Scheduling multicore workload on shared multipurpose clusters

A. Perez-Calero Yzquierdo, J.Flix Molina, C. Acosta Silva (PIC)

A. Forti (Manchester)

R. Starink, *Jeff Templon* (Nikhef)

2015.04.14

- OUTLINE:
- Intro scheduling theory
- Background of multicore scheduling
- How mcfloat works
- Performance of mcfloat
- How to tune & coupling with theory

Scheduling Theory 101

- A/S/C
  - A arrival time distribution
  - S size of jobs
  - C number of servers (ie worker nodes)
- Some are solved eg M/D/k
  - Poisson arrival time dist (M = Markov), deterministic size of jobs (D), k WNs
- G/G/k is *not solved*
- Best you can do is statistics

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Statistics reminder

What wins in statistics, are configurations (ways to partition a system) in which there are *many* ways (permutations) to achieve the desired result.

Think about rolling 100 dice … answer will nearly always be between 300 and 400, even though a roll of "100" is just as likely as any *particular* roll of 350. Just many, many more ways to get 350 than to get 100.

Try "distribution 100 dice" on Wolfram Alpha
Try other numbers than 100 …

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Multicore in Practice: Boundary Conditions

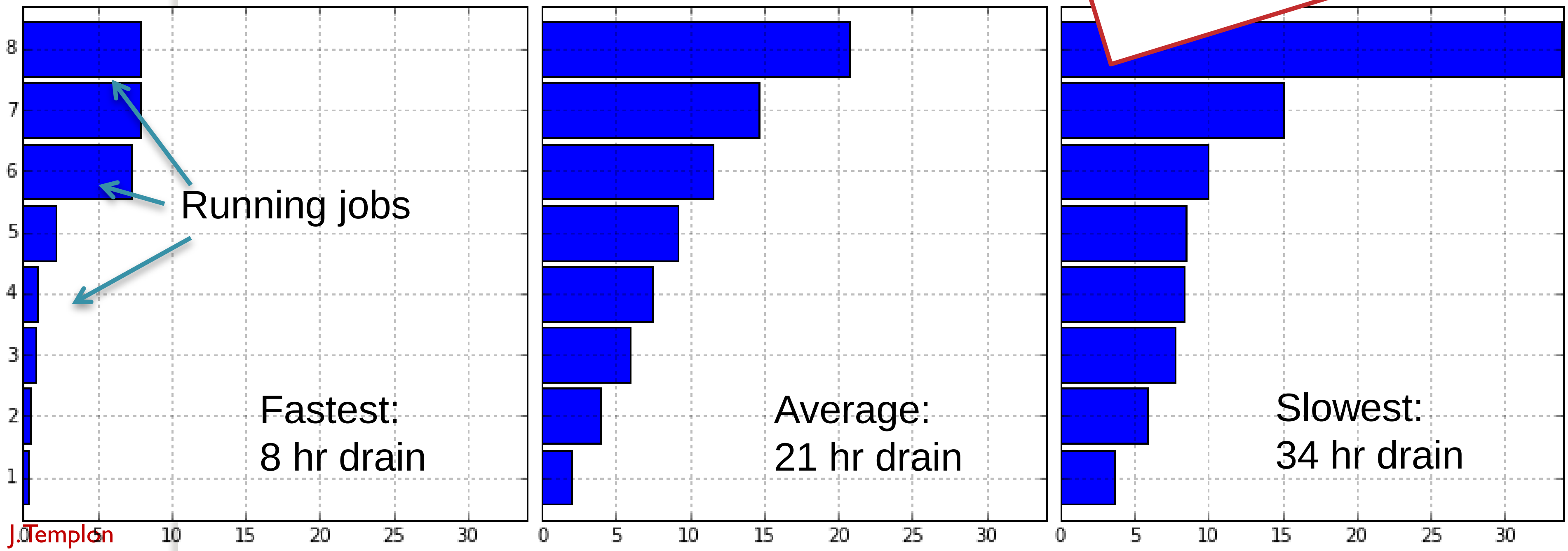
- Important customers want multicore
- No easy solution @ Nikhef:
    - Usually >7 groups active (also important)
    - Almost never empty (99% used)
    - Funding is for shared facility: cannot dedicate slots statically
- Typical jobs on system don't allow scheduler to progress on multicore problem (ie no backfilling)

