Lustre と NetApp Eシリーズ ストレージ

NetApp artificial intelligence solutions

NetApp
August 31, 2026

目次

Lustre と NetApp Eシリーズ ストレージ

Lustre と NetApp Eシリーズストレージ - 概要

Lustre と NetApp Eシリーズストレージは、HAサーバーペアとEシリーズオールフラッシュアレイの繰り返し可能な構成要素から構築された、AIおよびHPCワークロード向けの検証済み並列ファイルシステムソリューションです。

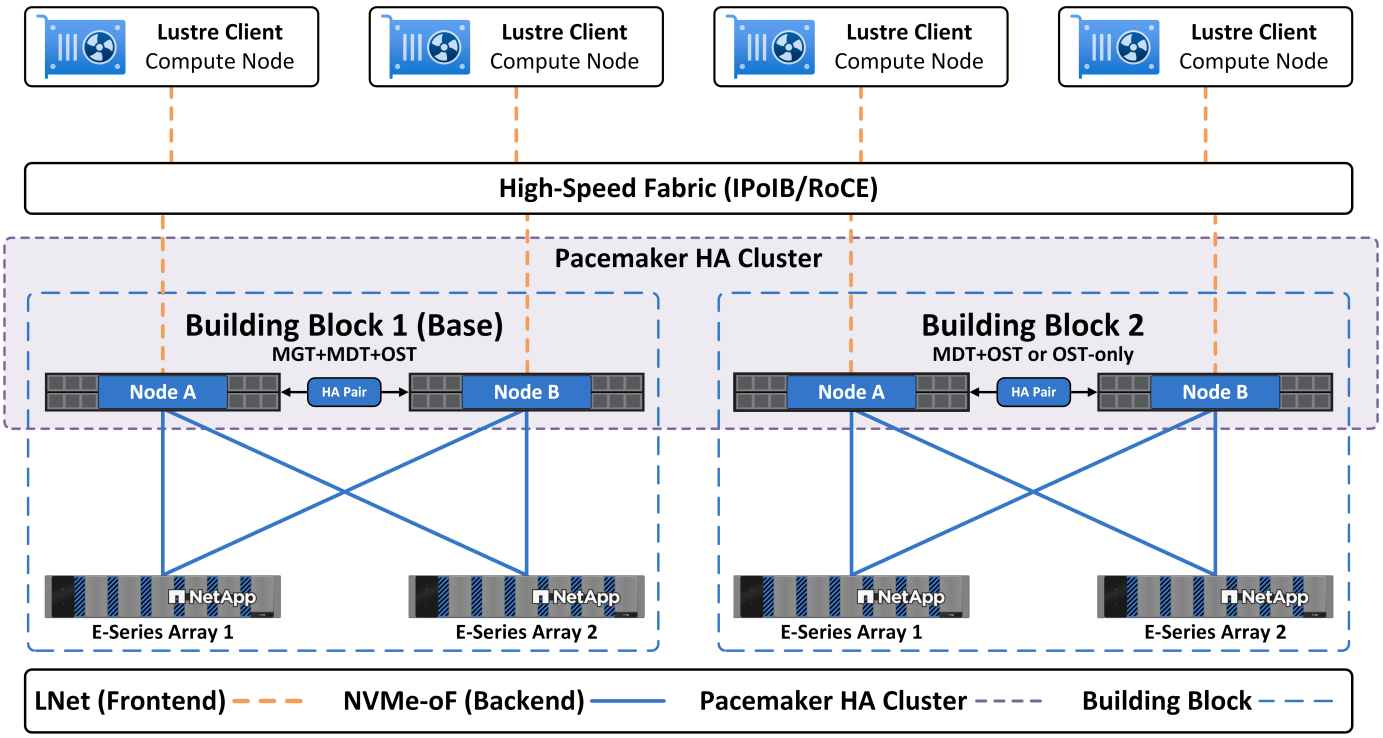
ソリューションの概要

Lustre と NetApp Eシリーズストレージはオープンソースの Lustre並列ファイルシステムと NetApp Eシリーズブロックストレージを組み合わせることで、データ集約型ワークロード向けのスケーラブルなハイパフォーマンス基盤を提供します。各ビルディングブロックは、高可用性（HA）用に構成された2つのOSS/MDSサーバーノードと2つのEシリーズ オールフラッシュアレイをペアにしています。バックエンドのNVMe-oF接続によりブロックI/Oがアレイに伝送され、別のLNetファブリックがクライアントとOSS/MDSサーバーノードを接続して並列ファイルシステムアクセスを実現します。

基本構成要素は、管理サーバー（MGS）、初期メタデータターゲット（MDT）、およびオブジェクトストレージターゲット（OST）をホストします。追加の構成要素は、ベースとなるMGSに登録され、ワークロードの需要増加に応じてメタデータ容量、データ容量、および総スループットを拡張します。

次の図は、複数のビルディングブロックとLustreクライアントがLNet経由で接続されたデプロイメントの例を示しています。

Lustre と NetApp Eシリーズストレージソリューションの概要



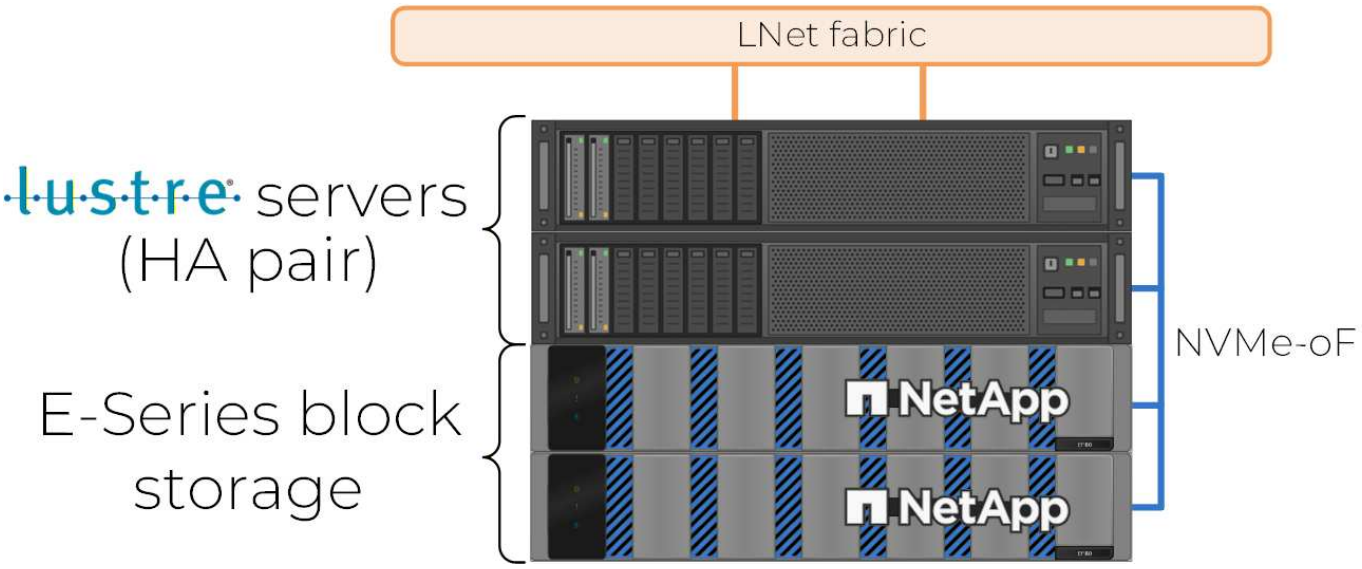
Lustre と NetApp Eシリーズ ストレージ - 解決策アーキテクチャ

Lustre と NetApp Eシリーズ ストレージが、構成要素、高可用性、ネットワークトポロジー、およびクライアントをどのように活用して、容量、パフォーマンス、およびフェイルオーバーを計画できるようにするかについて説明します。

ビルディングブロック設計

ビルディングブロックは、NetApp Eシリーズ Lustre 解決策の基本的なスケーラブルな単位です。各ビルディングブロックは、Lustre オブジェクトストレージサーバー（OSS）およびメタデータサーバー（MDS）機能を提供する高可用性（HA）ペアとして構成された2つのサーバーノード、2つのNetApp Eシリーズ オールフラッシュアレイ、サーバーとアレイ間の専用バックエンド NVMe over Fabrics（NVMe-oF）接続、およびクライアントとノード間通信のための専用フロントエンド Lustre ネットワーキング（LNet）接続で構成されています。PacemakerとCorosyncのHAクラスタは、1つ以上のビルディングブロックからのサーバーペアを含むことができます。NetApp `ansible-lustre`コレクションは、Ansible 自動化を使用して Lustre サービスをデプロイおよび構成します。

Lustre ビルディングブロック



構成要素は、ファイルシステムの成長要件に合わせて拡張可能です。すべてのファイルシステムには、1つの基本構成要素が必要です。基本構成要素は、管理ターゲット（MGT）上に管理サーバー（MGS）をホストし、初期メタデータターゲット（MDT）およびオブジェクトストレージターゲット（OST）の容量を提供します。追加の構成要素はファイルシステムを拡張し、基本構成要素のMGSに対してMDTおよびOSTターゲットを登録します。モジュール式のアプローチにより、ワークロードのニーズに応じて容量とパフォーマンスを個別に拡張できます。

NetAppは、ほとんどのデプロイメントに対して以下の構成要素構成プロファイルを推奨します。MGS、MDT、およびOSTのカウントは、Eシリーズ ストレージアレイのパフォーマンスと容量を最大化するように最適化された推奨デフォルト値です。ワークロードのニーズに合わせて、Ansibleインベントリ内のターゲット数、ボリュームサイズ、およびEシリーズのドライブ割り当てを調整してください。たとえば、より大きなメタデータストレージ容量を必要とするデプロイメントでは、同じビルディングブロックハードウェア内でより多くのMDTをプロビジョニングし、OSTを少なくすることができます。

以下の表は、構成要素の種類と推奨される目標数をまとめたものです。

タイプ	MGS	MDT	OST	用途
基本	1	8	32	ファイルシステムの最初の構成要素。MGSおよび初期のMDTとOSTの容量をホストします。
MDT+OST	0	8	32	メタデータとデータ容量を追加します。基本構成要素MGSに対して登録します。
OST のみ	0	0	32	データ容量のみを追加します。基本構成要素MGSに対して登録します。

- ドライブの種類とプールレイアウト： Eシリーズは従来のRAIDボリュームグループとダイナミックディスクプール（DDP）をサポートします。TLC NVMeドライブの場合、OSTストレージにはRAID 6ボリュームグループを、MGSおよびMDTストレージにはRAID 1ボリュームグループを使用してください。QLC NVMeドライブの場合は、共有ドライブプールにDDPを使用してください。
- ＊ターゲットの分散：＊各ビルディングブロック内のLustreターゲットは、両方のOSS/MDSサーバーノードと両方のEシリーズ アレイ間でバランスよく配置されます。このレイアウトにより、I/OがサーバーのCPUとアレイコントローラーに分散され、NVMe-oFパスのパフォーマンスが向上し、冗長性が維持されます。"ハードウェア コンポーネント"の"ターゲット分散とNVMe-oF接続"を参照してください。

