```

6. Überprüfen Sie, ob die installierten Versionen von Ansible, Python und BeeGFS-Sammlung mit der übereinstimmen"Technische Anforderungen".

```
ansible --version
ansible-galaxy collection list netapp_eseries.beegfs
```

7. Richten Sie passwortloses SSH ein, damit Ansible vom Ansible-Steuerungsknoten aus auf die Remote-BeeGFS-Datei-Nodes zugreifen kann.
   a. Generieren Sie auf dem Ansible-Steuerungsknoten, falls erforderlich, ein Paar öffentlicher Schlüssel.

```
ssh-keygen
```

   b. Richten Sie passwortloses SSH für jeden der Dateiknoten ein.

```
ssh-copy-id <ip_or_hostname>
```

Richten Sie bei den Blockknoten eine passwortlose SSH ein. Dies wird weder unterstützt noch erforderlich.

Erstellen des Ansible-Inventars

Um die Konfiguration für Datei- und Block-Nodes zu definieren, erstellen Sie einen Ansible-Bestand für das BeeGFS-Dateisystem, das bereitgestellt werden soll. Der Bestand umfasst Hosts, Gruppen und Variablen, die das gewünschte BeeGFS-Dateisystem beschreiben.

Schritt 1: Konfiguration für alle Bausteine definieren

Legen Sie die Konfiguration fest, die für alle Bausteine gilt, unabhängig davon, welches Konfigurationsprofil Sie für sie einzeln anwenden können.

Bevor Sie beginnen

- Wählen Sie ein Subnetz-Adressierungsschema für Ihre Bereitstellung aus. Aufgrund der im aufgeführten Vorteile "Softwarearchitektur"wird empfohlen, ein einziges Subnetz-Adressierungsschema zu verwenden.

Schritte

1. Geben Sie auf dem Ansible-Steuerungsknoten ein Verzeichnis an, das Sie zum Speichern der Bestands- und Playbook-Dateien in Ansible verwenden möchten.

   Sofern nicht anders angegeben, werden alle in diesem Schritt erstellten Dateien und Verzeichnisse und die folgenden Schritte relativ zu diesem Verzeichnis erstellt.

2. Folgende Unterverzeichnisse erstellen:

   `host_vars`

   `group_vars`

   `packages`

3. Erstellen Sie ein Unterverzeichnis für Cluster-Passwörter und sichern Sie die Datei durch Verschlüsselung mit Ansible Vault (siehe "Verschlüsseln von Inhalten mit Ansible Vault"):

   a. Erstellen Sie das Unterverzeichnis `group_vars/all`.

   b. Erstellen Sie im `group_vars/all` Verzeichnis eine Passwortdatei mitder Bezeichnung `passwords.yml`.

   c. Füllen Sie den `passwords.yml file` mit den folgenden Angaben aus, und ersetzen Sie alle Benutzernamen- und Kennwortparameter entsprechend Ihrer Konfiguration:

```
# Credentials for storage system's admin password
eseries_password: <PASSWORD>

# Credentials for BeeGFS file nodes
ssh_ha_user: <USERNAME>
ssh_ha_become_pass: <PASSWORD>

# Credentials for HA cluster
ha_cluster_username: <USERNAME>
ha_cluster_password: <PASSWORD>
ha_cluster_password_sha512_salt: randomSalt

# Credentials for fencing agents
# OPTION 1: If using APC Power Distribution Units (PDUs) for fencing:
# Credentials for APC PDUs.
apc_username: <USERNAME>
apc_password: <PASSWORD>

# OPTION 2: If using the Redfish APIs provided by the Lenovo XCC (and
other BMCs) for fencing:
# Credentials for XCC/BMC of BeeGFS file nodes
bmc_username: <USERNAME>
bmc_password: <PASSWORD>
```

d. Führen Sie aus `ansible-vault encrypt passwords.yml`, und legen Sie ein Vault-Kennwort fest, wenn Sie dazu aufgefordert werden.

Schritt: Konfiguration für einzelne Datei- und Block-Nodes definieren

Legen Sie die Konfiguration für einzelne Datei-Nodes und einzelne Baustein-Nodes fest.

1. Unter `host_vars/`Erstellen Sie für jeden BeeGFS-Dateiknoten eine Datei mit dem Namen `<HOSTNAME>.yml` Mit dem folgenden Inhalt, besondere Aufmerksamkeit auf die Notizen über den Inhalt für BeeGFS Cluster-IPs und Host-Namen enden in ungerade oder gerade Zahlen.

   Zunächst stimmen die Schnittstellennamen der Dateiknoten mit dem überein, was hier aufgeführt ist (z. B. ib0 oder ibs1f0). Diese benutzerdefinierten Namen werden in konfiguriert die für alle Datei-Knoten gelten soll.

```
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: 100.127.100. <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
```

Wenn Sie bereits das BeeGFS-Cluster implementiert haben, müssen Sie das Cluster beenden, bevor Sie statisch konfigurierte IP-Adressen hinzufügen oder ändern, einschließlich Cluster-IPs und IPs für NVMe/IB. Dies ist erforderlich, damit diese Änderungen ordnungsgemäß wirksam werden und Cluster-Vorgänge nicht unterbrechen.

2. Unter `host_vars/`Erstellen Sie für jeden BeeGFS-Block-Knoten eine Datei mit dem Namen `<HOSTNAME>.yml` Und geben Sie den folgenden Inhalt ein.

   Achten Sie besonders auf die Hinweise zum Inhalt, die für Speicher-Array-Namen ausgefüllt werden müssen, die mit ungeraden oder geraden Zahlen enden.

   Erstellen Sie für jeden Block-Node eine Datei, und geben Sie den an `<MANAGEMENT_IP>` Für einen der beiden Controller (normalerweise 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
```

Schritt 3: Definieren Sie die Konfiguration, die für alle Datei- und Block-Nodes gelten soll

Unter können Sie die gemeinsame Konfiguration für eine Gruppe von Hosts definieren `group_vars` In einem Dateinamen, der der Gruppe entspricht. Dadurch wird verhindert, dass eine gemeinsame Konfiguration an mehreren Orten wiederholt wird.

Über diese Aufgabe

Hosts können sich in mehr als einer Gruppe befinden. Ansible zur Laufzeit wählt Ansible aus, welche Variablen auf Basis seiner variablen Rangfolge für einen bestimmten Host gelten. (Weitere Informationen zu diesen Regeln finden Sie in der Ansible-Dokumentation für "Variablen verwenden".)

Host-zu-Gruppe-Zuweisungen werden in der tatsächlichen Ansible-Bestandsdatei definiert, die gegen Ende dieses Vorgangs erstellt wird.

Schritt

In Ansible können alle Konfigurationen, die auf alle Hosts angewendet werden sollen, in einer Gruppe mit dem Namen definiert werden `All`. Erstellen Sie die Datei `group_vars/all.yml` Mit folgenden Inhalten:

```
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"
```

Schritt 4: Definieren Sie die Konfiguration, die für alle Datei-Knoten gelten soll

Die gemeinsame Konfiguration für Dateiknoten ist in einer Gruppe mit dem Namen definiert `ha_cluster`. In den Schritten in diesem Abschnitt wird die Konfiguration erstellt, die in der enthalten sein sollte `group_vars/ha_cluster.yml` Datei:

Schritte

1. Legen Sie oben in der Datei die Standardeinstellungen fest, einschließlich des Kennworts, das als verwendet werden soll `sudo` Benutzer auf den Datei-Nodes.

```
### ha_cluster Ansible group inventory file.
# Place all default/common variables for BeeGFS HA cluster resources
below.
### Cluster node defaults
ansible_ssh_user: {{ ssh_ha_user }}
ansible_become_password: {{ ssh_ha_become_pass }}
eseries_ipoib_default_hook_templates:
  - 99-multihoming.j2   # This is required for single subnet
deployments, where static IPs containing multiple IB ports are in the
same IPoIB subnet. i.e: cluster IPs, multirail, single subnet, etc.
# 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 status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
eseries_ib_opensm_options:
  virt_enabled: "2"
  virt_max_ports_in_process: "0"
```

> Wenn der `ansible_ssh_user` bereits ist `root`, können Sie optional die auslassen und beim Ausführen des Playbook die `ansible_become_password` Option angeben `--ask -become-pass`.

2. Konfigurieren Sie optional einen Namen für den Hochverfügbarkeits-Cluster und geben Sie einen Benutzer für die Cluster-interne Kommunikation an.

   Wenn Sie das private IP-Adressschema ändern, müssen Sie auch die Standardeinstellung aktualisieren `beegfs_ha_mgmtd_floating_ip`. Dies muss mit dem übereinstimmen, was Sie später für die BeeGFS Management Ressourcengruppe konfigurieren.

   Geben Sie eine oder mehrere E-Mails an, die Warnmeldungen für Cluster-Ereignisse mit empfangen sollen `beegfs_ha_alert_email_list`.

```
### 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: "{{ ha_cluster_username }}" # Parameter for
BeeGFS HA cluster username in the passwords file.
beegfs_ha_cluster_password: "{{ ha_cluster_password }}" # Parameter for
BeeGFS HA cluster username's password in the passwords file.
beegfs_ha_cluster_password_sha512_salt: "{{
ha_cluster_password_sha512_salt }}" # Parameter for BeeGFS HA cluster
username's password salt in the passwords file.
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
```

Während scheinbar redundant, `beegfs_ha_mgmtd_floating_ip` Ist wichtig, wenn Sie das BeeGFS-Dateisystem über einen einzelnen HA-Cluster hinaus skalieren. Nachfolgende HA-Cluster werden ohne zusätzlichen BeeGFS-Managementservice bereitgestellt und Punkt am Managementservice des ersten Clusters.

3. Konfigurieren Sie einen Fechtagenten. (Weitere Informationen finden Sie unter "Konfigurieren Sie Fechten in einem Red hat High Availability Cluster".) Die folgende Ausgabe zeigt Beispiele für die Konfiguration gängiger Fencing-Agenten. Wählen Sie eine dieser Optionen.

   Beachten Sie bei diesem Schritt Folgendes:

- Standardmäßig ist Fechten aktiviert, Sie müssen jedoch einen Fechten_Agent_ konfigurieren.
- Der `<HOSTNAME>` Angegeben in `pcmk_host_map` Oder `pcmk_host_list` Der Hostname in der Ansible-Bestandsaufnahme entspricht.
- Das BeeGFS-Cluster ohne Fencing wird insbesondere in der Produktion nicht unterstützt. Dies soll weitgehend sicherstellen, wenn BeeGFS-Services, einschließlich aller Ressourcenabhängigkeiten wie Blockgeräte, Failover aufgrund eines Problems durchführen, es besteht keine Möglichkeit des gleichzeitigen Zugriffs durch mehrere Nodes, die zu einer Beschädigung des Filesystems oder anderen unerwünschten oder unerwarteten Verhalten führen. Wenn das Fechten deaktiviert werden muss, lesen Sie die allgemeinen Hinweise in der BeeGFS HA-Rolle „erste Schritte“-Anleitung und „Set“ `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` Mit FALSE innen `ha_cluster.yml`.
- Es sind mehrere Fechtgeräte auf Node-Ebene verfügbar, und die BeeGFS HA-Rolle kann jeden Fechtagenten konfigurieren, der im Red hat HA Package Repository verfügbar ist. Wenn möglich, verwenden Sie einen Zaunsagenten, der über die unterbrechungsfreie Stromversorgung (USV) oder die Rack-Stromverteilereinheit (rPDU) arbeitet. Da einige Fechten-Agenten wie der Baseboard-Management-Controller (BMC) oder andere Lights-Out-Geräte, die in den Server integriert sind, möglicherweise nicht auf die Zaunanforderung unter bestimmten Ausfallszenarien reagieren.

```
### Fencing configuration:
# OPTION 1: To enable fencing using APC Power Distribution Units
(PDUs):
beegfs_ha_fencing_agents:
 fence_apc:
   - ipaddr: <PDU_IP_ADDRESS>
     login: "{{ apc_username }}" # Parameter for APC PDU username in
the passwords file.
     passwd: "{{ apc_password }}" # Parameter for APC PDU password in
the passwords file.
     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 }}" # Parameter for XCC/BMC username in
the passwords file.
  password: "{{ bmc_password }}" # Parameter for XCC/BMC password in
the passwords file.
    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/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Aktivieren Sie die empfohlene Performance-Optimierung im Linux-Betriebssystem.

   Viele Benutzer finden die Standardeinstellungen für die Performance-Parameter zwar im Allgemeinen gut, Sie können jedoch optional die Standardeinstellungen für einen bestimmten Workload ändern. Daher sind diese Empfehlungen in die BeeGFS-Rolle enthalten, jedoch sind sie nicht standardmäßig aktiviert, um sicherzustellen, dass Benutzer die auf ihr Dateisystem angewendete Einstellung kennen.

   Um das Performance-Tuning zu aktivieren, geben Sie Folgendes an:

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

5. (Optional) Sie können die Leistungsparameter im Linux-Betriebssystem nach Bedarf anpassen.

   Eine umfassende Liste der verfügbaren Tuning-Parameter, die Sie anpassen können, finden Sie im Abschnitt Performance Tuning Defaults der BeeGFS HA-Rolle in "E-Series BeeGFS GitHub-Website". Die Standardwerte können für alle Knoten im Cluster in dieser Datei oder für die Datei eines einzelnen Knotens überschrieben werden `host_vars` .

6. Um vollständige 200 GB/HDR-Konnektivität zwischen Block- und Dateiknoten zu ermöglichen, verwenden Sie das OpenSM-Paket (Open Subnetz Manager) aus der NVIDIA Open Fabrics Enterprise Distribution (MLNX_OFED). Die MLNX_OFED-Version in der Liste wird im Lieferumfang der "Anforderungen an den Datei-Node" empfohlenen OpenSM-Pakete enthalten. Obwohl die Implementierung mit Ansible unterstützt wird, müssen Sie zuerst den MLNX_OFED-Treiber auf allen Datei-Nodes installieren.

   a. Füllen Sie die folgenden Parameter in aus `group_vars/ha_cluster.yml` (Passen Sie Pakete nach Bedarf an):

      ```
      ### OpenSM package and configuration information
      eseries_ib_opensm_options:
        virt_enabled: "2"
        virt_max_ports_in_process: "0"
      ```

7. Konfigurieren Sie die `udev` Regel zur Sicherstellung einer konsistenten Zuordnung von logischen InfiniBand-Port-IDs zu zugrunde liegenden PCIe-Geräten.

   Der `udev` Die Regel muss für die PCIe-Topologie jeder Serverplattform, die als BeeGFS-Datei-Node verwendet wird, eindeutig sein.

   Für verifizierte Dateiknoten folgende Werte verwenden:

```
### Ensure Consistent Logical IB Port Numbering
# OPTION 1: Lenovo SR665 V3 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:01:00.0": i1a
  "0000:01:00.1": i1b
  "0000:41:00.0": i2a
  "0000:41:00.1": i2b
  "0000:81:00.0": i3a
  "0000:81:00.1": i3b
  "0000:a1:00.0": i4a
  "0000:a1:00.1": i4b

# OPTION 2: 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
```

8. (Optional) Aktualisieren des Metadaten-Zielauswahlalgorithmus.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

Während der Verifizierungstests `randomrobin` Wurde in der Regel verwendet, um sicherzustellen, dass Testdateien während des Performance-Benchmarking gleichmäßig auf alle BeeGFS-Speicherziele verteilt wurden (weitere Informationen zu Benchmarking finden Sie auf der BeeGFS-Website für "Benchmarking eines BeeGFS-Systems"). Bei der realen Welt könnte dies dazu führen, dass sich die niedrigeren nummerierten Ziele schneller füllen als die höher nummerierten Ziele. Auslassung `randomrobin` Und nur mit dem Standard `randomized` Der Wert zeigt sich, dass er eine gute Leistung bietet und gleichzeitig alle verfügbaren Ziele nutzt.

Schritt 5: Definieren Sie die Konfiguration für den gemeinsamen Block-Node

Die gemeinsame Konfiguration für Block-Knoten wird in einer Gruppe mit dem Namen definiert `eseries_storage_systems`. In den Schritten in diesem Abschnitt wird die Konfiguration erstellt, die in der enthalten sein sollte `group_vars/ eseries_storage_systems.yml` Datei:

Schritte

1. Setzen Sie die Ansible-Verbindung auf Local, geben Sie das Systemkennwort ein und geben Sie an, ob

SSL-Zertifikate verifiziert werden sollen. (Normalerweise verwendet Ansible SSH für die Verbindung zu gemanagten Hosts. Bei Storage-Systemen der NetApp E-Series, die als Block-Nodes verwendet werden, verwenden die Module JEDOCH die REST-API für die Kommunikation.) Fügen Sie oben in der Datei Folgendes hinzu:

```
### 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: {{ eseries_password }} # Parameter for E-Series
storage array password in the passwords file.
eseries_validate_certs: false
```

2. Installieren Sie die für Block-Nodes in aufgeführten Versionen, um eine optimale Performance zu gewährleisten "Technische Anforderungen".

   Laden Sie die entsprechenden Dateien aus dem herunter "NetApp Support Website". Sie können sie entweder manuell aktualisieren oder sie in das einbeziehen `packages/` Verzeichnis des Ansible-Steuerungsknotens, und füllen Sie dann die folgenden Parameter in aus `eseries_storage_systems.yml` So führen Sie ein Upgrade mit Ansible durch:

   ```
   # Firmware, NVSRAM, and Drive Firmware (modify the filenames as needed):
   eseries_firmware_firmware: "packages/RCB_11.80GA_6000_64cc0ee3.dlp"
   eseries_firmware_nvsram: "packages/N6000-880834-D08.dlp"
   ```

3. Laden Sie die neueste Laufwerksfirmware herunter, die für die in Ihren Blockknoten installierten Laufwerke verfügbar ist, und installieren Sie sie im "NetApp Support Website". Sie können sie entweder manuell aktualisieren oder in das Verzeichnis des Ansible-Steuerknotens aufnehmen `packages/` . Dann füllen Sie die folgenden Parameter in aus `eseries_storage_systems.yml` , um das Upgrade mit Ansible auszuführen:

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

   > Einstellung `eseries_drive_firmware_upgrade_drives_online` Bis `false` Beschleunigt das Upgrade, sollte aber erst nach dem Einsatz von BeeGFS durchgeführt werden. Der Grund dafür ist, dass bei dieser Einstellung sämtliche I/O-Vorgänge auf den Laufwerken vor dem Upgrade angehalten werden müssen, um Applikationsfehler zu vermeiden. Obwohl ein Online-Laufwerk-Firmware-Upgrade vor der Konfiguration von Volumes noch schnell durchgeführt wird, empfehlen wir Ihnen, diesen Wert immer auf zu setzen `true` Um später Probleme zu vermeiden.

4. Nehmen Sie zur Optimierung der Leistung folgende Änderungen an der globalen Konfiguration vor:

```
# 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. Um eine optimale Bereitstellung und ein optimales Verhalten von Volumes zu gewährleisten, geben Sie folgende Parameter an:

```
# 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"
```

> ⓘ Der für angegebene Wert `eseries_storage_pool_usable_drives` Gibt einen spezifischen Block-Node der NetApp EF600 an und steuert die Reihenfolge, in der Laufwerke neuen Volume-Gruppen zugewiesen werden. Durch diese Bestellung wird sichergestellt, dass der I/O zu jeder Gruppe gleichmäßig über die Kanäle des Backend-Laufwerks verteilt wird.

Definieren Sie den Ansible-Bestand für BeeGFS-Bausteine

Definieren Sie nach der Definition der allgemeinen Ansible-Bestandsstruktur die Konfiguration für jeden Baustein im BeeGFS-Dateisystem.

Diese Implementierungsanleitungen zeigen, wie Sie ein Filesystem implementieren, das aus einem Grundbaustein besteht, einschließlich Management-, Metadaten- und Storage-Services, einem zweiten Baustein mit Metadaten und Storage-Services und einem dritten Baustein nur für Storage.

Diese Schritte sollen den gesamten Bereich typischer Konfigurationsprofile anzeigen, mit denen Sie NetApp BeeGFS-Bausteine konfigurieren können, um die Anforderungen des gesamten BeeGFS-Dateisystems zu erfüllen.

> ⓘ Passen Sie in diesen und nachfolgenden Abschnitten nach Bedarf an, um den Bestand zu erstellen, der das BeeGFS-Dateisystem darstellt, das Sie bereitstellen möchten. Verwenden Sie insbesondere Ansible-Hostnamen, die jeden Block- oder Datei-Node darstellen, und das gewünschte IP-Adressschema für das Storage-Netzwerk, um sicherzustellen, dass es auf die Anzahl der BeeGFS-Datei-Nodes und -Clients skaliert werden kann.

Schritt: Die Ansible-Bestandsdatei erstellen

Schritte

1. Erstellen Sie eine neue `inventory.yml` Datei, und fügen Sie dann die folgenden Parameter ein, ersetzen Sie die Hosts unter `eseries_storage_systems` Nach Bedarf zur Darstellung der Block-Nodes in Ihrer Implementierung. Die Namen sollten dem Namen entsprechen, für den sie verwendet werden `host_vars/<FILENAME>.yml`.

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp_01:
        netapp_02:
        netapp_03:
        netapp_04:
        netapp_05:
        netapp_06:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
```

   In den nachfolgenden Abschnitten werden unter weitere Ansible-Gruppen erstellt `ha_cluster` Die die BeeGFS-Dienste darstellen, die im Cluster ausgeführt werden sollen.

Schritt: Inventar für einen Management-, Metadaten- und Storage-Baustein konfigurieren

Der erste Baustein im Cluster- oder Basis-Baustein muss den BeeGFS-Managementservice sowie Metadaten- und Storage-Services umfassen:

Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter `ha_cluster: children`:

```
      # beegfs_01/beegfs_02 HA Pair (mgmt/meta/storage building block):
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_01:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_01:
          hosts:
```

```
        beegfs_01:
        beegfs_02:
    meta_02:
      hosts:
        beegfs_01:
        beegfs_02:
    stor_02:
      hosts:
        beegfs_01:
        beegfs_02:
    meta_03:
      hosts:
        beegfs_01:
        beegfs_02:
    stor_03:
      hosts:
        beegfs_01:
        beegfs_02:
    meta_04:
      hosts:
        beegfs_01:
        beegfs_02:
    stor_04:
      hosts:
        beegfs_01:
        beegfs_02:
    meta_05:
      hosts:
        beegfs_02:
        beegfs_01:
    stor_05:
      hosts:
        beegfs_02:
        beegfs_01:
    meta_06:
      hosts:
        beegfs_02:
        beegfs_01:
    stor_06:
      hosts:
        beegfs_02:
        beegfs_01:
    meta_07:
      hosts:
        beegfs_02:
        beegfs_01:
```

```
        stor_07:
          hosts:
            beegfs_02:
            beegfs_01:
        meta_08:
          hosts:
            beegfs_02:
            beegfs_01:
        stor_08:
          hosts:
            beegfs_02:
            beegfs_01:
```

2. Erstellen Sie die Datei `group_vars/mgmt.yml` Und geben Sie Folgendes an:

```
# 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: 100.127.102.0/16
beegfs_service: management
beegfs_targets:
  netapp_01:
    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. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `meta_01` Bis `meta_08` Verwenden Sie die folgende Vorlage und füllen Sie dann die Platzhalterwerte für jeden Service aus, indem Sie auf die folgende Tabelle verweisen:

```
# 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>
```

Die Volume-Größe wird als Prozentsatz des gesamten Storage-Pools angegeben (auch als Volume-Gruppe bezeichnet). NetApp empfiehlt, freie Kapazitäten in jedem Pool zu belassen, um Platz für die SSD-Überprovisionierung zu haben (weitere Informationen finden Sie unter "Einführung in das NetApp EF600 Array"). Der Storage-Pool, `beegfs_m1_m2_m5_m6`, Weist auch 1% der Kapazität des Pools für den Management-Service. Somit für Metadaten-Volumes im Storage-Pool `beegfs_m1_m2_m5_m6`, Wenn 1,92-TB- oder 3,84-TB-Laufwerke verwendet werden, setzen Sie diesen Wert auf `21.25`; Für 7,5-TB-Laufwerke setzen Sie diesen Wert auf `22.25`; Und für 15,3-TB-Laufwerke ist dieser Wert auf festgelegt `23.75`.

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
meta_01.yml	8015	i1b:100.127.101.1/16 i2b:100.127.102.1/16	0	netapp_01	Beegfs_m1_m2_m5_m6	A
meta_02.yml	8025	i2b:100.127.102.2/16 i1b:100.127.101.2/16	0	netapp_01	Beegfs_m1_m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.101.3/16 i4b:100.127.102.3/16	1	netapp_02	Beegfs_m3_m4_m7_m8	A

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
meta_04.yml	8045	l4b:100.127.102.4/16 i3b:100.127.101.4/16	1	netapp_02	Beegfs_m3_m4_m7_m8	B
meta_05.yml	8055	i1b:100.127.101.5/16 i2b:100.127.102.5/16	0	netapp_01	Beegfs_m1_m2_m5_m6	A
meta_06.yml	8065	i2b:100.127.102.6/16 i1b:100.127.101.6/16	0	netapp_01	Beegfs_m1_m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.101.7/16 i4b:100.127.102.7/16	1	netapp_02	Beegfs_m3_m4_m7_m8	A
meta_08.yml	8085	l4b:100.127.102.8/16 i3b:100.127.101.8/16	1	netapp_02	Beegfs_m3_m4_m7_m8	B

4. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_01` Bis `stor_08` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```

Informationen zur richtigen Größe finden Sie unter "Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools".

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_01.yml	8013	i1b:100.127.103.1/16 i2b:100.127.104.1/16	0	netapp_01	Beegfs_s1_s2	A
stor_02.yml	8023	i2b:100.127.104.2/16 i1b:100.127.103.2/16	0	netapp_01	Beegfs_s1_s2	B
stor_03.yml	8033	i3b:100.127.103.3/16 i4b:100.127.104.3/16	1	netapp_02	Beegfs_s3_s4	A
stor_04.yml	8043	l4b:100.127.104.4/16 i3b:100.127.103.4/16	1	netapp_02	Beegfs_s3_s4	B

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_05.yml	8053	i1b:100.127.103.5/16 i2b:100.127.104.5/16	0	netapp_01	Beegfs_s5_s6	A
stor_06.yml	8063	i2b:100.127.104.6/16 i1b:100.127.103.6/16	0	netapp_01	Beegfs_s5_s6	B
stor_07.yml	8073	i3b:100.127.103.7/16 i4b:100.127.104.7/16	1	netapp_02	Beegfs_s7_s8	A
stor_08.yml	8083	l4b:100.127.104.8/16 i3b:100.127.103.8/16	1	netapp_02	Beegfs_s7_s8	B

Schritt 3: Konfigurieren Sie den Bestand für einen Baustein Metadaten + Speicher

In diesen Schritten wird beschrieben, wie ein Ansible-Inventar für BeeGFS-Metadaten + Storage-Baustein eingerichtet wird.

Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter der vorhandenen Konfiguration:

```
        meta_09:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_09:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_10:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_10:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_11:
          hosts:
            beegfs_03:
```

```
        beegfs_04:
    stor_11:
      hosts:
        beegfs_03:
        beegfs_04:
    meta_12:
      hosts:
        beegfs_03:
        beegfs_04:
    stor_12:
      hosts:
        beegfs_03:
        beegfs_04:
    meta_13:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_13:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_14:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_14:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_15:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_15:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_16:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_16:
      hosts:
        beegfs_04:
        beegfs_03:
```

2. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `meta_09` Bis `meta_16` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```

Informationen zur richtigen Größe finden Sie unter "Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools".

