Analysing I/O bottlenecks in LHC data analysis on grid storage resources

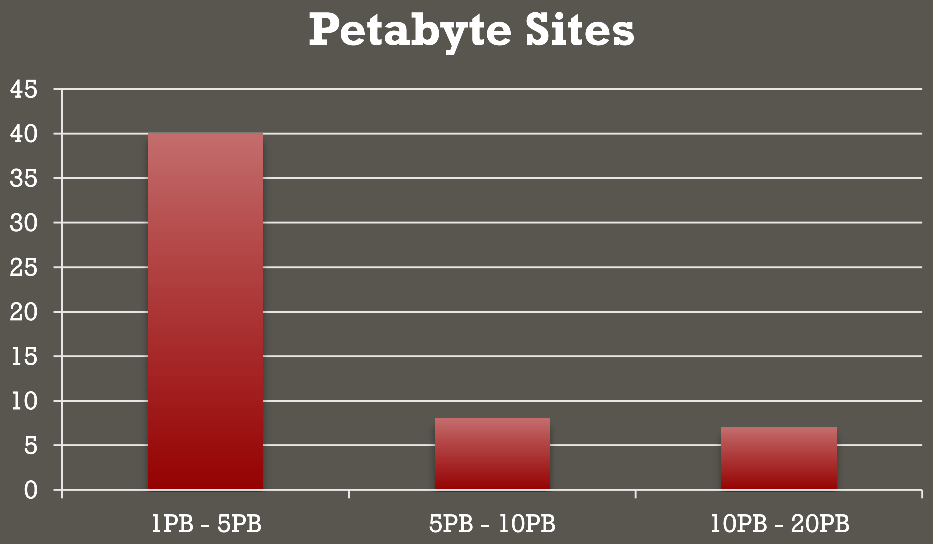
Wahid Bhimji
University of Edinburgh

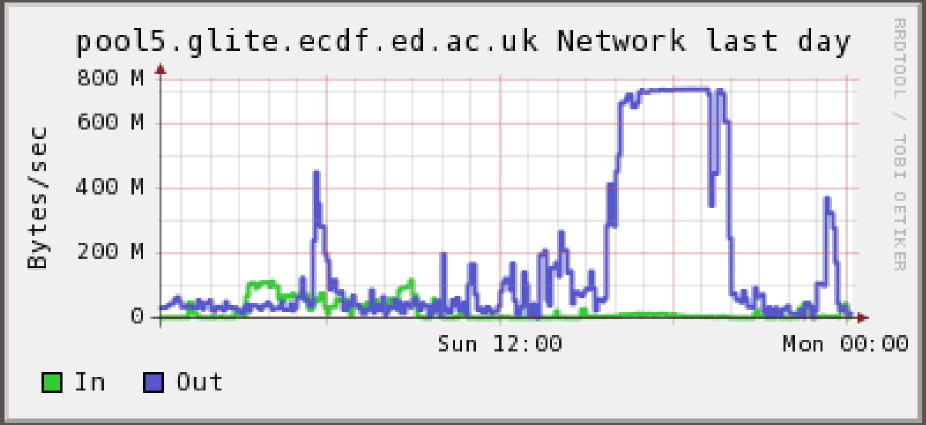
P. Clark, M. Doidge, M. P. Hellmich,
S. Skipsey and I. Vukotic

Introduction

- LHC experiments now have multi-petabytes of storage at multiple sites

- A lot of activity at sites is I/O heavy





Introduction

WLCG Storage Technical Evolution Group recently recommended:

- I/O benchmarks be developed,
- Experiments communicate I/O requirements,
- Applications and storage systems provide tools for understanding I/O.

This talk :

- Perspectives on analyzing I/O usage
- Using examples of work undertaken
  - With some results too!
- Comparing and contrasting the approaches

Why analyze I/O? Different perspectives

Sites:

- Vendor supplied storage / purchasing decisions
- Site tuning (hardware/ middleware)

Storage Middleware Developers

- Tuning system for use in WLCG environment
  - Basic functionality testing for new releases
  - Scale testing of low-level operations
- Choice of protocols / technologies etc.

Experiments

- Applications
- Data models / file structure
- Chasing sites to ensure resources utilized

Examples presented here

- Vendor storage testing: evaluating suitability of suggested storage for a Tier 2 site.
- Low-level middleware testing: to improve scalability for use in bigger sites.
- ROOT I/O testing framework: for evaluating changes in ROOT-based applications and data structures.
- Middleware testing framework: for releases and new features.