サポートされているオペレーティングシステム、Lustreバージョン、アレイファームウェア、および関連するビルディングブロックコンポーネントは、"NetApp Interoperability Matrix Tool（IMT）"の **E-Series Lustre** に記載されています。詳細なボリューム数、ドライブレイアウト、およびサイジングガイドラインについては、"ハードウェア コンポーネント"を参照してください。ソフトウェアコンポーネントについては、"ソフトウェアコンポーネント"を参照してください。

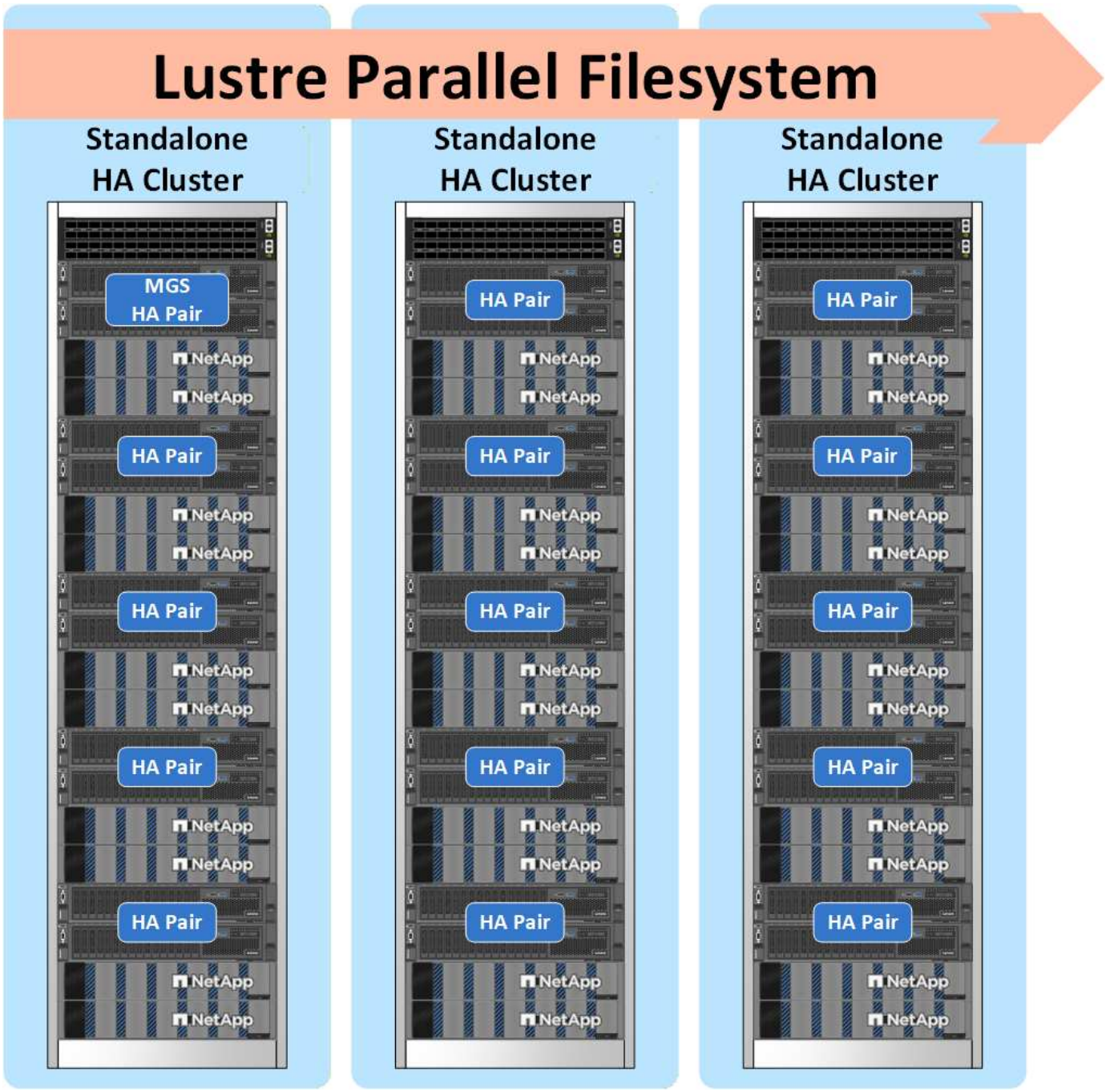
スケーリングに関する推奨事項

Lustreファイルシステムは、メタデータ容量、データ容量、またはその両方が制限要因となった場合に、構成要素を追加することで拡張します。MDTはファイル数とメタデータIOPSに基づいてスケーリングし、OSTは容量と集約スループット要件に基づいてスケーリングします。

1. ファイルシステムごとに**1**つの基本ビルディングブロック： 正確に1つの基本ビルディングブロックをデプロイします。これは、MGS、初期MDT、およびOSTターゲットをホストします。
2. 追加のビルディングブロックプロファイル： 既存のMDT容量が十分で、データ容量またはスループット要件を増やす必要がある場合は、OST専用のビルディングブロックを追加します。メタデータのパフォーマンスや名前空間の容量がボトルネックになった場合は、MDT+OSTのビルディングブロックを追加します。
3. **HA**クラスタの制限： PacemakerおよびCorosyncの各HAクラスタを5つのビルディングブロック（10個のLustreサーバーノード）に制限します。大規模な構成では、リソースの制約を回避するために、大規模なデプロイメントを複数のHAクラスタに均等に分割します。

次の図は、42Uラックに最大5つのビルディングブロックを積み重ねた状態を示しています（ラックごとに1つのPacemaker HAクラスタ）。複数のラックが単一のLustreファイルシステムに参加できますが、MGSをホストするのは基本ビルディングブロックのみです。

Lustreパラレルファイルシステムラックスケーリング



サイズの例については、"サイズガイド"を参照してください。

HAアーキテクチャ

Lustre with NetApp E-Series Storageは、NetApp Eシリーズ ストレージと統合された共有ディスク高可用性（HA）アーキテクチャを使用します。PacemakerがHAリソースを管理し、Corosyncがクラスタメンバーシップとメッセージングを提供します。これらが連携して、各ビルディングブロック内の2つのOSS/MDSサーバノード間におけるLustreターゲットのフェイルオーバーを管理します。マルチパスNVMe-oFが両方のOSS/MDSサーバノードを同じEシリーズ ボリュームに接続するため、必要に応じてどちらのノードでもターゲットサービスを引き継ぐことができます。

MGS、MDT、およびOSTはそれぞれ、Pacemakerリソースグループ内の依存関係とともにHAリソースとして

構成されます。Pacemakerは、リソースが正しい順序で起動および停止し、同じノード上に常駐し、一度に1つのノードでのみ実行されることを保証します。両方のノードから同じターゲットに同時にアクセスすると、基盤となるファイルシステムが破損する可能性があります。

- ターゲットのフェイルオーバー： OSS/MDSサーバーノードは両方ともクラスタ内でピアですが、各Lustreターゲットは一度に1つのノードでのみアクティブになります。Pacemakerの監視リソースは、各ターゲットとその依存関係の状態を監視し、ターゲットが現在のノードで利用できなくなった場合にフェイルオーバーをトリガーします。障害発生時、Pacemakerは影響を受けるリソースグループをパートナーノード上で正しい順序で再起動し、サービスが再開されるとクライアントは透過的にターゲットの新しい場所に再接続します。各ターゲットは単一のノード上でアクティブ化されるため、ファイルシステムが両方のノードからの同時書き込みにさらされることはありません。
- 共有ストレージ アクセス： マルチパスNVMe-oFパスは、各OSS/MDSサーバーノードをビルディングブロック内のすべてのEシリーズコントローラに接続するため、どのノードからでもすべてのターゲットボリュームにアクセスできます。サーバーノードに障害が発生した場合、パートナーノードは既に同じボリュームへのアクティブなパスを保持しているため、ストレージ接続を再構成することなくターゲットの所有権を引き継ぐことができます。アレイコントローラまたはパスに障害が発生した場合、マルチパスはI/Oを稼働中のコントローラにリダイレクトします。この二重冗長性により、LustreターゲットはOSS/MDSサーバーノード間またはアレイコントローラー間で、バッキングボリュームへのアクセスを失うことなくフェイルオーバーできます。
- **STONITH** フェンシング： 障害が発生すると、Pacemaker は障害のあるノードと通信して、ターゲットが停止していることを確認できない場合があります。Pacemaker は、他の場所でこれらのターゲットを再起動する前に、障害が発生したノードをフェンスし、理想的には電源を遮断して、そのノードが確実に停止していることを確認します。フェンシングは、両方のノードが同時に同じターゲットにアクセスして基盤となるファイルシステムを破損させるスプリットブレイン状態を防ぎ、フェイルオーバー時のデータ整合性を維持します。NetApp は、Redfish 対応のベースボード管理コントローラー（BMC）を搭載したサーバーに `fence_redfish`を推奨します。その他のフェンスエージェント（ `fence_apc`など）もサポートされています。

クラスタ管理とフェンシング設定については、"ansible-lustreコレクション"の"HAユーザーガイド"を参照してください。

ネットワーク アーキテクチャ

このソリューションでは、ストレージバックエンドのトラフィックとLNetフロントエンドのトラフィックにそれぞれ別のネットワークパスを使用します。

- バックエンド（**NVMe-oF**）： OSS/MDSサーバーノードは、NVMe/InfiniBandまたはNVMe/RoCEを使用して、NVMe-oF経由でEシリーズ アレイに接続します。NVMe-oFパスは、LustreターゲットサービスとEシリーズ ボリュームの間でブロックI/Oを転送します。EF80の6つのHCAバックエンドのケーブル接続については、"Lustre ハードウェアのラックとケーブル接続"を参照してください。ノードとアレイ間での推奨されるターゲット配置については、"ハードウェア コンポーネント"の"ターゲット分布"を参照してください。
- フロントエンド（**LNet**）： LustreクライアントとOSS/MDSサーバーノードは、NVIDIA OFEDスタックの`@o2ib`ネットワークタイプを使用してLNet経由で通信します。InfiniBandとRoCEはどちらも検証済みのフロントエンドトランスポートです。ansible-lustreコレクションは、マルチレールLNet用のすべてのフロントエンドインターフェースを設定します。マルチレールLNetは、複数のポートを集約することで帯域幅を拡大し、クライアントおよびノード間のトラフィックに対してパスの冗長性を提供します。
- 管理とクラスター： アウトオブバンド管理ネットワークは、サーバー管理、BMC アクセス、およびアレイ管理を提供します。STONITH フェンシングエージェント（ `fence_redfish`など）は、このネットワークを使用してサーバーの BMC にアクセスし、障害が発生したノードから電源を遮断します。Corosync は、フロントエンドファブリックに加えて、このネットワーク上でクラスタ通信を追加リングとして実行

