Dynamic partitioning for Multi-core jobs at INFN Tier–1

Stefano Dal Pra

INFN–T1
stefano.dalpra@cnaf.infn.it

Multicore Task Force meeting, September 16th 2014

INFN–T1 Farm

INFN–T1 Farm

Resources

- 6 × CREAM–CE, LSF 7.06, ~14K slots, 165 KHS06

Usage

- 4 × LHC + ~20 minor experiments.
  - Almost no dedicated WNs
  - single–core jobs
  - fairshare scheduling
- Saturation (having free slots is usually a symptom of problems)

Multicore

Multi–core early tests

- Ten dedicated 8-core WN from Jan 2014
- Job submission to a dedicated mcore queue
- enabled to CMS and ATLAS

Dynamic partitioning

- Activated on August 1^{st}, 2014
- Enabled on a set of racks, three WN flavours
- 8 core; 12+HT $\rightarrow$ 16 slot; 24 core, $> 3GB$ RAM
- Current status: production, no babysitting
  - still *raw* on some component
- tunable by configuration file

Features – goals

Features

- Dedicates WNs to (only) multicore jobs on need.
- Reclaim them back to ordinary single-core jobs when they are still free after a given time.
- Deployed and working with 8-core jobs only

Goals

- Reduce to a minimum the number of unused cores due to WN draining phases or underutilization.
- DO NOT stress the batch system (reduce interactions at the lightest possible ones)

Implementation

Components

- Three python scripts (esub, elim, mcdir)
- two `C` programs (using `lsf/lsbatch.h` api)
- one configuration file

Input

- WNs status (num cores, num slots, jobs)
- Pending and running job list with resource request

Output

- A JSON status file with three WN lists:
  - D: Draining, P: Purged, R: mcore–full
- Accounting logfile, debugging logfile

LSF Implementation
External Load Index script

`elim.mcore`

Runs an endless loop on each WN. Prints to `stdout` its mcore status (1 or 0) every 60 secs, which is then collected by the LSF master.

- At start, the node takes its initial status by looking for its hostname on a mcore members list
  - If NOT found, then it sets `mcore` $\leftarrow 0$ without any further check.
  - Else it checks for its hostname in the JSON statusfile produced by `mcdir` and set–or–updates its status as a result.

LSF Implementation
External SUBmission script

Alter submission parameter: `esub.mcore`

Resource requests are modifyed on each submitted job.
Inspect `LSB_SUB_RES_REQ` and `LSB_SUB_NUM_PROCESSORS` to identify multicore jobs.
When like: `bsub -n 8 -R "select span[ptile=8]"`

- Then: `LSB_SUB_RES_REQ` ← `"select [mcore==1] span[ptile=8]"`
- Else: `LSB_SUB_RES_REQ` ← `"select [mcore!=1]"`

- This does NOT depend on the queue name. Only on the submission parameters.
- Need to deal with CPU & MEM queue limits. Monitoring more difficult.

Implementation
The MCORE Director

It manages node transitions to–from the *mcore* partition.

Collected data

- **Configuration info:** `hostgroups, max_hostdrain, max_emptyslots, max_empty_ratio...`
- **dynamic info:** WNs status (num cores, num slots, running jobs), Pending and running job list.
- dynamic data are Updated every 6 minutes

WN subset management

WNs are partitioned on four logical subsets.

- M, P: nodes having mcore status set to 0
- D, R: nodes having mcore status set to 1

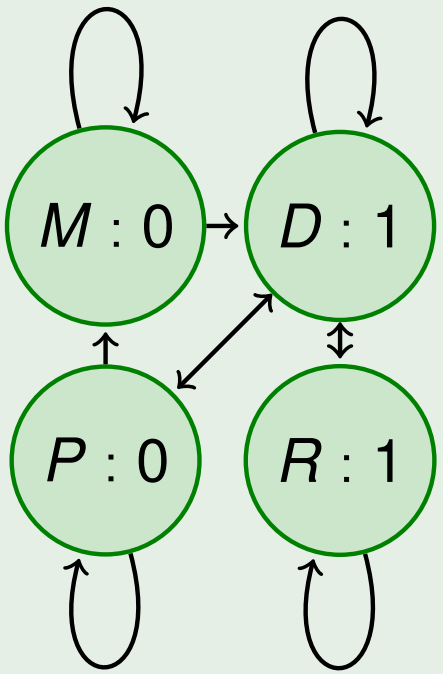
Transitions
moving WNs to–from the partition

Mcore status

WNs are moved across the following sets:

- *M*: available for multicore
- *D*: assigned to multicore
- *R*: running only multicore
- *P*: purged from multicore

Mcore Status Transition Map



Dynamic of the partition

1. At $T = 0$, all WNs are w_i in the set $M = \{w_1, \ldots, w_N\}$
2. When $Q_m > 0$ multicore jobs are queued, k WN are moved from M to $D = \{w_1, \ldots, w_k\}$ by the director.
3. When a node is full of multicore, it is moved from D to R.
4. When a node $w_i \in D$ has free room for a multicore and no jobs starts there after a timeout, it is moved from D to P.
5. When more multicore nodes are needed, they are moved from P and M to D.
6. The elim script on each node w_i updates its mcore status:

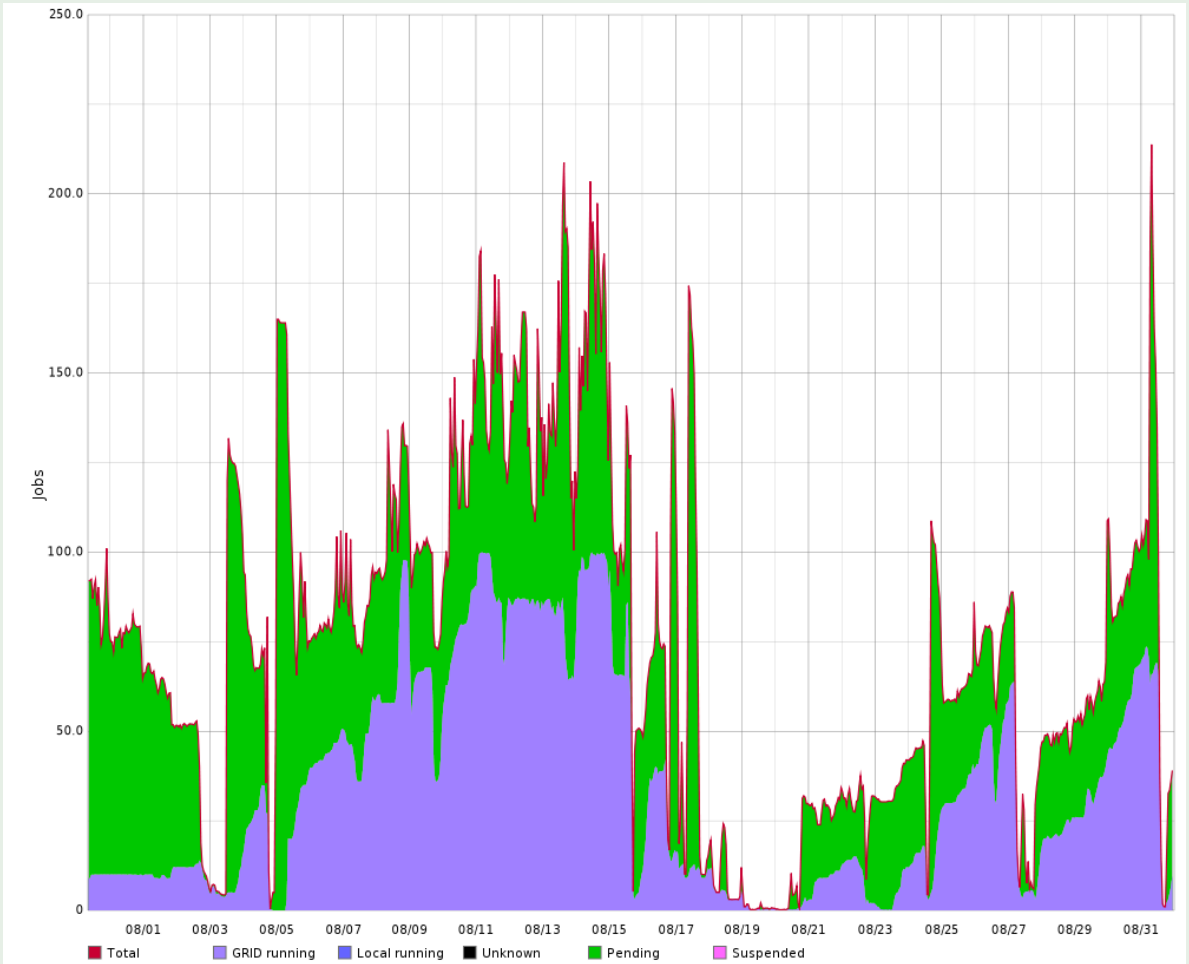
$$mcore(w_i) = \begin{cases} 1 & if\ w_i \in D \cup R \\ 0 & if\ w_i \in M \cup P \end{cases}$$

Configuration file (JSON syntax)

```
"mode":"auto", "mcore_libpath": ".",
"mc_runjobs_fn": "mc_runjobs.txt",
"mc_pendjobs_fn": "mc_pendjobs.txt",
"infodir": "info-dynamic-lsf",
"mcd_json": "mc_dir.json",
"log_fn": "mcore.log","log_dbg" : "mcore_act.log",
"hist_fn": "mcore_hist.json",
"num_cores":8, "reduce_f":16, "max_hostdrain":18,
"max_emptyslots" : 157, "max_empty_ratio" : 0.3,
"mcore_groups": ["rack20603", "rack20501"],
"lsf_hostgroups": "long_filename_bmgroups.cache",
"cachedir": "/usr/share/lsf/var/cache/",
"machinejob_nodeinfo_fn": "node_hs06_cores.txt",
"badhosts_fn": "badhosts.txt"
```