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
meta_09.yml	8015	i1b:100.127.101.9/16 i2b:100.127.102.9/16	0	netapp_03	Beegfs_m9_m10_m13_m14	A
meta_10.yml	8025	i2b:100.127.102.10/16 i1b:100.127.101.10/16	0	netapp_03	Beegfs_m9_m10_m13_m14	B
meta_11.yml	8035	i3b:100.127.101.11/16 i4b:100.127.102.11/16	1	netapp_04	Beegfs_m11_m12_m15_m16	A
meta_12.yml	8045	l4b:100.127.102.12/16 i3b:100.127.101.12/16	1	netapp_04	Beegfs_m11_m12_m15_m16	B

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
meta_13.yml	8055	i1b:100.127.101.13/16 i2b:100.127.102.13/16	0	netapp_03	Beegfs_m9_m10_m13_m14	A
meta_14.yml	8065	i2b:100.127.102.14/16 i1b:100.127.101.14/16	0	netapp_03	Beegfs_m9_m10_m13_m14	B
meta_15.yml	8075	i3b:100.127.101.15/16 i4b:100.127.102.15/16	1	netapp_04	Beegfs_m11_m12_m15_m16	A
meta_16.yml	8085	l4b:100.127.102.16/16 i3b:100.127.101.16/16	1	netapp_04	Beegfs_m11_m12_m15_m16	B

3. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_09` Bis `stor_16` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```

Die richtige Größe finden Sie unter "Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools" ..

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_09.yml	8013	i1b:100.127.103.9/16 i2b:100.127.104.9/16	0	netapp_03	Beegfs_s9_s10	A
stor_10.yml	8023	i2b:100.127.104.10/16 i1b:100.127.103.10/16	0	netapp_03	Beegfs_s9_s10	B
stor_11.yml	8033	i3b:100.127.103.11/16 i4b:100.127.104.11/16	1	netapp_04	Beegfs_s11_s12	A
stor_12.yml	8043	l4b:100.127.104.12/16 i3b:100.127.103.12/16	1	netapp_04	Beegfs_s11_s12	B
stor_13.yml	8053	i1b:100.127.103.13/16 i2b:100.127.104.13/16	0	netapp_03	Beegfs_s13_s14	A
stor_14.yml	8063	i2b:100.127.104.14/16 i1b:100.127.103.14/16	0	netapp_03	Beegfs_s13_s14	B
stor_15.yml	8073	i3b:100.127.103.15/16 i4b:100.127.104.15/16	1	netapp_04	Beegfs_s15_s16	A
stor_16.yml	8083	l4b:100.127.104.16/16 i3b:100.127.103.16/16	1	netapp_04	Beegfs_s15_s16	B

Schritt 4: Konfigurieren Sie den Bestand für einen nur-Storage-Baustein

In diesen Schritten wird beschrieben, wie Sie einen Ansible-Bestand für einen einzigen BeeGFS-Baustein einrichten. Der Hauptunterschied zwischen der Konfiguration für Metadaten + Storage und einem rein Storage-basierten Baustein besteht darin, dass alle Metadaten-Ressourcengruppen und Änderungen nicht mehr berücksichtigt werden `criteria_drive_count` Von 10 bis 12 für jeden Speicherpool.

Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter der vorhandenen Konfiguration:

```
# beegfs_05/beegfs_06 HA Pair (storage only building block):
  stor_17:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_18:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_19:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_20:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_21:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_22:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_23:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_24:
    hosts:
      beegfs_06:
      beegfs_05:
```

2. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_17` Bis `stor_24` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```

Die richtige Größe finden Sie unter "Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools" .

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_17.yml	8013	i1b:100.127.103.17/16 i2b:100.127.104.17/16	0	netapp_05	Beegfs_s17_s18	A
stor_18.yml	8023	i2b:100.127.104.18/16 i1b:100.127.103.18/16	0	netapp_05	Beegfs_s17_s18	B
stor_19.yml	8033	i3b:100.127.103.19/16 i4b:100.127.104.19/16	1	netapp_06	Beegfs_s19_s20	A
stor_20.yml	8043	l4b:100.127.104.20/16 i3b:100.127.103.20/16	1	netapp_06	Beegfs_s19_s20	B

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_21.yml	8053	i1b:100.127.103.21/16 i2b:100.127.104.21/16	0	netapp_05	Beegfs_s21_s22	A
stor_22.yml	8063	i2b:100.127.104.22/16 i1b:100.127.103.22/16	0	netapp_05	Beegfs_s21_s22	B
stor_23.yml	8073	i3b:100.127.103.23/16 i4b:100.127.104.23/16	1	netapp_06	Beegfs_s23_s24	A
stor_24.yml	8083	l4b:100.127.104.24/16 i3b:100.127.103.24/16	1	netapp_06	Beegfs_s23_s24	B

BeeGFS bereitstellen

Zur Implementierung und zum Management der Konfiguration werden ein oder mehrere Playbooks ausgeführt, die die Aufgaben enthalten, die Ansible ausführen muss, und das gesamte System in den gewünschten Zustand bringen.

Zwar können alle Aufgaben in einem einzigen Playbook enthalten sein, doch bei komplexen Systemen ist dies schnell und schwerfällig. Mit Ansible können Sie Rollen erstellen und verteilen, um wiederverwendbare Playbooks und verwandte Inhalte (z. B. Standardvariablen, Aufgaben und Handler) zu verpacken. Weitere Informationen finden Sie in der Ansible-Dokumentation für "Rollen".

Rollen werden häufig im Rahmen einer Ansible Sammlung mit zugehörigen Rollen und Modulen verteilt. Daher importieren diese Playbooks in erster Linie mehrere Rollen, die in den verschiedenen NetApp E-Series Ansible Sammlungen verteilt sind.

> Derzeit sind mindestens zwei Bausteine (vier Datei-Nodes) für die Bereitstellung von BeeGFS erforderlich, es sei denn, ein separates Quorum-Gerät ist als Tiebreaker konfiguriert, um Probleme beim Einrichten von Quorum mit einem Cluster mit zwei Nodes zu minimieren.

Schritte

1. Erstellen Sie eine neue `playbook.yml` Datei und schließen Sie Folgendes ein:

```
# 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.
      ansible.builtin.import_role:
        name: netapp_eseries.beegfs.beegfs_ha_7_4
```

In diesem Playbook wird ein paar ausgeführt `pre_tasks` Überprüfen Sie, ob Python 3 auf den Datei-Nodes installiert ist, und überprüfen Sie, ob die angegebenen Ansible-Tags unterstützt werden.

2. Verwenden Sie die `ansible-playbook` Befehl mit den Inventar- und Playbook-Dateien, wenn Sie bereit sind BeeGFS zu implementieren.

   Die Bereitstellung wird komplett ausgeführt `pre_tasks`, Und dann zur Bestätigung des Benutzers aufgefordert, bevor mit der tatsächlichen BeeGFS-Bereitstellung.

   Führen Sie den folgenden Befehl aus, indem Sie die Anzahl der Gabeln nach Bedarf anpassen (siehe Hinweis unten):

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

   Insbesondere bei größeren Bereitstellungen `forks` wird empfohlen, die Standardanzahl von Gabeln (5) mit dem Parameter zu überschreiben, um die Anzahl der Hosts zu erhöhen, die Ansible parallel konfiguriert. (Weitere Informationen finden Sie unter "Kontrolle der Playbook-Ausführung".) Die Einstellung für den maximalen Wert hängt von der Verarbeitungsleistung ab, die auf dem Ansible-Steuerungsknoten verfügbar ist. Das oben genannte Beispiel von 20 wurde auf einem virtuellen Ansible-Steuerungsknoten mit 4 CPUs (Intel® Xeon® Gold 6146 CPU @ 3,20 GHz) ausgeführt.

   Je nach Größe der Implementierung und Netzwerk-Performance zwischen dem Ansible Control Node und BeeGFS File- und Block-Nodes kann die Implementierungszeit variieren.

Konfigurieren Sie BeeGFS-Clients

Sie müssen den BeeGFS-Client auf allen Hosts installieren und konfigurieren, die Zugriff auf das BeeGFS-Dateisystem benötigen, z. B. Compute- oder GPU-Nodes. Für diese

Aufgabe können Sie Ansible und die BeeGFS-Sammlung verwenden.

Schritte

1. Richten Sie bei Bedarf über den Ansible-Steuerungsknoten passwortlose SSH für jeden Host ein, den Sie als BeeGFS-Clients konfigurieren möchten:

   ```
   ssh-copy-id <user>@<HOSTNAME_OR_IP>
   ```

2. Unter `host_vars/`Erstellen Sie für jeden BeeGFS-Client eine Datei mit dem Namen `<HOSTNAME>.yml` Füllen Sie den Platzhaltertext mit den korrekten Informationen für Ihre Umgebung aus:

   ```
   # 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>
       address: <IP>/<SUBNET_MASK>
   ```

   Bei der Bereitstellung mit zwei Subnetzadressierungsschemata müssen auf jedem Client zwei InfiniBand-Schnittstellen konfiguriert werden, eine in jedem der beiden Storage-IPoIB-Subnetze. Wenn Sie die Beispiel-Subnetze und empfohlenen Bereiche für jeden hier aufgeführten BeeGFS-Dienst verwenden, sollten Clients eine Schnittstelle im Bereich von bis und die andere in bis konfigurieren `100.127.1.0` `100.127.99.255` `100.128.1.0` `100.128.99.255`.

3. Erstellen Sie eine neue Datei `client_inventory.yml`, Und dann füllen Sie die folgenden Parameter an der Spitze:

   ```
   # 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.
   ```

Speichern Sie Passwörter nicht im Klartext. Verwenden Sie stattdessen den Ansible Vault (siehe Ansible-Dokumentation für) "Verschlüsseln von Inhalten mit Ansible Vault") Oder verwenden Sie die `--ask-become-pass` Option beim Ausführen des Playbooks.

4. Im `client_inventory.yml` Datei, Listen Sie alle Hosts auf, die als BeeGFS-Clients unter dem konfiguriert werden sollen `beegfs_clients` Gruppe, und geben Sie dann alle zusätzlichen Konfigurationen an, die zum Erstellen des BeeGFS-Client-Kernelmoduls erforderlich sind.

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        beegfs_01:
        beegfs_02:
        beegfs_03:
        beegfs_04:
        beegfs_05:
        beegfs_06:
        beegfs_07:
        beegfs_08:
        beegfs_09:
        beegfs_10:
      vars:
        # OPTION 1: If you're using the NVIDIA 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.
```

Wenn Sie die NVIDIA OFED-Treiber verwenden, stellen Sie sicher, dass `beegfs_client_ofed_include_path` sie auf den richtigen „Header include path“ für Ihre Linux-Installation verweist. Weitere Informationen finden Sie in der BeeGFS-Dokumentation für "RDMA-Unterstützung".

5. Im `client_inventory.yml` Datei, Listen Sie die BeeGFS-Dateisysteme auf, die am unteren Rand eines zuvor definierten gemountet werden sollen `vars`.

```
        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
```

Der `beegfs_client_config` Stellt die Einstellungen dar, die getestet wurden. Lesen Sie die Dokumentation im `netapp_eseries.beegfs` Kollektion's `beegfs_client` Rolle für einen umfassenden Überblick über alle Optionen. Dies enthält Details zum Mounten mehrerer BeeGFS-Dateisysteme oder zum mehrere Male das gleiche BeeGFS-Dateisystem.

6. Erstellen Sie eine neue `client_playbook.yml` Datei, und füllen Sie dann die folgenden Parameter aus:

```
# 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
```

Beim Importieren des weglassen `netapp_eseries.host` Sammlung und `ipoib` Rolle, wenn Sie bereits die erforderlichen IB/RDMA-Treiber und IP-Adressen auf den entsprechenden IPoIB-Schnittstellen installiert haben.

7. Führen Sie den folgenden Befehl aus, um den Client zu installieren und zu erstellen und BeeGFS zu mounten:

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

8. Bevor Sie das BeeGFS-Dateisystem in Produktion setzen, empfehlen wir * dringend*, sich bei allen Clients anzumelden und zu starten `beegfs-fsck --checkfs` Um sicherzustellen, dass alle Knoten erreichbar sind und keine Probleme gemeldet werden.

Skalierung auf mehr als fünf Bausteine

Pacemaker und Corosync können so konfiguriert werden, dass sie über fünf Bausteine (10 Datei-Knoten) hinausgehen. Allerdings gibt es Nachteile zu größeren Clustern, und schließlich Pacemaker und Corosync auferlegen ein Maximum von 32 Knoten.

NetApp hat nur BeeGFS HA Cluster für bis zu 10 Nodes getestet. Es wird nicht empfohlen, einzelne Cluster über dieses Limit hinaus zu skalieren. BeeGFS-Filesysteme müssen jedoch immer noch weit über 10 Nodes skalieren, und NetApp hat dies in der BeeGFS on NetApp Lösung berücksichtigt.

Durch die Implementierung mehrerer HA-Cluster mit einer Teilmenge der Bausteine in jedem Filesystem können Sie das gesamte BeeGFS Filesystem unabhängig von empfohlenen oder festen Grenzwerten für die zugrunde liegenden HA-Clustering-Mechanismen skalieren. Gehen Sie in diesem Szenario wie folgt vor:

- Erstellen Sie einen neuen Ansible-Bestand, der die zusätzlichen HA-Cluster darstellt, und geben Sie dann das Konfigurieren eines anderen Managementservices nicht ein. Zeigen Sie stattdessen auf `beegfs_ha_mgmtd_floating_ip` Variable in jedem zusätzlichen Cluster enthalten `ha_cluster.yml` An die IP für den ersten BeeGFS Management Service.

- Wenn Sie dem selben Dateisystem zusätzliche HA-Cluster hinzufügen, stellen Sie Folgendes sicher:
  - Die BeeGFS-Knoten-IDs sind eindeutig.
  - Die Dateinamen, die den einzelnen Diensten unter entsprechen `group_vars` Ist für alle Cluster eindeutig.
  - Die BeeGFS-Client- und Server-IP-Adressen sind für alle Cluster eindeutig.
  - Das erste HA-Cluster mit dem BeeGFS-Managementservice wird ausgeführt, bevor versucht wird, zusätzliche Cluster zu implementieren oder zu aktualisieren.
- Inventarisierung für jedes HA-Cluster getrennt in der eigenen Verzeichnisstruktur

  Wenn Sie versuchen, die Bestandsdateien für mehrere Cluster in einem Verzeichnisbaum zu mischen, kann dies zu Problemen führen, wie die BeeGFS HA-Rolle die auf ein bestimmtes Cluster angewendete Konfiguration aggregiert.

> Es ist nicht erforderlich, dass jedes HA-Cluster auf fünf Bausteine skaliert werden kann, bevor ein neues erstellt wird. In vielen Fällen lässt sich das Management mit weniger Bausteinen pro Cluster vereinfachen. Ein Ansatz besteht darin, die Bausteine in jedem einzelnen Rack als HA-Cluster zu konfigurieren.

Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools

Wenn Sie den standardmäßigen vier Volumes pro Storage-Pool-Konfiguration für Bausteine der zweiten Generation folgen, finden Sie in der folgenden Tabelle.

Diese Tabelle enthält empfohlene Prozentsätze, die als Volume-Größe im verwendet werden können `eseries_storage_pool_configuration` Für jede BeeGFS-Metadaten oder jedes Storage-Ziel:

Laufwerkgröße	Größe
1,92 TB	18
3,84 TB	21.5
7,68 TB	22.5
15,3 TB	24

> Die oben stehende Anleitung gilt nicht für den Speicherpool, der den Management-Service enthält. Dieser sollte die Größe von über 25 % verringern, um 1 % des Speicherpools für Management-Daten zuzuweisen.

Um zu verstehen, wie diese Werte ermittelt wurden, lesen Sie "TR-4800: Anhang A: Verständnis von SSD-Ausdauer und -Überprovisionierung".

Baustein mit hoher Kapazität

Im Standard BeeGFS Solution Deployment Guide werden Verfahren und Empfehlungen für High-Performance-Workload-Anforderungen beschrieben. Kunden, die hohe Kapazitätsanforderungen erfüllen möchten, sollten die hier aufgeführten Variationen bei Implementierung und Empfehlungen beobachten.

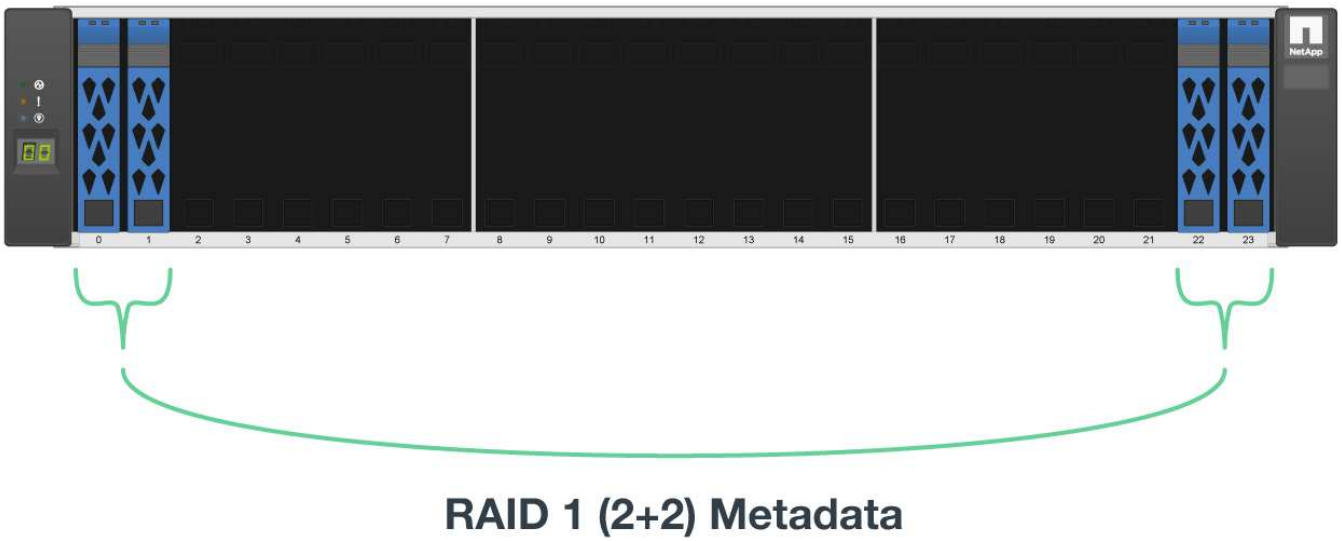
Controller

Wenn Sie Bausteine mit hoher Kapazität benötigen, sollten EF600 Controller durch EF300 Controller ersetzt werden, wobei jeweils eine Cascade HIC zur SAS-Erweiterung installiert ist. Jeder Block-Node verfügt über eine minimale Anzahl von NVMe-SSDs, die im Array-Gehäuse für BeeGFS-Metadaten-Storage befüllt sind, und wird an Erweiterungs-Shelfs mit NL-SAS-HDDs für BeeGFS-Storage-Volumes angeschlossen.

Die Konfiguration des Datei-Node zu Block-Nodes bleibt unverändert.

Laufwerkplatzierung

In jedem Block-Node sind mindestens 4 NVMe-SSDs für BeeGFS-Metadaten-Storage erforderlich. Diese Laufwerke sollten in den äußeren Steckplätzen des Gehäuses platziert werden.



Erweiterungsfächer

Der Baustein mit hoher Kapazität kann mit 1-7, 60 Laufwerkserweiterungsfächern pro Speicher-Array dimensioniert werden.

Anweisungen zum Kabelanschluss der einzelnen Erweiterungsfächer finden Sie unter "Siehe EF300-Verkabelung für Laufwerk-Shelfs".

Verwenden Sie individuelle Architekturen

Überblick und Anforderungen

Verwenden Sie alle NetApp E/EF-Series Storage-Systeme als BeeGFS-Block-Nodes und x86-Server als BeeGFS-Datei-Nodes, wenn Sie BeeGFS High Availability-Cluster mithilfe von Ansible implementieren.

Definitionen für die in diesem Abschnitt verwendete Terminologie finden Sie auf der "Begriffe und Konzepte" Seite.

Einführung

"NetApp-verifizierte Architekturen"Einige Kunden und Partner bieten zwar vordefinierte Referenzkonfigurationen und Hinweise zur Größenbestimmung, möchten aber möglicherweise lieber benutzerdefinierte Architekturen entwickeln, die sich besser an die speziellen Anforderungen oder Hardwarepräferenzen anpassen. Einer der Hauptvorteile der Entscheidung für BeeGFS auf NetApp ist die Möglichkeit, BeeGFS HA-Cluster auf Shared-Festplatten mit Ansible zu implementieren. Dadurch wird das Cluster-Management vereinfacht und die Zuverlässigkeit mithilfe von NetApp entwickelten HA-Komponenten gesteigert. Die Implementierung benutzerdefinierter BeeGFS-Architekturen auf NetApp erfolgt noch immer mithilfe von Ansible und den Appliance-ähnlichen Ansatz auf einem flexiblen Spektrum an Hardware.

Dieser Abschnitt beschreibt die allgemeinen Schritte, die zur Implementierung von BeeGFS-Dateisystemen auf NetApp Hardware und zur Verwendung von Ansible zur Konfiguration von BeeGFS-Dateisystemen erforderlich sind. Details zu Best Practices für das Design von BeeGFS-Dateisystemen und optimierten Beispielen finden Sie im "NetApp-verifizierte Architekturen" Abschnitt.

Implementierungsübersicht

Die Bereitstellung eines BeeGFS-Dateisystems umfasst im Allgemeinen die folgenden Schritte:

- Ersteinrichtung:
  - Hardware installieren/verkabeln.
  - Richten Sie Datei- und Block-Nodes ein.
  - Richten Sie einen Ansible-Steuerungsknoten ein.
- Definieren Sie das BeeGFS-Dateisystem als Ansible-Inventar.
- Ausführen von Ansible für Datei- und Block-Nodes zur Implementierung von BeeGFS
  - Optional zum Einrichten von Clients und Mounten BeeGFS.

In den nachfolgenden Abschnitten werden diese Schritte näher beschrieben.

Ansible übernimmt alle Softwareprovisionierung- und Konfigurationsaufgaben, z. B.:

- Erstellen/Zuordnen von Volumes auf Block-Nodes
- Formatieren/Tuning von Volumes auf Datei-Nodes
- Installation/Konfiguration von Software auf Datei-Nodes
- Einrichten des HA-Clusters und Konfigurieren von BeeGFS-Ressourcen und File-System-Services

Anforderungen

Unterstützung für BeeGFS in Ansible ist veröffentlicht am "Ansible-Galaxie" Als Sammlung von Rollen und Modulen zur Automatisierung der End-to-End-Implementierung und des Managements von BeeGFS HA-Clustern.

BeeGFS selbst ist nach einem <major>.<minor>.<patch> Versioning Schema versioniert und die Sammlung pflegt Rollen für jede unterstützte <major>.<minor> Version von BeeGFS, zum Beispiel BeeGFS 7.2 oder BeeGFS 7.3. Da Updates für die Sammlung veröffentlicht werden, wird die Patch-Version in jeder Rolle aktualisiert, um auf die neueste verfügbare BeeGFS-Version für diesen Release Branch (Beispiel: 7.2.8) zu verweisen. Jede Version der Sammlung wird auch mit bestimmten Linux-Distributionen und -Versionen getestet und unterstützt, derzeit Red Hat für Dateiknoten und Red Hat und Ubuntu für Clients. Die Ausführung anderer Distributionen wird nicht unterstützt, und die Ausführung anderer Versionen (insbesondere anderer Hauptversionen) wird nicht empfohlen.

Ansible-Steuerungsknoten

Dieser Node enthält Inventar und Playbooks, die zum Managen von BeeGFS verwendet werden. Dazu benötigen Sie:

- Ansible, 6.x (ansible-Core, 2.13)
- Python 3.6 (oder höher)
- Python (Pip) Pakete: Ipaddr und netaddr

Es empfiehlt sich auch, passwortlose SSH vom Steuerungsknoten auf alle BeeGFS Datei-Knoten und -Clients einzurichten.

BeeGFS-Dateiknoten

Dateiknoten müssen Red Hat Enterprise Linux (RHEL) 9.4 ausführen und Zugriff auf das HA-Repository mit den erforderlichen Paketen (Pacemaker, Corosync, Fence-Agents-All, Resource-Agents) haben. Beispielsweise kann der folgende Befehl ausgeführt werden, um das entsprechende Repository unter RHEL 9 zu aktivieren:

