The ATLAS Production System

MC and Data Reprocessing

Rodney Walker

Ludwig Maximilians Universität
Munich

2nd Feb, 2009 / DESY Computing Seminar

Outline

ATLAS Distributed Computing

- MC Production
- Data Reprocessing
- Group and user analysis, calibration

Outline

MC Production for ATLAS

- Traditionally a distributed activity, e.g. ZEUS Funnel. The difference is in the volume
- Plan to simulate 10% of 200Hz data taking rate
- Geant4 - 10mins per event, 800MB RAM, reconstruction - 1min per event, 2-3GB RAM
- Steady state of 20[ev/s]*600[s/ev] = 12000 cpu cores
- Production is bursty due to release cycle. Currently fill up to 25000 cores - but includes T1 resources to be used for reprocessing.

Outline

Computing Model

- Expect to reprocess all data twice per year
  - apply new alignment, calibration, algorithm improvements
  - more often during commisioning
- Historically not distributed
- Co-located with RAW data and databases - re-use experiment primary reconstruction infra-structure
- ATLAS will reprocess at T1s to maximise cpu and tape resources
- Reprocess while taking data - quicker and more often
- New challenges

Distributed Reprocessing Challenges

- Large(2GB) input files different for each job, c.f. 50 MC jobs skip through 1 EVNT file
- Data is stored on tape at the T1s
  - also at T0 - write once, read never
  - need automated pre-stage to keep cpus full
  - varied T1 experience of non-archival tape systems
- Conditions Data access
  - cannot use T0 Db directly - RTT, load, scaling
  - distributed(Streams) to T1 Oracle RACs
- Haven't been practising for 5yrs

Outline

History of ProdSys

- Always consisted of 3 distinct Grids/middleware: OSG, LCG, NDGF
- OSG is LCG minus gliteWMS. NDGF very different.
- Handle heterogeneity using central database of job definitions. Each grid executes them, in their own way, and updates Db.
- Test in DC1, DC2, CSC challenges/productions.
- OSG and LCG performed badly. NDGF worked well.

