BeeGFS su NetApp con storage e-Series

BeeGFS on NetApp with E-Series Storage

NetApp
March 21, 2024

This PDF was generated from https://docs.netapp.com/it-it/beegfs/index.html on March 21, 2024. Always check docs.netapp.com for the latest.

Sommario

BeeGFS su NetApp con storage e-Series

Inizia subito

Contenuto di questo sito

Questo sito descrive come implementare e gestire BeeGFS su NetApp utilizzando architetture NetApp Verified Architectures (NVA) e architetture personalizzate. I progetti NVA sono testati a fondo e offrono ai clienti configurazioni di riferimento e indicazioni per il dimensionamento, in modo da ridurre al minimo i rischi di implementazione e accelerare il time-to-market. NetApp supporta inoltre architetture BeeGFS personalizzate eseguite su hardware NetApp, offrendo a clienti e partner flessibilità nella progettazione di file system per soddisfare un'ampia gamma di requisiti. Entrambi gli approcci sfruttano Ansible per l'implementazione, offrendo un approccio simile ad appliance per la gestione di BeeGFS su qualsiasi scala in una gamma flessibile di hardware.

Termini e concetti

I seguenti termini e concetti si applicano alla soluzione BeeGFS su NetApp.

Vedere "Amministrare i cluster BeeGFS" Sezione per ulteriori dettagli su termini e concetti specifici per l'interazione con cluster BeeGFS ad alta disponibilità (ha).

Termine	Descrizione
AI	Intelligenza artificiale.
Inventario di Ansible	Struttura di directory contenente file YAML utilizzati per descrivere il cluster BeeGFS ha desiderato.
BMC	Controller di gestione della baseboard. Talvolta indicato come processore di servizi.
nodi a blocchi	Sistemi storage.
client	Nodi nel cluster HPC che eseguono applicazioni che devono utilizzare il file system. A volte chiamato anche nodi di calcolo o GPU.
DL	Deep Learning.
nodi di file	File server BeeGFS.
HA	Alta disponibilità.
HIC	Scheda di interfaccia host.
HPC	High-Performance Computing.

Termine	Descrizione
Workload di tipo HPC	I carichi di lavoro in stile HPC sono in genere caratterizzati da più nodi di calcolo o GPU che hanno tutti bisogno di accedere allo stesso dataset in parallelo per facilitare un calcolo distribuito o un lavoro di training. Questi set di dati sono spesso costituiti da file di grandi dimensioni che devono essere sottoposti a striping su più nodi di storage fisici per eliminare i tradizionali colli di bottiglia hardware che impedirebbero l'accesso simultaneo a un singolo file.
ML	Apprendimento automatico.
NLP	Elaborazione del linguaggio naturale.
NLU	Comprensione del linguaggio naturale.
NVA	Il programma NetApp Verified Architecture (NVA) fornisce configurazioni di riferimento e indicazioni per il dimensionamento per carichi di lavoro e casi di utilizzo specifici. Queste soluzioni sono testate a fondo e sono progettate per ridurre al minimo i rischi di implementazione e accelerare il time-to-market.
rete storage/rete client	Rete utilizzata dai client per comunicare con il file system BeeGFS. Si tratta spesso della stessa rete utilizzata per la comunicazione MPI (Parallel message Passing Interface) e altre comunicazioni applicative tra i nodi del cluster HPC.

Utilizzare architetture verificate

Panoramica e requisiti

Panoramica della soluzione

La soluzione BeeGFS su NetApp combina il file system parallelo BeeGFS con i sistemi storage NetApp EF600 per un'infrastruttura affidabile, scalabile e conveniente che tiene il passo con i carichi di lavoro esigenti.

Questo design sfrutta la densità delle performance offerta dalle più recenti velocità di rete e hardware per server e storage Enterprise, Richiede file node dotati di doppi processori AMD EPYC 7003 "Milano" e supporto per PCIe 4.0 con connessioni dirette che utilizzano InfiniBand da 200 GB (HDR) per bloccare i nodi che forniscono NVMe e NVMeOF end-to-end utilizzando il protocollo NVMe/IB.

Programma NVA

La soluzione BeeGFS su NetApp fa parte del programma NetApp Verified Architecture (NVA), che offre ai clienti configurazioni di riferimento e indicazioni sul dimensionamento per carichi di lavoro e casi di utilizzo specifici. Le soluzioni NVA sono testate e progettate per ridurre al minimo i rischi di implementazione e accelerare il time-to-market.

Casi di utilizzo

I seguenti casi di utilizzo si applicano alla soluzione BeeGFS su NetApp:

- Intelligenza artificiale (ai), tra cui apprendimento automatico (ML), apprendimento approfondito (DL), elaborazione del linguaggio naturale (NLP) su larga scala e comprensione del linguaggio naturale (NLU). Per ulteriori informazioni, vedere "BeeGFS per l'ai: Fatti e finzione".
- High-performance computing (HPC) che include applicazioni accelerate da MPI (message passing interface) e altre tecniche di calcolo distribuito. Per ulteriori informazioni, vedere "Perché BeeGFS va oltre l'HPC".
- Carichi di lavoro delle applicazioni caratterizzati da:
  - Lettura o scrittura su file di dimensioni superiori a 1 GB
  - Lettura o scrittura sullo stesso file da parte di più client (10, 100 e 1000)
- Set di dati multi-terabyte o multi-petabyte.
- Ambienti che richiedono un singolo spazio dei nomi dello storage, ottimizzabili per una combinazione di file di grandi e piccoli dimensioni.

Benefici

I vantaggi principali dell'utilizzo di BeeGFS su NetApp includono:

- Disponibilità di progetti hardware verificati che forniscono la completa integrazione di componenti hardware e software per garantire performance e affidabilità prevedibili.
- Implementazione e gestione con Ansible per semplicità e coerenza su larga scala.
- Monitoraggio e osservabilità forniti con e-Series Performance Analyzer e plug-in BeeGFS. Per ulteriori informazioni, vedere "Presentazione di un framework per il monitoraggio delle soluzioni NetApp e-Series".

- Alta disponibilità con un'architettura a disco condiviso che garantisce la durata e la disponibilità dei dati.
- Supporto per la moderna gestione e orchestrazione dei workload con container e Kubernetes. Per ulteriori informazioni, vedere "Kubernetes incontra BeeGFS: Una storia di investimenti a prova di futuro".

Architettura HA

BeeGFS su NetApp espande le funzionalità dell'edizione Enterprise di BeeGFS creando una soluzione completamente integrata con l'hardware NetApp che consente un'architettura ha (Shared Disk High Availability).

> Sebbene l'edizione della community BeeGFS possa essere utilizzata gratuitamente, l'edizione Enterprise richiede l'acquisto di un contratto di abbonamento al supporto professionale da parte di un partner come NetApp. L'edizione Enterprise consente di utilizzare diverse funzionalità aggiuntive, tra cui resilienza, applicazione delle quote e pool di storage.

La figura seguente confronta le architetture ha shared-nothing e shared-disk.

[]

Per ulteriori informazioni, vedere "Annuncio dell'alta disponibilità per BeeGFS supportato da NetApp".

Ansible

BeeGFS su NetApp viene fornito e implementato utilizzando l'automazione Ansible, che è ospitata su GitHub e Ansible Galaxy (la raccolta BeeGFS è disponibile presso "Ansible Galaxy" e. "GitHub e-Series di NetApp"). Sebbene Ansible sia testato principalmente con l'hardware utilizzato per assemblare i blocchi di base BeeGFS, è possibile configurarlo per l'esecuzione su qualsiasi server basato su x86 utilizzando una distribuzione Linux supportata.

Per ulteriori informazioni, vedere "Implementazione di BeeGFS con storage e-Series".

Generazioni di design

La soluzione BeeGFS su NetApp si trova attualmente nella sua seconda progettazione generazionale.

Sia la prima che la seconda generazione includono un'architettura di base che incorpora un file system BeeGFS e un sistema storage NVMe EF600. Tuttavia, la seconda generazione si basa sulla prima per includere questi benefici aggiuntivi:

- Raddoppiare le prestazioni e la capacità aggiungendo solo 2U di spazio rack
- Alta disponibilità (ha) basata su un design hardware a due livelli e a disco condiviso
- Qualifica esterna per architetture NVIDIA DGX A100 SuperPOD e NVIDIA BasePOD

Secondo design generazionale

La seconda generazione di BeeGFS su NetApp è ottimizzata per soddisfare i requisiti di performance dei carichi di lavoro più esigenti, tra cui HPC (high-performance computing) e ML (machine learning) di tipo HPC, DL (deep learning) e tecniche di intelligenza artificiale (ai) simili. Incorporando un'architettura ad alta disponibilità (ha) a dischi condivisi, la soluzione BeeGFS su NetApp soddisfa anche i requisiti di durata e disponibilità dei dati di aziende e altre organizzazioni che non possono permettersi downtime o perdita di dati in cerca di storage scalabile per tenere il passo con i propri carichi di lavoro e casi di utilizzo. Questa soluzione

non solo è stata verificata da NetApp, ma ha anche superato la qualifica esterna come opzione di storage per NVIDIA DGX SuperPOD e DGX BasePOD.

Primo design generazionale

La prima generazione di BeeGFS su NetApp è stata progettata per i carichi di lavoro di apprendimento automatico (ML) e intelligenza artificiale (ai) utilizzando i sistemi storage NetApp EF600 NVMe, il file system parallelo BeeGFS, i sistemi NVIDIA DGX™ A100 e gli switch NVIDIA® Mellanox® Quantum™ QM8700 200Gbps IB. Questo design include anche InfiniBand (IB) a 200 Gbps per il fabric di interconnessione di cluster di calcolo e storage, per offrire un'architettura completamente basata su IB per carichi di lavoro dalle performance elevate.

Per ulteriori informazioni sulla prima generazione, vedere "NetApp EF-Series ai con sistemi NVIDIA DGX A100 e BeeGFS".

Panoramica dell'architettura

La soluzione BeeGFS su NetApp include considerazioni di progettazione architetturale utilizzate per determinare le apparecchiature, il cablaggio e le configurazioni specifiche richieste per supportare i carichi di lavoro validati.

Architettura a blocchi

Il file system BeeGFS può essere implementato e scalato in diversi modi, a seconda dei requisiti di storage. Ad esempio, i casi di utilizzo che includono principalmente numerosi file di piccole dimensioni trarranno beneficio da una maggiore capacità e performance dei metadati, mentre i casi di utilizzo che presentano meno file di grandi dimensioni potrebbero favorire una maggiore capacità di storage e performance per i contenuti dei file effettivi. Queste considerazioni multiple hanno un impatto sulle diverse dimensioni dell'implementazione del file system parallelo, che aggiunge complessità alla progettazione e all'implementazione del file system.

Per affrontare queste sfide, NetApp ha progettato un'architettura standard basata su building block che consente di scalare ciascuna di queste dimensioni. In genere, gli elementi di base BeeGFS vengono implementati in uno dei tre profili di configurazione seguenti:

- Un singolo building block di base, che include gestione BeeGFS, metadati e servizi di storage
- Metadati BeeGFS e building block dello storage
- Un building block per lo storage BeeGFS

L'unica modifica hardware tra queste tre opzioni è l'utilizzo di dischi più piccoli per i metadati BeeGFS. In caso contrario, tutte le modifiche di configurazione vengono applicate tramite software. Inoltre, con Ansible come motore di implementazione, la configurazione del profilo desiderato per un particolare building block rende semplici le attività di configurazione.

Per ulteriori dettagli, vedere Progettazione hardware verificata.

Servizi del file system

Il file system BeeGFS include i seguenti servizi principali:

- **Servizio di gestione.** registra e monitora tutti gli altri servizi.
- **Servizio di storage.** Memorizza il contenuto del file utente distribuito, noto come file di blocco dei dati.
- **Servizio metadati.** tiene traccia del layout del file system, della directory, degli attributi dei file e così via.

- **Servizio client.** consente di attivare il file system per accedere ai dati memorizzati.

La figura seguente mostra i componenti della soluzione BeeGFS e le relazioni utilizzate con i sistemi NetApp e-Series.

[]

In qualità di file system parallelo, BeeGFS esegue lo striping dei propri file su più nodi server per massimizzare le performance di lettura/scrittura e la scalabilità. I nodi server lavorano insieme per fornire un singolo file system che può essere montato e accessibile contemporaneamente da altri nodi server, comunemente noti come *client*. Questi client possono visualizzare e utilizzare il file system distribuito in modo simile a un file system locale come NTFS, XFS o ext4.

I quattro servizi principali vengono eseguiti su un'ampia gamma di distribuzioni Linux supportate e comunicano tramite qualsiasi rete compatibile con TCP/IP o RDMA, tra cui InfiniBand (IB), Omni-Path (OPA) e RDMA over Converged Ethernet (RoCE). I servizi server BeeGFS (gestione, storage e metadati) sono daemon dello spazio utente, mentre il client è un modulo kernel nativo (senza patch). Tutti i componenti possono essere installati o aggiornati senza riavviare ed è possibile eseguire qualsiasi combinazione di servizi sullo stesso nodo.

Nodi verificati

La soluzione BeeGFS su NetApp include i seguenti nodi verificati: Il sistema di storage NetApp EF600 (nodo a blocchi) e il server Lenovo ThinkSystem SR665 (nodo file).

Nodo a blocchi: Sistema storage EF600

L'array all-flash NetApp EF600 offre un accesso coerente e quasi in tempo reale ai dati, supportando contemporaneamente qualsiasi numero di workload. Per consentire un'alimentazione rapida e continua dei dati alle applicazioni ai e HPC, i sistemi storage EF600 offrono fino a due milioni di IOPS in lettura memorizzati nella cache, tempi di risposta inferiori a 100 microsecondi e larghezza di banda in lettura sequenziale di 42 Gbps in un'unica enclosure.

Nodo file: Server Lenovo ThinkSystem SR665

SR665 è un server 2U a due socket dotato di PCIe 4.0. Se configurato per soddisfare i requisiti di questa soluzione, offre prestazioni elevate per l'esecuzione dei file service BeeGFS in una configurazione ben bilanciata con la disponibilità di throughput e IOPS forniti dai nodi e-Series collegati direttamente.

Per ulteriori informazioni su Lenovo SR665, vedere "Il sito Web di Lenovo".

Progettazione hardware verificata

Gli elementi di base della soluzione (illustrati nella figura seguente) utilizzano due server PCIe 4.0 a doppio socket per il file layer BeeGFS e due sistemi storage EF600 come layer di blocchi.

[]

> Poiché ogni building block include due nodi di file BeeGFS, per stabilire il quorum nel cluster di failover sono necessari almeno due building block. Sebbene sia possibile configurare un cluster a due nodi, questa configurazione presenta dei limiti che potrebbero impedire il failover. Se si richiede un cluster a due nodi, è possibile incorporare un terzo dispositivo come spareggio (tuttavia, tale progetto non è trattato in questo sito).

Ogni building block offre alta disponibilità attraverso una progettazione hardware a due livelli che separa i domini di errore per i layer di file e blocchi. Ogni Tier può eseguire il failover in modo indipendente, fornendo una maggiore resilienza e riducendo il rischio di guasti a cascata. L'utilizzo di HDR InfiniBand in combinazione con NVMeOF offre un throughput elevato e una latenza minima tra file e nodi a blocchi, con ridondanza completa e un'oversubscription di collegamento sufficiente per evitare che il design disaggregato diventi un collo di bottiglia, anche quando il sistema è parzialmente degradato.

La soluzione BeeGFS su NetApp viene eseguita in tutti gli elementi di base dell'implementazione. Il primo building block implementato deve eseguire i servizi di gestione, metadati e storage BeeGFS (indicati come building block di base). Tutti gli elementi di base successivi vengono configurati tramite software per eseguire i metadati e i servizi storage BeeGFS o solo i servizi storage. La disponibilità di diversi profili di configurazione per ciascun building block consente la scalabilità dei metadati del file system o della capacità e delle performance dello storage utilizzando le stesse piattaforme hardware sottostanti e la stessa progettazione di building block.

Fino a cinque building block vengono combinati in un cluster Linux ha standalone, garantendo un numero ragionevole di risorse per cluster resource manager (pacemaker) e riducendo l'overhead di messaggistica richiesto per mantenere i membri del cluster sincronizzati (Corosync). Si consiglia di utilizzare almeno due building block per cluster per consentire a un numero sufficiente di membri di stabilire il quorum. Uno o più di questi cluster BeeGFS ha standalone vengono combinati per creare un file system BeeGFS (mostrato nella figura seguente) accessibile ai client come singolo namespace dello storage.

[]

Anche se in ultima analisi il numero di building block per rack dipende dai requisiti di alimentazione e raffreddamento per un determinato sito, La soluzione è stata progettata in modo da poter implementare fino a cinque building block in un singolo rack 42U, pur fornendo spazio per due switch InfiniBand 1U utilizzati per la rete storage/dati. Ogni building block richiede otto porte IB (quattro per switch per la ridondanza), quindi cinque building block lasciano metà delle porte su uno switch InfiniBand HDR a 40 porte (come NVIDIA QM8700) disponibile per implementare una topologia fat-tree o simile senza blocchi. Questa configurazione garantisce che il numero di rack di storage o di calcolo/GPU possa essere aumentato senza colli di bottiglia nella rete. In alternativa, è possibile utilizzare un fabric di storage con overoveroveroveroveroversured su raccomandazione del vendor del fabric di storage.

La seguente immagine mostra una topologia ad albero adiposo a 80 nodi.

[]

Utilizzando Ansible come motore di implementazione per implementare BeeGFS su NetApp, gli amministratori possono mantenere l'intero ambiente utilizzando l'infrastruttura moderna come procedure di codice. In questo modo si semplifica drasticamente quello che altrimenti sarebbe un sistema complesso, consentendo agli amministratori di definire e regolare la configurazione in un'unica posizione, garantendo che venga applicata in modo coerente indipendentemente dalla scalabilità dell'ambiente. La raccolta BeeGFS è disponibile all'interno del sito "Ansible Galaxy" e. "GitHub e-Series di NetApp".

Requisiti tecnici

Per implementare la soluzione BeeGFS su NetApp, assicurati che il tuo ambiente soddisfi i requisiti tecnologici.

Requisiti hardware

La seguente tabella elenca i componenti hardware necessari per implementare una singola progettazione di building block di seconda generazione della soluzione BeeGFS su NetApp.

I componenti hardware utilizzati in qualsiasi implementazione specifica della soluzione potrebbero variare in base ai requisiti del cliente.

Conta	Componente hardware	Requisiti
2	Nodi di file BeeGFS.	Per ottenere le prestazioni previste, ciascun nodo di file deve soddisfare o superare la seguente configurazione. **Processori:** • 2 processori AMD EPEYC 7343 16C da 3.2 GHz. • Configurato come due zone NUMA. **Memoria:** • 256 GB. • 16x 16 GB TruDDR4 da 200 MHz (2Rx8 1,2 V) RDIMM-A (preferiscono DIMM più piccoli rispetto a meno DIMM più grandi). • Popolato per massimizzare la larghezza di banda della memoria. **Espansione PCIe: Quattro slot PCE Gen4 x16:** • Due slot per zona NUMA. • Ogni slot deve fornire alimentazione/raffreddamento sufficiente per l'adattatore Mellanox MCX653106A-HDAT. **Varie:** • Due dischi SATA da 1 TB e 7.200 rpm (o simili) configurati in RAID 1 per il sistema operativo. • Adattatore OCP 3.0 10 GbE (o equivalente) per la gestione del sistema operativo in-band. • BMC 1GbE con API Redfish per la gestione dei server out-of-band. • Doppi alimentatori hot swap e ventole ad alte prestazioni. • Per raggiungere gli switch InfiniBand storage, è necessario supportare i cavi ottici Mellanox InfiniBand. **Lenovo SR665:** • Un modello NetApp personalizzato include la versione richiesta del firmware del controller XClarity necessaria per supportare gli adattatori Mellanox ConnectX-6 a due porte. Contatta NetApp per informazioni sugli ordini.
8	HCA Mellanox ConnectX-6 (per nodi di file).	• MCX653106A-HDAT host Channel Adapter (HDR IB 200 GB, QSFP56 a due porte, PCIe4,0 x16).

Conta	Componente hardware	Requisiti
8	Cavi HDR InfiniBand da 1 m (per connessioni dirette a nodi di file/blocchi).	• MCP1650-H001E30 (cavo di rame passivo Mellanox da 1 m, IB HDR, fino a 200 Gbps, QSFP56, 30 AWG). La lunghezza può essere regolata in modo da tenere conto delle distanze maggiori tra il file e i nodi di blocco, se necessario.
8	Cavi HDR InfiniBand (per connessioni nodo file/switch storage)	Richiede cavi InfiniBand HDR (ricetrasmettitori QSFP56) della lunghezza appropriata per collegare i nodi di file agli switch Leaf dello storage. Le opzioni possibili includono: • MCP1650-H002E26 (cavo di rame passivo Mellanox da 2 m, IB HDR, fino a 200 GB/s, QSFP56, 30 AWG). • MFS1S00-H003E (cavo in fibra attiva Mellanox da 3 m, IB HDR, fino a 200 GB/s, QSFP56).
2	Nodi a blocchi e-Series	Due controller EF600 configurati come segue: • Memoria: 256 GB (128 GB per controller). • Adattatore: 2 porte 200 GB/HDR (NVMe/IB). • Dischi: Configurati in modo da corrispondere alla capacità desiderata.

Requisiti software

Per ottenere performance e affidabilità prevedibili, le release della soluzione BeeGFS su NetApp vengono testate con versioni specifiche dei componenti software necessari per implementare la soluzione.

Requisiti di implementazione del software

La seguente tabella elenca i requisiti software implementati automaticamente nell'ambito dell'implementazione di Ansible-Based BeeGFS.

Software	Versione
BeeGFS	7.2.6
Corosync	3.1.5-1
Pacemaker	2.1.0-8
Opensm	opensm-5.9.0 (da mlnx_ofed 5.4-1.0.3.0) Necessario solo per le connessioni dirette per abilitare la virtualizzazione.

Requisiti dei nodi di controllo Ansible

La soluzione BeeGFS su NetApp viene implementata e gestita da un nodo di controllo Ansible. Per ulteriori informazioni, consultare "Documentazione Ansible".

I requisiti software elencati nelle tabelle seguenti sono specifici della versione della raccolta NetApp BeeGFS Ansible elencata di seguito.

Software	Versione
Ansible	2.11 installato tramite pip: ansible-4.7.0 e ansible-core < 2.12,>=2.11.6
Python	3.9
Pacchetti Python aggiuntivi	Crittografia-35.0.0, netaddr-0.8.0
BeeGFS Ansible Collection	3.0.0

Requisiti del nodo del file

Software	Versione
RedHat Enterprise Linux	RedHat 8.4 Server Physical con alta disponibilità (2 socket). I file node richiedono un abbonamento valido a RedHat Enterprise Linux Server e Red Hat Enterprise Linux High Availability Add-on.
Kernel Linux	4.18.0-305.25.1.el8_4.x86_64
Driver InfiniBand/RDMA	Posta in arrivo
Firmware ConnectX-6 HCA	PM: 20.31.1014
PXE: 3.6.0403	UEFI: 14.24.0013

Requisiti dei nodi a blocchi EF600

Software	Versione
Sistema operativo SANtricity	11.70.2
NVSRAM	N6000-872834-D06.dlp
Firmware del disco	Più recente disponibile per i modelli di unità in uso.

Requisiti aggiuntivi

Per la convalida sono state utilizzate le apparecchiature elencate nella seguente tabella, ma è possibile utilizzare le alternative appropriate in base alle necessità. In generale, NetApp consiglia di eseguire le versioni software più recenti per evitare problemi imprevisti.

Componente hardware	Software installato
• 2 switch Mellanox MQM8700 da 200 GB InfiniBand	• Firmware 3.9.2110

Componente hardware	Software installato
1x nodo di controllo Ansible (virtualizzato): • Processori: Intel® Xeon® Gold 6146 CPU @ 3,20 GHz • Memoria: 8 GB • Storage locale: 24 GB	• CentOS Linux 8.4.2105 • Kernel 4.18.0-305.3.1.el8.x86_64 Le versioni di Ansible e Python installate corrispondono a quelle della tabella precedente.
10x BeeGFS Client (nodi CPU): • Processore: 1 CPU AMD EPEYC a 7302 16 core a 3,0 GHz • Memoria: 128 GB • Rete: 2 Mellanox MCX653106A-HDAT (una porta collegata per adattatore).	• Ubuntu 20.04 • Kernel: 5.4.0-100-generic • Driver InfiniBand: Mellanox OFED 5.4-1.0.3.0
1x BeeGFS Client (nodo GPU): • Processori: 2 CPU AMD EPYC a 7742 64 core a 2,25 GHz • Memoria: 1 TB • Rete: 2 Mellanox MCX653106A-HDAT (una porta collegata per adattatore). Questo sistema è basato sulla piattaforma NVDIA HGX A100 e include quattro GPU A100.	• Ubuntu 20.04 • Kernel: 5.4.0-100-generic • Driver InfiniBand: Mellanox OFED 5.4-1.0.3.0

Esaminare la progettazione della soluzione

Panoramica del design

Per supportare la soluzione BeeGFS su NetApp, che combina il file system parallelo BeeGFS con i sistemi storage NetApp EF600, sono necessarie apparecchiature, cablaggi e configurazioni specifiche.

Scopri di più:

- "Configurazione dell'hardware"
- "Configurazione del software"
- "Verifica del progetto"
- "Linee guida per il dimensionamento"
- "Tuning delle performance"

Architetture derivate con variazioni di design e performance:

- "Building Block ad alta capacità"

Configurazione dell'hardware

La configurazione hardware per BeeGFS su NetApp include nodi di file e cablaggio di rete.

Configurazione del nodo del file

I file node hanno due socket CPU configurati come zone NUMA separate, che includono l'accesso locale a un numero uguale di slot PCIe e memoria.

Gli adattatori InfiniBand devono essere inseriti nei riser o negli slot PCI appropriati, in modo da bilanciare il carico di lavoro sulle corsie PCIe e sui canali di memoria disponibili. Il carico di lavoro viene bilanciato isolando completamente il lavoro per i singoli servizi BeeGFS in un nodo NUMA specifico. L'obiettivo è quello di ottenere performance simili da ogni nodo di file come se si trattasse di due server a socket singolo indipendenti.

La figura seguente mostra la configurazione NUMA del nodo del file.

[]

I processi BeeGFS vengono bloccati in una particolare zona NUMA per garantire che le interfacce utilizzate si trovino nella stessa zona. Questa configurazione evita la necessità di un accesso remoto sulla connessione tra socket. La connessione tra socket è talvolta nota come collegamento QPI o GMI2; anche nelle moderne architetture di processore, può essere un collo di bottiglia quando si utilizza una rete ad alta velocità come HDR InfiniBand.

Configurazione del cablaggio di rete

All'interno di un building block, ogni nodo di file è connesso a due nodi a blocchi utilizzando un totale di quattro connessioni InfiniBand ridondanti. Inoltre, ciascun nodo di file dispone di quattro connessioni ridondanti alla rete di storage InfiniBand.

Nella figura seguente, notare che:

- Tutte le porte dei nodi di file delineate in verde vengono utilizzate per la connessione al fabric di storage; tutte le altre porte dei nodi di file sono le connessioni dirette ai nodi di blocco.
- Due porte InfiniBand in una zona NUMA specifica si collegano ai controller A e B dello stesso nodo a blocchi.
- Le porte nel nodo NUMA 0 si connettono sempre al primo nodo a blocchi.
- Le porte nel nodo NUMA 1 si connettono al secondo nodo a blocchi.

[]

Per le reti di storage con switch ridondanti, le porte evidenziate in verde chiaro devono essere collegate a uno switch e le porte in verde scuro a un altro switch.

La configurazione del cablaggio illustrata nella figura consente a ciascun servizio BeeGFS di:

- Viene eseguito nella stessa zona NUMA indipendentemente dal nodo del file che esegue il servizio BeeGFS.
- Disporre di percorsi secondari ottimali per la rete di storage front-end e per i nodi a blocchi back-end, indipendentemente da dove si verifica un guasto.

- Ridurre al minimo gli effetti delle performance se un nodo di file o un controller in un nodo a blocchi richiede manutenzione.

