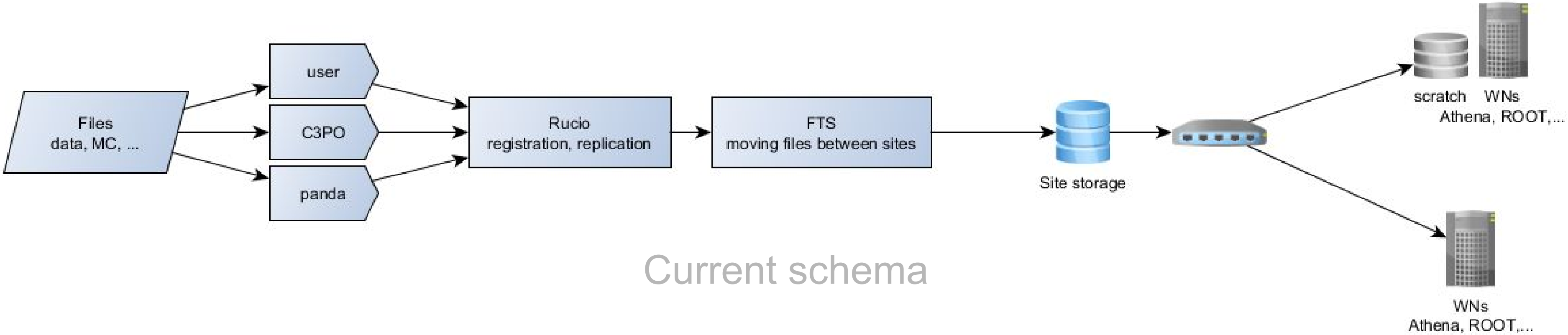
# WAN/LAN IO

Past, future

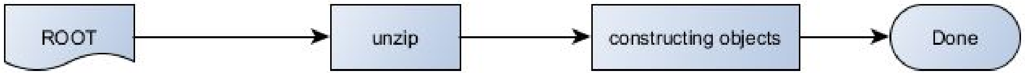
**Ilija, Vakho, Tomas**

# IO chain review

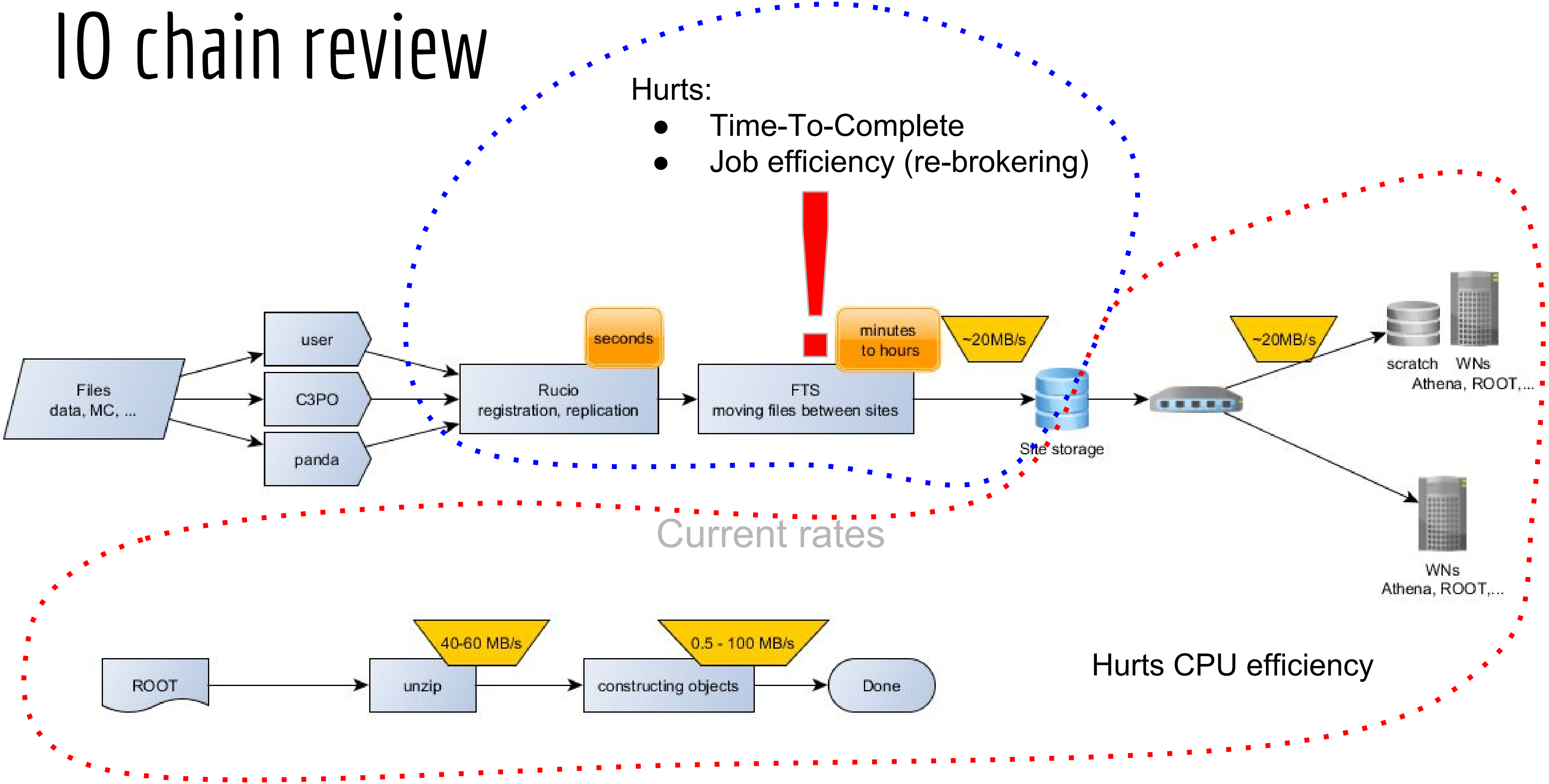
Our business is moving data as quickly as possible to the users code.



