The ATLAS ROOT-based data formats: recent improvements and performance measurements

Wahid Bhimji
University of Edinburgh

J. Cranshaw, P. van Gemmeren, D. Malon,
R. D. Schaffer, and I. Vukotic
On behalf of the ATLAS collaboration

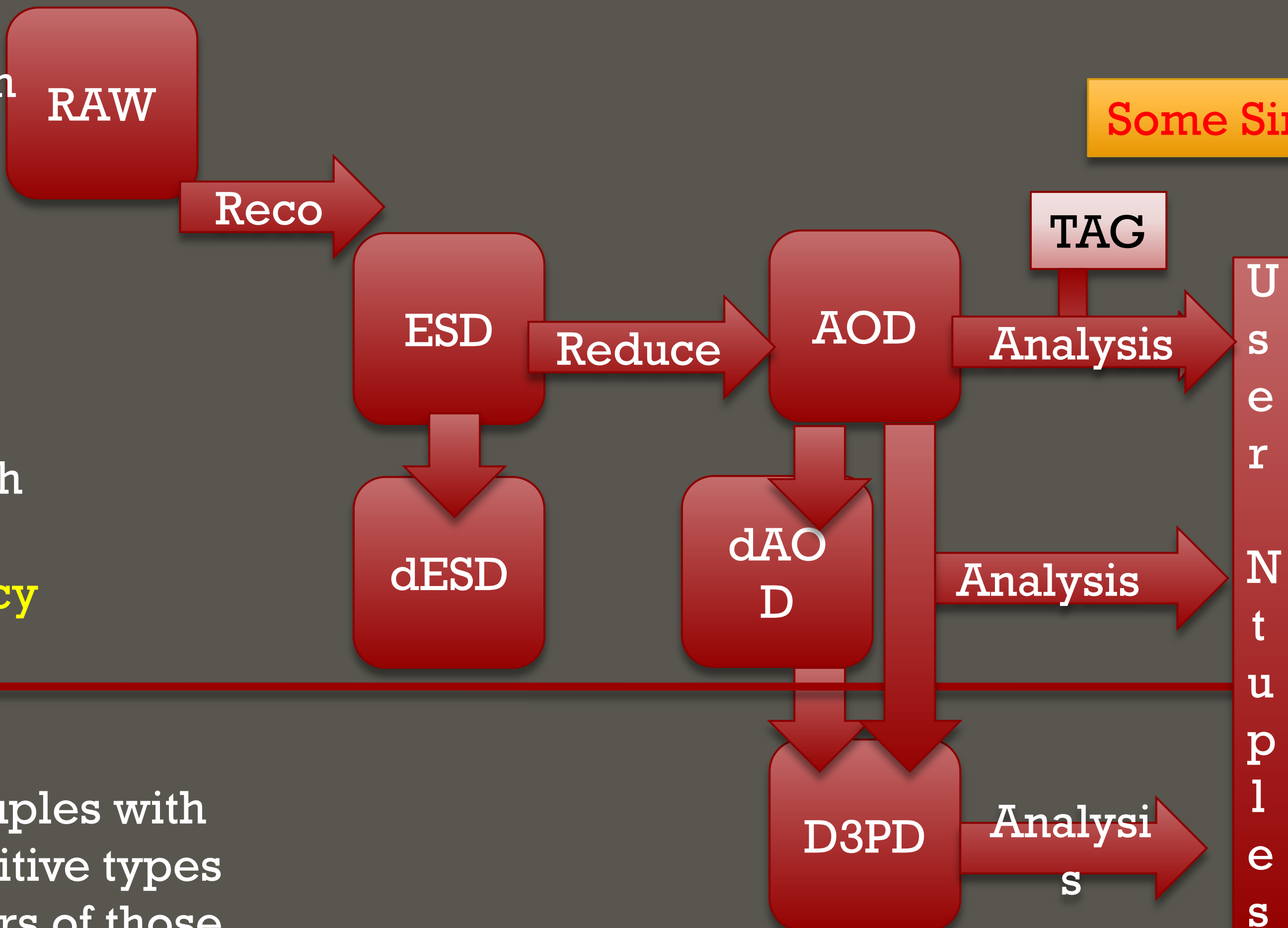
CHEP 2012

Overview

- ATLAS analysis workflows and ROOT usage
- Optimisations of ATLAS data formats
  - "AOD/ESD" and "D3PD"
- An ATLAS ROOT I/O testing framework

ATLAS ROOT I/O data flow

Bytestream
Not ROOT



Some Simplification!

AOD/ESD
ROOT with
POOL
Persistency

Athena
Software
Framework

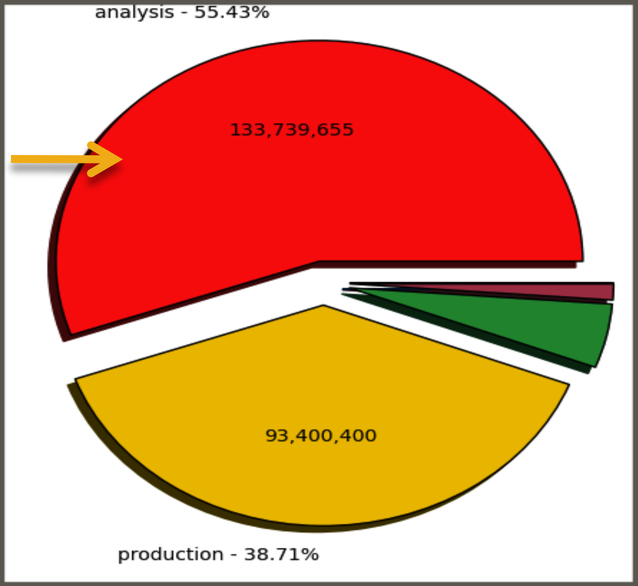
D3PD
ROOT ntuples with
only primitive types
and vectors of those

Non-Athena
User Code
(standard tools
and examples)

