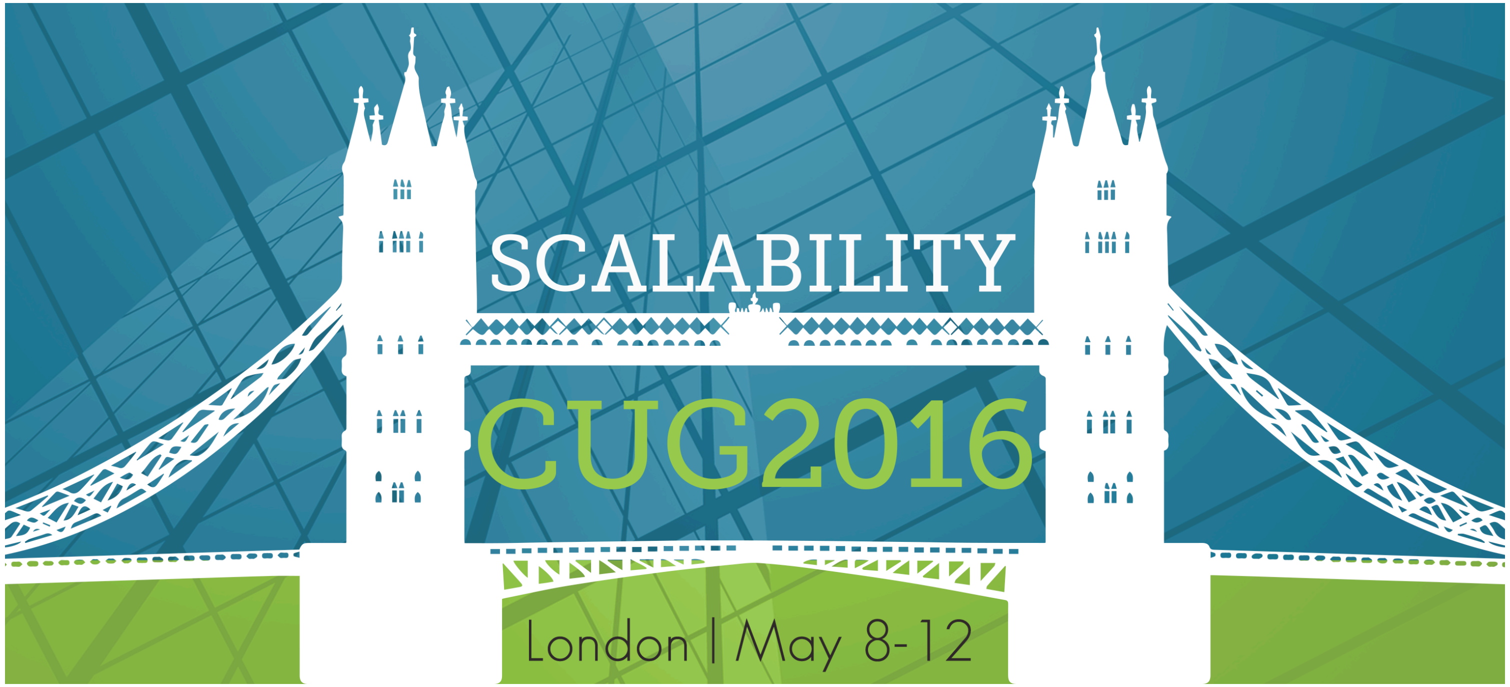
CSCS NERSC KAUST OAK RIDGE National Laboratory SEAGATE CRAY



# Jobs I/O monitoring for Lustre at scale

London, CUG2016

Matteo Chesi (CSCS), Tina Declerck (NERSC), Bilel Hadri (KAUST),
Jason Hill (ORNL), Torben Kling Petersen (Seagate) and Sven Trautmann (CRAY)

