ENDIT 2.0

NeIC NT1 Manager
Mattias Wadenstein
<maswan@ndgf.org>

2023-11-14
NDGF AHM, Linköping, Sweden

NORDIC E-INFRASTRUCTURE COLLABORATION

Overview

- What is ENDIT
- Design
- Changes in 2.0
- Benchmark results

What is ENDIT

- Efficient Nordic Dcache Interface to TSM
  - Or, well, IBM Storage Protect as it is called these days
- A package to use a TSM controlled tape library as an HSM backend for dCache
  - Most of our sites run TSM for backups, so using existing infra
- Designed for efficiency and speed
- In production use by NDGF-T1 for a decade
  - Several other sites also use the plugin, either as is or modified

ENDIT design ideas

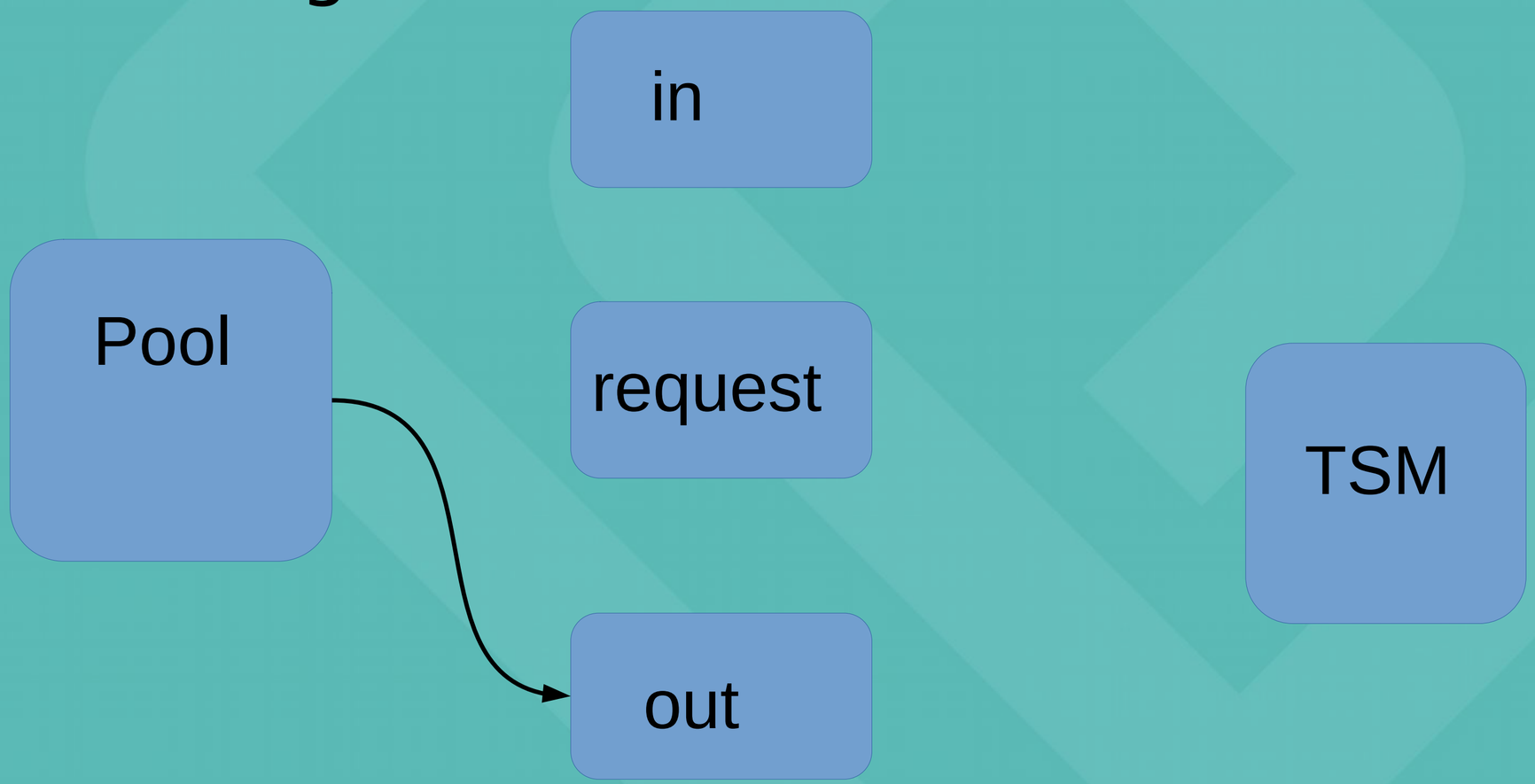
- Using the dsmc command line client to get/put/rm
  - Assumption: Unlikely to lose data due to weird corner cases
  - Using intermediate directories to create batching for efficiency
- Thresholds for when to act in size, time, etc
- Use of dedicated tape read and write nodes
  - Mostly a consideration for performance with small SSD-based nodes for high throughput
  - At NDGF we then do a pool2pool copy for reads, so the clients hit the same disk pools as disk data for slow reads

ENDIT parts

- dCache Plugin
  - A dCache HSM plugin that is used instead of the reference script plugin, this is a jar loaded into the dCache pool and configured by “hsm register ...”
- Endit daemons
  - A set of scripts that check for requests, batch them into good sized groups, and then issues dsmc archive/retrieve/delete commands
  - Configured with endit.conf
- Auxiliary script
  - Tape hints generator, tells the retriever how to split requests into one retrieve per tape for parallelism etc

