This document describes an HPC storage solution based on a HUAWEI OceanStor V3 converged storage system and the Lustre distributed file system. Specifically, it provides the best practices for the design, deployment, and optimization of a distributed file system. The document first introduces HPC and the Lustre file system in detail and then presents the verification results of deploying an HPC distributed file system based on OceanStor V3 storage systems, providing reference for HPC distributed file system deployment.

Wang Chao

Storage Solutions, IT, Huawei Enterprise Group

2015-03-17 V1.0
## Contents

1 About This Document ........................................................................................................ 4
  1.1 Overview ......................................................................................................................... 4
  1.2 Purpose ............................................................................................................................ 4
  1.3 Intended Audience .......................................................................................................... 5
  1.4 Customer Benefits ......................................................................................................... 5

2 Products and Technologies .............................................................................................. 6
  2.1 OceanStor V3 Storage Systems ....................................................................................... 6
    2.1.1 Product Positioning .................................................................................................... 6
    2.1.2 Advantages ................................................................................................................ 7
  2.2 High-Performance Computing (HPC) ............................................................................ 8
    2.2.1 System Structure ....................................................................................................... 8
    2.2.2 Hardware System ...................................................................................................... 9
    2.2.3 Software System ....................................................................................................... 10
    2.2.4 Interconnect ............................................................................................................. 11
  2.3 Distributed File Systems ............................................................................................... 12
    2.3.1 Lustre ......................................................................................................................... 13
    2.3.2 Intel Enterprise Edition for Lustre Software .............................................................. 16
  2.4 Introduction to Distributed File System Applications for HPC ....................................... 18
    2.4.1 HPC Distributed File System Analysis .................................................................... 19
    2.4.2 Application Simulation Methods .............................................................................. 23

3 HUAWEI HPC Storage Solution ..................................................................................... 27
  3.1 Solution Overview .......................................................................................................... 27
  3.2 Networking Mode .......................................................................................................... 28
  3.3 System Expansion ......................................................................................................... 29
  3.4 Reference Configuration .............................................................................................. 30

4 Reference Design of an HPC Distributed File System .................................................. 32
  4.1 Hardware Selection ....................................................................................................... 32
    4.1.1 Storage Hardware Selection for MGT and MDT ....................................................... 33
    4.1.2 Storage Hardware Selection for the OST ................................................................. 33
  4.2 Storage Space Requirements ....................................................................................... 33
    4.2.1 MGT Storage Space Requirements ....................................................................... 33
    4.2.2 MDT Storage Space Requirements ....................................................................... 34
4.2.3 OST Space Requirements ................................................................. 34
4.3 Choosing a File System Format ............................................................. 34
4.3.1 MDT ................................................................................................. 34
4.3.2 OST.................................................................................................... 35
4.4 File and File System Restrictions ............................................................ 35
4.5 Memory Requirement ........................................................................... 36
4.5.1 Client Memory Requirement .............................................................. 36
4.5.2 MDS Memory Requirement ................................................................. 36
4.5.3 OSS Memory Requirements ................................................................. 37
4.6 Network Requirement ........................................................................... 38
4.7 Security Configuration .......................................................................... 38

5 Reference Deployment of an HPC Distributed File System .......................... 40
5.1 Application Requirements .................................................................... 40
5.2 Deployment Plan ................................................................................... 40
5.2.1 Solution Architecture ....................................................................... 40
5.2.2 Solution Configuration ..................................................................... 41
5.3 Deployment Procedure ......................................................................... 44
5.3.1 Procedure ........................................................................................ 44
5.4 Testing and Acceptance ....................................................................... 45
5.5 Performance and Capacity Expansion ................................................. 45
5.6 Solution Benefits .................................................................................. 46

6 Reference Optimization of an HPC Distributed File System ....................... 47
6.1 Hardware Optimization ......................................................................... 47
6.2 Computing Node .................................................................................. 47
6.3 Storage Nodes ..................................................................................... 47
6.3.1 Load Balancing ................................................................................ 47
6.3.2 Multipath Optimization ................................................................. 48
6.4 Network Nodes .................................................................................... 49
6.4.1 IP Network Performance Optimization ........................................... 49
6.4.2 IB Network Optimization ................................................................. 49
6.4.3 Fibre Channel Network Optimization .............................................. 49
6.5 Application ........................................................................................... 49
6.5.1 Basic Performance Optimization ...................................................... 49
6.5.2 Small File Performance Optimization .............................................. 50
6.5.3 OSS and Client Nodes ...................................................................... 50

7 Appendix .................................................................................................. 51
7.1 Reference Documents .......................................................................... 51
7.2 Terminology ........................................................................................ 51
1. About This Document

1.1 Overview

The OceanStor V3 is a next-generation storage product series designed for enterprise-class applications. It incorporates a cloud-oriented storage operating system, a next-generation powerful hardware platform, and a suite of intelligent management software. The OceanStor V3 leads the industry in functionality, performance, efficiency, reliability, and ease of use. The OceanStor V3 adequately meets data storage needs of various applications such as OLTP/OLAP databases, file sharing, and cloud computing.

As a branch of scientific computing, HPC meets computing needs of computing-intensive applications that involve the processing of large volumes of data. HPC aims to improve the computation efficiency and accuracy. It has been widely used in a number of sectors, including oil exploration, earthquake prediction, aviation and astronomy, mechanical manufacturing, and animation rendering.

As informatization continues to be a popular trend and generates huge volumes of data, industries have sensed development opportunities where HPC can help enterprises extract more value from the data. To do so, HPC applications require large-I/O, high-bandwidth, and low-latency storage systems, posing a considerable challenge to enterprise IT deployment.

To help enterprises address this challenge, Huawei has launched an HPC distributed file system solution that combines Lustre distributed file system and the OceanStor V3. Built on a flexible architecture, the scale-out solution can be expanded by a maximum unit of 1 PB capacity and 10 GB/s performance. The superb-performance, high-reliability, and low-latency HPC storage system meets diversified storage performance needs.

This document starts with the application environment of an HPC storage system and focuses on how to use an OceanStor V3 to deploy an HPC distributed file system. It contains the following contents:

- Products and technologies related to HPC
- Analysis and simulation of primary HPC applications
- Reference design of an HPC distributed file system
- Reference deployment of an HPC distributed file system
- Reference optimization of an HPC distributed file system

1.2 Purpose

This document describes the market status and application models of HPC, as well as the design, deployment, and optimization of an HPC distributed file system, aiming to provide
reference for the IT system solutions of Huawei's partners and customers. The reference is
epected to lower the spending on the design, acceptance, delivery, and maintenance of HPC
storage solutions.

1.3 Intended Audience

This document is intended for Huawei's employees, partners, and customer IT engineers who
want to use a distributed file system solution in an HPC environment. The document
introduces HPC applications and provides reference design, deployment, and optimization of
an HPC distributed file system.

It is assumed that the readers are familiar with the following products and technologies:

- HPC basics
- Linux operating systems
- Storage systems

1.4 Customer Benefits

After reading this document, you will be able to understand:

- Implementation models of HPC applications and HPC distributed file systems
- How HPC is used and simulated in related industries
- Huawei's HPC solutions
- How to design an HPC distributed file system
- Huawei's reference architecture for HPC distributed file systems
- How to optimize HPC distributed file systems
2 Products and Technologies

2.1 OceanStor V3 Storage Systems

2.1.1 Product Positioning

The OceanStor V3 storage systems (OceanStor V3) are next-generation mid-range storage systems launched by Huawei in alignment with the status quo and future trends of storage technologies. The OceanStor V3 addresses the requirements of medium and large-sized businesses for massive data storage, high-speed data access, high availability, adequate utilization, energy efficiency, and ease of use.

Business development results in increasing volumes of data, which pose ever high demands on storage systems. Traditional storage systems fail to meet these demands and encounter the following bottlenecks: inflexible storage performance expansion, complex management of various devices, failure to utilize legacy devices, and increasing maintenance costs occupying a large part of Total Cost of Ownership (TCO). The HUAWEI OceanStor V3 is a proven storage system to solve those challenges.

Designed to be highly flexible and scalable, the OceanStor V3 boasts industry-leading hardware specifications, RAID 2.0+ underlying virtualization, and multiple data protection technologies. It is intended for use in a variety of scenarios, including Online Transaction Processing (OLTP)/Online Analytical Processing (OLAP) databases, high-performance computing (HPC), digital media, Internet operations, centralized storage, backup, disaster recovery (DR), and data migration.

Figure 2-1 OceanStor V3 storage systems
2.1.2 Advantages

Converged

- Convergence of SAN and NAS: SAN and NAS services are converged to provide elastic storage, simplify service deployment, improve storage resource utilization, and reduce total cost of ownership (TCO). Underlying storage resource pools provide both block service and file service and shorten storage resource access paths to ensure that the two services are equally efficient.

- Convergence of heterogeneous storage systems: The OceanStor V3 provides heterogeneous virtualization software that is compatible with third-party storage devices (different models or brands). In addition to ease of management and space utilization improvement, the OceanStor V3 supports data migration between heterogeneous storage systems, minimizing the risk and impact on customer business of data migration.

- Convergence of entry-level, mid-range, and high-end storage devices: Users can save data in storage devices of varying grades and brands depending on how valuable the data is. Third-party storage devices can utilize the backup and disaster recovery features of Huawei to implement local and remote data protection.

- Convergence of SSDs and HDDs: The OceanStor V3 implements deep optimization for HDDs and intelligently uses different cache algorithms and service life optimization algorithms in the case of combined HDDs and SSDs. In so doing, the OceanStor V3 unleashes the performance of SSDs, utilizes large storage capacity of HDDs, and manages data in different disk tiers based on data value.

