OSiRIS - Open Storage Research Infrastructure

Some Ceph "Lightning" Topics

Shawn McKee
University of Michigan
USATLAS Facilities Mtg @ OSG AHM, UCSD
March 6, 2017

Overview

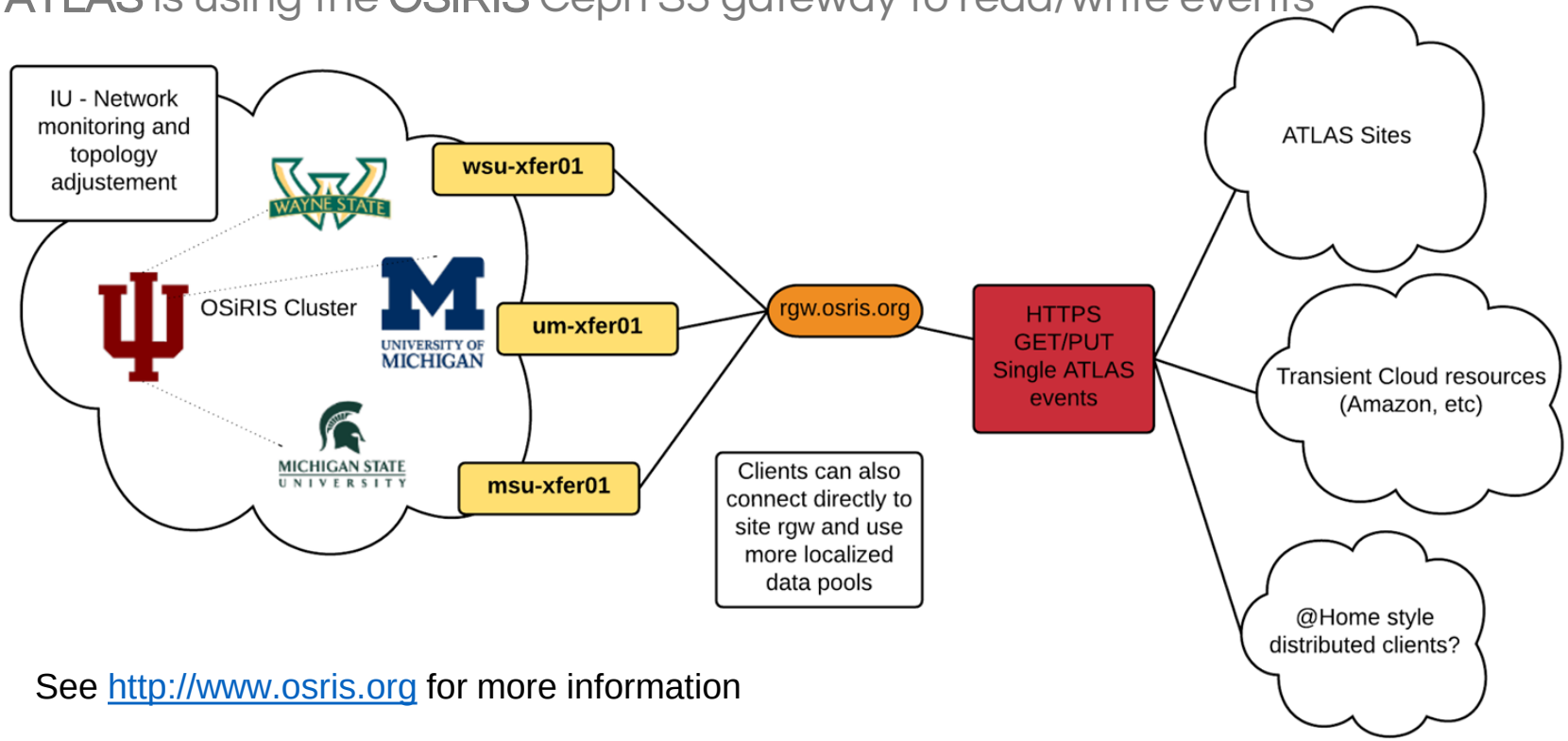
Two topics to cover regarding Ceph:

1. USATLAS testing of Ceph on OSiRIS
2. Our experience upgrading from Jewel to Kraken

3. Thanks to Doug Benjamin for slides on the testing and to Ben Meekhof for info on the testing and slides on the upgrade (he has presented them on the monthly HEP Ceph call today)

ATLAS S3 Event Service

ATLAS is using the OSiRIS Ceph S3 gateway to read/write events



See http://www.osris.org for more information

OSiRIS Load generator at ANL

- Uses 1-15 two socket (8 physical cores- HT one) 10 Gbe interconnect – through two fabric replicators – max 20 Gb/sec to WAN
- HTCondor batch system used to launch whole node jobs virtually simultaneously.
  - Python Boto library v2.4.5 used to communicate and write to the remote OS
- On each node "typical" event stored in RAM disk before writing to remote OSiRIS instance.
- Each node connects to the same bucket in OSiRIS.
- Each node runs 64 subprocesses (4 subprocess per logical Core).
- Within each subprocess a fixed number of threads is used to connect and write to the OSiRIS instance.

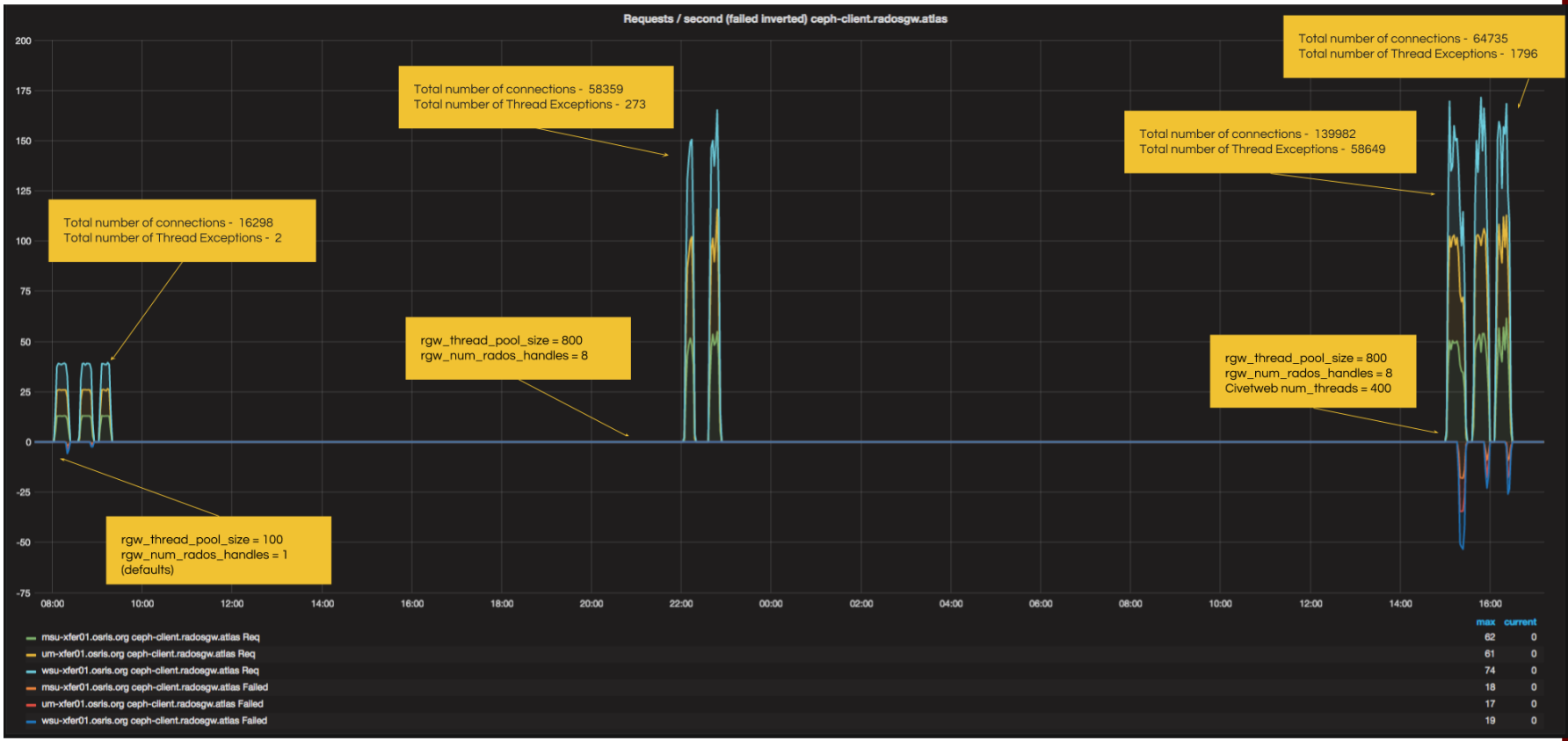
OSiRIS Testing Program

- Based initially on code from Wen Guan in the pilot.
- In a loop that runs until countdown time is finished
  - Loop duration varies if some connections have slow response
- Each tread makes connection to a gateway node. (randomize connections across gateway nodes)
  - Trap connection errors
  - Abort thread on connection errors
- Get the bucket for write
  - Trap for get bucket errors
- Write Object to bucket
  - Trap for write errors
- Retry thread on Get Bucket and write object errors
- Scan job logs for errors and characterize them.

OSiRIS Testing Program

- Initially problems with firewalls and certificates from "Unknown" CAs
  - **All fixed quickly**
- Doug's initial testing showed high failure rates around 7% (21025 con)
  - Cutting back the number of connections by 9% resulted in 50% less failures (19135 con)
