ENDIT 2.0

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

2023-10-18
HEPiX, Victoria, Canada

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

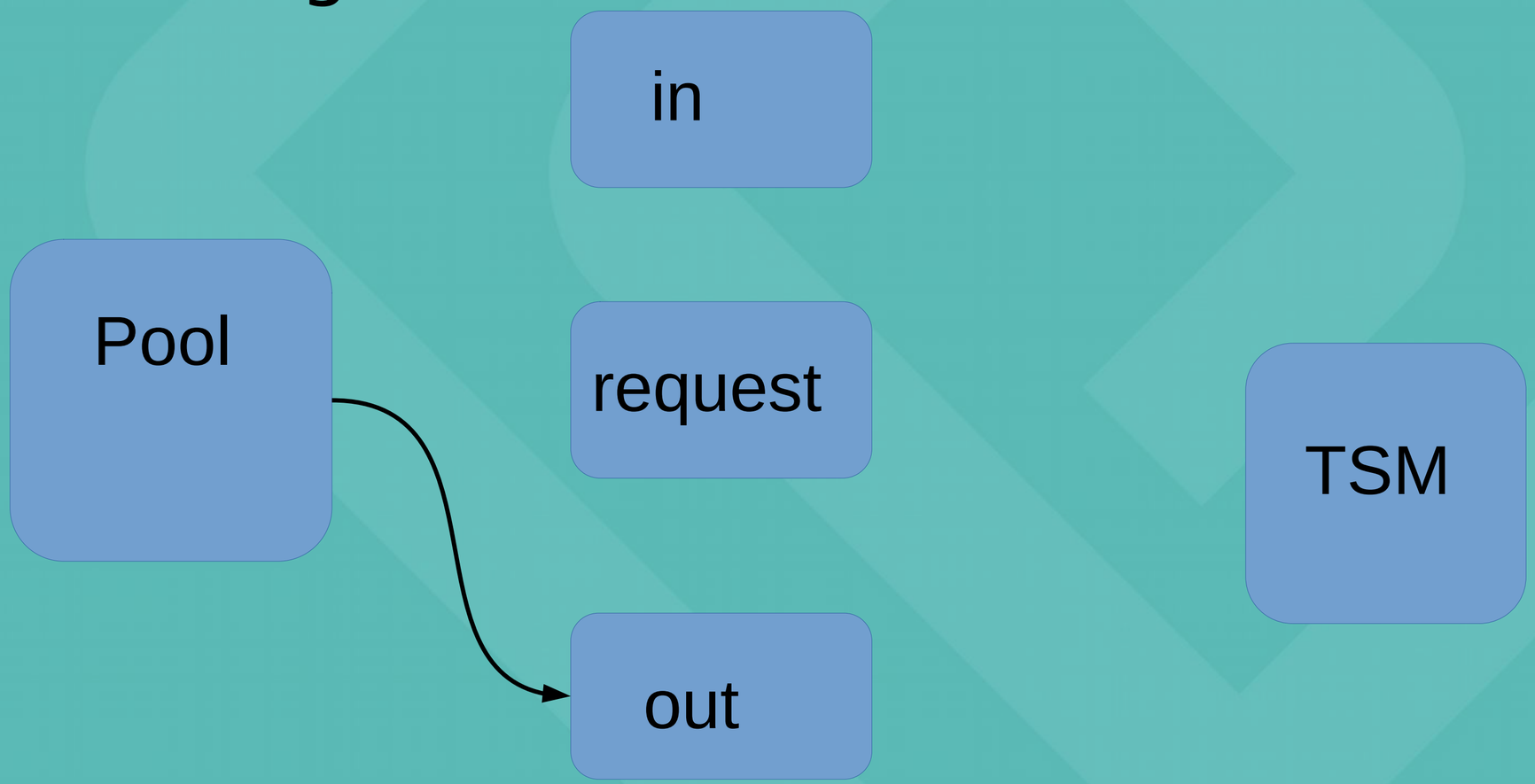
- Using the dsmc command line client to get/put/rm
  - Assumption: Unlikely to lose data due to weird corner cases
  - Using intermediate directories to create batching for efficiency
- Thresholds for when to act in size, time, etc
- Use of dedicated tape read and write nodes
  - Mostly a consideration 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