May 9th, 2016

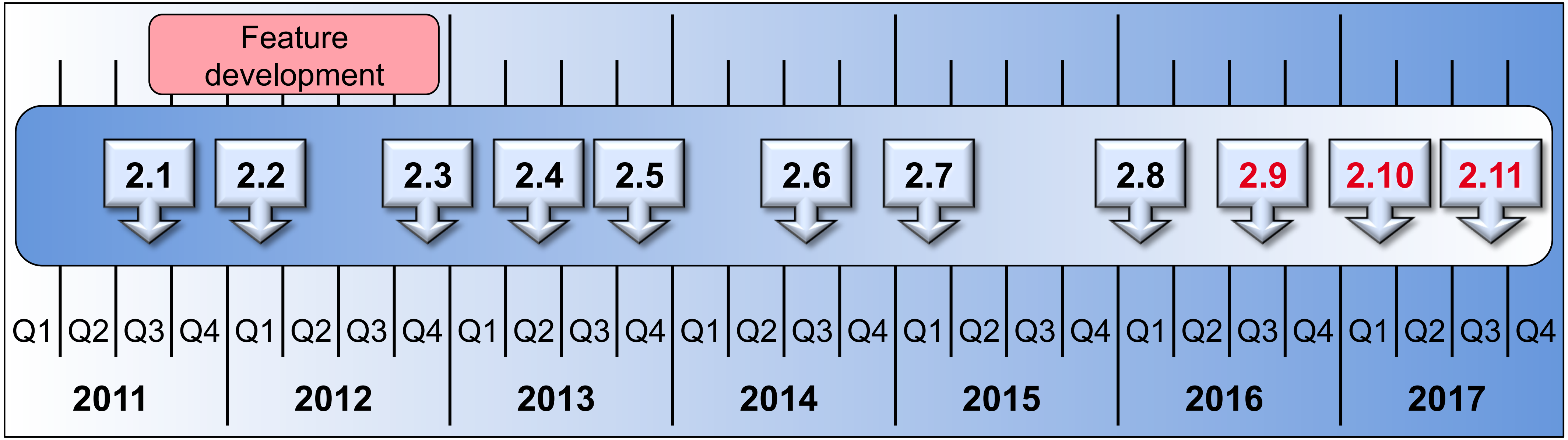
# Jobs I/O monitoring for Lustre

Is it slow ?

**Why ?**

- Parallel filesystems are **SHARED** resources in HPC clusters
- Lustre I/O performances are **USAGE DEPENDENT**
- Fast detection of disturbing workloads helps preventing major issues

# Lustre Jobs Stats at Scale


Feature development
2.1
2.2
2.3
2.4
2.5
2.6
2.7
2.8
2.9
2.10
2.11
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
2011
2012
2013
2014
2015
2016
2017

- Lustre Job Stats are not a brand new feature (LU-694)
- Using the feature on petascale systems is not trivial

# Lustre stats

- Environment: Lustre 2.5+ with 3 MDT, 26 OST, 7800-ish clients, no jobstats
- Client (total: >3k stats files)
  - `services/ldlm_cbd/stats`
  - osc, mdc, mgc : `osp/<type>/stats` + `ldlm/namespaces/<type>/pool/stats`
  - FS: `statahead_stats`, `stats_track_gid`, `stats_track_ppid`, `stats_track_pid`, `read_ahead_stats`, `stats`, `offset_stats`, `extents_stats_per_process`, `extents_stats`
  - osc: `stats`, `rpc_stats`, `osc_stats` + mdc: `stats`
- MDS (total: >15k stats files)
  - client: `stats` + `ldlm_stats` file
  - ldiskfs device: `stats` + `brw_stats`
  - osc, lwp, mdt, mgc: `osp/<type>/stats` + `ldlm/namespaces/<type>/pool/stats`
  - ldlm `services/ldlm_canceld/stats` + `services/ldlm_cbd/stats`
  - MDT `rename_stats`, `job_stats`, `md_stats`, `hash_stats`, and `site_stats`
  - MDS `mdt_fld/stats`, `mdt_seqm/stats`, `mdt_seqs/stats`, `mdt_out/stats`, `mdt_setattr/stats`, `mdt_readpage/stats`, `mdt/stats`

# Lustre stats (continued)

- OSS (total: >15k stats files)
  - client and OST: `stats` + `ldlm_stats` file
  - ldiskfs device: `stats` + `brw_stats`
  - ldlm `services/ldlm_canceld/stats` + `services/ldlm_cbd/stats`
  - lwp,mdt,mgc: `osp/<type>/stats` + `ldlm/namespaces/<osc>[/pool]/stats`
  - OST: `brw_stats`, `job_stats`, `stats`
  - OSS: `ost_out/stats`, `ost_seq/stats`, `ost_io/stats`, `ost_create/stats`, `ost/stats`
- in common
  - plain text format, easy to parse
  - same information is stored in different counters, some are easier to collect
  - counters can be reset by re-mount or manually
  - mapping stats to jobs can be done in some environments without jobstats
  - jobstats are a special case