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Doing it without scheduler: draining

Can't do better than statistics anyway

8-core nodes



Job completion time (hours into the future)

Real data, production cluster – 8 groups active, Snapshot 31 march 2015 15:42:02 CEST

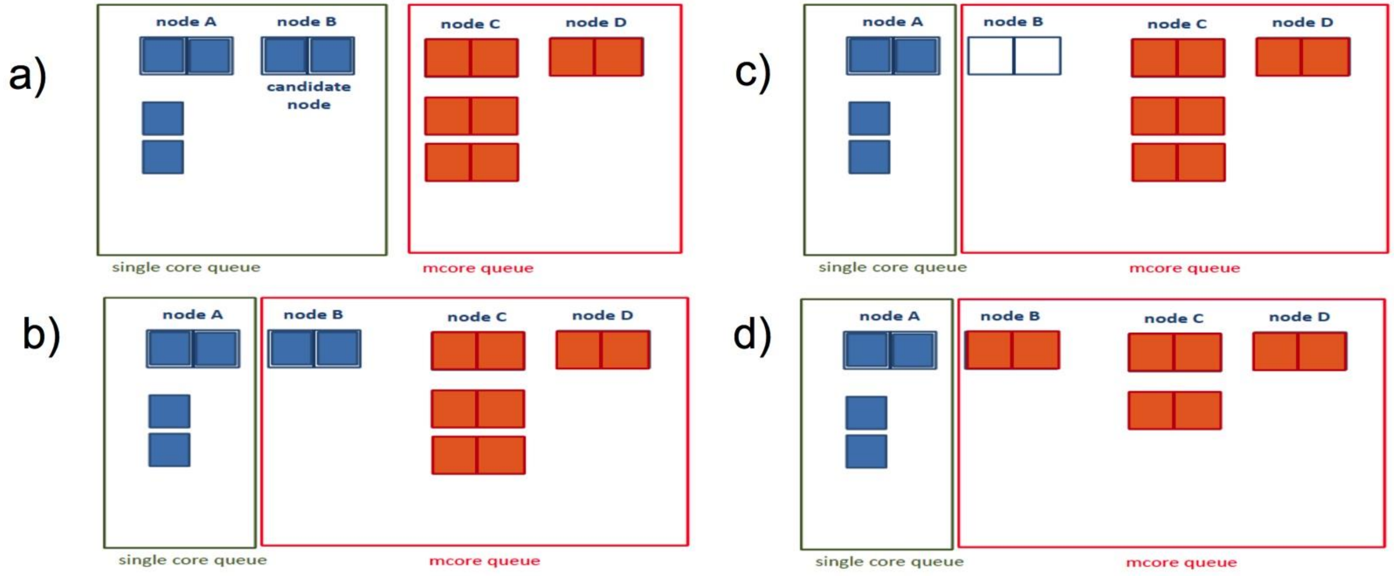
J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Multicore slot conservation with mcfloat

Multicore slot conservation can be achieved with **dynamic partitioning** of site resources: implemented by **mcfloat tool for Torque/Maui sites**