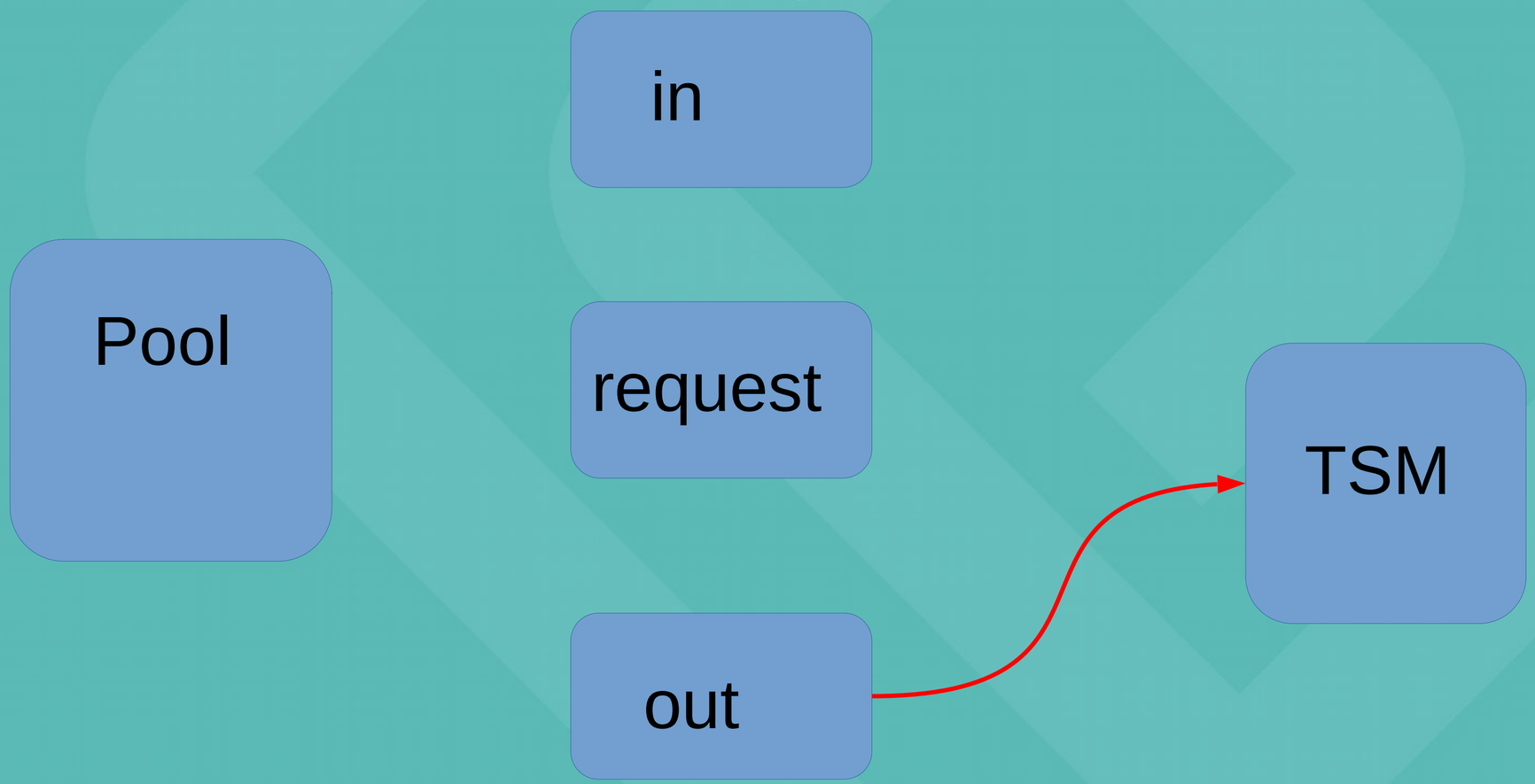
ENDIT design

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



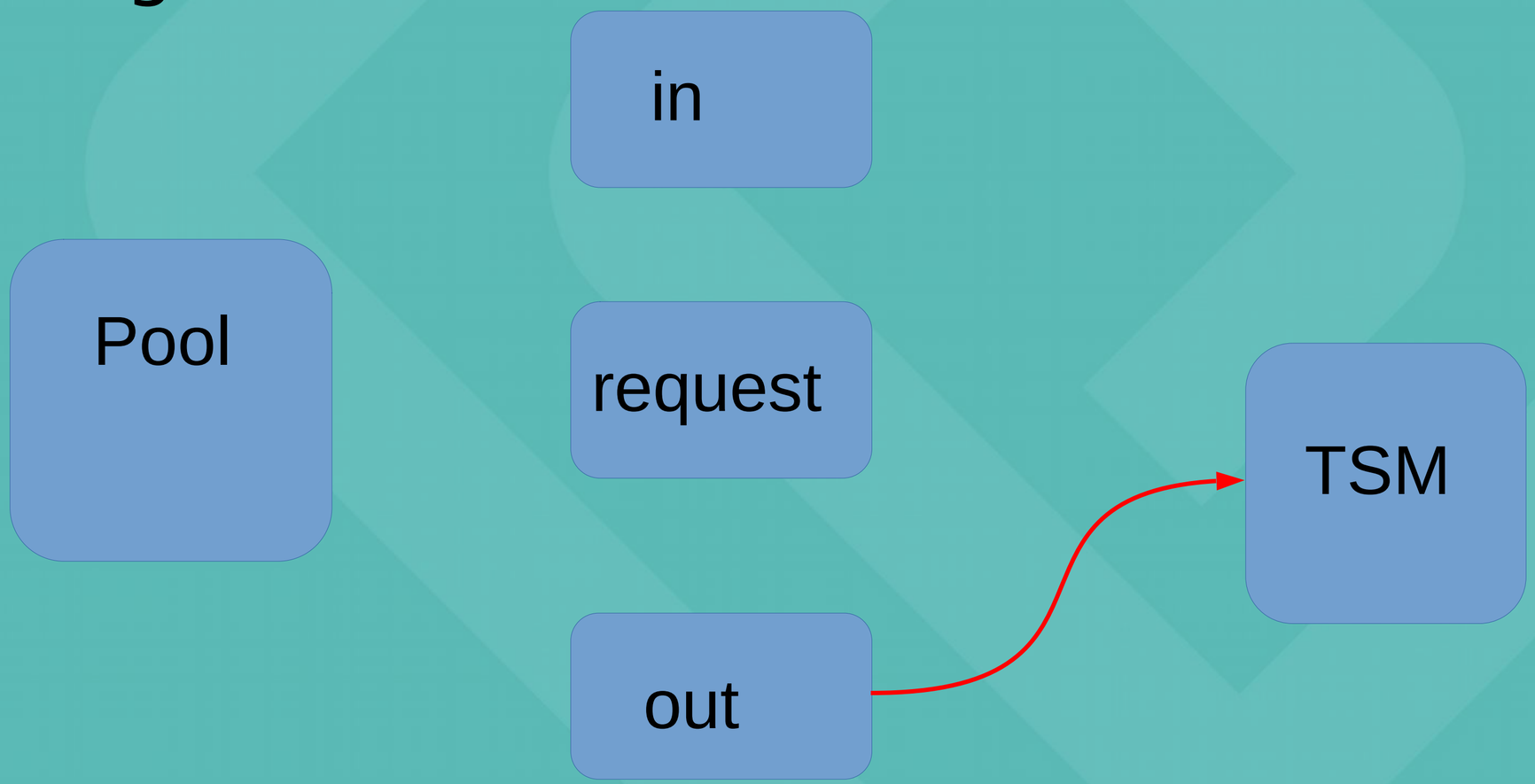
