Use arquiteturas verificadas

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Índice

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 .

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 .

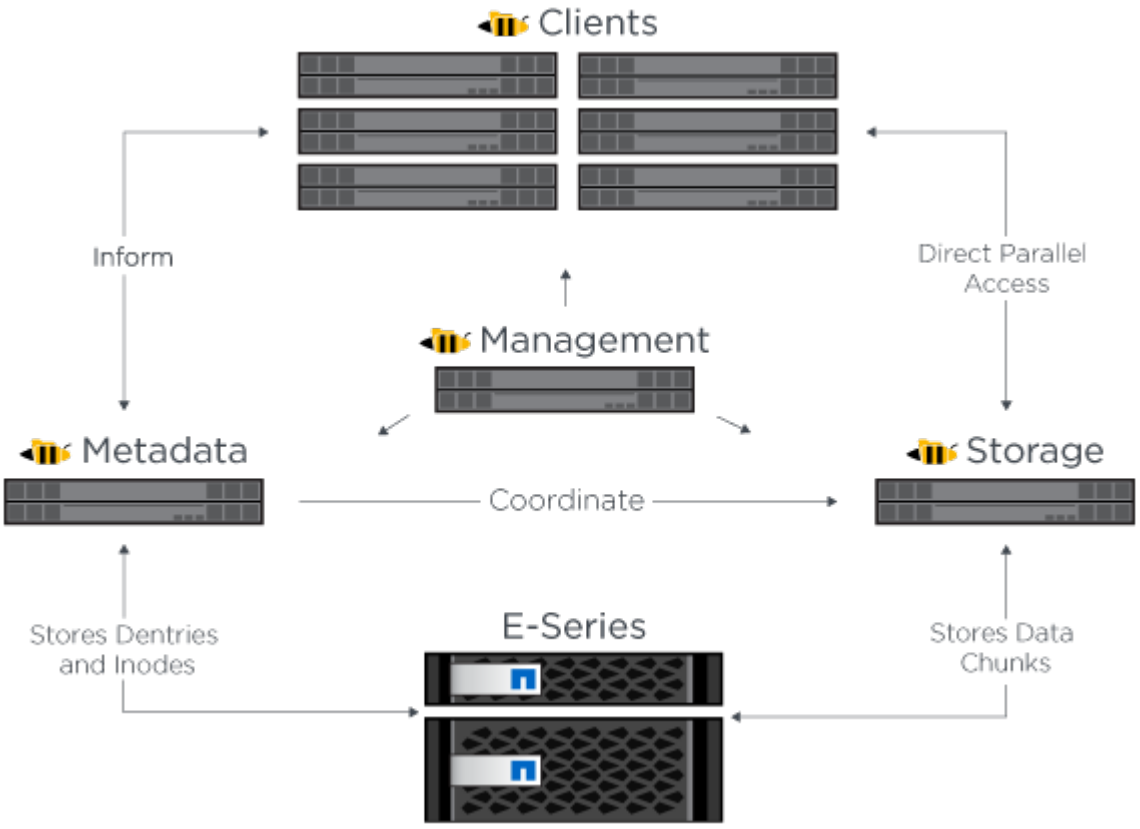
Serviços de sistema de arquivos

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.

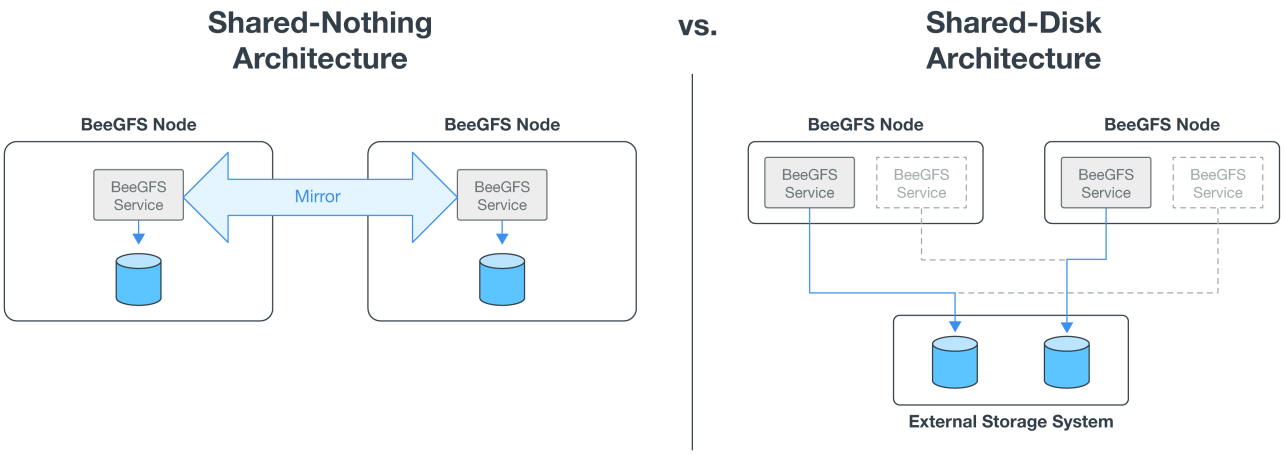
Os quatro principais serviços são executados em uma ampla variedade de distribuições Linux suportadas e se comunicam por qualquer rede compatível com TCP/IP ou RDMA, incluindo InfiniBand (IB), Omni-Path (OPA) e RDMA sobre Ethernet convergente (RoCE). Os serviços de servidor BeeGFS (gerenciamento, storage 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 .

