Monitoring an heterogenous Lustre environment

LUG 2015

frederick.lefebvre@calculquebec.ca

Plan

Site description

A bit of history

Real time utilization monitoring

Moving forward

Who we are

Advanced computing center @Université Laval

Part of Calcul Québec & Compute Canada

Operate 2 Clusters & 4 Lustre fs

Our infrastructure

1000+ lustre clients (2.5.3 & 2.1.6)

On 2 IB fabrics - 6 LNET routers

4 Lustre file systems (~2.5 PB)

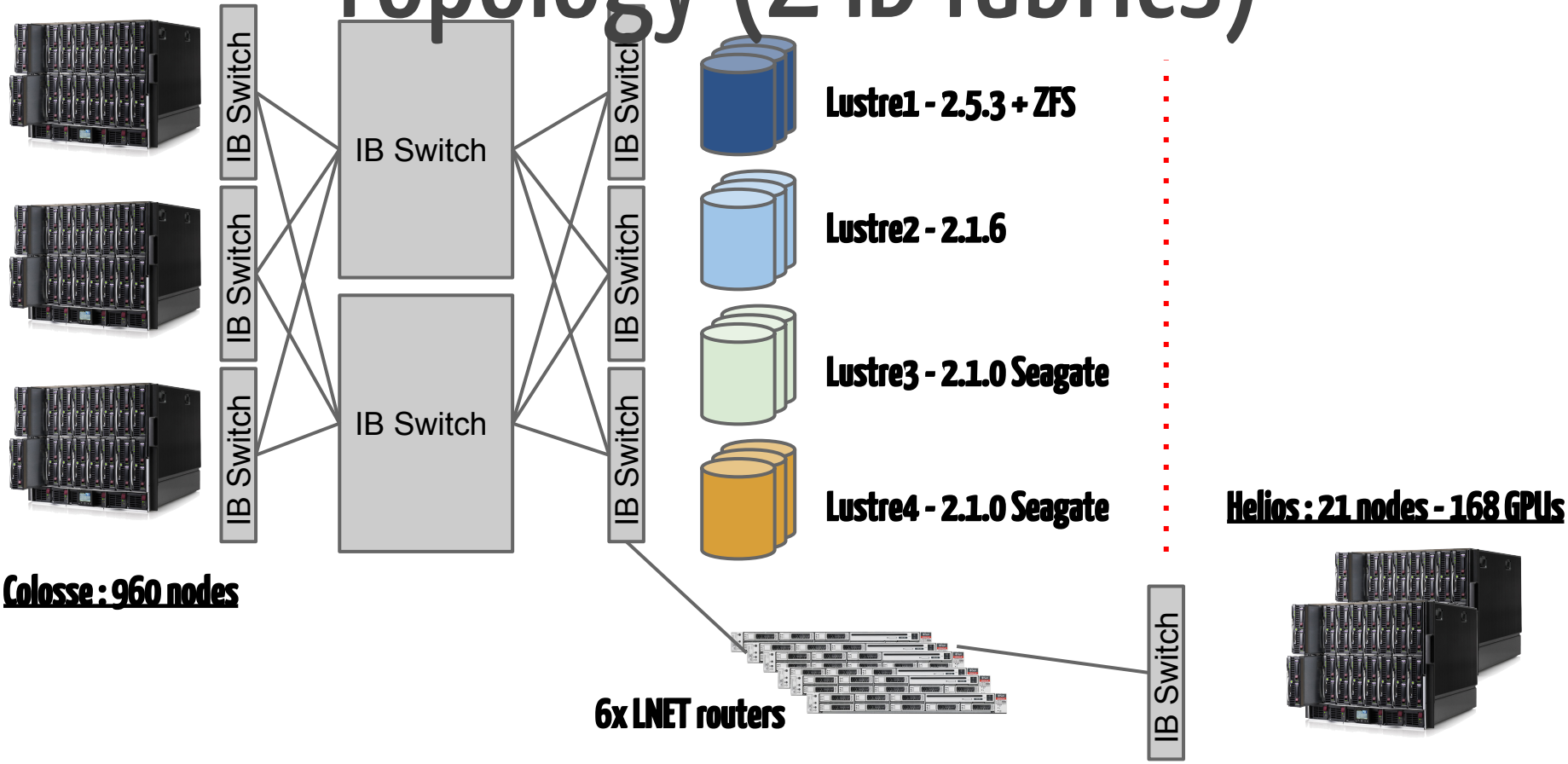
500 TB - Lustre 2.5.3 + ZFS - 16 OSSs

500 TB - Lustre 2.1.6 - 6 OSSs

1.4 PB - Lustre 2.1.0 (Seagate) - 12 OSS

120 TB - Lustre 2.1.0 (Seagate) - 2 OSS

Topology (2 IB fabrics)



An history of monitoring

We initially monitored because

- We knew we should
- We wanted to identify failures

Tried multiple options over the years

- collectl
- cerebro +lmt

An history of monitoring (cont.)

Converged towards custom gmetrics scripts for Ganglia

- Monitored IB
- Lustre IOPS and throughput

An history of monitoring (cont.)