ENDIT design

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



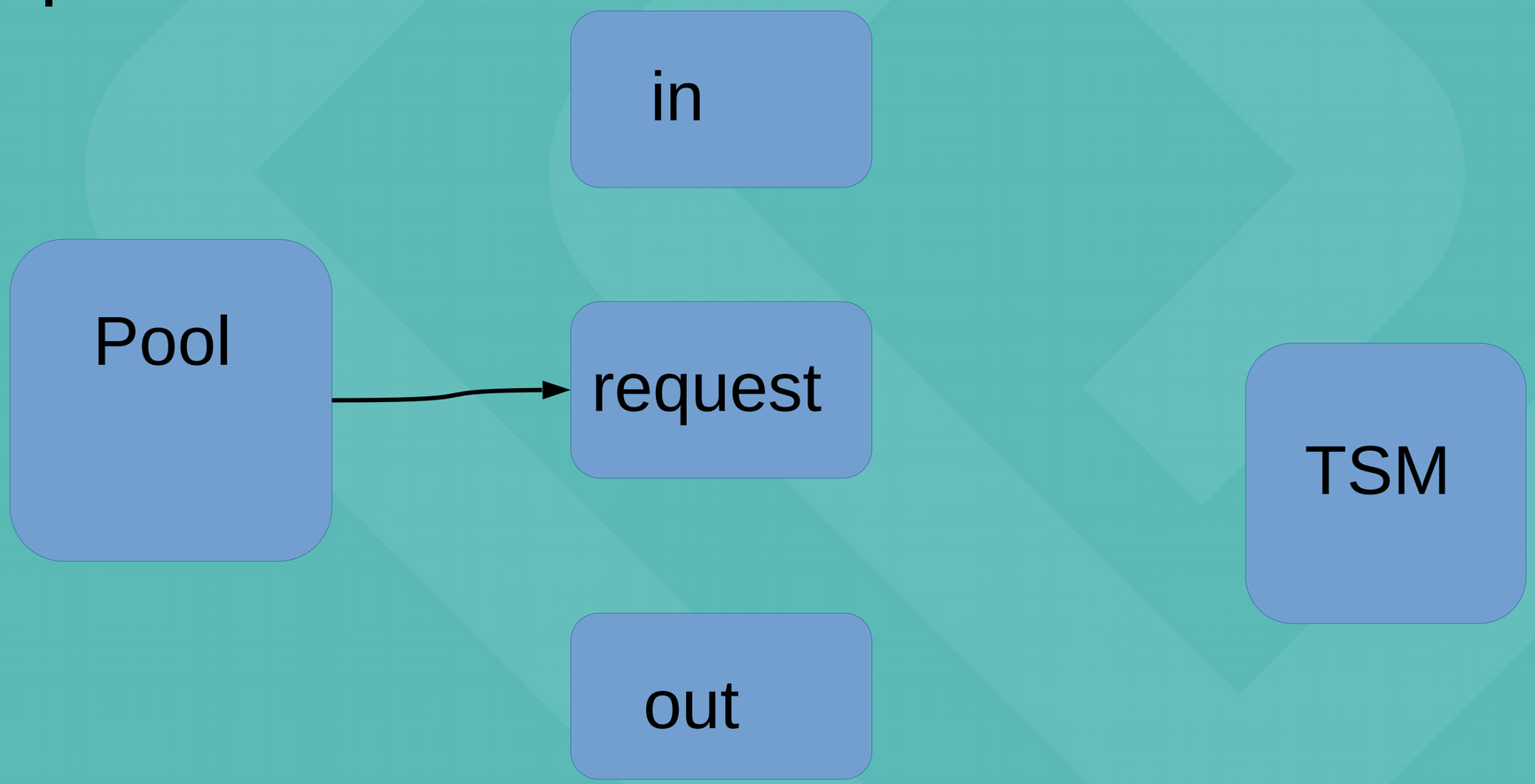
ENDIT design

