Condor at the RACF

Multicore jobs in our workflow and other places for input-driven scheduling

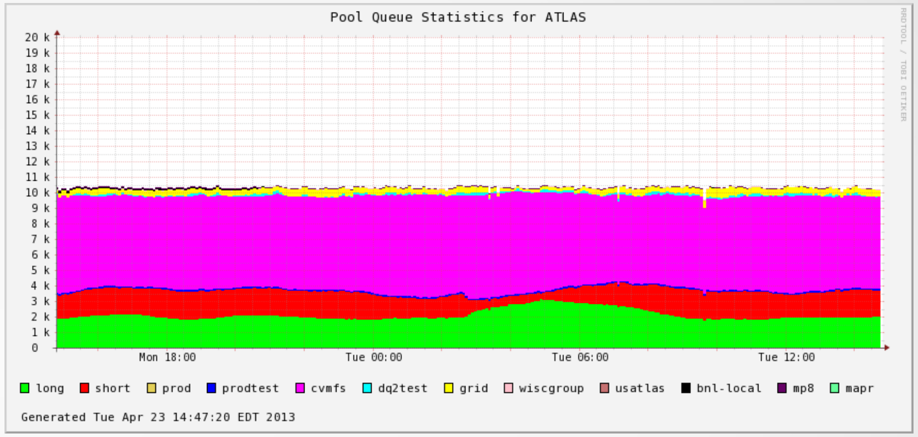
May 2013

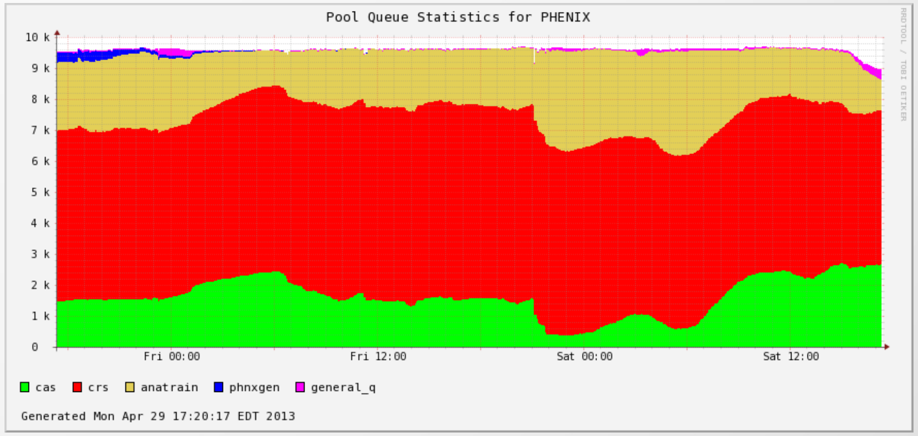
William Strecker-Kellogg
Brookhaven National Lab

Talk Outline

RHIC/ATLAS Computing Facility

- Who are we?
  - Offline computing for RHIC
  - Tier-1 for ATLAS in US
- Condor pools at the RACF
  - 18.5kCPU RHIC
    - 9.7kCPU (PHENIX)
    - 8.8kCPU (STAR)
  - 11.0kCPU ATLAS
- Characteristics
  - RHIC—federation of individual users, some central control, data on nodes
  - ATLAS—tightly controlled, master batch system (PANDA), central data