Current schema



# IO chain review



Hurts:

- Time-To-Complete
- Job efficiency (re-brokering)

Current rates

Hurts CPU efficiency

# IO chain - important facts

Average file size ~200 MB

Inter-site links 10-100Gbps

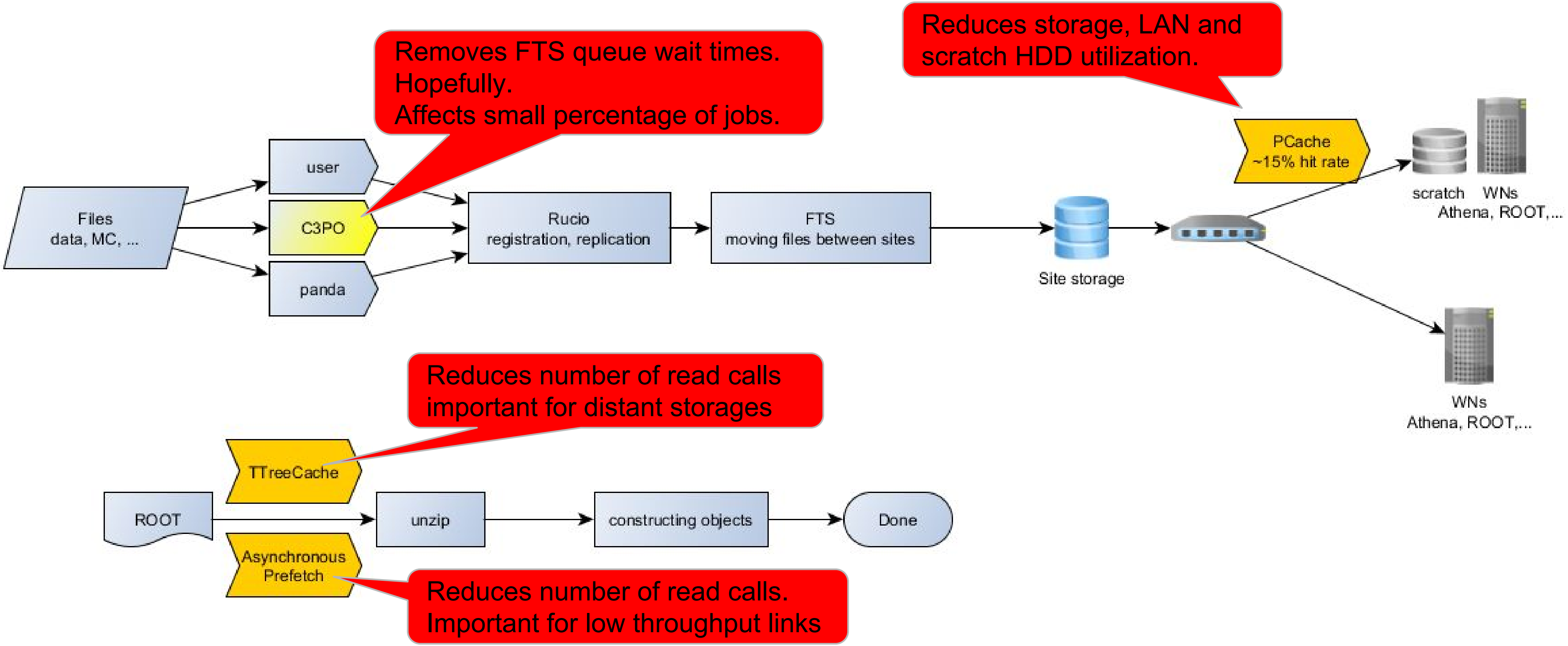
Storage servers 10-40Gbps

WNs - 1 or 10 Gbps

WNs scratch disks speeds writing - up to 40MB/s, reading - up to 80 MB/s

100% CPU efficiency demands read speeds of up to 20MB/s

# IO chain - current optimizations



Removes FTS queue wait times.
Hopefully.
Affects small percentage of jobs.

Reduces storage, LAN and scratch HDD utilization.

Reduces number of read calls important for distant storages

Reduces number of read calls.
Important for low throughput links

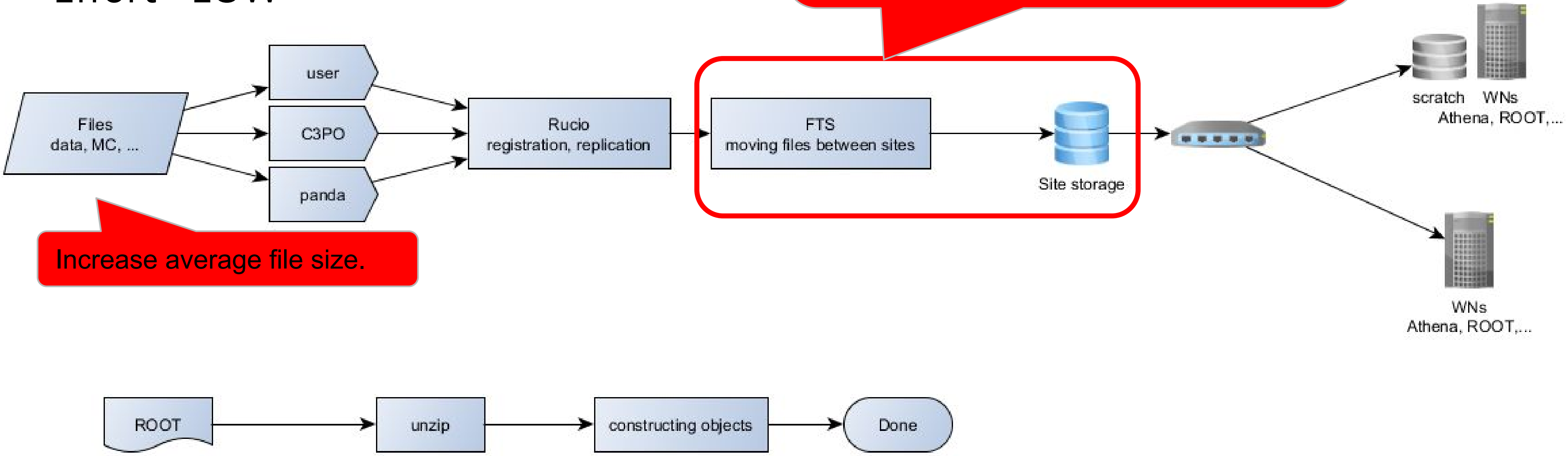
# Possible optimizations - I

Impact - HIGH

Effort - LOW

Reduce errors from ~15% to <1 permil
Make sure FTS queues are empty when links and src/dest storages are not saturated.