Converged towards custom gmetrics scripts for Ganglia

- Monitored IB
- Lustre IOPS and throughput

Worked reasonably well

- But quality of our scripts was inconsistent
- Some used a surprising amount of resources when running

Issues

Users and staff keep asking questions

- Why is it so long navigating my files ?
- Why is my app so slow at writing results ?
- Which filesystem should I use ?

We had most of the data “somewhere” but searching for it was always painful

Issues (cont.)

Data is everywhere

- Performance data from compute node is in mongoDB
- Lustre client and server data in Ganglia (RRD)
- Scheduler data in text log files

Easy to end up with a spiderweb of complex script to integrate all data

New reqs

1. Give staff a near-realtime view of the filesystems utilisation and load

2. Provide end-users with data to identify their utilisation of the Lustre filesystems
   a. Bonus point for cross-referencing with the scheduler's stats

New reqs (cont.)

We want to keep “historical” data for future analysis and comparison

With the amount of storage now available, it didn’t make sense to keep losing granularity over time with RRD.

Enter Splunk

We have been using Splunk to store and visualize system logs and energy efficiency data.

It's well suited to indexing a lot of unstructured data

Plan

Collect data from /proc on all nodes and send it to the system's log with logger

The logs are indexed and visualized with Splunk

- We use a separate index for performance data

Data collection

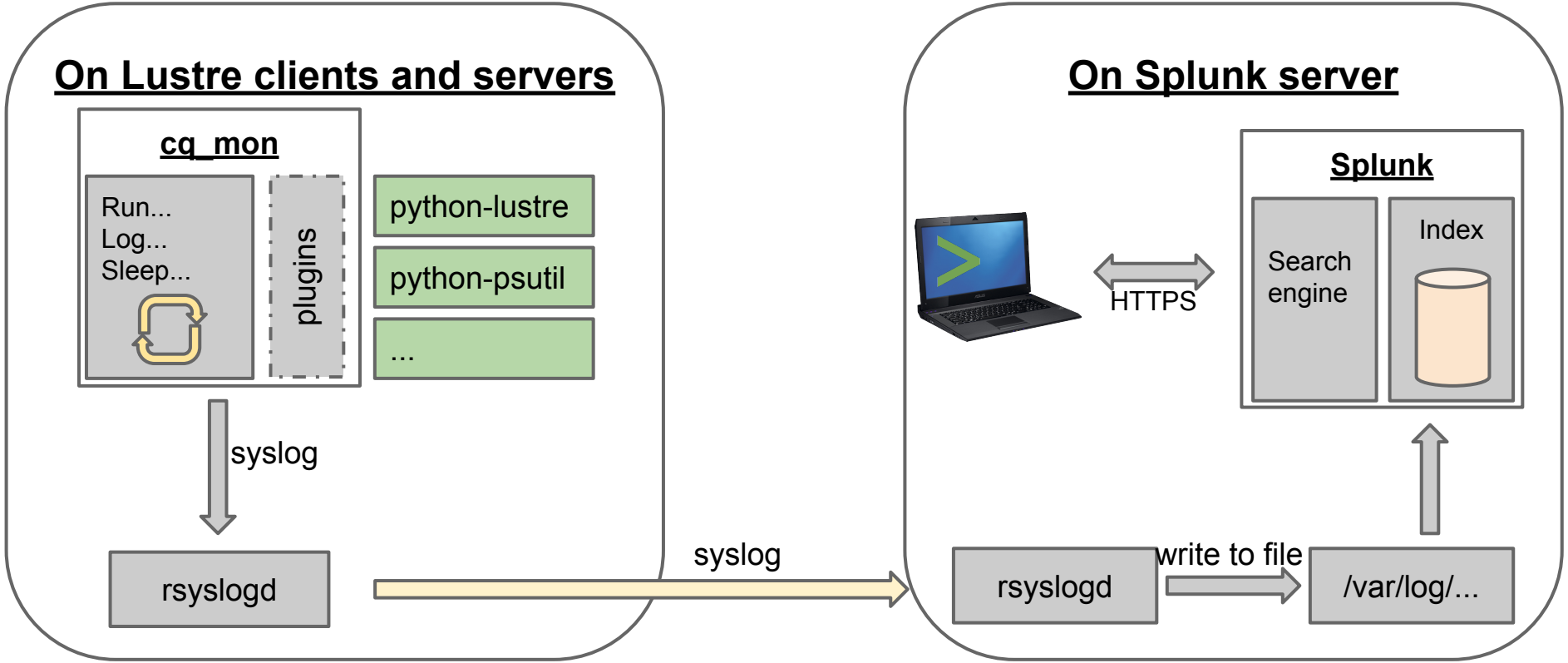
Existing options to monitor Lustre tend to be tied to a specific backend (RRD or DB)