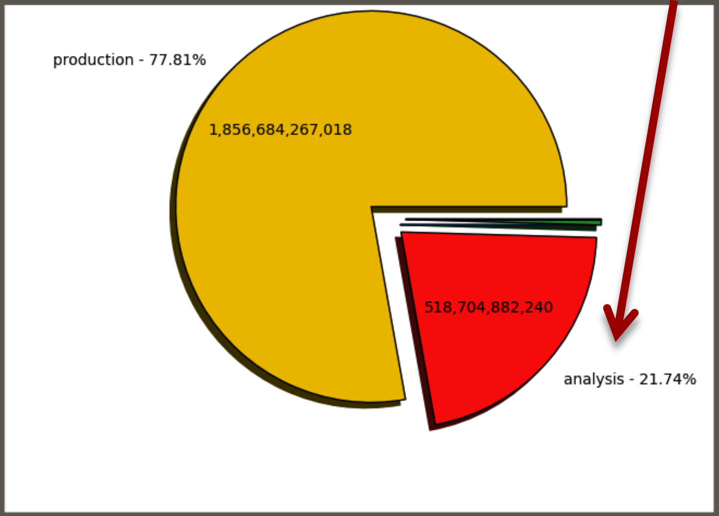
ATLAS analysis

- User analysis here called "analysis" as opposed to "production" is:
  - Not centrally organised
  - Heavy on I/O – all ATLAS analysis is ROOT I/O

By no. of jobs, Analysis =55%



By wallclock time, Analysis=22%



ATLAS ROOT I/O: POOL Formats

POOL:

- AOD/ESD use ROOT I/O via the POOL persistency framework.
- Could use other technologies for object streaming into files but in practice ROOT is the only supported.

ROOT versions in Athena s/w releases:

- 2011 data (Tier 0 processing) : ROOT 5.26
- 2011 data (Reprocessing): ROOT 5.28
- 2012 data : ROOT 5.30

ROOT I/O features we use

See e.g. P. Canal's talk at CHEP10

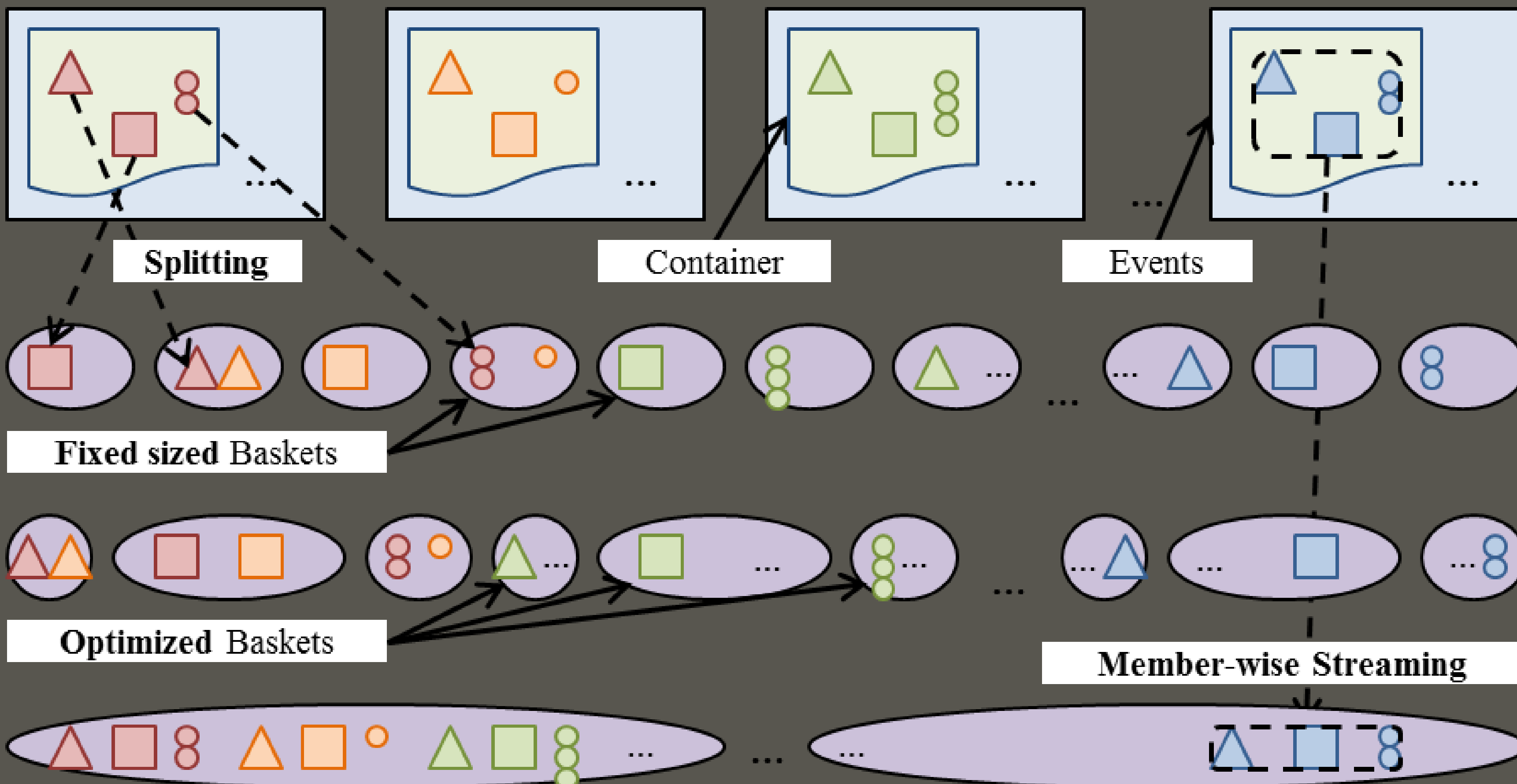
Writing files:

- Split level: object data members placed in separate branches
  - Initial 2011 running: AOD /ESD fully-split (99) into primitive data
- From 2011 use ROOT "autoflush" and "optimize baskets"
  - Baskets (buffers) resized so they have similar number of entries
  - Flushed to disk automatically once a certain amount of data is buffered to create a cluster that can be read back in a single read
  - Initial 2011 running we used the default 30 MB
- Also use member-wise streaming
  - Data members of object stored together on disk

Reading files:

- There is a memory buffer TTreeCache (TTC) which learns used branches and then pre-fetches them
- Used by default for AOD->D3PD in Athena
- For user code its up to them