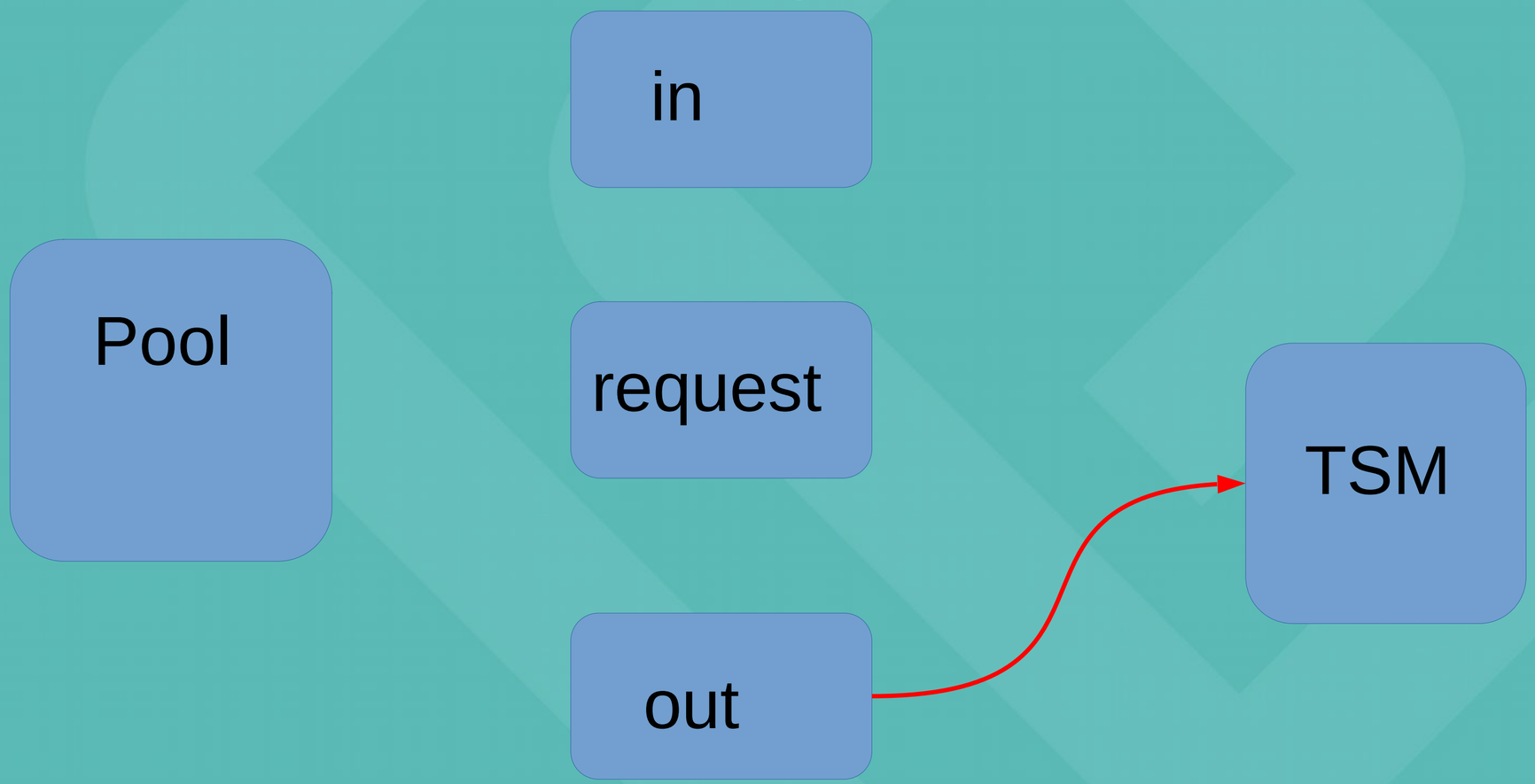
ENDIT design

- Put, step 1: A hardlink is created in “out” for the file staged when dCache flushes it



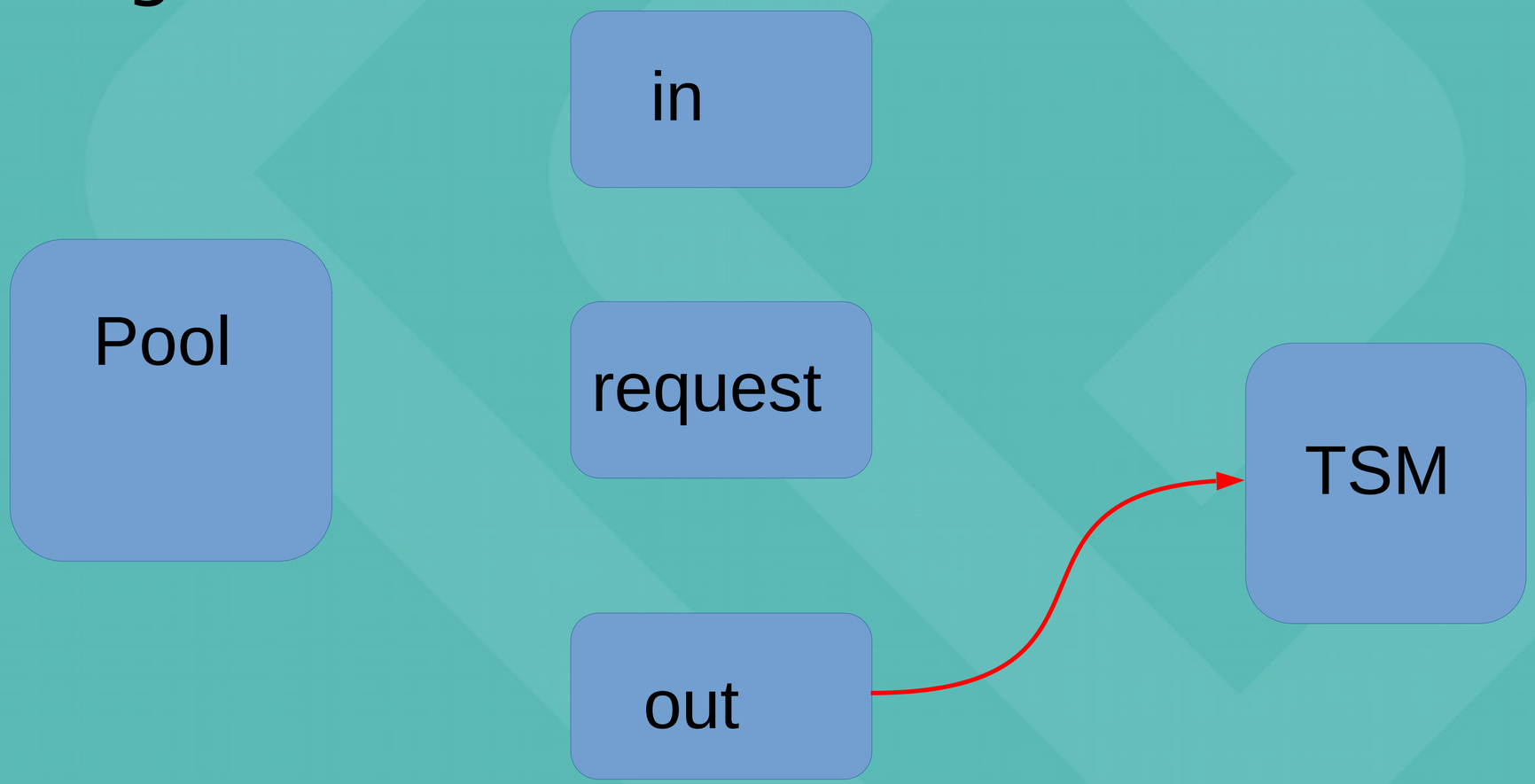
ENDIT design

- Put, step 2: Time passes. When there is more than X GB files or Y time, dsmc archive -delete out/*