RACF Overview

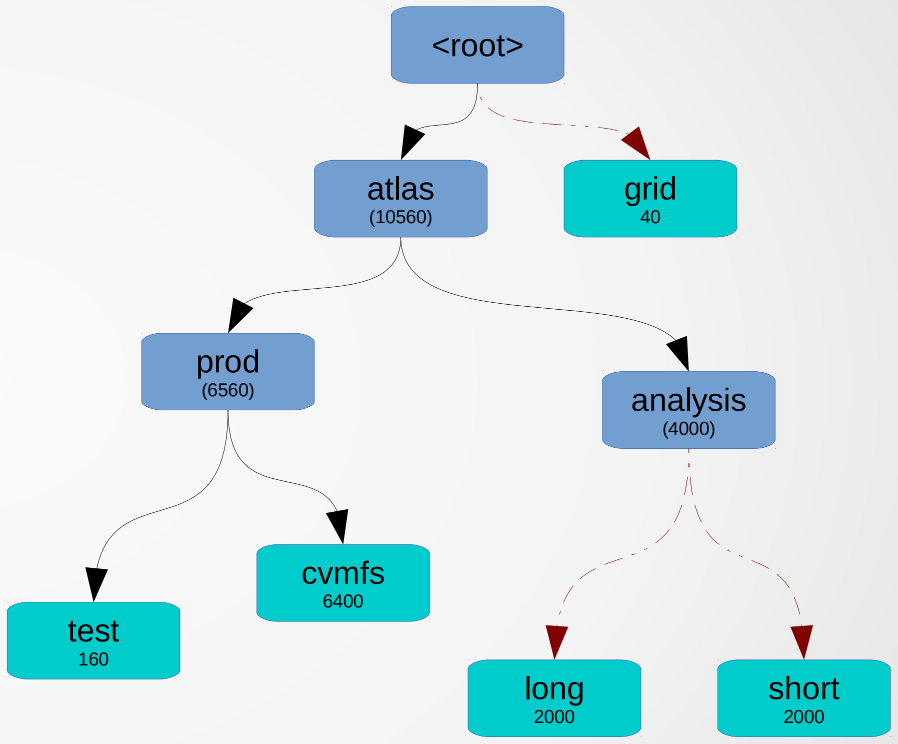
- Old RHIC detectors
  - PHOBOS
  - BRAHMS
- Smaller experiments, neutrino and astro
  - LBNE
  - DAYABAY
  - LSST
  - EIC
  - Theory group

- ATLAS supports various smaller groups
  - Local Tier-3
  - Wisconsin

- Separate cluster for some, others integrated into ATLAS
- Total of smaller groups around 1.5k CPUs

ATLAS Structure

- Flat, uniform farm in both hardware and software
- PANDA Queues map to AccountingGroup(s)
- Hierarchical structure
- Only leaf nodes have jobs submitted to them
- Spillover between arbitrary (related) groups
  - *short* and *long* can share but are constrained to 4k by parent (*analysis*)
  - *grid* can accept all surplus not used by ATLAS
- Version Makeup
  - Farm: 7.6.6 SL5.3
  - Central Manager: 7.6.9 SL5.3
  - Submit Nodes: 7.6.10 SL6.3



Key

Turquoise
    leaf group with jobs
Blue
    middle group, quota is sum of children, no jobs
Red arrow
    group has accept_surplus on

Node Consistency

- Theme: keep nodes the same!
  - Even with tools like puppet, partitioning the farm by config is inefficient
- Balance between queues changes frequently
  - Made 9 adjustments this year so far
- Queues with non-standard config still need restart
  - Can't change slot count or make pslots
- Restart = full drain = inefficient
- Even harder for cloud nodes
  - Maintain balance with nodes coming and going

Multicore Jobs in ATLAS Workflow

- Initially a test queue with a group under production
  - Static 24-core machines with 2x8CPU and 8x1CPU slots
- Discovered problem with groups—wanted quota usage to be #CPUs (default slot-weight)—but jobs wouldn't match correctly (see ticket #2958)
  - Fix fails when *any* group has **accept_surplus** enabled
- We need accept_surplus *and* multicore jobs in groups
  - Kludge fix: set slot-weight to 1

Q: How to integrate multicore jobs into existing groups?

Q: How to integrate high-memory jobs into existing groups?

A: Partitionable Slots (pslots)!