- Convergence of primary storage and backup storage: The OceanStor V3 provides a snapshot-based backup function to notably lower data protection costs.

Reliable

- Failure rate reduction: The OceanStor V3 leverages RAID 2.0+ block virtualization technology to automatically and evenly distribute data to disks in a storage pool, preventing disk hot spots and lowering the overall failure rate of the storage system. Huawei's patented quake-proof and corrosion-proof technology further brings down the failure rate.

- Fast self-healing: The OceanStor V3 employs multi-level fault-tolerant and hot spare space design, enabling fast self-check and self-healing of faulty blocks. The disk reconstruction duration is shortened by 20 times, ensuring a high system reliability.

- Disaster recovery in seconds: The OceanStor V3 provides multi-timestamp caching technology to reduce latency and performance overhead, delivering a second-level RPO.

Efficient

- The OceanStor V3 has industry-leading specifications with up to eight controllers, 4 TB of cache (1 TB for dual-controller 6900 V3 and 4 TB for eight-controller 6900 V3), 160 host ports, and 8 PB of storage capacity.

- The OceanStor V3 is among the first storage products to be equipped with state-of-the-art Intel Ivy Bridge CPU, doubling performance to 2.2 million IOPS and 40 GB/s of bandwidth.

- The OceanStor V3 uses software-defined interface cards. The Huawei-developed chip in the SmartIO card supports three protocols: 10GE, 16 Gbit/s Fibre Channel, and FCoE.
Therefore, users can use any of these protocols through software settings, without the need to change the interface card.

2.2 High-Performance Computing (HPC)

Computational science, theoretical science, and experimental science are the three pillars that help the human kind better understand the world. Traditional scientific research methods usually combine theories and experiments. However, if no existing theory reference is available and experiments are overly expensive or unavailable, the only choice is to test the correctness of a theory through computational simulation.

As a branch of scientific computing, HPC meets computing needs of computing-intensive applications that involve the processing of large volumes of data. HPC aims to shorten the computing time and improve computing accuracy. It has been widely used in a number of sectors, including oil exploration, earthquake prediction, aviation and astronomy, mechanical manufacturing, and animation rendering.

The following sections introduce and analyze HPC in four aspects: system structure, hardware system, software system, and interconnect.

2.2.1 System Structure

Since the 1970s, modern high-performance computers have experienced decades of development, with their system structures changing from vector machine, symmetrical multi-processing (SMP), massively parallel processing (MPP), constellation, to cluster.

A cluster system uses an external interconnect (as opposed to the storage interconnect) to connect a group of computer nodes. Certain software is installed to enable synergy between these nodes to complete computing tasks. Featuring cost-effectiveness, flexibility, and scalability, and thanks to the rapid development of high-speed interconnects, cluster systems have become the mainstream architecture in the HPC sector. Over 80% the world's top 500 supercomputers employ cluster architecture.
2.2.2 Hardware System

Since 2000, cluster architecture has gradually become the norm in the HPC sector. In the meantime, x86 processors have rapidly expanded its share in the RISC processor market, thanks to their advantages in universal application and cost performance.

x86 processors account for the lion's share of 90% in the world's top 500 supercomputers. In particular, Intel processors with excellent performance have a dominant position.

Figure 2-3 Processors used in world's top 500 supercomputers
2.2.3 Software System

Operating system is the most underlying software system of a computer. In the early times, most high-performance computers used dedicated Unix operating systems. With the rise of cluster systems and constant development of x86 processors, Linux systems have gradually replaced Unix as the mainstream operating system in the HPC sector. Advantages of Linux include open source, high efficiency, high security, and multiple-user support. Support from a great deal of open-source and commercial software also contributes the rapid increasing market share of Linux. More than 90% of world’s top 500 cluster systems run Linux.

Figure 2-4 Operating systems employed by world’s top 500 supercomputers

Modern high-performance computers divide a single task into sub-tasks and execute them in parallel, thereby shortening the processing time, improving computational accuracy, or expanding the scale. To that end, a proper parallel programming model is needed. The programming model is closely associated with the parallel computer system structure. The mainstream parallel programming models include shared memory programming model and distributed memory programming model.

The mainstream programming model under the shared memory architecture (such as NUMA and SMP) is the shared memory programming model. This model has a single address space and a simple programming method, but is inferior in portability and scalability. A widely used shared memory programming model is OpenMP. OpenMP implements thread-level parallelizing by adding pragma statements in serial scripts to instruct parallelizing.

A programming model under distributed memory architecture (such as MMP and cluster) is called distributed memory programming. A common example is message passing programming model, which features multiple address spaces, complex programming, high portability, and high scalability. A widely used message passing model is Message Passing Interface (MPI) that is invoked in the form of a function library by applications. MPI implements process-level parallelizing. Since the launch of MPI-1.0 in February 1993, MPI has been developing for over two decades. The latest MPI-3.0 standard was released in September 2012. Based on this standard, many types of open-source and commercial MPI...
implementations have emerged, including the open-source MPICH, MVAPICH, and OpenMPI, as well the commercial Intel-MPI and Platform MPI.

**Figure 2-5** Evolution of the MPI standard

Current high-performance clusters have a multi-layer structure. Cluster systems boast the advantages of both shared memory and distributed memory architectures, resulting in the emergence of a hybrid model. Shared memory programming is used inside nodes, and distributed memory programming works between nodes. This model is called OpenMP+MPI. The advantage of this model lies in that it combines the thread-level fine-grained parallelizing and process-level rough-grained parallelizing. In most cases, OpenMP+MPI outperforms pure OpenMP or MPI.

### 2.2.4 Interconnect

High-performance clusters connect computing nodes over a network and use MPI or other software to pass messages for parallel computing. As modern high-performance clusters constantly increase in scale, networks have a growing impact on parallelizing efficiency.

The impact mainly lies in latency and bandwidth. MPI exchanges data between processes in the form of point-to-point or collective communication (broadcast, gather, scatter, forwarding, and reduction). A low exchange rate results in a small impact of the network on the parallelizing efficiency. Likewise, a high exchange frequency indicates a great impact. In this case, if a number of small data packets are exchanged between processes of different nodes, applications are sensitive to network latency. If the data packets are large, applications are sensitive to network bandwidth. Therefore, network-intensive applications require high-performance clusters to provide a low-latency, high-bandwidth interconnect.

InfiniBand (IB) is an open standard that builds such a type of network. The original purpose of IB is to convert the server bus into a network. IB leverages Remote Direct Memory Access (RDMA) to greatly lower network latency. A common type of IB technology is FDR, which delivers a mere 0.7 μs of latency during point-to-point communication and up to 56 Gbit/s bandwidth.
Since 2000, the market share of IB has been increasing along with the rapid development of cluster architecture. Of the world's top 500 high-performance computers, 40% use IB devices. In addition to IB, Gigabit Ethernet (GE) commands a market share of around 30%. Due to its universal applicability and cost-effectiveness, GE is well-suited for high-performance clusters that have modest network requirements. However, its market share has been declining over the years. 10GE has been enjoying rapid development since 2009 and now accounts for 10% market share in the top 500 supercomputers.

**Figure 2-6** Interconnect networks employed by world's top 500 supercomputers

Data in the preceding graph is quoted from www.top500.org.

### 2.3 Distributed File Systems

Computers manage and store data in the form of file system. In the era of information explosion, people have access to huge volumes of data that is still increasing exponentially. The traditional method of expanding storage capacity for a file system is to add disks. However, this simple method hardly satisfies user requirements for storage capacity, capacity growth speeds, data backup, and data security.

Distributed file system is an effective solution to the storage and management issues of massive data. It expands one file system at one location to multiple file systems at multiple locations. The nodes at these locations form a file system network. Each node can be located in a different place, and they communicate and transfer data over an interconnect. When using a distributed file system, users do not need to know where data is stored or obtained. Instead, they can manage and store all the data as if it were in a local file system. Distributed file systems are designed to work in client/server model. A typical network possibly contains multiple servers for access from multiple users.
The mainstream file systems include Lustre, Hadoop, MogileFS, FreeNAS, FastDFS, NFS, OpenAFS, MooseFS, pNFS, and GoogleFS. The following section describes Lustre.

### 2.3.1 Lustre

**Overview**

Lustre is a storage architecture for clusters. Its central component is the Lustre file system that runs on a Linux operating system, provides a POSIX-compliant Unix file system interface, and complies with GPL 2.0. IDC statistics show that Lustre is the most widely used file system in the HPC field. Of the 50 fastest supercomputer websites, 60% use Lustre.

**NOTE**

Due to Lustre's structural characteristics, every two devices serve as a backup for each other. Therefore, Lustre can be called a clustered file system. In terms of I/O processing model, Lustre implements parallel processing. So it can also be called a parallel file system. In addition, because I/Os in Lustre are usually distributed to different I/O nodes, it is also a distributed file system. These three names represent different perspectives, and there is no right or wrong between them.

In the cluster computer field, the improvement in data exchange between computer and disks cannot keep pace with the performance growth of micro-processors and memory, bottlenecking the performance of applications. The Lustre file system eliminates this bottleneck because it provides PB-level storage capacity and hundreds of TB/s of I/O throughput, making it able to serve tens of thousands of clients.

The Lustre file system employs client/server model. Its object storage features (separation of files on clients and disks, no impact of storage upgrade and change on client use) have largely reduced the storage device purchase cost and transformed the ways enterprises buy storage. The distributed and parallel working mode of Lustre avoids the need to buy expensive high-end storage systems to meet the performance requirement. Instead, customers can buy an entry-level storage system and dynamically expand I/O nodes to address growing application requirements. The customer can create a file system for each cluster application, or use a file system that has a unified global name. The customer can set file system properties such as stripe for each file, thereby reducing inter-cluster data copy times and simplifying cluster management. As the storage capacity of servers and clusters is consolidated, resource waste is minimized, facilitating dynamic performance increase.

