Monitoring the Lustre* file system to maintain optimal performance

Gabriele Paciucci, Andrew Uselton

Outline

- Lustre* metrics
- Monitoring tools
- Analytics and presentation
- Conclusion and Q&A

Why Monitor Lustre*?

Continual monitoring provides an indication of system health.

Very high values for various metrics show when the file system is under load.

Even in the absence of user complaints there may be faults or degraded performance that can be addressed proactively.

When a problem is reported the data being gathered can help in diagnosis.

Outline

- Lustre* metrics
- Monitoring tools
- Analytics and presentation
- Conclusion and Q&A

Metrics

Bulk I/O rate

```
# ls -1 /proc/fs/lustre/obdfilter/
num_refs
scratch-OST0000
scratch-OST0001
```

```
# llstat -g /proc/fs/lustre/obdfilter/scratch-OST0000
/usr/bin/llstat: STATS on 09/03/13 /proc/fs/lustre/obdfilter/scratch-OST0000/stats on 192.168.2.21@tcp
snapshot_time        1378243939.365733
read_bytes           336107
write_bytes          234181
get_info             2122
set_info_async       2
connect              6
disconnect           1
...
```

Number of bytes read or written since boot or last reset

Metrics

Bulk I/O rate (continued)

```
# llstat -g -i 1 /proc/fs/lustre/obdfilter/scratch-OST0000
...
0    read_bytes_rq 0 0 336107 [reqs]   read_bytes 0 0.00 220986105856 ...
1    read_bytes_rq 0 0 336107 [reqs]   read_bytes 0 0.00 220986105856 ...
...
```

Output suitable for graphing

Metrics

Bulk I/O rate (continued)

- llstat queries the ‘stats’ /proc file so the actual file name is optional
- llstat –g <path> : one snapshot
- Equivalent to: cat <path>/stats
- llstat –i 2 <path> : a snapshot every two seconds with differentials (rates)
- llstat –c <path> : reset the counters (clear them)

Metrics

OSS activity

```
# llstat /proc/fs/lustre/ost/OSS/ost/stats
/usr/bin/llstat: STATS on 09/03/13 /proc/fs/lustre/ost/OSS/ost/stats on 192.168.2.21@tcp
snapshot_time           1378243036.805175
req_waittime            602959
req_qdepth              602959
req_active              602959
req_timeout             602959
reqbuf_avail            1453522
ldlm_glimpse_enqueue    532
ldlm_extent_enqueue     1865
ost_setattr             4
ost_create              73
ost_destroy             2380
...
```

Metrics

CHANGED in 2.4

OST “back-end” statistics

```
# cat /proc/fs/lustre/obdfilter/scratch-OST0000/brw_stats
snapshot_time:        1378246506.227395 (secs.usecs)

                           read      |    write
pages per bulk r/w       rpcs  % cum % | rpcs     % cum %
1:                     125018  38  38 |   5   0   0
2:                         21   0  38 |   2   0   0
4:                         47   0  38 |   0   0   0
8:                         41   0  38 |   0   0   0
16:                        39   0  38 |   3   0   0
32:                       113   0  38 |   3   0   0
64:                       138   0  38 |   4   0   0
128:                      449   0  38 |  14   0   0
256:                   198698  61 100 |234142  99 100
...
```

Metrics

OST "back-end" statistics (continued)

There are seven histograms in brw_stats

- **pages per bulk r/w:** Did the RPC come in with 1MB of data?
- **discontiguous pages:** Is the above 1MB contiguous in the file
- **discontiguous blocks**: Is the above 1MB contiguous on disk?
- **disk fragmented I/Os**: How often is it discontiguous?
- **disk I/Os in flight**: How much parallelism in the disk activity?
- **I/O time**: How long do individual I/Os take?
- **disk I/O size**: How many I/Os of each size range are there?

Metrics

CHANGED in 2.4

MDT statistics

```
# llstat /proc/fs/lustre/mdt/scratch-MDT0000/md_stats
/usr/bin/llstat: STATS on 09/12/13 /proc/fs/lustre/mdt/scratch-MDT0000/md_stats on 192.168.2.201@o2ib
snapshot_time         1378979155.736074
open                  23658911
close                 20804619
mknod                 23923
link                  209
unlink                17823891
mkdir                 16740332
rmdir                 16596811
rename                1770750
getattr               53041694
setattr               7515845
getxattr              3124526
setxattr              10835
statfs                181
sync                  3218
samedir_rename        1762931
crossdir_rename       7819
```

Metrics

MDS statistics

```
# llstat /proc/fs/lustre/mds/MDS/mdt/stats
/usr/bin/llstat: STATS on 09/03/13 /proc/fs/lustre/mds/MDS/mdt/stats on 192.168.2.11@tcp
snapshot_time          1378249371.898363
req_waittime           724865
req_qdepth             724865
req_active             724865
req_timeout            724865
reqbuf_avail           1779535
ldlm_ibits_enqueue     33956
mds_getattr            510
mds_getattr_lock       346
mds_connect            11
...
```

Metrics

There are many others. What you monitor will depend on what you want to know and on what you think the problems are you need to investigate. What you want to measure will also guide your choice of tools for collecting, analyzing, and presenting the data.

Outline

- Lustre* metrics
- Monitoring tools
- Analytics and presentation
- Conclusion and Q&A

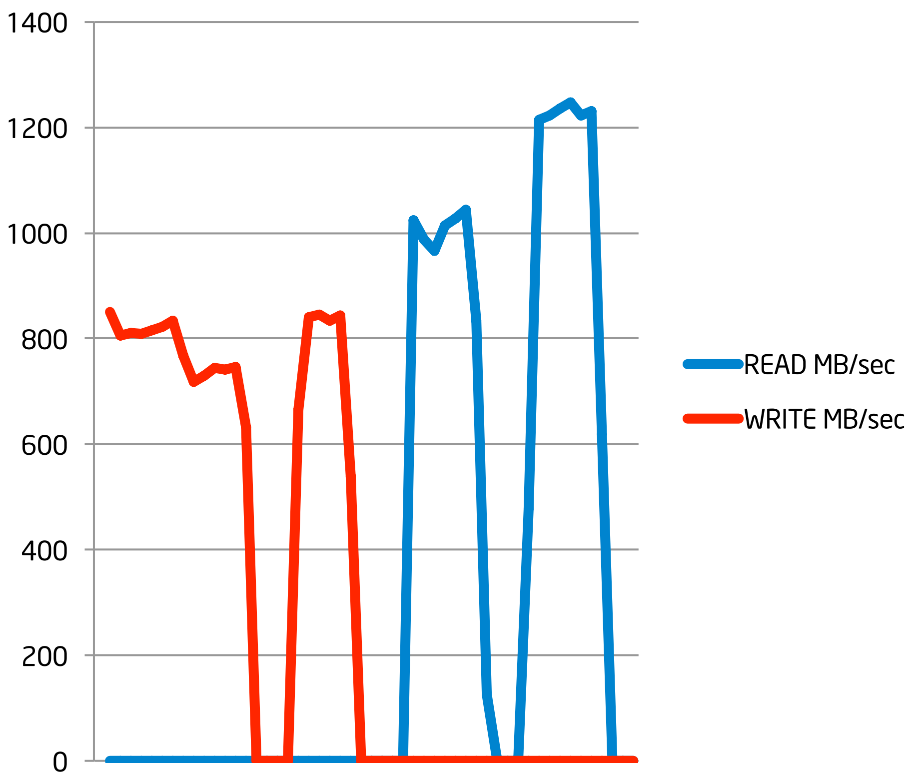
Tools

Plot-llstat parses and graphs the output of llstat using gnuplot, and generates .dat files for use in spreadsheets.

```
# llstat -i2 -g -c lustrefs-OST0000 >/tmp/log

# plot-llstat /tmp/log 3

# gnuplot /tmp/log.scr
```