# Lustre jobstats

- jobstats are available in Lustre 2.5+
- connect to your scheduling system (job ID environment variable)
- clients include job ID into Lustre traffic and server sums up requests
- beware, there may be bugs
    - LU-6659
    - LU-5179
- counters are not updated but the jobstats file gets another entry with each job
- jobstat information will be hold for a specific time (job_cleanup_interval)
- combining the jobstat information from all MDTs and OSTs will give you good data on what your applications are doing
- not very fine-grained information. May hide intense IO phases

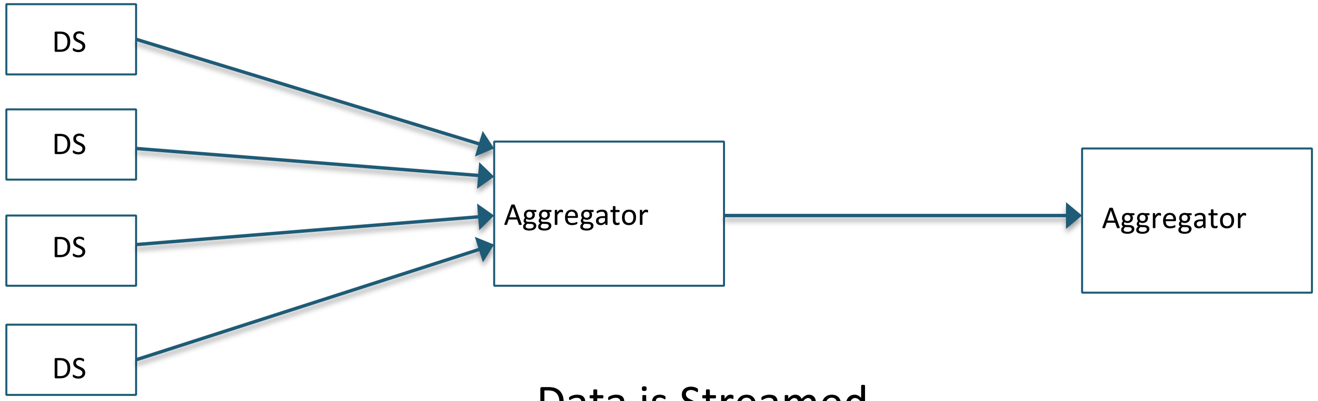
# Consuming Stats at Scale – Machine Interface

- **As established earlier, there are thousands of stats available, so**

**What Are Your Options?**

# Consuming Stats at Scale

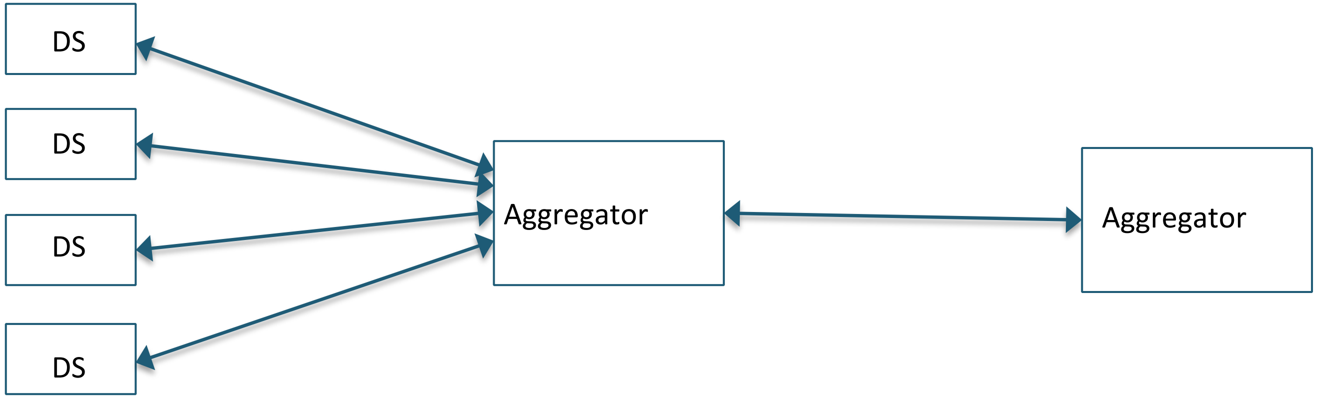
**PUSH?**


DS
DS
DS
DS
Aggregator
Aggregator

Data is Streamed

**or**

**PULL?**


DS
DS
DS
DS
Aggregator
Aggregator

Data is Polled

# Streaming Advantages & Disadvantages

- **Advantages:**
  - **Near immediate access to data**
  - **Low impact on data source (no requirement to store data in memory)**
  - **Open source software available**
- **Disadvantages:**
  - **Potential for loss of data if aggregator is not keeping up**