Cablaggio per sfruttare la larghezza di banda

Per sfruttare l'intera larghezza di banda bidirezionale PCIe, assicurarsi che una porta di ciascun adattatore InfiniBand si colleghi al fabric di storage e l'altra porta si colleghi a un nodo a blocchi. La velocità massima teorica di una porta HDR InfiniBand è di 25 Gbps (senza tenere conto della segnalazione e di altri overhead). La larghezza di banda massima single-direction di uno slot PCIe 4.0 x16 è di 32 Gbps, creando un potenziale collo di bottiglia durante l'implementazione di nodi di file che incorporano adattatori InfiniBand a doppia porta in grado teoricamente di gestire 50 GBps di larghezza di banda.

La figura seguente mostra il design del cablaggio utilizzato per sfruttare l'intera larghezza di banda bidirezionale PCIe.

[]

Per ogni servizio BeeGFS, utilizzare lo stesso adattatore per connettere la porta preferita utilizzata per il traffico client con il percorso al controller dei nodi a blocchi che è il principale proprietario di tali volumi di servizi. Per ulteriori informazioni, vedere "Configurazione del software".

Configurazione del software

La configurazione software per BeeGFS su NetApp include componenti di rete BeeGFS, nodi a blocchi EF600, nodi di file BeeGFS, gruppi di risorse e servizi BeeGFS.

Configurazione di rete BeeGFS

La configurazione di rete di BeeGFS è costituita dai seguenti componenti.

- **IP mobili** gli IP mobili sono un tipo di indirizzo IP virtuale che può essere instradato dinamicamente a qualsiasi server della stessa rete. Più server possono avere lo stesso indirizzo IP mobile, ma possono essere attivi solo su un server alla volta.

  Ciascun servizio del server BeeGFS dispone di un proprio indirizzo IP che può spostarsi tra i nodi di file in base alla posizione di esecuzione del servizio del server BeeGFS. Questa configurazione IP mobile consente a ciascun servizio di eseguire il failover in modo indipendente sull'altro nodo di file. Il client deve semplicemente conoscere l'indirizzo IP di un particolare servizio BeeGFS; non deve sapere quale nodo di file sta eseguendo quel servizio.

- **Configurazione multi-homing del server BeeGFS** per aumentare la densità della soluzione, ciascun nodo di file dispone di più interfacce di storage con IP configurati nella stessa subnet IP.

  È necessaria un'ulteriore configurazione per garantire che questa configurazione funzioni come previsto con lo stack di rete Linux, perché per impostazione predefinita, le richieste a un'interfaccia possono essere risposte su un'interfaccia diversa se i relativi IP si trovano nella stessa subnet. Oltre ad altri inconvenienti, questo comportamento predefinito rende impossibile stabilire o mantenere correttamente le connessioni RDMA.

  L'implementazione basata su Ansible gestisce il rafforzamento del comportamento del percorso inverso (RP) e del protocollo di risoluzione degli indirizzi (ARP), oltre a garantire che quando gli IP mobili vengono avviati e arrestati; i corrispondenti percorsi e le regole IP vengono creati dinamicamente per consentire alla configurazione di rete multihomed di funzionare correttamente.

- **La configurazione multi-rail del client BeeGFS** *Multi-rail* si riferisce alla capacità di un'applicazione di

utilizzare più "guide" di rete indipendenti per aumentare le performance.

Sebbene BeeGFS possa utilizzare RDMA per la connettività, BeeGFS utilizza IPoIB per semplificare il rilevamento e la creazione di connessioni RDMA. Per consentire ai client BeeGFS di utilizzare più interfacce InfiniBand, è possibile configurare ciascun client con un indirizzo IP situato in una subnet diversa e quindi configurare le interfacce preferite per metà dei servizi del server BeeGFS in ciascuna subnet.

Nel diagramma seguente, le interfacce evidenziate in verde chiaro si trovano in una subnet IP (ad esempio, `100.127.0.0/16`) e le interfacce di colore verde scuro si trovano in un'altra subnet (ad esempio, `100.128.0.0/16`).

La figura seguente mostra il bilanciamento del traffico tra più interfacce client BeeGFS.

[]

Poiché ogni file in BeeGFS viene in genere sottoposto a striping su più servizi di storage, la configurazione multi-rail consente al client di ottenere un throughput superiore a quello possibile con una singola porta InfiniBand. Ad esempio, il seguente esempio di codice mostra una configurazione di striping comune dei file che consente al client di bilanciare il traffico tra entrambe le interfacce:

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

L'utilizzo di due subnet IPoIB è una distinzione logica. Se lo si desidera, è possibile utilizzare una singola subnet fisica InfiniBand (rete di storage).

> Il supporto multi-rail è stato aggiunto in BeeGFS 7.3.0 per consentire l'utilizzo di più interfacce IB in una singola subnet IPoIB. La progettazione della soluzione BeeGFS su NetApp è stata sviluppata prima della disponibilità generale di BeeGFS 7.3.0, dimostrando così l'utilizzo di due subnet IP per utilizzare due interfacce IB sui client BeeGFS. Uno dei vantaggi dell'approccio a più subnet IP consiste nell'eliminare la necessità di configurare il multihoming sui nodi client BeeGFS (per ulteriori informazioni, vedere "Supporto di BeeGFS RDMA").

Configurazione del nodo a blocchi EF600

I nodi a blocchi sono costituiti da due controller RAID attivi/attivi con accesso condiviso allo stesso set di dischi. In genere, ciascun controller possiede la metà dei volumi configurati sul sistema, ma può sostituire l'altro controller in base alle necessità.

Il software multipathing sui nodi di file determina il percorso attivo e ottimizzato per ciascun volume e si sposta automaticamente sul percorso alternativo in caso di guasto di un cavo, di un adattatore o di un controller.

Il seguente diagramma mostra il layout del controller nei nodi a blocchi EF600.

[]

Per facilitare la soluzione ha con disco condiviso, i volumi vengono mappati su entrambi i nodi di file in modo che possano assumere il controllo reciproco in base alle necessità. Il seguente diagramma mostra un esempio di configurazione del servizio BeeGFS e della proprietà del volume preferita per ottenere le massime prestazioni. L'interfaccia a sinistra di ciascun servizio BeeGFS indica l'interfaccia preferita che i client e gli altri servizi utilizzano per contattarlo.

[]

Nell'esempio precedente, client e servizi server preferiscono comunicare con il servizio di storage 1 utilizzando l'interfaccia i1b. Il servizio di storage 1 utilizza l'interfaccia i1a come percorso preferito per comunicare con i suoi volumi (storage_tgt_101, 102) sul controller A del primo nodo a blocchi. Questa configurazione utilizza l'intera larghezza di banda PCIe bidirezionale disponibile per l'adattatore InfiniBand e ottiene prestazioni migliori da un adattatore HDR InfiniBand a due porte rispetto a quanto sarebbe altrimenti possibile con PCIe 4.0.

Configurazione del nodo del file BeeGFS

I nodi di file BeeGFS sono configurati in un cluster ad alta disponibilità (ha) per facilitare il failover dei servizi BeeGFS tra più nodi di file.

La progettazione del cluster ha si basa su due progetti Linux ha ampiamente utilizzati: Corosync per l'appartenenza al cluster e Pacemaker per la gestione delle risorse del cluster. Per ulteriori informazioni, vedere "Training Red Hat per add-on ad alta disponibilità".

NetApp ha creato ed esteso diversi agenti di risorse OCF (Open Cluster Framework) per consentire al cluster di avviare e monitorare in modo intelligente le risorse BeeGFS.

Cluster BeeGFS ha

In genere, quando si avvia un servizio BeeGFS (con o senza ha), è necessario disporre di alcune risorse:

- Indirizzi IP raggiungibili dal servizio, generalmente configurati da Network Manager.
- File system sottostanti utilizzati come destinazione per BeeGFS per l'archiviazione dei dati.

  Questi sono in genere definiti in `/etc/fstab` E montato da Systemd.

- Un servizio Systemd responsabile dell'avvio dei processi BeeGFS quando le altre risorse sono pronte.

  Senza software aggiuntivo, queste risorse vengono avviate solo su un singolo nodo di file. Pertanto, se il nodo del file non è in linea, una parte del file system BeeGFS non è accessibile.

Dato che più nodi possono avviare ciascun servizio BeeGFS, Pacemaker deve assicurarsi che ogni servizio e le risorse dipendenti siano in esecuzione su un solo nodo alla volta. Ad esempio, se due nodi tentano di avviare lo stesso servizio BeeGFS, esiste il rischio di danneggiamento dei dati se entrambi tentano di scrivere negli stessi file sulla destinazione sottostante. Per evitare questo scenario, Pacemaker si affida a Corosync per mantenere in modo affidabile lo stato del cluster complessivo in sincronia tra tutti i nodi e stabilire il quorum.

Se si verifica un errore nel cluster, Pacemaker reagisce e riavvia le risorse BeeGFS su un altro nodo. In alcuni scenari, Pacemaker potrebbe non essere in grado di comunicare con il nodo guasto originale per confermare che le risorse sono state interrotte. Per verificare che il nodo sia inattivo prima di riavviare le risorse BeeGFS altrove, Pacemaker interrompe il nodo difettoso, idealmente rimuovendo l'alimentazione.

Sono disponibili molti agenti di scherma open-source che consentono a Pacemaker di recinzione di un nodo con un'unità di distribuzione dell'alimentazione (PDU) o utilizzando il server Baseboard Management Controller (BMC) con API come Redfish.

Quando BeeGFS viene eseguito in un cluster ha, tutti i servizi BeeGFS e le risorse sottostanti vengono gestiti da Pacemaker in gruppi di risorse. Ogni servizio BeeGFS e le risorse da cui dipende sono configurati in un gruppo di risorse, che garantisce che le risorse vengano avviate e interrotte nell'ordine corretto e collocate sullo stesso nodo.

Per ciascun gruppo di risorse BeeGFS, Pacemaker esegue una risorsa di monitoraggio BeeGFS personalizzata responsabile del rilevamento delle condizioni di guasto e dell'attivazione intelligente dei failover quando un servizio BeeGFS non è più accessibile su un nodo specifico.

La figura seguente mostra i servizi e le dipendenze BeeGFS controllati da pacemaker.

[]

Per avviare più servizi BeeGFS dello stesso tipo sullo stesso nodo, Pacemaker è configurato per avviare i servizi BeeGFS utilizzando il metodo di configurazione Multi Mode. Per ulteriori informazioni, consultare "Documentazione BeeGFS su Multi Mode".

Poiché i servizi BeeGFS devono essere in grado di avviarsi su più nodi, il file di configurazione per ciascun servizio (normalmente situato in `/etc/beegfs`) Viene memorizzato in uno dei volumi e-Series utilizzati come destinazione BeeGFS per quel servizio. In questo modo, la configurazione e i dati di un particolare servizio BeeGFS sono accessibili a tutti i nodi che potrebbero aver bisogno di eseguire il servizio.

```
# 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
```

Verifica del progetto

La progettazione di seconda generazione per la soluzione BeeGFS su NetApp è stata verificata utilizzando tre profili di configurazione a blocchi.

I profili di configurazione includono quanto segue:

- Un singolo building block di base, che include gestione BeeGFS, metadati e servizi di storage.
- Metadati BeeGFS più un building block per lo storage.
- Un building block BeeGFS solo per lo storage.

I blocchi costitutivi erano collegati a due switch Mellanox Quantum InfiniBand (MQM8700). Dieci client BeeGFS sono stati collegati anche agli switch InfiniBand e utilizzati per eseguire utility di benchmark sintetiche.

La figura seguente mostra la configurazione BeeGFS utilizzata per validare la soluzione BeeGFS su NetApp.

[]

Striping del file BeeGFS

Un vantaggio dei file system paralleli è la capacità di eseguire lo striping di singoli file su più destinazioni di storage, che potrebbero rappresentare volumi sullo stesso sistema di storage sottostante o su sistemi di storage diversi.

In BeeGFS, è possibile configurare lo striping per directory e per file per controllare il numero di destinazioni utilizzate per ogni file e per controllare la dimensione del blocco (o dimensione del blocco) utilizzata per ogni stripe di file. Questa configurazione consente al file system di supportare diversi tipi di carichi di lavoro e profili i/o senza dover riconfigurare o riavviare i servizi. È possibile applicare le impostazioni di stripe utilizzando `beegfs-ctl` Tool della riga di comando o con applicazioni che utilizzano l'API di striping. Per ulteriori informazioni, consultare la documentazione di BeeGFS per "Striping" e. "API di striping".

Per ottenere le migliori prestazioni, i modelli di stripe sono stati regolati durante l'intero test e vengono annotati i parametri utilizzati per ciascun test.

Test della larghezza di banda IOR: Client multipli

I test della larghezza di banda IOR hanno utilizzato OpenMPI per eseguire lavori paralleli dello strumento IOR (Synthetic i/o Generator Tool), disponibile presso "HPC GitHub") Su tutti i 10 nodi client a uno o più blocchi di base BeeGFS. Se non diversamente specificato:

- Tutti i test utilizzavano l'i/o diretto con una dimensione di trasferimento di 1 MiB.
- Lo striping dei file BeeGFS è stato impostato su una dimensione di blocco di 1 MB e su una destinazione per file.

I seguenti parametri sono stati utilizzati per IOR con il conteggio dei segmenti regolato per mantenere la dimensione del file aggregato a 5TiB per un building block e 40TiB per tre building block.

```
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
```

Un building block di base BeeGFS (gestione, metadati e storage)

La figura seguente mostra i risultati del test IOR con un singolo building block di base BeeGFS (gestione, metadati e storage).

[]

Metadati BeeGFS + building block dello storage

La figura seguente mostra i risultati del test IOR con un singolo metadata BeeGFS + building block dello storage.

[]

Building block BeeGFS solo per lo storage

La figura seguente mostra i risultati del test IOR con un singolo building block BeeGFS solo per lo storage.

[]

Tre blocchi di base BeeGFS

La figura seguente mostra i risultati del test IOR con tre blocchi di base BeeGFS.

[]

Come previsto, la differenza di performance tra il building block di base e i successivi metadati + building block di storage è trascurabile. Il confronto tra i metadati e il building block di storage e un building block di solo storage mostra un leggero aumento delle performance di lettura dovuto ai dischi aggiuntivi utilizzati come target di storage. Tuttavia, non vi è alcuna differenza significativa nelle prestazioni di scrittura. Per ottenere performance più elevate, è possibile aggiungere più elementi di base insieme per scalare le performance in modo lineare.

Test della larghezza di banda IOR: Singolo client

Il test della larghezza di banda IOR ha utilizzato OpenMPI per eseguire più processi IOR utilizzando un singolo server GPU dalle performance elevate per esplorare le performance ottenibili da un singolo client.

Questo test confronta anche il comportamento di rilettura e le performance di BeeGFS quando il client è configurato per utilizzare la paging-cache del kernel Linux (`tuneFileCacheType = native`) rispetto al valore predefinito `buffered` impostazione.

La modalità di caching nativa utilizza la paging-cache del kernel Linux sul client, consentendo alle operazioni di rilettura di provenire dalla memoria locale invece di essere ritrasmesse sulla rete.

Il diagramma seguente mostra i risultati del test IOR con tre blocchi di base BeeGFS e un singolo client.

[]

Lo striping di BeeGFS per questi test è stato impostato su una dimensione del blocco di 1 MB con otto destinazioni per file.

Sebbene le performance di scrittura e lettura iniziale siano più elevate utilizzando la modalità buffer predefinita, per i carichi di lavoro che rileggono gli stessi dati più volte, la modalità caching nativa offre un significativo miglioramento delle performance. Questo miglioramento delle performance di rilettura è importante per i carichi di lavoro come il deep learning che rileggono lo stesso set di dati più volte in diverse epoche.

Test delle performance dei metadati

I test delle prestazioni dei metadati hanno utilizzato lo strumento MDTest (incluso come parte di IOR) per misurare le prestazioni dei metadati di BeeGFS. I test hanno utilizzato OpenMPI per eseguire lavori paralleli su tutti e dieci i nodi client.

I seguenti parametri sono stati utilizzati per eseguire il test di benchmark con il numero totale di processi scalati da 10 a 320 in incrementi di 2x e con una dimensione del file di 4k.

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

Le performance dei metadati sono state misurate prima con uno e due metadati + building block di storage per mostrare come le performance si ridimensionano aggiungendo ulteriori building block.

Un solo metadata BeeGFS + building block di storage

Il seguente diagramma mostra i risultati di MDTest con un solo metadata BeeGFS + blocchi di base dello storage.

[]

Due metadati BeeGFS + blocchi di base per lo storage

Il seguente diagramma mostra i risultati di MDTest con due metadati BeeGFS + blocchi di base dello storage.

[]

Validazione funzionale

Nell'ambito della convalida di questa architettura, NetApp ha eseguito diversi test funzionali, tra cui:

- Errore di una singola porta InfiniBand client disattivando la porta dello switch.
- Errore di una porta InfiniBand di un singolo server disattivando la porta dello switch.
- Attivazione dello spegnimento immediato del server mediante BMC.
- Posizionamento corretto di un nodo in standby e failover del servizio su un altro nodo.
- Posizionamento corretto di un nodo di nuovo online e fallimento dei servizi di back nel nodo originale.
- Spegnere uno degli switch InfiniBand utilizzando la PDU. Tutti i test sono stati eseguiti mentre era in corso il test di stress con `sysSessionChecksEnabled: false` Set di parametri sui client BeeGFS. Non sono stati osservati errori o interruzioni dell'i/O.

> Si è verificato un problema noto (vedere "Changelog") Quando le connessioni RDMA client/server BeeGFS vengono interrompute inaspettatamente, a causa della perdita dell'interfaccia primaria (come definito nella `connInterfacesFile`) O un server BeeGFS non funzionante; l'i/o client attivo può bloccarsi per un massimo di dieci minuti prima della ripresa. Questo problema non si verifica quando i nodi BeeGFS vengono posizionati correttamente in standby o non sono in standby per la manutenzione pianificata o se TCP è in uso.

Convalida NVIDIA DGX A100 SuperPOD e BasePOD

NetApp ha validato una soluzione di storage per NVDIA DGX A100 SuperPOD utilizzando un file system BeeGFS simile costituito da tre blocchi di base con i metadati e il profilo di configurazione dello storage

applicato. Il lavoro di qualificazione ha comportato il test della soluzione descritta da questo NVA con venti server GPU DGX A100 che eseguono una varietà di storage, machine learning e benchmark di deep learning. Tutto lo storage certificato per l'utilizzo in NVIDIA DGX A100 SuperPOD è certificato automaticamente per l'utilizzo anche nelle architetture NVIDIA BasePOD.

Per ulteriori informazioni, vedere "NVIDIA DGX SuperPOD con NetApp" e. "NVIDIA DGX BasePOD".

Linee guida per il dimensionamento

La soluzione BeeGFS include consigli per il dimensionamento delle performance e della capacità basati su test di verifica.

L'obiettivo di un'architettura building-block è creare una soluzione semplice da dimensionare aggiungendo più building block per soddisfare i requisiti di un particolare sistema BeeGFS. Utilizzando le linee guida riportate di seguito, è possibile stimare la quantità e i tipi di blocchi di base BeeGFS necessari per soddisfare i requisiti del proprio ambiente.

Tenere presente che queste stime sono le migliori performance del caso. Le applicazioni di benchmarking sintetico vengono scritte e utilizzate per ottimizzare l'utilizzo dei file system sottostanti in modi diversi dalle applicazioni reali.

Dimensionamento delle performance

La seguente tabella fornisce il dimensionamento delle performance consigliato.

Profilo di configurazione	1 MIB letture	1MiB scrive
Metadati + storage	62GiBps	21 GiBps
Solo storage	64 GiBps	21 GiBps

Le stime di dimensionamento della capacità dei metadati si basano sulla "regola generale" secondo cui 500 GB di capacità sono sufficienti per circa 150 milioni di file in BeeGFS. Per ulteriori informazioni, consultare la documentazione di BeeGFS per "Requisiti di sistema".)

L'utilizzo di funzionalità come gli elenchi di controllo degli accessi e il numero di directory e file per directory influisce anche sulla velocità di utilizzo dello spazio dei metadati. Le stime della capacità dello storage tengono conto della capacità utilizzabile del disco insieme all'overhead di RAID 6 e XFS.

Dimensionamento della capacità per metadati + building block dello storage

La seguente tabella fornisce il dimensionamento della capacità consigliato per i metadati e gli elementi di base dello storage.

Dimensioni del disco (2+2 RAID 1) gruppi di volumi di metadati	Capacità dei metadati (numero di file)	Dimensioni del disco (8+2 RAID 6) gruppi di volumi di storage	Capacità dello storage (contenuto del file)
1,92 TB	1,938,577,200	1,92 TB	51,77 TB
3,84 TB	3,880,388,400	3,84 TB	103,55 TB
7,68 TB	8,125,278,000	7,68 TB	216,74 TB
15,3 TB	17,269,854,000	15,3 TB	460,60 TB

Quando si ridimensionano i metadati e gli elementi di base dello storage, è possibile ridurre i costi utilizzando unità più piccole per i gruppi di volumi di metadati rispetto ai gruppi di volumi di storage.

Dimensionamento della capacità per gli elementi di base solo per lo storage

La seguente tabella fornisce il dimensionamento della capacità a regola d'uso per gli elementi di base solo per lo storage.

Dimensioni del disco (10+2 RAID 6) gruppi di volumi di storage	Capacità dello storage (contenuto del file)
1,92 TB	59,89 TB
3,84 TB	119,80 TB
7,68 TB	251,89 TB
15,3 TB	538,55 TB

L'overhead di performance e capacità dell'inclusione del servizio di gestione nel building block di base (primo) è minimo, a meno che non sia attivato il blocco globale dei file.

Tuning delle performance

La soluzione BeeGFS include consigli per l'ottimizzazione delle performance basati su test di verifica.

Sebbene BeeGFS fornisca performance ragionevoli, NetApp ha sviluppato una serie di parametri di tuning consigliati per massimizzare le performance. Questi parametri tengono conto delle funzionalità dei nodi a blocchi e-Series sottostanti e di eventuali requisiti speciali necessari per eseguire BeeGFS in un'architettura ha a disco condiviso.

Ottimizzazione delle performance per i file node

I parametri di tuning disponibili che è possibile configurare includono:

1. **Impostazioni di sistema in UEFI/BIOS dei nodi di file.** per massimizzare le prestazioni, si consiglia di configurare le impostazioni di sistema sul modello di server utilizzato come nodi di file. È possibile configurare le impostazioni di sistema quando si impostano i nodi di file utilizzando il programma di configurazione del sistema (UEFI/BIOS) o le API Redfish fornite dal controller di gestione della baseboard (BMC).

   Le impostazioni di sistema variano a seconda del modello di server utilizzato come nodo di file. Le impostazioni devono essere configurate manualmente in base al modello di server in uso. Per informazioni su come configurare le impostazioni di sistema per i nodi di file Lenovo SR665 validati, vedere "Ottimizzare le impostazioni del sistema del nodo di file per le performance".

2. **Impostazioni predefinite per i parametri di configurazione richiesti.** i parametri di configurazione richiesti influiscono sulla modalità di configurazione dei servizi BeeGFS e sulla modalità di formattazione e montaggio dei volumi e-Series (dispositivi a blocchi) da parte di Pacemaker. Questi parametri di configurazione richiesti includono:
   - Parametri di configurazione del servizio BeeGFS

È possibile ignorare le impostazioni predefinite per i parametri di configurazione in base alle esigenze. Per i parametri che è possibile regolare per i carichi di lavoro o i casi di utilizzo specifici, vedere "Parametri di configurazione del servizio BeeGFS".

- I parametri di montaggio e formattazione del volume sono impostati sui valori predefiniti consigliati e devono essere regolati solo in caso di utilizzo avanzato. I valori predefiniti sono i seguenti:
  - Ottimizza la formattazione iniziale del volume in base al tipo di destinazione (ad esempio gestione, metadati o storage), oltre alla configurazione RAID e alle dimensioni dei segmenti del volume sottostante.
  - Regolare il modo in cui pacemaker monta ciascun volume per assicurarsi che le modifiche vengano immediatamente applicate ai nodi a blocchi e-series. In questo modo si evita la perdita di dati quando i nodi di file non funzionano con scritture attive in corso.

    Per i parametri che è possibile regolare per i carichi di lavoro o i casi di utilizzo specifici, vedere "formattazione dei volumi e parametri di configurazione del montaggio".

3. **Impostazioni di sistema nel sistema operativo Linux installato sui nodi di file.** è possibile ignorare le impostazioni predefinite del sistema operativo Linux quando si crea l'inventario Ansible nel passaggio 4 di "Creare l'inventario Ansible".

   Le impostazioni predefinite sono state utilizzate per validare la soluzione BeeGFS su NetApp, ma è possibile modificarle per adattarle ai carichi di lavoro o ai casi di utilizzo specifici. Di seguito sono riportati alcuni esempi delle impostazioni di sistema del sistema operativo Linux che è possibile modificare:

   - Code i/o su dispositivi a blocchi e-Series.

     È possibile configurare le code i/o sui dispositivi a blocchi e-Series utilizzati come destinazioni BeeGFS per:

     - Regolare l'algoritmo di scheduling in base al tipo di dispositivo (NVMe, HDD e così via).
     - Aumentare il numero di richieste in sospeso.
     - Regolare le dimensioni della richiesta.
     - Ottimizza il comportamento di Read ahead.
   - Impostazioni della memoria virtuale.

     È possibile regolare le impostazioni della memoria virtuale per ottenere performance di streaming ottimali e costanti.

   - Impostazioni della CPU.

     È possibile regolare il regolatore di frequenza della CPU e altre configurazioni della CPU per ottenere le massime prestazioni.

   - Dimensione richiesta di lettura.

     È possibile aumentare la dimensione massima della richiesta di lettura per gli HCA Mellanox.

Ottimizzazione delle performance per i nodi a blocchi

In base ai profili di configurazione applicati a un particolare building block BeeGFS, i gruppi di volumi configurati sui nodi a blocchi cambiano leggermente. Ad esempio, con un nodo a blocchi EF600 a 24 dischi:

- Per il singolo building block di base, inclusi i servizi di gestione, metadati e storage BeeGFS:
  - 1 gruppo di volumi RAID 10 2+2 per la gestione di BeeGFS e i servizi di metadati
  - 2 gruppi di volumi 8+2 RAID 6 per i servizi di storage BeeGFS
- Per i metadati BeeGFS + building block di storage:
  - 1 gruppo di volumi RAID 10 2+2 per i servizi di metadati BeeGFS
  - 2 gruppi di volumi 8+2 RAID 6 per i servizi di storage BeeGFS
- Solo per lo storage BeeGFS building block:
  - 2 gruppi di volumi RAID 6 da 10+2 per i servizi di storage BeeGFS

> Poiché BeeGFS ha bisogno di una quantità di spazio di storage significativamente inferiore per la gestione e i metadati rispetto allo storage, un'opzione è quella di utilizzare dischi più piccoli per i gruppi di volumi RAID 10. I dischi più piccoli devono essere inseriti negli slot più esterni. Per ulteriori informazioni, consultare "istruzioni per l'implementazione".

Questi sono tutti configurati dall'implementazione basata su Ansible, insieme a diverse altre impostazioni generalmente consigliate per ottimizzare performance/comportamento, tra cui:

- Regolare le dimensioni del blocco della cache globale a 32 KiB e regolare il vampate della cache basato sulla domanda al 80%.
- Disattivazione del bilanciamento del carico automatico (per garantire che le assegnazioni dei volumi dei controller rimvengano come previsto).
- Abilitare il caching in lettura e disabilitare il caching Read-ahead.
- Abilitare il caching in scrittura con mirroring e richiedere il backup della batteria, in modo che le cache persistano in caso di guasto di un controller di nodi a blocchi.
- Specifica dell'ordine in cui i dischi vengono assegnati ai gruppi di volumi, bilanciando i/o tra i canali di disco disponibili.

Building block ad alta capacità

Il design della soluzione BeeGFS standard è costruito tenendo in considerazione i carichi di lavoro dalle performance elevate. I clienti che cercano casi di utilizzo ad alta capacità devono osservare le variazioni nelle caratteristiche di progettazione e performance qui delineate.