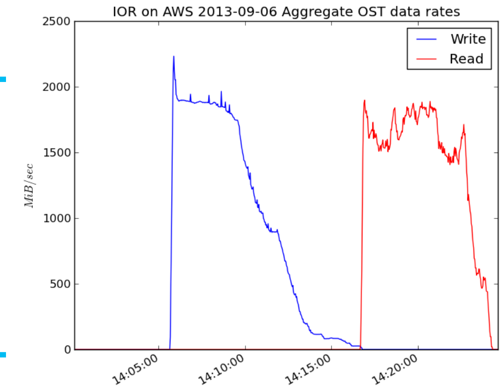


Tools (continued)

python/matplotlib

- Matplotlib.org
- matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell

```
plt.xlabel('time')
plt.ylabel(r'$MiB/sec$')
plt.setp( ax.get_xticklabels(), rotation=30, horizontalalignment='right')
plt.title("%s on AWS %s Aggregate OST data rates" % (self.application, dayStr))
plt.legend()
plt.savefig(plot)
plt.cla()
```



collectl

- collectl.sourceforge.net
- CollectL is a tool that can be used to monitor Lustre. You can run CollectL on a Lustre system that has any combination of MDSs, OSTs and clients. The collected data can be written to a file for continuous logging and played back at a later time. It can also be converted to a format suitable for plotting.

```
[oss]# collectl -scdl -i 3
#<--------CPU--------><----------Disks-----------><---------Lustre OST--------->
#cpu sys inter  ctxsw KBRead  Reads KBWrit Writes  KBRead  Reads  KBWrit Writes
  19  19 1930    563     0      0  27211   251      0      0  28701    28
   9   8 1346    239     0      0  17269   165      0      0   9225     9

[client]# collectl -sl --lustopts R -oTm
#            <---------------Lustre Client--------------->
#Time         KBRead  Reads  KBWrite Writes  Hits Misses
12:20:50.003   17138     8    12854    13   4100     0
12:20:51.002   18450     9    20500    20   4349     0
12:20:52.003   32735    16    20460    20   8447     0
```

LMT

- github.com/chaos/lmt/wiki
- The Lustre Monitoring Tool (LMT) monitors Lustre File System servers (MDT, OST, and LNET routers). It collects data using the Cerebro monitoring system and stores it in a MySQL database. Graphical and text clients are provided which display historical and real time data pulled from the database.

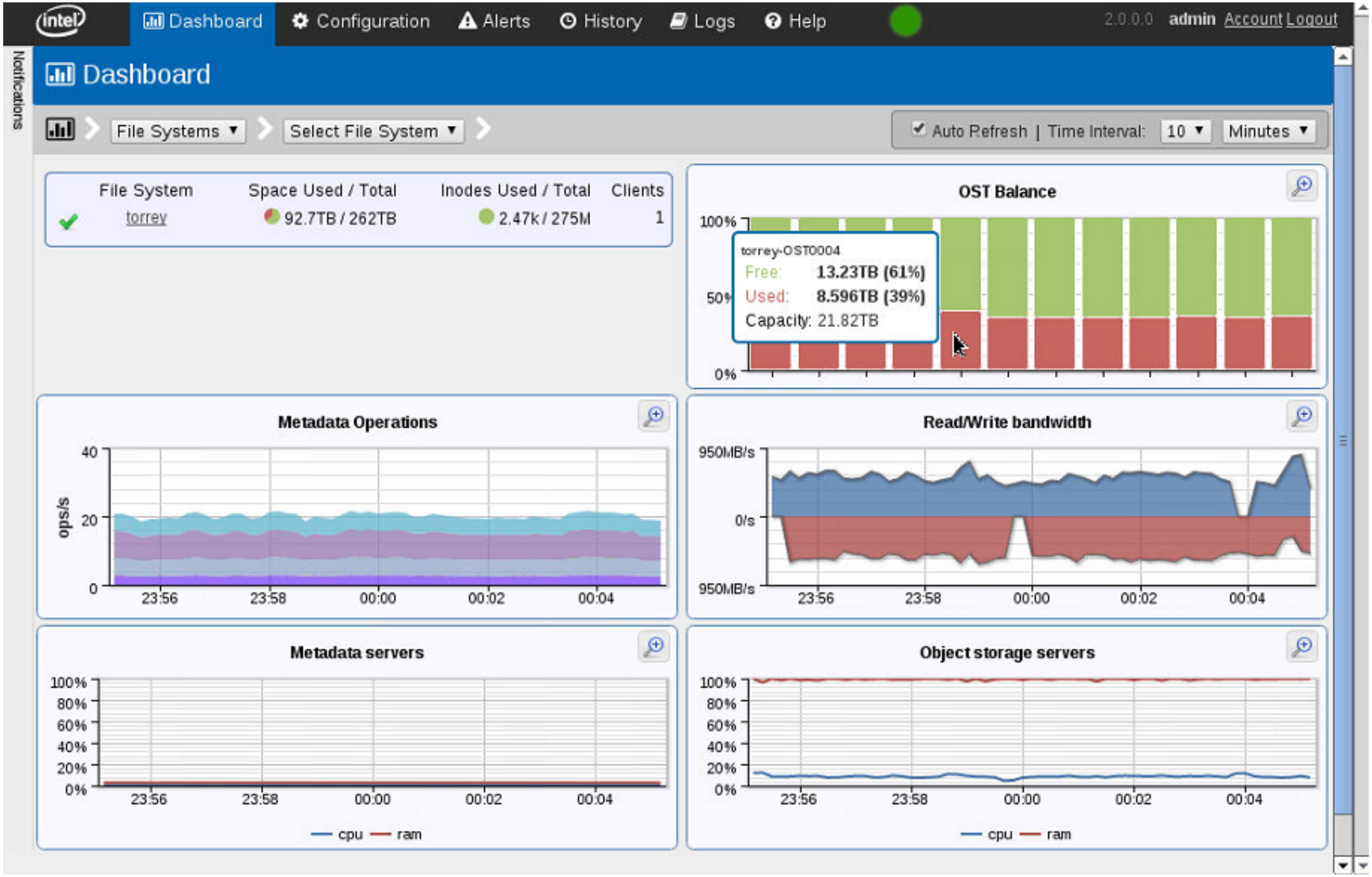
```
Filesystem: scratch
    Inodes:    160.000m total,      0.000m used (  0%),    160.000m free
     Space:      1.245t total,      0.014t used (  1%),      1.232t free
   Bytes/s:      0.000g read,       0.000g write,                0 IOPS
   MDops/s:      0 open,        0 close,       0 getattr,        0 setattr
                 0 link,        0 unlink,      0 mkdir,          0 rmdir
                 0 statfs,      0 rename,      0 getxattr
>OST S        OSS    Exp   CR rMB/s wMB/s  IOPS   LOCKS  LGR  LCR %cpu %mem %spc
0000         oss0      4    0     0     0     0       0    0    0    0    8    1
0001         oss1      4    0     0     0     0       0    0    0    0    8    1
0002         oss2      4    0     0     0     0       0    0    0    0    8    1
0003         oss3      4    0     0     0     0       0    0    0    0    8    1
0004         oss0      4    0     0     0     0       0    0    0    0    8    1
0005         oss1      4    0     0     0     0       0    0    0    0    8    1
```