LCG problems

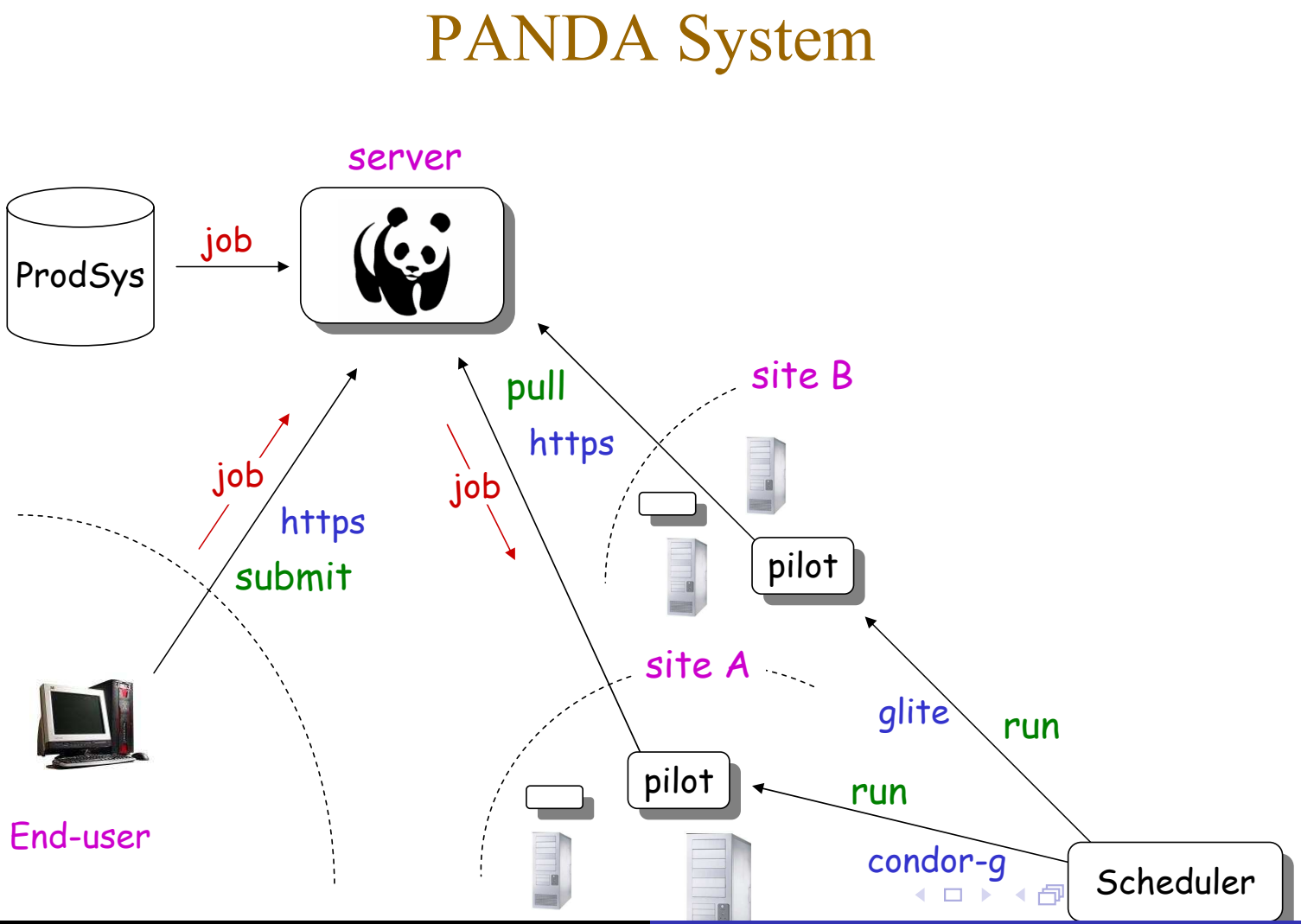
- EDG WMS slow to submit. Could not fill cpus. High failure rate. Poor brokerage - bad info. No data-job co-location. Job priorities not followed.
- No data placement - did not move input data near free cpu. Accessing data remotely increased failures, SE overload, confusion.
- Pure Condorg submission addresses WMS issues but not data placement
- Tinkering around the edges did not improve it.

OSG solution

- OSG performed slightly worse than LCG and this prompted major rethink. Less Grid-baggage.
- Designed data-centric production system, factorising out grid frailties
- Uses new data handling system(DDM) to move data to cpu
- Uses pull-method - pilot job has survived Grid middleware when it asks for real job. Nod to LHCb Dirac system.
- Used very successfully for OSG production in 2007
- Canada switched from LCG to Panda production, late 2007, with immediate and impressive improvement. Led to proposal to use Panda everywhere - transition early 2008.