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



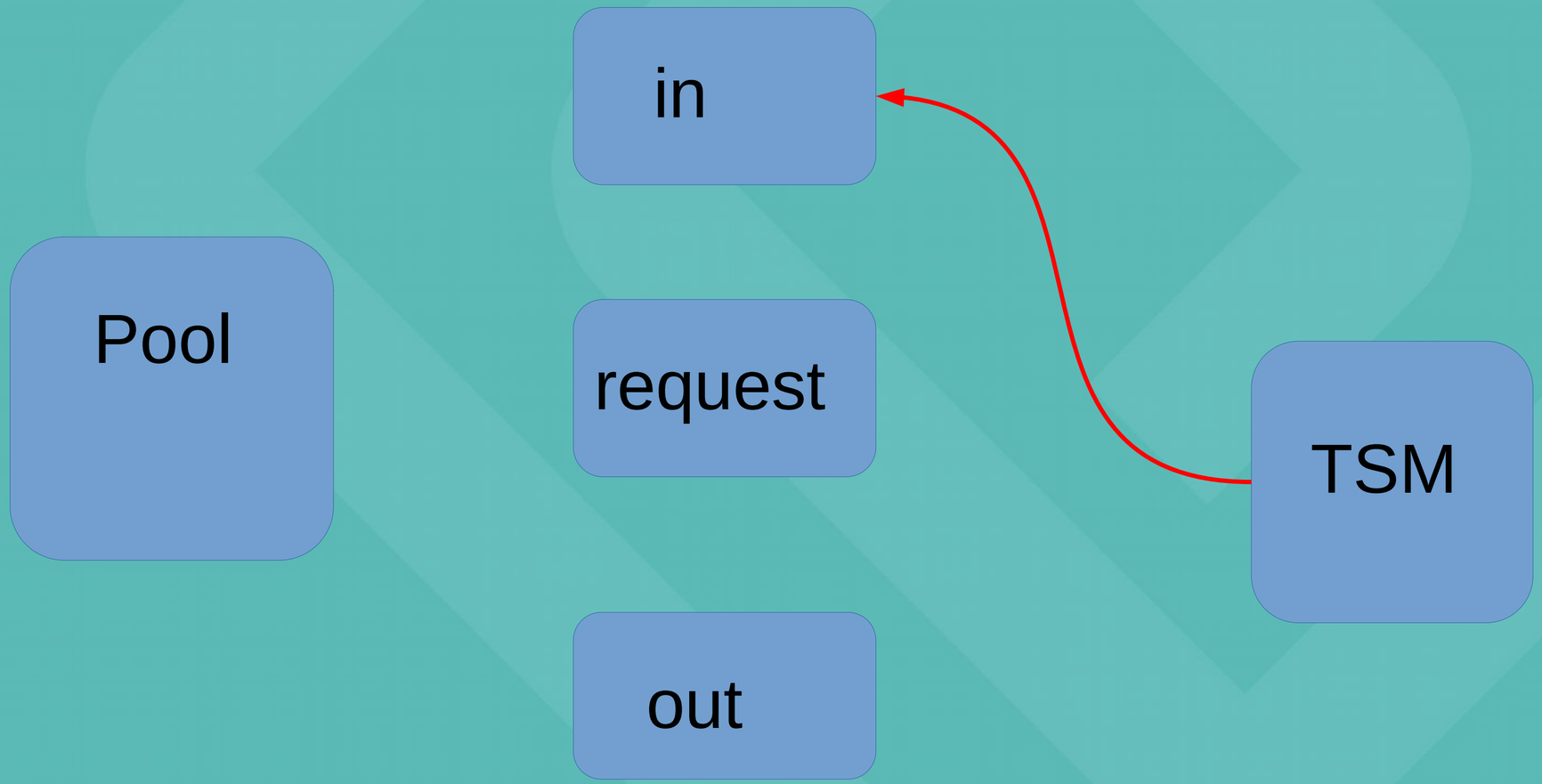
ENDIT design

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



