Lessons learned from an ad hoc ATLAS data challenge to the Lancaster WLCG Tier2 site.

HEPiX 2022.
Matt Doidge (Lancaster University), James Walder (STFC), Duncan Rand (JISC/Imperial College London).
with contributions from Gerard Hand and Steven Simpson (Lancaster University).

Background (bordering on a site report)

- Lancaster (or “UKI-NORTHGRID-LANCS-HEP”) is a long established WLCG Tier-2 gridsite in the UK, at Lancaster University.
- Provide ~8000 cores of compute.
- Until recently used DPM for our grid storage needs.
- Our primary WLCG usergroup is ATLAS.
- In late 2021 deployed a new storage solution, consisting of a 10PB CephFS volume with an XRootD frontend.
- Another recent change was the increase of our connection to our NREN from 10 Gb/s to 40 Gb/s (4x10).
- The storage (and the site as a whole) is outside the institution firewall, and has an almost dedicated network link (it’s the University’s backup link).

Motivation.

We've had two big changes that could affect our data taking rates in unknown ways, a new storage solution and a network link upgrade. With that in mind:

How much data can we shovel to our site?

- Perfsonar only gives us a limited picture of our network capacity.
- Manual tests tend to only show functionality, not capacity.
- Atlas have the infrastructure to push around a lot of data however (FTS).

Where's the inevitable bottleneck?

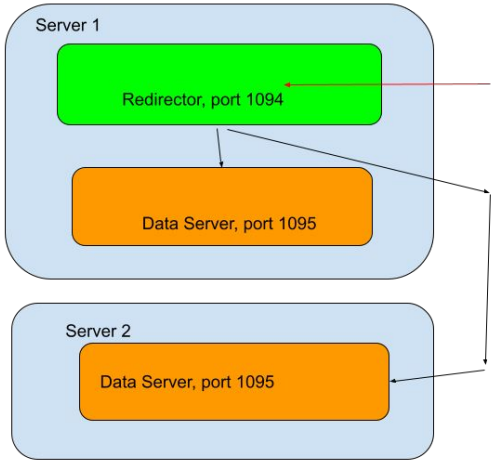
- The site network link (40Gb/s)?
- The XRootD server's network (Nx25Gb/s, where N is the number of xroot servers)?
- CEPHFS write speed? (dd tests fill the NIC bandwidth, suggesting this won't be the case).
- The XRootD server's capacity to "process" transfers?

Are we monitoring everything we should monitor?

- System Load, Network Load, Log file error message flags.

Some technical details (xroot endpoints)

- xroot server 1 (redirector and data server): Midrange Dell “Pizza Box”, 24 Cores (48 Threads), 128GB RAM, 2 x 25Gb Broadcom NICs (1 world facing, 1 internal mounting CEPHFS)
- xroot server 2 (data server): Another, slightly more modest pizza box (technically a retasked NFS server), Same CPU, 96GB of RAM (this became a factor), same networking.
- OS: CentOS7
- xrootd version: 5.4.3
- configured for https tpc
- Config files in my github (see references).

