# Parallel distributed file systems

Michael Kuhn

Research Group Scientific Computing
Department of Informatics
Universität Hamburg

2016-03-03

# Definition

- Parallel file systems
  - Allow parallel access to shared resources
  - Access should be as efficient as possible
- Distributed file systems
  - Data and metadata are distributed across servers
  - Single servers do not have a complete view
- Naming is inconsistent
  - Often just “parallel file system” or “cluster file system”

# Definition…

- Parallel distributed file systems are typically used in high performance computing
  - NFS only for non-(performance-)critical applications
- Local file systems also allow parallel access
  - Locks, e.g. via `flock` or `lockf`

# Definition…

- Storage Area Network (SAN)
  - Provides block devices via the network
  - File systems can be put on top of the devices
  - Parallel distributed file systems can use SANs
- Network Attached Storage (NAS)
  - Abstracts storage devices
  - Directly provides a file system
    - Typically NFS or SMB

# Basics

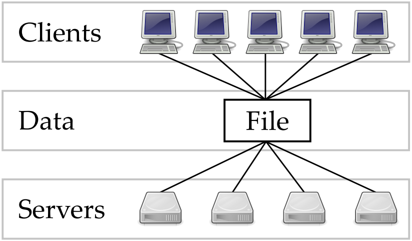



Figure: Parallel access and data distribution

- Clients access a shared file in parallel
- File is distributed across multiple servers and storage devices
  - Higher throughput and capacity

# Architecture

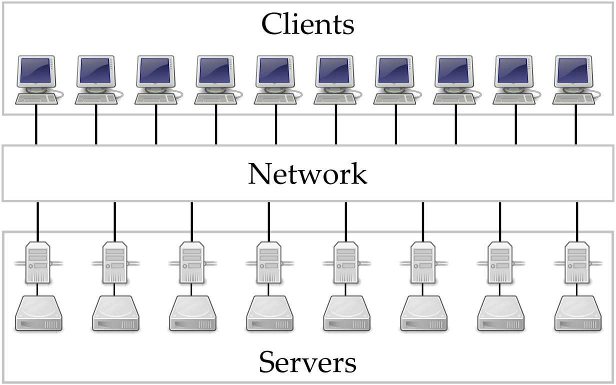



Figure: Parallel distributed file system

# Architecture…

- Clients and servers
  - Clients execute parallel applications
  - Have access to the file system via the network
  - Typically no local storage and no direct access to the storage devices
  - No influence on each other
- Data and metadata servers
  - Simple controllers or full-fletched servers
  - Differences between servers for data and metadata
    - Produce different access patterns

# Architecture…

- Different communication schemes
  - Clients know responsible server (more common)
  - Clients contact a random server
- Different server behavior
  - Servers notify clients about responsible server
  - Servers transparently relay requests
- Advantages and disadvantages in both cases

# Architecture…

- Clients know responsible server
  - Advantages: Simple communication protocol (one-to-one), no communication between servers
  - Disadvantages: Distribution logic has to be implemented by the clients, additional client-side information required
- Clients contact a random server
  - Advantages: Clients do not need to know data/metadata distribution, load balancing is easier to implement
  - Disadvantages: Access requires additional messages, more complex communication protocol

# Interface

- Access to file system through I/O interface
  - Usually standardized
  - Proprietary interfaces with more functionality or performance
- Interfaces comprise syntax and semantics
  - Syntax defines available operations
  - Semantics defines operations' behavior
- Often POSIX interface
  - Standardized and portable

# Interface…

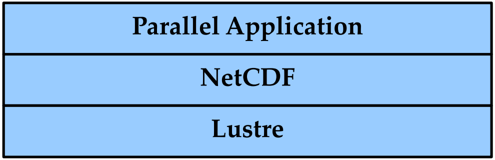



Figure: I/O stack from the applications' point of view

- Applications use abstract interfaces
  - NetCDF provides self-describing data format
- Parallel distributed file system handles efficient access and distribution

# Semantics

- POSIX has strict consistency requirements
  - Changes have to be visible globally after `write`
  - I/O should be atomic
- POSIX for local file systems
  - Requirements are easy to fulfill
  - Everything is handled by the VFS
- Small aspects are changeable
  - `strictatime`, `relatime` and `noatime` for behavior regarding timestamps
  - `posix_fadvise` for announcing the access pattern

# Semantics. . .

- Contrast: Network File System (NFS)
  - Same syntax, considerably different semantics
- So-called session semantics
  - Changes are not visible to other clients
    - Only within session
  - Other clients only see changes after session ends
  - `close` writes changes and returns potential errors
- Later: MPI-IO
  - Less strict for higher scalability

