Solving of IO crisis with ATLAS production at OLCF

Danila Oleynik UTA

Initial setup of ATLAS production at OLCF

- On initial steps of placing of ATLAS payloads to Titan was discovered that ATLAS software generate high loading to Lustre metadata servers

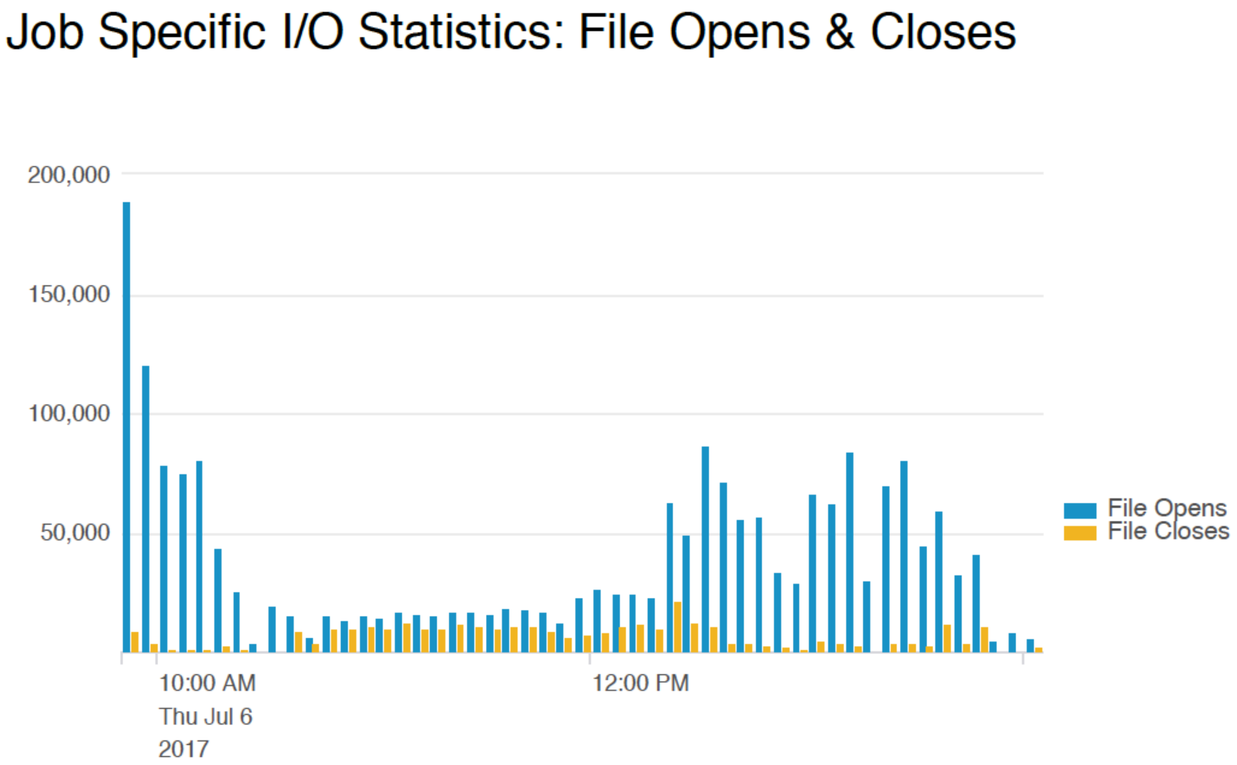
- To cope this loading ATLAS software was moved to NFS filesystem, which available at OLCF, but have limited quota: 50Gb

- All transient data, sqlite DB, jobs working directories etc. were placed to Lustre, and in general should cause reasonable IO

Identifying of crisis

- At May 2017 we got permission to use up to 20 slots for PanDA pilots in local batch queue of Titan: with our other constrains it give as possibility to launch up to 6000 ATLAS jobs simultaneously (in case resources will be available)
- It allowed to complete about 900K jobs in May. We collect about 20M Titan CPU*hours
- June was a bit less successful, given available backfill: 690K completed jobs, and some increasing of number of failed jobs.
- Increasing of execution time for some jobs to 90-120 minutes (and more)
- At July 6 I got complaint from OLCF support about highly IO metadata intensive jobs, which impacted other users

IO crisis



Start Time: Jul 06, 2017 09:52:55
End Time: Jul 06, 2017 15:57:31
Nodes Used: 300

"System administrators have alerted us that your jobs on Titan are currently doing over 2 million metadata operations per job and causing congestion on the metadata server. This is impacting other users' jobs.