```
subscription-manager repo-override repo=rhel-9-for-x86_64-
highavailability-rpms --add=enabled:1
```

BeeGFS Client-Knoten

Eine BeeGFS-Client-Ansible-Rolle steht zur Verfügung, um das BeeGFS-Client-Paket zu installieren und BeeGFS-Mount(s) zu verwalten. Diese Rolle wurde mit RHEL 9.4 und Ubuntu 22.04 getestet.

Wenn Sie nicht Ansible verwenden, um den BeeGFS-Client einzurichten und BeeGFS zu mounten, any "BeeGFS unterstützte Linux-Distribution und Kernel" Kann verwendet werden.

Ersteinrichtung

Installieren und verkabeln Sie die Hardware

Schritte erforderlich, um Hardware zu installieren und zu verkabeln, die zum Ausführen von BeeGFS auf NetApp verwendet wird.

Planen Sie die Installation

Jedes BeeGFS-Dateisystem besteht aus einer Anzahl von Datei-Nodes, auf denen BeeGFS-Dienste über Backend-Storage ausgeführt werden, der von einer Anzahl von Block-Nodes bereitgestellt wird. Die Datei-Nodes sind in einem oder mehreren Hochverfügbarkeits-Clustern konfiguriert, um Fehlertoleranz für BeeGFS-Services zu bieten. Jeder Block-Node ist bereits ein aktiv/aktiv-HA-Paar. Die Mindestanzahl unterstützter File-Nodes in jedem HA-Cluster beträgt drei und die maximale Anzahl unterstützter File-Nodes in jedem Cluster ist zehn. BeeGFS-Filesysteme können über zehn Nodes hinaus skaliert werden, indem mehrere unabhängige HA-Cluster implementiert werden, die zusammen einen Single Filesystem Namespace bieten.

Normalerweise wird jedes HA-Cluster als eine Reihe von „Bausteinen“ bereitgestellt, in denen einige File-Nodes (x86-Server) direkt mit einer Reihe von Block-Nodes verbunden sind (in der Regel E-Series Storage-Systeme). Diese Konfiguration erzeugt ein asymmetrisches Cluster, in dem BeeGFS-Services nur auf bestimmten Datei-Nodes ausgeführt werden können, die Zugriff auf den Back-End-Block-Storage haben, der für BeeGFS-Ziele verwendet wird. Die Balance zwischen Datei- und Block-Nodes in jedem Baustein und dem für die direkte Verbindung verwendeten Storage-Protokoll hängen von den Anforderungen einer bestimmten Installation ab.

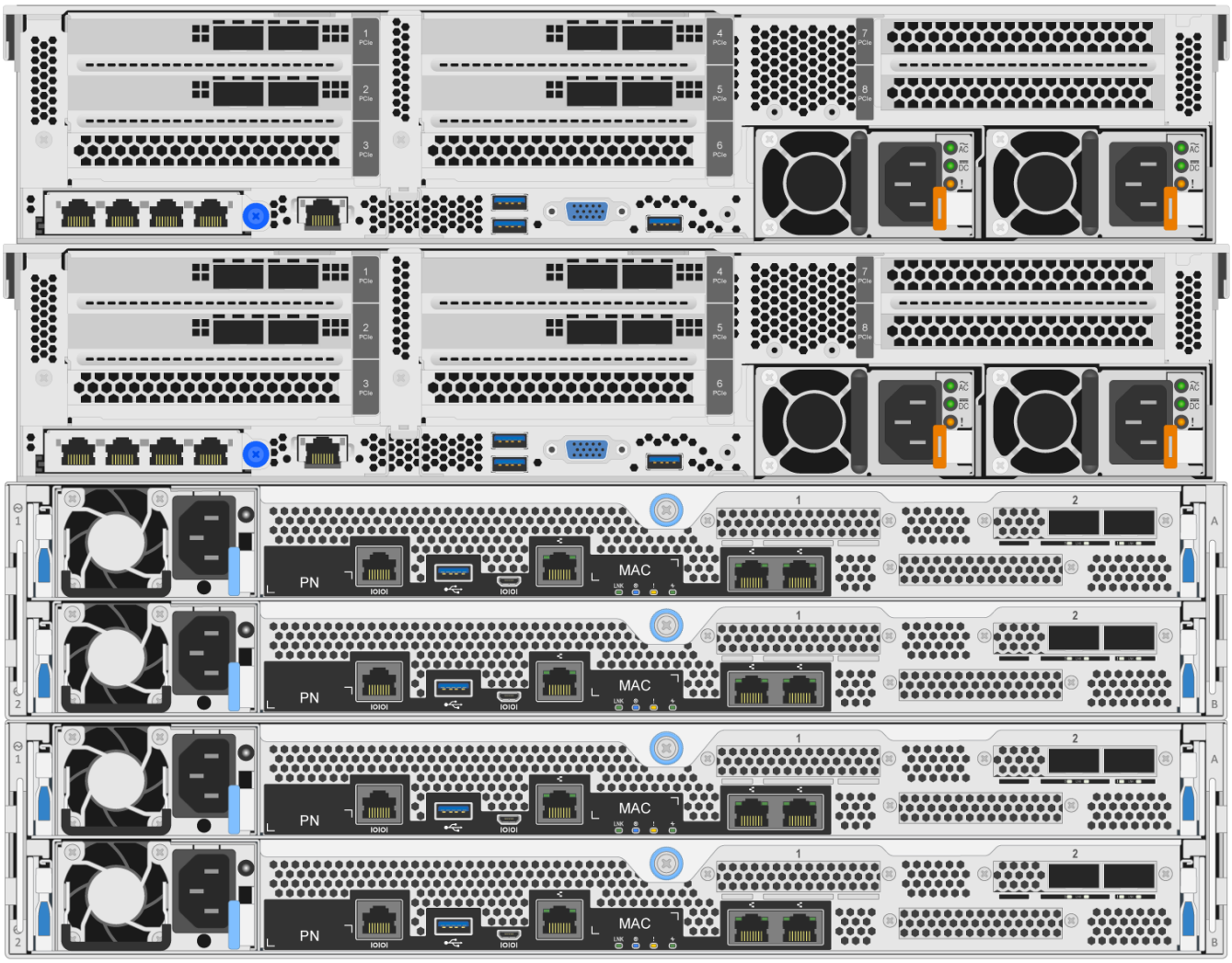
Eine alternative HA-Cluster-Architektur verwendet ein Storage-Fabric (auch als Storage Area Network oder SAN bekannt) zwischen den Datei- und Block-Nodes, um ein symmetrisches Cluster herzustellen. So können BeeGFS-Services auf jedem Datei-Node in einem bestimmten HA-Cluster ausgeführt werden. Da symmetrische Cluster aufgrund der zusätzlichen SAN-Hardware nicht so kostengünstig sind, setzt diese Dokumentation den Einsatz eines asymmetrischen Clusters voraus, der als eine Reihe von einem oder mehreren Bausteinen implementiert wird.

Stellen Sie sicher, dass die gewünschte Dateisystemarchitektur für eine bestimmte BeeGFS-Bereitstellung gut verstanden wird, bevor Sie mit der Installation fortfahren.

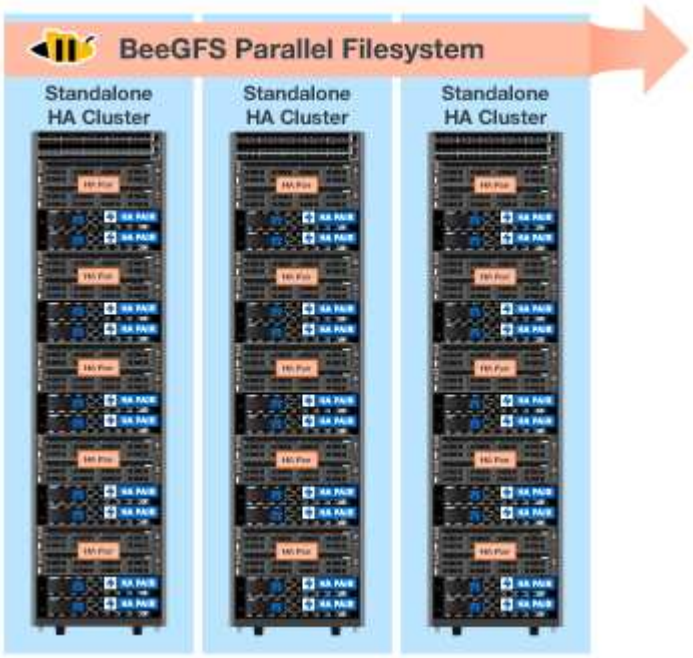
Rack-Hardware

Bei der Planung der Installation ist es wichtig, dass alle Geräte in jedem Baustein in benachbarten Rack-Einheiten verfügbar sind. Als Best Practice empfiehlt es sich, Datei-Nodes sofort über Block-Nodes in jedem Baustein verfügbar zu machen. Befolgen Sie die Dokumentation für die Modelle der Datei und "Block-Storage" Knoten, die Sie verwenden, wenn Sie Schienen und Hardware im Rack installieren.

Beispiel für einen einzelnen Baustein:

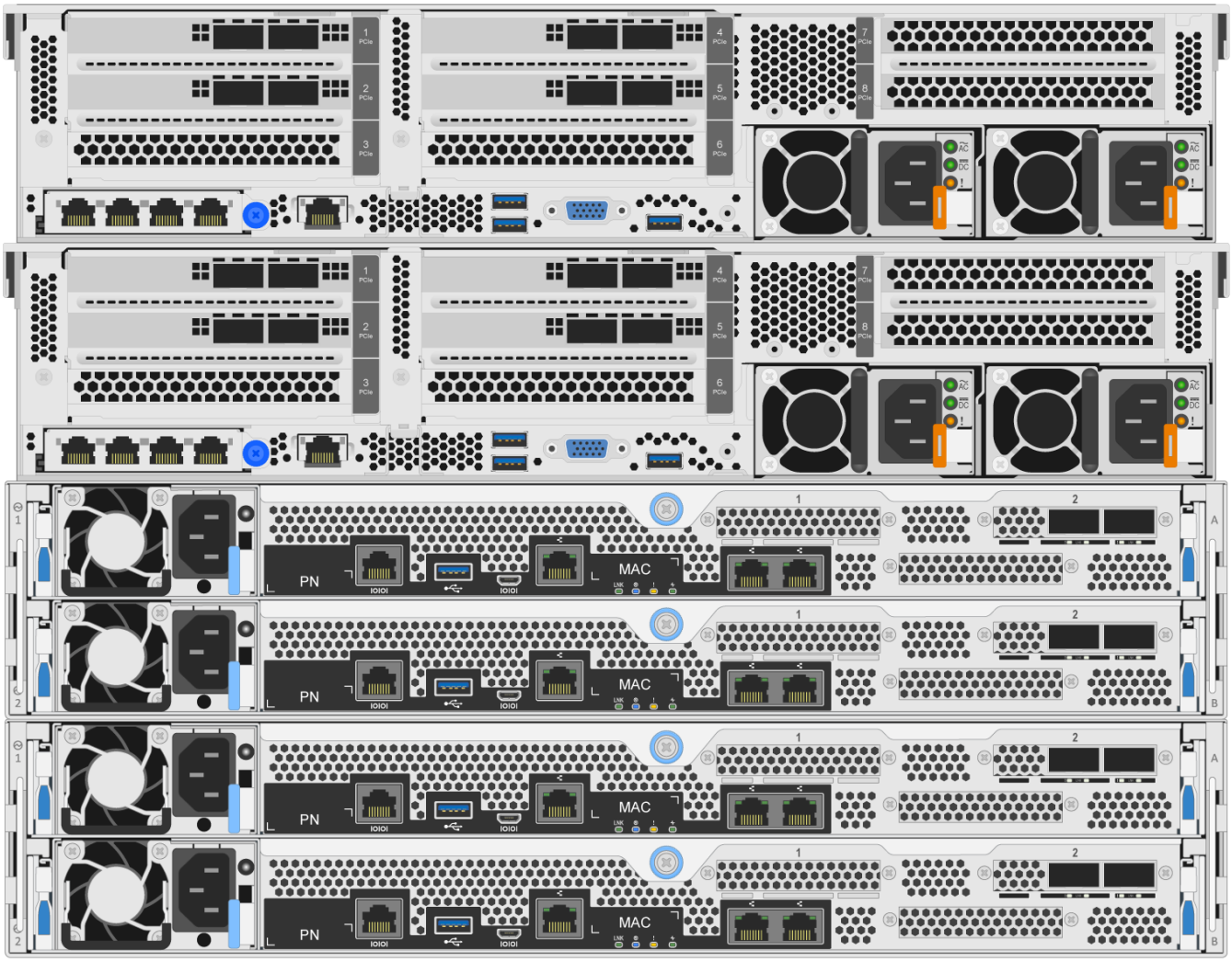


Beispiel für eine große BeeGFS-Installation, bei der es in jedem HA-Cluster mehrere Bausteine und mehrere HA-Cluster im Filesystem gibt:



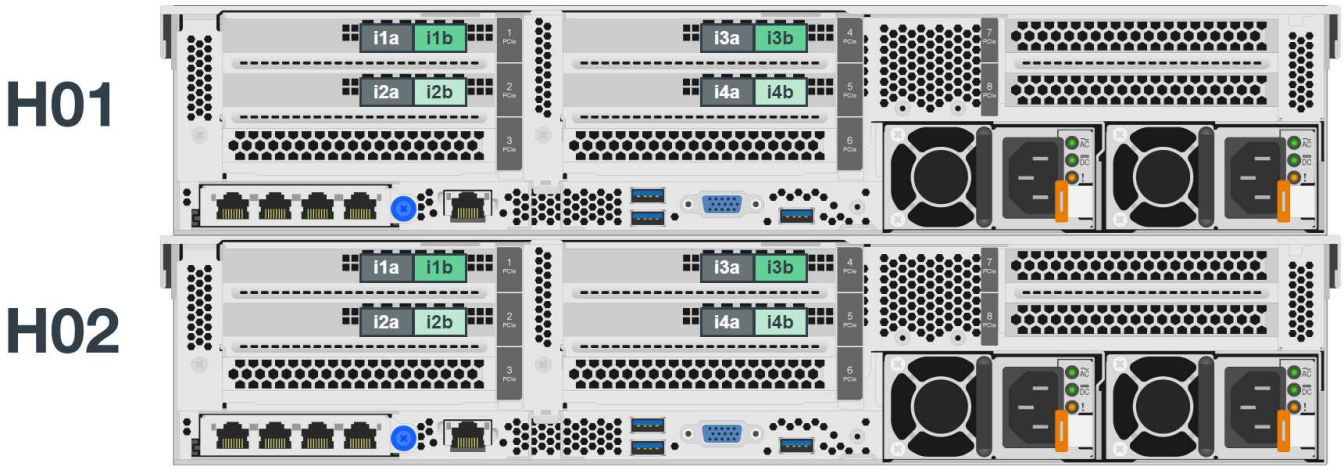
Kabeldatei- und Blockknoten

Sie werden die HIC-Ports der Block-Nodes der E-Series normalerweise mit dem vorgesehenen Host Channel Adapter (für InfiniBand-Protokolle) oder den Host-Bus-Adaptern (für Fibre Channel und andere Protokolle) der Datei-Nodes verbinden. Die genaue Art und Weise, diese Verbindungen herzustellen, hängt von der gewünschten Dateisystemarchitektur ab, hier ist ein Beispiel"Basierend auf BeeGFS der zweiten Generation auf NetApp Verified Architecture":



Dateiknoten mit dem Client-Netzwerk verkabeln

Jeder Datei-Node verfügt über eine bestimmte Anzahl von InfiniBand- oder Ethernet-Ports für BeeGFS-Client-Traffic. Je nach Architektur verfügt jeder Datei-Node über eine oder mehrere Verbindungen zu einem hochperformanten Client-/Storage-Netzwerk, möglicherweise zu mehreren Switches für Redundanz und höhere Bandbreite. Hier sehen Sie ein Beispiel für die Client-Verkabelung mithilfe redundanter Netzwerk-Switches, bei denen die in Dunkelgrün bzw. hellgrün hervorgehobenen Ports mit separaten Switches verbunden sind:



Verbindung zwischen Management-Netzwerk und Stromversorgung

Stellen Sie alle erforderlichen Netzwerkverbindungen für in-Band- und Out-of-Band-Netzwerke her.

Schließen Sie alle Netzteile an, um sicherzustellen, dass jeder Datei- und Block-Knoten Verbindungen zu mehreren Stromverteilungs-Einheiten hat, um Redundanz zu gewährleisten (falls verfügbar).

Datei- und Block-Knoten einrichten

Manuelle Schritte zur Einrichtung von Datei- und Block-Nodes vor der Ausführung von Ansible erforderlich

File-Nodes

Konfigurieren des Baseboard Management Controllers (BMC)

Ein Baseboard Management Controller (BMC), der manchmal als Service-Prozessor bezeichnet wird, ist der generische Name für die Out-of-Band-Management-Funktion, die in verschiedenen Server-Plattformen integriert ist, die Remote-Zugriff bieten können, selbst wenn das Betriebssystem nicht installiert ist oder nicht zugänglich ist. Anbieter vermarkten diese Funktionalität in der Regel mit ihrem eigenen Branding. Auf dem Lenovo SR665 wird beispielsweise der BMC als Lenovo XClarity Controller (XCC) bezeichnet.

Befolgen Sie die Dokumentation des Serveranbieters, um alle erforderlichen Lizenzen für den Zugriff auf diese Funktionalität zu aktivieren und sicherzustellen, dass der BMC mit dem Netzwerk verbunden und für den Remote-Zugriff entsprechend konfiguriert ist.

> Wenn ein BMC-basiertes Fechten mit Redfish gewünscht wird, stellen Sie sicher, dass Redfish aktiviert ist und die BMC-Schnittstelle über das auf dem Dateiknoten installierte Betriebssystem zugänglich ist. Auf dem Netzwerk-Switch kann eine spezielle Konfiguration erforderlich sein, wenn BMC und der Betrieb dieselbe physische Netzwerkschnittstelle nutzen.

Systemeinstellungen Einstellen

Stellen Sie mithilfe der Benutzeroberfläche des System-Setup (BIOS/UEFI) sicher, dass Einstellungen auf maximale Leistung eingestellt sind. Die genauen Einstellungen und optimalen Werte variieren je nach verwendetes Servermodell. Es wird eine Anleitung zur Verfügung gestellt"Verifizierte Datei-Node-Modelle", andernfalls beziehen Sie sich auf die Dokumentation und Best Practices des Serverherstellers, die auf Ihrem Modell basieren.

Installieren Sie ein Betriebssystem

Installieren Sie ein unterstütztes Betriebssystem basierend auf den aufgeführten Dateiknoten"Hier" -Anforderungen. Beachten Sie die nachfolgenden Schritte, die auf Ihrer Linux-Distribution basieren.

Red Hat

Verwenden Sie den Red Hat Subscription Manager, um das System zu registrieren und zu abonnieren, damit die erforderlichen Pakete aus den offiziellen Red Hat-Repositorys installiert werden können und um Updates auf die unterstützte Version von Red Hat zu beschränken: `subscription-manager release --set=<MAJOR_VERSION>.<MINOR_VERSION>` . Anweisungen finden Sie unter "Registrieren und Abonnieren eines RHEL Systems" Und "Einschränken von Aktualisierungen" .

Red hat Repository mit den für hohe Verfügbarkeit erforderlichen Paketen aktivieren:

```
subscription-manager repo-override --repo=rhel-9-for-x86_64
-highavailability-rpms --add=enabled:1
```

Managementnetzwerk Konfigurieren

Konfigurieren Sie alle erforderlichen Netzwerkschnittstellen für die bandinterne Verwaltung des Betriebssystems. Die genauen Schritte hängen von der jeweiligen Linux-Distribution und der verwendeten Version ab.

> Vergewissern Sie sich, dass SSH aktiviert ist und alle Managementoberflächen über den Ansible Kontroll-Node zugänglich sind.

Aktualisieren der HCA- und HBA-Firmware

Stellen Sie sicher, dass auf allen HBAs und HCAs unterstützte Firmware-Versionen ausgeführt "NetApp Interoperabilitätsmatrix"werden, die auf dem aufgeführt sind, und aktualisieren Sie ggf.. Weitere Empfehlungen für NVIDIA ConnectX Adapter finden Sie "Hier".

Block-Nodes

Befolgen Sie die Schritte zu "Die Inbetriebnahme ist möglich mit E-Series" Um den Managementport an jedem Block Node Controller zu konfigurieren und optional den Namen des Storage-Arrays für jedes System festzulegen.

> Es ist keine zusätzliche Konfiguration erforderlich, die darüber hinaus sicherstellt, dass alle Block-Nodes über den Ansible-Kontroll-Node zugänglich sind. Die verbleibende Systemkonfiguration wird mit Ansible angewendet/gewartet.

Ansible-Steuerungsknoten Einrichten

Richten Sie einen Ansible-Steuerungsknoten ein, um das Dateisystem zu implementieren und zu managen.

Überblick

Ein Ansible-Steuerungsknoten ist eine physische oder virtuelle Linux-Maschine, die zum Verwalten des

Clusters verwendet wird. Er muss folgende Anforderungen erfüllen:

- Lernen Sie die "Anforderungen"Rolle für die BeeGFS HA kennen, einschließlich der installierten Versionen von Ansible, Python und zusätzlichen Python-Paketen.
- Treffen Sie den Beamten "Ansible-Control-Node-Anforderungen" Einschließlich Betriebssystemversionen.
- SSH- und HTTPS-Zugriff auf alle Datei- und Block-Nodes

Detaillierte Installationsschritte finden Sie "Hier".

Definieren Sie das BeeGFS-Dateisystem

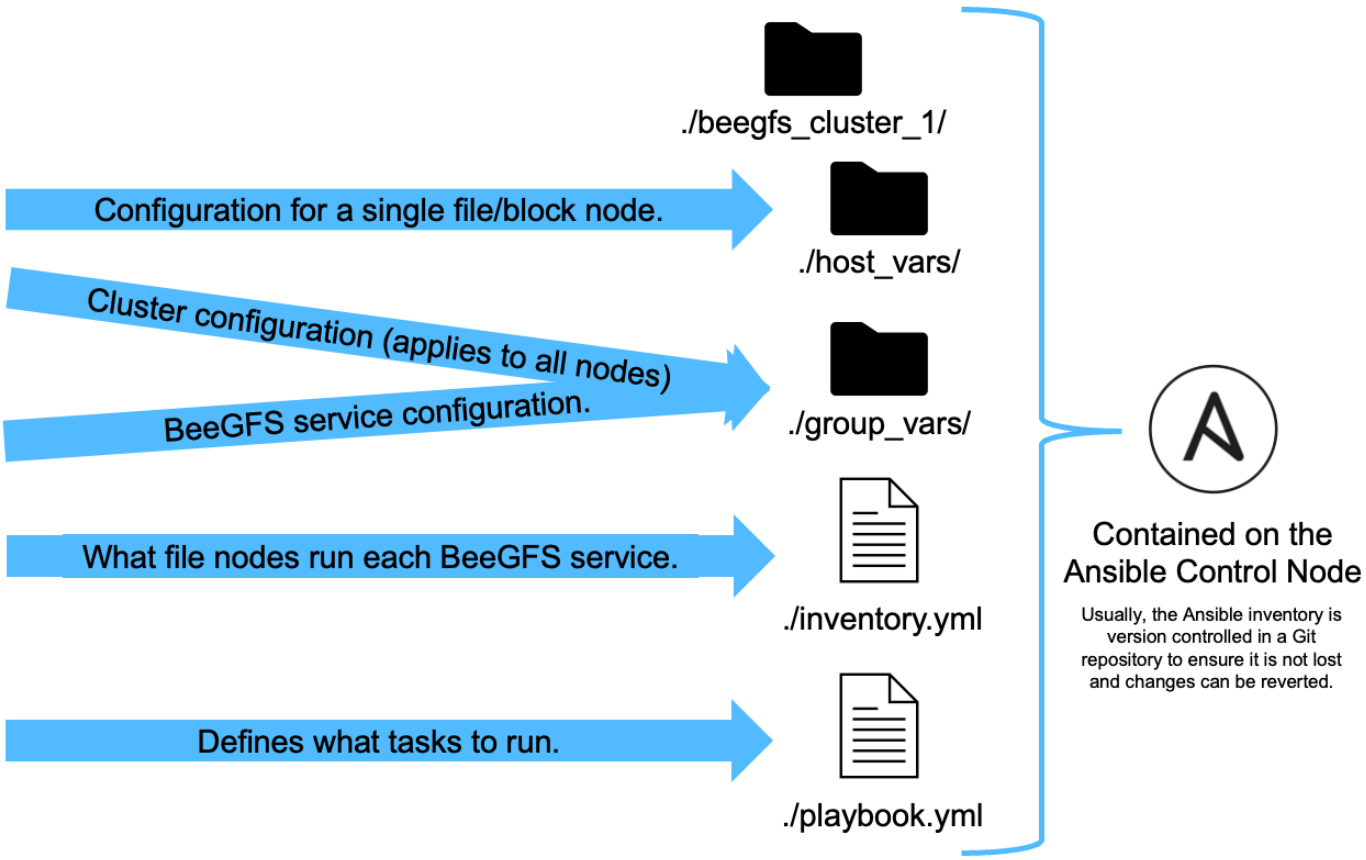
Ansible-Bestandsübersicht

Der Ansible-Bestand ist ein Satz von Konfigurationsdateien, die das gewünschte BeeGFS HA-Cluster definieren.

Überblick

Es wird empfohlen, die standardmäßigen Ansible-Methoden für die Organisation des zu befolgen "Inventar", Einschließlich der Verwendung von "Unterverzeichnisse/Dateien" Anstatt den gesamten Bestand in einer Datei zu speichern.

Der Ansible-Bestand für ein einzelnes BeeGFS HA-Cluster ist wie folgt organisiert:



Da ein einzelnes BeeGFS-Dateisystem mehrere HA-Cluster umfassen kann, können große Installationen mehrere Ansible-Inventare durchführen. Im Allgemeinen wird es nicht empfohlen, mehrere HA-Cluster als einen einzelnen Ansible-Bestand zu definieren, um Probleme zu vermeiden.

Schritte

1. Erstellen Sie auf Ihrem Ansible-Kontroll-Node ein leeres Verzeichnis, das den Ansible-Bestand für das BeeGFS-Cluster enthält, das bereitgestellt werden soll.
   a. Wenn Ihr Filesystem schließlich mehrere HA-Cluster enthalten soll/kann, wird empfohlen, zuerst ein Verzeichnis für das Dateisystem zu erstellen und dann Unterverzeichnisse für den Bestand, der die einzelnen HA-Cluster darstellt, einzutragen. Beispiel:

```
beegfs_file_system_1/
    beegfs_cluster_1/
    beegfs_cluster_2/
    beegfs_cluster_N/
```

2. Erstellen Sie im Verzeichnis, das den Bestand für den HA-Cluster enthält, den Sie bereitstellen möchten, zwei Verzeichnisse `group_vars` Und `host_vars` Und zwei Dateien `inventory.yml` Und `playbook.yml`.

Die folgenden Abschnitte gehen durch die Definition des Inhalts jeder dieser Dateien.

Planen Sie das Dateisystem

Planen Sie die Filesystem-Implementierung, bevor Sie den Ansible-Bestand aufbauen.

Überblick

Vor der Bereitstellung des Dateisystems sollten Sie festlegen, welche IP-Adressen, Ports und andere Konfigurationen für alle Datei-Nodes, Block-Nodes und BeeGFS-Services im Cluster erforderlich sind. Während die genaue Konfiguration je nach Architektur des Clusters variiert, werden in diesem Abschnitt Best Practices und Schritte definiert, die allgemein anwendbar sind.

Schritte

1. Wenn Sie zum Verbinden von Datei-Nodes mit Block-Nodes ein IP-basiertes Storage-Protokoll (z. B. iSER, iSCSI, NVMe/IB oder NVMe/RoCE) verwenden, füllen Sie für jeden Baustein das folgende Arbeitsblatt aus. Jede direkte Verbindung in einem einzelnen Baustein sollte ein eigenes Subnetz haben, und es sollte keine Überschneidung mit Subnetzen geben, die für die Client-Server-Konnektivität verwendet werden.

Datei-Node	IB-Port	IP-Adresse	Block-Node	IB-Port	Physische IP-Adresse	Virtuelle IP (nur für EF600 mit HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

Wenn Datei- und Block-Nodes in jedem Baustein direkt verbunden sind, können für mehrere Bausteine oft dieselben IPs/Schemas verwendet werden.

2. Füllen Sie unabhängig davon aus, ob Sie InfiniBand oder RDMA over Converged Ethernet (RoCE) für das Storage-Netzwerk verwenden, das folgende Arbeitsblatt aus, um die IP-Bereiche zu ermitteln, die für HA-Cluster-Services, BeeGFS-Fileservices und Clients verwendet werden, um zu kommunizieren:

Zweck	InfiniBand-Port	IP-Adresse oder Bereich
BeeGFS Cluster-IP(s)	<INTERFACE(s)>	<RANGE>
BeeGFS Management	<INTERFACE(s)>	<IP(s)>
BeeGFS-Metadaten	<INTERFACE(s)>	<RANGE>
BeeGFS-Speicherung	<INTERFACE(s)>	<RANGE>
BeeGFS-Clients	<INTERFACE(s)>	<RANGE>

   a. Wenn Sie ein einzelnes IP-Subnetz verwenden, ist nur ein Arbeitsblatt erforderlich. Füllen Sie andernfalls auch ein Arbeitsblatt für das zweite Subnetz aus.

3. Füllen Sie für jeden Baustein im Cluster das folgende Arbeitsblatt aus, um festzulegen, welche BeeGFS-Services ausgeführt werden sollen. Geben Sie für jeden Service die bevorzugten/sekundären Dateiknoten, Netzwerkport, fließende IP(s), NUMA-Zonenzuweisung (falls erforderlich) und welche Block-Nodes für ihre Ziele verwendet werden. Beachten Sie beim Ausfüllen des Arbeitsblatts die folgenden Richtlinien:

   a. Geben Sie BeeGFS-Dienste als entweder an `mgmt.yml`, `meta_<ID>.yml`, Oder `storage_<ID>.yml` Wobei ID eine eindeutige Nummer für alle BeeGFS-Dienste dieses Typs in diesem Dateisystem darstellt. Dieses Übereinkommen erleichtert die Rückverweisen auf dieses Arbeitsblatt in den nachfolgenden Abschnitten, während Dateien für die Konfiguration der einzelnen Dienste erstellt werden.

   b. Ports für BeeGFS-Dienste müssen nur in einem bestimmten Baustein einzigartig sein. Stellen Sie sicher, dass Dienste mit derselben Portnummer nicht jemals auf demselben Dateiknoten ausgeführt werden können, um Portkonflikte zu vermeiden.

   c. Bei Bedarf können Services Volumes von mehr als einem Block-Node und/oder Storage-Pool nutzen (und nicht alle Volumes müssen Eigentum desselben Controllers sein). Zudem können mehrere Services denselben Block-Node und/oder dieselbe Storage-Pool-Konfiguration nutzen (einzelne Volumes werden in einem späteren Abschnitt definiert).

BeeGFS-Dienst (Dateiname)	File-Nodes	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
<SERVICE TYPE>_<ID>.yml	<PREFERRED FILE NODE> <SECONDARY FILE NODE(s)>	<PORT>	<INTERFACE>:<IP/SUBNET> <INTERFACE>:<IP/SUBNET>	<NUMA NODE/ZONE>	<BLOCK NODE>	<STORAGE POOL/VOLUME GROUP>	<A OR B>

Weitere Details zu Standardkonventionen, Best Practices und ausgefüllten Arbeitsblättern finden Sie in den "Best Practices in sich vereint""Definieren Sie BeeGFS-Bausteine"Abschnitten und der BeeGFS on NetApp Verified Architecture.

Datei- und Blockknoten definieren

Konfigurieren Einzelner Dateiknoten

Legen Sie die Konfiguration für einzelne Datei-Nodes mithilfe von Host-Variablen fest (Host_vars).

Überblick

In diesem Abschnitt wird das Ausfüllen von erläutert `host_vars/<FILE_NODE_HOSTNAME>.yml` Datei für jeden Datei-Node im Cluster. Diese Dateien sollten nur die für einen bestimmten Dateiknoten spezifische Konfiguration enthalten. Hierzu zählen folgende allgemein:

- Die Definition der IP oder des Hostnamen Ansible sollte für die Verbindung mit dem Node verwendet werden.
- Konfiguration zusätzlicher Schnittstellen und Cluster-IPs, die für HA-Cluster-Services (Pacemaker und Corosync) zur Kommunikation mit anderen Datei-Nodes verwendet werden Standardmäßig verwenden diese Dienste dasselbe Netzwerk wie die Managementoberfläche, aber für Redundanz sollten zusätzliche Schnittstellen verfügbar sein. Es ist üblich, zusätzliche IPs im Storage-Netzwerk zu definieren, ohne dass ein zusätzliches Cluster- oder Management-Netzwerk erforderlich ist.
  - Die Performance aller Netzwerke, die für die Cluster-Kommunikation verwendet werden, ist für die Performance des Filesystems nicht von entscheidender Bedeutung. Bei der Standardkonfiguration des Clusters bietet ein Netzwerk mit mindestens 1 GB/s im Allgemeinen ausreichende Performance für Cluster-Vorgänge, z. B. die Synchronisierung von Node-Status und die Koordinierung von Änderungen des Clusterressourcenstatus. Langsame/überlastete Netzwerke können dazu führen, dass Änderungen des Ressourcenzustands länger dauern als üblich, und in extremen Fällen können Nodes aus dem Cluster entfernt werden, wenn sie in einem angemessenen Zeitrahmen keine Herzschläge senden können.
- Konfigurieren von Schnittstellen, die für die Verbindung zu Block-Nodes über das gewünschte Protokoll verwendet werden (z. B. iSCSI/iSER, NVMe/IB, NVMe/RoCE, FCP usw.)

Schritte

Wenn Sie auf das im "Planen Sie das Dateisystem" Abschnitt definierte IP-Adressierungsschema verweisen, erstellen Sie für jeden Dateiknoten im Cluster eine Datei `host_vars/<FILE_NODE_HOSTNAME>/yml` und füllen Sie sie wie folgt aus:

1. Geben Sie oben die IP oder den Hostnamen an, den Ansible für den Node mit SSH verwenden und verwalten soll:

```
ansible_host: "<MANAGEMENT_IP>"
```

2. Konfigurieren Sie zusätzliche IPs, die für den Cluster-Datenverkehr verwendet werden können:
   a. Wenn der Netzwerktyp ist "InfiniBand (mit IPoIB)":

```
eseries_ipoib_interfaces:
- name: <INTERFACE>  # Example: ib0 or i1b
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

b. Wenn der Netzwerktyp ist "RDMA über Converged Ethernet (RoCE)":

```
eseries_roce_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

c. Wenn der Netzwerktyp ist "Ethernet (nur TCP, kein RDMA)":

```
eseries_ip_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

3. Geben Sie an, welche IPs für Cluster-Datenverkehr verwendet werden sollen, wobei die bevorzugten IPs höher aufgeführt sind:

```
beegfs_ha_cluster_node_ips:
- <MANAGEMENT_IP> # Including the management IP is typically but not
required.
- <IP_ADDRESS>    # Ex: 100.127.100.1
- <IP_ADDRESS>    # Additional IPs as needed.
```

IPS, die in Schritt zwei konfiguriert sind, werden nicht als Cluster-IPs verwendet, es sei denn, sie sind im enthalten `beegfs_ha_cluster_node_ips` Liste. So können mithilfe von Ansible zusätzliche IPs/Schnittstellen konfiguriert werden, die bei Bedarf für andere Zwecke verwendet werden können.

4. Wenn der Datei-Node über ein IP-basiertes Protokoll mit Block-Nodes kommunizieren muss, müssen IPs auf der entsprechenden Schnittstelle konfiguriert werden. Für dieses installierte/konfigurierte Protokoll sind alle Pakete erforderlich.

a. Bei Verwendung von "ISCSI":

```
eseries_iscsi_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
```

b. Bei Verwendung von "ISER":

```
eseries_ib_iser_interfaces:
- name: <INTERFACE>  # Example: ib0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
  configure: true # If the file node is directly connected to the
block node set to true to setup OpenSM.
```

c. Bei Verwendung von "NVMe/IB":

```
eseries_nvme_ib_interfaces:
- name: <INTERFACE>  # Example: ib0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
  configure: true # If the file node is directly connected to the
block node set to true to setup OpenSM.
```

d. Bei Verwendung von "NVMe/RoCE":

```
eseries_nvme_roce_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
```

e. Andere Protokolle:

i. Bei Verwendung von "NVMe/FC", Die Konfiguration einzelner Schnittstellen ist nicht erforderlich. Die BeeGFS-Cluster-Implementierung erkennt das Protokoll automatisch und installiert/konfiguriert die Anforderungen nach Bedarf. Wenn Sie eine Fabric zum Verbinden von Datei- und Block-Nodes verwenden, stellen Sie sicher, dass die Switches im Rahmen der Best Practices von NetApp und dem Switch-Anbieter ordnungsgemäß begrenzt sind.

ii. Bei der Verwendung von FCP oder SAS muss keine zusätzliche Software installiert oder konfiguriert werden. Wenn Sie FCP verwenden, stellen Sie sicher, dass die Switches Folgendes ordnungsgemäß begrenzt sind "NetApp" Und die Best Practices Ihres Switch-Anbieters berücksichtigen.

iii. Die Verwendung von IB-SRP wird derzeit nicht empfohlen. NVMe/IB oder iSER nutzen, je nachdem, was die Block-Nodes der E-Series unterstützen.

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen einzelnen Dateiknoten darstellt.

Erweitert: Zwischen Ethernet- und InfiniBand-Modus können NVIDIA ConnectX VPI-Adapter eingesetzt werden

NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) Adapter unterstützen sowohl InfiniBand als auch Ethernet als Transportebene. Das Umschalten zwischen den Modi wird nicht automatisch ausgehandelt und muss mit dem in enthaltenen Werkzeug konfiguriert werden <code>mstconfig</code> <code>mstflint</code>, einem Open-Source-Paket, das Teil des ist "NVIDIA Firmare Tools (MFT)". Das Ändern des Modus der Adapter muss nur einmal vorgenommen werden. Dies kann manuell vorgenommen oder als Teil aller Schnittstellen, die mithilfe des Abschnitts des Bestands konfiguriert wurden, in das Ansible-Inventar aufgenommen <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> werden, um es automatisch prüfen/anwenden zu lassen.

So kann beispielsweise die aktuelle Schnittstellenspannung im InfiniBand-Modus in Ethernet geändert werden, damit sie für RoCE verwendet werden kann:

1. Für jede Schnittstelle, die Sie angeben möchten `mstconfig` Als Zuordnung (oder Wörterbuch), das angibt `LINK_TYPE_P<N>` Wo `<N>` Wird durch die Anschlussnummer des HCA für die Schnittstelle bestimmt. Der `<N>` Wert kann durch Ausführen bestimmt werden `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` Und fügen Sie 1 zur letzten Nummer aus dem PCI-Steckplatznamen hinzu und konvertieren Sie auf dezimal.
   a. Beispiel angegeben `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → HCA-Port 3) → `LINK_TYPE_P3: eth`:

```
eseries_roce_interfaces:
- name: <INTERFACE>
  address: <IP/SUBNET>
  mstconfig:
    LINK_TYPE_P3: eth
```

Weitere Details finden Sie im "Dokumentation der NetApp E-Series Host-Sammlung" Für den Schnittstellentyp/das Protokoll, das Sie verwenden.

Konfigurieren Einzelner Blockknoten

Legen Sie die Konfiguration für einzelne Block-Nodes mithilfe von Host-Variablen fest (Host_vars).

Überblick

In diesem Abschnitt wird das Ausfüllen von erläutert `host_vars/<BLOCK_NODE_HOSTNAME>.yml` Datei für jeden Block-Node im Cluster. Diese Dateien sollten nur die Konfiguration enthalten, die für einen bestimmten Block-Node eindeutig ist. Hierzu zählen folgende allgemein:

- Der Systemname (wie in System Manager angezeigt).
- Die HTTPS-URL für einen der Controller (wird zum Verwalten des Systems mit seiner REST-API verwendet).
- Welche Storage-Protokoll-Datei-Nodes verwenden für die Verbindung zu diesem Block-Node?

- Konfigurieren von Ports für die Host-Schnittstelle (HIC), z. B. IP-Adressen (falls erforderlich)

Schritte

Wenn Sie auf das im "Planen Sie das Dateisystem" Abschnitt definierte IP-Adressierungsschema verweisen, erstellen Sie für jeden Block-Node im Cluster eine Datei `host_vars/<BLOCK_NODE_HOSTNAME>/yml` und füllen Sie sie wie folgt aus:

1. Geben Sie oben den Systemnamen und die HTTPS-URL für einen Controller an:

```
eseries_system_name: <SYSTEM_NAME>
eseries_system_api_url:
https://<MANAGEMENT_HOSTNAME_OR_IP>:8443/devmgr/v2/
```

2. Wählen Sie die aus "Protokoll" Dateiknoten werden für die Verbindung zu diesem Block-Knoten verwendet:
   a. Unterstützte Protokolle: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. Je nach verwendetem Protokoll erfordern die HIC-Ports unter Umständen zusätzliche Konfigurationen. Bei Bedarf sollte die HIC-Port-Konfiguration definiert werden, sodass der oberste Eintrag in der Konfiguration für jeden Controller dem physischen, am meisten linken Port auf jedem Controller entspricht, und der untere Port dem fast rechten Port. Alle Ports erfordern eine gültige Konfiguration, auch wenn sie derzeit nicht verwendet werden.

   ⓘ Wenn Sie HDR (200 GB) InfiniBand oder 200 GB RoCE mit EF600 Block-Nodes verwenden, sehen Sie den folgenden Abschnitt.

   a. Für iSCSI:

```
eseries_controller_iscsi_port:
  controller_a:         # Ordered list of controller A channel
definition.
    - state:            # Whether the port should be enabled.
Choices: enabled, disabled
      config_method:    # Port configuration method Choices: static,
dhcp
      address:          # Port IPv4 address
      gateway:          # Port IPv4 gateway
      subnet_mask:      # Port IPv4 subnet_mask
      mtu:              # Port IPv4 mtu
    - (...)             # Additional ports as needed.
  controller_b:         # Ordered list of controller B channel
definition.
    - (...)             # Same as controller A but for controller B

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_iscsi_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_iscsi_port_config_method: dhcp    # General port
configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_iscsi_port_gateway:               # General port
IPv4 gateway for both controllers.
eseries_controller_iscsi_port_subnet_mask:           # General port
IPv4 subnet mask for both controllers.
eseries_controller_iscsi_port_mtu: 9000              # General port
maximum transfer units (MTU) for both controllers. Any value greater
than 1500 (bytes).
```

b. Für iSER:

```
eseries_controller_ib_iser_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

c. Für NVMe/IB:

```
eseries_controller_nvme_ib_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

d. Für NVMe/RoCE:

```
eseries_controller_nvme_roce_port:
  controller_a:          # Ordered list of controller A channel
definition.
    - state:             # Whether the port should be enabled.
      config_method:     # Port configuration method Choices: static,
dhcp
      address:           # Port IPv4 address
      subnet_mask:       # Port IPv4 subnet_mask
      gateway:           # Port IPv4 gateway
      mtu:               # Port IPv4 mtu
      speed:             # Port IPv4 speed
  controller_b:          # Ordered list of controller B channel
definition.
    - (...)              # Same as controller A but for controller B

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_nvme_roce_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_nvme_roce_port_config_method: dhcp    # General
port configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_nvme_roce_port_gateway:               # General
port IPv4 gateway for both controllers.
eseries_controller_nvme_roce_port_subnet_mask:           # General
port IPv4 subnet mask for both controllers.
eseries_controller_nvme_roce_port_mtu: 4200              # General
port maximum transfer units (MTU). Any value greater than 1500
(bytes).
eseries_controller_nvme_roce_port_speed: auto            # General
interface speed. Value must be a supported speed or auto for
automatically negotiating the speed with the port.
```

e. FC- und SAS-Protokolle erfordern keine zusätzliche Konfiguration. SRP wird nicht richtig empfohlen.

Weitere Optionen zum Konfigurieren von HIC-Ports und Hostprotokollen, einschließlich der Möglichkeit zum Konfigurieren von iSCSI-CHAP finden Sie im "Dokumentation" In der SANtricity Kollektion enthalten. Hinweis: Bei der Bereitstellung von BeeGFS werden der Speicherpool, die Volume-Konfiguration und andere Aspekte des Bereitstellungsspeicher an anderer Stelle konfiguriert und in dieser Datei nicht definiert.

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen einzelnen Block-Knoten darstellt.

Mit HDR (200 GB) InfiniBand oder 200 GB RoCE mit NetApp EF600 Block-Nodes:

Um HDR (200 GB) InfiniBand mit der EF600 zu verwenden, muss für jeden physischen Port eine zweite „virtuelle“ IP konfiguriert werden. Nachfolgend sehen Sie ein Beispiel für die korrekte Konfiguration eines EF600 mit Dual-Port InfiniBand HDR HIC:

```
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101    # Port 2a (virtual)
    - 192.168.2.101    # Port 2b (virtual)
    - 192.168.1.100    # Port 2a (physical)
    - 192.168.2.100    # Port 2b (physical)
  controller_b:
    - 192.168.3.101    # Port 2a (virtual)
    - 192.168.4.101    # Port 2b (virtual)
    - 192.168.3.100    # Port 2a (physical)
    - 192.168.4.100    # Port 2b (physical)
```

Festlegen Der Konfiguration Des Gemeinsamen Dateiknotens

Geben Sie unter Verwendung von Gruppenvariablen (Group_vars) die Konfiguration allgemeiner Dateiknoten an.

Überblick

Konfiguration, die auf alle Datei-Nodes Apfel soll, wird bei definiert `group_vars/ha_cluster.yml`. Dazu gehören in der Regel:

- Details zur Verbindung und Anmeldung zu den einzelnen Dateiknoten.
- Gängige Netzwerkkonfiguration.
- Gibt an, ob ein automatischer Neustart zulässig ist.
- Wie Firewall- und selinux-Status konfiguriert werden sollen.
- Cluster-Konfiguration mit Warn- und Fechten
- Performance-Optimierung:
- Allgemeine BeeGFS-Servicekonfiguration.

Die in dieser Datei festgelegten Optionen können auch auf einzelnen Datei-Nodes definiert werden, z. B. wenn gemischte Hardware-Modelle verwendet werden oder Sie unterschiedliche Passwörter für jeden Knoten haben. Die Konfiguration auf einzelnen Datei-Knoten hat Vorrang vor der Konfiguration in dieser Datei.

Schritte

Erstellen Sie die Datei `group_vars/ha_cluster.yml` Und füllen Sie es wie folgt aus:

1. Geben Sie an, wie sich der Ansible Control-Node mit den Remote-Hosts authentifizieren soll:

```
ansible_ssh_user: root
ansible_become_password: <PASSWORD>
```

Speichern Sie Passwörter insbesondere für Produktionsumgebungen nicht im Klartext. Verwenden Sie stattdessen Ansible Vault (siehe "Verschlüsseln von Inhalten mit Ansible Vault") Oder der `--ask-become-pass` Option beim Ausführen des Playbooks. Wenn der `ansible_ssh_user` Ist bereits root, dann können Sie optional auslassen `ansible_become_password`.

2. Wenn Sie statische IPs in ethernet- oder InfiniBand-Schnittstellen konfigurieren (zum Beispiel Cluster-IPs) und mehrere Schnittstellen im gleichen IP-Subnetz sind (z. B. wenn ib0 192.168.1.10/24 verwendet und ib1 192.168.1.11/24 verwendet), Zusätzliche IP-Routing-Tabellen und -Regeln müssen so eingerichtet sein, dass Multi-Homed-Unterstützung ordnungsgemäß funktioniert. Aktivieren Sie einfach den mitgelieferten Konfigurationshaken für Netzwerkschnittstellen wie folgt:

```
eseries_ip_default_hook_templates:
  - 99-multihoming.j2
```

3. Bei der Implementierung des Clusters müssen je nach Storage-Protokoll Nodes neu gestartet werden, um die Erkennung von Remote-Block-Geräten (E-Series Volumes) zu erleichtern oder andere Aspekte der Konfiguration anzuwenden. Standardmäßig werden vor dem Neubooten von Nodes angezeigt. Sie können Nodes jedoch durch Angabe des folgenden Verfahrens automatisch neu starten:

```
eseries_common_allow_host_reboot: true
```

   a. Standardmäßig nach einem Neustart, um sicherzustellen, dass Block-Geräte und andere Services bereit sind Ansible wartet, bis das System `default.target` Erreicht wird, bevor die Implementierung fortgesetzt wird. In manchen Szenarien, in denen NVMe/IB verwendet wird, ist dies möglicherweise nicht lang genug, um Remote-Geräte zu initialisieren, zu erkennen und eine Verbindung zu herstellen. Dies kann dazu führen, dass die automatisierte Implementierung vorzeitig ausfällt und ausfällt. Um dies bei der Nutzung von NVMe/IB zu vermeiden, müssen Sie außerdem Folgendes definieren:

```
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
```

4. Für die Kommunikation mit BeeGFS und HA-Cluster-Services sind mehrere Firewall-Ports erforderlich. Wenn Sie die Firwewall nicht manuell konfigurieren möchten (nicht empfohlen), geben Sie Folgendes an, damit erforderliche Firewall-Zonen erstellt und Ports automatisch geöffnet werden:

```
beegfs_ha_firewall_configure: True
```

5. Derzeit wird SELinux nicht unterstützt, und es wird empfohlen, den Status auf deaktiviert zu setzen, um Konflikte zu vermeiden (insbesondere, wenn RDMA verwendet wird). Stellen Sie Folgendes ein, um sicherzustellen, dass SELinux deaktiviert ist:

```
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
```

6. Konfigurieren Sie die Authentifizierung so, dass Dateiknoten kommunizieren können und passen Sie die Standardeinstellungen entsprechend Ihren Unternehmensrichtlinien an:

```
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.
```

7. "Planen Sie das Dateisystem"Legen Sie auf der Grundlage des Abschnitts die BeeGFS-Management-IP für dieses Dateisystem fest:

```
beegfs_ha_mgmtd_floating_ip: <IP ADDRESS>
```

> Während scheinbar redundant, `beegfs_ha_mgmtd_floating_ip` Ist wichtig, wenn Sie das BeeGFS-Dateisystem über einen einzelnen HA-Cluster hinaus skalieren. Nachfolgende HA-Cluster werden ohne zusätzlichen BeeGFS-Managementservice bereitgestellt und Punkt am Managementservice des ersten Clusters.

8. Aktivieren Sie bei Bedarf E-Mail-Alarme:

```
beegfs_ha_enable_alerts: True
# E-mail recipient list for notifications when BeeGFS HA resources
change or fail.
beegfs_ha_alert_email_list: ["<EMAIL>"]
# This dictionary is used to configure postfix service
(/etc/postfix/main.cf) which is required to set email alerts.
beegfs_ha_alert_conf_ha_group_options:
      # This parameter specifies the local internet domain name. This is
optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com)
      mydomain: <MY_DOMAIN>
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
```

9. Es wird dringend empfohlen, Fechten zu aktivieren, da bei Ausfall des primären Knotens Services vom Starten auf sekundären Knoten blockiert werden können.
   a. Aktivieren Sie das globale Fechten, indem Sie Folgendes angeben:

```
beegfs_ha_cluster_crm_config_options:
  stonith-enabled: True
```

      i. Hinweis: Bei Bedarf können auch alle unterstützten "Cluster-Eigenschaft" Daten hier angegeben werden. Diese Anpassungen sind in der Regel nicht notwendig, da die BeeGFS HA-Rolle mit einer Reihe gut getesteter ausgeliefert "Standardwerte"wird.

   b. Wählen Sie anschließend einen Fechten-Agent aus und konfigurieren Sie ihn:
      i. OPTION 1: Ermöglicht das Fechten mit 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>
"
```

      ii. OPTION 2: Ermöglicht das Fechten mit den vom Lenovo XCC (und anderen BMCs) bereitgestellten Redfish APIs:

```
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
```

iii. Weitere Informationen zum Konfigurieren anderer Fencing-Agenten finden Sie im "Red Hat-Dokumentation".

10. Die BeeGFS HA-Rolle kann viele verschiedene Tuning-Parameter anwenden, um die Leistung weiter zu optimieren. Dazu gehören unter anderem die Optimierung der Kernel-Speicherauslastung und die E/A von Blockgeräten. Die Rolle wird mit einem angemessenen Satz von basierend auf Tests mit NetApp E-Series Block-Nodes ausgeliefert "Standardwerte" . Diese werden standardmäßig jedoch nicht angewendet, es sei denn, Sie geben Folgendes an:

```
beegfs_ha_enable_performance_tuning: True
```

a. Geben Sie bei Bedarf auch hier Änderungen an der Standard-Performance-Optimierung an. Weitere Informationen finden Sie in der vollständigen "Parameter für die Performance-Optimierung" Dokumentation.

11. Damit schwebende IP-Adressen (manchmal auch als logische Schnittstellen bekannt), die für BeeGFS-Dienste verwendet werden, zwischen Datei-Nodes ausfallen können, müssen alle Netzwerkschnittstellen konsistent benannt werden. Standardmäßig werden Netzwerkschnittstellennamen vom Kernel generiert, was nicht garantiert ist, dass konsistente Namen generiert werden, auch bei identischen Servermodellen mit Netzwerkadaptern, die in denselben PCIe-Steckplätzen installiert sind. Dies ist auch nützlich, wenn Vorräte erstellt werden, bevor das Gerät bereitgestellt wird und generierte Schnittstellennamen bekannt sind. Um konsistente Gerätenamen auf der Grundlage eines Blockdiagramms des Servers oder sicherzustellen `lshw -class network -businfo` Ausgabe, geben Sie die gewünschte PCIe-Adresse-zu-logische Schnittstellenzuordnung wie folgt an:

a. Für InfiniBand (IPoIB)-Netzwerkschnittstellen:

```
eseries_ipoib_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:01:00.0: i1a
```

b. Bei Ethernet-Netzwerkschnittstellen:

```
eseries_ip_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:01:00.0: e1a
```

Um Konflikte zu vermeiden, wenn Schnittstellen umbenannt werden (um zu verhindern, dass sie umbenannt werden), sollten Sie keine möglichen Standardnamen wie eth0, ens9f0, ib0 oder ibs4f0 verwenden. Eine häufige Namenskonvention besteht darin, „e“ oder „i“ für Ethernet oder InfiniBand zu verwenden, gefolgt von der PCIe-Steckplatznummer und einem Buchstaben zur Angabe des Ports. Zum Beispiel wäre der zweite Port eines InfiniBand-Adapters, der in Steckplatz 3 installiert ist: i3b.

Wenn Sie ein verifiziertes Datei-Node-Modell verwenden, klicken Sie auf "Hier" Beispiel für Zuordnungen von PCIe-Adressen zu logischen Ports

12. Geben Sie optional die Konfiguration an, die für alle BeeGFS-Dienste im Cluster gelten soll. Die Standardkonfigurationswerte können gefunden werden "Hier", und die Konfiguration pro Service wird an anderer Stelle angegeben:

    a. BeeGFS Management-Service:

    ```
    beegfs_ha_beegfs_mgmtd_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

    b. BeeGFS Metadata Services:

    ```
    beegfs_ha_beegfs_meta_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

    c. BeeGFS Storage-Services:

    ```
    beegfs_ha_beegfs_storage_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

13. Ab BeeGFS 7.2.7 und 7.3.1 "Verbindungsauthentifizierung" Muss konfiguriert oder explizit deaktiviert werden. Es gibt einige Konfigurationsmöglichkeiten, die mit der Ansible-basierten Implementierung konfiguriert werden können:

    a. Standardmäßig konfiguriert die Bereitstellung die Verbindungsauthentifizierung automatisch und erstellt ein `connauthfile` Die auf alle Datei-Nodes verteilt und mit den BeeGFS-Diensten verwendet werden. Diese Datei wird auch auf dem Ansible-Steuerungsknoten in abgelegt/gepflegt `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` Wo Sie Daten für die Wiederverwendung mit Clients, die auf dieses Filesystem zugreifen müssen, aufbewahren (sicher).

        i. Zum Generieren eines neuen Schlüssels angeben `-e "beegfs_ha_conn_auth_force_new=True` Wenn Sie das Ansible-Playbook ausführen. Beachten Sie, dass dies bei einem ignoriert wird `beegfs_ha_conn_auth_secret` Definiert ist.

        ii. Weitere Optionen finden Sie in der vollständigen Liste der Standardwerte, die im enthalten

"BeeGFS HA-Rolle"sind.

b. Ein benutzerdefiniertes Geheimnis kann verwendet werden, indem Sie Folgendes in definieren `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. Die Verbindungsauthentifizierung kann vollständig deaktiviert werden (NICHT empfohlen):

```
beegfs_ha_conn_auth_enabled: false
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die die allgemeine Konfiguration des Dateiknoten darstellt.

Verwendung von HDR (200 GB) InfiniBand mit NetApp EF600 Block-Nodes:

Um HDR (200 GB) InfiniBand mit der EF600 zu verwenden, muss der Subnetzmanager die Virtualisierung unterstützen. Wenn Datei- und Block-Knoten über einen Switch verbunden sind, muss dies im Subnetz Manager für die Gesamtstruktur aktiviert sein.

Wenn Block- und Datei-Nodes direkt über InfiniBand verbunden sind, `opensm` muss auf jedem Datei-Node für jede Schnittstelle, die direkt mit einem Block-Node verbunden ist, eine Instanz von konfiguriert werden. Dies geschieht durch Angabe von `configure: true` wann "Konfigurieren von File-Node-Storage-Schnittstellen".

Derzeit unterstützt die Inbox-Version von `opensm` , die mit unterstützten Linux-Distributionen ausgeliefert wurde, keine Virtualisierung. Stattdessen ist es erforderlich, dass Sie die Version von über die NVIDIA OpenFabrics Enterprise Distribution (OFED) installieren und konfigurieren `opensm` . Obwohl die Implementierung mit Ansible weiterhin unterstützt wird, sind einige weitere Schritte erforderlich:

1. Laden Sie die Pakete für die Version von OpenSM, die im Abschnitt von der NVIDIA-Website aufgeführt sind, mithilfe von Curl oder Ihrem gewünschten Tool in das Verzeichnis herunter "Technologieanforderungen erfüllt" `<INVENTORY>/packages/` . Beispiel:

```
curl -o packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
curl -o packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
```

2. Unter `group_vars/ha_cluster.yml` Definieren Sie die folgende Konfiguration:

```
### 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-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
          "packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
    - packages:
        add:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
  uninstall:
    - packages:
        remove:
          - opensm
          - opensm-libs
      files:
        remove:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm

eseries_ib_opensm_options:
  virt_enabled: "2"
```

Festlegen Der Konfiguration Allgemeiner Blockknoten

Geben Sie unter Verwendung von Gruppenvariablen (Group_vars) die allgemeine Konfiguration von Blockknoten an.