1. Vendor Storage Testing

AIM: Test disk server from Dell

- 2 Dell R710, each with 4 x 24 TB MD3200/1200 storage arrays
- Dense storage of the kind in use at many sites

- Sent to many sites with different storage systems
  - We used as a DPM disk pool (most popular SE at UK Tier 2s)
  - Servers partitioned into virtual disks 9-41TB :
    - Range of RAID configurations and underlying filesystems
  - Tested in Lancaster's smaller production cluster (512 cores)
- Tests (wrapped in scripts to submit to batch queue)
  - Rfcp: copy using rfio: 250 clients per disk server.
  - ROOT RFIO read: 2G file, 100 clients per filesystem
- Dell Whitepaper – including source code for tests:
  - http://www.dellhpcsolutions.com/

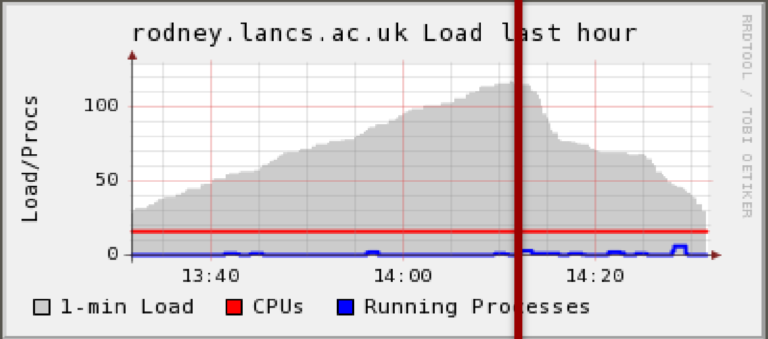
Vendor Storage: Some Results

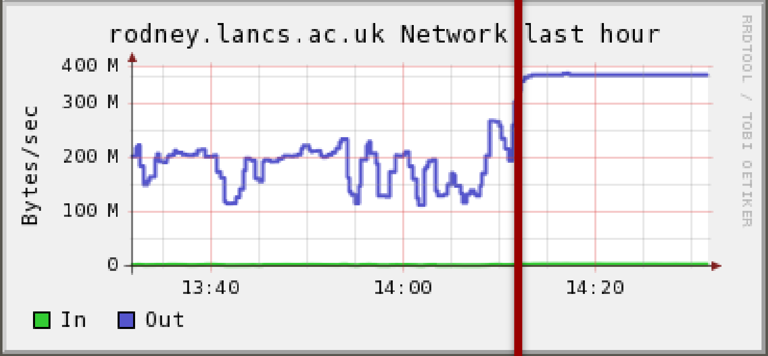
Artificially created load seen on T2 production systems and similar tuning effects e.g. readaheads:

so effective test for new hardware

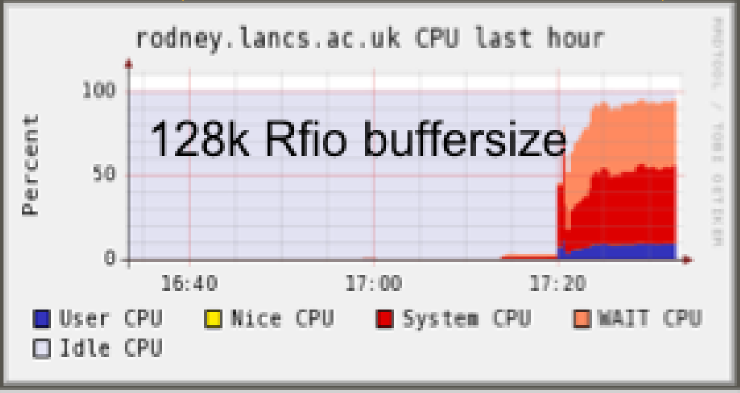
Block device (rfcp tests)

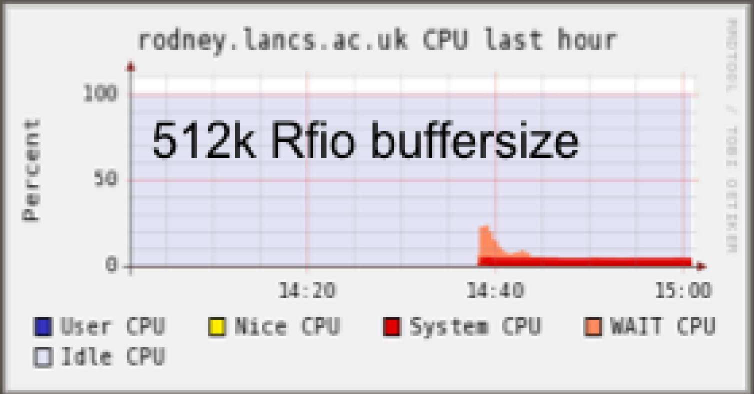
blockdev
--setra 8MB





Rfio (ROOT direct read test)


128k Rfio buffersize


512k Rfio buffersize

2. Middleware Scale Testing

AIM: Find limits of the DPM storage element's internal ways of dealing with requests when stressed in a realistic fashion