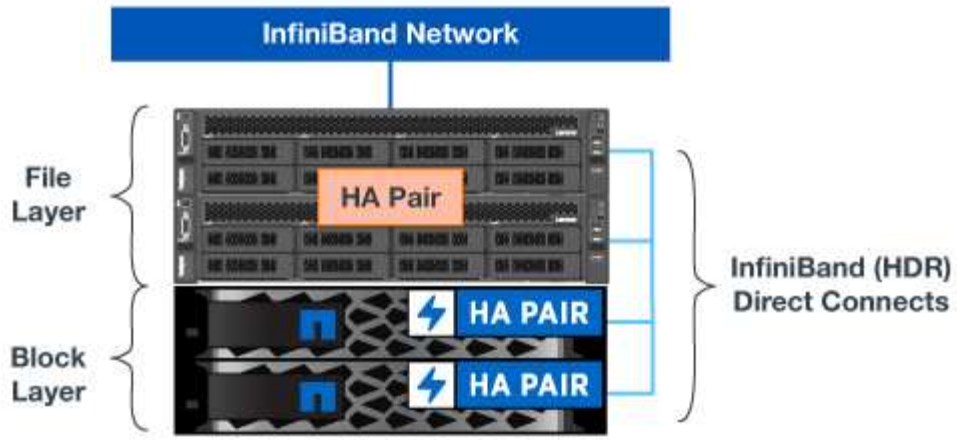
Nós verificados

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

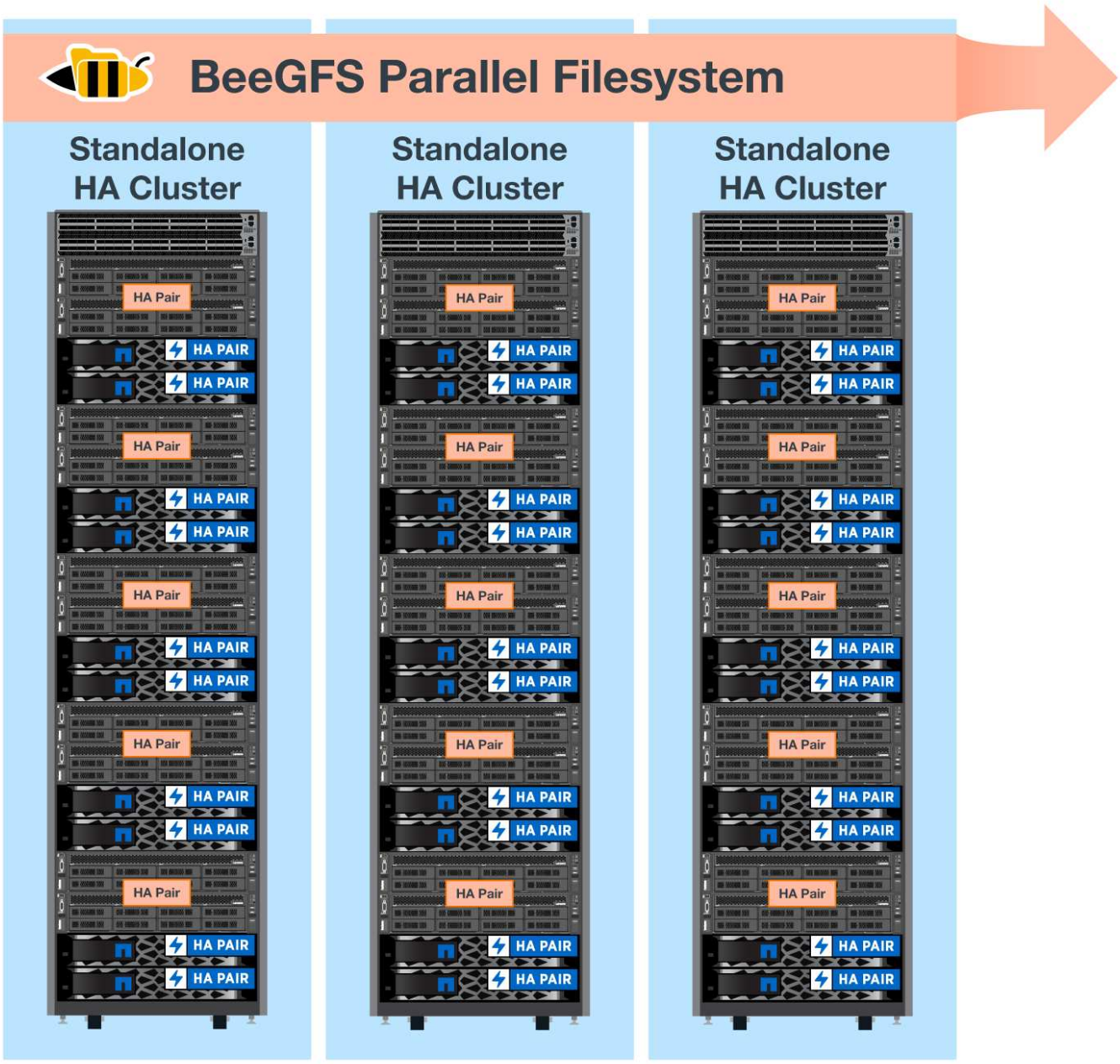
Os componentes básicos da solução (mostrados na figura a seguir) usam os servidores de nós de arquivo verificados para a camada de arquivo BeeGFS e dois sistemas de storage EF600 como a camada de bloco.



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
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.
8	Adaptadores de placa de host InfiniBand (para nós de arquivo).	Os adaptadores de placa host podem variar dependendo do modelo de servidor do 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
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	Tipo de cabo	Quantidade	Número de peça
NVIDIA QM9700	Fibra ativa (incluindo transcetores)	2	MMA4Z00-NS (multimodo, IB/ETH, 800GB GB/s 2x400Gb/s OSFP de porta dupla)
		4	MFP7E20-Nxxx (cabo de fibra divisor de 2 canais multimodo, 4 canais para dois)
		8	MMA1Z00-NS400 (multimodo, IB/ETH, QSFP-112 de porta única de 400GB GB/s)
	Cobre passivo	2	MCP7Y40-N002 (cabo divisor de cobre passivo NVIDIA, InfiniBand de 800GB GB/s para 4x 200GB GB/s, OSFP para 4x QSFP112 GB)
NVIDIA MQM8700	Fibra ativa	8	MFS1S00-H003E (cabo de fibra ativa NVIDIA, InfiniBand de 200GB GB/s, QSFP56 GB)
	Cobre passivo	8	MCP1650-H002E26 (cabo de cobre passivo NVIDIA, InfiniBand de 200GB GB/s, QSFP56 GB)

Requisitos de software e firmware

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

Requisitos de nó de arquivo

Software	Versão
Red Hat Enterprise Linux (RHEL)	Servidor RHEL 9.4 físico com alta disponibilidade (2 soquetes). **Observação:** Os nós de arquivo exigem uma assinatura válida do Red Hat Enterprise Linux Server e o complemento de alta disponibilidade do Red Hat Enterprise Linux.
Kernel do Linux	5.14.0-427.42.1.el9_4.x86_64
Firmware HCA	**Firmware ConnectX-7 HCA** FW: 28.45.1200 + PXE: 3.7.0500 + UEFI: 14.38.0016 • Firmware HCA * ConnectX-6 FW: 20.43.2566 e PXE: 3.7.0500 e UEFI: 14.37.0013

Requisitos de nó de bloco de EF600 U.

Software	Versão
Sistema operacional SANtricity	11.90R3
NVSRAM	N6000-890834-D02.dlp
Firmware da unidade	Mais recente disponível para os modelos de acionamento em uso. Consulte "Site de firmware de disco e-Series".

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	7.4.6
Corosync	3,1.8-1
Pacemaker	2,1.7-5,2
PCS	0,11.7-2
Agentes de vedação (peixe-vermelho/apc)	4,10.0-62
Drivers InfiniBand / RDMA	MLNX_OFED_LINUX-23,10-3,2.2,1-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	10.x
Ansible-core	> 2.13.0
Python	3,10
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

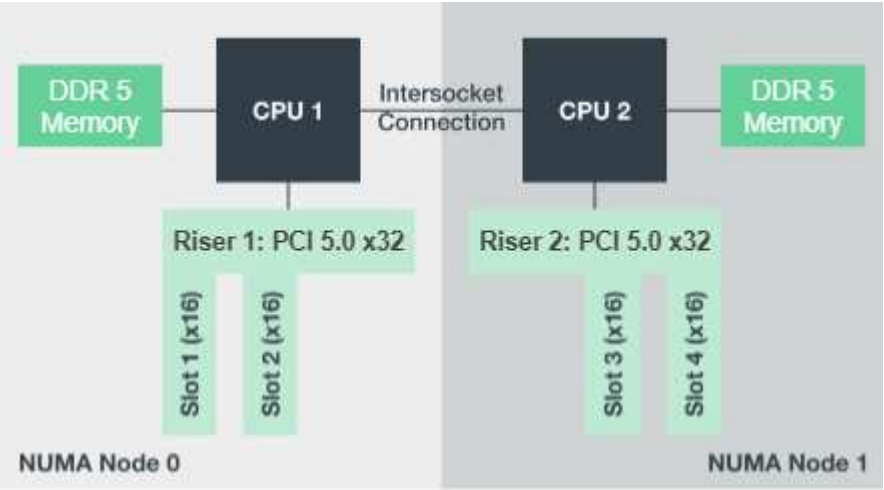
A configuração de hardware do BeeGFS no NetApp inclui nós de arquivo e cabeamento de rede.

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.



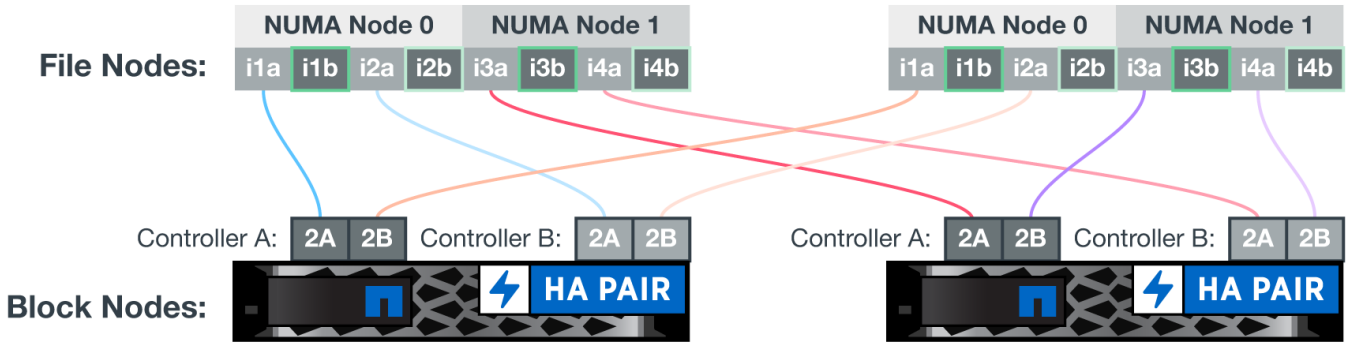
Os processos BeeGFS são fixados a uma zona NUMA específica para garantir que as interfaces usadas estejam na mesma zona. Esta configuração evita a necessidade de acesso remoto através da ligação entre sockets. A conexão entre soquetes às vezes é conhecida como QPI ou link GMI2; mesmo em arquiteturas 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:

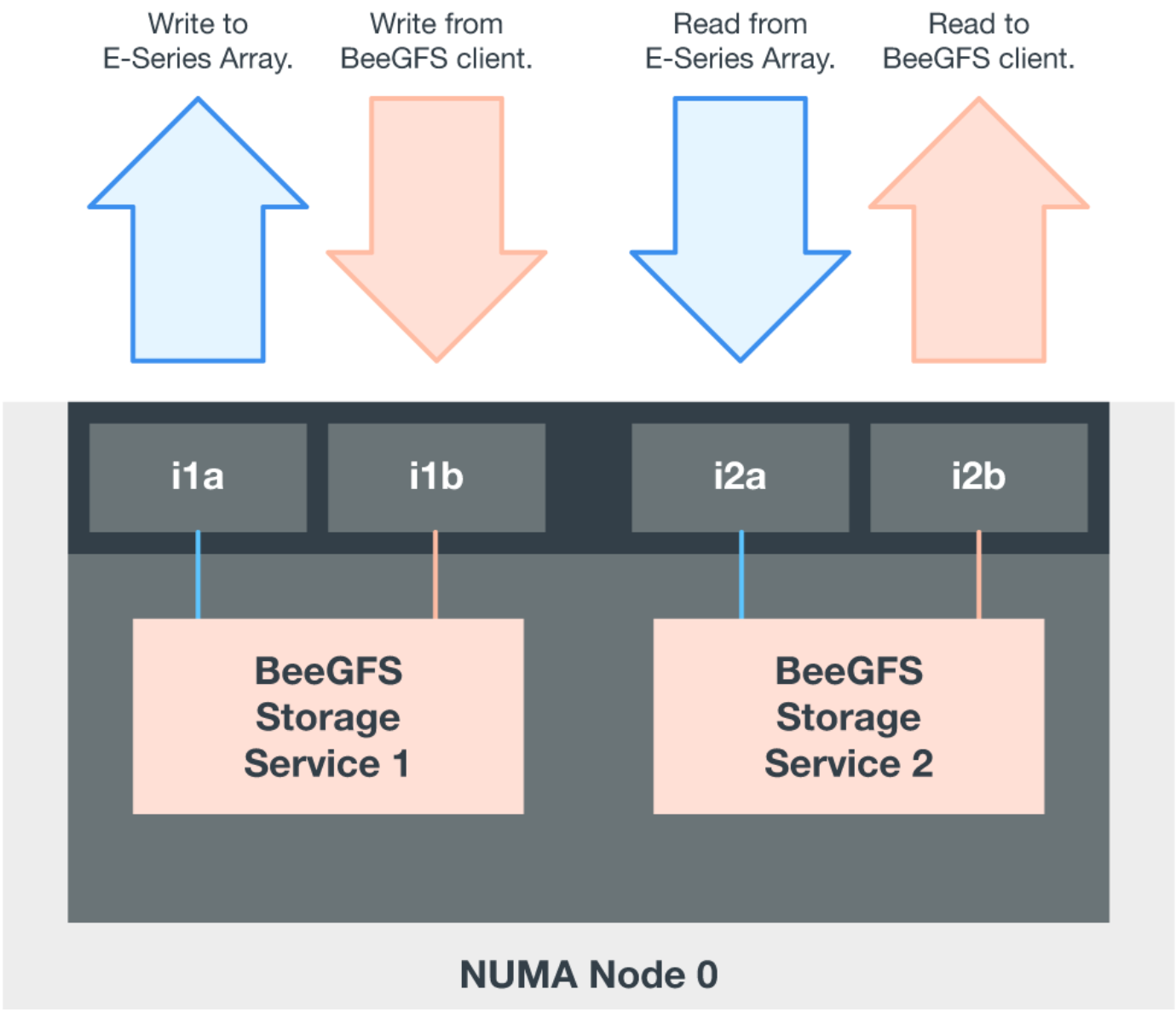
- Execute na mesma zona NUMA, independentemente do nó de arquivo que está executando o serviço BeeGFS.
- Ter caminhos secundários ideais para a rede de storage de front-end e para os nós de bloco de back-end, independentemente de onde ocorra uma falha.
- Minimize os efeitos de desempenho se um nó de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

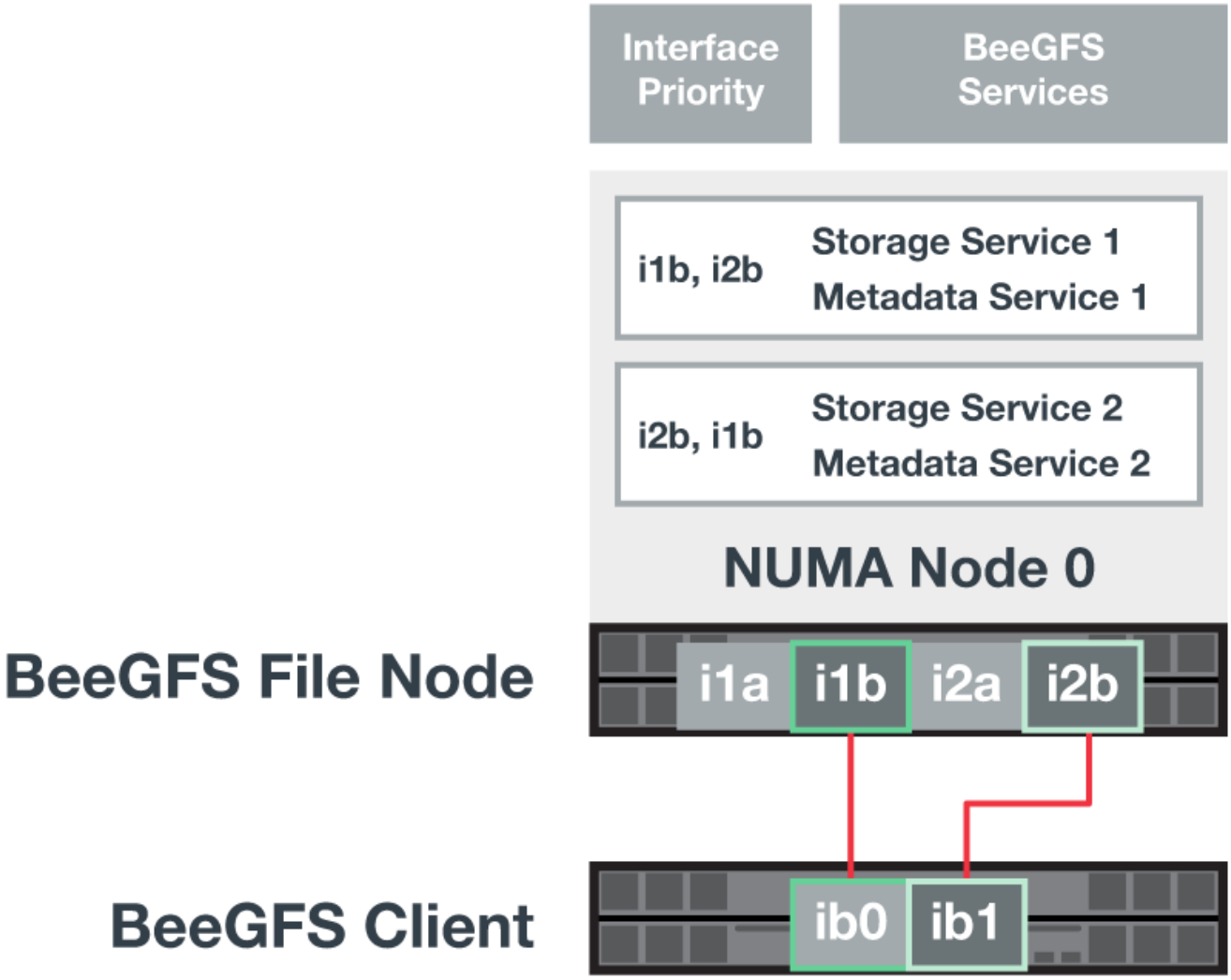
  A implantação baseada em Ansible lida com o aperto do comportamento do caminho reverso (RP) e do protocolo de resolução de endereço (ARP), além de garantir quando os IPs flutuantes são iniciados e parados; as rotas e regras IP correspondentes são criadas dinamicamente para permitir que a configuração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