ROOT streaming



ESD/AOD Optimisations

- ATLAS Athena processes event by event – no partial object retrieval
- Previous layout (fully-split, 30 MB AutoFlush): many branches and many events per basket
- Non–optimal particularly for event picking:
  - E.g. selecting with TAG:
    - Using event metadata: e.g. on trigger, event or object
    - No payload data is retrieved for unselected events
    - Can make overall analysis much faster (despite slower data rate)
  - Also multi-processor AthenaMP framework:
    - Multiple workers, each read a non-sequential part of input

2011 ESD/AOD Optimisations

- Switched **splitting off but kept member-wise streaming**
  - Each collection stored in a single basket
  - Except for largest container (to avoid file size increase)
- Number of baskets reduced from ~10,000 to ~800,
  - Increases average size by more than x10
  - Lowers the number of reads
- Write **fewer events per basket in optimisation:**
  - ESD flush every **5 events**
  - AOD every **10 events**
- Less data needed if selecting events when reading

Performance Results

Repeated local disk read; Controlled environment; Cache cleaned

AOD Layout	Reading all events	Selective 1% read
OLD: Fully split, 30 MB Auto-flush	55 (±3) ms/ev.	270 ms /ev.
CURRENT: No split, 10 event Auto-flush	35 (±2) ms /ev.	60 ms/ev.

- Reading all events is ~30% faster
- Selective reading (1%) using TAGs: 4-5 times faster

Further Performance Results

- **File Size** is **very similar** in old and current format.
- **Virtual Memory** foot print **reduced by about 50-100 MB** for writing and reading:
  - Fewer baskets loaded into memory.
- **Write speed increased** by about **20%**.
  - The write speed was increased even further (to almost 50%), as the compression level was relaxed.

New scheme used for autumn 2011 reprocessing

Athena AOD read speed (including Transient/Persistent conversion and reconstruction of some objects)
~ 5 MB/s from ~3 MB/s in original processing
(including ROOT 5.26 to 5.28 as well as layout change)

Testing Framework

- ROOT I/O changes affect different storage systems on the Grid differently
  - E.g. TTC with Rfio/DPM needed some fixes
- Also seen cases where AutoFlush and TTC don't reduce HDD reads/time as expected

Need regular tests on all systems used (in addition to controlled local tests) to avoid I/O "traps"

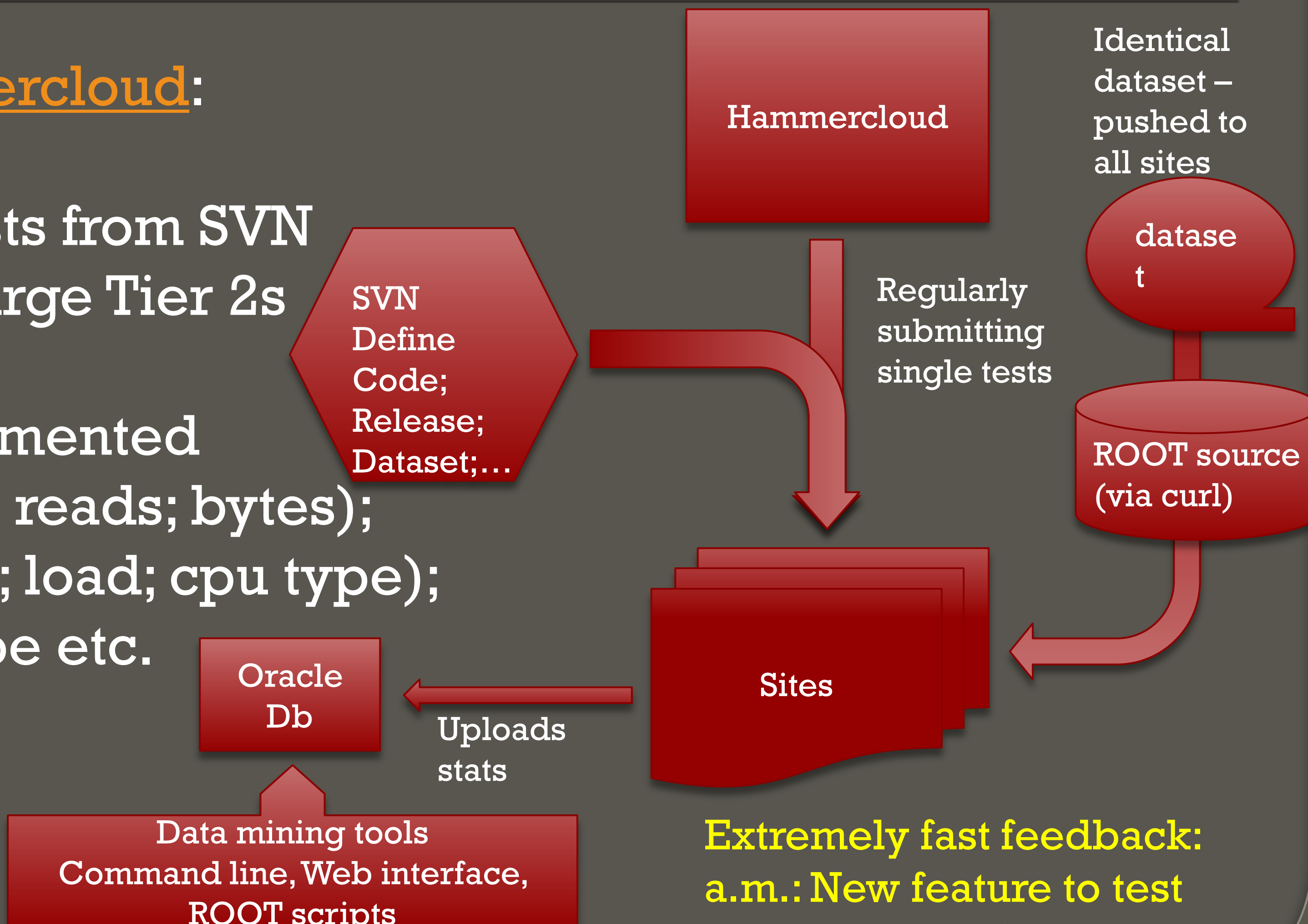
- Also now have a ROOT IO group well attended by ROOT developers ; ATLAS; CMS and LHCb
  - Coming up with a rewritten basket optimization
  - We promised to test any developments rapidly

Built a Testing Framework

Using hammercloud:

Takes our tests from SVN
Runs on all large Tier 2s

Highly instrumented
- ROOT (e.g. reads; bytes);
- WN (traffic; load; cpu type);
- storage type etc.