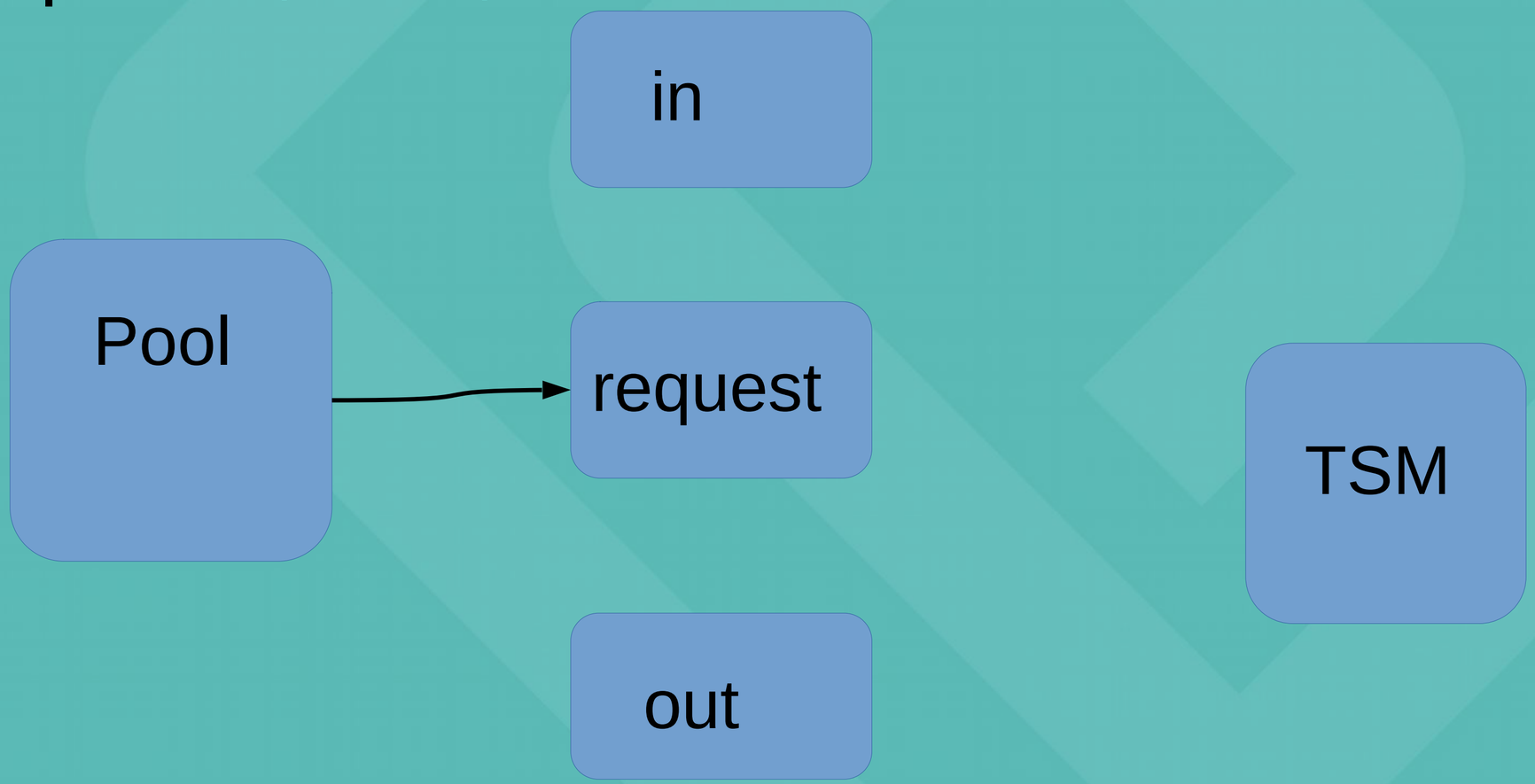
ENDIT design

- Put, step 3: the ENDIT plugin discovers that the file is gone from out and considers it successfully put