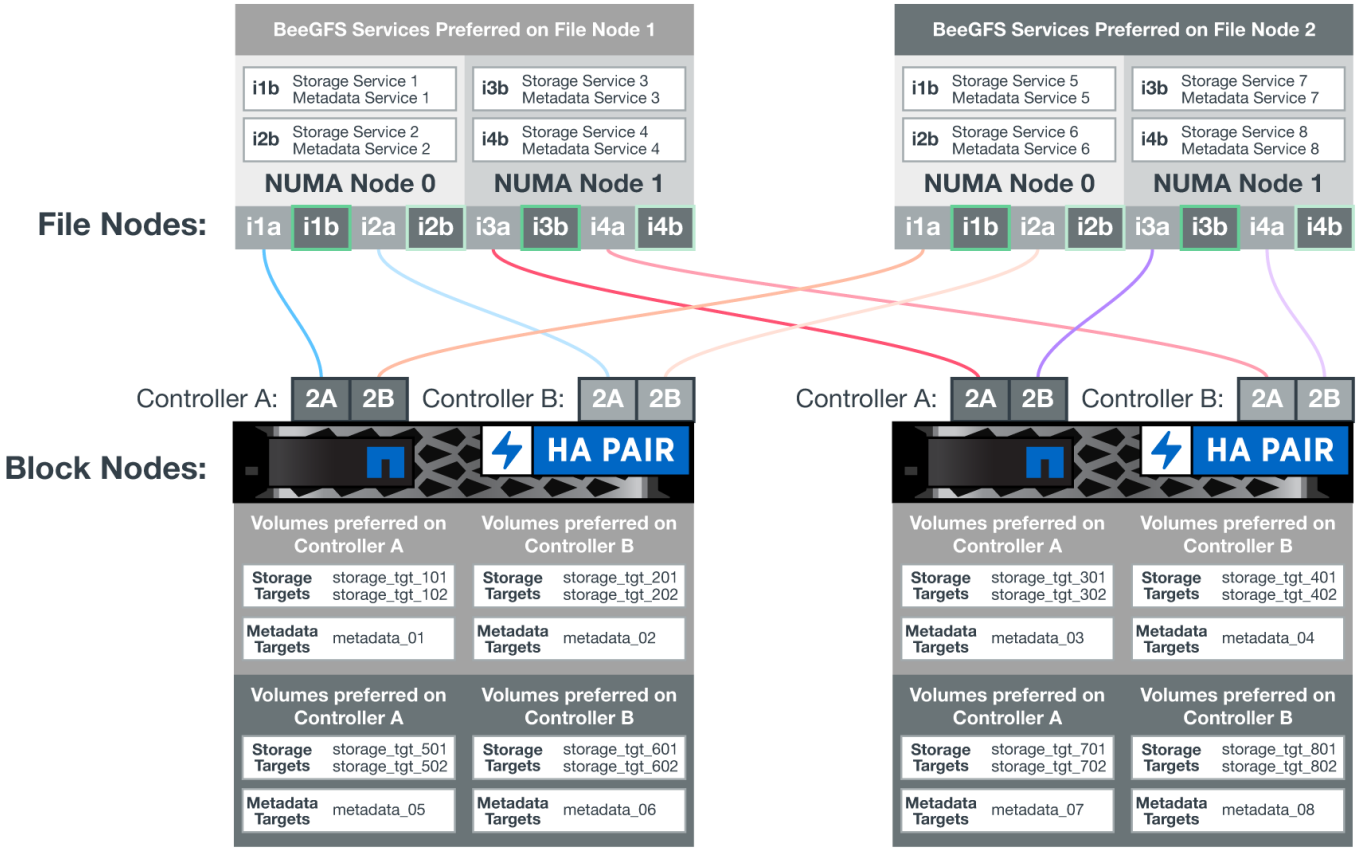
Configuração de nó de bloco de EF600 U.

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.

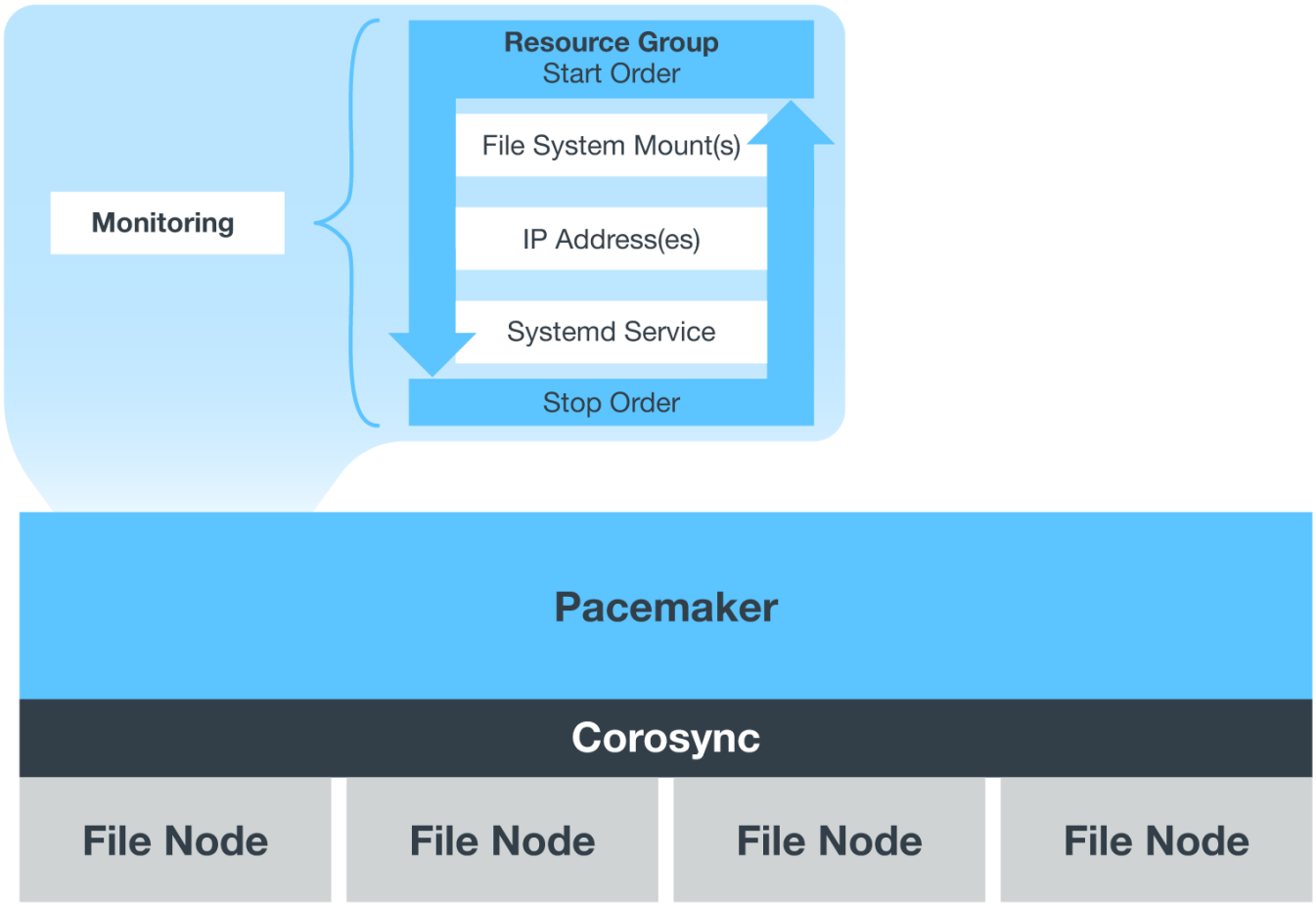
Se houver uma falha no cluster, o fabricante de Paz reagirá e reiniciará os recursos do BeeGFS em outro nó. Em alguns cenários, o pacemaker pode não ser capaz de se comunicar com o nó defeituoso original para confirmar que os recursos estão parados. Para verificar se o nó está inativo antes de reiniciar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