# Polling Advantages & Disadvantages

- Advantages
  - Potential for higher resolution data
  - Deterministic – could have less impact on data source
- Disadvantages
  - Aggregator must scale faster than required with push model
  - Either store or over-write data if aggregator isn't keeping up
  - Delay (potentially minor) in access to data
  - Requires specialized software (LDMS)

# The basic question:

- **How fast do you want to act on the data being collected?**

# What’s going on the Parallel File System ?

**Bilel Hadri**, Maciej Olchowik
KAUST Supercomputing Laboratory

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

# The Good, the Bad, and the Ugly

The Good

- Sonexion booted without problems after power blip (>70 OSSes down)
- No issue noticed by users, all their jobs were running fine !

The Bad

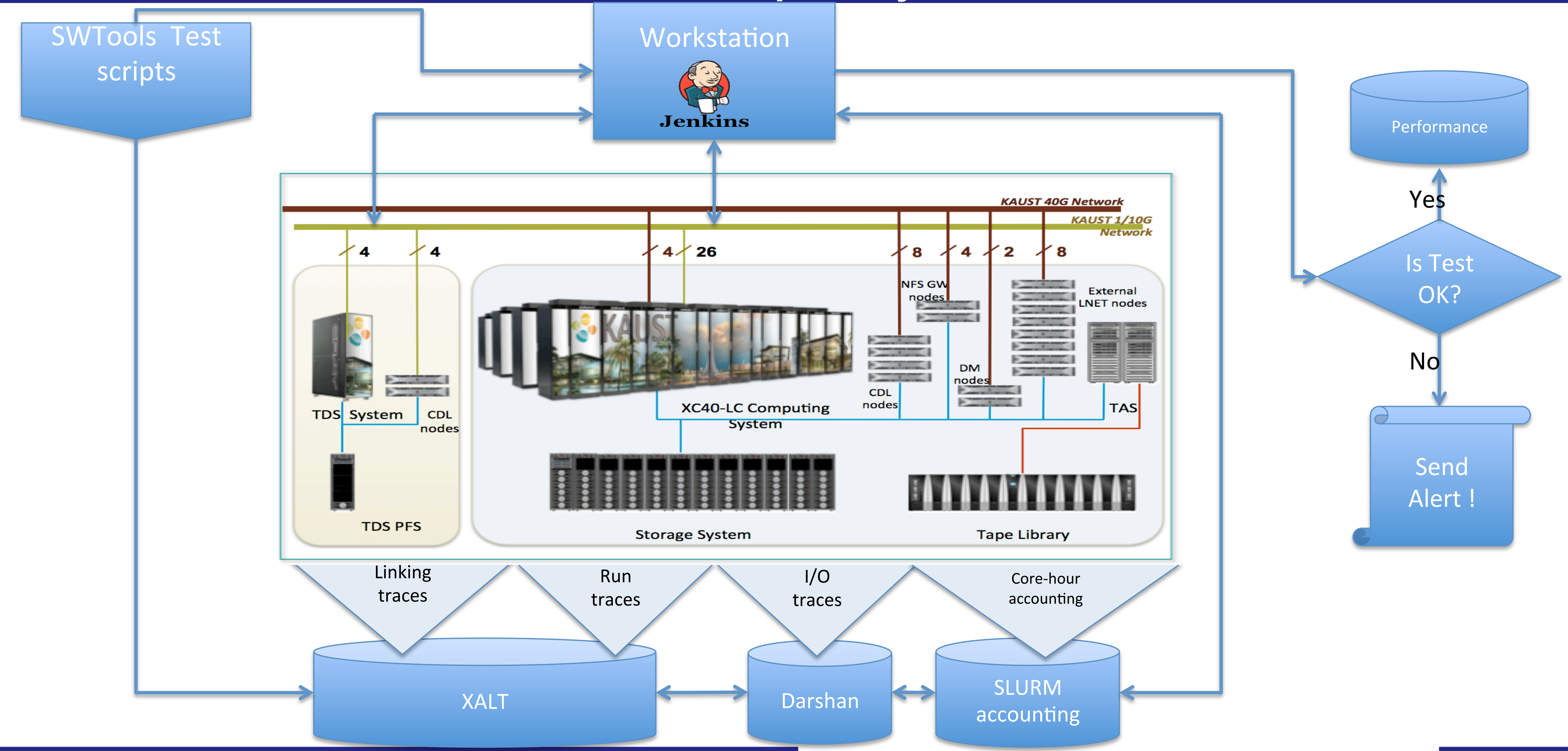
- Poor metadata performance (MDS crashing)
- Problems with handling small files

The Ugly

- User can crash Lustre however difficult to track it
- Need to correlate Cray/Sonexion/Sys-admin/CS data
  - What is .exe or a.out code ?