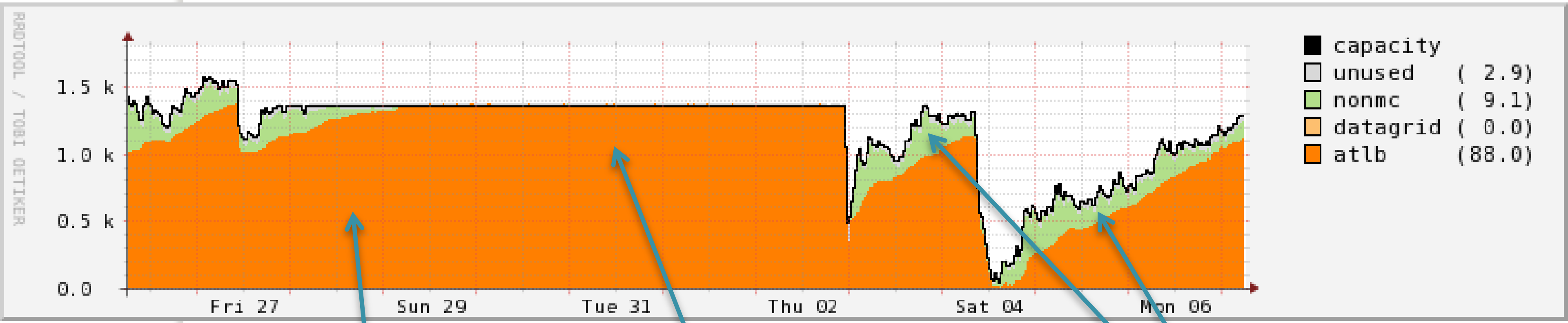
Principles:

- Moving WNs between separated pools for single and multicore jobs
- The boundary between the two partitions is adjusted dynamically to load variations:
  - **no draining is needed to support a constant multicore job load**
  - **draining in a very controlled amount:** only a small percentage of the total number of cores in a site being drained simultaneously



A week at Nikhef

- Note "unused" here is unused fraction of multicore pool. Also note: pool includes both draining and drained nodes.