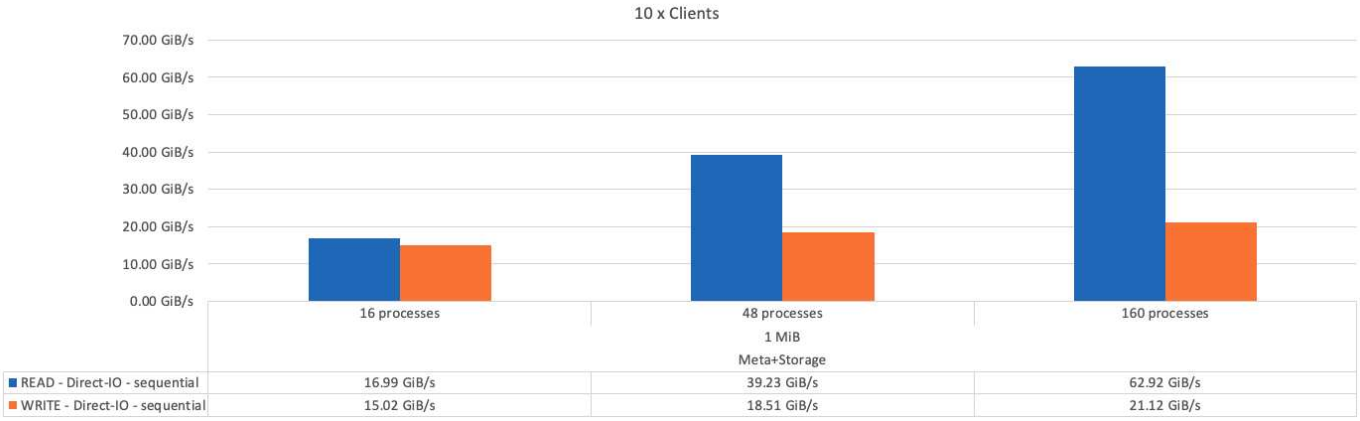
Uma base do BeeGFS (gerenciamento, metadados e storage) componente básico

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.

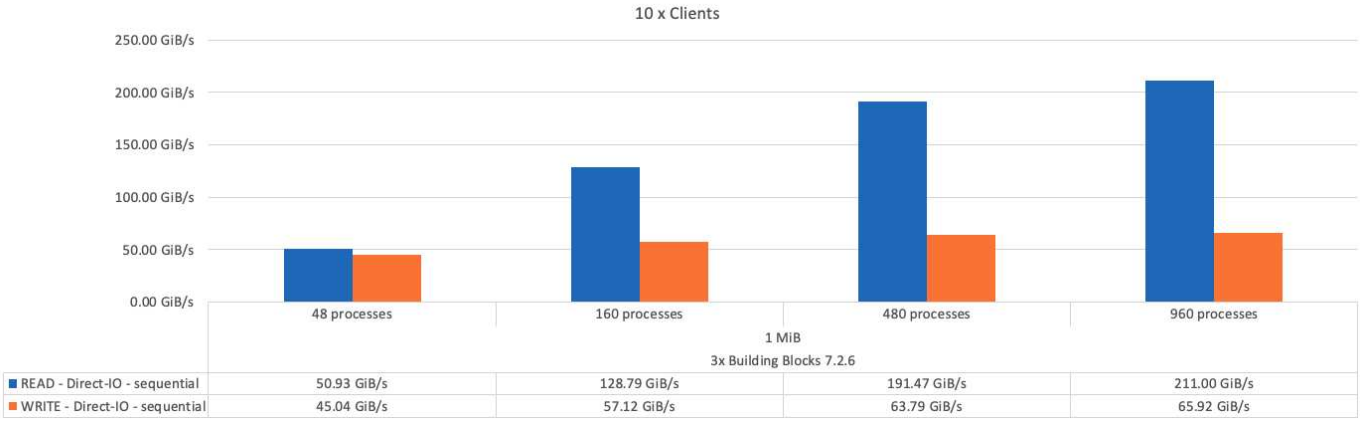


Componente básico somente de storage do BeeGFS

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.

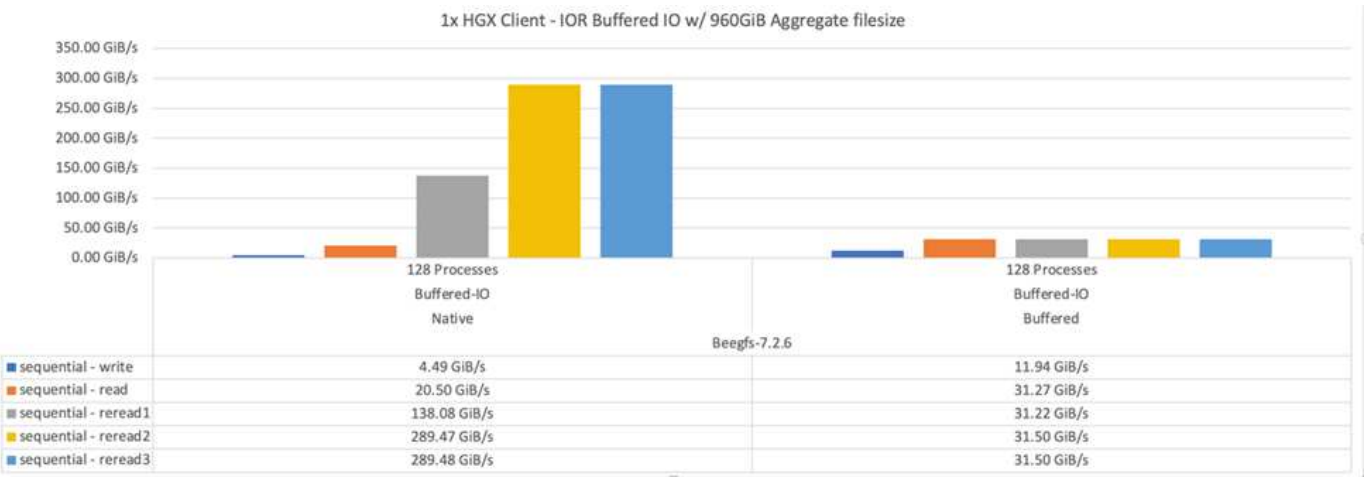
Testes de largura de banda IOR: Cliente único

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.