することもできます。Corosync を複数のリングで構成することで、クラスタメッセージングの冗長性が向上し、単一のネットワーク障害によってクォーラムが中断されるリスクが軽減されます。

Lustre クライアント

Lustreクライアントは、クライアントカーネルモジュールをロードし、OSS/MDSサーバーノードへのLNet接続を確立し、ファイルシステムをマウントして、単一の一貫性のあるPOSIX準拠の名前空間をアプリケーションに提供します。入出力は、アクティブなOST全体に並列に分散されます。クライアントノードはビルディングブロックの外部にあり、フロントエンドのLNetファブリックを介してファイルシステムを使用します。クライアントオペレーティングシステムはこの解決策のIMTには記載されていません。テスト済みのクライアントディストリビューションと、デプロイされたLustreリリースでサポートされているカーネルバージョン（例：Red Hat Enterprise Linux 9、SUSE Linux Enterprise Server 15、Ubuntu 24.04）については、"Lustre サポートマトリックス"を参照してください。

クライアントノードは、OSS/MDSサーバーノードと同じLNetファブリックおよびネットワークタイプへの接続を必要とします。必要に応じて、LNetルーターは複数のLNetサブネットまたはファブリックをブリッジ接続できます。

NetAppは、Rocky Linux 9.8およびRed Hat Enterprise Linux 9.8用のLustreサーバーRPMパッケージ（ビルド済み）を提供します。NetAppは、事前に構築されたクライアントパッケージを提供していません。"netapp-lustreリポジトリ"にあるLustreソースコードから、クライアントオペレーティングシステムとカーネル用のクライアントパッケージをビルドしてください。

クライアント パッケージのインストール後、"ansible-lustreコレクション" のオプションの `lustre_client` ロールが、クライアント ネットワーク インターフェイスと LNet を設定し、選択されている場合はマルチレール LNet を有効にし、接続性を検証して、永続的な systemd マウント ユニットを管理します。このロールは Lustre をビルドしません。このロールは、 `lustre_client_packages` が設定されている場合にのみクライアント パッケージをインストールします。必要に応じて、クライアントを手動で設定することもできます。段階的な手順については、"解決策をデプロイする" および "lustre_client ロールのドキュメント" を参照してください。

Lustre と NetApp Eシリーズストレージ - ハードウェアコンポーネント

サーバー、NetApp Eシリーズ アレイ、ストレージ レイアウト、およびネットワークのハードウェア要件は、NetApp Eシリーズ ストレージを使用して Lustre のサイズ設定と注文を行う際にご使用ください。ソフトウェア コンポーネントについては、"ソフトウェアコンポーネント"を参照してください。

サーバー要件

このソリューションは、BYOS（Bring Your Own Server：サーバー持ち込み）モデルを採用しています。各構成要素には、以下の仕様を満たすか、それを上回る2つのOSS/MDSサーバーノードが必要です。

ノード仕様

以下の表は、OSS/MDSノードごとの最小サーバー要件を示しています。

コンポーネント	要件
数量	ビルディングブロックごとに2つのノード
CPU	AMD EPYCまたはIntel Xeon、32コア以上
メモリ	256 GB DDR5（最小）
ネットワークHCA	6つのデュアルポート200Gb HCA（HCA接続を参照）
PCIeスロット	2× PCIe Gen5 x16 および 4× PCIe Gen5 x8 （PCIeスロットの要件 を参照）
ブートドライブ	RAID 1構成のドライブ2台（ソフトウェアRAIDまたはハードウェアRAIDを推奨）

NetAppは、Lenovo ThinkSystem SR665 V3サーバーを使用してソリューションを検証しました。これらの要件を満たす場合、他社製の同等サーバーもサポート対象となります。

HCA接続

以下の表は、OSS/MDSサーバーノードごとのHCA、LNetフロントエンドポート、およびNVMe-oFパスの要件を示しています。

ノードあたりの**HCA**数	**Lustre**ノードあたりの**LNet**ポート数	**Lustre**ノードあたりの**NVMe-oF**パス数	**HCA**の例
6（12ポート）	4	合計8個（各アレイに4個ずつ）	MCX755106AS-HEAT（デュアルポート200Gb PCIe gen5カード）

NetApp は、EF80ストレージ アレイの帯域幅を最大化するために、ノードあたり6台のHCAを推奨します。

PCIeスロットの要件

EF80ビルディングブロック内の各OSS/MDSサーバーノードには、6つのデュアルポートHCAが搭載されています。内訳は、LNet用が2つ、NVMe-oF用が4つです。スロット幅は各HCAで使用可能な帯域幅を決定するため、カード単体の幅ではなく、各スロットとライザーの電気的幅を確認してください。Gen5 x8スロットにGen5 x16カードを挿入すると、x8で動作します。

- **LNet HCAs:** フロントエンドのスループットを最大限に引き出すには、PCIe Gen5 x16 スロットにインストールしてください。
- **NVMe-oF HCAs:** PCIe Gen5 x8 または PCIe Gen5 x16 スロットに取り付けます。
- **NUMA**バランス： HCAを2つのNUMAゾーンに均等に分割し、各ゾーンに2つのLNetインターフェースと4つのNVMe-oFインターフェースを配置します。

以下の表は、EF80ビルディングブロック内の各OSS/MDSサーバーノードに必要な最小スロット数を示しています。

トラフィックタイプ	ノードあたりの**HCA**数	最小スロット幅	**NUMA**ゾーンごとのスロット数
LNet	2	PCIe Gen5 x16	1
NVMe-oF	4	PCIe Gen5 x8	2

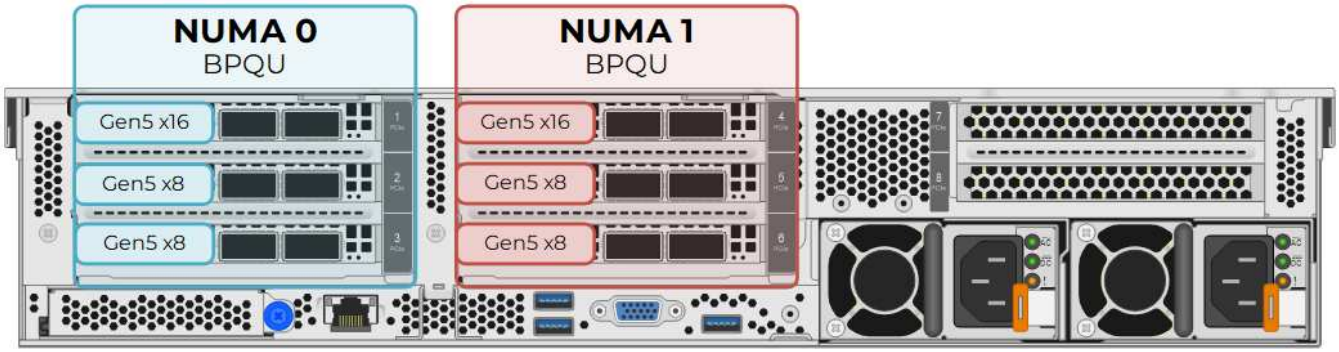
オプションとして、デュアルポートHCAのポートを分割し、一方のポートでLNetトラフィックを、もう一方のポートでNVMe-oFトラフィックを伝送するように設定できます。これら4つのHCAをすべてPCIe Gen5 x16スロットに取り付けて、すべてのLNetポートがフルスループットに達するようにします。その後、残りのNVMe-oFポート用に、さらに2つのHCAをGen5 x8またはGen5 x16スロットに追加します。

▼ **Lenovo ThinkSystem SR665 V3** のライザー構成例

以下の2つのライザーの組み合わせはどちらも、EF80ビルディングブロックのスロット要件を満たしています。

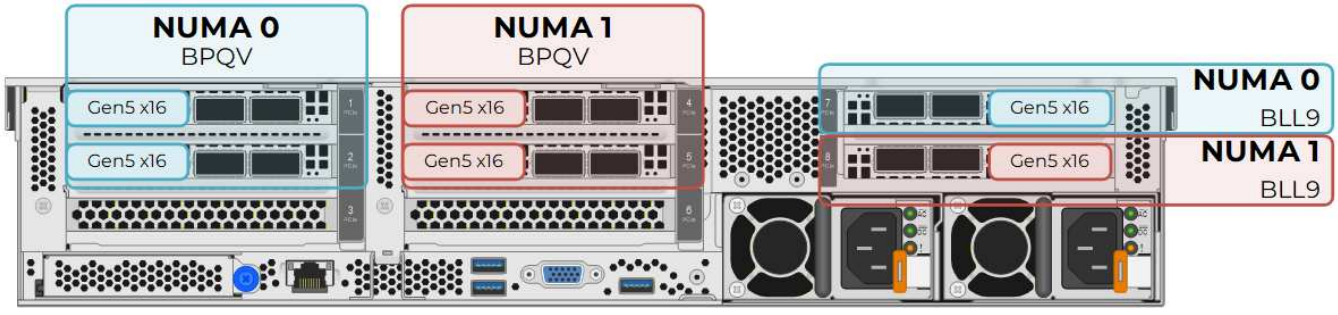
最小スロット幅

ライザー位置1と2にBPQUライザーを取り付けてください。各BPQUライザーには、PCIe Gen5 x16スロットが1つとPCIe Gen5 x8スロットが2つ搭載されています。これらのライザーを組み合わせることで、LNet用のGen5 x16スロットが2つ、NVMe-oF用のGen5 x8スロットが4つ提供され、これらはNUMAゾーン間で均等に分配されます。



すべての**Gen5 x16**スロット

ライザー位置1と2にはBPQVライザーを、ライザー位置3にはBLL9ライザーを取り付けてください。各BPQVライザーには、2つのPCIe Gen5 x16スロットが搭載されています。BLL9ライザーは、NUMAゾーン0にスロット7、NUMAゾーン1にスロット8（いずれもPCIe Gen5 x16）を提供します。この組み合わせにより、NUMAゾーンごとに3つずつ、合計6つのGen5 x16スロットが提供され、同じHCAのポート間でLNetとNVMe-oFのトラフィックを分割することをサポートします。



EF80標準ビルディングブロック

EF80は、このソリューションリリースにおいて検証済みの標準プラットフォームです。

アレイ仕様

以下の表は、EF80アレイの構成要素（2つのアレイ）の仕様を示しています。

コンポーネント	仕様詳細
モデル	NetApp EF80
フォーム ファクタ	2Uベースシャーシ、24個の内部NVMe SSDスロット
コントローラ	デュアルコントローラー（AとB）
システムの	アレイあたり24× NVMe SSD
入出力接続	検証済みのLustre設計では、アレイあたり8× 200Gb NVMe/IBまたはNVMe/RoCEホストポート