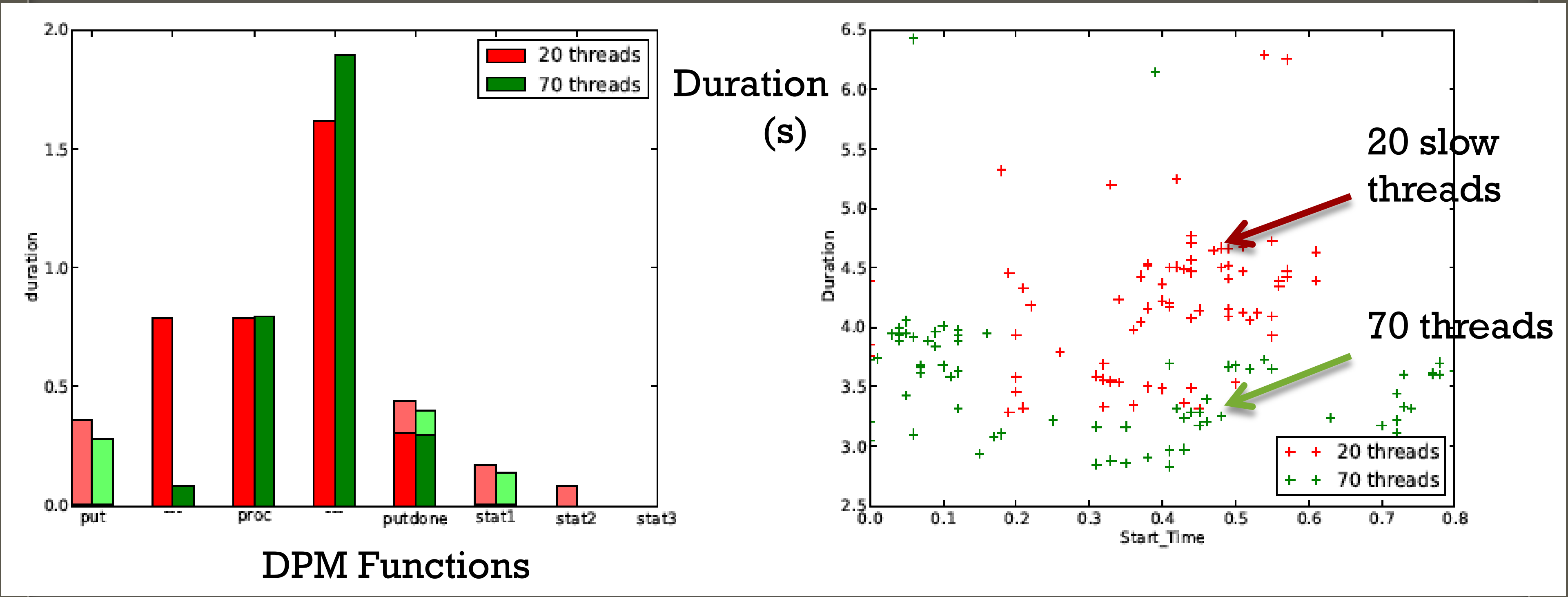
- Added tests to DPM package perfsuite
  - File copy and ROOT direct RFIO access (as before)
  - But also a "pretend" rfcp test
    - All DPM calls performed but no actual transfer
    - Explore DPM limits without hitting hardware bottlenecks
- Also added detailed log-file parsing
- Full details see:

http://www.ph.ed.ac.uk/~wbhimji/GridStorage/StressTestingAndDevelopingDistributedDataStorage-MH.pdf

M/ware Scaling: Some Results

Found improvements to DPM daemon:

- Increase socket queue
- Increasing number of (slow) threads:



See also more recent stuff from Martin Hellmich on the DPM poster

3. ROOT I/O Testing Framework

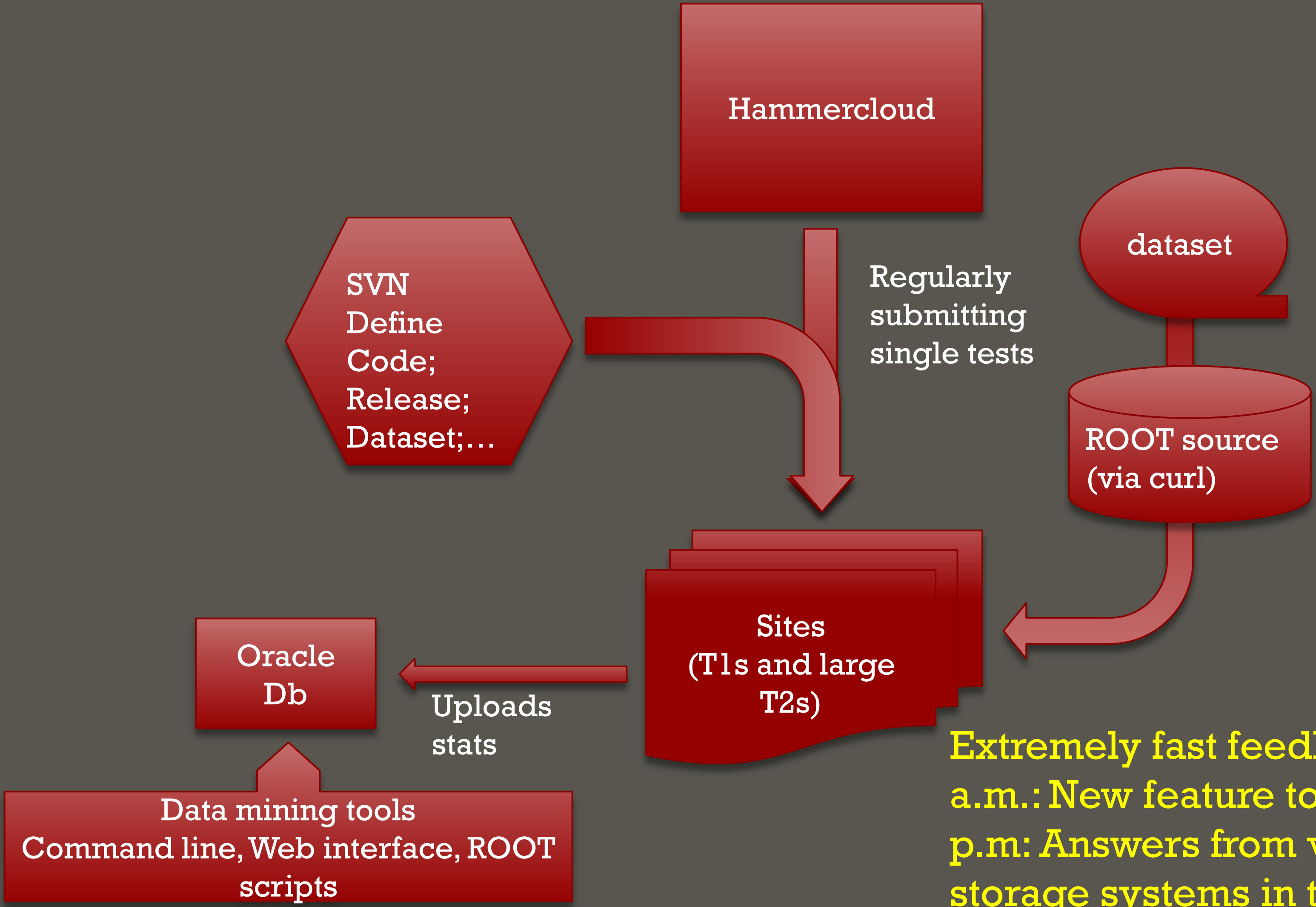
AIM: Rapidly test ROOT I/O developments in real production environments and monitor ATLAS SW I/O.

- Using hammercloud (HC):
  - Automatically submits functional or stress tests to a site.
  - Already of course a powerful tool for I/O testing used for site tuning; experiment blacklisting and middleware development

Modified HC to:

- Take our tests from SVN.
- Use identical data files: new versions pushed to sites.
- Heavily instrument tests. Upload stats to an oracle db
  - ROOT (e.g. reads; bytes);
  - WN (traffic; load; cpu type);
  - Storage type, access protocol etc.
- New web page for monitoring.

ROOT I/O Testing Framework



Hammercloud

Regularly submitting single tests

SVN Define Code; Release; Dataset;…

dataset

ROOT source (via curl)

Sites (T1s and large T2s)

Oracle Db

Uploads stats

Data mining tools
Command line, Web interface, ROOT scripts

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

ROOT I/O: Examples of Tests

- ROOT based reading of file with a simple TTree:
  - Provides metrics from ROOT (no. of reads/ read speed)
  - Like a user analysis
  - Reading all branches and 100% or 10% events (at random);
  - Reading limited 2% branches (those used in a real analysis)
- Using different ROOT versions
  - Including option of trunk of ROOT for feature testing
- ATLAS Specific Tests:
  - E.g Ntuple making in framework
- Instrumented user code examples
- Wide-Area-Network Tests