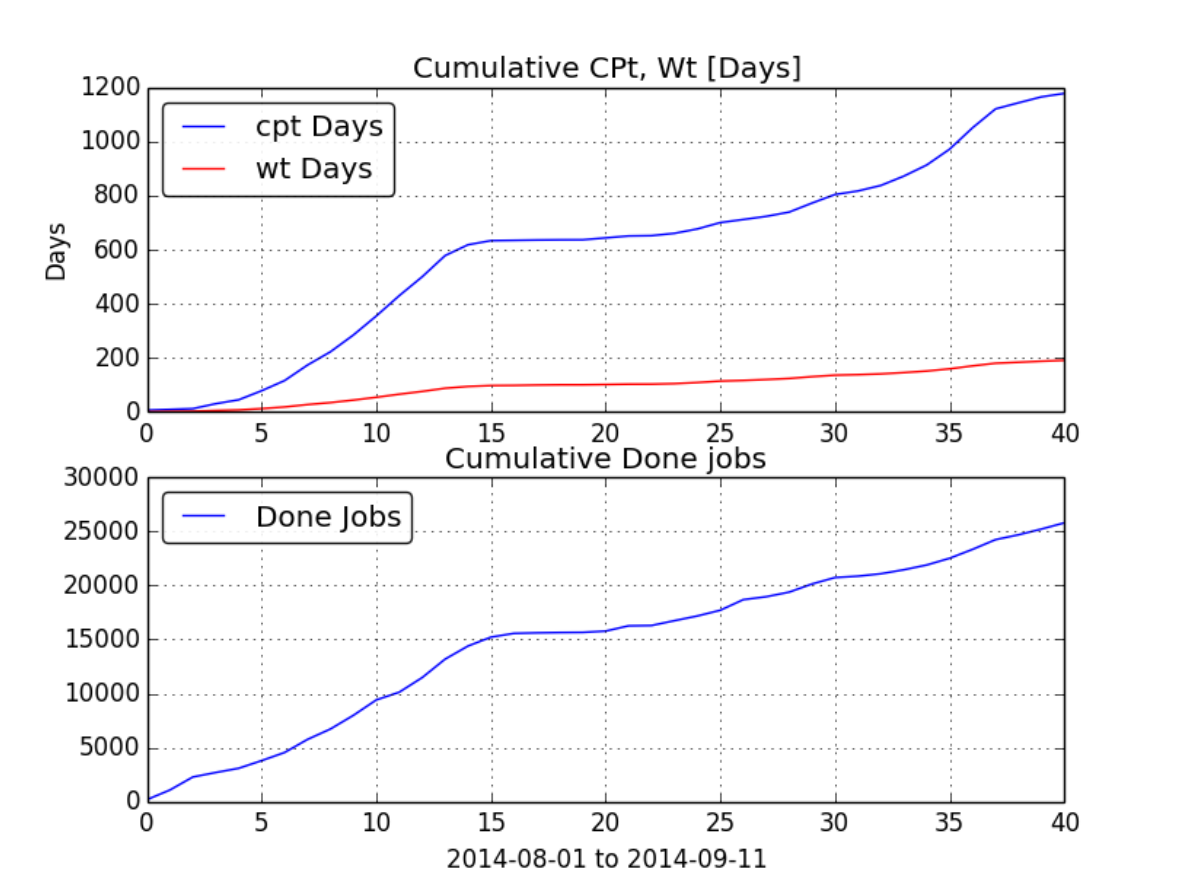
Dynamic partition
mcore queue activity

Multicore running and pending Jobs, August 2014



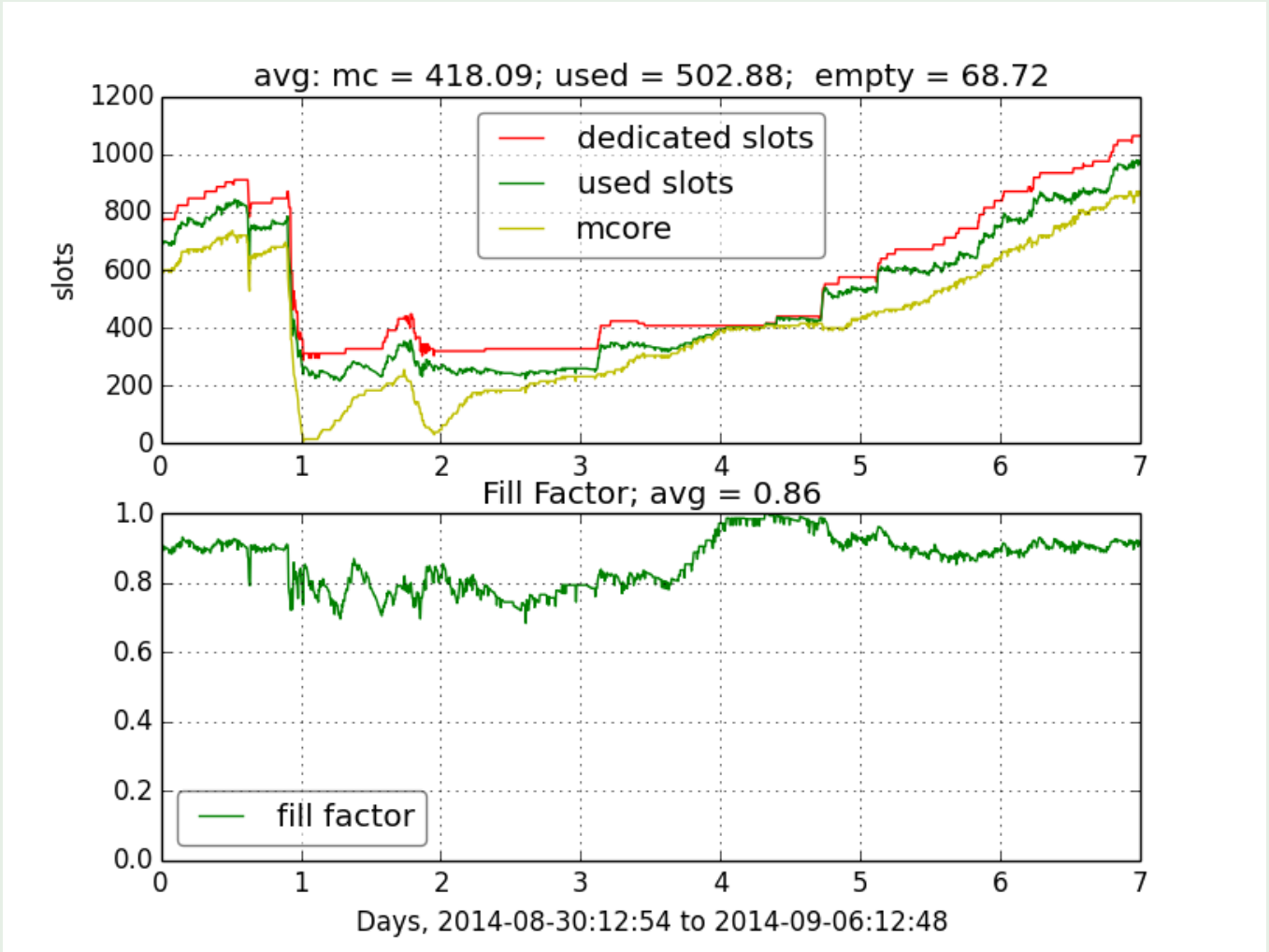
Dynamic partition