EF80プラットフォームは、各コントローラに2ポートのホストI/Oモジュールを3つ搭載した場合、アレイあたり最大12個のホストポートをサポートします。検証済みのLustre設計では、スロット1と2にホストI/Oモジュールを使用し、アレイあたり8つのホストポートを提供します。

各EF80アレイは、コントローラ間のミラーリングのために専用のスロット4 I/Oモジュールも使用します。コントローラAのポート4aをコントローラBのポート4aに、コントローラAのポート4bをコントローラBのポート4bに接続してください。これらの接続はキャッシュミラーリングとI/O転送を提供するものであり、ホストのNVMe-oFトラフィックには使用されません。参照してください"EF50とEF80のコントローラ間ミラーリング接続ケーブルを接続する"。

EFシリーズの全モデルの詳細仕様については、"NetApp EFシリーズ オールフラッシュ アレイ データシート"を参照してください。

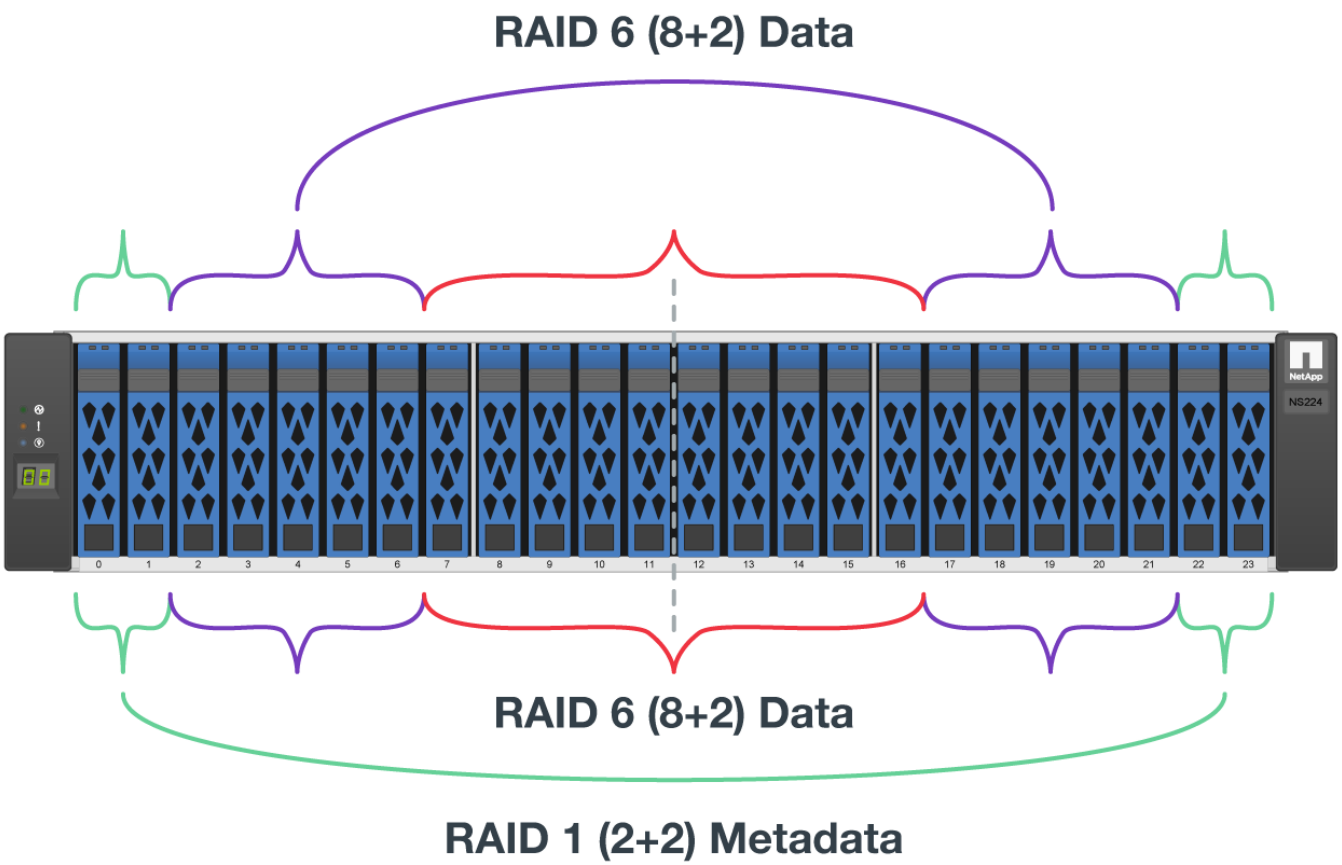
ドライブ構成（アレイあたり**24**台のドライブ）

基本ビルディングブロック内の各EF80アレイは、24個のNVMeドライブスロットすべてを使用します。

- MGS/MDTストレージ用のRAID 1構成のドライブ4台（共有ボリュームグループまたはDDP割り当て）
- OSTストレージ用にRAID 6構成のドライブ10台（最初のOSTプール）
- OSTストレージ用（セカンドOSTプール）にRAID 6構成のドライブ10台を使用

このストレージ レイアウトは、3.84 TB、7.68 TB、または 15.3 TB のドライブを従来のボリュームグループで使用する場合に適用されます。30.7 TB または 61.4 TB の Capacity Flash （QLC） ドライブを使用する場合は、24 台のドライブすべてにわたって単一のダイナミック ディスク プールをプロビジョニングし、プール内に MGS および MDT ボリューム用の RAID 1 ボリュームを作成し、OST にはデフォルトの RAID 6 ボリュームを使用します。

1.92 TBのドライブは、現時点ではこのソリューションには推奨されていません。上記に記載されている、検証済みのドライブ容量のいずれかを使用してください。



ボリュームとターゲットカウント（基本構成要素）

以下の表は、基本構成要素ごとのMGS、MDT、およびOSTの数を示しています。

対象タイプ	**BB1**個あたりのカウント	**RAID /** プール	**Notes**
MGS	1	RAID 1	基本構成要素のみ；アレイ1
MDT	8	RAID 1	配列あたり4個
OST	32	RAID 6 (TLC) または DDP (QLC)	アレイあたり16

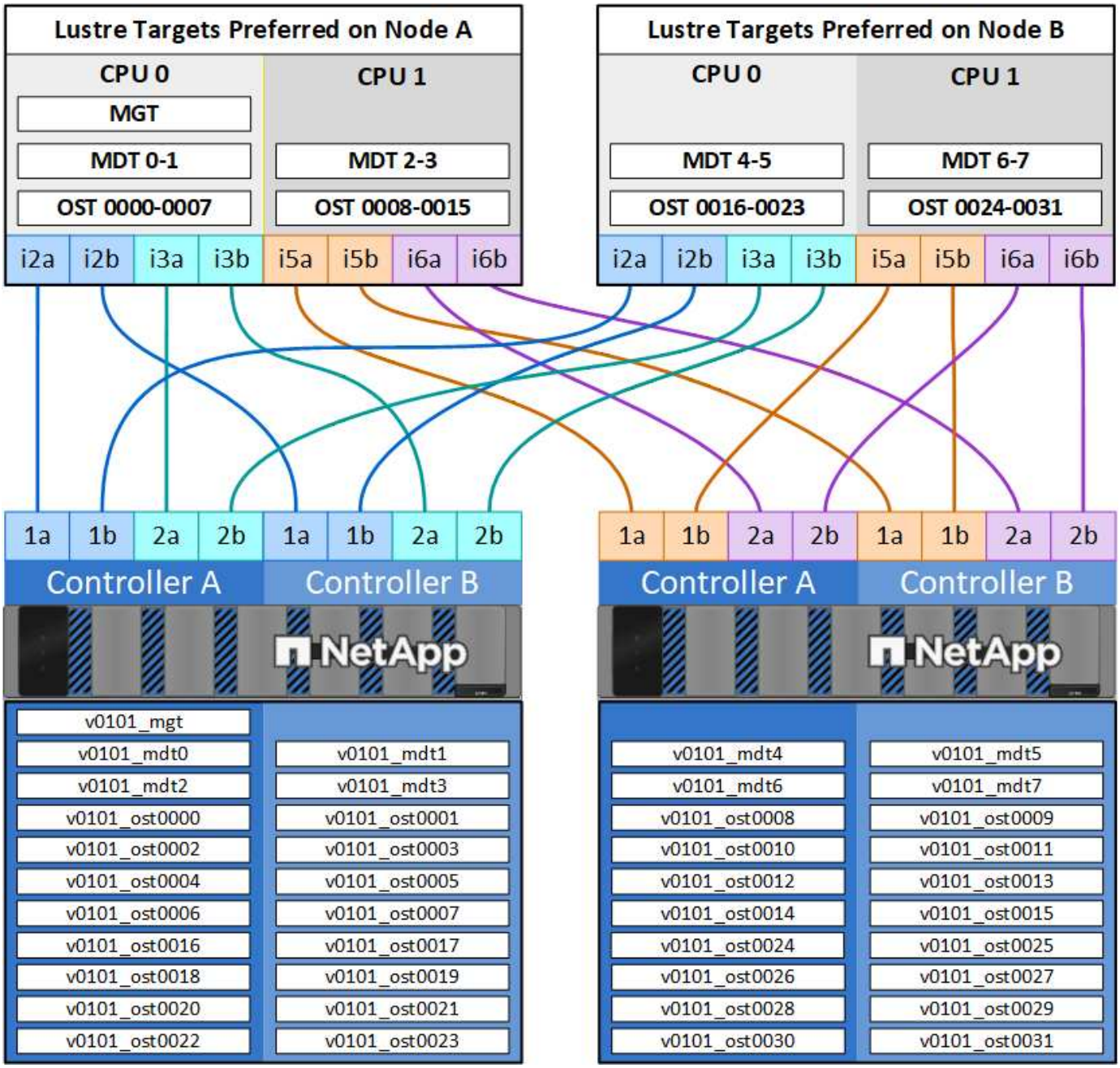
Ansibleインベントリを作成する際の出発点として、以下のボリュームサイズ設定ガイドラインを使用してください。

Volume	推奨サイズ	**Notes**
MGS	5～10 GiB	設定データのみ
MDT	RAID 1容量 ÷ MDT数	それぞれ約1～2 TiB（標準値）
OST	RAID 6またはDDP容量÷プールあたりのOST数	ドライブ容量に応じてスケーリングします。"サイズガイド"を参照してください。

プライマリビルディングブロックのボリューム分散

次の図は、基本EF80ビルディングブロックにおけるOSS/MDSサーバーノードおよびEシリーズアレイ全体にわたるLustreターゲットの推奨配置とNVMe-oF接続を示しています。図では、管理ターゲットを **MGT**（管理ターゲット）と表記しています。1つのMGTは、このドキュメントの別の箇所で参照されているMGS（管理サーバー）サービスをホストします。