# Reality

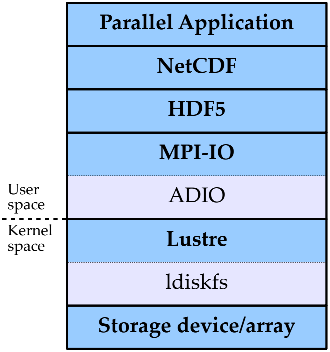



Figure: I/O stack in HPC

- Data transformation
  - Transport through all layers
  - Loss of information
- Complex interaction
  - Optimizations and workarounds per layer
  - Information about other layers
- Comfort vs. performance
  - Structured data in application
  - Byte stream in POSIX

# Reality…

| Abstraction | Interface | Data types | Control |
|---|---|---|---|
| High | NetCDF | Structures | Coarse-grained |
| | MPI-IO | Elements | |
| Low | POSIX | Bytes | Fine-grained |

Figure: Level of abstraction

- High abstraction offers comfort but little control
- Low abstraction allows exact tuning

# Examples

- GPFS (IBM)
- Lustre (Intel)
- OrangeFS (formerly PVFS)
- CephFS (Red Hat)
- BeeGFS (Fraunhofer, formerly FhGFS)
- GlusterFS (Red Hat)

# General

- I/O is expensive in comparison to computation
  - Context switch, slow storage devices etc.
- Parallel distributed file systems have to do I/O via the network
  - Restrictions regarding throughput and latency
- Novel concepts like burst buffers
  - Store data temporarily
  - Steady forwarding to the file system
  - E.g. node-local non-volatile memory (NVRAM)

# Data

- Data is potentially accessed by multiple clients
  - Overlapping vs. non-overlapping
  - Read accesses typically unproblematic
- Overlapping write accesses
  - Heavily dependent on I/O semantics
  - Usually requires locks
  - Often requires distributed lock management

# Data...

- Data distribution is relevant for performance
  - Number of data servers to contact
  - Realized using distribution functions
    - Typically simple round robin
    - Sometimes controllable by the user
    - E.g. to support heterogeneous access patterns

# Data…

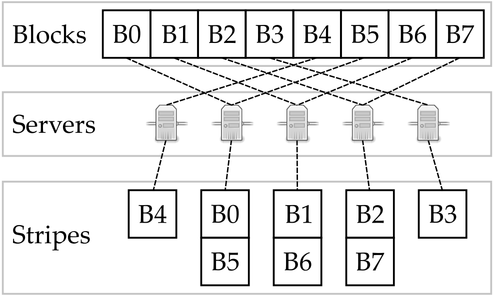



Figure: Round robin data distribution

- File consists of blocks, is distributed across servers in stripes
    - Block size is equal to the stripe size in this case
- Distribution does not have to start on the first server

# Metadata

- Metadata is accessed by multiple clients
  - Read accesses are unproblematic again
  - Parallel modification of size, timestamps etc.
- Metadata of one file is managed by one server
  - Updates are typically serial
- Potentially millions of clients
  - DDoS caused by clients on the metadata server

# Metadata…

- Distributed approaches to metadata management
  - Changeable metadata not stored centrally
    - E.g. size and timestamps
  - Calculation during runtime
    - Client contacts all relevant data servers
    - Updates sped up by making requests slower
- Metadata distribution analogous to data distribution
  - Determine server by e.g. hashing the path
  - Typically has to be deterministic
    - Clients have to be able to autonomously determine the responsible servers
  - One object usually managed by one server
    - Exceptions for directories as of late

# Metadata...

- Many metadata operations are inherently serial
  - E.g. path resolution

1. `/foo/bar`
   1. Read root directory's inode
   2. Check access permissions
   3. Read root directory and search for `foo`
2. `/foo/bar`
   1. Read directory's inode
   2. Check access permissions
   3. Read directory and search for `bar`
3. `/foo/bar`
   1. Read file's inode
   2. Check access permissions
   3. Access file

# Metadata…

| Technology | Device | IOPS |
|---|---|---|
| HDD | 7,200 RPM | 75–100 |
| | 10,000 RPM | 125–150 |
| | 15,000 RPM | 175–210 |
| SSD | Intel X25-M G2 | 8,600 |
| | OCZ Vertex 4 | 85,000–90,000 |
| | Fusion-io ioDrive Octal | 1,000,000+ |

Table: IOPS for selected HDDs and SSDs

- Separated servers allow targeted optimizations
  - E.g. hard disk drives for data, flash devices for metadata
  - Different prices (factor $\geq 10$)
  - Metadata make up $\approx 5\,\%$ of overall data