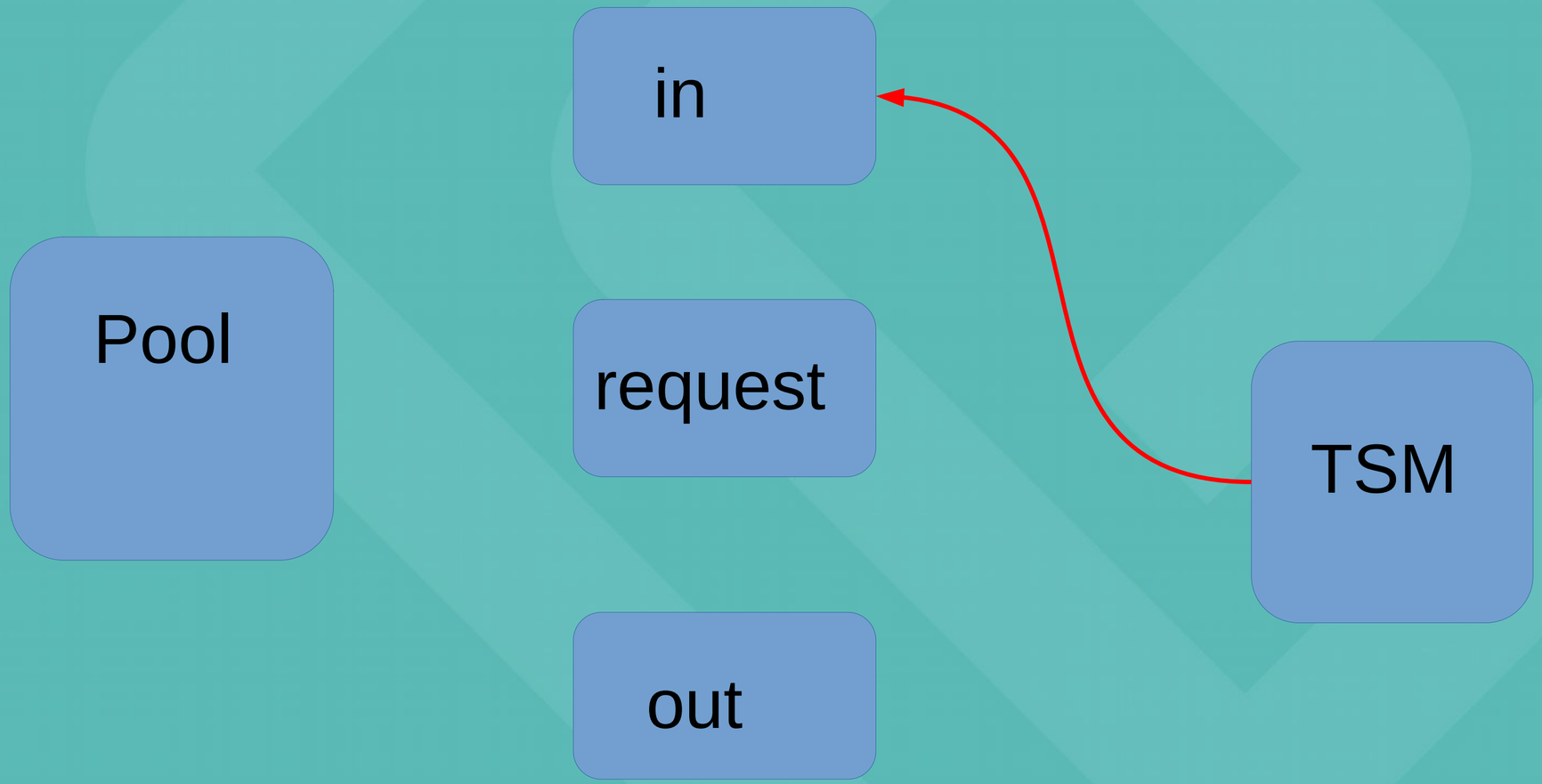
ENDIT design

- Get, step 1: The plugin creates a request file with pnfsid, size, etc



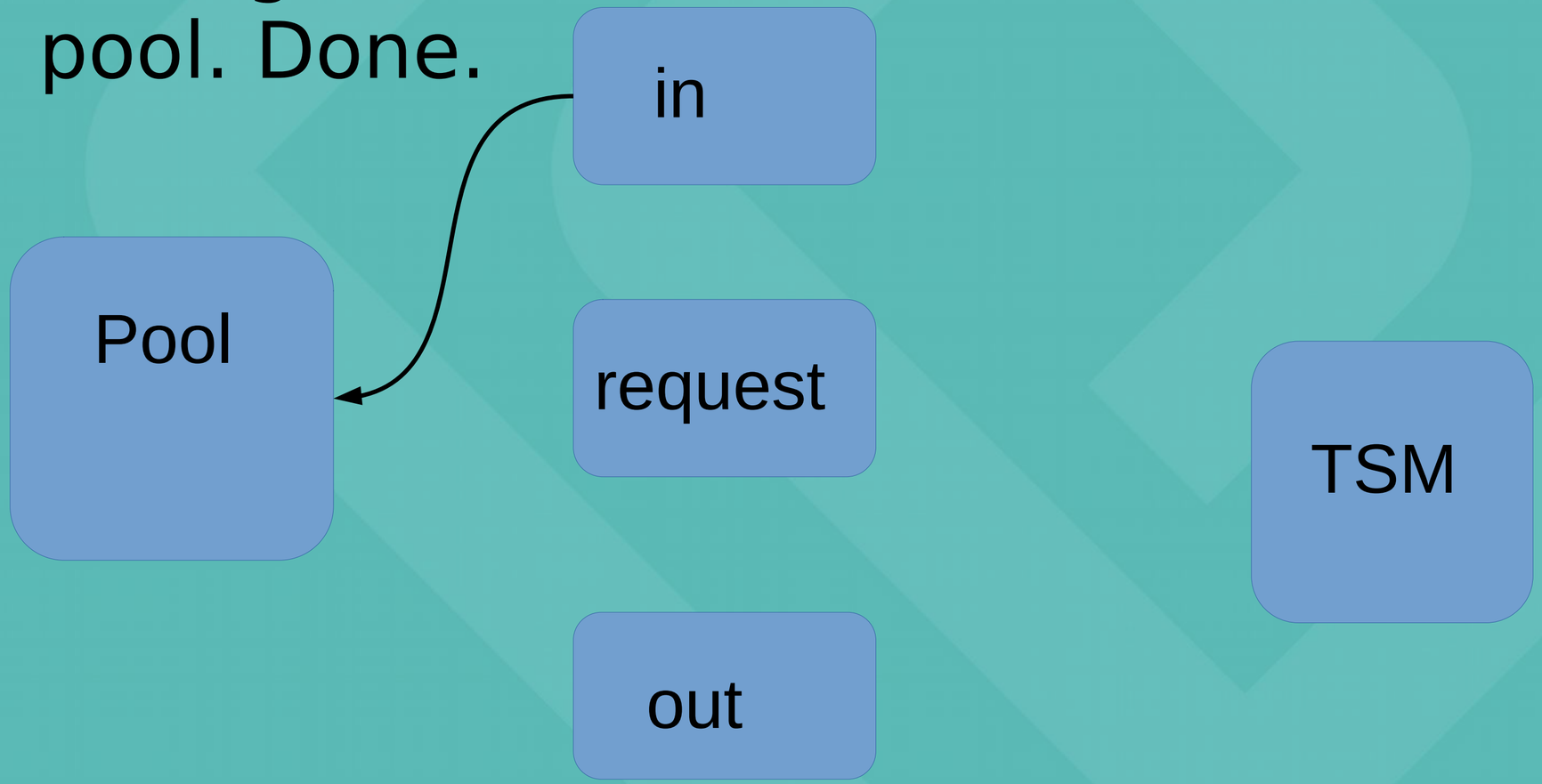
ENDIT design

- Get, step 2: Time passes, X or Y then the endit daemon retrieves the files from TSM to in/



ENDIT design

- Get, step 3: When the plugin discovers a file with the right name and size in “in/”, rename it into the pool. Done.



ENDIT 2.0 Changes

- Create an option to not reserve space for files in the dCache pool until just before the rename() from the in/ directory
  - Necessary to be able to push sufficient number of requests to ENDIT daemons to get good throughput
  - Required adding a buffer space and breaks in the endit daemons
    - Can break pools (filling filesystems) if you run new plugin with old scripts!
- Create a json file with attributes in the out/ directory for writes
  - Not used by our endit daemons, but makes life easier for others
  - Hopefully not a major performance impact
  - Can break existing scripts!