Configurazione hardware e software

La configurazione hardware e software per l'building block ad alta capacità è standard, ad eccezione del fatto che i controller EF600 devono essere sostituiti con controller EF300 con un'opzione per il collegamento tra 1 e 7 tray di espansione IOM con 60 unità ciascuno per ciascun array di storage, totale da 2 a 14 vassoi di espansione per building block.

I clienti che implementano una progettazione di building block ad alta capacità probabilmente utilizzeranno solo la configurazione di base basata su building block, costituita da servizi di gestione, metadati e storage BeeGFS per ciascun nodo. Per un'efficienza dei costi, i nodi di storage ad alta capacità devono eseguire il provisioning dei volumi di metadati sui dischi NVMe nell'enclosure di controller EF300 e fornire i volumi di storage ai dischi NL-SAS nei vassoi di espansione.

[]

Linee guida per il dimensionamento

Queste linee guida sul dimensionamento presuppongono che gli building block ad alta capacità siano configurati con un gruppo di volumi 2+2 NVMe SSD per i metadati nell'enclosure EF300 di base e 6 gruppi di volumi 8+2 NL-SAS per tray di espansione IOM per lo storage.

Dimensioni del disco (HDD con capacità)	Capacità per BB (1 vassoio)	Capacità per BB (2 tray)	Capacità per BB (3 tray)	Capacità per BB (4 tray)
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

Implementare la soluzione

Panoramica dell'implementazione

È possibile implementare BeeGFS su NetApp in nodi di file e blocchi validati utilizzando la progettazione di blocchi di costruzione BeeGFS di seconda generazione di NetApp.

Raccolte e ruoli Ansible

La soluzione BeeGFS su NetApp viene implementata utilizzando Ansible, un noto motore DI automazione IT utilizzato per automatizzare le implementazioni delle applicazioni. Ansible utilizza una serie di file denominati collettivamente inventario, che modellano il file system BeeGFS che si desidera implementare.

Ansible consente ad aziende come NetApp di espandere le funzionalità integrate utilizzando le raccolte di Ansible Galaxy (vedere "Raccolta NetApp e-Series BeeGFS"). Le raccolte includono moduli che eseguono alcune funzioni o attività specifiche (come la creazione di un volume e-Series) e includono ruoli che possono chiamare più moduli e altri ruoli. Questo approccio automatizzato riduce il tempo necessario per implementare il file system BeeGFS e il cluster ha sottostante. Inoltre, semplifica l'aggiunta di building block per espandere i file system esistenti.

Per ulteriori informazioni, vedere "Scopri di più sull'inventario Ansible".

> Poiché l'implementazione della soluzione BeeGFS su NetApp richiede numerosi passaggi, NetApp non supporta l'implementazione manuale della soluzione.

Profili di configurazione per gli elementi di base BeeGFS

Le procedure di implementazione coprono i seguenti profili di configurazione:

- Un building block di base che include servizi di gestione, metadati e storage.
- Un secondo building block che include metadati e servizi di storage.
- Un terzo building block che include solo i servizi di storage.

Questi profili mostrano l'intera gamma di profili di configurazione consigliati per gli elementi di base di NetApp BeeGFS. Per ogni implementazione, il numero di metadati e di building block dello storage o building block dei soli servizi di storage può variare nelle procedure, a seconda dei requisiti di capacità e performance.

Panoramica delle fasi di implementazione

L'implementazione prevede le seguenti attività di alto livello:

Implementazione dell'hardware

1. Assemblare fisicamente ciascun blocco edificio.
2. Hardware per rack e cavi. Per le procedure dettagliate, vedere "Implementare l'hardware".

Implementazione del software

1. "Impostare i nodi di file e blocchi".
   - Configurare gli IP BMC sui nodi di file
   - Installare un sistema operativo supportato e configurare la rete di gestione sui nodi di file
   - Configurare gli IP di gestione sui nodi a blocchi
2. "Impostare un nodo di controllo Ansible".
3. "Ottimizzare le impostazioni di sistema per le prestazioni".
4. "Creare l'inventario Ansible".
5. "Definire l'inventario Ansible per gli elementi di base BeeGFS".
6. "Implementare BeeGFS utilizzando Ansible".
7. "Configurare i client BeeGFS".

> Le procedure di implementazione includono diversi esempi in cui il testo deve essere copiato in un file. Prestare particolare attenzione a eventuali commenti inline contrassegnati da " n." o "//" per qualsiasi elemento che debba o possa essere modificato per un'implementazione specifica. Ad esempio:
>
> ```
> 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"
> ```

Architetture derivate con variazioni nelle raccomandazioni di implementazione:

- "Building Block ad alta capacità"

Scopri di più sull'inventario Ansible

Prima di iniziare l'implementazione, assicurati di comprendere come utilizzare Ansible per configurare e implementare la soluzione BeeGFS su NetApp utilizzando la progettazione di blocchi di costruzione BeeGFS di seconda generazione.

L'inventario Ansible definisce la configurazione per i nodi di file e blocchi e rappresenta il file system BeeGFS che si desidera implementare. L'inventario include host, gruppi e variabili che descrivono il file system BeeGFS desiderato. Gli inventari di esempio possono essere scaricati da "NetApp e-Series BeeGFS GitHub".

Moduli e ruoli Ansible

Per applicare la configurazione descritta dall'inventario Ansible, utilizzare i vari moduli e ruoli Ansible forniti nella raccolta NetApp e-Series Ansible, in particolare il ruolo BeeGFS ha 7.2 (disponibile presso "NetApp e-Series BeeGFS GitHub") che implementa la soluzione end-to-end.

Ogni ruolo nella raccolta NetApp e-Series Ansible è un'implementazione end-to-end completa della soluzione BeeGFS su NetApp. I ruoli utilizzano le raccolte NetApp e-Series SANtricity, host e BeeGFS che consentono di configurare il file system BeeGFS con ha (alta disponibilità). È quindi possibile eseguire il provisioning e il mapping dello storage, assicurandosi che lo storage del cluster sia pronto per l'uso.

Sebbene i ruoli siano corredati da una documentazione approfondita, le procedure di implementazione descrivono come utilizzare il ruolo per implementare un'architettura verificata di NetApp utilizzando la progettazione di blocchi di costruzione BeeGFS di seconda generazione.

> Anche se le fasi di implementazione tentano di fornire dettagli sufficienti per evitare che l'esperienza precedente con Ansible sia un prerequisito, si dovrebbe avere una certa familiarità con Ansible e la terminologia correlata.

Layout dell'inventario per un cluster BeeGFS ha

Utilizzare la struttura di inventario Ansible per definire un cluster BeeGFS ha.

Chiunque abbia esperienza precedente con Ansible deve essere consapevole del fatto che il ruolo BeeGFS ha implementa un metodo personalizzato per scoprire quali variabili (o fatti) si applicano a ciascun host. Ciò è necessario per semplificare la creazione di un inventario Ansible che descriva le risorse che possono essere eseguite su più server.

Un inventario Ansible è costituito in genere dai file in `host_vars` e. `group_vars`e un `inventory.yml` file che assegna host a gruppi specifici (e potenzialmente gruppi ad altri gruppi).

> Non creare alcun file con il contenuto di questa sottosezione, che è da intendersi solo come esempio.

Sebbene questa configurazione sia predeterminata in base al profilo di configurazione, è necessario avere una comprensione generale del modo in cui tutto viene presentato come inventario Ansible, come segue:

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

Per ogni servizio, viene creato un file aggiuntivo in `group_vars` descrizione della configurazione:

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

Questo layout consente di definire la configurazione del servizio, della rete e dello storage BeeGFS per ciascuna risorsa in un'unica posizione. Dietro le quinte, il ruolo BeeGFS aggrega la configurazione necessaria per ogni nodo di file e blocchi in base a questa struttura di inventario. Per ulteriori informazioni, consulta questo post del blog: "NetApp accelera l'implementazione di ha per BeeGFS con Ansible".

> L'ID del nodo BeeGFS numerico e stringa per ciascun servizio viene configurato automaticamente in base al nome del gruppo. Pertanto, oltre al requisito generale Ansible per l'univoci nome di gruppo, i gruppi che rappresentano un servizio BeeGFS devono terminare con un numero univoco per il tipo di servizio BeeGFS rappresentato dal gruppo. Ad esempio, sono consentiti meta_01 e stor_01, ma i metadati_01 e meta_01 non lo sono.

Esaminare le Best practice

Seguire le linee guida delle Best practice per l'implementazione della soluzione BeeGFS su NetApp.

Convenzioni standard

Quando si assembla e crea fisicamente il file di inventario Ansible, attenersi alle seguenti convenzioni standard (per ulteriori informazioni, vedere "Creare l'inventario Ansible").

- I nomi host dei nodi di file sono numerati in sequenza (h01-HN) con numeri inferiori nella parte superiore del rack e numeri superiori nella parte inferiore.

  Ad esempio, la convenzione di naming `[location][row][rack]hN` assomiglia a: `ictad22h01`.

- Ciascun nodo a blocchi è composto da due controller di storage, ciascuno con il proprio nome host.

  Il nome di un array di storage viene utilizzato per fare riferimento all'intero sistema di storage a blocchi come parte di un inventario Ansible. I nomi degli array di storage devono essere numerati in sequenza (a01 - AN) e i nomi host dei singoli controller derivano da tale convenzione di naming.

  Ad esempio, un nodo del blocco denominato `ictad22a01` in genere, è possibile configurare i nomi host per ciascun controller come `ictad22a01-a` e. `ictad22a01-b`, Ma in un inventario Ansible deve essere indicato come `ictad22a01`.

- I nodi di file e blocchi all'interno dello stesso building block condividono lo stesso schema di numerazione e sono adiacenti l'uno all'altro nel rack con entrambi i nodi di file in cima ed entrambi i nodi di blocco direttamente sotto di essi.

  Ad esempio, nel primo building block, i nodi di file h01 e h02 sono entrambi collegati direttamente ai nodi di blocco a01 e a02. Dall'alto verso il basso, i nomi host sono h01, h02, a01 e a02.

- I building block vengono installati in ordine sequenziale in base ai nomi host, in modo che i nomi host con numero inferiore si trovino nella parte superiore del rack e i nomi host con numero superiore nella parte inferiore.

  L'obiettivo è ridurre al minimo la lunghezza del cavo che va verso la parte superiore degli switch rack e definire una pratica di implementazione standard per semplificare la risoluzione dei problemi. Per i datacenter in cui ciò non è consentito a causa di problemi relativi alla stabilità del rack, è certamente consentito l'inverso, popolando il rack dal basso verso l'alto.

Configurazione della rete storage InfiniBand

Metà delle porte InfiniBand su ciascun nodo di file vengono utilizzate per connettersi direttamente ai nodi di blocco. L'altra metà è collegata agli switch InfiniBand e viene utilizzata per la connettività client-server BeeGFS. Quando si determinano le dimensioni delle subnet IPoIB utilizzate per client e server BeeGFS, è necessario considerare la crescita prevista del cluster di calcolo/GPU e del file system BeeGFS. Se si deve discostarsi dagli intervalli IP consigliati, tenere presente che ogni connessione diretta in un singolo building block ha una subnet univoca e non esiste alcuna sovrapposizione con le subnet utilizzate per la connettività client-server.

Connessioni dirette

I nodi di file e blocchi all'interno di ciascun building block utilizzano sempre gli IP nella tabella seguente per le loro connessioni dirette.

> Questo schema di indirizzamento rispetta la seguente regola: Il terzo ottetto è sempre dispari o pari, a seconda che il nodo del file sia dispari o pari.

Nodo del file	Porta IB	Indirizzo IP	Nodo del blocco	Porta IB	IP fisico	IP virtuale
Dispari (h1)	i1a	192.168.1.10	Dispari (c1)	2a	192.168.1.100	192.168.1.101
Dispari (h1)	i2a	192.168.3.10	Dispari (c1)	2a	192.168.3.100	192.168.3.101
Dispari (h1)	i3a	192.168.5.10	Pari (c2)	2a	192.168.5.100	192.168.5.101
Dispari (h1)	i4a	192.168.7.10	Pari (c2)	2a	192.168.7.100	192.168.7.101

Nodo del file	Porta IB	Indirizzo IP	Nodo del blocco	Porta IB	IP fisico	IP virtuale
Pari (h2)	i1a	192.168.2.10	Dispari (c1)	2b	192.168.2.100	192.168.2.101
Pari (h2)	i2a	192.168.4.10	Dispari (c1)	2b	192.168.4.100	192.168.4.101
Pari (h2)	i3a	192.168.6.10	Pari (c2)	2b	192.168.6.100	192.168.6.101
Pari (h2)	i4a	192.168.8.10	Pari (c2)	2b	192.168.8.100	192.168.8.101

Schema di indirizzamento IPoIB client-server BeeGFS (due subnet)

Per consentire ai client BeeGFS di utilizzare due porte InfiniBand, sono necessarie due subnet IPoIB con la metà dei servizi server BeeGFS configurati con un IP preferito su ciascuna subnet, in modo da garantire che i client utilizzino due porte InfiniBand per massimizzare la ridondanza e il possibile throughput per il file system.

Ogni nodo di file esegue più servizi server BeeGFS (gestione, metadati o storage). Per consentire a ciascun servizio di eseguire il failover in modo indipendente sull'altro nodo di file, ciascuno è configurato con indirizzi IP univoci che possono fluttuare tra entrambi i nodi (a volte definiti interfaccia logica o LIF).

Sebbene non sia obbligatorio, questa implementazione presuppone che i seguenti intervalli di subnet IPoIB siano in uso per queste connessioni e definisce uno schema di indirizzamento standard che applica le seguenti regole:

- Il secondo ottetto è sempre dispari o pari, a seconda che la porta InfiniBand del nodo del file sia pari o dispari.
- Gli IP del cluster BeeGFS sono sempre `xxx. 127.100.yyy` oppure `xxx.128.100.yyy`.

> Oltre all'interfaccia utilizzata per la gestione del sistema operativo in-band, Corosync può utilizzare interfacce aggiuntive per la sincronizzazione e il battito cardiaco del cluster. In questo modo, la perdita di una singola interfaccia non riduce l'intero cluster.

- Il servizio BeeGFS Management è sempre attivo `xxx.yyy.101.0` oppure `xxx.yyy.102.0`.
- I servizi di metadati BeeGFS sono sempre attivi `xxx.yyy.101.zzz` oppure `xxx.yyy.102.zzz`.
- I servizi di storage BeeGFS sono sempre attivi `xxx.yyy.103.zzz` oppure `xxx.yyy.103.zzz`.
- Indirizzi compresi nell'intervallo `100.xxx.1.1` attraverso `100.xxx.99.255` sono riservati ai clienti.

Subnet A: 100.127.0.0/16

La seguente tabella fornisce l'intervallo per la subnet A: 100.127.0.0/16.

Scopo	Porta InfiniBand	Indirizzo IP o intervallo
IP cluster BeeGFS	i1b	100.127.100.1 - 100.127.100.255
Gestione di BeeGFS	i1b	100.127.101.0
Metadati BeeGFS	i1b o i3b	100.127.101.1 - 100.127.101.255
Storage BeeGFS	i1b o i3b	100.127.103.1 - 100.127.103.255
Client BeeGFS	(varia in base al client)	100.127.1.1 - 100.127.99.255

Subnet B: 100.128.0.0/16

La seguente tabella fornisce l'intervallo per la subnet B: 100.128.0.0/16.

Scopo	Porta InfiniBand	Indirizzo IP o intervallo
IP cluster BeeGFS	i4b	100.128.100.1 - 100.128.100.255
Gestione di BeeGFS	i2b	100.128.102.0
Metadati BeeGFS	i2b o i4b	100.128.102.1 - 100.128.102.255
Storage BeeGFS	i2b o i4b	100.128.104.1 - 100.128.104.255
Client BeeGFS	(varia in base al client)	100.128.1.1 - 100.128.99.255

Non tutti gli IP compresi negli intervalli sopra indicati vengono utilizzati in questa architettura verificata di NetApp. Dimostrano come gli indirizzi IP possono essere pre-allocati per consentire una facile espansione del file system utilizzando uno schema di indirizzamento IP coerente. In questo schema, i nodi di file BeeGFS e gli ID di servizio corrispondono al quarto ottetto di un intervallo ben noto di IP. Il file system potrebbe certamente scalare oltre 255 nodi o servizi, se necessario.

Implementare l'hardware

Ciascun building block è costituito da due nodi di file x86 validati collegati direttamente a due nodi a blocchi utilizzando cavi HDR (200 GB) InfiniBand.

Poiché ogni building block include due nodi di file BeeGFS, per stabilire il quorum nel cluster di failover sono necessari almeno due building block. Sebbene sia possibile configurare un cluster a due nodi, esistono limitazioni a questa configurazione che possono impedire il failover in alcuni scenari. Se è necessario un cluster a due nodi, è anche possibile incorporare un terzo dispositivo come spareggio, sebbene ciò non sia trattato in questa procedura di implementazione.

Se non diversamente specificato, i seguenti passaggi sono identici per ciascun building block del cluster, indipendentemente dal fatto che venga utilizzato per eseguire metadati e servizi di storage BeeGFS o solo servizi di storage.

Fasi

1. Configurare ciascun nodo di file BeeGFS con quattro HCA (host Channel Adapter) a doppia porta PCIe 4.0 ConnectX-6 in modalità InfiniBand e installarli negli slot PCIe 2, 3, 5 e 6.
2. Configurare ciascun nodo a blocchi BeeGFS con una scheda HIC (host Interface Card) da 200 GB a doppia porta e installare l'HIC in ciascuno dei due controller storage.

   Rack dei blocchi in modo che i due nodi di file BeeGFS si trovino sopra i nodi di blocco BeeGFS. La figura seguente mostra la configurazione hardware corretta per l'building block BeeGFS (vista posteriore).

   []

   La configurazione dell'alimentatore per i casi di utilizzo in produzione dovrebbe in genere utilizzare PSU ridondanti.

3. Se necessario, installare i dischi in ciascuno dei nodi a blocchi BeeGFS.
   a. Se il building block verrà utilizzato per eseguire i metadati e i servizi di storage BeeGFS e le unità più

piccole verranno utilizzate per i volumi di metadati, verificare che siano popolate negli slot più esterni, come mostrato nella figura seguente.

b. Per tutte le configurazioni di building block, se un enclosure di dischi non è completamente popolato, assicurarsi che negli slot 0–11 e 12–23 venga inserito un numero uguale di dischi per ottenere prestazioni ottimali.

[]

4. Per collegare i nodi di file e blocchi, utilizzare cavi di rame a collegamento diretto da 1 m InfiniBand HDR 200 GB, in modo che corrispondano alla topologia illustrata nella figura seguente.

[]

> I nodi di più building block non sono mai connessi direttamente. Ogni building block deve essere trattato come un'unità standalone e tutte le comunicazioni tra building block avvengono tramite switch di rete.

5. Utilizzare cavi di rame a collegamento diretto da 200 GB InfiniBand HDR da 2 m (o della lunghezza appropriata) per collegare le restanti porte InfiniBand su ciascun nodo di file agli switch InfiniBand che verranno utilizzati per la rete di storage.

   Se sono in uso switch InfiniBand ridondanti, collegare le porte evidenziate in verde chiaro nella figura seguente a diversi switch.

   []

6. Se necessario, assemblare gli elementi di base aggiuntivi seguendo le stesse linee guida per il cablaggio.

> Il numero totale di building block implementabili in un singolo rack dipende dall'alimentazione e dal raffreddamento disponibili in ogni sito.

Implementare il software

Impostare nodi di file e nodi di blocco

Sebbene la maggior parte delle attività di configurazione del software sia automatizzata utilizzando le raccolte Ansible fornite da NetApp, è necessario configurare il networking sul BMC (Baseboard Management Controller) di ciascun server e configurare la porta di gestione su ciascun controller.

Configurare i nodi di file

1. Configurare il networking sul BMC (Baseboard Management Controller) di ciascun server.

   Per informazioni su come configurare la rete per i file node Lenovo SR665 validati, consultare la "Documentazione di Lenovo ThinkSystem".

> Un BMC (Baseboard Management Controller), a volte chiamato Service Processor, è il nome generico della funzionalità di gestione out-of-band integrata in varie piattaforme server che possono fornire accesso remoto anche se il sistema operativo non è installato o accessibile. I vendor in genere commercializzano questa funzionalità con un proprio marchio. Ad esempio, su Lenovo SR665, il BMC viene definito *Lenovo XClarity Controller (XCC)*.

2. Configurare le impostazioni di sistema per ottenere le massime prestazioni.

   È possibile configurare le impostazioni di sistema utilizzando il setup UEFI (precedentemente noto come BIOS) o le API Redfish fornite da molti BMC. Le impostazioni di sistema variano in base al modello di server utilizzato come nodo di file.

   Per informazioni su come configurare le impostazioni di sistema per i nodi di file Lenovo SR665 validati, vedere "Ottimizzare le impostazioni di sistema per le prestazioni".

3. Installare Red Hat 8.4 e configurare il nome host e la porta di rete utilizzati per gestire il sistema operativo, inclusa la connettività SSH dal nodo di controllo Ansible.

   Non configurare gli IP su nessuna delle porte InfiniBand in questo momento.

   > Sebbene non sia strettamente necessario, le sezioni successive presumono che i nomi host siano numerati in sequenza (ad esempio h1-HN) e si riferiscono alle attività che devono essere completate su host con numero pari o dispari.

4. Utilizza RedHat Subscription Manager per registrare e sottoscrivere il sistema per consentire l'installazione dei pacchetti richiesti dai repository ufficiali Red Hat e per limitare gli aggiornamenti alla versione supportata di Red Hat: `subscription-manager release --set=8.4`. Per istruzioni, vedere "Come registrarsi e sottoscrivere un sistema RHEL" e. "Come limitare gli aggiornamenti".
5. Abilitare il repository Red Hat contenente i pacchetti richiesti per l'alta disponibilità.

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

6. Aggiornare il firmware di ConnectX-6 HCA alla versione consigliata in "Requisiti tecnologici".

   Questo aggiornamento può essere eseguito scaricando ed eseguendo una versione del tool mlxup che integra il firmware consigliato. È possibile scaricare questo tool da "Mlxup - Utility di aggiornamento e query" ("guida per l'utente").

Impostare i nodi a blocchi

Configurare i nodi a blocchi EF600 configurando la porta di gestione su ciascun controller.

1. Configurare la porta di gestione su ciascun controller EF600.

   Per istruzioni sulla configurazione delle porte, consultare "Centro di documentazione e-Series".

2. Facoltativamente, impostare il nome dell'array di storage per ciascun sistema.

   L'impostazione di un nome può semplificare il riferimento a ciascun sistema nelle sezioni successive. Per

istruzioni sull'impostazione del nome dell'array, consultare "Centro di documentazione e-Series".

> Sebbene non sia strettamente necessario, gli argomenti successivi presumono che i nomi degli array di storage siano numerati in sequenza (ad esempio c1 - CN) e fanno riferimento ai passaggi da completare sui sistemi con numero pari o dispari.

Impostare un nodo di controllo Ansible

Per impostare un nodo di controllo Ansible, è necessario identificare una macchina virtuale o fisica con accesso di rete alle porte di gestione di tutti i nodi di file e blocchi che possono essere utilizzati per configurare la soluzione.

I seguenti passaggi sono stati testati su CentOS 8.4. Per i passaggi specifici della distribuzione Linux preferita, consultare la "Documentazione Ansible".

1. Installare Python 3.9 e assicurarsi che la versione corretta di `pip` è installato.

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

2. Creare collegamenti simbolici, assicurandosi che il binario Python 3.9 venga utilizzato ogni volta `python3` oppure `python` viene chiamato.

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

3. Installare i pacchetti Python richiesti dalle raccolte NetApp BeeGFS.

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

> Per assicurarsi di installare una versione supportata di Ansible e tutti i pacchetti Python richiesti, fare riferimento al file Readme della raccolta BeeGFS. Le versioni supportate sono indicate anche nella "Requisiti tecnici".

4. Verificare che siano installate le versioni corrette di Ansible e Python.

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

5. Memorizzare gli inventari Ansible utilizzati per descrivere l'implementazione di BeeGFS nei sistemi di controllo del codice sorgente come Git o BitBucket, quindi installare Git per interagire con tali sistemi.

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

6. Impostare SSH senza password. Questo è il modo più semplice per consentire ad Ansible di accedere ai nodi di file BeeGFS remoti dal nodo di controllo Ansible.
   a. Nel nodo di controllo Ansible, se necessario, generare una coppia di chiavi pubbliche utilizzando `ssh-keygen`
   b. Impostare SSH senza password su ciascuno dei nodi di file utilizzando `ssh-copy-id <ip_or_hostname>`

      Non impostare * SSH senza password sui nodi di blocco. Questo non è supportato né richiesto.

7. Utilizzare Ansible Galaxy per installare la versione della raccolta BeeGFS elencata nella "Requisiti tecnici".

   Questa installazione include dipendenze Ansible aggiuntive, come il software NetApp SANtricity e le raccolte host.

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

Creare l'inventario Ansible

Per definire la configurazione per i nodi di file e blocchi, creare un inventario Ansible che rappresenti il file system BeeGFS che si desidera implementare. L'inventario include host, gruppi e variabili che descrivono il file system BeeGFS desiderato.

Fase 1: Definire la configurazione per tutti gli elementi di base

Definire la configurazione che si applica a tutti gli elementi di base, indipendentemente dal profilo di configurazione che è possibile applicare singolarmente.

Prima di iniziare

- Utilizzare un sistema di controllo del codice sorgente come BitBucket o Git per memorizzare il contenuto della directory contenente l'inventario Ansible e i file del playbook.
- Creare un `.gitignore` File che specifica i file che Git deve ignorare. In questo modo si evita di memorizzare file di grandi dimensioni in Git.

Fasi

1. Nel nodo di controllo Ansible, identificare una directory che si desidera utilizzare per memorizzare i file dell'inventario Ansible e del playbook.

   Se non diversamente specificato, tutti i file e le directory creati in questa fase e nelle fasi successive vengono creati in relazione a questa directory.

2. Creare le seguenti sottodirectory:

   ```
   host_vars

   group_vars

   packages
   ```

Fase 2: Definire la configurazione per i singoli nodi di file e blocchi

Definire la configurazione che si applica ai singoli nodi di file e ai singoli nodi building block.

1. Sotto `host_vars/`, Creare un file per ogni nodo di file BeeGFS denominato `<HOSTNAME>.yml` Con il seguente contenuto, prestare particolare attenzione alle note relative al contenuto da compilare per gli IP del cluster BeeGFS e i nomi host che terminano con numeri dispari e pari.

   Inizialmente, i nomi dell'interfaccia del nodo del file corrispondono a quelli elencati qui (ad esempio ib0 o ibs1f0). Questi nomi personalizzati sono configurati in Fase 4: Definire la configurazione da applicare a tutti i nodi di file.

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

Se il cluster BeeGFS è già stato implementato, è necessario arrestare il cluster prima di aggiungere o modificare gli indirizzi IP configurati staticamente, inclusi gli IP del cluster e gli IP utilizzati per NVMe/IB. Ciò è necessario per garantire che queste modifiche abbiano effetto corretto e non interrompano le operazioni del cluster.

2. Sotto `host_vars/`, Creare un file per ogni nodo del blocco BeeGFS denominato `<HOSTNAME>.yml` e compilarlo con il seguente contenuto.

   Prestare particolare attenzione alle note relative ai contenuti da inserire nei nomi degli array di storage che terminano con numeri pari o dispari.