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

# Correlation of data from Workflow Complete cycle



SWTools Test scripts

Workstation

Jenkins

Performance

Yes

Is Test OK?

No

Send Alert !

Linking traces

Run traces

I/O traces

Core-hour accounting

XALT

Darshan

SLURM accounting

# Lustre slow

- Not again… Why it's slow ?
- Is it temporary or a sign of big trouble ?

- What's the right strategy ?
- Which applications ? Who is doing it ?
  - If multiple times, ban user/code temporarily?
  - fix his code/ patch software system

- Shaheen faced major disruptions with Lustre in early phase of installation
  - Too long to diagnostic and isolate the issue ( 3 days !!!)

# Who is responsible ?

- Vendor
  - Provide live alert not only to sys-admin( they already received a lot), but also to Scientific team and users
  - What's the right metric ?
    - Focus on real applications benchmarks
- HPC staff
  - Track in live which jobs are causing issues
  - Efficient correlation of data to target the right cause
  - Integration with other monitoring tools (nagios)
- Users
  - Better training, Do & **don't**
  - Know better their application
  - Use and check performance of code ( Darshan )

# Roadmap for better IO monitoring

- Most Urgent: Vendor Support
  - Lustre/Commercial development
  - HPC sites/Scientific community  involvement in Lustre development
  - Other products/technologies to include/consider

# Goals

- Open Source – community support
- Not just Lustre monitoring – same tools for all site performance monitoring
  - Larger community base
  - Easier to manage
- Pluggable and customizable
  - More than just Lustre – and not just filesystems
- Ideally no impact on the remote client
- Want instantaneous *and* historical data
  - Don’t just need right now
  - Analysis for patterns (human/machine) is important

