ATLAS Autonomous Multicore Provisioning

Dynamic Allocation with Condor

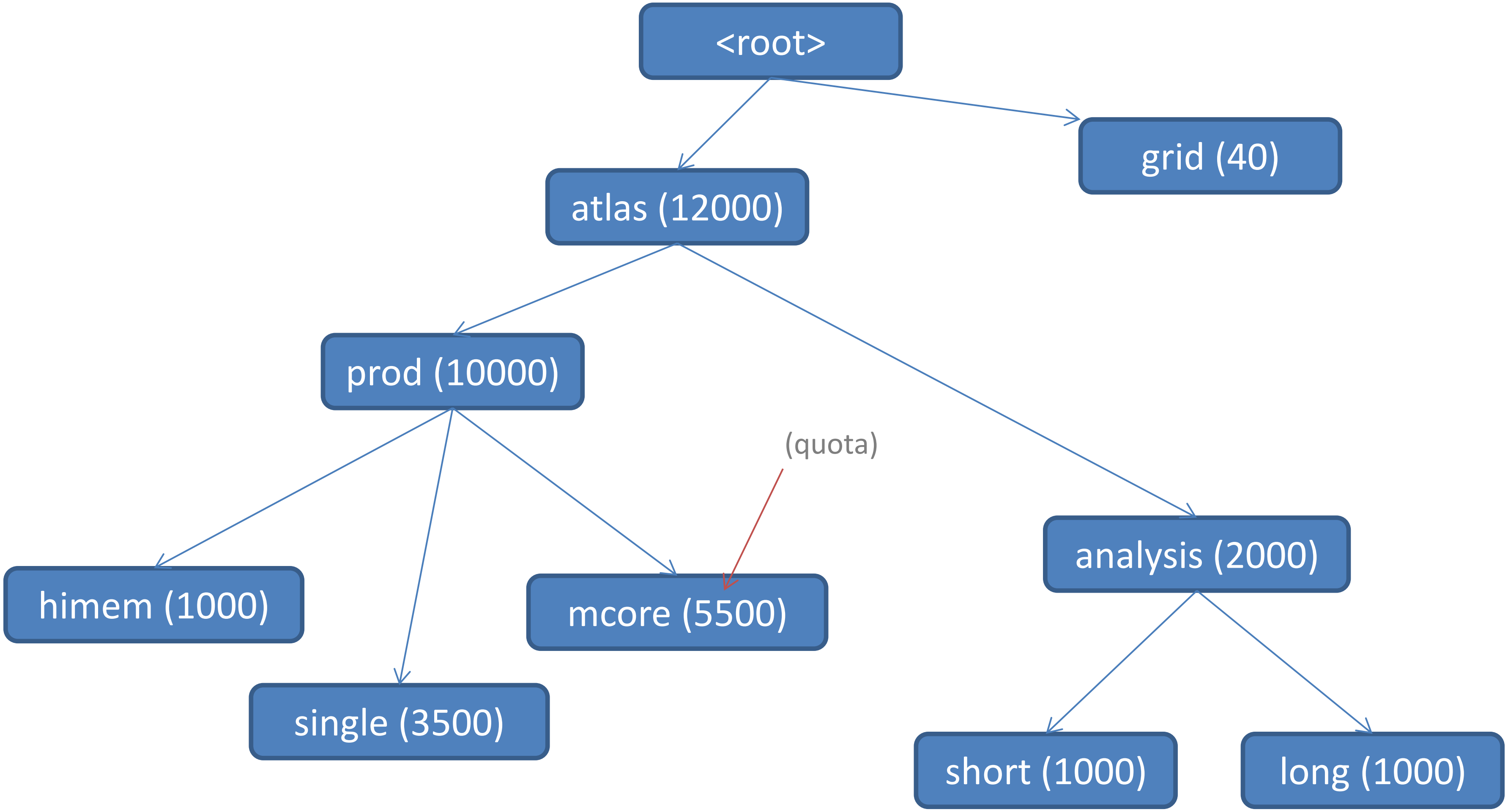
William Strecker-Kellogg

ATLAS Tree Structure

- Use hierarchical group-quotas in Condor
  - Leaf-nodes in the hierarchy get jobs submitted to them and correspond 1:1 with panda-queues
  - Surplus resources from underutilized queues are automatically allocated to other, busier queues
    - Quotas determine steady-state allocation when all queues are busy
  - Quota of parent groups are the sum of their children's quotas