  - Cutting back the number of connections another 23% eliminated the failures(16298 con)
- The OSiRIS end of the tests was only seeing about 4% resource use (CPU/memory). **Tuning was needed!**
- From the ceph-users list we got feedback on some variables to tweak
  - rgw_thread_pool_size defaults to 100; try **400**
  - rgw_num_rados_handles defaults to 1; try **8**
- These two changes made a huge improvement; 58359 con at 0.48% failures
  - Pushing on this config with 139982 con resulted in almost 50% failures
  - Tried updating
  - civetweb_num_threads = 400 and doubled thread pool to 800
  - Last test reached 64735 con with 0.7% failure (Doug tuned to use 25 threads/process)

Testing Graphic of Results



Testing Notes about Graphics

- Left off the beginning are some initial tests with high error rates.

The tests beginning 08:00 are the set where Doug stepped down until we achieved a near-zero error rate.

- From the last 3 test peaks I think we can conclude that adjusting civetweb num threads didn't effect things much. Our peak is not really higher.

- In the test with high threads and errors we saw nearly equal 403 forbidden responses and connection reset by peer responses (~ 20,000)

- in the final annotated test the thread errors were from 1500 x reset by peer responses and 15 x 403 forbidden. **We thought civetweb threads might be the reset issue but seems like not.**

Ceph Upgrade Overview

Typical storage: 48 HT cores, 60 OSD, 4 x NVMe journal devs

Components divided equally between University of Michigan, Wayne State University, and Michigan State University

Each Institution: 1 mon , 1 mds , 3-4 storage nodes

Jewel 10.2.5 -> Kraken 11.2.0

Minor Issues

Systemd unit attempts to auto-create ceph-mgr instances and ceph auth keys, doesn't respect cluster setting in /etc/sysconfig/ceph (commands do not pass --cluster)

ceph-mgr crashes immediately - turned out to work fine once kraken was applied to MDS cluster.