To help the current situation, could you stop the jobs you have running on Titan and limit to only 1 running job at a time? We have received several reports of hangs on the atlas2 partition.»

We was allowed to launch up to 20 submissions in row, so total number of metadata operations from ATLAS production may reach 40 millions operations.

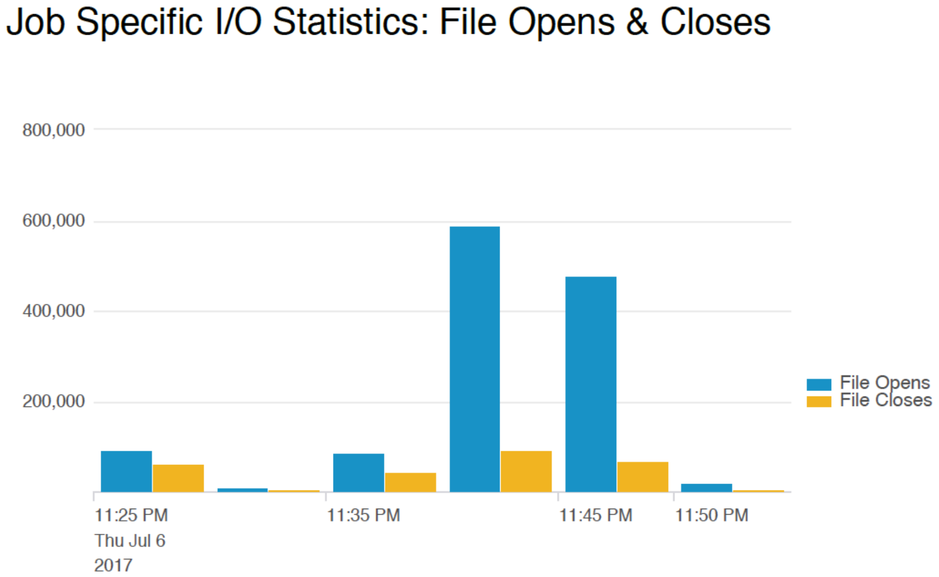
Lustre metadata servers were affected, and looks like that NFS some well with loading.

First action and starting of investigation

- Immediately maximum number of used slots was decreased from 20 to 5 (from 6000 nodes to 1500 nodes)
- Stripping of working directories was changed to serve big number of small files from one MPI rank (PanDA job)

```
lfs setstripe -c 1[directory]
```

- Execution time of job decreased almost immediately to 40-60 minutes. IO profile changed - became more ‘solid'
- Unfortunately we didn’t get detailed information about affected files etc. So we was needed to guess possible sources of issues



Start Time: Jul 06, 2017 23:26:16
End Time: Jul 07, 2017 00:03:47
Nodes Used: 300

Initial suspected

- One of first suspected source of issue was shared access to sqlite DB.
- Sqlite DB require write access, so it was placed to Lustre and linked from working directories through symlinks
- A set of changes was implemented:
  - deprecate symlinks
  - change priority of paths for looking up of DB
  - finally move DB to RAM disk of working node
  - Input data was moved to RAM disk of working data too
  - Unfortunately, IO profile did not changed significantly

IO profiling with ‘strace'

- It was clear, that most of IO produced by ATHENA. To get more detailed information typical job was wrapped with ‘strace’. Thanks for trick to Vakho.
  - 'Strace' produce high overhead and produce huge output
  - Only few jobs was finished successfully with 'strace' wrapping
- ATLAS job used for the study: http://bigpanda.cern.ch/job?pandaid=3558462700
- 'Strace' log file: https://cernbox.cern.ch/index.php/s/48s7ORjYww2F0hY

Information gathering from ‘strace' log

- Summary of analysis of one ATHENA MP job (16 cores):
  - Number of uniq files opened during execution: 6826
  - Number of uniq ‘*.so’ files opened during execution: 1899
  - Number of uniq ‘*.py(c)’ files opened during execution: 1475

Number of metadata operations from ATHENA MP job 16 cores

	Open ok	Open fail	Stat ok	Stat fail
ATLAS release area (NFS)	**109987**	**371688**	**124522**	**318206**
Lustre	2902	5860	2502	2356
Working directory (Lustre)	471	2605	365	995

- Reminder: one submission to Titan: from 15 to 350 ATHENA MP jobs
- Up to 20 simultaneous submissions

Why we have many calls to Lustre?