# State of the Art

- Parallel distributed file systems allow huge and high-performance storage systems
- Blizzard (DKRZ, GPFS)
  - Capacity: 7 PB
  - Throughput: 30 GB/s
- Mistral (DKRZ, Lustre)
  - Capacity: 50 PB
  - Throughput: 400 GB/s
  - IOPS: 80,000 operations/s
- Titan (ORNL, Lustre)
  - Capacity: 40 PB
  - Throughput: 1.4 TB/s

# Overview

- One of the most well-known parallel distributed file systems
- Open source (GPLv2)
  - $>$ 550,000 lines of code
- Supports Linux (exclusively)
  - Name is derived from Linux and Cluster
- Widely used
  - More than half of the TOP100
  - More than a third of the TOP500

# History

- 1999: Initial development
  - Research project at Carnegie Mellon University, lead by Peter Braam
- 2001: Formation of Cluster File Systems
- 2007: Acquisition by Sun
  - Integration into HPC hardware
  - Combined with ZFS
- 2010: Acquisition by Oracle
  - Development discontinued
- Further development by community
  - Intel (formerly Whamcloud), Seagate (formerly Xyratex), OpenSFS, EOFS etc.

# History…

- Version 2.3 (October 2012)
  - Experimental support for ZFS
- Version 2.4 (May 2013)
  - Distributed Namespace (DNE)
  - ZFS for data and metadata
- Version 2.5 (October 2013)
  - Hierarchical Storage Management (HSM)
- Version 2.6 (July 2014)
  - Experimental support for distributed directories
- Currently version 2.7 (March 2015)

# History…

```
$ lfs mkdir --index 0 /lustre/home
$ lfs mkdir --index 1 /lustre/scratch
```

Listing 1: DNE in Lustre

- DNE allows distributing different directories across different metadata servers
  - `/scratch` for large files
  - /home typically with many small files
- Static approach, manual configuration

# Architecture

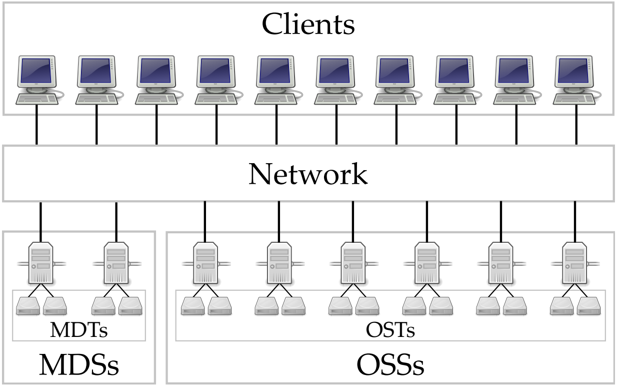



Figure: Lustre architecture

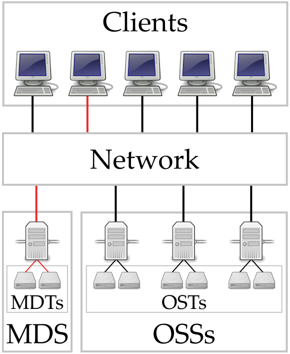
# Architecture…

- Object Storage Servers (OSSs)
  - Manage data
  - Object-based byte-level access
  - One or more Object Storage Targets (OSTs)
- Metadata Servers (MDSs)
  - Manage metadata
  - Not involved in the actual I/O
  - One or more Metadata Targets (MDTs)

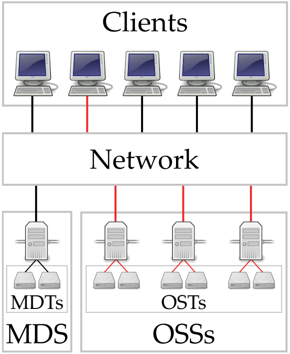
# Architecture. . .

- Data and metadata servers use an underlying local file system
  - Typically ldiskfs (ext4 fork)
  - Alternatively ZFS (Data Management Unit)
    - Avoids POSIX overhead
- No direct access to storage devices by clients
  - Clients send requests to the servers
  - Servers perform operations
  - Servers send results to clients

# Architecture…




(a) Metadata access




(b) Data access

- Metadata server only accessed for initial opening
- Direct parallel access to data servers afterwards

# Support

- Lustre is a kernel file system
  - Client and server
- Client supports (relatively) current kernels
  - Integrated into the kernel since 3.12 (outdated)
  - Support for newer kernels sometimes takes a while
- Server only supports selected enterprise kernels
  - E.g. Red Hat Enterprise Linux (or CentOS)
  - Mainly due to ldiskfs
  - Vanilla kernel supported with ZFS

# Functionality

- Distributed lock management
  - For data and metadata
  - Overlapping read locks and non-overlapping write locks on byte level
  - Mount options `flock` and `localflock`
- POSIX-compliant
  - POSIX interface via VFS
  - No native support for MPI-IO