Überblick

Die Konfiguration, die auf alle Block-Nodes Apfel soll, wird auf definiert `group_vars/eseries_storage_systems.yml`. Dazu gehören in der Regel:

- Details dazu, wie der Ansible-Kontroll-Node mit als Block-Nodes verwendeten E-Series Storage-Systemen verbunden werden soll.

- Welche Firmware-, NVSRAM- und Laufwerk-Firmware-Versionen die Nodes ausführen sollten.
- Globale Konfiguration einschließlich Cache-Einstellungen, Host-Konfiguration und Einstellungen für die Bereitstellung von Volumes

> Die in dieser Datei festgelegten Optionen können auch auf einzelnen Block-Nodes definiert werden, z. B. wenn gemischte Hardware-Modelle verwendet werden oder Sie unterschiedliche Passwörter für jeden Knoten haben. Die Konfiguration auf einzelnen Block-Knoten hat Vorrang vor der Konfiguration in dieser Datei.

Schritte

Erstellen Sie die Datei `group_vars/eseries_storage_systems.yml` Und füllen Sie es wie folgt aus:

1. Ansible verwendet SSH nicht für die Verbindung mit Block-Nodes und verwendet stattdessen REST-APIs. Um dies zu erreichen, müssen wir Folgendes festlegen:

```
ansible_connection: local
```

2. Geben Sie den Benutzernamen und das Passwort an, um jeden Knoten zu verwalten. Der Benutzername kann optional ausgelassen werden (und wird standardmäßig auf admin gesetzt), andernfalls können Sie jedes Konto mit Administratorrechten angeben. Geben Sie außerdem an, ob SSL-Zertifikate überprüft oder ignoriert werden sollen:

```
eseries_system_username: admin
eseries_system_password: <PASSWORD>
eseries_validate_certs: false
```

> Es wird nicht empfohlen, Kennwörter im Klartext zu verwenden. Verwenden Sie einen Ansible-Vault, oder stellen Sie die bereit `eseries_system_password` Wenn Sie Ansible mit --extra-Vars verwenden.

3. Geben Sie optional an, welche Controller-Firmware, NVSRAM und Laufwerk-Firmware auf den Nodes installiert werden soll. Diese müssen auf das heruntergeladen werden `packages/` Verzeichnis vor der Ausführung von Ansible. NVSRAM und E-Series Controller Firmware können heruntergeladen werden "Hier" Und Laufwerk-Firmware "Hier":

```
eseries_firmware_firmware: "packages/<FILENAME>.dlp" # Ex.
"packages/RCB_11.80GA_6000_64cc0ee3.dlp"
eseries_firmware_nvsram: "packages/<FILENAME>.dlp" # Ex.
"packages/N6000-880834-D08.dlp"
eseries_drive_firmware_firmware_list:
  - "packages/<FILENAME>.dlp"
  # Additional firmware versions as needed.
eseries_drive_firmware_upgrade_drives_online: true # Recommended unless
BeeGFS hasn't been deployed yet, as it will disrupt host access if set
to "false".
```

Wenn diese Konfiguration angegeben wird, aktualisiert Ansible automatisch alle Firmware einschließlich des Neubootens von Controllern (falls erforderlich), ohne zusätzliche Eingabeaufforderungen. Dies wird für BeeGFS/Host-I/O voraussichtlich ohne Unterbrechung ausgeführt, kann jedoch zu einer vorübergehenden Abnahme der Performance führen.

4. Passen Sie die Standardeinstellungen für die globale Systemkonfiguration an. Die hier aufgeführten Optionen und Werte werden häufig für BeeGFS auf NetApp empfohlen, können jedoch bei Bedarf angepasst werden:

```
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 by default.
```

5. Konfigurieren Sie die Standardeinstellungen für die globale Volume-Bereitstellung. Die hier aufgeführten Optionen und Werte werden häufig für BeeGFS auf NetApp empfohlen, können jedoch bei Bedarf angepasst werden:

```
eseries_volume_size_unit: pct # Required by default. This allows volume
capacities to be specified as a percentage, simplifying putting together
the inventory.
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
```

6. Passen Sie bei Bedarf die Reihenfolge an, in der Ansible Laufwerke für Storage Pools und Volume-Gruppen wählt, und berücksichtigen Sie die folgenden Best Practices:

   a. Nennen Sie alle (möglicherweise kleineren) Laufwerke, die zuerst für Management- und/oder Metadaten-Volumes verwendet werden sollten, und Storage Volumes zuletzt.

b. Stellen Sie sicher, dass Sie die Reihenfolge der Festplattenauswahl auf der Grundlage der Modelle für Festplatten-Shelfs/Festplattengehäuse ausgleichen, um die Laufwerksauswahl über verfügbare Laufwerkskanäle auszugleichen. Beispielsweise befinden sich Laufwerke 0-11 mit der EF600 und ohne Erweiterungen auf Laufwerkskanal 1 und Laufwerke 12-23 auf dem Laufwerkskanal. Daher ist eine Strategie zur Balance der Antriebsauswahl zu wählen `disk shelf:drive` 99:0, 99:23, 99:1, 99:22 usw. Wenn mehr als ein Gehäuse vorhanden ist, steht die erste Ziffer für die Laufwerk-Shelf-ID.

```
# Optimal/recommended order for the EF600 (no expansion):
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"
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die die allgemeine Block-Node-Konfiguration darstellt.

BeeGFS-Dienste definieren

Definieren Sie den BeeGFS-Managementdienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Managementservice erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte nur ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Der Servicetyp (Management).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Geben Sie an, wo/wie ein Volume Daten für diesen Service speichern soll (das BeeGFS-Managementziel).

Schritte

Erstellen Sie eine neue Datei `group_vars/mgmt.yml`, und verweisen Sie auf den "Planen Sie das Dateisystem" Abschnitt. Füllen Sie diese wie folgt aus:

1. Geben Sie diese Datei an, um die Konfiguration für einen BeeGFS-Managementdienst anzuzeigen:

```
beegfs_service: management
```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Dies ist in der Regel nicht für den Management-Service erforderlich, es sei denn, Sie müssen Quoten aktivieren, jedoch alle unterstützten Konfigurationsparameter von `beegfs-mgmtd.conf` Kann enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, und `connNetFilterFile`.

```
beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
```

3. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.0/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. Geben Sie das BeeGFS-Managementziel an, auf dem dieser Service Daten gemäß den folgenden Richtlinien speichert:

   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).

   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).

   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_m1_m2_m5_m6
        raid_level: <LEVEL> # One of: raid1, raid5, raid6, raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Managementdienst darstellt.

Definieren Sie den BeeGFS-Metadatendienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Metadatendienstes erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte mindestens ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Der Servicetyp (Metadaten).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Festlegen, wo/wie ein Volume Daten für diesen Service speichern soll (das BeeGFS-Metadatenziel).

Schritte

"Planen Sie das Dateisystem"Erstellen Sie `group_vars/meta_<ID>.yml` für jeden Metadatendienst im Cluster eine Datei unter, und füllen Sie sie wie folgt aus, um auf den Abschnitt zu verweisen:

1. Geben Sie an, dass diese Datei die Konfiguration für einen BeeGFS-Metadatendienst darstellt:

   ```
   beegfs_service: metadata
   ```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Mindestens müssen Sie den gewünschten TCP- und UDP-Port angeben, jedoch alle unterstützten Konfigurationsparameter von `beegfs-meta.conf` Kann ebenfalls enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`,

`connInterfacesFile`, und `connNetFilterFile`.

```
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <TCP PORT>
  connMetaPortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.1/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. Geben Sie das BeeGFS-Metadatenziel an, bei dem dieser Dienst Daten gemäß den folgenden Richtlinien speichert (dies konfiguriert auch automatisch den `storeMetaDirectory` Option):

   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).

   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).

   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_m1_m2_m5_m6
        raid_level: <LEVEL> # One of: raid1, raid5, raid6, raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Metadatendienst darstellt.

Definieren Sie den BeeGFS-Speicherdienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Speicherdienstes erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte mindestens ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Den Servicetyp (Storage).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Geben Sie an, wo/wie Volumen(en) Daten für diesen Dienst speichern sollen (die BeeGFS-Speicherziele).

Schritte

"Planen Sie das Dateisystem"Erstellen Sie `group_vars/stor_<ID>.yml` für jeden Storage-Service im Cluster eine Datei unter, und füllen Sie sie wie folgt aus, um auf den Abschnitt Bezug zu nehmen:

1. Geben Sie diese Datei für die Konfiguration eines BeeGFS-Speicherdienstes an:

   ```
   beegfs_service: storage
   ```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Mindestens müssen Sie den gewünschten TCP- und UDP-Port angeben, jedoch alle unterstützten Konfigurationsparameter von `beegfs-storage.conf` Kann ebenfalls enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, und `connNetFilterFile`.

```
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <TCP PORT>
  connStoragePortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.1/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. Geben Sie die BeeGFS-Speicherziele an, in denen dieser Service Daten gemäß den folgenden Richtlinien speichert (dies konfiguriert auch automatisch den `storeStorageDirectory` Option):
   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).
   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).
   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_s1_s2
        raid_level: <LEVEL> # One of: raid1, raid5, raid6,
raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
        # Multiple storage targets are supported / typical:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Speicherdienst darstellt.

Zuordnen von BeeGFS-Services zu Datei-Nodes

Geben Sie an, welche Datei-Nodes jeden BeeGFS-Dienst mit dem ausführen können `inventory.yml` Datei:

Überblick

In diesem Abschnitt wird die Erstellung des erläutert `inventory.yml` Datei: Dazu gehören alle Block-Nodes und die Angabe, welche Datei-Nodes jeden BeeGFS-Service ausführen können.

Schritte

Erstellen Sie die Datei `inventory.yml` Und füllen Sie es wie folgt aus:

1. Erstellen Sie oben in der Datei die Ansible-Standardinventarstruktur:

```
# BeeGFS HA (High_Availability) cluster inventory.
all:
  children:
```

2. Erstellen Sie eine Gruppe mit allen Block-Nodes, die an diesem HA-Cluster teilnehmen:

```
# Ansible group representing all block nodes:
eseries_storage_systems:
  hosts:
    <BLOCK NODE HOSTNAME>:
    <BLOCK NODE HOSTNAME>:
    # Additional block nodes as needed.
```

3. Erstellen Sie eine Gruppe, die alle BeeGFS-Dienste im Cluster und die Datei-Nodes enthält, auf denen sie ausgeführt werden:

```
# Ansible group representing all file nodes:
ha_cluster:
  children:
```

4. Definieren Sie für jeden BeeGFS-Service im Cluster die bevorzugten und sekundären Dateiknoten, die diesen Service ausführen sollen:

```
    <SERVICE>: # Ex. "mgmt", "meta_01", or "stor_01".
      hosts:
        <FILE NODE HOSTNAME>:
        <FILE NODE HOSTNAME>:
        # Additional file nodes as needed.
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei.

Stellen Sie das BeeGFS-Dateisystem bereit

Ansible – Playbook-Überblick

BeeGFS HA-Cluster implementieren und managen mithilfe von Ansible

Überblick

In den vorherigen Abschnitten wurden die Schritte aufgeführt, mit denen ein Ansible-Inventar erstellt werden konnte, der ein BeeGFS HA-Cluster darstellt. In diesem Abschnitt wird die von NetApp entwickelte Ansible-Automatisierung für die Implementierung und das Management des Clusters vorgestellt.

Ansible – Wichtige Konzepte

Bevor Sie fortfahren, ist es hilfreich, sich mit ein paar Schlüsselkonzepten von Ansible vertraut zu machen:

- Aufgaben, die für einen Ansible-Bestand ausgeführt werden müssen, werden in einem sogenannten **Playbook** definiert.
  - Die meisten Aufgaben in Ansible sind **idempotent**, d. h., sie können mehrmals ausgeführt werden, um zu überprüfen, ob die gewünschte Konfiguration/der gewünschte Zustand noch angewendet wird, ohne

dass Dinge zu stören oder unnötige Updates zu machen.

- Die kleinste Ausführungseinheit in Ansible ist ein **Modul**.
  - Typische Playbooks nutzen mehrere Module.
    - Beispiele: Laden Sie ein Paket herunter, aktualisieren Sie eine Konfigurationsdatei, starten/aktivieren Sie einen Dienst.
  - NetApp verteilt Module zur Automatisierung von NetApp E-Series Systemen.
- Komplexe Automatisierungsoptionen sind besser als Rollen integriert.
  - Im Wesentlichen ein Standardformat zur Verteilung eines wiederverwendbaren Playbooks.
  - NetApp verteilt Rollen für Linux-Hosts und BeeGFS-Filesysteme.

BeeGFS HA-Rolle für Ansible: Schlüsselkonzepte

Die gesamte Automatisierung, die für das Implementieren und Managen jeder Version von BeeGFS auf NetApp erforderlich ist, ist als Ansible-Rolle verpackt und im Rahmen der verteilt "NetApp E-Series Ansible Collection für BeeGFS":

- Diese Rolle kann als irgendwo zwischen einem **Installer** und einer modernen **Deployment/Management** Engine für BeeGFS gedacht werden.
  - Nutzt moderne Infrastruktur als Code-Praktiken und -Philosophien um das Management der Storage-Infrastruktur in jeder Größenordnung zu vereinfachen
  - Das "Kubesbete"Projekt ermöglicht Benutzern die Implementierung und Wartung einer gesamten Kubernetes-Distribution für die Scale-out-Computing-Infrastruktur.
- Dabei handelt es sich um das **softwaredefinierte**-Format von NetApp zur Verpackung, Verteilung und Wartung von BeeGFS auf NetApp Lösungen.
  - Versuchen Sie, eine „Appliance-ähnliche“ Erfahrung zu schaffen, ohne eine gesamte Linux-Distribution oder ein großes Bild zu verteilen.
  - Dazu gehören die von NetApp entwickelten Open Cluster Framework (OCF)-konformen Cluster-Ressourcen-Agents für benutzerdefinierte BeeGFS-Ziele, IP-Adressen und Monitoring für intelligente Pacemaker/BeeGFS-Integration.
- Diese Rolle spielt nicht einfach die Implementierung der „Automatisierung“ und soll den gesamten Lebenszyklus des Filesystems managen, darunter:
  - Konfigurationsänderungen und Updates pro Service oder Cluster-weite Konfiguration
  - Automatisierung von Cluster-Reparatur und Recovery nach Behebung von Hardware-Problemen
  - Vereinfachte Performance-Optimierung mit Standardeinstellungen, die auf umfangreichen Tests mit BeeGFS und NetApp Volumes basieren
  - Überprüfung und Korrektur von Konfigurationstendenzen.

NetApp bietet auch eine Ansible-Rolle für "BeeGFS-Clients", Die optional verwendet werden kann, um BeeGFS zu installieren und mounten Sie Dateisysteme auf Compute/GPU/Login-Nodes.

Implementieren Sie das BeeGFS HA-Cluster

Geben Sie an, welche Aufgaben ausgeführt werden sollen, um BeeGFS HA-Cluster mithilfe eines Playbooks zu implementieren.

Überblick

In diesem Abschnitt wird beschrieben, wie Sie das Standard-Playbook zur Bereitstellung/zum Managen von BeeGFS auf NetApp zusammenstellen können.

Schritte

Erstellen Sie das Ansible Playbook

Erstellen Sie die Datei `playbook.yml` Und füllen Sie es wie folgt aus:

1. Definieren Sie zunächst einen Satz von Aufgaben (allgemein als A bezeichnet) "Spielen") Die nur auf Block-Nodes der NetApp E-Series ausgeführt werden sollte. Wir verwenden eine Pause-Aufgabe, um vor dem Ausführen der Installation eine Aufforderung zu geben (um versehentliche Playbook-Läufe zu vermeiden), und importieren dann die `nar_santricity_management` Rolle: Diese Rolle übernimmt die Anwendung aller in definierten allgemeinen Systemkonfiguration `group_vars/eseries_storage_systems.yml` Oder einzeln `host_vars/<BLOCK NODE>.yml` Dateien:

```
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  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: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

2. Definieren Sie die Wiedergabe, die für alle Datei- und Blockknoten ausgeführt wird:

```
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
```

3. In diesem Sales Play können wir optional einen Satz von „Voraufgaben“ definieren, die vor der Bereitstellung des HA-Clusters ausgeführt werden sollten. Dies kann nützlich sein, um alle Voraussetzungen wie Python zu überprüfen/zu installieren. Zudem können Überprüfungen vor dem Flug durchgeführt werden, beispielsweise die Unterstützung der bereitgestellten Ansible-Tags:

```
```

```
  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 }}"
```

4. Schließlich importiert dieses Spiel die BeeGFS HA-Rolle für die Version von BeeGFS, die Sie bereitstellen möchten:

```
tasks:
  - name: Verify the BeeGFS HA cluster is properly deployed.
    import_role:
      name: beegfs_ha_7_4 # Alternatively specify: beegfs_ha_7_3.
```

> Für jede unterstützte Major.Minor Version von BeeGFS wird eine BeeGFS HA-Rolle beibehalten. Auf diese Weise können Benutzer festlegen, wann ein Upgrade von Major-/Minor-Versionen durchgeführt werden soll. Derzeit (`beegfs_7_3`(`beegfs_7_2`werden BeeGFS 7.3.x) oder BeeGFS 7.2.x) unterstützt. Standardmäßig werden beide Rollen zum Zeitpunkt der Veröffentlichung die neueste BeeGFS-Patch-Version bereitstellen. Benutzer können dies jedoch überschreiben und gegebenenfalls den neuesten Patch bereitstellen. "Upgrade-Leitfaden"Weitere Informationen finden Sie auf dem neuesten Stand.

5. Optional: Wenn Sie zusätzliche Aufgaben definieren möchten, sollten Sie beachten, ob die Aufgaben an geleitet werden sollen `all` Hosts (einschließlich der E-Series Storage-Systeme) oder nur die Datei-Nodes Definieren Sie bei Bedarf ein neues Spiel speziell für Dateiknoten mit `- hosts: ha_cluster`.

Klicken Sie Auf "Hier" Beispiel für eine vollständige Playbook-Datei.

NetApp Ansible Sammlungen installieren

Die BeeGFS-Sammlung für Ansible, und alle Abhängigkeiten werden aufrechterhalten "Ansible-Galaxie". Führen Sie auf Ihrem Ansible-Steuerungsknoten den folgenden Befehl aus, um die neueste Version zu installieren:

```
ansible-galaxy collection install netapp_eseries.beegfs
```

Obwohl nicht in der Regel empfohlen, ist es auch möglich, eine bestimmte Version der Sammlung zu installieren:

```
ansible-galaxy collection install netapp_eseries.beegfs:
==<MAJOR>.<MINOR>.<PATCH>
```

Führen Sie das Playbook aus

Aus dem Verzeichnis auf Ihrem Ansible-Steuerungsknoten, der den enthält `inventory.yml` Und `playbook.yml` Dateien, führen Sie das Playbook wie folgt aus:

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

Basierend auf der Cluster-Größe kann die ursprüngliche Implementierung 20+ Minuten dauern. Wenn die Implementierung aus irgendeinem Grund fehlschlägt, korrigieren Sie einfach Probleme (z. B. Fehlverkabelung, Knoten wurde nicht gestartet usw.) und starten Sie das Ansible Playbook neu.

"Allgemeine Konfiguration der Datei-Nodes"Wenn Sie angeben , wenn Sie die Standardoption wählen, damit Ansible die verbindungsbasierte Authentifizierung automatisch verwaltet, `connAuthFile` kann ein als gemeinsamer Schlüssel verwendet jetzt unter `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (standardmäßig) gefunden werden. Alle Clients, die auf das Dateisystem zugreifen müssen, müssen diesen gemeinsam genutzten Schlüssel verwenden. Dies wird automatisch verarbeitet, wenn Clients über die konfiguriert werden"BeeGFS-Client-Rolle".

Bereitstellen von BeeGFS-Clients

Optional kann Ansible verwendet werden, um BeeGFS-Clients zu konfigurieren und das Dateisystem zu mounten.

Überblick

Für den Zugriff auf BeeGFS-Dateisysteme muss der BeeGFS-Client auf jedem Node installiert und konfiguriert werden, der das Dateisystem bereitstellen muss. In diesem Abschnitt wird beschrieben, wie Sie diese Aufgaben mit dem verfügbaren ausführen "Ansible-Rolle".

Schritte

Erstellen Sie die Client-Bestandsdatei

1. Richten Sie bei Bedarf über den Ansible-Steuerungsknoten passwortlose SSH für jeden Host ein, den Sie als BeeGFS-Clients konfigurieren möchten:

   ```
   ssh-copy-id <user>@<HOSTNAME_OR_IP>
   ```

2. Unter `host_vars/`Erstellen Sie für jeden BeeGFS-Client eine Datei mit dem Namen `<HOSTNAME>.yml` Füllen Sie den Platzhaltertext mit den korrekten Informationen für Ihre Umgebung aus:

   ```
   # BeeGFS Client
   ansible_host: <MANAGEMENT_IP>
   ```

3. Geben Sie optional eine der folgenden Optionen ein, wenn Sie die Rollen der NetApp E-Series Host Collection verwenden möchten, um InfiniBand- oder Ethernet-Schnittstellen für Clients zu konfigurieren, damit eine Verbindung mit BeeGFS File-Nodes hergestellt werden kann:

   a. Wenn der Netzwerktyp ist "InfiniBand (mit IPoIB)":

```
eseries_ipoib_interfaces:
- name: <INTERFACE>  # Example: ib0 or i1b
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

b. Wenn der Netzwerktyp ist "RDMA über Converged Ethernet (RoCE)":

```
eseries_roce_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

c. Wenn der Netzwerktyp ist "Ethernet (nur TCP, kein RDMA)":

```
eseries_ip_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

4. Erstellen Sie eine neue Datei `client_inventory.yml` Geben Sie den Benutzer an, den Ansible für die Verbindung mit den einzelnen Clients verwenden soll, und das Passwort, das Ansible zur Eskalation von Berechtigungen verwenden soll (dies erfordert `ansible_ssh_user` „Root“ oder „Sudo“-Berechtigungen besitzen):

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER>
    ansible_become_password: <PASSWORD>
```

Speichern Sie Passwörter nicht im Klartext. Verwenden Sie stattdessen den Ansible-Vault (siehe "Ansible-Dokumentation" Für Inhalte mit Ansible Vault) oder verwenden Sie den `--ask-become-pass` Option beim Ausführen des Playbooks.

5. Im `client_inventory.yml` Datei, Listen Sie alle Hosts auf, die als BeeGFS-Clients unter dem konfiguriert werden sollen `beegfs_clients` Gruppe, und dann beachten Sie die Inline-Kommentare und Uncomment zusätzliche Konfiguration erforderlich, um das BeeGFS-Client-Kernel-Modul auf Ihrem System zu erstellen:

```
children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        <CLIENT HOSTNAME>:
        # Additional clients as needed.

      vars:
        # OPTION 1: If you're using the NVIDIA 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.
```

Wenn Sie die NVIDIA OFED-Treiber verwenden, stellen Sie sicher, dass beegfs_Client_ofed_include_PATH auf den korrekten „Header include path“ für Ihre Linux-Installation verweist. Weitere Informationen finden Sie in der BeeGFS-Dokumentation für "RDMA-Unterstützung".

6. Im `client_inventory.yml` Datei, Listen Sie die BeeGFS-Dateisysteme auf, die Sie unter einem zuvor definierten gemountet haben möchten `vars`:

```
        beegfs_client_mounts:
          - sysMgmtdHost: <IP ADDRESS>  # 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
        # Specify additional file system mounts for this or other file
systems.
```

7. Ab BeeGFS 7.2.7 und 7.3.1 "Verbindungsauthentifizierung" müssen konfiguriert oder explizit deaktiviert sein. Je nachdem, wie Sie die verbindungsbasierte Authentifizierung bei der Angabe konfigurieren"Allgemeine Konfiguration der Datei-Nodes", müssen Sie möglicherweise Ihre Clientkonfiguration anpassen:

   a. Standardmäßig konfiguriert die HA-Cluster-Implementierung die Verbindungsauthentifizierung automatisch und generiert einen `connauthfile` Die auf dem Ansible-Kontroll-Node bei platziert/gewartet werden `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Standardmäßig ist die BeeGFS-Client-Rolle so eingerichtet, dass sie diese Datei an die in definierten Clients liest/verteilt `client_inventory.yml`, Und es ist keine zusätzliche Aktion erforderlich.

      i. Weitere Optionen finden Sie in der vollständigen Liste der Standardwerte, die im enthalten sind "BeeGFS-Client-Rolle".

   b. Wenn Sie ein benutzerdefiniertes Geheimnis mit angeben `beegfs_ha_conn_auth_secret` Geben Sie ihn im an `client_inventory.yml` Außerdem:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Wenn Sie die verbindungsbasierte Authentifizierung vollständig mit deaktivieren `beegfs_ha_conn_auth_enabled`, Geben Sie das im an `client_inventory.yml` Außerdem:

```
beegfs_ha_conn_auth_enabled: false
```

Eine vollständige Liste der unterstützten Parameter und weitere Details finden Sie im "Vollständige BeeGFS-Client-Dokumentation". Klicken Sie für ein vollständiges Beispiel eines Clientbestands auf "Hier".

Erstellen Sie die BeeGFS Client Playbook-Datei

1. Erstellen Sie eine neue Datei `client_playbook.yml`

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
```

2. Optional: Wenn Sie die Rollen der NetApp E-Series Host Collection verwenden möchten, um Schnittstellen für Clients zu konfigurieren, mit denen sich eine Verbindung zu BeeGFS-Dateisystemen herstellen lässt, importieren Sie die Rolle entsprechend dem Schnittstellentyp, den Sie konfigurieren:
   a. Wenn Sie InfiniBand (IPoIB) verwenden:

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
```

   b. Bei Verwendung von RDMA over Converged Ethernet (RoCE):

```
    - name: Ensure IPoIB is configured
      import_role:
        name: roce
```

   c. Wenn Sie Ethernet verwenden (nur TCP, kein RDMA):

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ip
```

3. Schließlich importieren Sie die BeeGFS-Client-Rolle, um die Client-Software zu installieren und das Dateisystem-Mounts einzurichten:

```
    # REQUIRED: Install the BeeGFS client and mount the BeeGFS file
system.
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

Klicken Sie zum vollständigen Beispiel für ein Client-Playbook auf "Hier".

Führen Sie das BeeGFS Client Playbook aus

Führen Sie den folgenden Befehl aus, um den Client zu installieren/zu erstellen und BeeGFS zu mounten:

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

Überprüfen Sie die BeeGFS-Bereitstellung

Überprüfen Sie die Bereitstellung des Dateisystems, bevor Sie das System in die Produktion bringen.

Überblick

Bevor Sie das BeeGFS-Dateisystem in Produktion setzen, führen Sie einige Überprüfungsprüfungen durch.

Schritte

1. Melden Sie sich bei einem Client an und führen Sie Folgendes aus, um sicherzustellen, dass alle erwarteten Knoten vorhanden/erreichbar sind, und es werden keine Inkonsistenzen oder andere Probleme gemeldet:

   ```
   beegfs-fsck --checkfs
   ```

2. Fahren Sie den gesamten Cluster herunter und starten Sie ihn dann neu. Führen Sie von jedem beliebigen Dateiknoten Folgendes aus:

   ```
   pcs cluster stop --all # Stop the cluster on all file nodes.
   pcs cluster start --all  # Start the cluster on all file nodes.
   pcs status # Verify all nodes and services are started and no failures
   are reported (the command may need to be reran a few times to allow time
   for all services to start).
   ```

3. Platzieren Sie jeden Knoten in den Standby-Modus, und überprüfen Sie, ob BeeGFS-Dienste einen Failover auf sekundäre Knoten ausführen können. So melden Sie sich an einem beliebigen Dateiknoten an und führen Sie Folgendes aus:

```
pcs status # Verify the cluster is healthy at the start.
pcs node standby <FILE NODE HOSTNAME> # Place the node under test in
standby.
pcs status # Verify services are started on a secondary node and no
failures are reported.
pcs node unstandby <FILE NODE HOSTNAME> # Take the node under test out
of standby.
pcs status # Verify the file node is back online and no failures are
reported.
pcs resource relocate run # Move all services back to their preferred
nodes.
pcs status # Verify services have moved back to the preferred node.
```

4. Verwenden Sie Tools zum Leistungsvergleich wie IOR und MDTest, um zu überprüfen, ob die Performance des Dateisystems den Erwartungen entspricht. Beispiele für häufige Tests und Parameter, die mit BeeGFS verwendet werden"Design-Verifizierung", finden Sie im Abschnitt BeeGFS on NetApp Verified Architecture.

Zusätzliche Tests sollten auf Grundlage der Abnahmekriterien durchgeführt werden, die für einen bestimmten Standort/eine bestimmte Installation definiert sind.

BeeGFS-Cluster verwalten

Übersicht, Schlüsselkonzepte und Terminologie

Lesen Sie, wie BeeGFS HA-Cluster nach der Implementierung verwaltet werden.

Überblick

Dieser Abschnitt richtet sich an Cluster-Administratoren, die BeeGFS HA-Cluster nach ihrer Bereitstellung verwalten müssen. Selbst diejenigen, die mit Linux HA-Clustern vertraut sind, sollten diesen Leitfaden genau lesen, da es verschiedene Unterschiede beim Management des Clusters gibt, insbesondere im Hinblick auf die Neukonfiguration aufgrund der Verwendung von Ansible.

Schlüsselkonzepte

Während einige dieser Konzepte auf der Hauptseite vorgestellt werden"Begriffe und Konzepte", ist es hilfreich, sie im Kontext eines BeeGFS HA-Clusters neu einzuführen:

Cluster Node: Ein Server, auf dem Pacemaker- und Corosync-Dienste ausgeführt und am HA-Cluster beteiligt sind.

Datei-Node: Ein Clusterknoten, mit dem ein oder mehrere BeeGFS-Management-, Metadaten- oder Storage-Services ausgeführt werden.

Block-Node: Ein Storage-System der NetApp E-Series, das Block-Storage für Datei-Nodes bereitstellt. Diese Nodes nehmen nicht am BeeGFS HA-Cluster Teil, da sie eigene Standalone-HA-Funktionen bereitstellen. Jeder Node besteht aus zwei Storage Controllern, die auf Blockebene Hochverfügbarkeit bieten.

BeeGFS-Service: Ein BeeGFS-Management, Metadaten- oder Speicherservice. Auf jedem Datei-Node wird ein oder mehrere Services ausgeführt, die Volumes auf dem Block-Node zum Speichern ihrer Daten verwenden.

Baustein: Eine standardisierte Implementierung von BeeGFS-Datei-Nodes, E-Series Block-Nodes und auf ihnen ausgeführten BeeGFS-Services zur Vereinfachung der Skalierung eines BeeGFS HA-Clusters/-Filesystems nach einer NetApp Verified Architecture. Kundenspezifische HA-Cluster werden ebenfalls unterstützt, verfolgen jedoch oft einen ähnlichen Bausteinansatz, der die Skalierung vereinfacht.

BeeGFS HA Cluster: Eine skalierbare Anzahl von Datei-Nodes, die für die Ausführung von BeeGFS-Diensten verwendet werden, die von Block-Nodes gesichert werden, um BeeGFS-Daten auf hochverfügbare Weise zu speichern. Basiert auf bewährten Open-Source-Komponenten Pacemaker und Corosync mit Ansible für Verpackung und Bereitstellung.

Cluster Services: bezieht sich auf Pacemaker- und Corosync-Dienste, die auf jedem Knoten ausgeführt werden, der am Cluster teilnimmt. Hinweis: Es ist möglich, dass ein Node keine BeeGFS-Services ausführen kann und nur als „Tiebreaker“-Node im Cluster teilnimmt, wenn es nur zwei Datei-Nodes benötigt.

Cluster-Ressourcen: für jeden BeeGFS-Dienst, der im Cluster ausgeführt wird, wird eine BeeGFS-Monitorressource und eine Ressourcengruppe mit Ressourcen für BeeGFS-Ziele, IP-Adressen (fließende IPs) und den BeeGFS-Service selbst angezeigt.

Ansible: Ein Tool für die Softwarebereitstellung, das Konfigurationsmanagement und den Applikationseinsatz, das Infrastruktur als Code ermöglicht. Die Pakete von BeeGFS HA-Clustern vereinfachen die Bereitstellung, Neukonfiguration und Aktualisierung von BeeGFS auf NetApp.

Stk: Eine Befehlszeilenoberfläche, die von einem der Dateiknoten im Cluster zur Abfrage und Kontrolle des Status von Knoten und Ressourcen im Cluster verfügbar ist.

Allgemeine Terminologie

Failover: jeder BeeGFS-Dienst hat einen bevorzugten Dateiknoten, auf dem er ausgeführt wird, es sei denn, der Knoten schlägt fehl. Wenn ein BeeGFS-Service auf einem nicht-bevorzugten/sekundären Dateiknoten ausgeführt wird, muss er sich im Failover befinden.

Failback: der Akt, BeeGFS-Dienste von einem nicht bevorzugten Dateiknoten zurück zu ihrem bevorzugten Knoten zu verschieben.

HA-Paar: zwei Datei-Nodes, die auf den gleichen Satz von Block-Nodes zugreifen können, werden manchmal als HA-Paar bezeichnet. Dieser Begriff wird von NetApp häufig verwendet, um zwei Storage-Controller oder Nodes zu bezeichnen, die untereinander „übernommen“ können.

Wartungsmodus: deaktiviert die gesamte Ressourcenüberwachung und verhindert, dass Pacemaker Ressourcen im Cluster verschieben oder anderweitig verwalten kann (siehe auch den Abschnitt auf "Wartungsmodus").

HA-Cluster: ein oder mehrere Dateiknoten mit BeeGFS-Diensten, die ein Failover zwischen mehreren Knoten im Cluster ausführen können, um ein hochverfügbares BeeGFS-Dateisystem zu erstellen. Häufig sind Datei-Nodes in HA-Paaren konfiguriert, die in der Lage sind, eine Untergruppe der BeeGFS-Dienste im Cluster auszuführen.

Wann Ansible im Vergleich zum Tool PCs verwendet werden soll

Wann sollten Sie Ansible im Vergleich zum PCs-Befehlszeilungstool für das Management des HA-Clusters verwenden?

Alle Cluster-Implementierungs- und Neukonfigurierungsaufgaben sollten mit Ansible von einem externen Ansible-Kontroll-Node abgeschlossen werden. Temporäre Änderungen im Clusterstatus (z. B. ein- und Ausstellen von Knoten in den Standby-Modus) werden in der Regel durch Anmeldung an einem Knoten des Clusters (vorzugsweise einer, der nicht beeinträchtigt ist oder sich über die Wartung befindet) und unter Verwendung des Befehlszeilen-Tools PCs durchgeführt.

Das Ändern einer beliebigen Cluster-Konfiguration einschließlich Ressourcen, Einschränkungen, Eigenschaften und der BeeGFS Services selbst sollte immer mit Ansible erfolgen. Das Verwalten einer aktuellen Kopie des Ansible-Bestands und Playbook (ideal zur Versionskontrolle, um Änderungen zu verfolgen) ist Teil der Wartung des Clusters. Wenn Sie Änderungen an der Konfiguration vornehmen müssen, aktualisieren Sie den Bestand und führen Sie das Ansible-Playbook aus, das die BeeGFS HA-Rolle importiert.

Die HA-Rolle verarbeitet, das Cluster in den Wartungsmodus zu platzieren und anschließend alle erforderlichen Änderungen vorzunehmen, bevor BeeGFS oder Cluster-Services neu gestartet werden, um die neue Konfiguration anzuwenden. Da in der Regel keine vollständigen Node-Neustarts außerhalb der ursprünglichen Implementierung erforderlich sind, wird das Rerunning von Ansible in der Regel als „sicheres“ Verfahren angesehen. Für den Fall, dass BeeGFS-Services neu gestartet werden müssen, wird jedoch immer während Wartungsfenster oder außerhalb der Geschäftszeiten empfohlen. Diese Neustarts sollten in der Regel keine Anwendungsfehler verursachen, können aber die Leistung beeinträchtigen (was einige Anwendungen besser verarbeiten können als andere).

Die erneute Ausführung von Ansible ist auch eine Option, wenn Sie den gesamten Cluster wieder in einen

vollkommen optimalen Zustand zurückversetzen möchten, und kann in einigen Fällen den Status des Clusters einfacher wiederherstellen als PCs. Insbesondere in einem Notfall, in dem der Cluster aus irgendeinem Grund ausgefallen ist, kann, wenn alle Knoten gesichert werden, Ansible neu zu starten, den Cluster schneller und zuverlässiger wiederherstellen, als zu versuchen, PCs zu verwenden.

Untersuchen Sie den Status des Clusters

Verwenden Sie PCs, um den Status des Clusters anzuzeigen.

Überblick

Wird Ausgeführt `pcs status` Von jedem Cluster-Node aus können Sie den Gesamtstatus des Clusters und den Status jeder Ressource (z. B. BeeGFS-Services und deren Abhängigkeiten) am einfachsten einsehen. In diesem Abschnitt wird erklärt, was Sie in der Ausgabe von finden `pcs status` Befehl.

Allgemeines zur Ausgabe von `pcs status`

Laufen `pcs status` Auf jedem Clusterknoten, auf dem die Cluster-Dienste (Pacemaker und Corosync) gestartet werden. Oben in der Ausgabe wird eine Zusammenfassung des Clusters angezeigt:

```
[root@beegfs_01 ~]# pcs status
Cluster name: hacluster
Cluster Summary:
  * Stack: corosync
  * Current DC: beegfs_01 (version 2.0.5-9.el8_4.3-ba59be7122) - partition
with quorum
  * Last updated: Fri Jul  1 13:37:18 2022
  * Last change:  Fri Jul  1 13:23:34 2022 by root via cibadmin on
beegfs_01
  * 6 nodes configured
  * 235 resource instances configured
```

Im folgenden Abschnitt werden Nodes im Cluster aufgeführt:

```
Node List:
  * Node beegfs_06: standby
  * Online: [ beegfs_01 beegfs_02 beegfs_04 beegfs_05 ]
  * OFFLINE: [ beegfs_03 ]
```

Dies zeigt insbesondere alle Knoten an, die sich im Standby- oder Offline-Modus befinden. Nodes im Standby-Modus sind weiterhin am Cluster beteiligt, sind jedoch als nicht zur Ausführung von Ressourcen geeignet. Nodes, die offline sind, geben an, dass auf diesem Node keine Cluster-Services ausgeführt werden, entweder da sie manuell angehalten werden, oder weil der Node neu gebootet/heruntergefahren wurde.

Beim ersten Starten von Nodes werden Cluster-Services angehalten und müssen manuell gestartet werden, um zu vermeiden, dass versehentlich Ressourcen auf einen nicht funktionsuntüchtigen Node zurückfallen.

Wenn sich Knoten aufgrund eines nicht-administrativen Grund im Standby- oder Offline-Modus befinden (zum Beispiel ein Ausfall), wird neben dem Status des Node in Klammern zusätzlicher Text angezeigt. Wenn beispielsweise das Fechten deaktiviert ist und eine Ressource auf einen Fehler stößt, wird angezeigt `Node <HOSTNAME>: standby (on-fail)`. Ein anderer möglicher Zustand ist `Node <HOSTNAME>: UNCLEAN (offline)`, Die kurz als ein Knoten angezeigt wird, wird eingezäunt, aber bleibt bestehen, wenn das Fechten fehlgeschlagen zeigt, dass der Cluster den Status des Knotens nicht bestätigen kann (dies kann verhindern, dass die Ressourcen auf anderen Knoten beginnen).

Im nächsten Abschnitt werden alle Ressourcen im Cluster und ihre Status angezeigt:

```
Full List of Resources:
  * mgmt-monitor    (ocf::eseries:beegfs-monitor):   Started beegfs_01
  * Resource Group: mgmt-group:
    * mgmt-FS1  (ocf::eseries:beegfs-target):    Started beegfs_01
    * mgmt-IP1  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-IP2  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-service  (systemd:beegfs-mgmtd):  Started beegfs_01
[...]
```

Ähnlich wie bei Knoten wird neben dem Ressourcenzustand in Klammern zusätzlicher Text angezeigt, wenn Probleme mit der Ressource auftreten. Wenn z. B. Pacemaker einen Ressourcenstopp anfordert und dieser nicht innerhalb der zugewiesenen Zeit abgeschlossen werden kann, versucht Pacemaker, den Knoten einzuzäunen. Wenn das Fechten deaktiviert ist oder der Fechten-Vorgang fehlschlägt, wird der Ressourcenzustand angezeigt `FAILED <HOSTNAME> (blocked)` Pacemaker kann ihn nicht auf einem anderen Knoten starten.

Es ist erwähnenswert BeeGFS HA-Cluster nutzen eine Reihe von BeeGFS optimiert benutzerdefinierte OCF-Ressourcen-Agenten. Insbesondere ist der BeeGFS-Monitor für das Auslösen eines Failover verantwortlich, wenn BeeGFS-Ressourcen auf einem bestimmten Knoten nicht verfügbar sind.

Konfigurieren Sie HA-Cluster und BeeGFS neu

Verwenden Sie Ansible, um das Cluster neu zu konfigurieren.

Überblick

Generell sollten Sie jeden Aspekt des BeeGFS HA-Clusters neu konfigurieren, indem Sie Ihren Ansible-Bestand aktualisieren und den `ansible-playbook` Befehl erneut ausführen. Dazu gehören das Aktualisieren von Warnungen, das Ändern der Konfiguration für permanente Fechten oder das Anpassen der BeeGFS-Servicekonfiguration. Diese werden über die `group_vars/ha_cluster.yml` Datei angepasst und eine vollständige Liste der Optionen finden Sie im "Festlegen Der Konfiguration Des Gemeinsamen Dateiknotens" Abschnitt.

Weitere Informationen zu ausgewählten Konfigurationsoptionen finden Sie unten, die Administratoren bei der Wartung oder Wartung des Clusters beachten sollten.

So deaktivieren und aktivieren Sie Fechten

Beim Einrichten des Clusters ist Fechten standardmäßig aktiviert/erforderlich. In einigen Fällen ist es wünschenswert, Fechten vorübergehend zu deaktivieren, um sicherzustellen, dass Knoten nicht versehentlich heruntergefahren werden, wenn bestimmte Wartungsvorgänge ausgeführt werden (z. B. ein Upgrade des Betriebssystems). Auch wenn dies manuell deaktiviert werden kann, sollte es auf die Kompromisse-Administratoren achten.

OPTION 1: Deaktivieren Sie Fechten mit Ansible (empfohlen).

Wenn das Fechten mit Ansible deaktiviert wird, wird die on-Fail-Aktion des BeeGFS-Monitors von „Zaun“ in „Standby“ geändert. Wenn der BeeGFS-Monitor einen Fehler erkennt, versucht er, den Knoten in den Standby-Modus zu stellen und alle BeeGFS-Dienste zu ausfallsicher. Außerhalb aktiver Fehlerbehebung/Tests ist dies in der Regel wünschenswerter als Option 2. Der Nachteil ergibt sich daraus, dass eine Ressource auf dem ursprünglichen Knoten nicht stoppt, dass sie an einem anderen Ort gestartet werden kann (weshalb normalerweise ein Fechten für Produktionscluster erforderlich ist).

1. In Ihrem Ansible-Inventar unter `groups_vars/ha_cluster.yml` Fügen Sie die folgende Konfiguration hinzu:

```
beegfs_ha_cluster_crm_config_options:
    stonith-enabled: False
```

2. Führen Sie das Ansible-Playbook erneut aus, um die Änderungen auf das Cluster anzuwenden.

OPTION 2: Manuelle Abwahl deaktivieren.

In einigen Fällen möchten Sie die Fechten unter Umständen vorübergehend deaktivieren, ohne Ansible neu zu verwenden, um die Fehlerbehebung oder das Testen des Clusters zu erleichtern.

> Wenn der BeeGFS-Monitor in dieser Konfiguration einen Fehler erkennt, versucht das Cluster, die entsprechende Ressourcengruppe zu stoppen. Es wird KEIN vollständiger Failover ausgelöst oder versucht, die betroffene Ressourcengruppe auf einen anderen Host neu zu starten oder zu verschieben. Zur Wiederherstellung sollten Sie alle Probleme beheben und anschließend ausführen `pcs resource cleanup` Oder setzen Sie den Knoten manuell in den Standby-Modus.

Schritte

1. So legen Sie fest, ob Fechten (stonith) global aktiviert oder deaktiviert ist: `pcs property show stonith-enabled`
2. So deaktivieren Sie den Fechtlauf: `pcs property set stonith-enabled=false`
3. So aktivieren Sie den Fechtlauf: `pcs property set stonith-enabled=true`

> Diese Einstellung wird beim nächsten Ausführen des Ansible-Playbooks überschrieben.

Aktualisieren Sie die HA-Cluster-Komponenten

BeeGFS-Version aktualisieren

Führen Sie die folgenden Schritte aus, um die BeeGFS-Version des HA-Clusters mithilfe von Ansible zu aktualisieren.

Überblick

BeeGFS folgt einem `major.minor.patch` Versionsschema. Die BeeGFS HA-Ansible-Rollen werden für jede unterstützte `major.minor` Version (z. B. `beegfs_ha_7_2` und `beegfs_ha_7_3`) bereitgestellt. Jede HA-Rolle ist auf die neueste BeeGFS-Patch-Version fixiert, die zum Zeitpunkt der Veröffentlichung der Ansible Sammlung verfügbar ist.

Ansible sollte für alle BeeGFS Upgrades verwendet werden, einschließlich dem Verschieben zwischen größeren, kleineren und Patch-Versionen von BeeGFS. Um BeeGFS zu aktualisieren, müssen Sie zuerst die BeeGFS Ansible-Sammlung aktualisieren, die außerdem die neuesten Fixes und Verbesserungen an der Implementierungs-/Management-Automatisierung und dem zugrunde liegenden HA-Cluster heraufgibt. Selbst nach der Aktualisierung auf die neueste Version der Kollektion wird BeeGFS erst aktualisiert `ansible-playbook` Wird mit dem ausgeführt `-e "beegfs_ha_force_upgrade=true"` Einstellen.

Weitere Informationen zu BeeGFS-Versionen finden Sie im "BeeGFS Upgrade-Dokumentation".

Getestete Upgrade-Pfade

Jede Version der BeeGFS-Kollektion wird mit spezifischen Versionen von BeeGFS getestet, um die Interoperabilität zwischen allen Komponenten zu gewährleisten. Außerdem werden Tests durchgeführt, um sicherzustellen, dass Upgrades von der von der letzten Version der Sammlung unterstützten BeeGFS-Version(en) auf die in der neuesten Version unterstützten durchgeführt werden können.

Originalversion	Upgrade-Version	Multirail	Details
7.2.6	7.3.2	Ja.	Beegfs-Sammlung von v3.0.1 auf v3.1.0, multirail hinzugefügt
7.2.6	7.2.8	Nein	Beegfs-Sammlung wird von v3.0.1 auf v3.1 aktualisiert
7.2.8	7.3.1	Ja.	Upgrade mit beegfs Collection v3.1.0, multirail hinzugefügt
7.3.1	7.3.2	Ja.	Upgrade mit beegfs Collection v3.1.0
7.3.2	7.4.1	Ja.	Upgrade mit beegfs Collection v3.2.0
7.4.1	7.4.2	Ja.	Upgrade mit beegfs Collection v3.2.0

Schritte beim BeeGFS-Upgrade

In den folgenden Abschnitten werden die Schritte zum Aktualisieren der BeeGFS Ansible Sammlung und BeeGFS selbst beschrieben. Achten Sie besonders auf zusätzliche Schritte für die Aktualisierung von BeeGFS Major oder Minor Versionen.

Schritt: Upgrade der BeeGFS-Sammlung

Bei Erfassungs-Upgrades mit Zugriff auf "Ansible-Galaxie", Ausführen des folgenden Befehls:

```
ansible-galaxy collection install netapp_eseries.beegfs --upgrade
```

Laden Sie die Sammlung von herunter, um Offline-Sammlungs-Upgrades von zu erhalten "Ansible-Galaxie" Durch Klicken auf das gewünschte `Install Version` Und dann `Download tarball`. Übertragen Sie den Tarball auf Ihren Ansible-Steuerungsknoten und führen Sie den folgenden Befehl aus.

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

Siehe "Sammlungen Werden Installiert" Finden Sie weitere Informationen.

Schritt 2: Aktualisieren Sie den Ansible-Bestand

Nehmen Sie alle erforderlichen oder gewünschten Aktualisierungen der Ansible-Bestandsdateien Ihres Clusters vor. Im "Hinweise Zum Versionsupgrade"folgenden Abschnitt finden Sie Einzelheiten zu Ihren spezifischen Upgrade-Anforderungen. "Ansible-Bestandsübersicht"Allgemeine Informationen zur Konfiguration Ihres BeeGFS HA-Bestands finden Sie im Abschnitt.

Schritt 3: Ansible-Playbook aktualisieren (nur bei Aktualisierung von Haupt- oder Nebenversionen)

Wenn Sie zwischen Haupt- oder Unterversionen wechseln, aktualisieren Sie in der `playbook.yml` Datei, die zum Bereitstellen und Warten des Clusters verwendet wird, den Namen der `beegfs_ha_<VERSION>` Rolle, damit die gewünschte Version angezeigt wird. Wenn Sie beispielsweise BeeGFS 7.4 bereitstellen möchten, wäre dies `beegfs_ha_7_4`:

```
- hosts: all
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.beegfs
  tasks:
    - name: Ensure BeeGFS HA cluster is setup.
      ansible.builtin.import_role:  # import_role is required for tag
availability.
        name: beegfs_ha_7_4
```

Weitere Informationen zum Inhalt dieser Playbook-Datei finden Sie im "Implementieren Sie das BeeGFS HA-Cluster" Abschnitt.

Schritt 4: Führen Sie das BeeGFS-Upgrade aus

So wenden Sie das BeeGFS-Update an:

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

Hinter den Kulissen übernimmt die BeeGFS HA-Rolle:

- Stellen Sie sicher, dass sich das Cluster in einem optimalen Zustand befindet, wobei sich jeder BeeGFS-Service auf seinem bevorzugten Node befindet.
- Versetzen Sie das Cluster in den Wartungsmodus.
- Aktualisieren der HA-Cluster-Komponenten (falls erforderlich)
- Aktualisieren Sie jeden Dateiknoten nacheinander wie folgt:
  - Setzen Sie ihn in den Standby-Modus und führen Sie ein Failover seiner Dienste zum sekundären Knoten durch.
  - BeeGFS-Pakete aktualisieren.
  - Fallback-Services.
- Verschieben Sie das Cluster aus dem Wartungsmodus.

Hinweise zur Versionsaktualisierung

Upgrade von BeeGFS Version 7.2.6 oder 7.3.0

Änderungen an verbindungsbasierter Authentifizierung

BeeGFS-Versionen, die nach 7.3.1 veröffentlicht wurden, erlauben nicht mehr, dass Dienste ohne Angabe von A gestartet werden `connAuthFile` Oder Einstellung `connDisableAuthentication=true` In der Konfigurationsdatei des Dienstes. Es wird dringend empfohlen, die verbindungsbasierte Authentifizierungssicherheit zu aktivieren. Siehe "BeeGFS-Verbindungsbasierte Authentifizierung" Finden Sie weitere Informationen.

Standardmäßig ist der festgelegt `beegfs_ha*` Rollen generieren und verteilen diese Datei und fügen sie auch zum Ansible-Steuerungsknoten bei hinzu `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`. Der `beegfs_client` Die Rolle überprüft auch, ob diese Datei vorhanden ist, und liefert sie an die Clients, sofern verfügbar.

> Wenn der `beegfs_client` Die Rolle wurde nicht zur Konfiguration von Clients verwendet. Diese Datei muss manuell auf jeden Client und auf den verteilt werden `connAuthFile` Konfiguration in `beegfs-client.conf` Dateisatz für die Verwendung. Beim Upgrade von einer früheren Version von BeeGFS, bei der die verbindungsbasierte Authentifizierung nicht aktiviert war, verlieren Clients den Zugriff, es sei denn, die auf der Verbindung basierende Authentifizierung ist als Teil des Upgrades durch die Einstellung deaktiviert `beegfs_ha_conn_auth_enabled: false` In `group_vars/ha_cluster.yml` (Nicht empfohlen).

Weitere Details und alternative Konfigurationsoptionen finden "Festlegen Der Konfiguration Des Gemeinsamen Dateiknotens"Sie im Abschnitt zum Konfigurieren der Verbindungsauthentifizierung.

Aktualisieren Sie Pacemaker- und Corosync-Pakete in einem HA-Cluster

Führen Sie diese Schritte aus, um Pacemaker- und Corosync-Pakete in einem HA-Cluster zu aktualisieren.

Überblick

Durch ein Upgrade von Pacemaker und Corosync wird sichergestellt, dass der Cluster von neuen Funktionen, Sicherheits-Patches und Leistungsverbesserungen profitiert.

Upgrade-Ansatz

Es gibt zwei empfohlene Ansätze für das Upgrade eines Clusters: Ein rollierendes Upgrade oder eine vollständige Abschaltung des Clusters. Jeder Ansatz hat seine eigenen vor- und Nachteile. Der Aktualisierungsvorgang kann je nach Ihrer Pacemaker-Version variieren. Bestimmen Sie anhand der Dokumentation von ClusterLabs"Aktualisieren eines Pacemaker-Clusters", welche Vorgehensweise verwendet werden soll. Bevor Sie einen Upgrade-Ansatz verfolgen, müssen Sie Folgendes überprüfen:

- Die neuen Pacemaker- und Corosync-Pakete werden von der NetApp BeeGFS-Lösung unterstützt.
- Für das BeeGFS-Dateisystem und die Pacemaker-Cluster-Konfiguration sind gültige Backups vorhanden.
- Das Cluster befindet sich in einem ordnungsgemäßen Zustand.

Rollierendes Upgrade

Bei dieser Methode wird jeder Node aus dem Cluster entfernt, aktualisiert und anschließend wieder in das Cluster eingeführt, bis die neue Version auf allen Nodes ausgeführt wird. Dieser Ansatz sorgt für einen unterbrechungsfreien Cluster, was ideal für größere HA-Cluster ist, birgt aber auch das Risiko, dass während des Prozesses gemischte Versionen ausgeführt werden. Dieser Ansatz sollte in einem Cluster mit zwei Nodes vermieden werden.

1. Vergewissern Sie sich, dass sich das Cluster in einem optimalen Zustand befindet, wobei jeder BeeGFS-Service auf seinem bevorzugten Node ausgeführt wird. Weitere Informationen finden Sie unter "Untersuchen Sie den Status des Clusters" .
2. Platzieren Sie den Node für das Upgrade in den Standby-Modus, um alle BeeGFS-Services zu leeren (oder zu verschieben):

   ```
   pcs node standby <HOSTNAME>
   ```

3. Überprüfen Sie, ob die Services des Node durch Ausführen von abgelaufen sind:

   ```
   pcs status
   ```

   Stellen Sie sicher, dass keine Dienste als auf dem Node im Standby gemeldet werden `Started`.

   Je nach Clustergröße kann es Sekunden oder Minuten dauern, bis Dienste zum Schwesterknoten verschoben werden. Wenn ein BeeGFS-Dienst auf dem Schwesterknoten nicht gestartet werden kann, lesen Sie die "Leitfäden Zur Fehlerbehebung".

4. Fahren Sie das Cluster auf dem Node herunter:

   ```
   pcs cluster stop <HOSTNAME>
   ```

5. Aktualisieren Sie die Pacemaker-, Corosync- und PCs-Pakete auf dem Knoten:

   Die Befehle des Package Managers variieren je nach Betriebssystem. Die folgenden Befehle gelten für Systeme, auf denen RHEL 8 und höher ausgeführt wird.

   ```
   dnf update pacemaker-<version>
   ```

   ```
   dnf update corosync-<version>
   ```

   ```
   dnf update pcs-<version>
   ```

6. Starten Sie die Pacemaker-Clusterdienste auf dem Knoten:

   ```
   pcs cluster start <HOSTNAME>
   ```

7. Wenn das `pcs` Paket aktualisiert wurde, authentifizieren Sie den Node erneut beim Cluster:

   ```
   pcs host auth <HOSTNAME>
   ```

8. Überprüfen Sie, ob die Pacemaker-Konfiguration mit dem Werkzeug noch gültig `crm_verify` ist.

   Dies muss nur einmal während des Cluster-Upgrades überprüft werden.

   ```
   crm_verify -L -V
   ```

9. Beenden Sie den Standby-Modus des Node:

   ```
   pcs node unstandby <HOSTNAME>
   ```

10. Verschieben Sie alle BeeGFS-Services zurück auf ihren bevorzugten Node:

    ```
    pcs resource relocate run
    ```

11. Wiederholen Sie die vorherigen Schritte für jeden Knoten im Cluster, bis auf allen Knoten die gewünschten Pacemaker-, Corosync- und PCs-Versionen ausgeführt werden.
12. Führen Sie abschließend den Cluster aus `pcs status`, und überprüfen Sie, ob er ordnungsgemäß ist, und der `Current DC` meldet die gewünschte Pacemaker-Version.

Wenn der `Current DC` Bericht „Misted-Version“ meldet, wird ein Knoten im Cluster weiterhin mit der vorherigen Pacemaker-Version ausgeführt und muss aktualisiert werden. Wenn ein aktualisierter Node nicht in der Lage ist, dem Cluster erneut beizutreten, oder wenn die Ressourcen nicht gestartet werden können, prüfen Sie die Cluster-Protokolle, und lesen Sie die Pacemaker-Versionshinweise oder Benutzerhandbücher nach bekannten Upgrade-Problemen.

Schließen Sie den Cluster ab

Bei diesem Ansatz werden alle Cluster Nodes und Ressourcen heruntergefahren, die Nodes aktualisiert und das Cluster anschließend neu gestartet. Dieser Ansatz ist erforderlich, wenn die Pacemaker- und Corosync-Versionen keine Konfiguration mit gemischten Versionen unterstützen.

1. Vergewissern Sie sich, dass sich das Cluster in einem optimalen Zustand befindet, wobei jeder BeeGFS-Service auf seinem bevorzugten Node ausgeführt wird. Weitere Informationen finden Sie unter "Untersuchen Sie den Status des Clusters" .
2. Fahren Sie die Cluster-Software (Pacemaker und Corosync) auf allen Knoten herunter.

   Je nach Cluster-Größe kann es Sekunden oder Minuten dauern, bis das gesamte Cluster angehalten wurde.

   ```
   pcs cluster stop --all
   ```

3. Sobald Cluster-Services auf allen Knoten heruntergefahren sind, aktualisieren Sie die Pacemaker-, Corosync- und PCs-Pakete auf jedem Knoten entsprechend Ihren Anforderungen.

   Die Befehle des Package Managers variieren je nach Betriebssystem. Die folgenden Befehle gelten für Systeme, auf denen RHEL 8 und höher ausgeführt wird.

   ```
   dnf update pacemaker-<version>
   ```

   ```
   dnf update corosync-<version>
   ```

   ```
   dnf update pcs-<version>
   ```

4. Starten Sie nach dem Upgrade aller Nodes die Cluster-Software auf allen Nodes:

   ```
   pcs cluster start --all
   ```

5. Wenn das `pcs` Paket aktualisiert wurde, authentifizieren Sie jeden Node im Cluster erneut:

```
pcs host auth <HOSTNAME>
```

6. Führen Sie abschließend den Cluster aus `pcs status`, und überprüfen Sie, ob er in Ordnung ist, und der `Current DC` meldet die korrekte Pacemaker-Version.

> Wenn der `Current DC` Bericht „Misted-Version“ meldet, wird ein Knoten im Cluster weiterhin mit der vorherigen Pacemaker-Version ausgeführt und muss aktualisiert werden.

Aktualisiert die Datei-Node-Adapter-Firmware

Führen Sie die folgenden Schritte aus, um die ConnectX-7-Adapter des Datei-Knotens auf die neueste Firmware zu aktualisieren.

Überblick

Um einen neuen MLNX_OFED-Treiber zu unterstützen, neue Funktionen zu aktivieren oder Fehler zu beheben, ist möglicherweise eine Aktualisierung der ConnectX-7-Adapter-Firmware erforderlich. In diesem Handbuch wird das Dienstprogramm von NVIDIA für Adapteraktualisierungen aufgrund seiner Benutzerfreundlichkeit und Effizienz verwendet `mlxfwmanager`.

Upgrade-Überlegungen

In diesem Handbuch werden zwei Ansätze zur Aktualisierung der ConnectX-7-Adapter-Firmware beschrieben: Ein laufendes Update und ein zwei-Knoten-Cluster-Update. Wählen Sie den passenden Aktualisierungsansatz gemäß der Clustergröße aus. Bevor Sie Firmware-Aktualisierungen durchführen, stellen Sie sicher, dass:

- Ein unterstützter MLNX_OFED-Treiber ist installiert, siehe "Technologieanforderungen erfüllt".
- Für das BeeGFS-Dateisystem und die Pacemaker-Cluster-Konfiguration sind gültige Backups vorhanden.
- Das Cluster befindet sich in einem ordnungsgemäßen Zustand.

Vorbereitung des Firmware-Updates

Es wird empfohlen, das NVIDIA-Dienstprogramm zu verwenden `mlxfwmanager`, um die Adapter-Firmware eines Knotens zu aktualisieren, die mit dem NVIDIA-Treiber MLNX_OFED gebündelt ist. Laden Sie vor dem Starten der Updates das Firmware-Image des Adapters von herunter"Die Support-Website von NVIDIA", und speichern Sie es auf jedem Datei-Node.

> Für Lenovo ConnectX-7 Adapter, verwenden Sie das `mlxfwmanager_LES` Tool, das auf der NVIDIA-Seite zur Verfügung steht"OEM-Firmware".

Rollierender Aktualisierungsansatz

Dieser Ansatz wird für alle HA-Cluster mit mehr als zwei Nodes empfohlen. Dieser Ansatz beinhaltet die Aktualisierung der Adapter-Firmware auf einem Datei-Node, sodass das HA-Cluster Anforderungen weiterhin erfüllen kann. Allerdings wird empfohlen, um I/O-Anfragen während dieser Zeit zu vermeiden.

1. Vergewissern Sie sich, dass sich das Cluster in einem optimalen Zustand befindet, wobei jeder BeeGFS-Service auf seinem bevorzugten Node ausgeführt wird. Weitere Informationen finden Sie unter "Untersuchen Sie den Status des Clusters" .

2. Wählen Sie einen Datei-Node aus, um ihn zu aktualisieren und in den Standby-Modus zu versetzen, der alle BeeGFS-Services von diesem Node entfernt (oder verschiebt):

```
pcs node standby <HOSTNAME>
```

3. Überprüfen Sie, ob die Dienste des Node abgelaufen sind, indem Sie Folgendes ausführen:

```
pcs status
```

Vergewissern Sie sich, dass keine Services als auf dem Node im Standby-Modus melden `Started`.

> Je nach Cluster-Größe kann es Sekunden oder Minuten dauern, bis die BeeGFS-Dienste zum Schwesterknoten verschoben werden. Wenn ein BeeGFS-Dienst auf dem Schwesterknoten nicht gestartet werden kann, lesen Sie die "Leitfäden Zur Fehlerbehebung".

4. Aktualisieren Sie die Adapter-Firmware mit `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

Beachten Sie `PCI Device Name` für jeden Adapter, der Firmware-Updates empfängt.

5. Setzen Sie jeden Adapter mithilfe des Dienstprogramms zurück `mlxfwreset`, um die neue Firmware anzuwenden.

> Einige Firmware-Aktualisierungen erfordern möglicherweise einen Neustart, um das Update anzuwenden. Weitere Informationen finden Sie unter"Die Einschränkungen von NVIDIA mlxfwreset". Wenn ein Neustart erforderlich ist, führen Sie einen Neustart durch, anstatt die Adapter zurückzusetzen.

a. Beenden Sie den opensm-Dienst:

```
systemctl stop opensm
```

b. Führen Sie den folgenden Befehl für jeden `PCI Device Name` zuvor genannten aus.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Starten Sie den opensm-Dienst:

```
systemctl start opensm
```

d. Starten Sie den `eseries_nvme_ib.service` .

```
systemctl restart eseries_nvme_ib.service
```

e. Überprüfen Sie, ob die Volumes des E-Series-Speicherarrays vorhanden sind.

```
multipath -ll
```

1. Führen Sie aus `ibstat`, und überprüfen Sie, ob alle Adapter mit der gewünschten Firmware-Version ausgeführt werden:

```
ibstat
```

2. Starten Sie die Pacemaker-Clusterdienste auf dem Knoten:

```
pcs cluster start <HOSTNAME>
```

3. Beenden Sie den Standby-Modus des Node:

```
pcs node unstandby <HOSTNAME>
```

4. Verschieben Sie alle BeeGFS-Services zurück auf ihren bevorzugten Node:

```
pcs resource relocate run
```

Wiederholen Sie diese Schritte für jeden Datei-Node im Cluster, bis alle Adapter aktualisiert wurden.

Update für Cluster mit zwei Nodes

Dieser Ansatz wird für HA-Cluster mit nur zwei Nodes empfohlen. Dieser Ansatz ähnelt einem rollierenden Update, enthält jedoch zusätzliche Schritte zur Vermeidung von Service-Ausfallzeiten, wenn die Cluster-Services eines Node angehalten werden.

1. Vergewissern Sie sich, dass sich das Cluster in einem optimalen Zustand befindet, wobei jeder BeeGFS-Service auf seinem bevorzugten Node ausgeführt wird. Weitere Informationen finden Sie unter "Untersuchen Sie den Status des Clusters" .
2. Wählen Sie einen Datei-Node aus, um den Node zu aktualisieren und in den Standby-Modus zu versetzen, der alle BeeGFS-Services von diesem Node entfernt (oder verschiebt):

```
pcs node standby <HOSTNAME>
```

3. Überprüfen Sie, ob die Ressourcen des Node abgelaufen sind, indem Sie Folgendes ausführen:

```
pcs status
```

Vergewissern Sie sich, dass keine Services als auf dem Node im Standby-Modus melden `Started`.

Je nach Cluster-Größe kann es Sekunden oder Minuten dauern, bis BeeGFS-Dienste als auf dem Schwesternknoten melden `Started`. Wenn ein BeeGFS-Dienst nicht gestartet werden kann, lesen Sie die "Leitfäden Zur Fehlerbehebung".

4. Versetzen Sie das Cluster in den Wartungsmodus.

```
pcs property set maintenance-mode=true
```

5. Aktualisieren Sie die Adapter-Firmware mit `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

Beachten Sie `PCI Device Name` für jeden Adapter, der Firmware-Updates empfängt.

6. Setzen Sie jeden Adapter mithilfe des Dienstprogramms zurück `mlxfwreset`, um die neue Firmware anzuwenden.

Einige Firmware-Aktualisierungen erfordern möglicherweise einen Neustart, um das Update anzuwenden. Weitere Informationen finden Sie unter"Die Einschränkungen von NVIDIA mlxfwreset". Wenn ein Neustart erforderlich ist, führen Sie einen Neustart durch, anstatt die Adapter zurückzusetzen.

a. Beenden Sie den opensm-Dienst:

```
systemctl stop opensm
```

b. Führen Sie den folgenden Befehl für jeden `PCI Device Name` zuvor genannten aus.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Starten Sie den opensm-Dienst:

```
systemctl start opensm
```

7. Führen Sie aus `ibstat`, und überprüfen Sie, ob alle Adapter mit der gewünschten Firmware-Version ausgeführt werden:

```
ibstat
```

8. Starten Sie die Pacemaker-Clusterdienste auf dem Knoten:

```
pcs cluster start <HOSTNAME>
```

9. Beenden Sie den Standby-Modus des Node:

```
pcs node unstandby <HOSTNAME>
```

10. Beenden Sie das Cluster aus dem Wartungsmodus.

```
pcs property set maintenance-mode=false
```

11. Verschieben Sie alle BeeGFS-Services zurück auf ihren bevorzugten Node:

```
pcs resource relocate run
```

Wiederholen Sie diese Schritte für jeden Datei-Node im Cluster, bis alle Adapter aktualisiert wurden.

Upgrade von E-Series Storage-Arrays

Führen Sie die folgenden Schritte aus, um die Komponenten des HA-Clusters des E-Series Storage-Arrays zu aktualisieren.

Überblick

Die NetApp E-Series Storage Arrays Ihres HA Clusters mit der neuesten Firmware auf dem neuesten Stand zu halten, gewährleistet optimale Performance und verbesserte Sicherheit. Firmware-Updates für das Storage Array werden über SANtricity OS-, NVSRAM- und Festplatten-Firmware-Dateien angewendet.

Obwohl ein Upgrade der Storage Arrays während des Online-Betriebs des HA-Clusters möglich ist, sollte das Cluster bei allen Upgrades in den Wartungsmodus versetzt werden.

Upgrade-Schritte für Block-Nodes

Im Folgenden wird beschrieben, wie die Firmware der Storage-Arrays mithilfe der `Netapp_Eseries.Santricity` Ansible-Sammlung aktualisiert wird. Bevor Sie fortfahren, lesen "Upgrade-Überlegungen"Sie das zur Aktualisierung von E-Series Systemen.

Ein Upgrade auf SANtricity OS 11.80 oder höhere Versionen ist nur ab 11.70.5P1 möglich. Das Speicher-Array muss vor der Anwendung weiterer Upgrades zuerst auf 11.70.5P1 aktualisiert werden.

1. Überprüfen Sie den Ansible Control-Node mithilfe der neuesten SANtricity Ansible Sammlung.
   - Bei Erfassungs-Upgrades mit Zugriff auf "Ansible-Galaxie", Ausführen des folgenden Befehls:

     ```
     ansible-galaxy collection install netapp_eseries.santricity --upgrade
     ```

   - Laden Sie für Offline-Upgrades den Sammeltarball von herunter"Ansible-Galaxie", übertragen Sie ihn auf Ihren Steuerungsknoten und führen Sie Folgendes aus:

     ```
     ansible-galaxy collection install netapp_eseries-santricity-
     <VERSION>.tar.gz --upgrade
     ```

     Siehe "Sammlungen Werden Installiert" Finden Sie weitere Informationen.

2. Holen Sie sich die neueste Firmware für Ihr Speicher-Array und die Laufwerke.
   a. Laden Sie die Firmware-Dateien herunter.
      - **SANtricity OS und NVSRAM:** Navigieren "NetApp Support Website"Sie zum und laden Sie die neueste Version von SANtricity OS und NVSRAM für Ihr Speicherarray-Modell herunter.
      - **Laufwerksfirmware:** Navigieren "E-Series Festplatten-Firmware-Website"Sie zum und laden Sie die neueste Firmware für jedes Laufwerkmodell Ihres Speicherarrays herunter.

   b. Speichern Sie SANtricity OS-, NVSRAM- und Laufwerk-Firmware-Dateien im `<inventory_directory>/packages` Verzeichnis Ihres Ansible Control Node.
3. Bei Bedarf aktualisieren Sie die Ansible-Bestandsdateien Ihres Clusters, damit alle Storage-Arrays (Block-Nodes), die aktualisiert werden müssen, einbezogen werden. Weitere Informationen finden Sie im "Ansible-Bestandsübersicht" Abschnitt.
4. Stellen Sie sicher, dass sich das Cluster mit jedem BeeGFS-Service auf seinem bevorzugten Node in einem optimalen Zustand befindet. Weitere Informationen finden Sie unter "Untersuchen Sie den Status des Clusters" .
5. Versetzen Sie das Cluster gemäß den Anweisungen in in "Versetzen Sie das Cluster in den Wartungsmodus"den Wartungsmodus.
6. Erstellen Sie ein neues Ansible-Playbook mit dem Namen `update_block_node_playbook.yml`. Füllen Sie das Playbook mit den folgenden Inhalten aus und ersetzen Sie die Versionen des SANtricity Betriebssystems, des NVSRAM und der Festplatten-Firmware auf Ihren gewünschten Upgrade-Pfad:

```
- hosts: eseries_storage_systems
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.santricity
  vars:
    eseries_firmware_firmware: "packages/<SantricityOS>.dlp"
    eseries_firmware_nvsram: "packages/<NVSRAM>.dlp"
    eseries_drive_firmware_firmware_list:
      - "packages/<drive_firmware>.dlp"
    eseries_drive_firmware_upgrade_drives_online: true

  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

7. Führen Sie über Ihren Ansible-Steuerungsknoten den folgenden Befehl aus, um die Updates zu starten:

```
ansible-playbook -i inventory.yml update_block_node_playbook.yml
```

8. Überprüfen Sie nach Abschluss des Playbook, ob sich jedes Speicher-Array in einem optimalen Zustand befindet.
9. Entfernen Sie das Cluster aus dem Wartungsmodus und überprüfen Sie, ob sich das Cluster in einem optimalen Zustand befindet, wobei sich jeder BeeGFS-Service auf seinem bevorzugten Node befindet.

Service und Wartung

Failover- und Failback-Services

BeeGFS-Services zwischen Cluster-Nodes verschieben

Überblick

BeeGFS-Services können ein Failover zwischen den Nodes im Cluster durchführen, um sicherzustellen, dass die Clients weiterhin auf das Filesystem zugreifen können, wenn ein Node einen Fehler aufweist, oder Sie müssen eine geplante Wartung durchführen. In diesem Abschnitt werden verschiedene Möglichkeiten beschrieben, wie Administratoren das Cluster nach der Wiederherstellung nach einem Ausfall reparieren oder Services manuell zwischen Nodes verschieben können.

Schritte

Failover und Failback

Failover (Geplant)

Wenn Sie einen einzelnen Datei-Node zur Wartung offline schalten müssen, möchten Sie in der Regel alle BeeGFS-Dienste von diesem Node verschieben (oder ablassen). Dies kann erreicht werden, indem zunächst der Knoten in den Standby-Modus versetzt wird:

```
pcs node standby <HOSTNAME>
```

Nach der Überprüfung mit `pcs status` Alle Ressourcen wurden auf dem alternativen Datei-Node neu gestartet. Sie können je nach Bedarf weitere Änderungen am Node vornehmen.

Failback (nach einem geplanten Failover)

Wenn Sie bereit sind, die BeeGFS-Dienste zuerst auf den bevorzugten Knoten wiederherzustellen `pcs status` Und überprüfen Sie in der „Knotenliste“, ob der Status Standby lautet. Wenn der Node neu gebootet wurde, wird er offline angezeigt, bis Sie die Cluster-Services in den Online-Modus versetzen:

```
pcs cluster start <HOSTNAME>
```

Sobald der Node online ist, bringen Sie ihn aus dem Standby-Modus mit:

```
pcs node unstandby <HOSTNAME>
```

Schließlich verlagern alle BeeGFS-Dienste wieder auf ihre bevorzugten Knoten mit:

```
pcs resource relocate run
```

Failback (nach einem ungeplanten Failover)

Wenn auf einem Node ein Hardware- oder ein anderer Fehler auftritt, sollte der HA-Cluster automatisch reagieren und seine Services auf einen gesunden Node verschieben. So bleibt den Administratoren Zeit für Korrekturmaßnahmen. Bevor Sie fortfahren, lesen "Fehlerbehebung"Sie den Abschnitt, um die Ursache des Failovers zu ermitteln und alle offenen Probleme zu beheben. Sobald der Knoten wieder eingeschaltet ist und sich in einem ordnungsgemäßen Zustand befindet, können Sie mit dem Failback fortfahren.

Wenn ein Node nach einem ungeplanten (oder geplanten) Neubooten gebootet wird, werden Cluster-Services nicht automatisch gestartet. Sie müssen daher den Node zuerst in den Online-Modus versetzen:

```
pcs cluster start <HOSTNAME>
```

Bei der nächsten Bereinigung werden alle Ressourcenfehler behoben, und der Fechtverlauf des Node wird zurückgesetzt:

```
pcs resource cleanup node=<HOSTNAME>
pcs stonith history cleanup <HOSTNAME>
```

Verifizieren in `pcs status` Der Knoten ist online und in einem ordnungsgemäßen Zustand. Standardmäßig werden BeeGFS-Dienste nicht automatisch Failback durchführen, um zu vermeiden, dass Ressourcen versehentlich auf einen ungesunden Knoten zurückverschoben werden. Wenn Sie bereit sind, alle Ressourcen im Cluster wieder an die bevorzugten Nodes zurückzugeben, mit den folgenden Funktionen:

```
pcs resource relocate run
```

Einzelne BeeGFS-Services werden auf alternative Datei-Nodes verschoben

Verschieben Sie einen BeeGFS-Service dauerhaft auf einen neuen Datei-Node

Wenn Sie den bevorzugten Datei-Node für einen einzelnen BeeGFS-Service dauerhaft ändern möchten, passen Sie den Ansible-Bestand an, sodass der bevorzugte Node zuerst aufgelistet wird, und führen Sie das Ansible-Playbook erneut aus.

In dieser Beispieldatei ist beegfs_01 beispielsweise `inventory.yml` der bevorzugte Datei-Node zum Ausführen des BeeGFS-Managementservice:

```
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
```

Durch eine Umkehrung des Auftrags würden die Managementservices am beegfs_02 bevorzugt werden:

```
        mgmt:
          hosts:
            beegfs_02:
            beegfs_01:
```

Verschieben Sie einen BeeGFS-Service vorübergehend auf einen alternativen Datei-Node

Im Allgemeinen, wenn ein Knoten gerade gewartet wird, möchten Sie die Schritte [Failover und Failback](#Failover-and-Failback) verwenden, um alle Dienste von diesem Knoten weg zu verschieben.

Wenn Sie aus irgendeinem Grund einen einzelnen Service auf einen anderen Dateiknoten verschieben müssen, führen Sie:

```
pcs resource move <SERVICE>-monitor <HOSTNAME>
```

Geben Sie keine einzelnen Ressourcen oder die Ressourcengruppe an. Geben Sie immer den Namen des Monitors für den BeeGFS-Dienst an, den Sie verschieben möchten. Um zum Beispiel den BeeGFS-Managementdienst auf beegfs_02 zu verschieben, führen Sie: Aus `pcs resource move mgmt-monitor beegfs_02`. Dieser Prozess kann wiederholt werden, um einen oder mehrere Services von den bevorzugten Nodes weg zu verschieben. Überprüfen Sie, ob `pcs status` die Services auf dem neuen Node verlegt/gestartet wurden.

Wenn Sie einen BeeGFS-Service wieder auf den bevorzugten Node verschieben möchten, löschen Sie zuerst die temporären Ressourcenbeschränkungen (diesen Schritt wird bei mehreren Services wiederholt):

```
pcs resource clear <SERVICE>-monitor
```

Wenn Sie bereit sind, den Service(s) dann wieder zurück zu den bevorzugten Knoten zu verschieben, werden die folgenden Aktionen ausgeführt:

```
pcs resource relocate run
```

Hinweis: Mit diesem Befehl werden Services verschoben, bei denen keine temporären Ressourcenbeschränkungen mehr vorhanden sind, die sich nicht auf den bevorzugten Nodes befinden.

Versetzen Sie das Cluster in den Wartungsmodus

Verhindern Sie, dass das HA-Cluster versehentlich auf geplante Änderungen in der Umgebung reagiert.

Überblick

Wenn Sie das Cluster in den Wartungsmodus versetzen, werden die gesamte Ressourcenüberwachung deaktiviert und Pacemaker kann nicht mehr Ressourcen im Cluster verschieben oder anderweitig verwalten. Alle Ressourcen werden auf den ursprünglichen Nodes weiterhin ausgeführt, unabhängig davon, ob es eine temporäre Ausfallbedingung gibt, die den Zugriff auf sie verhindern würde. Dies wird empfohlen/ist u. a.:

- Netzwerkwartung, die vorübergehend Verbindungen zwischen Datei-Nodes und BeeGFS-Diensten unterbrechen kann.
- Block-Node-Upgrades:
- Dateiknoten-Betriebssystem, Kernel oder andere Paketaktualisierungen.

Im Allgemeinen ist der einzige Grund, das Cluster manuell in den Wartungsmodus zu versetzen, um zu verhindern, dass es auf externe Änderungen in der Umgebung reagiert. Wenn für einen einzelnen Node im Cluster die physische Reparatur erforderlich ist, verwenden Sie keinen Wartungsmodus und platzieren Sie den Node einfach gemäß dem oben beschriebenen Verfahren in Standby. Beachten Sie, dass bei der Umleitung von Ansible der Cluster automatisch der Wartungsmodus für die meisten Softwarewartungsarbeiten einschließlich Upgrades und Konfigurationsänderungen durchgeführt wird.

Schritte

So überprüfen Sie, ob das Cluster sich im Wartungsmodus befindet:

```
pcs property config
```

Die `maintenance-mode` Eigenschaft wird nicht angezeigt, wenn das Cluster ordnungsgemäß ausgeführt wird. Wenn sich der Cluster derzeit im Wartungsmodus befindet, wird die Eigenschaft als gemeldet `true`. Um den Wartungsmodus zu aktivieren, führen Sie folgende Schritte aus:

```
pcs property set maintenance-mode=true
```

Sie können überprüfen, indem Sie den PC-Status ausführen und sicherstellen, dass alle Ressourcen „(nicht verwaltet)“ anzeigen. Um das Cluster aus dem Wartungsmodus zu nehmen, führen Sie folgende Schritte aus:

```
pcs property set maintenance-mode=false
```

Beenden Sie den Cluster und starten Sie den Cluster

Graziös wird das HA-Cluster angehalten und gestartet.

Überblick

In diesem Abschnitt wird beschrieben, wie das BeeGFS-Cluster ordnungsgemäß heruntergefahren und neu gestartet wird. Beispielszenarien, bei denen dies möglicherweise erforderlich ist, sind beispielsweise die elektrische Wartung oder die Migration zwischen Rechenzentren oder Racks.