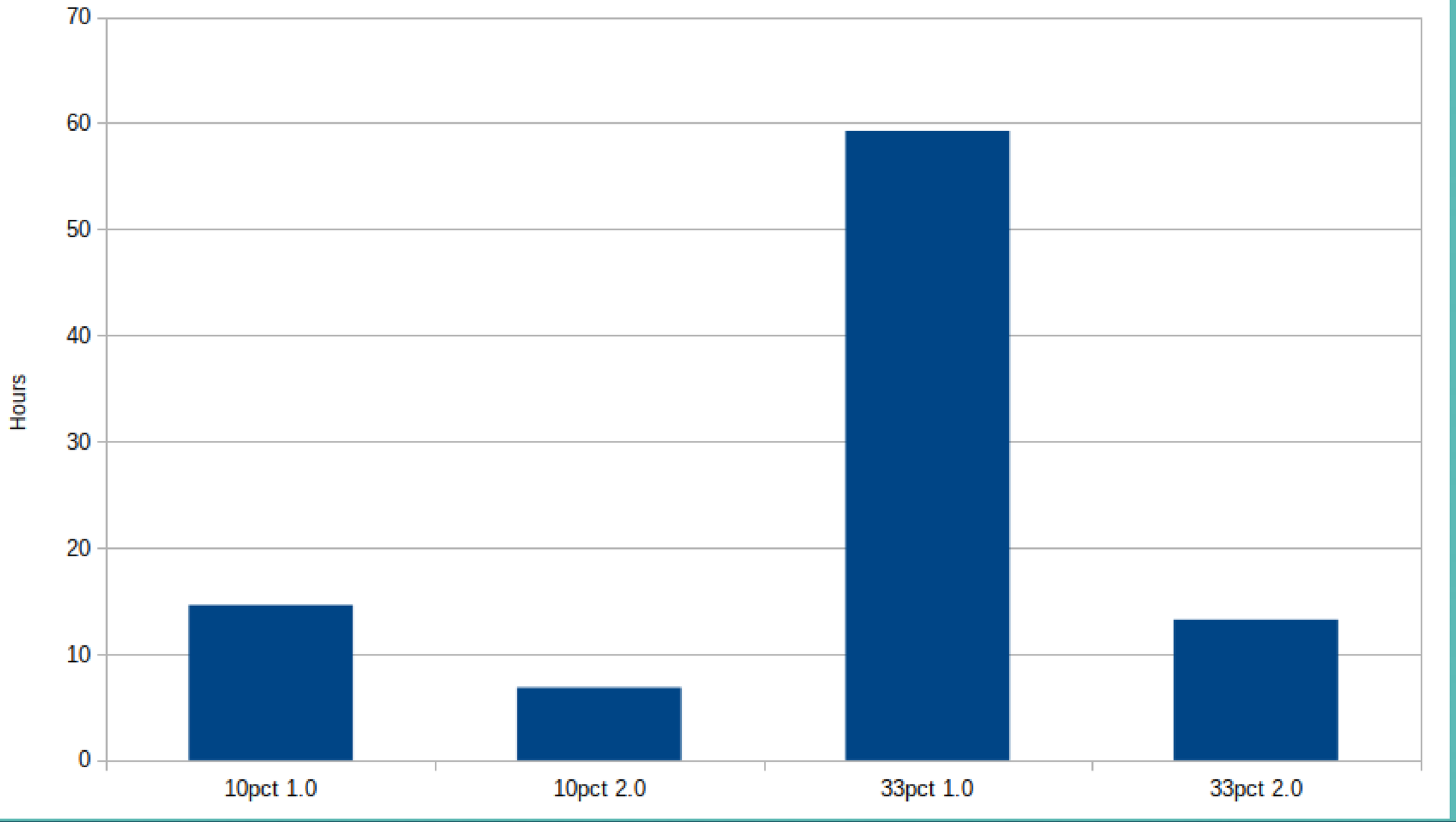
dCache plugin

- Instead of a HSM script like most dCache installations, we use a dCache plugin
  - https://github.com/neicnordic/dcache-endit-provider/
  - AGPL just like dCache
  - Just unpack the plugin in the plugin directory
  - Then configure through the dCache admin interface
- Much better scalability than the script
  - Tested to 100k outstanding read requests per pool
  - Can do restores as fast as the rest of dCache can handle it (probably latency bound to namespace from a single pool)