Tools (continued)

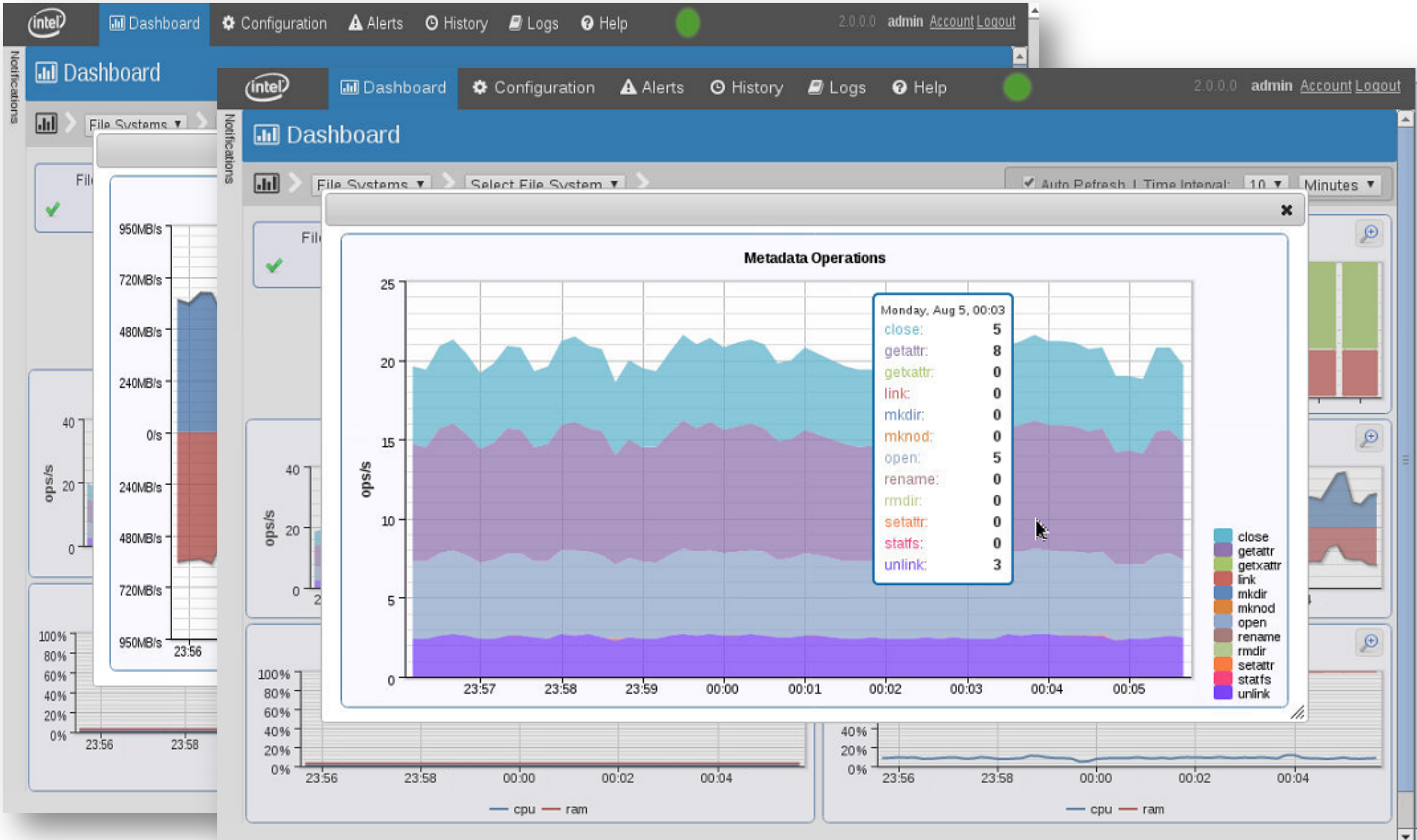
Intel Manager for Lustre

- Provisions and monitors Lustre file systems
- Storage hardware neutral
- Modern webapp built on REST API
- Intuitive GUI
- Fully featured CLI
- Provides plugin interface for integration with storage and other software tools
- Bundled with Intel Enterprise Edition for Lustre

Tools (continued)



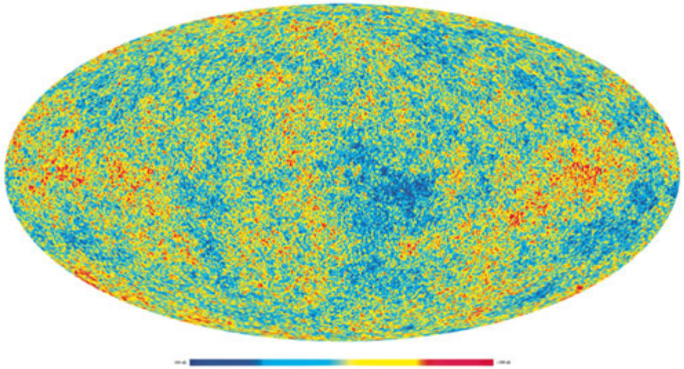
Tools (continued)



Outline

- Lustre* metrics
- Monitoring tools
- Analytics and presentation
- Conclusion and Q&A

