

# Evaluating Lustre's Metadata Server on a Multi-Socket Platform

Konstantinos Chasapis

Scientific Computing  
Department of Informatics  
University of Hamburg

9th Parallel Data Storage Workshop

# Motivation

Metadata performance can be crucial to overall system performance

- Applications create thousands of files (file-per-process)
  - Normal output files
  - Store their current state, snapshots files
- Stat operation, to check application's state
- Clean "old" - snapshot files and temporary files

Solutions:

- Improve metadata architectures and algorithms
- Use more sophisticated hardware on the metadata servers
  - Increase processing power in the same server, add cores and sockets
  - Replace HDDs with SSDs

# Our work

## Evaluate Lustre Metadata Server Performance when using Multi-Socket Platforms

### Contributions:

- An extensive performance evaluation and analysis of the *create* and *unlink* operations in Lustre
- A comparison of Lustre's metadata performance with the local file systems *ext4* and *XFS*
- The identification of hardware best suited for Lustre's metadata server

- 1 Motivation
- 2 Related Work
- 3 Lustre Overview
- 4 Methodology
- 5 Experimental Results
- 6 Conclusions

# Related Work

## Lustre Metadata performance:

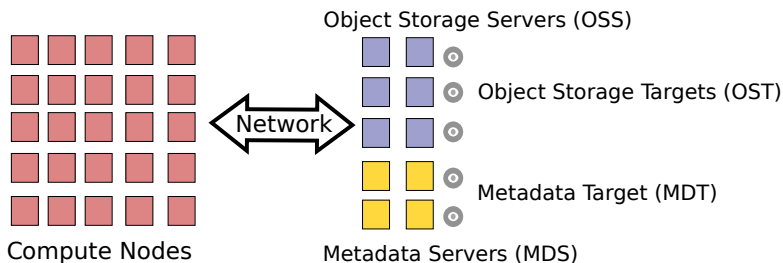
- Alam et al. “Parallel I/O and the metadata wall,” PDSW '11
  - Measured the implications of the network overhead on the file systems' scalability
  - Evaluate performance improvements when using SSDs instead of HDDs
- Shipman et al. “Lessons Learned in Deploying the World s Largest Scale Lustre File System,” ORNL Tech. Rep. 2010
  - Configurations that can optimize metadata performance in Lustre

## Performance scaling with the number of cores:

- Boyd-Wickizer et al. “An analysis of Linux scalability to many cores,” OSDI'10
  - Performed an analysis of the Linux kernel while running on a 48-core server

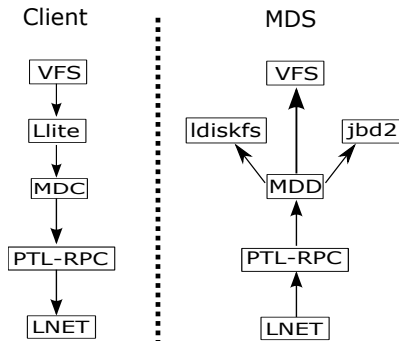
# Lustre Overview

- Parallel distributed file system
- Separates metadata to data servers
- 2.6 version supports distributed metadata
- Uses back end file-system to store data (ldiskfs and ZFS)



# Lustre metadata operation path

- Complex path - goes through many layers
- VFS since it is POSIX compliant
- LNET network communication protocol
- File data stored in OSSs
- Lustre inode store object-id ost mapping



# Methodology

- Use a single multi-socket server
- Use mdtest metadata generator benchmark to stress the MDS
- Compare with XFS and ext4
  - ext4, since `ldiskfs` is based on it
  - XFS, that is a high-performance local node file system
  - ZFS, has poor metadata performance that's we do not include it
- Measure *creat* and *unlink* operations
  - *stat* performance heavily depends in the OSSs

Collect system statistics:

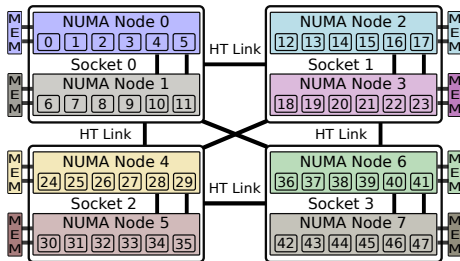
- CPU utilization
- Block device utilization
- OProfile stats: % of CPU consumed by Lustre's modules



# Hardware specifications

We use a four socket server consisting of:

- Supermicro motherboard model H8QG6
- 4× AMD Opteron 6168 Magny-Cours 12-core processors 1.9 GHz
- 128 GB of DDR3 main memory running at 1,333 MHz
- Western Digital Caviar Green 2 TB SATA2 HDD and
- 2× Samsung 840 Pro Series 128 GB SATA3 SSDs
- Memory throughput: 8.7 GB/s for local and 4.0 GB/s for remote



# Testbed Setup

- CentOS 6.4 with the patched Lustre kernel version 2.6.32-358.6.2.el6
- Lustre 2.4 version RPMS provided by Intel (Whamcloud)
- Linux governor is set to `performance`, which operates all the cores at the maximum frequency and gets the maximum memory bandwidth
- An SSD for the OST
- For the `ext4` and `XFS` experiments, we use an SSD
- `cfg` device scheduler for the MDT

# mdtest Benchmark

MPI-parallelized benchmark that runs in phases, where in each phase a single type of POSIX metadata operation is issued to the underlying file system.

