LSF@INFN-T1

Pre-GDB on batch systems

Bologna – 11-3-2014

Stefano Dal Pra: Stefano.dalpra@cnaf.infn.it

Andrea Chierici: Andrea.chierici@cnaf.infn.it

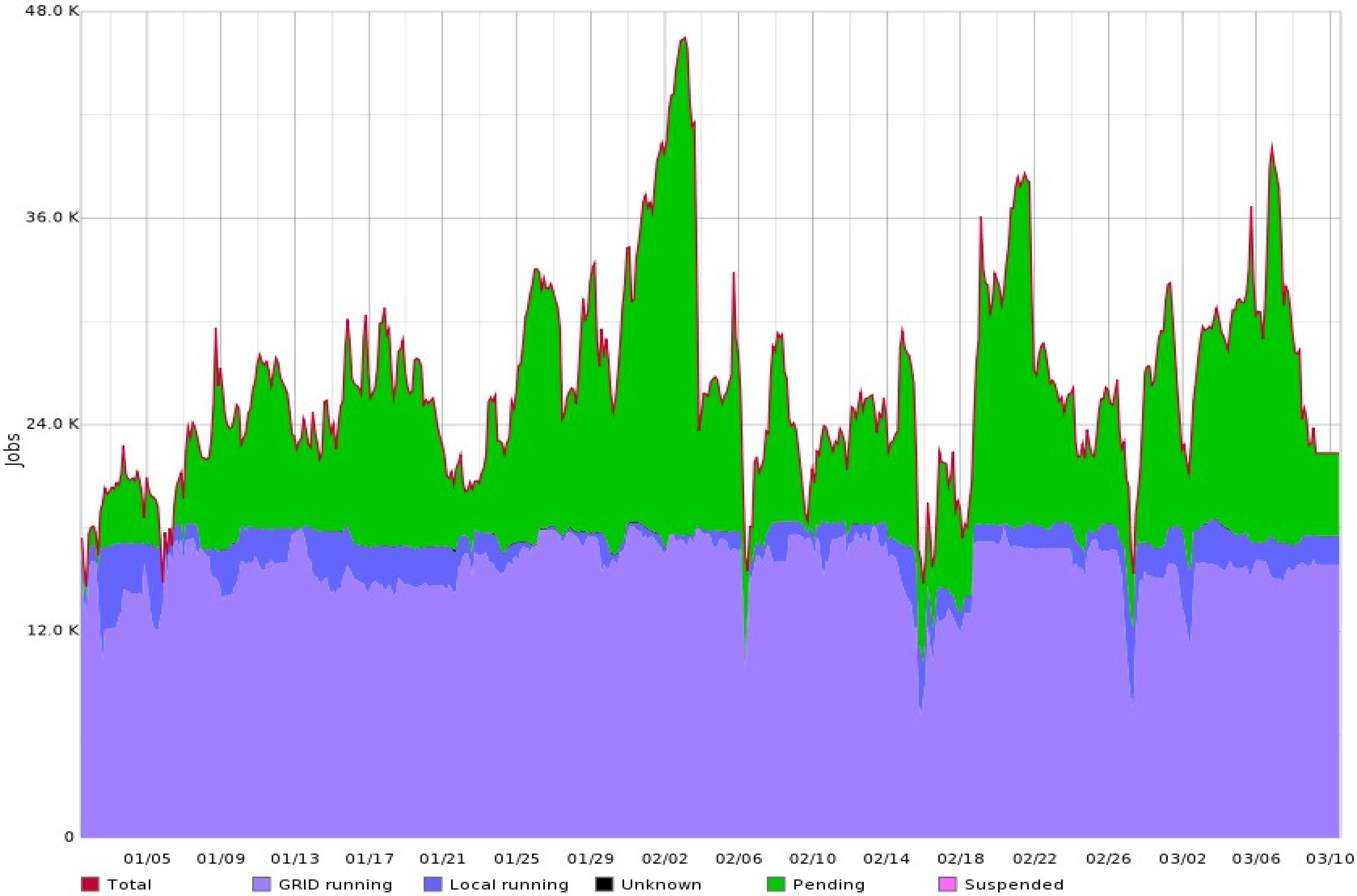
Outline

- LSF current size and operational setup
- Load, inefficiences and bottlenecks
- Special usecases
- External Load Indexes