Not Working With Group Quotas

Partitionable Slot Requirements

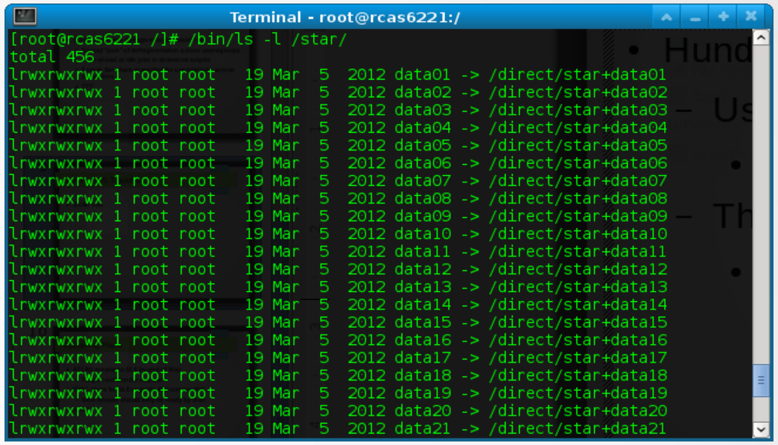
- Want to be able to slice by RAM, CPU, and possibly Disk
  - In the future slicing by any local-resource (GPU...)
- Want sane (configurable) defaults for existing job-configs
  - Request: 1 core, TotalRam/TotalCPU memory, etc...
- Want no complete starvation of larger jobs that can be accommodated somewhere
  - Implies some form of defragmentation/draining
- Ideally defragmentation would be group-aware
- Every node would become one big pslot

Defragmentation In Detail

- Scheduler-aware defragmentation would help
  1. Spread “pain” of defragmentation across users/groups
  2. Ensure fair-share respected for users/groups across schedulers
- Implementation ideas
  1. Look-ahead at queue to determine defrag targets
     - Looking at demand from idle jobs in queue, or allowing users to provide targets
  2. Keep historical data to improve heuristics
     - “This user's jobs in this cluster typically run for X hours”, etc...

STAR's NFS Handling

- Hundreds of NFS filesystems from 2Tb to 10Tb each
  - Users can access them freely
    - ...so they can easily break them
- There is no global picture of resource-usage at the filesystem level
- Concurrency limits are nice but users can easily lie (or be ignorant) about what their jobs are doing
  - Would be a large maintenance burden as these change somewhat often



```
[root@rcas6221 /]# /bin/ls -l /star/
total 456
lrwxrwxrwx 1 root root   19 Mar  5  2012 data01 -> /direct/star+data01
lrwxrwxrwx 1 root root   19 Mar  5  2012 data02 -> /direct/star+data02
lrwxrwxrwx 1 root root   19 Mar  5  2012 data03 -> /direct/star+data03
lrwxrwxrwx 1 root root   19 Mar  5  2012 data04 -> /direct/star+data04
lrwxrwxrwx 1 root root   19 Mar  5  2012 data05 -> /direct/star+data05
lrwxrwxrwx 1 root root   19 Mar  5  2012 data06 -> /direct/star+data06
lrwxrwxrwx 1 root root   19 Mar  5  2012 data07 -> /direct/star+data07
lrwxrwxrwx 1 root root   19 Mar  5  2012 data08 -> /direct/star+data08
lrwxrwxrwx 1 root root   19 Mar  5  2012 data09 -> /direct/star+data09
lrwxrwxrwx 1 root root   19 Mar  5  2012 data10 -> /direct/star+data10
lrwxrwxrwx 1 root root   19 Mar  5  2012 data11 -> /direct/star+data11
lrwxrwxrwx 1 root root   19 Mar  5  2012 data12 -> /direct/star+data12
lrwxrwxrwx 1 root root   19 Mar  5  2012 data13 -> /direct/star+data13
lrwxrwxrwx 1 root root   19 Mar  5  2012 data14 -> /direct/star+data14
lrwxrwxrwx 1 root root   19 Mar  5  2012 data15 -> /direct/star+data15
lrwxrwxrwx 1 root root   19 Mar  5  2012 data16 -> /direct/star+data16
lrwxrwxrwx 1 root root   19 Mar  5  2012 data17 -> /direct/star+data17
lrwxrwxrwx 1 root root   19 Mar  5  2012 data18 -> /direct/star+data18
lrwxrwxrwx 1 root root   19 Mar  5  2012 data19 -> /direct/star+data19
lrwxrwxrwx 1 root root   19 Mar  5  2012 data20 -> /direct/star+data20
lrwxrwxrwx 1 root root   19 Mar  5  2012 data21 -> /direct/star+data21
```