Extremely fast feedback:
a.m.: New feature to test
p.m.: Answers for all storage systems in the world.

Examples of Tests

1. ROOT based reading of D3PD:
   - Provides metrics from ROOT (no. of reads/ read speed)
   - Like a user D3PD analysis
   - Reading all branches and 100% or 10% events (at random);
   - Reading limited 2% branches (those used in a real Higgs analysis)
2. Using different ROOT versions
   - Latest Athena Releases.
   - Using 5.32 (not yet in Athena)
   - Using trunk of ROOT
3. Athena D3PD making from AOD
4. Instrumented user code examples
5. Wide-Area-Network Tests

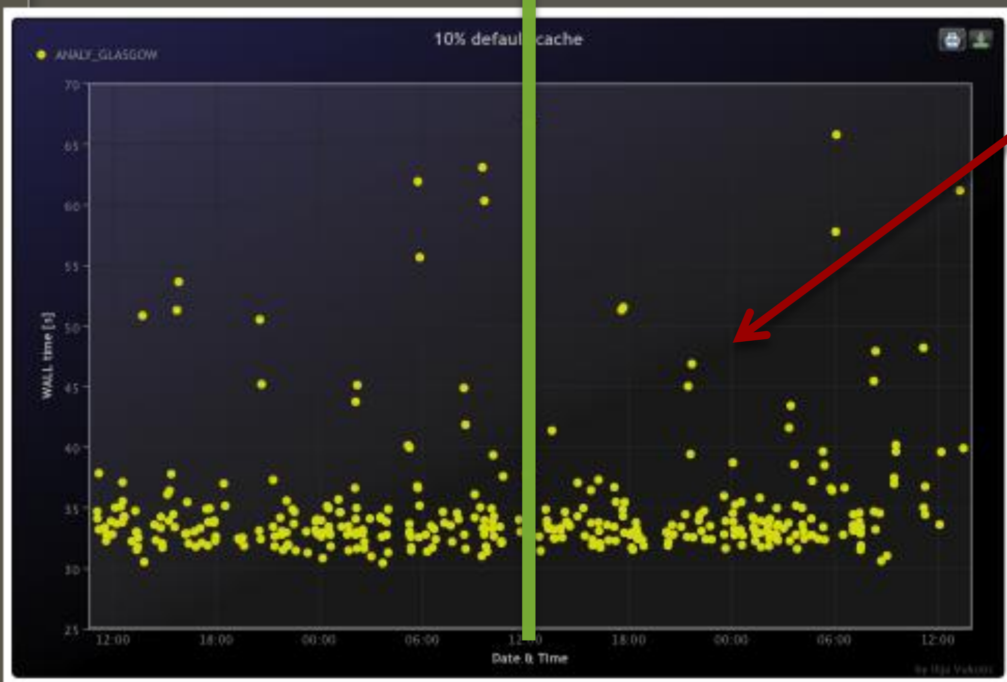
http://ivukotic.web.cern.ch/ivukotic/HC/index.asp


Login
From
2012-04-09
to
2012-04-17
100% default cache
x: root bytesread [bytes]
y: CPU time [s]
vs Time
line
avg.
proj. x
proj. y
Refresh
Annotations
Sites
CPUs
ROOT versions
tags/v5-32-01 v:43181
tags/v5-30-05 v:42230
d v:0
Storage
Input files
100% default cache
ANALY_CERN_XROOTD
ANALY_CSCS
ANALY_DESY-ZN
ANALY_GLASGOW
ANALY_QMUL
CPU time [s]
180
160
140
120
100
80
60
40
20
500M
525M
550M
575M
600M
625M
650M
675M
700M
725M
750M
775M
800M
root bytesread [bytes]
by Ilija Vukotic
CPU time [s] for 100% default cache
ANALY_CERN_XROOTD 62.26
ANALY_CSCS 72.16
ANALY_DESY-ZN 56.24
ANALY_GLASGOW 68.63
ANALY_QMUL 51.81
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80

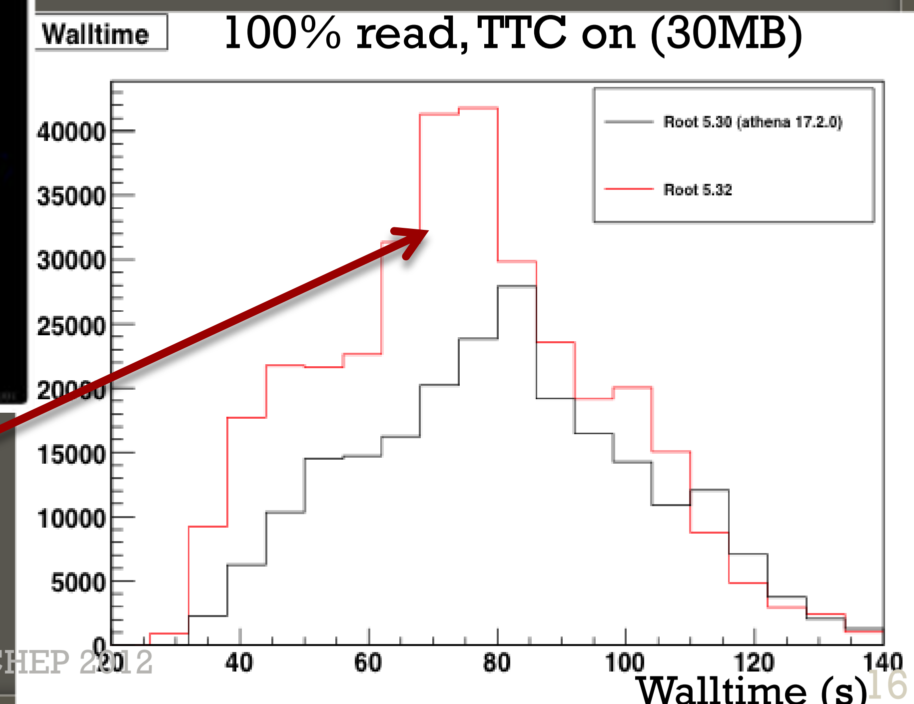
Testing ROOT Versions

ROOT 5.28 (Athena 17.0.4)