   Per ogni nodo del blocco, creare un file e specificare `<MANAGEMENT_IP>` Per uno dei due controller (di solito 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
```

Fase 3: Definire la configurazione da applicare a tutti i nodi di file e blocchi

È possibile definire la configurazione comune a un gruppo di host in `group_vars` in un nome di file che corrisponde al gruppo. In questo modo si evita di ripetere una configurazione condivisa in più posizioni.

A proposito di questa attività

Gli host possono trovarsi in più di un gruppo e, in fase di esecuzione, Ansible sceglie le variabili da applicare a un determinato host in base alle regole di precedenza delle variabili. Per ulteriori informazioni su queste regole, consultare la documentazione Ansible per "Utilizzo delle variabili".)

Le assegnazioni host-to-group sono definite nel file di inventario Ansible effettivo, creato verso la fine di questa procedura.

Fase

In Ansible, qualsiasi configurazione che si desidera applicare a tutti gli host può essere definita in un gruppo chiamato `All`. Creare il file `group_vars/all.yml` con i seguenti contenuti:

```
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"
```

Fase 4: Definire la configurazione da applicare a tutti i nodi di file

La configurazione condivisa per i nodi di file viene definita in un gruppo chiamato `ha_cluster`. La procedura descritta in questa sezione illustra la configurazione da includere in `group_vars/ha_cluster.yml` file.

Fasi

1. Nella parte superiore del file, definire le impostazioni predefinite, inclusa la password da utilizzare come `sudo` utente sui nodi del file.

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

In particolare per gli ambienti di produzione, non memorizzare le password in testo normale. Utilizzare invece il vault Ansible (vedere "Crittografia del contenuto con Ansible Vault") o il `--ask-become-pass` quando si esegue il playbook. Se il `ansible_ssh_user` è già `root`, quindi è possibile omettere il `ansible_become_password`.

2. Facoltativamente, configurare un nome per il cluster ad alta disponibilità (ha) e specificare un utente per la comunicazione intra-cluster.

   Se si sta modificando lo schema di indirizzamento IP privato, è necessario aggiornare anche il valore predefinito `beegfs_ha_mgmtd_floating_ip`. Questo valore deve corrispondere a quello configurato in seguito per il gruppo di risorse BeeGFS Management.

   Specificare una o più e-mail che devono ricevere avvisi per gli eventi del cluster utilizzando `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: hacluster                # BeeGFS HA cluster
username.
beegfs_ha_cluster_password: hapassword               # BeeGFS HA cluster
username's password.
beegfs_ha_cluster_password_sha512_salt: randomSalt # BeeGFS HA cluster
username's password salt.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0           # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
      mydomain: “example.com”
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
beegfs_ha_alert_verbosity: 3
#  1) high-level node activity
#  3) high-level node activity + fencing action information + resources
(filter on X-monitor)
#  5) high-level node activity + fencing action information + resources
```

Anche se apparentemente ridondante, `beegfs_ha_mgmtd_floating_ip` È importante quando si scala il file system BeeGFS oltre un singolo cluster ha. I cluster ha successivi vengono implementati senza un servizio di gestione BeeGFS aggiuntivo e puntano al servizio di gestione fornito dal primo cluster.

3. Configurare un agente di scherma. Per ulteriori informazioni, vedere "Configurare la scherma in un cluster Red Hat High Availability".) Il seguente output mostra esempi per la configurazione degli agenti di scherma comuni. Scegliere una di queste opzioni.

   Per questa fase, tenere presente che:

   - Per impostazione predefinita, la funzione di scherma è attivata, ma è necessario configurare un *Agent*

di scherma.

- Il `<HOSTNAME>` specificato in `pcmk_host_map` oppure `pcmk_host_list` Deve corrispondere al nome host nell'inventario Ansible.
- L'esecuzione del cluster BeeGFS senza scherma non è supportata, in particolare in produzione. In questo modo si garantisce in gran parte che quando i servizi BeeGFS, incluse eventuali dipendenze di risorse come i dispositivi a blocchi, si verifichi un failover a causa di un problema, non vi sia alcun rischio di accesso simultaneo da parte di più nodi che si traducono in un danneggiamento del file system o in altri comportamenti indesiderati o imprevisti. Se la scherma deve essere disattivata, fare riferimento alle note generali nella guida introduttiva e nel set del ruolo BeeGFS ha `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` a false in `ha_cluster.yml`.
- Sono disponibili più dispositivi di scherma a livello di nodo e il ruolo BeeGFS ha può configurare qualsiasi agente di scherma disponibile nel repository dei pacchetti Red Hat ha. Se possibile, utilizzare un agente di scherma che lavori attraverso l'UPS (Uninterruptible Power Supply) o l'unità di distribuzione dell'alimentazione rack (rPDU), Perché alcuni agenti di scherma, come il BMC (Baseboard Management Controller) o altri dispositivi di illuminazione integrati nel server, potrebbero non rispondere alla richiesta di fence in determinati scenari di errore.

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

4. Abilitare l’ottimizzazione delle performance consigliata nel sistema operativo Linux.

   Mentre molti utenti trovano che le impostazioni predefinite per i parametri delle performance funzionino generalmente bene, è possibile modificare le impostazioni predefinite per un particolare carico di lavoro. Di conseguenza, questi consigli sono inclusi nel ruolo BeeGFS, ma non sono abilitati per impostazione predefinita per garantire che gli utenti siano a conoscenza della messa a punto applicata al file system.

   Per attivare l’ottimizzazione delle performance, specificare:

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

5. (Facoltativo) è possibile regolare i parametri di ottimizzazione delle performance nel sistema operativo Linux in base alle esigenze.

Per un elenco completo dei parametri di tuning disponibili che è possibile regolare, vedere la sezione Performance Tuning Defaults del ruolo BeeGFS ha in "Sito e-Series BeeGFS GitHub". I valori predefiniti possono essere sovrascritti per tutti i nodi nel cluster in questo file o in `host_vars` file per un singolo nodo.

6. Per consentire una connettività completa da 200 GB/HDR tra nodi di file e blocchi, utilizzare il pacchetto Open Subnet Manager (opensm) di Mellanox Open Fabrics Enterprise Distribution (MLNX_OFED). (La casella di posta in arrivo `opensm` il pacchetto non supporta le funzionalità di virtualizzazione necessarie). Sebbene sia supportata la distribuzione con Ansible, è necessario prima scaricare i pacchetti desiderati nel nodo di controllo Ansible utilizzato per eseguire il ruolo BeeGFS.
   a. Utilizzo di `curl` In alternativa, scaricare i pacchetti per la versione di opensm elencata nella sezione relativa ai requisiti tecnologici dal sito Web di Mellanox al `packages/` directory. Ad esempio:

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

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

   b. Compilare i seguenti parametri in `group_vars/ha_cluster.yml` (regolare i pacchetti in base alle esigenze):

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

7. Configurare `udev` Regola per garantire la mappatura coerente degli identificatori di porta logici InfiniBand ai dispositivi PCIe sottostanti.

   Il `udev` La regola deve essere univoca per la topologia PCIe di ciascuna piattaforma server utilizzata come nodo di file BeeGFS.

   Utilizzare i seguenti valori per i nodi di file verificati:

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

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

8. (Facoltativo) aggiornare l'algoritmo di selezione dei metadati.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

> Durante i test di verifica, `randomrobin` In genere, è stato utilizzato per garantire che i file di test fossero distribuiti in modo uniforme tra tutti gli obiettivi di storage BeeGFS durante il benchmarking delle performance (per ulteriori informazioni sul benchmarking, visitare il sito BeeGFS per "Benchmarking di un sistema BeeGFS"). Con un utilizzo reale, questo potrebbe causare il riempimento più rapido dei target con un numero inferiore rispetto ai target con un numero superiore. Omettere `randomrobin` e utilizzando solo il valore predefinito `randomized` è stato dimostrato che il valore offre buone performance pur continuando a utilizzare tutti gli obiettivi disponibili.

Fase 5: Definire la configurazione per il nodo a blocchi comune

La configurazione condivisa per i nodi a blocchi viene definita in un gruppo chiamato `eseries_storage_systems`. La procedura descritta in questa sezione illustra la configurazione da includere in `group_vars/ eseries_storage_systems.yml` file.

Fasi

1. Impostare la connessione Ansible su locale, fornire la password di sistema e specificare se i certificati SSL devono essere verificati. (In genere, Ansible utilizza SSH per connettersi agli host gestiti, ma nel caso dei sistemi storage NetApp e-Series utilizzati come nodi a blocchi, i moduli utilizzano l'API REST per la comunicazione). Nella parte superiore del file, aggiungere quanto segue:

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

Si sconsiglia di elencare le password in testo non crittografato. Utilizzare Ansible vault o fornire il `eseries_system_password` Quando si esegue Ansible utilizzando `--extra -vars`.

2. Per garantire prestazioni ottimali, installare le versioni elencate per i nodi a blocchi in "Requisiti tecnici".

   Scaricare i file corrispondenti da "Sito di supporto NetApp". È possibile aggiornarli manualmente o includerli in `packages/` Directory del nodo di controllo Ansible, quindi popolare i seguenti parametri in `eseries_storage_systems.yml` Per eseguire l'aggiornamento utilizzando Ansible:

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

3. Scaricare e installare il firmware più recente disponibile per le unità installate nei nodi a blocchi da "Sito di supporto NetApp". È possibile aggiornarli manualmente o includerli in `packages/` Directory del nodo di controllo Ansible, quindi popolare i seguenti parametri in `eseries_storage_systems.yml` Per eseguire l'aggiornamento utilizzando Ansible:

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

   Impostazione `eseries_drive_firmware_upgrade_drives_online` a. `false` Accelera l'aggiornamento, ma non deve essere eseguito fino a quando non viene implementato BeeGFS. Questo perché questa impostazione richiede l'interruzione di tutti i/o sui dischi prima dell'aggiornamento per evitare errori dell'applicazione. Sebbene l'esecuzione di un aggiornamento online del firmware del disco prima della configurazione dei volumi sia ancora rapida, si consiglia di impostare sempre questo valore su `true` per evitare problemi in un secondo momento.

4. Per ottimizzare le performance, apportare le seguenti modifiche alla configurazione globale:

```
# 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. Per garantire un provisioning e un comportamento ottimali dei volumi, specificare i seguenti parametri:

```
# 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"
```

Il valore specificato per `eseries_storage_pool_usable_drives` È specifico per i nodi a blocchi NetApp EF600 e controlla l'ordine in cui i dischi vengono assegnati a nuovi gruppi di volumi. Questo ordine garantisce che l'i/o per ciascun gruppo sia distribuito uniformemente tra i canali di dischi back-end.

Definire l'inventario Ansible per gli elementi di base BeeGFS

Dopo aver definito la struttura generale di inventario Ansible, definire la configurazione per ciascun building block nel file system BeeGFS.

Queste istruzioni di implementazione mostrano come implementare un file system costituito da un building block di base che include servizi di gestione, metadati e storage, un secondo building block con metadati e servizi di storage e un terzo building block di solo storage.

Questi passaggi hanno lo scopo di mostrare l'intera gamma di profili di configurazione tipici che è possibile utilizzare per configurare gli elementi di base di NetApp BeeGFS in modo da soddisfare i requisiti del file system generale BeeGFS.

In questa e nelle sezioni successive, modificare in base alle necessità per creare l'inventario che rappresenta il file system BeeGFS che si desidera implementare. In particolare, utilizzare i nomi host Ansible che rappresentano ciascun nodo di file o blocco e lo schema di indirizzamento IP desiderato per la rete di storage per garantire che possa scalare in base al numero di nodi di file e client BeeGFS.

Fase 1: Creare il file di inventario Ansible

Fasi

1. Creare un nuovo `inventory.yml` quindi inserire i seguenti parametri, sostituendo gli host in `eseries_storage_systems` in base alle necessità per rappresentare i nodi a blocchi nell'implementazione. I nomi devono corrispondere al nome utilizzato per `host_vars/<FILENAME>.yml`.

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

   Nelle sezioni successive, verranno creati ulteriori gruppi Ansible in `ha_cluster` Che rappresentano i servizi BeeGFS che si desidera eseguire nel cluster.

Fase 2: Configurare l'inventario per un building block di gestione, metadati e storage

Il primo building block nel cluster o nel building block di base deve includere il servizio di gestione BeeGFS insieme ai metadati e ai servizi di storage:

Fasi

1. Poll `inventory.yml`, compilare i seguenti parametri in `ha_cluster: children`:

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

```
        ictad22h01:
        ictad22h02:
    meta_02:
      hosts:
        ictad22h01:
        ictad22h02:
    stor_02:
      hosts:
        ictad22h01:
        ictad22h02:
    meta_03:
      hosts:
        ictad22h01:
        ictad22h02:
    stor_03:
      hosts:
        ictad22h01:
        ictad22h02:
    meta_04:
      hosts:
        ictad22h01:
        ictad22h02:
    stor_04:
      hosts:
        ictad22h01:
        ictad22h02:
    meta_05:
      hosts:
        ictad22h02:
        ictad22h01:
    stor_05:
      hosts:
        ictad22h02:
        ictad22h01:
    meta_06:
      hosts:
        ictad22h02:
        ictad22h01:
    stor_06:
      hosts:
        ictad22h02:
        ictad22h01:
    meta_07:
      hosts:
        ictad22h02:
        ictad22h01:
```

```
        stor_07:
          hosts:
            ictad22h02:
            ictad22h01:
        meta_08:
          hosts:
            ictad22h02:
            ictad22h01:
        stor_08:
          hosts:
            ictad22h02:
            ictad22h01:
```

2. Creare il file `group_vars/mgmt.yml` e includere quanto segue:

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

3. Sotto `group_vars/`, creare i file per i gruppi di risorse `meta_01` attraverso `meta_08` utilizzando il seguente modello, inserire i valori segnaposto per ogni servizio che fa riferimento alla tabella seguente:

```
# 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>
```

Le dimensioni del volume vengono specificate come percentuale del pool di storage complessivo (definito anche gruppo di volumi). NetApp consiglia vivamente di lasciare una certa capacità libera in ogni pool per consentire lo spazio necessario per l'overprovisioning SSD (per ulteriori informazioni, vedere "Introduzione all'array NetApp EF600"). Il pool di storage, `beegfs_m1_m2_m5_m6`, alloca inoltre l'1% della capacità del pool per il servizio di gestione. Pertanto, per i volumi di metadati nel pool di storage, `beegfs_m1_m2_m5_m6`, Se si utilizzano dischi da 1,92 TB o 3,84 TB, impostare questo valore su `21.25`; Per dischi da 7,65 TB, impostare questo valore su `22.25`; E per i dischi da 15,3 TB, impostare questo valore su `23.75`.

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
meta_01.yml	8015	i1b:100.127.101.1/16 i2b:100.128.102.1/16	0	ictad22a01	beegfs_m1_m2_m5_m6	R
meta_02.yml	8025	i2b:100.128.102.2/16 i1b:100.127.101.2/16	0	ictad22a01	beegfs_m1_m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.101.3/16 i4b:100.128.102.3/16	1	ictad22a02	beegfs_m3_m4_m7_m8	R

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
meta_04.yml	8045	i4b:100.128.102.4/16 i3b:100.127.101.4/16	1	ictad22a02	beegfs_m3_m4_m7_m8	B
meta_05.yml	8055	i1b:100.127.101.5/16 i2b:100.128.102.5/16	0	ictad22a01	beegfs_m1_m2_m5_m6	R
meta_06.yml	8065	i2b:100.128.102.6/16 i1b:100.127.101.6/16	0	ictad22a01	beegfs_m1_m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.101.7/16 i4b:100.128.102.7/16	1	ictad22a02	beegfs_m3_m4_m7_m8	R
meta_08.yml	8085	i4b:100.128.102.8/16 i3b:100.127.101.8/16	1	ictad22a02	beegfs_m3_m4_m7_m8	B

4. Sotto `group_vars/`, creare i file per i gruppi di risorse `stor_01` attraverso `stor_08` utilizzando il seguente modello, inserire i valori segnaposto per ciascun servizio che fa riferimento all'esempio:

```
# 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>
```

Per le dimensioni corrette da utilizzare, vedere "Percentuali consigliate di overprovisioning del pool di storage".

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
stor_01.yml	8013	i1b:100.127.103.1/16 i2b:100.128.104.1/16	0	ictad22a01	beegfs_s1_s2	R
stor_02.yml	8023	i2b:100.128.104.2/16 i1b:100.127.103.2/16	0	ictad22a01	beegfs_s1_s2	B
stor_03.yml	8033	i3b:100.127.103.3/16 i4b:100.128.104.3/16	1	ictad22a02	beegfs_s3_s4	R
stor_04.yml	8043	i4b:100.128.104.4/16 i3b:100.127.103.4/16	1	ictad22a02	beegfs_s3_s4	B

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
stor_05.yml	8053	i1b:100.127.103.5/16 i2b:100.128.104.5/16	0	ictad22a01	beegfs_s5_s6	R
stor_06.yml	8063	i2b:100.128.104.6/16 i1b:100.127.103.6/16	0	ictad22a01	beegfs_s5_s6	B
stor_07.yml	8073	i3b:100.127.103.7/16 i4b:100.128.104.7/16	1	ictad22a02	beegfs_s7_s8	R
stor_08.yml	8083	i4b:100.128.104.8/16 i3b:100.127.103.8/16	1	ictad22a02	beegfs_s7_s8	B

Fase 3: Configurare l'inventario per un building block di metadati + storage

Questi passaggi descrivono come configurare un inventario Ansible per un building block di storage + metadati BeeGFS.

Fasi

1. Poll `inventory.yml`, inserire i seguenti parametri nella configurazione esistente:

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

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

2. Sotto `group_vars/`, creare i file per i gruppi di risorse `meta_09` attraverso `meta_16` utilizzando il seguente modello, inserire i valori segnaposto per ciascun servizio che fa riferimento all'esempio:

```
# 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>
```

Per le dimensioni corrette da utilizzare, vedere "Percentuali consigliate di overprovisioning del pool di storage".

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
meta_09.yml	8015	i1b:100.127.101.9/16 i2b:100.128.102.9/16	0	ictad22a03	beegfs_m9_m10_m13_m14	R
meta_10.yml	8025	i2b:100.128.102.10/16 i1b:100.127.101.10/16	0	ictad22a03	beegfs_m9_m10_m13_m14	B
meta_11.yml	8035	i3b:100.127.101.11/16 i4b:100.128.102.11/16	1	ictad22a04	beegfs_m11_m12_m15_m16	R
meta_12.yml	8045	i4b:100.128.102.12/16 i3b:100.127.101.12/16	1	ictad22a04	beegfs_m11_m12_m15_m16	B

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
meta_13.yml	8055	i1b:100.127.101.13/16 i2b:100.128.102.13/16	0	ictad22a03	beegfs_m9_m10_m13_m14	R
meta_14.yml	8065	i2b:100.128.102.14/16 i1b:100.127.101.14/16	0	ictad22a03	beegfs_m9_m10_m13_m14	B
meta_15.yml	8075	i3b:100.127.101.15/16 i4b:100.128.102.15/16	1	ictad22a04	beegfs_m11_m12_m15_m16	R
meta_16.yml	8085	i4b:100.128.102.16/16 i3b:100.127.101.16/16	1	ictad22a04	beegfs_m11_m12_m15_m16	B

3. Sotto `group_vars/`, creare file per gruppi di risorse `stor_09` attraverso `stor_16` utilizzando il seguente modello, inserire i valori segnaposto per ciascun servizio che fa riferimento all'esempio:

```
# 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>
```

Per le dimensioni corrette da utilizzare, vedere "Percentuali consigliate di overprovisioning del pool di storage"..

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
stor_09.yml	8013	i1b:100.127.103.9/16 i2b:100.128.104.9/16	0	ictad22a03	beegfs_s9_s10	R
stor_10.yml	8023	i2b:100.128.104.10/16 i1b:100.127.103.10/16	0	ictad22a03	beegfs_s9_s10	B
stor_11.yml	8033	i3b:100.127.103.11/16 i4b:100.128.104.11/16	1	ictad22a04	beegfs_s11_s12	R
stor_12.yml	8043	i4b:100.128.104.12/16 i3b:100.127.103.12/16	1	ictad22a04	beegfs_s11_s12	B
stor_13.yml	8053	i1b:100.127.103.13/16 i2b:100.128.104.13/16	0	ictad22a03	beegfs_s13_s14	R
stor_14.yml	8063	i2b:100.128.104.14/16 i1b:100.127.103.14/16	0	ictad22a03	beegfs_s13_s14	B
stor_15.yml	8073	i3b:100.127.103.15/16 i4b:100.128.104.15/16	1	ictad22a04	beegfs_s15_s16	R
stor_16.yml	8083	i4b:100.128.104.16/16 i3b:100.127.103.16/16	1	ictad22a04	beegfs_s15_s16	B

Fase 4: Configurare l'inventario per un building block di solo storage

Questi passaggi descrivono come configurare un inventario Ansible per un building block BeeGFS solo storage. La differenza principale tra l'impostazione della configurazione per un metadata + storage rispetto a un building block solo storage è l'omissione di tutti i gruppi di risorse di metadati e la modifica `criteria_drive_count` da 10 a 12 per ogni pool di storage.

Fasi

1. Poll `inventory.yml`, inserire i seguenti parametri nella configurazione esistente:

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

2. Sotto `group_vars/`, creare i file per i gruppi di risorse `stor_17` attraverso `stor_24` utilizzando il seguente modello, inserire i valori segnaposto per ciascun servizio che fa riferimento all'esempio:

```
# 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>
```

Per le dimensioni corrette da utilizzare, vedere "Percentuali consigliate di overprovisioning del pool di storage".

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
stor_17.yml	8013	i1b:100.127.103.17/16 i2b:100.128.104.17/16	0	ictad22a05	beegfs_s17_s18	R
stor_18.yml	8023	i2b:100.128.104.18/16 i1b:100.127.103.18/16	0	ictad22a05	beegfs_s17_s18	B
stor_19.yml	8033	i3b:100.127.103.19/16 i4b:100.128.104.19/16	1	ictad22a06	beegfs_s19_s20	R
stor_20.yml	8043	i4b:100.128.104.20/16 i3b:100.127.103.20/16	1	ictad22a06	beegfs_s19_s20	B

Nome del file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
stor_21.yml	8053	i1b:100.127.103.21/16 i2b:100.128.104.21/16	0	ictad22a05	beegfs_s21_s22	R
stor_22.yml	8063	i2b:100.128.104.22/16 i1b:100.127.103.22/16	0	ictad22a05	beegfs_s21_s22	B
stor_23.yml	8073	i3b:100.127.103.23/16 i4b:100.128.104.23/16	1	ictad22a06	beegfs_s23_s24	R
stor_24.yml	8083	i4b:100.128.104.24/16 i3b:100.127.103.24/16	1	ictad22a06	beegfs_s23_s24	B

Implementare BeeGFS

L'implementazione e la gestione della configurazione implica l'esecuzione di uno o più playbook contenenti le attività che Ansible deve eseguire e portare il sistema nello stato desiderato.

Anche se tutte le attività possono essere incluse in un singolo playbook, per i sistemi complessi, ciò diventa rapidamente poco pratico da gestire. Ansible consente di creare e distribuire i ruoli come metodo per il packaging di playbook riutilizzabili e contenuti correlati (ad esempio: Variabili predefinite, attività e gestori). Per ulteriori informazioni, consultare la documentazione Ansible per "Ruoli".

I ruoli vengono spesso distribuiti come parte di un insieme Ansible contenente ruoli e moduli correlati. Pertanto, questi playbook importano principalmente solo diversi ruoli distribuiti nelle varie raccolte NetApp e-Series Ansible.

> Attualmente, per implementare BeeGFS sono necessari almeno due building block (quattro nodi di file), a meno che un dispositivo di quorum separato non sia configurato come un interruttore a più livelli per mitigare eventuali problemi quando si stabilisce il quorum con un cluster a due nodi.

Fasi

1. Creare un nuovo `playbook.yml` archiviare e includere quanto segue:

```
# BeeGFS HA (High Availability) cluster playbook.
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
```

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

```
your playbook command with --list-tags to see all valid playbook tags."
      when: 'item not in ["all", "storage", "beegfs_ha",
"beegfs_ha_package", "beegfs_ha_configure",
"beegfs_ha_configure_resource", "beegfs_ha_performance_tuning",
"beegfs_ha_backup", "beegfs_ha_client"]'
      loop: "{{ ansible_run_tags }}"
  tasks:
    - name: Verify before proceeding.
      pause:
        prompt: "Are you ready to proceed with running the BeeGFS HA
role? Depending on the size of the deployment and network performance
between the Ansible control node and BeeGFS file and block nodes this
can take awhile (10+ minutes) to complete."
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_2
```

Questo playbook ne fa parte `pre_tasks` Verificare che Python 3 sia installato sui nodi di file e che i tag Ansible forniti siano supportati.

2. Utilizzare `ansible-playbook` Controlla con i file di inventario e playbook quando sei pronto per implementare BeeGFS.

   L'implementazione verrà eseguita completamente `pre_tasks`, Quindi richiedere la conferma dell'utente prima di procedere con l'effettiva implementazione di BeeGFS.

   Eseguire il seguente comando, regolando il numero di forche secondo necessità (vedere la nota seguente):

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

   In particolare per implementazioni più grandi, sovrascrivendo il numero predefinito di forcelle (5) utilizzando `forks` Si consiglia di aumentare il numero di host configurati in parallelo da Ansible. Per ulteriori informazioni, vedere "Ottimizzazione delle performance di Ansible" e. "Controllo dell'esecuzione del playbook".) L'impostazione del valore massimo dipende dalla potenza di elaborazione disponibile sul nodo di controllo Ansible. L'esempio precedente di 20 è stato eseguito su un nodo di controllo virtuale Ansible con 4 CPU (Intel® Xeon® Gold 6146 CPU @ 3,20 GHz).

   A seconda delle dimensioni dell'implementazione e delle prestazioni di rete tra il nodo di controllo Ansible e i nodi di blocco e file BeeGFS, il tempo di implementazione potrebbe variare.

Configurare i client BeeGFS

È necessario installare e configurare il client BeeGFS su tutti gli host che necessitano dell'accesso al file system BeeGFS, come i nodi di calcolo o GPU. Per questa attività, è possibile utilizzare Ansible e l'insieme BeeGFS.

Fasi

1. Se necessario, impostare SSH senza password dal nodo di controllo Ansible a ciascuno degli host che si desidera configurare come client BeeGFS:

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

2. Sotto `host_vars/`, Creare un file per ogni client BeeGFS denominato `<HOSTNAME>.yml` con il seguente contenuto, inserendo il testo segnaposto con le informazioni corrette per il tuo ambiente:

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

   Attualmente, su ciascun client devono essere configurate due interfacce InfiniBand, una per ciascuna delle due subnet IPoIB di storage. Se si utilizzano le subnet di esempio e gli intervalli consigliati per ciascun servizio BeeGFS qui elencati, i client devono avere un'interfaccia configurata nell'intervallo di `100.127.1. 0` a. `100.127.99.255` e l'altro in `100.128.1. 0` a. `100.128. 99.255`.

3. Creare un nuovo file `client_inventory.yml`, quindi inserire i seguenti parametri nella parte superiore:

   ```
   # 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.
   ```

   Non memorizzare le password in testo normale. Utilizzare invece Ansible Vault (vedere la documentazione Ansible per "Crittografia del contenuto con Ansible Vault") o utilizzare `--ask-become-pass` quando si esegue il playbook.

4. In `client_inventory.yml` File, elenca tutti gli host che devono essere configurati come client BeeGFS

in `beegfs_clients` E specificare eventuali configurazioni aggiuntive richieste per creare il modulo del kernel del client BeeGFS.

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

Quando si utilizzano i driver Mellanox OFED, assicurarsi che `beegfs_client_ofed_include_path` Indica il corretto "header include path" per l'installazione di Linux. Per ulteriori informazioni, consultare la documentazione di BeeGFS per "Supporto RDMA".

5. In `client_inventory.yml` Elencare i file system BeeGFS che si desidera montare nella parte inferiore di qualsiasi file definito in precedenza `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
```

Il `beegfs_client_config` rappresenta le impostazioni testate. Consultare la documentazione fornita con `netapp_eseries.beegfs` di raccolta `beegfs_client` ruolo per una panoramica completa di tutte le opzioni. Sono inclusi i dettagli sul montaggio di più file system BeeGFS o sul montaggio dello stesso file system BeeGFS più volte.

6. Creare un nuovo `client_playbook.yml` e compilare i seguenti parametri:

```
# 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
```

Omettere l'importazione di `netapp_eseries.host` raccolta e. `ipoib` Ruolo se sono già stati installati i driver IB/RDMA richiesti e gli IP configurati sulle interfacce IPoIB appropriate.

7. Per installare e creare il client e montare BeeGFS, eseguire il seguente comando:

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

8. Prima di mettere in produzione il file system BeeGFS, si consiglia di eseguire l'accesso a qualsiasi client `beegfs-fsck --checkfs` per garantire che tutti i nodi siano raggiungibili e che non vi siano problemi segnalati.

Scala oltre cinque elementi di base

È possibile configurare Pacemaker e Corosync per scalare oltre cinque blocchi costitutivi (10 nodi di file). Tuttavia, i cluster più grandi presentano alcuni inconvenienti e, alla fine, Pacemaker e Corosync impongono un massimo di 32 nodi.