The Lustre file system supports high-performance, low-latency interconnects, such as IB based on OpenFabrics Enterprise Distribution (OFED). Different RDMA networks can be bridged by Lustre routing, ensuring sufficient network bandwidth for Lustre.

Shared storage partitions of the Lustre file system support active-active failover of object storage targets (OSTs) so that an underlying error is transparent to applications. In addition, using multiple mount protection (MMP) technology offers complete protection against errors that may cause file system failure.

**Components**

A Lustre file system mainly contains the following components: Metadata Server (MDS), Metadata Target (MDT), Object Storage Target (OST), and clients. Figure 2-7 illustrates the relationships between these components.

MDS: The MDS server manages metadata information about the Lustre file system, including file name, directory, permission, and file structure. It makes metadata stored in one or more MDTs available to Lustre clients. There can be multiple MDSs, but only one of them is active. The other MDSs work in passive mode.

MDT: Each file system has one MDT. If only one MDS exists, the MDT can be a local disk of the MDS or a LUN in a remote storage system. One MDT can be mapped to two hosts for
access by multiple MDSs. However, only one MDS can access the MDT at a time, for the purpose of high availability.

OSS: The OSS provides file I/O service for clients. Clients obtain metadata information from the MDS and access file data on the OSS. The file data is finally stored on the OSTs that are connected to the OSS. Typically an OSS has 2 to 8 OSTs and a maximum of 16 TB of capacity.

OST: The OST stores file data as data objects on one or more OSSs. A file can be stored in one OST or across many OSTs. One OST can be mapped to two hosts for high OSS availability.

Lustre clients: Lustre clients are computational, visualization, or desktop nodes that run Lustre software and mount the Lustre file system. In a high-performance cluster, Lustre clients are usually computational, management, or login nodes.

**Figure 2-7** Lustre components in a basic cluster

![Lustre components in a basic cluster](image)

**File storage**

The Lustre file system uses file identifiers (FID) to identify files or objects. The MDT saves information about where file data is stored in the OST. The information is saved as an attribute of FID, called layout extended attribute (EA).

Assume that the MDT stores information about where a data file (not a directory or connector) is saved. If the MDT points to a single OST, the entire data file is stored on the OST. If it points to multiple OSTs, the file is striped across the OSTs in the RAID 0 pattern. As shown in Figure 2-8, a client must obtain layout EA information from the MDT before it can read or write a data file. The layout EA information shows which OSTs store the data.
The high performance of the Lustre file system is primarily attributed to its data segmentation or chunk striping implemented between OSTs using a round-robin algorithm. Users can set the stripe quantity, size, and target OST for each file. When a single file is accessed, striping can improve the file system performance to be higher than the bandwidth of a single OST. Striping allows parts of files to be stored on different OSTs, as shown in Figure 2-9.

For example, in a Lustre file system, file C has a larger stripe size than file A, so it can store more data in a single stripe. The stripe count of file A is 3, namely stored across three OSTs. The stripe counts of file B and file C are both 1. When the first I/O request is delivered, file A is striped and stored on OST1, OST2, and OST3. When the second I/O request is delivered, files A, B, and C are written concurrently. Based on a balancing algorithm, two stripes of file A are stored on OST1, file B is stored on OST2, and file C is stored on OST3. In nature, OST1, OST2, and OST3 store the same amount of data.

The preceding example shows that the file size in a Lustre file system is not restricted by the size of an OST, because files can be striped and stored on multiple OSTS (up to 200). When ldiskfs is used, the size of each OST is up to 16 TB, and the total maximum file storage space.
is 31.25 PB. When ZFS is used, the maximum size per OST is 256 PB, and the maximum total size is 8 EB.

**File Read/Write**

In a Lustre file system, a file is read/written in the following procedure:

1. A Lustre client sends a file open request to the MDT.
2. Upon receiving the request, the MDT searches for the corresponding layout EA of the file.
3. The MDT sends the FID containing the layout EA to the Lustre client.
4. The client finds the corresponding OSS and sends an I/O request.
5. The OSS finds the corresponding OST, reads/writes the file, and returns the result to the Lustre.

![Image of file reading/writing in Lustre](image)

**Figure 2-10 File reading/writing in Lustre**

### 2.3.2 Intel Enterprise Edition for Lustre Software

Intel® Enterprise Edition for Lustre (Intel EE for Lustre) software is dedicated to helping enterprises and large-scale organizations to improve the performance and scalability of parallel file systems. On top of open source software, Intel EE for Lustre software greatly simplifies installation, configuration, and monitoring. The software optimization enables large-sized applications and high-bandwidth storage systems to collaborate with Lustre for higher performance. Intel EE for Lustre software provides open interfaces that allow efficient integration with existing infrastructure. In addition, the open source nature of the software helps IT more conveniently utilize network and hardware improvements to expand storage systems in a data center.

The following figure shows Intel's customization of the Lustre software:
The key features of Intel EE for Lustre software are as follows:

Intuitive management interface based on web browser
- Simple yet powerful GUI and script-supported CLI
- Cluster configuration and management performed with simple mouse clicks
- Centralized definition and management of routine tasks

Quasi real-time file system monitoring
- Quasi real-time storage health and key performance indicators
- High-level system performance monitoring data

Enhanced fault detection tool
- Enhanced log file view for cluster storage
- Intelligent log display for fault isolation and analysis
- Configurable event notification messages

Open documents and APIs
- APIs for integration of Intel management architecture
- Storage plug-in architecture for detailed hardware reports

Compared with the open-source Lustre, Intel EE for Lustre software requires one more Lustre management server. The network topology is as follows:
2.4 Introduction to Distributed File System Applications for HPC

As data analysis and modeling exert an increasing influence on enterprise business and production, HPC applications start expanding into a number of industries. The following lists the mainstream HPC applications in different industries:

Table 2-1 Mainstream HPC applications

<table>
<thead>
<tr>
<th>Industry</th>
<th>Computational Task</th>
<th>Mainstream HPC Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing</td>
<td>• Fluid mechanics</td>
<td>• Ansys</td>
</tr>
<tr>
<td></td>
<td>• Finite element analysis</td>
<td>• Nastran</td>
</tr>
<tr>
<td></td>
<td>• Impact simulation</td>
<td>• Fluent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Abaqus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LS-Dyna</td>
</tr>
<tr>
<td>Oil and natural</td>
<td>• Energy reservoir simulation</td>
<td>• ECLIPSE</td>
</tr>
<tr>
<td>gas</td>
<td>• Earthquake and oil reservoir analysis and simulation</td>
<td>• VIP</td>
</tr>
<tr>
<td></td>
<td>• Oil reservoir data visual display</td>
<td>• DMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MEPO</td>
</tr>
<tr>
<td>Life science</td>
<td>• Pharmaceutical R&amp;D</td>
<td>• Accelrys</td>
</tr>
<tr>
<td></td>
<td>• Genetic research</td>
<td>• Schrodinger</td>
</tr>
<tr>
<td></td>
<td>• Medical data informatics</td>
<td>• Simulia</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SAS</td>
</tr>
</tbody>
</table>
## 2.4.1 HPC Distributed File System Analysis

### CAE Emulation

Computer Aided Engineering (CAE) uses computers to aide complex analysis engineering and structural mechanics performance, and to optimize structural performance. It is primarily applied to aviation, astronomy, automobile, shipping, mechanics, architecture, and electronics.

The CAE process is as follows: Implement geometry modeling and grid generation. > Specify load and boundary conditions and submit the product to servers for analysis. > Display results and assess product performance.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Scalability</th>
<th>Memory</th>
<th>Storage</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invisible Finite Element Analysis (IFEA)</td>
<td>ANSYS</td>
<td>General-purpose invisible finite element software</td>
<td>&lt; 16 CPUs</td>
<td>These applications require large memory, and 1 million node flexibilities require 1 to 10 GB of memory.</td>
<td>High I/O requirement</td>
<td>IB still has a distinct advantage over GE.</td>
</tr>
<tr>
<td></td>
<td>NASTRAN</td>
<td>Linear structural analysis software</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABAQUS</td>
<td>General-purpose invisible/evident finite element software</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evident Finite Element Analysis (EFEA)</td>
<td>LS-DYNA</td>
<td>The most outstanding evident analysis software</td>
<td>64+ CPUs</td>
<td>These applications require relatively small</td>
<td>High I/O requirement</td>
<td>IB is recommended for 32 processes or more.</td>
</tr>
<tr>
<td></td>
<td>PAM-CRA</td>
<td>Collision and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HUAWEI OceanStor V3 Converged Storage Systems — HPC Distributed File System Reference Architecture

### Table 2-3 Oil business requirement analysis

<table>
<thead>
<tr>
<th>Application Industry</th>
<th>Computing Node</th>
<th>Interconnect</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oil exploration</td>
<td>Multi-core/Multi-core/High-density server</td>
<td>10GE</td>
<td>Large storage capacity</td>
</tr>
<tr>
<td></td>
<td>GPU/GPGPU technology</td>
<td>IB</td>
<td>SSD acceleration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I/O-intensive</td>
</tr>
</tbody>
</table>

### Oil industry

In the oil industry, HPC mainly serves oil exploration operations, including outdoor operations, earthquake data processing, earthquake data explanation, and oil reservoir simulation. These operations require HPC to provide floating-point operations, multi-core scalability, large memory, and high bandwidth.

The workflow of oil business is as follows: Explore oil. > Locate the oil well. > Mine the oil. > Refine the oil. > Sell the oil.

### Animation and Multimedia

Animation and multimedia is another major application sector of HPC, where it is primarily used for large graphic rendering projects.