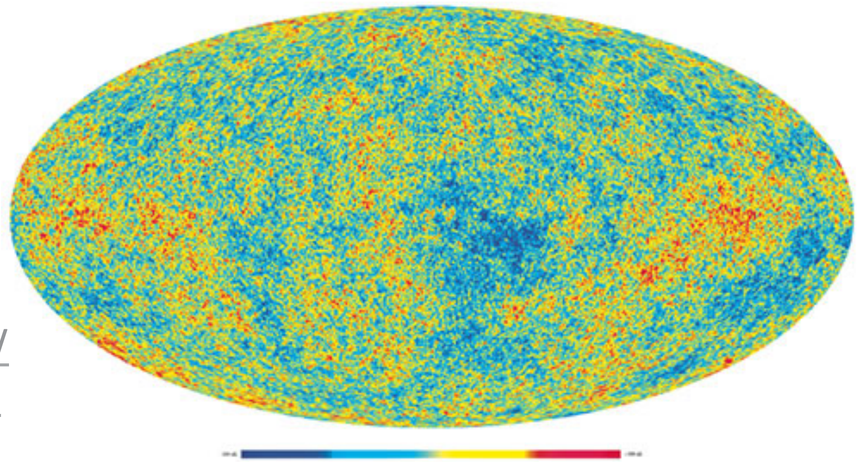
Analytics – scenario 1



- Scenario: MADbench2 - A Cosmic Microwave Background Radiation (CMBR) application distributed across 16 nodes of a virtual cluster constructed from Amazon Web Services resources and including a Lustre* file system with 8 OSSs and 8 OSTs per OSS. All nodes are connected via Ethernet links limited to 110/120 MB/sec

- Tools: lltop (part of LMT), gnuplot, and an ad hoc python/numpy/matplotlib script for presenting data from the log file.

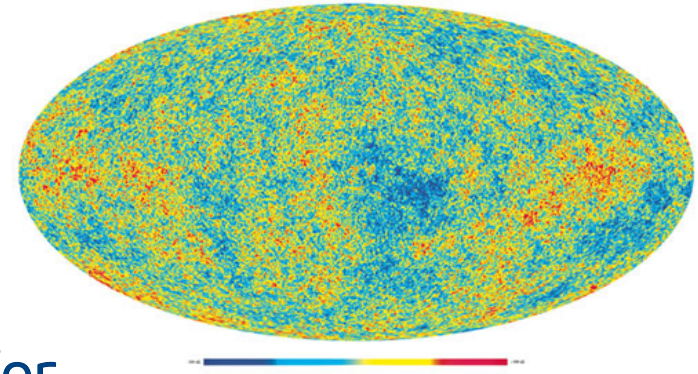
Analytics – scenario 1



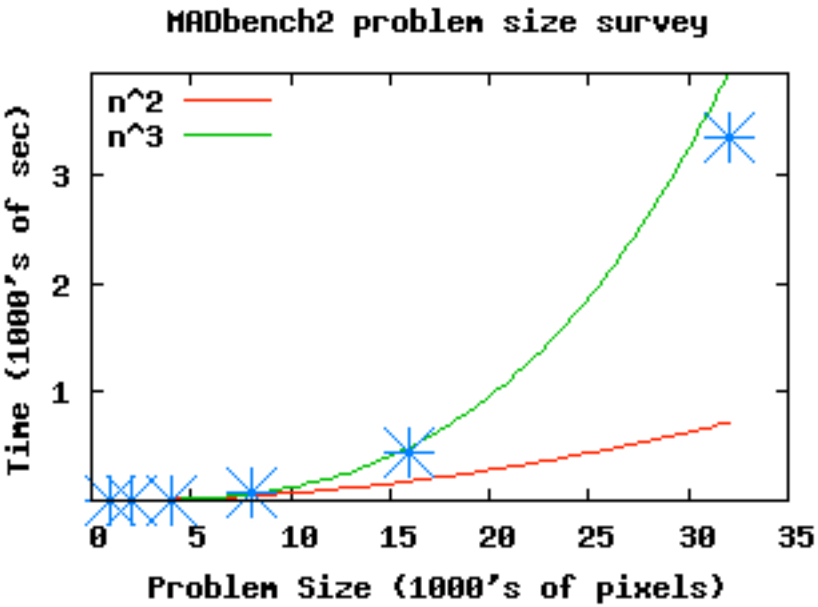
http://crd-legacy.lbl.gov/~borrill/cmb/madcap/
http://crd-legacy.lbl.gov/~borrill/MADbench2/

- MADbench2 carries out a sequence of matrix inversion calculations using an out-of-core algorithm. That phase of the application can be I/O intensive and scales as n^2 for a problem of size n (n is the number of pixels in the map). It also has a communication phase that scales as n^{3}.
- We want to know how the application scales in this environment and what its workload looks like.

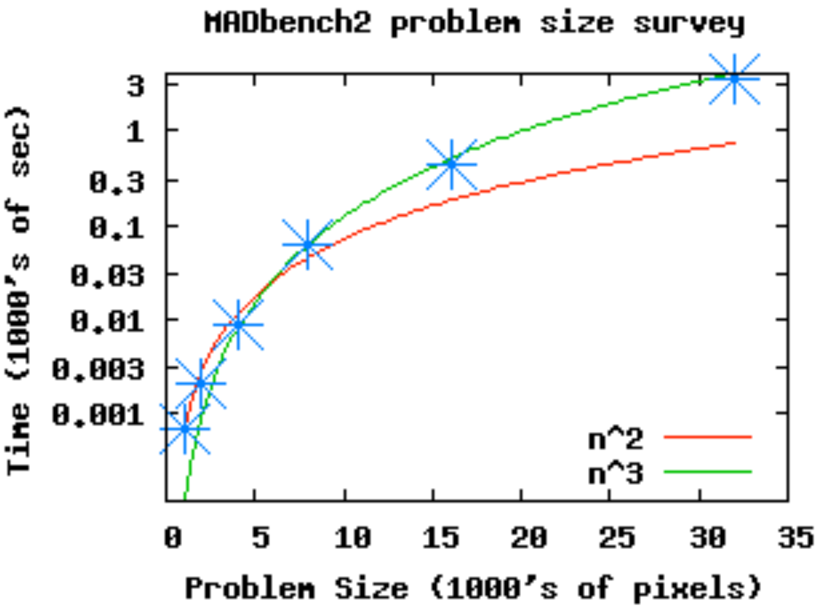
Analytics – scenario 1 (gnuplot)



- The MADbench application does appear to scale as n^3 for larger instances, though not necessarily for the smaller ones.
- This is what we expected.
- But the scaling study doesn't illuminate why.