LSF current setup

- V7.06, SLC 6.4, 1 Master, 2 shadow
- 3 Grid sites:INFN-T1, INFN-CNAF-LHCB, INFN-T3-BO
- 10 CREAM CEs (emi-3)
- 46 queues (6 LHC, 6 test - operations)
- ~ 1400 WNs, 18Kslots, 180K HS06, ~ 100K Jobs/day
- [Almost] no dedicated resources
- Fairshare, single-core jobs
  - With exceptions.

Activity 2014

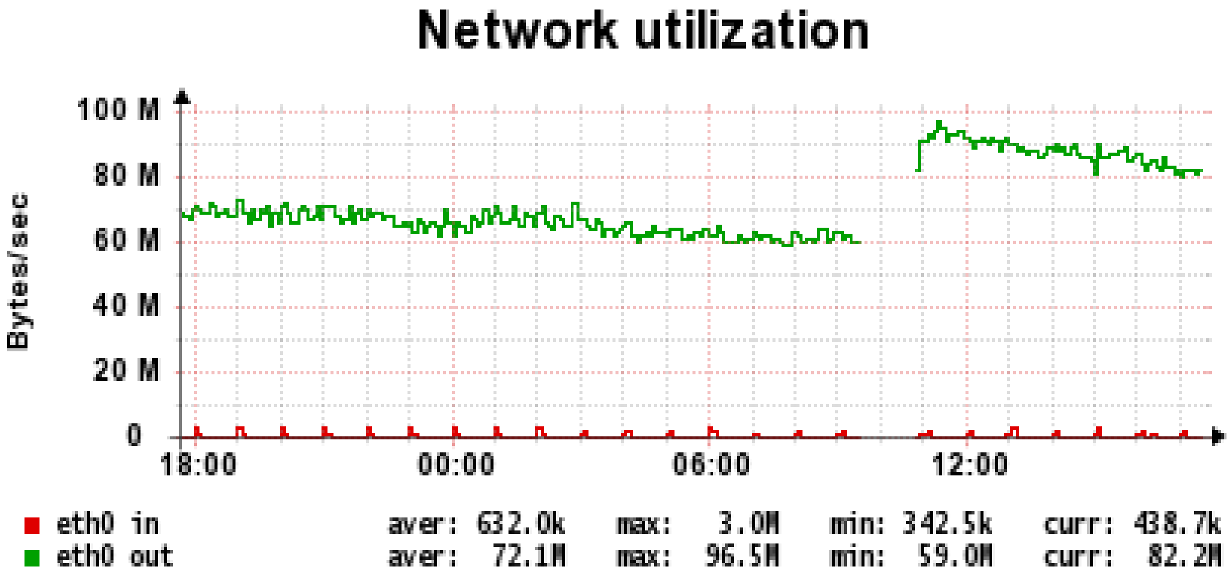


Special use cases

- 1 SL5 rack, to be phased-out soon (WNoDeS for those still needing SL5 or custom setup: cdf, babar)
- 1 WnoDes rack
  - Virtual wn management redesigned for scalability, using LSF "*External Load Indexes*"
  - Mixed mode
  - *auger_db* early adopter of the latest release
- Small HPC cluster
  - 8 host 16core 48GBRam, Nvidia GPU, 8 x K40 , 2 x K20
  - Early setup, being added to LSF soon
- mcore queue

Load Issues

- Current bottleneck, at times, has been bandwidth
  - Master answering too many requests from clients
  - Occasional net saturation experienced



Load Issues (2)