Business workflow: Abstract the original model. > Submit the task. > Wait in the queue. > Activate the task. > Start rendering. > Archive the rendering.
Table 2-4 Animation and multimedia requirement analysis

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Function</th>
<th>Computing Node</th>
<th>Storage</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D production software</td>
<td>Maya</td>
<td>3D image design, image modeling</td>
<td>Cluster rendering requires each node to deliver powerful image processing capabilities, large memory capacity, and high memory bandwidth.</td>
<td>Image and video data processing requires large storage capacity, high bandwidth, and high performance against bursts of demands.</td>
<td>Task activation requires high network bandwidth.</td>
</tr>
<tr>
<td></td>
<td>3Ds Max</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XSI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lightwave 3D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Renderer</td>
<td>Renderman</td>
<td>3D image rendering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mental Ray</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rendering management software</td>
<td>Enfuzion</td>
<td>Rendering task distribution, scheduling, and management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Qube</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Muster</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drqueue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Weather Forecast

Weather forecast is the science of predicting the changes and state of weather by importing meteorological data and boundary parameters into an equation built by using mathematical methods.

Workflow: Collect and pre-process meteorological data. > Start the digital weather forecast process. > Generate forecast results by combining the disciplines of digital weather forecasting, synoptic, and statistics.

Table 2-5 Weather forecasting requirements analysis

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Scalability</th>
<th>Computing</th>
<th>Storage</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather forecasting</td>
<td>MM5</td>
<td>The most widely used mid-scale model, which will transition to WRF</td>
<td>The software provides good scalability. Most modes have achieved parallelism.</td>
<td>Computing load is heavy. Each time the forecasting accuracy improves by 100%, the computing load increases exponentially.</td>
<td>Most programs in oceanography mode have high I/O performance requirements . Distributed I/O processing or distributed file system is usually</td>
<td>Extremely intensive communication poses stringent network performance requirements.</td>
</tr>
<tr>
<td></td>
<td>WRF</td>
<td>A new mid-scale model developed based on MM5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRAPES</td>
<td>New numerical forecasting system developed by China Meteorological Administration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Life Sciences**

Life sciences can be divided into three sectors:

- **Biology informatics**: uses HPC for biological gene sequencing, splicing, and comparison and provides genome information and related data systems to resolve major issues in biological, medical, and industrial fields.

- **Molecular dynamics simulation**: leverages HPC to simulate molecular dynamics on a large scale and uses the simulation results to analyze and verify protein changes on molecular and atomic levels.

- **New medicine R&D**: uses HPC to rapidly complete virtual selection of high-throughput medicines, shortening average R&D period by 1.5 years and saving RMB 100 million on investment.

**Table 2-6 Application requirement analysis for life sciences**

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Description</th>
<th>Scalability</th>
<th>Computing</th>
<th>Storage</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bioinformatics — DNA gene comparison</td>
<td>BLAST, FASTA</td>
<td>Sequence similarity searching software</td>
<td>Large memory capacity</td>
<td>High I/O requirement</td>
<td>As the applications do not require frequent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ClustalW</td>
<td>Multi-sequence</td>
<td>The applications mainly involve</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 2.4.2 Application Simulation Methods

#### Overview

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Description</th>
<th>Scalability</th>
<th>Computing</th>
<th>Storage</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pharmaceutical R&amp;D</strong></td>
<td>Censor, RepeatMasker</td>
<td>Repeated sequence testing software</td>
<td>fixed-point calculation.</td>
<td>Parallel computing tasks require a high concurrency storage system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PHYLIP, PALM</td>
<td>System evolution tree construction software</td>
<td></td>
<td>High memory requirement, 1 GB for each processor core</td>
<td></td>
<td>As data communication is not frequent, GE is sufficient.</td>
</tr>
<tr>
<td><strong>Molecular dynamics</strong></td>
<td>DOCK, AutoDock, FlexX</td>
<td>Semi-flexible docking program</td>
<td>Parallel computing tasks require a high scalability.</td>
<td>High I/O requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Discovery, Studio</td>
<td>Different scales of molecular docking</td>
<td></td>
<td>As point-to-point communication is frequent, IB is recommended.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ZDOCK, RDOCK</td>
<td>Firm docking, protein docking</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MORDOR</td>
<td>Flexible docking program</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pharmaceutical R&amp;D</strong></td>
<td>NAMD</td>
<td>Simulating parallel fluid dynamics code of macromolecular structures</td>
<td>Storage solution must provide a very high parallel acceleration ratio.</td>
<td>Modest</td>
<td>High I/O requirement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GROMACS</td>
<td>Molecular dynamics package for researching biological molecular systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHARM</td>
<td>Commercial software based on CHARMM force fields</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMBER</td>
<td>Commercial software based on AMBER force fields</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LAMMPS</td>
<td>Large-scale atomic/molecular massively parallel simulator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Before the deployment of an HPC application system or in a test environment, a test tool for HPC distributed file systems is used to simulate an abstracted workload model to assess whether the HPC distributed file system meets application requirements. For optimum test results of different applications, the parameters of the test tool can be properly adjusted.

The following table lists a few commonly used tools for HPC distributed file systems:

Table 2-7 File system test tools

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General-purpose file system testing tools</td>
<td>IOzone</td>
<td>IOzone is a file system benchmarking tool used to test the read/write performance of file systems in different operating systems.</td>
</tr>
<tr>
<td></td>
<td>bonnie++</td>
<td>Bonnie++ is a tool for testing Unix file system performance.</td>
</tr>
<tr>
<td>Dedicated computing environment testing tools</td>
<td>IOR</td>
<td>Parallel file system testing tool</td>
</tr>
<tr>
<td></td>
<td>mdtst</td>
<td>Metadata benchmarking tool</td>
</tr>
<tr>
<td></td>
<td>FDtree</td>
<td>FDtree is a tool used for testing file system metadata performance.</td>
</tr>
</tbody>
</table>

How to Use IOzone

IOzone is a file system benchmarking tool used to test the read/write performance of file systems in different operating systems. It supports the following testing modes:

Table 2-8 Testing modes of IOzone

<table>
<thead>
<tr>
<th>Read/Write Mode</th>
<th>Description</th>
<th>Value of Parameter -I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Tests the performance of writing a new file.</td>
<td>0</td>
</tr>
<tr>
<td>Re-write</td>
<td>Tests the performance of writing a file that has been written.</td>
<td>0</td>
</tr>
<tr>
<td>Read</td>
<td>Tests the performance of reading a file.</td>
<td>1</td>
</tr>
<tr>
<td>Re-Read</td>
<td>Tests the performance of reading a file that has been recently read.</td>
<td>1</td>
</tr>
<tr>
<td>Random Read</td>
<td>Tests the performance of reading the random offset of a file.</td>
<td>2</td>
</tr>
<tr>
<td>Random Write</td>
<td>Tests the performance of writing the random offset of a file.</td>
<td>2</td>
</tr>
<tr>
<td>Random Mix</td>
<td>Tests the performance of reading and writing the random offset of a file.</td>
<td>8</td>
</tr>
<tr>
<td>Backwards Read</td>
<td>Tests the performance of reading a file backwards.</td>
<td>3</td>
</tr>
</tbody>
</table>
## Read/Write Mode

<table>
<thead>
<tr>
<th>Description</th>
<th>Value of Parameter -I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests the performance of writing and overwriting a specific part of a file.</td>
<td>4</td>
</tr>
<tr>
<td>Tests the performance of invoking function fwrite() to write a file.</td>
<td>6</td>
</tr>
<tr>
<td>Tests the performance of invoking function fwrite() to re-write a file.</td>
<td>6</td>
</tr>
<tr>
<td>Tests the performance of invoking function fread() to re-write a file.</td>
<td>7</td>
</tr>
<tr>
<td>Tests the performance of reading a file that has been recently read.</td>
<td>7</td>
</tr>
</tbody>
</table>

### Procedure for installing IOzone:

- Download the IOzone installation package.
- Run `rpm -ivh iozone-3.315-1.e15.rf.x86_64.rpm` to install IOzone.

The following is an example of IOzone command:

```
iozone -t 24 -F 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 -i 0 -o -r 4m -s 240G -w
```

### IOzone commands involve the following parameters:

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOzone</td>
<td><code>-t</code></td>
<td>Indicates the number of threads in multi-thread mode. In the preceding example, there are 24 threads.</td>
</tr>
<tr>
<td></td>
<td><code>-F</code></td>
<td>Indicates the name of the temporary files that correspond to the threads, namely the name of the files generated during writing.</td>
</tr>
<tr>
<td></td>
<td><code>-i</code></td>
<td>Indicates the type of testing. Possible values are listed in Table 2-8.</td>
</tr>
<tr>
<td></td>
<td><code>-o</code></td>
<td>Indicates synchronous writing to a disk.</td>
</tr>
<tr>
<td></td>
<td><code>-r</code></td>
<td>Indicates the size of each block for testing.</td>
</tr>
<tr>
<td></td>
<td><code>-s</code></td>
<td>Indicates the size of each file for testing.</td>
</tr>
<tr>
<td></td>
<td><code>-w</code></td>
<td>Saves files after the test is completed.</td>
</tr>
</tbody>
</table>

**NOTE**

The size of the files for testing must be larger than the server memory (two times recommended). Otherwise, Linux will cache data reading and writing, affecting the accuracy of the test results.
How to Use IOR

IOR is a parallel file system testing tool and supports file systems that use POSIX, MPIIO, or HDF5 interface. IOR can work with MPI commands to test the Lustre file system.

Procedure for installing IOR:

- Run `tar xvzf IOR-2.10.3.tgz` to decompress the downloaded IOR software package.
- Run `cd IOR/src/C` to go to the decompressed directory.
- Run `make`.
- Run `./IOR`.