Multicore jobs

Other jobs still running on draining nodes

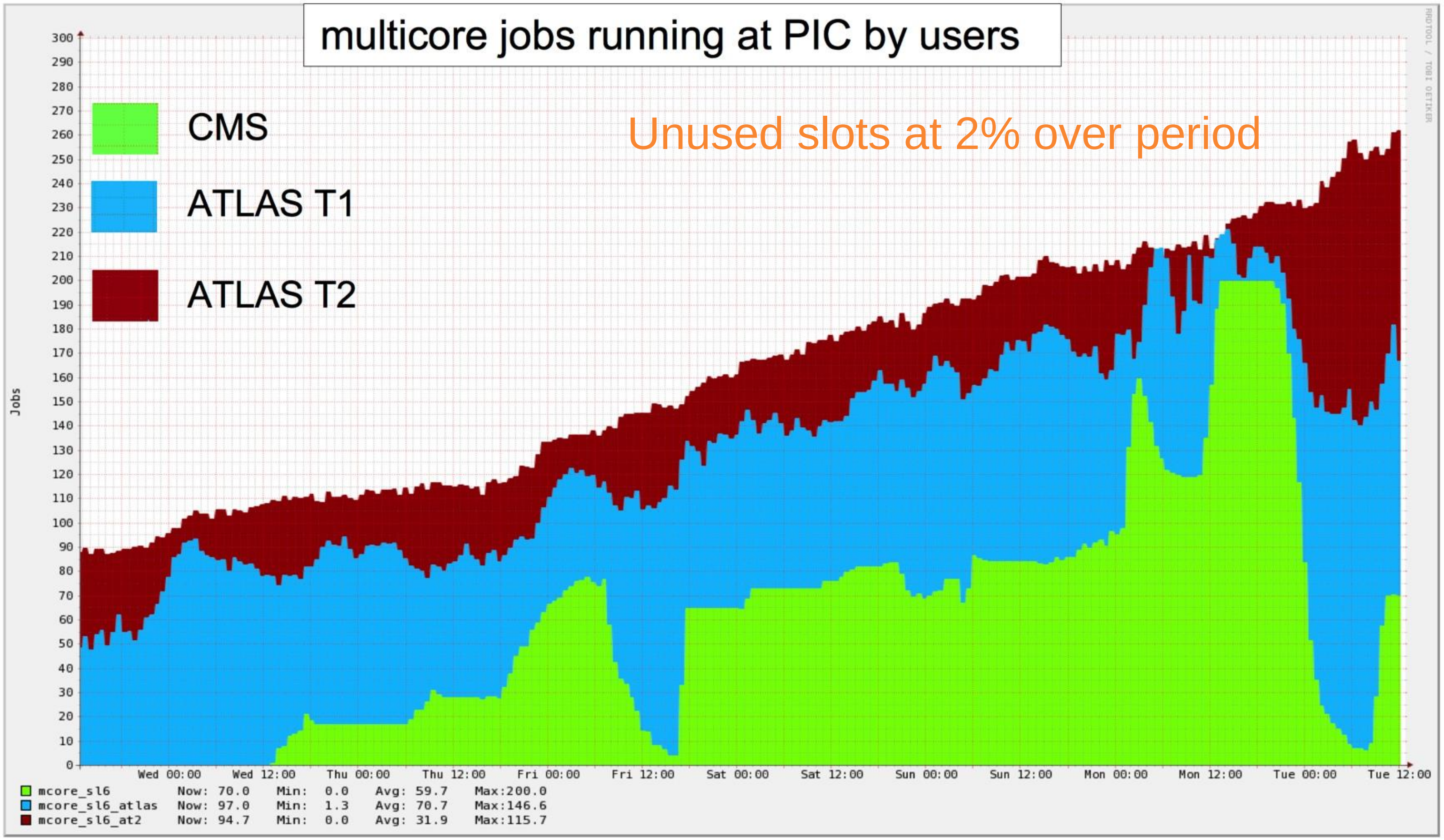
Period in which all nodes 100% drained

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Mcfloat performance at PIC

A week of multicore jobs running at PIC

- Continuous ramp up of multicore resources being offered by the site as a response to user pressure
- Robust with respect to indivudual users bursty job submission patterns