## Configuration:

- Private directories per process
- 500.000 files for Lustre
- 3.000.000 files for ext4 and XFS
- Unmount the file system and flush kernel caches after each operation

# Configurations

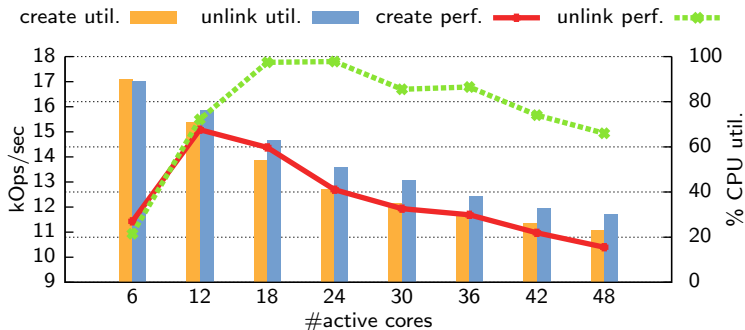
- 1 Scaling with the number of cores
  - Increase the workload and the active cores
- 2 Bind mdtest processes to specific sockets
  - Same workload and divide the mdtest processes among the active sockets
- 3 Use of multiple mount points
  - Increase the mount points used to access the file system
- 4 Back-end device limitation
  - Measure MDS performance while using kernel RAMDISK as MDT

Configuration:

**Scaling with the number of cores**

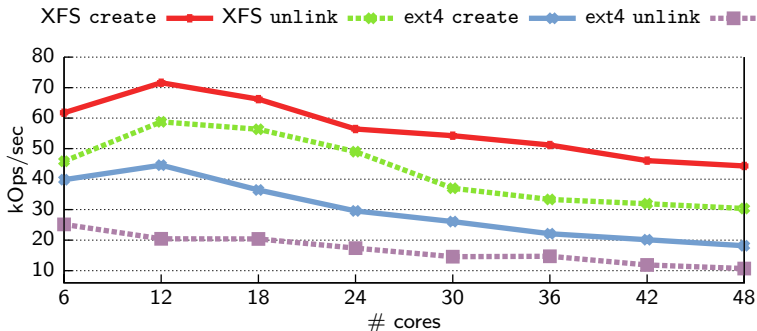
# Lustre's performance vs. Active cores

- #mdtest processes equals  $2 \times$  #active cores
- 6 mount points
- Lustre's modules CPU usage drops by  $2 \times$  from 12 to 24 sockets



## ext4 and XFS performance vs. Active cores

- #mdtest processes equals  $2 \times$  #active cores
- 6 mount points



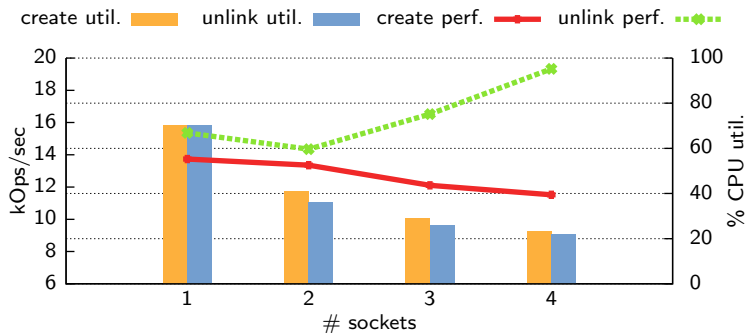
Configuration:

**Bind mdtest processes per socket**



# Lustre performance, bind per socket configuration

- All cores are active
- Group mdtest processes per socket
- 12 mdtest processes
- 12 mount points

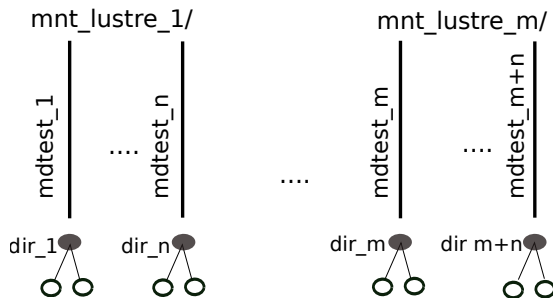


Configuration:

**Use of multiple mount points (MPs)**

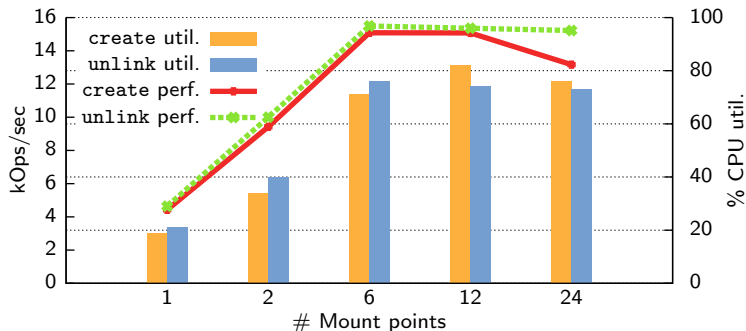
# Multiple mount point configuration

- Mount the file system in several directories
- Access the file system from different paths



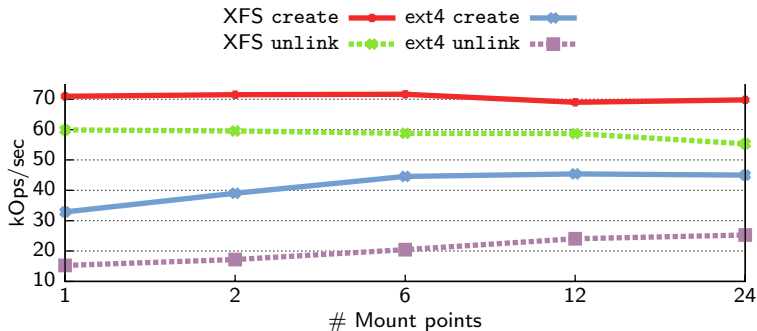
# Lustre's performance vs. Mount points

- 24 mdtest processes
- 12 active cores
- Lustre's modules CPU usage increases by 5x from 1 MP to 12 MPs



## ext4 and XFS performance vs. Mount points

- 24 mdtest processes
- 12 active cores

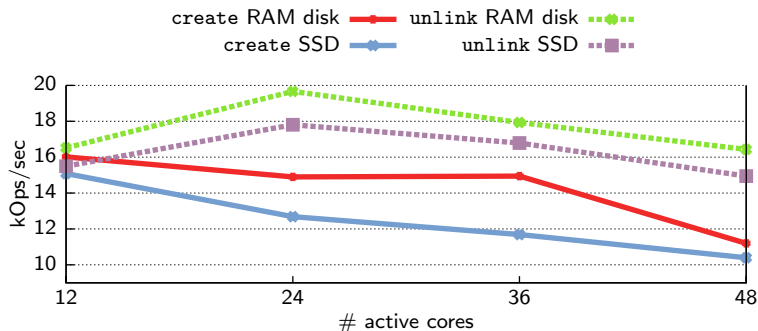


Configuration:

## **Back-end device limitation**

# Lustre's performance using different back-end devices

- #mdtest processes equals  $2 \times$  #active cores
- 6 mount points



# Conclusions

## Main observations:

- Lustre MDS performance improvement is limited to a single socket
- MDT device does not seem to be the bottleneck
- Using multiple mount points per client can significantly increase performance

## Previous work:

- The number of cores is less significant than the CPU clock

Thank you - Questions?

[konstantinos.chasapis@informatik.uni-hamburg.de](mailto:konstantinos.chasapis@informatik.uni-hamburg.de)



## Lustre modules usage increasing the number of active cores

	create		unlink	
<b>Module</b>	<b>12 C</b>	<b>24 C</b>	<b>12 C</b>	<b>24 C</b>
obdclass	6.38	3.58	12.96	8.57
mdtest	4.22	3.59	<0.1	<0.1
ldiskfs	5.30	2.26	3.01	1.70
lnet	<0.1	<0.1	3.24	2.17
libcfs	2.61	1.28	3.30	2.04
osd_ldiskfs	2.16	0.86	2.06	1.11
jbd2	1.76	0.97	1.23	0.78
lvfs	1.41	0.98	1.93	1.60
lustre	1.32	0.56	1.81	0.88
mdd	1.65	0.30	0.9	0.42
mdt	1.65	0.73	1.84	0.97

## Lustre modules usage increasing the number of MPs

	create		unlink	
<b>Module</b>	<b>1 M</b>	<b>12 M</b>	<b>1 M</b>	<b>12 M</b>
ptl-rcp	2.38	11.38	3.04	11.67
obdclass	1.56	7.53	3.03	12.86
ldiskfs	1.30	6.53	0.62	3.12
lnet	0.82	3.75	<0.1	<0.1
libcfs	0.68	3.06	<0.1	<0.1
osd_ldiskfs	0.47	2.61	0.43	2.10
jbd2	0.45	2.15	0.38	1.26
mdt	0.36	2.11	<0.1	<0.1
lvfs	0.30	1.74	0.44	1.97
lustre	0.29	1.70	0.42	1.84
lod	0.18	1.06	<0.1	<0.1
mdd	0.18	1.03	<0.1	<0.1
mdc	0.13	0.71	<0.1	<0.1
mdt	<0.1	<0.1	0.42	1.87