(see next slide for diagram)

ATLAS Tree Structure

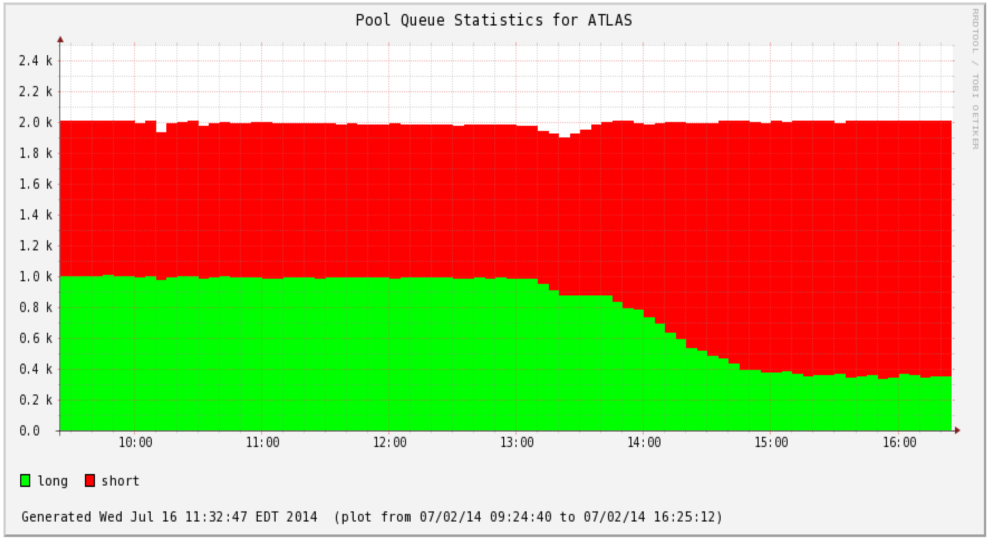


Surplus Sharing

- Surplus sharing is controlled by boolean *accept_surplus* flag on each queue
  - Quotas / surplus are normalized in units of CPUs
- Groups with flag can share with their siblings
  - Parent groups with flag allow surplus to “flow down” the tree from their siblings to their children
  - Parent groups without *accept_surplus* flag constrain surplus-sharing to among their children

Surplus Sharing

- Scenario: **analysis** has quota of *2000* and no *accept_surplus*; **short** and **long** have a quota of *1000* each and *accept_surplus* on
  - short=1600, long=400…possible
  - short=1500, long=700…impossible (violates analysis quota)



Partitionable Slots

- Each batch node is configured to be partitioned into arbitrary slices of CPUs
  - Condor terminology:
    - Partitionable slots are automatically sliced into dynamic slots
- Multicore jobs are thus accommodated with no administrative effort
  - Farm is filled depth first (default is breadth first) to reduce fragmentation
    - Only minimal (~1-2%) defragmentation necessary

Where’s the problem?

- Everything works perfectly with all single-core
- However… Multicore jobs will not be able to compete for surplus resources fairly
  - Negotiation is greedy, if 7 slots are free, they won’t match an 8-core job but will match 7 single-core jobs in the same cycle
    - If any multicore queues compete for surplus with single core queues, the multicore will always lose

- A solution outside Condor is needed
  - Ultimate goal is to maximize farm utilization