NetApp ha testato solo i cluster BeeGFS ha per un massimo di 10 nodi; la scalabilità dei singoli cluster oltre questo limite non è consigliata o supportata. Tuttavia, i file system BeeGFS devono ancora scalare ben oltre 10 nodi, e NetApp lo ha considerato nella soluzione BeeGFS su NetApp.

Implementando più cluster ha contenenti un sottoinsieme dei blocchi costitutivi in ciascun file system, è possibile scalare il file system BeeGFS globale indipendentemente da qualsiasi limite consigliato o limite massimo sui meccanismi di clustering ha sottostanti. In questo scenario, procedere come segue:

- Creare un nuovo inventario Ansible che rappresenti i cluster ha aggiuntivi, quindi omettere la configurazione di un altro servizio di gestione. Puntare invece il `beegfs_ha_mgmtd_floating_ip` variabile in ogni cluster aggiuntivo `ha_cluster.yml` All'IP per il primo servizio di gestione BeeGFS.
- Quando si aggiungono cluster ha aggiuntivi allo stesso file system, assicurarsi di quanto segue:
  - Gli ID del nodo BeeGFS sono univoci.

- I nomi dei file corrispondenti a ciascun servizio in `group_vars` è unico in tutti i cluster.
- Gli indirizzi IP del client e del server BeeGFS sono univoci in tutti i cluster.
- Il primo cluster ha contenente il servizio di gestione BeeGFS è in esecuzione prima di tentare di implementare o aggiornare altri cluster.

- Gestire gli inventari per ciascun cluster ha separatamente nel proprio albero di directory.

  Il tentativo di combinare i file di inventario per più cluster in un unico albero di directory potrebbe causare problemi con il modo in cui il ruolo BeeGFS ha aggrega la configurazione applicata a un determinato cluster.

> Non è necessario che ogni cluster ha si adatti a cinque building block prima di crearne uno nuovo. In molti casi, l'utilizzo di un numero inferiore di building block per cluster è più semplice da gestire. Un approccio consiste nel configurare gli elementi di base in ogni singolo rack come cluster ha.

Percentuali consigliate di overprovisioning del pool di storage

Se si seguono le configurazioni standard dei quattro volumi per pool di storage per gli elementi di base di seconda generazione, fare riferimento alla tabella seguente.

Questa tabella fornisce le percentuali consigliate da utilizzare come dimensione del volume in `eseries_storage_pool_configuration` Per ogni destinazione di storage o metadati BeeGFS:

Dimensioni del disco	Dimensione
1,92 TB	18
3,84 TB	21.5
7,68 TB	22.5
15,3 TB	24

> Le indicazioni riportate sopra non si applicano al pool di storage contenente il servizio di gestione, che dovrebbe ridurre le dimensioni di cui sopra di .25% per allocare l'1% del pool di storage per i dati di gestione.

Per capire come sono stati determinati questi valori, vedere "TR-4800: Appendice A: Comprensione della durata e dell'overprovisioning degli SSD".

Building block ad alta capacità

La guida all'implementazione della soluzione BeeGFS standard delinea procedure e raccomandazioni per i requisiti di workload ad alte performance. I clienti che desiderano soddisfare requisiti di capacità elevata devono osservare le variazioni nell'implementazione e i consigli descritti di seguito.

[]

Controller

Per gli building block ad alta capacità, i controller EF600 devono essere sostituiti con controller EF300, ciascuno con un HIC Cascade installato per l'espansione SAS. Ogni nodo a blocchi avrà un numero minimo di SSD NVMe popolati nell'enclosure dell'array per lo storage dei metadati BeeGFS e saranno collegati agli shelf di espansione popolati con HDD NL-SAS per i volumi di storage BeeGFS.

La configurazione da nodo file a nodo di blocco rimane la stessa.

Posizionamento del disco

Per lo storage dei metadati BeeGFS sono richiesti almeno 4 SSD NVMe in ciascun nodo a blocchi. Questi dischi devono essere posizionati negli slot più esterni dell'enclosure.

[]

Vassoi di espansione

Il building block ad alta capacità può essere dimensionato con 1-7, 60 tray di espansione per unità per array di storage.

Per istruzioni su come collegare ciascun vassoio di espansione, "Fare riferimento al cablaggio EF300 per gli shelf di dischi".

Utilizzare architetture personalizzate

Panoramica e requisiti

Utilizzare qualsiasi sistema storage NetApp e/EF-Series come nodi a blocchi BeeGFS e server x86 come nodi di file BeeGFS quando si implementano cluster ad alta disponibilità BeeGFS utilizzando Ansible.

Le definizioni della terminologia utilizzata in questa sezione sono disponibili sul "termini e concetti" pagina.

Introduzione

Mentre "Architetture verificate da NetApp" fornire configurazioni di riferimento predefinite e indicazioni sul dimensionamento, alcuni clienti e partner potrebbero preferire progettare architetture personalizzate più adatte a requisiti o preferenze hardware particolari. Uno dei principali vantaggi della scelta di BeeGFS su NetApp è la capacità di implementare cluster ha a disco condiviso BeeGFS utilizzando Ansible, semplificando la gestione dei cluster e migliorando l'affidabilità con i componenti ha creati da NetApp. L'implementazione di architetture BeeGFS personalizzate su NetApp viene ancora eseguita utilizzando Ansible, mantenendo un approccio simile all'appliance su una gamma flessibile di hardware.

In questa sezione vengono descritti i passaggi generali necessari per implementare i file system BeeGFS sull'hardware NetApp e l'utilizzo di Ansible per configurare i file system BeeGFS. Per informazioni dettagliate sulle Best practice relative alla progettazione dei file system BeeGFS e per esempi ottimizzati, fare riferimento a. "Architetture verificate da NetApp" sezione.

Panoramica sull'implementazione

In genere, l'implementazione di un file system BeeGFS richiede i seguenti passaggi:

- Configurazione iniziale:
  - Installazione/cavo hardware.
  - Impostare i nodi di file e blocchi.
  - Impostare un nodo di controllo Ansible.
- Definire il file system BeeGFS come un inventario Ansible.
- Esegui Ansible su file e nodi a blocchi per implementare BeeGFS.
  - Facoltativamente per configurare i client e montare BeeGFS.

Le sezioni successive tratterà questi passaggi in modo più dettagliato.

Ansible gestisce tutte le attività di provisioning e configurazione del software, tra cui:

- Creazione/mappatura di volumi su nodi a blocchi.
- Formattazione/messa a punto di volumi su nodi di file.
- Installazione/configurazione del software sui nodi di file.
- Stabilire il cluster ha e configurare le risorse BeeGFS e i servizi del file system.

Requisiti

Il supporto per BeeGFS in Ansible viene rilasciato il "Ansible Galaxy" Insieme di ruoli e moduli che automatizzano l'implementazione e la gestione end-to-end dei cluster BeeGFS ha.

BeeGFS è dotato di versioni che seguono uno schema di controllo delle versioni di <major>.<minor>.<patch> e l'insieme mantiene i ruoli per ogni versione supportata di <major>.<minor> di BeeGFS, ad esempio BeeGFS 7.2 o BeeGFS 7.3. Man mano che vengono rilasciati gli aggiornamenti della raccolta, la versione della patch in ciascun ruolo verrà aggiornata in modo da indicare l'ultima versione di BeeGFS disponibile per tale ramo di release (esempio: 7.2.8). Ogni versione della raccolta viene testata e supportata anche con specifiche distribuzioni e versioni di Linux, attualmente Red Hat per i file node e RedHat e Ubuntu per i client. L'esecuzione di altre distribuzioni non è supportata e l'esecuzione di altre versioni (in particolare altre versioni principali) non è consigliata.

Nodo di controllo Ansible

Questo nodo conterrà l'inventario e i playbook utilizzati per gestire BeeGFS. Richiede:

- Ansible 6.x (ansible-core 2.13)
- Python 3.6 (o versione successiva)
- Pacchetti Python (pip): Ipaddr e netaddr

Si consiglia inoltre di impostare SSH senza password dal nodo di controllo a tutti i nodi di file e client BeeGFS.

Nodi di file BeeGFS

I file node devono eseguire RedHat 8.4 e avere accesso al repository ha contenente i pacchetti richiesti (pacemaker, corosync, fence-agents-all, resource-agents). Ad esempio, è possibile eseguire il seguente comando per abilitare il repository appropriato su RedHat 8:

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

Nodi client BeeGFS

È disponibile un ruolo Ansible del client BeeGFS per installare il pacchetto client BeeGFS e gestire i mount BeeGFS. Questo ruolo è stato testato con RedHat 8.4 e Ubuntu 22.04.

Se non si utilizza Ansible per configurare il client BeeGFS e montare BeeGFS, qualsiasi "BeeGFS supporta la distribuzione e il kernel Linux" può essere utilizzato.

Configurazione iniziale

Installazione e cavo hardware

Procedure necessarie per installare e collegare l'hardware utilizzato per eseguire BeeGFS su NetApp.

Pianificare l'installazione

Ciascun file system BeeGFS è costituito da un certo numero di nodi di file che eseguono i servizi BeeGFS utilizzando lo storage back-end fornito da un certo numero di nodi a blocchi. I file node sono configurati in uno o più cluster ad alta disponibilità per fornire la fault tolerance per i servizi BeeGFS. Ogni nodo a blocchi è già una coppia ha attiva/attiva. Il numero minimo di nodi di file supportati in ciascun cluster ha è tre e il numero massimo di nodi di file supportati in ciascun cluster è dieci. I file system BeeGFS possono scalare oltre dieci nodi implementando più cluster ha indipendenti che lavorano insieme per fornire un singolo namespace del file system.

In genere, ogni cluster ha viene implementato come una serie di "building block" in cui alcuni nodi di file (server x86) sono collegati direttamente a un certo numero di nodi a blocchi (in genere sistemi storage e-Series). Questa configurazione crea un cluster asimmetrico, in cui i servizi BeeGFS possono essere eseguiti solo su alcuni nodi di file che hanno accesso allo storage a blocchi di back-end utilizzato per le destinazioni BeeGFS. Il bilanciamento dei nodi file-to-block in ciascun building block e del protocollo di storage in uso per le connessioni dirette dipende dai requisiti di una particolare installazione.

Un'architettura di cluster ha alternativa utilizza un fabric di storage (noto anche come SAN (Storage Area Network) tra i nodi di file e blocchi per stabilire un cluster simmetrico. Ciò consente l'esecuzione dei servizi BeeGFS su qualsiasi nodo di file in un cluster ha specifico. Poiché i cluster generalmente simmetrici non sono così convenienti a causa dell'hardware SAN aggiuntivo, questa documentazione presuppone l'utilizzo di un cluster asimmetrico implementato come una serie di uno o più building block.

> Assicurarsi che l'architettura del file system desiderata per una particolare implementazione di BeeGFS sia ben compresa prima di procedere con l'installazione.

Hardware per rack

Quando si pianifica l'installazione, è importante che tutte le apparecchiature di ciascun building block siano installate in rack adiacenti. La procedura consigliata prevede il racking dei nodi di file immediatamente sopra i nodi di blocco in ciascun building block. Seguire la documentazione relativa ai modelli di file e. "blocco" nodi utilizzati durante l'installazione di guide e hardware nel rack.

Esempio di un singolo building block:

[esempio di building block]

Esempio di un'installazione BeeGFS di grandi dimensioni in cui sono presenti più elementi di base in ciascun cluster ha e più cluster ha nel file system:

[Esempio di implementazione di BeeGFS]

Nodi di blocco e file via cavo

In genere, le porte HIC dei nodi a blocchi e-Series vengono collegate direttamente alle porte dell'adattatore del canale host designato (per i protocolli InfiniBand) o dell'adattatore del bus host (per Fibre Channel e altri protocolli) dei nodi di file. Il modo esatto per stabilire queste connessioni dipende dall'architettura del file system desiderata, ecco un esempio "Basato sull'architettura verificata di seconda generazione di BeeGFS su NetApp":

[Esempio di file BeeGFS per bloccare il cablaggio del nodo]

Collegare i nodi di file alla rete client

Ogni nodo di file avrà un certo numero di porte InfiniBand o Ethernet designate per il traffico del client BeeGFS. A seconda dell'architettura, ciascun nodo di file disporrà di una o più connessioni a una rete client/storage dalle performance elevate, potenzialmente a più switch per la ridondanza e l'aumento della larghezza di banda. Di seguito viene riportato un esempio di cablaggio del client che utilizza switch di rete ridondanti, in cui le porte evidenziate in verde scuro e verde chiaro sono collegate a switch separati:

[Esempio di cablaggio del client BeeGFS]

Gestione delle connessioni rete e alimentazione

Stabilire le connessioni di rete necessarie per le reti in-band e out-of-band.

Collegare tutti gli alimentatori assicurandosi che ciascun nodo di file e blocchi sia collegato a più unità di distribuzione dell'alimentazione per la ridondanza (se disponibile).

Impostare i nodi di file e blocchi

Operazioni manuali necessarie per impostare i nodi di file e blocchi prima di eseguire Ansible.

Nodi di file

Configurare Baseboard Management Controller (BMC)

Un BMC (Baseboard Management Controller), a volte chiamato Service Processor, è il nome generico della funzionalità di gestione out-of-band integrata in varie piattaforme server che possono fornire accesso remoto anche se il sistema operativo non è installato o accessibile. I vendor in genere commercializzano questa funzionalità con un proprio marchio. Ad esempio, su Lenovo SR665, BMC viene definito XCC (Lenovo XClarity Controller).

Seguire la documentazione del vendor del server per abilitare le licenze necessarie per accedere a questa funzionalità e assicurarsi che il BMC sia connesso alla rete e configurato in modo appropriato per l'accesso remoto.

> Se si desidera utilizzare Redfish per la scherma basata su BMC, assicurarsi che Redfish sia attivato e che l'interfaccia BMC sia accessibile dal sistema operativo installato nel nodo file. Potrebbe essere necessaria una configurazione speciale sullo switch di rete se BMC e il sistema operativo condividono la stessa interfaccia di rete fisica.

Mettere a punto le impostazioni di sistema

Utilizzando l'interfaccia del programma di configurazione del sistema (BIOS/UEFI), assicurarsi che le impostazioni siano impostate per massimizzare le prestazioni. Le impostazioni esatte e i valori ottimali variano in base al modello di server in uso. Vengono fornite indicazioni per "modelli di file node verificati", in caso contrario, fare riferimento alla documentazione del vendor del server e alle best practice in base al modello in uso.

Installare un sistema operativo

Installare un sistema operativo supportato in base ai requisiti del nodo file elencati "qui". Fare riferimento a eventuali passaggi aggiuntivi riportati di seguito in base alla distribuzione Linux.

RedHat

Utilizza RedHat Subscription Manager per registrare e sottoscrivere il sistema per consentire l'installazione dei pacchetti richiesti dai repository ufficiali Red Hat e per limitare gli aggiornamenti alla versione supportata di Red Hat: `subscription-manager release --set=<MAJOR_VERSION>.<MINOR_VERSION>`. Per istruzioni, vedere "Come registrarsi e sottoscrivere un sistema RHEL" e. "Come limitare gli aggiornamenti".

Abilitare il repository Red Hat contenente i pacchetti richiesti per l'alta disponibilità:

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

Configurare la rete di gestione

Configurare le interfacce di rete necessarie per consentire la gestione in banda del sistema operativo. I passaggi esatti dipendono dalla distribuzione e dalla versione di Linux in uso.

> Assicurarsi che SSH sia attivato e che tutte le interfacce di gestione siano accessibili dal nodo di controllo Ansible.

Aggiornare il firmware HCA e HBA

Assicurarsi che tutti gli HBA e gli HCA eseguano le versioni firmware supportate elencate nella "Matrice di interoperabilità NetApp" e, se necessario, eseguire l'upgrade. Ulteriori consigli per gli adattatori NVIDIA ConnectX sono disponibili "qui".

Nodi a blocchi

Seguire i passi da a. "Inizia a lavorare con e-Series" per configurare la porta di gestione su ciascun controller di nodi a blocchi e, facoltativamente, impostare il nome dell'array di storage per ciascun sistema.

> Non è necessaria alcuna configurazione aggiuntiva oltre a garantire che tutti i nodi a blocchi siano accessibili dal nodo di controllo Ansible. La configurazione di sistema rimanente verrà applicata/mantenuta utilizzando Ansible.

Impostare Ansible Control Node

Impostare un nodo di controllo Ansible per implementare e gestire il file system.

Panoramica

Un nodo di controllo Ansible è una macchina Linux fisica o virtuale utilizzata per gestire il cluster. Deve soddisfare i seguenti requisiti:

- Scopri "requisiti" Per il ruolo BeeGFS ha, incluse le versioni installate di Ansible, Python e qualsiasi pacchetto Python aggiuntivo.
- Incontra il funzionario "Requisiti dei nodi di controllo Ansible" incluse le versioni del sistema operativo.
- Avere accesso SSH e HTTPS a tutti i nodi di file e blocchi.

È possibile trovare i passaggi dettagliati per l'installazione "qui".

Definire il file system BeeGFS

Panoramica di Ansible Inventory

L'inventario Ansible è un insieme di file di configurazione che definiscono il cluster BeeGFS ha desiderato.

Panoramica

Si consiglia di seguire le procedure standard di Ansible per organizzare il "inventario", incluso l'utilizzo di "sottodirectory/file" invece di memorizzare l'intero inventario in un file.

L'inventario Ansible per un singolo cluster BeeGFS ha è organizzato come segue:

[Panoramica di Ansible Inventory]

> Poiché un singolo file system BeeGFS può comprendere più cluster ha, è possibile che installazioni di grandi dimensioni dispongano di più inventari Ansible. In genere, non è consigliabile cercare di definire più cluster ha come un singolo inventario Ansible per evitare problemi.

Fasi

1. Nel nodo di controllo Ansible creare una directory vuota contenente l'inventario Ansible per il cluster BeeGFS che si desidera implementare.
   a. Se il file system conterrà più cluster ha, si consiglia di creare prima una directory per il file system, quindi sottodirectory per l'inventario che rappresenta ciascun cluster ha. Ad esempio:

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

2. Nella directory contenente l'inventario per il cluster ha che si desidera implementare, creare due directory `group_vars` e. `host_vars` e due file `inventory.yml` e. `playbook.yml`.

Nelle sezioni seguenti viene illustrata la definizione del contenuto di ciascuno di questi file.

Pianificare il file system

Pianificare l'implementazione del file system prima di creare l'inventario Ansible.

Panoramica

Prima di implementare il file system, è necessario definire gli indirizzi IP, le porte e le altre configurazioni richieste da tutti i nodi di file, i nodi di blocco e i servizi BeeGFS in esecuzione nel cluster. Anche se la configurazione esatta varia in base all'architettura del cluster, questa sezione definisce le Best practice e i passaggi da seguire che sono generalmente applicabili.

Fasi

1. Se si utilizza un protocollo di storage basato su IP (come iSER, iSCSI, NVMe/IB o NVMe/RoCE) per connettere i nodi di file ai nodi di blocco, compilare il seguente foglio di lavoro per ciascun building block. Ogni connessione diretta in un singolo building block deve avere una subnet univoca e non deve esserci alcuna sovrapposizione con le subnet utilizzate per la connettività client-server.

Nodo del file	Porta IB	Indirizzo IP	Nodo del blocco	Porta IB	IP fisico	Virtual IP (solo per EF600 con HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> Se i nodi di file e blocchi di ciascun building block sono collegati direttamente, spesso è possibile riutilizzare gli stessi IP/schema per più building block.

2. Indipendentemente dall'utilizzo di InfiniBand o RDMA su RoCE (Converged Ethernet) per la rete di storage, compilare il seguente foglio di lavoro per determinare gli intervalli IP che verranno utilizzati per i servizi cluster ha, i file service BeeGFS e i client per comunicare:

Scopo	**Porta InfiniBand**	**Indirizzo IP o intervallo**
IP cluster BeeGFS	<INTERFACE(s)>	<RANGE>
Gestione di BeeGFS	<INTERFACE(s)>	<IP(s)>
Metadati BeeGFS	<INTERFACE(s)>	<RANGE>
Storage BeeGFS	<INTERFACE(s)>	<RANGE>
Client BeeGFS	<INTERFACE(s)>	<RANGE>

   a. Se si utilizza una singola subnet IP, è necessario un solo foglio di lavoro, altrimenti compilare un foglio di lavoro per la seconda subnet.

3. In base a quanto sopra, per ciascun building block del cluster, compilare il seguente foglio di lavoro che definisce i servizi BeeGFS che verranno eseguiti. Per ogni servizio, specificare i nodi di file preferiti/secondari, la porta di rete, gli IP mobili, l'assegnazione di zona NUMA (se necessario) e i nodi di blocco da utilizzare per le destinazioni. Per compilare il foglio di lavoro, fare riferimento alle seguenti linee guida:

   a. Specificare i servizi BeeGFS come uno dei due `mgmt.yml`, `meta_<ID>.yml`, o. `storage_<ID>.yml` Dove ID rappresenta un numero univoco per tutti i servizi BeeGFS di quel tipo in questo file system. Questa convenzione semplifica il riferimento a questo foglio di lavoro nelle sezioni successive durante la creazione di file per configurare ciascun servizio.

   b. Le porte per i servizi BeeGFS devono essere univoche solo in un particolare building block. Assicurarsi che i servizi con lo stesso numero di porta non possano mai essere eseguiti sullo stesso nodo di file per evitare conflitti di porta.

   c. Se necessario, i servizi possono utilizzare volumi da più di un nodo a blocchi e/o pool di storage (e non tutti i volumi devono essere di proprietà dello stesso controller). Più servizi possono anche condividere lo stesso nodo a blocchi e/o la stessa configurazione del pool di storage (i singoli volumi verranno definiti in una sezione successiva).

Servizio BeeGFS (nome file)	Nodi di file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
<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>

Per ulteriori informazioni su convenzioni standard, Best practice e fogli di lavoro di esempio completi, fare riferimento a. "best practice" e. "Definire i blocchi di base BeeGFS" Sezioni di BeeGFS su NetApp Verified Architecture.

Definire i nodi di file e blocchi

Configurare singoli nodi di file

Specificare la configurazione per i singoli nodi di file utilizzando le variabili host (host_vars).

Panoramica

In questa sezione viene illustrata la compilazione di un `host_vars/<FILE_NODE_HOSTNAME>.yml` per ogni nodo di file nel cluster. Questi file devono contenere solo configurazioni univoche per un particolare nodo di file. Ciò comprende in genere:

- Definizione dell'IP o del nome host che Ansible deve utilizzare per connettersi al nodo.
- Configurazione di interfacce aggiuntive e IP del cluster utilizzati per i servizi cluster ha (Pacemaker e Corosync) per comunicare con altri nodi di file. Per impostazione predefinita, questi servizi utilizzano la stessa rete dell'interfaccia di gestione, ma devono essere disponibili interfacce aggiuntive per la ridondanza. La pratica comune consiste nel definire IP aggiuntivi sulla rete di storage, evitando la necessità di un cluster aggiuntivo o di una rete di gestione.
  - Le performance di qualsiasi rete utilizzata per la comunicazione in cluster non sono critiche per le performance del file system. Con la configurazione predefinita del cluster, in genere, almeno una rete a 1 Gbps fornirà prestazioni sufficienti per le operazioni del cluster, come la sincronizzazione degli stati dei nodi e il coordinamento delle modifiche dello stato delle risorse del cluster. Le reti lente/occupate possono richiedere più tempo del solito per le modifiche dello stato delle risorse e, in casi estremi, potrebbero causare l'eliminazione dei nodi dal cluster se non riescono a inviare heartbeat in un intervallo di tempo ragionevole.
- Configurazione delle interfacce utilizzate per la connessione ai nodi a blocchi sul protocollo desiderato (ad esempio: ISCSI/iSER, NVMe/IB, NVMe/RoCE, FCP, ecc.)

Fasi

Facendo riferimento allo schema di indirizzamento IP definito in "Pianificare il file system" per ciascun nodo del file nel cluster, creare un file `host_vars/<FILE_NODE_HOSTNAME>/yml` e compilarlo come segue:

1. In alto, specificare l'IP o il nome host che Ansible deve utilizzare per SSH al nodo e gestirlo:

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

2. Configurare IP aggiuntivi che possono essere utilizzati per il traffico del cluster:
   a. Se il tipo di rete è "InfiniBand (con 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. Se il tipo di rete è "RDMA su Ethernet convergente (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. Se il tipo di rete è "Ethernet (solo TCP, senza 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. Indicare gli IP da utilizzare per il traffico del cluster, con gli IP preferiti elencati più in alto:

```
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.
```

> Gli IPS configurati nella fase due non verranno utilizzati come IP del cluster a meno che non siano inclusi in `beegfs_ha_cluster_node_ips` elenco. Ciò consente di configurare IP/interfacce aggiuntive utilizzando Ansible che possono essere utilizzati per altri scopi, se necessario.