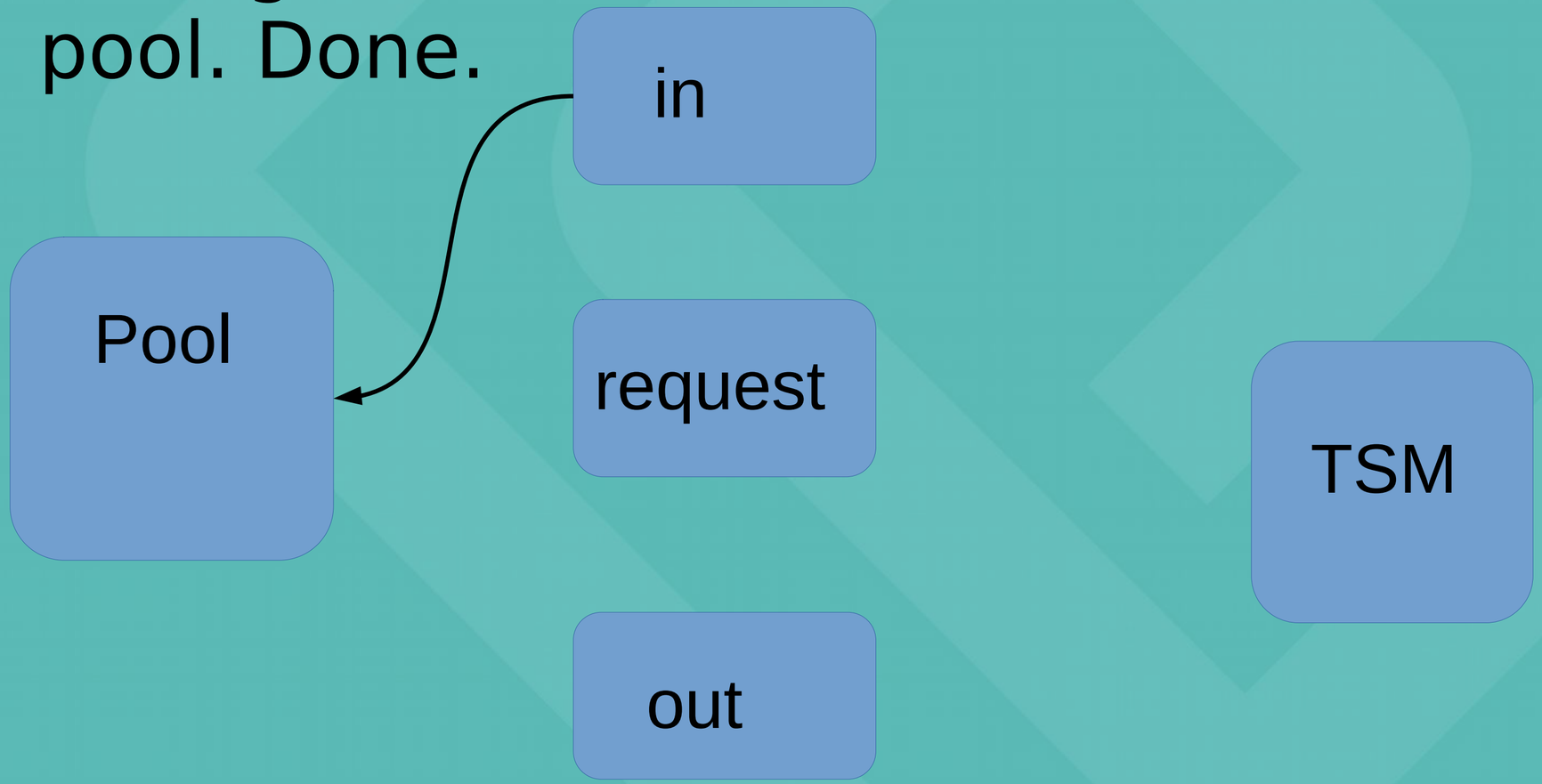
ENDIT design

- 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.