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

ROOT I/O: Example Results

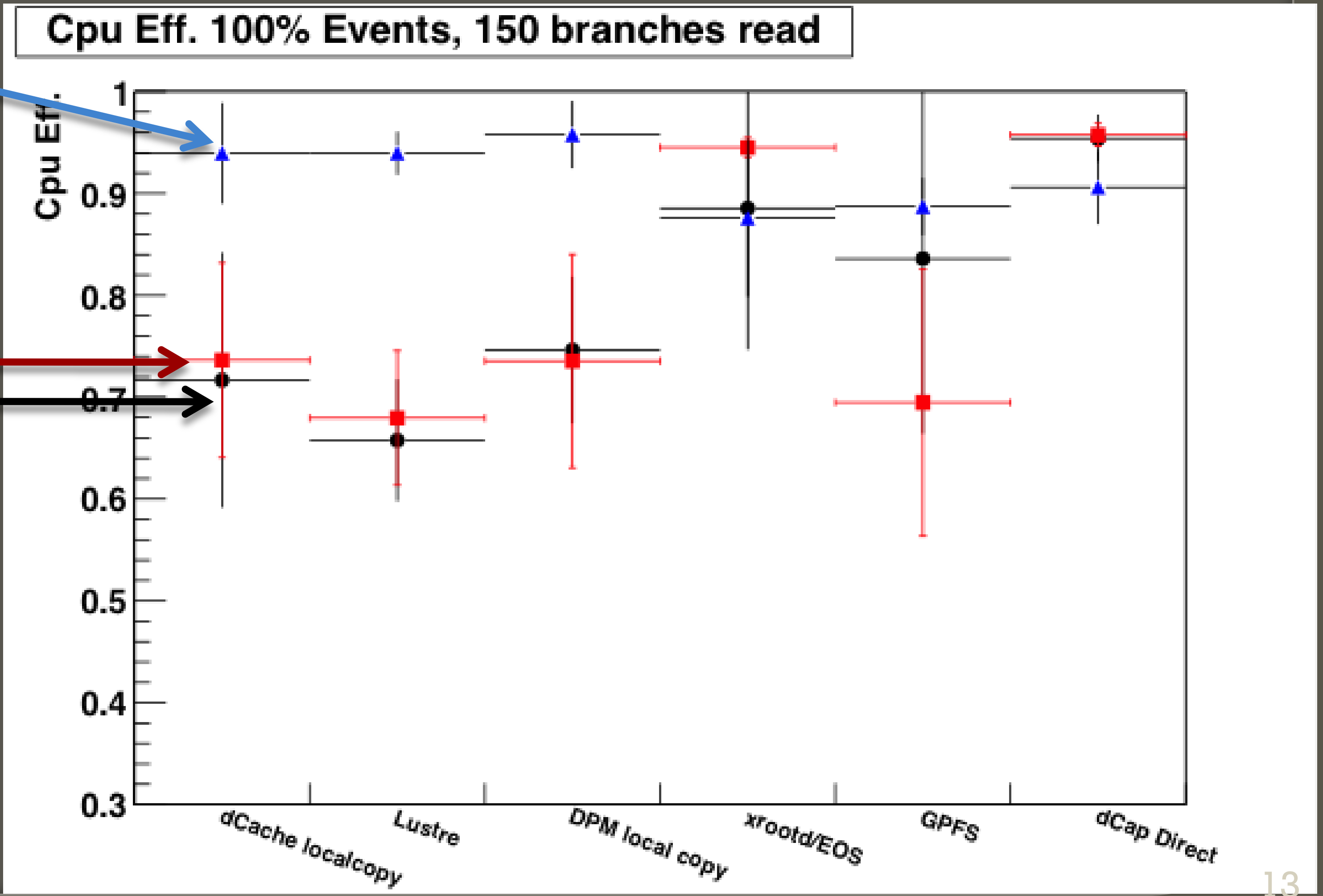
Tree with simple objects; ~ 12k events; 6k branches; 800M total size

Reading all branches TTreeCache (TTC) (30MB)

Reading only 2% of branches: 300 MB TTC or 30 MB TTC

Drop in cpu eff for limited braches except for some sites with vector read (dCap; xrootd)

Lots more data to mine!