Plan already in place

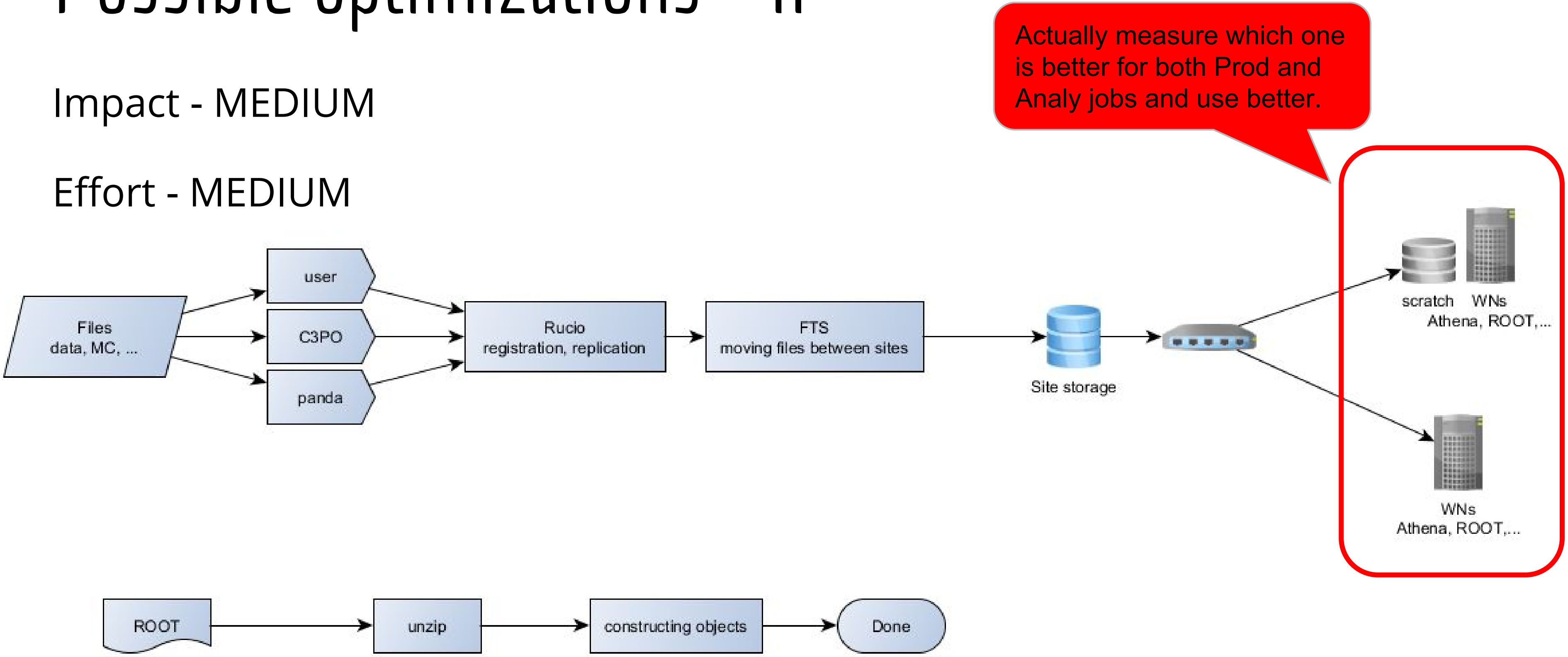


Increase average file size.

# Possible optimizations - II

Impact - MEDIUM

Effort - MEDIUM



Actually measure which one is better for both Prod and Analy jobs and use better.