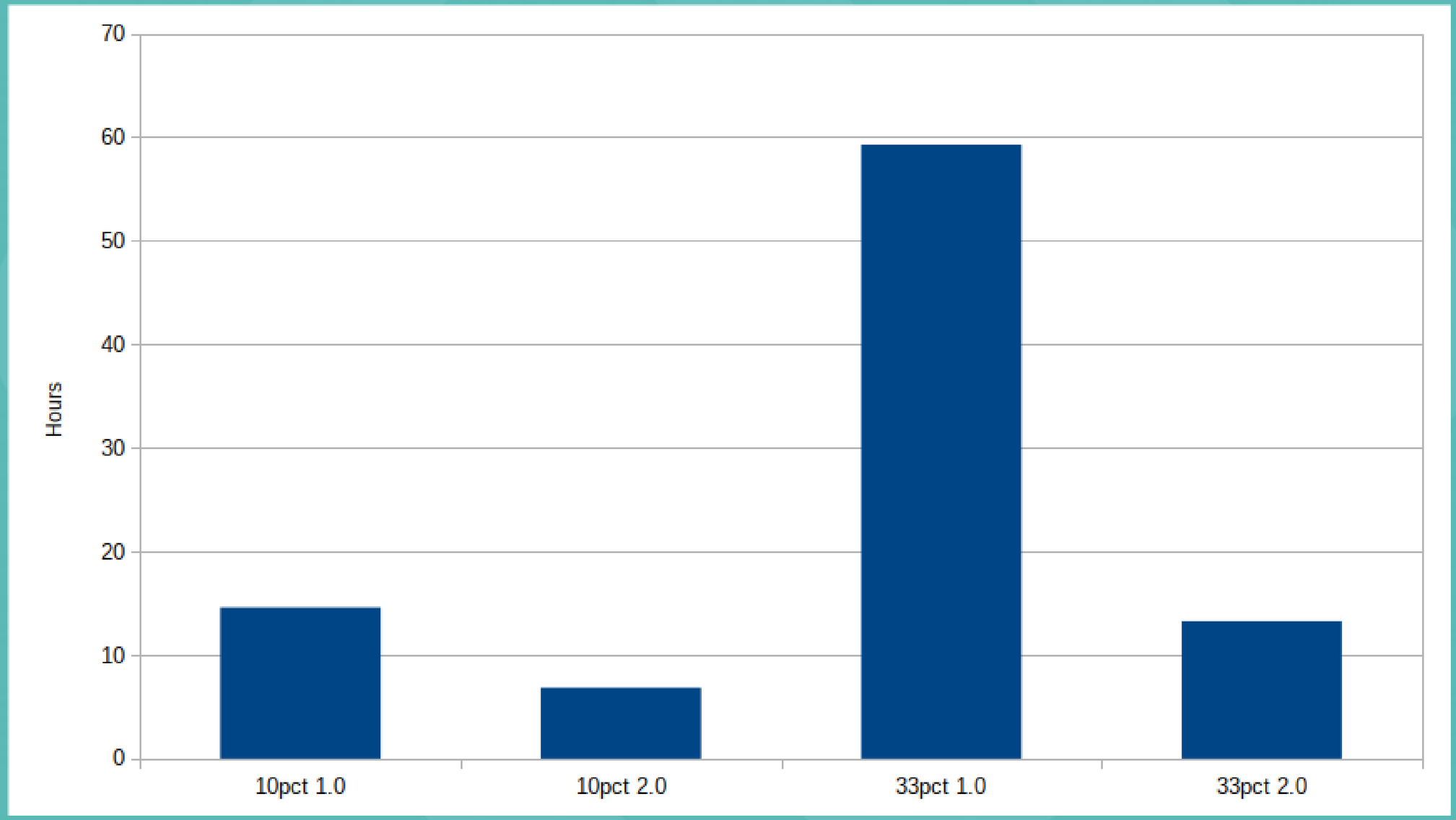
ENDIT 2.0 Changes

- 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 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!