Outline

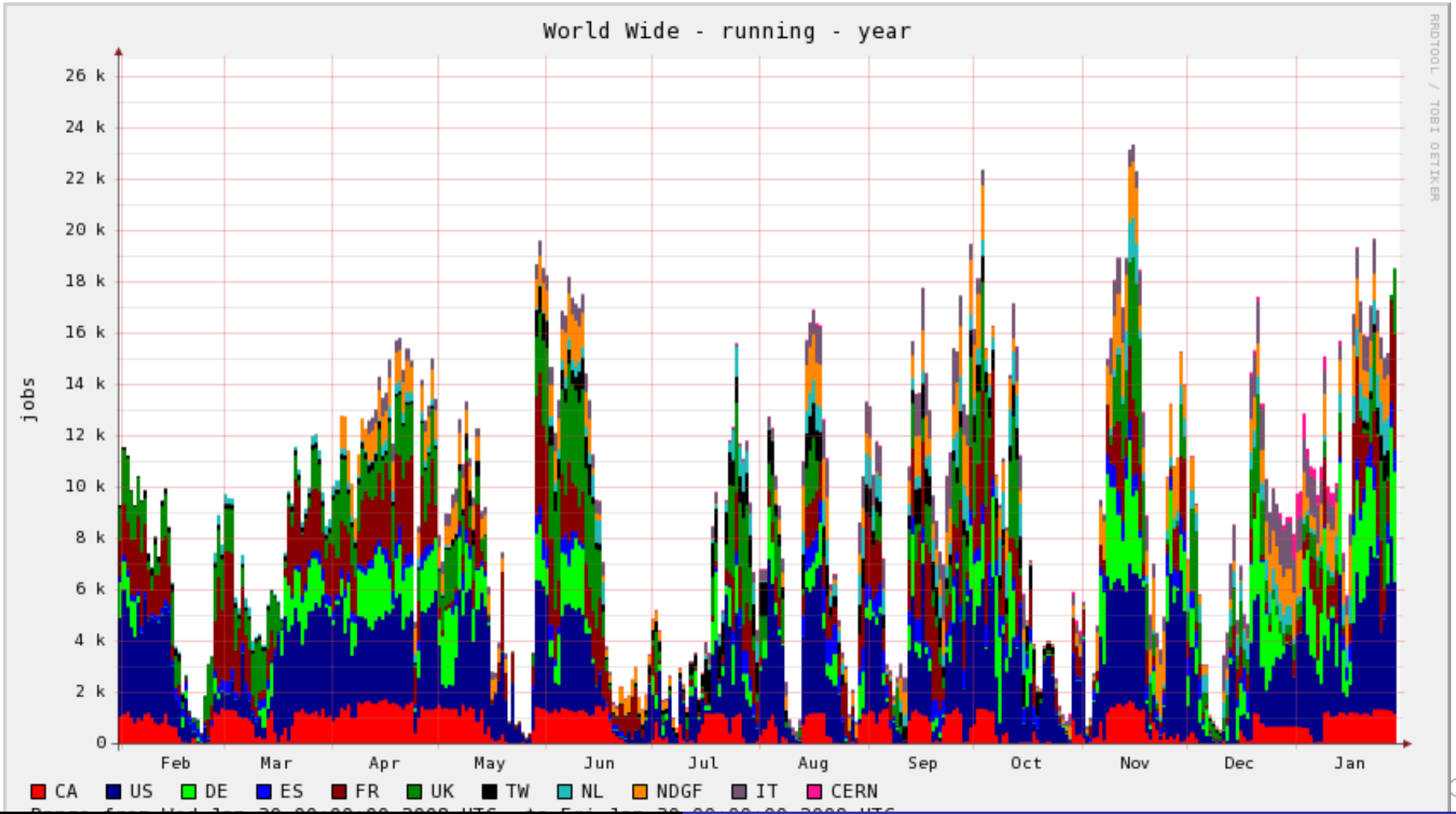
Panda


PANDA System
server
ProdSys
job
pull
https
site B
job
https
submit
job
pilot
site A
glite
run
End-user
pilot
run
condor-g
Scheduler