More flexibility is needed

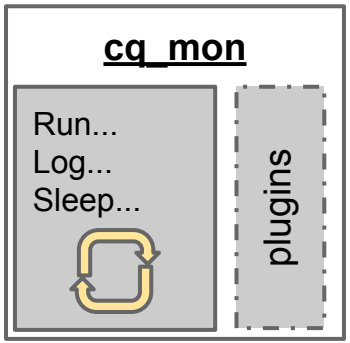
We decided to write another one

We could have modified an existing application

Overview


On Lustre clients and servers
cq_mon
Run...
Log...
Sleep...
plugins
python-lustre
python-psutil
...
syslog
rsyslogd
syslog
On Splunk server
Splunk
Search engine
Index
HTTPS
rsyslogd
write to file
/var/log/...

cq_mon

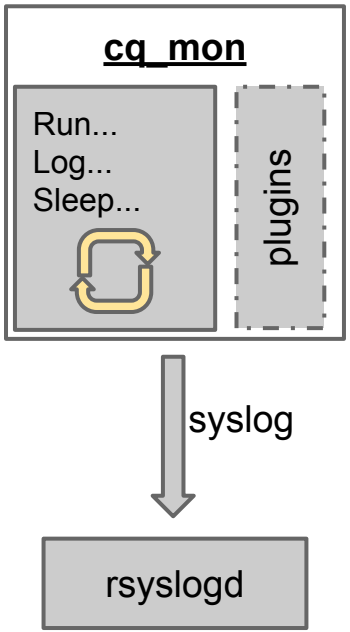


Writen in Python

Configuration specifies which plugins to run

Runs each plugin at interval and log the results as ‘key=values’ to syslog

cq_mon



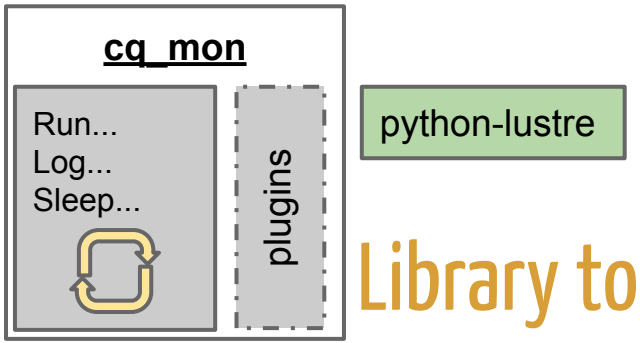
Writen in Python

Configuration specifies which plugins to run

Runs each plugin at interval and log the results as ‘key=values’ to syslog

Add syslog filter so as not to ‘polute’ system logs with performance data

python-lustre_util



Library to interact with lustre through /proc

Loads stats for different Lustre components make them available through an appropriate object (ie: lustre_mdt, lustre_ost, etc)

python-lustre_util (cont.)

Work in progress

- Tested with Lustre 2.1.6, 2.4.x, 2.5.x+ZFS and 2.1.0-xyratex
- Available for download through Github: https://github.com/calculquebec

Offer stats for:

- MDS, OSS, MDT, OST
- Client stats from the client and not the OSS/MDS

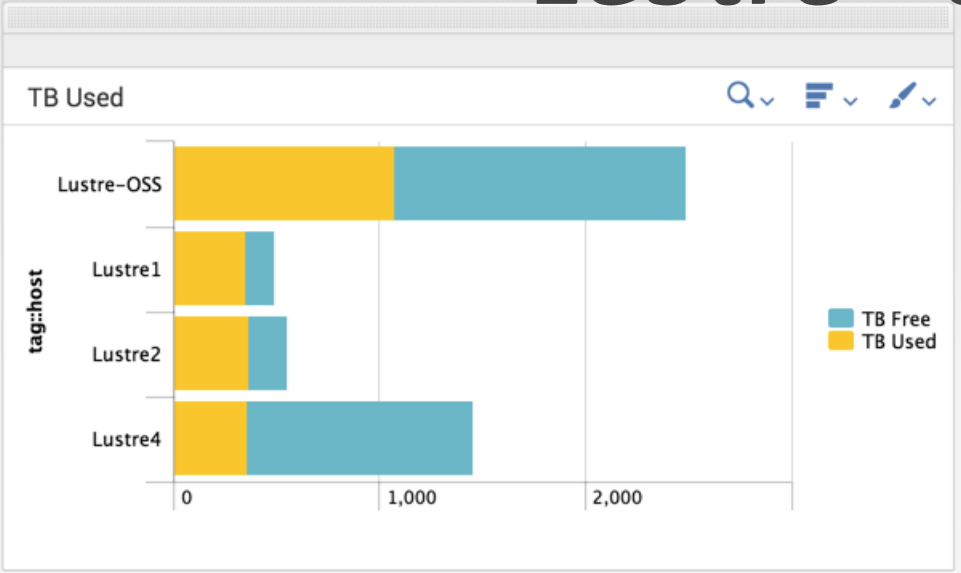
Splunk dashboards

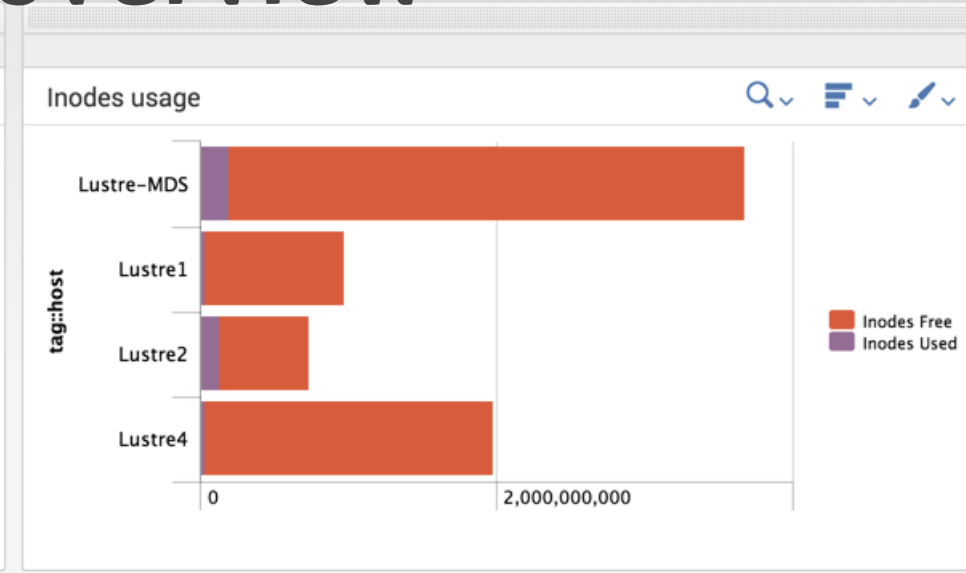
Put together custom dashboards for our needs

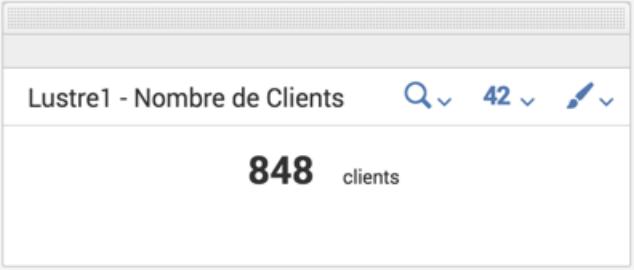
- Lustre filesystems overview
- 1 detailed dashboard for each Lustre filesystem
- Job report

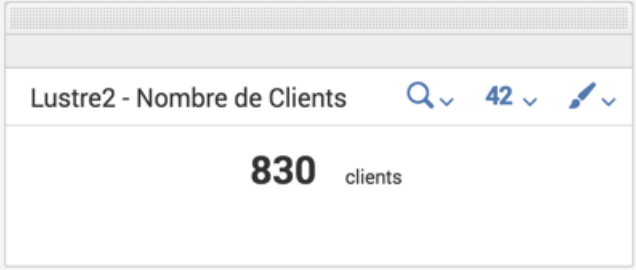
Some reports are slower than we would like