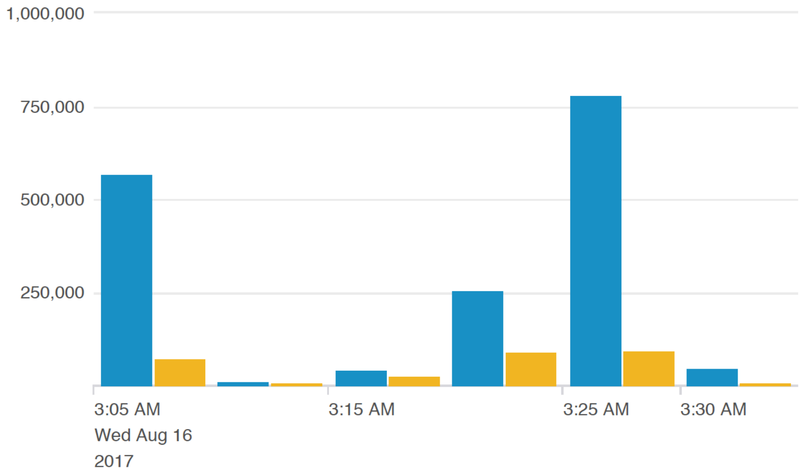
- Traversing of shared libraries through PYTHONPATH, PATH and LD_LIBRARY_PATH
  - Some of this variables was extended with paths to third party libraries (GRID stuff etc.) These paths not needed for payload itself and can be cleaned
- Checking of current working directory first to find shared libraries:
  - Small hack in athena.py helps to reduce it (thanks to Vakho)

```
import sys
sys.path=sys.path[1:]
```

Hack applied only at OLCF as experimental solution to see could it help with crisis and not propagated to the Atlas code repository yet.

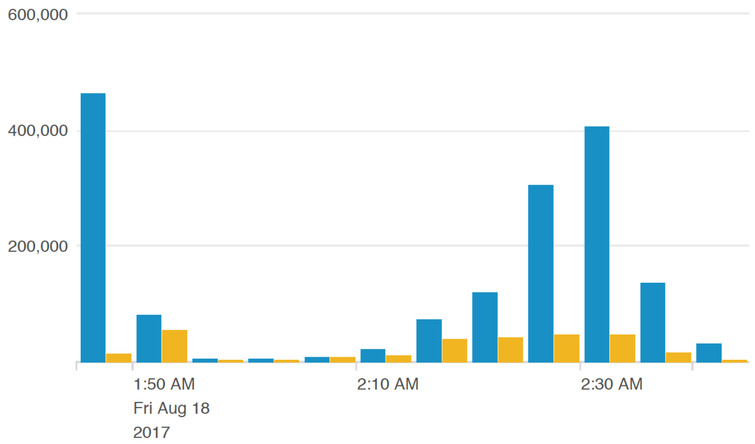
Results of optimization

Before optimization



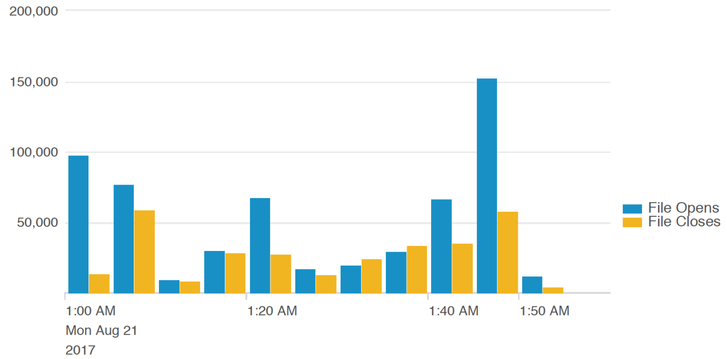
Start Time: Aug 15, 2017 23:56:11
End Time: Aug 16, 2017 01:01:16
Nodes Used: 350
batch id: 3567126

Environment cleanup



Start Time: Aug 18, 2017 01:46:04
End Time: Aug 18, 2017 02:58:22
Nodes Used: 350
batch id: 3570319

ATHENA 'hack'