基本構成要素の目標分布（**EF80**）



以下の表は、ボリュームが2つのアレイ間でどのように分散されているか、および各ターゲットがどちらのサーバーを優先するかを示しています。

対象タイプ	アレイ**1**	アレイ**2**	サーバー**1**	サーバー**2**	**Notes**
MGS	1	0	1	0	基本ビルディングブロックのみ

対象タイプ	アレイ1	アレイ2	サーバー1	サーバー2	Notes
MDT	4	4	4	4	各サーバーは、各アレイから2つのMDTを優先的に選択します。
OST	16	16	16	16	RAID 6プールあたり8 × アレイあたり2プール、または同等のDDP割り当て
総ボリューム数	**21**	**20**	**21**	**20**	

ストレージプールの選択：TLC vs QLC

アレイ内のNVMeドライブ容量に基づいてEシリーズのプールタイプを選択してください：

- **3.84 TB**、**7.68 TB**、および **15.3 TB** ドライブ： OST ボリュームには **RAID 6** ボリュームグループを、MGS および MDT ボリュームには **RAID 1** ボリュームグループ を使用するか、共有ドライブプールには DDP を使用します。
- **30.7 TB** および **61.4 TB** 容量のフラッシュ **(QLC)** ドライブ： ダイナミック ディスク プール **(DDP)** のみを使用してください。MGS および MDT ストレージ用に、DDP 内に RAID 1 ボリュームを作成します。DDP から OST ストレージ用の RAID 6 ボリュームを作成します。

ドライブのサイズとレイアウト別の構築ブロックごとの使用可能容量の見積もりについては、"サイズガイド"を参照してください。

ネットワーク要件

バックエンド（**NVMe-oF**）

- NVMe/InfiniBandまたは各OSS/MDSノードと各EシリーズアレイとのNVMe/RoCE
- NVMe/RoCE バックエンド インターフェイス上の MTU 9000 （通常）
- 各Lustreノードからストレージアレイへの8つのパス（各アレイへ4つ）
- EF80は、ノードごとに6つのHCAを使用します（NVMe-oFの場合はi2、i3、i5、i6、LNetの場合はi1、i4）。ノードAを、ラベルの末尾が `a` であるコントローラポートにケーブル接続します（例： `1a` や `2a`）。ノードBを、ラベルの末尾が `b` であるコントローラポートにケーブル接続します（例： `1b` や `2b`）。"EF80 6 HCAバックエンドケーブル接続"を参照してください。

フロントエンド（**LNet**）

- LNet の IPoIB または RoCE (`@o2ib` ネットワークタイプ)
- RoCEフロントエンドインターフェースのMTU 9000（標準）
- フロントエンドポートが4つ利用可能な場合（EF80：i1およびi4 HCA、両方のポート）、マルチレールLNetの使用を推奨します。
- 専用のInfiniBandまたはRoCEスイッチファブリックで、OSS/MDSサーバーノードとLustreクライアントを接続
- RoCEファブリックではロスレスRoCE（PFC）を推奨します

管理

- サーバーBMCおよびアレイ管理ポート用の帯域外管理ネットワーク
- 専用のCorosyncネットワークまたは共有フロントエンドファブリック（サイトによって異なる）

Lustre と NetApp Eシリーズストレージ - ソフトウェアコンポーネント

NetApp Eシリーズストレージを使用してLustreをデプロイするソフトウェアスタックとAnsibleオートメーションを確認してください。サーバー、ストレージ、およびネットワークについては、"ハードウェア コンポーネント"を参照してください。

ビルディングブロックソフトウェア

以下の表は、EシリーズアレイおよびOSS/MDSサーバーノードで使用されるソフトウェアコンポーネントの一覧です。サポートされているバージョンとコンポーネントの組み合わせについては、"NetApp Interoperability Matrix Tool（IMT）"を信頼できる情報源として参照してください。**E-Series Lustre** を検索し、展開前にオペレーティングシステム、Lustre、SANtricity OS、DOCA-OFED、HCAファームウェア、およびHAパッケージのバージョンを確認してください。

コンポーネント	機能
SANtricity OS	各Eシリーズ アレイ上で動作し、SANtricity System Manager、デュアルアクティブ コントローラによる高可用性と自動I/Oパス フェイルオーバー、自動ロード バランシングとパス監視、ダイナミック ディスク プールと従来のRAID、自動ドライブ再構築、ミラーリング キャッシュ保護、データ保証、プロアクティブなドライブ正常性監視、オンラインでのソフトウェア、ファームウェア、および構成変更を提供します。
Lustre	クライアントには単一のPOSIX準拠の名前空間を提供すると同時に、メタデータとファイルデータを複数のターゲットに分散させて並列アクセスを可能にします。管理サーバー（MGS）はファイルシステム構成を保存し、メタデータターゲット（MDT）は名前空間を管理し、オブジェクトストレージターゲット（OST）はEシリーズボリュームにファイルデータを保存します。この分離により、構成要素を追加することでパフォーマンスと容量を拡張できます。
Red Hat Enterprise Linux (RHEL) または Rocky Linux	Lustreターゲットサービス、HAクラスタリング、ストレージおよびネットワーク接続のためのカーネル、システムサービス、およびパッケージフレームワークを提供します。サポートされているオペレーティングシステムのリポジトリには、Pacemaker、Corosync、フェンスエージェント、NVMe-oF、およびマルチパスのパッケージも含まれており、これらはソリューションスタックとしてまとめてテストされます。
高可用性クラスタサービス（Pacemaker、Corosync、およびフェンスエージェント）	これらのサービスを組み合わせることで、OSS/MDSサーバーノード全体にわたるLustreターゲットに対して、連携のとれた高可用性が実現されます。クラスタのメンバーシップとクォーラムを維持し、リソースを監視し、順序付けられたフェイルオーバーを調整し、応答しないノードを隔離してから、他の場所で共有ターゲットをアクティブ化します。この調整により、Eシリーズボリュームへのスプリットブレインアクセスを防止し、障害発生時にファイルシステムの整合性を保護します。

コンポーネント	機能
NVIDIA DOCA-OFED	HCAドライバ、RDMAライブラリ、およびInfiniBandとRoCEファブリック向けの管理ユーティリティを提供します。同じソフトウェアスタックが、Eシリーズアレイへのバックエンド NVMe-oFの低レイテンシアクセスと、Lustreクライアントとのフロントエンド LNet通信の高スループットをサポートします。

NetAppは、Rocky Linux 9.8およびRed Hat Enterprise Linux 9.8用のLustreサーバーRPMパッケージ（ビルド済み）を提供しています。

Ansibleによる自動化

Lustre と NetApp Eシリーズ ストレージは、Eシリーズ アレイおよび OSS/MDS サーバー ノードへの管理アクセスを持つコントロール ノードから Ansible オートメーションを使用して提供およびデプロイされます。Ansible インベントリは、アレイ、サーバー、ストレージ割り当て、ネットワーク インターフェイス、HA クラスタ リソースなど、目的のソリューションをモデル化します。

NetApp EシリーズLustre Ansibleコレクションは、SANtricityおよびホストコレクションを使用してエンドツーエンド展開をオーケストレーションします。これらのコレクションを組み合わせることで、Eシリーズ ストレージのプロビジョニングとマッピング、サーバーオペレーティングシステムの準備、NVMe-oFおよびLNet接続の構成、Lustreターゲットのデプロイ、およびPacemakerリソースの作成が行われます。この自動化により、設定の再現性が向上し、既存のファイルシステムに構成要素を追加する作業が簡素化されます。

以下の表は、Ansibleコントロールノードで使用される主要なソフトウェアパッケージをまとめたものです。現在のパッケージの依存関係については、"NetApp EシリーズLustre Ansibleコレクション"を参照してください。

パッケージ	依存または目的
Python	Ansibleの実行環境と必要なPythonモジュールを提供します。
ansible-core	Pythonが必要で、NetApp EシリーズのAnsibleコレクションを実行します。
追加のPythonパッケージ	`cryptography`、 `ipaddr`、 `netaddr`、 `passlib`、 `resolvelib`、 `jinja2`、および `pyyaml`
NetApp EシリーズLustre Ansibleコレクション	Lustre、LNet、NVMe-oFホスト接続、およびPacemakerリソースを設定します。
NetApp Eシリーズ SANtricity Ansible コレクション	SANtricity Web Services API を使用して、Eシリーズ アレイ、ストレージ プール、ボリューム、およびホスト マッピングを設定します。
NetApp Eシリーズ ホスト Ansible コレクション	OSS/MDSサーバーノードをEシリーズアレイへのLNetおよびNVMe-oF接続用に準備します。

Lustreクライアントソフトウェア

Lustreクライアントソフトウェアを使用すると、システムはフロントエンドのLNetファブリックを介して並列ファイルシステムにアクセスできます。各クライアントは、実行中のカーネルとサーバーにデプロイされているLustreリリースの両方と互換性のあるLustreカーネルモジュールとLustreクライアントユーティリティが必要です。

"Lustre サポートマトリックス"を使用して、お使いのLustreバージョンに対応するクライアントオペレーティ

ングシステムとカーネルを見つけてください。NetAppは、事前にビルドされたLustreクライアントパッケージを提供していません。"netapp-lustreリポジトリ"にあるNetApp提供のLustreソースコードから、クライアントオペレーティングシステムとカーネル用のクライアントパッケージをビルドしてください。Lustreクライアントパッケージがインストールされると、 `lustre_client`ロールはネットワークインターフェースとLNetを設定し、Lustreファイルシステムのためのpersistent systemdマウントユニットを作成できます。

クライアントの展開手順については、"解決策をデプロイする"を参照してください。

Lustre と NetApp Eシリーズストレージ - サイジングガイダンス

容量とメタデータのニーズに基づいて NetApp Eシリーズストレージのビルディングブロックで Lustreのサイズを決定し、容量を追加するタイミングを判断して、パフォーマンスに影響する要因を確認します。

容量サイジング

2台のEF80アレイで構成される基本ビルディングブロックは、次のLustreターゲットレイアウトを提供します。推奨されるターゲットの配置とNVMe-oFパスについては、"ハードウェア コンポーネント"の"プライマリビルディングブロックのボリューム分散"を参照してください。

コンポーネント	数	標準的なボリュームサイズ（ターゲットあたり）
MGS	1	5～10 GiB
MDT	8	サイズはドライブ容量とRAID構成によって異なります
OST	32	サイズはドライブ容量とRAID構成によって異なります

各EF80アレイは24台のNVMeドライブを使用します。対応容量は、従来のボリュームグループ（MGS/MDTの場合はRAID 1、OSTの場合はRAID 6）またはDDPを使用した場合、3.84 TB、7.68 TB、15.3 TBです。また、DDPのみを使用した場合、30.7 TBまたは61.4 TBのCapacity Flash（QLC）ドライブが対応します。1.92 TBドライブは、現時点ではこのソリューションには推奨されていません。ドライブスロットのレイアウトとプールの選択については、"ハードウェア コンポーネント"を参照してください。