ceph-mgr startup, non-fatal: ImportError: No module named cherrypy

Major Issue - Kraken CPU load

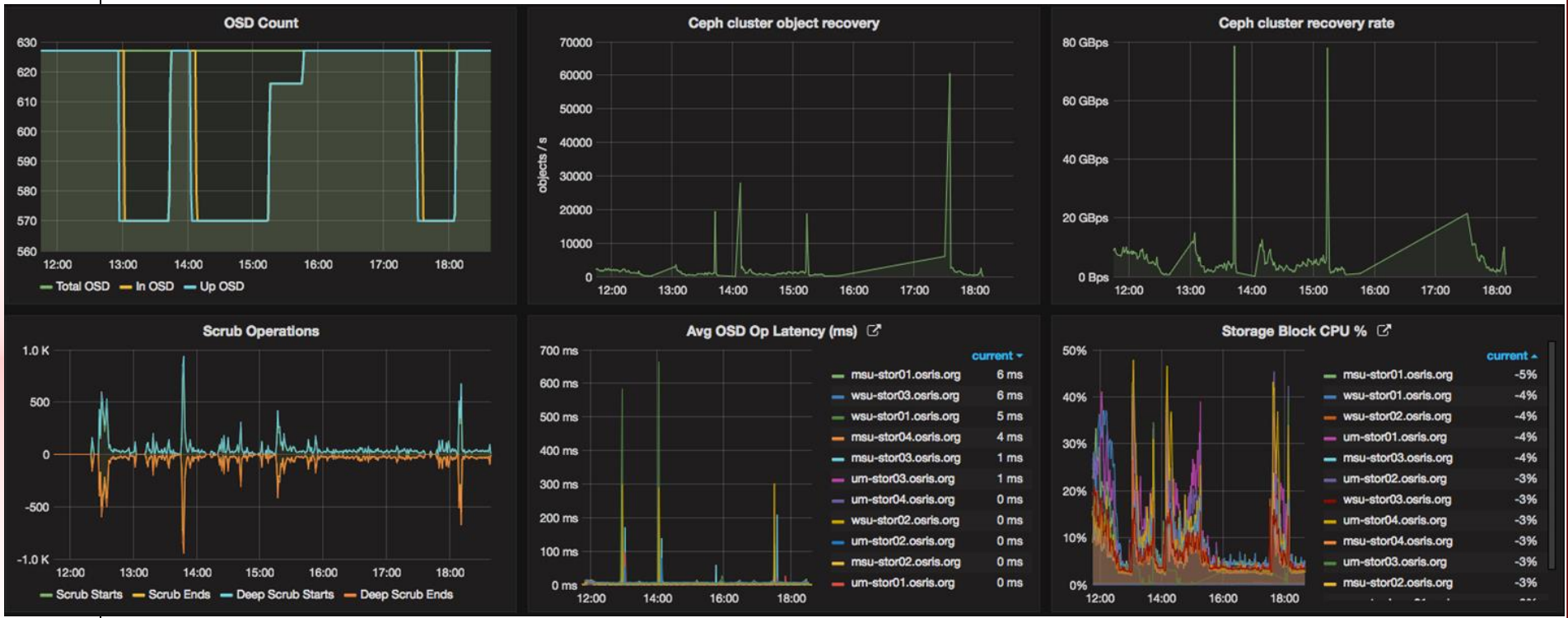
Under Jewel even our dense nodes didn't have major CPU issues during recovery

Updated some nodes to Kraken, and those nodes would be overwhelmed by CPU usage if sufficient number of OSD marked out

- Shutdown of 1 storage node, 60 OSD, loads the rest enough to start flapping if noout/nodown isn't set
- Starting OSD on a node takes hours (competing load)

Major Issue - Kraken CPU load

Contrast - Jewel CPU



Mitigation

We were able to mitigate the issue by reducing async messenger instance threads and osd recovery/op threads.

