BeeGFS no NetApp com storage e-Series

BeeGFS on NetApp with E-Series Storage

NetApp
October 22, 2024

Índice

BeeGFS no NetApp com storage e-Series

Comece agora

O que está incluído neste site

Este site documenta como implantar e gerenciar BeeGFS no NetApp usando arquiteturas verificadas (NVAs) do NetApp e arquiteturas personalizadas. Os designs da NVA são completamente testados e fornecem aos clientes configurações de referência e orientações de dimensionamento para minimizar o risco de implantação e acelerar o time-to-market. O NetApp também é compatível com arquiteturas BeeGFS personalizadas executadas no hardware da NetApp, oferecendo aos clientes e parceiros flexibilidade no design de sistemas de arquivos para atender a uma ampla gama de requisitos. Ambas as abordagens utilizam o Ansible para implantação, fornecendo uma abordagem semelhante a um dispositivo que gerencia o BeeGFS em qualquer escala em uma variedade flexível de hardware.

Termos e conceitos

Os termos e conceitos a seguir se aplicam à solução BeeGFS on NetApp.

Consulte "Administrar clusters BeeGFS"a seção para obter detalhes adicionais sobre termos e conceitos específicos para interagir com clusters de alta disponibilidade (HA) do BeeGFS.

Prazo	Descrição
AI	Inteligência artificial.
Inventário do Ansible	Estrutura de diretório contendo arquivos YAML que são usados para descrever o cluster BeeGFS HA desejado.
BMC	Controlador de gerenciamento de placa de base. Às vezes referido como um processador de serviço.
nós de bloco	Sistemas de storage.
clientes	Nós no cluster HPC que executam aplicativos que precisam utilizar o sistema de arquivos. Às vezes também conhecido como nós de computação ou GPU.
DL	Aprendizagem profunda.
nós de arquivo	Servidores de arquivos BeeGFS.
HA	Alta disponibilidade.
HIC	Placa de interface do host.

Prazo	Descrição
HPC	Computação de alta performance.
Workloads em estilo HPC	Os workloads no estilo HPC são normalmente caraterizados por vários nós de computação ou GPUs que precisam acessar o mesmo conjunto de dados em paralelo para facilitar um trabalho de computação ou treinamento distribuído. Esses conjuntos de dados costumam ser compostos por arquivos grandes que devem ser distribuídos por vários nós de storage físico para eliminar os gargalos tradicionais de hardware que impediriam o acesso simultâneo a um único arquivo.
ML	Aprendizado de máquina.
PNL	Processamento de linguagem natural.
NLU	Compreensão da linguagem natural.
NVA	O programa NetApp Verified Architecture (NVA) fornece configurações de referência e orientações de dimensionamento para workloads e casos de uso específicos. Essas soluções são completamente testadas e foram projetadas para minimizar os riscos de implantação e acelerar o time-to-market.
rede de armazenamento / rede cliente	Rede usada para que os clientes se comuniquem com o sistema de arquivos BeeGFS. Esta é frequentemente a mesma rede usada para a Interface de passagem de mensagens paralela (MPI) e outra comunicação de aplicativos entre nós de cluster HPC.

Use arquiteturas verificadas

Visão geral e requisitos

Visão geral da solução

A solução BeeGFS on NetApp combina o sistema de arquivos paralelos do BeeGFS com os sistemas de storage NetApp EF600 para uma infraestrutura confiável, dimensionável e econômica que acompanha os workloads exigentes.

Programa NVA

A solução BeeGFS on NetApp faz parte do programa NetApp Verified Architecture (NVA), que fornece aos clientes configurações de referência e orientações de dimensionamento para workloads e casos de uso específicos. As soluções NVA são completamente testadas e projetadas para minimizar os riscos de implantação e acelerar o time-to-market.

Visão geral do design

A solução BeeGFS on NetApp foi projetada como uma arquitetura de componentes básicos dimensionável, configurável para uma variedade de workloads exigentes. Seja lidando com muitos arquivos pequenos, gerenciando grandes operações de arquivos ou uma carga de trabalho híbrida, o sistema de arquivos pode ser personalizado para atender a essas necessidades. A alta disponibilidade é incorporada ao design com o uso de uma estrutura de hardware de duas camadas que permite failover independente em várias camadas de hardware e garante desempenho consistente, mesmo durante degradações parciais do sistema. O sistema de arquivos BeeGFS oferece um ambiente dimensionável e de alta performance em diferentes distribuições do Linux. Além disso, apresenta aos clientes um namespace de storage único de fácil acesso. Saiba mais no "visão geral da arquitetura".

Casos de uso

Os seguintes casos de uso se aplicam à solução BeeGFS no NetApp:

- Os sistemas NVIDIA DGX SuperPOD apresentam DGX com GPU A100, H100, H200 e B200.
- Inteligência artificial (AI), incluindo aprendizado de máquina (ML), aprendizado profundo (DL), processamento de linguagem natural em larga escala (PNL) e compreensão de linguagem natural (NLU). Para obter mais informações, "BeeGFS para IA: Fato versus ficção"consulte .
- Computação de alto desempenho (HPC), incluindo aplicativos acelerados por MPI (interface de passagem de mensagens) e outras técnicas de computação distribuída. Para obter mais informações, "Por que BeeGFS vai além da HPC"consulte .
- Workloads de aplicação caraterizados por:
  - Leitura ou escrita em arquivos maiores que 1GB
  - Leitura ou escrita no mesmo arquivo por vários clientes (10s, 100s e 1000s)
- Conjuntos de dados com vários terabytes ou multipetabytes.
- Ambientes que precisam de um único namespace de armazenamento otimizado para uma combinação de arquivos grandes e pequenos.

Benefícios

Os principais benefícios do uso do BeeGFS no NetApp incluem:

- Disponibilidade de designs de hardware verificados que fornecem integração total de componentes de hardware e software para garantir desempenho e confiabilidade previsíveis.
- Implantação e gerenciamento com o Ansible para oferecer simplicidade e consistência em escala.
- Monitoramento e observabilidade fornecidos com o e-Series Performance Analyzer e o plug-in BeeGFS. Para obter mais informações, "Apresentando uma estrutura para monitorar as soluções NetApp e-Series"consulte .
- Alta disponibilidade com uma arquitetura de disco compartilhado que fornece durabilidade e disponibilidade de dados.
- Suporte para gerenciamento e orquestração modernos de workloads usando contêineres e Kubernetes. Para obter mais informações, "Conheça o BeeGFS: Uma história dos investimentos prontos para o futuro"consulte .

Projete gerações

A solução BeeGFS on NetApp está atualmente em seu segundo projeto geracional.

A primeira e a segunda geração incluem uma arquitetura básica que incorpora um sistema de arquivos BeeGFS e um sistema de storage NVMe EF600. No entanto, a segunda geração baseia-se na primeira a incluir estes benefícios adicionais:

- Duplique o desempenho e a capacidade, adicionando apenas 2U GB de espaço em rack
- Alta disponibilidade (HA) com base em um design de hardware de duas camadas de disco compartilhado
- Arquitetura projetada para os sistemas NVIDIA DGX SuperPOD A100, H100, H200 e B200, que foi validada anteriormente em um cluster de aceitação dedicado na NVIDIA. Leia mais sobre o NVIDIA DGX SuperPOD com a NetApp no "guia de design".

Segundo design geracional

A segunda geração do BeeGFS no NetApp é otimizada para atender aos requisitos de performance de workloads exigentes, incluindo computação de alta performance (HPC), aprendizado de máquina (ML), deep learning (DL) e outras técnicas de inteligência artificial (AI). Ao incorporar uma arquitetura de alta disponibilidade (HA) de disco compartilhado, esse design garante a durabilidade e a disponibilidade dos dados, tornando-o ideal para empresas e outras organizações que não podem pagar tempo de inatividade ou perda de dados. O projeto de segunda geração inclui componentes como servidores PCIe Gen5 e suporte para switches InfiniBand NVIDIA Quantum QM9700 400GBGb/s. Essa solução não só foi verificada pela NetApp, mas também passou na qualificação externa como uma opção de storage para o SuperPOD do NVIDIA DGX A100, com certificação estendida para sistemas DGX SuperPOD H100, H200 e B200.

Primeiro design geracional

A primeira geração de BeeGFS no NetApp foi projetada para workloads de aprendizado de máquina (ML) e inteligência artificial (AI) usando os sistemas de storage NVMe NetApp EF600, o sistema de arquivos paralelos BeeGFS, os sistemas NVIDIA A100 e os switches IB NVIDIA Mellanox Quantum QM8700 200GBGb/s. Esse design também oferece InfiniBand (IB) de 200GB GB/s para a malha de interconexão de cluster de computação e storage para fornecer uma arquitetura totalmente baseada em IB para workloads de alto desempenho.

Para obter mais informações sobre a primeira geração, "NetApp EF-Series AI com os sistemas DGX A100 e

BeeGFS da NVIDIA"consulte .

Visão geral da arquitetura

A solução BeeGFS on NetApp inclui considerações de design de arquitetura usadas para determinar o equipamento, o cabeamento e as configurações específicos necessários para dar suporte a workloads validados.

Arquitetura de componentes básicos

O sistema de arquivos BeeGFS pode ser implantado e dimensionado de diferentes maneiras, dependendo dos requisitos de storage. Por exemplo, os casos de uso que apresentam principalmente vários arquivos pequenos se beneficiarão do desempenho e capacidade extra dos metadados, enquanto os casos de uso com menos arquivos grandes podem favorecer mais capacidade de armazenamento e desempenho para o conteúdo real dos arquivos. Essas várias considerações afetam diferentes dimensões da implantação do sistema de arquivos paralelo, o que aumenta a complexidade ao projetar e implantar o sistema de arquivos.

Para lidar com esses desafios, a NetApp projetou uma arquitetura padrão de componentes básicos usada para dimensionar cada uma dessas dimensões. Normalmente, os componentes básicos do BeeGFS são implantados em um de três perfis de configuração:

- Um componente básico único, incluindo gerenciamento, metadados e serviços de storage do BeeGFS
- Metadados do BeeGFS, além de componente básico de storage
- Um componente básico de storage do BeeGFS

A única alteração de hardware entre essas três opções é o uso de unidades menores para metadados do BeeGFS. Caso contrário, todas as alterações de configuração são aplicadas através do software. E, com o Ansible como mecanismo de implantação, a configuração do perfil desejado para um componente básico específico simplifica as tarefas de configuração.

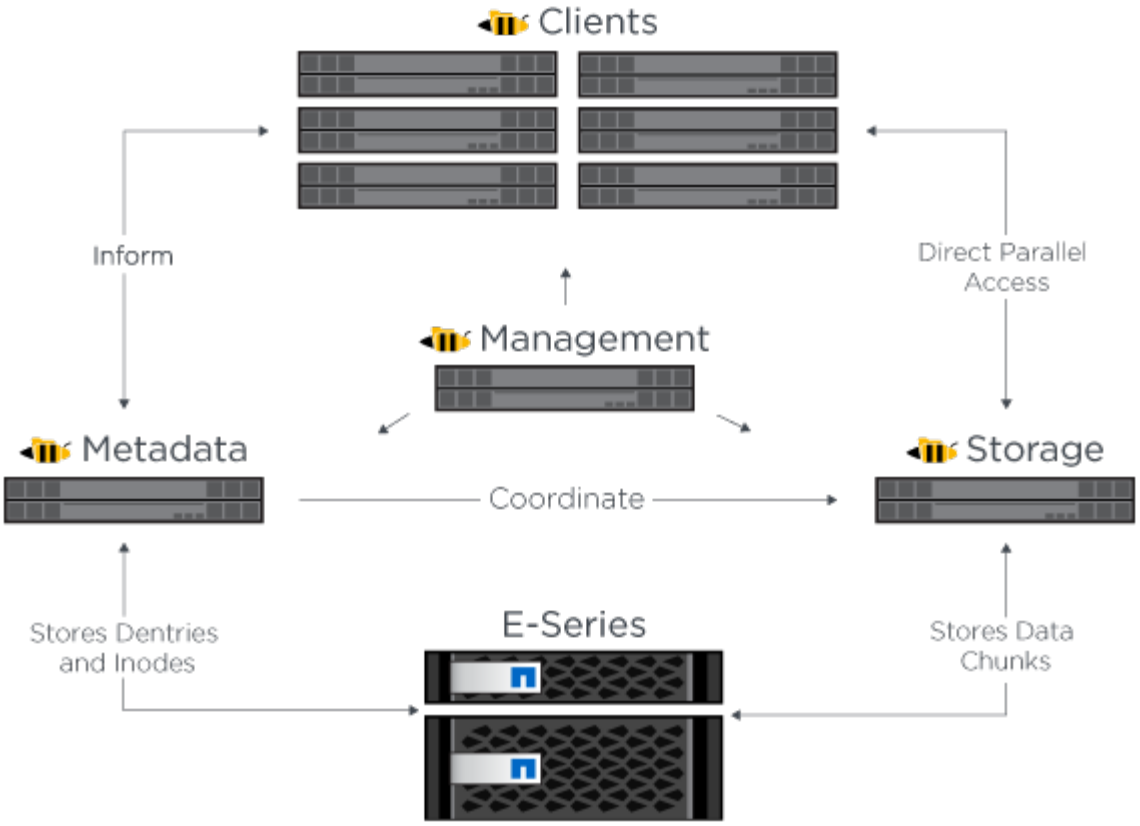
Para obter mais detalhes, Design de hardware verificadoconsulte .

Serviços de sistema de arquivos

O sistema de arquivos BeeGFS inclui os seguintes serviços principais:

- **Serviço de gestão.** Registra e monitora todos os outros serviços.
- **Serviço de armazenamento.** Armazena o conteúdo do arquivo de usuário distribuído conhecido como arquivos de bloco de dados.
- **Serviço de metadados.** Mantém o controle do layout do sistema de arquivos, diretório, atributos de arquivo e assim por diante.
- **Serviço de atendimento ao cliente.** Monta o sistema de arquivos para acessar os dados armazenados.

A figura a seguir mostra os componentes e as relações da solução BeeGFS usadas com os sistemas NetApp e-Series.



Como um sistema de arquivos paralelo, o BeeGFS distribui seus arquivos em vários nós de servidor para maximizar a performance de leitura/gravação e a escalabilidade. Os nós de servidor trabalham juntos para fornecer um único sistema de arquivos que pode ser simultaneamente montado e acessado por outros nós de servidor, comumente conhecidos como *clients*. Esses clientes podem ver e consumir o sistema de arquivos distribuídos da mesma forma que um sistema de arquivos local, como NTFS, XFS ou ext4.

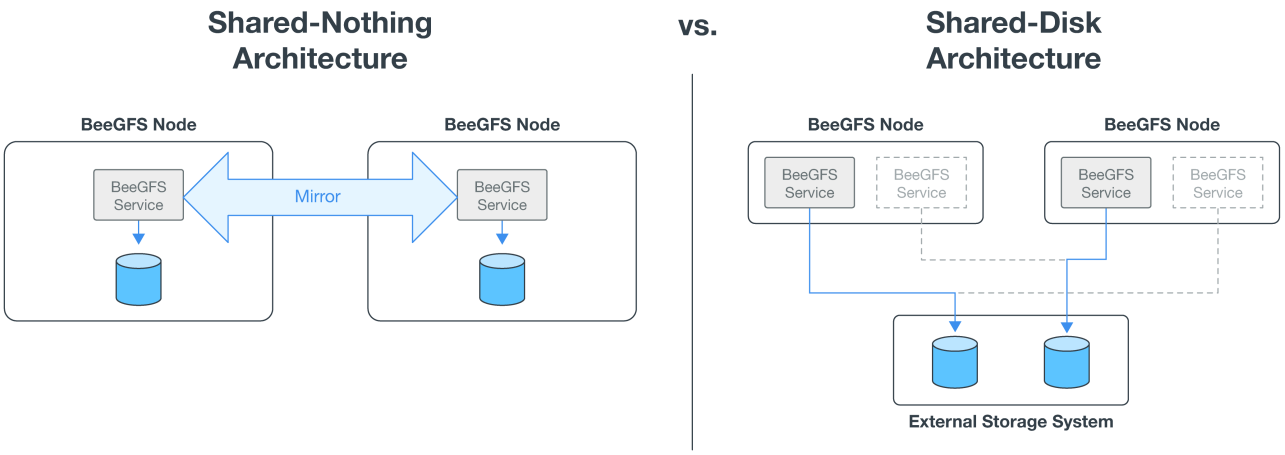
Os quatro principais serviços são executados em uma ampla variedade de distribuições Linux suportadas e se comunicam por qualquer rede compatível com TCP/IP ou RDMA, incluindo InfiniBand (IB), Omni-Path (OPA) e RDMA sobre Ethernet convergente (RoCE). Os serviços de servidor BeeGFS (gerenciamento, storage e metadados) são daemons de espaço do usuário, enquanto o cliente é um módulo de kernel nativo (sem patchless). Todos os componentes podem ser instalados ou atualizados sem reinicialização, e você pode executar qualquer combinação de serviços no mesmo nó.

Arquitetura HA

O BeeGFS no NetApp expande a funcionalidade da edição empresarial BeeGFS ao criar uma solução totalmente integrada com o hardware da NetApp que habilita uma arquitetura de alta disponibilidade (HA) de disco compartilhado.

> Embora a edição da comunidade BeeGFS possa ser usada gratuitamente, a edição empresarial exige a compra de um contrato de assinatura de suporte profissional de um parceiro como a NetApp. A edição corporativa permite o uso de vários recursos adicionais, incluindo resiliência, imposição de cotas e pools de armazenamento.

A figura a seguir compara as arquiteturas de HA de disco compartilhado e de disco compartilhado.



Para obter mais informações, "Anúncio de alta disponibilidade para o BeeGFS com suporte da NetApp"consulte .

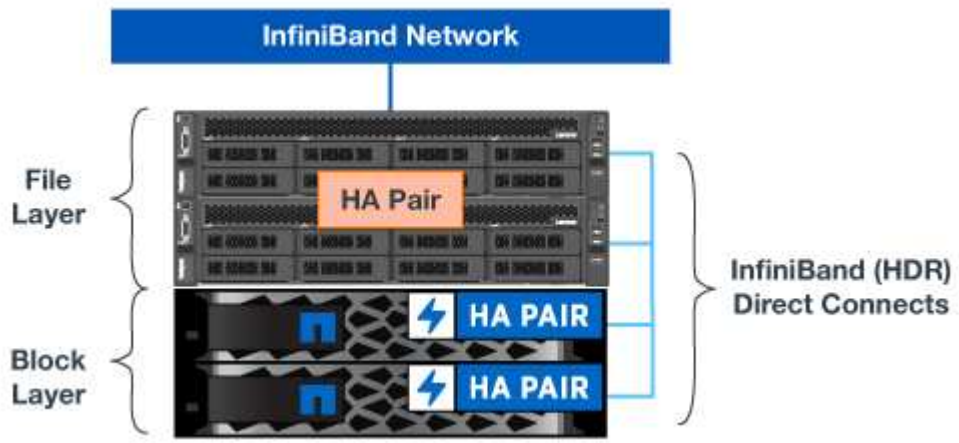
Nós verificados

A solução BeeGFS on NetApp verificou os nós listados abaixo.

Nó	Hardware	Detalhes
Bloco	Sistema de storage NetApp EF600	Um storage array totalmente NVMe 2U de alta performance desenvolvido para workloads exigentes.
Ficheiro	Servidor Lenovo ThinkSystem SR665 V3	Um servidor 2U de dois soquetes com PCIe 5,0, dois processadores AMD EPYC 9124. Para obter mais informações sobre o Lenovo SR665 V3, "Website da Lenovo" consulte .
	Servidor Lenovo ThinkSystem SR665	Um servidor 2U de dois soquetes com PCIe 4,0, dois processadores AMD EPYC 7003. Para obter mais informações sobre o Lenovo SR665, "Website da Lenovo" consulte .

Design de hardware verificado

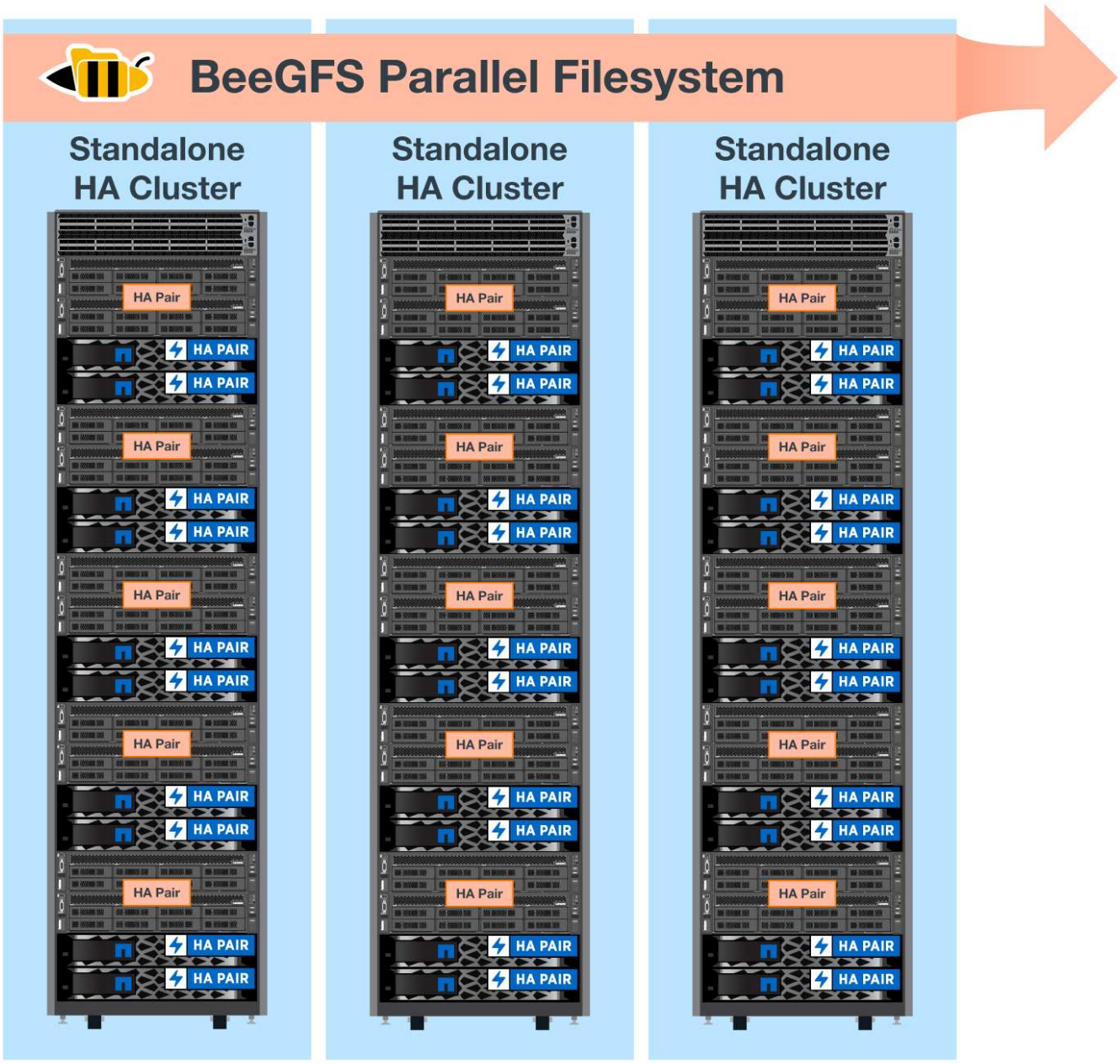
Os componentes básicos da solução (mostrados na figura a seguir) usam os servidores de nós de arquivo verificados para a camada de arquivo BeeGFS e dois sistemas de storage EF600 como a camada de bloco.



A solução BeeGFS on NetApp é executada em todos os componentes básicos da implantação. O primeiro componente básico implantado deve executar os serviços de gerenciamento, metadados e storage do BeeGFS (conhecido como componente básico). Todos os componentes básicos subsequentes podem ser configurados por meio de software para estender metadados e serviços de storage ou para fornecer exclusivamente serviços de storage. Essa abordagem modular permite dimensionar o sistema de arquivos de acordo com as necessidades de um workload e, ao mesmo tempo, usar as mesmas plataformas de hardware subjacentes e o design de componentes básicos.

Até cinco componentes básicos podem ser implantados para formar um cluster Linux HA autônomo. Isso otimiza o gerenciamento de recursos com a Pacemaker e mantém sincronização eficiente com o Corosync. Um ou mais clusters autônomos do BeeGFS HA são combinados para criar um sistema de arquivos BeeGFS acessível aos clientes como um namespace de storage único. No lado do hardware, um único rack de 42U U pode acomodar até cinco componentes básicos, juntamente com dois switches InfiniBand de 1U GB para a rede de dados/storage. Veja o gráfico abaixo para uma representação visual.

É necessário um mínimo de dois componentes básicos para estabelecer quorum no cluster de failover. Um cluster de dois nós tem limitações que podem impedir que ocorra um failover bem-sucedido. Você pode configurar um cluster de dois nós incorporando um terceiro dispositivo como um tiebreaker. No entanto, esta documentação não descreve esse design.



Ansible

O BeeGFS no NetApp é fornecido e implantado usando a automação do Ansible, hospedada no GitHub e no Ansible Galaxy (a coleção BeeGFS está disponível na "Ansible Galaxy" e "NetApp's e-Series GitHub"). Embora o Ansible seja testado principalmente com o hardware usado para montar os componentes básicos do BeeGFS, é possível configurá-lo para ser executado em praticamente qualquer servidor baseado em x86 usando uma distribuição Linux compatível.

Para obter mais informações, "Implantando o BeeGFS com o storage e-Series"consulte .

Requisitos técnicos

Para implementar a solução BeeGFS no NetApp, garanta que seu ambiente atenda aos requisitos de tecnologia descritos neste documento.

Requisitos de hardware

Antes de começar, certifique-se de que seu hardware atenda às seguintes especificações para um único design de componente básico de segunda geração da solução BeeGFS on NetApp. Os componentes exatos para uma determinada implantação podem variar com base nos requisitos do cliente.

Quantidade	Componente de hardware	Requisitos
2	Nós de arquivo BeeGFS	Cada nó de arquivo deve atender ou exceder as especificações dos nós de arquivo recomendados para obter a performance esperada. • Opções de nó de arquivo recomendadas:* • **Lenovo ThinkSystem SR665 V3** ◦ * Processadores:* 2x AMD EPYC 9124 16C 3,0 GHz (configurado como duas zonas NUMA). ◦ **Memória:** 256GBGB (16x 16GB TruDDR5 4800MHzGB RDIMM-A) ◦ **Expansão PCIe:** quatro slots PCIe Gen5 x16 (dois por zona NUMA) ◦ **Diversos:** ▪ Duas unidades em RAID 1 para os (1TB 7,2K SATA ou superior) ▪ Porta de 1GbE GbE para gerenciamento de SO na banda ▪ 1GbE BMC com API Redfish para gerenciamento de servidores fora da banda ▪ Fontes de alimentação duplas hot swap e ventoinhas de desempenho • **Lenovo ThinkSystem SR665** ◦ * Processadores:* 2x AMD EPYC 7343 16C 3,2 GHz (configurado como duas zonas NUMA). ◦ **Memória:** 256GBGB (16x 16GB TruDDR4 3200MHzGB RDIMM-A) ◦ **Expansão PCIe:** quatro slots PCIe Gen4 x16 (dois por zona NUMA) ◦ **Diversos:** ▪ Duas unidades em RAID 1 para os (1TB 7,2K SATA ou superior) ▪ Porta de 1GbE GbE para gerenciamento de SO na banda ▪ 1GbE BMC com API Redfish para gerenciamento de servidores fora da banda ▪ Fontes de alimentação duplas hot swap e ventoinhas de desempenho
2	Nós de bloco do e-Series (array de EF600 U)	**Memória:** 256GB GB (128GB GB por controlador). **Adaptador:** 200GB/HDR de 2 portas (NVMe/IB). **Drives:** configurado para corresponder aos metadados e à capacidade de armazenamento desejados.

Quantidade	Componente de hardware	Requisitos
8	Adaptadores de placa de host InfiniBand (para nós de arquivo).	Os adaptadores de placa de host variam de acordo com o modelo de servidor usado para o nó de arquivo. As recomendações para nós de arquivos verificados incluem: • **Servidor Lenovo ThinkSystem SR665 V3:** ◦ MCX755106AS-Heat ConnectX-7, NDR200, QSFP112, 2 portas, PCIe Gen5 x16, adaptador InfiniBand • **Servidor Lenovo ThinkSystem SR665:** ◦ MCX653106A-HDAT ConnectX-6, HDR, QSFP-56, 2 portas, PCIe Gen4 x16, adaptador InfiniBand
1	Switch de rede de armazenamento	O switch de rede de storage deve ter capacidade para velocidades InfiniBand de 200GB GB/s. Os modelos de interrutores recomendados incluem: • **NVIDIA QM9700 Quantum 2 NDR switch InfiniBand** • **NVIDIA MQM8700 Quantum HDR InfiniBand switch**

Requisitos de cabeamento

Conexões diretas de nós de bloco para nós de arquivo.

Quantidade	Número de peça	Comprimento
8	MCP1650-H001E30 (cabo de cobre passivo NVIDIA, QSFP56, 200GBm/s)	1 m

Conexões de nós de arquivo para o switch de rede de storage. Selecione a opção de cabo apropriada na tabela a seguir de acordo com o switch de armazenamento InfiniBand. O comprimento recomendado do cabo é de 2m mm; no entanto, isso pode variar de acordo com o ambiente do cliente.

Modelo do interrutor	Quantidade	Tipo de cabo	Número de peça
NVIDIA QM9700	4	Fibra ativa	MFA7U10-H002 (cabo de fibra ativa NVIDIA, InfiniBand de 400GB GB/s a 2x 200GB GB/s, OSFP a 2x QSFP56 GB)
NVIDIA QM9700	4	Cobre passivo	MCP7Y60-H002 (cabo de cobre passivo NVIDIA, InfiniBand de 400GB GB/s a 2x 200GB GB/s, OSFP a 2x QSFP56 GB)
NVIDIA MQM8700	8	Fibra ativa	MFS1S00-H003E (cabo de fibra ativa NVIDIA, InfiniBand de 200GB GB/s, QSFP56 GB)
NVIDIA MQM8700	8	Cobre passivo	MCP1650-H002E26 (cabo de cobre passivo NVIDIA, InfiniBand de 200GB GB/s, QSFP56 GB)

Requisitos de software

Para performance e confiabilidade previsíveis, as versões da solução BeeGFS on NetApp são testadas com versões específicas dos componentes de software necessárias para implementar a solução.

Requisitos de nó de arquivo

Software	Versão
RedHat Enterprise Linux	Servidor RedHat 9,3 físico com alta disponibilidade (soquete 2). Os nós de arquivo exigem uma assinatura válida do RedHat Enterprise Linux Server e o Red Hat Enterprise Linux High Availability Add-on.
Kernel do Linux	5.14.0-362.24.1.el9_3.x86_64
Drivers InfiniBand / RDMA	MLNX_OFED_LINUX-23,10-3,2.2,0-LTS
Firmware HCA	• Firmware HCA * ConnectX-7 FW: 28.39.1002 e PXE: 3.7.0201 e UEFI: 14.32.0012 • Firmware HCA * ConnectX-6 FW: 20.31.1014 e PXE: 3.6.0403 e UEFI: 14.24.0013

Requisitos de nó de bloco de EF600 U.

Software	Versão
Sistema operacional SANtricity	11.80.0
NVSRAM	N6000-880834-D08.dlp
Firmware da unidade	Mais recente disponível para os modelos de acionamento em uso.

Requisitos de implantação de software

A tabela a seguir lista os requisitos de software implantados automaticamente como parte da implantação do BeeGFS baseada em Ansible.

Software	Versão
BeeGFS	7.4.4
Corosync	3,1.5-4
Pacemaker	2,1.4-5
OpenSM	opensm-5.17.2 (de MLNX_OFED_Linux-23,10-3,2.2,0-LTS)

Requisitos de nó de controle do Ansible

A solução BeeGFS no NetApp é implantada e gerenciada a partir de um nó de controle do Ansible. Para obter mais informações, consulte "Documentação do Ansible" .

Os requisitos de software listados nas tabelas a seguir são específicos da versão da coleção Ansible do

NetApp BeeGFS listada abaixo.

Software	Versão
Ansible	6.x quando instalado através do pip: Ansible-6.0.0 e Ansible-core > 2.13.0
Python	3,9 (ou posterior)
Pacotes Python adicionais	Criptografia-43,0.0, netaddr-1,3.0, ipaddr-2.2.0
Coleção BeeGFS do NetApp e-Series	3.2.0

Rever o design da solução

Visão geral do design

Equipamentos, cabeamento e configurações específicos são necessários para dar suporte à solução BeeGFS on NetApp, que combina o sistema de arquivos paralelos do BeeGFS com os sistemas de storage NetApp EF600.

Saiba mais:

- "Configuração de hardware"
- "Configuração de software"
- "Verificação do design"
- "Diretrizes de dimensionamento"
- "Ajuste de desempenho"

Arquiteturas derivadas com variações de design e desempenho:

- "Bloco de construção de alta capacidade"

Configuração de hardware

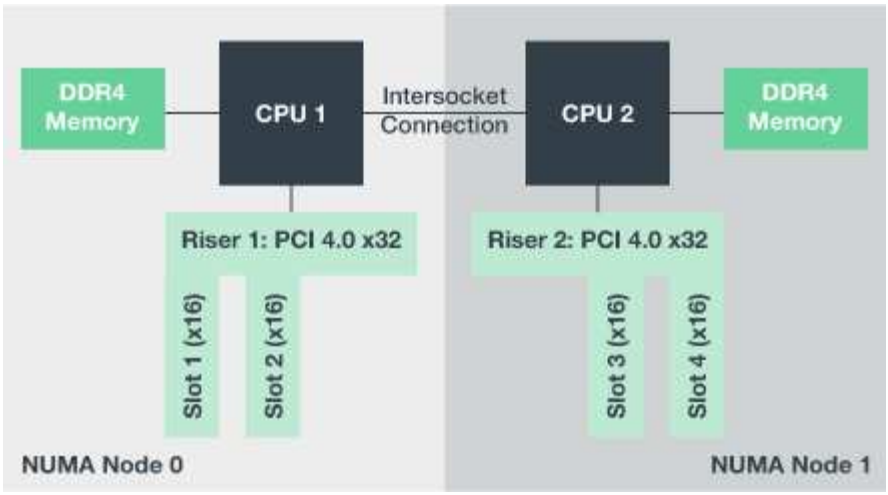
A configuração de hardware do BeeGFS no NetApp inclui nós de arquivo e cabeamento de rede.

Configuração do nó do arquivo

Os nós de arquivo têm dois soquetes de CPU configurados como zonas NUMA separadas, que incluem acesso local a um número igual de slots PCIe e memória.

Os adaptadores InfiniBand devem ser preenchidos nos risers ou slots PCI apropriados, de modo que a carga de trabalho seja equilibrada sobre as faixas PCIe e os canais de memória disponíveis. Você equilibra o workload com o isolamento total do trabalho de serviços individuais do BeeGFS para um nó específico. O objetivo é alcançar um desempenho semelhante de cada nó de arquivo como se fossem dois servidores de soquete único independentes.

A figura a seguir mostra a configuração do nó de arquivo NUMA.