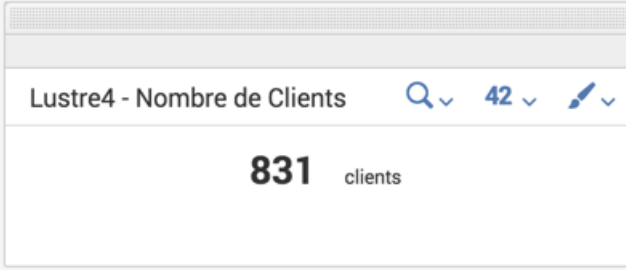
Lustre - overview



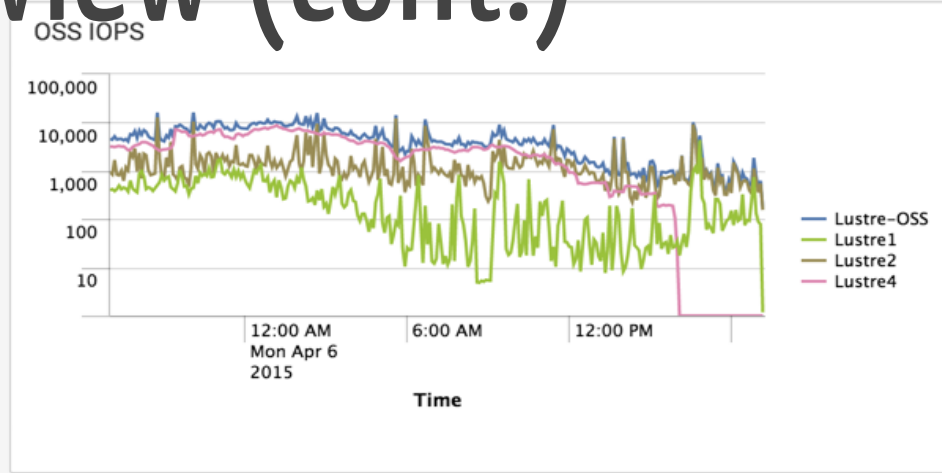


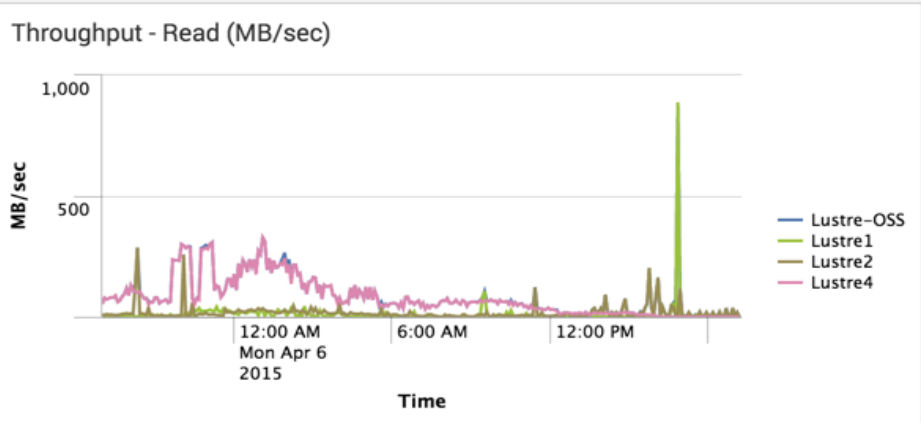


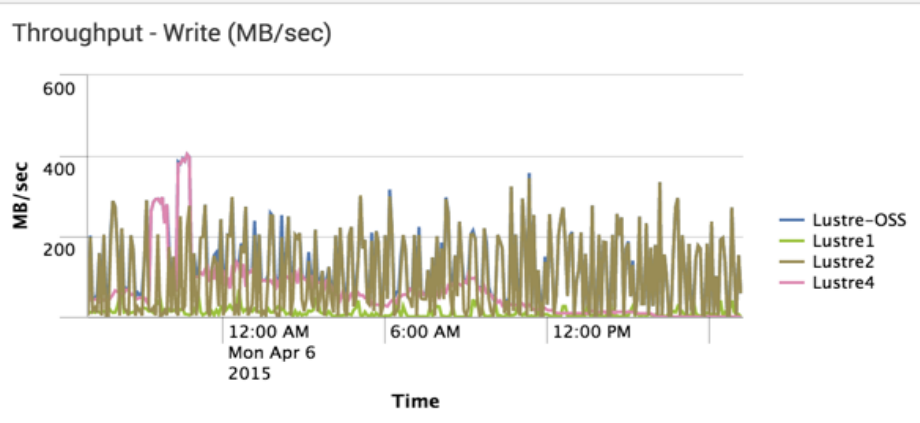




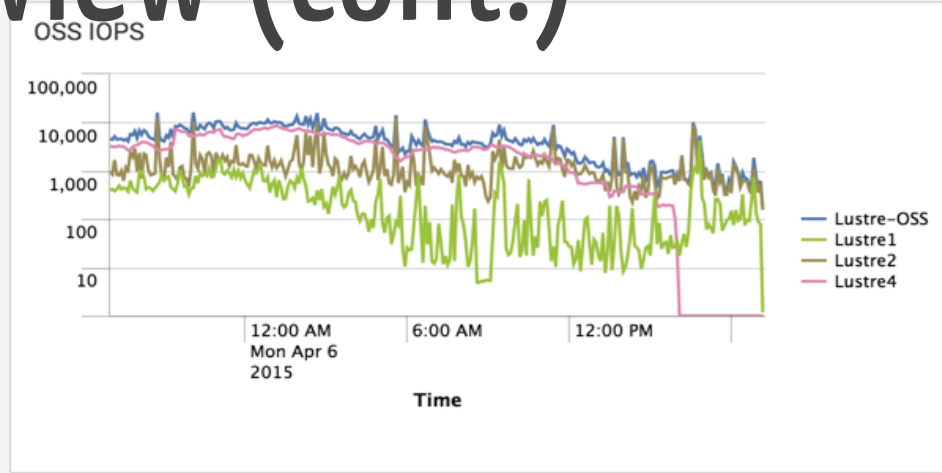
Lustre overview (cont.)