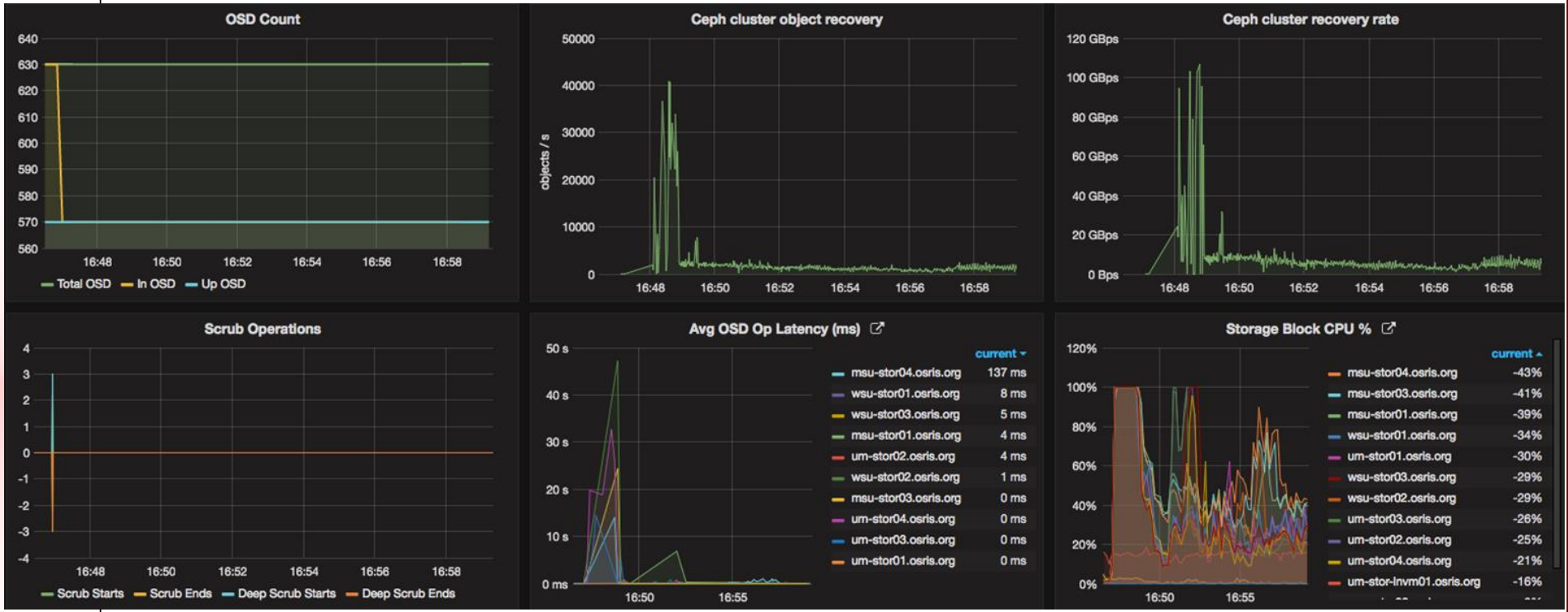
Still unusually high, but brief, peaks in CPU usage during cluster state changes (OSD out, in, recovery)

Have since updated remaining OSD to Kraken

ceph.conf

```
[osd]
ms_async_max_op_threads = 1
ms_async_op_threads = 1
osd_recovery_max_active = 1
osd_op_threads = 1
```

Kraken CPU - With thread reduction



Bad timing - data center outage

During our update we had many PG left in non-ideal states.

- flapping OSD
- intentionally stopping/restarting OSD
- starting up a storage node is now many hours of CPU churn, were trying to wait it out
- multiplied by 5 nodes updated to kraken

And then disaster struck! Vendor doing scheduled equipment test in UM datacenter flipped the wrong switch, datacenter power out 1 minute.

Bad timing - data center outage

OSD logs were warning about auth key timestamps, possible clock skew, not starting up.

The 2 monitors left were restarted and wouldn't form quorom

- (i think) monitor leveldb too large, could not process before election call
- On one mon, set: mon lease = 20
- Startup then proceeded and cluster was available

No Data Lost!

We didn't really do anything special to recover

Started up storage nodes, waited out the CPU churn over a day or so with noout/nodown set. Had some unfound objects but all came back.

Subsequently did some controlled 'outages' of OSD or whole nodes and eventually landed on the thread tuning settings to control the issue

Or rather...to compensate for our limited CPU resources ?

- Under jewel our storage blocks ran 100,000+ threads
- Under Kraken about 3000
- Was the OS just so choked before that it produced a throttle effect?

Discussion

Comments or Questions?