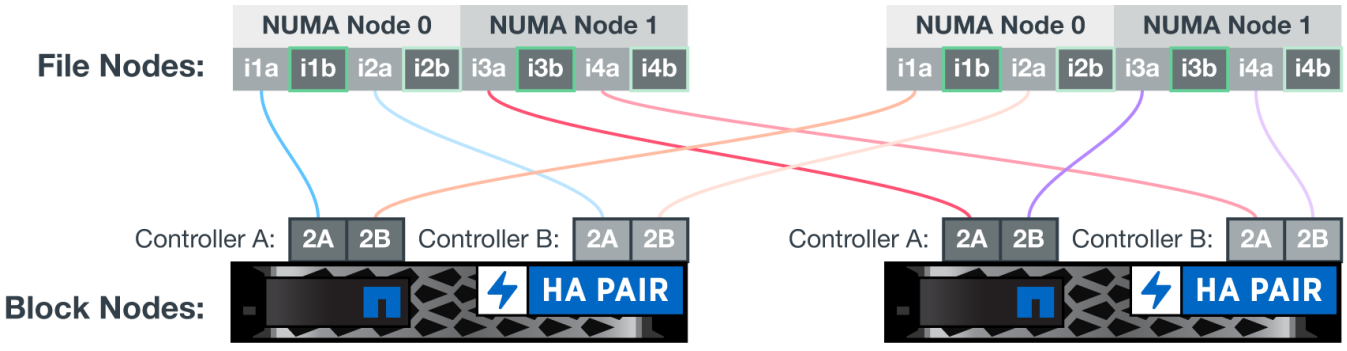
Os processos BeeGFS são fixados a uma zona NUMA específica para garantir que as interfaces usadas estejam na mesma zona. Esta configuração evita a necessidade de acesso remoto através da ligação entre sockets. A conexão entre soquetes às vezes é conhecida como QPI ou link GMI2; mesmo em arquiteturas de processador modernas, eles podem ser um gargalo ao usar redes de alta velocidade como HDR InfiniBand.

Configuração de cabeamento de rede

Em um componente básico, cada nó de arquivo é conetado a dois nós de bloco usando um total de quatro conexões InfiniBand redundantes. Além disso, cada nó de arquivo tem quatro conexões redundantes com a rede de storage InfiniBand.

Na figura a seguir, observe que:

- Todas as portas de nós de arquivo descritas em verde são usadas para se conectar à malha de storage. Todas as outras portas de nós de arquivo são as conexões diretas aos nós de bloco.
- Duas portas InfiniBand em uma zona NUMA específica se conetam aos controladores A e B do mesmo nó de bloco.
- As portas no nó NUMA 0 sempre se conetam ao primeiro nó de bloco.
- As portas no nó NUMA 1 conetam-se ao segundo nó de bloco.



Ao usar cabos divisores para conetar o switch de armazenamento a nós de arquivo, um cabo deve ramificar e conetar-se às portas delineadas em verde claro. Outro cabo deve ramificar e conetar-se às portas delineadas em verde escuro. Além disso, para redes de armazenamento com switches redundantes, as portas delineadas em verde claro devem se conetar a um switch, enquanto as portas em verde escuro devem se conetar a outro switch.

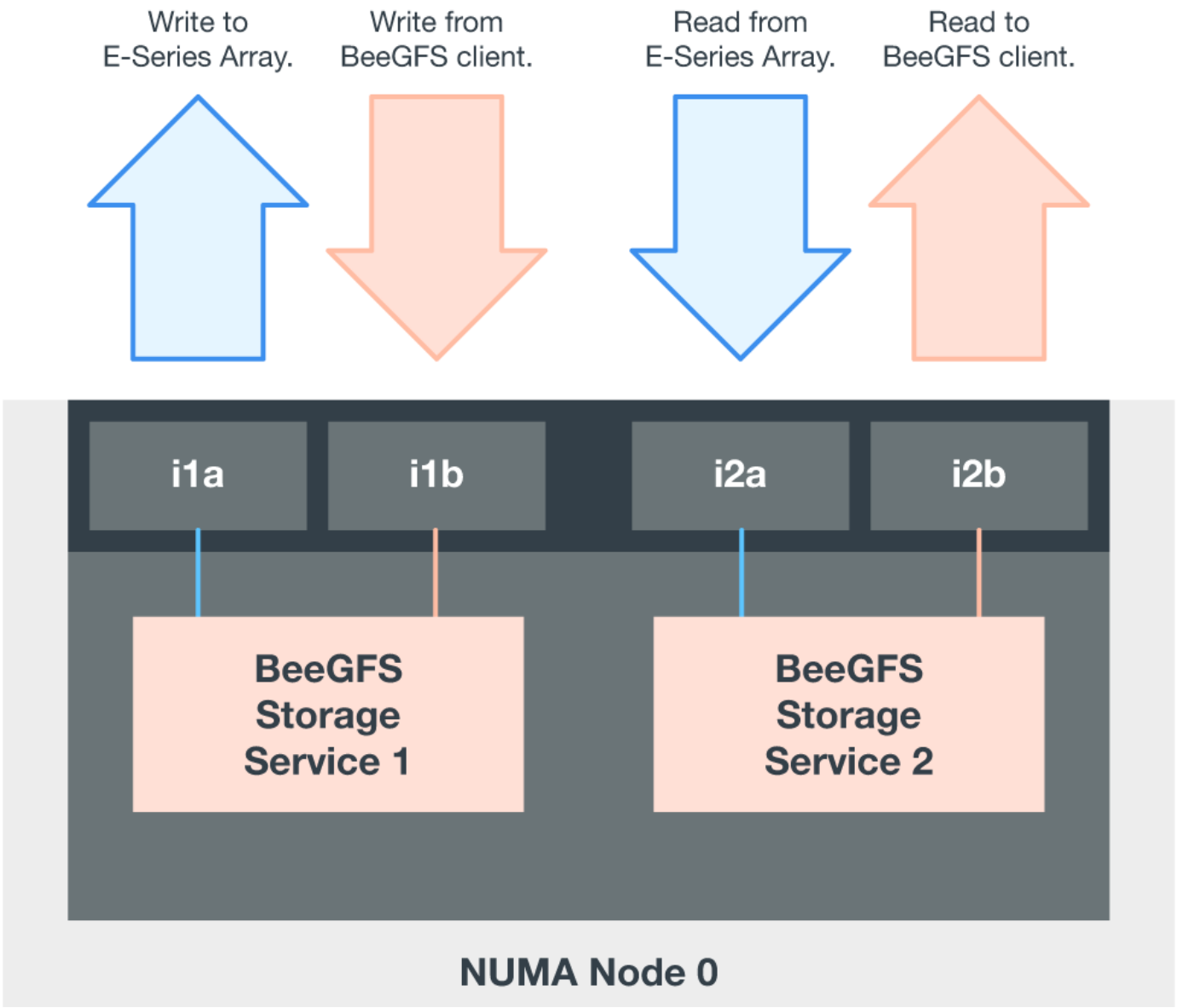
A configuração de cabeamento ilustrada na figura permite que cada serviço BeeGFS:

- Execute na mesma zona NUMA, independentemente do nó de arquivo que está executando o serviço BeeGFS.
- Ter caminhos secundários ideais para a rede de storage de front-end e para os nós de bloco de back-end, independentemente de onde ocorra uma falha.
- Minimize os efeitos de desempenho se um nó de arquivo ou controlador em um nó de bloco exigir manutenção.

Cabeamento para aproveitar a largura de banda

Para utilizar a largura de banda bidirecional PCIe completa, verifique se uma porta em cada adaptador InfiniBand se conecta à malha de storage e se a outra porta se conecta a um nó de bloco.

A figura a seguir mostra o projeto de cabeamento usado para aproveitar a largura de banda bidirecional PCIe completa.



Para cada serviço BeeGFS, use o mesmo adaptador para conetar a porta preferida usada para o tráfego do cliente com o caminho para a controladora de nós de bloco que é o principal proprietário desses volumes de

serviços. Para obter mais informações, "Configuração de software"consulte .

Configuração de software

A configuração de software do BeeGFS no NetApp inclui componentes de rede BeeGFS, nós de arquivo de EF600 blocos, nós de arquivo BeeGFS, grupos de recursos e serviços BeeGFS.

Configuração de rede BeeGFS

A configuração de rede BeeGFS consiste nos seguintes componentes.

- **IPs flutuantes** os IPs flutuantes são uma espécie de endereço IP virtual que pode ser roteado dinamicamente para qualquer servidor na mesma rede. Vários servidores podem possuir o mesmo endereço IP flutuante, mas só podem estar ativos em um servidor a qualquer momento.

  Cada serviço de servidor BeeGFS tem seu próprio endereço IP que pode se mover entre nós de arquivo, dependendo do local de execução do serviço de servidor BeeGFS. Esta configuração IP flutuante permite que cada serviço faça failover independentemente para o outro nó de arquivo. O cliente simplesmente precisa saber o endereço IP de um determinado serviço BeeGFS; ele não precisa saber qual nó de arquivo está executando esse serviço no momento.

- **Configuração multihoming do servidor BeeGFS** para aumentar a densidade da solução, cada nó de arquivo tem várias interfaces de storage com IPs configurados na mesma sub-rede IP.

  Configuração adicional é necessária para garantir que essa configuração funcione como esperado com a pilha de rede Linux, porque por padrão, as solicitações para uma interface podem ser respondidas em uma interface diferente se seus IPs estiverem na mesma sub-rede. Além de outras desvantagens, esse comportamento padrão torna impossível estabelecer ou manter adequadamente conexões RDMA.

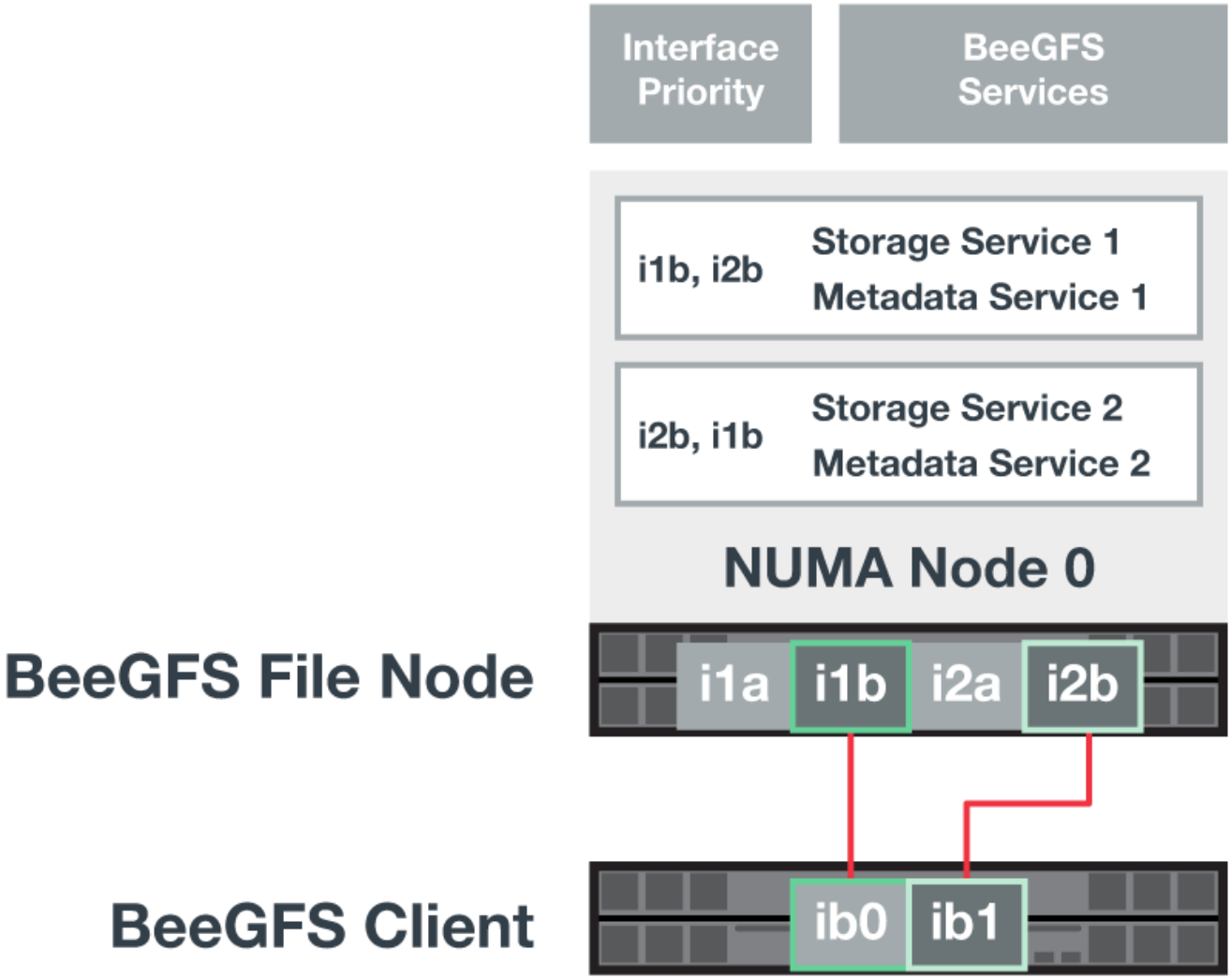
  A implantação baseada em Ansible lida com o aperto do comportamento do caminho reverso (RP) e do protocolo de resolução de endereço (ARP), além de garantir quando os IPs flutuantes são iniciados e parados; as rotas e regras IP correspondentes são criadas dinamicamente para permitir que a configuração de rede multihomed funcione corretamente.

- **A configuração multitrilho do cliente BeeGFS** *Multi-rail* refere-se à capacidade de um aplicativo usar várias conexões de rede independentes, ou "trilhos", para aumentar o desempenho.

  BeeGFS implementa suporte multi-trilho para permitir o uso de várias interfaces IB em uma única sub-rede IPoIB. Essa capacidade permite recursos como balanceamento dinâmico de carga entre NICs de RDMA, otimizando o uso de recursos de rede. Ele também se integra ao armazenamento GPUDirect NVIDIA (GDS), que oferece maior largura de banda do sistema e diminui a latência e a utilização na CPU do cliente.

  Esta documentação fornece instruções para configurações de sub-rede IPoIB únicas. As configurações de sub-rede IPoIB duplas são suportadas, mas não fornecem as mesmas vantagens que as configurações de sub-rede única.

  A figura a seguir mostra o balanceamento de tráfego entre várias interfaces de cliente BeeGFS.



Como cada arquivo no BeeGFS geralmente é distribuído em vários serviços de storage, a configuração de vários trilhos permite que o cliente obtenha mais taxa de transferência do que o possível com uma única porta InfiniBand. Por exemplo, a amostra de código a seguir mostra uma configuração comum de distribuição de arquivos que permite ao cliente equilibrar o tráfego entre ambas as interfaces:

E

```
root@beegfs01:/mnt/beegfs# beegfs-ctl --getentryinfo myfile
Entry type: file
EntryID: 11D-624759A9-65
Metadata node: meta_01_tgt_0101 [ID: 101]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 1M
+ Number of storage targets: desired: 4; actual: 4
+ Storage targets:
  + 101 @ stor_01_tgt_0101 [ID: 101]
  + 102 @ stor_01_tgt_0101 [ID: 101]
  + 201 @ stor_02_tgt_0201 [ID: 201]
  + 202 @ stor_02_tgt_0201 [ID: 201]
```

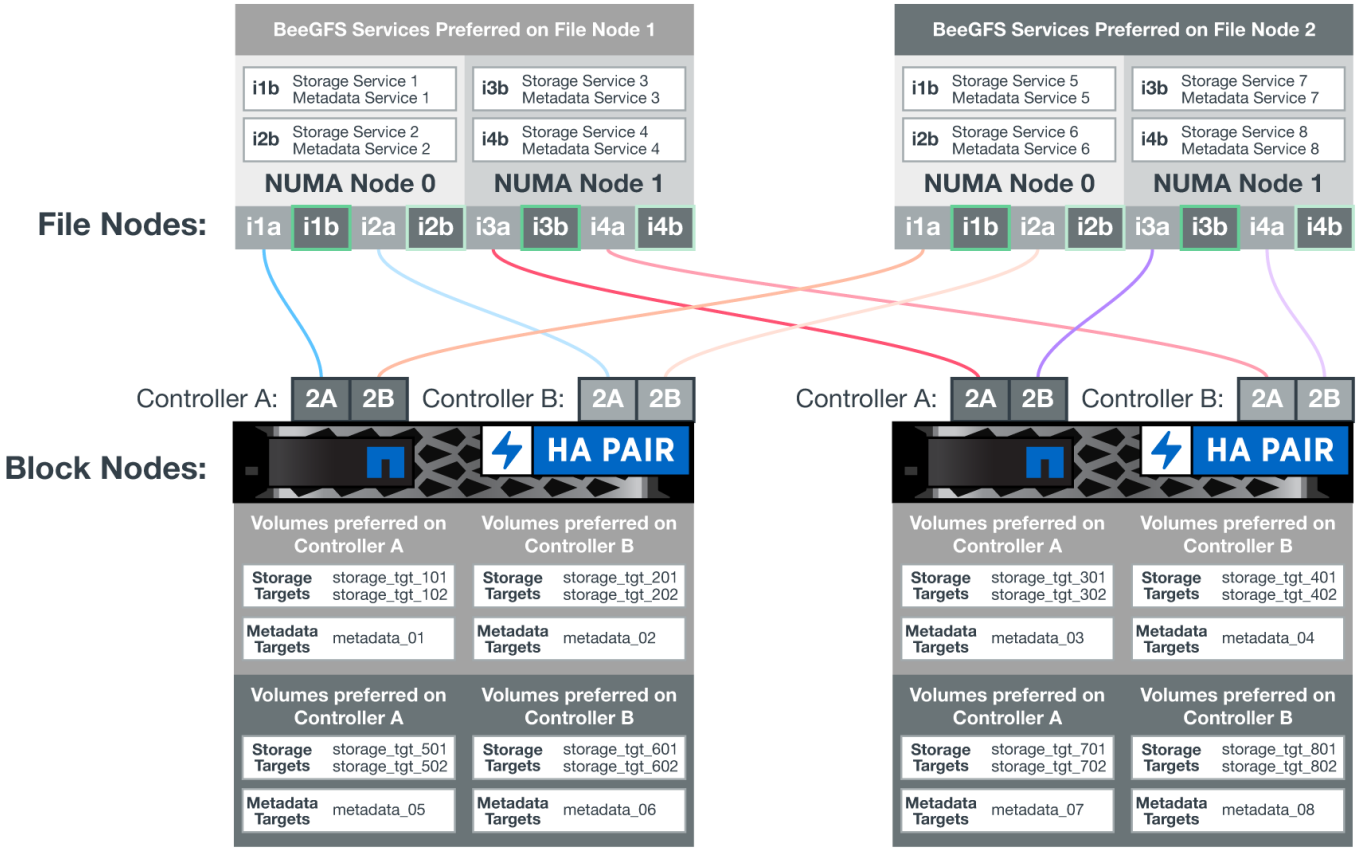
Configuração de nó de bloco de EF600 U.

Os nós de bloco são compostos por duas controladoras RAID ativas/ativas com acesso compartilhado ao mesmo conjunto de unidades. Normalmente, cada controlador possui metade dos volumes configurados no sistema, mas pode assumir o controle para o outro controlador conforme necessário.

O software multipathing nos nós de arquivo determina o caminho ativo e otimizado para cada volume e se move automaticamente para o caminho alternativo no caso de uma falha de cabo, adaptador ou controlador.

O diagrama a seguir mostra o layout do controlador em EF600 nós de bloco.

Para facilitar a solução de HA de disco compartilhado, os volumes são mapeados para os dois nós de arquivo, para que eles possam assumir o controle uns dos outros conforme necessário. O diagrama a seguir mostra um exemplo de como o serviço BeeGFS e a propriedade de volume preferencial são configurados para obter o máximo de performance. A interface à esquerda de cada serviço BeeGFS indica a interface preferida que os clientes e outros serviços usam para contatá-lo.



No exemplo anterior, clientes e serviços de servidor preferem se comunicar com o serviço de armazenamento 1 usando a interface i1b. O serviço de armazenamento 1 usa a interface i1a como o caminho preferido para se comunicar com seus volumes (storage_tgt_101, 102) no controlador A do primeiro nó de bloco. Esta configuração utiliza a largura de banda PCIe bidirecional completa disponível para o adaptador InfiniBand e consegue um melhor desempenho a partir de um adaptador InfiniBand HDR de porta dupla do que seria possível com PCIe 4,0.

Configuração do nó de arquivo BeeGFS

Os nós de arquivos BeeGFS são configurados em um cluster de alta disponibilidade (HA) para facilitar o failover de serviços BeeGFS entre vários nós de arquivo.

O projeto de cluster HA é baseado em dois projetos Linux HA amplamente utilizados: Corosync para associação de cluster e Pacemaker para gerenciamento de recursos de cluster. Para obter mais informações, "Treinamento da Red Hat para complementos de alta disponibilidade"consulte .

A NetApp criou e estendeu vários agentes de recursos da estrutura de cluster aberta (OCF) para permitir que o cluster inicie e monitore de forma inteligente os recursos do BeeGFS.

Clusters de HA do BeeGFS

Normalmente, quando você inicia um serviço BeeGFS (com ou sem HA), alguns recursos devem estar implementados:

- Endereços IP onde o serviço é acessível, normalmente configurados pelo Network Manager.
- Sistemas de arquivos subjacentes usados como destino para o BeeGFS armazenar dados.

  Estes são tipicamente definidos em `/etc/fstab` e montados pelo Systemd.

- Um serviço Systemd responsável por iniciar processos BeeGFS quando os outros recursos estiverem prontos.

  Sem software adicional, esses recursos começam apenas em um único nó de arquivo. Portanto, se o nó de arquivo ficar offline, uma parte do sistema de arquivos BeeGFS fica inacessível.

Como vários nós podem iniciar cada serviço BeeGFS, o fabricante de pacemaker precisa garantir que cada serviço e recursos dependentes estejam sendo executados apenas em um nó de cada vez. Por exemplo, se dois nós tentarem iniciar o mesmo serviço BeeGFS, há o risco de corrupção de dados se ambos tentarem gravar nos mesmos arquivos no destino subjacente. Para evitar esse cenário, a Pacemaker confia no Corosync para manter o estado geral do cluster em sincronia entre todos os nós e estabelecer quórum.

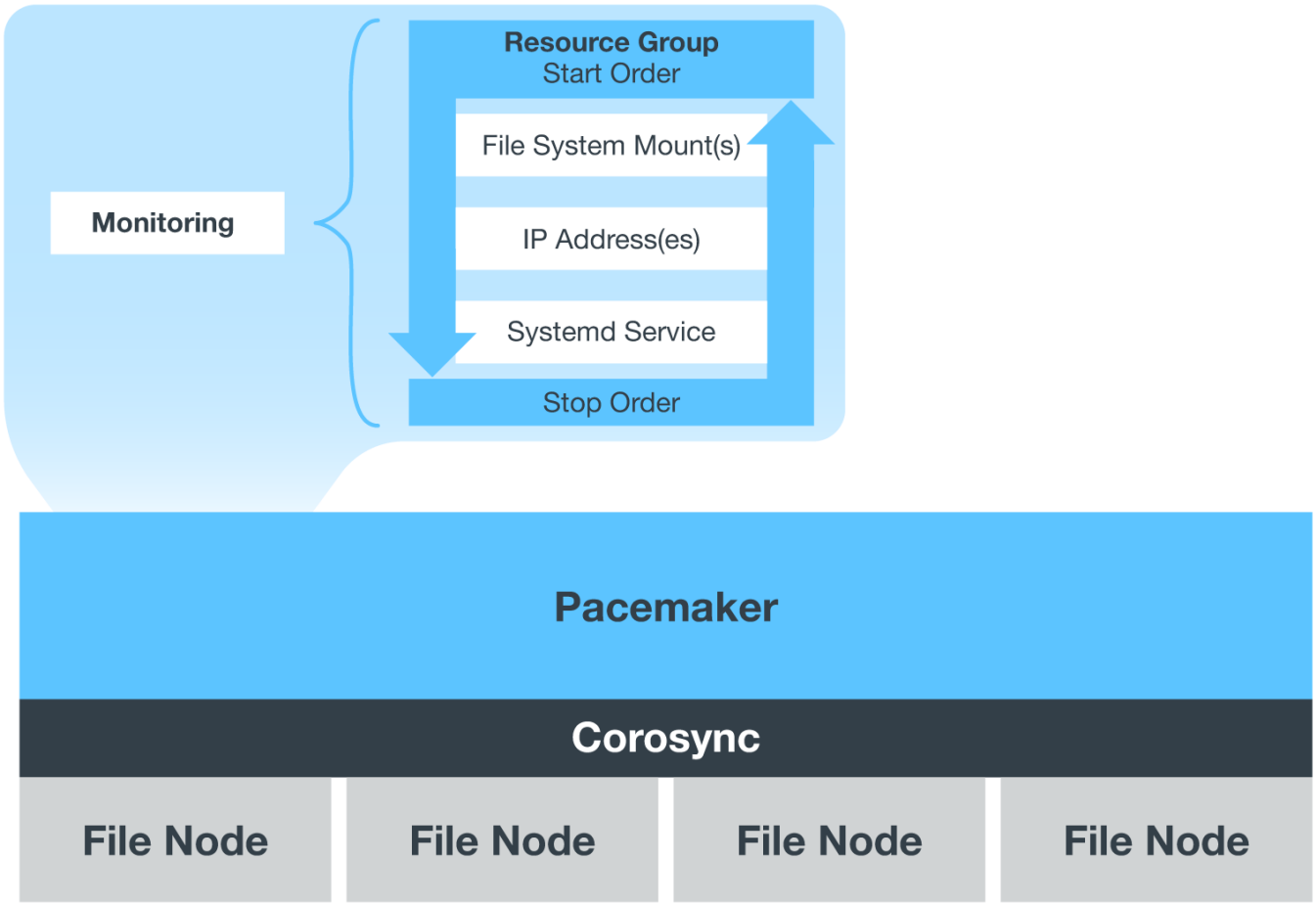
Se houver uma falha no cluster, o fabricante de Paz reagirá e reiniciará os recursos do BeeGFS em outro nó. Em alguns cenários, o pacemaker pode não ser capaz de se comunicar com o nó defeituoso original para confirmar que os recursos estão parados. Para verificar se o nó está inativo antes de reiniciar os recursos do BeeGFS em outro lugar, o fabricante de pacemaker bloqueia o nó com falha, idealmente removendo energia.

Muitos agentes de esgrima de código aberto estão disponíveis que permitem que o pacemaker cerque um nó com uma unidade de distribuição de energia (PDU) ou usando o controlador de gerenciamento de placa base (BMC) do servidor com APIs como o Redfish.

Quando o BeeGFS está em execução em um cluster de HA, todos os serviços do BeeGFS e os recursos subjacentes são gerenciados pelo Pacemaker em grupos de recursos. Cada serviço BeeGFS e os recursos dos quais depende são configurados em um grupo de recursos, o que garante que os recursos sejam iniciados e parados na ordem correta e colocados no mesmo nó.

Para cada grupo de recursos BeeGFS, o pacemaker executa um recurso de monitoramento personalizado BeeGFS, responsável por detetar condições de falha e acionar failovers de forma inteligente quando um serviço BeeGFS não estiver mais acessível em um nó específico.

A figura a seguir mostra os serviços e dependências do BeeGFS controlados pelo pacemaker.



Para que vários serviços BeeGFS do mesmo tipo sejam iniciados no mesmo nó, o pacemaker é configurado para iniciar os serviços BeeGFS usando o método de configuração Multi Mode. Para obter mais informações, consulte "Documentação BeeGFS no modo Multi" .

Como os serviços BeeGFS precisam ser capazes de iniciar em vários nós, o arquivo de configuração de cada serviço (normalmente localizado em `/etc/beegfs`) é armazenado em um dos volumes do e-Series usados como destino do BeeGFS para esse serviço. Isso torna a configuração, juntamente com os dados de um serviço BeeGFS específico, acessível a todos os nós que possam precisar para executar o serviço.

```
# 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
```

Verificação do design

O design de segunda geração da solução BeeGFS on NetApp foi verificado usando três perfis de configuração de componentes básicos.

Os perfis de configuração incluem o seguinte:

- Um componente básico único, incluindo gerenciamento, metadados e serviços de storage do BeeGFS.
- Metadados do BeeGFS, além de um componente básico de storage.
- Componente básico somente de storage do BeeGFS.

Os componentes básicos foram anexados a dois switches NVIDIA Quantum InfiniBand (MQM8700). Dez clientes BeeGFS também foram anexados aos switches InfiniBand e usados para executar utilitários de benchmark sintéticos.

A figura a seguir mostra a configuração BeeGFS usada para validar a solução BeeGFS no NetApp.

Distribuição de arquivos BeeGFS

Um benefício dos sistemas de arquivos paralelos é a capacidade de distribuir arquivos individuais entre vários destinos de storage, o que pode representar volumes no mesmo ou em diferentes sistemas de storage subjacentes.

No BeeGFS, você pode configurar a distribuição por diretório e por arquivo para controlar o número de destinos usados para cada arquivo e controlar o tamanho do bloco (ou tamanho do bloco) usado para cada faixa de arquivo. Essa configuração permite que o sistema de arquivos ofereça suporte a diferentes tipos de cargas de trabalho e perfis de e/S sem a necessidade de reconfigurar ou reiniciar serviços. Você pode aplicar configurações de stripe usando a `beegfs-ctl` ferramenta de linha de comando ou com aplicativos que usam a API de striping. Para obter mais informações, consulte a documentação do BeeGFS para "Riscar" e "Striping API".

Para alcançar o melhor desempenho, padrões de faixa foram ajustados ao longo dos testes, e os parâmetros usados para cada teste são observados.

Testes de largura de banda IOR: Vários clientes

Os testes de largura de banda IOR usaram o OpenMPI para executar trabalhos paralelos da ferramenta de gerador de e/S sintética IOR (disponível a partir de "HPC GitHub") em todos os nós de cliente 10 para um ou mais blocos de construção BeeGFS. Salvo indicação em contrário:

- Todos os testes usaram I/o direto com um tamanho de transferência de 1MiB.
- A distribuição de arquivos BeeGFS foi definida como um tamanho de 1MB chunksize e um destino por arquivo.

Os seguintes parâmetros foram utilizados para IOR com a contagem de segmentos ajustada para manter o tamanho do arquivo agregado para 5TiB para um bloco de construção e 40TiB para três blocos de construção.

```
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
```

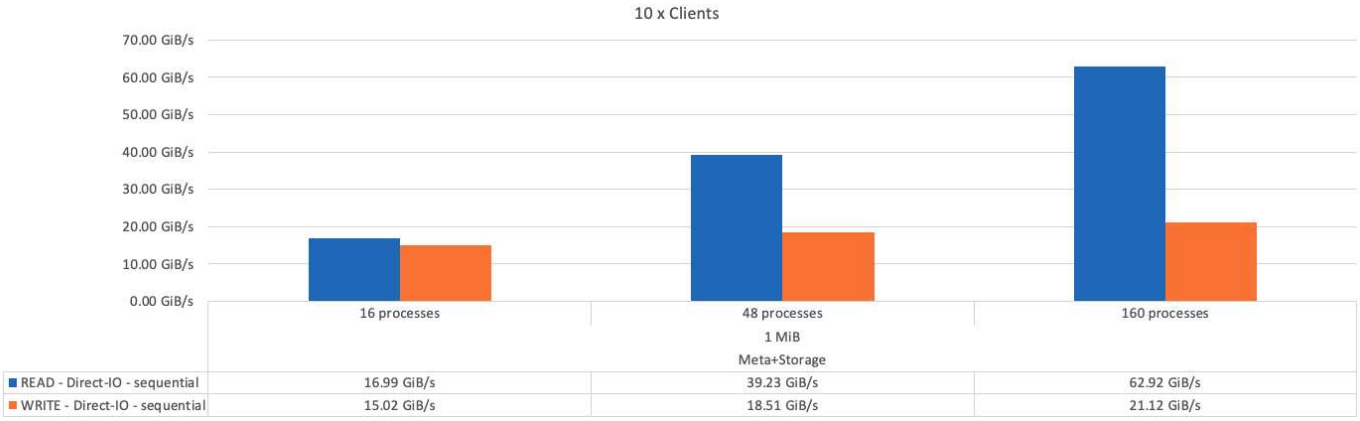
Uma base do BeeGFS (gerenciamento, metadados e storage) componente básico

A figura a seguir mostra os resultados do teste de IOR com um único componente básico do BeeGFS

(gerenciamento, metadados e storage).

Componentes básicos de storage e metadados do BeeGFS

A figura a seguir mostra os resultados do teste de IOR com um único componente básico de storage e metadados do BeeGFS.

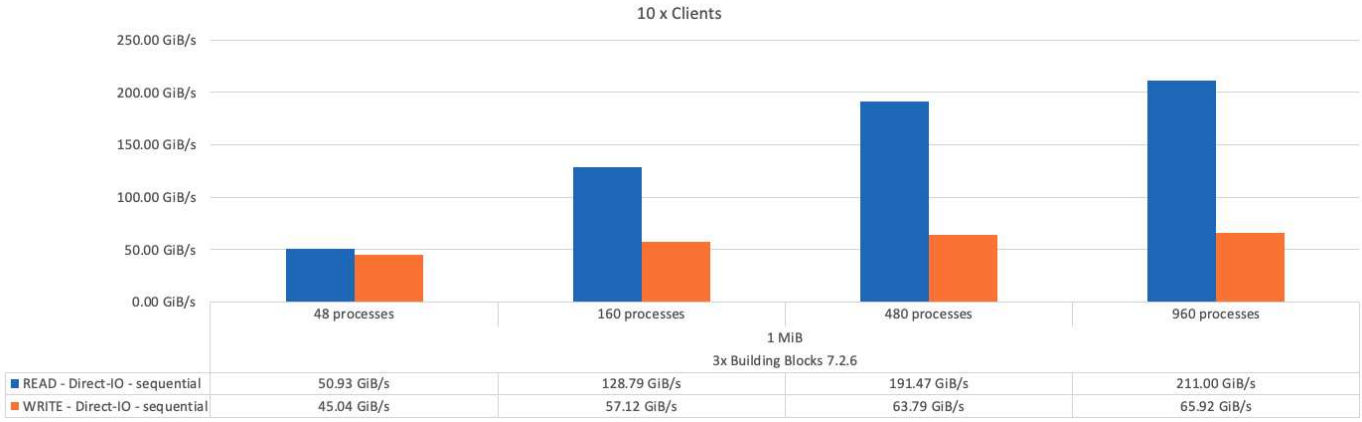


Componente básico somente de storage do BeeGFS

A figura a seguir mostra os resultados do teste de IOR com um único componente básico somente de storage do BeeGFS.

Três blocos de construção BeeGFS

A figura a seguir mostra os resultados do teste de IOR com três componentes básicos do BeeGFS.



Como esperado, a diferença de desempenho entre o componente básico e o componente básico de armazenamento de metadados subsequentes é insignificante. Comparar os metadados e o componente básico de armazenamento e um componente básico somente de armazenamento mostra um ligeiro aumento no desempenho de leitura devido às unidades adicionais usadas como destinos de armazenamento. No entanto, não há diferença significativa no desempenho de gravação. Para obter um desempenho mais alto, você pode adicionar vários blocos de construção juntos para dimensionar o desempenho de forma linear.

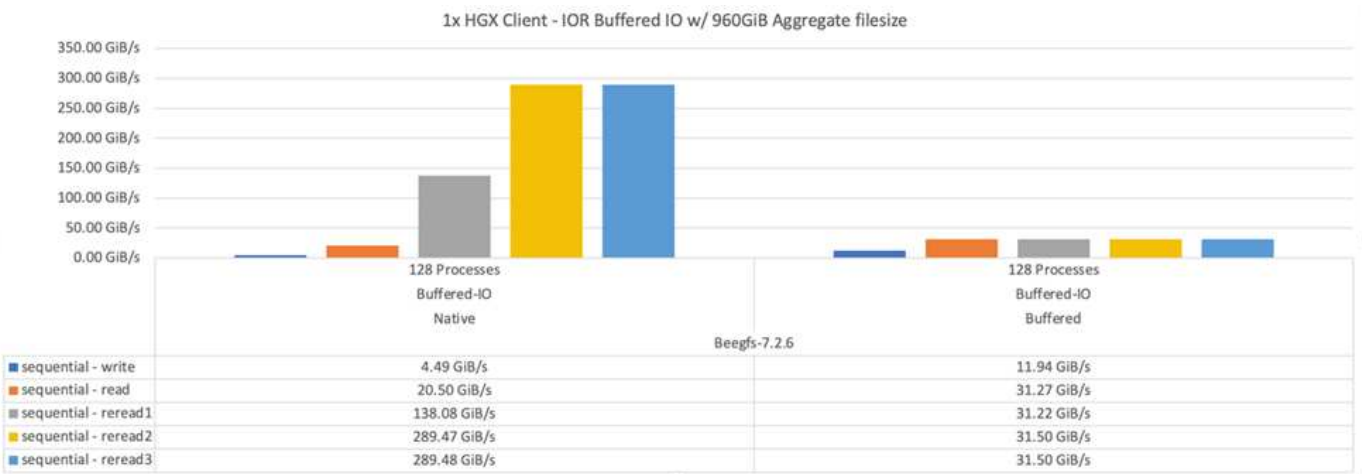
Testes de largura de banda IOR: Cliente único

O teste de largura de banda IOR usou o OpenMPI para executar vários processos IOR usando um único servidor GPU de alto desempenho para explorar o desempenho possível para um único cliente.