mcore queue activity

Multicore done Jobs, August 2014



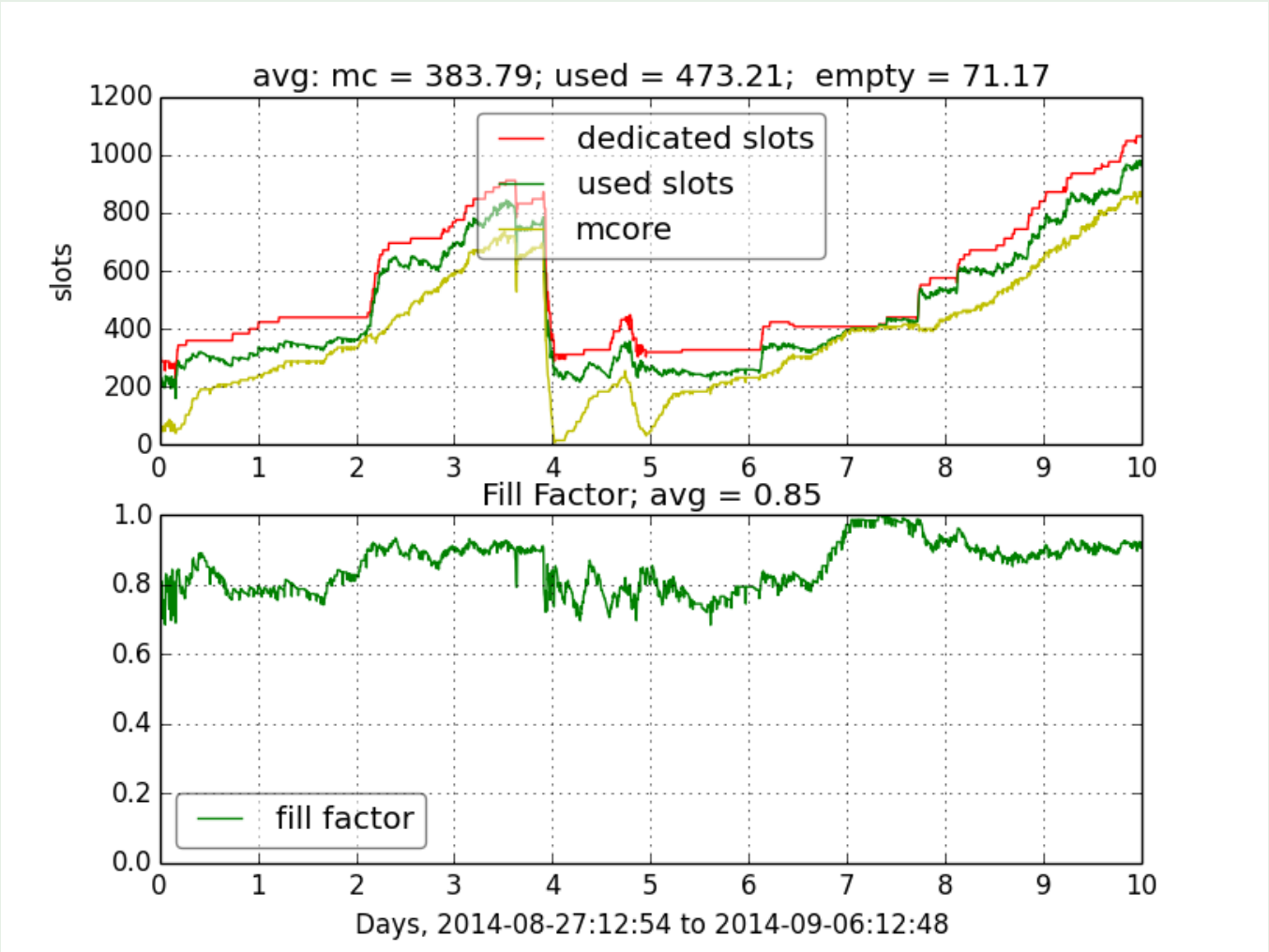
Mcore jobs

Mcore partition, 7 days



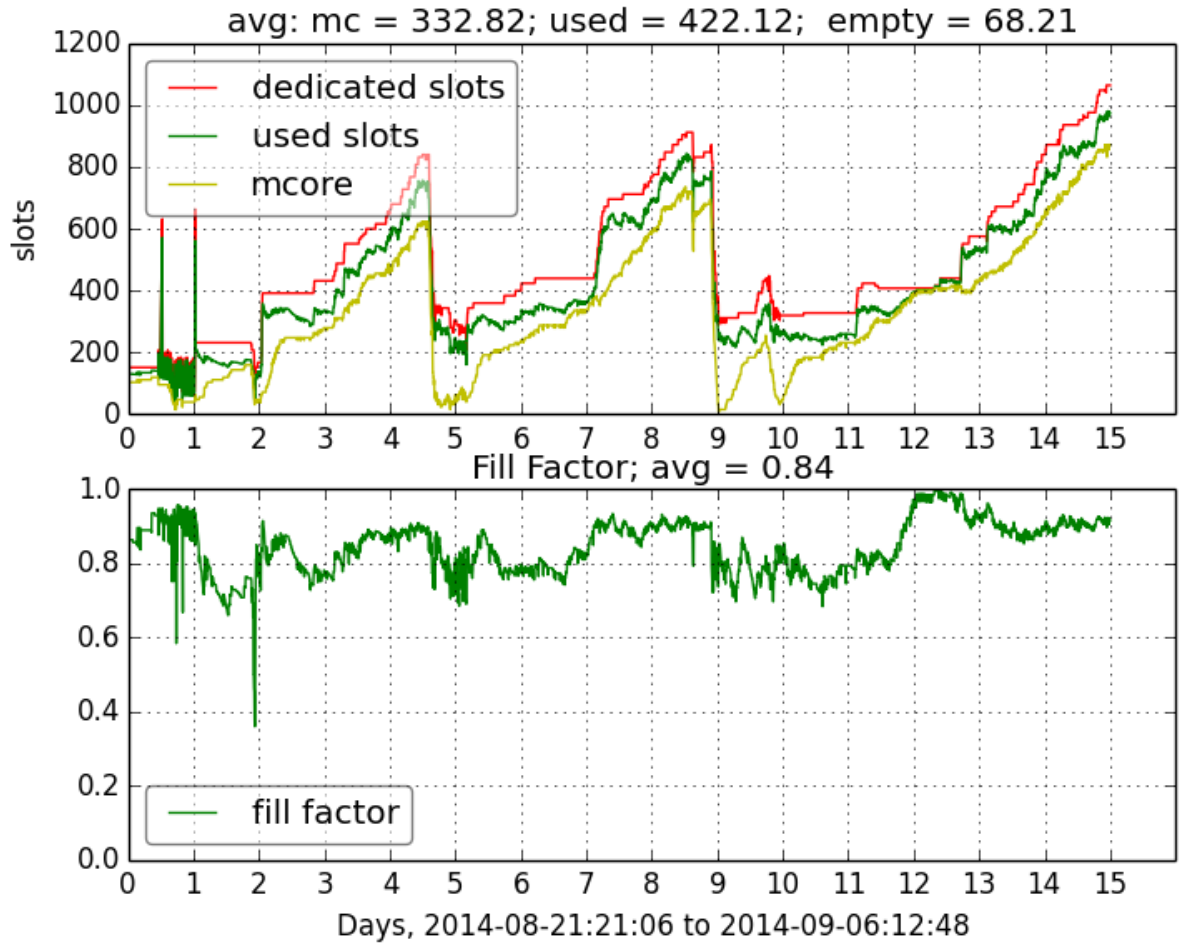