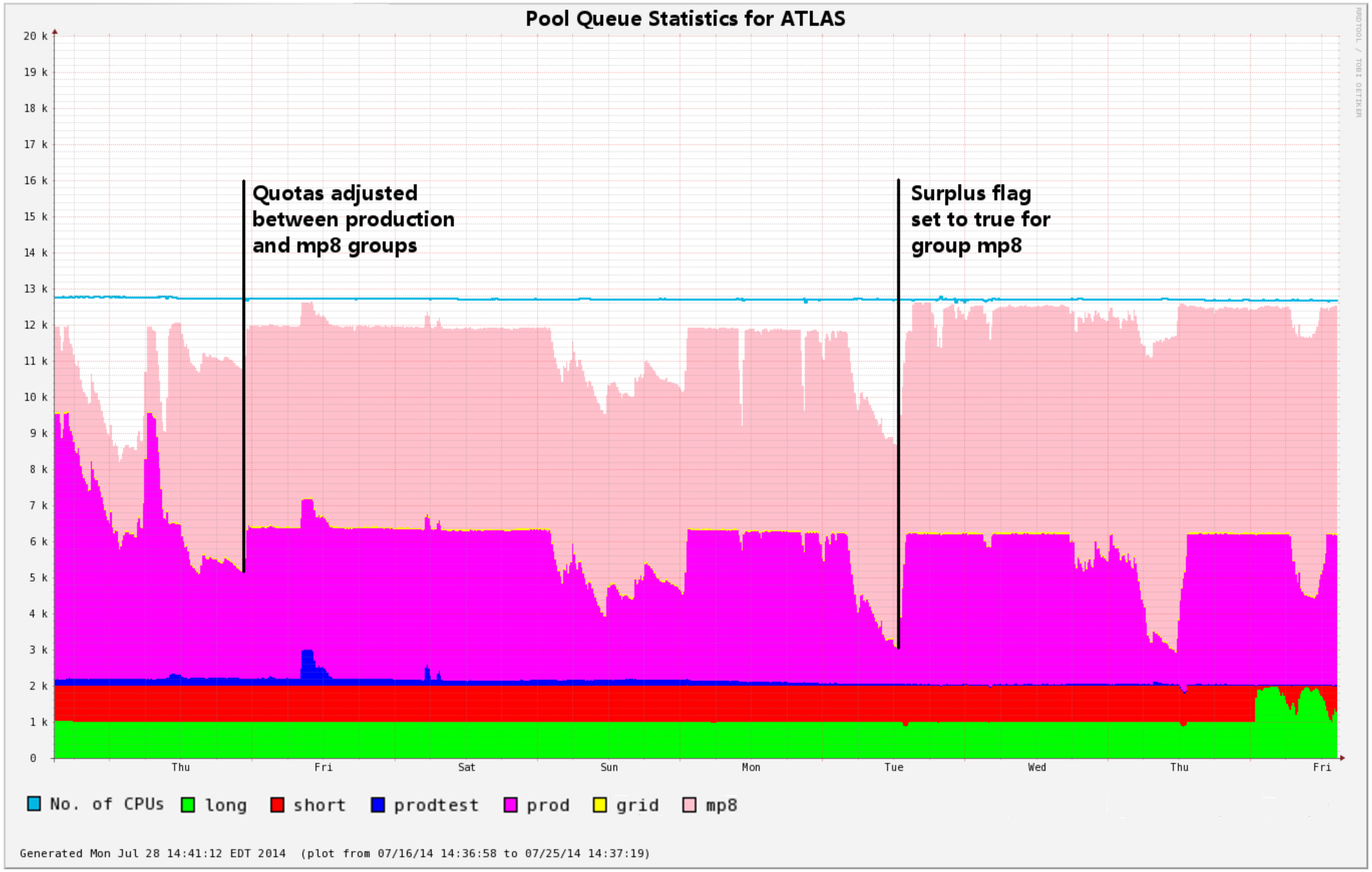
Dynamic Allocation

- A script watches panda queues for demand
  - Queues that have few or no pending jobs are considered empty
  - Short spikes are smoothed out in demand calculation
- Script is aware of Condor’s group-structure
  - Builds tree dynamically from database
    - This facilitates altering the group hierarchy with no rewriting of the script

Dynamic Allocation

- Script figures out which queues are able to *accept_surplus*
  - Based on comparing “weight” of queues
    - Weight defined as size of job in queue (# cores)
  - Able to cope with any combination of demands
  - Prevents starvation by allowing surplus into “heaviest” queues first
    - Avoids both single-core and multicore queues competing for the same resources
  - Can shift balance between entire sub-trees in hierarchy (e.g. analysis <--> production)

Results



Quotas adjusted between production and mp8 groups

Surplus flag set to true for group mp8

Results

- Dips in regular production (magenta) are filled in by multicore jobs (pink)
  - Some inefficiency remains due to fragmentation
    - There is some irreducible average wait-time for 8 cores on a single machine to become free
- Results look promising, will even allow grid resources to backfill if all ATLAS queues drain
  - Currently impossible as Condor doesn't support preemption of dynamic slots… they are working on it.

THANK YOU!

Dynamic allocation script and images credit to my summer intern:

Mark Jensen (SUNY Stony Brook)

Questions? → <willsk@bnl.gov>