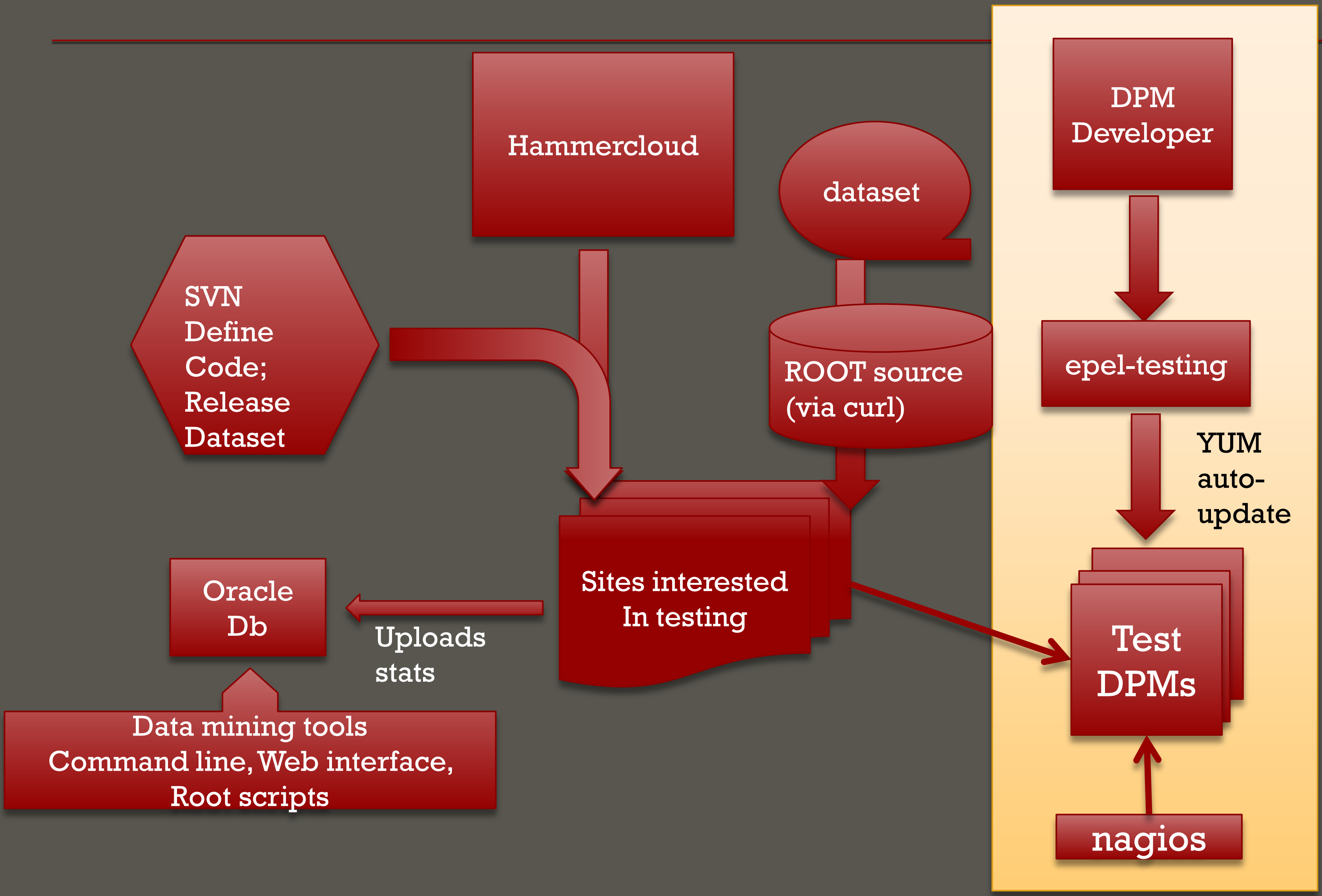
ROOT I/O Testing: plans

- Within ROOT I/O working group
- Test and develop core ROOT I/O:
  - Basket Optimisation
  - Asynchronous Prefetching
- Broaden tests:
  - More generic benchmark and /or
  - Real examples from other HEP experiments:
    - Happy to take examples to test…
- Use for site tuning:
  - As requested…
  - Need to compare to storage client/server monitoring. E.g. xrootd (see poster of Matevz Tadel) and http.