Mcore jobs

Mcore partition, 10 days



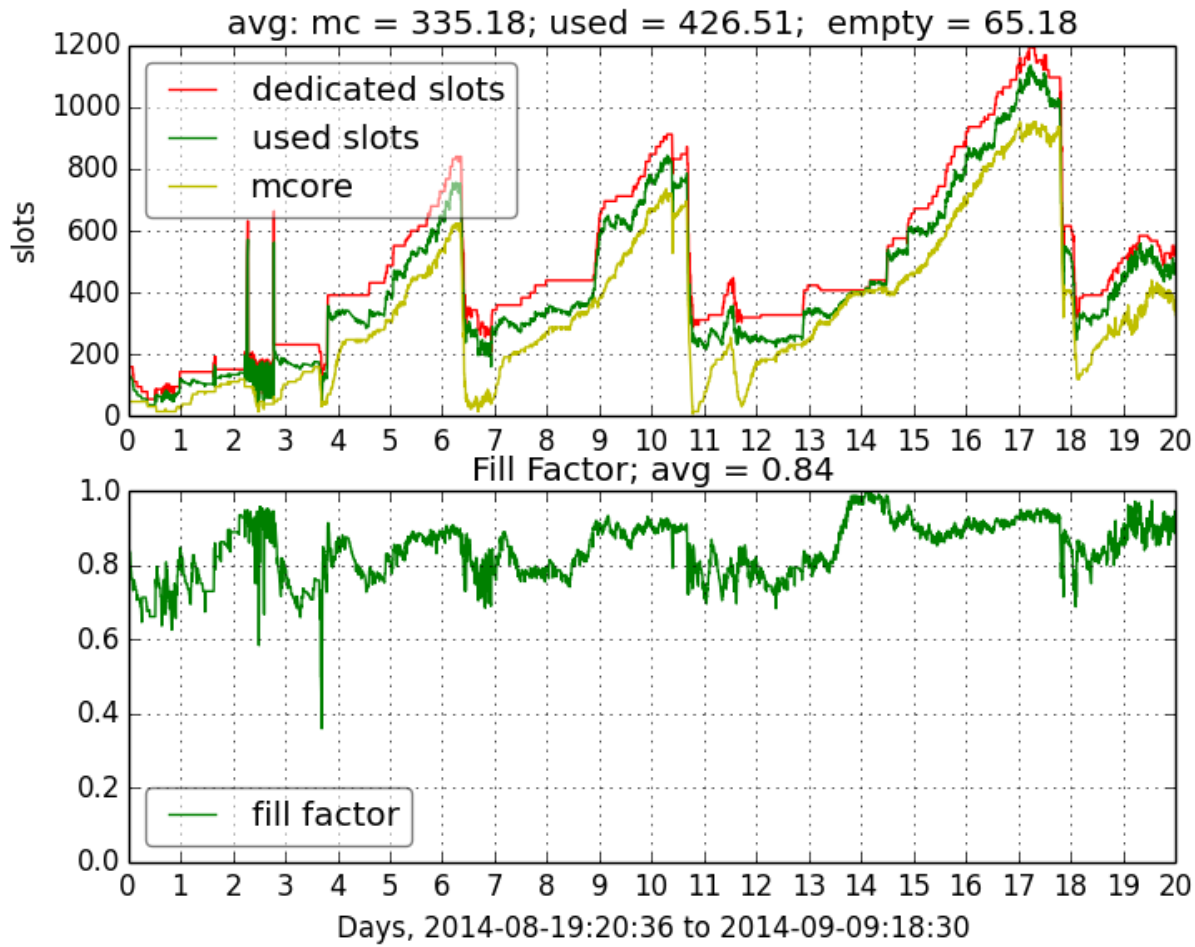
Mcore jobs

Mcore partition, 15 days



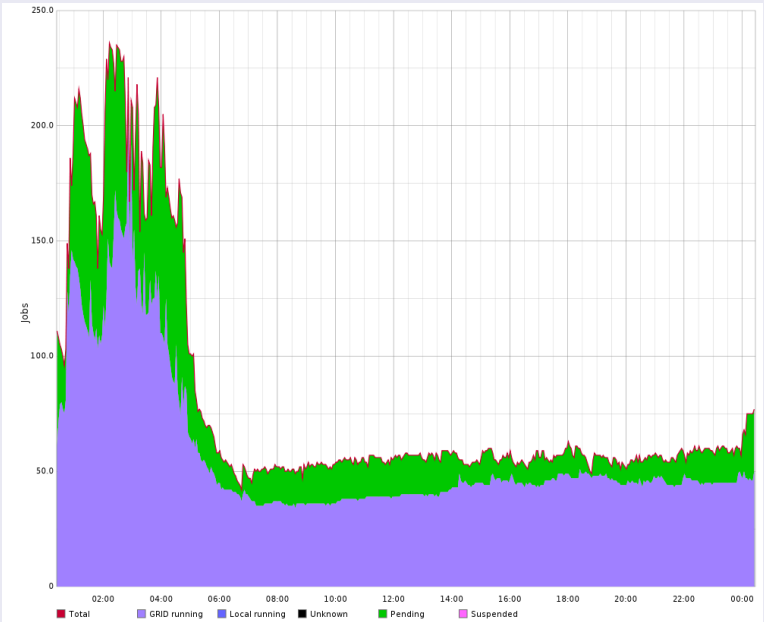
Mcore jobs