Este teste também compara o comportamento de releitura e o desempenho do BeeGFS quando o cliente está configurado para usar o cache de página do kernel Linux (`tuneFileCacheType = native`) versus a configuração padrão `buffered`.

O modo de cache nativo usa o cache de página do kernel Linux no cliente, permitindo que as operações de releitura venham da memória local em vez de serem retransmitidas pela rede.

O diagrama a seguir mostra os resultados do teste de IOR com três componentes básicos do BeeGFS e um único cliente.



BeeGFS striping para esses testes foi definido para um tamanho de 1MB chunksize com oito alvos por arquivo.

Embora o desempenho de gravação e leitura inicial seja maior usando o modo de buffer padrão, para cargas

de trabalho que releram os mesmos dados várias vezes, um aumento de desempenho significativo é observado no modo de armazenamento em cache nativo. Essa performance de releitura aprimorada é importante para workloads como o deep learning que releiam o mesmo conjunto de dados várias vezes em várias épocas.

Teste de desempenho de metadados

Os testes de performance de metadados usaram a ferramenta MDTest (incluída como parte da IOR) para medir a performance de metadados do BeeGFS. Os testes utilizaram OpenMPI para executar trabalhos paralelos em todos os dez nós de cliente.

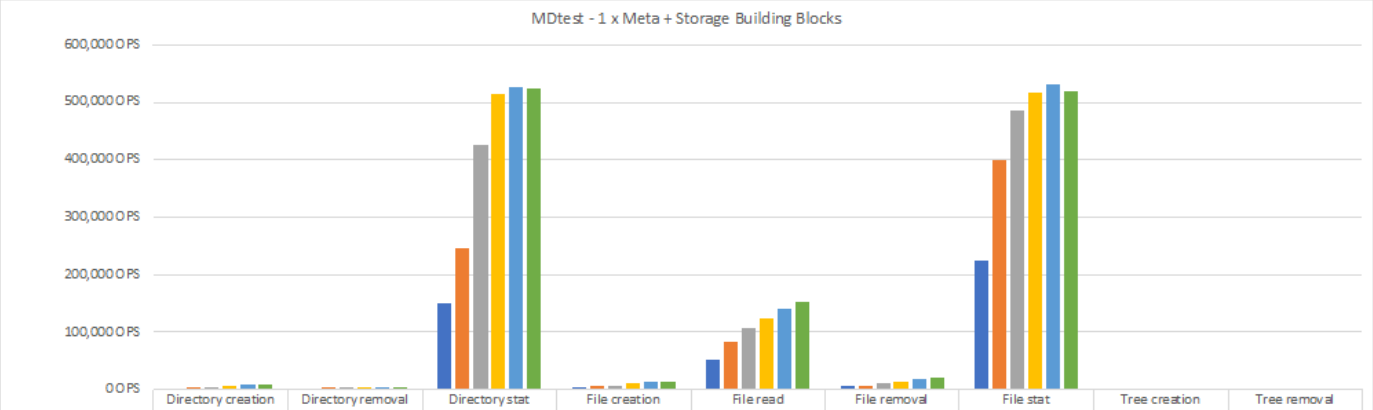
Os seguintes parâmetros foram utilizados para executar o teste de benchmark com o número total de processos dimensionados de 10 a 320 na etapa de 2x e com um tamanho de arquivo de 4K.

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

A performance dos metadados foi medida primeiro com um e dois componentes básicos de storage e metadados para mostrar como a performance é dimensionada com a adição de componentes básicos adicionais.

Um componente básico de storage e metadados do BeeGFS

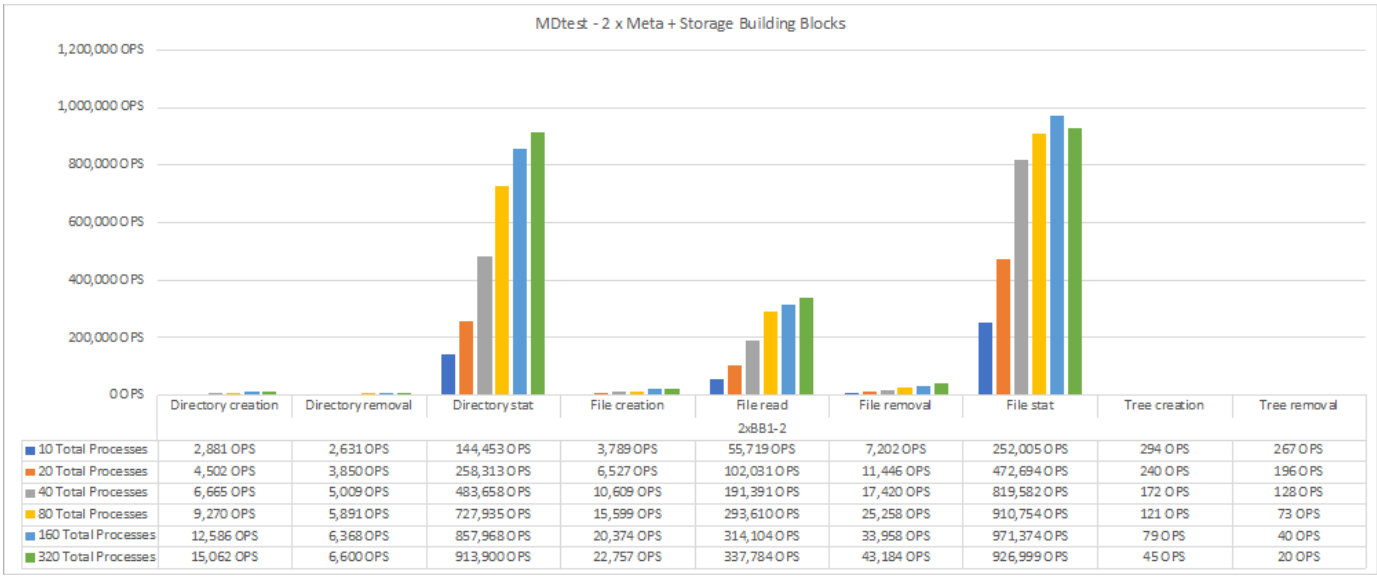
O diagrama a seguir mostra os resultados do MDTest com os componentes básicos de storage e metadados do BeeGFS.



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

Dois componentes básicos de storage e metadados do BeeGFS

O diagrama a seguir mostra os resultados do MDTest com dois componentes básicos de storage e metadados do BeeGFS.



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

Validação funcional

Como parte da validação desta arquitetura, o NetApp executou vários testes funcionais, incluindo os seguintes:

- Falha em uma única porta InfiniBand de cliente desativando a porta do switch.
- Falha em uma única porta InfiniBand de servidor desativando a porta do switch.
- Acionando uma desativação imediata do servidor usando o BMC.
- Colocação graciosa de um nó em standby e falha no serviço para outro nó.
- Colocando um nó de volta on-line e falhando serviços de volta para o nó original.
- Desligar um dos switches InfiniBand usando a PDU. Todos os testes foram realizados enquanto o teste de estresse estava em andamento com o `sysSessionChecksEnabled: false` parâmetro definido nos clientes BeeGFS. Não foram observados erros ou interrupções na e/S.

> Há um problema conhecido (consulte a "Changelog") quando as conexões RDMA cliente/servidor BeeGFS são interrompidas inesperadamente, seja pela perda da interface principal (conforme definido na `connInterfacesFile`) ou por falha de um servidor BeeGFS; e/S cliente ativo pode travar por até dez minutos antes de retomar. Esse problema não ocorre quando os nós BeeGFS são colocados graciosamente dentro e fora de espera para manutenção planejada ou se o TCP estiver em uso.

Validação do NVIDIA DGX SuperPOD e BasePOD

A NetApp validou uma solução de storage para NVIDIAs DGX A100 SuperPOD usando um sistema de arquivos BeeGFS semelhante, que consiste em três componentes básicos com os metadados e o perfil de configuração de storage aplicado. O esforço de qualificação envolveu o teste da solução descrita por esse NVA com vinte servidores de GPU DGX A100 que executam diversos benchmarks de storage, aprendizado de máquina e deep learning. Com base na validação estabelecida pelo SuperPOD DGX A100 da NVIDIA, a solução BeeGFS on NetApp foi aprovada para os sistemas DGX SuperPOD H100, H200 e B200. Essa extensão se baseia no cumprimento dos requisitos de sistema e de benchmark estabelecidos anteriormente, conforme validado com o NVIDIA DGX A100.

Para obter mais informações, "NVIDIA DGX SuperPOD com NetApp" consulte e "NVIDIA DGX BasePOD".

Diretrizes de dimensionamento

A solução BeeGFS inclui recomendações para o dimensionamento da performance e da capacidade com base nos testes de verificação.

Com uma arquitetura de componentes básicos, o objetivo é criar uma solução simples de dimensionar adicionando vários componentes básicos para atender aos requisitos de um sistema BeeGFS específico. Usando as diretrizes abaixo, você pode estimar a quantidade e os tipos de componentes básicos do BeeGFS necessários para atender aos requisitos do seu ambiente.

Tenha em mente que essas estimativas são o melhor desempenho. Aplicativos de benchmarking sintéticos são escritos e utilizados para otimizar o uso de sistemas de arquivos subjacentes de maneiras que aplicativos do mundo real podem não.

Dimensionamento da performance

A tabela a seguir fornece o dimensionamento de desempenho recomendado.

Perfil de configuração	1MiB leituras	1MiB grava
Metadados e armazenamento	62GiBps	21GiBps
Apenas armazenamento	64GiBps	21GiBps

As estimativas de dimensionamento da capacidade dos metadados baseiam-se na "regra geral" de que 500GB TB de capacidade é suficiente para cerca de 150 milhões de arquivos no BeeGFS. (Para obter mais informações, consulte a documentação do BeeGFS para "Requisitos do sistema".)

O uso de recursos como listas de controle de acesso e o número de diretórios e arquivos por diretório também afetam a rapidez com que o espaço de metadados é consumido. As estimativas de capacidade de armazenamento são responsáveis pela capacidade utilizável da unidade, juntamente com a sobrecarga RAID 6 e XFS.

Dimensionamento da capacidade para componentes básicos de storage e metadados

A tabela a seguir fornece o dimensionamento de capacidade recomendado para metadados, além de componentes básicos de storage.

Grupos de volume de metadados de tamanho da unidade (2 RAID 1 2)	Capacidade de metadados (número de arquivos)	Grupos de volume de armazenamento de tamanho da unidade (8 RAID 2 6)	Capacidade de armazenamento (conteúdo do arquivo)
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

Ao dimensionar metadados e componentes básicos de storage, você pode reduzir custos usando unidades menores para grupos de volumes de metadados em vez de grupos de volumes de storage.

Dimensionamento da capacidade para componentes básicos somente de storage

A tabela a seguir fornece o dimensionamento da capacidade de regra geral para componentes básicos somente de storage.

Grupos de volume de armazenamento de tamanho da unidade (10 RAID 2 6)	Capacidade de armazenamento (conteúdo do arquivo)
1,92 TB	59,89 TB
3,84 TB	119,80 TB
7,68 TB	251,89 TB
15,3 TB	538,55 TB

A sobrecarga de desempenho e capacidade de incluir o serviço de gerenciamento no componente básico básico (primeiro) são mínimas, a menos que o bloqueio global de arquivos esteja habilitado.

Ajuste de desempenho

A solução BeeGFS inclui recomendações para ajuste de performance com base nos testes de verificação.

Embora o BeeGFS forneça desempenho razoável pronto para uso, a NetApp desenvolveu um conjunto de parâmetros de ajuste recomendados para maximizar a performance. Esses parâmetros levam em consideração as funcionalidades dos nós de bloco e-Series subjacentes e todos os requisitos especiais necessários para executar o BeeGFS em uma arquitetura de HA de disco compartilhado.

Ajuste de performance para nós de arquivos

Os parâmetros de ajuste disponíveis que você pode configurar incluem o seguinte:

1. * Configurações do sistema na UEFI/BIOS de nós de arquivo.* Para maximizar o desempenho, recomendamos configurar as configurações do sistema no modelo de servidor que você usa como nós de arquivo. Você configura as configurações do sistema quando configura seus nós de arquivo usando a configuração do sistema (UEFI/BIOS) ou as APIs do Redfish fornecidas pelo controlador de gerenciamento de placa base (BMC).

   As configurações do sistema variam dependendo do modelo de servidor que você usa como nó de arquivo. As configurações devem ser configuradas manualmente com base no modelo de servidor em uso. Para saber como configurar as configurações do sistema para os nós de arquivo validados do Lenovo SR665, "Ajuste as configurações do sistema do nó de arquivo para obter desempenho"consulte .

2. * Configurações padrão para os parâmetros de configuração necessários.* Os parâmetros de configuração necessários afetam a forma como os serviços BeeGFS são configurados e como os volumes do e-Series (dispositivos de bloco) são formatados e montados pelo pacemaker. Estes parâmetros de configuração necessários incluem o seguinte:
   - Parâmetros de configuração do BeeGFS Service

     Você pode substituir as configurações padrão para os parâmetros de configuração, conforme necessário. Para obter os parâmetros que podem ser ajustados para suas cargas de trabalho ou casos de uso específicos, consulte o "Parâmetros de configuração do serviço BeeGFS".

- Os parâmetros de formatação e montagem do volume são definidos como padrões recomendados e devem ser ajustados apenas para casos de uso avançados. Os valores padrão farão o seguinte:
  - Otimize a formatação inicial do volume com base no tipo de destino (como gerenciamento, metadados ou armazenamento), juntamente com a configuração RAID e o tamanho do segmento do volume subjacente.
  - Ajuste a forma como o pacemaker monta cada volume para garantir que as alterações sejam imediatamente lavadas para nós de bloco da série E. Isso evita a perda de dados quando os nós de arquivo falham com gravações ativas em andamento.

    Para obter os parâmetros que podem ser ajustados para suas cargas de trabalho ou casos de uso específicos, consulte o "formatação de volume e parâmetros de configuração de montagem".

3. * Configurações do sistema no sistema operacional Linux instalado nos nós de arquivo.* Você pode substituir as configurações padrão do sistema operacional Linux ao criar o inventário do Ansible na etapa 4 do "Crie o inventário do Ansible".

   As configurações padrão foram usadas para validar a solução BeeGFS no NetApp, mas você pode alterá-las para ajustá-las aos seus workloads ou casos de uso específicos. Alguns exemplos das configurações do sistema operacional Linux que você pode alterar incluem o seguinte:

   - Filas de e/S em dispositivos de bloco e-Series.

     Você pode configurar filas de e/S nos dispositivos de bloco e-Series usados como destinos BeeGFS para:

     - Ajuste o algoritmo de agendamento com base no tipo de dispositivo (NVMe, HDD, etc.).
     - Aumentar o número de pedidos pendentes.
     - Ajuste os tamanhos dos pedidos.
     - Otimizar o comportamento de leitura antecipada.
   - Definições de memória virtual.

     Você pode ajustar as configurações de memória virtual para um desempenho otimizado de streaming contínuo.

   - Definições da CPU.

     Você pode ajustar o regulador de frequência da CPU e outras configurações da CPU para obter o máximo desempenho.

   - Leia o tamanho da solicitação.

     Você pode aumentar o tamanho máximo da solicitação de leitura para HCAs NVIDIA.

Ajuste de desempenho para nós de bloco

Com base nos perfis de configuração aplicados a um componente básico do BeeGFS específico, os grupos de volume configurados nos nós de bloco mudam ligeiramente. Por exemplo, com um nó de bloco de EF600 unidades de 24 unidades:

- Para o componente básico único, incluindo gerenciamento, metadados e serviços de storage do BeeGFS:
  - Grupo de volumes RAID 10, mais de 1x 2 2 vezes, para serviços de gerenciamento e metadados do

BeeGFS

  - Grupos de volumes RAID 6 de 2x 8 2 vezes mais para serviços de storage BeeGFS
- Para um componente básico de storage e metadados do BeeGFS:
  - Grupo de volumes RAID 10, mais de 1x 2 2 vezes, para serviços de metadados BeeGFS
  - Grupos de volumes RAID 6 de 2x 8 2 vezes mais para serviços de storage BeeGFS
- Para o storage BeeGFS apenas componente básico:
  - Grupos de volumes RAID 6 de 2x 10 2 vezes mais para serviços de storage BeeGFS

Como o BeeGFS precisa de muito menos espaço de storage para gerenciamento e metadados em vez de storage, uma opção é usar unidades menores para os grupos de volumes RAID 10. As unidades menores devem ser preenchidas nos slots de unidade mais externos. Para obter mais informações, consulte "instruções de implantação".

Todos eles são configurados pela implantação baseada em Ansible, juntamente com várias outras configurações geralmente recomendadas para otimizar a performance/o comportamento, incluindo:

- Ajustar o tamanho do bloco de cache global para 32KiB e ajustar a descarga de cache baseada na demanda para 80%.
- Desativar o balanceamento automático (garantindo que as atribuições de volume do controlador permaneçam como pretendido).
- Ativar o cache de leitura e desativar o cache de leitura antecipada.
- Ativar o armazenamento em cache de gravação com espelhamento e exigir backup de bateria, para que os caches persistam por falha de um controlador de nó de bloco.
- Especificar a ordem pela qual as unidades são atribuídas a grupos de volume, equilibrando e/S entre os canais de unidade disponíveis.

Componente básico de alta capacidade

O design da solução padrão BeeGFS foi desenvolvido com os workloads de alta performance em mente. Os clientes que procuram casos de uso de alta capacidade devem observar as variações nas caraterísticas de design e desempenho descritas aqui.

Configuração de hardware e software

A configuração de hardware e software para o componente básico de alta capacidade é padrão, exceto que os controladores EF600 devem ser substituídos por EF300 controladores com a opção de anexar entre 1 e 7 bandejas de expansão IOM com 60 unidades cada para cada storage de armazenamento, totalizando 2 a 14 bandejas de expansão por bloco de construção.

Os clientes que implantam um design de componente básico de alta capacidade provavelmente usarão apenas a configuração básica em estilo de componente básico que consiste no gerenciamento, metadados e serviços de storage do BeeGFS para cada nó. Para obter eficiência de custos, os nós de storage de alta capacidade devem provisionar volumes de metadados nas unidades NVMe no compartimento da controladora EF300 e provisionar volumes de storage para as unidades NL-SAS nas bandejas de expansão.

[]

Diretrizes de dimensionamento

Essas diretrizes de dimensionamento presumem que os componentes básicos de alta capacidade são configurados com um grupo de volume SSD NVMe de mais de 2 GB para metadados no compartimento EF300 básico e 2 6x 8 grupos de volume NL-SAS de mais de 2 TB por bandeja de expansão IOM para storage.

Tamanho da unidade (HDDs de capacidade)	Capacidade por BB (1 tabuleiro)	Capacidade por BB (2 bandejas)	Capacidade por BB (3 bandejas)	Capacidade por BB (4 bandejas)
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

Implantar a solução

Visão geral da implantação

É possível implantar o BeeGFS no NetApp para nós de arquivos e blocos validados usando a segunda geração do design de componentes básicos do BeeGFS da NetApp.

Coleções e funções do Ansible

Com o Ansible, você implanta a solução BeeGFS no NetApp, que é um mecanismo popular de automação DE TI usado para automatizar as implantações de aplicações. O Ansible usa uma série de arquivos coletivamente conhecidos como um inventário, que modela o sistema de arquivos BeeGFS que você deseja implantar.

O Ansible permite que empresas como o NetApp expandam a funcionalidade incorporada usando coleções no Ansible Galaxy ("Coleção BeeGFS da NetApp e-Series"consulte). As coleções incluem módulos que executam alguma função ou tarefa específica (como criar um volume e-Series) e incluem funções que podem chamar vários módulos e outras funções. Essa abordagem automatizada reduz o tempo necessário para implantar o sistema de arquivos BeeGFS e o cluster de HA subjacente. Além disso, simplifica a adição de componentes básicos para expandir os sistemas de arquivos existentes.

Para obter mais detalhes, "Saiba mais sobre o inventário do Ansible"consulte .

> Como várias etapas estão envolvidas na implantação da solução BeeGFS no NetApp, o NetApp não oferece suporte para a implantação manual da solução.

Perfis de configuração para blocos de construção BeeGFS

Os procedimentos de implantação abrangem os seguintes perfis de configuração:

- Um componente básico que inclui serviços de gerenciamento, metadados e storage.
- Um segundo componente básico que inclui metadados e serviços de storage.

- Um terceiro componente básico que inclui apenas serviços de storage.

Esses perfis demonstram a gama completa de perfis de configuração recomendados para os blocos de construção BeeGFS do NetApp. Para cada implantação, o número de componentes básicos de storage e metadados ou componentes básicos apenas de serviços de storage pode variar nos procedimentos, dependendo dos requisitos de capacidade e desempenho.

Visão geral das etapas de implantação

A implantação envolve as seguintes tarefas de alto nível:

Implantação de hardware

1. Monte fisicamente cada bloco de construção.
2. Hardware de rack e cabo. Para obter procedimentos detalhados, "Implantar hardware"consulte .

Implantação de software

1. "Configurar nós de arquivo e bloco".
   - Configurar IPs BMC em nós de arquivo
   - Instale um sistema operacional suportado e configure a rede de gerenciamento em nós de arquivos
   - Configurar IPs de gerenciamento em nós de bloco
2. "Configure um nó de controle do Ansible".
3. "Ajuste as configurações do sistema para obter desempenho".
4. "Crie o inventário do Ansible".
5. "Definir o inventário do Ansible para os componentes básicos do BeeGFS".
6. "Implante o BeeGFS com o Ansible".
7. "Configurar clientes BeeGFS".

> Os procedimentos de implantação incluem vários exemplos em que o texto precisa ser copiado para um arquivo. Preste muita atenção a quaisquer comentários inline indicados por carateres "///" para qualquer coisa que deve ou pode ser modificada para uma implantação específica. Por exemplo:
>
> ```
> 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"
> ```

Arquiteturas derivadas com variações nas recomendações de implantação:

- "Bloco de construção de alta capacidade"

Saiba mais sobre o inventário do Ansible

Antes de começar a implantação, entenda como usar o Ansible para configurar e implantar a solução BeeGFS no NetApp usando o design de componentes básicos do BeeGFS de segunda geração.

O inventário do Ansible define a configuração para nós de arquivo e bloco e representa o sistema de arquivos BeeGFS que você deseja implantar. O inventário inclui hosts, grupos e variáveis descrevendo o sistema de arquivos BeeGFS desejado. Os inventários de amostra podem ser baixados de "NetApp e-Series BeeGFS

GitHub".

Módulos e funções do Ansible

Para aplicar a configuração descrita pelo inventário do Ansible, use os vários módulos e funções do Ansible fornecidos na coleção do NetApp e-Series Ansible, em particular a função BeeGFS HA 7,4 (disponível no "NetApp e-Series BeeGFS GitHub") que implanta a solução completa.

Cada função na coleção Ansible do NetApp e-Series é uma implantação completa completa e completa da solução BeeGFS no NetApp. As funções usam as coleções SANtricity, host e BeeGFS do NetApp e-Series que permitem configurar o sistema de arquivos BeeGFS com HA (alta disponibilidade). Em seguida, você pode provisionar e mapear o armazenamento e garantir que o armazenamento de cluster esteja pronto para uso.

Embora a documentação detalhada seja fornecida com as funções, os procedimentos de implantação descrevem como usar a função para implantar uma arquitetura verificada do NetApp usando o design de componentes básicos do BeeGFS de segunda geração.

> Embora as etapas de implantação tentem fornecer detalhes suficientes para que a experiência anterior com o Ansible não seja um pré-requisito, você deve estar familiarizado com o Ansible e com a terminologia relacionada.

Layout de inventário para um cluster BeeGFS HA

Use a estrutura de inventário do Ansible para definir um cluster BeeGFS HA.

Qualquer pessoa com experiência anterior do Ansible deve estar ciente de que a função BeeGFS HA implementa um método personalizado para descobrir quais variáveis (ou fatos) se aplicam a cada host. Isso é necessário para simplificar a criação de um inventário do Ansible que descreve recursos que podem ser executados em vários servidores.

Um inventário do Ansible normalmente consiste nos arquivos no `host_vars` eno e `group_vars` em um `inventory.yml` arquivo que atribui hosts a grupos específicos (e potencialmente grupos a outros grupos).

> Não crie nenhum arquivo com o conteúdo desta subseção, que se destina apenas a exemplo.

Embora essa configuração seja predeterminada com base no perfil de configuração, você deve ter uma compreensão geral de como tudo é definido como um inventário do Ansible, da seguinte forma:

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

Para cada serviço, um arquivo adicional é criado em `group_vars` descrever sua configuração:

```
# meta_01 - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: 8015
  connMetaPortUDP: 8015
  tuneBindToNumaZone: 0
floating_ips:
  - i1b: <IP>/<SUBNET_MASK>
  - i2b: <IP>/<SUBNET_MASK>
# Type of BeeGFS service the HA resource group will manage.
beegfs_service: metadata # Choices: management, metadata, storage.
# What block node should be used to create a volume for this service:
beegfs_targets:
  netapp01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.25
            owning_controller: A
```

Esse layout permite que a configuração de serviço, rede e storage do BeeGFS para cada recurso seja definida em um único lugar. Nos bastidores, a função BeeGFS agrega a configuração necessária para cada nó de arquivo e bloco com base nessa estrutura de inventário. Para obter mais informações, consulte este post no blog: "O NetApp acelera a implantação do HA para BeeGFS com o Ansible".

> O ID numérico e do nó de string BeeGFS para cada serviço é configurado automaticamente com base no nome do grupo. Assim, além do requisito geral do Ansible para que os nomes de grupos sejam únicos, os grupos que representam um serviço BeeGFS precisam terminar em um número exclusivo para o tipo de serviço BeeGFS que o grupo representa. Por exemplo, meta_01 e stor_01 são permitidos, mas metadata_01 e meta_01 não são.

Reveja as práticas recomendadas

Siga as diretrizes de práticas recomendadas ao implantar a solução BeeGFS no NetApp.

Convenções padrão

Ao montar e criar fisicamente o arquivo de inventário do Ansible, siga estas convenções padrão (para obter mais informações, "Crie o inventário do Ansible"consulte).

- Os nomes de host de nó de arquivo são numerados sequencialmente (H01-HN) com números menores na parte superior do rack e números mais altos na parte inferior.

  Por exemplo, a convenção de nomenclatura `[location][row][rack]hN` é semelhante a: `beegfs_01`.

- Cada nó de bloco é composto por duas controladoras de storage, cada uma com seu próprio nome de host.

  O nome de um storage array é usado para referir-se a todo o sistema de storage de bloco como parte de um inventário do Ansible. Os nomes dos storage arrays devem ser numerados sequencialmente (A01 - an), e os nomes dos hosts para controladores individuais são derivados dessa convenção de nomenclatura.

  Por exemplo, um nó de bloco chamado `ictad22a01` normalmente pode ter nomes de host configurados para cada controlador, como `ictad22a01-a` e `ictad22a01-b`, mas ser referido em um inventário do Ansible como `netapp_01`.

- Nós de arquivo e bloco dentro do mesmo bloco de construção compartilham o mesmo esquema de numeração e são adjacentes um ao outro no rack com ambos os nós de arquivo na parte superior e ambos os nós de bloco diretamente abaixo deles.

  Por exemplo, no primeiro componente básico, os nós de arquivo H01 e H02 são conetados diretamente aos nós de bloco A01 e A02. De cima para baixo, os nomes de host são H01, H02, A01 e A02.

- Os blocos de construção são instalados em ordem sequencial com base nos nomes de host, de modo que os nomes de host de número inferior estejam na parte superior do rack e os nomes de host de número superior estejam na parte inferior.

  O objetivo é minimizar o comprimento do cabo que corre até a parte superior dos switches de rack e definir uma prática de implantação padrão para simplificar a solução de problemas. Para datacenters onde isso não é permitido devido a preocupações em torno da estabilidade do rack, o inverso é certamente permitido, preenchendo o rack de baixo para cima.

Configuração de rede de storage InfiniBand

Metade das portas InfiniBand em cada nó de arquivo é usada para se conectar diretamente aos nós de bloco. A outra metade está conetada aos switches InfiniBand e é usada para a conetividade cliente-servidor BeeGFS. Ao determinar o tamanho das sub-redes IPoIB usadas para clientes e servidores BeeGFS, você deve considerar o crescimento esperado do cluster de computação/GPU e do sistema de arquivos BeeGFS. Se você tiver que se desviar dos intervalos de IP recomendados, lembre-se de que cada conexão direta em um único bloco de construção tem uma sub-rede exclusiva e não há sobreposição com sub-redes usadas para conetividade cliente-servidor.

Ligações diretas

Os nós de arquivo e bloco em cada bloco de construção sempre usam os IPs na tabela a seguir para suas conexões diretas.

> Este esquema de endereçamento segue a seguinte regra: O terceiro octeto é sempre ímpar ou par, o que depende se o nó do arquivo é ímpar ou par.

Nó de arquivo	Porta de IB	Endereço IP	Nó de bloco	Porta de IB	IP físico	IP virtual
Bola de Futsal (H1)	i1a	192.168.1.10	Bola de Futsal (C1)	2a	192.168.1.100	192.168.1.101
Bola de Futsal (H1)	i2a	192.168.3.10	Bola de Futsal (C1)	2a	192.168.3.100	192.168.3.101

Nó de arquivo	Porta de IB	Endereço IP	Nó de bloco	Porta de IB	IP físico	IP virtual
Bola de Futsal (H1)	i3a	192.168.5.10	Kit de meia (C2)	2a	192.168.5.100	192.168.5.101
Bola de Futsal (H1)	i4a	192.168.7.10	Kit de meia (C2)	2a	192.168.7.100	192.168.7.101
Kit de meia (H2)	i1a	192.168.2.10	Bola de Futsal (C1)	2b	192.168.2.100	192.168.2.101
Kit de meia (H2)	i2a	192.168.4.10	Bola de Futsal (C1)	2b	192.168.4.100	192.168.4.101
Kit de meia (H2)	i3a	192.168.6.10	Kit de meia (C2)	2b	192.168.6.100	192.168.6.101
Kit de meia (H2)	i4a	192.168.8.10	Kit de meia (C2)	2b	192.168.8.100	192.168.8.101

Esquemas de endereçamento IPoIB cliente-servidor BeeGFS

Cada nó de arquivo executa vários serviços de servidor do BeeGFS (gerenciamento, metadados ou storage). Para permitir que cada serviço faça failover independentemente para o outro nó de arquivo, cada um é configurado com endereços IP exclusivos que podem flutuar entre ambos os nós (às vezes chamado de interface lógica ou LIF).

Embora não seja obrigatório, essa implantação presume que os seguintes intervalos de sub-rede IPoIB estão em uso para essas conexões e define um esquema de endereçamento padrão que aplica as seguintes regras:

- O segundo octeto é sempre ímpar ou par, com base se a porta InfiniBand do nó de arquivo é ímpar ou par.
- Os IPs de cluster do BeeGFS são sempre `xxx. 127.100.yyy` ou `xxx.128.100.yyy`.

> Além da interface usada para o gerenciamento de SO na banda, interfaces adicionais podem ser usadas pelo Corosync para batimento cardíaco de cluster e sincronização. Isso garante que a perda de uma única interface não derrube todo o cluster.

- O serviço BeeGFS Management está sempre em `xxx.yyy.101.0` ou `xxx.yyy.102.0`.
- Os serviços de metadados BeeGFS estão sempre em `xxx.yyy.101.zzz` ou `xxx.yyy.102.zzz`.
- Os serviços do BeeGFS Storage estão sempre na `xxx.yyy.103.zzz` ou `xxx.yyy.104.zzz`.
- Os endereços no `100.xxx.1.1` intervalo até `100.xxx.99.255` são reservados para os clientes.

Esquema de endereçamento de sub-rede única IPoIB

Este guia de implantação utilizará um único esquema de sub-rede, dadas as vantagens listadas "arquitetura de software"no .

Sub-rede: 100.127.0.0/16

A tabela a seguir fornece o intervalo para uma única sub-rede: 100.127.0.0/16.

Finalidade	Porta InfiniBand	Endereço IP ou intervalo
IP do cluster do BeeGFS	i1b ou i4b	100.127.100.1 - 100.127.100.255
Gestão BeeGFS	i1b	100.127.101.0
	i2b	100.127.102.0
Metadados BeeGFS	i1b ou i3b	100.127.101.1 - 100.127.101.255
	i2b ou i4b	100.127.102.1 - 100.127.102.255
Storage BeeGFS	i1b ou i3b	100.127.103.1 - 100.127.103.255
	i2b ou i4b	100.127.104.1 - 100.127.104.255
Clientes BeeGFS	(varia de acordo com o cliente)	100.127.1.1 - 100.127.99.255

Esquema de endereçamento de sub-rede IPoIB dois

Um esquema de endereçamento de duas sub-redes não é mais recomendado, mas ainda pode ser implementado. Consulte as tabelas abaixo para obter um esquema recomendado de duas sub-redes.

Sub-rede A: 100.127.0.0/16

A tabela a seguir fornece o intervalo para a sub-rede A: 100.127.0.0/16.

Finalidade	Porta InfiniBand	Endereço IP ou intervalo
IP do cluster do BeeGFS	i1b	100.127.100.1 - 100.127.100.255
Gestão BeeGFS	i1b	100.127.101.0
Metadados BeeGFS	i1b ou i3b	100.127.101.1 - 100.127.101.255
Storage BeeGFS	i1b ou i3b	100.127.103.1 - 100.127.103.255
Clientes BeeGFS	(varia de acordo com o cliente)	100.127.1.1 - 100.127.99.255

Sub-rede B: 100.128.0.0/16

A tabela a seguir fornece o intervalo para a sub-rede B: 100.128.0.0/16.

Finalidade	Porta InfiniBand	Endereço IP ou intervalo
IP do cluster do BeeGFS	i4b	100.128.100.1 - 100.128.100.255
Gestão BeeGFS	i2b	100.128.102.0
Metadados BeeGFS	i2b ou i4b	100.128.102.1 - 100.128.102.255
Storage BeeGFS	i2b ou i4b	100.128.104.1 - 100.128.104.255
Clientes BeeGFS	(varia de acordo com o cliente)	100.128.1.1 - 100.128.99.255

Nem todos os IPs nos intervalos acima são usados nesta arquitetura verificada do NetApp. Eles demonstram como os endereços IP podem ser pré-alocados para permitir uma fácil expansão do sistema de arquivos usando um esquema de endereçamento IP consistente. Nesse esquema, os nós de arquivo BeeGFS e as IDs de serviço correspondem ao quarto octeto de um intervalo bem conhecido de IPs. O sistema de arquivos certamente pode ser dimensionado além de 255 nós ou serviços, se necessário.

Implantar hardware

Cada componente básico consiste em dois nós de arquivo x86 validados diretamente conectados a dois nós de bloco usando cabos InfiniBand HDR (200GB).