- Solution was to harvest NFS usage with *lsof* and adjust users' prio-factor accordingly
- Overall lack of visibility in condor into what a job is doing
- Another opportunity for user-provided data to drive scheduling
  - Adjusting prio-factor is inelegant
  - So is passing data in tons of custom classads

Data Driven Scheduling

- Common theme $\longrightarrow$ user data can improve scheduling
  - Collected data more accurate then what the user will claim if asked
  - Users cannot mislead in stating job requirements
    - Concurrency limits require jobs to ask for resources
- Condor often running under other batch systems with better insight into upcoming work
  - PANDA in ATLAS
  - STAR scheduler
  - VM Provisioning
- A flexible method for condor to harvest/accept more data?

Data Driven Scheduling (cont...)

- More cases where statistics can help
  - Given a queue of idle work, no a priori knowledge of the throughput requirement
    - Context: VM provisioning for a given work queue
  - Historical data collection can help—up to a point
    - Most users are not malicious and can be trusted to honestly represent what their jobs do
    - Combination of heuristics and trusting users could be more effective than either

Virtualization Testbed

- Described last year—see my CHEP2012 paper
  - Thin wrapper around condor to allow *trusted* VM execution inside our firewall
  - No restrictions on access to NFS/other UID-based services
  - Usual problems and limitations from NAT
- STAR is using on 480 cores to re-run some 2004 production code in Scientific Linux 4
- SL6—could replace with a container-based approach
  - CGroups and libvirt leveraged to make it easy with a minimum of extra coding

Checkpointing

- Testing DMTCP checkpointing, mainly for RHIC users
  - ATLAS case is too complex and there is no storage easily available for images
  - Cloud context even trickier, no local storage, bandwidth usage charges
- Images on the submit node would require user-aware disk-space monitoring and fairness (feature in 7.9.x?)
- Images in NFS would be easier—developing a DMTCP wrapper that places images in a user-designated NFS directory
  - NFS Quotas provide fairness/limits outside condor

Virtualization Testbed

- Described last year—see my CHEP2012 paper
  - Thin wrapper around condor to allow *trusted* VM execution inside our firewall
  - No restrictions on access to NFS/other UID-based services
  - Usual problems and limitations from NAT
- STAR is using on 480 cores to re-run some 2004 production code in Scientific Linux 4
- SL6—could replace with a container-based approach
  - CGroups and libvirt leveraged to make it easy with a minimum of extra coding

High CM Availability

- Port channel blew on line card connecting our central managers
- Current Status: condor_had
  - Not possible since we use flocking extensively
    - all RHIC → all RHIC
    - ATLAS → RHIC (PHENIX)
- Condor View and Tiered Collectors
  - Replicate collector data across nodes
  - Bring up a negotiator on one
  - Don't want to partition pool by config

Data Collection Troubles

- Attempted to collect all ClassAd data into MongoDB instance
- Parse each schedd's *history* file and dump to DB
- Encountered scalability problems
  - Data growth—MongoDB stores keys for *every* field
    - Many Gb every day—lots of short-running jobs
  - No Collection-level locking—very poor write performance without multiple databases
    - Default partitioning was collection-per-experiment
    - Not worth the hardware to throw more hardware at it
- Will investigate plumage—does it store <u>everything</u>?

Upcoming Plans

- ATLAS moving to SL6 by end of May
  - Target next Condor release?
- RHIC plans for SL6 upgrade this summer/fall
  - Next release should long be ready by then
- PHENIX Mapping jobs to data with job-RANK and network-topology-aware scheduling
  - Plans are for this summer/fall.

Thank You!

Questions? Comments?