Benchmarks

- Artificial 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



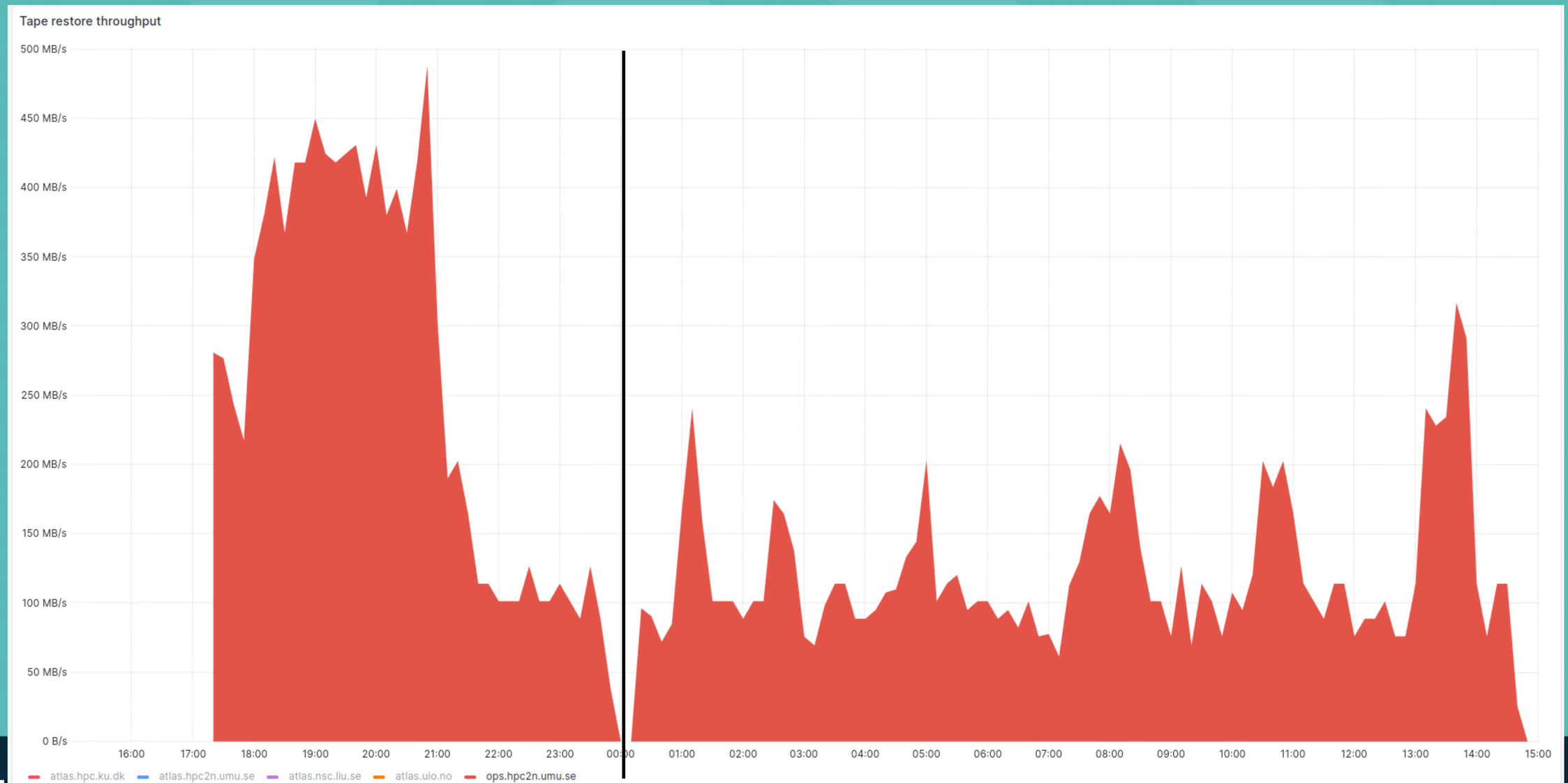
Throughput graph

- 10% recalls, ordered. First 2.0, then 1.0

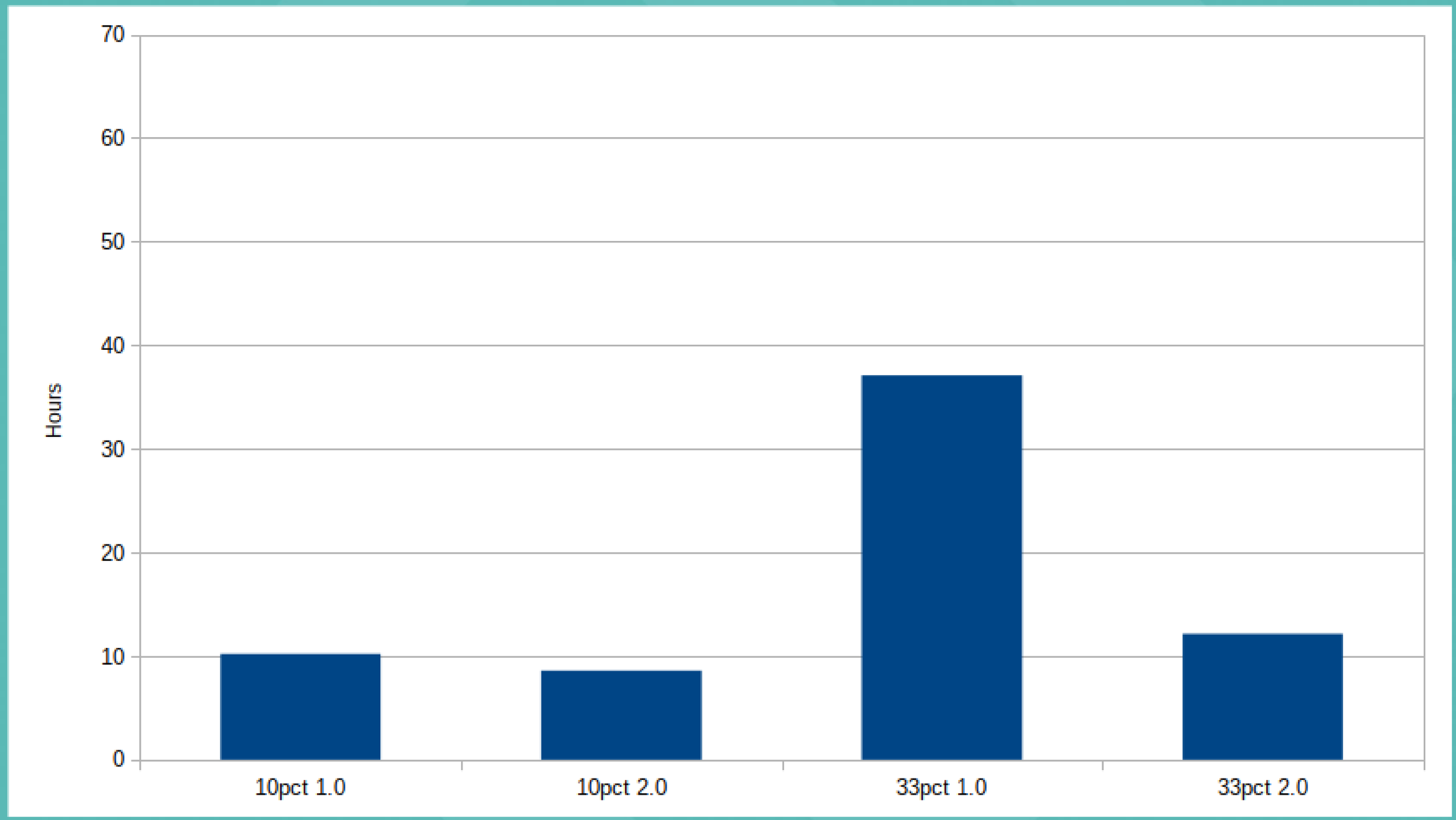
400

MB/s

100

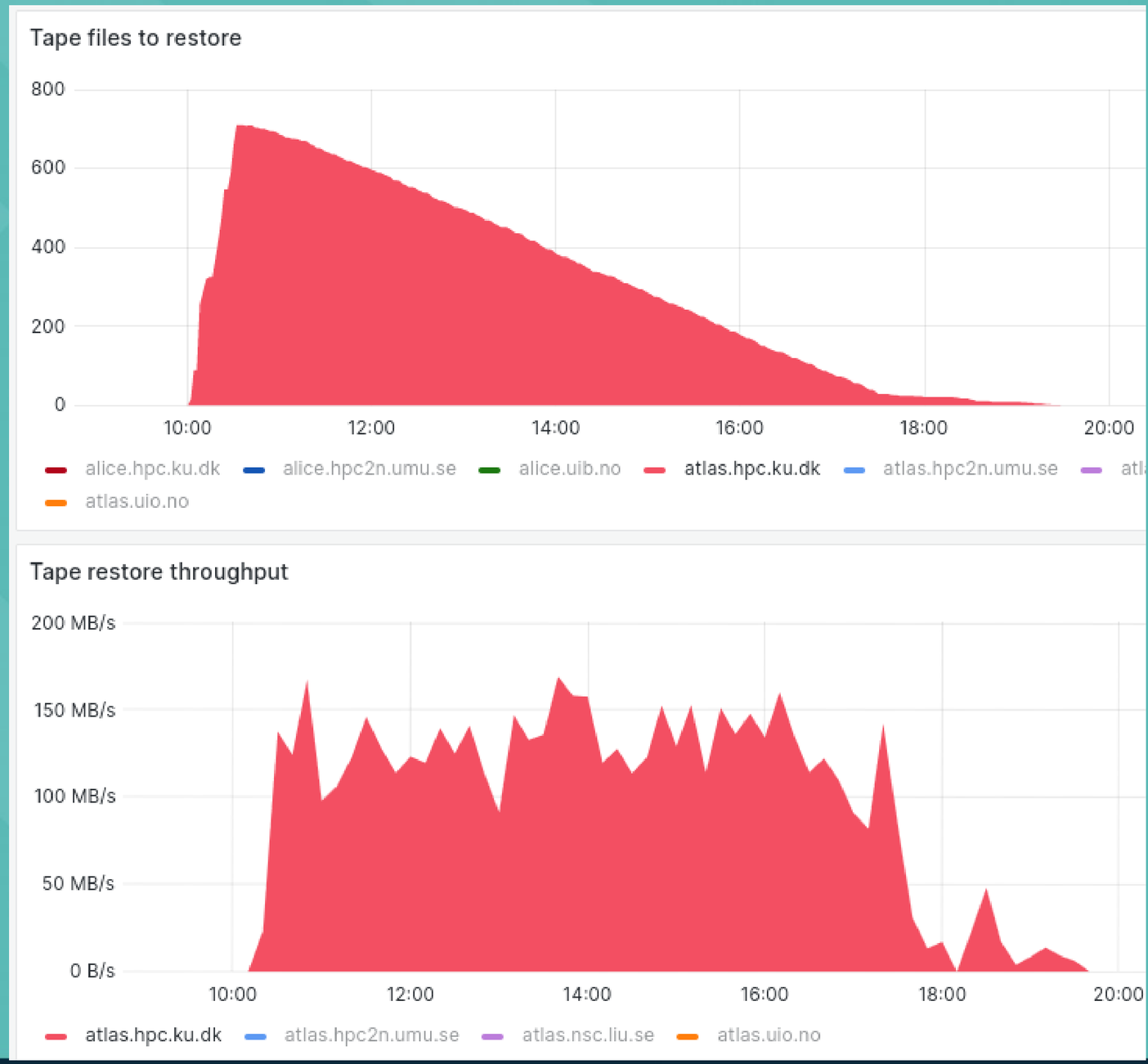


Random order



Production use

- On 1 of 5 tape sites
- Restores working
- No big queues yet
  - Queue < pool size:
    no difference 1.0 vs 2.0
- Write problems?
  - Discovered this week
  - No significant changes to write codepaths
  - Diagnosis not yet complete
  - Worked 2 weeks ago!



Benchmark comments

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

Future work

- A bit more documentation
- Good default values
- Deploy on all our tape pools in production
  - Currently 1 out of 5 tape libraries
- Decide if we can make a mode switch option or keep the 1.0 series for old behaviour
- Proper release tagging etc
- Figure out why we somehow broke writes with no significant changes on the write side??

Questions?

NORDIC E-INFRASTRUCTURE COLLABORATION