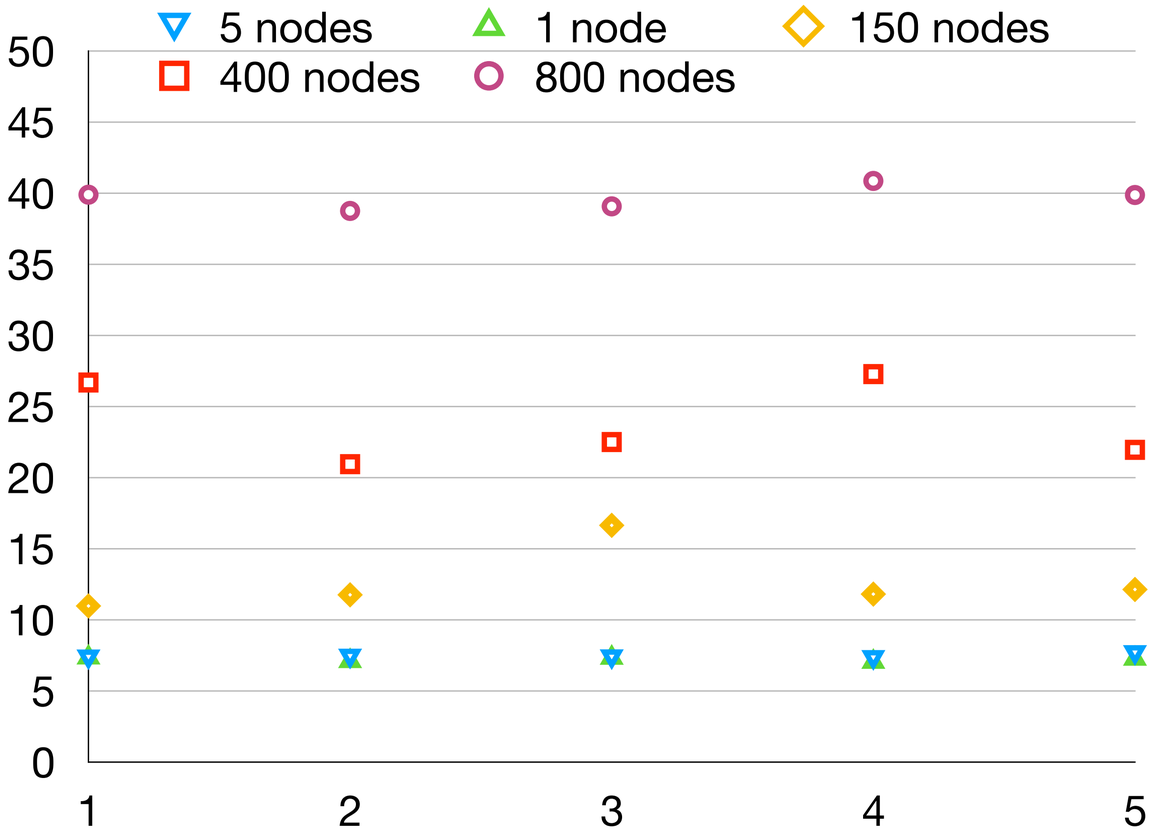


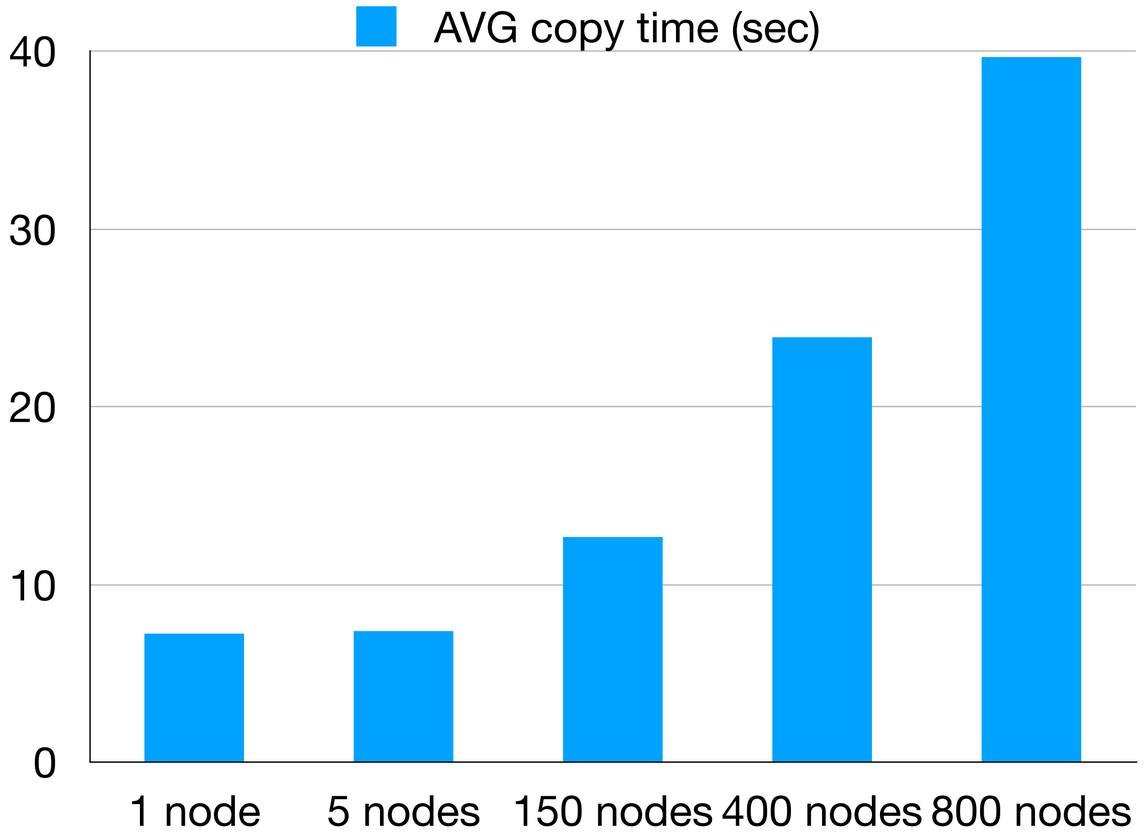
Start Time: Aug 21, 2017 01:02:42
End Time: Aug 21, 2017 02:16:04
Nodes Used: 350
batch id: 3573737

Good, but not enough

- Even with decreasing of IO with ~factor 4 it still was high.
- Next step was moving of whole working directory to RAM disk of computing node