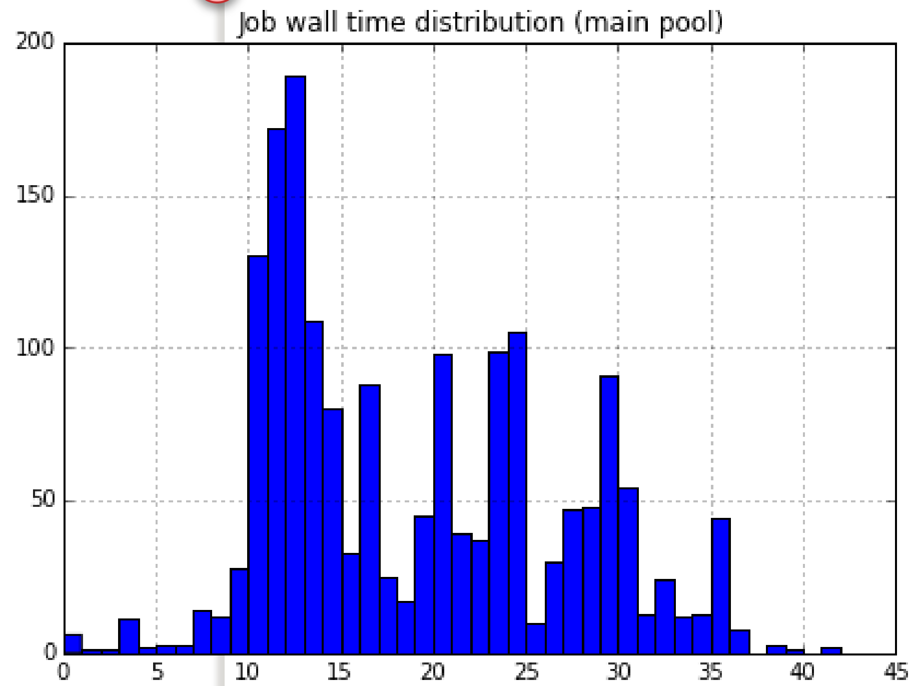


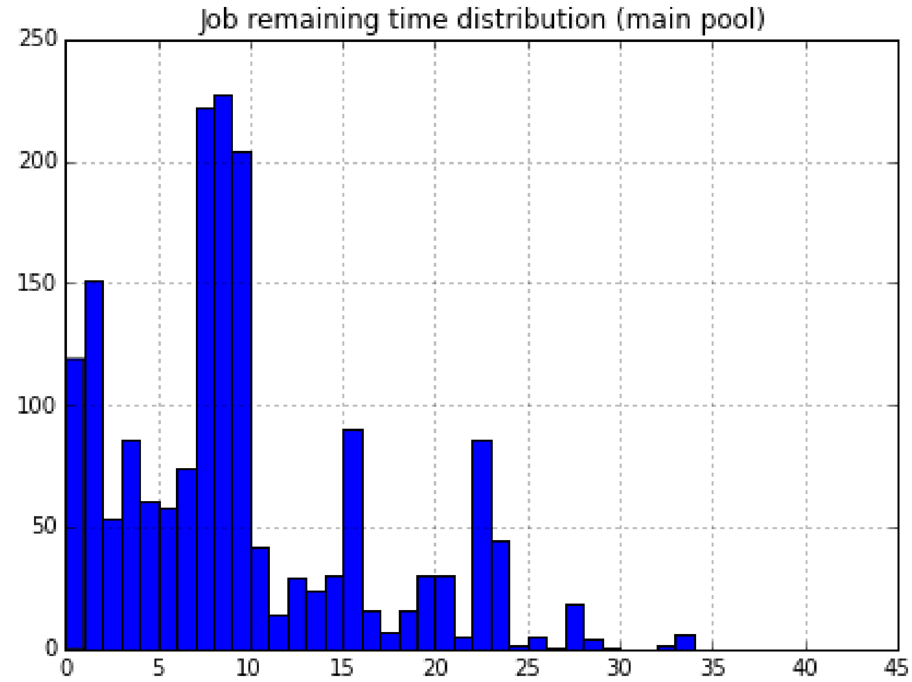
Dynamic partition works

- Mcfloat works
- Rest: How to optimize
  - Operation of dynamic multicore pool
  - Acquisition of running multicore slots

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

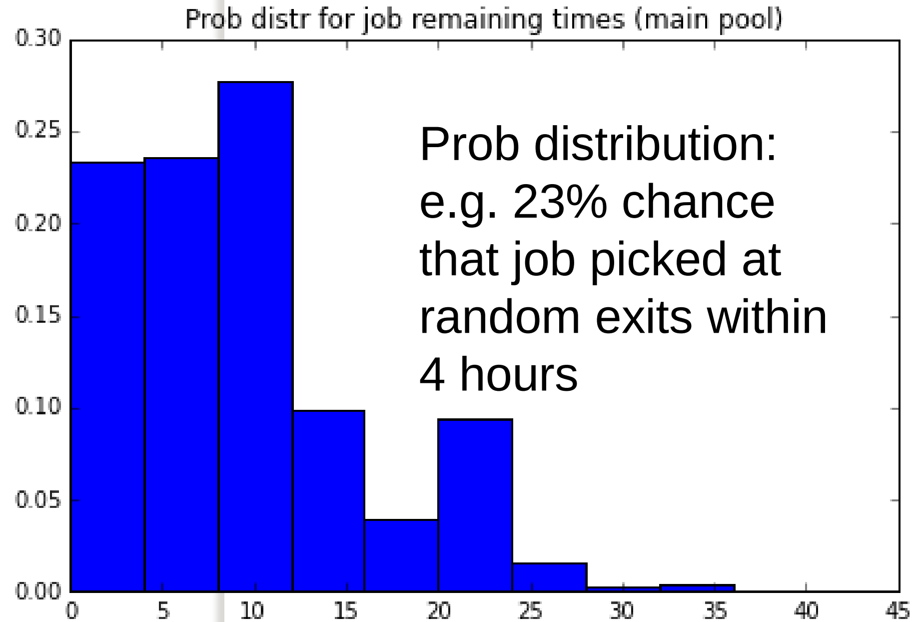
Examine job distributions: what do they tell us?

