Implementare la soluzione

BeeGFS on NetApp with E-Series Storage

NetApp
June 18, 2024

Sommario

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

Nodo del file	Porta IB	Indirizzo IP	Nodo del blocco	Porta IB	IP fisico	IP virtuale
Dispari (h1)	i4a	192.168.7.10	Pari (c2)	2a	192.168.7.100	192.168.7.101
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".

Informazioni sul copyright

Informazioni sul marchio commerciale