with synchronisation of files to Luster at the end of job.
- Understanding of overhead of this operations require scalability tests:
  - Copy of ~4Gb files for each node for: 1,5,150,400 and 800 nodes job and measure spent time.
  - 5 measurements for each scale

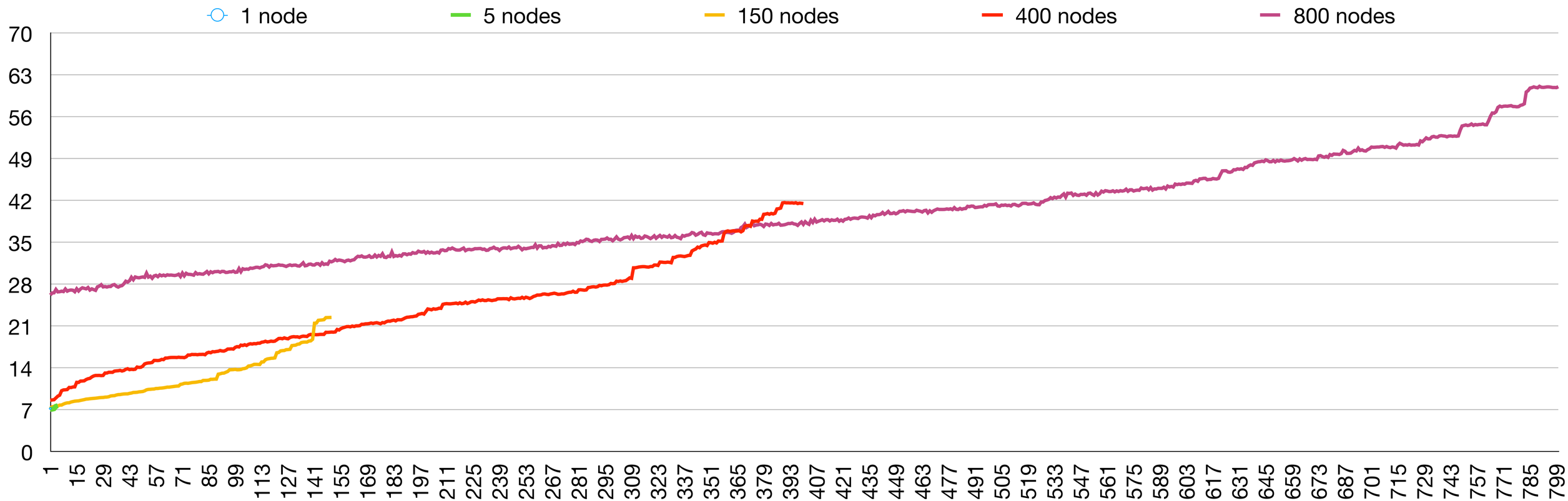
Results combined (average copy time)