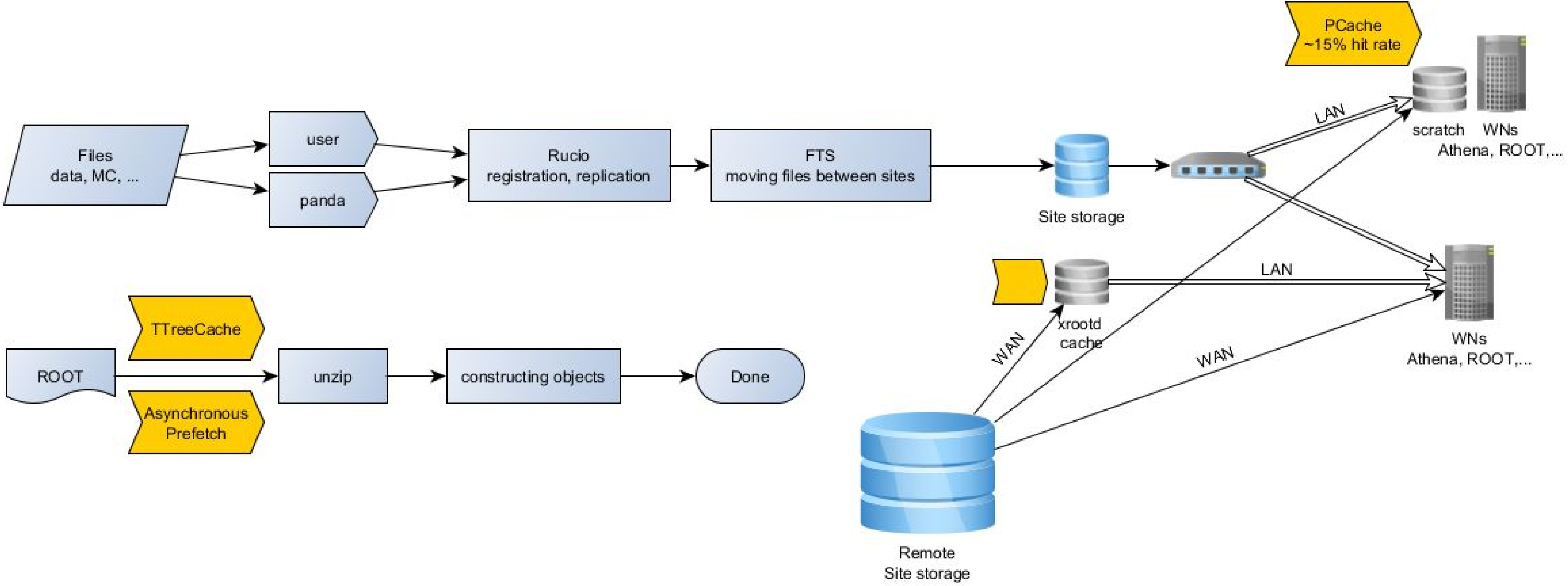
# Possible optimizations - III

**Requirements:**
TTC and AsyncP must be enabled.
Rucio must return correct paths.
Robust feedback on Network utilization

**Nice to have:**
Xrootd caches (rucio aware or not)

Impact - LARGE

Effort - LARGE



# Conclusions

IO influences TimeToComplete, CPU efficiency, Wall time left on the floor.

A lot of systems in play. All optimizations are stepwise, influencing each other.

A number of assumptions not really tested.

A lot of space for optimization of individual systems.  Still some low hanging fruit.

We should remove systems not having significant positive influence.

Continuously measure few KPIs (task TTC, CPU eff., network utilization).

# To Do list (next few weeks)

FTS optimizations:

- Hiro - update of FTS3 on BNL, tuning settings
- Ilija - continuous monitoring
- Mario - fixing endpoints mapping, checking non-recoverable errors
- Alejandro, Oliver K - tuning on their side

Regaining remote access possibility:

- Mario - changes in rucio client
- Ilija - validation
- Paul - pilot changes to use new rucio options
- Ilija - validation, testing on few sites, small scale

Measuring C3PO impact:

- From all datasets that fulfil replication requirements only 50% will be replicated (selection will be deterministic on hashed  dataset name)
- Will compare relevant metrics of jobs based for these two classes.

# Appendix I

Studies of ROOT Asynchronous Prefetching Performance
G. Papadrosou (Google Summer of Code student)

- Access an xAOD file over WAN
- 950 branches, 10K entries
- Cache size automatically set to the auto-flush setting used at writing
- Tested the same file at 3 sites: Chicago, New York and Paris
- Fluctuating network speed (from 200KB/sec to 200MB/sec)
- Tests run on a machine inside CERN Network with no other users logged in

| | **Standard** | **Asynchronous** | **Notes** |
|---|---|---|---|
| Read time (avg)<br>Worst<br>Best<br>**Chicago** | 2460 sec<br>6120 sec<br>153 sec | 227 sec<br>506 sec<br>41 sec | Slowest link, Large fluctuations In network speed |
| Read time (avg)<br>Worst<br>Best<br>**New York** | 688 sec<br>1116 sec<br>65 sec | 227 sec<br>810 sec<br>34 sec | Large fluctuations In network speed |
| Read time (avg)<br>Worst<br>Best<br>**Paris** | 77 sec<br>110 sec<br>65 sec | 37 sec<br>52 sec<br>32 sec | Fastest link, Somewhat consistent network speed |
| CPU time | 32.8 | 30.6 | |
| Read transactions | 1096 | 1097 | |

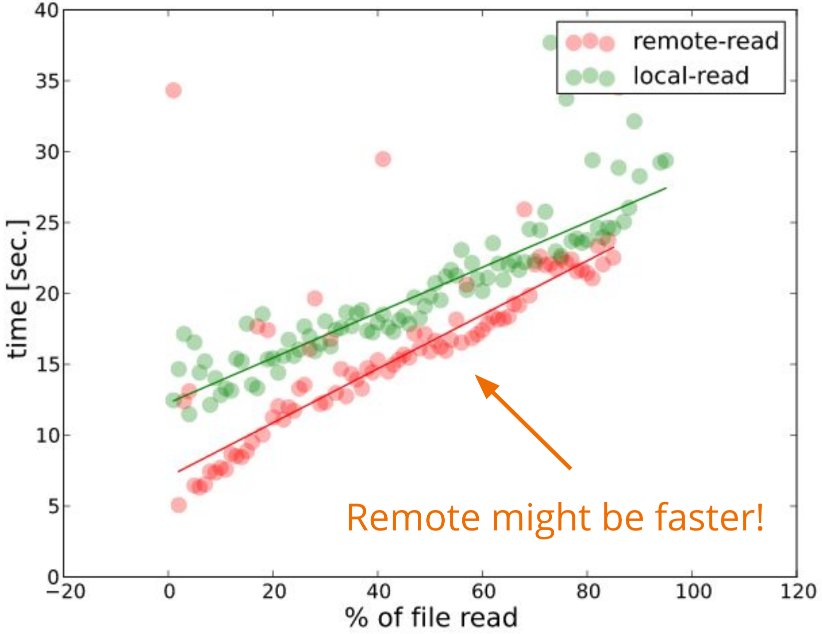
# Status and perspectives

- The code for activating ROOT Asynchronous Prefetching in Athena jobs is available in the master release branch (can be backported to other release branches)
- Our tests so far demonstrated that the Asynchronous Prefetching can bring significant speedup in remote data reading
- On the other hand
  - We have observed some instabilities (random failures), which require running more tests and understanding the problem
  - We have not yet measured memory overhead of the Asynchronous Prefetching
- More testing/studies is required before deciding whether or not it’s feasible to use this feature in production
- This requires running tests to check the stability and performance of the Asynchronous Prefetching. **Perhaps the DDM group can allocate some manpower for this task?**