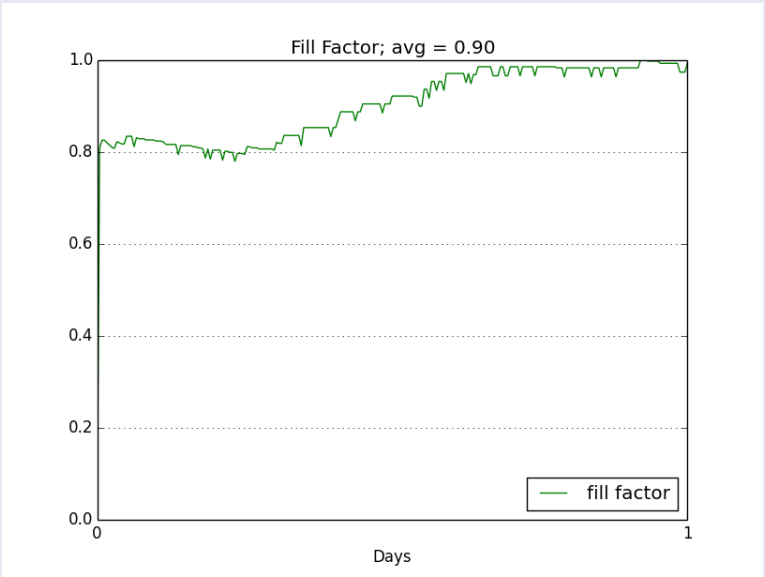
Mcore partition, 20 days



Mcore jobs

Multicore Activity and Fill Factor (24h)





Stability

With steady submission flow, $FF \rightarrow 1$.