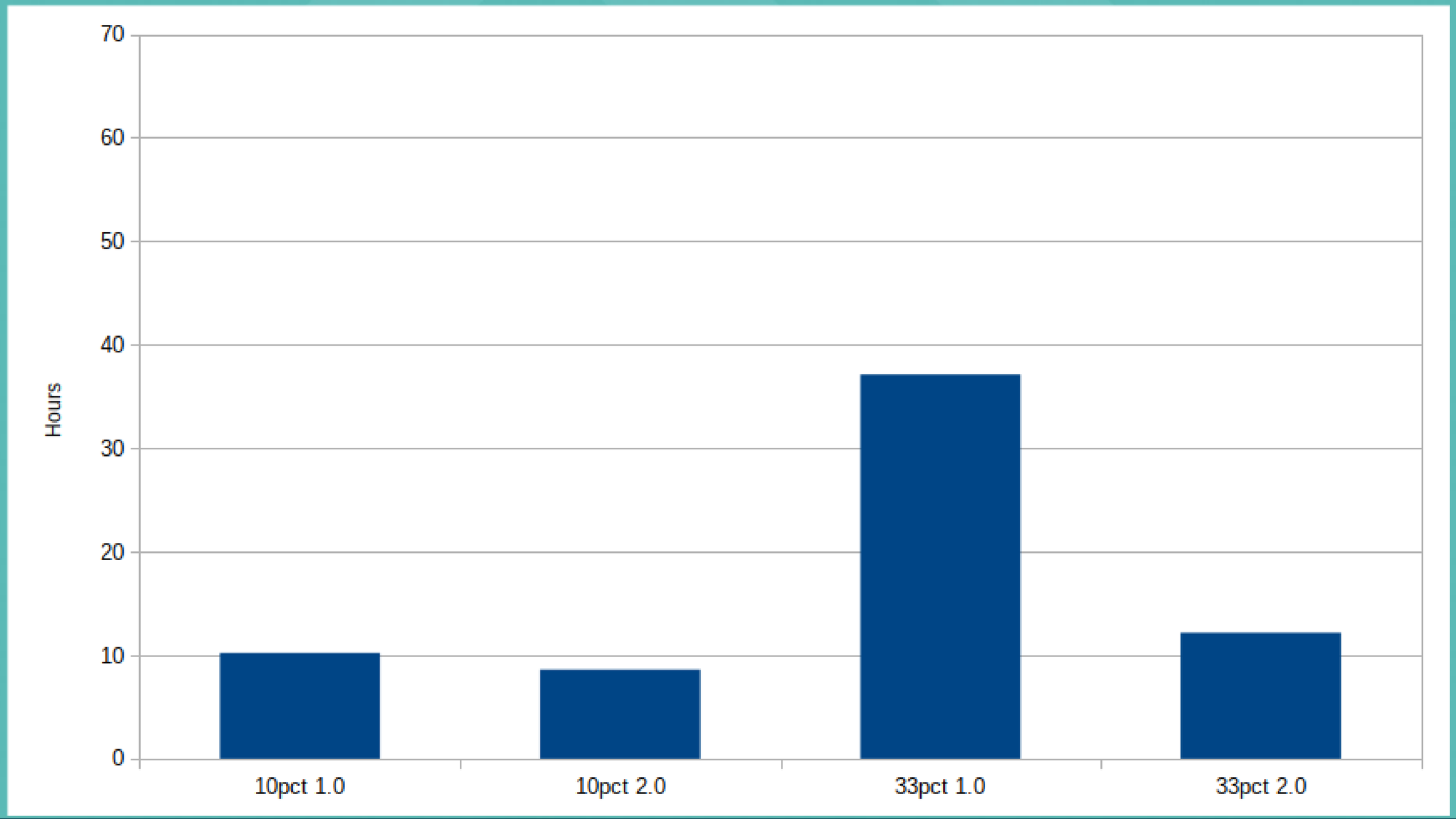
Benchmarks

- Artifical benchmark:
  - 3 full tapes and 3 half-full tapes on last generation jaguar (TS1160)
  - Reading back a random selection of files per tape (10% or 33%)
  - Requests either issued tape by tape, or in randomized order
- Benchmark run against a tape library with production loads
  - Some variance expected since tape drives might be busy doing other stuff