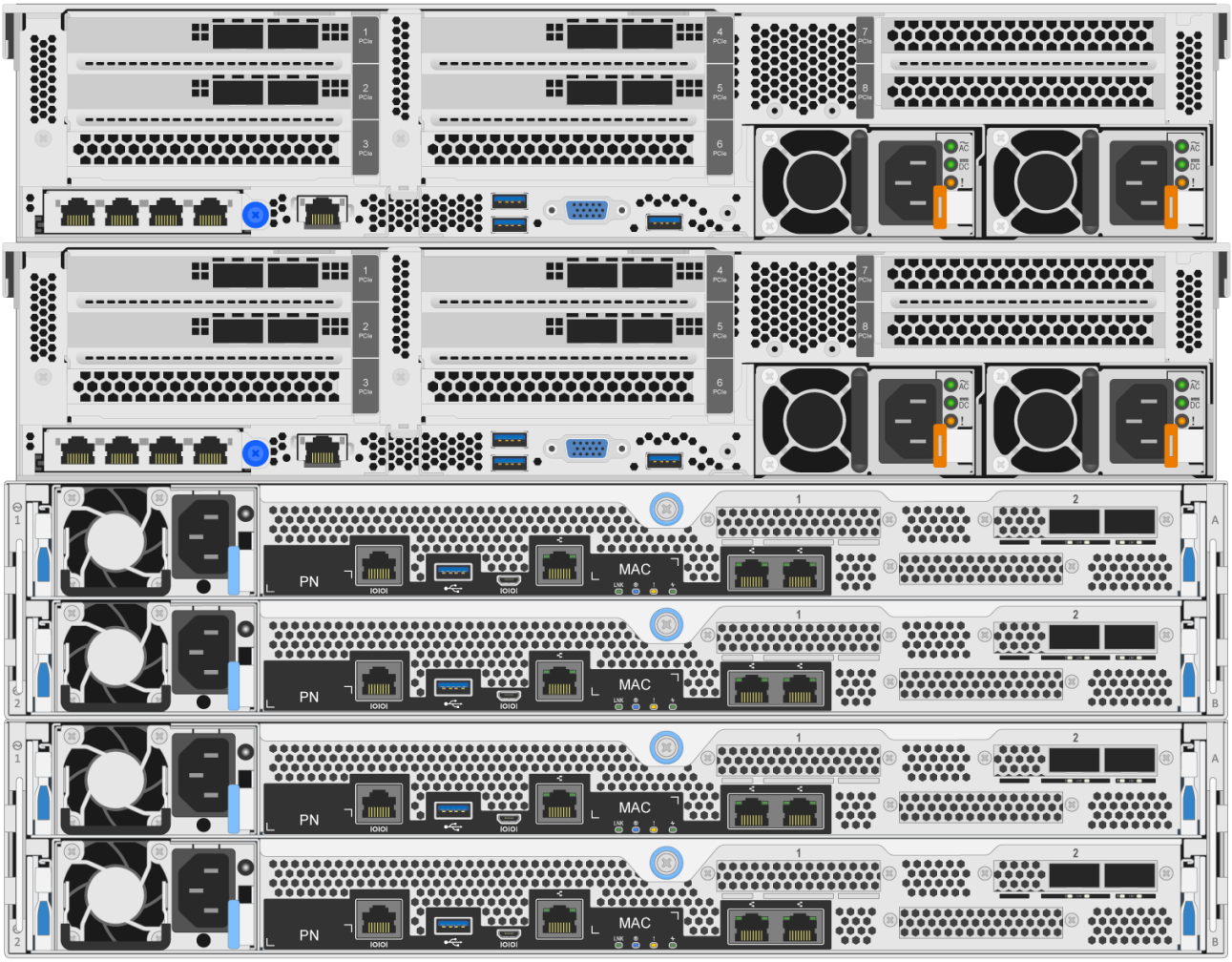
> É necessário um mínimo de dois componentes básicos para estabelecer quorum no cluster de failover. Um cluster de dois nós tem limitações que podem impedir que ocorra um failover bem-sucedido. Você pode configurar um cluster de dois nós incorporando um terceiro dispositivo como um tiebreaker. No entanto, esta documentação não descreve esse design.

As etapas a seguir são idênticas para cada componente básico no cluster, independentemente de ser usado para executar serviços de storage e metadados do BeeGFS, ou apenas serviços de storage, a menos que seja indicado de outra forma.

Passos

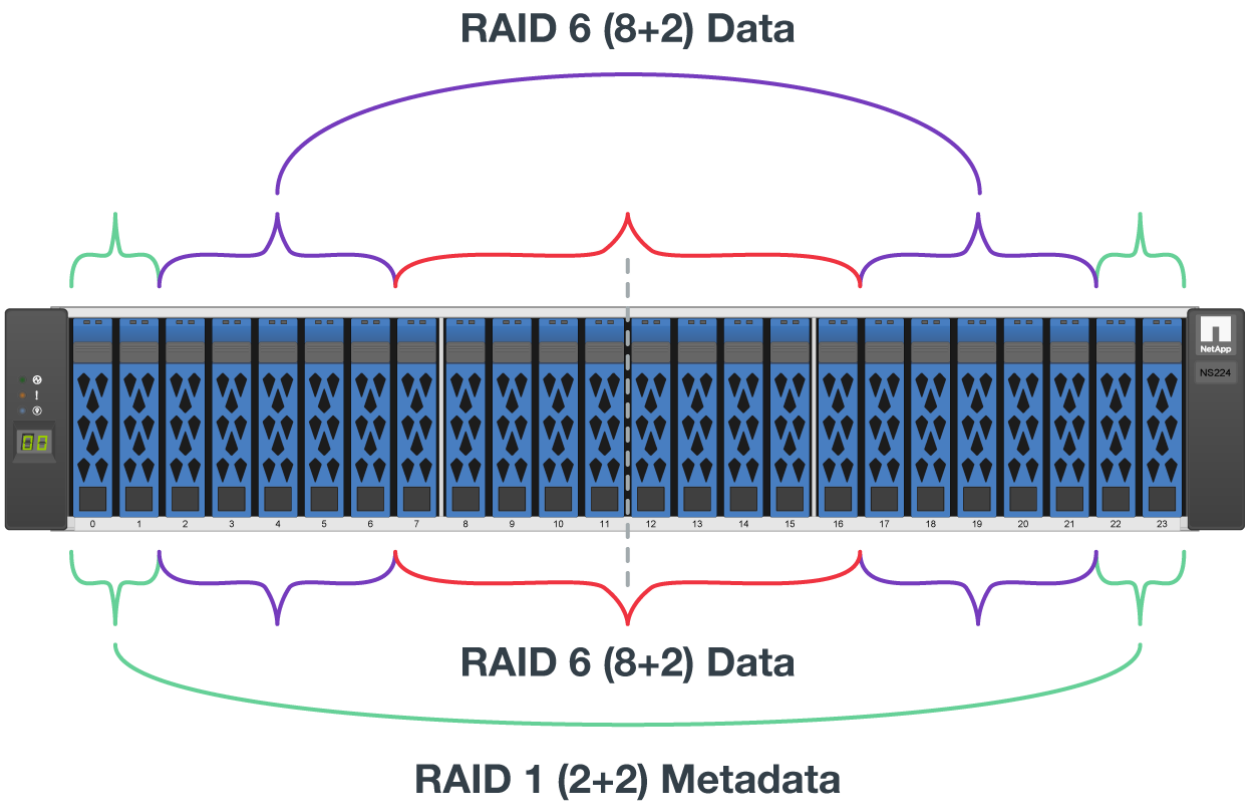
1. Configure cada nó de arquivo BeeGFS com quatro adaptadores de canal de host (HCAs) usando os modelos especificados no "Requisitos técnicos". Insira os HCAs nos slots PCIe do nó de arquivo de acordo com as especificações abaixo:
   - **Servidor Lenovo ThinkSystem SR665 V3:** Use slots PCIe 1, 2, 4 e 5.
   - **Servidor Lenovo ThinkSystem SR665:** Use slots PCIe 2, 3, 5 e 6.
2. Configure cada nó de bloco BeeGFS com uma placa de interface de host (HIC) 200GB de porta dupla e instale o HIC em cada uma de suas duas controladoras de storage.

   Agrupe os componentes básicos para que os dois nós de arquivo BeeGFS fiquem acima dos nós de bloco BeeGFS. A figura a seguir mostra a configuração correta de hardware para o componente básico BeeGFS usando servidores Lenovo ThinkSystem SR665 V3 como nós de arquivo (visão traseira).

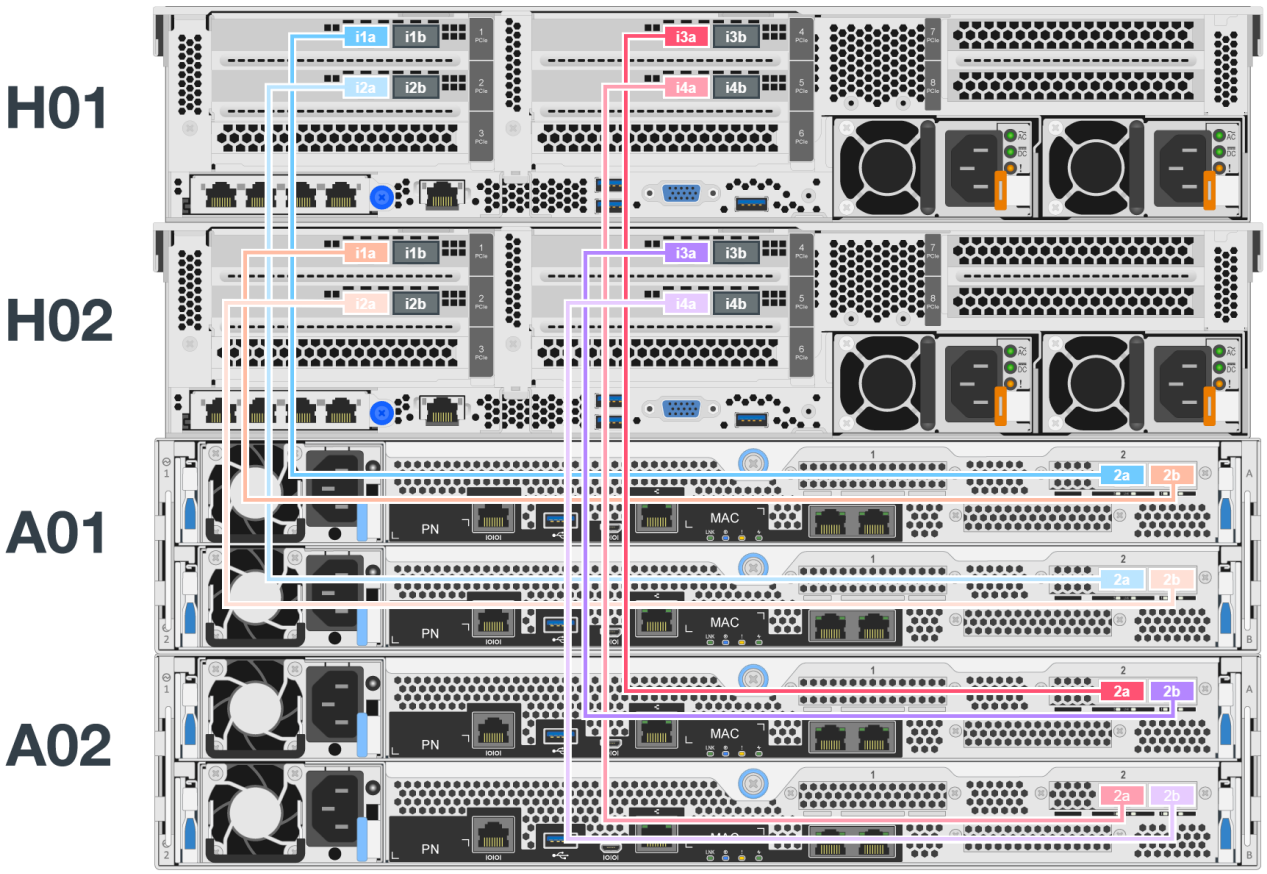


A configuração da fonte de alimentação para casos de uso de produção normalmente deve usar PSUs redundantes.

3. Se necessário, instale as unidades em cada um dos nós de bloco BeeGFS.
   a. Se o componente básico for usado para executar metadados e serviços de storage do BeeGFS e unidades menores forem usados para volumes de metadados, verifique se eles estão preenchidos nos slots de unidade mais externos, como mostra a figura abaixo.
   b. Para todas as configurações de componentes básicos, se um compartimento de unidade não estiver totalmente preenchido, certifique-se de que um número igual de unidades esteja preenchido nos slots 0–11 e 12–23 para obter um desempenho ideal.



4. Conete os nós de bloco e arquivo usando o "Cabos de cobre de conexão direta InfiniBand HDR 200GB de 1m GB", de modo que correspondam à topologia mostrada na figura a seguir.

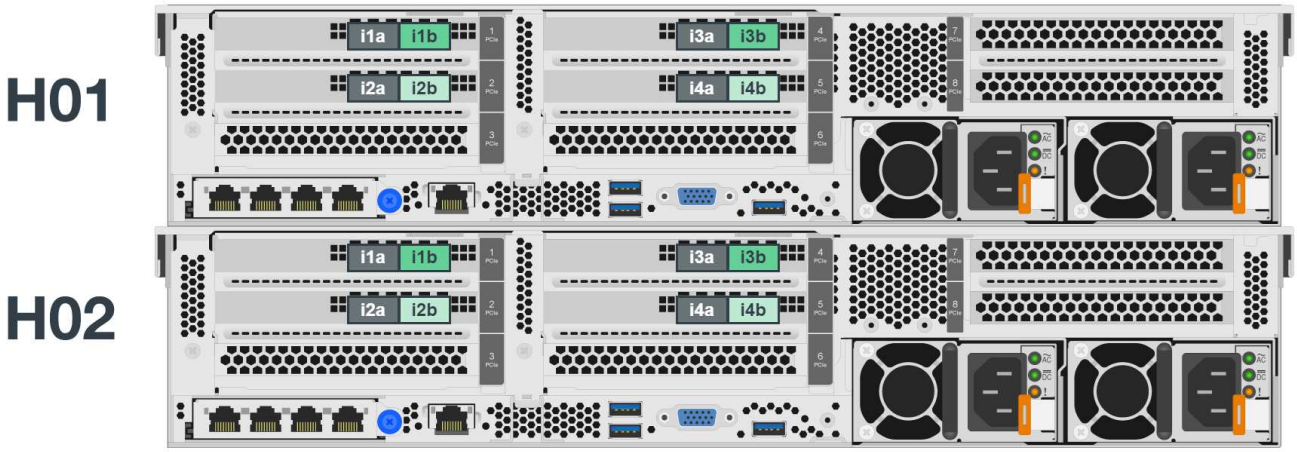


Os nós em vários componentes básicos nunca são conetados diretamente. Cada bloco de construção deve ser Tratado como uma unidade autônoma e toda a comunicação entre blocos de construção ocorre através de switches de rede.

5. Conete as portas InfiniBand restantes no nó de arquivo ao switch InfiniBand da rede de storage usando o "Cabos InfiniBand de 2M GB" específico do switch de storage InfiniBand.

   Ao usar cabos divisores para conetar o switch de armazenamento a nós de arquivo, um cabo deve se ramificar para fora do switch e se conetar às portas delineadas em verde claro. Outro cabo divisor deve ser ramificado para fora do switch e conetar-se às portas delineadas em verde escuro.

   Além disso, para redes de armazenamento com switches redundantes, as portas delineadas em verde claro devem se conetar a um switch, enquanto as portas em verde escuro devem se conetar a outro switch.



6. Conforme necessário, monte blocos de construção adicionais seguindo as mesmas diretrizes de cabeamento.

O número total de componentes básicos que podem ser implantados em um único rack depende da energia e refrigeração disponíveis em cada local.

Implantar software

Configurar nós de arquivo e nós de bloco

Embora a maioria das tarefas de configuração de software seja automatizada com as coleções do Ansible fornecidas pela NetApp, é necessário configurar a rede na controladora de gerenciamento de placa base (BMC) de cada servidor e configurar a porta de gerenciamento em cada controladora.

Configurar nós de arquivo

1. Configure a rede no controlador de gerenciamento de placa base (BMC) de cada servidor.

   Para saber como configurar a rede para os nós de arquivo validados do Lenovo SR665 V3, consulte o "Documentação do Lenovo ThinkSystem".

Um controlador de gerenciamento de placa base (BMC), às vezes chamado de processador de serviço, é o nome genérico para o recurso de gerenciamento fora da banda incorporado em várias plataformas de servidor que podem fornecer acesso remoto, mesmo que o sistema operacional não esteja instalado ou acessível. Normalmente, os fornecedores comercializam essa funcionalidade com sua própria marca. Por exemplo, no Lenovo SR665, o BMC é chamado de *controlador XClarity (XCC)*.

2. Configure as definições do sistema para obter o máximo desempenho.

   Você configura as configurações do sistema usando a configuração UEFI (anteriormente conhecida como BIOS) ou usando as APIs do Redfish fornecidas por muitos BMCs. As configurações do sistema variam de acordo com o modelo de servidor usado como nó de arquivo.

   Para saber como configurar as configurações do sistema para os nós de arquivo validados do Lenovo SR665, "Ajuste as configurações do sistema para obter desempenho"consulte .

3. Instale o Red Hat 9,3 e configure o nome do host e a porta de rede usados para gerenciar o sistema operacional, incluindo a conetividade SSH do nó de controle do Ansible.

   Não configure IPs em nenhuma das portas InfiniBand no momento.

   Embora não sejam estritamente necessárias, as seções subsequentes presumem que os nomes de host são numerados sequencialmente (como H1-HN) e referem-se a tarefas que devem ser concluídas em hosts ímpares versus mesmo numerados.

4. Use o RedHat Subscription Manager para Registrar e assinar o sistema para permitir a instalação dos pacotes necessários dos repositórios oficiais da Red Hat e limitar as atualizações da versão suportada do Red Hat: `subscription-manager release --set=9.3`. Para obter instruções, consulte "Como Registrar e assinar um sistema RHEL" e "Como limitar as atualizações".
5. Ative o repositório Red Hat que contém os pacotes necessários para alta disponibilidade.

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

6. Atualize todo o firmware HCA para a versão recomendada em "Requisitos de tecnologia".

   Esta atualização pode ser feita baixando e executando uma versão da ferramenta mlxup que agrupa o firmware recomendado. Pode transferir esta ferramenta a partir de "Mlxup - Utilitário de atualização e consulta" ("manual do utilizador").

Configurar nós de bloco

Configure os nós de bloco EF600 configurando a porta de gerenciamento em cada controlador.

1. Configure a porta de gerenciamento em cada controlador EF600.

   Para obter instruções sobre como configurar portas, vá para o "E-Series Documentation Center".

2. Opcionalmente, defina o nome do storage array para cada sistema.

   Definir um nome pode facilitar a consulta a cada sistema nas seções subsequentes. Para obter instruções

sobre como definir o nome do array, vá para "E-Series Documentation Center" .

Embora não sejam estritamente necessários, os tópicos subsequentes presumem que os nomes de matrizes de armazenamento são numerados sequencialmente (como C1 - CN) e referem-se às etapas que devem ser concluídas em sistemas ímpares versus pares.

Ajuste as configurações do sistema do nó de arquivo para obter desempenho

Para maximizar o desempenho, recomendamos configurar as configurações do sistema no modelo de servidor que você usa como nós de arquivo.

As configurações do sistema variam dependendo do modelo de servidor que você usa como nó de arquivo. Este tópico descreve como configurar as definições do sistema para os nós de ficheiro de servidor Lenovo ThinkSystem SR665 validados.

Utilize a interface UEFI para ajustar as definições do sistema

O firmware do sistema do servidor Lenovo SR665 contém vários parâmetros de ajuste que podem ser definidos através da interface UEFI. Esses parâmetros de ajuste podem afetar todos os aspetos de como o servidor funciona e o desempenho do servidor.

Em **Configuração UEFI > Definições do sistema**, ajuste as seguintes definições do sistema:

Menu modo de funcionamento

Configuração do sistema	Mude para
Modo de funcionamento	Personalizado
CTDP	Manual
Manual do cTDP	350
Limite de potência do pacote	Manual
Modo de eficiência	Desativar
Controle Global-Cstate	Desativar
estados SOC P	P0
DF C-Estados	Desativar
P-State	Desativar
Ativação da desativação da memória	Desativar
NUMA NODES por socket	NPS1

Menu dispositivos e portas de e/S.

Configuração do sistema	Mude para
IOMMU	Desativar

Menu de alimentação

Configuração do sistema	Mude para
Travão de potência PCIe	Desativar

Menu processadores

Configuração do sistema	Mude para
Controlo global do estado C	Desativar
DF C-Estados	Desativar
Modo SMT	Desativar
CPPC	Desativar

Use Redfish API para ajustar as configurações do sistema

Além de usar a Configuração UEFI, você pode usar a API Redfish para alterar as configurações do sistema.

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

Para obter informações detalhadas sobre o esquema do Red Fish, consulte "Website da DMTF" .

Configure um nó de controle do Ansible

Para configurar um nó de controle do Ansible, você precisa identificar uma máquina virtual ou física com acesso de rede às portas de gerenciamento de todos os nós de arquivo e bloco que podem ser usados para configurar a solução.

As etapas a seguir foram testadas no Ubuntu 22,04.04. Para obter passos específicos para a distribuição Linux preferida, consulte "Documentação do Ansible" .

1. Instale o Python 3,10 e certifique-se de que a versão correta do `pip` está instalada.

```
sudo apt install python3.10 -y
sudo apt install python3-pip
sudo apt install sshpass
```

2. Crie links simbólicos, garantindo que o binário Python 3,10 seja usado sempre que `python3` ou `python` seja chamado.

```
sudo ln -sf /usr/bin/python3.10 /usr/bin/python3
sudo ln -sf /usr/bin/python3 /usr/bin/python
```

3. Instale os pacotes Python exigidos pelas coleções BeeGFS do NetApp.

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

> Para garantir que você esteja instalando uma versão compatível do Ansible e todos os pacotes Python necessários, consulte o arquivo Readme da coleção BeeGFS. As versões suportadas também são anotadas no "Requisitos técnicos".

4. Verifique se as versões corretas do Ansible e do Python estão instaladas.

```
ansible --version
ansible [core 2.17.2]
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.10/dist-
packages/ansible
  ansible collection location =
/root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0]
(/usr/bin/python3)
  jinja version = 3.1.4
  libyaml = True
```

5. Armazene os inventários Ansible usados para descrever a implantação do BeeGFS em sistemas de controle de origem, como Git ou BitBucket, e instale o Git para interagir com esses sistemas.

```
sudo apt install git -y
```

6. Configure o SSH sem senha. Essa é a maneira mais fácil de permitir que o Ansible acesse os nós de arquivo BeeGFS remotos a partir do nó de controle do Ansible.
   a. No nó de controle do Ansible, se necessário, gere um par de chaves públicas usando `ssh-keygen`
   b. Configure o SSH sem senha para cada um dos nós de arquivo usando `ssh-copy-id <ip_or_hostname>`

      Not configure SSH sem senha para os nós de bloco. Isto não é suportado nem necessário.

7. Use o Ansible Galaxy para instalar a versão da coleção BeeGFS listada no "Requisitos técnicos".

Essa instalação inclui dependências adicionais do Ansible, como o software NetApp SANtricity e as coleções de host.

```
ansible-galaxy collection install netapp_eseries.beegfs:==3.2.0
```

Crie o inventário do Ansible

Para definir a configuração de nós de arquivo e bloco, crie um inventário do Ansible que represente o sistema de arquivos BeeGFS que você deseja implantar. O inventário inclui hosts, grupos e variáveis descrevendo o sistema de arquivos BeeGFS desejado.

Passo 1: Defina a configuração para todos os blocos de construção

Defina a configuração que se aplica a todos os blocos de construção, independentemente do perfil de configuração que você possa aplicar a eles individualmente.

Antes de começar

- Escolha um esquema de endereçamento de sub-rede para sua implantação. Devido aos benefícios listados no "arquitetura de software", recomenda-se usar um único esquema de endereçamento de sub-rede.

Passos

1. No nó de controle do Ansible, identifique um diretório que você deseja usar para armazenar os arquivos de estratégia e inventário do Ansible.

   Salvo indicação em contrário, todos os arquivos e diretórios criados nesta etapa e as etapas a seguir são criados em relação a este diretório.

2. Crie os seguintes subdiretórios:

   `host_vars`

   `group_vars`

   `packages`

Etapa 2: Defina a configuração para nós individuais de arquivo e bloco

Defina a configuração que se aplica a nós de arquivo individuais e nós de bloco de construção individuais.

1. Em `host_vars/`, crie um arquivo para cada nó de arquivo BeeGFS nomeado `<HOSTNAME>.yml` com o seguinte conteúdo, prestando especial atenção às notas sobre o conteúdo a serem preenchidas para IPs de cluster BeeGFS e nomes de host que terminam em números ímpares versus pares.

   Inicialmente, os nomes da interface do nó de arquivo correspondem ao que está listado aqui (como ib0 ou ibs1f0). Esses nomes personalizados são configurados no Etapa 4: Defina a configuração que deve ser aplicada a todos os nós de arquivo.

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

Se você já implantou o cluster BeeGFS, será necessário interromper o cluster antes de adicionar ou alterar endereços IP configurados estaticamente, incluindo IPs e IPs do cluster usados para NVMe/IB. Isso é necessário para que essas alterações entrem em vigor corretamente e não interrompam as operações do cluster.

2. Em `host_vars/`, crie um arquivo para cada nó de bloco BeeGFS nomeado `<HOSTNAME>.yml` e o preencha com o seguinte conteúdo.

   Preste atenção especial às notas sobre o conteúdo a ser preenchido para nomes de storage arrays que terminam em números ímpares versus pares.

   Para cada nó de bloco, crie um arquivo e especifique o `<MANAGEMENT_IP>` para um dos dois controladores (geralmente 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
```

Etapa 3: Defina a configuração que deve ser aplicada a todos os nós de arquivo e bloco

Você pode definir a configuração comum a um grupo de hosts em em `group_vars` um nome de arquivo que corresponde ao grupo. Isso impede a repetição de uma configuração compartilhada em vários locais.

Sobre esta tarefa

Os hosts podem estar em mais de um grupo e, em tempo de execução, o Ansible escolhe quais variáveis se aplicam a um host específico com base em suas regras de precedência de variáveis. (Para obter mais informações sobre essas regras, consulte a documentação do Ansible para "Usando variáveis".)

As atribuições de host para grupo são definidas no arquivo de inventário real do Ansible, que é criado no final deste procedimento.

Passo

No Ansible, qualquer configuração que você deseja aplicar a todos os hosts pode ser definida em um grupo `All` chamado . Crie o arquivo `group_vars/all.yml` com o seguinte conteúdo:

```
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"
```

Etapa 4: Defina a configuração que deve ser aplicada a todos os nós de arquivo

A configuração compartilhada para nós de arquivo é definida em um grupo `ha_cluster` chamado . As etapas nesta seção compilam a configuração que deve ser incluída no `group_vars/ha_cluster.yml` arquivo.

Passos

1. Na parte superior do arquivo, defina os padrões, incluindo a senha a ser usada como `sudo` usuário nos nós de arquivo.

```
### 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 for single subnet
deployments, where static IPs containing multiple IB ports are in the
same IPoIB subnet. i.e: cluster IPs, multirail, single subnet, etc.
# If the following options are specified, then Ansible will
automatically reboot nodes when necessary for changes to take effect:
eseries_common_allow_host_reboot: true
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
eseries_ib_opensm_options:
  virt_enabled: "2"
  virt_max_ports_in_process: "0"
```

Particularmente para ambientes de produção, não armazene senhas em texto simples. Em vez disso, use o Ansible Vault ("Criptografia de conteúdo com o Ansible Vault"consulte) ou a `--ask-become-pass` opção ao executar o manual de estratégia. Se o `ansible_ssh_user` já estiver `root` , você poderá omitir opcionalmente o `ansible_become_password`.

2. Opcionalmente, configure um nome para o cluster de high-availability (HA) e especifique um utilizador para comunicação intra-cluster.

   Se você estiver modificando o esquema de endereçamento IP privado, também deverá atualizar o padrão `beegfs_ha_mgmtd_floating_ip`. Isso deve corresponder ao que você configurar mais tarde para o grupo de recursos do BeeGFS Management.

   Especifique um ou mais e-mails que devem receber alertas para eventos de cluster usando `beegfs_ha_alert_email_list`o .

```
### 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
```

Embora pareça redundante, `beegfs_ha_mgmtd_floating_ip` é importante quando você escala o sistema de arquivos BeeGFS além de um único cluster de HA. Os clusters de HA subsequentes são implantados sem um serviço de gerenciamento BeeGFS adicional e apontam para o serviço de gerenciamento fornecido pelo primeiro cluster.

3. Configurar um agente de vedação. (Para obter mais detalhes, "Configure o esgrima em um cluster Red Hat High Availability" consulte .) A saída a seguir mostra exemplos para a configuração de agentes de vedação comuns. Escolha uma destas opções.

   Para esta etapa, esteja ciente de que:

   - Por padrão, o esgrima está habilitado, mas você precisa configurar um *agente* de esgrima.

- O `<HOSTNAME>` especificado no `pcmk_host_map` ou `pcmk_host_list` deve corresponder ao nome do host no inventário do Ansible.
- A execução do cluster BeeGFS sem cercas não é suportada, especialmente na produção. Isso é em grande parte para garantir quando os serviços BeeGFS, incluindo quaisquer dependências de recursos, como dispositivos de bloco, fazem failover devido a um problema, não há risco de acesso simultâneo por vários nós que resultam em corrupção do sistema de arquivos ou outro comportamento indesejável ou inesperado. Se o esgrima tiver de ser desativado, consulte as notas gerais no guia de introdução da função BeeGFS HA e defina `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` como false no `ha_cluster.yml`.
- Há vários dispositivos de esgrima no nível do nó disponíveis e a função BeeGFS HA pode configurar qualquer agente de esgrima disponível no repositório de pacotes Red Hat HA. Quando possível, use um agente de vedação que funcione através da fonte de alimentação ininterrupta (UPS) ou da unidade de distribuição de energia em rack (rPDU), porque alguns agentes de vedação, como o controlador de gerenciamento de placa base (BMC) ou outros dispositivos de iluminação integrados no servidor, podem não responder à solicitação de vedação sob certos cenários de falha.

```
### 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/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Ative o ajuste de desempenho recomendado no sistema operacional Linux.

   Embora muitos usuários encontrem as configurações padrão para os parâmetros de desempenho geralmente funcionem bem, você pode opcionalmente alterar as configurações padrão para uma determinada carga de trabalho. Como tal, essas recomendações são incluídas na função BeeGFS, mas não são habilitadas por padrão para garantir que os usuários estejam cientes do ajuste aplicado ao sistema de arquivos.

   Para ativar o ajuste de desempenho, especifique:

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

5. (Opcional) você pode ajustar os parâmetros de ajuste de desempenho no sistema operacional Linux conforme necessário.

Para obter uma lista abrangente dos parâmetros de ajuste disponíveis que você pode ajustar, consulte a seção padrões de ajuste de desempenho da função de HA BeeGFS em "Site do e-Series BeeGFS GitHub". os valores padrão podem ser substituídos para todos os nós do cluster neste arquivo ou `host_vars` para um nó individual.

6. Para permitir a conetividade 200GBK/HDR completa entre nós de bloco e arquivo, use o pacote Open Subnet Manager (OpenSM) da NVIDIA Open Fabrics Enterprise Distribution (MLNX_OFED). A versão MLNX_OFED na lista "requisitos de nó de arquivo"vem junto com os pacotes OpenSM recomendados. Embora a implantação usando Ansible seja compatível, primeiro você precisa instalar o driver MLNX_OFED em todos os nós de arquivo.

   a. Preencha os seguintes parâmetros em `group_vars/ha_cluster.yml` (ajuste pacotes conforme necessário):

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

7. Configure a `udev` regra para garantir o mapeamento consistente de identificadores de porta InfiniBand lógicos para dispositivos PCIe subjacentes.

   A `udev` regra deve ser exclusiva da topologia PCIe de cada plataforma de servidor usada como nó de arquivo BeeGFS.

   Use os seguintes valores para nós de arquivo verificados:

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

# OPTION 2: Lenovo SR665 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:41:00.0": i1a
  "0000:41:00.1": i1b
  "0000:01:00.0": i2a
  "0000:01:00.1": i2b
  "0000:a1:00.0": i3a
  "0000:a1:00.1": i3b
  "0000:81:00.0": i4a
  "0000:81:00.1": i4b
```

8. (Opcional) Atualize o algoritmo de seleção de destino de metadados.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

> ⓘ No teste de verificação, `randomrobin` o era normalmente usado para garantir que os arquivos de teste fossem distribuídos uniformemente por todos os destinos de storage do BeeGFS durante o benchmark de desempenho (para obter mais informações sobre benchmarking, consulte o site BeeGFS para "Benchmarking de um sistema BeeGFS"). Com o uso do mundo real, isso pode fazer com que alvos com números mais baixos preencham mais rápido do que alvos com números mais altos. Omitir e `randomrobin` apenas usar o valor padrão `randomized` foi mostrado para fornecer bom desempenho enquanto ainda utiliza todos os alvos disponíveis.

Etapa 5: Defina a configuração para o nó de bloco comum

A configuração compartilhada para nós de bloco é definida em um grupo `eseries_storage_systems` chamado . As etapas nesta seção compilam a configuração que deve ser incluída no `group_vars/eseries_storage_systems.yml` arquivo.

Passos

1. Defina a conexão Ansible como local, forneça a senha do sistema e especifique se os certificados SSL

devem ser verificados. (Normalmente, o Ansible usa SSH para se conectar a hosts gerenciados. No entanto, no caso dos sistemas de storage do NetApp e-Series usados como nós de bloco, os módulos usam a API REST para comunicação.) Na parte superior do arquivo, adicione o seguinte:

```
### 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
```

Listar senhas em texto simples não é recomendado. Use o cofre do Ansible ou forneça o `eseries_system_password` ao executar o Ansible `--extra-vars` usando o .

2. Para garantir o desempenho ideal, instale as versões listadas para nós de bloco "Requisitos técnicos"no .

   Transfira os ficheiros correspondentes a partir do "Site de suporte da NetApp". Você pode atualizá-los manualmente ou incluí-los `packages/` no diretório do nó de controle do Ansible e preencher os seguintes parâmetros `eseries_storage_systems.yml` para atualizar usando o Ansible:

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

3. Transfira e instale o firmware de unidade mais recente disponível para as unidades instaladas nos nós de bloco a partir do "Site de suporte da NetApp". Você pode atualizá-los manualmente ou incluí-los `packages/` no diretório do nó de controle do Ansible e preencher os seguintes parâmetros `eseries_storage_systems.yml` para atualizar usando o Ansible:

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

A configuração `eseries_drive_firmware_upgrade_drives_online` para `false` acelerará a atualização, mas não deverá ser feita depois que o BeeGFS for implantado. Isso ocorre porque essa configuração requer a interrupção de todas as I/o para as unidades antes da atualização para evitar erros de aplicativo. Embora a execução de uma atualização de firmware de unidade online antes de configurar volumes ainda seja rápida, recomendamos que você sempre defina esse valor para `true` evitar problemas mais tarde.

4. Para otimizar o desempenho, faça as seguintes alterações na configuração global:

```
# 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. Para garantir o provisionamento e o comportamento ideais de volume, especifique os seguintes parâmetros:

```
# 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"
```

ⓘ O valor especificado para `eseries_storage_pool_usable_drives` é específico para nós de bloco do NetApp EF600 e controla a ordem pela qual as unidades são atribuídas a novos grupos de volumes. Esse pedido garante que a e/S para cada grupo seja distribuída uniformemente pelos canais de unidade de back-end.

Definir o inventário do Ansible para os componentes básicos do BeeGFS

Depois de definir a estrutura geral de inventário do Ansible, defina a configuração para cada componente básico no sistema de arquivos BeeGFS.

Essas instruções de implantação demonstram como implantar um sistema de arquivos que consiste em um componente básico, incluindo gerenciamento, metadados e serviços de storage, um segundo componente básico com metadados e serviços de storage e um terceiro componente básico apenas de storage.

Essas etapas destinam-se a mostrar toda a gama de perfis de configuração típicos que você pode usar para configurar os componentes básicos do NetApp BeeGFS para atender aos requisitos do sistema de arquivos BeeGFS geral.

ⓘ Nesta e nas seções subsequentes, ajuste conforme necessário para criar o inventário que representa o sistema de arquivos BeeGFS que você deseja implantar. Em especial, use nomes de host do Ansible que representam cada bloco ou nó de arquivo e o esquema de endereçamento IP desejado para a rede de storage. Assim, ela pode ser dimensionada para o número de nós de arquivos BeeGFS e clientes.

Etapa 1: Crie o arquivo de inventário do Ansible

Passos

1. Crie um novo `inventory.yml` arquivo e, em seguida, insira os seguintes parâmetros, substituindo os hosts em `eseries_storage_systems` conforme necessário para representar os nós de bloco em sua implantação. Os nomes devem corresponder ao nome utilizado para `host_vars/<FILENAME>.yml`.

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

   Nas seções subsequentes, você criará grupos adicionais do Ansible `ha_cluster` que representam os serviços BeeGFS que deseja executar no cluster.