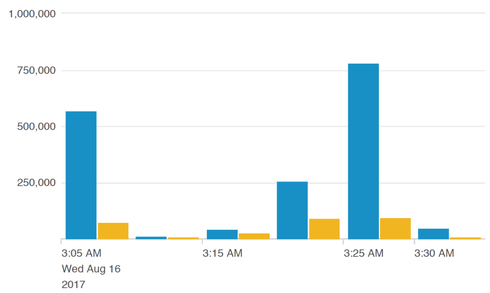
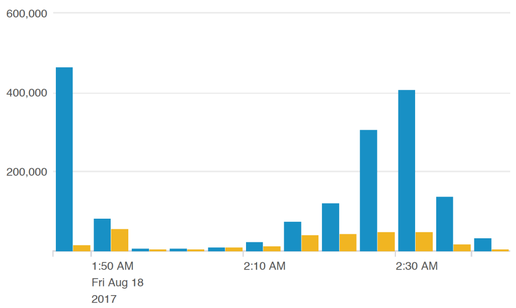
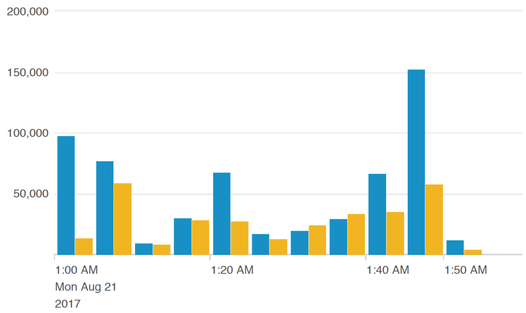
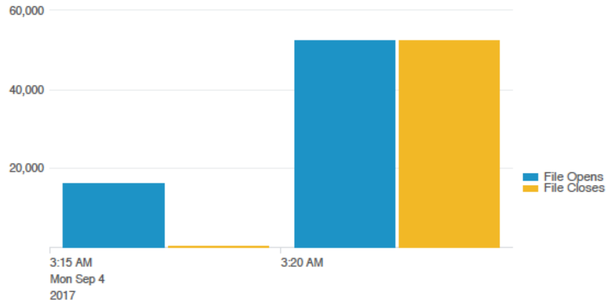
- Slow (non proportional) increasing of copy time

Results combined (average copy time)



- Acceptable delay even for big jobs

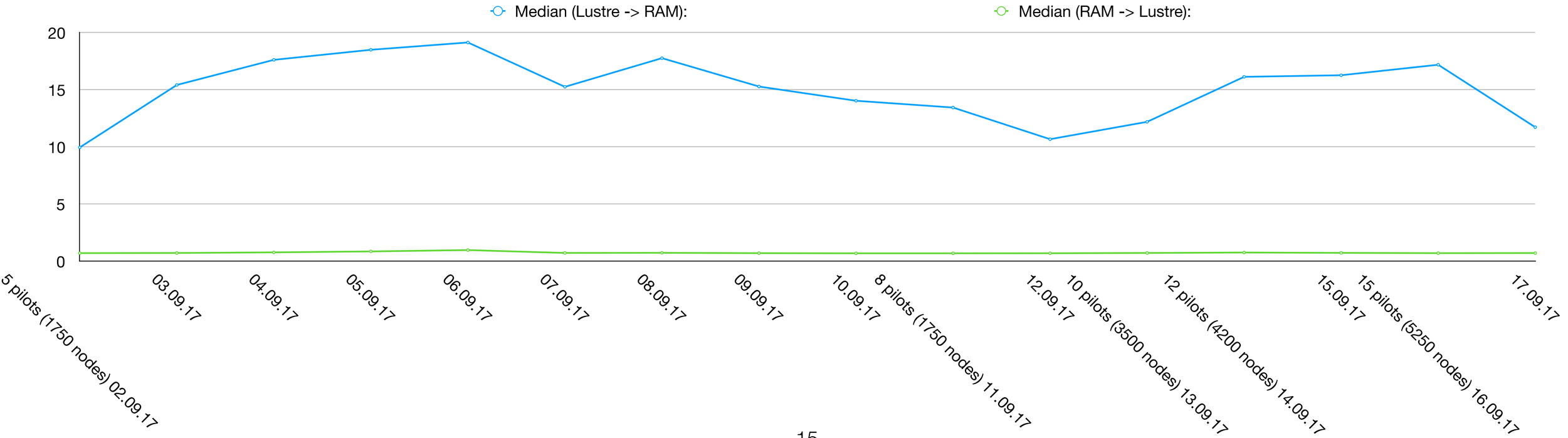
Moving of working directory to RAM-disk. Results