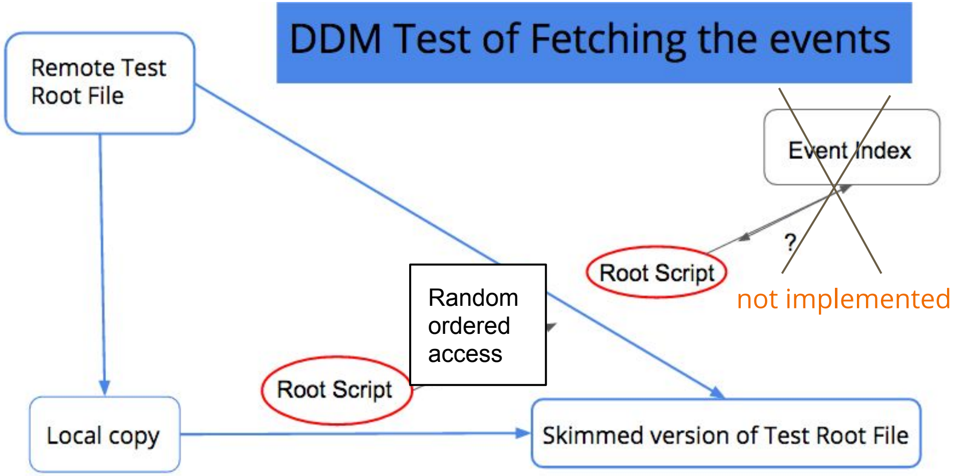
# Appendix II

Tomas Javurek
Continuing IO studies

# Remote vs. Copy To Scratch



Remote might be faster!



## DDM Test of Fetching the events

Remote Test Root File

Event Index

?

Root Script

not implemented

Random ordered access

Root Script

Local copy

Skimmed version of Test Root File

- Very naive test was performed, see schema
- Very complex problem! (technicalities not here)
- No quantitative results yet.
- Large potential to speed up job walltime.
- More efficient balancing of jobs.
- BUT, more challenging for network
- Our approach is purely empirical: Let’s try it in small scale!

# How to proceed:

- Implementing test to Hammer Cloud:
  - Remote vs. Copy to Scratch configurable in Pilot (being discussed with Paul for Pilot and Jarka for HC)
  - Using existing template in order to have comparison.
  - Testing against small set of site with careful monitoring.
- Aim is to build a grid map suggesting which method is better to use in which situation. Very complex problem:
  - Type of job
  - Distance to input
  - Network ...
- General studies + ML, Volodimir Begy
- Discussion:
  - Changes in software?
  - Pilot?
  - DDM?

# Appendix III

Ale D. G.
Q&A session

**what were the topics of Ilija's studies on WAN/LAN from few years ago?**

- Optimization of Persistent objects to improve IO, TP conv.
- Optimization of ATLAS file structure - basket reordering
- Optimizing how we use ROOT (TTC, splitting, compression factor, basket sizes,...)
- Measuring performance (in memory, scratch disk, NAS, xrootd, dcache, dpm)
- Measuring what branches are used.

**which of Ilija's results have proved to be correct?**

- How result can be incorrect? It We can like it or not but results were correct.

**which of Ilija's results were already improved?**

- A number of optimizations are used. But not all (eg. we don’t use knowledge about branches)

**which of Ilija's results still need improvement?**

- Some things should be redone from time to time. Performance tend to degrade with time.

**which aspects of Ilija's results has Tomas continued to do now?**

- Tomas will re-create a HC test template for easy measure of remote IO

**are they in line with the old studies or have things changed?**

- George’s and Tomas’ studies had no big surprises.

**who are the involved people right now and what exactly are they working on (Philippe C? Axel N? Peter VG & Co ?)**

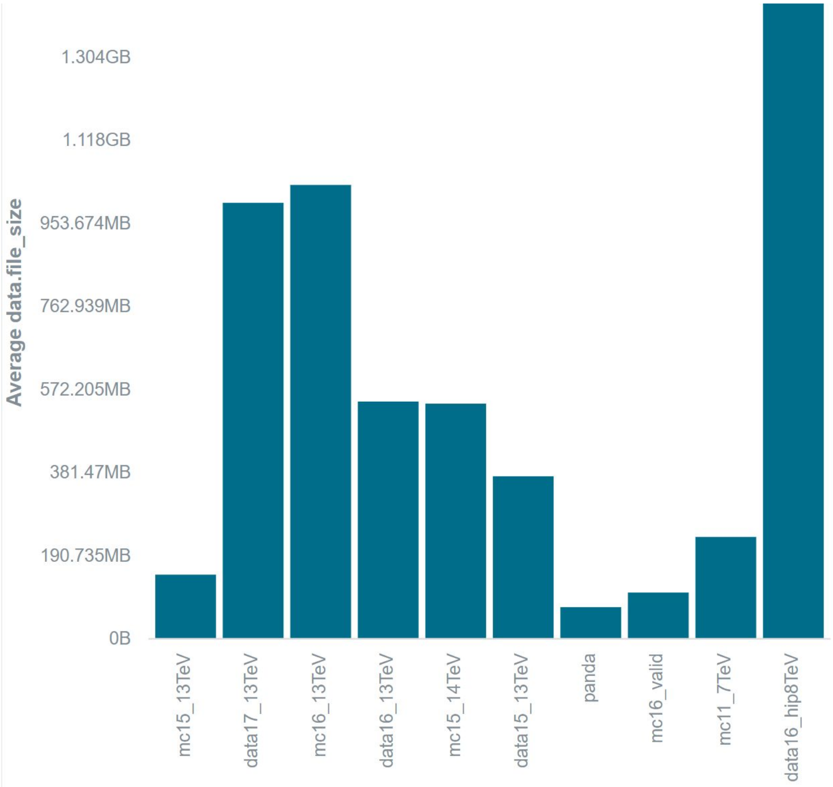
- Good question.