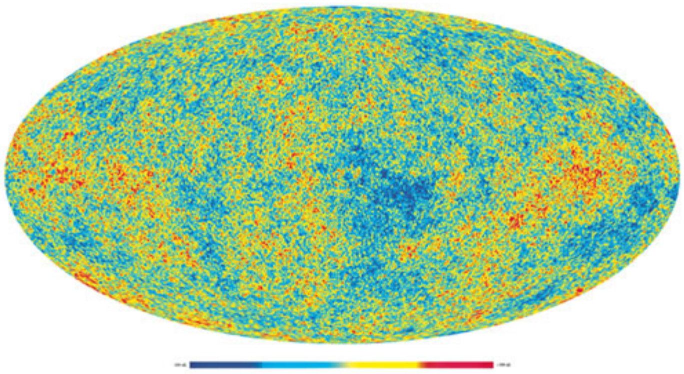


Linear scale



Log scale

Analytics – scenario 1

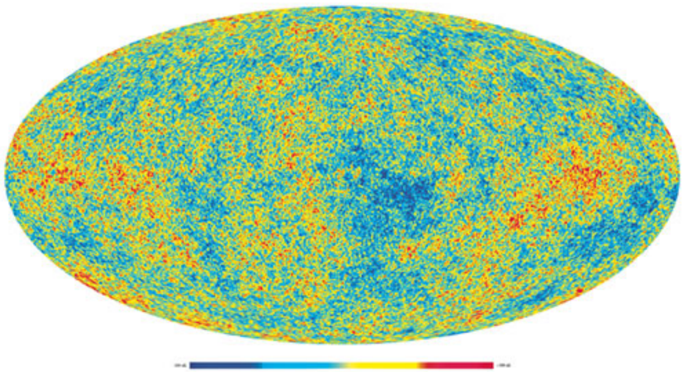


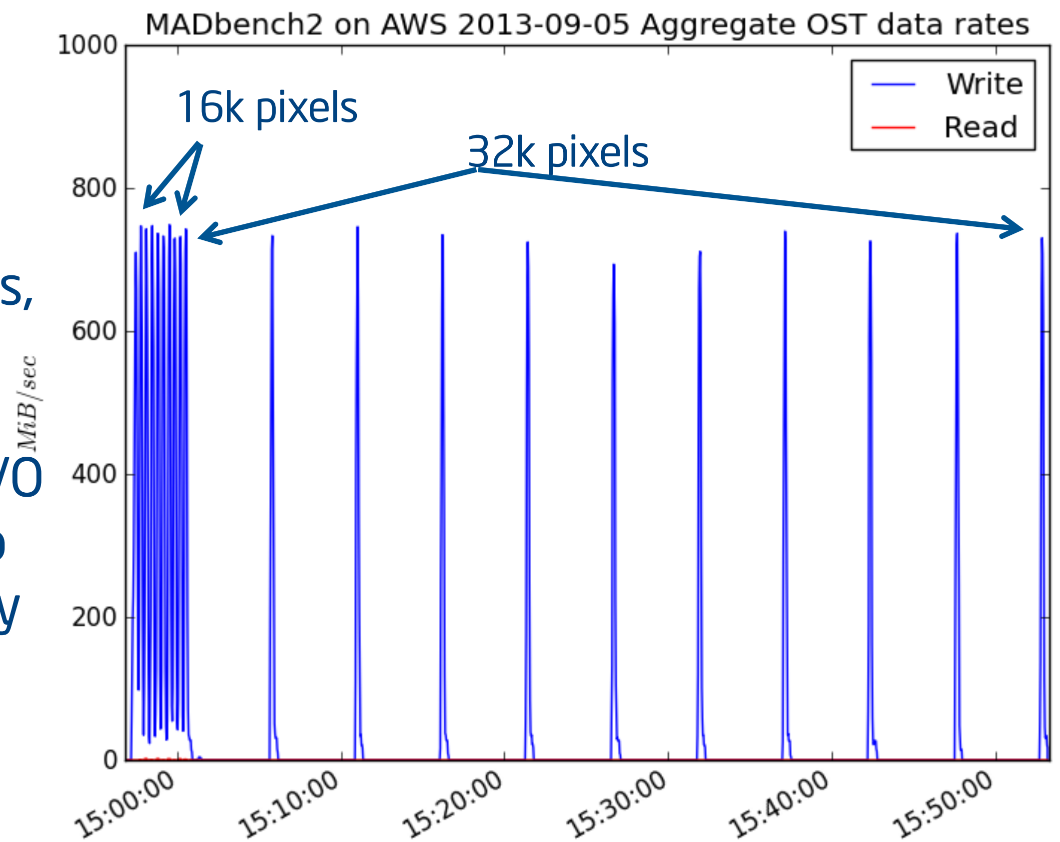
- The ltop utility in the LMT package records OST stat file contents much like llstat.

- An ad hoc python script (with numpy and matplotlib) that parses the output recorded by ltop can present a variety of special-case views of the data.

Analytics – scenario 1

- The smaller instances ran quickly (not shown)
- The 16k and 32k pixel instances became communication bound
- The MADbench2 application reads just as much as it writes, but no reads appear, why?
- Even at this larger scale the I/O fits entirely in client cache, so the reads do not generate any traffic to the servers (where ltop is listening)