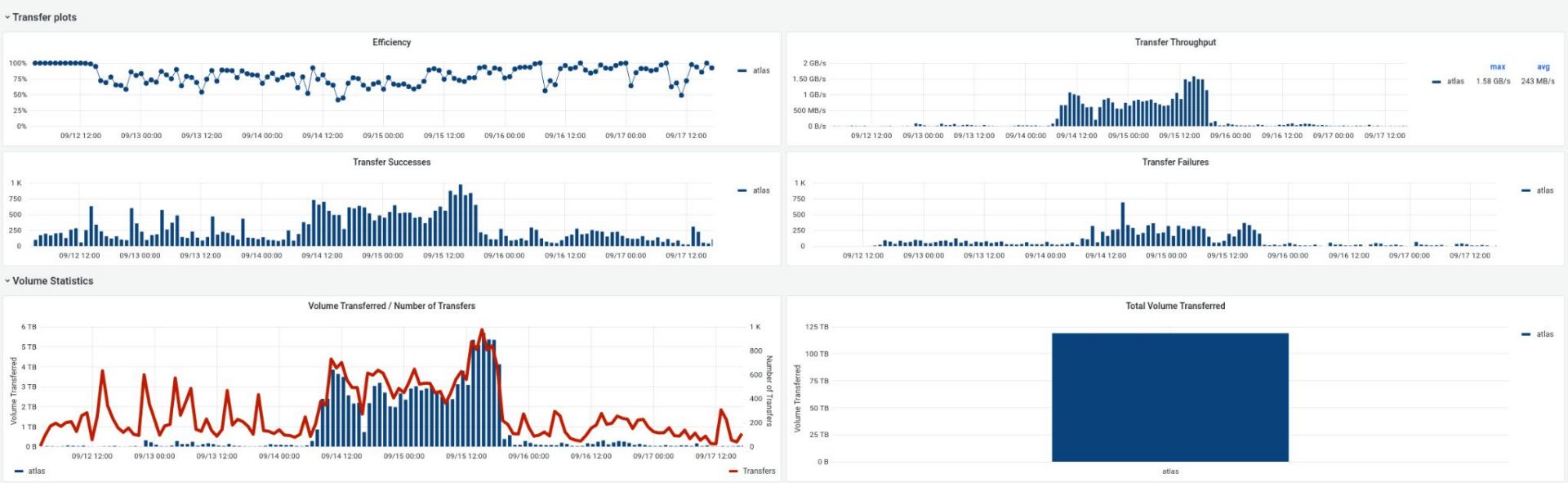


Method

- Working with Duncan (one of our NREN reps) James (the UK Atlas rep) subscribed 100TB of data (4 ~25GB datasets) located at CERN to Lancaster.
  - This had to be approved by the ATLAS DDM team.
  - The data sets purposely consisted of a mix of large and less-large files (3 had ~9GB, 1 had ~4GB).
  - Approximately 9k of the 9GB files, and 6k of the 4GB files.
  - The files were “T-tbar MC, derived AOD format (PhysVal)”.
  - The transfer was predicted to take a few days, allowing us time to react, adjust and review.
- When transfers were ready the number of simultaneous FTS transfers between CERN and Lancaster were increased (from 100 to 200 max transfers, 300 max connections)
  - This encouraged FTS to throw as much data as we could our way.
- Our plan was to then sit back and watch the FTS plots, keeping each other informed of our observations.

…I’m not entirely surely that’s enough to qualify as a plan!

The Plots (test period ~8am 14th Sept to ~7pm 15th Sept)



The maximum data rate over the first day was 1.07 GB/s (8Gb), with an average of 0.75 GB/s.
(we're also aware that our efficiency isn't ever very good even outside of the test period).

What were we seeing?

- The FTS was regularly throttling back transfers as it hit the efficiency threshold.
  - FTS makes decisions on whether to ramp up or down the number of concurrent transfers by the efficiency (i.e. % failed transfers). There are other factors, but this was the one we hit.
  - Sadly the decision level isn't indicated in these plots.
- Transfers were failing due to "checksum timeout" errors.
  - This is not a new error, but we thought we had tuned it out through tweaking xrootd settings.
- The xrootd servers themselves were heavily loaded, with corresponding errors in the logs.

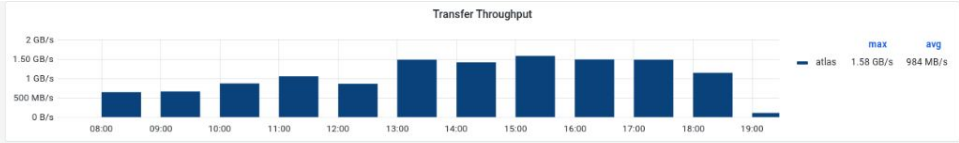
Solution: Try to increase the checksumming capacity for our servers.

Mid-test Tuning (First Day)