ROOT 5.30 (Athena 17.1.4)



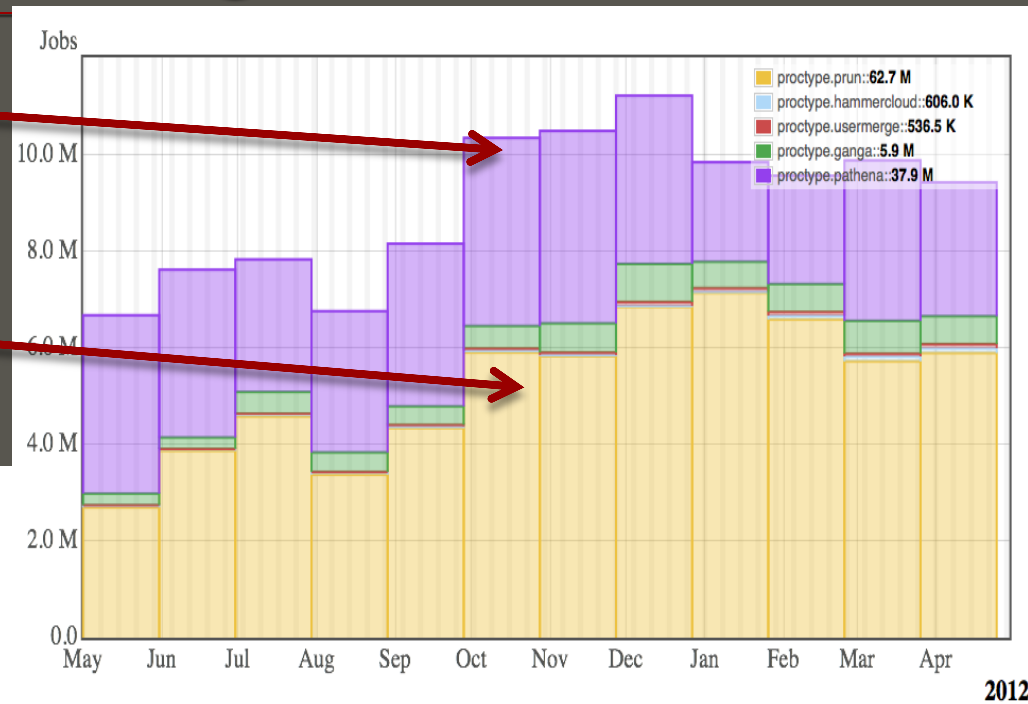
Tracking change to ROOT 5.30 in Athena – no significant changes in wall times for D3PD read on any storage system

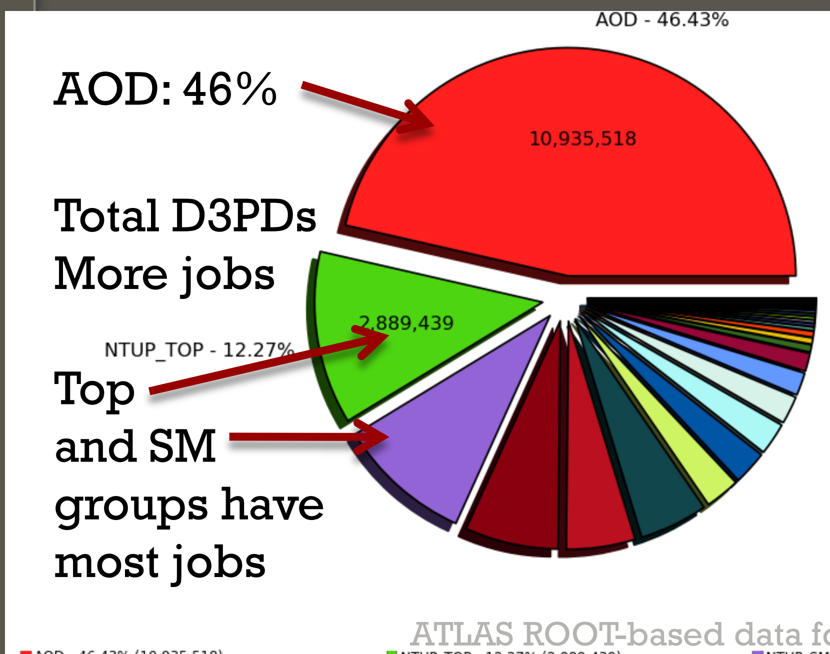


ROOT 5.32 (red) again no big change on average over all sites

Tuning D3PDs: growth of D3PD

- "pathena" jobs
  - running in framework
  - Eg. Private Simulation ; AOD Analysis
- "prun" jobs
  - Whatever user wants!
  - Mostly D3PD analysis