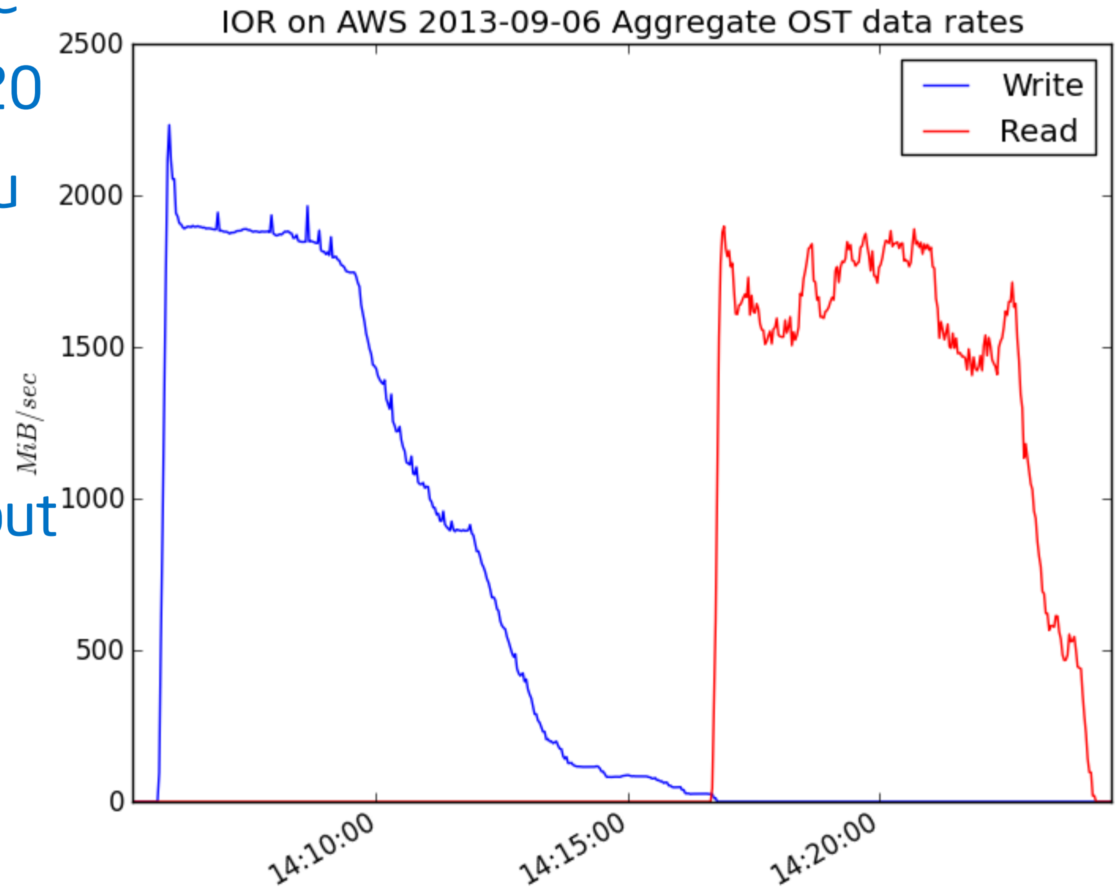


Analytics – scenario 2

- Scenario: Amazon Lustre Cluster with 16 OSS and 32 clients. The parallel I/O benchmark IOR exercises the file system at its best possible rate, and we'd like to get a detailed look at what it is doing.
- Tools: (As before) lltop and an ad hoc python/numpy/matplotlib script for presenting data from the log file.

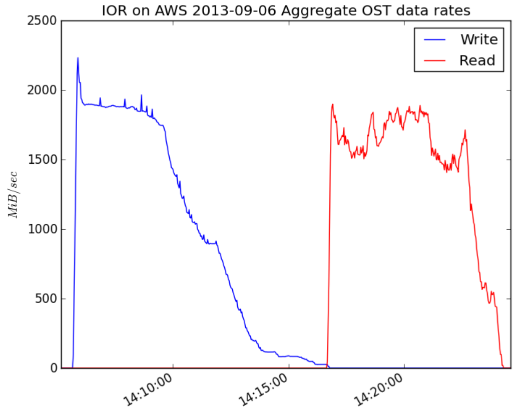
Analytics – scenario 2

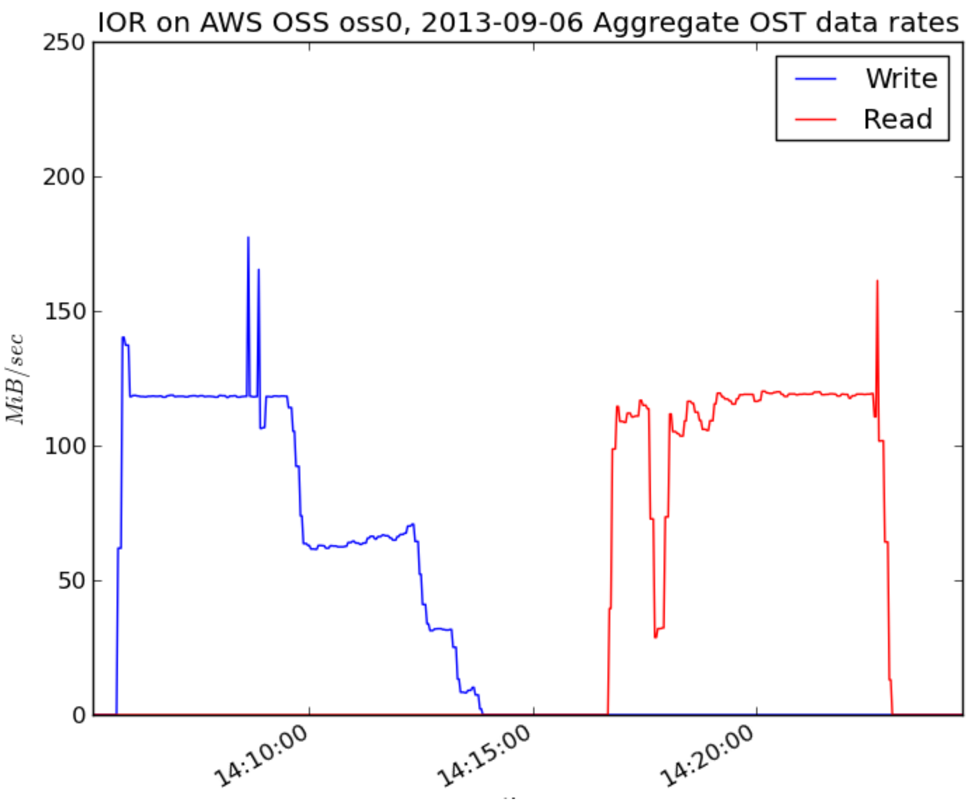
- The peak sustained rate for the I/O is about 1920 MB/s, which is what you would expect from sixteen links
- Reads are little lower, but not much.
- Why the long tail in the writes?



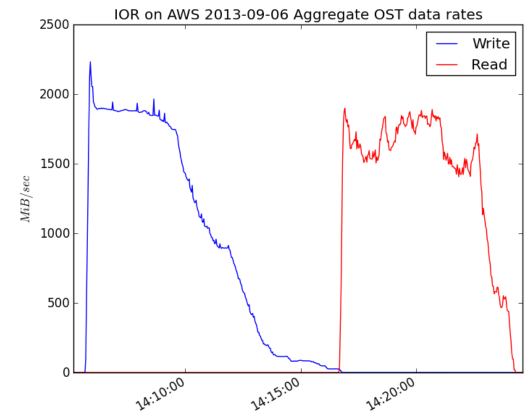
Analytics – scenario 2

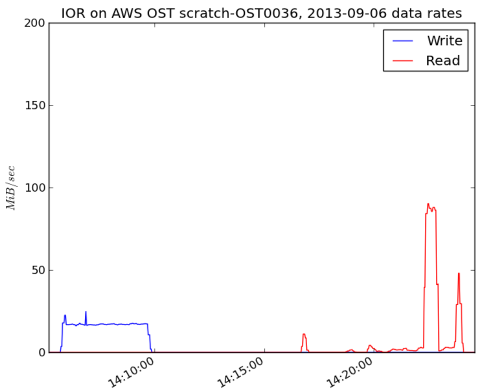
- This is OSS 0, it is typical of the others.
- The writes proceed at about 110/120 MB/s at the beginning, then fall off significantly at some point.
- The reads are mostly at 110/120 MB/s but occasionally fall significantly.
- Why?