Etapa 2: Configurar o inventário para um componente básico de gerenciamento, metadados e armazenamento

O primeiro componente básico do cluster ou componente básico deve incluir o serviço de gerenciamento do BeeGFS, além de serviços de metadados e storage:

Passos

1. No `inventory.yml`, preencha os seguintes parâmetros em `ha_cluster: children`:

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

```
    beegfs_02:
meta_02:
  hosts:
    beegfs_01:
    beegfs_02:
stor_02:
  hosts:
    beegfs_01:
    beegfs_02:
meta_03:
  hosts:
    beegfs_01:
    beegfs_02:
stor_03:
  hosts:
    beegfs_01:
    beegfs_02:
meta_04:
  hosts:
    beegfs_01:
    beegfs_02:
stor_04:
  hosts:
    beegfs_01:
    beegfs_02:
meta_05:
  hosts:
    beegfs_02:
    beegfs_01:
stor_05:
  hosts:
    beegfs_02:
    beegfs_01:
meta_06:
  hosts:
    beegfs_02:
    beegfs_01:
stor_06:
  hosts:
    beegfs_02:
    beegfs_01:
meta_07:
  hosts:
    beegfs_02:
    beegfs_01:
stor_07:
```

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

2. Crie o arquivo `group_vars/mgmt.yml` e inclua o seguinte:

```
# mgmt - BeeGFS HA Management Resource Group
# OPTIONAL: Override default BeeGFS management configuration:
# beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
#  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
floating_ips:
  - i1b: 100.127.101.0/16
  - i2b: 100.127.102.0/16
beegfs_service: management
beegfs_targets:
  netapp_01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 1
            owning_controller: A
```

3. Em `group_vars/`, crie arquivos para grupos de recursos `meta_01` `meta_08` usando o modelo a seguir e preencha os valores de espaço reservado para cada serviço que faz referência à tabela abaixo:

```
# 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>
```

O tamanho do volume é especificado como uma porcentagem do conjunto de armazenamento geral (também conhecido como um grupo de volumes). A NetApp recomenda fortemente que você deixe alguma capacidade livre em cada pool para permitir espaço para provisionamento excessivo de SSD (para obter mais informações, "Introdução ao array NetApp EF600"consulte). O pool de armazenamento, `beegfs_m1_m2_m5_m6`, também aloca 1% da capacidade do pool para o serviço de gerenciamento. Assim, para volumes de metadados no pool de armazenamento, `beegfs_m1_m2_m5_m6`, quando 1,92TB ou 3,84TB unidades forem usadas, defina esse valor como `21.25`; para unidades 7,65TB, defina esse valor como `22.25`; e para unidades 15,3TB, defina esse valor como `23.75`.

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
meta_01.yml	8015	i1b:100.127.101.1/16 i2b:100.127.102.1/16	0	netapp_01	beegfs_m1_m2_m5_m6	A
meta_02.yml	8025	i2b:100.127.102.2/16 i1b:100.127.101.2/16	0	netapp_01	beegfs_m1_m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.101.3/16 i4b:100.127.102.3/16	1	netapp_02	beegfs_m3_m4_m7_m8	A

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
meta_04.yml	8045	i4b:100.127.102.4/16 i3b:100.127.101.4/16	1	netapp_02	beegfs_m3_m4_m7_m8	B
meta_05.yml	8055	i1b:100.127.101.5/16 i2b:100.127.102.5/16	0	netapp_01	beegfs_m1_m2_m5_m6	A
meta_06.yml	8065	i2b:100.127.102.6/16 i1b:100.127.101.6/16	0	netapp_01	beegfs_m1_m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.101.7/16 i4b:100.127.102.7/16	1	netapp_02	beegfs_m3_m4_m7_m8	A
meta_08.yml	8085	i4b:100.127.102.8/16 i3b:100.127.101.8/16	1	netapp_02	beegfs_m3_m4_m7_m8	B

4. Em `group_vars/`, crie arquivos para grupos de recursos `stor_01` `stor_08` usando o modelo a seguir e preencha os valores de espaço reservado para cada serviço que referencie o exemplo:

```
# 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>
```

Para obter o tamanho correto a ser usado, "Porcentagens recomendadas de provisionamento de pool de storage"consulte .

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
stor_01.yml	8013	i1b:100.127.103.1/16 i2b:100.127.104.1/16	0	netapp_01	beegfs_s1_s2	A
stor_02.yml	8023	i2b:100.127.104.2/16 i1b:100.127.103.2/16	0	netapp_01	beegfs_s1_s2	B
stor_03.yml	8033	i3b:100.127.103.3/16 i4b:100.127.104.3/16	1	netapp_02	beegfs_s3_s4	A
stor_04.yml	8043	i4b:100.127.104.4/16 i3b:100.127.103.4/16	1	netapp_02	beegfs_s3_s4	B

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
stor_05.yml	8053	i1b:100.127.103.5/16 i2b:100.127.104.5/16	0	netapp_01	beegfs_s5_s6	A
stor_06.yml	8063	i2b:100.127.104.6/16 i1b:100.127.103.6/16	0	netapp_01	beegfs_s5_s6	B
stor_07.yml	8073	i3b:100.127.103.7/16 i4b:100.127.104.7/16	1	netapp_02	beegfs_s7_s8	A
stor_08.yml	8083	i4b:100.127.104.8/16 i3b:100.127.103.8/16	1	netapp_02	beegfs_s7_s8	B

Passo 3: Configure o inventário para um bloco de construção de metadados e armazenamento

Estas etapas descrevem como configurar um inventário do Ansible para um componente básico de storage e metadados do BeeGFS.

Passos

1. No `inventory.yml`, preencha os seguintes parâmetros sob a configuração existente:

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

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

2. Em `group_vars/`, crie arquivos para grupos de recursos `meta_09` `meta_16` usando o modelo a seguir e preencha os valores de espaço reservado para cada serviço que referencie o exemplo:

```
# 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>
```

Para obter o tamanho correto a ser usado, "Porcentagens recomendadas de provisionamento de pool de storage"consulte .

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
meta_09.yml	8015	i1b:100.127.101.9/16 i2b:100.127.102.9/16	0	netapp_03	beegfs_m9_m10_m13_m14	A
meta_10.yml	8025	i2b:100.127.102.10/16 i1b:100.127.101.10/16	0	netapp_03	beegfs_m9_m10_m13_m14	B
meta_11.yml	8035	i3b:100.127.101.11/16 i4b:100.127.102.11/16	1	netapp_04	beegfs_m11_m12_m15_m16	A
meta_12.yml	8045	i4b:100.127.102.12/16 i3b:100.127.101.12/16	1	netapp_04	beegfs_m11_m12_m15_m16	B

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
meta_13.yml	8055	i1b:100.127.101.13/16 i2b:100.127.102.13/16	0	netapp_03	beegfs_m9_m10_m13_m14	A
meta_14.yml	8065	i2b:100.127.102.14/16 i1b:100.127.101.14/16	0	netapp_03	beegfs_m9_m10_m13_m14	B
meta_15.yml	8075	i3b:100.127.101.15/16 i4b:100.127.102.15/16	1	netapp_04	beegfs_m11_m12_m15_m16	A
meta_16.yml	8085	i4b:100.127.102.16/16 i3b:100.127.101.16/16	1	netapp_04	beegfs_m11_m12_m15_m16	B

3. Em `group_vars/`, criar arquivos para grupos de recursos `stor_09` `stor_16` usando o modelo a seguir e preencha os valores de espaço reservado para cada serviço que referencie o exemplo:

```
# 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>
```

Para obter o tamanho correto a ser usado, "Porcentagens recomendadas de provisionamento de pool de storage"consulte ..

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
stor_09.yml	8013	i1b:100.127.103.9/16 i2b:100.127.104.9/16	0	netapp_03	beegfs_s9_s10	A
stor_10.yml	8023	i2b:100.127.104.10/16 i1b:100.127.103.10/16	0	netapp_03	beegfs_s9_s10	B
stor_11.yml	8033	i3b:100.127.103.11/16 i4b:100.127.104.11/16	1	netapp_04	beegfs_s11_s12	A
stor_12.yml	8043	i4b:100.127.104.12/16 i3b:100.127.103.12/16	1	netapp_04	beegfs_s11_s12	B
stor_13.yml	8053	i1b:100.127.103.13/16 i2b:100.127.104.13/16	0	netapp_03	beegfs_s13_s14	A
stor_14.yml	8063	i2b:100.127.104.14/16 i1b:100.127.103.14/16	0	netapp_03	beegfs_s13_s14	B
stor_15.yml	8073	i3b:100.127.103.15/16 i4b:100.127.104.15/16	1	netapp_04	beegfs_s15_s16	A
stor_16.yml	8083	i4b:100.127.104.16/16 i3b:100.127.103.16/16	1	netapp_04	beegfs_s15_s16	B

Etapa 4: Configure o inventário para um componente básico somente de armazenamento

Estas etapas descrevem como configurar um inventário do Ansible para um componente básico somente de storage do BeeGFS. A principal diferença entre configurar a configuração de metadados e armazenamento versus um componente básico somente de armazenamento é a omissão de todos os grupos de recursos de metadados e a alteração de `criteria_drive_count` 10 para 12 para cada pool de armazenamento.

Passos

1. No `inventory.yml`, preencha os seguintes parâmetros sob a configuração existente:

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

2. Em `group_vars/`, crie arquivos para grupos de recursos `stor_17` `stor_24` usando o modelo a seguir e preencha os valores de espaço reservado para cada serviço que referencie o exemplo:

```
# 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>
```

Para obter o tamanho correto a ser usado, "Porcentagens recomendadas de provisionamento de pool de storage"consulte .

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
stor_17.yml	8013	i1b:100.127.103.17/16 i2b:100.127.104.17/16	0	netapp_05	beegfs_s17_s18	A
stor_18.yml	8023	i2b:100.127.104.18/16 i1b:100.127.103.18/16	0	netapp_05	beegfs_s17_s18	B
stor_19.yml	8033	i3b:100.127.103.19/16 i4b:100.127.104.19/16	1	netapp_06	beegfs_s19_s20	A
stor_20.yml	8043	i4b:100.127.104.20/16 i3b:100.127.103.20/16	1	netapp_06	beegfs_s19_s20	B

Nome do ficheiro	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
stor_21.yml	8053	i1b:100.127.103.21/16 i2b:100.127.104.21/16	0	netapp_05	beegfs_s21_s22	A
stor_22.yml	8063	i2b:100.127.104.22/16 i1b:100.127.103.22/16	0	netapp_05	beegfs_s21_s22	B
stor_23.yml	8073	i3b:100.127.103.23/16 i4b:100.127.104.23/16	1	netapp_06	beegfs_s23_s24	A
stor_24.yml	8083	i4b:100.127.104.24/16 i3b:100.127.103.24/16	1	netapp_06	beegfs_s23_s24	B

Implantar o BeeGFS

A implantação e o gerenciamento da configuração envolve a execução de um ou mais playbooks que contêm as tarefas que o Ansible precisa executar e colocar o sistema geral no estado desejado.

Embora todas as tarefas possam ser incluídas em um único manual, para sistemas complexos, isso rapidamente se torna difícil de gerenciar. O Ansible permite que você crie e distribua funções como uma forma de empacotar playbooks reutilizáveis e conteúdo relacionado (por exemplo: Variáveis padrão, tarefas e manipuladores). Para obter mais informações, consulte a documentação do Ansible para "Funções".

As funções geralmente são distribuídas como parte de uma coleção do Ansible que contém funções e módulos relacionados. Assim, esses playbooks apenas importam várias funções distribuídas nas várias coleções do NetApp e-Series Ansible.

> ⓘ Atualmente, pelo menos dois componentes básicos (quatro nós de arquivo) são necessários para implantar o BeeGFS, a menos que um dispositivo de quorum separado seja configurado como um tiebreaker para mitigar quaisquer problemas ao estabelecer quorum com um cluster de dois nós.

Passos

1. Crie um novo `playbook.yml` arquivo e inclua o seguinte:

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

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

```
your playbook command with --list-tags to see all valid playbook tags."
      when: 'item not in ["all", "storage", "beegfs_ha",
"beegfs_ha_package", "beegfs_ha_configure",
"beegfs_ha_configure_resource", "beegfs_ha_performance_tuning",
"beegfs_ha_backup", "beegfs_ha_client"]'
      loop: "{{ ansible_run_tags }}"
  tasks:
    - name: Verify before proceeding.
      pause:
        prompt: "Are you ready to proceed with running the BeeGFS HA
role? Depending on the size of the deployment and network performance
between the Ansible control node and BeeGFS file and block nodes this
can take awhile (10+ minutes) to complete."
    - name: Verify the BeeGFS HA cluster is properly deployed.
      ansible.builtin.import_role:
        name: netapp_eseries.beegfs.beegfs_ha_7_4
```

Esse manual de estratégia executa alguns `pre_tasks` que verificam se o Python 3 está instalado nos nós de arquivo e verificam se as tags do Ansible fornecidas são compatíveis.

2. Use o `ansible-playbook` comando com os arquivos de inventário e manual de estratégia quando estiver pronto para implantar o BeeGFS.

   A implantação executará tudo `pre_tasks`e solicitará a confirmação do usuário antes de prosseguir com a implantação real do BeeGFS.

   Execute o seguinte comando, ajustando o número de garfos conforme necessário (veja a nota abaixo):

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

   Especialmente para implantações maiores, a substituição do número padrão de bifurcações (5) usando o `forks` parâmetro é recomendada para aumentar o número de hosts que o Ansible configura em paralelo. (Para obter mais informações, "Controlar a execução do manual de estratégia"consulte .) A configuração de valor máximo depende da potência de processamento disponível no nó de controle do Ansible. O exemplo acima de 20 foi executado em um nó de controle virtual do Ansible com 4 CPUs (CPU Intel® Xeon® Gold 6146 com 3,20GHz GB).

   Dependendo do tamanho da implantação e da performance de rede entre o nó de controle do Ansible e os nós de bloco e arquivo do BeeGFS, o tempo de implantação pode variar.

Configurar clientes BeeGFS

Você precisa instalar e configurar o cliente BeeGFS em todos os hosts que precisam ter acesso ao sistema de arquivos BeeGFS, como nós de computação ou GPU. Para essa tarefa, use o Ansible e a coleção BeeGFS.

Passos

1. Se necessário, configure o SSH sem senha do nó de controle do Ansible para cada um dos hosts que você deseja configurar como clientes BeeGFS:

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

2. Em `host_vars/`, crie um arquivo para cada cliente BeeGFS nomeado `<HOSTNAME>.yml` com o seguinte conteúdo, preenchendo o texto do espaço reservado com as informações corretas para o seu 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>
       address: <IP>/<SUBNET_MASK>
   ```

   Se estiver implantando com um esquema de endereçamento de duas sub-redes, duas interfaces InfiniBand devem ser configuradas em cada cliente, uma em cada uma das duas sub-redes IPoIB de storage. Se estiver usando as sub-redes de exemplo e os intervalos recomendados para cada serviço BeeGFS listado aqui, os clientes devem ter uma interface configurada no intervalo de `100.127.1.0` a `100.127.99.255` e a outra em `100.128.1.0` para `100.128.99.255`.

3. Crie um novo `client_inventory.yml` arquivo e preencha os seguintes parâmetros na parte superior:

   ```
   # 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.
   ```

   Não armazene senhas em texto simples. Em vez disso, use o Ansible Vault (consulte a documentação do Ansible para "Criptografia de conteúdo com o Ansible Vault") ou use a `--ask-become-pass` opção ao executar o manual de estratégia.

4. No `client_inventory.yml` arquivo, liste todos os hosts que devem ser configurados como clientes BeeGFS no `beegfs_clients` grupo e especifique qualquer configuração adicional necessária para criar o módulo do kernel do cliente BeeGFS.

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        beegfs_01:
        beegfs_02:
        beegfs_03:
        beegfs_04:
        beegfs_05:
        beegfs_06:
        beegfs_07:
        beegfs_08:
        beegfs_09:
        beegfs_10:
      vars:
        # OPTION 1: If you're using the NVIDIA OFED drivers and they are
already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        beegfs_client_ofed_enable: True
        beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"
        # OPTION 2: If you're using inbox IB/RDMA drivers and they are
already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        # OPTION 3: If you want to use inbox IB/RDMA drivers and need
them installed/configured.
        eseries_ib_skip: False # Default value.
        beegfs_client_ofed_enable: False # Default value.
```

> Ao usar os drivers NVIDIA OFED, certifique-se de que `beegfs_client_ofed_include_path` aponta para o "caminho de inclusão de cabeçalho" correto para a instalação do Linux. Para obter mais informações, consulte a documentação do BeeGFS para "Suporte RDMA".

5. No `client_inventory.yml` arquivo, liste os sistemas de arquivos BeeGFS que você deseja montar na parte inferior de qualquer um definido anteriormente `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
```

`beegfs_client_config`O representa as definições que foram testadas. Veja a documentação incluída com `netapp_eseries.beegfs` a função da coleção `beegfs_client` para uma visão geral abrangente de todas as opções. Isso inclui detalhes sobre a montagem de vários sistemas de arquivos BeeGFS ou a montagem do mesmo sistema de arquivos BeeGFS várias vezes.

6. Crie um novo `client_playbook.yml` arquivo e preencha os seguintes parâmetros:

```
# 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
```

Omitir a importação da `netapp_eseries.host` coleção e `ipoib` da função se você já tiver instalado os drivers IB/RDMA necessários e IPs configurados nas interfaces IPoIB apropriadas.

7. Para instalar e construir o cliente e montar o BeeGFS, execute o seguinte comando:

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

8. Antes de colocar o sistema de arquivos BeeGFS em produção, nós * fortemente * recomendamos que você faça login em qualquer cliente e execute `beegfs-fsck --checkfs` para garantir que todos os nós estejam acessíveis e não haja problemas relatados.

Escala além de cinco componentes básicos

Você pode configurar o Pacemaker e o Corosync para escalar além de cinco blocos de construção (10 nós de arquivo). No entanto, há desvantagens para clusters maiores e, eventualmente, a Pacemaker e a Corosync impõem um máximo de 32 nós.

A NetApp testou apenas clusters de HA do BeeGFS para até 10 nós. O dimensionamento de clusters individuais além desse limite não é recomendado ou compatível. No entanto, os sistemas de arquivos BeeGFS ainda precisam ser dimensionados para além de 10 nós, e a NetApp foi responsável pela solução BeeGFS on NetApp.

Com a implantação de vários clusters de HA que contêm um subconjunto dos componentes básicos em cada sistema de arquivos, é possível dimensionar o sistema de arquivos BeeGFS geral independentemente de quaisquer limites físicos ou recomendados pelos mecanismos subjacentes de clustering HA. Neste cenário, faça o seguinte:

- Crie um novo inventário do Ansible que represente o(s) cluster(s) de HA adicional(s) e, em seguida, omite a configuração de outro serviço de gerenciamento. Em vez disso, aponte a `beegfs_ha_mgmtd_floating_ip` variável em cada cluster adicional `ha_cluster.yml` para o IP do primeiro serviço de gerenciamento BeeGFS.

- Ao adicionar clusters de HA adicionais ao mesmo sistema de arquivos, verifique o seguinte:
  - As IDs de nó do BeeGFS são exclusivas.
  - Os nomes de arquivo correspondentes a cada serviço em `group_vars` são exclusivos em todos os clusters.
  - Os endereços IP do cliente e do servidor BeeGFS são exclusivos em todos os clusters.
  - O primeiro cluster de HA que contém o serviço de gerenciamento BeeGFS está sendo executado antes de tentar implantar ou atualizar clusters adicionais.
- Mantenha os inventários de cada cluster de HA separadamente em sua própria árvore de diretórios.

  Tentar misturar arquivos de inventário para vários clusters em uma árvore de diretório pode causar problemas em como a função de HA BeeGFS agrega a configuração aplicada a um cluster específico.

> Não é necessário que cada cluster de HA escale até cinco componentes básicos antes de criar um novo. Em muitos casos, é mais fácil gerenciar o uso de menos componentes básicos por cluster. Uma abordagem é configurar os componentes básicos em cada rack único como um cluster de HA.

Porcentagens recomendadas de provisionamento de pool de storage

Ao seguir a configuração padrão de quatro volumes por pool de storage para componentes básicos de segunda geração, consulte a tabela a seguir.

Essa tabela fornece as porcentagens recomendadas a serem usadas como o tamanho do volume em `eseries_storage_pool_configuration` cada metadados do BeeGFS ou destino de storage:

Tamanho da unidade	Tamanho
1,92 TB	18
3,84 TB	21,5
7,68 TB	22,5
15,3 TB	24

> As orientações acima não se aplicam ao pool de storage que contém o serviço de gerenciamento, o que deve reduzir os tamanhos acima em 25% para alocar 1% do pool de storage para dados de gerenciamento.

Para entender como esses valores foram determinados, "TR-4800: Apêndice A: Compreender a resistência e o provisionamento de SSD"consulte .

Componente básico de alta capacidade

O guia de implantação padrão da solução BeeGFS descreve os procedimentos e recomendações para requisitos de workloads de alta performance. Os clientes que desejam atender a requisitos de alta capacidade devem observar as variações na implantação e recomendações descritas aqui.

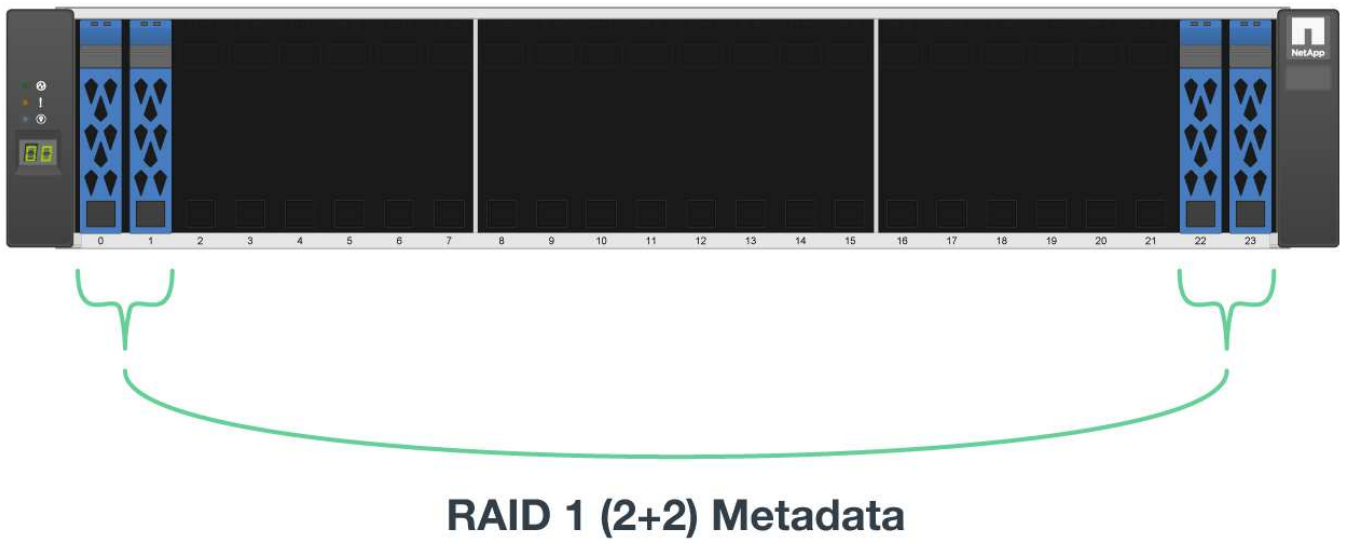
Controladores

Para blocos de construção de alta capacidade, os controladores EF600 devem ser substituídos por controladores EF300, cada um com um Cascade HIC instalado para expansão SAS. Cada nó de bloco terá um número mínimo de SSDs NVMe preenchidos no compartimento do array para o storage de metadados do BeeGFS e será anexado aos compartimentos de expansão preenchidos com HDDs NL-SAS para volumes de storage BeeGFS.

A configuração do nó do arquivo para o nó do bloco permanece a mesma.

Posicionamento da unidade

Para o storage de metadados do BeeGFS, são necessários no mínimo 4 SSD NVMe em cada nó de bloco. Essas unidades devem ser colocadas nos slots mais externos do gabinete.



Bandejas de expansão

O componente básico de alta capacidade pode ser dimensionado com bandejas de expansão de 1 a 7 unidades e 60 por storage array.

Para obter instruções sobre o cabo de cada bandeja de expansão, "Consulte o cabeamento EF300 para ver as gavetas de unidades".

Use arquiteturas personalizadas

Visão geral e requisitos

Usar os sistemas de storage do NetApp e/EF-Series como nós de bloco do BeeGFS e x86 servidores como nós de arquivo BeeGFS ao implantar clusters de alta disponibilidade do BeeGFS usando o Ansible.

As definições de terminologia usadas nesta seção podem ser encontradas na "termos e conceitos"página.

Introdução

"Arquiteturas verificadas da NetApp"Embora forneça configurações de referência predefinidas e orientações para dimensionamento, alguns clientes e parceiros podem preferir projetar arquiteturas personalizadas mais adequadas a requisitos específicos ou preferências de hardware. Um dos principais benefícios da escolha do BeeGFS no NetApp é a capacidade de implantar clusters de HA de disco compartilhado BeeGFS com o Ansible. Isso simplifica o gerenciamento do cluster e aumenta a confiabilidade com componentes de HA de autoria da NetApp. A implantação de arquiteturas BeeGFS personalizadas no NetApp ainda é feita usando o Ansible. Isso mantém uma abordagem semelhante a um dispositivo em uma variedade flexível de hardware.

Esta seção descreve as etapas gerais necessárias para implantar os sistemas de arquivos BeeGFS no hardware do NetApp e o uso do Ansible para configurar os sistemas de arquivos BeeGFS. Para obter detalhes sobre as melhores práticas em torno do design dos sistemas de arquivos BeeGFS e exemplos otimizados, consulte "Arquiteturas verificadas da NetApp"a seção.

Visão geral da implantação

Geralmente, a implantação de um sistema de arquivos BeeGFS envolve as seguintes etapas:

- Configuração inicial:
  - Instale/cabo de hardware.
  - Configurar nós de arquivo e bloco.
  - Configure um nó de controle do Ansible.
- Defina o sistema de arquivos BeeGFS como um inventário do Ansible.
- Execute o Ansible em nós de arquivo e bloco para implantar o BeeGFS.
  - Opcionalmente, para configurar clientes e montar BeeGFS.

As seções subsequentes cobrirão esses passos com mais detalhes.

O Ansible gerencia todas as tarefas de configuração e provisionamento de software, incluindo:

- Criando/mapeando volumes em nós de bloco.
- Formatação/ajuste de volumes em nós de arquivo.
- Instalação/configuração de software em nós de arquivo.
- Estabelecimento do cluster de HA e configuração de recursos do BeeGFS e serviços de sistema de arquivos.

Requisitos

O suporte ao BeeGFS no Ansible é lançado "Ansible Galaxy"como uma coleção de funções e módulos que automatizam a implantação e o gerenciamento completos dos clusters de HA do BeeGFS.

O BeeGFS em si é versionado seguindo um esquema de controle de versão do <major> <major>.<minor> <minor>.<patch> e a coleção mantém funções para cada versão suportada do BeeGFS, por exemplo, BeeGFS 7,2 ou BeeGFS 7,3. À medida que as atualizações da coleção são lançadas, a versão de patch em cada função será atualizada para apontar a versão BeeGFS mais recente disponível para essa ramificação de lançamento (exemplo: 7,2.8). Cada versão da coleção também é testada e suportada com distribuições e versões específicas do Linux, atualmente Red Hat para nós de arquivos, RedHat e Ubuntu para clientes. A execução de outras distribuições não é suportada e a execução de outras versões (especialmente outras versões principais) não é recomendada.

Nó de controle do Ansible

Esse nó conterá o inventário e os playbooks usados para gerenciar o BeeGFS. Requer:

- Ansible-core 6.x (Ansible-core 2,13)
- Python 3,6 (ou posterior)
- Pacotes Python (PIP): Ipaddr e netaddr

Também é recomendável configurar SSH sem senha do nó de controle para todos os nós de arquivo BeeGFS e clientes.

Nós de arquivos BeeGFS

Os nós de arquivo devem executar o RedHat 9,3 e ter acesso ao repositório HA que contém os pacotes necessários (pacemaker, corosync, fence-agents-all, resource-maents). Por exemplo, o seguinte comando pode ser executado para ativar o repositório apropriado no RedHat 9:

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

Nós de cliente BeeGFS

Uma função Ansible do cliente BeeGFS está disponível para instalar o pacote cliente BeeGFS e gerenciar a(s) montagem(s) do BeeGFS. Esta função foi testada com RedHat 8,4 e Ubuntu 22,04.

Se você não estiver usando o Ansible para configurar o cliente BeeGFS e montar o BeeGFS, é possível usar qualquer um"BeeGFS suporta distribuição Linux e kernel".

Configuração inicial

Instalação e hardware do cabo

As etapas necessárias para instalar e o hardware do cabo usado para executar o BeeGFS no NetApp.

Planeie a instalação

Cada sistema de arquivos BeeGFS consistirá em alguns nós de arquivo que executam serviços BeeGFS usando storage de back-end fornecido por alguns nós de bloco. Os nós de arquivo são configurados em um ou mais clusters de alta disponibilidade para fornecer tolerância de falhas para serviços BeeGFS. Cada nó de bloco já é um par de HA ativo/ativo. O número mínimo de nós de arquivos com suporte em cada cluster de HA é de três e o número máximo de nós de arquivos com suporte em cada cluster é de dez. Os sistemas de arquivos BeeGFS podem ser dimensionados além de dez nós, implantando vários clusters de HA independentes que trabalham juntos para fornecer um namespace único do sistema de arquivos.

Normalmente, cada cluster de HA é implantado como uma série de "componentes básicos", onde alguns nós de arquivos (x86 servidores) são diretamente conectados a algum número de nós de bloco (normalmente sistemas de storage e-Series). Essa configuração cria um cluster assimétrico, no qual os serviços do BeeGFS só podem ser executados em certos nós de arquivo que têm acesso ao storage de bloco de back-end usado nos destinos do BeeGFS. O equilíbrio de nós de arquivo para bloco em cada componente básico e o protocolo de storage em uso para conexões diretas dependem dos requisitos de uma instalação específica.

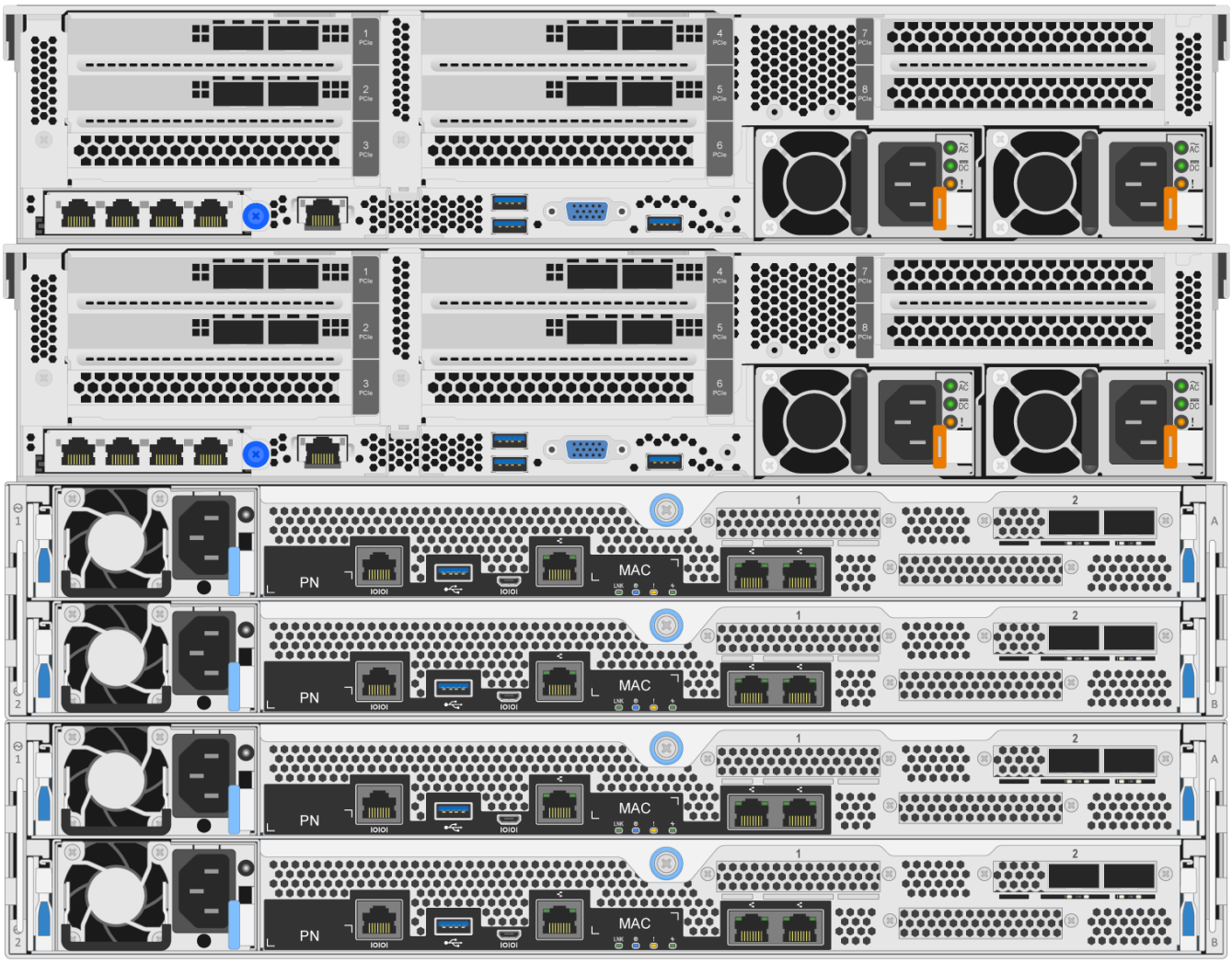
Uma arquitetura de cluster de HA alternativa usa uma malha de storage (também conhecida como rede de área de storage ou SAN) entre os nós de arquivo e bloco para estabelecer um cluster simétrico. Isso permite que os serviços BeeGFS sejam executados em qualquer nó de arquivo em um cluster de HA específico. Como os clusters simétricos geralmente não são tão econômicos devido ao hardware SAN extra, esta documentação presume o uso de um cluster assimétrico implantado como uma série de um ou mais blocos de construção.

Certifique-se de que a arquitetura do sistema de arquivos desejada para uma determinada implantação do BeeGFS seja bem compreendida antes de prosseguir com a instalação.

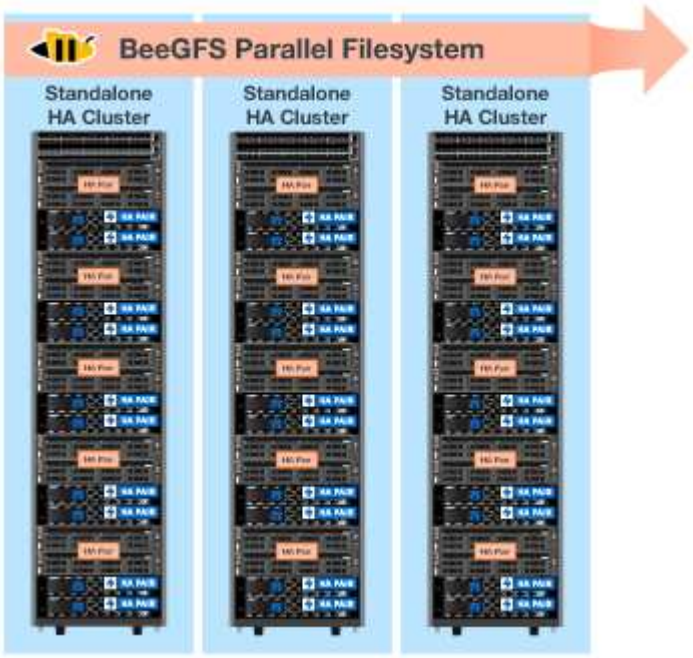
Hardware de rack

Ao Planejar a instalação, é importante que todos os equipamentos em cada bloco de construção sejam montados em rack adjacentes. A prática recomendada é que os nós de arquivo sejam colocados imediatamente acima dos nós de bloco em cada componente básico. Siga a documentação do(s) modelo(s) de arquivo e "bloco" nós que você está usando ao instalar trilhos e hardware no rack.

Exemplo de um único componente básico:

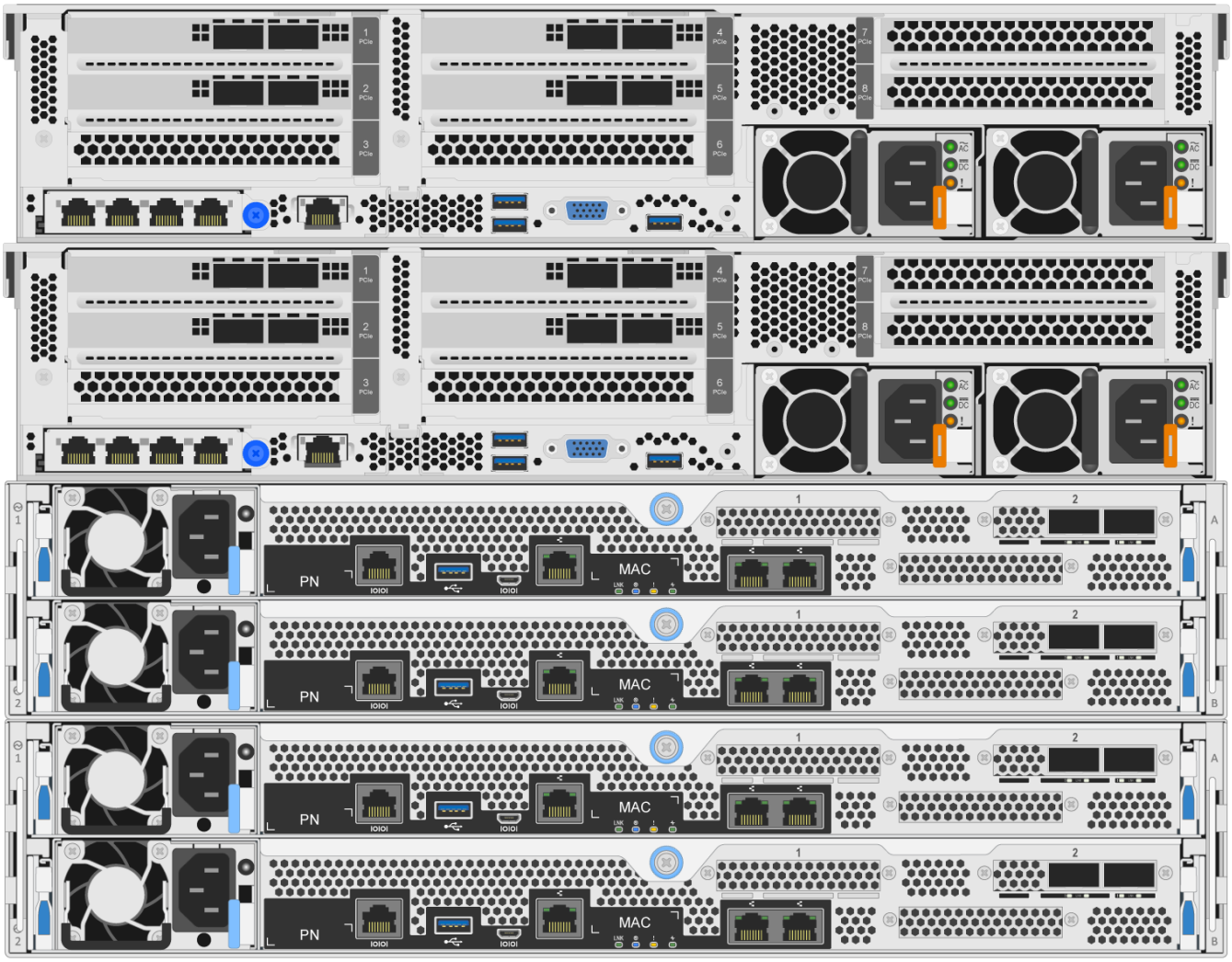


Exemplo de uma instalação grande do BeeGFS onde há vários componentes básicos em cada cluster de HA e vários clusters de HA no sistema de arquivos:



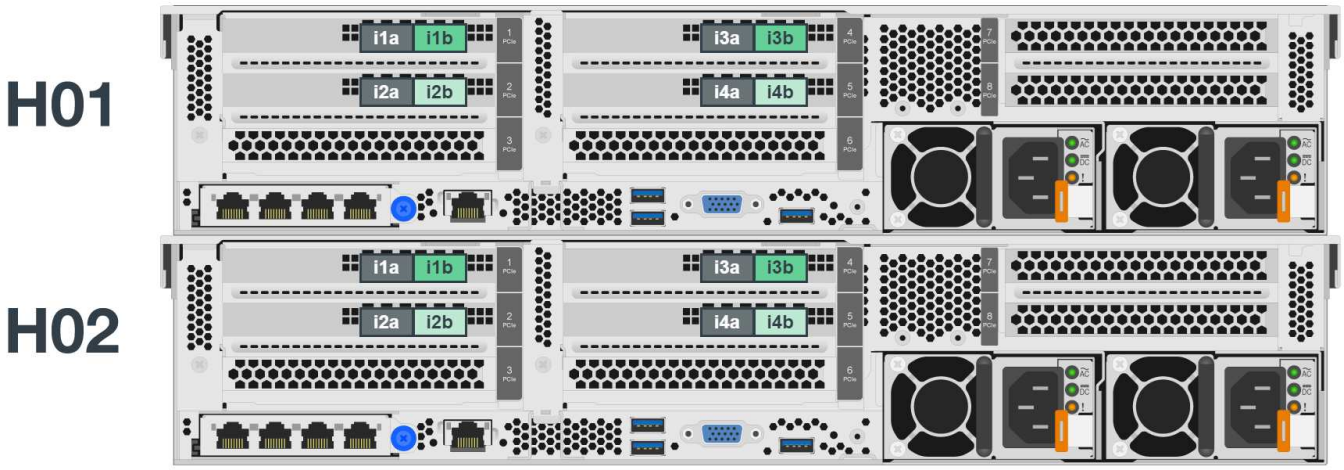
Nós de bloco e arquivo de cabo

Normalmente, você conetará diretamente as portas HIC dos nós de bloco e-Series às portas designadas do adaptador de canal do host (para protocolos InfiniBand) ou do adaptador de barramento do host (para Fibre Channel e outros protocolos) dos nós de arquivo. A maneira exata de estabelecer essas conexões dependerá da arquitetura desejada do sistema de arquivos, aqui está um "Baseado na arquitetura BeeGFS de segunda geração na NetApp Verified"exemplo :



Nós de arquivo de cabo para a rede do cliente

Cada nó de arquivo terá algum número de portas InfiniBand ou Ethernet designadas para o tráfego de clientes BeeGFS. Dependendo da arquitetura, cada nó de arquivo terá uma ou mais conexões com uma rede cliente/storage de alto desempenho, potencialmente com vários switches para redundância e maior largura de banda. Aqui está um exemplo de cabeamento de cliente usando switches de rede redundantes, onde as portas destacadas em verde escuro versus verde claro estão conetadas a switches separados:



Ligar a rede de gestão e energia

Estabeleça todas as conexões de rede necessárias para a rede dentro e fora da banda.

Conete todas as fontes de alimentação garantindo que cada nó de arquivo e bloco tenha conexões com várias unidades de distribuição de energia para redundância (se disponível).

Configurar nós de arquivo e bloco

Etapas manuais necessárias para configurar nós de arquivo e bloco antes de executar o Ansible.

Nós de arquivo

Configurar o controlador de gerenciamento de placa base (BMC)

Um controlador de gerenciamento de placa base (BMC), às vezes chamado de processador de serviço, é o nome genérico para o recurso de gerenciamento fora da banda incorporado em várias plataformas de servidor que podem fornecer acesso remoto, mesmo que o sistema operacional não esteja instalado ou acessível. Normalmente, os fornecedores comercializam essa funcionalidade com sua própria marca. Por exemplo, no Lenovo SR665, o BMC é conhecido como o controlador XClarity (XCC) da Lenovo.

Siga a documentação do fornecedor do servidor para habilitar as licenças necessárias para acessar essa funcionalidade e garantir que o BMC esteja conetado à rede e configurado adequadamente para acesso remoto.

> Se for desejado esgrima baseada em BMC usando o Red Fish, certifique-se de que o Redfish esteja habilitado e que a interface BMC esteja acessível a partir do sistema operacional instalado no nó do arquivo. Pode ser necessária uma configuração especial no comutador de rede se o BMC e o sistema operativo partilharem a mesma interface de rede física.

Sintonize as definições do sistema

Usando a interface de configuração do sistema (BIOS/UEFI), verifique se as configurações estão definidas para maximizar o desempenho. As configurações exatas e os valores ideais variam de acordo com o modelo do servidor em uso. As orientações são fornecidas para "modelos de nó de arquivo verificados", caso contrário, consulte a documentação do fornecedor do servidor e as práticas recomendadas com base no seu modelo.

Instale um sistema operativo

Instale um sistema operacional suportado com base nos requisitos de nó de arquivo "aqui"listados . Consulte as etapas adicionais abaixo com base na sua distribuição Linux.

RedHat

Use o RedHat Subscription Manager para Registrar e assinar o sistema para permitir a instalação dos pacotes necessários dos repositórios oficiais da Red Hat e limitar as atualizações da versão suportada do Red Hat: `subscription-manager release --set=<MAJOR_VERSION>.<MINOR_VERSION>`. Para obter instruções, consulte "Como Registrar e assinar um sistema RHEL" e "Como limitar as atualizações".

Ative o repositório Red Hat que contém os pacotes necessários para alta disponibilidade:

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

Configurar a rede de gestão

Configure todas as interfaces de rede necessárias para permitir o gerenciamento na banda do sistema operacional. As etapas exatas dependerão da distribuição e versão específica do Linux em uso.

> Certifique-se de que o SSH esteja ativado e que todas as interfaces de gerenciamento estejam acessíveis a partir do nó de controle do Ansible.

Atualize o firmware HCA e HBA

Certifique-se de que todos os HBAs e HCAs estão a executar versões de firmware suportadas listadas no "Matriz de interoperabilidade do NetApp" e, se necessário, atualize. Recomendações adicionais para adaptadores NVIDIA ConnectX podem ser encontradas "aqui".

Nós de bloco

Siga as etapas para "Comece a trabalhar com o e-Series"configurar a porta de gerenciamento em cada controlador de nó de bloco e, opcionalmente, definir o nome do storage array para cada sistema.

> Nenhuma configuração adicional além de garantir que todos os nós de bloco estejam acessíveis a partir do nó de controle do Ansible. A configuração restante do sistema será aplicada/mantida com o Ansible.

Configurar o nó de controle do Ansible

Configure um nó de controle do Ansible para implantar e gerenciar o sistema de arquivos.

Visão geral

Um nó de controle do Ansible é uma máquina Linux física ou virtual usada para gerenciar o cluster. Deve cumprir os seguintes requisitos:

- Conheça a "requisitos"função de HA do BeeGFS, incluindo as versões instaladas do Ansible, Python e

quaisquer pacotes Python adicionais.

- Conheça o oficial"Requisitos de nó de controle do Ansible", incluindo versões de sistema operacional.
- Tenha acesso SSH e HTTPS a todos os nós de arquivos e blocos.

Etapas detalhadas da instalação podem ser encontradas "aqui".

Defina o sistema de arquivos BeeGFS

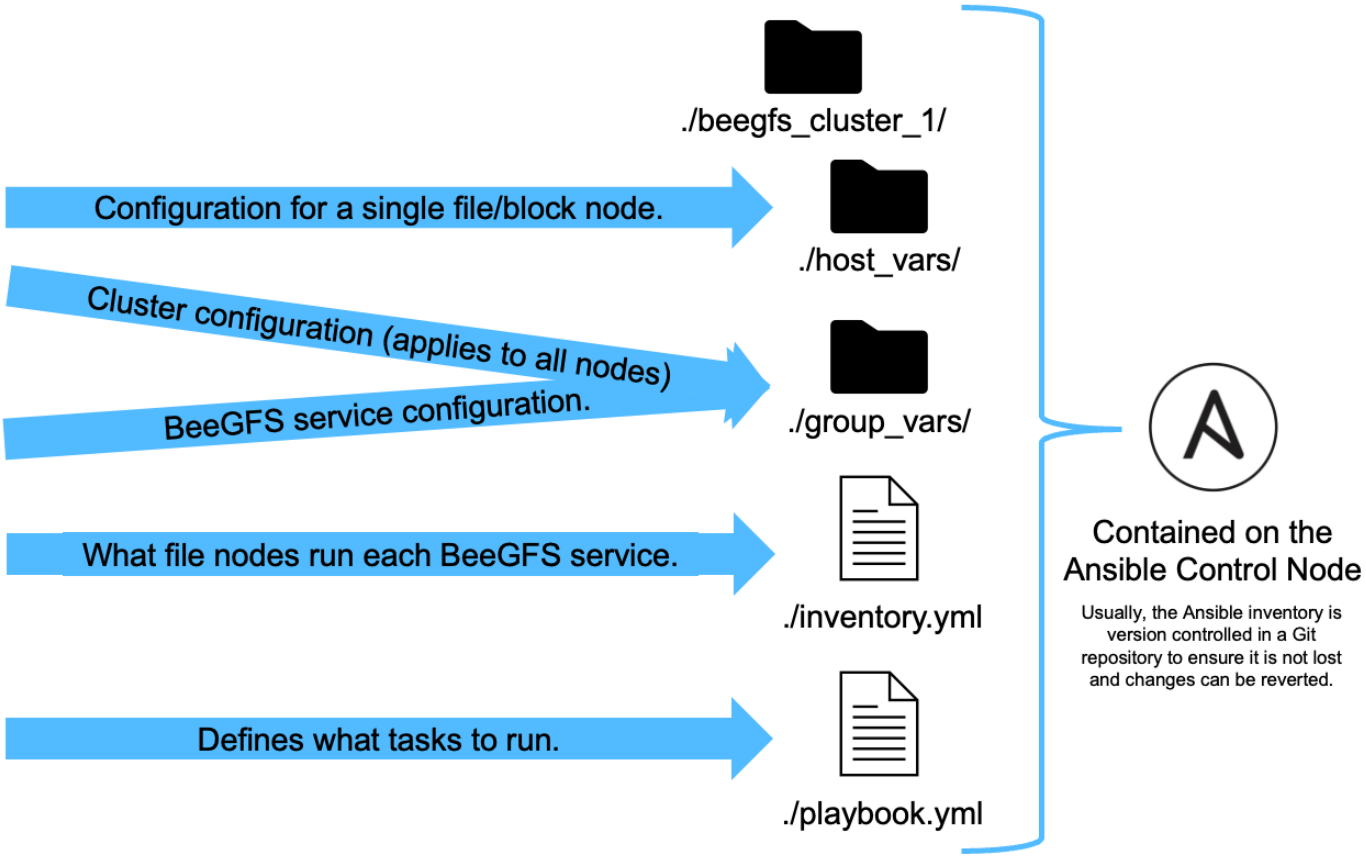
Visão geral do Ansible Inventory

O inventário do Ansible é um conjunto de arquivos de configuração que definem o cluster de HA do BeeGFS desejado.

Visão geral

Recomenda-se seguir as práticas padrão do Ansible para organizar o "inventário", incluindo o uso do , "subdiretórios/ficheiros"em vez de armazenar todo o inventário em um arquivo.

O inventário do Ansible para um único cluster BeeGFS HA está organizado da seguinte forma:



Como um único sistema de arquivos BeeGFS pode abranger vários clusters de HA, é possível que grandes instalações tenham vários inventários do Ansible. Geralmente, não é recomendável definir vários clusters de HA como um único inventário do Ansible para evitar problemas.

Passos

1. No nó de controle do Ansible, crie um diretório vazio que conterá o inventário do Ansible para o cluster BeeGFS que você deseja implantar.
   a. Se o seu sistema de arquivos eventualmente contiver vários clusters de HA, é recomendável criar primeiro um diretório para o sistema de arquivos e, em seguida, subdiretórios para o inventário que representa cada cluster de HA. Por exemplo:

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

2. No diretório que contém o inventário do cluster HA que deseja implantar, crie dois diretórios `group_vars` e `host_vars` dois arquivos `inventory.yml` e `playbook.yml`.

As seções a seguir descrevem o conteúdo de cada um desses arquivos.

Planeie o sistema de ficheiros

Planeje a implantação do sistema de arquivos antes de desenvolver o inventário do Ansible.

Visão geral

Antes de implantar o sistema de arquivos, você deve definir quais endereços IP, portas e outras configurações serão exigidas por todos os nós de arquivo, nós de bloco e serviços BeeGFS executados no cluster. Embora a configuração exata varie com base na arquitetura do cluster, esta seção define as práticas recomendadas e as etapas a seguir que geralmente são aplicáveis.

Passos

1. Se você estiver usando um protocolo de storage baseado em IP (como iSER, iSCSI, NVMe/IB ou NVMe/RoCE) para conectar nós de arquivo a nós de bloco, preencha a Planilha a seguir para cada bloco básico. Cada conexão direta em um único bloco de construção deve ter uma sub-rede única, e não deve haver sobreposição com sub-redes usadas para conetividade cliente-servidor.

Nó de arquivo	Porta de IB	Endereço IP	Nó de bloco	Porta de IB	IP físico	IP virtual (apenas para EF600K com HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> Se os nós de arquivo e bloco em cada bloco de construção estiverem diretamente conetados, você pode frequentemente reutilizar os mesmos IPs/esquema para vários blocos de construção.

2. Independentemente de você estar usando InfiniBand ou RDMA em Converged Ethernet (RoCE) para a

rede de storage, preencha a seguinte Planilha para determinar os intervalos de IP que serão usados para serviços de cluster de HA, serviços de arquivos BeeGFS e clientes para se comunicar:

Finalidade	Porta InfiniBand	Endereço IP ou intervalo
IP(s) de cluster do BeeGFS	<INTERFACE(s)>	<RANGE>
Gestão BeeGFS	<INTERFACE(s)>	<IP(s)>
Metadados BeeGFS	<INTERFACE(s)>	<RANGE>
Storage BeeGFS	<INTERFACE(s)>	<RANGE>
Clientes BeeGFS	<INTERFACE(s)>	<RANGE>

a. Se você estiver usando uma única sub-rede IP, apenas uma Planilha será necessária, caso contrário, também preencha uma Planilha para a segunda sub-rede.

3. Com base no exposto, para cada componente básico no cluster, preencha a seguinte Planilha que define quais serviços BeeGFS ele será executado. Para cada serviço, especifique o(s) nó(s) de arquivo preferencial/secundário(s), a porta de rede, IP(s) flutuante(s), a atribuição de zona NUMA (se necessário) e que nó(s) de bloco serão usados para seus destinos. Consulte as seguintes diretrizes ao preencher a Planilha:

   a. Especifique os serviços BeeGFS como `mgmt.yml` , `meta_<ID>.yml` `storage_<ID>.yml` ou onde o ID representa um número exclusivo em todos os serviços BeeGFS desse tipo neste sistema de arquivos. Esta convenção simplificará a referência a esta Planilha em seções subsequentes ao criar arquivos para configurar cada serviço.

   b. As portas para serviços BeeGFS precisam ser exclusivas em um componente básico específico. Certifique-se de que os serviços com o mesmo número de porta nunca possam ser executados no mesmo nó de arquivo para evitar conflitos de porta.

   c. Se necessário, os serviços podem usar volumes de mais de um nó de bloco e/ou pool de storage (e nem todos os volumes precisam pertencer à mesma controladora). Vários serviços também podem compartilhar o mesmo nó de bloco e/ou configuração de pool de storage (volumes individuais serão definidos em uma seção posterior).

Serviço BeeGFS (nome do arquivo)	Nós de arquivo	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
<SERVICE TYPE>_<ID>.yml	NÓ(S) DE ARQUIVO SECUNDÁRIO(S)>	<PORT>	<INTERFACE>:<IP/SUBNET> <INTERFACE>:<IP/SUBNET>	<NUMA NODE/ZONE>	<BLOCK NODE>	POOL DE ARMAZENAMENTO/ GRUPO DE VOLUME>	A OU B>

Para obter mais detalhes sobre convenções padrão, práticas recomendadas e planilhas de exemplo preenchidas, consulte "práticas recomendadas"as seções e "Defina os componentes básicos do BeeGFS"do BeeGFS na arquitetura verificada do NetApp.

Definir nós de arquivo e bloco

Configurar nós de arquivo individuais

Especifique a configuração para nós de arquivo individuais usando variáveis de host (host_vars).

Visão geral

Esta seção aborda o preenchimento de um `host_vars/<FILE_NODE_HOSTNAME>.yml` arquivo para cada nó de arquivo no cluster. Esses arquivos só devem conter configuração exclusiva de um nó de arquivo específico. Isso geralmente inclui:

- Definindo o IP ou o nome do host que o Ansible deve usar para se conetar ao nó.
- Configuração de interfaces adicionais e IPs de cluster usados para serviços de cluster HA (Pacemaker e Corosync) para se comunicar com outros nós de arquivo. Por padrão, esses serviços usam a mesma rede que a interface de gerenciamento, mas interfaces adicionais devem estar disponíveis para redundância. A prática comum é definir IPs adicionais na rede de armazenamento, evitando a necessidade de um cluster adicional ou rede de gerenciamento.
  - O desempenho de qualquer rede usada para comunicação em cluster não é crítico para o desempenho do sistema de arquivos. Com a configuração padrão do cluster, geralmente, pelo menos uma rede de 1GB GB/s fornecerá desempenho suficiente para operações do cluster, como a sincronização dos estados dos nós e a coordenação das alterações do estado dos recursos do cluster. Redes lentas/ocupadas podem fazer com que as alterações de estado de recursos demorem mais tempo do que o normal, e em casos extremos podem resultar em nós sendo despejados do cluster se não puderem enviar batimentos cardíacos em um período de tempo razoável.
- Configuração de interfaces usadas para conexão a nós de bloco pelo protocolo desejado (por exemplo: ISCSI/iSER, NVMe/IB, NVMe/RoCE, FCP, etc.)

Passos

Fazendo referência ao esquema de endereçamento IP definido na "Planeie o sistema de ficheiros"seção, para cada nó de arquivo no cluster, crie um arquivo `host_vars/<FILE_NODE_HOSTNAME>/yml` e o preencha da seguinte forma:

1. Na parte superior, especifique o IP ou o nome de host que o Ansible deve usar para SSH para o nó e gerenciá-lo:

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

2. Configure IPs adicionais que podem ser usados para tráfego de cluster:
   a. Se o tipo de rede for "InfiniBand (usando 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 o tipo de rede for "RDMA em 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 o tipo de rede for "Ethernet (apenas TCP, sem 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. Indique quais IPs devem ser usados para tráfego de cluster, com IPs preferenciais listados acima:

```
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.
```

Os IPS configurados na etapa dois não serão usados como IPs de cluster, a menos que sejam incluídos na `beegfs_ha_cluster_node_ips` lista. Isso permite configurar IPs/interfaces adicionais usando o Ansible que podem ser usados para outros fins, se desejado.

4. Se o nó de arquivo precisar se comunicar com nós de bloco por um protocolo baseado em IP, os IPs precisarão ser configurados na interface apropriada e quaisquer pacotes necessários para esse protocolo instalado/configurado.

a. Se estiver a utilizar "ISCSI":

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

b. Se estiver a utilizar "Iser":

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

c. Se estiver a utilizar "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 estiver a utilizar "NVMe/RoCE":

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

e. Outros protocolos:

   i. Se estiver usando "NVMe/FC"o , a configuração de interfaces individuais não é necessária. A implantação do cluster do BeeGFS detetará automaticamente o protocolo e instalará/configurará os requisitos conforme necessário. Se você estiver usando uma malha para conectar nós de arquivo e bloco, verifique se os switches estão adequadamente zoneados seguindo as práticas recomendadas do fornecedor de switch e do NetApp.

   ii. O uso de FCP ou SAS não requer a instalação ou configuração de software adicional. Se estiver usando FCP, verifique se os switches estão adequadamente zoneados seguindo "NetApp" e as práticas recomendadas do fornecedor de switch.

   iii. O uso do SRP IB não é recomendado neste momento. Use o NVMe/IB ou iSER dependendo do suporte dos nós de bloco do e-Series.

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um único nó de arquivo.

Avançado: Alternando os adaptadores VPI do NVIDIA ConnectX entre o modo Ethernet e InfiniBand

Os adaptadores NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) suportam InfiniBand e Ethernet como a camada de transporte. A troca entre modos não é negociada automaticamente e deve ser configurada usando a <code>mstconfig</code> ferramenta incluída no <code>mstflint</code>, um pacote de código aberto que faz parte "Ferramentas NVIDIA Firmare (MFT)"do . Alterar o modo dos adaptadores só precisa ser feito uma vez. Isso pode ser feito manualmente ou incluído no inventário do Ansible como parte de qualquer interface configurada usando a <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> seção do inventário, para que ele seja verificado/aplicado automaticamente.

Por exemplo, para alterar uma interface atual no modo InfiniBand para Ethernet, para que possa ser usada no RoCE:

1. Para cada interface que você deseja configurar especifique `mstconfig` como um mapeamento (ou dicionário) que especifica `LINK_TYPE_P<N>` onde `<N>` é determinado pelo número de porta do HCA para a interface. O `<N>` valor pode ser determinado executando `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` e adicionando 1 ao último número do nome do slot PCI e convertendo para decimal.
   a. Por exemplo, fornecido `PCI_SLOT_NAME=0000:2f:00.2` (2 e 1 → porta HCA 3) → `LINK_TYPE_P3: eth`:

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

Para obter mais detalhes, consulte a para obter informações "Documentação da coleção de hosts do NetApp e-Series"sobre o tipo/protocolo de interface que está a utilizar.

Configurar nós de bloco individual

Especifique a configuração para nós de bloco individuais usando variáveis de host (host_vars).

Visão geral

Esta seção aborda o preenchimento de um `host_vars/<BLOCK_NODE_HOSTNAME>.yml` arquivo para cada nó de bloco no cluster. Esses arquivos só devem conter configuração exclusiva de um nó de bloco específico. Isso geralmente inclui:

- O nome do sistema (conforme apresentado no System Manager).
- O URL HTTPS para um dos controladores (usado para gerenciar o sistema usando sua API REST).
- Quais nós de arquivo de protocolo de storage usam para se conectar a esse nó de bloco.
- Configuração de portas de placa de interface do host (HIC), como endereços IP (se necessário).

Passos

Fazendo referência ao esquema de endereçamento IP definido na "Planeie o sistema de ficheiros"seção, para cada nó de bloco no cluster, crie um arquivo `host_vars/<BLOCK_NODE_HOSTNAME>/yml` e o preencha da seguinte forma:

1. Na parte superior, especifique o nome do sistema e o URL HTTPS para um dos controladores:

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

2. Selecione os "protocolo"nós de arquivo que serão usados para se conetar a esse nó de bloco:

   a. Protocolos suportados: `auto` `iscsi` , , `fc`, `sas`, , , `ib_srp`, `ib_iser` `nvme_ib`, , `nvme_fc`, `nvme_roce`.

   ```
   eseries_initiator_protocol: <PROTOCOL>
   ```

3. Dependendo do protocolo em uso, as portas HIC podem exigir configuração adicional. Quando necessário, a configuração da porta HIC deve ser definida de modo que a entrada superior na configuração para cada controlador corresponda com a porta física mais à esquerda em cada controlador, e a porta inferior a porta mais à direita. Todas as portas requerem uma configuração válida mesmo que não estejam em uso no momento.

   > ⓘ Consulte também a seção abaixo se você estiver usando HDR (200GB) InfiniBand ou 200GB RoCE com EF600 nós de bloco.

   a. Para 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. Para 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. Para 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. Para 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. Os protocolos FC e SAS não exigem configuração adicional. SRP não é recomendado corretamente.

Para opções adicionais para configurar portas HIC e protocolos host, incluindo a capacidade de configurar CHAP iSCSI, consulte o "documentação" incluído com a coleção SANtricity. Observação ao implantar o BeeGFS, o pool de storage, a configuração de volume e outros aspectos do provisionamento de storage serão configurados em outro lugar e não deverão ser definidos nesse arquivo.

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um nó de bloco único.

Usando InfiniBand HDR (200GBK) ou RoCE de 200GB GB com nós de bloco NetApp EF600:

Para usar o InfiniBand HDR (200GB) com o EF600, um segundo IP "virtual" deve ser configurado para cada porta física. Abaixo está um exemplo da maneira correta de configurar um EF600 equipado com a porta dupla InfiniBand HDR HIC:

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

Especifique a Configuração do nó de ficheiro Comum

Especifique a configuração de nó de arquivo comum usando variáveis de grupo (group_vars).

Visão geral

A configuração que deve ser Apple para todos os nós de arquivo é definida em `group_vars/ha_cluster.yml`. Geralmente inclui:

- Detalhes sobre como conetar e fazer login em cada nó de arquivo.
- Configuração de rede comum.
- Se as reinicializações automáticas são permitidas.
- Como o firewall e os estados selinux devem ser configurados.
- Configuração de cluster, incluindo alertas e cercas.
- Ajuste de desempenho.
- Configuração comum do serviço BeeGFS.

As opções definidas neste arquivo também podem ser definidas em nós de arquivo individuais, por exemplo, se modelos de hardware mistos estiverem em uso ou se você tiver senhas diferentes para cada nó. A configuração em nós de arquivo individuais terá precedência sobre a configuração neste arquivo.

Passos

Crie o arquivo `group_vars/ha_cluster.yml` e preencha-o da seguinte forma:

1. Indique como o nó Ansible Control deve se autenticar com os hosts remotos:

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

Particularmente para ambientes de produção, não armazene senhas em texto simples. Em vez disso, use o Ansible Vault ("Criptografia de conteúdo com o Ansible Vault"consulte) ou a `--ask-become-pass` opção ao executar o manual de estratégia. Se o `ansible_ssh_user` já for root, você pode omitir opcionalmente o `ansible_become_password`.

2. Se você estiver configurando IPs estáticos em interfaces ethernet ou InfiniBand (por exemplo, IPs de cluster) e várias interfaces estiverem na mesma sub-rede IP (por exemplo, se ib0 estiver usando 192.168.1.10/24 e IB1 estiver usando 192.168.1.11/24), tabelas e regras de roteamento IP adicionais devem ser configuradas para que o suporte multi-homed funcione corretamente. Basta ativar o gancho de configuração da interface de rede fornecido da seguinte forma:

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

3. Ao implantar o cluster, dependendo do protocolo de storage, pode ser necessário reiniciar os nós para facilitar a descoberta de dispositivos de bloco remoto (volumes e-Series) ou aplicar outros aspetos da configuração. Por padrão, os nós serão solicitados antes da reinicialização, mas você pode permitir que os nós sejam reiniciados automaticamente especificando o seguinte:

```
eseries_common_allow_host_reboot: true
```

   a. Por padrão, após uma reinicialização, para garantir que os dispositivos de bloco e outros serviços estejam prontos, o Ansible aguardará até que o systemd `default.target` seja alcançado antes de continuar com a implantação. Em alguns cenários em que o NVMe/IB está em uso, isso pode não ser longo o suficiente para inicializar, descobrir e se conetar a dispositivos remotos. Isto pode resultar na continuação prematura e falha da implementação automatizada. Para evitar isso ao usar o NVMe/IB, defina o seguinte:

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

4. Várias portas de firewall são necessárias para que os serviços do cluster BeeGFS e HA se comuniquem. A menos que você deseje configurar o firwewall manualmente (não recomendado), especifique o seguinte para que as zonas de firewall necessárias sejam criadas e as portas abertas automaticamente:

```
beegfs_ha_firewall_configure: True
```

5. Neste momento, o SELinux não é suportado, e é recomendável que o estado seja definido como desativado para evitar conflitos (especialmente quando o RDMA está em uso). Defina o seguinte para garantir que o SELinux esteja desativado:

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

6. Configure a autenticação para que os nós de arquivo possam se comunicar, ajustando os padrões conforme necessário com base nas políticas da sua organização:

```
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. Com base na "Planeie o sistema de ficheiros" seção, especifique o IP de gerenciamento do BeeGFS para este sistema de arquivos:

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

> Embora pareça redundante, `beegfs_ha_mgmtd_floating_ip` é importante quando você escala o sistema de arquivos BeeGFS além de um único cluster de HA. Os clusters de HA subsequentes são implantados sem um serviço de gerenciamento BeeGFS adicional e apontam para o serviço de gerenciamento fornecido pelo primeiro cluster.

8. Ative alertas de e-mail, se desejado:

```
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. A ativação do esgrima é fortemente recomendada, caso contrário, os serviços podem ser bloqueados de iniciar em nós secundários quando o nó primário falhar.
   a. Ative a vedação globalmente especificando o seguinte:

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

      i. Observação qualquer suporte "propriedade cluster" também pode ser especificado aqui, se necessário. Normalmente, não é necessário ajustá-los, uma vez que a função BeeGFS HA é fornecida com uma série de testes bem testados"predefinições".

   b. Em seguida, selecione e configure um agente de esgrima:
      i. Opção 1: Para ativar a vedação utilizando unidades de distribuição de energia (PDUs) da 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. Opção 2: Para habilitar o esgrima usando as APIs do Redfish fornecidas pelo Lenovo XCC (e outros 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
```

iii. Para obter detalhes sobre a configuração de outros agentes de vedação, consulte o "Documentação RedHat".

10. A função BeeGFS HA pode aplicar muitos parâmetros de ajuste diferentes para ajudar a otimizar ainda mais a performance. Estes incluem a otimização da utilização da memória do kernel e a e/S do dispositivo de bloco, entre outros parâmetros. A função é fornecida com um conjunto razoável de "predefinições" com base em testes com os nós de bloco do NetApp e-Series, mas por padrão estes não são aplicados a menos que você especifique:

```
beegfs_ha_enable_performance_tuning: True
```

a. Se necessário, especifique também quaisquer alterações ao ajuste de desempenho padrão aqui. Consulte a documentação completa "parâmetros de ajuste de desempenho"para obter detalhes adicionais.

11. Para garantir que os endereços IP flutuantes (às vezes conhecidos como interfaces lógicas) usados para serviços BeeGFS possam fazer failover entre nós de arquivo, todas as interfaces de rede devem ser nomeadas de forma consistente. Por padrão, os nomes de interface de rede são gerados pelo kernel, o que não é garantido para gerar nomes consistentes, mesmo em modelos de servidor idênticos com adaptadores de rede instalados nos mesmos slots PCIe. Isso também é útil ao criar inventários antes que o equipamento seja implantado e os nomes de interface gerados sejam conhecidos. Para garantir nomes de dispositivos consistentes, com base em um diagrama de blocos do servidor ou `lshw -class network -businfo` saída, especifique o mapeamento de interface lógica de endereço PCIe desejado da seguinte forma:

a. Para interfaces de rede InfiniBand (IPoIB):

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

b. Para interfaces de rede Ethernet:

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

Para evitar conflitos quando as interfaces são renomeadas (impedindo-as de serem renomeadas), você não deve usar nomes padrão potenciais como eth0, ens9f0, ib0 ou ibs4f0. Uma convenção de nomenclatura comum é usar 'e' ou 'i' para Ethernet ou InfiniBand, seguido do número do slot PCIe e uma letra para indicar a porta. Por exemplo, a segunda porta de um adaptador InfiniBand instalado no slot 3 seria: i3b.

Se você estiver usando um modelo de nó de arquivo verificado, clique "aqui" em exemplo mapeamentos de endereço PCIe para porta lógica.