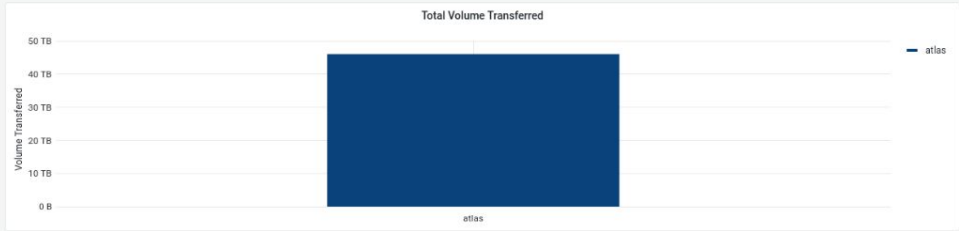
- With the “checksumming capacity” being the heavily limiting factor we tried to tune (i.e. raise) the relevant xrootd settings:
  - xrootd.chksum max XX (the number of simultaneous checksum operations)
  - ofs.cksrdsz XXXXm (AIUI the size, in MB, of each file chunk stored in RAM for the operation).
    - The total memory consumed by checksumming operations = Nchksums * Size
- These have had to be raised before, in “normal” operations, due to the quite low defaults.
- There is also the chksumming digest used - the inbuilt xrootd adler32 in our case.
  - There is a possibility to replace this with a custom adler32 script, but this was a bigger change then we could make in a short space of time.
- However we took things too far on our initial tests, and started getting out of RAM errors on our xrootd servers. Hence we throttled back some numbers.
  - The “sweet spot” appears to be:
    - maxchksums * cksrdsz < MachineRAM/2
    - maxchksum ≈ nMachineThreads.
- And sadly these changes only had a small affect, if any.
  - Essentially they were about as “tuned” as they could be.

Throwing more servers at the problem (12.00 2nd Day)

We had one drained DPM disk server laying around, originally earmarked for a CEPH testbed. So we quickly reinstalled it and threw it in to production as a third xrootd server. The results were almost immediate and quite pronounced (although somewhat expected) - with the maximum rate jumping to 1.58 GB/s (average 1.09 GB/s).



Volume Statistics



Conclusions 1

The bottleneck for our site appears to be the lack of CPU to process checksums in a timely manner.

- A back-of-an-envelope guesstimate suggests to take in data at a rate of ~40Gb/s we would need approximately 6-8 medium quality servers (~24 cores, 128GB of RAM, 25Gb NICs).
  - This is a lot more than we the 3-4 xrootd servers we originally planned (and that was for redundancy, not capacity).
  - There is a concern that at higher numbers of xrootd servers the load on the CEPHFS system would be non-negligible
    - Data is written, then re-read for the checksumming process.
    - Lucky for us we have a few in-warranty DPM disk servers that can eventually fill this gap.
- Improvements could possibly be made if we could write a plugin to have xrootd checksum “over the wire”.
  - This is available for some endpoints (HDFS), but not for plain POSIX (presumably to allow for the possibility of asynchronous writes).
  - This would also have the benefit of reducing the load of our xroot gateways on the underlaying filesystem.
- Of course there may be other config changes to improve the picture, but we believe that these will be small scale benefits..