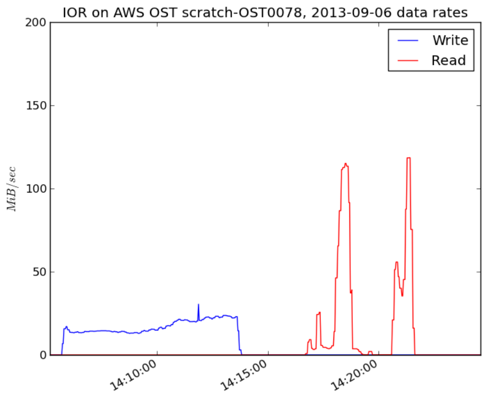


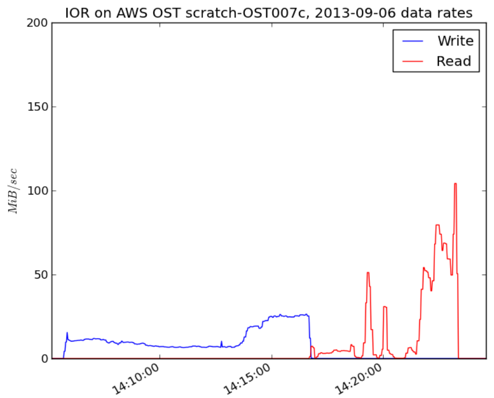


Analytics – scenario 2



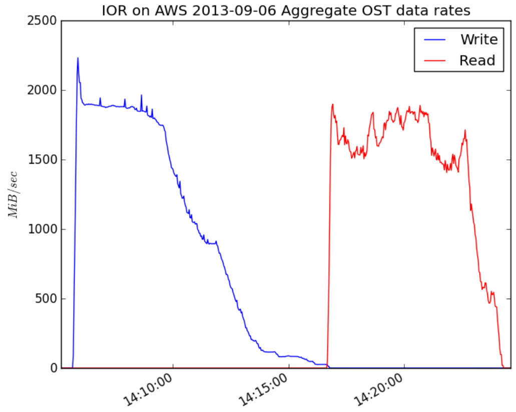


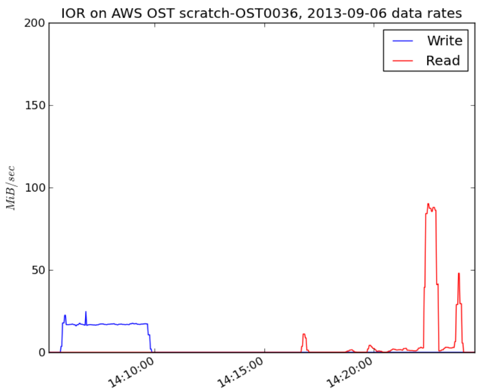


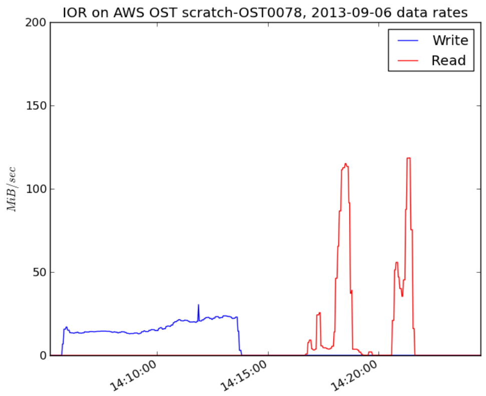


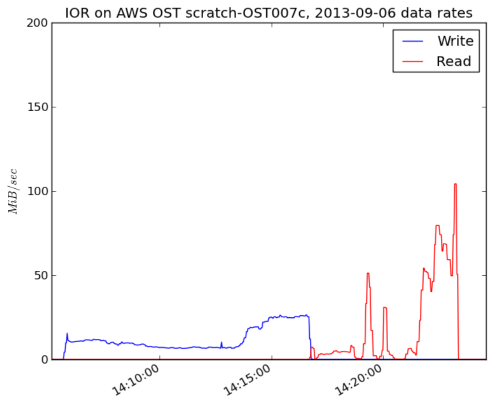
- Three OSTs show very different behavior.
- The writes proceed at a steady (or even increasing) pace until they are done, and it looks like some OSTs have more work to do than others.
- The reads proceed at near the full bandwidth of the OSS, but only in short bursts.
- What gives?

Analytics – scenario 2



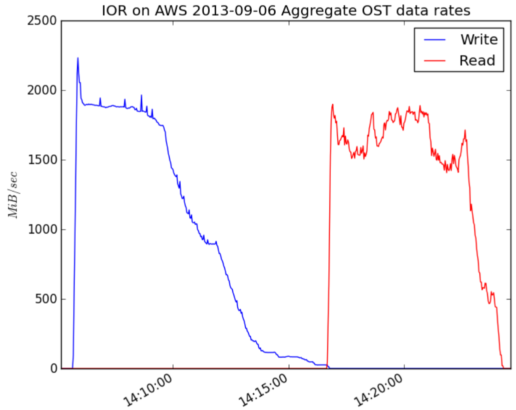


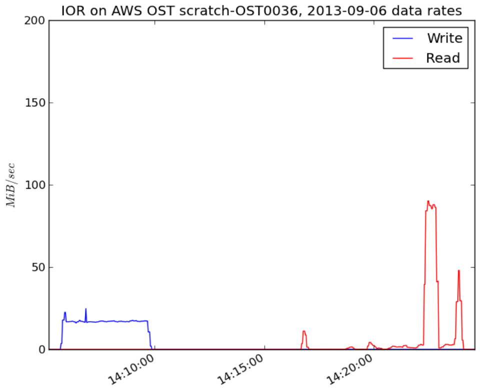


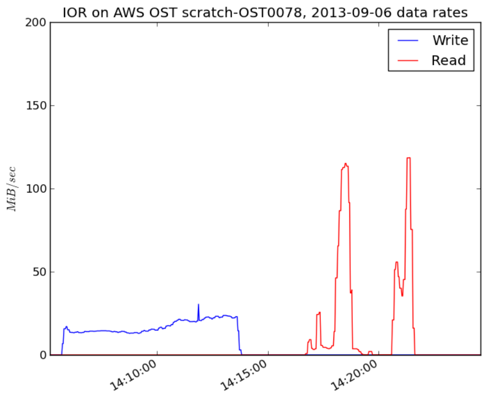


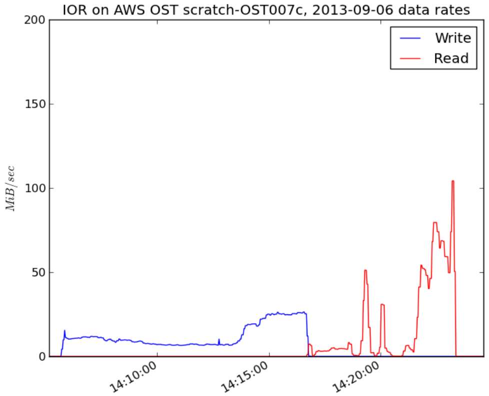
- Writes all take place with data arriving from every client, and the OSS ends up servicing all clients as quickly as the requests can move through its queue. They share bandwidth, so all the OSTs are busy all the time.
- Each client can only have 8 RPCs outstanding at a time.