12. Especifique opcionalmente a configuração que deve ser aplicada a todos os serviços BeeGFS no cluster. Os valores de configuração padrão podem ser "aqui"encontrados e a configuração por serviço é especificada em outro lugar:
    a. Serviço de gerenciamento BeeGFS:

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

    b. Serviços de metadados BeeGFS:

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

    c. Serviços BeeGFS Storage:

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

13. A partir do BeeGFS 7.2.7 e 7.3.1 "autenticação de conexão"devem ser configurados ou explicitamente desativados. Há algumas maneiras de configurar isso usando a implantação baseada no Ansible:
    a. Por padrão, a implantação configurará automaticamente a autenticação de conexão e gerará uma `connauthfile` que será distribuída para todos os nós de arquivos e usada com os serviços BeeGFS. Esse arquivo também será colocado/mantido no nó de controle do Ansible `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` onde ele deve ser mantido (com segurança) para reutilização com clientes que precisam acessar esse sistema de arquivos.
        i. Para gerar uma nova chave, especifique `-e "beegfs_ha_conn_auth_force_new=True` ao executar o manual de estratégia do Ansible. Nota isto é ignorado se a `beegfs_ha_conn_auth_secret` estiver definida.
        ii. Para opções avançadas, consulte a lista completa de padrões incluídos no "BeeGFS HA função".
    b. Um segredo personalizado pode ser usado definindo o seguinte em `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. A autenticação de conexão pode ser totalmente desativada (NÃO recomendada):

```
beegfs_ha_conn_auth_enabled: false
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa a configuração comum do nó de arquivo.

Usando InfiniBand HDR (200GBK) com nós de bloco NetApp EF600:

Para usar o InfiniBand HDR (200GB) com o EF600, o gerenciador de sub-rede deve suportar a virtualização. Se os nós de arquivo e bloco estiverem conetados usando um switch, isso precisará ser ativado no gerenciador de sub-rede para a malha geral.

Se os nós de bloco e arquivo estiverem diretamente conetados usando InfiniBand, uma instância de `opensm` deve ser configurada em cada nó de arquivo para cada interface diretamente conetada a um nó de bloco. Isso é feito especificando `configure: true` quando "configurando interfaces de storage de nós de arquivo".

Atualmente, a versão da caixa de entrada `opensm` fornecida com distribuições Linux suportadas não suporta virtualização. Em vez disso, é necessário instalar e configurar a versão do `opensm` a partir do NVIDIA OpenFabrics Enterprise Distribution (OFED). Embora a implantação usando Ansible ainda seja compatível, algumas etapas adicionais são necessárias:

1. Usando curl ou a ferramenta desejada, baixe os pacotes para a versão do OpenSM listados na "requisitos de tecnologia"seção do site do NVIDIA para `<INVENTORY>/packages/` o diretório. Por exemplo:

```
curl -o packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.3/x86_64/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm

curl -o packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.3/x86_64/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
```

2. Em `group_vars/ha_cluster.yml` Definir a seguinte configuração:

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

eseries_ib_opensm_options:
  virt_enabled: "2"
```

Especifique Common Block Node Configuration

Especifique a configuração de nó de bloco comum usando variáveis de grupo (group_vars).

Visão geral

A configuração que deve ser Apple para todos os nós de bloco é definida em `group_vars/eseries_storage_systems.yml`. Geralmente inclui:

- Detalhes sobre como o nó de controle do Ansible deve se conectar aos sistemas de storage e-Series usados como nós de bloco.

- Quais versões de firmware, NVSRAM e firmware da unidade os nós devem ser executados.
- Configuração global, incluindo configurações de cache, configuração de host e configurações de como os volumes devem ser provisionados.

> As opções definidas neste arquivo também podem ser definidas em nós de bloco individuais, por exemplo, se modelos de hardware mistos estiverem em uso ou se você tiver senhas diferentes para cada nó. A configuração em nós de bloco individuais terá precedência sobre a configuração neste arquivo.

Passos

Crie o arquivo `group_vars/eseries_storage_systems.yml` e preencha-o da seguinte forma:

1. O Ansible não usa SSH para se conectar a nós de bloco e, em vez disso, usa APIs REST. Para conseguir isso, devemos definir:

```
ansible_connection: local
```

2. Especifique o nome de usuário e a senha para gerenciar cada nó. O nome de usuário pode ser omitido opcionalmente (e será padrão para admin), caso contrário, você pode especificar qualquer conta com admin Privileges. Especifique também se os certificados SSL devem ser verificados ou ignorados:

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

> Listar senhas em texto simples não é recomendado. Use o Ansible Vault ou forneça o `eseries_system_password` ao executar o Ansible usando --extra-vars.

3. Opcionalmente, especifique o firmware da controladora, NVSRAM e da unidade que devem ser instalados nos nós. Eles precisarão ser baixados para `packages/` o diretório antes de executar o Ansible. O firmware da controladora e-Series e a NVSRAM podem ser baixados "aqui" e o firmware da unidade "aqui":

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

Se essa configuração for especificada, o Ansible atualizará automaticamente todo o firmware, incluindo a reinicialização de controladores (se necessário) sem prompts adicionais. Espera-se que isso não cause interrupções na e/S do BeeGFS/host, mas possa causar uma diminuição temporária na performance.

4. Ajuste os padrões globais de configuração do sistema. As opções e valores listados aqui são comumente recomendados para BeeGFS no NetApp, mas podem ser ajustados se necessário:

```
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. Configurar padrões globais de provisionamento de volume. As opções e valores listados aqui são comumente recomendados para BeeGFS no NetApp, mas podem ser ajustados se necessário:

```
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 necessário, ajuste a ordem pela qual o Ansible selecionará unidades para pools de storage e grupos de volumes tendo em mente as práticas recomendadas a seguir:
   a. Liste as unidades (potencialmente menores) que devem ser usadas primeiro para gerenciamento e/ou volumes de metadados e, por último, os volumes de storage.
   b. Certifique-se de equilibrar a ordem de seleção da unidade entre os canais de unidade disponíveis com base no(s) modelo(s) de compartimento de disco/compartimento de unidade. Por exemplo, com o EF600 e sem expansões, as unidades 0-11 estão no canal de unidade 1 e as unidades 12-23 estão no canal de unidade. Assim, uma estratégia para equilibrar a seleção da unidade é selecionar `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, etc. caso haja mais de um compartimento, o primeiro dígito representa a ID do compartimento da unidade.

```
# 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"
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa a configuração de nó

de bloco comum.

Defina os serviços BeeGFS

Definir o serviço de gerenciamento BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de gerenciamento BeeGFS. Apenas um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a definição de:

- O tipo de serviço (gerenciamento).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como um volume deve ser armazenado os dados para esse serviço (o destino de gerenciamento do BeeGFS).

Passos

Crie um novo arquivo `group_vars/mgmt.yml` e consulte a "Planeie o sistema de ficheiros" seção preencha-o da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de gerenciamento BeeGFS:

```
beegfs_service: management
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. Normalmente, isso não é necessário para o serviço de gerenciamento, a menos que você precise ativar cotas, no entanto, qualquer parâmetro de configuração suportado do `beegfs-mgmtd.conf` pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `storeMgmtdDirectory` `connAuthFile` , , `connDisableAuthentication`, , `connInterfacesFile` E `connNetFilterFile`.

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

3. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o destino de gerenciamento do BeeGFS no qual esse serviço armazenará dados de acordo com as seguintes diretrizes:
   a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).
   b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).
   c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

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

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de gerenciamento BeeGFS.

Defina o serviço de metadados do BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de metadados do BeeGFS. Pelo menos um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a definição de:

- O tipo de serviço (metadados).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como um volume deve ser armazenado os dados para esse serviço (o destino de metadados do BeeGFS).

Passos

Fazendo referência à "Planeie o sistema de ficheiros" seção, crie um arquivo em `group_vars/meta_<ID>.yml` para cada serviço de metadados no cluster e preencha-os da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de metadados do BeeGFS:

```
beegfs_service: metadata
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. No mínimo, você deve especificar a porta TCP e UDP desejada, no entanto, qualquer parâmetro de configuração suportado `beegfs-meta.conf` também pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `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. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o destino de metadados do BeeGFS no qual esse serviço armazenará dados de acordo com as diretrizes a seguir (isso também configurará a opção automaticamente `storeMetaDirectory`):

a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).

b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).

c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

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

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de metadados BeeGFS.

Defina o serviço de storage BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de storage BeeGFS. Pelo menos um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a definição de:

- O tipo de serviço (armazenamento).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como os volumes devem ser armazenados para esse serviço (os destinos de storage do BeeGFS).

Passos

Fazendo referência à "Planeie o sistema de ficheiros" seção, crie um arquivo em `group_vars/stor_<ID>.yml` para cada serviço de armazenamento no cluster e preencha-os da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de storage BeeGFS:

```
beegfs_service: storage
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. No mínimo, você deve especificar a porta TCP e UDP desejada, no entanto, qualquer parâmetro de configuração suportado `beegfs-storage.conf` também pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `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. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o(s) destino(s) de storage BeeGFS em que esse serviço armazenará dados de acordo com as diretrizes a seguir (isso também configurará automaticamente a `storeStorageDirectory` opção):

   a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).

   b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é

recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).

c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

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

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de storage BeeGFS.

Mapear os serviços BeeGFS para nós de arquivo

Especifique quais nós de arquivo podem executar cada serviço BeeGFS usando o `inventory.yml` arquivo.

Visão geral

Esta seção descreve como criar o `inventory.yml` arquivo. Isso inclui listar todos os nós de bloco e especificar quais nós de arquivo podem executar cada serviço BeeGFS.

Passos

Crie o arquivo `inventory.yml` e preencha-o da seguinte forma:

1. Na parte superior do arquivo, crie a estrutura de inventário padrão do Ansible:

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

2. Crie um grupo que contenha todos os nós de bloco participantes deste cluster HA:

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

3. Crie um grupo que contenha todos os serviços BeeGFS no cluster e os nós de arquivo que os executarão:

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

4. Para cada serviço BeeGFS no cluster, defina o(s) nó(s) de arquivo preferencial e secundário(s) que deve(m) executar esse serviço:

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

Clique "aqui" para ver um exemplo de um arquivo de inventário completo.

Implante o sistema de arquivos BeeGFS

Visão geral do Playbook do Ansible

Implantação e gerenciamento de clusters de HA do BeeGFS com o Ansible.

Visão geral

As seções anteriores descreveram as etapas necessárias para criar um inventário do Ansible que representa um cluster de HA do BeeGFS. Esta seção apresenta a automação do Ansible desenvolvida pela NetApp para implantar e gerenciar o cluster.

Ansible: Conceitos-chave

Antes de prosseguir, é útil estar familiarizado com alguns conceitos-chave do Ansible:

- As tarefas a serem executadas em um inventário do Ansible são definidas no que é conhecido como **manual de estratégia**.
  - A maioria das tarefas no Ansible foi projetada para ser **idempotente**, o que significa que elas podem ser executadas várias vezes para verificar se a configuração/estado desejado ainda é aplicada sem quebrar coisas ou fazer atualizações desnecessárias.
- A menor unidade de execução no Ansible é um **módulo**.
  - Os playbooks típicos usam vários módulos.
    - Exemplos: Baixar um pacote, atualizar um arquivo de configuração, iniciar/ativar um serviço.
  - O NetApp distribui módulos para automatizar os sistemas NetApp e-Series.
- A automação complexa é melhor empacotada como uma função.
  - Essencialmente um formato padrão para distribuir um manual reutilizável.
  - O NetApp distribui funções para hosts Linux e sistemas de arquivos BeeGFS.

Papel BeeGFS HA para Ansible: Conceitos-chave

Toda a automação necessária para implantar e gerenciar cada versão do BeeGFS no NetApp funciona como uma função do Ansible e é distribuída como parte "Coleção Ansible do NetApp e-Series para BeeGFS"do :

- Essa função pode ser pensada como algures entre um **instalador** e um moderno mecanismo de implantação/gerenciamento* para o BeeGFS.
  - Aplica infraestrutura moderna como práticas e filosofias de código para simplificar o gerenciamento da infraestrutura de armazenamento em qualquer escala.
  - Semelhante a como o "Kubespray" projeto permite que os usuários implantem/mantenham toda uma distribuição do Kubernetes para uma infraestrutura de computação com escalabilidade horizontal.
- Essa função é o formato **definido por software** que o NetApp usa para empacotar, distribuir e manter o BeeGFS nas soluções NetApp.
  - Esforce-se para criar uma experiência "semelhante a um dispositivo" sem a necessidade de distribuir uma distribuição Linux inteira ou uma imagem grande.
  - Inclui agentes de recursos de cluster compatíveis com OCF (Open Cluster Framework) da NetApp para destinos BeeGFS personalizados, endereços IP e monitoramento que fornecem integração inteligente com o fabricante de pacemaker/BeeGFS.
- Esta função não é simplesmente "automação" de implantação e destina-se a gerenciar todo o ciclo de vida do sistema de arquivos, incluindo:
  - Aplicação de alterações e atualizações de configuração por serviço ou em todo o cluster.
  - Automatizar a recuperação e a recuperação de clusters após problemas de hardware serem resolvidos.
  - Simplificação do ajuste de performance com base em valores padrão definidos com base em testes abrangentes com o BeeGFS e o NetApp volumes.
  - Verificando e corrigindo o desvio de configuração.

O NetApp também fornece uma função Ansible para "Clientes BeeGFS", que pode ser usada, como opção, para instalar o BeeGFS e montar sistemas de arquivos nos nós de computação/GPU/login.

Implante o cluster BeeGFS HA

Especifique quais tarefas devem ser executadas para implantar o cluster BeeGFS HA usando um manual de estratégia.

Visão geral

Esta seção aborda como montar o manual de estratégia padrão usado para implantar/gerenciar o BeeGFS no NetApp.

Passos

Crie o Playbook do Ansible

Crie o arquivo `playbook.yml` e preencha-o da seguinte forma:

1. Primeiro defina um conjunto de tarefas (comumente chamadas de a "reproduzir") que devem ser executadas somente em nós de bloco do NetApp e-Series. Usamos uma tarefa de pausa para avisar antes de executar a instalação (para evitar execuções acidentais de playbooks) e depois importar a `nar_santricity_management` função. Essa função lida com a aplicação de qualquer configuração geral do sistema definida em `group_vars/eseries_storage_systems.yml` arquivos individuais ou individuais `host_vars/<BLOCK NODE>.yml`.

```
- 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. Defina a reprodução que será executada em todos os nós de arquivo e bloco:

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

3. Dentro desta jogada, podemos opcionalmente definir um conjunto de "pré-tarefas" que devem ser

executadas antes de implantar o cluster HA. Isso pode ser útil para verificar/instalar quaisquer pré-requisitos como Python. Também podemos injetar quaisquer verificações antes do voo, por exemplo, verificando as tags Ansible fornecidas são compatíveis:

```
  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. Por último, essa peça importa a função BeeGFS HA para a versão do BeeGFS que você deseja implantar:

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

> Uma função BeeGFS HA é mantida para cada versão maior.menor suportada do BeeGFS. Isso permite que os usuários escolham quando querem atualizar versões principais/secundárias. Atualmente, o BeeGFS 7,3.x (`beegfs_7_3`) ou o BeeGFS 7,2.x (`beegfs_7_2`) são compatíveis. Por padrão, ambas as funções implantarão a versão mais recente do patch BeeGFS no momento do lançamento, embora os usuários possam optar por substituir isso e implantar o patch mais recente, se desejado. Consulte o mais recente "guia de atualização" para obter mais detalhes.

5. Opcional: Se você deseja definir tarefas adicionais, lembre-se de que as tarefas devem ser direcionadas a `all` hosts (incluindo os sistemas de storage e-Series) ou apenas aos nós de arquivos. Se necessário, defina uma nova reprodução segmentando especificamente os nós de arquivos usando `- hosts: ha_cluster`o .

Clique "aqui" para obter um exemplo de um arquivo de manual completo.

Instalar as coleções do NetApp Ansible

A coleção BeeGFS para Ansible e todas as dependências são mantidas "Ansible Galaxy"no . No nó de controle do Ansible, execute o seguinte comando para instalar a versão mais recente:

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

Embora normalmente não seja recomendado, também é possível instalar uma versão específica da coleção:

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

Execute o Playbook

A partir do diretório no nó de controle do Ansible que contém os `inventory.yml` arquivos e `playbook.yml`, execute o manual de estratégia da seguinte forma:

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

Com base no tamanho do cluster, a implantação inicial pode levar mais de 20 minutos. Se a implantação falhar por qualquer motivo, basta corrigir quaisquer problemas (por exemplo, falta de cabeamento, nó não foi iniciado, etc.) e reiniciar o manual de estratégia do Ansible.

Ao "configuração comum de nó de arquivo"especificar o , se você escolher a opção padrão para que o Ansible gerencie automaticamente a autenticação baseada na conexão, um `connAuthFile` segredo usado como um segredo compartilhado agora pode ser encontrado em `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (por padrão). Qualquer cliente que precise acessar o sistema de arquivos precisará usar esse segredo compartilhado. Isso é Tratado automaticamente se os clientes forem configurados usando o "Função de cliente BeeGFS".

Implante clientes BeeGFS

Como opção, o Ansible pode ser usado para configurar clientes BeeGFS e montar o sistema de arquivos.

Visão geral

Acessar os sistemas de arquivos BeeGFS requer a instalação e a configuração do cliente BeeGFS em cada nó que precisa montar o sistema de arquivos. Esta seção documenta como executar essas tarefas usando o "Função do Ansible"disponível .

Passos

Crie o Arquivo de Inventário do Cliente

1. Se necessário, configure o SSH sem senha do nó de controle do Ansible para cada um dos hosts que você deseja configurar como clientes BeeGFS:

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

2. Em `host_vars/`, crie um arquivo para cada cliente BeeGFS nomeado `<HOSTNAME>.yml` com o seguinte conteúdo, preenchendo o texto do espaço reservado com as informações corretas para o seu ambiente:

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

3. Inclua, opcionalmente, uma das opções a seguir, se quiser usar as funções da coleção de hosts do NetApp e-Series para configurar interfaces InfiniBand ou Ethernet para clientes se conetarem a nós de arquivos BeeGFS:

   a. Se o tipo de rede for "InfiniBand (usando 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 o tipo de rede for "RDMA em 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 o tipo de rede for "Ethernet (apenas TCP, sem 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. Crie um novo `client_inventory.yml` arquivo e especifique o usuário que o Ansible deve usar para se conetar a cada cliente. A senha que o Ansible deve usar para encaminhar privilégios (isso requer `ansible_ssh_user` root ou ter sudo Privileges):

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

Não armazene senhas em texto simples. Em vez disso, use o Ansible Vault (consulte o "Documentação do Ansible" para criptografar conteúdo com o Ansible Vault) ou use a `--ask-become-pass` opção ao executar o manual de estratégia.

5. No `client_inventory.yml` arquivo, liste todos os hosts que devem ser configurados como clientes BeeGFS no `beegfs_clients` grupo e, em seguida, consulte os comentários inline e descomente qualquer configuração adicional necessária para criar o módulo do kernel do cliente BeeGFS no seu 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 NVIDIA OFED drivers and they are
already installed:
        #eseries_ib_skip: True # Skip installing inbox drivers when
using the IPoIB role.
        #beegfs_client_ofed_enable: True
        #beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"

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

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

Ao usar os drivers NVIDIA OFED, certifique-se de que beegfs_client_ofed_include_path aponte para o "caminho de inclusão de cabeçalho" correto para a instalação do Linux. Para obter mais informações, consulte a documentação do BeeGFS para "Suporte RDMA".

6. No `client_inventory.yml` arquivo, liste os sistemas de arquivos BeeGFS que você deseja montar em qualquer um definido anteriormente `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 partir do BeeGFS 7.2.7 e 7.3.1 "autenticação de conexão"devem ser configurados ou explicitamente desativados. Dependendo de como você escolhe configurar a autenticação baseada em conexão ao especificar "configuração comum de nó de arquivo", talvez seja necessário ajustar a configuração do cliente:
   a. Por padrão, a implantação do cluster de HA configurará automaticamente a autenticação de conexão e gerará uma `connauthfile` que será colocada/mantida no nó de controle do Ansible em `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Por padrão, a função de cliente BeeGFS é configurada para ler/distribuir esse arquivo aos clientes definidos no `client_inventory.yml`, e nenhuma ação adicional é necessária.
      i. Para opções avançadas, consulte a lista completa de padrões incluídos no "Função de cliente BeeGFS".
   b. Se você optar por especificar um segredo personalizado com `beegfs_ha_conn_auth_secret` especifique-o `client_inventory.yml` no arquivo também:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Se você optar por desativar totalmente a autenticação baseada em conexão com `beegfs_ha_conn_auth_enabled` o , especifique isso também no `client_inventory.yml` arquivo:

```
beegfs_ha_conn_auth_enabled: false
```

Para obter uma lista completa dos parâmetros suportados e detalhes adicionais, consulte o "Documentação completa do cliente BeeGFS". Para obter um exemplo completo de um inventário de cliente, clique "aqui"em .

Crie o arquivo de Playbook do cliente BeeGFS

1. Crie um novo arquivo `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. Opcional: Se você quiser usar as funções da coleção de hosts do NetApp e-Series para configurar interfaces para que os clientes se conetem aos sistemas de arquivos BeeGFS, importe a função correspondente ao tipo de interface que você está configurando:
   a. Se você estiver usando o InfiniBand (IPoIB):

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

   b. Se você estiver usando o RDMA em Converged Ethernet (RoCE):

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

   c. Se você estiver usando Ethernet (somente TCP, sem RDMA):

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

3. Por último, importe a função de cliente BeeGFS para instalar o software cliente e configurar as montagens do sistema de arquivos:

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

Para obter um exemplo completo de um manual de estratégia de cliente, clique "aqui"em .

Execute o Playbook do cliente BeeGFS

Para instalar/construir o cliente e montar o BeeGFS, execute o seguinte comando:

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

Verifique a implantação do BeeGFS

Verifique a implantação do sistema de arquivos antes de colocar o sistema em produção.

Visão geral

Antes de colocar o sistema de arquivos BeeGFS em produção, execute algumas verificações.

Passos

1. Faça login em qualquer cliente e execute o seguinte para garantir que todos os nós esperados estejam presentes/acessíveis, e não haja inconsistências ou outros problemas relatados:

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

2. Encerre todo o cluster e reinicie-o. A partir de qualquer nó de arquivo, execute o seguinte:

```
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. Coloque cada nó em standby e verifique se os serviços do BeeGFS são capazes de fazer failover para nós secundários. Para fazer esse login em qualquer um dos nós de arquivo e executar o seguinte:

```
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. Use ferramentas de benchmarking de desempenho, como IOR e MDTest, para verificar se o desempenho do sistema de arquivos atende às expetativa. Exemplos de testes e parâmetros comuns usados com o BeeGFS podem ser encontrados na "Verificação de design"seção do BeeGFS na arquitetura verificada do NetApp.

Testes adicionais devem ser realizados com base nos critérios de aceitação definidos para um determinado local/instalação.

Administrar clusters BeeGFS

Visão geral, conceitos-chave e terminologia

Saiba como administrar clusters de HA do BeeGFS após a implantação.

Visão geral

Esta seção destina-se aos administradores de cluster que precisam gerenciar clusters de HA do BeeGFS após a implantação. Mesmo aqueles que estão familiarizados com os clusters de HA do Linux devem ler cuidadosamente este guia, pois há várias diferenças em como gerenciar o cluster, especialmente em relação à reconfiguração devido ao uso do Ansible.

Conceitos-chave

Embora alguns desses conceitos sejam introduzidos na página principal"termos e conceitos", é útil reintroduzi-los no contexto de um cluster BeeGFS HA:

Nó de cluster: Um servidor executando serviços de pacemaker e Corosync e participando do cluster HA.

Nó de arquivo: Um nó de cluster usado para executar um ou mais serviços de gerenciamento, metadados ou storage do BeeGFS.

Nó de bloco: Um sistema de storage do NetApp e-Series que fornece storage de bloco para nós de arquivos. Esses nós não participam do cluster BeeGFS HA, pois fornecem suas próprias funcionalidades de HA autônomas. Cada nó consiste em duas controladoras de storage que fornecem alta disponibilidade na camada de bloco.

Serviço BeeGFS: Um serviço de gerenciamento, metadados ou storage do BeeGFS. Cada nó de arquivo executará um ou mais serviços que usarão volumes no nó de bloco para armazenar seus dados.

Bloco básico: Uma implantação padronizada de nós de arquivo BeeGFS, nós de bloco de e-Series e serviços BeeGFS executados neles, o que simplifica o dimensionamento de um sistema de arquivos/cluster BeeGFS HA seguindo uma arquitetura verificada pelo NetApp. Clusters de HA personalizados também são compatíveis, mas geralmente seguem uma abordagem de componentes básicos semelhante para simplificar o dimensionamento.

Cluster do BeeGFS HA: Um número dimensionável de nós de arquivo usados para executar serviços do BeeGFS com o respaldo de nós de bloco para armazenar dados do BeeGFS de maneira altamente disponível. Desenvolvido com base em componentes de código aberto comprovados pela indústria, a Pacemaker e o Corosync, usando o Ansible para embalagem e implantação.

Serviços de cluster: refere-se aos serviços de pacemaker e Corosync executados em cada nó que participa do cluster. Observação é possível que um nó não execute nenhum serviço BeeGFS e apenas participe do cluster como nó "tiebreaker", caso haja apenas a necessidade de dois nós de arquivo.

Recursos de cluster: para cada serviço BeeGFS executado no cluster, você verá um recurso de monitoramento BeeGFS e um grupo de recursos contendo recursos para destino(s) BeeGFS, endereço(s) IP (IPs flutuantes) e o próprio serviço BeeGFS.

Ansible: Uma ferramenta para provisionamento de software, gerenciamento de configurações e implantação de aplicações, habilitando a infraestrutura como código. É como os clusters do BeeGFS são empacotados para simplificar o processo de implantação, reconfiguração e atualização do BeeGFS no NetApp.

PCs: Uma interface de linha de comando disponível a partir de qualquer um dos nós de arquivo no cluster usado para consultar e controlar o estado de nós e recursos no cluster.

Terminologia comum

Failover: cada serviço BeeGFS tem um nó de arquivo preferido em que será executado, a menos que esse nó falhe. Quando um serviço BeeGFS está sendo executado no nó de arquivo secundário/não preferencial, diz-se que está em failover.

Failback: o ato de mover os serviços BeeGFS de um nó de arquivo não preferencial de volta para o nó preferido.

Par de HA: dois nós de arquivos que podem acessar o mesmo conjunto de nós de bloco às vezes são referidos como um par de HA. Este é um termo comum usado em todo o NetApp para se referir a dois controladores de storage ou nós que podem "assumir o controle" uns dos outros.

Modo de manutenção: desativa todo o monitoramento de recursos e impede que o pacemaker mova ou gerencie recursos no cluster (consulte também a seção em "modo de manutenção").

Cluster de HA: um ou mais nós de arquivos executando serviços BeeGFS que podem fazer o failover entre vários nós no cluster para criar um sistema de arquivos BeeGFS altamente disponível. Normalmente, os nós de arquivos são configurados em pares de HA que podem executar um subconjunto dos serviços BeeGFS no cluster.

Quando usar o Ansible versus a ferramenta PCs

Quando você deve usar o Ansible versus a ferramenta de linha de comando PCs para gerenciar o cluster de HA?

Todas as tarefas de implantação e reconfiguração de cluster devem ser concluídas usando o Ansible a partir de um nó de controle externo do Ansible. Alterações temporárias no estado do cluster (por exemplo, colocar nós dentro e fora de espera) normalmente serão realizadas fazendo login em um nó do cluster (de preferência um que não esteja degradado ou prestes a ser submetido a manutenção) e usando a ferramenta de linha de comando PCs.

Usar o Ansible, modificar qualquer configuração do cluster, incluindo recursos, restrições, propriedades e serviços do BeeGFS. Manter uma cópia atualizada do inventário e do manual de estratégia do Ansible (idealmente no controle de origem para controlar alterações) faz parte da manutenção do cluster. Quando você precisar fazer alterações na configuração, atualize o inventário e execute novamente o manual de estratégia do Ansible que importa a função de HA do BeeGFS.

A função de HA tratará da colocação do cluster no modo de manutenção e depois fará as alterações necessárias antes de reiniciar os serviços do BeeGFS ou do cluster para aplicar a nova configuração. Como as reinicializações completas de nós geralmente não são necessárias fora da implantação inicial, a reinstalação do Ansible geralmente é considerada um procedimento "seguro", mas é sempre recomendada durante períodos de manutenção ou horas extras, caso os serviços do BeeGFS precisem ser reiniciados. Essas reinicializações geralmente não devem causar erros de aplicativo, mas podem prejudicar o desempenho (o que alguns aplicativos podem lidar melhor que outros).

A reexecução do Ansible também é uma opção quando você deseja retornar todo o cluster a um estado totalmente ideal e pode, em alguns casos, ser capaz de recuperar o estado do cluster com mais facilidade do que o uso de PCs. Especialmente durante uma emergência em que o cluster está inativo por algum motivo, uma vez que todos os nós estão voltando a executar o Ansible pode recuperar o cluster de forma mais rápida e confiável do que tentar usar PCs.

Examine o estado do cluster

Use PCs para visualizar o estado do cluster.

Visão geral

Executar `pcs status` a partir de qualquer um dos nós de cluster é a maneira mais fácil de ver o estado geral do cluster e o status de cada recurso (como os serviços BeeGFS e suas dependências). Esta seção percorre o que você encontrará na saída do `pcs status` comando.

Compreender a saída de `pcs status`

Execute `pcs status` em qualquer nó de cluster onde os serviços de cluster (Pacemaker e Corosync) são iniciados. A parte superior da saída irá mostrar-lhe um resumo do cluster:

```
[root@beegfs_01 ~]# pcs status
Cluster name: hacluster
Cluster Summary:
  * Stack: corosync
  * Current DC: beegfs_01 (version 2.0.5-9.el8_4.3-ba59be7122) - partition
with quorum
  * Last updated: Fri Jul  1 13:37:18 2022
  * Last change:  Fri Jul  1 13:23:34 2022 by root via cibadmin on
beegfs_01
  * 6 nodes configured
  * 235 resource instances configured
```

A seção abaixo lista os nós no cluster:

```
Node List:
  * Node beegfs_06: standby
  * Online: [ beegfs_01 beegfs_02 beegfs_04 beegfs_05 ]
  * OFFLINE: [ beegfs_03 ]
```

Isso indica notavelmente todos os nós que estão em standby ou offline. Os nós em modo de espera ainda estão participando do cluster, mas marcados como não qualificados para executar recursos. Os nós que estão offline indicam que os serviços de cluster não estão sendo executados nesse nó, seja devido a ser parado manualmente ou porque o nó foi reinicializado/encerrado.

Quando os nós são iniciados pela primeira vez, os serviços de cluster serão interrompidos e precisam ser iniciados manualmente para evitar falhas acidentais de recursos para um nó que não seja saudável.

Se os nós estiverem em standby ou offline devido a um motivo não administrativo (por exemplo, uma falha), o texto adicional será exibido ao lado do estado do nó entre parênteses. Por exemplo, se o esgrima estiver desativado e um recurso encontrar uma falha, você verá `Node <HOSTNAME>: standby (on-fail)`. Outro

estado possível é `Node <HOSTNAME>: UNCLEAN (offline)`, que será visto brevemente como um nó está sendo cercado, mas persistirá se o fencing falhar, indicando que o cluster não pode confirmar o estado do nó (isso pode impedir que os recursos comecem em outros nós).

A próxima seção mostra uma lista de todos os recursos no cluster e seus estados:

```
Full List of Resources:
  * mgmt-monitor    (ocf::eseries:beegfs-monitor):   Started beegfs_01
  * Resource Group: mgmt-group:
    * mgmt-FS1  (ocf::eseries:beegfs-target):    Started beegfs_01
    * mgmt-IP1  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-IP2  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-service  (systemd:beegfs-mgmtd):  Started beegfs_01
[...]
```

Semelhante aos nós, o texto adicional será exibido ao lado do estado do recurso entre parênteses se houver algum problema com o recurso. Por exemplo, se o pacemaker solicitar uma parada de recurso e ele não for concluído dentro do tempo alocado, o pacemaker tentará cercar o nó. Se o esgrima estiver desativado ou a operação de esgrima falhar, o estado do recurso será `FAILED <HOSTNAME> (blocked)` e o pacemaker não poderá iniciá-lo em um nó diferente.

Vale a pena notar que os clusters de HA do BeeGFS utilizam vários agentes de recursos OCF personalizados otimizados pelo BeeGFS. Em particular, o monitor BeeGFS é responsável por acionar um failover quando os recursos do BeeGFS em um nó específico não estiverem disponíveis.

Reconfigure o cluster de HA e o BeeGFS

Use o Ansible para reconfigurar o cluster.

Visão geral

De um modo geral, a reconfiguração de qualquer aspecto do cluster BeeGFS HA deve ser feita atualizando seu inventário do Ansible e executando novamente `ansible-playbook` o comando. Isso inclui atualização de alertas, alteração da configuração de cercas permanentes ou ajuste da configuração do serviço BeeGFS. Estes são ajustados usando o `group_vars/ha_cluster.yml` arquivo e uma lista completa de opções pode ser encontrada "Especifique a Configuração do nó de ficheiro Comum"na seção.

Consulte abaixo para obter detalhes adicionais sobre opções de configuração selecionadas que os administradores devem estar cientes ao executar a manutenção ou a manutenção do cluster.

Como desativar e ativar o Esgrima

O esgrima é ativado/exigido por padrão ao configurar o cluster. Em alguns casos, pode ser desejável desativar temporariamente o esgrima para garantir que os nós não sejam acidentalmente desligados ao executar determinadas operações de manutenção (como atualizar o sistema operacional). Embora isso possa ser desativado manualmente, há compensações que os administradores devem estar cientes.

Opção 1: Desativar cercas usando o Ansible (recomendado).

Quando a vedação é desativada usando o Ansible, a ação on-fail do monitor BeeGFS é alterada de "cerca"

para "espera". Isso significa que, se o monitor BeeGFS detetar uma falha, ele tentará colocar o nó em espera e fazer o failover de todos os serviços BeeGFS. Fora da solução de problemas/testes ativos, isso geralmente é mais desejável do que a opção 2. A desvantagem é que se um recurso não parar no nó original, ele será bloqueado de começar em outro lugar (é por isso que o esgrima é normalmente necessário para clusters de produção).

1. No inventário do Ansible, `groups_vars/ha_cluster.yml` adicione a seguinte configuração:

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

2. Execute novamente o manual de estratégia do Ansible para aplicar as alterações ao cluster.

Opção 2: Desativar a vedação manualmente.

Em alguns casos, você pode desativar temporariamente o esgrima sem executar novamente o Ansible, talvez para facilitar a solução de problemas ou o teste do cluster.

> Nessa configuração, se o monitor BeeGFS detetar uma falha, o cluster tentará interromper o grupo de recursos correspondente. Ele NÃO acionará um failover completo ou tentará reiniciar ou mover o grupo de recursos afetado para outro host. Para recuperar, solucione quaisquer problemas em seguida, execute `pcs resource cleanup` ou coloque manualmente o nó em espera.

Passos:

1. Para determinar se a vedação (stonith) está globalmente ativada ou desativada, execute: `pcs property show stonith-enabled`
2. Para desativar a execução de esgrima: `pcs property set stonith-enabled=false`
3. Para ativar a execução de esgrima: `pcs property set stonith-enabled=true`

Observação: Essa configuração será substituída na próxima vez que você executar o manual do Ansible.

Atualizar os componentes do cluster HA

Atualize a versão BeeGFS

Siga estas etapas para atualizar a versão BeeGFS do cluster de HA usando o Ansible.

Visão geral

BeeGFS segue um `major.minor.patch` esquema de controle de versão. As funções do BeeGFS HA Ansible são fornecidas para cada versão com suporte `major.minor` (por exemplo, `beegfs_ha_7_2` e `beegfs_ha_7_3`). Cada função de HA é fixada à versão de patch BeeGFS mais recente disponível no momento do lançamento da coleção Ansible.

O Ansible deve ser usado em todas as atualizações do BeeGFS, incluindo a migração entre as versões principal, secundária e de patch do BeeGFS. Para atualizar o BeeGFS, primeiro você precisará atualizar a coleção BeeGFS Ansible, que também abordará as correções e os aprimoramentos mais recentes da

automação de implantação/gerenciamento e do cluster de HA subjacente. Mesmo depois de atualizar para a versão mais recente da coleção, BeeGFS não será atualizado até `ansible-playbook` que seja executado com o `-e "beegfs_ha_force_upgrade=true"` conjunto.

> Para obter mais informações sobre as versões BeeGFS, consulte "Documentação do BeeGFS Upgrade".

Caminhos de atualização testados

Cada versão da coleção BeeGFS é testada com versões específicas do BeeGFS para garantir a interoperabilidade entre todos os componentes. Os testes também são realizados para garantir que as atualizações possam ser realizadas a partir das versões do BeeGFS compatíveis com a última versão da coleção, para as compatíveis na versão mais recente.