# Functionality…

- Hierarchical Storage Management
  - Important requirement for large storage systems
  - Supports multiple tiers
    - Hard disk drives, tapes etc.
  - Metadata is still managed by Lustre
  - Data is transparently moved to different tiers
- High availability
  - Supports failover mechanisms
  - Active/passive and active/active configurations

# Functionality...

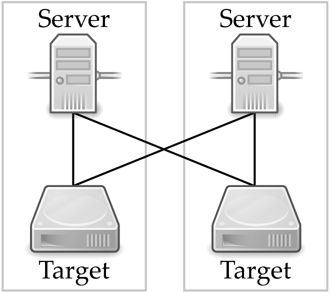



Figure: Active/active failover configuration

- Servers can take over role/work of each other
- Can be used to support uninterrupted upgrades

# Overview

- Open source (LGPL)
  - $>$ 250.000 lines of code
- Developed by Clemson University, Argonne National Laboratory and Omnibond
- Successor of PVFS
  - 2007: Starts as development branch
  - 2010: Replaces PVFS as main version

# Functionality

- Distributed metadata and directories
    - Distributed directories since version 2.9
- Runs completely in user space
- Very good MPI-IO support
    - Native backend in ROMIO
- Support for POSIX interface
    - FUSE file system
    - Optional kernel module

# Functionality...

- OrangeFS is not POSIX-compliant
  - Guarantees atomic execution of non-contiguous and non-overlapping accesses
  - Applies to read and write operations
  - Therefore supports (non-atomic) MPI-IO
- Sufficient for many use cases
  - Stricter semantics is not supported
  - MPI-IO atomic mode not supported due to missing support for locks

# Dependencies

```
systemctl stop firewalld
systemctl disable firewalld
sed -i '/^SELINUX=/s/.*/SELINUX=disabled/'
    ↪ /etc/selinux/config
setenforce 0
```

Listing 2: Prepare system

```
yum -y upgrade
yum -y groupinstall 'Development Tools'
yum -y install xmlto asciidoc elfutils-libelf-devel
    ↪ zlib-devel binutils-devel newt-devel python-devel
    ↪ hmaccalc perl-ExtUtils-Embed bison elfutils-devel
    ↪ audit-libs-devel python-docutils sg3_utils expect
    ↪ attr lsof quilt libselinux-devel
```

Listing 3: Install dependencies

# ZFS

```
URL='http://archive.zfsonlinux.org'

yum -y install epel-release
yum -y install --nogpgcheck
    ↪ $URL/epel/zfs-release.el7.noarch.rpm
yum -y install zfs zfs-dkms libzfs2-devel libuuid-devel

systemctl start zfs.target
```

Listing 4: Set up ZFS

# Lustre

```
git clone --depth 1 git://git.hpdd.intel.com/fs/lustre-release.git
cd lustre-release
sh ./autogen.sh
./configure --disable-ldiskfs
make rpms
yum -y install *.$(arch).rpm
```

Listing 5: Set up Lustre

# Lustre configuration

```
mkfs.lustre --mgs --backfstype=zfs --fsname=lustre --device-size=1048576 lustre-mgs/mgs /tmp/lustre-mgs
mkfs.lustre --mdt --backfstype=zfs --fsname=lustre --index=0 --mgsnode=$(hostname)@tcp --device-size=1048576 lustre-mdt0/mdt0 /tmp/lustre-mdt0
mkfs.lustre --ost --backfstype=zfs --fsname=lustre --index=0 --mgsnode=$(hostname)@tcp --device-size=1048576 lustre-ost0/ost0 /tmp/lustre-ost0
```

Listing 6: Format devices

# Lustre configuration…

```
echo "$(hostname) - mgs  zfs:lustre-mgs/mgs" >> /etc/ldev.conf
echo "$(hostname) - mdt0 zfs:lustre-mdt0/mdt0" >> /etc/ldev.conf
echo "$(hostname) - ost0 zfs:lustre-ost0/ost0" >> /etc/ldev.conf

systemctl daemon-reload
systemctl start lustre
```

Listing 7: Start services

```
mkdir -p /mnt/lustre/client
mount -t lustre $(hostname):/lustre /mnt/lustre/client
```

Listing 8: Mount Lustre

# Summary

- Parallel distributed file systems offer simultaneous access for many clients
  - Scalable parallel access is hard to support
  - Distribute data and metadata for improved throughput and capacity
- Typically separated into data and metadata servers
- Access via I/O interface
  - Often POSIX or MPI-IO
- Important representatives are Lustre and GPFS
  - OrangeFS offers an alternative approach