WN comparison
by model and job type

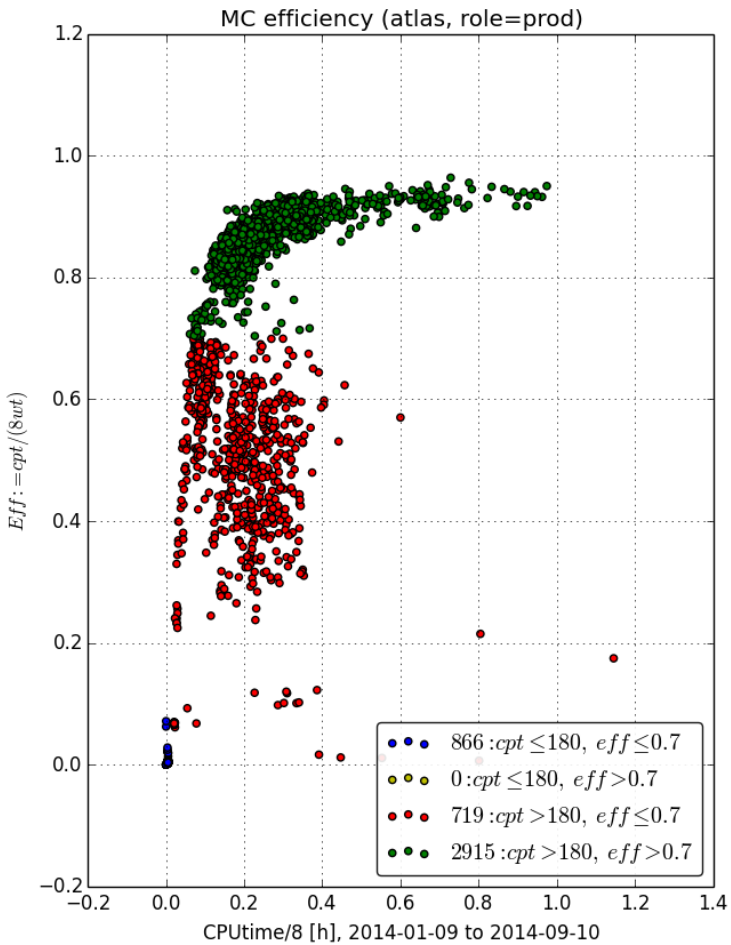
Multicore Jobs (Sep 1 to Sep 13)

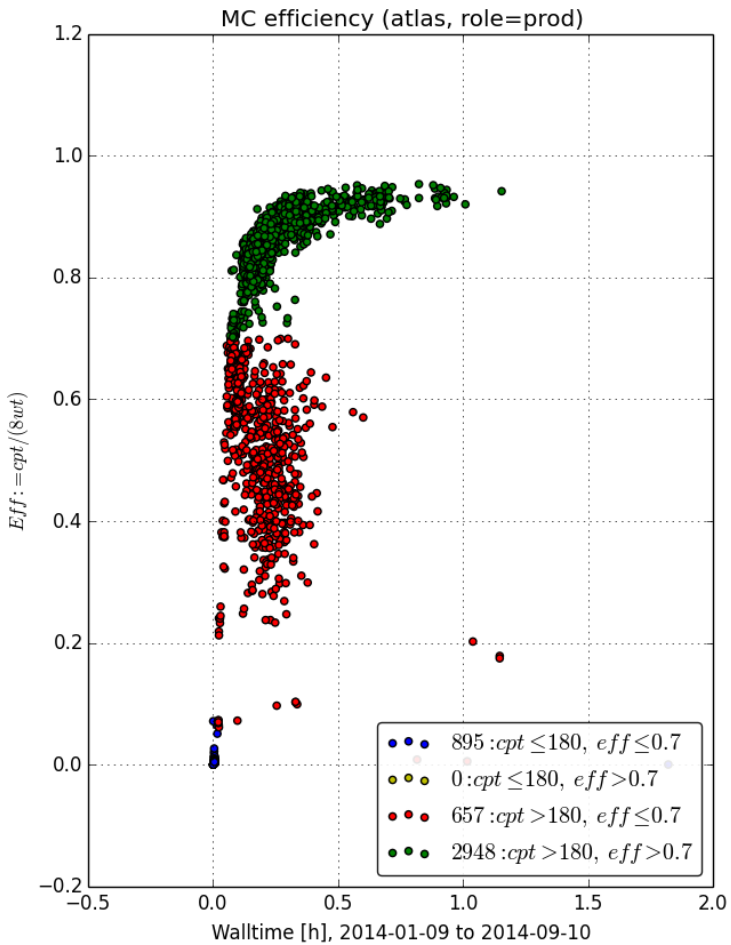
jobs	%eff	wntype	core	slot	grp	role
62	87.27	wn-205-04	8	8	atlas	prod
2863	89.28	wn-205-01	12,HT	16	atlas	prod
1973	87.96	wn-206-03	24	24	atlas	prod
147	68.83	wn-206-03	24	24	cms	prod

singlecore Jobs (Sep 1 to Sep 13)

