

# Task driven framework for Lustre monitoring

**Gabriele Iannetti**

High Performance Computing  
GSI Helmholtz Centre for Heavy Ion Research  
Darmstadt, Germany

LAD'17 Paris, France

# Agenda

1. Lustre Production Environment
2. Motivation
3. Software Architecture
4. Technical Details
5. Example for an IO-Task
6. Future Work

# Lustre Production Environment

## Clients

- ~1000 clients v. 2.6.92 but moving to v. 2.10
- Running on Debian Jessie

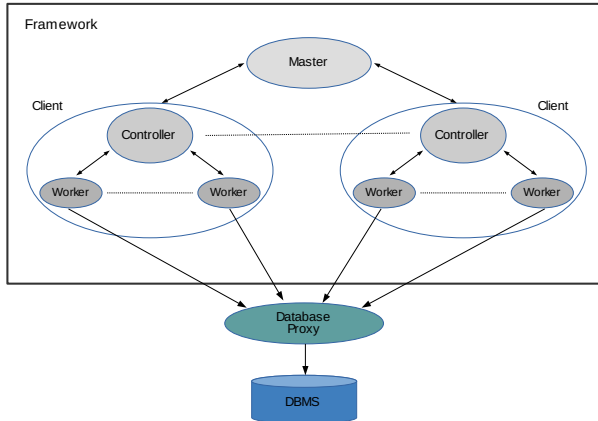
## Servers

- Total storage capacity of 14.7PB
- Pair of active/passive meta data server v. 2.5.3.90 with manual failover
- 78 file server v. 2.5.3.90 with ZFS v. 0.6.3
- 546 OSTs - 7 OSTs per one OSS
- Running on Debian Wheezy

# Motivation

- Monitoring the availability of the file servers
- Measuring IO performance per OSTs continuously
- Collecting measurement results for later analysis per OSS/OST
- Scheduling and execution of generic tasks

# Software Architecture



- Based on a master-client architecture
- Clients are divided into a controller with multiple workers
- Bottom-Up communication model via message passing

# The Master Component

- Creates tasks within a specific measure interval for all OSTs
- Schedules tasks to controller on demand when tasks are available
- Keeps track of scheduled tasks for rescheduling

# The Client Component

## Controller

- Creates a pool of workers
- Requests tasks from master
- Provides tasks to workers over a shared queue

## Worker

Responsible for executing tasks from the shared queue.

# Technical Details

Free available as open source project on GitHub at:

[https://github.com/GSI-HPC/lustre\\_task\\_driven\\_monitoring\\_framework](https://github.com/GSI-HPC/lustre_task_driven_monitoring_framework)

It is still under development...

## Mandatory Requirements

- Python Standard Library
- ZeroMQ for distributed messaging
- lctl from Lustre utils for determining OSTs and OSSs

## Optional Requirements for running Sample Task

- Python interface to MySQL (MySQLdb) / MySQL database server
- lfs from Lustre utils for setting file stripes



# Example for an IO-Task (1)

Measure interval is 15 minutes in this example.

## Task Implementation

A task implements an interface method of the generic task class.

1. Checks if OST is in active state for doing IO tests
2. Writes data in 1MB blocks to a target OST with a total of 8MB payload
3. Reads the file content block-wise from the target OST back
4. The measured metrics are pushed to the database proxy

## Collecting and Storing Measurements

- This is done outside and independently of the framework.
- A proxy buffers incoming messages and does bulk inserts into a database.

## Example for an IO-Task (2)

Simplified database table schema for storing IO measurements:

Field	Description
id	Primary key
read_timestamp	Timestamp for start of the read operation
write_timestamp	Timestamp for start of the write operation
ost	Target OST name
ip	IP address of the OSS
size	Total payload size in bytes
read_throughput	Average read throughput in bytes per seconds
write_throughput	Average write throughput in bytes per seconds
read_duration	Total read duration in seconds
write_duration	Total write duration in seconds

## Example for an IO-Task (3)

As a first step query the database for file server  
where write duration or read duration  $\geq 10$ :

Date	IP	max_write_dur_sec	max_read_dur_sec	count
2017-09-11	1.2.3.4	12	0	21
2017-09-13	1.2.3.4	20	0	17
2017-09-14	1.2.3.4	25	0	58
2017-09-15	1.2.3.4	17	0	26

Further investigation of the 15th of September 2017 for the file server can be done by more precise database query...

## Example for an IO-Task (4)

As a second step query the database for the date '2017-09-15' and IP '1.2.3.4' for the following information:

- Target OST
- Min and max timestamps(ts)/durations(dur)
- Count of IO measurements

OST	min_ts	max_ts	min_dur	max_dur	count
OST001f	13:37:56	14:23:07	5	17	4
OST0022	13:37:56	14:23:07	5	17	4
OST0021	13:37:56	14:23:07	4	17	4
OST0020	13:37:56	14:23:07	5	17	4
OST001c	13:37:56	14:23:07	5	17	4
OST001d	13:52:58	14:23:07	5	17	3
OST001e	13:52:58	14:23:07	5	17	3

# Future Work

- Task description language
- Creation of different tasks at runtime
- Providing a complete documentation

**Thank you!**