# Appendix IV

Plots backing up some of the claims

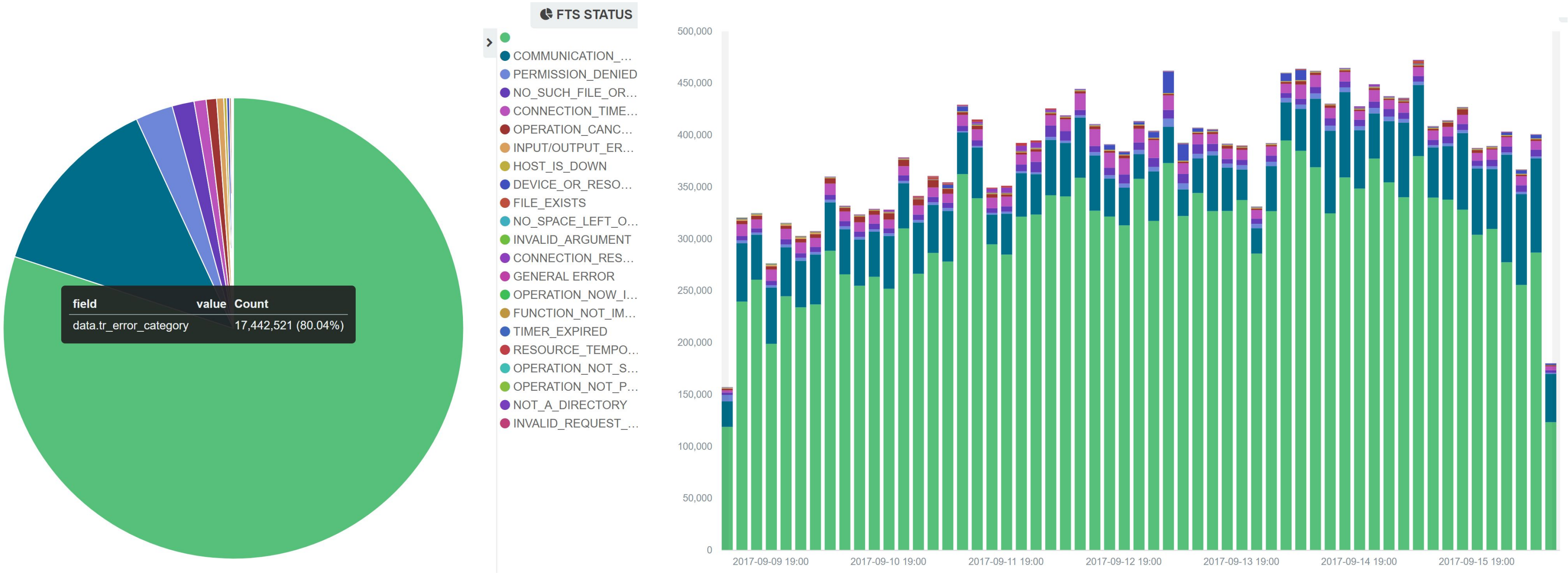
# Average file size

Files transferred by FTS during last week



# FTS error rate

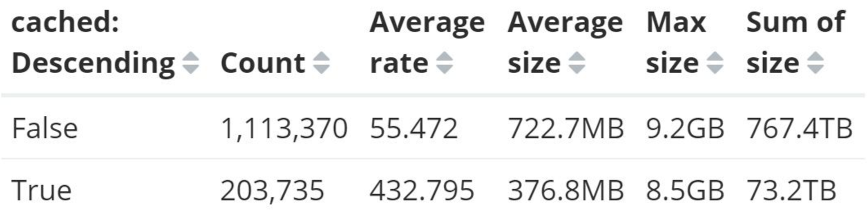
Continuously high error rate. Errors reduce limits on maximal active transfers.



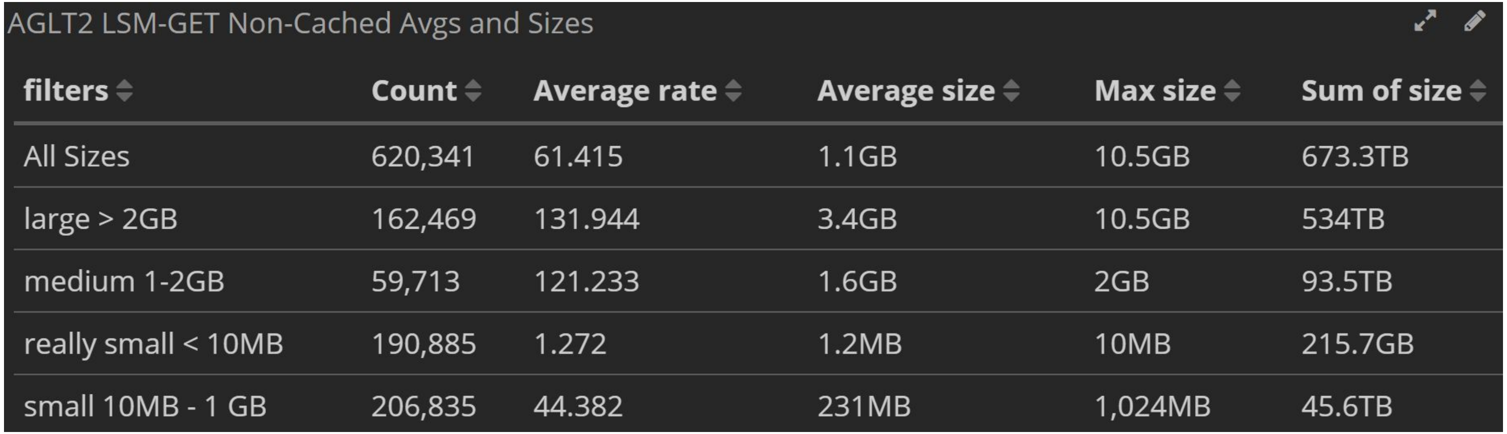
| field | value | Count |
|---|---|---|
| data.tr_error_category | | 17,442,521 (80.04%) |