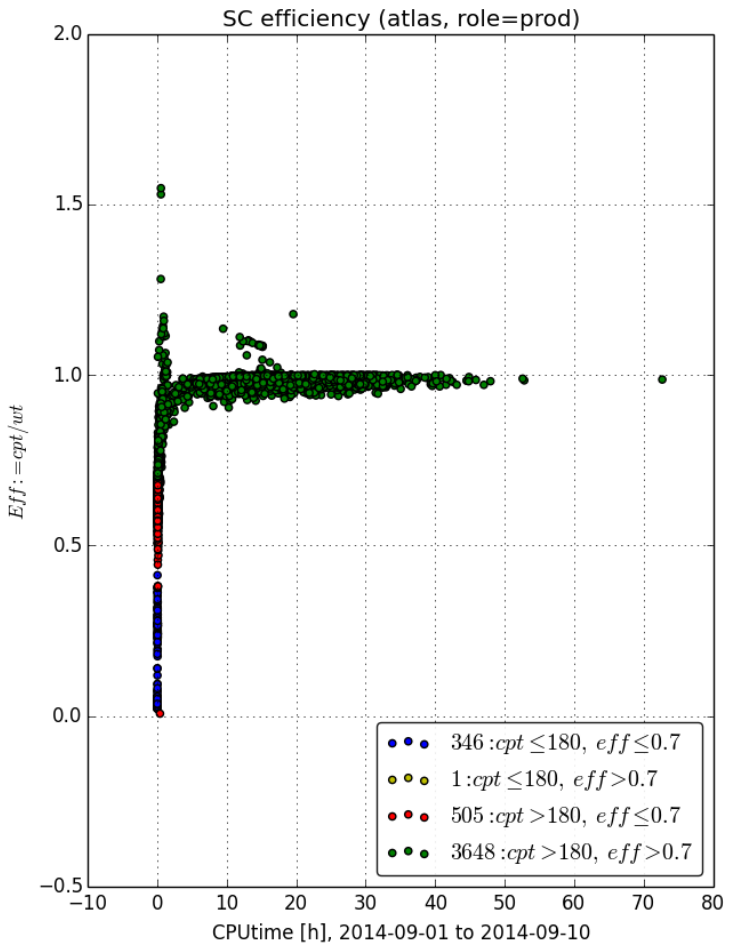
jobs	%eff	wntype	core	slot	grp	role
6842	92.85	wn-205-01	12,HT	16	atlas	prod
5636	91.76	wn-206-03	24	24	atlas	prod
4576	90.22	wn-205-01	12,HT	16	cms	prod
3781	88.61	wn-206-03	24	24	cms	prod

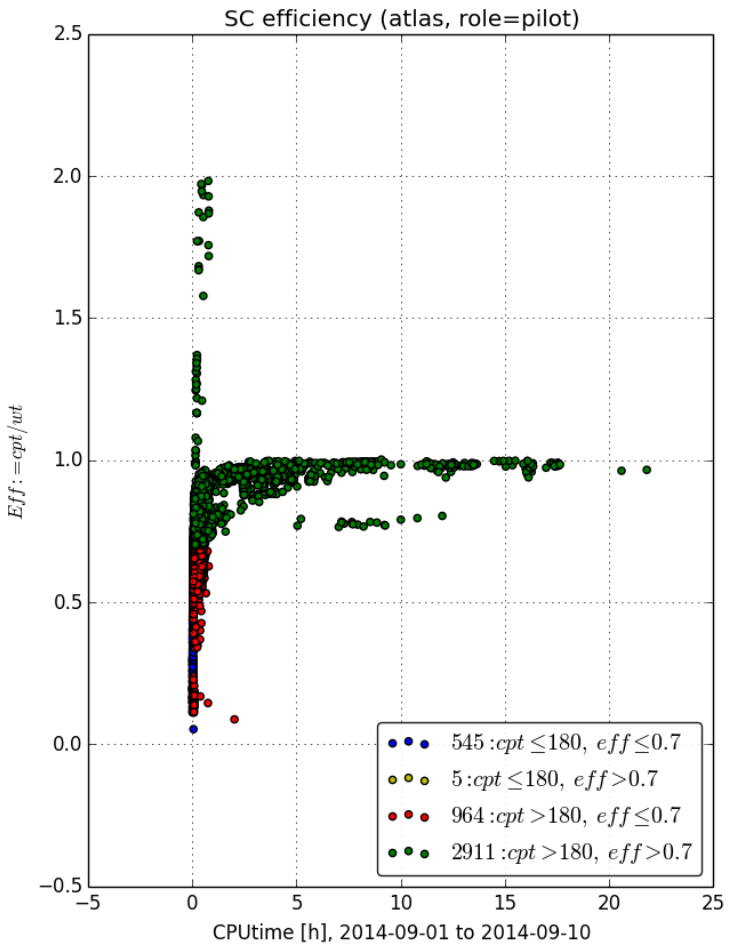
Mcore jobs (atlas)
Eff vs CPT, Wt





Single core jobs (atlas)
Eff vs CPT, prod vs pilot





Optimizations
Filling the gaps

High Memory jobs

- Requirement from atlas to provide resources for single–core jobs requiring 2× RAM than usual.
- Done: tag them as mcore, requiring 2 slots.
  - `bsub -n 2 -R "select [mcore==1] span[ptile=2]"`
- Constraint: no more than 4 HM jobs per node (8 slots)

Job Packing (Keep jobs together)

Modify elim.mcore to publish hm = # HM jobs, and esub.mcore to add an `order` clause:

- `bsub -n 2 -R "select [mcore==1] span[ptile=2] order[-hm]"`

TODO

director

- smarter host selection.
  - Director has enough information to order nodes by maximum or expected draintime.

esub, elim

- packing: keep together 2–slot jobs
- keep away longest jobs from running on mcore node candidates

backfilling

- attempt to enable jobs from *short* queues

Summary

+ A dedicated queue eases configuration and monitoring
- Negative impact on m–core efficiency from
  - Long–living single–core jobs (higher drainining time)
  - hiccups on multicore submissions (more draining needed)
+ Positive impact on m–core efficiency from *almost* steady submission rate.
+ evaluating Jobsize mix (up to 4×2–slots or 2×4–slots per node)
+ The dynamic partitioning mechanism is quite generic and may be exploited for other applications too (provisioning nodes for cloud usage)