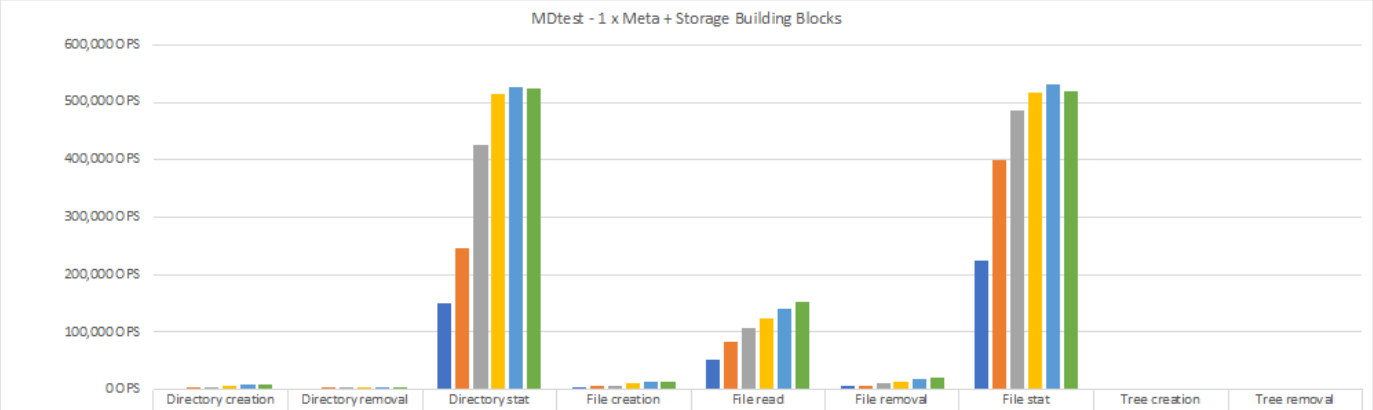
Os seguintes parâmetros foram utilizados para executar o teste de benchmark com o número total de processos dimensionados de 10 a 320 na etapa de 2x e com um tamanho de arquivo de 4K.

```
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

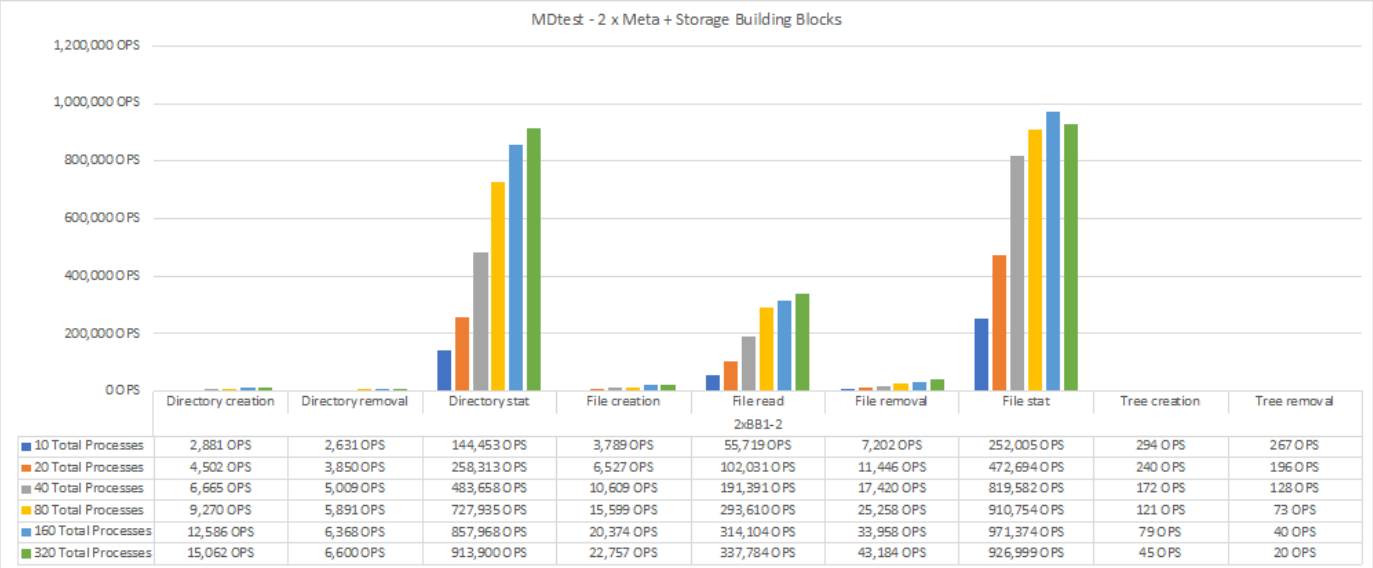
O diagrama a seguir mostra os resultados do MDTest com os componentes básicos de storage e metadados do BeeGFS.



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 Lenovo SR665 V3 validados, consulte "Ajuste as configurações do sistema do nó de arquivo para obter desempenho" .

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

O BeeGFS no NetApp pode ser implantado para nós de arquivo e bloco validados usando o Ansible com o design de componentes básicos do BeeGFS da NetApp.

Coleções e funções do Ansible

A solução BeeGFS no NetApp é implantada com o Ansible, um mecanismo de automação DE TI popular que automatiza as implantações de aplicações. O Ansible usa uma série de arquivos coletivamente conhecidos como 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 disponíveis no Ansible Galaxy ("Coleção BeeGFS da NetApp e-Series"consulte). As coleções incluem módulos que executam funções ou tarefas específicas (como criar um volume e-Series) e 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 manutenção e a expansão do cluster e do sistema de arquivos BeeGFS.

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 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 iniciar uma implantação, familiarize-se com a forma como o Ansible é configurado e usado para implantar a solução BeeGFS no NetApp.

O inventário do Ansible é uma estrutura de diretório que lista os nós de arquivo e bloco para o sistema de

arquivos BeeGFS a ser implantado. Ele inclui hosts, grupos e variáveis descrevendo o sistema de arquivos BeeGFS desejado. O inventário do Ansible precisa ser armazenado no nó de controle do Ansible, que é qualquer máquina com acesso aos nós de bloco e arquivo usados para executar o manual de estratégia do Ansible. Os inventários de amostras podem ser transferidos a partir do "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 Ansible do NetApp e-Series (disponível na "NetApp e-Series BeeGFS GitHub") que implantam 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

Defina um cluster BeeGFS HA com a estrutura de inventário do Ansible.

Qualquer pessoa com experiência anterior do Ansible deve estar ciente de que a função de HA do BeeGFS implementa um método personalizado para descobrir quais variáveis (ou fatos) se aplicam a cada host. Esse design simplifica a estruturação do inventário do Ansible para descrever recursos que podem ser executados em vários servidores.

Um inventário do Ansible normalmente consiste nos arquivos no `host_vars` e `group_vars`, juntamente com 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.

> 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).

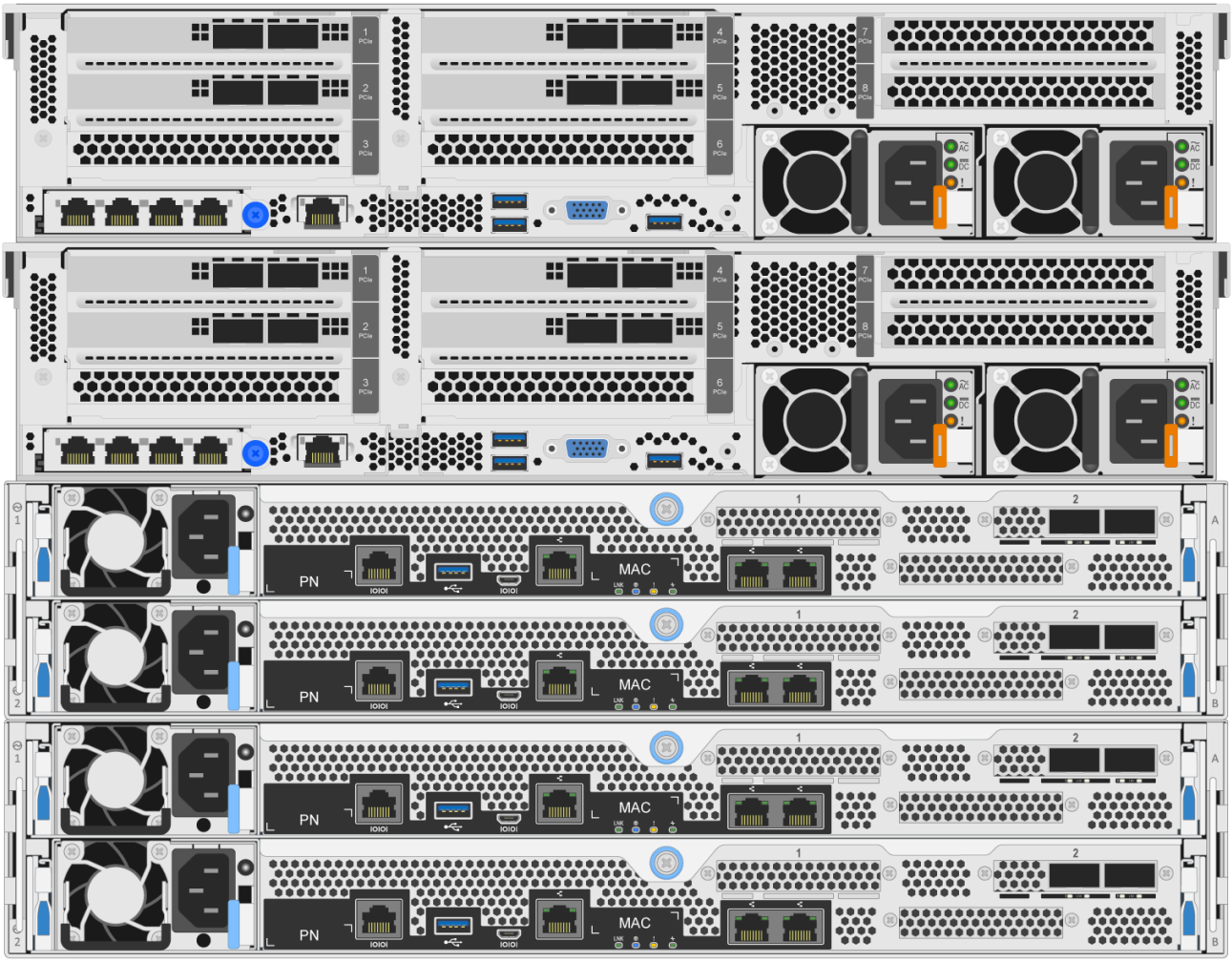