Analytics – scenario 2





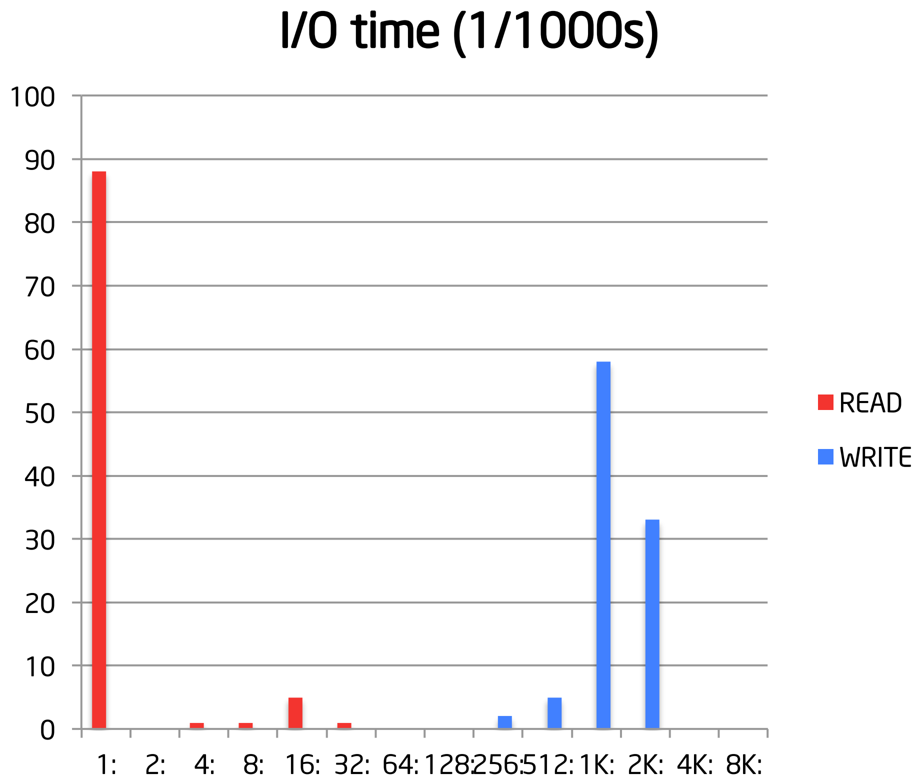




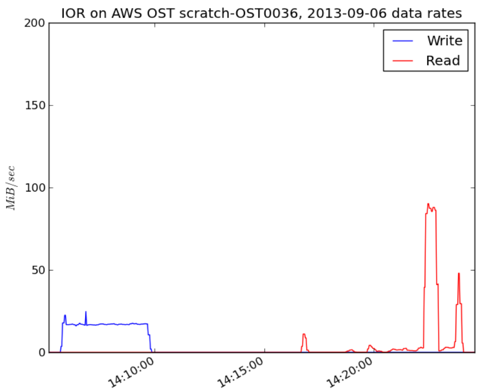
- Reads benefit from read-ahead, so a lucky early read request will prime the OSS with lots of extra data to serve for that particular client.
- That client makes progress quickly, since the OSS can service its requests right away, and the client keeps sending more requests as the original eight are completed.
- The reads either finish or some other OST on the OSS gets lucky.

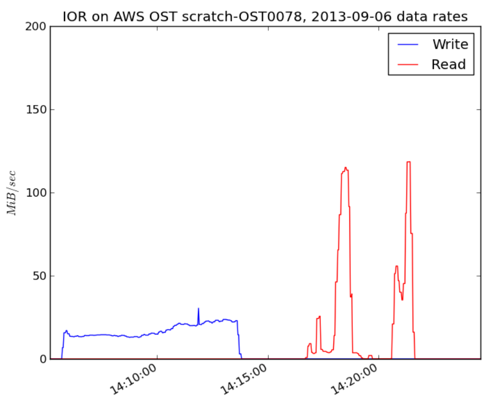
Analytics – scenario 2

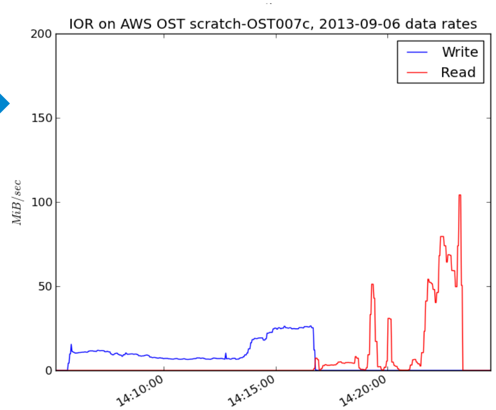
- I/O latency from brw_stats
- the latency and the limit on outstanding RPCs combine to create the situation where reads are bursty rather than steady.

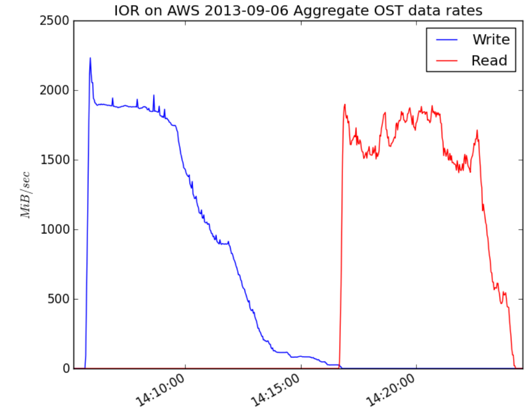


Analytics – scenario 2









- But why is the write tail so long? After all, OST 7c's writes (bottom) actually speed up.
- The OSTs with less work to do finish early and the clients needing that data no longer need to communicate.
- The OSSs could go faster, but the clients have the same link speed as the OSSs. Some clients are still trying to talk to many OSTs so some OSSs aren't seeing enough traffic from them to stay busy.

Outline

- Lustre* metrics
- Monitoring tools
- Analytics and presentation
- Conclusion and Q&A

Conclusion

- There is a wealth of information about the health and performance of Lustre* available in /proc.
- Proactively tracking the changes in that information will allow system staff to anticipate and repair problems.
- Knowing the tools for gathering, analyzing, and presenting the information will help with system issues and with the impacts of user codes.
- In the event that a fault is reported, the monitoring telemetry can help quickly isolate a specific root cause.

Thank You

Questions?

Legal Disclaimer

Risk Factors

The above statements and any others in this document that refer to plans and expectations for the third quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as "anticipates," "expects," "intends," "plans," "believes," "seeks," "estimates," "may," "will," "should" and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel's actual results, and variances from Intel's current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company's expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel's and competitors' products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel's response to such actions; and Intel's ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. Intel's results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel's ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel's results is included in Intel's SEC filings, including the company's most recent reports on Form 10-Q, Form 10-K and earnings release.

Rev. 7/17/13

intel®