以下の表は、EF80ビルディングブロック（アレイ2基、ドライブ48台）のおおよその使用可能容量を示しています。アレイの使用可能容量は、最終的なLustreファイルシステムの使用可能容量とは異なります。MDTのinode割り当て、ジャーナル、オーバープロビジョニング、およびインベントリボリュームの割合は、アプリケーションで使用可能な容量を減少させます。

ドライブ容量	レイアウト（アレイごと）	おおよそ使用可能な量（構成要素）
3.84 TB	RAID 1（2+2）+ RAID 6（10）+ RAID 6（10）	138.08 TB
3.84 TB	DDP（24ドライブ、2予約済み）	133.63 TB
7.68 TB	RAID 1（2+2）+ RAID 6（10）+ RAID 6（10）	276.35 TB
7.68 TB	DDP（24）	267.47 TB

ドライブ容量	レイアウト（アレイごと）	おおよそ使用可能な量（構成要素）
15.3 TB	RAID 1（2+2）+ RAID 6（10）+ RAID 6（10）	552.88 TB
15.3 TB	DDP（24）	535.15 TB
30.7 TB	DDPのみ	1,070.35 TB
61.4 TB	DDPのみ	2,162.70 TB

メタデータとデータのサイズを個別に設定し、その後、Ansibleインベントリでボリュームサイズを設定します。デプロイメント テンプレートは ldiskfs を使用してターゲットをフォーマットします。MDT は Lustre のデフォルトの inode 比率を使用し、OST テンプレートは `-i 4096`を設定します。

- メタデータ（**MDT**）： 予想されるファイル数に基づいてMDTのサイズを決定します。ファイル数の増加を見込んで、予想されるファイル数の約2倍のファイル数を確保してください。MDTのサイズが不足していると、OSTの容量に余裕があっても、新しいファイルの作成ができなくなる場合があります。
- データ（**OST**）： 使用可能な容量とスループットのニーズに基づいてOSTのサイズを決定します。
- **MGS:** ファイルシステム構成のためだけに、小さな管理ターゲット（5～10 GiB）を使用します。

選択したドライブ容量に合わせて、インベントリ内のMDTおよびOSTボリュームのサイズを調整してください。 `format_options.mkfsoptions`の `eseries_lustre_filesystem_mdt`または `eseries_lustre_filesystem_ost`は、サイトが異なるinode比率を必要とする場合にのみ変更してください。inode比率の背景については、"Lustre Wiki：Lustreチューニング"を参照してください。ビルディングブロックを追加するタイミングについては、[スケーリング]を参照してください。

スケーリング

容量やメタデータの負荷に応じて、構成要素を追加してください。

- **OST**のみ： ファイル数とメタデータの読み込みは既存のMDT容量内に収まっており、より多くのデータ容量または集計スループットが必要です。OSTを32台、OSS/MDSサーバーノードを2台追加します。
- **MDT+OST:** ファイル数またはメタデータレートがMDTの容量に近づいているか、メタデータのスループットとデータ容量の両方を増やす必要があります。MDTを8個、OSTを32個追加します。

既存のMDT上でメタデータが飽和状態にあるかどうかを確認するには、 `mdt.*.md_stats`を使用してください。各Pacemaker/Corosyncクラスターは、5つの構成要素（10個のOSS/MDSサーバーノード）に制限してください。大規模な展開の場合は、構成要素を複数のHAクラスターに均等に分割してください。スケーリングルールについては、"ソリューションアーキテクチャ"を参照してください。

以下の例は、1つのHAクラスタ内に、基本構成要素とOST専用の構成要素を備えたファイルシステムを示しています。

構成要素	タイプ	サーバ	配列	**MGS**	**MDT**	**OST**
BB1を使用したチャンク アップロード署名要求がサポートされるようになりました。	基本	2	2	1	8	32

構成要素	タイプ	サーバ	配列	MGS	MDT	OST
BB2	OST のみ	2	2	0	0	32
合計		4	4	1	8	64

BB2は、 `mgsnode=`を使用して、BB1のMGSに対してOSTターゲットを登録します。BB2のOSTインデックスはグローバルに割り当てられます（例えば、OST0032からOST0063まで）。これにより、どのインデックスもBB1と競合しません。

パフォーマンス

正式な検証では、LustreクライアントのIOR、mdtest、fioを使用して、相対的なスループット、IOPS、メタデータの動作を特性評価し、HAフェイルオーバーを検証します。これらのテストでは、性能を保証する数値は公表されていません。合成ベンチマークは最良のケースの動作を測定するものであり、実際のアプリケーションのパフォーマンスを反映していない可能性があります。

パフォーマンスは、構成要素の数にほぼ比例します。実際の結果は、ドライブの種類とプールのレイアウト、NVMe-oFパス数、LNetファブリックとクライアント数、およびデータI/Oとメタデータ I/Oの比率によって異なります。

ワークロード固有のサイジングおよびパフォーマンスに関する推奨事項については、NetApp アカウントチームにお問い合わせください。

ドライブのレイアウトとボリューム数については、"ハードウェア コンポーネント"を参照してください。デプロイ手順については、"解決策をデプロイする"を参照してください。現場レベルの在庫管理に関するガイダンスについては、"Lustre Ansibleインベントリをカスタマイズする"を参照してください。

解決策をデプロイする

NetApp Eシリーズストレージで**Lustre**をデプロイする

Lustre を NetApp Eシリーズストレージとともにデプロイするには、ハードウェア、環境準備、インベントリ、Ansible デプロイ、および検証の各タスクを順番に完了してください。

前提条件

デプロイを開始する前に、以下の点を確認してください。

- "サイズガイド"のサイジングとビルディングブロックのガイダンスに従い、"ハードウェア コンポーネント"に一致するハードウェアを選択しました。
- サーバー、HCA、Eシリーズ、オペレーティングシステム、Lustre、SANtricity OS、およびDOCA-OFEDの組み合わせが、"NetApp Interoperability Matrix Tool"の **E-Series Lustre** に記載されていることを確認しました。
- 計画されているビルディングブロック用のすべてのOSS/MDSサーバーノードとEシリーズアレイは現場に到着しており、ラックに取り付ける準備ができています。
- サーバーBMC、アレイ管理ポート、およびAnsibleコントロールノードとして機能するホストの資格情報とネットワーク アクセスがあります。

NetAppは、"NetApp EシリーズLustre Ansibleコレクション"を使用したLustre HAクラスターとLustreクライアントのデプロイをサポートします。Lustreターゲット、LNet、NVMe-oFホスト接続、またはPacemakerリソースを手動で設定しないでください。

導入ワークフロー

以下のタスクを順番に完了してください。

1. "Lustre ハードウェアのラックとケーブル接続"。

   LustreサーバーとEF80アレイをラックに設置し、バックエンドのNVMe-oFファブリックとフロントエンドのLNetファブリックをケーブルで接続します。

2. "Lustreのデプロイ環境を準備する"。

   アレイとサーバーを設定した後、コントロールノードに Ansible とその依存関係をインストールします。

3. "Lustre Ansibleインベントリをカスタマイズする"。

   サイトの構成に使用するテンプレートを選択し、サーバー、アレイ、ネットワーク、ビルディングブロック、および認証情報を入力します。

4. "Lustre HAクラスタとクライアントをデプロイする"。

   サーバーオペレーティングシステムを準備し、NVIDIA DOCA-OFEDをインストールし、メインのデプロイメントプレイブックを実行して、Lustreクライアントを設定します。

5. "Lustreのデプロイメントを検証および拡張する"。

   本番環境で使用する前に、クラスターの状態、マウントポイント、および容量を確認してください。ファイルシステムに新たな構成要素が必要になった場合は、既存のインベントリを拡張します。

関連情報

- "NetApp EシリーズLustre Ansibleコレクション"
- "NetApp Lustre ソースリポジトリ"
- "HA管理ユーザーガイド"
- "パフォーマンスチューニングガイド"
- "Lustreクライアント展開テンプレート"

Lustre ハードウェアのラックとケーブル接続

Lustre サーバーと NetApp EF80 アレイをラックに搭載し、各ビルディングブロックのバックエンド NVMe-oF ファブリックとフロントエンド LNet ファブリックをケーブル接続します。

開始する前に

HCA 数、PCIe スロット要件、パス要件、および MTU ガイダンスを "ハードウェア コンポーネント" で確認してください。

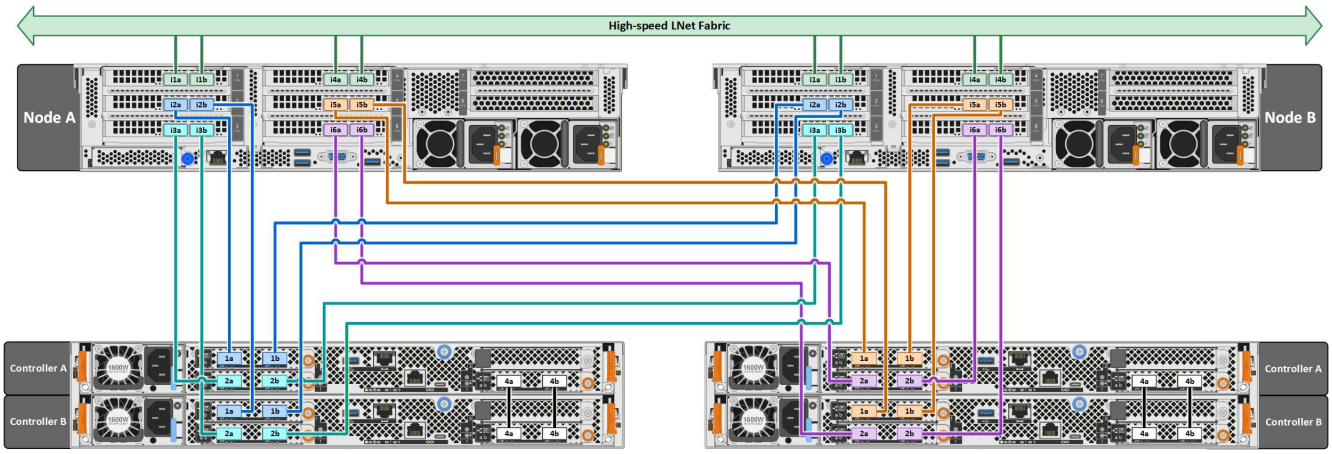
手順

1. すべての Lustre サーバーと E シリーズ ストレージ アレイを、構成要素のトポロジーに従ってラックに搭載し、ケーブルを接続します。

2. 各EF80アレイについて、コントローラAのポート4aをコントローラBのポート4aに、およびコントローラAのポート4bをコントローラBのポート4bにケーブル接続します。これらの専用接続はコントローラ間のミラーリング機能を提供するものであり、ホストトラフィックには使用されません。を参照してください。"EF50とEF80のコントローラ間ミラーリング接続ケーブルを接続する"