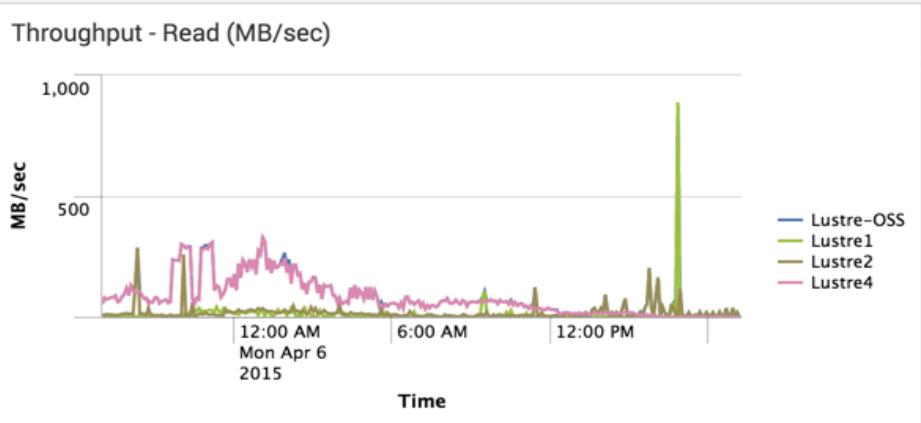


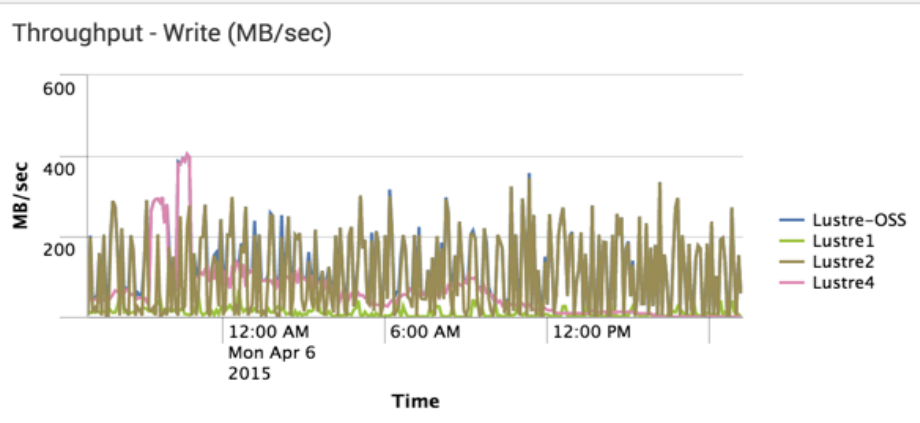




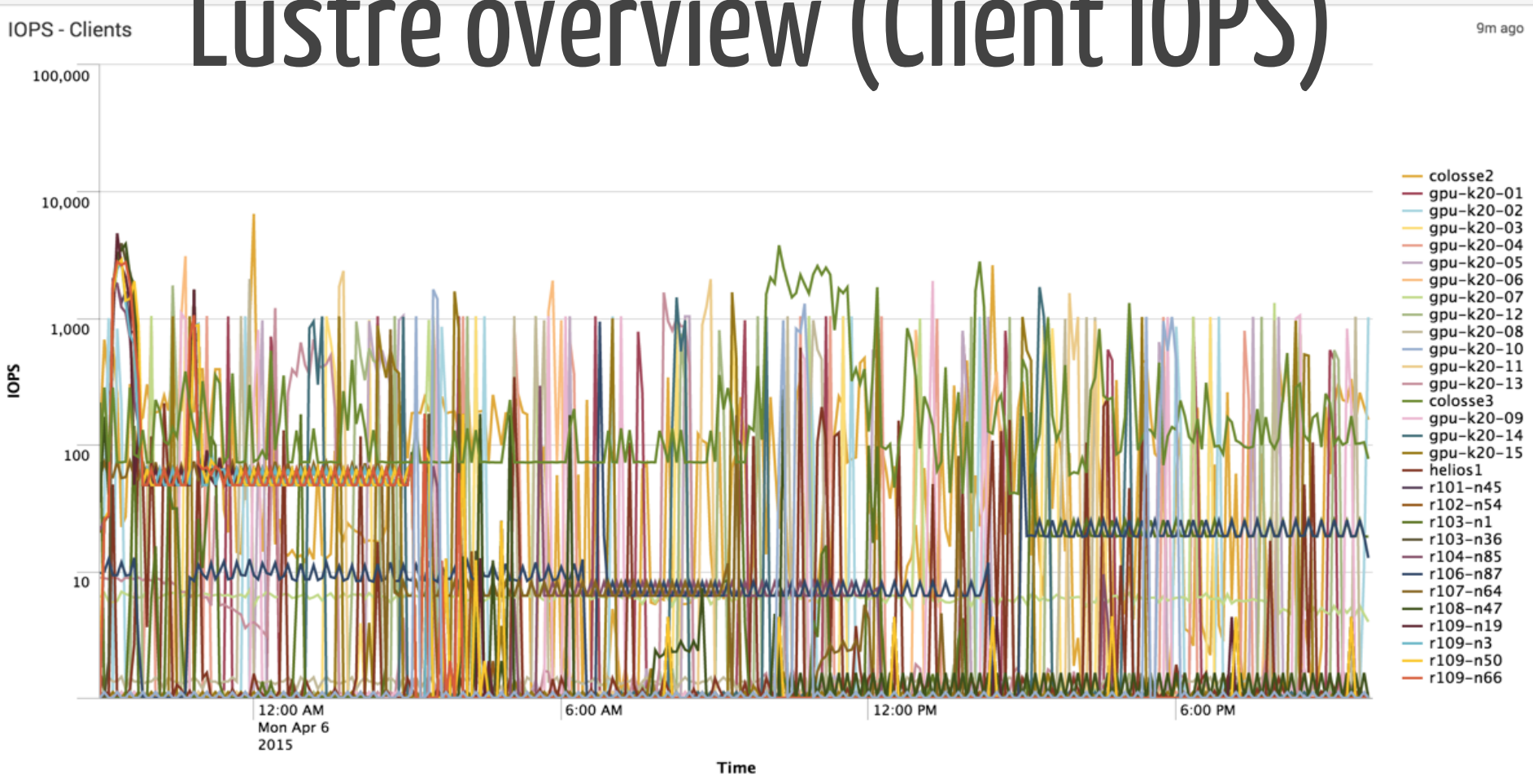
Lustre overview (cont.)