Data placement

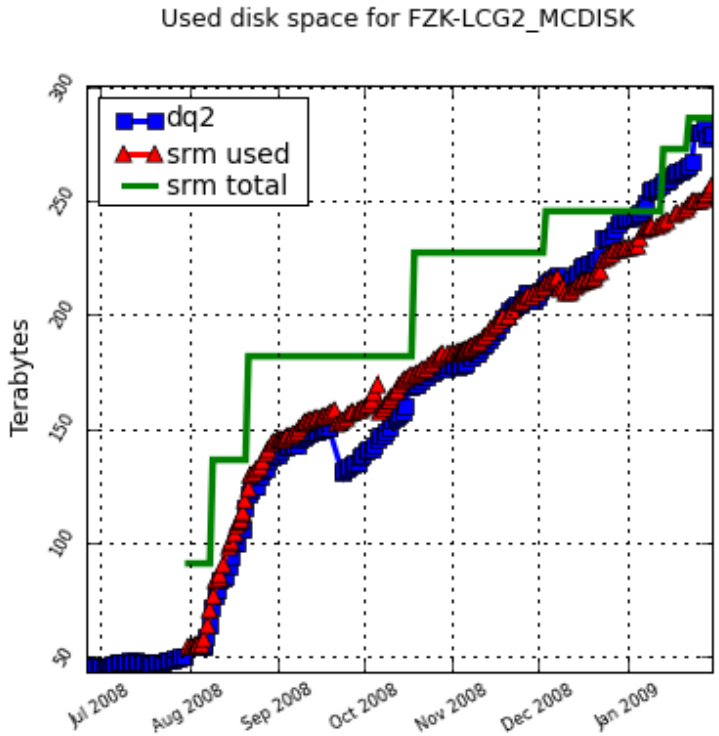
- T1 cloud model imposed. Input data is located at T1 and output data must be copied there for job to finish successfully.
- Jobs are assigned to T2 and input data is copied there before the job can run. Input guaranteed to be local when pilot picks up job.
- Output data first stored to local SRM, and then asynchronously to T1. Job not finished until output arrives at T1.
- If local SRM is broken then jobs fail and no more assigned. Previously employed failover methods led to poor tracability.

Performance



Data Volume

- Accumulation of MC data at T1s
- 1-2TB per day, from T1 and it's T2s
- Timely addition of disk capacity
- Deletion campaigns have negligible impact

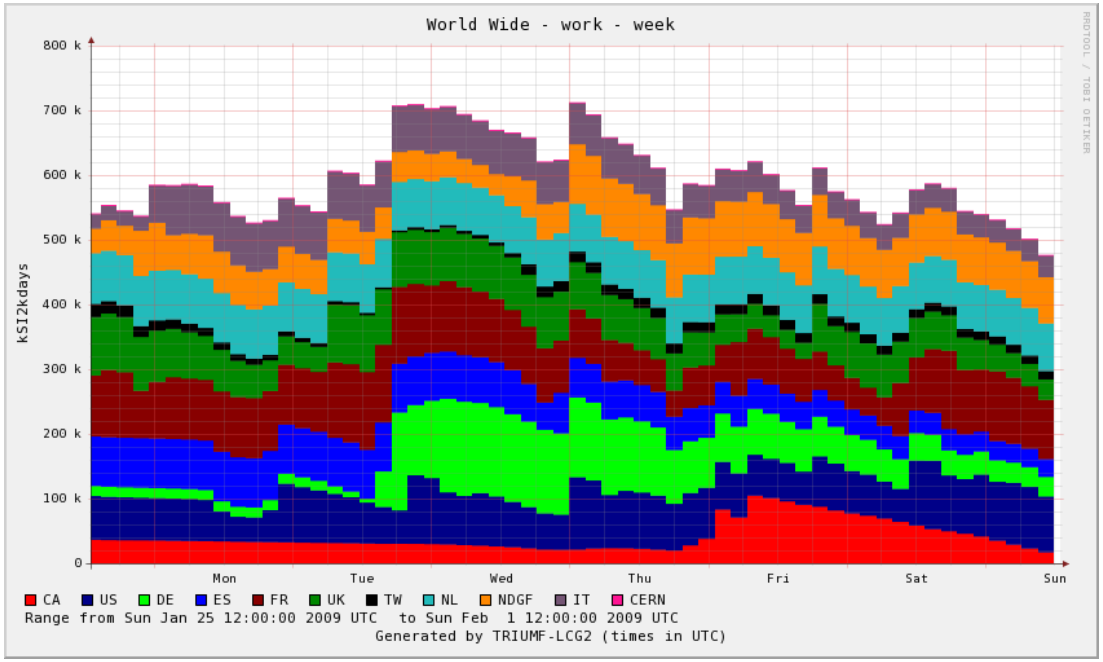


Task Assignment to Clouds

- Only consider cloud with necessary input data
- Work to be done in cloud is calculated using cpu time per job, in SI2kdays
- Sum over assigned tasks, $work = \sum(Njobs_i - Ndone_i).cpu_i$
- only count tasks with equal or higher priority. High priority task assignment blind to lower priority tasks.
- Max work per cloud based on MoU shares, e.g.US 20%, max is 20kSI2kdays
- cpu time per job is rough guess and usually way off. Plan feedback mechanism to reset to actual time taken by first jobs.