Versão original	Versão de atualização	Multirail	Detalhes
7.2.6	7.3.2	Sim	Atualizando a coleção beegfs de v3,0.1 para v3,1.0, multirail adicionado
7.2.6	7.2.8	Não	Atualizando a coleção beegfs de v3,0.1 para v3,1.0
7.2.8	7.3.1	Sim	Atualização usando beegfs coleção v3,1.0, multi-rail adicionado
7.3.1	7.3.2	Sim	Atualize usando a coleção beegfs v3,1.0
7.3.2	7.4.1	Sim	Atualize usando a coleção beegfs v3,2.0
7.4.1	7.4.2	Sim	Atualize usando a coleção beegfs v3,2.0

Etapas de atualização do BeeGFS

As seções a seguir fornecem etapas para atualizar a coleção BeeGFS Ansible e o próprio BeeGFS. Preste atenção especial a qualquer passo extra para atualizar as versões BeeGFS Major ou menor.

Passo 1: Atualize a coleção BeeGFS

Para atualizações de coleção com acesso ao "Ansible Galaxy", execute o seguinte comando:

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

Para atualizações de coleção offline, faça o download da coleção "Ansible Galaxy"clicando no desejado `Install Version` e,em seguida `Download tarball`, . Transfira o tarball para o nó de controle do Ansible e execute o seguinte comando.

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

Consulte "Instalando coleções" para obter mais informações.

Etapa 2: Atualize o inventário do Ansible

Faça todas as atualizações necessárias ou desejadas para os arquivos de inventário do Ansible do cluster. Consulte "Notas de atualização da versão"a seção abaixo para obter detalhes sobre seus requisitos de atualização específicos. Consulte "Visão geral do Ansible Inventory"a seção para obter informações gerais sobre como configurar seu inventário BeeGFS HA.

Etapa 3: Atualizar o manual do Ansible (somente ao atualizar versões principais ou secundárias)

Se você estiver se movendo entre versões maiores ou menores, no `playbook.yml` arquivo usado para implantar e manter o cluster, atualize o nome da `beegfs_ha_<VERSION>` função para refletir a versão desejada. Por exemplo, se você quiser implantar o BeeGFS 7,4, isso `beegfs_ha_7_4` seria :

```
- hosts: all
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.beegfs
  tasks:
    - name: Ensure BeeGFS HA cluster is setup.
      ansible.builtin.import_role:  # import_role is required for tag
availability.
        name: beegfs_ha_7_4
```

Para obter mais detalhes sobre o conteúdo deste arquivo de manual de estratégia, consulte "Implante o cluster BeeGFS HA"a seção.

Passo 4: Execute a atualização BeeGFS

Para aplicar a atualização BeeGFS:

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

Nos bastidores, o papel BeeGFS HA vai lidar com:

- Verifique se o cluster está no estado ideal com cada serviço BeeGFS localizado no nó preferido.
- Coloque o cluster no modo de manutenção.
- Atualize os componentes do cluster HA (se necessário).
- Atualize cada nó de arquivo, um de cada vez, da seguinte forma:
  - Coloque-a em standby e faça failover de seus serviços para o nó secundário.
  - Atualize os pacotes BeeGFS.
  - Serviços de retorno.
- Mova o cluster para fora do modo de manutenção.

Notas de atualização da versão

Atualização do BeeGFS versão 7.2.6 ou 7.3.0

Alterações na autenticação baseada em conexão

As versões BeeGFS lançadas após 7.3.1 não permitirão mais que os serviços iniciem sem especificar uma `connAuthFile` configuração ou `connDisableAuthentication=true` no arquivo de configuração do serviço. É altamente recomendável habilitar a segurança de autenticação baseada em conexão. Consulte "Autenticação baseada em conexão BeeGFS" para obter mais informações.

Por padrão, as `beegfs_ha*` funções gerarão e distribuirão esse arquivo, adicionando-o também ao nó de controle do Ansible em `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`. A `beegfs_client` função também irá verificar a presença deste ficheiro e fornecê-lo aos clientes, se disponível.

> Se a `beegfs_client` função não foi usada para configurar clientes, esse arquivo precisará ser distribuído manualmente para cada cliente e a `connAuthFile` configuração no `beegfs-client.conf` conjunto de arquivos para usá-lo. Ao atualizar a partir de uma versão anterior do BeeGFS onde a autenticação baseada em conexão não foi ativada, os clientes perderão o acesso a menos que a autenticação baseada em conexão seja desativada como parte da atualização definindo `beegfs_ha_conn_auth_enabled: false` em `group_vars/ha_cluster.yml` (não recomendado).

Para obter detalhes adicionais e opções de configuração alternativas, consulte a etapa para configurar a autenticação de conexão na "Especifique a Configuração do nó de ficheiro Comum"seção.

Atualizar o storage array do e-Series

Siga estas etapas para atualizar os storage arrays e-Series do cluster de HA (nós de bloco).

Visão geral

Manter os storage arrays NetApp e-Series atualizados do seu cluster de HA com o firmware mais recente garante desempenho ideal e segurança aprimorada. As atualizações de firmware para a matriz de armazenamento são aplicadas usando os arquivos SANtricity os, NVSRAM e firmware da unidade.

> Embora os storage arrays possam ser atualizados on-line com o cluster de HA, é recomendável colocar o cluster no modo de manutenção para todas as atualizações.

Bloquear etapas de atualização do nó

As etapas a seguir descrevem como atualizar o firmware dos storages de armazenamento usando a `Netapp_Eseries.Santricity` coleção Ansible. Antes de prosseguir, reveja o "Considerações sobre a atualização" para atualizar os sistemas e-Series.

> A atualização para o SANtricity os 11,80 ou versões posteriores só é possível a partir de 11.71.5P1. O storage array deve primeiro ser atualizado para 11.70.5P1 antes de aplicar novas atualizações.

1. Verifique se o nó de controle do Ansible está usando a coleção mais recente do SANtricity Ansible.
   - Para atualizações de coleção com acesso ao "Ansible Galaxy", execute o seguinte comando:

     ```
     ansible-galaxy collection install netapp_eseries.santricity --upgrade
     ```

   - Para atualizações off-line, baixe o tarball de coleta de "Ansible Galaxy", transfira-o para o nó de controle e execute:

     ```
     ansible-galaxy collection install netapp_eseries-santricity-
     <VERSION>.tar.gz --upgrade
     ```

     Consulte "Instalando coleções" para obter mais informações.

2. Obtenha o firmware mais recente para a sua matriz de armazenamento e unidades.
   a. Transfira os ficheiros de firmware.
      - **SANtricity os e NVSRAM:** navegue até o "Site de suporte da NetApp" e faça o download da versão mais recente do SANtricity os e NVSRAM para seu modelo de storage array.
      - **Drive firmware:** navegue até o "Site de firmware de disco e-Series" e faça o download do firmware mais recente para cada um dos modelos de unidade da matriz de armazenamento.
   b. Armazene os arquivos de firmware da unidade, NVSRAM e do sistema operacional SANtricity no diretório do nó de controle do Ansible `<inventory_directory>/packages`.
3. Se necessário, atualize os arquivos de inventário do Ansible do cluster para incluir todos os storage arrays (nós de bloco) que exigem atualizações. Para obter orientação, consulte "Visão geral do Ansible Inventory"a secção.
4. Garantir que o cluster esteja no estado ideal, com cada serviço BeeGFS no nó de sua preferência. "Examine o estado do cluster"Consulte para obter detalhes.
5. Coloque o cluster no modo de manutenção seguindo as instruções na "Coloque o cluster no modo de manutenção".
6. Crie um novo manual do Ansible chamado `update_block_node_playbook.yml`. Preencha o manual com o seguinte conteúdo, substituindo as versões do SANtricity os, NVSRAM e firmware da unidade para o caminho de atualização desejado:

```
- hosts: eseries_storage_systems
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.santricity
  vars:
    eseries_firmware_firmware: "packages/<SantricityOS>.dlp"
    eseries_firmware_nvsram: "packages/<NVSRAM>.dlp"
    eseries_drive_firmware_firmware_list:
      - "packages/<drive_firmware>.dlp"
    eseries_drive_firmware_upgrade_drives_online: true

  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

7. Para iniciar as atualizações, execute o seguinte comando a partir do nó de controle do Ansible:

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

8. Depois que o manual de estratégia for concluído, verifique se cada storage array está no estado ideal.
9. Mova o cluster para fora do modo de manutenção e valide que o cluster está no estado ideal. Cada serviço BeeGFS está no nó de sua preferência.

Manutenção e manutenção

Serviços de failover e failback

Movimentação de serviços do BeeGFS entre nós de cluster.

Visão geral

Os serviços BeeGFS podem fazer o failover entre nós no cluster para garantir que os clientes possam continuar acessando o sistema de arquivos em caso de falha ou que você precise executar a manutenção planejada. Esta seção descreve várias maneiras pelas quais os administradores podem curar o cluster depois de se recuperar de uma falha ou mover serviços manualmente entre nós.

Passos

Failover e failback

Failover (planejado)

Geralmente, quando você precisa colocar um único nó de arquivo off-line para manutenção, você vai querer mover (ou drenar) todos os serviços BeeGFS desse nó. Isso pode ser feito colocando primeiro o nó em

standby:

```
pcs node standby <HOSTNAME>
```

Depois de verificar o uso `pcs status` de todos os recursos terem sido reiniciados no nó de arquivo alternativo, você pode encerrar ou fazer outras alterações no nó conforme necessário.

Failback (após um failover planejado)

Quando você estiver pronto para restaurar os serviços BeeGFS para o nó preferido, primeiro execute `pcs status` e verifique na "Lista de nós" o status está em espera. Se o nó foi reinicializado, ele será exibido offline até que você coloque os serviços de cluster online:

```
pcs cluster start <HOSTNAME>
```

Quando o nó estiver online, retire-o do modo de espera com:

```
pcs cluster node unstandby <HOSTNAME>
```

Por último, reposicione todos os serviços BeeGFS de volta aos seus nós preferidos com:

```
pcs resource relocate run
```

Failback (após um failover não planejado)

Se um nó apresentar uma falha de hardware ou outra, o cluster de HA deve reagir automaticamente e mover seus serviços para um nó íntegro, fornecendo tempo para os administradores tomarem as medidas corretivas. Antes de prosseguir, consulte "solução de problemas"a seção para determinar a causa do failover e resolver quaisquer problemas pendentes. Depois que o nó estiver ligado novamente e saudável, você poderá prosseguir com o failback.

Quando um nó é inicializado após uma reinicialização não planejada (ou planejada), os serviços de cluster não são configurados para serem iniciados automaticamente, então você primeiro precisará colocar o nó on-line com:

```
pcs cluster start <HOSTNAME>
```

Em seguida, limpe todas as falhas de recursos e redefina o histórico de esgrima do nó:

```
pcs resource cleanup node=<HOSTNAME>
pcs stonith history cleanup <HOSTNAME>
```

Verifique se `pcs status` o nó está on-line e saudável. Por padrão, os serviços BeeGFS não irão fazer o failback automaticamente para evitar mover acidentalmente recursos de volta para um nó que não está saudável. Quando estiver pronto, retorne todos os recursos no cluster de volta aos nós preferidos com:

```
pcs resource relocate run
```

Movendo serviços individuais BeeGFS para nós de arquivos alternativos

Mova permanentemente um serviço BeeGFS para um novo nó de arquivo

Se você quiser alterar permanentemente o nó de arquivo preferido para um serviço BeeGFS individual, ajuste o inventário do Ansible para que o nó preferido seja listado primeiro e execute novamente o manual de estratégia do Ansible.

Por exemplo, neste arquivo de exemplo `inventory.yml`, beegfs_01 é o nó de arquivo preferido para executar o serviço de gerenciamento BeeGFS:

```
mgmt:
  hosts:
    beegfs_01:
    beegfs_02:
```

Reverter a ordem faria com que os serviços de gerenciamento fossem preferidos no beegfs_02:

```
mgmt:
  hosts:
    beegfs_02:
    beegfs_01:
```

Mova temporariamente um serviço BeeGFS para um nó de arquivo alternativo

Geralmente, se um nó estiver em manutenção, você vai querer usar as [etapas de failover e failback] para afastar todos os serviços desse nó.

Se, por algum motivo, você precisar mover um serviço individual para uma execução diferente do nó de arquivo:

```
pcs resource move <SERVICE>-monitor <HOSTNAME>
```

> Não especifique recursos individuais ou o grupo de recursos. Sempre especifique o nome do monitor para o serviço BeeGFS que deseja realocar. Por exemplo, para mover o serviço de gerenciamento BeeGFS para beegfs_02 execute: `pcs resource move mgmt-monitor beegfs_02`. Esse processo pode ser repetido para afastar um ou mais serviços de seus nós preferidos. Verifique se o uso `pcs status` dos serviços foi relocado/iniciado no novo nó.

Para mover um serviço BeeGFS de volta para o nó preferido, primeiro limpe as restrições de recursos temporários (repetindo essa etapa conforme necessário para vários serviços):

```
pcs resource clear <SERVICE>-monitor
```

Então, quando estiver pronto para realmente mover o(s) serviço(s) de volta para o(s) nó(s) preferido(s) executado(s):

```
pcs resource relocate run
```

Observação esse comando irá realocar quaisquer serviços que não tenham mais restrições de recursos temporários que não estejam localizados em seus nós preferidos.

Coloque o cluster no modo de manutenção

Impedir que o cluster de HA reaja acidentalmente às alterações pretendidas no ambiente.

Visão geral

Colocar o cluster no modo de manutenção desativa todo o monitoramento de recursos e impede que o pacemaker mova ou gerencie recursos no cluster. Todos os recursos permanecerão em execução em seus nós originais, independentemente de haver uma condição de falha temporária que os impeça de serem acessíveis. Os cenários em que isso é recomendado/útil incluem:

- Manutenção de rede que pode interromper temporariamente as conexões entre nós de arquivo e serviços BeeGFS.
- Atualizações do nó de bloco.
- File Node sistema operacional, kernel ou outras atualizações de pacote.

Geralmente, a única razão para colocar manualmente o cluster no modo de manutenção é impedir que ele reaja a alterações externas no ambiente. Se um nó individual no cluster exigir reparo físico, não use o modo de manutenção e simplesmente coloque esse nó em espera seguindo o procedimento acima. Observe que a reexecução do Ansible coloca o cluster automaticamente em modo de manutenção, facilitando a manutenção de software, incluindo atualizações e alterações de configuração.

Passos

Para verificar se o cluster está no modo de manutenção, execute:

```
pcs property show maintenance-mode
```

Isso retornará FALSE quando o cluster estiver operando normalmente. Para ativar a execução do modo de manutenção:

```
pcs property set maintenance-mode=true
```

Você pode verificar executando o status dos PCs e garantindo que todos os recursos mostrem "(unmanaged)". Para retirar o cluster do modo de manutenção, execute:

```
pcs property set maintenance-mode=false
```

Pare e inicie o cluster

Parar e iniciar o cluster HA com simplicidade.

Visão geral

Esta seção descreve como encerrar e reiniciar o cluster BeeGFS com simplicidade. Exemplos de cenários onde isso pode ser necessário incluem manutenção elétrica ou migração entre data centers ou racks.

Passos

Se, por algum motivo, você precisar parar todo o cluster BeeGFS e encerrar todos os serviços serão executados:

```
pcs cluster stop --all
```

Também é possível parar o cluster em nós individuais (que farão failover automático de serviços para outro nó), embora seja recomendável colocar primeiro o nó em standby (consulte "failover"a seção):

```
pcs cluster stop <HOSTNAME>
```

Para iniciar os serviços e recursos do cluster em todos os nós executados:

```
pcs cluster start --all
```

Ou inicie serviços em um nó específico com:

```
pcs cluster start <HOSTNAME>
```

Nesse momento, execute `pcs status` e verifique se o cluster e os serviços BeeGFS começam em todos os nós, e os serviços estão sendo executados nos nós que você espera.

Dependendo do tamanho do cluster, ele pode levar algum tempo (segundos a minutos) para que todo o cluster pare, ou mostrar iniciado em `pcs status`. se `pcs cluster <COMMAND>` travar por mais de cinco minutos, antes de executar "Ctrl C" para cancelar o comando, faça login em cada nó do cluster e use `pcs status` para ver se os serviços de cluster (Corosync/Pacemaker) ainda estão em execução nesse nó. De qualquer nó em que o cluster ainda esteja ativo, você pode verificar quais recursos estão bloqueando o cluster. Aborde manualmente o problema e o comando deve ser concluído ou pode ser executado novamente para parar quaisquer serviços restantes.

Substituir nós de arquivo

Substituindo um nó de arquivo se o servidor original estiver com defeito.

Visão geral

Esta é uma visão geral das etapas necessárias para substituir um nó de arquivo no cluster. Essas etapas presumem que o nó do arquivo falhou devido a um problema de hardware e foi substituído por um novo nó de arquivo idêntico.

Passos:

1. Substitua fisicamente o nó de arquivo e restaure todo o cabeamento para o nó de bloco e rede de armazenamento.
2. Reinstale o sistema operacional no nó de arquivo, incluindo a adição de assinaturas Red Hat.
3. Configurar o gerenciamento e a rede BMC no nó de arquivo.
4. Atualize o inventário do Ansible se o nome de host, IP, mapeamentos de interface PCIe para lógica ou qualquer outra coisa mudou sobre o novo nó de arquivo. Geralmente, isso não é necessário se o nó foi substituído por hardware de servidor idêntico e você estiver usando a configuração de rede original.
   a. Por exemplo, se o nome do host mudou, crie (ou renomeie) o arquivo de inventário do nó (`host_vars/<NEW_NODE>.yml`) e, em seguida, no arquivo de inventário do Ansible (`inventory.yml`), substitua o nome do nó antigo pelo novo nome do nó:

```
all:
    ...
    children:
    ha_cluster:
        children:
        mgmt:
            hosts:
            node_h1_new:    # Replaced "node_h1" with "node_h1_new"
            node_h2:
```

5. De um dos outros nós no cluster, remova o nó antigo: `pcs cluster node remove <HOSTNAME>`.

   NÃO PROSSIGA ANTES DE EXECUTAR ESTE PASSO.

6. No nó de controle do Ansible:
   a. Remova a chave SSH antiga com:

```
`ssh-keygen -R <HOSTNAME_OR_IP>`
```

   b. Configure o SSH sem senha para o nó Substituir por:

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

7. Execute novamente o manual de estratégia do Ansible para configurar o nó e adicioná-lo ao cluster:

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

8. Neste ponto, execute `pcs status` e verifique se o nó substituído está listado e executando serviços.

Expanda ou diminua o cluster

Adicione ou remova blocos de construção do cluster.

Visão geral

Esta seção documenta várias considerações e opções para ajustar o tamanho do cluster BeeGFS HA. Normalmente, o tamanho do cluster é ajustado adicionando ou removendo componentes básicos, que geralmente são dois nós de arquivo configurados como um par de HA. Também é possível adicionar ou remover nós de arquivo individuais (ou outros tipos de nós de cluster), se necessário.

Adicionando um Building Block ao cluster

Considerações

Aumentar o cluster adicionando componentes básicos adicionais é um processo simples. Antes de começar, lembre-se das restrições relativas ao número mínimo e máximo de nós de cluster em cada cluster de HA individual e determine se você deve adicionar nós ao cluster de HA existente ou criar um novo cluster de HA. Normalmente, cada componente básico consiste em dois nós de arquivo, mas três nós é o número mínimo de nós por cluster (para estabelecer quorum) e dez é o máximo recomendado (testado). Para cenários avançados, é possível adicionar um único nó "tiebreaker" que não executa nenhum serviço BeeGFS ao implantar um cluster de dois nós. Entre em Contato com o suporte da NetApp se você estiver pensando em tal implantação.

Tenha em mente essas restrições e qualquer crescimento futuro esperado de cluster ao decidir como expandir o cluster. Por exemplo, se você tiver um cluster de seis nós e precisar adicionar mais quatro nós, seria recomendável apenas iniciar um novo cluster de HA.

Lembre-se: Um único sistema de arquivos BeeGFS pode consistir em vários clusters de HA independentes. Isso permite que os sistemas de arquivos continuem dimensionando muito além dos limites recomendados/físicos dos componentes subjacentes do cluster de HA.

Passos

Ao adicionar um componente básico ao cluster, você precisará criar os `host_vars` arquivos para cada um dos novos nós de arquivo e nós de bloco (arrays e-Series). Os nomes desses hosts precisam ser adicionados ao inventário, juntamente com os novos recursos que devem ser criados. Os arquivos correspondentes `group_vars` precisarão ser criados para cada novo recurso. Consulte "use arquiteturas personalizadas"a secção para obter detalhes.

Depois de criar os arquivos corretos, tudo o que é necessário é executar novamente a automação usando o comando:

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

Removendo um Building Block do cluster

Há uma série de considerações a ter em mente quando você precisa aposentar um bloco de construção, por exemplo:

- Quais serviços BeeGFS estão sendo executados nesse componente básico?
- Apenas os nós de arquivo estão se aposentando e os nós de bloco devem ser anexados a novos nós de arquivo?
- Se todo o componente básico estiver sendo aposentado, os dados devem ser movidos para um novo componente básico, dispersos em nós existentes no cluster ou movidos para um novo sistema de arquivos BeeGFS ou outro sistema de storage?
- Isso pode acontecer durante uma interrupção ou deve ser feito sem interrupções?
- O componente básico está ativamente em uso ou contém dados que não estão mais ativos?

Devido aos diversos pontos de partida possíveis e aos estados finais desejados, entre em Contato com o suporte da NetApp para que possamos identificar e ajudar a implementar a melhor estratégia com base em seu ambiente e requisitos.

Solucionar problemas

Solução de problemas de um cluster BeeGFS HA.

Visão geral

Esta seção descreve como investigar e solucionar problemas de várias falhas e outros cenários que podem surgir ao operar um cluster BeeGFS HA.

Guias de solução de problemas

Investigando failovers inesperados

Quando um nó é fechado inesperadamente e seus serviços são movidos para outro nó, a primeira etapa deve ser verificar se o cluster indica falhas de recursos na parte inferior `pcs status`do . Normalmente, nada estará presente se o esgrima for concluído com sucesso e os recursos forem reiniciados em outro nó.

Geralmente, o próximo passo será pesquisar através dos logs do systemd usando `journalctl` em qualquer um dos nós de arquivo restantes (os logs do pacemaker são sincronizados em todos os nós). Se você souber a hora em que ocorreu a falha, você pode iniciar a pesquisa imediatamente antes da falha ocorrer (geralmente, pelo menos dez minutos antes é recomendado):

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>"
```

As seções a seguir mostram o texto comum que você pode grep nos logs para restringir ainda mais a investigação.

Passos para investigar/resolver

Passo 1: Verifique se o monitor BeeGFS detetou uma falha:

Se o failover tiver sido acionado pelo monitor BeeGFS, você verá um erro (se não prosseguir para a próxima etapa).

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i unexpected
[...]
Jul 01 15:51:03 beegfs_01 pacemaker-schedulerd[9246]:  warning: Unexpected
result (error: BeeGFS service is not active!) was recorded for monitor of
meta_08-monitor on beegfs_02 at Jul  1 15:51:03 2022
```

Neste caso, o serviço BeeGFS meta_08 parou por algum motivo. Para continuar a solução de problemas, devemos inicializar o beegfs_02 e revisar os logs do serviço em `/var/log/beegfs-meta-meta_08_tgt_0801.log`. Por exemplo, o serviço BeeGFS pode ter encontrado um erro de aplicação devido a um problema interno ou problema com o nó.

Diferentemente dos logs do pacemaker, os logs dos serviços BeeGFS não são distribuídos para todos os nós do cluster. Para investigar esses tipos de falhas, os logs do nó original onde a falha ocorreu são necessários.

Possíveis problemas que podem ser relatados pelo monitor incluem:

- O(s) alvo(s) não estão acessíveis!
  - Descrição: Indica que os volumes de bloco não estavam acessíveis.
  - Resolução de problemas:
    - Se o serviço também não conseguir iniciar no nó de arquivo alternativo, confirme se o nó de bloco está em bom estado.
    - Verifique se há problemas físicos que impeçam o acesso aos nós de bloco a partir deste nó de arquivo, por exemplo, adaptadores InfiniBand com defeito ou cabos.
- A rede não está acessível!
  - Descrição: Nenhum dos adaptadores usados pelos clientes para se conetar a este serviço BeeGFS estava online.
  - Resolução de problemas:
    - Se vários/todos os nós de arquivo foram afetados, verifique se houve uma falha na rede usada para conetar os clientes BeeGFS e o sistema de arquivos.
    - Verifique se há problemas físicos que impeçam o acesso aos clientes a partir deste nó de arquivo, por exemplo, adaptadores InfiniBand ou cabos defeituosos.
- O serviço BeeGFS não está ativo!
  - Descrição: Um serviço BeeGFS parou inesperadamente.
  - Resolução de problemas:
    - No nó de arquivo que relatou o erro, verifique os logs para o serviço BeeGFS impactado para ver se ele relatou uma falha. Se isso aconteceu, abra um caso com o suporte do NetApp para que a falha possa ser investigada.

- Se não houver erros relatados no log BeeGFS, verifique os logs do diário para ver se systemd registrou um motivo pelo qual o serviço foi interrompido. Em alguns cenários, o serviço BeeGFS pode não ter tido a chance de Registrar quaisquer mensagens antes do processo ser encerrado (por exemplo, se alguém executou `kill -9 <PID>`).

Etapa 2: Verifique se o nó deixou o cluster inesperadamente

Caso o nó tenha sofrido alguma falha catastrófica de hardware (por exemplo, a placa do sistema morreu) ou tenha ocorrido um problema de pânico do kernel ou de software semelhante, o monitor BeeGFS não reportará um erro. Em vez disso, procure o nome do host e você deve ver mensagens do Pacemaker indicando que o nó foi perdido inesperadamente:

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i <HOSTNAME>
[...]
Jul 01 16:18:01 beegfs_01 pacemaker-attrd[9245]:  notice: Node beegfs_02
state is now lost
Jul 01 16:18:01 beegfs_01 pacemaker-controld[9247]:  warning:
Stonith/shutdown of node beegfs_02 was not expected
```

Passo 3: Verifique se o pacemaker foi capaz de cercar o nó

Em todos os cenários, você deve ver o pacemaker tentar cercar o nó para verificar se ele está realmente offline (as mensagens exatas podem variar por causa da esgrima):

```
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Cluster
node beegfs_02 will be fenced: peer is no longer part of the cluster
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Node
beegfs_02 is unclean
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Scheduling
Node beegfs_02 for STONITH
```

Se a ação de esgrima for concluída com sucesso, você verá mensagens como:

```
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
[2214070] (call 27 from pacemaker-controld.9247) for host 'beegfs_02' with
device 'fence_redfish_2' returned: 0 (OK)
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
targeting beegfs_02 on beegfs_01 for pacemaker-
controld.9247@beegfs_01.786df3a1: OK
Jul 01 16:18:14 beegfs_01 pacemaker-controld[9247]:  notice: Peer
beegfs_02 was terminated (off) by beegfs_01 on behalf of pacemaker-
controld.9247: OK
```

Se a ação de esgrima falhou por algum motivo, os serviços BeeGFS não poderão reiniciar em outro nó para evitar o risco de corrupção de dados. Isso seria um problema para investigar separadamente, se, por exemplo, o dispositivo de vedação (PDU ou BMC) estivesse inacessível ou mal configurado.

Ações recurso Falha Endereço (encontradas na parte inferior do status PCs)

Se um recurso necessário para executar um serviço BeeGFS falhar, um failover será acionado pelo monitor BeeGFS. Se isso ocorrer, provavelmente não haverá "ações de recurso com falha" listadas na parte inferior do `pcs status` e você deve consultar as etapas sobre como "failback após um failover não planejado".

Caso contrário, geralmente deve haver apenas dois cenários onde você verá "ações de recurso falhadas".

Passos para investigar/resolver

Cenário 1: Um problema temporário ou permanente foi detetado com um agente de esgrima e foi reiniciado ou movido para outro nó.

Alguns agentes de vedação são mais confiáveis do que outros, e cada um implementará seu próprio método de monitoramento para garantir que o dispositivo de vedação esteja pronto. Em particular, o agente de esgrima do redfish foi visto para relatar ações de recursos falhadas como as seguintes, mesmo que ele ainda mostre iniciado:

```
  * fence_redfish_2_monitor_60000 on beegfs_01 'not running' (7):
call=2248, status='complete', exitreason='', last-rc-change='2022-07-26
08:12:59 -05:00', queued=0ms, exec=0ms
```

Não é esperado que um agente de esgrima que relata ações de recursos com falha em um nó específico acione um failover dos serviços BeeGFS executados nesse nó. Ele deve simplesmente ser reiniciado automaticamente no mesmo nó ou em um nó diferente.

Passos para resolver:

1. Se o agente de esgrima se recusar a executar consistentemente em todos ou em um subconjunto de nós, verifique se esses nós são capazes de se conetar ao agente de esgrima e verifique se o agente de esgrima está configurado corretamente no inventário do Ansible.
   a. Por exemplo, se um agente de esgrima de peixe vermelho (BMC) estiver sendo executado no mesmo nó que é responsável por esgrima, e o gerenciamento de SO e IPs BMC estiverem na mesma interface física, algumas configurações de switch de rede não permitirão a comunicação entre as duas interfaces (para evitar loops de rede). Por padrão, o cluster de HA tentará evitar colocar agentes de vedação no nó que são responsáveis por cercas, mas isso pode acontecer em alguns cenários/configurações.
2. Uma vez que todos os problemas são resolvidos (ou se o problema parecia efêmero), execute `pcs resource cleanup` para redefinir as ações de recursos com falha.

Cenário 2: O monitor BeeGFS detetou um problema e acionou um failover, mas por algum motivo os recursos não puderam ser iniciados em um nó secundário.

Desde que o esgrima esteja habilitado e o recurso não tenha sido bloqueado de parar no nó original (consulte a seção de solução de problemas para "standby (on-fail)"), as razões mais prováveis incluem problemas para iniciar o recurso em um nó secundário porque:

- O nó secundário já estava offline.
- Um problema de configuração físico ou lógico impediu que o secundário acessasse os volumes de bloco usados como destinos BeeGFS.

Passos para resolver:

1. Para cada entrada nas ações de recursos com falha:
   a. Confirme se a ação de recurso falhou foi uma operação de início.
   b. Com base no recurso indicado e no nó especificado nas ações de recurso com falha:
      i. Procure e corrija quaisquer problemas externos que impeçam o nó de iniciar o recurso especificado. Por exemplo, se o endereço IP BeeGFS (IP flutuante) não foi iniciado, verifique se pelo menos uma das interfaces necessárias está conetada/on-line e cabeada ao switch de rede direito. Se um destino BeeGFS (dispositivo de bloco/volume e-Series) falhar, verifique se as conexões físicas com os nós de bloco de back-end estão conetadas conforme o esperado e verifique se os nós de bloco estão íntegros.
   c. Se não houver problemas externos óbvios e você desejar uma causa raiz para esse incidente, é sugerido que você abra um caso com suporte do NetApp para investigar antes de prosseguir, pois as etapas a seguir podem tornar a análise de causa raiz (RCA) desafiadora/impossível.
2. Depois de resolver quaisquer problemas externos:
   a. Comente todos os nós não funcionais do arquivo Ansible inventory.yml e execute novamente o manual completo do Ansible para garantir que toda a configuração lógica esteja configurada corretamente nos nós secundários.
      i. Observação: Não se esqueça de descomentar esses nós e executar novamente o manual de estratégia quando os nós estiverem saudáveis e você estiver pronto para o failback.
   b. Como alternativa, você pode tentar recuperar manualmente o cluster:
      i. Coloque todos os nós offline de volta online usando: `pcs cluster start <HOSTNAME>`
      ii. Limpar todas as ações de recursos com falha usando: `pcs resource cleanup`
      iii. Execute o status dos PCs e verifique se todos os serviços começam conforme esperado.
      iv. Se necessário, execute `pcs resource relocate run` para mover os recursos de volta para o nó preferido (se ele estiver disponível).

Questões comuns

Os serviços BeeGFS não fazem failover ou failback quando solicitados

- Problema provável:* o `pcs resource relocate` comando run foi executado, mas nunca terminou com sucesso.

Como verificar: Executar `pcs constraint --full` e verificar quaisquer restrições de localização com um ID de `pcs-relocate-<RESOURCE>`.

How to resolve: execute `pcs resource relocate clear` e execute novamente `pcs constraint --full` para verificar se as restrições extras são removidas.

Um nó no estado dos PCes mostra "standby (on-fail)" quando a vedação está desativada

Problema provável: o pacemaker não conseguiu confirmar com êxito todos os recursos foram parados no nó que falhou.

Como resolver:

1. Execute `pcs status` e verifique se há recursos que não são "iniciados" ou mostram erros na parte

inferior da saída e resolva quaisquer problemas.

2. Para colocar o nó novamente online, execute `pcs resource cleanup --node=<HOSTNAME>`.

Após um failover inesperado, os recursos mostram "Started (on-fail)" no status PCs quando o esgrima está ativado

Problema provável: ocorreu Um problema que desencadeou um failover, mas o pacemaker não conseguiu verificar se o nó estava vedado. Isso pode acontecer porque o esgrima foi mal configurado ou houve um problema com o agente de esgrima (exemplo: O PDU foi desconetado da rede).

Como resolver:

1. Verifique se o nó está realmente desligado.

> Se o nó especificado não estiver realmente desativado, mas executando serviços ou recursos de cluster, ocorrerá corrupção de dados/falha de cluster.

2. Confirme manualmente a vedação com: `pcs stonith confirm <NODE>`

Neste ponto, os serviços devem terminar de falhar e ser reiniciados em outro nó saudável.

Tarefas comuns de resolução de problemas

Reinicie os serviços BeeGFS individuais

Normalmente, se um serviço BeeGFS precisar ser reiniciado (por exemplo, para facilitar uma alteração de configuração), isso deve ser feito atualizando o inventário do Ansible e executando novamente o manual de estratégia. Em alguns cenários, pode ser desejável reiniciar serviços individuais para facilitar a solução de problemas mais rápida, por exemplo, para alterar o nível de log sem precisar esperar que todo o manual de estratégia seja executado.

> A menos que quaisquer alterações manuais também sejam adicionadas ao inventário do Ansible, elas serão revertidas na próxima vez que o manual de estratégia do Ansible for executado.

Opção 1: Reinício controlado pelo sistema

Se houver um risco de o serviço BeeGFS não reiniciar corretamente com a nova configuração, primeiro coloque o cluster no modo de manutenção para impedir que o monitor BeeGFS detete que o serviço seja interrompido e acione um failover indesejado:

```
pcs property set maintenance-mode=true
```

Se necessário, faça alterações na configuração dos serviços em `/mnt/<SERVICE_ID>/_config/beegfs-.conf` (exemplo: `/mnt/meta_01_tgt_0101/metadata_config/beegfs-meta.conf`), em seguida, use systemd para reiniciá-lo:

```
systemctl restart beegfs-*@<SERVICE_ID>.service
```

Exemplo: `systemctl restart beegfs-meta@meta_01_tgt_0101.service`

Opção 2: Reinício controlado pelo pacemaker

Se você não estiver preocupado com a nova configuração pode fazer com que o serviço pare inesperadamente (por exemplo, simplesmente mudando o nível de log), ou você está em uma janela de manutenção e não está preocupado com o tempo de inatividade, você pode simplesmente reiniciar o monitor BeeGFS para o serviço que deseja reiniciar:

```
pcs resource restart <SERVICE>-monitor
```

Por exemplo, para reiniciar o serviço de gerenciamento BeeGFS: `pcs resource restart mgmt-monitor`

Avisos legais

Avisos legais fornecem acesso a declarações de direitos autorais, marcas registradas, patentes e muito mais.

Direitos de autor

"https://www.netapp.com/company/legal/copyright/"

Marcas comerciais

NetApp, o logotipo DA NetApp e as marcas listadas na página de marcas comerciais da NetApp são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respetivos proprietários.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

Uma lista atual de patentes de propriedade da NetApp pode ser encontrada em:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidade

"https://www.netapp.com/company/legal/privacy-policy/"

Código aberto

Os arquivos de aviso fornecem informações sobre direitos autorais de terceiros e licenças usadas no software NetApp.

"Aviso para o e-Series/EF-Series SANtricity os"

Informações sobre direitos autorais

Informações sobre marcas comerciais