# Copy to scratch rates

MWT2



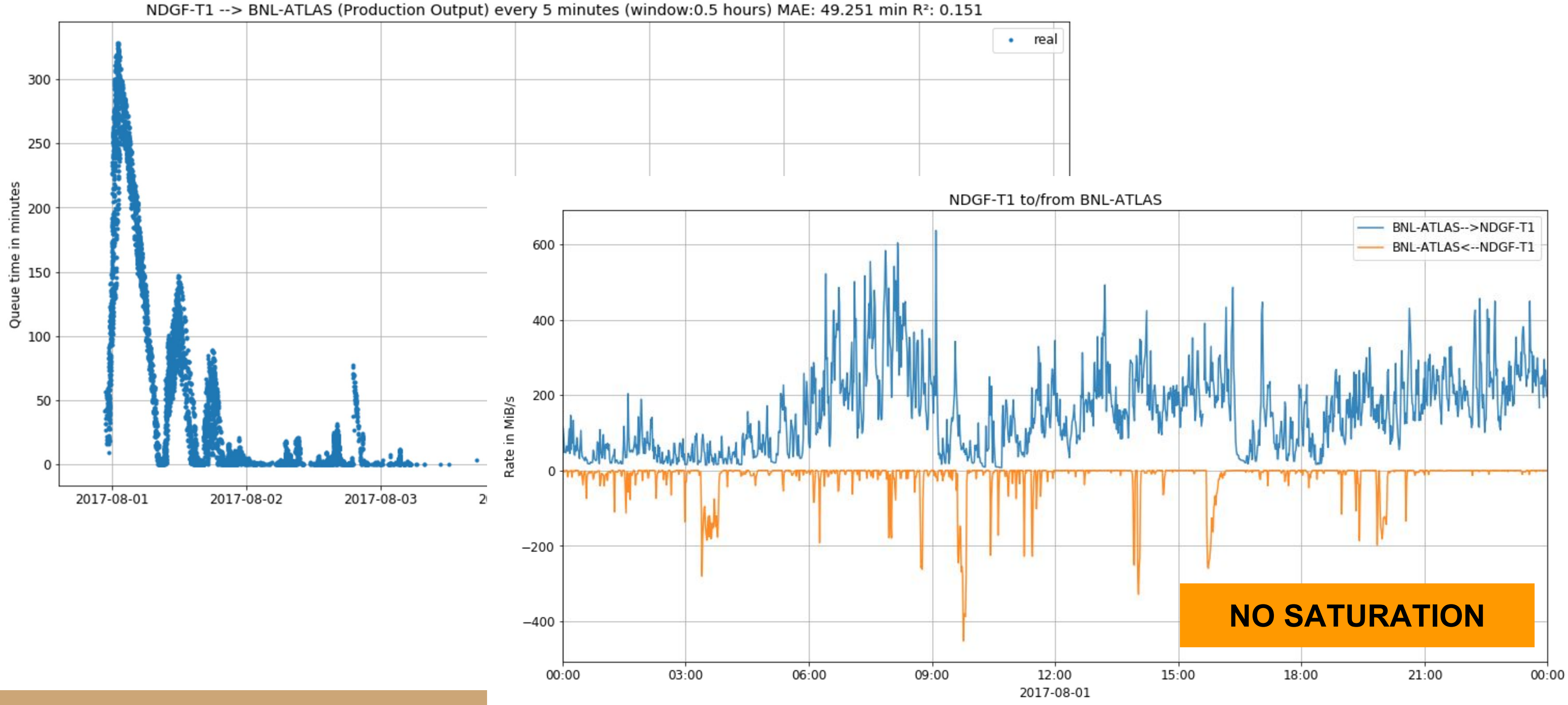
| cached: Descending | Count | Average rate | Average size | Max size | Sum of size |
|---|---|---|---|---|---|
| False | 1,113,370 | 55.472 | 722.7MB | 9.2GB | 767.4TB |
| True | 203,735 | 432.795 | 376.8MB | 8.5GB | 73.2TB |



AGLT2 LSM-GET Non-Cached Avgs and Sizes

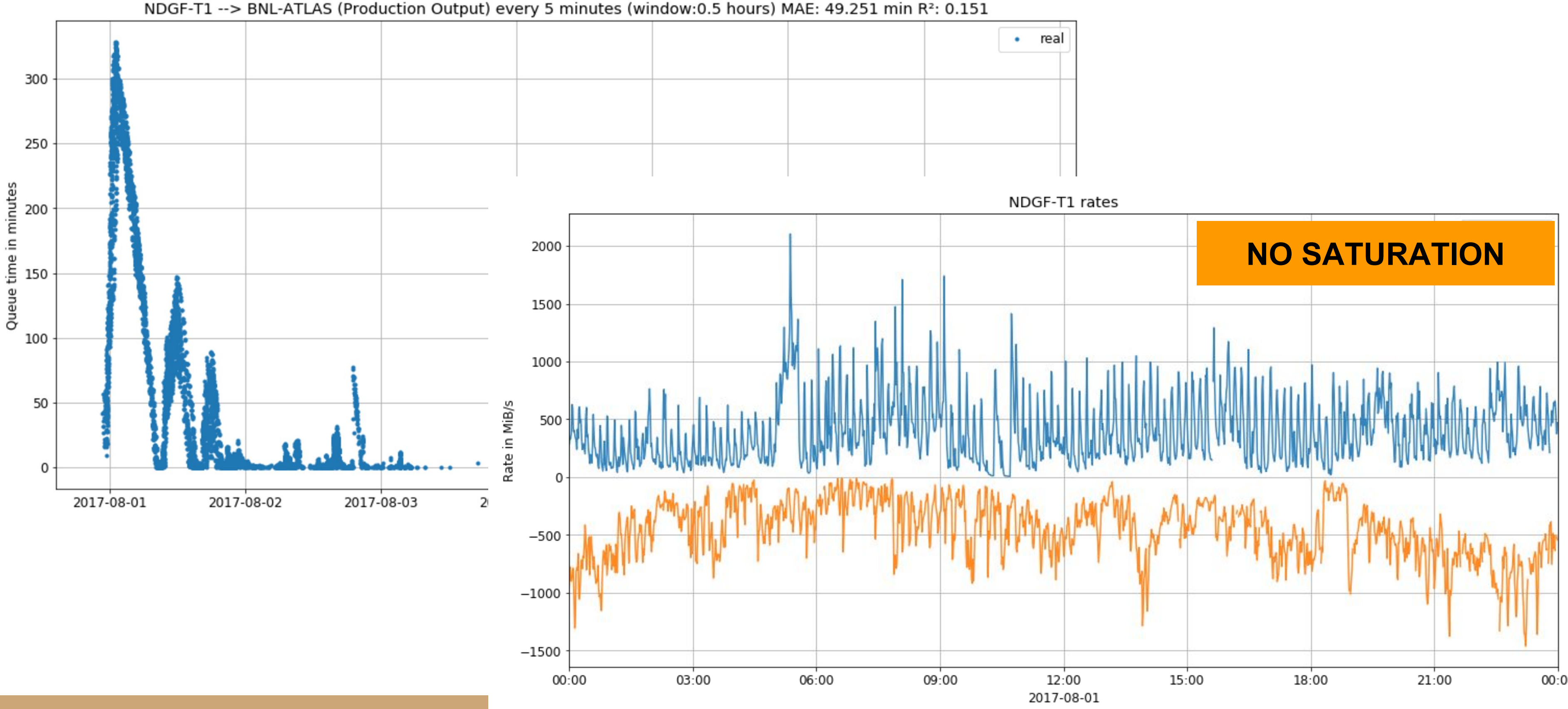
| filters | Count | Average rate | Average size | Max size | Sum of size |
|---|---|---|---|---|---|
| All Sizes | 620,341 | 61.415 | 1.1GB | 10.5GB | 673.3TB |
| large > 2GB | 162,469 | 131.944 | 3.4GB | 10.5GB | 534TB |
| medium 1-2GB | 59,713 | 121.233 | 1.6GB | 2GB | 93.5TB |
| really small < 10MB | 190,885 | 1.272 | 1.2MB | 10MB | 215.7GB |
| small 10MB - 1 GB | 206,835 | 44.382 | 231MB | 1,024MB | 45.6TB |