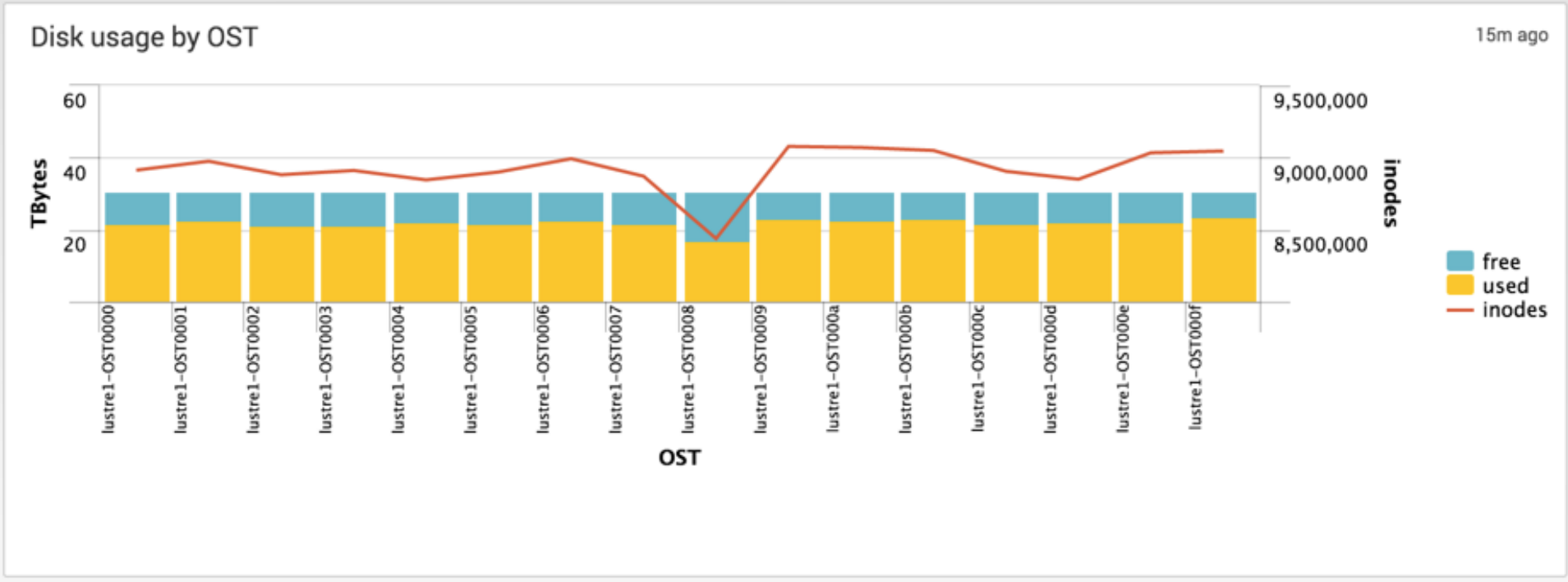




Lustre overview (Client IOPS)



Per FS view (Utilization)

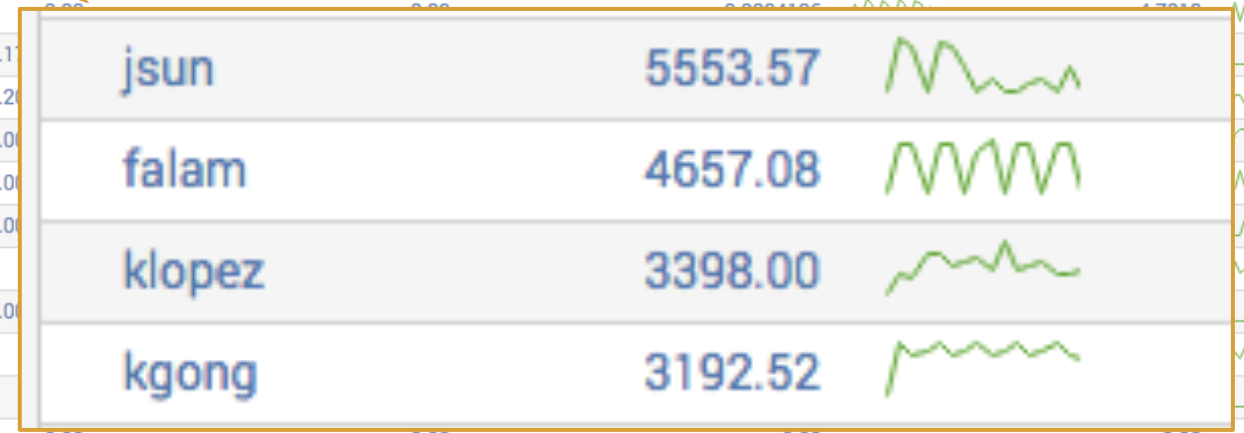


Per FS view (top users)