4. Se il nodo del file deve comunicare per bloccare i nodi su un protocollo basato su IP, gli IP dovranno essere configurati sull'interfaccia appropriata e tutti i pacchetti richiesti per quel protocollo

installati/configurati.

a. Se in uso "ISCSI":

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

b. Se in uso "Er":

```
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. Se in uso "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. Se in uso "NVMe/RoCE":

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

e. Altri protocolli:

i. Se in uso "NVMe/FC", la configurazione di singole interfacce non è richiesta. L'implementazione del cluster BeeGFS rileverà automaticamente il protocollo e installerà/configurerà i requisiti in base alle necessità. Se si utilizza un fabric per connettere file e nodi a blocchi, assicurarsi che gli switch siano correttamente suddivisi in zone seguendo le Best practice di NetApp e del vendor di switch.

ii. L'utilizzo di FCP o SAS non richiede l'installazione o la configurazione di software aggiuntivo. Se si utilizza FCP, assicurarsi che gli switch siano correttamente zonati "NetApp" e le best practice del tuo fornitore di switch.

iii. Al momento non si consiglia l'utilizzo di IB SRP. Utilizzare NVMe/IB o iSER a seconda di ciò che i nodi a blocchi e-Series supportano.

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta un singolo nodo di file.

Advanced (Avanzate): Passaggio tra le modalità Ethernet e InfiniBand degli adattatori VPI NVIDIA ConnectX

Gli adattatori NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) supportano sia InfiniBand che Ethernet come layer di trasporto. Il passaggio da una modalità all’altra non viene negoziato automaticamente e deve essere configurato utilizzando <code>mstconfig</code> strumento incluso in <code>mstflint</code>, un pacchetto open source che fa parte di "Mellanox firmare Tools (MFT)". La modifica della modalità degli adattatori deve essere eseguita una sola volta. Questa operazione può essere eseguita manualmente o inclusa nell’inventario Ansible come parte di qualsiasi interfaccia configurata con <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> sezione dell’inventario, per fare in modo che venga controllata/applicata automaticamente.

Ad esempio, per modificare una corrente di interfaccia in modalità InfiniBand su Ethernet in modo che possa essere utilizzata per RoCE:

1. Per ogni interfaccia che si desidera configurare, specificare `mstconfig` come un mapping (o dizionario) che specifica `LINK_TYPE_P<N>` dove `<N>` È determinato dal numero di porta dell'HCA per l'interfaccia. Il `<N>` il valore può essere determinato eseguendo `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` E aggiungendo 1 all'ultimo numero dal nome dello slot PCI e convertendo in decimale.
   a. Ad esempio `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → porta HCA 3) → `LINK_TYPE_P3: eth`:

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

Per ulteriori informazioni, consultare "Documentazione della raccolta di host NetApp e-Series" per il tipo di interfaccia/protocollo in uso.

Configurare singoli nodi a blocchi

Specificare la configurazione per i singoli nodi di blocco utilizzando le variabili host (host_vars).

Panoramica

In questa sezione viene illustrata la compilazione di un `host_vars/<BLOCK_NODE_HOSTNAME>.yml` file per ciascun nodo del blocco nel cluster. Questi file devono contenere solo una configurazione univoca per un nodo di blocco specifico. Ciò comprende in genere:

- Il nome del sistema (visualizzato in System Manager).
- L'URL HTTPS per uno dei controller (utilizzato per gestire il sistema utilizzando l'API REST).
- Quali nodi di file del protocollo di storage utilizzano per connettersi a questo nodo a blocchi.
- Configurazione delle porte della scheda di interfaccia host (HIC), ad esempio gli indirizzi IP (se necessario).

Fasi

Facendo riferimento allo schema di indirizzamento IP definito in "Pianificare il file system" per ogni nodo del blocco nel cluster creare un file `host_vars/<BLOCK_NODE_HOSTNAME>/yml` e compilarlo come segue:

1. Nella parte superiore, specificare il nome del sistema e l'URL HTTPS per uno dei controller:

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

2. Selezionare "protocollo" i nodi di file utilizzeranno per connettersi a questo nodo di blocco:
   a. Protocolli supportati: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. A seconda del protocollo in uso, le porte HIC potrebbero richiedere una configurazione aggiuntiva. Se necessario, definire la configurazione della porta HIC in modo che la voce superiore nella configurazione di ciascun controller corrisponda alla porta fisica più a sinistra su ciascun controller e la porta inferiore alla porta più a destra. Tutte le porte richiedono una configurazione valida anche se non sono attualmente in uso.

> Consultare anche la sezione seguente se si utilizza HDR (200 GB) InfiniBand o 200GB RoCE con nodi a blocchi EF600.

   a. Per 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. Per 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. Per 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. Per 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. I protocolli FC e SAS non richiedono ulteriori configurazioni. SRP non è consigliato correttamente.

Per ulteriori opzioni di configurazione delle porte HIC e dei protocolli host, inclusa la possibilità di configurare iSCSI CHAP, fare riferimento a. "documentazione" Incluso nella raccolta SANtricity. Nota durante l'implementazione di BeeGFS, il pool di storage, la configurazione del volume e altri aspetti del provisioning dello storage verranno configurati altrove e non devono essere definiti in questo file.

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta un singolo nodo a blocchi.

Utilizzando HDR (200 GB) InfiniBand o 200GB RoCE con i nodi a blocchi NetApp EF600:

Per utilizzare HDR (200 GB) InfiniBand con EF600, è necessario configurare un secondo IP "virtuale" per ciascuna porta fisica. Questo è un esempio del modo corretto per configurare un EF600 dotato di InfiniBand HDR HIC a due porte:

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

Specificare la configurazione del nodo file comune

Specificare la configurazione del nodo file comune utilizzando le variabili di gruppo (group_vars).

Panoramica

La configurazione che deve essere utilizzata per tutti i nodi di file è definita in `group_vars/ha_cluster.yml`. In genere include:

- Dettagli su come connettersi e accedere a ciascun nodo di file.
- Configurazione di rete comune.
- Se sono consentiti riavvii automatici.
- Modalità di configurazione degli stati firewall e selinux.
- Configurazione del cluster, inclusi avvisi e scherma.
- Tuning delle performance.
- Configurazione comune del servizio BeeGFS.

Le opzioni impostate in questo file possono essere definite anche su singoli nodi di file, ad esempio se sono in uso modelli hardware misti o se si dispone di password diverse per ciascun nodo. La configurazione sui singoli nodi di file avrà la precedenza sulla configurazione in questo file.

Fasi

Creare il file `group_vars/ha_cluster.yml` e compilarlo come segue:

1. Indicare come il nodo Ansible Control deve autenticare con gli host remoti:

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

In particolare per gli ambienti di produzione, non memorizzare le password in testo normale. Utilizzare invece Ansible Vault (vedere "Crittografia del contenuto con Ansible Vault") o il `--ask-become-pass` quando si esegue il playbook. Se il `ansible_ssh_user` è già root, quindi è possibile omettere il `ansible_become_password`.

2. Se si configurano IP statici sulle interfacce ethernet o InfiniBand (ad esempio gli IP del cluster) e più interfacce si trovano nella stessa subnet IP (ad esempio se ib0 utilizza 192.168.1.10/24 e ib1 utilizza 192.168.1.11/24), Per il corretto funzionamento del supporto multi-homed, è necessario configurare ulteriori tabelle e regole di routing IP. È sufficiente attivare il gancio di configurazione dell'interfaccia di rete fornito come segue:

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

3. Durante l'implementazione del cluster, a seconda del protocollo di storage potrebbe essere necessario riavviare i nodi per facilitare il rilevamento dei dispositivi a blocchi remoti (volumi e-Series) o applicare altri aspetti della configurazione. Per impostazione predefinita, i nodi richiedono prima del riavvio, ma è possibile consentire il riavvio automatico dei nodi specificando quanto segue:

```
eseries_common_allow_host_reboot: true
```

   a. Per impostazione predefinita, dopo un riavvio, per garantire che i dispositivi a blocchi e gli altri servizi siano pronti, Ansible attenderà fino al sistema `default.target` viene raggiunto prima di continuare con l'implementazione. In alcuni scenari, quando NVMe/IB è in uso, potrebbe non essere abbastanza lungo da inizializzare, rilevare e connettersi a dispositivi remoti. Ciò può causare la continuità prematura dell'implementazione automatica e il malfunzionamento. Per evitare questo problema quando si utilizza NVMe/IB, definire anche quanto segue:

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

4. Per comunicare con i servizi cluster BeeGFS e ha sono necessarie diverse porte firewall. A meno che non si desideri configurare il firwewall manualmente (non consigliato), specificare quanto segue per creare le zone firewall richieste e aprire automaticamente le porte:

```
beegfs_ha_firewall_configure: True
```

5. Al momento SELinux non è supportato e si consiglia di impostare lo stato su Disabled (disattivato) per evitare conflitti (soprattutto quando RDMA è in uso). Impostare quanto segue per assicurarsi che SELinux sia disattivato:

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

6. Configurare l'autenticazione in modo che i file node siano in grado di comunicare, regolando le impostazioni predefinite in base alle policy aziendali:

```
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. Basato su "Pianificare il file system" Sezione specificare l'IP di gestione BeeGFS per questo file system:

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

> Anche se apparentemente ridondante, `beegfs_ha_mgmtd_floating_ip` È importante quando si scala il file system BeeGFS oltre un singolo cluster ha. I cluster ha successivi vengono implementati senza un servizio di gestione BeeGFS aggiuntivo e puntano al servizio di gestione fornito dal primo cluster.

8. Attivare gli avvisi e-mail se lo si desidera:

```
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. Si consiglia vivamente di abilitare la scherma, altrimenti i servizi possono essere bloccati per l'avvio sui nodi secondari quando il nodo primario non funziona.
   a. Abilitare la scherma a livello globale specificando quanto segue:

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

      i. Prendere nota di eventuali supporti "proprietà del cluster" può anche essere specificato qui, se necessario. In genere, la regolazione di questi elementi non è necessaria, in quanto il ruolo BeeGFS ha viene fornito con una serie di elementi ben testati "valori predefiniti".

   b. Selezionare e configurare un agente di scherma:
      i. OPZIONE 1: Per abilitare la recinzione utilizzando le PDU (Power Distribution Unit) APC:

```
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. OPZIONE 2: Per abilitare la scherma utilizzando le API Redfish fornite da Lenovo XCC (e da altri BMC):

```
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. Per ulteriori informazioni sulla configurazione di altri agenti di scherma, fare riferimento a. "Documentazione RedHat".

10. Il ruolo BeeGFS ha può applicare diversi parametri di tuning per ottimizzare ulteriormente le performance. Questi includono l'ottimizzazione dell'utilizzo della memoria del kernel e l'i/o dei dispositivi a blocchi, tra gli altri parametri. Il ruolo viene fornito con una serie ragionevole di "valori predefiniti" In base ai test eseguiti con i nodi a blocchi NetApp e-Series, ma per impostazione predefinita questi non vengono applicati a meno che non si specifichi:

```
beegfs_ha_enable_performance_tuning: True
```

a. Se necessario, specificare qui eventuali modifiche all'ottimizzazione predefinita delle prestazioni. Consulta l'articolo completo "parametri di ottimizzazione delle performance" documentazione per ulteriori dettagli.

11. Per garantire che gli indirizzi IP mobili (talvolta noti come interfacce logiche) utilizzati per i servizi BeeGFS possano eseguire il failover tra i nodi di file, tutte le interfacce di rete devono essere denominate in modo coerente. Per impostazione predefinita, i nomi delle interfacce di rete vengono generati dal kernel, che non è garantito per generare nomi coerenti, anche su modelli di server identici con adattatori di rete installati negli stessi slot PCIe. Ciò è utile anche quando si creano inventari prima dell'implementazione dell'apparecchiatura e si conoscono i nomi delle interfacce generate. Per garantire nomi di dispositivi coerenti, in base a un diagramma a blocchi del server o. `lshw -class network -businfo` Output, specificare il mapping indirizzo PCIe desiderato per l'interfaccia logica come segue:

a. Per le interfacce di rete InfiniBand (IPoIB):

```
eseries_ipoib_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:41:00.0: i1a
```

b. Per le interfacce di rete Ethernet:

```
eseries_ip_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:41:00.0: e1a
```

Per evitare conflitti quando le interfacce vengono rinominate (impedendone la ridenominazione), non utilizzare nomi predefiniti potenziali come eth0, ens9f0, ib0 o ibs4f0. Una convenzione di denominazione comune prevede l'utilizzo di 'e' o 'i' per Ethernet o InfiniBand, seguito dal numero dello slot PCIe e da una lettera che indica la porta. Ad esempio, la seconda porta di un adattatore InfiniBand installato nello slot 3 è: I3b.

Se si utilizza un modello di nodo di file verificato, fare clic su "qui" Esempio di mapping indirizzo-porta logica PCIe.

12. Specificare facoltativamente la configurazione da applicare a tutti i servizi BeeGFS nel cluster. È possibile trovare i valori di configurazione predefiniti "qui"e la configurazione per servizio viene specificata altrove:
    a. Servizio di gestione BeeGFS:

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

    b. Servizi di metadati BeeGFS:

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

    c. Servizi di storage BeeGFS:

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

13. A partire da BeeGFS 7.2.7 e 7.3.1 "autenticazione della connessione" deve essere configurato o disabilitato esplicitamente. Esistono alcuni modi per configurarlo utilizzando la distribuzione basata su Ansible:
    a. Per impostazione predefinita, l'implementazione configura automaticamente l'autenticazione della connessione e genera un `connauthfile` Che verranno distribuiti a tutti i nodi di file e utilizzati con i servizi BeeGFS. Questo file verrà anche posizionato/mantenuto nel nodo di controllo Ansible all'indirizzo `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` dove deve essere mantenuto (in modo sicuro) per il riutilizzo con i client che devono accedere a questo file system.
        i. Per generare una nuova chiave, specificare `-e "beegfs_ha_conn_auth_force_new=True` Quando si esegue il playbook Ansible. Nota: Questa operazione viene ignorata se si seleziona `beegfs_ha_conn_auth_secret` è definito.
        ii. Per le opzioni avanzate, fare riferimento all'elenco completo dei valori predefiniti inclusi in "Ruolo BeeGFS ha".

b. È possibile utilizzare un segreto personalizzato definendo quanto segue in `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. L'autenticazione della connessione può essere disattivata completamente (NON consigliata):

```
beegfs_ha_conn_auth_enabled: false
```

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta la configurazione di un nodo di file comune.

Utilizzo di HDR (200 GB) InfiniBand con i nodi a blocchi NetApp EF600:

Per utilizzare HDR (200 GB) InfiniBand con EF600, il gestore di subnet deve supportare la virtualizzazione. Se i nodi di file e blocchi sono collegati mediante uno switch, questo deve essere attivato nel gestore di subnet per il fabric complessivo.

Se i nodi di blocco e di file sono collegati direttamente mediante InfiniBand, un'istanza di `opensm` deve essere configurato su ogni nodo di file per ogni interfaccia direttamente connessa a un nodo di blocco. Per eseguire questa operazione, specificare `configure: true` quando "configurazione delle interfacce di storage dei nodi di file".

Attualmente la versione inbox di `opensm` Fornito con le distribuzioni Linux supportate non supporta la virtualizzazione. È invece necessario installare e configurare la versione di `opensm` Da Mellanox OpenFabrics Enterprise Distribution (OFED). Sebbene la distribuzione con Ansible sia ancora supportata, sono necessari alcuni passaggi aggiuntivi:

1. Utilizzando curl o il tool desiderato, scaricare i pacchetti per la versione di opensm elencata nella "requisiti tecnologici" Dal sito Web di Mellanox al `<INVENTORY>/packages/` directory. Ad esempio:

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

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

2. Sotto `group_vars/ha_cluster.yml` definire la seguente configurazione:

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

eseries_ib_opensm_options:
  virt_enabled: "2"
```

Specificare la configurazione del nodo del blocco comune

Specificare la configurazione del nodo a blocchi comune utilizzando le variabili di gruppo (group_vars).

Panoramica

La configurazione che deve essere utilizzata per tutti i nodi a blocchi è definita in `group_vars/eseries_storage_systems.yml`. In genere include:

- Dettagli su come il nodo di controllo Ansible deve connettersi ai sistemi storage e-Series utilizzati come nodi a blocchi.
- Quali versioni del firmware, DI NVSRAM e del firmware del disco devono essere eseguite dai nodi.
- Configurazione globale, incluse le impostazioni della cache, la configurazione dell'host e le modalità di provisioning dei volumi.

Le opzioni impostate in questo file possono essere definite anche su singoli nodi a blocchi, ad esempio se sono in uso modelli hardware misti o se si dispone di password diverse per ciascun nodo. La configurazione sui singoli nodi a blocchi avrà la precedenza sulla configurazione in questo file.

Fasi

Creare il file `group_vars/eseries_storage_systems.yml` e compilarlo come segue:

1. Ansible non utilizza SSH per connettersi ai nodi a blocchi, ma le API REST. Per ottenere questo risultato, dobbiamo stabilire:

```
ansible_connection: local
```

2. Specificare il nome utente e la password per gestire ciascun nodo. Il nome utente può essere omesso (e l'impostazione predefinita è admin), altrimenti è possibile specificare qualsiasi account con privilegi di amministratore. Specificare inoltre se i certificati SSL devono essere verificati o ignorati:

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

Si sconsiglia di elencare le password in testo non crittografato. Utilizzare Ansible vault o fornire il `eseries_system_password` Quando si esegue Ansible con --extra-vars.

3. Facoltativamente, specificare il firmware del controller, NVSRAM e il firmware del disco da installare sui nodi. Questi dovranno essere scaricati su `packages/` Prima di eseguire Ansible. È possibile scaricare il firmware del controller e-Series e NVSRAM "qui" e firmware del disco "qui":

```
eseries_firmware_firmware: "packages/<FILENAME>.dlp" # Ex.
"packages/RCB_11.70.2_6000_61b1131d.dlp"
eseries_firmware_nvsram: "packages/<FILENAME>.dlp" # Ex.
"packages/N6000-872834-D06.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".
```

Se viene specificata questa configurazione, Ansible aggiorna automaticamente tutto il firmware, incluso il riavvio dei controller (se necessario) con senza richieste aggiuntive. Si prevede che ciò non causi interruzioni per BeeGFS/host i/o, ma può causare una temporanea diminuzione delle performance.

4. Regolare le impostazioni predefinite della configurazione globale del sistema. Le opzioni e i valori elencati di seguito sono generalmente consigliati per BeeGFS su NetApp, ma possono essere modificati se necessario:

```
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. Configurare le impostazioni predefinite per il provisioning di volumi globali. Le opzioni e i valori elencati di seguito sono generalmente consigliati per BeeGFS su NetApp, ma possono essere modificati se necessario:

```
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. Se necessario, modificare l'ordine in cui Ansible selezionerà le unità per i pool di storage e i gruppi di volumi tenendo presente le seguenti Best practice:
   a. Elencare tutte le unità (potenzialmente più piccole) che devono essere utilizzate per i volumi di gestione e/o metadati e i volumi di storage durano.
   b. Assicurarsi di bilanciare l'ordine di selezione dei dischi tra i canali disponibili in base ai modelli di shelf di dischi/enclosure di dischi. Ad esempio, con EF600 e senza espansioni, i dischi 0-11 si trovano sul canale 1 del disco e i dischi 12-23 sul canale del disco. Pertanto, è necessario scegliere una strategia per bilanciare la selezione del disco `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, ecc. In caso di più enclosure, la prima cifra rappresenta l'ID dello shelf del disco.

```
# 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"
```

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta la configurazione di un nodo a blocchi comune.

Definire i servizi BeeGFS

Definire il servizio di gestione BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di gestione BeeGFS. Nel cluster ha per un file system specifico dovrebbe esistere un solo servizio di questo tipo. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (gestione).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come deve essere un volume per memorizzare i dati per questo servizio (la destinazione di gestione di BeeGFS).

Fasi

Creare un nuovo file `group_vars/mgmt.yml` e facendo riferimento a. "Pianificare il file system" compilare la sezione come segue:

1. Indicare che questo file rappresenta la configurazione per un servizio di gestione BeeGFS:

```
beegfs_service: management
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. Questo non è generalmente richiesto per il servizio di gestione a meno che non sia necessario attivare le quote, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-mgmtd.conf` può essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `connNetFilterFile`.

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

3. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile`

opzione):

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

5. Specificare l'obiettivo di gestione di BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida:
   a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).
   b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di pool/volumi di storage. Nota quando si utilizzano gli SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'SSD (fare clic "qui" per ulteriori dettagli).
   c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: ictad22a01
    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
```

Fare clic su "qui" Per un esempio di un file di inventario completo che rappresenta un servizio di gestione BeeGFS.

Definire il servizio di metadati BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di metadati BeeGFS. Almeno un servizio di questo tipo dovrebbe esistere nei cluster ha per un particolare file system. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (metadati).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come deve essere un volume per memorizzare i dati per questo servizio (la destinazione dei metadati BeeGFS).

Fasi

Facendo riferimento a. "Pianificare il file system" creare un file in `group_vars/meta_<ID>.yml` per ogni servizio di metadati nel cluster e compilarli come segue:

1. Indica che questo file rappresenta la configurazione per un servizio di metadati BeeGFS:

```
beegfs_service: metadata
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. È necessario specificare almeno la porta TCP e UDP desiderata, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-meta.conf` può anche essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `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. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile` opzione):

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

5. Specificare la destinazione dei metadati BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida (questa operazione configurerà automaticamente anche `storeMetaDirectory` opzione):

a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).

b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di pool/volumi di storage. Nota quando si utilizzano gli SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'SSD (fare clic "qui" per ulteriori dettagli).

c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: ictad22a01
    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
```

Fare clic su "qui" Per un esempio di un file di inventario completo che rappresenta un servizio di metadati BeeGFS.

Definire il servizio di storage BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di storage BeeGFS. Almeno un servizio di questo tipo dovrebbe esistere nei cluster ha per un particolare file system. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (storage).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come devono essere i volumi per memorizzare i dati per questo servizio (le destinazioni di storage BeeGFS).

Fasi

Facendo riferimento a. "Pianificare il file system" creare un file in `group_vars/stor_<ID>.yml` per ogni servizio di storage nel cluster e compilarli come segue:

1. Indicare che questo file rappresenta la configurazione per un servizio di storage BeeGFS:

```
beegfs_service: storage
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. È necessario specificare almeno la porta TCP e UDP desiderata, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-storage.conf` può anche essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `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. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile` opzione):

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

5. Specificare le destinazioni di storage BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida (questa operazione configurerà automaticamente anche `storeStorageDirectory` opzione):

   a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).

   b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di

pool/volumi di storage. Nota quando si utilizzano gli SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'SSD (fare clic "qui" per ulteriori dettagli).

c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: ictad22a01
    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
```

Fare clic su "qui" Esempio di un file di inventario completo che rappresenta un servizio di storage BeeGFS.

Mappare i servizi BeeGFS ai nodi di file

Specificare quali nodi di file possono eseguire ciascun servizio BeeGFS utilizzando `inventory.yml` file.

Panoramica

In questa sezione viene illustrato come creare `inventory.yml` file. Ciò include l'elenco di tutti i nodi a blocchi e la specifica dei nodi di file che possono eseguire ciascun servizio BeeGFS.

Fasi

Creare il file `inventory.yml` e compilarlo come segue:

1. Dall'inizio del file, creare la struttura di inventario Ansible standard:

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

2. Creare un gruppo contenente tutti i nodi a blocchi che partecipano a questo cluster ha:

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

3. Creare un gruppo che conterrà tutti i servizi BeeGFS nel cluster e i nodi di file che li eseguiranno:

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

4. Per ogni servizio BeeGFS nel cluster, definire il nodo/i file preferito/i e secondario/i che deve eseguire tale servizio:

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

Fare clic su "qui" per un esempio di file di inventario completo.

Implementare il file system BeeGFS

Panoramica di Ansible Playbook

Implementazione e gestione di cluster BeeGFS ha con Ansible.

Panoramica

Nelle sezioni precedenti sono illustrate le fasi necessarie per creare un inventario Ansible che rappresenti un cluster BeeGFS ha. Questa sezione presenta l'automazione Ansible sviluppata da NetApp per implementare e gestire il cluster.

Ansible: Concetti chiave

Prima di procedere, è utile acquisire familiarità con alcuni concetti chiave di Ansible:

- Le attività da eseguire su un inventario Ansible sono definite in ciò che è noto come **playbook**.
  - La maggior parte delle attività di Ansible è progettata per essere **idempotent**, il che significa che possono essere eseguite più volte per verificare che la configurazione/stato desiderato sia ancora applicato senza interrompere le operazioni o eseguire aggiornamenti non necessari.
- La più piccola unità di esecuzione di Ansible è un modulo *.
  - I playbook tipici utilizzano più moduli.
    - Esempi: Scaricare un pacchetto, aggiornare un file di configurazione, avviare/abilitare un servizio.
  - NetApp distribuisce i moduli per automatizzare i sistemi NetApp e-Series.
- L'automazione complessa è meglio integrata come ruolo.
  - Essenzialmente un formato standard per la distribuzione di un playbook riutilizzabile.
  - NetApp distribuisce i ruoli per host Linux e file system BeeGFS.

Ruolo BeeGFS ha per Ansible: Concetti chiave

Tutta l'automazione necessaria per implementare e gestire ogni versione di BeeGFS su NetApp viene fornita come ruolo Ansible e distribuita come parte di "NetApp e-Series Ansible Collection per BeeGFS":

- Questo ruolo può essere considerato tra un **installer** e un moderno motore di **implementazione/gestione** per BeeGFS.
  - Applica l'infrastruttura moderna come pratiche di codice e filosofie per semplificare la gestione dell'infrastruttura di storage su qualsiasi scala.
  - Simile a come "Kubespray" Project consente agli utenti di implementare/gestire un'intera distribuzione Kubernetes per un'infrastruttura di calcolo scale-out.
- Questo ruolo è il formato * software-defined* utilizzato da NetApp per il packaging, la distribuzione e la manutenzione di BeeGFS su soluzioni NetApp.
  - Cerca di creare un'esperienza simile a quella di un'appliance senza dover distribuire un'intera distribuzione Linux o un'immagine di grandi dimensioni.
  - Include agenti di risorse cluster compatibili con Open Cluster Framework (OCF) creati da NetApp per destinazioni BeeGFS personalizzate, indirizzi IP e monitoraggio che forniscono un'integrazione intelligente di Pacemaker/BeeGFS.
- Questo ruolo non è semplicemente "automazione" dell'implementazione ed è destinato a gestire l'intero ciclo di vita del file system, tra cui:
  - Applicazione di modifiche e aggiornamenti della configurazione per servizio o a livello di cluster.
  - Automazione della riparazione e del ripristino del cluster dopo la risoluzione dei problemi hardware.
  - Semplificazione dell'ottimizzazione delle performance con valori predefiniti impostati in base a test approfonditi con volumi BeeGFS e NetApp.
  - Verifica e correzione della deriva della configurazione.

NetApp offre anche un ruolo Ansible per "Client BeeGFS", Che può essere utilizzato facoltativamente per installare BeeGFS e montare file system su nodi di calcolo/GPU/login.

Implementare il cluster BeeGFS ha

Specificare le attività da eseguire per implementare il cluster BeeGFS ha utilizzando un manuale.

Panoramica

Questa sezione descrive come assemblare il manuale standard utilizzato per implementare/gestire BeeGFS su NetApp.

Fasi

Creare il manuale Ansible Playbook

Creare il file `playbook.yml` e compilarlo come segue:

1. Per prima cosa, definire una serie di attività (comunemente denominate a. "gioca") Che dovrebbe essere eseguito solo sui nodi a blocchi NetApp e-Series. Viene utilizzata un'attività di pausa per richiedere conferma prima di eseguire l'installazione (per evitare l'esecuzione accidentale di un playbook), quindi importare `nar_santricity_management` ruolo. Questo ruolo gestisce l'applicazione di qualsiasi configurazione generale del sistema definita in `group_vars/eseries_storage_systems.yml` o individuale `host_vars/<BLOCK NODE>.yml` file.

```
- 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. Definire il gioco che verrà eseguito su tutti i nodi di file e blocchi:

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

3. All'interno di questo gioco è possibile definire facoltativamente un set di "pre-task" che devono essere

eseguiti prima di implementare il cluster ha. Questo può essere utile per verificare/installare qualsiasi prerequisito come Python. Possiamo anche effettuare controlli prima del volo, ad esempio verificando che i tag Ansible forniti siano supportati:

```
  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. Infine, questo gioco importa il ruolo BeeGFS ha per la versione di BeeGFS che si desidera implementare:

```
  tasks:
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_3 # Alternatively specify: beegfs_ha_7_2.
```

> Viene mantenuto un ruolo BeeGFS ha per ciascuna versione principale.minore supportata di BeeGFS. Questo consente agli utenti di scegliere quando aggiornare le versioni principali/secondarie. Attualmente BeeGFS 7.3.x (`beegfs_7_3`) O BeeGFS 7.2.x (`beegfs_7_2`) sono supportati. Per impostazione predefinita, entrambi i ruoli implementeranno la versione più recente delle patch BeeGFS al momento del rilascio, anche se gli utenti possono scegliere di eseguire l'override e distribuire la patch più recente, se lo desiderano. Fare riferimento alla versione più recente "guida all'upgrade" per ulteriori dettagli.

5. Facoltativo: Se si desidera definire attività aggiuntive, tenere presente se le attività devono essere indirizzate a. `all` Host (inclusi i sistemi storage e-Series) o solo i nodi di file. Se necessario, definire un nuovo gioco specifico per i nodi di file utilizzando `- hosts: ha_cluster`.

Fare clic su "qui" per un esempio di un file di playbook completo.

Installare NetApp Ansible Collections

L'insieme BeeGFS per Ansible e tutte le dipendenze vengono mantenute su "Ansible Galaxy". Sul nodo di controllo Ansible eseguire il seguente comando per installare la versione più recente:

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

Sebbene non sia generalmente consigliato, è anche possibile installare una versione specifica della raccolta:

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

Eseguire il Playbook

Dalla directory del nodo di controllo Ansible contenente `inventory.yml` e. `playbook.yml` eseguire il playbook come segue:

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

In base alle dimensioni del cluster, l'implementazione iniziale può richiedere oltre 20 minuti. Se l'implementazione non riesce per qualsiasi motivo, correggere eventuali problemi (ad esempio, cablaggio errato, nodo non avviato, ecc.) e riavviare il playbook Ansible.

Quando si specifica "configurazione di un nodo di file comune", Se si sceglie l'opzione predefinita per fare in modo che Ansible gestisca automaticamente l'autenticazione basata sulla connessione, un `connAuthFile` utilizzato come segreto condiviso è ora disponibile all'indirizzo `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (per impostazione predefinita). Tutti i client che hanno bisogno di accedere al file system dovranno utilizzare questo segreto condiviso. Questo viene gestito automaticamente se i client vengono configurati utilizzando "Ruolo del client BeeGFS".

Implementare i client BeeGFS

In alternativa, è possibile utilizzare Ansible per configurare i client BeeGFS e montare il file system.

Panoramica

L'accesso ai file system BeeGFS richiede l'installazione e la configurazione del client BeeGFS su ciascun nodo che deve montare il file system. In questa sezione viene descritto come eseguire queste attività utilizzando la disponibile "Ruolo Ansible".