Before optimization	Environment cleanup	ATHENA 'hack'	Working dir in RAM Disk
			
Start Time: Aug 15, 2017 23:56:11 End Time: Aug 16, 2017 01:01:16 Nodes Used: 350 batch id: 3567126	Start Time: Aug 18, 2017 01:46:04 End Time: Aug 18, 2017 02:58:22 Nodes Used: 350 batch id: 3570319	Start Time: Aug 21, 2017 01:02:42 End Time: Aug 21, 2017 02:16:04 Nodes Used: 350 batch id: 3573737	Start Time: Sep 04, 2017 03:19:01 End Time: Sep 04, 2017 04:00:00 Nodes Used: 350 batch id 3602038

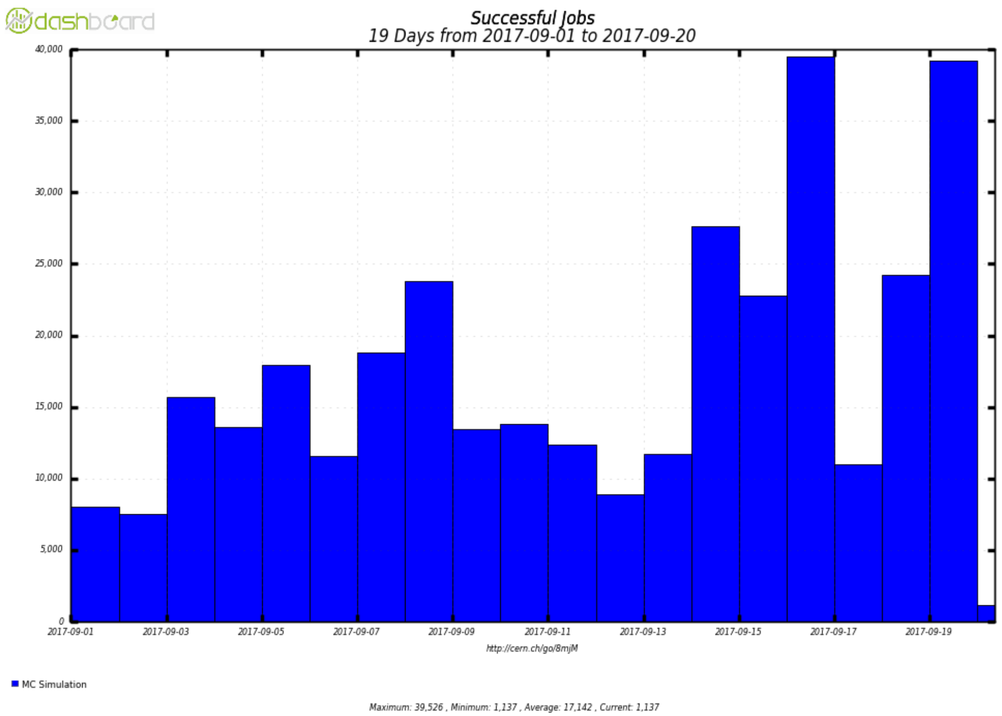
- Significant reduction of IO. Number of ‘open' operations almost matches with number of close operations.
  - Initial spike came form MPI wrapper which used to launch ATHENA Job on computing node. Already reduced with same fix like athena.py
- Current setup of ATLAS production at OLCF
  - ATLAS releases: NFS
  - Job working directories and input data: RAM disk of computing node
  - Output data moved to Lustre at the end of the job