# Queue length VS link rate



NO SATURATION

# Queue length VS **source** read rate

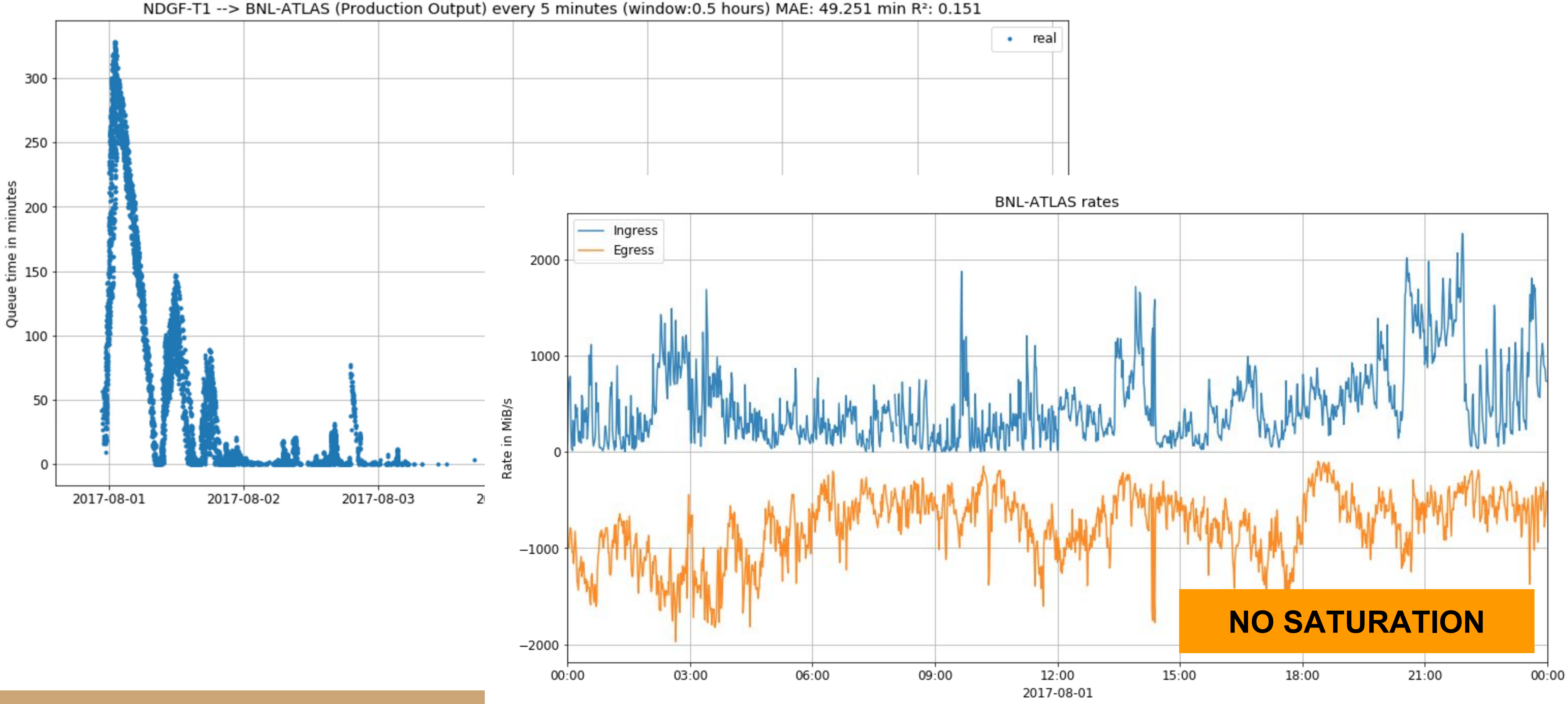


NDGF-T1 --> BNL-ATLAS (Production Output) every 5 minutes (window:0.5 hours) MAE: 49.251 min R²: 0.151

NDGF-T1 rates

NO SATURATION

# Queue length VS **destination** write rate



NO SATURATION