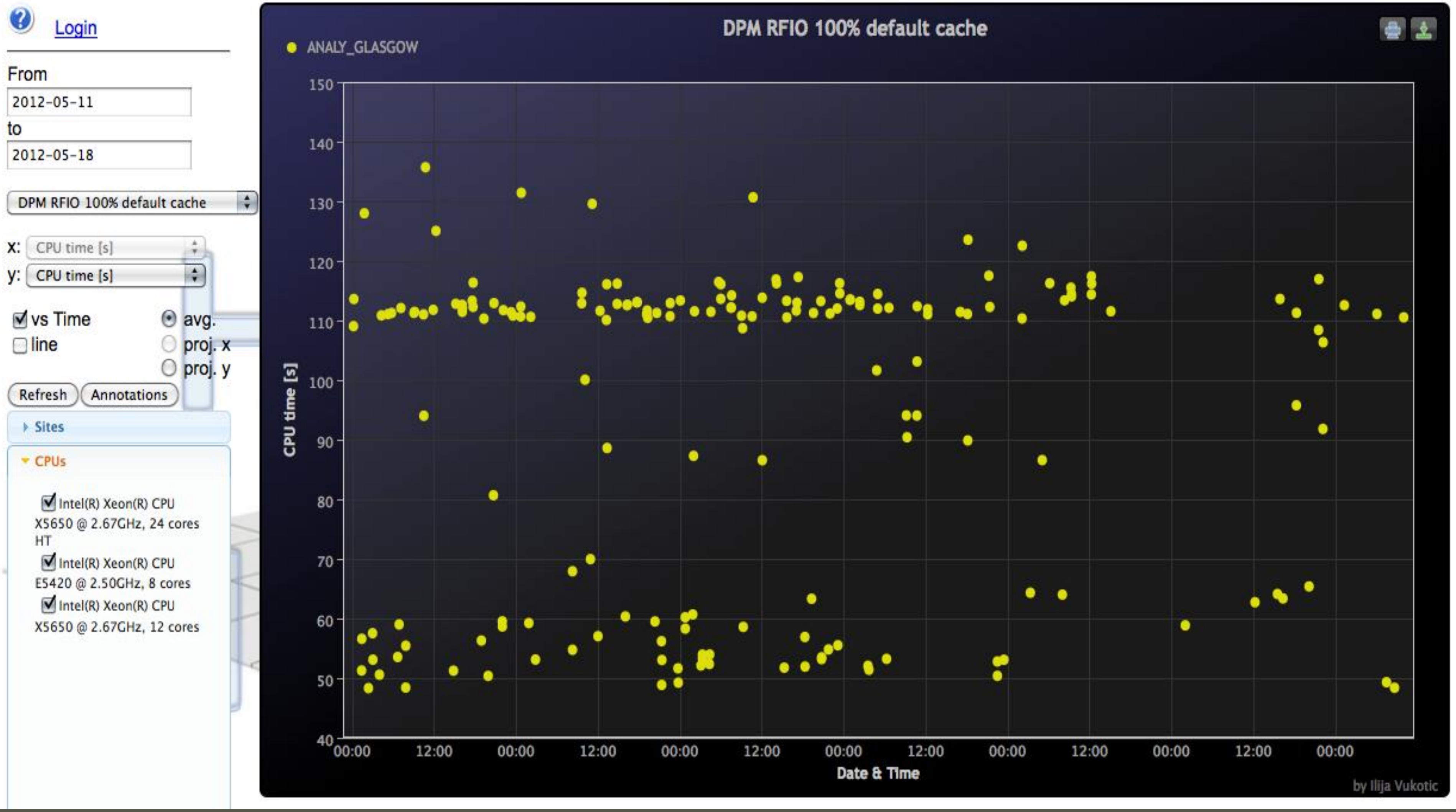
4. Middleware Testing Framework

- AIM: Make sure DPM releases work in the production environment. Test new features
- HC + SVN + custom stats uploaded to a new DB
- Similar tests as ROOT f/w but now evaluate
  - Current functionality:
    - rfio read and copy; gsiftp copy
  - New features / protocols:
    - WebDav: implemented;
    - NFS4.1 , xrootd (inc redirection): to come
- Point at test DPMs:
  - Currently Glasgow (SL5) and Edinburgh (SL6); other sites interested
  - Auto yum update from epel-testing repo
- Webpage for test results (as well as nagios)

DPM Testing Framework


Hammercloud
dataset
DPM Developer
SVN Define Code; Release Dataset
ROOT source (via curl)
epel-testing
YUM auto-update
Sites interested In testing
Oracle Db
Uploads stats
Test DPMs
Data mining tools Command line, Web interface, Root scripts
nagios