> É necessário um mínimo de dois componentes básicos para estabelecer quorum no cluster de failover. Um cluster de dois nós tem limitações que podem impedir que ocorra um failover bem-sucedido. Você pode configurar um cluster de dois nós incorporando um terceiro dispositivo como um tiebreaker. No entanto,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

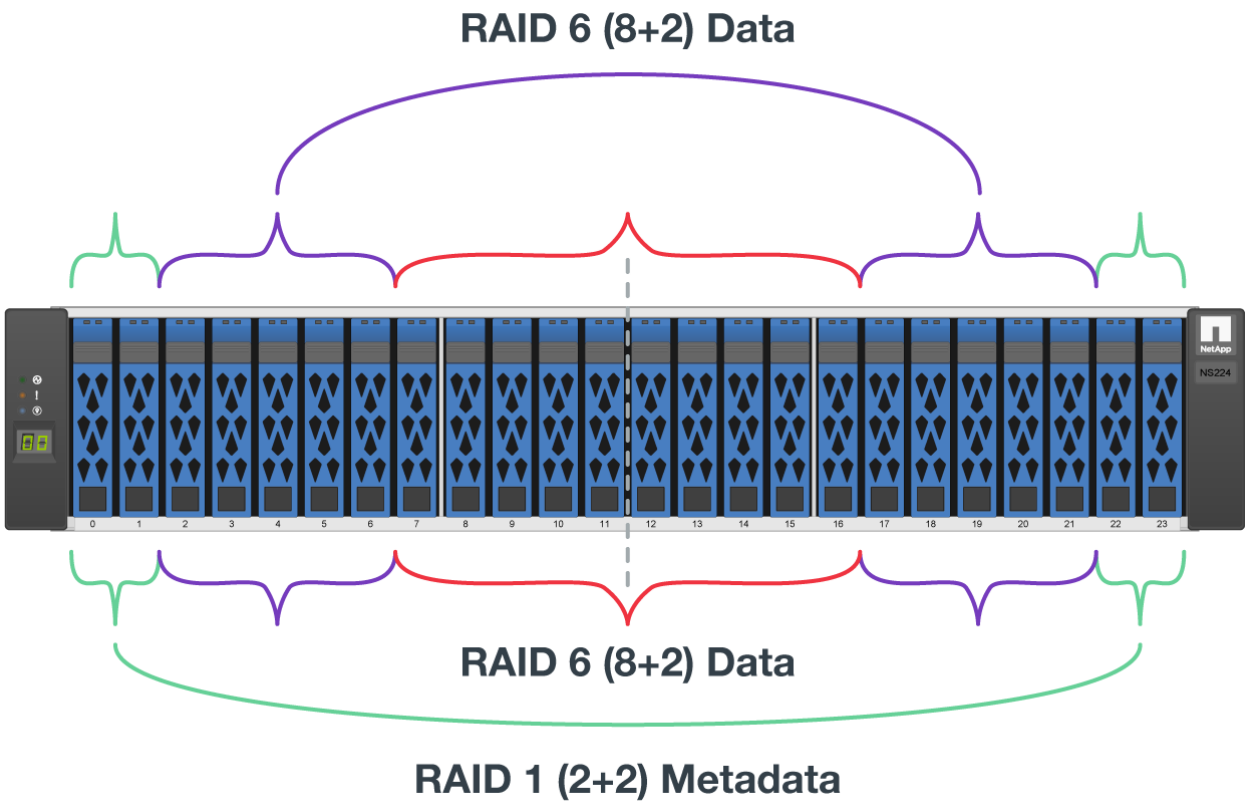
1. Configure cada nó de arquivo BeeGFS com quatro adaptadores de canal de host (HCAs) usando os modelos especificados no "Requisitos técnicos". Insira os HCAs nos slots PCIe do nó de arquivo de acordo com as especificações abaixo:
   - **Servidor Lenovo ThinkSystem SR665 V3:** Use slots PCIe 1, 2, 4 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).

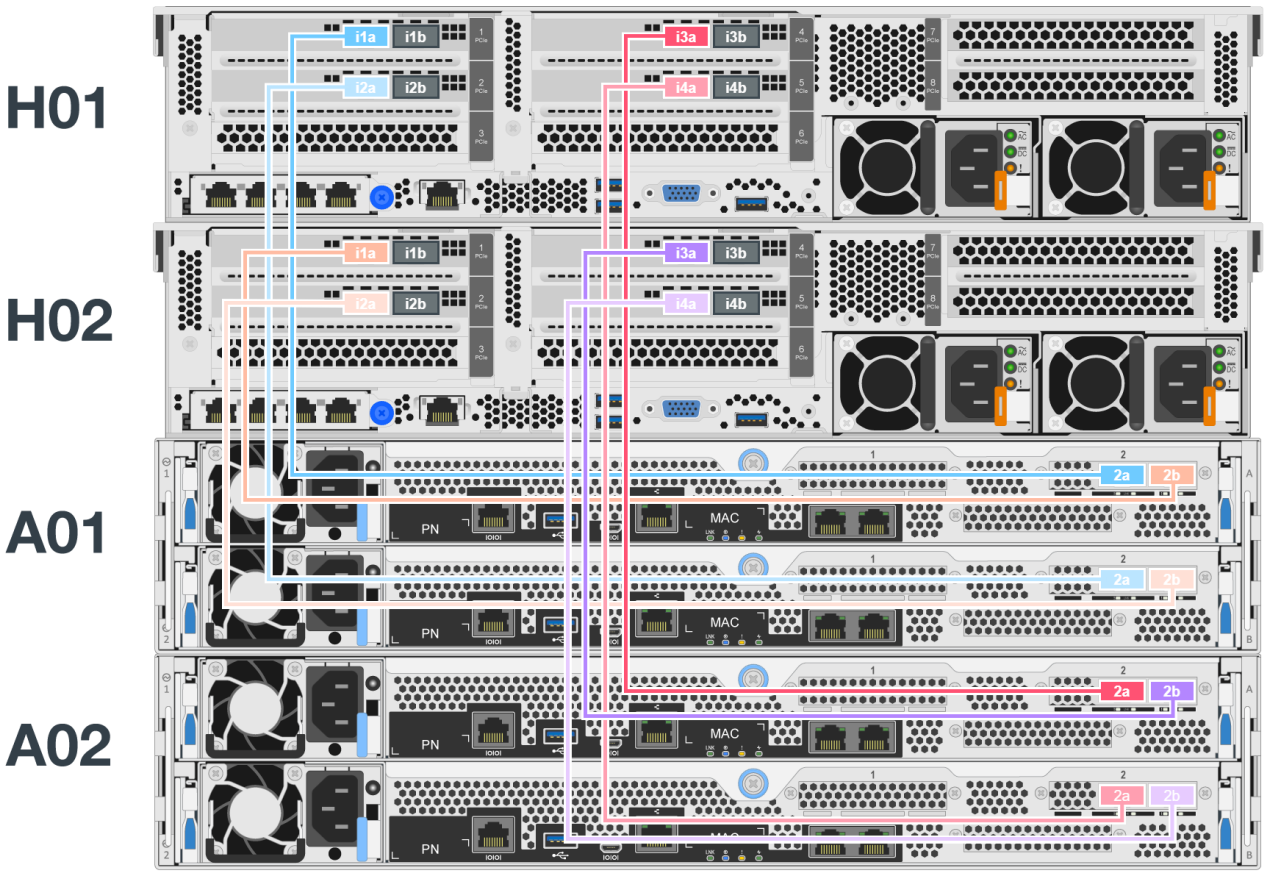


A configuração da fonte de alimentação para casos de uso de produção normalmente deve usar PSUs redundantes.

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.

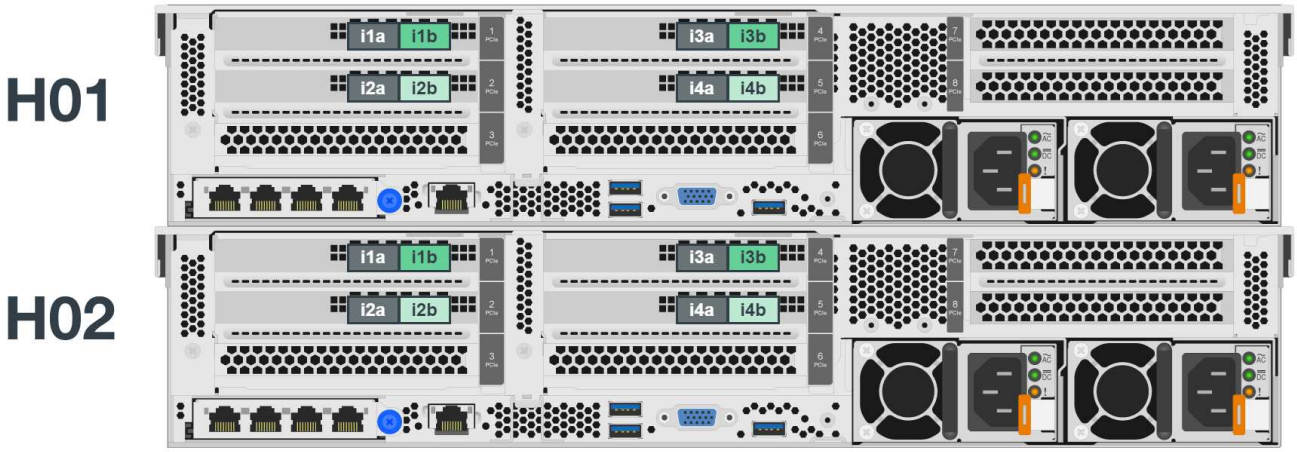


Os nós em vários componentes básicos nunca são conetados diretamente. Cada bloco de construção deve ser Tratado como uma unidade autônoma e toda a comunicação entre blocos de construção ocorre através de switches de rede.

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 Lenovo SR665 V3 validados, consulte "Ajuste as configurações do sistema para obter desempenho" .

3. Instale o Red Hat Enterprise Linux (RHEL) 9.4 e configure o nome do host e a porta de rede usados para gerenciar o sistema operacional, incluindo a conectividade 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 Red Hat Subscription Manager para registrar e assinar o sistema para permitir a instalação dos pacotes necessários dos repositórios oficiais do Red Hat e limitar as atualizações à versão suportada do Red Hat: `subscription-manager release --set=9.4` . 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 no "Requisitos de tecnologia" uso do "Atualize o firmware do adaptador do nó de arquivo" guia.

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 V3 contém vários parâmetros de ajuste que podem ser definidos por meio 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 designar uma máquina virtual ou física com acesso de rede a todos os nós de arquivo e bloco implantados na solução BeeGFS no NetApp.

Consulte "Requisitos técnicos"a para obter uma lista das versões de pacotes recomendadas. 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. A partir do nó de controle do Ansible, instale os seguintes pacotes Python e Python Virtual Environment.

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

2. Crie um ambiente virtual Python.

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

3. Ative o ambiente virtual.

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

4. Instale os pacotes Python necessários dentro do ambiente virtual ativado.

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

5. Instale a coleção BeeGFS usando o Ansible Galaxy.

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

6. Verifique se as versões instaladas do Ansible, Python e a coleção BeeGFS correspondem ao "Requisitos técnicos".

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

7. Configure SSH sem senha para 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.