The IOR testing command is `mpiexec`. The following is an example:

```
mpiexec -H 10.10.10.11,10.10.10.12,10.10.10.13 -np 24 IOR -b 100g -t 10m -w -g -F -e -k -o /lustrefs/ior
```

The following table describes the parameters in the command:

<table>
<thead>
<tr>
<th>Table 2-10 IOR command parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
</tr>
<tr>
<td>mpiexec</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
3 HUAWEI HPC Storage Solution

3.1 Solution Overview

The HUAWEI HPC Storage Solution is based on Lustre storage architecture and incorporates Intel EE for Lustre software. Other components include Huawei-developed server series and cluster management software.

Figure 3-1 HUAWEI HPC Storage Solution

System Structure

The HUAWEI HPC Storage Solution employs the mainstream cluster+distributed model and Lustre architecture. One MDS and one OSS comprise a cluster pair that provides active-active high availability. In normal cases, MDS and OSS share the load. If one server fails, the other continues to provide services. Performance scalability is not limited to one pair of servers, and multiple servers can be allocated for one task for distributed processing and maximum back-end performance.

Hardware System

The hardware system is the basis for a cluster system. The Huawei high-performance cluster system employs Huawei-developed servers, storage devices, switches, and infrastructure, meeting customer requirements in an end-to-end way. In addition, Huawei works closely with Intel, Mellanox, and NVIDIA to build an industry-leading hardware system.
Software System

In distributed storage management, Huawei collaborates with Intel to provide a stable, efficient HPC storage environment for customers.

To facilitate HPC management, Huawei provides its self-developed cluster management software that provides HPC management, monitoring, and other services required by the HPC cluster.

Interconnect

The HUAWEI HPC Storage Solution recommends IB networking for low-latency, high-bandwidth communication within an HPC cluster. For connection between I/O nodes and the storage system, 16 Gbit/s Fibre Channel is recommended for its high bandwidth and utilization.

The network diagram of the HUAWEI HPC Storage Solution is as follows:

**Figure 3-2** Network diagram of the HUAWEI HPC Storage Solution

![Network diagram of the HUAWEI HPC Storage Solution](image)

### 3.2 Networking Mode

The HUAWEI HPC Storage Solution provides a basic networking mode and flexible extensions for up to 100 Gbit/s throughput and PB-level storage capacity. The network components of the HUAWEI HPC Storage Solution are as follows:

- Computer node/Lustre client
- MDS: metadata server
- OSS pair: I/O nodes
- Management server
- OceanStor V3: File storage array that provides HPC distributed file systems
- IP switch: management network interconnect switch
- IB switch: computing network interconnect switch

Figure 3-3 Network diagram of the HUAWEI HPC Storage Solution

NOTE
[1] IB switches are used for interconnecting computing nodes in the scenario of HPC expansion. If only three or fewer computing nodes are used, the IB switch card embedded on the computing node blade server can be used for node interconnection.

3.3 System Expansion

The HUAWEI HPC distributed file system can scale up its performance and capacity by adding disks, storage nodes (engines), and I/O nodes (OSS). The post-expansion performance can be estimated as follows:

- If a disk enclosure has vacant disk slots, adding disks improves performance by: Number of disks x Read/Write performance of each disk.
- The read/write bandwidth increases by 10 GB/s every time a storage node is added, until the number of I/O nodes reaches the upper limit.
- If the disks, front-end links, and back-end links have no bottleneck, adding storage nodes improves performance by: Number of storage nodes x Performance of each node.
- If each OSS has one IB interface card and two 16 Gbit/s Fibre Channel interface cards, adding one pair of OSS nodes improves performance by: \( 2 \times \min \{ \text{Read/Write bandwidth of IB interface card}, 2 \times \text{Fibre Channel interface card bandwidth} \} \).

The file system performance and capacity can reach the optimum by combining the preceding expansion granularities. The following shows the expansion workflow:
3.4 Reference Configuration

The following table lists the reference configuration for the HPC storage solution based on the OceanStor V3:

### Table 3-1  HUAWEI HPC Storage Solution configuration

<table>
<thead>
<tr>
<th></th>
<th>Device Model</th>
<th>10 GB/s Configuration[1]</th>
<th>20 GB/s Configuration</th>
<th>30 GB/s Configuration</th>
<th>40 GB/s Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system size</td>
<td>/</td>
<td>1 PB</td>
<td>1.2 PB</td>
<td>1.8 PB</td>
<td>2.5 PB</td>
</tr>
<tr>
<td>Intel EE for Lustre</td>
<td>/</td>
<td>2.1.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of OSS nodes</td>
<td>RH2288 V3 (E2600 V3 CPU, 128 GB memory)</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Number of OSS storage arrays</td>
<td>OceanStor 5800 V3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Number of OSS disks</td>
<td>4 TB 7.2k rpm NL-SAS disks</td>
<td>250</td>
<td>500</td>
<td>750</td>
<td>1000</td>
</tr>
<tr>
<td>Number of front-end interface modules</td>
<td>16 Gbit/s FC HBA (dual ports)</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>14</td>
</tr>
</tbody>
</table>
### HUAWEI OceanStor V3 Converged Storage Systems — HPC Distributed File System Reference Architecture

<table>
<thead>
<tr>
<th>Device Model</th>
<th>10 GB/s Configuration[1]</th>
<th>20 GB/s Configuration</th>
<th>30 GB/s Configuration</th>
<th>40 GB/s Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>IB</td>
<td>56 Gbit/s FDR IB HBA</td>
<td>4</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>MDS node</td>
<td>RH2288 V3 (E2600 V3 CPU, 128 GB memory)</td>
<td>/</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MDS storage array</td>
<td>OceanStor 5300 V3</td>
<td>/</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Number of MDS disks</td>
<td>600 GB 15k rpm SAS disks</td>
<td>/</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Number of front-end interface modules</td>
<td>16 Gbit/s FC HBA</td>
<td>/</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IB</td>
<td>56 Gbit/s FDR IB HCA</td>
<td>/</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

1. Comprehensive read/write bandwidth: Comprehensive bandwidth for write, rewrite, read, and re-read in a workload test.

2. Each OSS node provides 6 GB/s I/O processing. If the required bandwidth is higher than 10 GB/s, OSS nodes must be expanded by pair.

3. If the required bandwidth is between 5 GB/s and 10 GB/s, you can add disks to improve the storage capacity of the HPC distributed file system. If the required bandwidth is higher than 10 GB/s, you need to add a new OceanStor 5800 V3 and interface modules every time the requirement increases by 10 GB/s.
4 Reference Design of an HPC Distributed File System

4.1 Hardware Selection

The following table lists the required hardware specifications of Lustre components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Storage Requirement</th>
<th>Recommended Hardware Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDSs</td>
<td>1% to 2% of the file system capacity</td>
<td>Sufficient CPU performance, sufficient memory, fast storage</td>
</tr>
<tr>
<td>OSSs</td>
<td>1 to 16 TB of storage capacity allocated to each OST</td>
<td>High bus bandwidth, balanced storage configuration for different OSSs</td>
</tr>
<tr>
<td></td>
<td>1 to 8 OSTs allocated to each OSS</td>
<td></td>
</tr>
<tr>
<td>Clients</td>
<td>None</td>
<td>Low-latency, high-bandwidth network</td>
</tr>
</tbody>
</table>

The Lustre file system can use different types of block storage devices, including single disk, software RAID, hardware RAID, and logical volume manager. In the Lustre file system, block devices only need to be mapped to the MDS and OSS nodes, without the need for clients to directly access the nodes. Block devices are accessed by one or two server nodes. Therefore, instead of using expensive switches, servers and the storage array simply use the simplest and most suitable point-to-point direct connection.

As for the production environment, an independent storage system is recommended for the MGS so that it can be expanded to multiple file systems. However, MDS nodes must be allowed to use the same storage devices as the MGS for the purpose of high availability. It is recommended that the Lustre file system is tested and maintained on a 64-bit CPU server. Using 64-bit clients avoids the restrictions on RAM size (not larger than 4 GB) and file size (not larger than 16 TB). If 32-bit clients are used for backup, certain restrictions are imposed on the files to be backed up.

The Lustre file system uses log file system technology on MDT and OST. As for MDT, 20% of performance resource is used to write logs to independent devices. Therefore, MDS consumes a part of CPU resource, and CPUs with at least four cores are recommended. In the case of multiple clients, it is necessary to enhance CPUs.
4.1.1 Storage Hardware Selection for MGT and MDT

An MGT requires only 100 MB of storage space even in a large Lustre file system. Data on an MGT is accessed only when it is mounted on a server or client. Therefore, it has a negligible impact on disk performance. However, the data on an MGT is critical to file system access. Therefore, it requires reliable storage, and RAID 1 is recommended.

The access model of MDS storage is similar to the database access model, both involving numerous queries, reads, and writes of small-sized data. The MDS does not require high throughput but a short seek time. For example, high-speed SAS disks or SSDs provide fast seek. For maximum performance, RAID 1 is recommended for the MDT, consisting two disks from different controllers. If a large MDT is required, it is recommended that multiple RAID 1 groups be created first and then implement RAID 0. In doing so, it is unlikely that the two disks of the same RAID 1 group fail at the same time. If RAID 0 is implemented before RAID 1, the rate of dual-disk failure within the same group is 50%, which would result data loss of the entire MDT. The first disk failure closes the mirror, and there is a 50% probability that the second failure will close the rest of the mirror and cause data loss.

4.1.2 Storage Hardware Selection for the OST

OSS storage employs the streaming I/O model, reliant on the access mode of the application. One OSS can manage multiple OSTs. Each volume has I/O load balancing between the server and target.