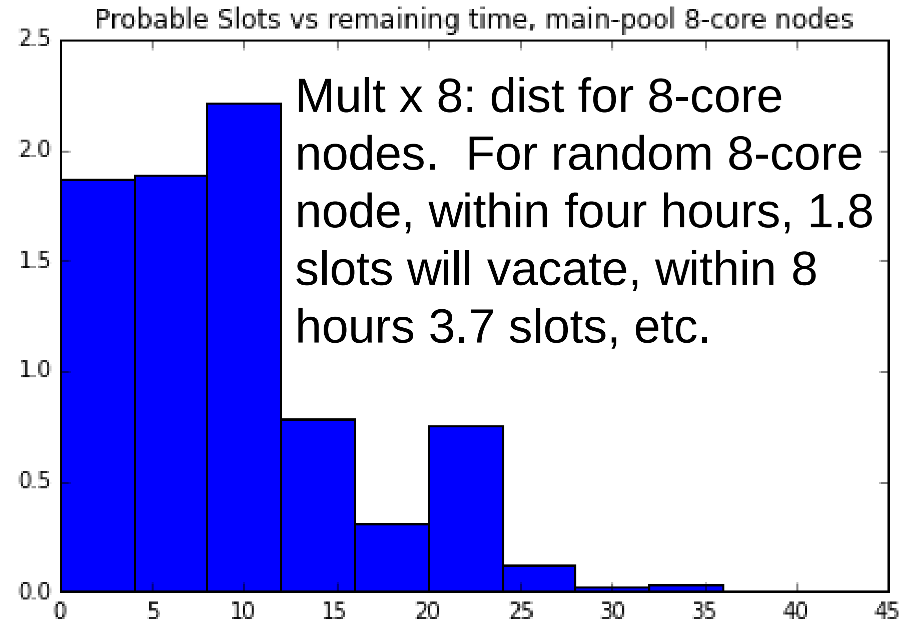


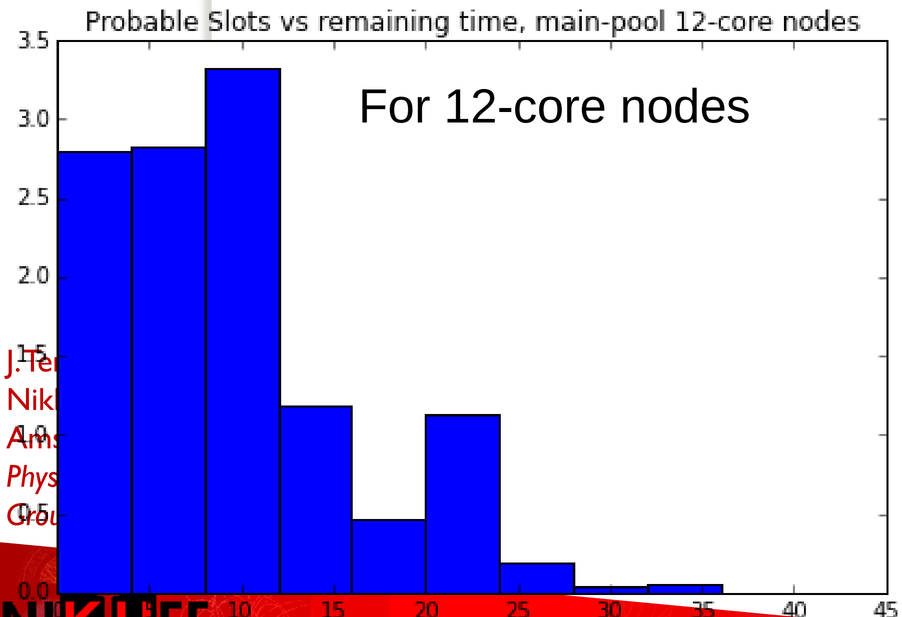


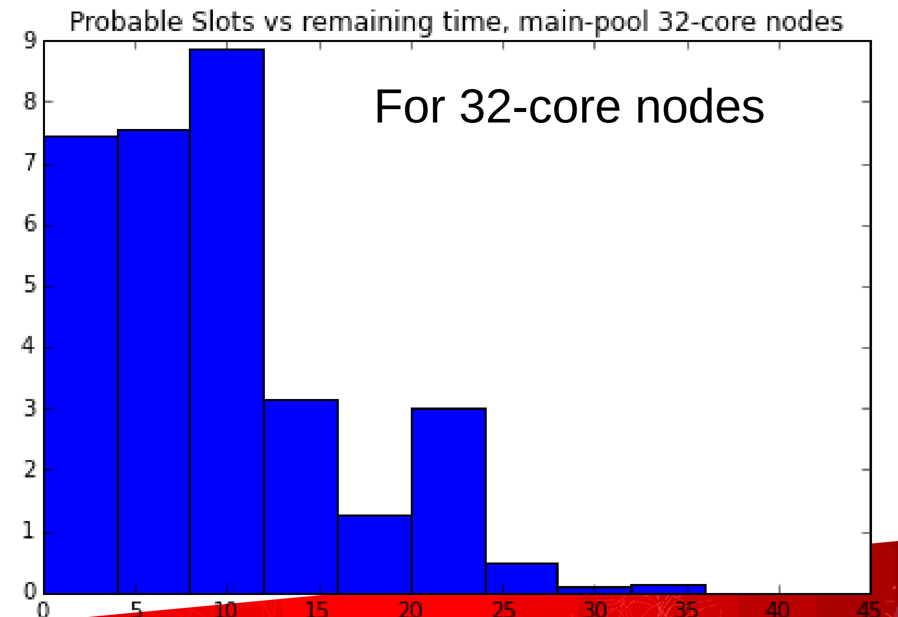
J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Time distributions on shared cluster – 8 groups active
Snapshot 31 march 2015 15:42:02 CEST


Prob distr for job remaining times (main pool)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0 5 10 15 20 25 30 35 40 45
Prob distribution: e.g. 23% chance that job picked at random exits within 4 hours


Probable Slots vs remaining time, main-pool 8-core nodes
2.5
2.0
1.5
1.0
0.5
0.0
0 5 10 15 20 25 30 35 40 45
Mult x 8: dist for 8-core nodes. For random 8-core node, within four hours, 1.8 slots will vacate, within 8 hours 3.7 slots, etc.