Fasi

Creare il file di inventario del client

1. Se necessario, impostare SSH senza password dal nodo di controllo Ansible a ciascuno degli host che si desidera configurare come client BeeGFS:

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

2. Sotto `host_vars/`, Creare un file per ogni client BeeGFS denominato `<HOSTNAME>.yml` con il seguente contenuto, inserendo il testo segnaposto con le informazioni corrette per il tuo ambiente:

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

3. Se si desidera utilizzare i ruoli di NetApp e-Series host Collection per configurare le interfacce InfiniBand o Ethernet per consentire ai client di connettersi ai nodi di file BeeGFS, è possibile includere uno dei seguenti elementi:

   a. Se il tipo di rete è "InfiniBand (con 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. Se il tipo di rete è "RDMA su Ethernet convergente (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. Se il tipo di rete è "Ethernet (solo TCP, senza 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. Creare un nuovo file `client_inventory.yml` E specificare l'utente che Ansible deve utilizzare per connettersi a ciascun client e la password che Ansible deve utilizzare per l'escalation dei privilegi (ciò richiede `ansible_ssh_user` essere root o avere privilegi sudo):

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

Non memorizzare le password in testo normale. Utilizzare invece il vault Ansible (vedere la "Documentazione Ansible" Per crittografare il contenuto con Ansible Vault) o utilizzare `--ask-become-pass` quando si esegue il playbook.

5. In `client_inventory.yml` File, elenca tutti gli host che devono essere configurati come client BeeGFS in `beegfs_clients` Fare riferimento ai commenti inline e rimuovere eventuali commenti aggiuntivi necessari per creare il modulo del kernel del client BeeGFS sul sistema:

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

      vars:
        # OPTION 1: If you're using the Mellanox OFED drivers and they
are already installed:
        #eseries_ib_skip: True # Skip installing inbox drivers when
using the IPoIB role.
        #beegfs_client_ofed_enable: True
        #beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"

        # OPTION 2: If you're using inbox IB/RDMA drivers and they are
already installed:
        #eseries_ib_skip: True # Skip installing inbox drivers when
using the IPoIB role.

        # OPTION 3: If you want to use inbox IB/RDMA drivers and need
them installed/configured.
        #eseries_ib_skip: False # Default value.
        #beegfs_client_ofed_enable: False # Default value.
```

Quando si utilizzano i driver Mellanox OFED, assicurarsi che beegfs_client_ofed_include_path punti al "header include path" corretto per l'installazione di Linux. Per ulteriori informazioni, consultare la documentazione di BeeGFS per "Supporto RDMA".

6. In `client_inventory.yml` Elencare i file system BeeGFS che si desidera montare sotto qualsiasi file definito in precedenza `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. A partire da BeeGFS 7.2.7 e 7.3.1 "autenticazione della connessione" deve essere configurato o disabilitato esplicitamente. A seconda di come si sceglie di configurare l'autenticazione basata sulla connessione quando si specifica "configurazione di un nodo di file comune", potrebbe essere necessario modificare la configurazione del client:
   a. Per impostazione predefinita, l'implementazione del cluster ha configurerà automaticamente l'autenticazione della connessione e genererà un `connauthfile` Che verranno posizionati/mantenuti sul nodo di controllo Ansible in `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Per impostazione predefinita, il ruolo del client BeeGFS è impostato per leggere/distribuire questo file ai client definiti in `client_inventory.yml`e non sono necessarie ulteriori azioni.
      i. Per le opzioni avanzate, fare riferimento all'elenco completo dei valori predefiniti inclusi in "Ruolo del client BeeGFS".
   b. Se si sceglie di specificare un segreto personalizzato con `beegfs_ha_conn_auth_secret` specificarlo in `client_inventory.yml` anche file:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Se si sceglie di disattivare completamente l'autenticazione basata sulla connessione con `beegfs_ha_conn_auth_enabled`, specificare che in `client_inventory.yml` anche file:

```
beegfs_ha_conn_auth_enabled: false
```

Per un elenco completo dei parametri supportati e ulteriori dettagli, fare riferimento a. "Documentazione completa del client BeeGFS". Per un esempio completo di un inventario client, fare clic su "qui".

Creare il file Playbook del client BeeGFS

1. Creare un nuovo file `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. Facoltativo: Se si desidera utilizzare i ruoli di NetApp e-Series host Collection per configurare le interfacce per la connessione dei client ai file system BeeGFS, importare il ruolo corrispondente al tipo di interfaccia che si sta configurando:
   a. Se si utilizza InfiniBand (IPoIB):

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

   b. Se si utilizza RDMA su Ethernet convergente (RoCE):

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

   c. Se si utilizza Ethernet (solo TCP, senza RDMA):

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

3. Infine, importare il ruolo del client BeeGFS per installare il software client e configurare i supporti del file system:

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

Per un esempio completo di un playbook client, fai clic "qui".

Eseguire il manuale BeeGFS Client Playbook

Per installare/creare il client e montare BeeGFS, eseguire il seguente comando:

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

Verificare l'implementazione di BeeGFS

Verificare l'implementazione del file system prima di mettere il sistema in produzione.

Panoramica

Prima di mettere il file system BeeGFS in produzione, eseguire alcuni controlli di verifica.

Fasi

1. Accedere a qualsiasi client ed eseguire quanto segue per assicurarsi che tutti i nodi previsti siano presenti/raggiungibili e che non siano segnalate incoerenze o altri problemi:

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

2. Arrestare l'intero cluster, quindi riavviarlo. Da qualsiasi nodo di file eseguire quanto segue:

```
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. Mettere ciascun nodo in standby e verificare che i servizi BeeGFS siano in grado di eseguire il failover su nodi secondari. Per eseguire questa operazione, accedere a uno dei nodi di file ed eseguire quanto segue:

```
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. Utilizza strumenti di benchmarking delle performance come IOR e MDTest per verificare che le performance del file system soddisfino le aspettative. Esempi di test e parametri comuni utilizzati con BeeGFS sono disponibili nella "Verifica del progetto" Sezione di BeeGFS su NetApp Verified Architecture.

È necessario eseguire test aggiuntivi in base ai criteri di accettazione definiti per un sito/installazione particolare.

Amministrare i cluster BeeGFS

Panoramica, concetti chiave e terminologia

Scopri come amministrare i cluster BeeGFS ha dopo che sono stati implementati.

Panoramica

Questa sezione è destinata agli amministratori dei cluster che devono gestire i cluster BeeGFS ha dopo la loro implementazione. Anche coloro che hanno familiarità con i cluster ha Linux dovrebbero leggere attentamente questa guida, poiché esistono diverse differenze nella gestione del cluster, in particolare per quanto riguarda la riconfigurazione dovuta all'utilizzo di Ansible.

Concetti chiave

Alcuni di questi concetti sono stati introdotti nel principale "termini e concetti" È utile reintrodurli nel contesto di un cluster BeeGFS ha:

Nodo cluster: un server che esegue i servizi Pacemaker e Corosync e che partecipa al cluster ha.

Nodo file: nodo cluster utilizzato per eseguire uno o più servizi di gestione, metadati o storage BeeGFS.

Nodo a blocchi: un sistema storage NetApp e-Series che fornisce storage a blocchi ai nodi di file. Questi nodi non fanno parte del cluster BeeGFS ha in quanto offrono funzionalità ha standalone proprie. Ciascun nodo è costituito da due storage controller che forniscono alta disponibilità al livello di blocco.

Servizio BeeGFS: Un servizio di gestione, metadati o storage BeeGFS. Ogni nodo di file eseguirà uno o più servizi che utilizzeranno i volumi sul nodo di blocco per memorizzare i propri dati.

Building Block: implementazione standardizzata di file node BeeGFS, nodi a blocchi e-Series e servizi BeeGFS in esecuzione su di essi che semplifica la scalabilità di un cluster/file system BeeGFS ha in base a un'architettura verificata di NetApp. Sono supportati anche i cluster ha personalizzati, ma spesso seguono un approccio simile a building block per semplificare la scalabilità.

Cluster BeeGFS ha: un numero scalabile di nodi di file utilizzati per eseguire i servizi BeeGFS supportati da nodi a blocchi per memorizzare i dati BeeGFS in modo altamente disponibile. Basato su componenti open-source collaudati nel settore, Pacemaker e Corosync utilizzano Ansible per il packaging e l'implementazione.

Servizi cluster: si riferisce ai servizi Pacemaker e Corosync in esecuzione su ciascun nodo che partecipa al cluster. Nota è possibile che un nodo non esegua alcun servizio BeeGFS e partecipi al cluster come nodo "Tiebreaker" nel caso in cui vi sia solo la necessità di due nodi di file.

Risorse del cluster: per ogni servizio BeeGFS in esecuzione nel cluster vengono visualizzate una risorsa di monitoraggio BeeGFS e un gruppo di risorse contenente risorse per destinazioni BeeGFS, indirizzi IP (IP mobili) e il servizio BeeGFS stesso.

Ansible: Uno strumento per il provisioning del software, la gestione della configurazione e l'implementazione delle applicazioni, che consente l'infrastruttura come codice. È il modo in cui i cluster BeeGFS ha vengono confezionati per semplificare il processo di implementazione, riconfigurazione e aggiornamento di BeeGFS su NetApp.

Pcs: interfaccia a riga di comando disponibile da qualsiasi nodo di file nel cluster utilizzato per eseguire query e controllare lo stato dei nodi e delle risorse nel cluster.

Terminologia comune

Failover: ciascun servizio BeeGFS dispone di un nodo di file preferito su cui verrà eseguito, a meno che tale nodo non si guasti. Quando un servizio BeeGFS viene eseguito su un nodo di file non preferito/secondario, si dice che sia in failover.

Failover: l'atto di spostare i servizi BeeGFS da un nodo di file non preferito al nodo preferito.

Coppia ha: due nodi di file che possono accedere allo stesso insieme di nodi a blocchi sono talvolta indicati come coppia ha. Si tratta di un termine comune utilizzato in NetApp per fare riferimento a due controller o nodi storage che possono "assumere" l'uno con l'altro.

Modalità di manutenzione: Disattiva il monitoraggio di tutte le risorse e impedisce a Pacemaker di spostare o gestire in altro modo le risorse nel cluster (vedere anche la sezione a. "modalità di manutenzione").

Cluster ha: uno o più file node che eseguono servizi BeeGFS che possono eseguire il failover tra più nodi nel cluster per creare un file system BeeGFS ad alta disponibilità. Spesso i file node sono configurati in coppie ha in grado di eseguire un sottoinsieme dei servizi BeeGFS nel cluster.

Quando utilizzare Ansible rispetto allo strumento PC

Quando si deve utilizzare Ansible rispetto allo strumento della riga di comando di PC per gestire il cluster ha?

Tutte le attività di implementazione e riconfigurazione del cluster devono essere completate utilizzando Ansible da un nodo di controllo Ansible esterno. Le modifiche temporanee nello stato del cluster (ad esempio, l'inserimento e l'uscita dei nodi in standby) vengono eseguite in genere accedendo a un nodo del cluster (preferibilmente uno che non è degradato o che sta per essere sottoposto a manutenzione) e utilizzando la riga di comando di PC.

La modifica di qualsiasi configurazione del cluster, incluse risorse, vincoli, proprietà e i servizi BeeGFS stessi, deve essere eseguita sempre utilizzando Ansible. La manutenzione di una copia aggiornata dell'inventario e del playbook Ansible (idealmente nel controllo del codice sorgente per tenere traccia delle modifiche) fa parte della manutenzione del cluster. Quando è necessario apportare modifiche alla configurazione, aggiornare l'inventario ed eseguire nuovamente il playbook Ansible che importa il ruolo BeeGFS ha.

Il ruolo ha gestirà il posizionamento del cluster in modalità di manutenzione e l'esecuzione delle modifiche necessarie prima di riavviare BeeGFS o i servizi cluster per applicare la nuova configurazione. Poiché i riavvii completi dei nodi non sono generalmente necessari al di fuori dell'implementazione iniziale, il rerunning di Ansible è generalmente considerato una procedura "sicura", ma è sempre consigliato durante le finestre di manutenzione o fuori orario nel caso in cui sia necessario riavviare qualsiasi servizio BeeGFS. In genere, questi riavvii non dovrebbero causare errori nelle applicazioni, ma potrebbero compromettere le prestazioni (che alcune applicazioni potrebbero gestire meglio di altre).

Il rerunning di Ansible è anche un'opzione quando si desidera riportare l'intero cluster a uno stato completamente ottimale e in alcuni casi potrebbe essere in grado di ripristinare lo stato del cluster più facilmente rispetto all'utilizzo di PC. Soprattutto in caso di emergenza in cui il cluster non è attivo per qualche motivo, una volta eseguito il backup di tutti i nodi, Ansible può ripristinare il cluster in modo più rapido e affidabile rispetto al tentativo di utilizzare i PC.

Esaminare lo stato del cluster

Utilizzare i PC per visualizzare lo stato del cluster.

Panoramica

In esecuzione `pcs status` Da qualsiasi nodo del cluster è il modo più semplice per visualizzare lo stato complessivo del cluster e lo stato di ciascuna risorsa (ad esempio i servizi BeeGFS e le relative dipendenze). In questa sezione vengono illustrate le informazioni disponibili nell'output di `pcs status` comando.

Comprendere l'output di `pcs status`

Eseguire `pcs status` Su qualsiasi nodo del cluster in cui vengono avviati i servizi del cluster (pacemaker e Corosync). La parte superiore dell'output mostra un riepilogo del cluster:

```
[root@ictad22h01 ~]# pcs status
Cluster name: hacluster
Cluster Summary:
  * Stack: corosync
  * Current DC: ictad22h01 (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
ictad22h01
  * 6 nodes configured
  * 235 resource instances configured
```

La sezione seguente elenca i nodi nel cluster:

```
Node List:
  * Node ictad22h06: standby
  * Online: [ ictad22h01 ictad22h02 ictad22h04 ictad22h05 ]
  * OFFLINE: [ ictad22h03 ]
```

Ciò indica in particolare i nodi in standby o offline. I nodi in standby partecipano ancora al cluster ma sono contrassegnati come non idonei per l'esecuzione delle risorse. I nodi offline indicano che i servizi cluster non sono in esecuzione su quel nodo, a causa di un arresto manuale o perché il nodo è stato riavviato/arrestato.

> Quando i nodi si avviano per la prima volta, i servizi del cluster vengono arrestati e devono essere avviati manualmente per evitare il failover accidentale delle risorse su un nodo non integro.

Se i nodi sono in standby o non in linea a causa di un motivo non amministrativo (ad esempio un errore), accanto allo stato del nodo viene visualizzato un testo aggiuntivo tra parentesi. Ad esempio, se la funzione di scherma è disattivata e una risorsa rileva un errore, viene visualizzato `Node <HOSTNAME>: standby (on-fail)`. Un altro stato possibile è `Node <HOSTNAME>: UNCLEAN (offline)`, che sarà visto brevemente come un nodo è in fase di recintamento, ma continuerà se la scherma non è riuscita, indicando che il cluster non può confermare lo stato del nodo (questo può bloccare l'avvio delle risorse su altri nodi).

La sezione successiva mostra un elenco di tutte le risorse del cluster e dei relativi stati:

```
Full List of Resources:
  * mgmt-monitor    (ocf::eseries:beegfs-monitor):   Started ictad22h01
  * Resource Group: mgmt-group:
    * mgmt-FS1  (ocf::eseries:beegfs-target):    Started ictad22h01
    * mgmt-IP1  (ocf::eseries:beegfs-ipaddr2):   Started ictad22h01
    * mgmt-IP2  (ocf::eseries:beegfs-ipaddr2):   Started ictad22h01
    * mgmt-service  (systemd:beegfs-mgmtd):  Started ictad22h01
[...]
```

In modo simile ai nodi, viene visualizzato un testo aggiuntivo accanto allo stato della risorsa tra parentesi in caso di problemi con la risorsa. Ad esempio, se il pacemaker richiede un arresto della risorsa e non riesce a completarlo entro il tempo assegnato, il pacemaker tenterà di individuare il nodo. Se la funzione di scherma è disattivata o l'operazione di scherma non riesce, lo stato della risorsa sarà `FAILED <HOSTNAME> (blocked)` E Pacemaker non sarà in grado di avviarlo su un nodo diverso.

Vale la pena notare che i cluster BeeGFS ha utilizzano diversi agenti di risorse OCF personalizzati ottimizzati per BeeGFS. In particolare, il monitor BeeGFS è responsabile dell'attivazione di un failover quando le risorse BeeGFS su un nodo specifico non sono disponibili.

Riconfigurare e aggiornare

Riconfigurare il cluster ha e BeeGFS

Utilizzare Ansible per riconfigurare il cluster.

Panoramica

Generalmente, la riconfigurazione di qualsiasi aspetto del cluster BeeGFS ha deve essere eseguita aggiornando l'inventario Ansible e rieseguendo `ansible-playbook` comando. Ciò include l'aggiornamento degli avvisi, la modifica della configurazione di scherma permanente o la regolazione della configurazione del servizio BeeGFS. Queste vengono regolate mediante `group_vars/ha_cluster.yml` il file e un elenco completo delle opzioni sono disponibili in "Specificare la configurazione del nodo file comune" sezione.

Per ulteriori informazioni sulle opzioni di configurazione selezionate di cui gli amministratori devono essere a conoscenza durante la manutenzione o la manutenzione del cluster, vedere di seguito.

Come disattivare e attivare la funzione di scherma

Per impostazione predefinita, la funzione di scherma viene attivata/richiesta durante la configurazione del cluster. In alcuni casi, potrebbe essere consigliabile disattivare temporaneamente la scherma per garantire che i nodi non vengano accidentalmente arrestati durante determinate operazioni di manutenzione (ad esempio l'aggiornamento del sistema operativo). Anche se questa funzione può essere disattivata manualmente, gli amministratori devono tenere presente che esistono compromessi.

OPZIONE 1: Disattiva scherma utilizzando Ansible (consigliato).

Quando la funzione di scherma viene disattivata utilizzando Ansible, l'azione on-fail del monitor BeeGFS passa da "fence" a "standby". Ciò significa che se il monitor BeeGFS rileva un errore, tenterà di mettere il nodo in standby e di eseguire il failover di tutti i servizi BeeGFS. Al di fuori del troubleshooting/test attivo, questo è in genere più desiderabile dell'opzione 2. Lo svantaggio è che se una risorsa non si ferma sul nodo originale,

viene impedita l'avvio da un'altra parte (motivo per cui la scherma è generalmente richiesta per i cluster di produzione).

1. Nel tuo inventario Ansible all'indirizzo `groups_vars/ha_cluster.yml` aggiungere la seguente configurazione:

```
beegfs_ha_cluster_crm_config_options:
    stonith-enabled: False
```

2. Rieseguire il manuale Ansible per applicare le modifiche al cluster.

OPZIONE 2: Disattiva manualmente la funzione di scherma.

In alcuni casi, potrebbe essere necessario disattivare temporaneamente la scherma senza eseguire nuovamente Ansible, ad esempio per facilitare la risoluzione dei problemi o il test del cluster.

> In questa configurazione, se il monitor BeeGFS rileva un errore, il cluster tenta di arrestare il gruppo di risorse corrispondente. Non attiverà un failover completo né tenterà di riavviare o spostare il gruppo di risorse interessato in un altro host. Per risolvere il problema, risolvere i problemi che si verificano in seguito `pcs resource cleanup` oppure mettere manualmente il nodo in standby.

Fasi:

1. Per determinare se la scherma (stonith) è attivata o disattivata globalmente, eseguire: `pcs property show stonith-enabled`
2. Per disattivare la funzione di scherma: `pcs property set stonith-enabled=false`
3. Per attivare la funzione di scherma: `pcs property set stonith-enabled=true`

Nota: Questa impostazione verrà ignorata alla prossima esecuzione del playbook Ansible.

Aggiornare il cluster ha e BeeGFS

Utilizzare Ansible per aggiornare BeeGFS e il cluster ha.

Panoramica

BeeGFS è dotato di una versione successiva a `major.minor.patch` Schema di versione e ruoli BeeGFS ha Ansible sono forniti per ogni BeeGFS supportato `major.minor` versione (ad esempio `beegfs_ha_7_2` e. `beegfs_ha_7_3`). Ogni ruolo ha viene associato all'ultima versione della patch BeeGFS al momento del rilascio della raccolta Ansible.

Ansible deve essere utilizzato per tutti gli aggiornamenti di BeeGFS, incluso il passaggio tra le versioni principali, minori e patch di BeeGFS. Per aggiornare BeeGFS, è necessario innanzitutto aggiornare la raccolta BeeGFS Ansible, che include anche le correzioni e i miglioramenti più recenti all'automazione della distribuzione/gestione e al cluster ha sottostante. Anche dopo l'aggiornamento alla versione più recente della raccolta, BeeGFS non verrà aggiornato fino a. `ansible-playbook` viene eseguito con `-e "beegfs_ha_force_upgrade=true"` impostare.

Per ulteriori informazioni sulle versioni di BeeGFS, vedere "Documentazione sull'aggiornamento di BeeGFS".

Percorsi di upgrade testati

Ogni versione dell'insieme BeeGFS viene testata con versioni specifiche di BeeGFS per garantire l'interoperabilità tra tutti i componenti. Viene inoltre eseguito un test per garantire che gli aggiornamenti possano essere eseguiti dalle versioni di BeeGFS supportate dall'ultima versione della raccolta a quelle supportate nell'ultima release.

Versione originale	Versione dell'aggiornamento	Multirrail	Dettagli
7.2.6	7.3.2	Sì	Aggiornamento della raccolta beegfs da v3.0.1 a v3.1.0, aggiunta di multi-rail
7.2.6	7.2.8	No	Aggiornamento della raccolta beegfs da v3.0.1 a v3.1.0
7.2.8	7.3.1	Sì	Aggiornamento con la raccolta beegfs v3.1.0, aggiunta di multi-rail
7.3.1	7.3.2	Sì	Eseguire l'aggiornamento utilizzando la raccolta beegfs v3.1.0

Passaggi per l'aggiornamento di BeeGFS

Le sezioni seguenti illustrano i passaggi per aggiornare l'insieme BeeGFS Ansible e BeeGFS stesso. Prestare particolare attenzione a eventuali passaggi aggiuntivi per l'aggiornamento delle versioni principali o secondarie di BeeGFS.

Fase 1: Aggiornamento di BeeGFS Collection

Per gli aggiornamenti del ritiro con accesso a. "Ansible Galaxy", eseguire il seguente comando:

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

Per gli aggiornamenti offline della raccolta, scarica la raccolta da "Ansible Galaxy" facendo clic sul pulsante desiderato `Install Version` e poi `Download tarball`. Trasferire il tarball al nodo di controllo Ansible ed eseguire il seguente comando.

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

Vedere "Installazione delle raccolte" per ulteriori informazioni.

Fase 2: Aggiornamento di Ansible Inventory

Eseguire gli aggiornamenti necessari o desiderati ai file di inventario Ansible del cluster. Vedere "Note sull'aggiornamento della versione" di seguito sono riportati i dettagli relativi ai requisiti di aggiornamento specifici. Vedere "USA architetture personalizzate" Sezione per informazioni generali sulla configurazione dell'inventario BeeGFS ha.

Fase 3: Aggiornamento di Ansible Playbook (solo quando si aggiornano versioni principali o minori)

Se si sta passando da una versione principale a una minore, in `playbook.yml` file utilizzato per implementare e gestire il cluster, aggiornare il nome di `beegfs_ha_<VERSION>` per riflettere la versione desiderata. Ad esempio, se si desidera implementare BeeGFS 7.3 `beegfs_ha_7_3`:

```
- 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_3
```

Per ulteriori informazioni sul contenuto di questo file di playbook, consultare "Implementare il cluster BeeGFS ha" sezione.

Fase 4: Eseguire BeeGFS Upgrade

Per applicare l'aggiornamento BeeGFS:

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

Dietro le quinte, il ruolo di BeeGFS ha gestirà:

- Assicurarsi che il cluster si trovi in uno stato ottimale con ciascun servizio BeeGFS situato sul nodo preferito.
- Impostare il cluster in modalità di manutenzione.
- Aggiornare i componenti del cluster ha (se necessario).
- Aggiornare ciascun nodo di file uno alla volta come segue:
  - Metterlo in standby e eseguire il failover dei servizi sul nodo secondario.
  - Aggiornare i pacchetti BeeGFS.
  - Servizi di fallback.
- Spostare il cluster fuori dalla modalità di manutenzione.

Note sull'aggiornamento della versione

Aggiornamento da BeeGFS versione 7.2.6 o 7.3.0

Modifiche all'autenticazione basata su connessione

Le versioni di BeeGFS rilasciate dopo la 7.3.1 non consentono più l'avvio dei servizi senza specificare un

`connAuthFile` o impostazione `connDisableAuthentication=true` nel file di configurazione del servizio. Si consiglia vivamente di attivare la protezione dell'autenticazione basata sulla connessione. Vedere "Autenticazione basata su connessione BeeGFS" per ulteriori informazioni.

Per impostazione predefinita, il `beegfs_ha*` I ruoli genereranno e distribuiranno questo file, aggiungendolo anche al nodo di controllo Ansible all'indirizzo `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`. Il `beegfs_client` role verificherà anche la presenza di questo file e lo fornirà ai client, se disponibili.

> Se il `beegfs_client` il ruolo non è stato utilizzato per configurare i client; questo file deve essere distribuito manualmente a ciascun client e a. `connAuthFile` configurazione in `beegfs-client.conf` file impostato per utilizzarlo. Quando si esegue l'aggiornamento da una versione precedente di BeeGFS in cui l'autenticazione basata sulla connessione non era abilitata, i client perderanno l'accesso a meno che l'autenticazione basata sulla connessione non sia disattivata come parte dell'aggiornamento mediante l'impostazione `beegfs_ha_conn_auth_enabled: false` poll `group_vars/ha_cluster.yml` (sconsigliato).

Per ulteriori dettagli e opzioni di configurazione alternative, vedere la procedura per configurare l'autenticazione della connessione in "Specificare la configurazione del nodo file comune" sezione.

Assistenza e manutenzione

Servizi di failover e failback

Spostamento dei servizi BeeGFS tra nodi cluster.

Panoramica

I servizi BeeGFS possono eseguire il failover tra i nodi del cluster per garantire che i client possano continuare ad accedere al file system in caso di guasto di un nodo o se è necessario eseguire una manutenzione pianificata. In questa sezione vengono descritti i vari modi in cui gli amministratori possono riparare il cluster dopo il ripristino da un errore o spostare manualmente i servizi tra i nodi.

Fasi

Failover e failover

Failover (pianificato)

In genere, quando si deve portare un singolo nodo di file offline per la manutenzione, si desidera spostare (o svuotare) tutti i servizi BeeGFS da quel nodo. Per eseguire questa operazione, mettere il nodo in standby:

```
pcs node standby <HOSTNAME>
```

Dopo aver verificato l'utilizzo `pcs status` tutte le risorse sono state riavviate sul nodo di file alternativo, è possibile chiudere o apportare altre modifiche al nodo in base alle necessità.

Failback (dopo un failover pianificato)

Quando si è pronti a ripristinare i servizi BeeGFS sul nodo preferito, eseguire prima `pcs status` E verificare in "Node List" (elenco nodi) che lo stato sia standby. Se il nodo è stato riavviato, viene visualizzato offline fino a

quando non si mettono in linea i servizi del cluster:

```
pcs cluster start <HOSTNAME>
```

Una volta che il nodo è online, portarlo fuori dallo standby con:

```
pcs cluster node unstandby <HOSTNAME>
```

Infine, ricollocare tutti i servizi BeeGFS nei nodi preferiti con:

```
pcs resource relocate run
```

Failback (dopo un failover non pianificato)

Se un nodo presenta un guasto hardware o di altro tipo, il cluster ha dovrebbe reagire automaticamente e spostare i propri servizi su un nodo integro, fornendo tempo agli amministratori per intraprendere azioni correttive. Prima di procedere, fare riferimento a. "risoluzione dei problemi" sezione per determinare la causa del failover e risolvere eventuali problemi in sospeso. Una volta riacceso il nodo e funzionante, è possibile procedere con il failback.

Quando un nodo viene avviato in seguito a un riavvio non pianificato (o pianificato), i servizi cluster non vengono impostati per avviarsi automaticamente, quindi è necessario prima portare il nodo online con:

```
pcs cluster start <HOSTNAME>
```

Quindi, ripulire gli eventuali errori delle risorse e reimpostare la cronologia delle scherma del nodo:

```
pcs resource cleanup node=<HOSTNAME>
pcs stonith history cleanup <HOSTNAME>
```

Verificare in `pcs status` il nodo è online e integro. Per impostazione predefinita, i servizi BeeGFS non eseguono automaticamente il failback per evitare di spostare accidentalmente le risorse in un nodo non integro. Quando si è pronti, restituire tutte le risorse del cluster ai nodi preferiti con:

```
pcs resource relocate run
```

Spostamento di singoli servizi BeeGFS in nodi di file alternativi

Spostare in modo permanente un servizio BeeGFS in un nuovo nodo di file

Se si desidera modificare in modo permanente il nodo di file preferito per un singolo servizio BeeGFS, regolare l'inventario Ansible in modo che il nodo preferito venga elencato per primo ed eseguire nuovamente il playbook Ansible.

Ad esempio in questo esempio `inventory.yml` File, ictad22h01 è il nodo di file preferito per eseguire il servizio di gestione BeeGFS:

```
        mgmt:
          hosts:
            ictad22h01:
            ictad22h02:
```

L'inversione dell'ordine causerebbe la preferenza dei servizi di gestione su ictad22h02:

```
        mgmt:
          hosts:
            ictad22h02:
            ictad22h01:
```

Spostare temporaneamente un servizio BeeGFS in un nodo di file alternativo

In genere, se un nodo è in fase di manutenzione, utilizzare i [passi di failover e failback] (failover e failback) per spostare tutti i servizi da quel nodo.

Se per qualche motivo è necessario spostare un singolo servizio in un nodo di file diverso, eseguire:

```
pcs resource move <SERVICE>-monitor <HOSTNAME>
```

> Non specificare singole risorse o il gruppo di risorse. Specificare sempre il nome del monitor per il servizio BeeGFS che si desidera trasferire. Ad esempio, per spostare il servizio di gestione BeeGFS in ictad22h02 eseguire: `pcs resource move mgmt-monitor ictad22h02`. Questo processo può essere ripetuto per spostare uno o più servizi lontano dai nodi preferiti. Verificare l'utilizzo di `pcs status` i servizi sono stati ricollocati/avviati sul nuovo nodo.

Per spostare di nuovo un servizio BeeGFS nel nodo preferito, eliminare prima i vincoli di risorsa temporanei (ripetendo questa operazione in base alle necessità per più servizi):

```
pcs resource clear <SERVICE>-monitor
```

Quindi, quando si è pronti a spostare di nuovo i servizi sui nodi preferiti, eseguire:

```
pcs resource relocate run
```

Nota: Questo comando consente di spostare i servizi che non hanno più vincoli di risorse temporanee e che non si trovano nei nodi preferiti.

Impostare il cluster in modalità di manutenzione

Evitare che il cluster ha reagisca accidentalmente alle modifiche previste nell'ambiente.

Panoramica

Impostando il cluster in modalità di manutenzione si disattiva il monitoraggio di tutte le risorse e si impedisce a Pacemaker di spostare o gestire in altro modo le risorse nel cluster. Tutte le risorse rimarranno in esecuzione sui nodi originali, indipendentemente dalla presenza di una condizione di guasto temporanea che ne impedirebbe l'accesso. Gli scenari in cui questo è consigliato/utile includono:

- Manutenzione della rete che potrebbe interrompere temporaneamente le connessioni tra i nodi di file e i servizi BeeGFS.
- Aggiornamenti del nodo a blocchi.
- File Node per aggiornamenti di sistemi operativi, kernel o altri pacchetti.

In genere, l'unico motivo per attivare manualmente la modalità di manutenzione è impedire che il cluster reagisca alle modifiche esterne dell'ambiente. Se un singolo nodo del cluster richiede una riparazione fisica, non utilizzare la modalità di manutenzione e posizionare semplicemente tale nodo in standby seguendo la procedura descritta in precedenza. Si noti che la riesecuzione di Ansible attiva automaticamente la modalità di manutenzione del cluster, facilitando la maggior parte della manutenzione del software, inclusi aggiornamenti e modifiche alla configurazione.

Fasi

Per verificare se il cluster è in modalità di manutenzione, eseguire:

```
pcs property show maintenance-mode
```

Questo restituisce false quando il cluster funziona normalmente. Per attivare la modalità di manutenzione, eseguire:

```
pcs property set maintenance-mode=true
```

Puoi verificare eseguendo lo stato dei PC e assicurandoti che tutte le risorse mostrino "(unmanaged)". Per uscire dalla modalità di manutenzione del cluster, eseguire:

```
pcs property set maintenance-mode=false
```

Arrestare e avviare il cluster

Arresto e avvio del cluster ha senza problemi.

Panoramica

In questa sezione viene descritto come arrestare e riavviare il cluster BeeGFS. Esempi di scenari in cui ciò potrebbe essere necessario includono la manutenzione elettrica o la migrazione tra datacenter o rack.

Fasi

Se per qualsiasi motivo è necessario arrestare l'intero cluster BeeGFS e arrestare tutti i servizi eseguiti:

```
pcs cluster stop --all
```

È anche possibile arrestare il cluster su singoli nodi (che esegue automaticamente il failover dei servizi su un altro nodo), sebbene si consiglia di mettere il nodo in standby (vedere la "failover" sezione):

```
pcs cluster stop <HOSTNAME>
```

Per avviare i servizi e le risorse del cluster su tutti i nodi eseguire:

```
pcs cluster start --all
```

Oppure avviare i servizi su un nodo specifico con:

```
pcs cluster start <HOSTNAME>
```

A questo punto eseguire `pcs status` Verificare inoltre che i servizi del cluster e BeeGFS vengano avviati su tutti i nodi e che i servizi siano in esecuzione sui nodi previsti.

A seconda delle dimensioni del cluster, l'interruzione dell'intero cluster può richiedere alcuni minuti (da secondi a minuti) o la visualizzazione avviata in `pcs status`. Se `pcs cluster <COMMAND>` Si blocca per più di cinque minuti, prima di eseguire "Ctrl+C" per annullare il comando, accedere a ciascun nodo del cluster e utilizzare `pcs status` Per verificare se i servizi cluster (Corosync/Pacemaker) sono ancora in esecuzione su quel nodo. Da qualsiasi nodo in cui il cluster è ancora attivo, è possibile controllare quali risorse bloccano il cluster. Risolvere manualmente il problema e il comando dovrebbe essere completo o può essere rieseguito per interrompere eventuali servizi rimanenti.

Sostituire i nodi del file

Sostituzione di un nodo di file se il server originale è guasto.

Panoramica

Di seguito viene fornita una panoramica dei passaggi necessari per sostituire un nodo di file nel cluster. Questi passaggi presumono che il nodo del file non sia riuscito a causa di un problema hardware ed è stato sostituito con un nuovo nodo del file identico.

Fasi:

1. Sostituire fisicamente il nodo del file e ripristinare tutti i cavi al nodo a blocchi e alla rete di storage.
2. Reinstallare il sistema operativo sul nodo di file, aggiungendo anche le sottoscrizioni Red Hat.

3. Configurare la gestione e la rete BMC sul nodo file.
4. Aggiornare l'inventario di Ansible se il nome host, l'IP, le mappature dell'interfaccia PCIe-to-logical o qualsiasi altra cosa è stata modificata in relazione al nuovo nodo del file. In genere, questo non è necessario se il nodo è stato sostituito con un hardware server identico e si sta utilizzando la configurazione di rete originale.
   a. Ad esempio, se il nome host è cambiato, creare (o rinominare) il file di inventario del nodo (`host_vars/<NEW_NODE>.yml`) Quindi nel file di inventario Ansible (`inventory.yml`), sostituire il nome del vecchio nodo con il nuovo nome del nodo:

```
all:
    ...
    children:
    ha_cluster:
        children:
        mgmt:
            hosts:
            node_h1_new:    # Replaced "node_h1" with "node_h1_new"
            node_h2:
```

5. Rimuovere il nodo precedente da uno degli altri nodi del cluster: `pcs cluster node remove <HOSTNAME>`.

   NON PROCEDERE PRIMA DI ESEGUIRE QUESTO PASSAGGIO.

6. Sul nodo di controllo Ansible:
   a. Rimuovere la vecchia chiave SSH con:

```
`ssh-keygen -R <HOSTNAME_OR_IP>`
```

   b. Configurare SSH senza password nel nodo di sostituzione con:

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

7. Eseguire nuovamente il playbook Ansible per configurare il nodo e aggiungerlo al cluster:

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

8. A questo punto, eseguire `pcs status` e verificare che il nodo sostituito sia ora elencato e che i servizi siano in esecuzione.

Espandere o ridurre il cluster

Aggiungere o rimuovere i building block dal cluster.

Panoramica

Questa sezione descrive varie considerazioni e opzioni per regolare le dimensioni del cluster BeeGFS ha. In genere, la dimensione del cluster viene regolata aggiungendo o rimuovendo gli elementi di base, che in genere sono due nodi di file configurati come coppia ha. È inoltre possibile aggiungere o rimuovere singoli nodi di file (o altri tipi di nodi di cluster), se necessario.

Aggiunta di un building block al cluster

Considerazioni

La crescita del cluster mediante l'aggiunta di ulteriori building block è un processo semplice. Prima di iniziare, tenere presenti le restrizioni relative al numero minimo e massimo di nodi del cluster in ciascun cluster ha singolo e determinare se è necessario aggiungere nodi al cluster ha esistente o creare un nuovo cluster ha. In genere, ciascun building block è costituito da due nodi di file, ma tre nodi sono il numero minimo di nodi per cluster (per stabilire il quorum) e dieci sono il numero massimo consigliato (testato). Per gli scenari avanzati è possibile aggiungere un singolo nodo "Tiebreaker" che non esegue alcun servizio BeeGFS durante l'implementazione di un cluster a due nodi. Contatta il supporto NetApp se stai prendendo in considerazione un'implementazione di questo tipo.

Quando si decide come espandere il cluster, tenere presente queste restrizioni e qualsiasi crescita futura prevista del cluster. Ad esempio, se si dispone di un cluster a sei nodi e si desidera aggiungere altri quattro nodi, si consiglia di avviare un nuovo cluster ha.

Tenere presente che un singolo file system BeeGFS può essere costituito da più cluster ha indipendenti. Ciò consente ai file system di continuare a scalare oltre i limiti consigliati/rigidi dei componenti del cluster ha sottostanti.

Fasi

Quando si aggiunge un building block al cluster, è necessario creare `host_vars` File per ciascuno dei nuovi nodi di file e nodi di blocco (array e-Series). I nomi di questi host devono essere aggiunti all'inventario, insieme alle nuove risorse da creare. Il corrispondente `group_vars` i file devono essere creati per ogni nuova risorsa. Vedere "Utilizzare architetture personalizzate" per ulteriori informazioni.

Dopo aver creato i file corretti, è sufficiente eseguire nuovamente l'automazione utilizzando il comando:

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

Rimozione di un Building Block dal cluster

È necessario tenere presente una serie di considerazioni quando è necessario dismettere un building block, ad esempio:

- Quali servizi BeeGFS vengono eseguiti in questo building block?
- I nodi di file vengono ritirati e i nodi di blocco devono essere collegati ai nuovi nodi di file?
- Se l'intero building block viene ritirato, i dati devono essere spostati in un nuovo building block, dispersi in nodi esistenti nel cluster o spostati in un nuovo file system BeeGFS o in un altro sistema storage?
- Questo può accadere durante un'interruzione o dovrebbe essere fatto senza interruzioni?
- Il building block è attivamente in uso o contiene principalmente dati che non sono più attivi?

A causa dei diversi possibili punti di partenza e degli stati finali desiderati, contatta il supporto NetApp in modo da poter identificare e implementare la strategia migliore in base al tuo ambiente e ai tuoi requisiti.

Risolvere i problemi

Risoluzione dei problemi di un cluster BeeGFS ha.

Panoramica

Questa sezione illustra come analizzare e risolvere i problemi di vari guasti e altri scenari che potrebbero verificarsi quando si utilizza un cluster BeeGFS ha.

Guide per la risoluzione dei problemi

Analisi di guasti imprevisti

Quando un nodo viene inaspettatamente recintato e i relativi servizi vengono spostati in un altro nodo, il primo passo dovrebbe essere vedere se il cluster indica eventuali guasti alle risorse nella parte inferiore di `pcs status`. In genere, se la scherma è stata completata correttamente e le risorse sono state riavviate su un altro nodo, non sarà presente nulla.

In genere, il passaggio successivo consiste nell'eseguire una ricerca nei log di sistema utilizzando `journalctl` Su uno qualsiasi dei nodi di file rimanenti (i registri di pacemaker sono sincronizzati su tutti i nodi). Se si conosce l'ora in cui si è verificato l'errore, è possibile avviare la ricerca poco prima che si sia verificato l'errore (generalmente si consiglia di almeno dieci minuti prima):

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>"
```

Le sezioni seguenti mostrano il testo comune che è possibile inserire nei registri per restringere ulteriormente l'analisi.

Procedure per investigare/risolvere

Fase 1: Controllare se il monitor BeeGFS ha rilevato un guasto:

Se il failover è stato attivato dal monitor BeeGFS, viene visualizzato un errore (in caso contrario, passare alla fase successiva).

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i unexpected
[...]
Jul 01 15:51:03 ictad22h01 pacemaker-schedulerd[9246]:  warning:
Unexpected result (error: BeeGFS service is not active!) was recorded for
monitor of meta_08-monitor on ictad22h02 at Jul  1 15:51:03 2022
```

In questo caso, il servizio BeeGFS meta_08 si è arrestato per qualche motivo. Per continuare con la risoluzione dei problemi, è necessario avviare ictad22h02 ed esaminare i registri per il servizio all'indirizzo `/var/log/beegfs-meta-meta_08_tgt_0801.log`. Ad esempio, il servizio BeeGFS potrebbe aver riscontrato un errore dell'applicazione a causa di un problema interno o del nodo.

A differenza dei registri di Pacemaker, i registri dei servizi BeeGFS non vengono distribuiti a tutti i nodi del cluster. Per analizzare questi tipi di errori, sono necessari i log del nodo originale in cui si è verificato l'errore.

I possibili problemi che potrebbero essere segnalati dal monitor includono:

- Destinazioni non accessibili.
  - Descrizione: Indica che i volumi a blocchi non erano accessibili.
  - Risoluzione dei problemi:
    - Se anche il servizio non è stato avviato sul nodo di file alternativo, verificare che il nodo di blocco sia integro.
    - Verificare l'eventuale presenza di problemi fisici che impediscano l'accesso ai nodi di blocco da questo nodo di file, ad esempio adattatori o cavi InfiniBand difettosi.
- Rete non raggiungibile.
  - Descrizione: Nessuno degli adattatori utilizzati dai client per connettersi a questo servizio BeeGFS era in linea.
  - Risoluzione dei problemi:
    - In caso di impatto su più/tutti i nodi di file, controllare se si è verificato un errore nella rete utilizzata per collegare i client BeeGFS e il file system.
    - Verificare l'eventuale presenza di problemi fisici che impediscano l'accesso ai client da questo nodo di file, ad esempio cavi o adattatori InfiniBand difettosi.
- Servizio BeeGFS non attivo.
  - Descrizione: Un servizio BeeGFS si è arrestato inaspettatamente.
  - Risoluzione dei problemi:
    - Nel nodo del file che ha riportato l'errore, controllare i log del servizio BeeGFS interessato per verificare se ha rilevato un blocco. In questo caso, aprire un caso con il supporto NetApp per poter indagare sul crash.
    - Se non vengono segnalati errori nel log di BeeGFS, controllare i log del journal per verificare se systemd ha registrato un motivo per cui il servizio è stato arrestato. In alcuni scenari, il servizio BeeGFS potrebbe non aver avuto la possibilità di registrare alcun messaggio prima che il processo venisse terminato (ad esempio, se qualcuno ha eseguito `kill -9 <PID>`).

Fase 2: Controllare se il nodo ha lasciato inaspettatamente il cluster

Nel caso in cui il nodo abbia subito un guasto hardware catastrofico (ad esempio, la scheda di sistema è morta) o si sia verificato un problema di kernel panic o software simile, il monitor BeeGFS non segnala alcun errore. Cercare invece il nome host e dovrebbero essere visualizzati messaggi da Pacemaker che indicano che il nodo è stato perso inaspettatamente:

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i <HOSTNAME>
[...]
Jul 01 16:18:01 ictad22h01 pacemaker-attrd[9245]:  notice: Node ictad22h02
state is now lost
Jul 01 16:18:01 ictad22h01 pacemaker-controld[9247]:  warning:
Stonith/shutdown of node ictad22h02 was not expected
```

Fase 3: Verificare che il pacemaker sia in grado di individuare il nodo

In tutti gli scenari si dovrebbe vedere il tentativo di pacemaker di recinzione del nodo per verificare che sia effettivamente offline (i messaggi esatti possono variare a seconda della causa del recinzione):

```
Jul 01 16:18:02 ictad22h01 pacemaker-schedulerd[9246]:  warning: Cluster
node ictad22h02 will be fenced: peer is no longer part of the cluster
Jul 01 16:18:02 ictad22h01 pacemaker-schedulerd[9246]:  warning: Node
ictad22h02 is unclean
Jul 01 16:18:02 ictad22h01 pacemaker-schedulerd[9246]:  warning:
Scheduling Node ictad22h02 for STONITH
```

Se l'azione di scherma viene completata correttamente, vengono visualizzati messaggi come:

```
Jul 01 16:18:14 ictad22h01 pacemaker-fenced[9243]:  notice: Operation
'off' [2214070] (call 27 from pacemaker-controld.9247) for host
'ictad22h02' with device 'fence_redfish_2' returned: 0 (OK)
Jul 01 16:18:14 ictad22h01 pacemaker-fenced[9243]:  notice: Operation
'off' targeting ictad22h02 on ictad22h01 for pacemaker-
controld.9247@ictad22h01.786df3a1: OK
Jul 01 16:18:14 ictad22h01 pacemaker-controld[9247]:  notice: Peer
ictad22h02 was terminated (off) by ictad22h01 on behalf of pacemaker-
controld.9247: OK
```

Se l'azione di scherma non è riuscita per qualche motivo, i servizi BeeGFS non potranno essere riavviati su un altro nodo per evitare il rischio di corruzione dei dati. Si tratta di un problema da analizzare separatamente, ad esempio se il dispositivo di scherma (PDU o BMC) non fosse accessibile o non fosse configurato correttamente.

Address Failed Resource Actions (azioni risorsa indirizzo non riuscito) (trovato in fondo allo stato di pcs)

Se una risorsa richiesta per eseguire un servizio BeeGFS non riesce, il monitor BeeGFS attiva un failover. In questo caso, probabilmente non sarà presente alcuna "azione risorsa non riuscita" nella parte inferiore di `pcs status` fare riferimento alla procedura "failback dopo un failover non pianificato".

In caso contrario, dovrebbero essere presenti solo due scenari in cui verranno visualizzate le "azioni delle risorse non riuscite".

Procedure per investigare/risolvere

Scenario 1: È stato rilevato un problema temporaneo o permanente con un agente di scherma che è stato riavviato o spostato in un altro nodo.

Alcuni agenti di scherma sono più affidabili di altri e ciascuno implementerà il proprio metodo di monitoraggio per garantire che il dispositivo di scherma sia pronto. In particolare, l'agente Redfish scherma ha rilevato azioni di risorse non riuscite come le seguenti, anche se continuerà a mostrare avviato:

```
  * fence_redfish_2_monitor_60000 on ictad22h01 'not running' (7):
call=2248, status='complete', exitreason='', last-rc-change='2022-07-26
08:12:59 -05:00', queued=0ms, exec=0ms
```

Un agente di scherma che segnala azioni di risorse non riuscite su un nodo particolare non dovrebbe attivare un failover dei servizi BeeGFS in esecuzione su quel nodo. Dovrebbe semplicemente essere riavviato automaticamente sullo stesso nodo o su un altro nodo.

Procedura per la risoluzione:

1. Se l'agente di scherma rifiuta costantemente di essere eseguito su tutti i nodi o su un sottoinsieme di nodi, controllare se tali nodi sono in grado di connettersi all'agente di scherma e verificare che l'agente di scherma sia configurato correttamente nell'inventario Ansible.
   a. Ad esempio, se un agente di scherma Redfish (BMC) è in esecuzione sullo stesso nodo in cui è responsabile della scherma e la gestione del sistema operativo e gli IP BMC si trovano sulla stessa interfaccia fisica, alcune configurazioni dello switch di rete non consentono la comunicazione tra le due interfacce (per evitare loop di rete). Per impostazione predefinita, il cluster ha tenterà di evitare di posizionare gli agenti di scherma sul nodo che sono responsabili della scherma, ma questo può accadere in alcuni scenari/configurazioni.
2. Una volta risolti tutti i problemi (o se il problema sembrava essere effimero), eseguire `pcs resource cleanup` per ripristinare le azioni delle risorse non riuscite.

Scenario 2: Il monitor BeeGFS ha rilevato un problema e ha attivato un failover, ma per qualche motivo le risorse non sono state avviate su un nodo secondario.

A condizione che sia attivata la funzione di scherma e che la risorsa non sia stata bloccata dall'arresto sul nodo originale (vedere la sezione relativa alla risoluzione dei problemi per "standby (on-fail)"), i motivi più probabili includono problemi di avvio della risorsa su un nodo secondario perché:

- Il nodo secondario era già offline.
- Un problema di configurazione fisica o logica ha impedito al secondario di accedere ai volumi di blocco utilizzati come destinazioni BeeGFS.

Procedura per la risoluzione:

1. Per ogni voce nelle azioni delle risorse non riuscite:
   a. Confermare che l'azione della risorsa non riuscita era un'operazione di avvio.
   b. In base alla risorsa indicata e al nodo specificato nelle azioni delle risorse non riuscite:
      i. Cercare e correggere eventuali problemi esterni che impediscano al nodo di avviare la risorsa specificata. Ad esempio, se l'indirizzo IP BeeGFS (floating IP) non si avvia, verificare che almeno una delle interfacce richieste sia connessa/online e cablata allo switch di rete corretto. Se una

destinazione BeeGFS (dispositivo a blocchi / volume e-Series) non funziona, verificare che le connessioni fisiche ai nodi di blocco back-end siano collegate come previsto e verificare che i nodi di blocco siano integri.

c. Se non ci sono problemi esterni evidenti e si desidera una causa principale per questo incidente, si consiglia di aprire un caso con il supporto NetApp per indagare prima di procedere, in quanto i seguenti passaggi potrebbero rendere difficile/impossibile l'analisi della causa principale (RCA).

2. Dopo aver risolto eventuali problemi esterni:

   a. Commentare eventuali nodi non funzionali dal file Ansible inventory.yml ed eseguire nuovamente il playbook Ansible completo per assicurarsi che tutte le configurazioni logiche siano configurate correttamente sui nodi secondari.

      i. Nota: Non dimenticare di rimuovere il commento da questi nodi e di eseguire nuovamente il playbook una volta che i nodi sono in buono stato e sei pronto per il failback.

   b. In alternativa, è possibile tentare di ripristinare manualmente il cluster:

      i. Posizionare di nuovo online i nodi offline utilizzando: `pcs cluster start <HOSTNAME>`

      ii. Cancellare tutte le azioni delle risorse non riuscite utilizzando: `pcs resource cleanup`

      iii. Eseguire lo stato dei PC e verificare che tutti i servizi inizano come previsto.

      iv. Se necessario, eseguire `pcs resource relocate run` per spostare nuovamente le risorse nel nodo preferito (se disponibile).

Problemi comuni

I servizi BeeGFS non eseguono il failover o il failback quando richiesto

Probabile problema: il `pcs resource relocate` il comando run è stato eseguito, ma non è mai stato completato correttamente.

Come controllare: Esegui `pcs constraint --full` E verificare la presenza di eventuali vincoli di posizione con un ID di `pcs-relocate-<RESOURCE>`.

Come risolvere: Esegui `pcs resource relocate clear` quindi rieseguire `pcs constraint --full` per verificare che i vincoli aggiuntivi vengano rimossi.

Un nodo nello stato di PC mostra "standby (on-fail)" quando la scherma è disattivata

Probabile problema: pacemaker non è riuscito a confermare che tutte le risorse sono state interrotte sul nodo che ha avuto esito negativo.

Come risolvere:

1. Eseguire `pcs status` e verificare la presenza di risorse che non sono "avviate" o che mostrano errori nella parte inferiore dell'output e risolvere eventuali problemi.
2. Per riportare il nodo in linea eseguire `pcs resource cleanup --node=<HOSTNAME>`.

Dopo un failover imprevisto, le risorse mostrano "Started (on-fail)" (avviato (on-fail)) in stato PC quando la scherma è attivata

Probabile problema: si è verificato Un problema che ha attivato un failover, ma Pacemaker non è riuscito a verificare che il nodo sia stato recintato. Questo potrebbe verificarsi a causa di una configurazione errata del recinto o di un problema con l'agente di recinzione (ad esempio: La PDU è stata disconnessa dalla rete).

Come risolvere:

1. Verificare che il nodo sia effettivamente spento.

> Se il nodo specificato non è effettivamente disattivato, ma esegue risorse o servizi cluster, si VERIFICHERÀ un danneggiamento dei dati o un errore del cluster.

2. Confermare manualmente la scherma con: `pcs stonith confirm <NODE>`

A questo punto i servizi dovrebbero terminare il failover e essere riavviati su un altro nodo integro.

Attività comuni di risoluzione dei problemi

Riavviare i singoli servizi BeeGFS

In genere, se un servizio BeeGFS deve essere riavviato (ad esempio per facilitare una modifica della configurazione), questa operazione deve essere eseguita aggiornando l'inventario Ansible e rieseguendo il manuale. In alcuni scenari potrebbe essere consigliabile riavviare singoli servizi per facilitare la risoluzione dei problemi più rapida, ad esempio per modificare il livello di registrazione senza dover attendere l'esecuzione dell'intero playbook.

> A meno che non vengano aggiunte modifiche manuali all'inventario Ansible, queste verranno ripristinate alla prossima esecuzione del playbook Ansible.

Opzione 1: Riavvio controllato dal sistema

Se esiste il rischio che il servizio BeeGFS non si riavvii correttamente con la nuova configurazione, impostare innanzitutto il cluster in modalità di manutenzione per evitare che il monitor BeeGFS rilevi che il servizio è stato arrestato e che venga attivato un failover indesiderato:

```
pcs property set maintenance-mode=true
```

Se necessario, apportare eventuali modifiche alla configurazione dei servizi all'indirizzo `/mnt/<SERVICE_ID>/_config/beegfs-.conf` (esempio: `/mnt/meta_01_tgt_0101/metadata_config/beegfs-meta.conf`) quindi utilizzare systemd per riavviarlo:

```
systemctl restart beegfs-*@<SERVICE_ID>.service
```

Esempio: `systemctl restart beegfs-meta@meta_01_tgt_0101.service`

Opzione 2: Riavvio controllato da pacemaker

Se non si è preoccupati per la nuova configurazione, il servizio potrebbe arrestarsi in modo imprevisto (ad esempio, semplicemente cambiando il livello di registrazione) oppure ci si trova in una finestra di manutenzione e non si è preoccupati per i tempi di inattività, è sufficiente riavviare il monitor BeeGFS per il servizio che si desidera riavviare:

```
pcs resource restart <SERVICE>-monitor
```

Ad esempio, per riavviare il servizio di gestione BeeGFS: `pcs resource restart mgmt-monitor`

Note legali

Le note legali forniscono l'accesso a dichiarazioni di copyright, marchi, brevetti e altro ancora.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marchi

NETAPP, il logo NETAPP e i marchi elencati nella pagina dei marchi NetApp sono marchi di NetApp, Inc. Altri nomi di società e prodotti potrebbero essere marchi dei rispettivi proprietari.

"https://www.netapp.com/company/legal/trademarks/"

Brevetti

Un elenco aggiornato dei brevetti di proprietà di NetApp è disponibile all'indirizzo:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Direttiva sulla privacy

"https://www.netapp.com/company/legal/privacy-policy/"

Open source

I file di avviso forniscono informazioni sul copyright e sulle licenze di terze parti utilizzate nel software NetApp.

"Avviso per i sistemi operativi SANtricity e-Series/EF-Series"

Informazioni sul copyright

Copyright © 2024 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEQUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina http://www.netapp.com/TM sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.