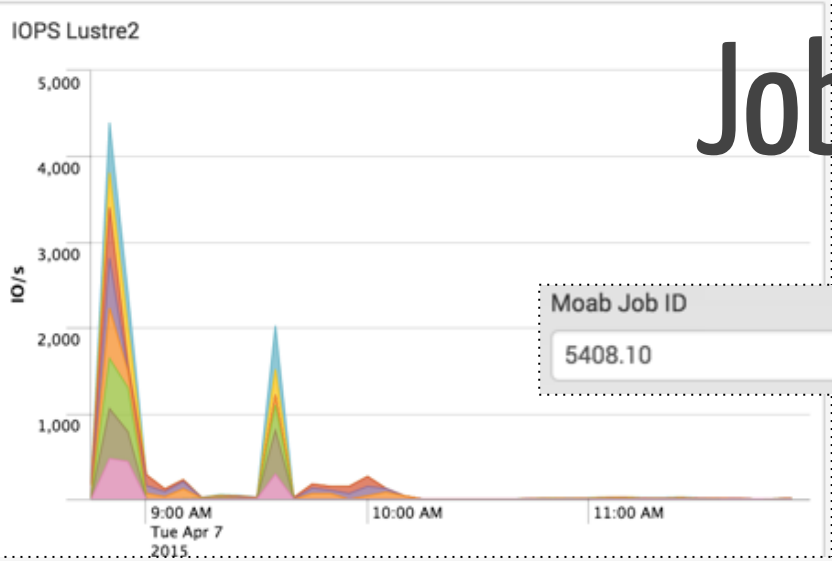
Top users - last 15 min

Real

username	IO load	iops	read MBps	read MBps s	read sec	read sec s	write MBps	write MBps s	write sec	write sec s
	18596		7.69		90.21		0.85		349.07	
jsun	5553.57		0.4606641		26.3926		0.0284726		5.9855	
falam	4657.08		0.4312121		23.4105		0.0137755		0.3344	
klopez	3398.00		0.0006615		0.045		0.00		0.0090	
kgong	3192.52		0.0000059		0.0063		0.00		26.6633	
etroc2	770.69		0.0755330		3.7519		0.0440753		1.2453	
zhangl	632.30		0.00		0.00		0.00		5.2852	
yzhou	156.94		1.345005		1.307		0.0003395		62.1898	
sghomesh	70.62									
mboisson	64.60		5.1							
gubea125	49.78		0.2							
zjendi	32.24		0.0							
jpicard	10.71		0.0							
marcandr	6.08		0.0							
rolina	0.76									
cneale	0.20		0.0							
afossi	0									
aj83310	0									
aldi	0		0.00		0.00		0.00		0.00	



Job report

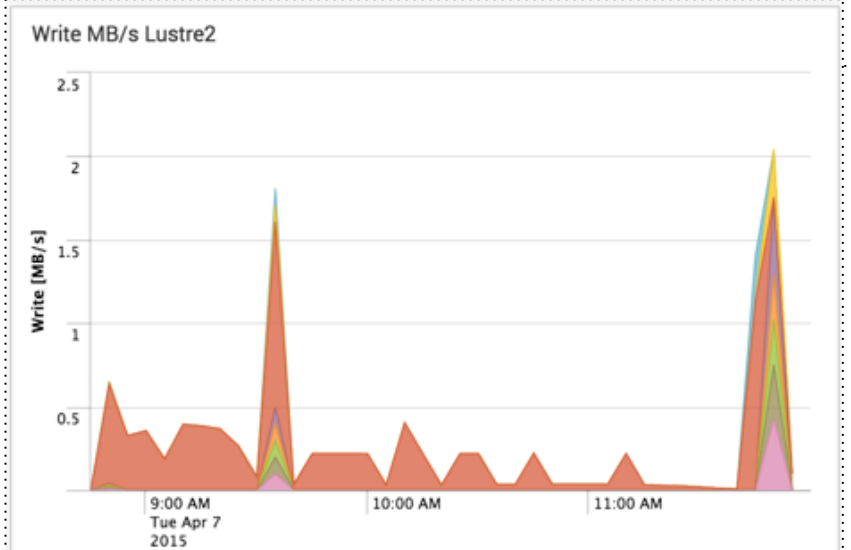


Moab Job ID: 5408.10

Cluster: Helios

Modules used

module
StdEnv
apps/buildtools/20140527
apps/moab
apps/ray/2.3.0
compilers/intel/14.0
mpi/openmpi/1.6.5



Limitations

The quantity of data collected is ‘scary’

- Especially true for clients (~1MB/client per day)

Impact of running extra codes on computes nodes

Still uncertainty on how to monitor or visualize some components

- ie: LNET routers, failovers, DNE setups, RPCs

Looking ahead

Add support for multiple MDS/MDT

Collect client's data on the servers

Cross-reference statistics with data from RobinHood DB

Make the code more resistant to failovers

Thank you

frederick.lefebvre@calculquebec.ca