```
ssh-keygen
```

   b. Configure o SSH sem senha para cada um dos nós de arquivo.

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

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

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
   ```

3. Crie um subdiretório para senhas de cluster e proteja o arquivo criptografando-o com o Ansible Vault ("Criptografia de conteúdo com o Ansible Vault"consulte):

   a. Crie o subdiretório . `group_vars/all`

   b. `group_vars/all`No diretório, crie um arquivo de senhas rotulado `passwords.yml`.

   c. Preencha o `passwords.yml file` com o seguinte, substituindo todos os parâmetros de nome de usuário e senha de acordo com sua configuração:

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

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

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

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

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

d. Execute `ansible-vault encrypt passwords.yml` e defina uma senha do Vault quando solicitado.

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: {{ ssh_ha_user }}
ansible_become_password: {{ ssh_ha_become_pass }}
eseries_ipoib_default_hook_templates:
  - 99-multihoming.j2   # This is required for single subnet
deployments, where static IPs containing multiple IB ports are in the
same IPoIB subnet. i.e: cluster IPs, multirail, single subnet, etc.
# If the following options are specified, then Ansible will
automatically reboot nodes when necessary for changes to take effect:
eseries_common_allow_host_reboot: true
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
eseries_ib_opensm_options:
  virt_enabled: "2"
  virt_max_ports_in_process: "0"
```

Se o `ansible_ssh_user` já estiver `root` , você poderá omitir opcionalmente o `ansible_become_password` e especificar a `--ask-become-pass` opção ao executar o manual de estratégia.

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: "{{ ha_cluster_username }}" # Parameter for
BeeGFS HA cluster username in the passwords file.
beegfs_ha_cluster_password: "{{ ha_cluster_password }}" # Parameter for
BeeGFS HA cluster username's password in the passwords file.
beegfs_ha_cluster_password_sha512_salt: "{{
ha_cluster_password_sha512_salt }}" # Parameter for BeeGFS HA cluster
username's password salt in the passwords file.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0          # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
      mydomain: "example.com"
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
beegfs_ha_alert_verbosity: 3
#  1) high-level node activity
#  3) high-level node activity + fencing action information + resources
(filter on X-monitor)
#  5) high-level node activity + fencing action information + resources
```

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: "{{ apc_username }}" # Parameter for APC PDU username in
the passwords file.
     passwd: "{{ apc_password }}" # Parameter for APC PDU password in
the passwords file.
     pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>"
# OPTION 2: To enable fencing using the Redfish APIs provided by the
Lenovo XCC (and other BMCs):
redfish: &redfish
  username: "{{ bmc_username }}" # Parameter for XCC/BMC username in
the passwords file.
  password: "{{ bmc_password }}" # Parameter for XCC/BMC password in
the passwords file.
    ssl_insecure: 1 # If a valid SSL certificate is not available
specify “1”.
beegfs_ha_fencing_agents:
  fence_redfish:
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
# For details on configuring other fencing agents see
https://access.redhat.com/documentation/en-
us/red_hat_enterprise_linux/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

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

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.

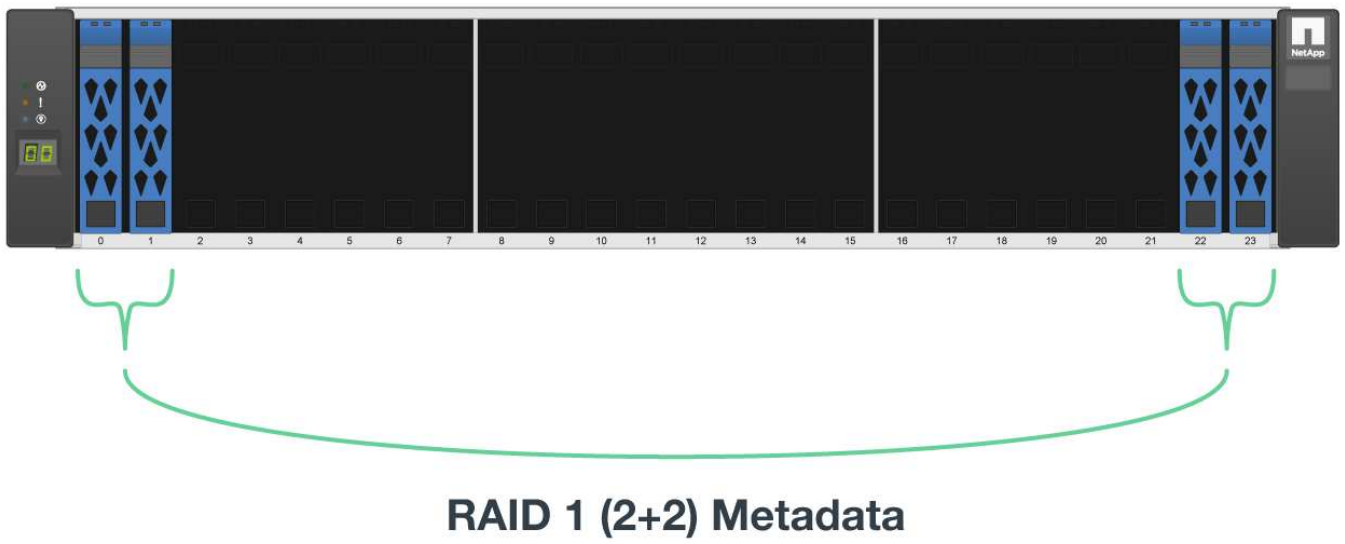
Controladores

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".

Informações sobre direitos autorais

Informações sobre marcas comerciais