The capacity of the Lustre file system is the sum of all targets. For example, each of 64 OSSs has two 8 TB targets. The total capacity is nearly 1 PB. If each OST uses 10 TB SATA disks (eight data disks + two parity disks, RAID 6), each disk has 50 MB/s of bandwidth, and each OST provides up to 400 MB/s of bandwidth. If the system is used for the back-end storage of an IB network, one OSS can provide 800 MB/s end-to-end throughput.

4.2 Storage Space Requirements

Estimation methods of MDT and OST storage requirements are different. The size of an MDT depends on the number of inodes required in the file system, whereas the total size of OSTs depends on the amount of data stored in the file system. The amount of data in an MDS is fixed. If MGS data is stored on the MDT, another 100 MB of storage space needs to be added to the MDT.

Every time a file is created in the Lustre file system, an inode is consumed on the MDT. Generally, the default number of file stripes is used. You can set `lfs setstripe` to change this parameter of each file.

When the Lustre ldiskfs file system is formatted initially, all inodes are distributed to MDTs and OSTs. Although storage space and inodes can be increased by adding OSTs, the total number of formatted inodes on MDTs and OSTs remains unchanged. Therefore, the inodes are sufficient to meet future expansion needs. When the file system is in use or a file is being created, files associated with metadata are stored in one of the pre-allocated inodes, without consuming the remaining storage space intended for data files.

4.2.1 MGT Storage Space Requirements

An MGT requires less than 100 MB of storage space, depending on the number of cluster servers managed by the MGS in the file system.
4.2.2 MDT Storage Space Requirements

When you calculate the size of an MDT, it is important to consider the number of files stored in the file system, because the number of inodes determines the size of the MDT. For security, each inode occupies 2 KB of space in the MDT by default. The file system metadata occupies 1% to 2% of the size of files stored in the file system. For example, if the average file size is 5 MB and you have 100 TB of free OST space, the required number of inodes is calculated as follows:

\[
\text{Required inodes} = \frac{100 \text{ TB} \times 1024 \text{ GB/TB} \times 1024 \text{ MB/GB}}{5 \text{ MB/inode}} = 20 \text{ million inodes}
\]

It is recommended that you use at least two times the calculated minimum inodes so that you can expand the storage space if the average file size is smaller than expected. The space occupation is calculated as follows:

\[
2 \text{ KB/inode} \times 40 \text{ million inodes} = 80 \text{ GB}
\]

If the average file size is small, for example 4 KB, MDTs will not deliver a high efficiency compared with OSTs. If the MDT is small-sized, a great deal of OST space becomes unavailable. Therefore, before formatting the system, ensure that the MDT space is sufficient for supporting the file system.

4.2.3 OST Space Requirements

The space required by OSTs depends on the user or application model. Lustre software uses a conservative method to estimate the size of an object (16 KB each object). If you are sure that the average file size is larger than 16 KB, you can specify a larger average file size (fewer inodes) to reduce file system workload and file check times.

4.3 Choosing a File System Format

By default, the Lustre uses the following parameters for storage data and metadata for enhanced performance and reliability.

- **flex_bg**: activates the flexible group function. Upon a write or read request, this parameter aggregates multiple block and inode bitmaps to minimize the seek time and reduce read, change, and write operations in RAID storage (with a RAID stripe depth of 1 MB). The flex_bg tag can be opened in OST and MDT file systems. The default value of flex_bg is 16 in an MDT file system and 256 in an OST file system. You can set all block and inode bitmaps to reside in the same flex_bg so that a single I/O can read/write common RAID storage.

- **huge_file**: sets the size of OST to be larger than 2 TB.

- **lazy_journal_init**: This extension can avoid writing the 400 MB log allocated by the Lustre file system, thereby reducing the system formatting time.

4.3.1 MDT

The number of inodes in an MDT depends on the total size of the formatted file system. By default, one inode represents 2048 bytes space. It is recommended that the default value be kept. This value affects the calculation of the metadata space requirement and cluster consistency.
4.3.2 OST

When an OST file system is formatted, inodes contribute to the calculation of local file system usage. When storage space is reserved for future expansion, reducing the inodes of the OST helps shorten the formatting and file system check duration and obtain more data storage space. The following table lists the default inode ratios of different OSTs:

<table>
<thead>
<tr>
<th>LUN/OST size</th>
<th>Inode ratio</th>
<th>Total inodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>over 10GB</td>
<td>1 inode/16KB</td>
<td>640 - 655k</td>
</tr>
<tr>
<td>10GB - 1TB</td>
<td>1 inode/64kB</td>
<td>153k - 15.7M</td>
</tr>
<tr>
<td>1TB - 8TB</td>
<td>1 inode/256kB</td>
<td>4.2M - 33.6M</td>
</tr>
<tr>
<td>over 8TB</td>
<td>1 inode/1MB</td>
<td>8.4M - 134M</td>
</tr>
</tbody>
</table>

If a number of small files exist, using the default inode ratio will require many more inodes than what is required if the average file size is used. Therefore, increasing the inode ratio can improve performance and file check efficiency on OSTs.

4.4 File and File System Restrictions

The following table describes the file and file system restrictions of the Lustre architecture of Linux virtual file system.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Design Specifications</th>
<th>Specifications in Production Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of clients</td>
<td>100 to 100,000</td>
<td>The maximum clients can be over 50,000, whereas most customers use 10,000 to 20,000 clients.</td>
</tr>
<tr>
<td>Number of OSSs</td>
<td>One OSS: Supports 1 to 32 OSTs, each providing up to 128 TB of capacity. Number of OSSs 500 OSSs with 4000 OSTs</td>
<td>One OSS A maximum of 8 OSTs, each providing 16 TB of capacity. Number of OSSs 450 OSSs with 1000 4 TB OSTs 192 OSSs with 1344 8 TB OSTs</td>
</tr>
<tr>
<td>Feature</td>
<td>Design Specifications</td>
<td>Specifications in Production Environment</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>OSS performance</td>
<td>One OSS</td>
<td>One OSS</td>
</tr>
<tr>
<td></td>
<td>5 GB/s</td>
<td>&gt; 2 GB/s</td>
</tr>
<tr>
<td></td>
<td>Aggregation</td>
<td>Aggregation</td>
</tr>
<tr>
<td></td>
<td>2.5 TB/s</td>
<td>240 GB/s</td>
</tr>
<tr>
<td>Number of MDSs</td>
<td>One MDS</td>
<td>One MDS</td>
</tr>
<tr>
<td></td>
<td>ldiskfs; 4 billion files</td>
<td>1 billion files</td>
</tr>
<tr>
<td></td>
<td>ZFS: 256 trillion files</td>
<td>Number of MDSs</td>
</tr>
<tr>
<td></td>
<td>Number of MDSs</td>
<td>1 active +1 standby</td>
</tr>
<tr>
<td></td>
<td>1 active +1 standby</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Up to 4096 MDTs and MDSs</td>
<td></td>
</tr>
<tr>
<td>MDS performance</td>
<td>35,000 creation operations per second</td>
<td>15,000 creation operations per second</td>
</tr>
<tr>
<td></td>
<td>100,000 metadata operations per second</td>
<td>35,000 metadata operations per second</td>
</tr>
<tr>
<td>File system capacity and</td>
<td>One file</td>
<td>One file</td>
</tr>
<tr>
<td>number of files</td>
<td>Up to 2.5 PB in size</td>
<td>TB-level in size</td>
</tr>
<tr>
<td></td>
<td>Aggregation</td>
<td>Aggregation</td>
</tr>
<tr>
<td></td>
<td>512 PB of storage space, 4 billion files</td>
<td>55 PB of storage space, 1 billion files</td>
</tr>
</tbody>
</table>

4.5 Memory Requirement

4.5.1 Client Memory Requirement

It is recommended that at least 2 GB of memory be configured for each client.

4.5.2 MDS Memory Requirement

The required MDS memory depends on the following requirements:

- Number of clients
- Size of directories
- Server workload

The MDS memory requirement depends on how many clients work in the system and how many files the file system contains. First, the MDS memory affects how many locks a client can have at a time. The number of client locks change depending on server workload and available memory. One client can maintain 10,000 locks at a time. In MDS, Lustre distributed lock management (DLM) memory uses around 2 KB for each file. Caching file data can improve metadata reading performance by 10 times or more. MDS memory requirements include:
- File system metadata: The memory requirement of file system data needs to be reasonable. If a great deal of memory is available and no hardware restriction is imposed, the disk I/Os of querying metadata decrease notably.
- Network transfer: Memory is used for sending and receiving cache for TCP protocol or other network transfer methods. Therefore, network transfer bandwidth is also considered.
- Log size: By default, the Lustre file system log size of each idiskfs is 400 MB. You can allocate memory to each file system using MDS memory.

HA configuration: If an MDS requires HA, the memory consumption of each log file will double. In doing so, if a primary server fails, the secondary server can take over. The default file system log size is 400 MB.

The remaining memory is used for caching file data of large tasks to keep cold data active for faster access. A file without any lock requires around 1.5 KB of storage space.

For example, one MDT connection to MDS has 1000 clients, 16 interaction nodes, 2 million files (400,000 files cached on clients. MDS memory requirements are as follows:

Operating system overhead = 512 MB
File system log = 400 MB
1000 x 4 core clients x 100 files/core x 2 KB = 800 MB
16 interactive clients x 10,000 files x 2 KB = 320 MB
1.6 million files x 1.5 KB/file = 2400 MB

Therefore, the minimum requirement is 4 GB of memory. However, more memory contributes to performance improvement. For example, when clients access 10 million files randomly, the excessive memory can improve the access speed.