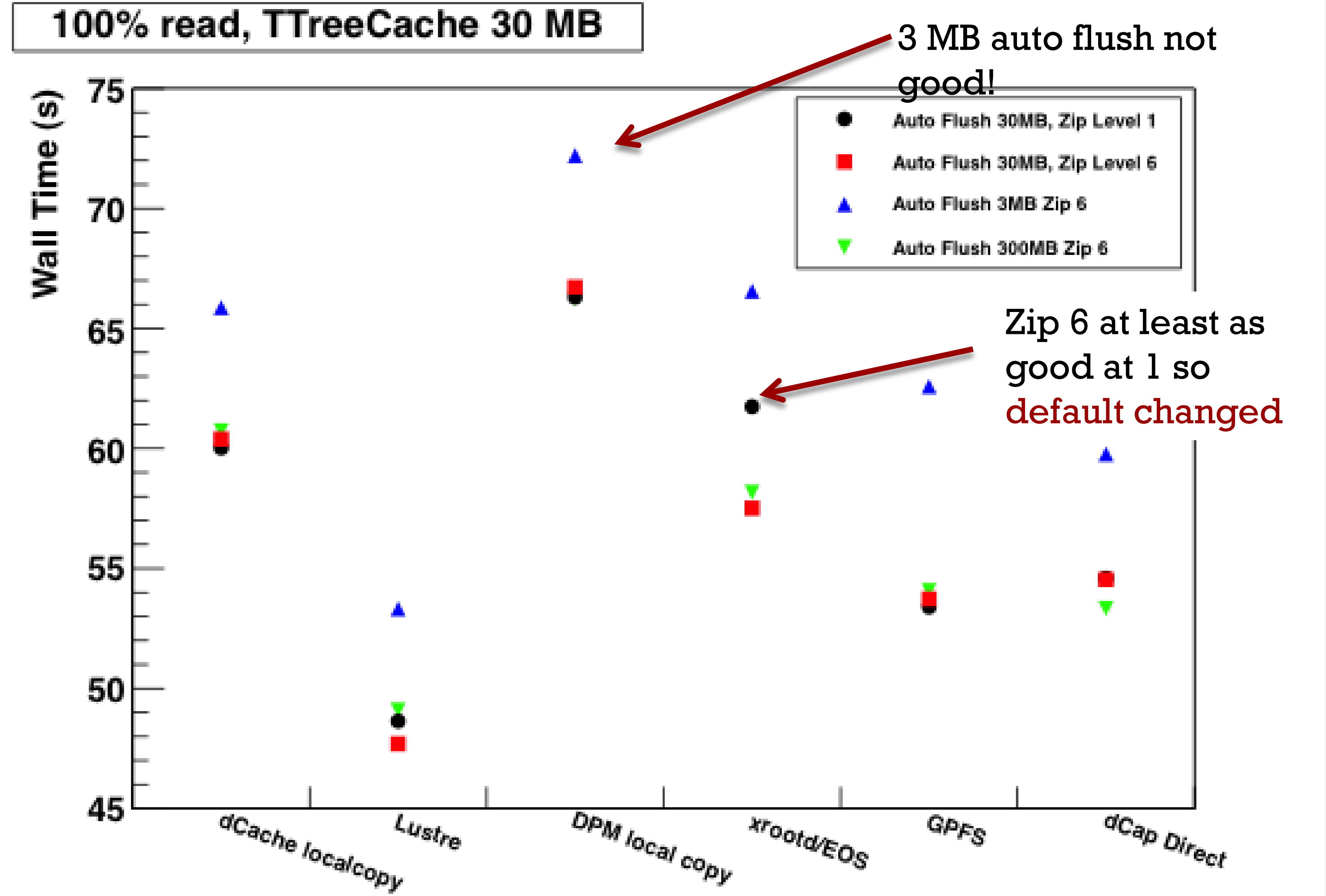




Optimisation of D3PD analysis becoming increasingly important

Tuning D3PDs: Compression and Auto-Flush

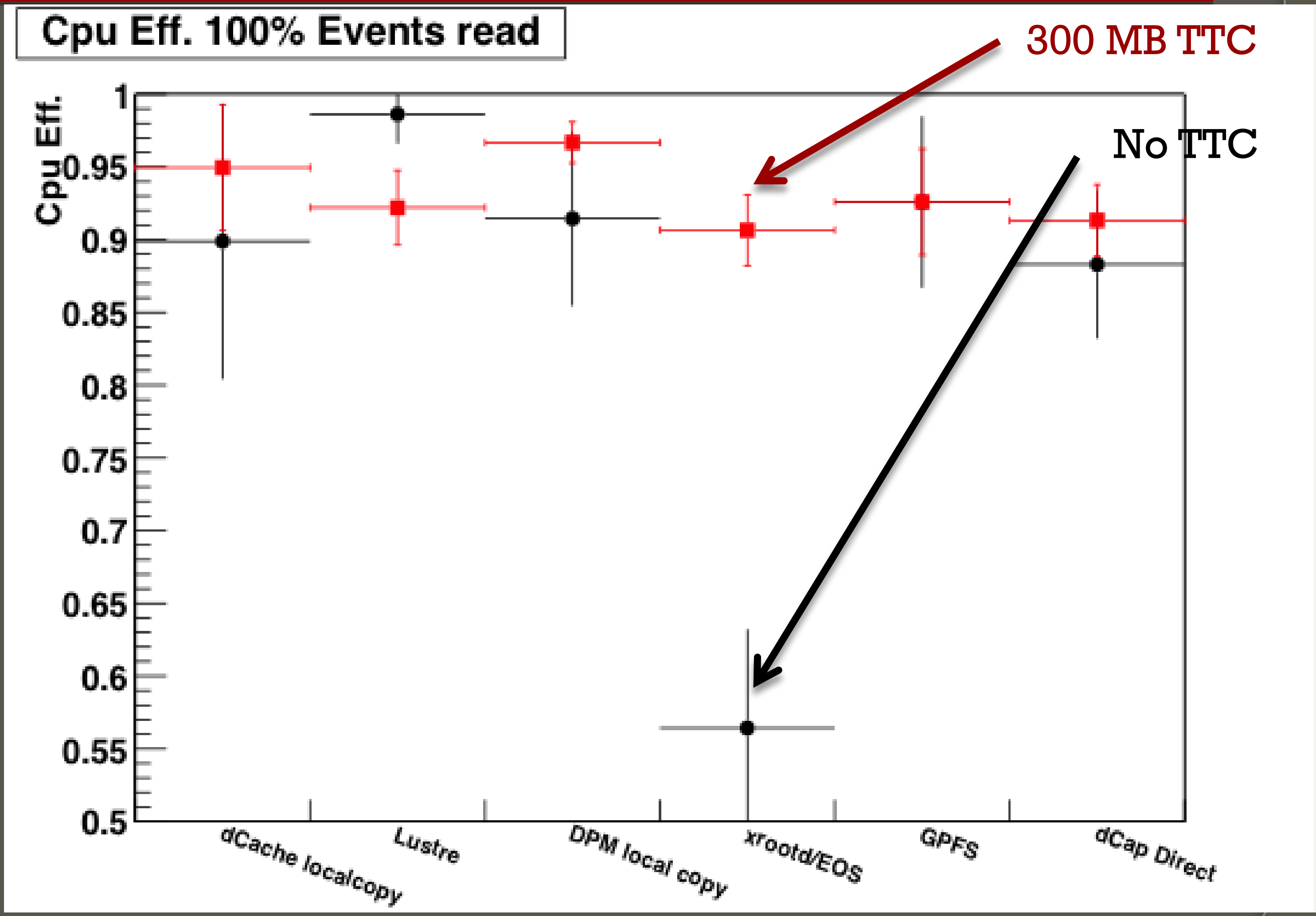
- Rewrite D3PD files:
  - Using ROOT 5.30 – current Athena release
- Try different zip levels, current default 1:
  - Local testing suggested "6" more optimal (in read speed) so copied this to all sites
  - Zip 6 files are ~5% smaller so also important gains in disk space and copy times
- Change autoflush setting, currently 30MB:
  - Try extreme values of 3 and 300 MB
- Showing results here for a few example sites
  - Labeled by the storage system they run
  - But use different hardware so this is not a measure of storage system or site performance



