Implantar software

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Índice

Implantar software . . . 1
- Configurar nós de arquivo e nós de bloco . . . 1
  - Configurar nós de arquivo . . . 1
  - Configurar nós de bloco . . . 2
- Ajuste as configurações do sistema do nó de arquivo para obter desempenho . . . 2
  - Utilize a interface UEFI para ajustar as definições do sistema . . . 2
  - Use Redfish API para ajustar as configurações do sistema . . . 3
- Configure um nó de controle do Ansible . . . 4
- Crie o inventário do Ansible . . . 5
  - Passo 1: Defina a configuração para todos os blocos de construção . . . 6
  - Etapa 2: Defina a configuração para nós individuais de arquivo e bloco . . . 7
  - Etapa 3: Defina a configuração que deve ser aplicada a todos os nós de arquivo e bloco . . . 9
  - Etapa 4: Defina a configuração que deve ser aplicada a todos os nós de arquivo . . . 10
  - Etapa 5: Defina a configuração para o nó de bloco comum . . . 16
- Definir o inventário do Ansible para os componentes básicos do BeeGFS . . . 18
  - Etapa 1: Crie o arquivo de inventário do Ansible . . . 19
  - Etapa 2: Configurar o inventário para um componente básico de gerenciamento, metadados e armazenamento . . . 19
  - Passo 3: Configure o inventário para um bloco de construção de metadados e armazenamento . . . 25
  - Etapa 4: Configure o inventário para um componente básico somente de armazenamento . . . 29
- Implantar o BeeGFS . . . 32
- Configurar clientes BeeGFS . . . 34

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

Configuração do sistema	Mude para
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.

Informações sobre direitos autorais

Informações sobre marcas comerciais