4.5.3 OSS Memory Requirements

When planning server node hardware, consider the usage of Lustre components (such as log, service thread, and file metadata). OSS hardware planning also affects effect of the memory-consuming OSS data caching feature. OSS memory requirements are as follows:
- Service thread: Service threads on OSS nodes allocate 4 MB of I/O buffer to each OST_IO service thread. Therefore, for each I/O request, the buffer does not need to be allocated or released.
- OSS read cache: OSS write cache uses regular Linux pages to cache storage data as if the cache were from the file system of a common Linux operating system. OSS read cache uses as much physical memory as possible.

OSSs and MDSs have the same file organization method. However, OSS nodes handle more workload than MDSs. Memory resource is primarily used for locks and inode cache. The following is the method for calculating the minimum memory when one OSS connects to two OSTs:

Sending and receiving cache of Ethernet/TCP (4 MB x 512 threads) = 2048 MB
400 MB log size x 2 OSTs = 800 MB
1.5 MB read and write/ OST I/O threads x 512 threads = 768 MB
600 MB file system read cache x 2 OSTs = 1200 MB
1000 x 4 core clients x 100 files/core x 2 KB = 800 MB
16 interactive clients x 10,000 files x 2 KB = 320 MB
1,600,000 additional files x 1.5 KB/file = 2400 MB
DLM locks + file system metadata = 3520 MB
Each OSS DLM lock + File system metadata = 3520 MB/6 OSTs = 600 MB (approximately)
Allocating buffer alone consumes around 1400 MB of memory, and the file system and kernel require extra 2 GB of memory at least. Therefore, for an HA file system with one OSS and two OSTs, the minimum required memory is 4 GB. Configuring more memory helps improve the performance of reading sequential small files.
For an HA system, minimum required memory (including extra memory for performance improvement) is 6 GB. For HA configuration where each OSS has four OSTs, at least 10 GB of memory is required. If an OSS is not handling any OST failover, the remaining memory is used as read cache. Typically, 2 GB of basic memory and 1 GB of memory for each OST are the minimum requirement. In the event of failover, each OST requires at least 2 GB of memory.

### 4.6 Network Requirement

As a high-performance file system, the Lustre file system carries a heavy network load. Therefore, network ports on each Lustre server and client are usually exclusively used by the Lustre file system. The subnet is also an exclusive TCP/IP network. A typical Lustre file system has the following networks:

The Lustre server has a high-performance back-end network, usually IB network.

A large-scale client network

A Lustre router is used to connect these two networks.

Complete the following steps to prepare for LNET configuration:

Confirm the port of each host used for running the Lustre file system.

**A network is a group of nodes that can directly communicate with each other.** Lustre network drivers (LNDs) support multiple types of networks and network hardware. The standard rule is to specify a network application for the LNET. For example, two different TCP subnets (TCP0 and TCP1) are used for two different LNETs. Do not enable Lustre for non-Lustre file system ports to avoid unnecessary overhead.

If you use multiple types of networks, only one router is required.

Any node with suitable ports can route LNET traffic between different types of network hardware or topologies. A router node can be a server, a client, or an independent router. The LNET can span different types of networks (such as TCP and IB) or different topologies, for example, bridge two IB or TCP/IP networks.

### 4.7 Security Configuration

During Lustre installation, the management server and each server must be mutually accessible. During Lustre operation, the MGS, MDS and each server must be accessible.
between each other. Most HPC environments use dedicated networks. Therefore, it is recommended that firewalls and SELinux be disabled.

If the firewall and SELinux cannot be disabled due to special requirements, allow port 988 of all hosts to be accessible. LNET will use free reserved ports (such as ports 1021, 1022, and 1023) to establish connections.
5 Reference Deployment of an HPC Distributed File System

5.1 Application Requirements

As informatization continues to be a popular trend and generates huge volumes of data, industries have sensed development opportunities where HPC can help enterprises extract more value from the data. Back-end data processing requires large I/Os, high bandwidth, and high throughput, posing a significant challenge to traditional storage.

This chapter uses HPC I/O model for meteorological analysis to describe how to use the HPC distributed file system of the OceanStor V3 to meet the challenge. The following figure shows the scenario:

![Application scenario](image)

**Figure 5-1** Application scenario

This test focuses on verifying the functions of the HPC distributed file system. The HPC computing and cluster functions are not described in detail.

5.2 Deployment Plan

5.2.1 Solution Architecture

The solution architecture is as follows:
Figure 5-2 Solution architecture

![Solution architecture diagram](image)

**NOTE**

1. **Server deployment**
   - Deploy one MDS and one MGS for processing metadata and management data.
   - Deploy four OSS nodes and use each two to set up a cluster. The four nodes share data I/Os.
   - Deploy a management server to manage the entire HPC distributed file system.
   - Deploy an E9000 server to simulate computing data and I/O initiator.

2. **OceanStor V3 deployment**
   - Deploy an OceanStor 5800 V3 storage system for storing metadata, management data, and computing data.

3. **Network deployment**
   - Deploy an IP switch to set up the management network of the HPC distributed file system.
   - Deploy an IB switch for interconnecting the computing clients (Lustre clients), MDS, MGS, and OSS nodes.
   - Use optical fibers to directly interconnect the OceanStor 5800 V3, MDS, MGS, and OSS for setting up the storage network.
   - Use Ethernet cables to directly connect MDS and MGS, and each pair of OSS nodes to set up the heartbeat network.

**5.2.2 Solution Configuration**

The configuration of the preceding hardware is as follows:
### Table 5-1 Hardware configuration

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
<th>Specifications</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>ManagerServer</td>
<td>RH 2288 V3&lt;br&gt;• CPU: Intel E5-2680 V3 2.50 GHz&lt;br&gt;• Memory: 64 GB&lt;br&gt;• Disk: 300 GB</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>MDS</td>
<td>RH 2288 V3&lt;br&gt;• CPU: Intel E5-2680 V3 2.50 GHz&lt;br&gt;• Memory: 128 GB&lt;br&gt;• Disk: 300 GB&lt;br&gt;• HBA: QLogic QLE2562 8 Gbit/s Fibre Channel&lt;br&gt;• HCA: Mellanox Connect-IB MT4113</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>OSS</td>
<td>E9000 chassis&lt;br&gt;CH220 V3 blade server&lt;br&gt;CH121 V3 blade server&lt;br&gt;CX611 1B switch module&lt;br&gt;CX110 IP switch module</td>
<td>1</td>
</tr>
<tr>
<td>Storage</td>
<td>OceanStor V3 array</td>
<td>OceanStor 5800 V3 engine&lt;br&gt;V3 4 U 3.5-inch disk enclosure&lt;br&gt;15k rpm 600 GB SAS disk&lt;br&gt;200 GB SLC HSSD&lt;br&gt;4 x 12 Gbit/s SAS disk module&lt;br&gt;4 x 8 Gbit/s FC interface module</td>
<td>1&lt;br&gt;8&lt;br&gt;156&lt;br&gt;15&lt;br&gt;4&lt;br&gt;6</td>
</tr>
</tbody>
</table>

The following table lists the software used during the solution deployment:

### Table 5-2 Software configuration

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>6.5</td>
<td>Operating system</td>
</tr>
<tr>
<td>Intel EE</td>
<td>2.0.1.2</td>
<td>Intel Enterprise Edition for Lustre software</td>
</tr>
<tr>
<td>IB driver</td>
<td>2.1</td>
<td>Mellanox HCA driver</td>
</tr>
<tr>
<td>IOR</td>
<td></td>
<td>Test software</td>
</tr>
</tbody>
</table>
The following table describes the version compatibility between Intel EE for Lustre software, Lustre, and operating system:

Table 5-3 Software compatibility

<table>
<thead>
<tr>
<th>IEEL Version</th>
<th>Lustre Base</th>
<th>Server Support</th>
<th>Client Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEL 1.0.X</td>
<td>2.3.X</td>
<td>RHEL 6.4 and CentOS 6.4</td>
<td>RHEL 6.4 and CentOS 6.4</td>
</tr>
<tr>
<td>IEEL 2.0.X</td>
<td>2.5.X</td>
<td>RHEL 6.5, CentOS 6.5, SLES 11 SP3</td>
<td>RHEL 6.5, CentOS 6.5, SLES 11 SP3</td>
</tr>
</tbody>
</table>

Figure 5-3 Storage configuration

NOTE
The configuration of the OceanStor 5800 V3 is as follows:
1. One disk domain
   - 156 x 15k 600 GB SAS disks
2. Two storage pools
   - Storage pool one uses RAID 10 and stores metadata and management data.
   - Storage pool 2 uses RAID 6 (8+2) and stores computing data.
3. 26 LUNs
   - One 10 GB LUN stores management data.
   - One 1000 GB LUN stores metadata.
   - Twenty-four 1 TB LUNs store computing data.
5.3 Deployment Procedure

5.3.1 Procedure

Step 1  Prepare an operating system.
1. Lustre supports Red Hat Enterprise Linux (RHEL), CentOS, and SUSE Linux Enterprise Server (SLES). In this document, CentOS is used. Note that CentOS must be installed using a set of basic software because the installation of Lustre requires certain dependent packages and kernel change.
2. Configure management network ports, heartbeat network ports, and domain name resolution to ensure connectivity between the management, computing, and heartbeat planes.
3. Disable the firewall.
4. Install the IB HCA driver.
5. Create a software repository to facilitate Lustre software installation.

Step 2  Prepare storage configuration.
1. Estimate the required storage capacity.
2. Configure storage.
3. Configure multipathing software.

Step 3  Install the management server.
1. Copy and decompress the `ieel-1.0.2.tar.gz` software package.
2. Execute the install script to start installation. During the installation, you are asked to create a new administrator account and set its password and email address.
3. Press the whitespace button to read the license agreement terms and accept the license.