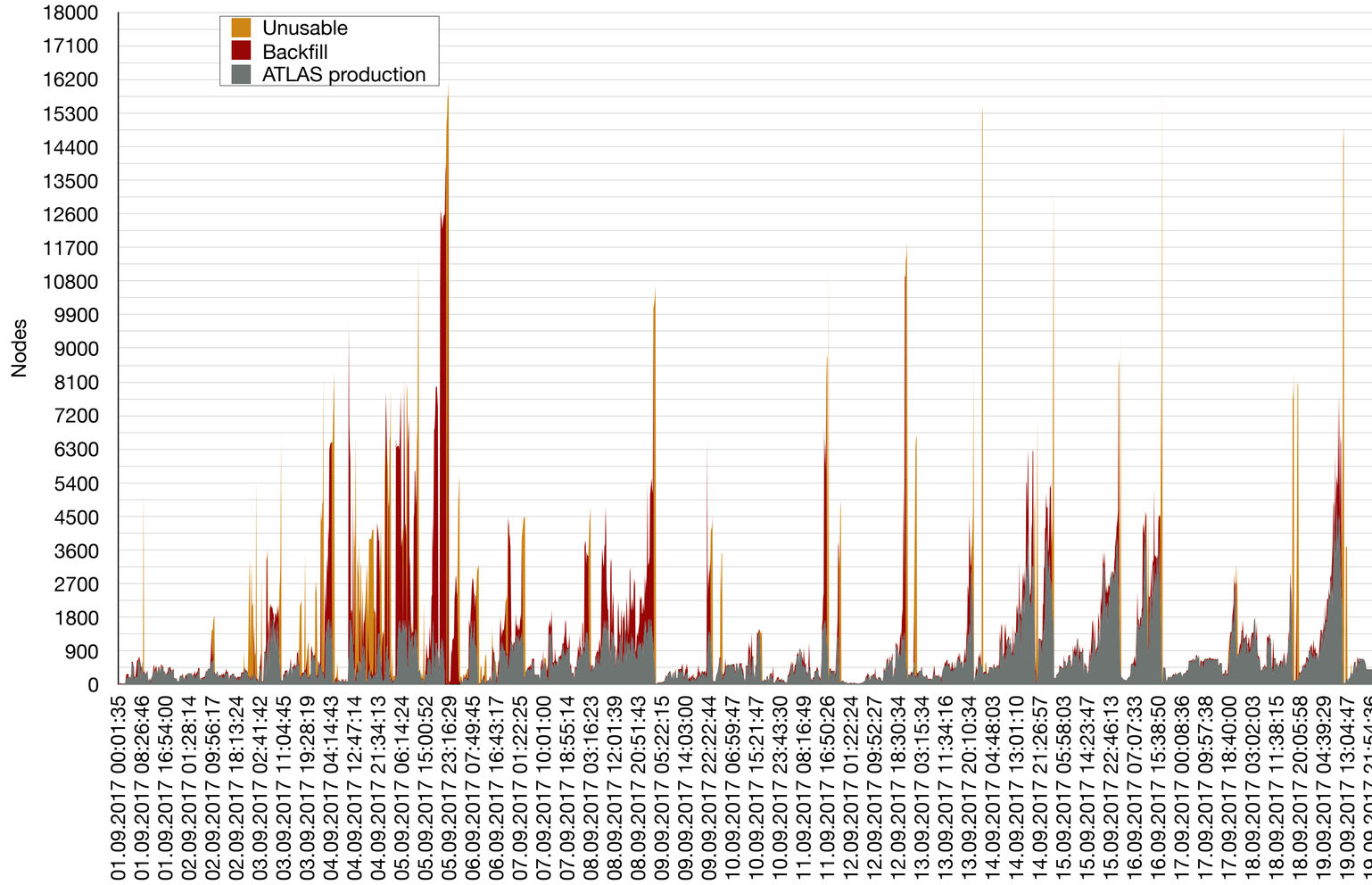
Increasing of loading

- 11 of September we got a 'green light’ from OLCF to increase loading.
- Number of Pilots increased step by step last week from 5 to 15 with maximum size of Pilot: 350 PanDA jobs.
  - No visible increasing of execution time
- IO carefully monitored: in general there is no increasing of median of copy latency: 10 - 18 seconds of copy of input data to RAM-disk, ~0,8 second of copy of output data to Lustre



Increasing of loading





Conclusion

- It took about 2 month of investigation, fixing and proving of solution
- Different specialist was involved: OLCF support team and sysadmins, ATHENA developers, PanDA team
- It’s site specific optimisation, but study can be used by other site and future development