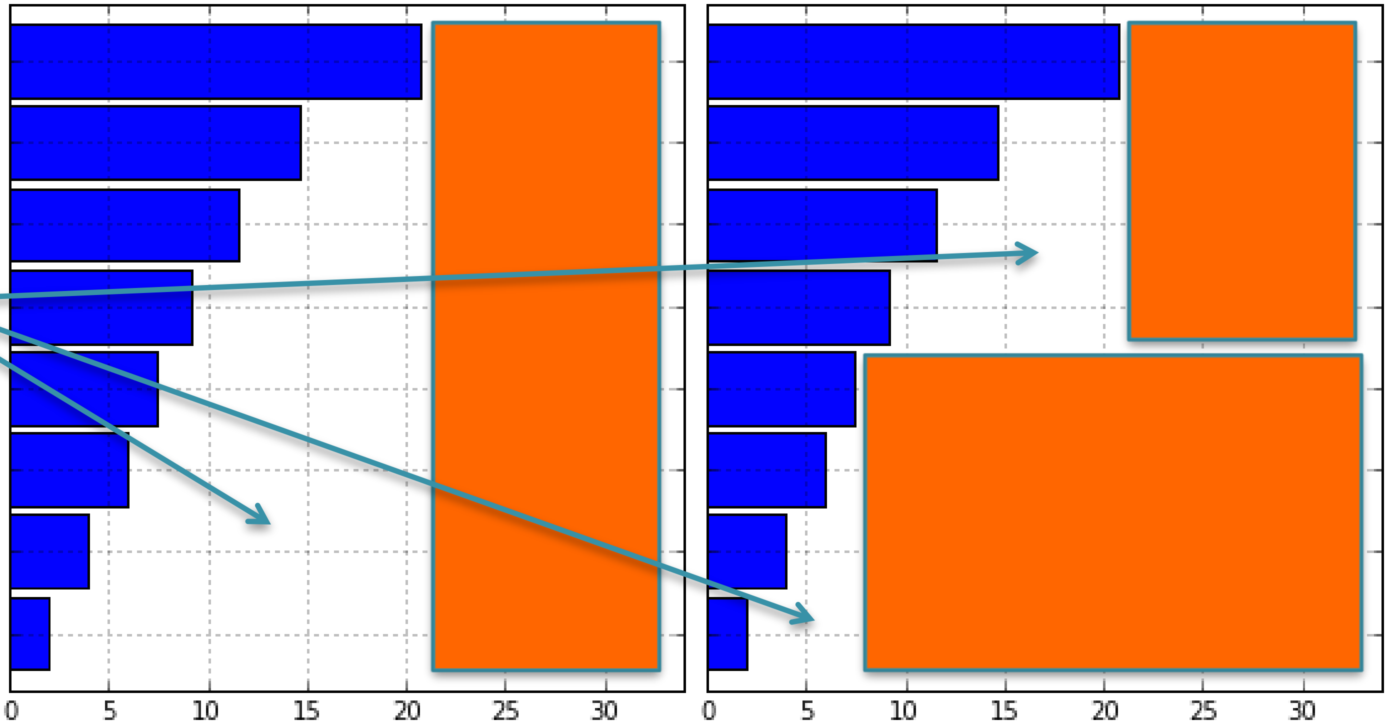

Probable Slots vs remaining time, main-pool 12-core nodes
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0 5 10 15 20 25 30 35 40 45
For 12-core nodes


Probable Slots vs remaining time, main-pool 32-core nodes
9
8
7
6
5
4
3
2
1
0
0 5 10 15 20 25 30 35 40 45
For 32-core nodes

Recall reminder from statistics beginning of talk!!!

Average time-to-start from real data

White: Wasted core hours

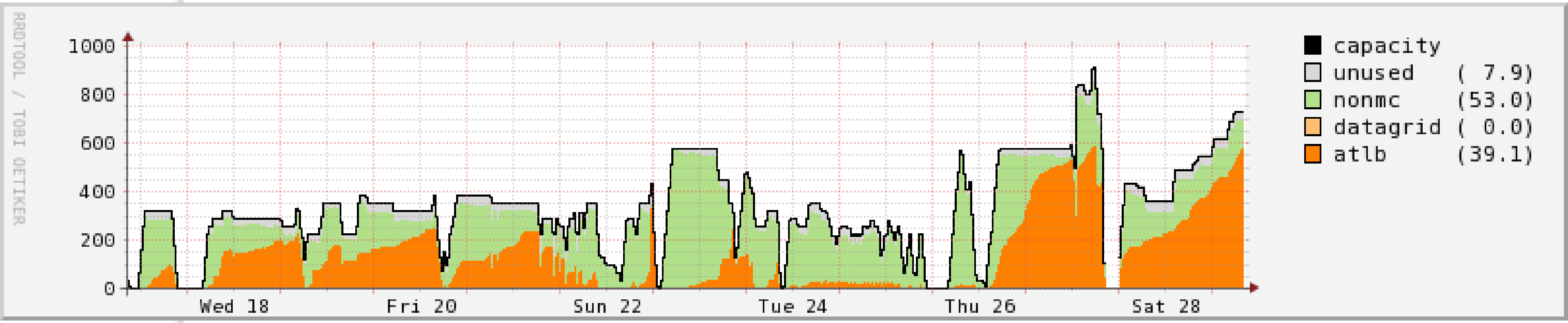


4-core jobs win twice: Start faster and waste fewer slots

Job size	t (hr) 8c nod	time 12c nod	time 32c nod
8 cores	21	10	4
4 cores	7	5	2

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

The switch at Nikhef



8 core jobs

4 core jobs

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Conclusions

- Dynamic partitioning with simple algorithm works well in practice
- Validated at PIC and Manchester
  - Since used at other sites as reported at HEPiX
- For fast growth & little waste in your pool, employ combinatorics:
  - Run jobs with as few cores as possible
  - More cores per node is better

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

Background info & exercises (if you like)

- Think about doodle poll difficulty:
  8 people vs 4 of 8
- Visit the wikipedia page on queueing theory
- Visit the Wolfram alpha dice roll page …
  try 1, 2, 5, 100 dice and see what happens to the probability distribution

J. Templon
Nikhef
Amsterdam
Physics Data Processing Group

NIKHEF