Step 4  Deploy the Lustre server.
1. Configure LNET.
2. Add the Lustre server node.
3. Configure volumes.
4. Configure IPMI.
5. Create a file system.

Step 5  Deploy Lustre clients.
1. Download the client source code and compile it.
2. Install the client software.
3. Mount the Lustre distributed file system.

---End
5.4 Testing and Acceptance

The distributed file system deployed in the preceding section meets the following requirements:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Test Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensive read/write bandwidth (read, write, re-read, and re-write)</td>
<td>10 GB/s</td>
</tr>
<tr>
<td>Comprehensive application performance per pair of OSS nodes</td>
<td>10 GB/s</td>
</tr>
<tr>
<td>I/O delivery per client</td>
<td>5.2 GB/s</td>
</tr>
<tr>
<td>Performance per disk</td>
<td>81 MB/s data reading and writing</td>
</tr>
</tbody>
</table>

5.5 Performance and Capacity Expansion

The HUAWEI HPC distributed file system can scale up its performance and capacity by adding disks, storage nodes (engines), and I/O nodes (OSS nodes). The post-expansion performance can be estimated as follows:

- If a disk enclosure has vacant disk slots, adding disks improves performance by: Number of disks x Read/Write performance of each disk.
- The read/write bandwidth increases by 10 GB/s every time a storage node is added, until the number of I/O nodes reaches the upper limit.
- If the disks, front-end links, and back-end links have no bottleneck, adding storage nodes improves performance by: Number of storage nodes x Performance of each node.
- If each OSS has one IB interface card and two 16 Gbit/s Fibre Channel interface cards, adding one pair of OSS nodes improves performance by: 2 x min [Read/Write bandwidth of IB interface card, 2 x Fibre Channel interface card bandwidth].

The file system performance and capacity can reach the optimum by combining the preceding expansion granularities. The following shows the expansion workflow:
5.6 Solution Benefits

Under the preceding reference architecture, you can easily set up your own HPC distributed file system, which brings the following benefits:

- **Flexible scale-out expansion**
  The HPC distributed file system supports flexible scale-out expansion by disk, storage node, and I/O node, meeting diversified application performance and capacity needs at different levels.

- **High-performance massive data storage**
  After expansion, the HPC distributed file system provides up to 100 Gbit/s of aggregate bandwidth and PB-level storage capacity, applicable to various high-performance and massive data scenarios.

- **Excellent network switching capacity**
  To help customers balance between cost and performance, the Huawei HPC distributed file system reference architecture supports GE, 10GE, 40GE, and QDR/FDR IB interconnect for the computing plane.
6 Reference Optimization of an HPC Distributed File System

6.1 Hardware Optimization

Hardware optimization indicates proper hardware selection to eliminate I/O path bottlenecks. The workflow of an HPC distributed file system experiences the following steps from the start of the application to the storage on disks.

The application is started. > The computing nodes prepare computing resources. > Data is obtained from the storage devices. > The computing nodes complete processing the data. > Data is returned to the storage for archiving.

From the perspective of the HPC distributed file system, I/O paths are used in the following steps:

The computing nodes (Lustre clients) deliver I/Os through the mounted HPC distributed file system. > The Lustre clients query configuration information on the MDS and MGS. > The Lustre clients decompress the I/Os and distribute them to each OSS node over the IB network. > The OSS nodes deliver I/Os to the OST over the IB or Fibre Channel network. > The storage engine allocates data blocks to each disk enclosure. > Data is written to disks in the disk enclosure.

The preceding analysis shows that I/O paths involve Lustre clients, IB HCAs, IB switches, MDS, MGS, OSS nodes, Fibre Channel cards, the storage engine, SAS cards, and disks. Resolve performance bottlenecks in each step to ensure smooth delivery of I/O load to disks.

6.2 Computing Node

When the Lustre file system is being read, computing nodes and the Lustre file system consume a large amount of CPU resources. Based on bandwidth requirements observe whether sufficient CPU resources are used by the application and Lustre clients.

6.3 Storage Nodes

6.3.1 Load Balancing

- Unify the type of disks (rotational speed and capacity) to balance data writing.
LUNs can be evenly owned by controllers. It is recommended that the number of LUNs be a multiple of the number of client CPUs.

Because, the Lustre distributed file system processes predominately large files, no prefetch is recommended for the storage nodes.

Map groups of LUNs to pairs of OSS nodes to ensure that both OSS nodes in a pair can read the share disks.

6.3.2 Multipath Optimization

Customers typically use the Multipath software embedded in CentOS. The following describes how to optimize Multipath:

- Use static bonding of Multipath to add LUNs and enable ALUA. The following method applies to all OceanStor V3 storage systems.

```bash
multipaths {
  multipath {
    wwid 36f84abf10056bfc0065bfe410000000c  //SCSI ID of the LUN
    alias ost1
  }
  multipath {
    wwid 36f84abf10056bfc0065c04dc0000000d  //SCSI ID of LUN
    alias ost2
  }
}
devices {
  device {
    vendor                  "HUAWEI"   //Storage supplier
    product                 "BXSG"     //Storage model
    path_grouping_policy    group_by_prio   //Paths are grouped by priority.
    getuid_callout          "/lib/udev/scsi_id --whitelisted
                          --device=/dev/%n"                                    //Obtain LUN WWN.
    path_checker            tur           //TUR checks path status.
    path_selector           "round-robin 0"  //Path selection algorithm
    failback                immediate      //path failover time
    prio                    alua         //invisible ALUA
  }
}
```

- Use Multipath to shield unnecessary LUNs to avoid using an incorrect LUN.
- It is recommended that you use two to six paths for each OSS node. An excessive number of paths will cause high performance usage during path selection.
- The preceding optimization methods apply to different scales of HPC distributed file systems.
6.4 Network Nodes

6.4.1 IP Network Performance Optimization

- If IP is used for networking, an optical network faster than 10GE is recommended over GE network.
- Due to bandwidth constrains, Ethernet uses 2.5 times the servers of IB network.
- If 10GE is used for Lustre network, ensure that you have enabled jumbo frame (by setting the MTU size on the server and enabling jumbo frame mode on the switch).
- Port bonding on a 10GE network does not greatly improve throughput.
- 10GE interface cards of the same model and brand are recommended.

6.4.2 IB Network Optimization

- If IB network is used, set the LNET mode in Lustre to o2ib instead of TCP.
- IB network products from Mellanox are recommended.
- FDR IB HCAs are recommended. Before use, verify the claimed bandwidth of the HCAs.
- If 56 Gbit/s to 120 Gbit/s IB cards are used, ensure that they match the PCIe bandwidth of the hosts.
- After the change of Lustre kernel, IB card drivers require re-compilation. Select proper driver versions.
- IB cards of the same model and brand are recommended.

6.4.3 Fibre Channel Network Optimization

- The Fibre Channel bandwidth must match the front-end host interconnect bandwidth. For example, if the hosts use FDR IB, one host needs two 8 Gbit/s Fibre Channel cards or one 16 Gbit/s Fibre Channel card. Otherwise, the back end will become a bottleneck.
- It is recommended that the Fibre Channel network use 16 Gbit/s Fibre Channel cards for saving PCI resources. IB cards can also be used.
- If dual Fibre Channel cards are used, it is recommended that each Fibre Channel card connects to one controller for optimum performance.
- Direct connection is recommended for Fibre Channel networking. Fibre Channel switches are not recommended.
- FC HBAs of the same model and brand are recommended.

6.5 Application

6.5.1 Basic Performance Optimization

- Lustre load balancing adjustment: After MDS and OSS nodes are added, Lustre automatically allocates volumes (or LUNs) to each node to ensure the LUNs are evenly distributed to each storage node.
- Lustre read performance optimization: Read performance is not optimized by default. To optimize it, change the read cache parameters `max_read_ahead_mb(800)` and `max_read_ahead_whole_mb(8)`. 
When creating and installing the Lustre file system, choose high-speed networks (such as IB or 10GE) as the communication network. Automatic network selection may be unsuitable.

6.5.2 Small File Performance Optimization

The Lustre file system performs well when read or writing large-sized files. In the case of small files, perform the following operations to optimize performance:

- Aggregate small I/Os before writing them to Lustre.
- Disable the LNET debug function of clients to reduce overheads.
- Increase the number of RPCs to improve concurrency capabilities.
- Adjust the Lustre stripe size.

6.5.3 OSS and Client Nodes

- Configure similar CPU and memory resource between OSS nodes and between clients and nodes.
- Properly increase the memory for OSS nodes because they consume a great deal of memory when reading and writing data.
7 Appendix

7.1 Reference Documents

HUAWEI OceanStor 5300 V3&5500 V3&5600 V3&5800 V3&6800 V3&6900 V3 Storage Systems V300R001 Product Documentation
Intel Enterprise Edition Lustre Brief
Lustre Manual

7.2 Terminology

The following table lists the terms used in this document:

Table 7-1 Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>Distributed file system</td>
<td>A distributed file system distributes files to multiple servers or storage systems for storage, delivering high performance and fault tolerance.</td>
</tr>
<tr>
<td>Lustre</td>
<td>A typical type of distributed file system</td>
</tr>
<tr>
<td>MDS</td>
<td>A metadata server processes the metadata of a file system.</td>
</tr>
<tr>
<td>MDT</td>
<td>A metadata target indicates the object that stores the metadata of a file system.</td>
</tr>
<tr>
<td>MGS</td>
<td>A management server is a software module that manages file system configuration.</td>
</tr>
<tr>
<td>MGT</td>
<td>A management target stores objects of management data.</td>
</tr>
<tr>
<td>OSS</td>
<td>An object storage server processes application data.</td>
</tr>
<tr>
<td>OST</td>
<td>An object storage target is an object that stores application data.</td>
</tr>
</tbody>
</table>