3. EF80の6 HCAトポロジーに従って、サーバーとアレイ間のバックエンドNVMe-oFネットワークをケーブル接続します。

EF80 6-HCA バックエンド ケーブル配線（**RoCE** または **InfiniBand**）



4. 各Lustreサーバー上の4つのフロントエンドLNetインターフェース（i1およびi4 HCA）とLustreクライアントを、専用のInfiniBandまたはRoCEスイッチファブリックに接続します。

5. RoCEを使用する場合は、優先フロー制御（PFC）を使用してロスレス動作のためにネットワーク ファブリックを設定してください。Ansibleプレイブックは、ビルディングブロック内のすべてのインターフェー

スでロスレスRoCEを設定します。

終了後の操作

"サーバー、アレイ、およびAnsibleコントロールノードを準備します"。

Lustreのデプロイメント環境を準備する

各 EF80 アレイと Lustre サーバーの管理アクセスおよび基本設定を構成し、ソリューションをデプロイするための Ansible コントロールノードを準備します。

Eシリーズアレイの準備

手順

1. 各コントローラーの管理インターフェイスを設定し、SANtricity System Manager に到達できることを確認します。
2. Ansibleインベントリで使用するアレイ名と管理者認証情報を設定します。
3. アレイが最適な状態であり、すべてのドライブが存在し、正しくレポートされていることを確認してください。

管理接続の設定方法については、"Eシリーズのマニュアル"を参照してください。

Lustreサーバーを準備する

手順

1. 各サーバーのベースボード管理コントローラー（BMC）を帯域外アクセス用に設定し、UEFI システム設定を最大パフォーマンスになるように設定します。
2. サポートされているオペレーティングシステム（RHELまたはRocky Linux）をインストールし、ホスト名と管理ネットワークポートを設定します。
3. **E-Series Lustre** ソリューションの要件を満たすように、すべてのファームウェアおよびソフトウェアパッケージを準備します。要件については、"ソフトウェアコンポーネント"を参照してください。

Ansibleコントロールノードを準備する

すべての Lustre サーバーと E シリーズアレイへの管理ネットワーク アクセスを持つ仮想または物理 Linux システムを、Ansible コントロールノードとして指定します。

以下の手順はUbuntu 24.04を使用します。別の Linux ディストリビューションの手順については、"Ansibleのインストールに関するドキュメント"を参照してください。

手順

1. Python、pip、およびPython仮想環境パッケージをインストールします。

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

2. Python仮想環境を作成してアクティブ化する：

```
python3 -m venv ~/pyenv
source ~/pyenv/bin/activate
```

3. 有効化された仮想環境にAnsibleと必要なPythonパッケージをインストールします：

```
pip install "ansible-core>=2.19" cryptography jinja2 ipaddr netaddr \
  passlib pyyaml resolvelib
```

4. NetApp EシリーズLustre Ansibleコレクションをインストールします。コレクションのインストールでは、SANtricityおよびHostコレクションを含む依存コレクションもインストールされます：

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

5. インストールされている Ansible、Python、およびコレクションのバージョンを確認します。

```
ansible --version
python3 --version
ansible-galaxy collection list netapp_eseries.lustre
```

6. コントロールノードから各Lustreサーバーへのパスワードなし SSH接続を設定します。`<ip_or_hostname>`をサーバーの管理IPアドレスまたはホスト名に置き換えます。

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

各Lustreサーバーについて、同じ手順を繰り返します `ssh-copy-id`。

> ⓘ EシリーズアレイへのパスワードレスSSHを設定しないでください。Ansibleは、インベントリで定義された認証情報を使用して、SANtricity Web Services APIを介してアレイを管理します。

終了後の操作

"Ansibleインベントリをカスタマイズする"。

Lustre Ansibleインベントリをカスタマイズする

EF80 デプロイメントテンプレートの作業用コピーを作成し、サーバー、アレイ、ネットワーク、および Lustre ビルディングブロックに合わせてインベントリをカスタマイズします。

選択したインベントリは、それぞれのファブリックとプラットフォームのディレクトリに保存してください。

共有 `playbooks`ディレクトリと `passwords.yml`ファイルを `building_blocks`のルートに保持してください。

デプロイメントテンプレートを選択してください

サイトのネットワーク ファブリックに一致するテンプレートから開始します：

- **InfiniBand**： "IB_H2_IB_DAC_A2_EF80 / lustre_building_block_cluster_1"
- **RoCE**： "ROCE_H2_ROCE_DAC_A2_EF80 / lustre_building_block_cluster_1"

サイト固有の設定を構成する

インベントリにサイト固有の値を設定し、ローカルリポジトリとudevルール設定も併せて設定してください。表中の行参照は、 `ansible-lustre` `release/1.0.0`タグ内のRoCE EF80テンプレートファイルを指しています。InfiniBandテンプレートは、表に記載されているLNetインターフェース変数を除いて、同じ変数名と構造を使用します。

フィールド	**Ansible**変数	テンプレートファイル(`release/1.0.0`)	**Scope**	**Notes**
ファイルシステム名	`eseries_lustre_filesystem_mgt.format_options.fsname`、`eseries_lustre_filesystem_mdt.format_options.fsname`、`eseries_lustre_filesystem_ost.format_options.fsname`	"ha_cluster.yml#L76"、"#L90"、"#L110"	クラスター全体	MGT、MDT、およびOSTマップ用の `format_options`の下にネストされています。最大8文字。3箇所すべてで同じ値を使用してください。最上位の `fsname`キーは効果がありません。
ターゲット登録用のMGS NID	`eseries_lustre_filesystem_mdt.format_options.mgsnode`, `eseries_lustre_filesystem_ost.format_options.mgsnode`	"ha_cluster.yml#L98"、"#L118"	クラスター全体	MDTとOSTの下にネストされています `format_options`。HAペアの両方のノードからフロントエンドのLNet NIDを含めてください。優先MGS所有者として設定されているノードのNIDを最初にリストし、次にパートナーノードのNIDをリストします。
Pacemakerクラスター名	`eseries_lustre_pacemaker_cib_properties_overrides.cluster_properties.cluster-name`	"ha_cluster.yml#L228"	クラスター全体	Pacemaker CIBプロパティのオーバーライドの下にネストされています。HAクラスタの一意の名前。

フィールド	Ansible変数	テンプレートファイル(release/1.0.0)	Scope	Notes
BMCフェンシングアドレス	eseries_lustre_pacemaker_fencing_agents.fence_redfish.ip	"ha_cluster.yml#L211"、"#L214"、"#L217"、"#L220"	Lustreサーバーごとに繰り返す	各サーバーノードのRedfish BMC IPアドレス。
アラートメールの受信者	eseries_lustre_alert_email_list	"ha_cluster.yml#L233"	クラスター全体	HAリソースおよび障害通知の受信者。
メールドメイン	eseries_lustre_mail_service_overrides.conf.mydomain	"ha_cluster.yml#L240"	クラスター全体	アラートメールの送信に使用するPostfixドメイン。
NTP時刻ソース	eseries_lustre_time_sync_overrides.server_pools	"ha_cluster.yml#L250"	クラスター全体	クラスタノードの時刻ソース。NTPプール、または1つ以上の個別のNTPサーバーを指定します。必要な `pool`または `server`ディレクティブと `iburst`などのオプションを含めます。
インターフェース PCI-to-name udev マップ	eseries_ip_udev_rules	"ha_cluster.yml#L172"	HCAスロットレイアウトごとに繰り返す	各PCIスロットを永続的なインターフェイス名（L182の値）にマッピングします。 `lspci -nnvmm`を使用して収集します。
ローカルパッケージリポジトリ（オプション）	eseries_common_yum_repos	"ha_cluster.yml#L256"、"#L261"	クラスター全体	エアギャップインストールの場合、ブロックのコメントを解除してリポジトリ `url`を設定します（L264）。
サーバー管理IP	ansible_host	"host_vars/lustre_01.yml#L10"	Lustreサーバーごとに繰り返す	サーバーごとに1つ `host_vars/lustre_NN.yml`のファイル。
NVMe-oFバックエンドインターフェイス	eseries_nvme_roce_interfaces（RoCE）/ eseries_nvme_ib_interfaces（InfiniBand）	"RoCE host_vars/lustre_01.yml#L21"、"InfiniBand host_vars/lustre_01.yml#L17"	Lustreサーバーごとに繰り返す	NVMe-oFストレージトラフィックに使用されるバックエンドインターフェイスアドレス（Eシリーズアレイ向け）。
LNetクラスタインターフェイス	eseries_roce_interfaces （RoCE） / eseries_ipoib_interfaces （InfiniBand）	"RoCE host_vars/lustre_01.yml#L47"、"InfiniBand host_vars/lustre_01.yml#L38"	Lustreサーバーごとに繰り返す	フロントエンド LNet インターフェイス アドレス（L56 の RoCE 値、L47 の InfiniBand 値）。

フィールド	Ansible変数	テンプレートファイル(release/1.0.0)	Scope	Notes
CorosyncノードのIPアドレス	eseries_lustre_corosync_node_ips	"host_vars/lustre_01.yml#L58"	Lustreサーバーごとに繰り返す	HAクラスタ内のすべてのノードIPアドレスを一覧表示します（L62の値）。
Lustre ターゲットサービス NID	lustre_target_service_nodes	"host_vars/lustre_01.yml#L64"	Lustreサーバーごとに繰り返す	HA ペアの両方のノードからフロントエンド LNet NID（(@o2ib）を含めて、フェイルオーバー後もターゲット サービスにアクセスできるようにします（L67 の値）。
アレイ管理IP	ansible_host	"host_vars/netapp_01.yml#L13"	Eシリーズアレイごとに繰り返す	コントローラー1つの管理IPアドレス。API URLはこのIPアドレスから取得されます。
構成要素グループへの参加	mgs / mds_NN / oss_NN	"lustre_inventory.yml"	ビルディングブロックごと	各インベントリホスト名は、対応する `host_vars`ファイルのベース名と一致する必要があります。例えば、ホスト `lustre_01`は `host_vars/lustre_01.yml`を使用し、 `netapp_01`は `host_vars/netapp_01.yml`を使用します。MGSグループは最初の構成要素にのみ含めてください。基本構成要素および各MDT+OST拡張構成要素にはMDSグループを含め、OSTのみの拡張構成要素からはMDSグループを除外してください。既存のMGSに対して拡張ターゲットを登録するには `mgsnode`を使用してください。

各ホストおよびビルディングブロックごとに設定を繰り返してください。Lustreサーバーごとに `host_vars/lustre_NN.yml`ファイルを1つ、アレイごとに `host_vars/netapp_NN.yml`ファイルを1つ作成し、追加のビルディングブロックごとにインベントリグループを繰り返します。

デプロイメント プレイブックは、共有 `building_blocks/passwords.yml`ファイルからアレイ、Pacemaker、およびBMCの認証情報を読み込みます。デプロイ前にこのファイルにデータを入力してAnsible Vaultで暗号化するか、 `--extra-vars`を使用して実行時に値を安全に上書きしてください。平文の認証情報をソース管理にコミットしないでください。