Restores issued tape by tape



Random order



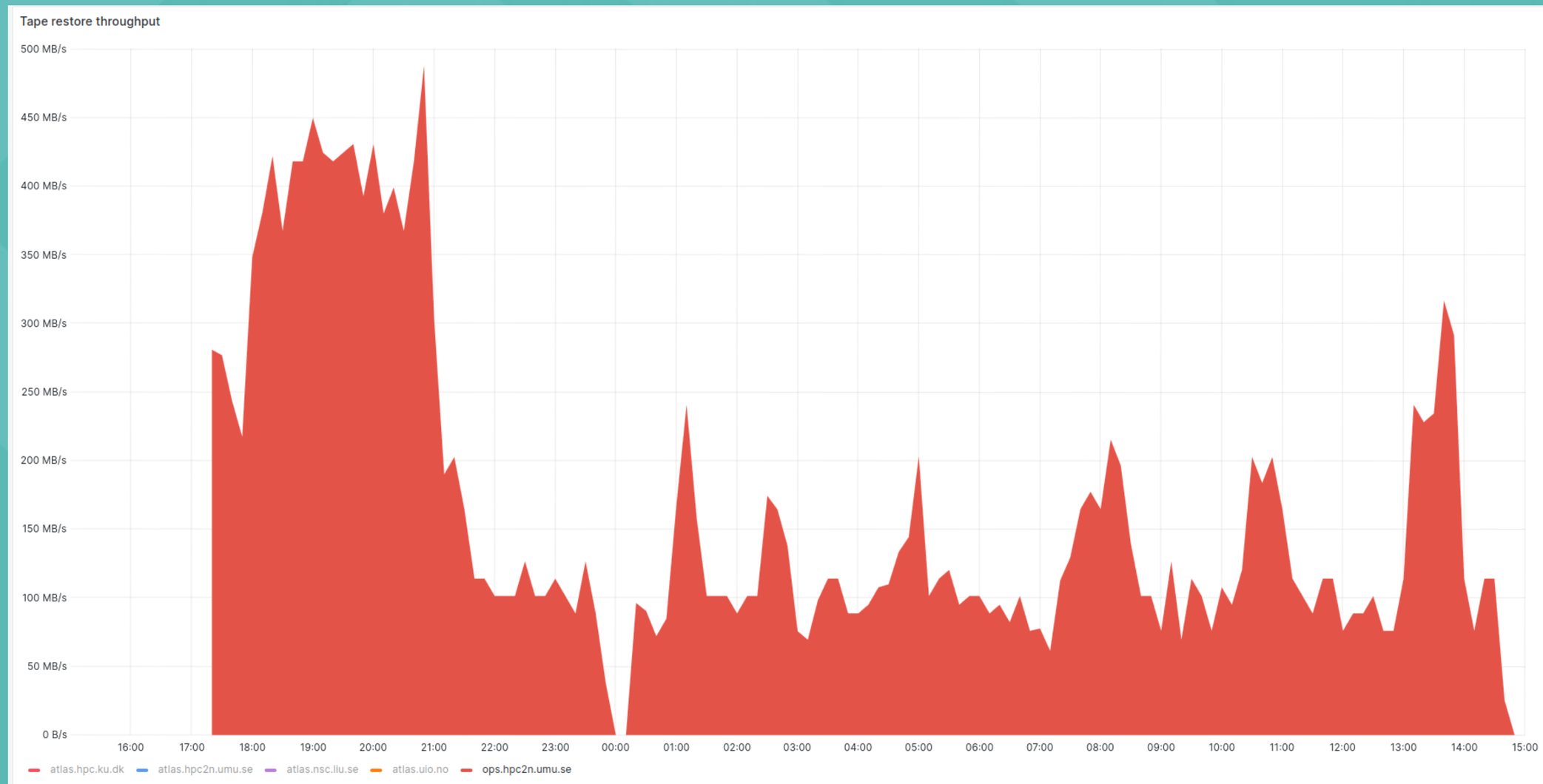
Benchmark hroughput graph

- 10% recalls, tape by tape. First 2.0, then 1.0

400

MB/s

100

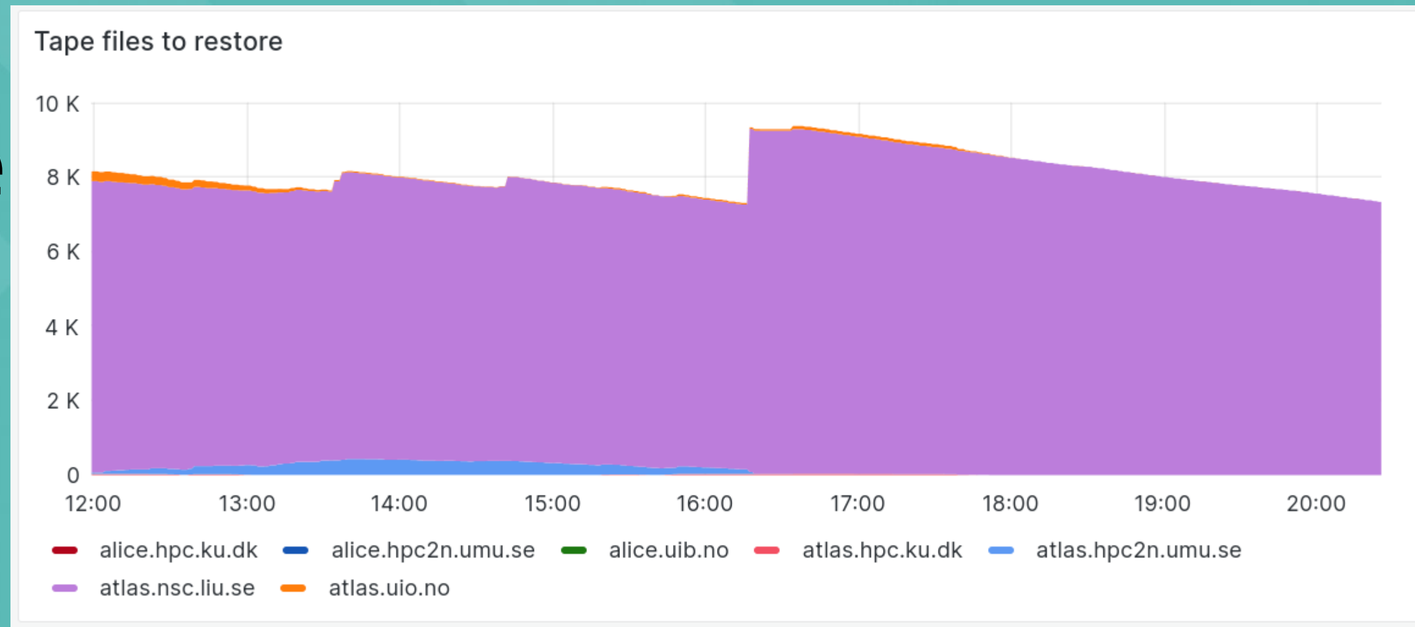


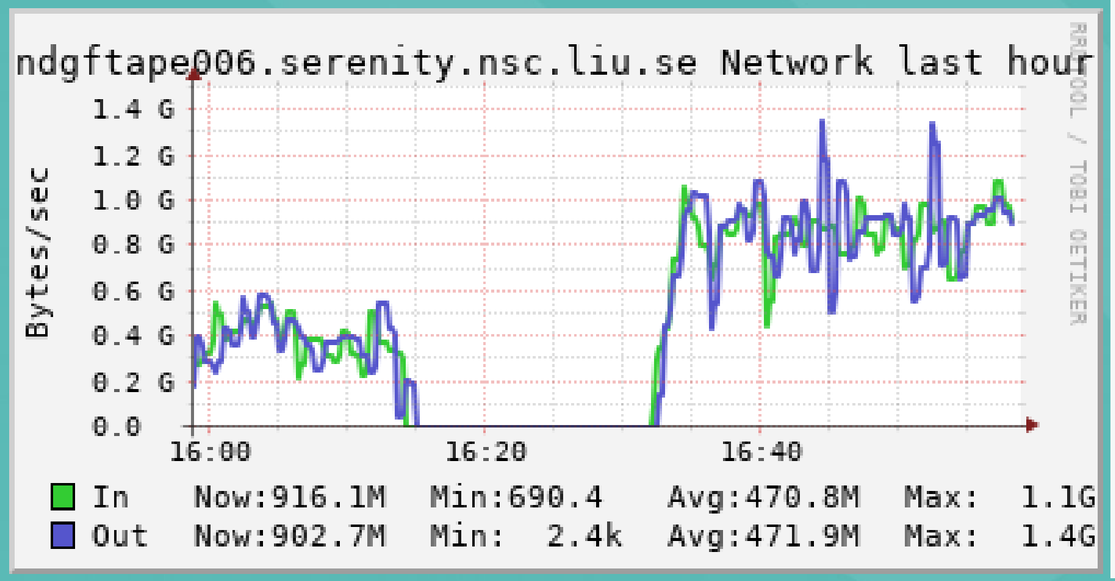
Benchmark comments

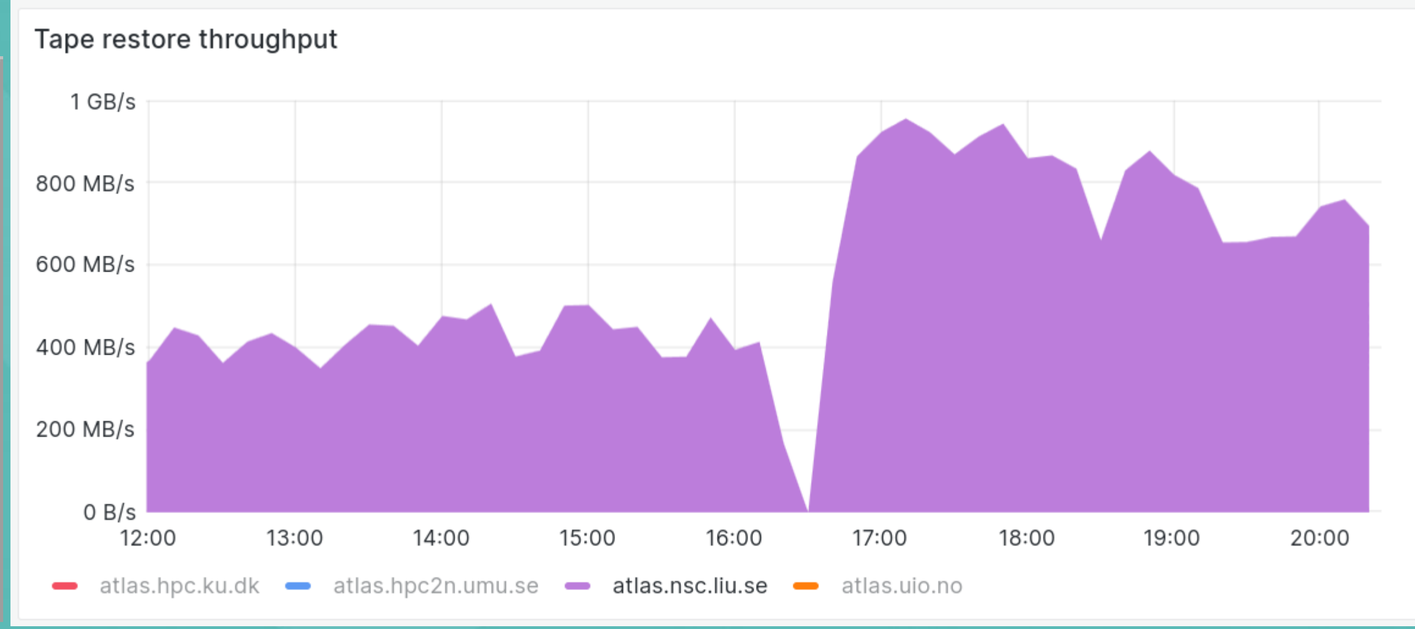
- Ordered restores worst case for 1.0
  - Reserved space for files on the first tape blocks all others
  - Only one drive is used most of the time
- Random order a bit more lenient
- Production recalls are in between these extremes
- 2.0 will outpace 1.0 when recall size >> pool size
  - The bigger the recall, the bigger the performance gap