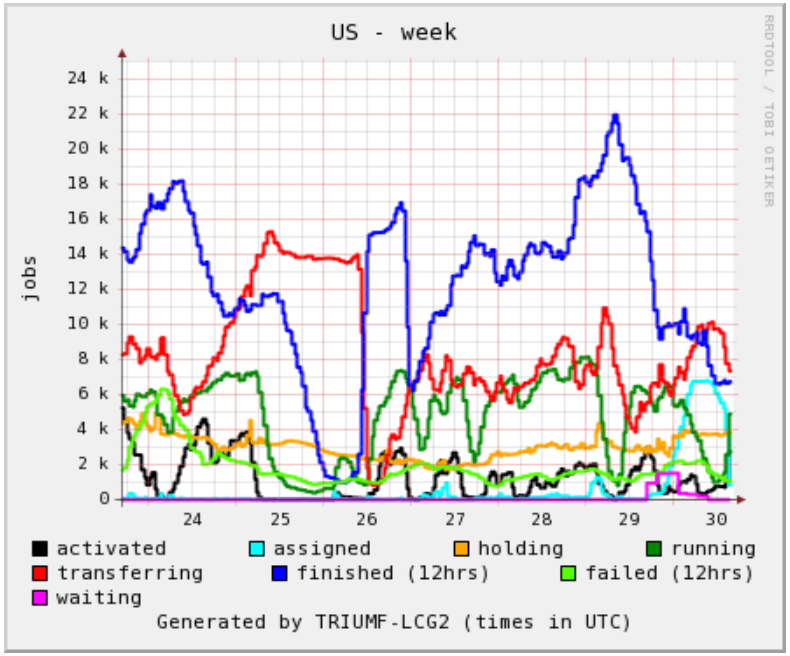
Task Assignment to Clouds II

- For clouds below their work threshold, choose one with lowest work/Nrunning
- Many ways to get it wrong, e.g. ignore low priority tasks in threshold, and assign in multiple parallel threads.



Tuning

- Transferring jobs contribute to cloud *work*
- Prevents new task assignment
- Plan to subtract transferring jobs contribution from *work*



Reprocessing using ProdSys

- Jobs to reprocess data are defined in Db, and tasks assigned to T1 based on data location - only 1 copy of RAW among T1s.
- Jobs run at T1 but data input is similar to MC at T2 case. Jobs remain 'assigned' while 10-20 file datasets are recalled from tape.

Outline

Pre-staging RAW data

- DDM uses srmBringOnline and srmLs
- Do not know tape distribution of files. Some assumptions about how they were written, i.e. time ordered during data taking.
- Must request large number of files and hope SRM and native tape system can optimize.
- But not more than read cache size. Files must stay on disk until processed. SRM pin does not work, so just periodically BringOnline.
- No manual T1 work, apart from recovering stuck/broken tapes.

Outline

Conditions Db Access

- Calibration data, mostly muon system(20MB), valid for 1 day
- Detector Control System(DCS), 1-2GB per day. Valid for few seconds to few hours.
- Flat files, LAr calibration, 100MB/day - pointers from Db
- Athena access CD before and during first event, but can go back for more
- 10mins of DCS taken, 1min before and 9 after 1st event timestamp
- Db resource usage large at start of job then little or none

Db Access Methods

- ATLAS uses COOL and CORAL to abstract access to Db
- COOL Db is in Oracle RAC - direct access. Hardware dependant. Works well for local, controlled access. Vulnerable to chaotic access.
- Create SQLite file for run range and access on local file system. Used in MC production, proven to scale perfectly. Good for well-defined uses, with small range. Needs auto-procedure to create and distribute files. Xmas 2008 reprocessing excercise done like this.
- FroNTier web-based proxy to Oracle RAC. Developed and used by CMS. May offer best of both worlds - under test for ATLAS. Suited for chaotic access from T2, e.g.user analysis.