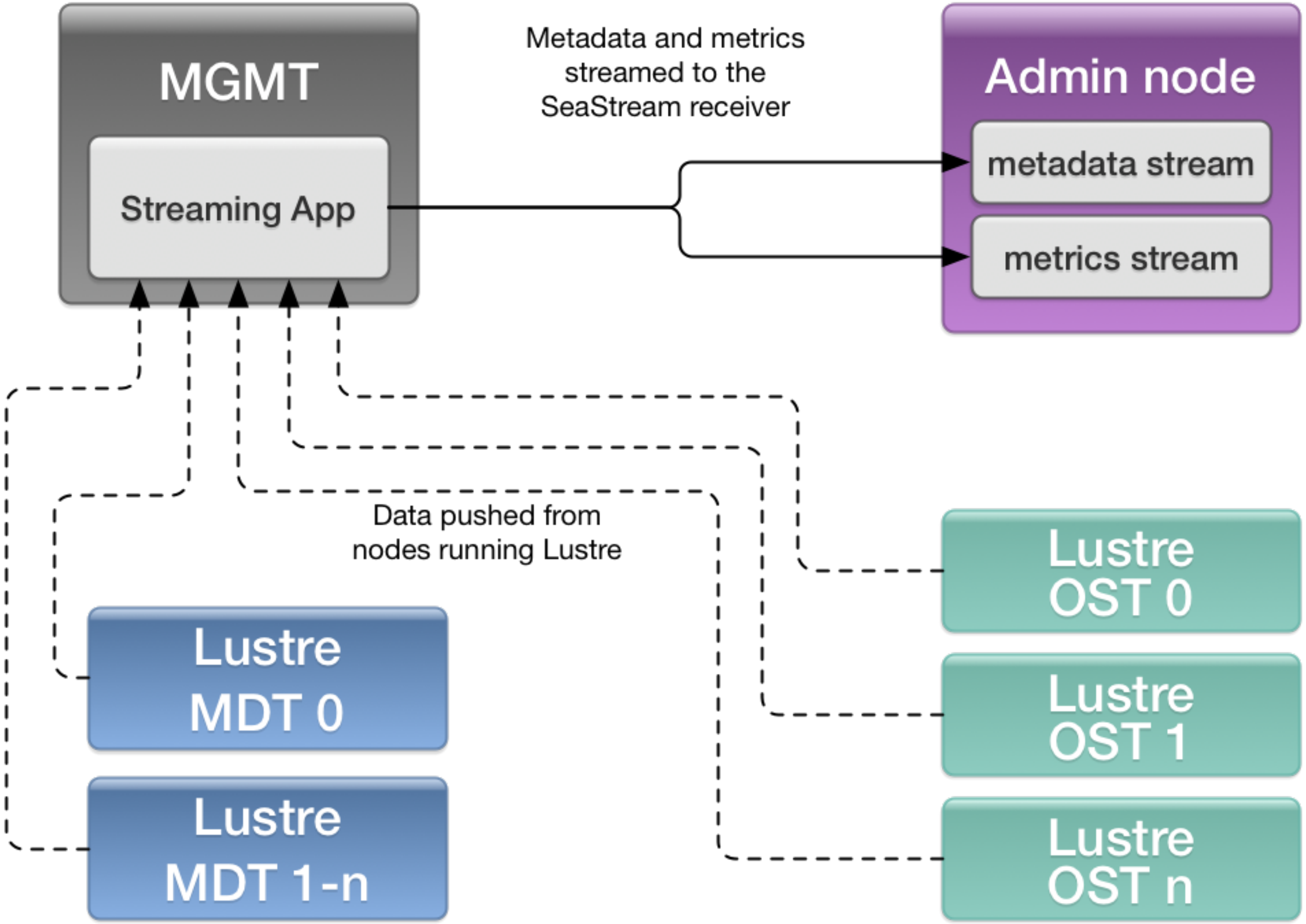
# Goals

- Visual Representation is often easier for finding trends
  - Ex. All jobs from a {user, project, domain, science, *}

# Jobstats in Clusterstor/Sonexion

- Jobstats in CSM
  - cscli based management
  - No historical data stored on CSM
  - No GUI based implementation (current POR)
  - User settable sampling frequency (15 to 600 sec)
  - Maintained under puppet control
- SeaStream API – ClusterStor Manager based export (streaming) API
  - REST streaming API (over https)
  - API forwards the data, not pull …
  - Sends only change information
  - Supports ClusterStor HA model

# Overall collection



MGMT

Streaming App

Metadata and metrics streamed to the SeaStream receiver

Admin node

metadata stream

metrics stream

Data pushed from nodes running Lustre

Lustre MDT 0

Lustre MDT 1-n

Lustre OST 0

Lustre OST 1

Lustre OST n

# New cscli commands

- Enable/Disable jobstats in ClusterStor

```
$ cscli jobstats collection --fs <fs-name> [--enable|--disable]
  [Enabling|Disabling] Lustre Job Statistics for <fs-name>
  Successfully [enabled|disabled] Lustre Job Statistics for <fs-name>
```

- Configuring jobstats in ClusterStor

```
$cscli jobstats configure --fs= <fs-name> --frequency=[15-600] --scheduler=[scheduler-
type]
  [Enabling|Disabling] Lustre Job Statistics for <fs-name>
  Successfully [enabled|disabled] Lustre Job Statistics for <fs-name>
```

- Show job status configuration in ClusterStor

```
$ cscli jobstats show
  <filesystem-name>: Enabled
    Frequency:    30 seconds
    Scheduler:    procname_uuid
```

Supported scheduler types:

| Job Scheduler | Environment variable |
|---|---|
| Simple Linux Utility for Resource Management | `SLURM_JOB_ID` |
| Sun Grid Engine (SGE) | `JOB_ID` |
| Load Sharing Facility (LSF) | `LSB_JOBID` |
| Loadleveler | `LOADL_STEP_ID` |
| Portable Batch Scheduler (PBS)/MAUI | `PBS_JOBID` |
| Cray Application Level Placement Scheduler (ALPS) | `ALPS_APP_ID` |

# Next steps and timeline …

- Development complete in June, 2016
  - Project started in February
  - Cray get full access (Cray timeline TBD)
- Tools to integrate and analyse data streams ??
  - Grafana, Splunk, others ??
- Additional QoS tools
  - Adaptive performance throttling ??

## Some Sources

- Daniel Kobras, Science and Computing, "Lustre – Finding the Filesystem Bottleneck", LAD 2012
- Lustre Wiki, http://wiki.lustre.org/Lustre_Monitoring_and_Statistics_Guide
- Roland Laifer, KIT, "Lustre tools for ldiskfs investigation and lightweight I/O statistics", LAD 2015
- many other LUG and LAD presentations.