Upgrade in production

- After fixing write issue
- NSC had 8k backlog:
- Roughly doubling of read throughput

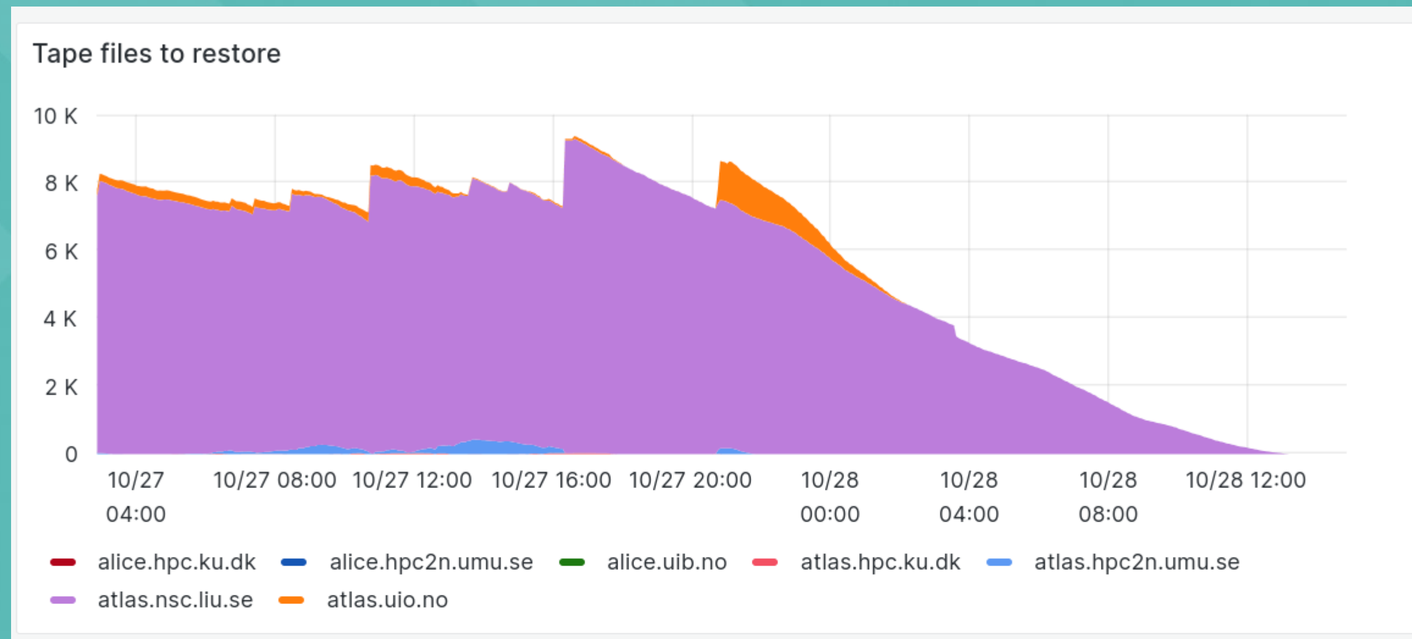


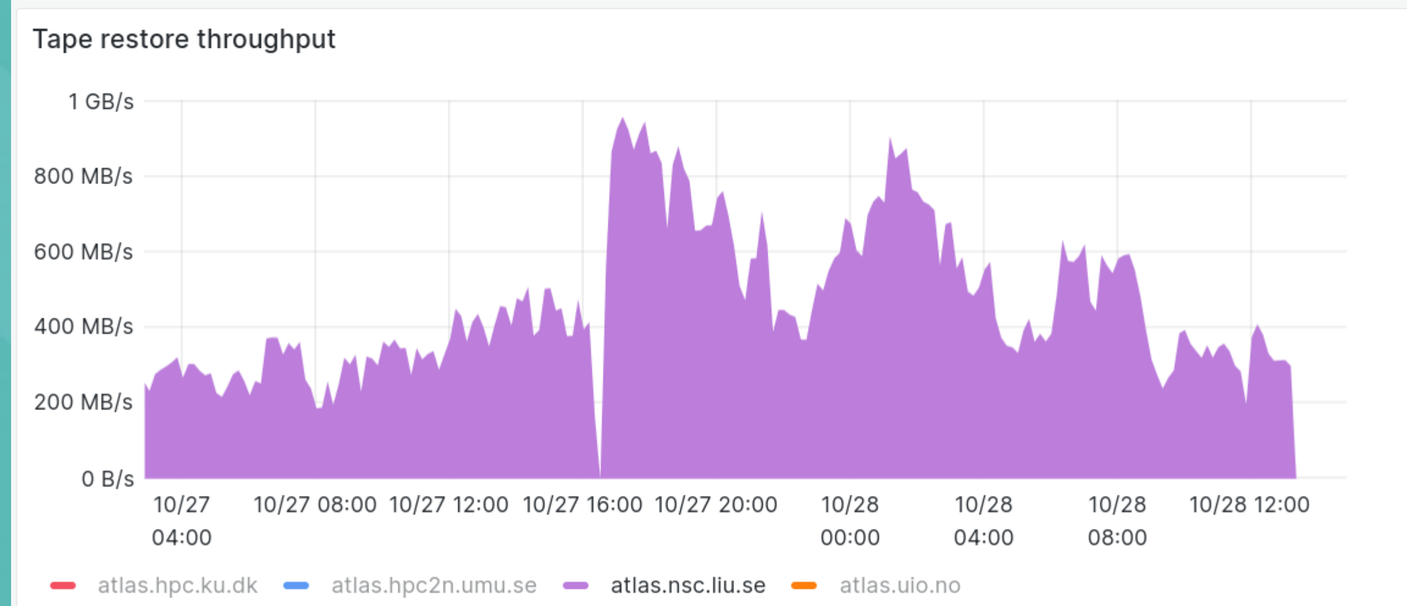




Upgrade in productoin

- Longer timescale
  - Pre-upgrade 200-400MB/s
  - Post-upgrade 400-800MB/s





Next steps

- Upgrade all our tape sites
- Request for a couple of runs of ATLAS tape carousel benchmarks
  - Investigate best settings for “max outstanding recalls” as well as our general performance
- Documentation and verify good defaults
- Package up new code in a proper release
  - Non-Nordic users would likely want a config switch for space reservation or not

Questions?

NORDIC E-INFRASTRUCTURE COLLABORATION