デプロイメントテンプレートにおけるIPアドレス、ホスト名、インターフェース名、ボリュームサイズなどは、その例です。プレイブックを実行する前に、環境に合わせてすべてのインベントリファイルを変更してください。

終了後の操作

"Lustre HAクラスタとクライアントをデプロイする"。

Lustre HAクラスタとクライアントをデプロイする

サーバーオペレーティングシステムを準備し、NVIDIA DOCA-OFEDをインストールし、Lustre HAクラスタをデプロイし、NetApp Ansibleプレイブックを使用してLustreクライアントを設定します。

始める前に、"Lustre Ansibleインベントリをカスタマイズする"に記載されているインベントリを完成させて保管してください。

Lustre HA クラスターをデプロイする

手順

1. 作業コピーの `building_blocks/playbooks`ディレクトリから、Lustreサーバー上のオペレーティングシステムを準備します。 `<inventory_path>`をカスタマイズしたインベントリディレクトリへのパスに置き換えます。

```
ansible-playbook -i <inventory_path>/lustre_inventory.yml
deploy_lustre_os/deploy_lustre_os_playbook.yml
```

2. 各 Lustre サーバーに NVIDIA DOCA-OFED をインストールします。前の手順でインストールしたカーネルに対して、カーネルモジュールをビルドします。カーネルの手順については、"NVIDIA DOCA-Hostのインストールとアップグレード" を参照してください。以下の例は、RHEL または Rocky Linux への DOCA-OFED のインストール方法を示しています。

   a. DOCAホストリポジトリパッケージをインストールし、パッケージキャッシュを更新してください。 `<doca_host_package>`をお使いのオペレーティングシステムとアーキテクチャに対応したDOCAホストパッケージに置き換えてください。

```
dnf install -y wget tar
wget https://www.mellanox.com/downloads/DOCA/<doca_host_package>.rpm
rpm -i <doca_host_package>.rpm
dnf clean all
dnf makecache
```

   b. 実行中のカーネル向けにDOCAカーネルモジュールをビルドします。このスクリプトは、作成したパッケージのパスを出力します。

```
dnf install -y doca-extra
/opt/mellanox/doca/tools/doca-kernel-support
```

   c. スクリプトが作成したカーネルモジュールリポジトリパッケージをインストールし、パッケージキャッシュを更新します。 `<doca_kernel_repo>`を前の手順のパスに置き換えます：

```
rpm -Uvh <doca_kernel_repo>.rpm
dnf makecache
```

d. DOCA-OFEDのユーザー空間、カーネル、およびNVMeパッケージをインストールします。`<kernel_version>`を実行中のカーネルバージョンに置き換えます：

```
dnf install -y doca-ofed-userspace
dnf install -y --disablerepo=doca doca-kernel-<kernel_version>
dnf install -y kmod-mlnx-nvme
```

3. メインのデプロイメントプレイブックを実行します。

```
ansible-playbook -i <inventory_path>/lustre_inventory.yml
lustre_playbook.yml
```

デプロイメントの規模に応じてAnsibleが並行して設定するホスト数を制御するには、`--forks`を調整します。たとえば、大規模なクラスタの場合は`--forks 20`を追加します。

プレイブックは、Eシリーズのボリュームグループまたは DDP プールとボリュームをプロビジョニングし、NVMe-oF ホスト接続を設定し、Lustre ターゲットのフォーマットとマウントを行い、LNet を設定し、パフォーマンスチューニングを適用し、Pacemaker HA リソースを設定します。

1つの構成要素からなるデプロイメントは、2ノードクラスタのPacemakerクラスタを形成します。フェイルオーバーが発生した場合、ノードAには追加の投票権が与えられます。2ノードクラスタで定足数を達成するには、qdeviceの設定を検討してください。デプロイメントに複数の構成要素が含まれる場合、各構成要素は、文書化されたクラスタの上限まで、より大きなPacemakerクラスタに2ノードのサーバーペアを提供します。フェンスとクォーラムの構成を"HA管理ユーザーガイド"で確認し、ノード障害発生時にクラスタがターゲットを安全に復旧できるようにしてください。

Lustreクライアントをデプロイする

ファイルシステムへのアクセスが必要なシステムであれば、クライアント展開テンプレートをカスタマイズし、クライアントプレイブックを実行することで、Lustreクライアントをあらゆるシステムに展開できます。

手順

1. 互換性のあるクライアントオペレーティングシステムとカーネルを"Lustre サポートマトリックス"から選択してください。そのカーネル用のLustreクライアントパッケージが未インストールの場合は、"netapp-lustre"からビルドしてインストールしてください。
2. `clients`テンプレートディレクトリ（共有`passwords.yml`ファイルを含む）の作業コピーを作成し、ファブリックに合ったテンプレートを選択してください。
   - **InfiniBand**： "IB_lustre_cluster_clients"
   - **RoCE**： "ROCE_lustre_cluster_clients"

3. クライアント用に `lustre_client_inventory.yml`、 `host_vars`、および `group_vars`をカスタマイズします。表中の行参照は、 `ansible-lustre`release/1.0.0`タグ内のRoCEクライアントテンプレートファイルを指します。InfiniBandクライアントテンプレートは、特に明記されていない限り、同じ変数を使用します。

フィールド	**Ansible**変数	テンプレートファイル(`release/1.0.0`)	**Scope**	**Notes**
クライアントホスト	`lustre_clients`	"lustre_client_inventory.yml#L4"	クライアントごと	各クライアントのホスト名を一覧表示してください。
クライアント SSH ユーザーと become パスワード	`ssh_client_user` / `ssh_client_become_pass`	"passwords.yml#L2"	クラスター全体	SSH ユーザーが `root`でない場合にのみ、become パスワードを設定してください。
LNetインターフェース	`eseries_lustre_lnet`	"group_vars/all.yml#L7"	クラスター全体	クライアント側のLNetインターフェース名（L13の値）。
MGS NIDsをマウント	`lustre_client_mounts.mgsnode`	"group_vars/all.yml#L17"	クラスター全体	ファイルシステムをマウントするために使用されるMGS NID（L20の値）。
ファイルシステム名	`lustre_client_mounts.fsname`	"group_vars/all.yml#L21"	クラスター全体	クラスターと一致する必要があります `fsname`。
マウント ポイント	`lustre_client_mounts.mount_point`	"group_vars/all.yml#L22"	クラスター全体	クライアントのマウントディレクトリ。
クライアント管理IP	`ansible_host`	"host_vars/client_01.yml#L4"	クライアントごとに繰り返す	クライアントごとに1つ`host_vars/client_NN.yml`のファイル。
ストレージファブリック インターフェース	`eseries_roce_interfaces`	"host_vars/client_01.yml#L10"	クライアントごとに繰り返す	フロントエンドインターフェースのアドレス（L15の値）。InfiniBand テンプレートはここで`eseries_ipoib_interfaces`を使用します。

各クライアントの設定を繰り返してください。クライアントごとに `host_vars/client_NN.yml`ファイルを1つ作成し、対応するホストを `lustre_client_inventory.yml`に追加してください。共有`clients/passwords.yml`ファイルに必要な情報を入力し、クライアントプレイブックを実行する前にAnsible Vaultで暗号化してください。

4. クライアントテンプレートディレクトリからクライアントプレイブックを実行します。

```
ansible-playbook -i lustre_client_inventory.yml
lustre_client_playbook.yml
```

クライアントプレイブックは、クライアントのネットワークインターフェースとLNetを設定し、Lustreファイルシステム用の永続的なsystemdマウントユニットを作成できます。

終了後の操作

"デプロイメントを確認する"。

Lustreデプロイメントの検証と拡張

本番環境で使用する前に、クラスターの状態、Lustreマウント、ファイルシステムの容量を確認し、必要に応じて同じインベントリを使用して構成要素を追加してください。

デプロイメントを確認する

手順

1. Lustreサーバーから、すべてのPacemakerリソースが想定されるノードで起動していることを確認します。

```
pcs status
```

2. Lustreクライアントから、Lustreファイルシステムがマウントされていることを確認します。

```
mount -t lustre
```

3. Lustreクライアントから、ファイルシステムの容量と、MDTおよびOSTターゲットが表示されていることを確認します。

```
lfs df -h
```

IOR、mdtest、fio を使用したパフォーマンス検証については、"サイズガイド"を参照してください。

制御されたターゲット移行または HA フェイルオーバー テストについては、"HA管理ユーザーガイド"の手順に従ってください。

ファイルシステムを拡張する

容量やメタデータのパフォーマンスを向上させるには、既存のインベントリにビルディングブロックを追加して拡張します。新しいビルディングブロック用に別のインベントリを作成しないでください。プレイブックは、固有のMDTおよびOSTインデックスを自動的に割り当て、既存のMGSに対して新しいターゲットを登録します。

手順

1. 新しいビルディングブロックホスト、アレイ、リソースグループ（MDT+OSTまたはOSTのみ）を同じインベントリに追加し、 `host_vars`および `group_vars`ファイルは元のデプロイで使用したものを使用します。
2. 更新されたインベントリに対して、メインのデプロイメントプレイブックを再実行します。

スケーリングルールについては"ソリューションアーキテクチャ"を、各構成要素タイプをいつ追加するかについてのガイダンスは"サイズガイド"を参照してください。

Lustre と NetApp Eシリーズストレージ - 関連リソース

Lustre と NetApp Eシリーズストレージのデプロイメントをプランまたは拡張する際は、関連するドキュメント、ツール、ソフトウェアについて、これらのリンクを使用してください。サイジングとデプロイの手順については、"サイズガイド" および "解決策をデプロイする" を参照してください。

NetAppのリソース

- "NetApp Interoperability Matrix Tool"。**E-Series Lustre** を検索します。
- "NetApp EFシリーズ オールフラッシュ アレイ データシート"
- "NetApp SANtricity System Manager ドキュメント"
- "NetApp ansible-lustreコレクション"。インベントリフィールドのガイダンスについては、"Lustre Ansible インベントリをカスタマイズする"を参照してください。
- "NetApp Lustre ソースリポジトリ"。NetAppは、Rocky Linux 9.8およびRed Hat Enterprise Linux 9.8用のLustreサーバーRPMパッケージ（ビルド済み）を提供します。NetAppソースから、クライアントオペレーティングシステムとカーネル用のLustreクライアントパッケージをビルドします。

Lustreコミュニティ

- "Lustreファイルシステム"
- "Lustre サポートマトリックス"クライアントオペレーティングシステムおよびカーネル向け
- "Lustre Wiki：Lustreチューニング"

関連する NetApp AI インフラストラクチャソリューション

- "Eシリーズストレージを搭載したNetApp上のBeeGFS"
- "NetApp Eシリーズストレージを使用したIBM Spectrum Scaleのデプロイ"
- "NVIDIA DGX SuperPODとNetApp EFシリーズ"

著作権に関する情報

商標に関する情報