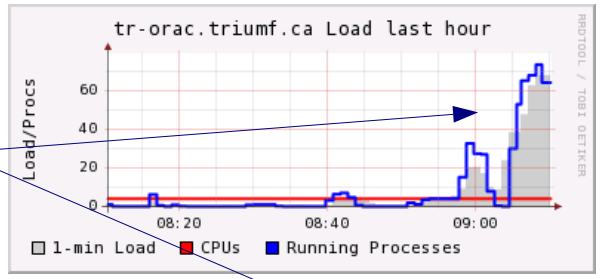
Frontier for Scaling Db Access

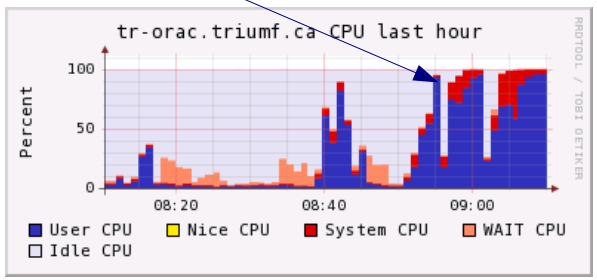
- Frontier client, integrated into CORAL, converts SQL query into http url.
- Frontier server decodes back to SQL and executes on close Db. Result is compressed and coded into http to send back.
- Identical SQL $\Longrightarrow$ identical url - cache result!
- Local caching web proxy (Squid) caches query results
- Configure to store ? results and also larger objects
- Avoids the multiple RTTs of SQL query
- Protects Oracle RAC from remote WN attack
- May benefit from caching (DCS 10min quantization)

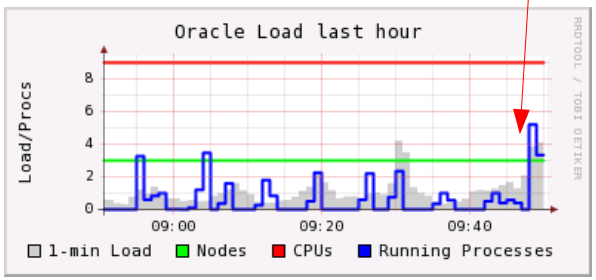
Frontier for Scaling Db Access

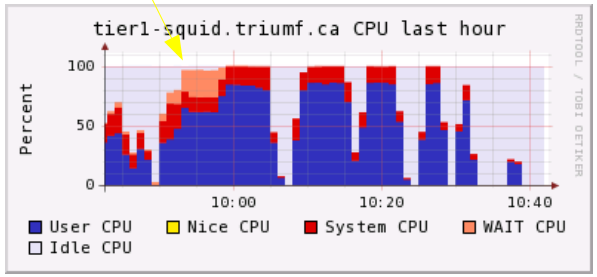
TRIUMF RAC vs Frontier scaling

10min quanta: bunches of 24 jobs use same CD









FroNTier Deployment

- Assuming test go well ...
- FroNTier servers required at each T1 with Squid (accelerator-mode)
- Squid caching proxy at T2s. CMS require this too, suggests a new EGEE service node.
- Testbed exists in DE, FZK and LMU - test with ID alignment and muon calibration.
- Proposal to reprocess also at T2(DESY-HH). Early adopters install squid?

Summary

- MC production system scales to meet ATLAS requirements
- With the addition of pre-staging, same system works for reprocessing needs too

- Outlook
  - Proliferation of small files overloads DDM, FTS, SRM, LFC. Must merge at T2 before upload to T1.
  - Pre-stage testing and tuning. Handling of lost files, broken tapes.
  - Enabling chaotic(non-T0/1) Conditions Data access