Middleware Testing Framework: Tracking RFIO reading



Login

From
2012-05-11
to
2012-05-18

DPM RFIO 100% default cache

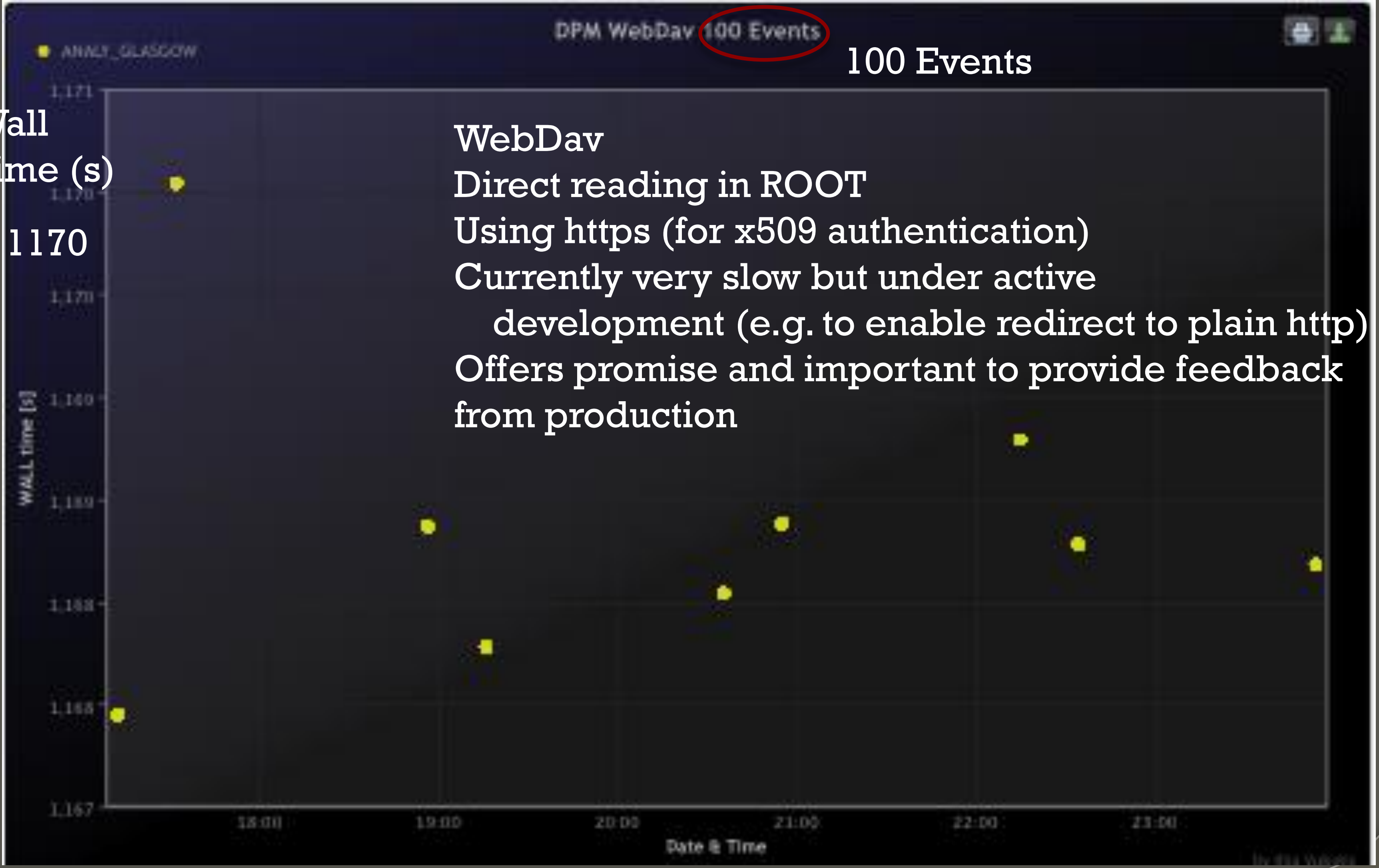
x: CPU time [s]
y: CPU time [s]

☑ vs Time
□ line

● avg.
○ proj. x
○ proj. y

☑ Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, 24 cores HT
☑ Intel(R) Xeon(R) CPU E5420 @ 2.50GHz, 8 cores
☑ Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, 12 cores

Middleware Testing Framework: Developing new protocols



100 Events

Wall Time (s)

1170

WebDav

Direct reading in ROOT

Using https (for x509 authentication)

Currently very slow but under active development (e.g. to enable redirect to plain http)

Offers promise and important to provide feedback from production

Comparing and contrasting

Different expertise and outcomes

Realism:

- Experiment can run its own s/w and want to: so need a “real” test.
- Site and developer may not: but need a “realistic” test.
- Vendor can’t run experiment code: need a synthetic benchmark.

Instrumentation:

- Site measurements of hardware performance.
- Middleware measurements of system internals.
- Experiment measurements of application.

Automation:

- Needed if system is to provide monitoring

Scale:

- Monitoring only requires single test at a time.
- Other testing: learn from both though contention only at scale.

Production:

- Site / Vendor / Developer may want to test outside production env.
- Specific examples like that here are easy.
- Generic hammercloud-in-a-box: requires experiment; m/ware tweaks

Comparing and Contrasting

Example	Vendor Storage	Low-level Middleware	Middleware framework		ROOT I/O Framework	
Use	Vendor Kit/ Site Tuning	Middleware Scale tests	M/ware Function	M/ware Features Protocols	Site quality level	VO soft / data
Automation	✖	✖	✔	✖	✔	✔
Scale	Stress	Stress	Single	Both	Both	Both
Environment	Test	Test	Production			
Instrumentation	Hardware	Middleware	Application			
Realism	✖	✖	✔	✔	✔	✔✔

Some reuse of tests but a lot of differences too

Conclusions

- I/O testing is important from a variety of perspectives.
- We have built tests for many of these
  - Used for vendor solutions; site tuning; middleware and application development.
- Much can be reused from these examples
  - But need for customizations remain.
- Working towards making it more generic
  - Towards meeting goals outlined in WLCG TEG