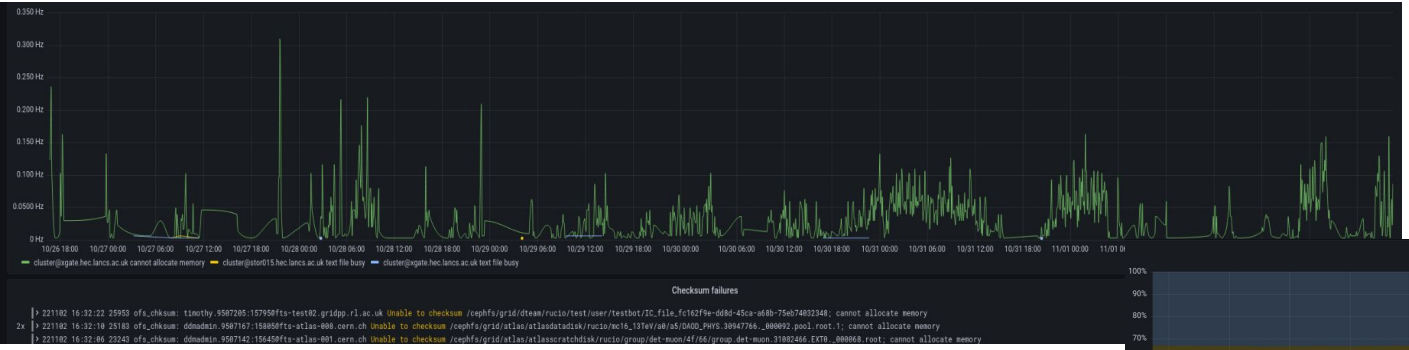
Conclusion 2

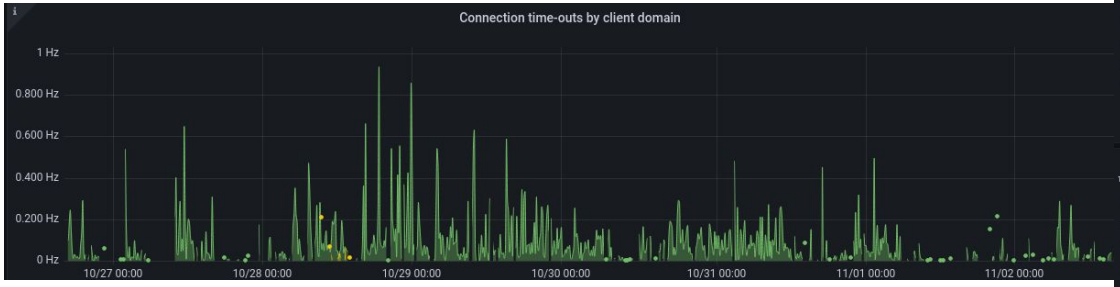
The test was incredibly useful, and revealed problems that would have only otherwise shown themselves at the height of production or in some large-scale data challenge. We plan to have a repeat performance after we've got a few more redirectors in place and polished our setup a bit more.

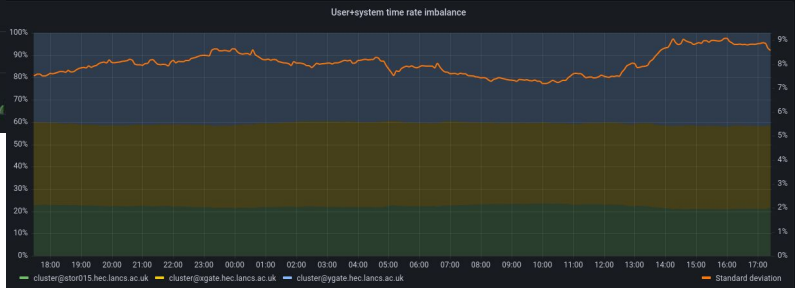
- But future tests don't need to be of the same scale - 100TB was almost certainly more then we needed.
- With coordination of the start (so data starts moving during office hours) we reckon ~10TB would be more than enough to provide useful statistics.
- The next test it would be advantageous to keep a "log" of changes and restarts.
  - server restarts noticeably muddy the transfer results, even if they're brief.
- Also I will take more screenshots - some data has a short lifespan (so I missed capturing it for this presentation).
- Of course we don't want to just fill up CERN FTS traffic with test transfers…

Monitoring Improvements

Another positive outcome of the exercise is that it helped us polish our monitoring.







Plots made using Prometheus + Grafana

Summary

- Much like how you can’t beat “real” job flows for testing compute, there is no substitute for real data flows for testing your storage.
  - Just crank things up to 11.
- The atlas data management system can easily provide this tool.
  - But please liaise with DDM and your local atlas reps first!
- For us, the bottleneck was no our network bandwidth or our file system performance, but our gateway CPU.
  - Not quite what we expected.
  - More gateway servers would help, but “over the wire” checksumming could help more.

Thanks for Listening!

References:

- Lancaster Xroot Config Github: https://github.com/mdoidge/lancsxroot
- Monit FTS Dashboard
- xrootd documentation: https://xrootd.slac.stanford.edu/docs.html
  - In particular:
  - https://xrootd.slac.stanford.edu/doc/dev54/xrd_config.htm#_Toc88513998
- Datasets used:
  - mc20_13TeV:mc20_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_PHYSVAL.e6337_s3681_r13144_r13146_p5057_tid28829126_00
  - mc20_13TeV:mc20_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_PHYSVAL.e6337_s3681_r13144_r13146_p5169_tid29398576_00
  - mc20_13TeV:mc20_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_PHYSVAL.e6337_s3681_r13144_p4856_tid27394621_00
  - mc20_13TeV:mc20_13TeV.410470.PhPy8EG_A14_ttbar_hdamp258p75_nonallhad.deriv.DAOD_PHYSVAL.e6337_s3681_r13144_r13146_p4931_tid27801838_00