Schritte

Wenn Sie aus irgendeinem Grund das gesamte BeeGFS-Cluster beenden und alle Dienste herunterfahren müssen, laufen:

```
pcs cluster stop --all
```

Es ist auch möglich, das Cluster auf einzelnen Nodes anzuhalten (wodurch automatisch ein Failover von Services auf einen anderen Node erfolgt). Es wird jedoch empfohlen, den Node zunächst in den Standby-Modus zu versetzen (siehe "Failover"Abschnitt):

```
pcs cluster stop <HOSTNAME>
```

So starten Sie Cluster Services und Ressourcen auf allen Nodes:

```
pcs cluster start --all
```

Oder starten Sie Services auf einem bestimmten Knoten mit:

```
pcs cluster start <HOSTNAME>
```

An dieser Stelle Lauf `pcs status` Überprüfen Sie, ob die Cluster- und BeeGFS-Services auf allen Nodes gestartet werden und die Services auf den erwarteten Nodes ausgeführt werden.

> Je nach Clustergröße kann es Sekunden oder Minuten dauern, bis der gesamte Cluster angehalten wird oder wie gestartet in angezeigt `pcs status` wird. Wenn `pcs cluster <COMMAND>` länger als fünf Minuten hängt, bevor Sie den Befehl mit „Strg+C“ abbrechen, melden Sie sich bei jedem Knoten des Clusters an und prüfen Sie mit `pcs status`, ob Clusterdienste (Corosync/Pacemaker) auf diesem Knoten noch ausgeführt werden. Von jedem Node, der das Cluster noch aktiv ist, können Sie überprüfen, welche Ressourcen das Cluster blockieren. Lösen Sie das Problem manuell, und der Befehl sollte entweder abgeschlossen werden oder kann erneut ausgeführt werden, um alle verbleibenden Services zu beenden.

Datei-Nodes ersetzen

Ersetzen eines Dateiknotens, wenn der ursprüngliche Server fehlerhaft ist.

Überblick

Dies bietet einen Überblick über die Schritte, die zum Austausch eines Datei-Nodes im Cluster erforderlich sind. Diese Schritte setzen voraus, dass der Datei-Node aufgrund eines Hardwareproblems ausgefallen ist und dass er durch einen neuen identischen File-Node ersetzt wurde.

Schritte

1. Ersetzen Sie den Datei-Node physisch und stellen Sie alle Kabel auf den Block-Node und das Storage-Netzwerk wieder her.
2. Installieren Sie das Betriebssystem auf dem Dateiknoten neu, einschließlich Hinzufügen von Red hat Subskriptionen.
3. Konfiguration von Management und BMC Networking auf dem Datei-Node
4. Aktualisieren Sie die Ansible-Bestandsaufnahme, wenn sich der Hostname, die IP, die Zuordnung der PCIe-zu-logischen Schnittstelle oder eine weitere Änderung bezüglich des neuen Datei-Nodes ergeben. Im Allgemeinen ist dies nicht erforderlich, wenn der Node durch identische Serverhardware ersetzt wurde und Sie die ursprüngliche Netzwerkkonfiguration verwenden.
   a. Wenn sich beispielsweise der Hostname geändert hat, erstellen Sie die Bestandsdatei des Node (oder benennen Sie sie um) (`host_vars/<NEW_NODE>.yml`) Und dann in der Ansible-Bestandsdatei (`inventory.yml`), ersetzen Sie den Namen des alten Knotens durch den neuen Knotennamen:

```
all:
    ...
    children:
    ha_cluster:
        children:
        mgmt:
            hosts:
            node_h1_new:    # Replaced "node_h1" with "node_h1_new"
            node_h2:
```

5. Entfernen Sie den alten Node von einem der anderen Nodes im Cluster: `pcs cluster node remove <HOSTNAME>`.

   ⚠ FAHREN SIE VOR AUSFÜHRUNG DIESES SCHRITTS NICHT FORT.

6. Auf dem Ansible-Steuerungsknoten:
   a. Entfernen Sie den alten SSH-Schlüssel mit:

   ```
   `ssh-keygen -R <HOSTNAME_OR_IP>`
   ```

   b. Konfigurieren Sie passwortloses SSH auf den Knoten Ersetzen mit:

   ```
   ssh-copy-id <USER>@<HOSTNAME_OR_IP>
   ```

7. Führen Sie das Ansible-Playbook erneut aus, um den Node zu konfigurieren und dem Cluster hinzuzufügen:

   ```
   ansible-playbook -i <inventory>.yml <playbook>.yml
   ```

8. An dieser Stelle, Lauf `pcs status` Und überprüfen Sie, ob der ersetzte Node jetzt aufgeführt ist und Services ausführt.

Erweitern oder verkleinern Sie den Cluster

Fügen Sie dem Cluster Bausteine hinzu oder entfernen Sie diese.

Überblick

In diesem Abschnitt werden verschiedene Überlegungen und Optionen dokumentiert, um die Größe Ihres BeeGFS HA-Clusters anzupassen. Normalerweise wird die Cluster-Größe durch Hinzufügen oder Entfernen von Bausteinen angepasst. Bei diesen handelt es sich in der Regel um zwei Datei-Nodes, die als HA-Paar eingerichtet wurden. Bei Bedarf können auch einzelne Datei-Nodes (oder andere Cluster-Nodes) hinzugefügt oder entfernt werden.

Hinzufügen eines Bausteins zum Cluster

Überlegungen

Das erweitern des Clusters durch Hinzufügen weiterer Bausteine ist ein unkomplizierter Prozess. Beachten Sie zunächst die Einschränkungen der minimalen und maximalen Anzahl von Cluster-Nodes in jedem einzelnen HA-Cluster und bestimmen Sie, ob Sie Nodes zum vorhandenen HA-Cluster hinzufügen oder ein neues HA-Cluster erstellen sollten. Normalerweise besteht jeder Baustein aus zwei Datei-Nodes, aber drei Nodes sind die Mindestanzahl an Nodes pro Cluster (um ein Quorum zu schaffen). Zehn davon ist das empfohlene Maximum (getestete). Für erweiterte Szenarien ist es möglich, einen einzelnen „Tiebreaker“ Node hinzuzufügen, auf dem keine BeeGFS-Services ausgeführt werden, wenn ein Cluster mit zwei Nodes implementiert wird. Bitte wenden Sie sich an den NetApp Support, wenn Sie eine solche Implementierung in Betracht ziehen.

Beachten Sie diese Einschränkungen und das erwartete zukünftige Cluster-Wachstum bei Ihrer Entscheidung über das erweitern des Clusters. Wenn Sie beispielsweise einen sechs-Node-Cluster haben und vier weitere Nodes hinzufügen müssen, empfiehlt es sich, nur einen neuen HA-Cluster zu starten.

Denken Sie daran, dass ein einziges BeeGFS-Dateisystem aus mehreren unabhängigen HA-Clustern bestehen kann. Dadurch können Filesysteme weit über die empfohlenen/harten Grenzen der zugrunde liegenden HA-Cluster-Komponenten hinaus skaliert werden.

Schritte

Wenn Sie dem Cluster einen Baustein hinzufügen, müssen Sie die `host_vars` Dateien für jeden der neuen Datei-Nodes und Block-Nodes (E-Series-Arrays) erstellen. Die Namen dieser Hosts müssen dem Bestand hinzugefügt werden, zusammen mit den neuen Ressourcen, die erstellt werden sollen. Die entsprechenden `group_vars` Dateien müssen für jede neue Ressource erstellt werden. "Nutzung benutzerdefinierter Architekturen"Weitere Informationen finden Sie im Abschnitt.

Nach dem Erstellen der richtigen Dateien müssen alle erforderlichen Dateien die Automatisierung mit dem Befehl erneut ausführen:

```
ansible-playbook -i <inventory>.yml <playbook>.yml
```

Entfernen eines Bausteins aus dem Cluster

Beachten Sie bei der Außerbetriebnahme eines Baublocks verschiedene Aspekte, z. B.:

- Welche BeeGFS-Services laufen in diesem Baustein?
- Werden nur die File-Nodes ausgemustert und die Block-Nodes mit neuen Datei-Nodes verbunden?
- Wenn der gesamte Baustein außer Betrieb genommen wird, sollten die Daten in einen neuen Baustein verschoben, in vorhandene Nodes im Cluster verteilt oder auf ein neues BeeGFS Filesystem oder ein anderes Storage-System verschoben werden?
- Kann dies bei einem Ausfall oder ohne Unterbrechung geschehen?
- Ist der Baustein aktiv genutzt oder enthält er in erster Linie Daten, die nicht mehr aktiv sind?

Aufgrund der vielfältigen möglichen Ausgangspunkte und gewünschten Endzustände wenden Sie sich bitte an den NetApp Support, damit wir die optimale Strategie basierend auf Ihrer Umgebung und Ihren Anforderungen identifizieren und implementieren können.

Fehlerbehebung

Fehlerbehebung für ein BeeGFS HA-Cluster.

Überblick

In diesem Abschnitt wird erläutert, wie verschiedene Fehler und andere Szenarien untersucht und behoben werden können, die beim Betrieb eines BeeGFS HA-Clusters auftreten können.

Leitfäden Zur Fehlerbehebung

Untersuchen Unerwarteter Failover

Wenn ein Node unerwartet eingezäunt ist und seine Services auf einen anderen Node verschoben werden, sollte der erste Schritt darin bestehen, zu überprüfen, ob das Cluster auf mögliche Ressourcenausfälle an der Unterseite von hinweist `pcs status`. Normalerweise gibt es keine Daten, wenn das Fechten erfolgreich abgeschlossen wurde und die Ressourcen auf einem anderen Knoten neu gestartet wurden.

Im Allgemeinen wird der nächste Schritt sein, durch die systemd-Logs mit zu suchen `journalctl` Auf einem beliebigen der übrigen Dateiknoten (Pacemaker-Protokolle werden auf allen Knoten synchronisiert). Wenn Sie wissen, wann der Fehler aufgetreten ist, können Sie die Suche kurz vor dem Auftreten des Fehlers starten (in der Regel mindestens zehn Minuten vor dem Auftreten des Fehlers empfohlen):

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>"
```

Die folgenden Abschnitte zeigen einen gemeinsamen Text, den Sie in den Protokollen grep können, um die Untersuchung weiter einzugrenzen.

Schritte zur Untersuchung/Lösung

Schritt 1: Prüfen, ob der BeeGFS-Monitor einen Fehler festgestellt hat:

Wenn das Failover vom BeeGFS-Monitor ausgelöst wurde, sollte ein Fehler angezeigt werden (wenn nicht mit dem nächsten Schritt fortfahren).

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i unexpected
[...]
Jul 01 15:51:03 beegfs_01 pacemaker-schedulerd[9246]:  warning: Unexpected
result (error: BeeGFS service is not active!) was recorded for monitor of
meta_08-monitor on beegfs_02 at Jul  1 15:51:03 2022
```

In diesem Fall hat der BeeGFS-Service meta_08 aus irgendeinem Grund gestoppt. Um mit der Fehlerbehebung fortzufahren, sollten wir beegfs_02 booten und Protokolle für den Dienst unter überprüfen `/var/log/beegfs-meta-meta_08_tgt_0801.log`. Beispiel: Aufgrund eines internen Problems oder eines Problems mit dem Node konnte für den BeeGFS-Service ein Applikationsfehler aufgetreten sein.

Im Gegensatz zu den Protokollen von Pacemaker werden Protokolle für BeeGFS-Services nicht auf alle Knoten im Cluster verteilt. Um diese Arten von Ausfällen zu untersuchen, sind die Protokolle vom ursprünglichen Knoten, auf dem der Fehler aufgetreten ist, erforderlich.

Mögliche Fehler, die vom Monitor gemeldet werden könnten:

- Auf Ziel(e) kann(n) nicht zugegriffen werden!
  - Beschreibung: Gibt an, auf die Block-Volumes nicht zugegriffen werden konnte.
  - Fehlerbehebung:
    - Wenn auch der Service am alternativen Datei-Node nicht gestartet werden konnte, vergewissern Sie sich, dass der Block-Node ordnungsgemäß ist.
    - Prüfen Sie auf physische Probleme, die den Zugriff auf die Block-Nodes durch diesen Datei-Node verhindern würden, z. B. fehlerhafte InfiniBand-Adapter oder Kabel.
- Netzwerk ist nicht erreichbar!
  - Beschreibung: Keiner der Adapter, die von Clients verwendet wurden, um sich mit diesem BeeGFS-Dienst zu verbinden, war online.
  - Fehlerbehebung:
    - Wenn mehrere/alle Dateiknoten betroffen waren, überprüfen Sie, ob ein Fehler im Netzwerk vorhanden ist, das zum Verbinden der BeeGFS-Clients und des Dateisystems verwendet wurde.
    - Prüfen Sie, ob physikalische Probleme den Zugriff auf die Clients durch diesen Dateiknoten verhindern würden, z. B. fehlerhafte InfiniBand-Adapter oder Kabel.
- BeeGFS-Service ist nicht aktiv!
  - Beschreibung: Ein BeeGFS-Dienst hat unerwartet gestoppt.
  - Fehlerbehebung:
    - Überprüfen Sie auf dem Datei-Node, der den Fehler gemeldet hat, die Protokolle für den betroffenen BeeGFS-Dienst, ob er einen Absturz gemeldet hat. Öffnen Sie in diesem Fall einen Fall mit NetApp Support, damit der Absturz untersucht werden kann.
    - Wenn im BeeGFS-Protokoll keine Fehler gemeldet werden, prüfen Sie in den Journalprotokollen, ob systemd einen Grund protokolliert hat, warum der Dienst angehalten wurde. In einigen Fällen wurde dem BeeGFS-Dienst möglicherweise keine Chance gegeben, Nachrichten zu protokollieren, bevor der Prozess beendet wurde (z. B. wenn jemand ausgeführt wurde `kill -9 <PID>`).

Schritt 2: Prüfen Sie, ob der Node das Cluster unerwartet verlassen hat

Falls auf dem Node ein schwerwiegender Hardware-Ausfall auftritt (z. B. die Systemplatine gestorben) oder ein Kernel-Panic oder ein ähnliches Softwareproblem auftritt, wird der BeeGFS-Monitor keinen Fehler melden. Suchen Sie stattdessen nach dem Hostnamen und Sie sollten Meldungen von Pacemaker sehen, die darauf hinweisen, dass der Knoten unerwartet verloren gegangen ist:

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i <HOSTNAME>
[...]
Jul 01 16:18:01 beegfs_01 pacemaker-attrd[9245]:  notice: Node beegfs_02
state is now lost
Jul 01 16:18:01 beegfs_01 pacemaker-controld[9247]:  warning:
Stonith/shutdown of node beegfs_02 was not expected
```

Schritt 3: Überprüfen Sie, ob Pacemaker in der Lage war, den Knoten einzuzäunen

In allen Szenarien sollten Sie sehen, dass Pacemaker versucht, den Knoten einzuzäunen, um zu überprüfen, ob er tatsächlich offline ist (genaue Meldungen können von der Ursache des Fechts abweichen):

```
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Cluster
node beegfs_02 will be fenced: peer is no longer part of the cluster
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Node
beegfs_02 is unclean
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Scheduling
Node beegfs_02 for STONITH
```

Wenn die Fechtaktion erfolgreich abgeschlossen ist, werden folgende Meldungen angezeigt:

```
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
[2214070] (call 27 from pacemaker-controld.9247) for host 'beegfs_02' with
device 'fence_redfish_2' returned: 0 (OK)
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
targeting beegfs_02 on beegfs_01 for pacemaker-
controld.9247@beegfs_01.786df3a1: OK
Jul 01 16:18:14 beegfs_01 pacemaker-controld[9247]:  notice: Peer
beegfs_02 was terminated (off) by beegfs_01 on behalf of pacemaker-
controld.9247: OK
```

Wenn die Fechten-Aktion aus irgendeinem Grund fehlgeschlagen ist, können die BeeGFS-Dienste auf einem anderen Node nicht neu starten, um Datenkorruption zu vermeiden. Das wäre ein Problem, separat zu untersuchen, wenn zum Beispiel das Fechten-Gerät (PDU oder BMC) unzugänglich oder falsch konfiguriert war.

Adressen fehlgeschlagener Ressourcen Aktionen (am Ende des Stk-Status gefunden)

Wenn eine Ressource, die zum Ausführen eines BeeGFS-Dienstes erforderlich ist, ausfällt, wird ein Failover durch den BeeGFS-Monitor ausgelöst. Wenn dies der Fall ist, werden wahrscheinlich keine „fehlgeschlagenen Ressourcenaktionen“ am Ende von aufgeführt `pcs status`, und Sie sollten die Schritte zum Thema "Failback nach einem ungeplanten Failover"lesen.

Ansonsten sollte es in der Regel nur zwei Szenarien geben, in denen Sie „Aktionen für fehlgeschlagene Ressourcen“ sehen.

Schritte zur Untersuchung/Lösung

Szenario 1: Bei einem Fechten-Agent wurde ein temporäres oder dauerhaftes Problem erkannt und es wurde neu gestartet oder auf einen anderen Knoten verschoben.

Einige Fechten-Agenten sind zuverlässiger als andere, und jeder implementiert seine eigene Überwachungsmethode, um sicherzustellen, dass die Fechtvorrichtung bereit ist. Insbesondere wurde festgestellt, dass der Fechtagent von Redfish fehlgeschlagene Ressourcenaktionen wie die folgenden meldet, obwohl er immer noch gestartet wird:

```
  * fence_redfish_2_monitor_60000 on beegfs_01 'not running' (7):
call=2248, status='complete', exitreason='', last-rc-change='2022-07-26
08:12:59 -05:00', queued=0ms, exec=0ms
```

Ein Fechten-Agent, der fehlgeschlagene Ressourcen-Aktionen auf einem bestimmten Knoten meldet, wird nicht erwartet, dass ein Failover der BeeGFS-Dienste ausgelöst wird, die auf diesem Knoten ausgeführt werden. Es sollte einfach automatisch auf demselben oder einem anderen Knoten neu gestartet werden.

Schritte zur Lösung:

1. Wenn der Fechtagent sich immer wieder weigert, auf allen oder einer Untermenge von Knoten ausgeführt zu werden, überprüfen Sie, ob diese Knoten eine Verbindung zum Fechtagenten herstellen können, und überprüfen Sie, ob der Fechtagent im Ansible-Bestand korrekt konfiguriert ist.
   a. Wenn z. B. ein Fechten-Agent von Redfish (BMC) auf demselben Knoten ausgeführt wird, wie er für das Fechten verantwortlich ist, und die Betriebssystemverwaltung und BMC-IPs auf derselben physischen Schnittstelle sind, ermöglichen einige Netzwerk-Switch-Konfigurationen keine Kommunikation zwischen den beiden Schnittstellen (um Netzwerkschleifen zu verhindern). Standardmäßig versucht das HA-Cluster, keine Fechten-Agenten auf dem Node zu platzieren, den sie für Fechten verantwortlich sind, aber dies kann in einigen Szenarien/Konfigurationen geschehen.
2. Sobald alle Probleme behoben sind (oder das Problem scheinbar kurzlebig zu sein schien), führen Sie den folgenden Lauf aus `pcs resource cleanup` So setzen Sie die fehlgeschlagenen Ressourcenaktionen zurück.

Szenario 2: Der BeeGFS-Monitor hat ein Problem erkannt und ein Failover ausgelöst, aber aus irgendeinem Grund konnte das System nicht auf einem sekundären Knoten starten.

Sofern das Fechten aktiviert ist und die Ressource nicht vom Stoppen auf dem ursprünglichen Knoten blockiert wurde (siehe Abschnitt Fehlerbehebung für „Standby (on-fail)“)), sind die wahrscheinlichsten Gründe, warum Probleme auftreten, die die Ressource auf einem sekundären Knoten zu starten, weil:

- Der sekundäre Node war bereits offline.
- Ein physisches oder logisches Konfigurationsproblem verhindert, dass das sekundäre System auf die als BeeGFS-Ziele verwendeten Block-Volumes zugreift.

Schritte zur Lösung:

1. Für jeden Eintrag in den Aktionen für fehlgeschlagene Ressourcen:
   a. Bestätigen Sie, dass die fehlgeschlagene Ressourcenaktion ein Startvorgang war.
   b. Basierend auf der in den Aktionen für fehlgeschlagene Ressourcen angegebenen Ressource und dem in den Knoten angegebenen Ressource:

i. Suchen Sie nach externen Problemen, die verhindern würden, dass der Knoten die angegebene Ressource startet, und beheben Sie diese. Wenn zum Beispiel BeeGFS IP-Adresse (Floating IP) nicht gestartet werden konnte, vergewissern Sie sich, dass mindestens eine der erforderlichen Schnittstellen angeschlossen/online ist und mit dem richtigen Netzwerk-Switch verbunden ist. Wenn ein BeeGFS-Ziel (Blockgerät/E-Series-Volume) fehlgeschlagen ist, überprüfen Sie, ob die physischen Verbindungen zu den Backend-Block-Nodes wie erwartet verbunden sind, und überprüfen Sie, ob die Block-Nodes ordnungsgemäß sind.

c. Wenn es keine offensichtlichen externen Probleme gibt und Sie eine Ursache für diesen Vorfall wünschen, sollten Sie einen Case mit dem NetApp Support eröffnen, um ihn zu untersuchen, bevor Sie fortfahren, da die folgenden Schritte eine Ursachenanalyse (Root Cause Analysis, RCA) schwierig/unmöglich machen können.

2. Nach der Lösung externer Probleme:

a. Kommentieren Sie alle nicht funktionierenden Nodes aus der Ansible Inventory.yml-Datei und führen Sie das vollständige Ansible-Playbook erneut aus, um sicherzustellen, dass die logische Konfiguration auf den/den sekundären Nodes korrekt eingerichtet ist.

i. Hinweis: Vergessen Sie nicht, diese Nodes zu kommentieren und das Playbook erneut auszuführen, sobald sich die Nodes in einem ordnungsgemäßen Zustand befinden und Sie zum Failback bereit sind.

b. Alternativ können Sie versuchen, das Cluster manuell wiederherzustellen:

i. Platzieren Sie alle Offline-Nodes wieder online mithilfe von: `pcs cluster start <HOSTNAME>`

ii. Löschen Sie alle fehlgeschlagenen Ressourcenaktionen mit: `pcs resource cleanup`

iii. Stk-Status ausführen und überprüfen, ob alle Dienste wie erwartet beginnen.

iv. Bei Bedarf ausführen `pcs resource relocate run` Verschieben von Ressourcen zurück auf den bevorzugten Node (sofern verfügbar)

Häufige Probleme

BeeGFS-Services führen bei Anforderung kein Failover oder Failback durch

Wahrscheinliche Ausgabe: das `pcs resource relocate` Befehl ausführen wurde ausgeführt, aber nie erfolgreich abgeschlossen.

So überprüfen Sie: Lauf `pcs constraint --full` Und überprüfen Sie auf alle Standortbeschränkungen mit einer ID von `pcs-relocate-<RESOURCE>`.

Wie löst man: Lauf `pcs resource relocate clear` Wiederholen Sie anschließend den Test `pcs constraint --full` Um zu überprüfen, ob die zusätzlichen Bedingungen entfernt wurden.

Ein Knoten im Stk-Status zeigt „Standby (ein-aus)“ an, wenn das Fechten deaktiviert ist

Wahrscheinliche Ursache: Pacemaker konnte nicht erfolgreich bestätigen, dass alle Ressourcen auf dem Knoten, der ausgefallen ist, angehalten wurden.

Wie löst man:

1. Laufen `pcs status` Und überprüfen Sie, ob die Ressourcen nicht „gestartet“ sind, oder zeigen Sie Fehler an der Unterseite der Ausgabe an, und beheben Sie eventuelle Probleme.
2. Um den Node wieder in den Online-Modus zu versetzen, wird ausgeführt `pcs resource cleanup --node=<HOSTNAME>`.

Nach einem unerwarteten Failover zeigen die Ressourcen „gestartet (ein-Fehler)“ im Stk-Status an, wenn das Fechten aktiviert ist

Wahrscheinliches Problem: Es trat ein Problem auf, das einen Failover auslöste, Pacemaker konnte jedoch nicht überprüfen, ob der Knoten eingezäunt war. Dies kann passieren, weil Fechten falsch konfiguriert war oder es ein Problem mit dem Fechten Agent gab (Beispiel: Die PDU wurde vom Netzwerk getrennt).

Wie löst man:

1. Vergewissern Sie sich, dass der Node tatsächlich ausgeschaltet ist.

   Wenn der von Ihnen angegebene Node nicht aktiv ist, der aber Cluster-Services oder -Ressourcen ausführt, treten Datenbeschädigungen/Cluster-Ausfälle auf.

2. Fechten manuell bestätigen mit: `pcs stonith confirm <NODE>`

An diesem Punkt sollten die Dienste den Failover beenden und auf einem anderen gesunden Knoten neu gestartet werden.

Häufige Fehlerbehebungsaufgaben

Starten Sie individuelle BeeGFS-Dienste neu

Normalerweise, wenn ein BeeGFS-Service neu gestartet werden muss (z. B. um eine Konfigurationsänderung zu ermöglichen), sollte dies durch Aktualisierung des Ansible-Bestands und durch erneute Ausführung des Playbooks geschehen. In manchen Szenarien kann es wünschenswert sein, einzelne Services neu zu starten, um eine schnellere Fehlerbehebung zu ermöglichen, beispielsweise um das Protokollierungsniveau zu ändern, ohne auf die Ausführung des gesamten Playbooks zu warten.

Wenn nicht auch manuelle Änderungen am Ansible-Inventar hinzugefügt werden, werden diese bei der nächsten Ausführung des Ansible-Playbooks zurückgesetzt.

Option 1: Systemgesteuerter Neustart

Wenn das Risiko besteht, dass der BeeGFS-Service mit der neuen Konfiguration nicht ordnungsgemäß neu gestartet wird, versetzen Sie das Cluster zuerst in den Wartungsmodus, um zu verhindern, dass der BeeGFS-Monitor den Service erkennt, angehalten wird und ein unerwünschtes Failover ausgelöst wird:

```
pcs property set maintenance-mode=true
```

Nehmen Sie ggf. Änderungen an der Servicekonfiguration unter vor `/mnt/<SERVICE_ID>/_config/beegfs-.conf` (Beispiel: `/mnt/meta_01_tgt_0101/metadata_config/beegfs-meta.conf`) Dann systemd verwenden, um es neu zu starten:

```
systemctl restart beegfs-*@<SERVICE_ID>.service
```

Beispiel: `systemctl restart beegfs-meta@meta_01_tgt_0101.service`

Option 2: Schrittmachergesteuerter Neustart

Wenn Sie keine Sorge haben, dass die neue Konfiguration dazu führen könnte, dass der Service unerwartet angehalten wird (z. B. einfach die Protokollierungsebene ändern), oder Sie sich in einem Wartungsfenster befinden und sich keine Gedanken über Ausfallzeiten machen, können Sie den BeeGFS-Monitor einfach für den Service neu starten, den Sie neu starten möchten:

```
pcs resource restart <SERVICE>-monitor
```

Zum Beispiel zum Neustart des BeeGFS-Managementdienstes: `pcs resource restart mgmt-monitor`

Rechtliche Hinweise

Rechtliche Hinweise ermöglichen den Zugriff auf Copyright-Erklärungen, Marken, Patente und mehr.

Urheberrecht

"https://www.netapp.com/company/legal/copyright/"

Marken

NetApp, das NETAPP Logo und die auf der NetApp Markenseite aufgeführten Marken sind Marken von NetApp Inc. Andere Firmen- und Produktnamen können Marken der jeweiligen Eigentümer sein.

"https://www.netapp.com/company/legal/trademarks/"

Patente

Eine aktuelle Liste der NetApp Patente finden Sie unter:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Datenschutzrichtlinie

"https://www.netapp.com/company/legal/privacy-policy/"

Open Source

In den Benachrichtigungsdateien finden Sie Informationen zu Urheberrechten und Lizenzen von Drittanbietern, die in der NetApp Software verwendet werden.

"Hinweis zum SANtricity OS für die E-Series/EF-Series"