100% read, TTreeCache 30 MB
Wall Time (s)
75
70
65
60
55
50
45
Auto Flush 30MB, Zip Level 1
Auto Flush 30MB, Zip Level 6
Auto Flush 3MB Zip 6
Auto Flush 300MB Zip 6
dCache localcopy
Lustre
DPM local copy
xrootd/EOS
GPFS
dCap Direct
3 MB auto flush not good!
Zip 6 at least as good at 1 so default changed

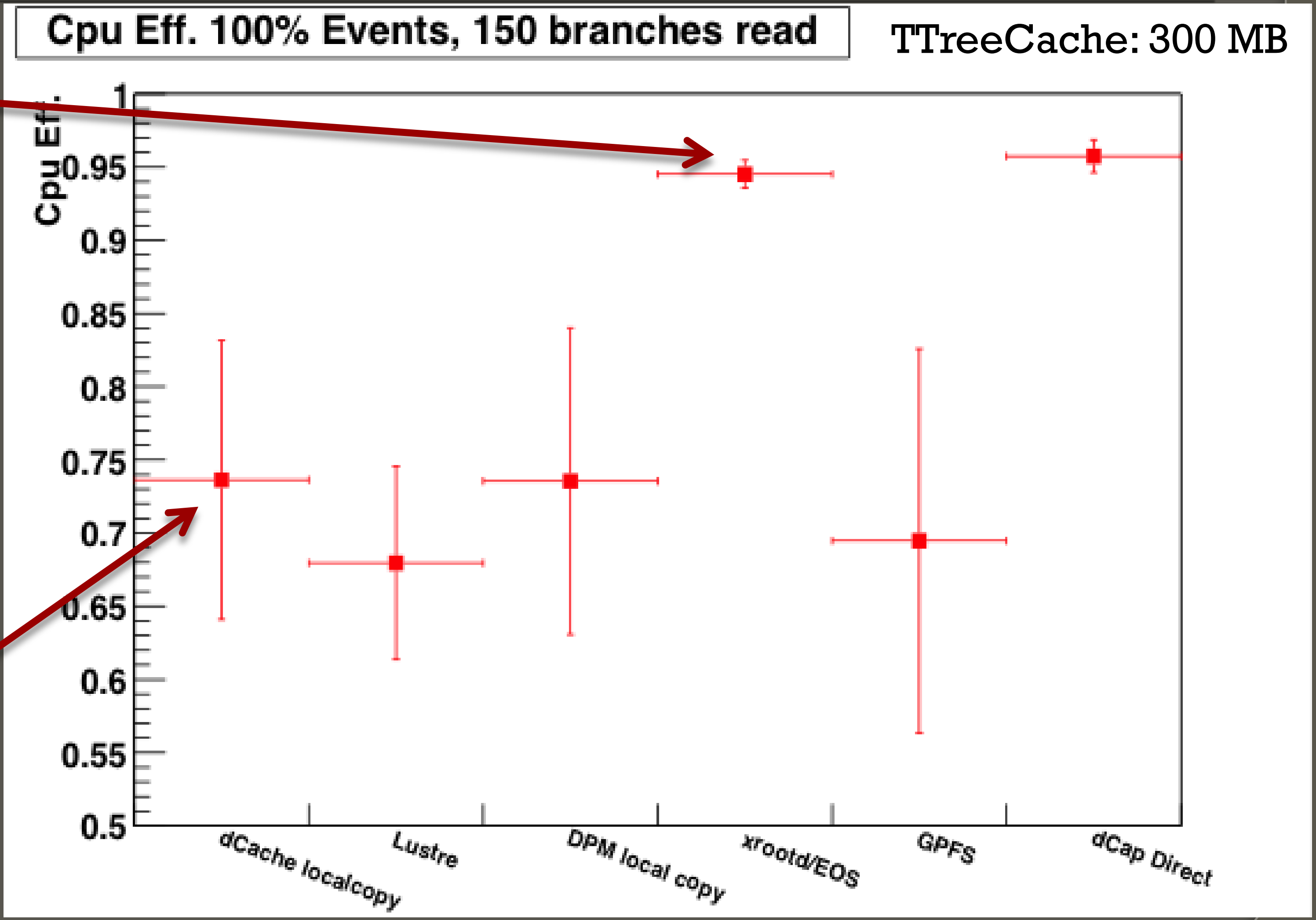
TTreeCache

- TTreeCache essential at some sites
- Users still don't set it
- Different optimal values per site
- Ability to set in job environment would be useful



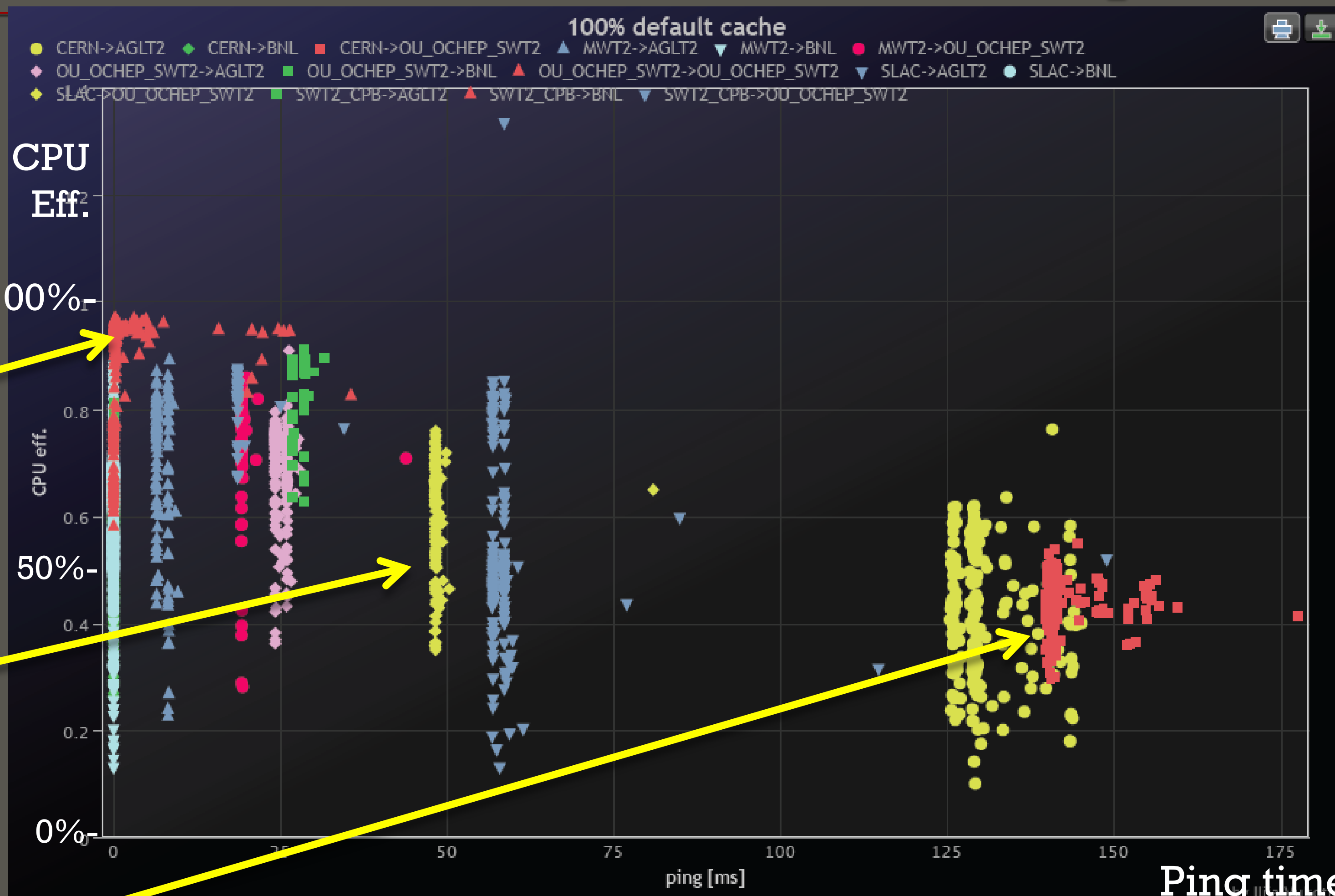
Reading limited branches

- Systems using vector reading protocol (dCap; xrootd) still have high eff.
- Drop in CPU efficiency on other storage systems



Wide-Area-Network Analysis

Running on various US sites

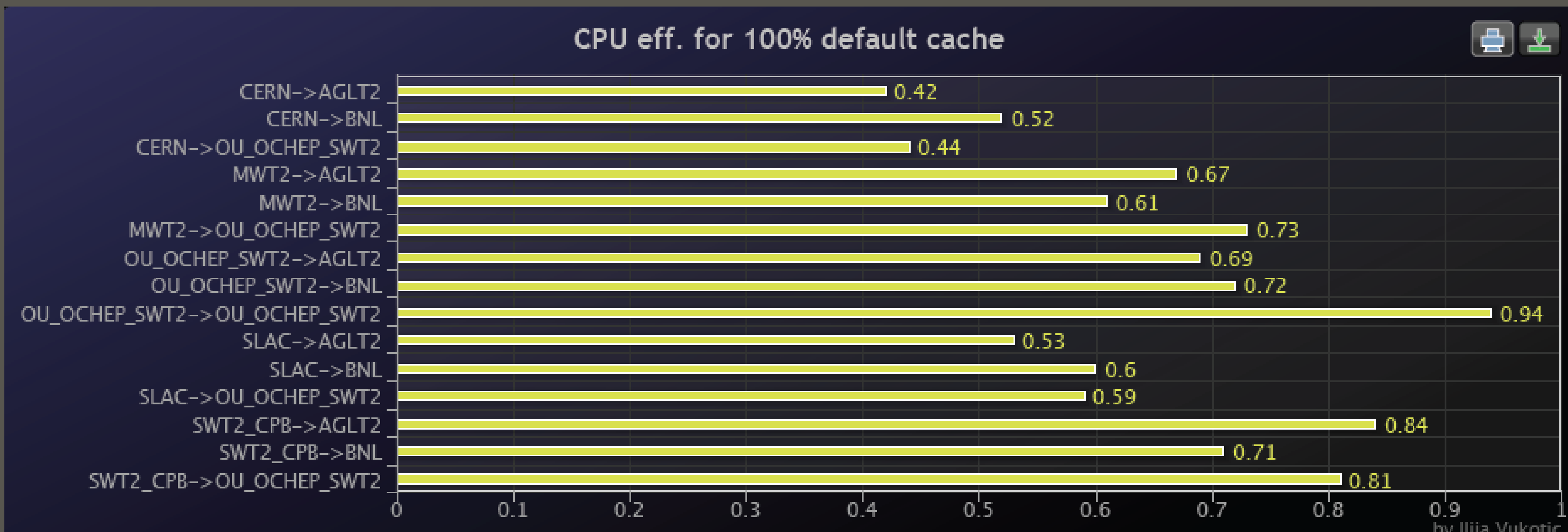


Local read

Reading from other US sites

Reading from CERN

CPU Efficiencies over WAN



- First measurements …. not bad rates
  - 94% local read eff. drops to 60%-80% for other US sites
  - and around 45% for reading from CERN
- Offers promise for this kind of running if needed
  - Plan to use such measurements for scheduling decisions

Conclusions

Made performance improvements in ATLAS ROOT I/O

Built I/O Testing framework: for monitoring and tuning

Plan to do:

- Lots more mining of our performance data ☺
- Test and develop core ROOT I/O with working group:
  - Basket Optimisation
  - Asynchronous Prefetching
- Provide sensible defaults for user analysis
- Further develop WAN reading
- Site tuning
- New I/O strategies for multicore (see next talk!)