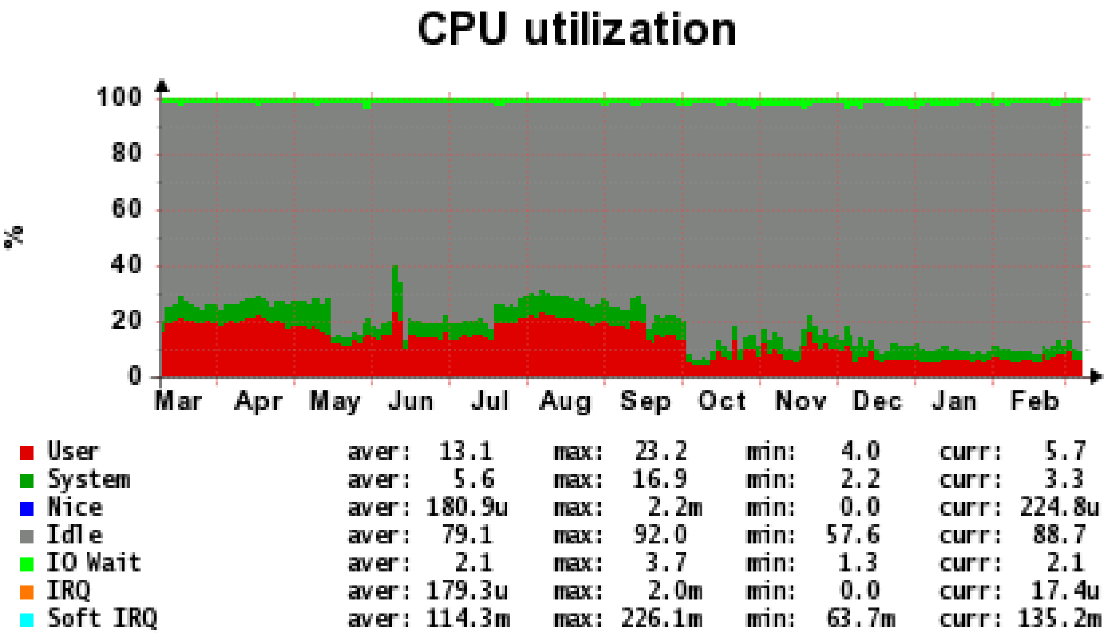
- Big bandwidth consumers are CEs (inspected with iftop)
  - Default: 1 “bjobs -l” every 120 sec
  - Reduced update rate; using btools
    - bjobsinfo → 80 x smaller output
  - 10 CEs (4 of them not managed directly) multiply the problem
    - Some sort of proxying may be helpful here

  Minor bandwidth consumers:

- A tipical Dirac pilot executes
  - 3 to 6 bqueues -l <queue> (one per payload)
  - 30 to 60 bjobs -W
  - 4000 jobs → 2 requests/sec;
  - If master is experiencing overload → answer delay → higher job WallTime

Customization (accounting)

- Removed accounting sensors from the CEs (Sep 2013)
  - Grid records (cream-blah) are matched with LSF logs on a PostgreSQL db on a standalone host, then propagated to our site-HLR (DGAS).
  - Healthy effect for the CPU load on our CEs



🟥	User	aver:	13.1	max:	23.2	min:	4.0	curr:	5.7
🟩	System	aver:	5.6	max:	16.9	min:	2.2	curr:	3.3
🟦	Nice	aver:	180.9u	max:	2.2m	min:	0.0	curr:	224.8u
⬛	Idle	aver:	79.1	max:	92.0	min:	57.6	curr:	88.7
🟩	IO Wait	aver:	2.1	max:	3.7	min:	1.3	curr:	2.1
🟧	IRQ	aver:	179.3u	max:	2.0m	min:	0.0	curr:	17.4u
🟦	Soft IRQ	aver:	114.3m	max:	226.1m	min:	63.7m	curr:	135.2m

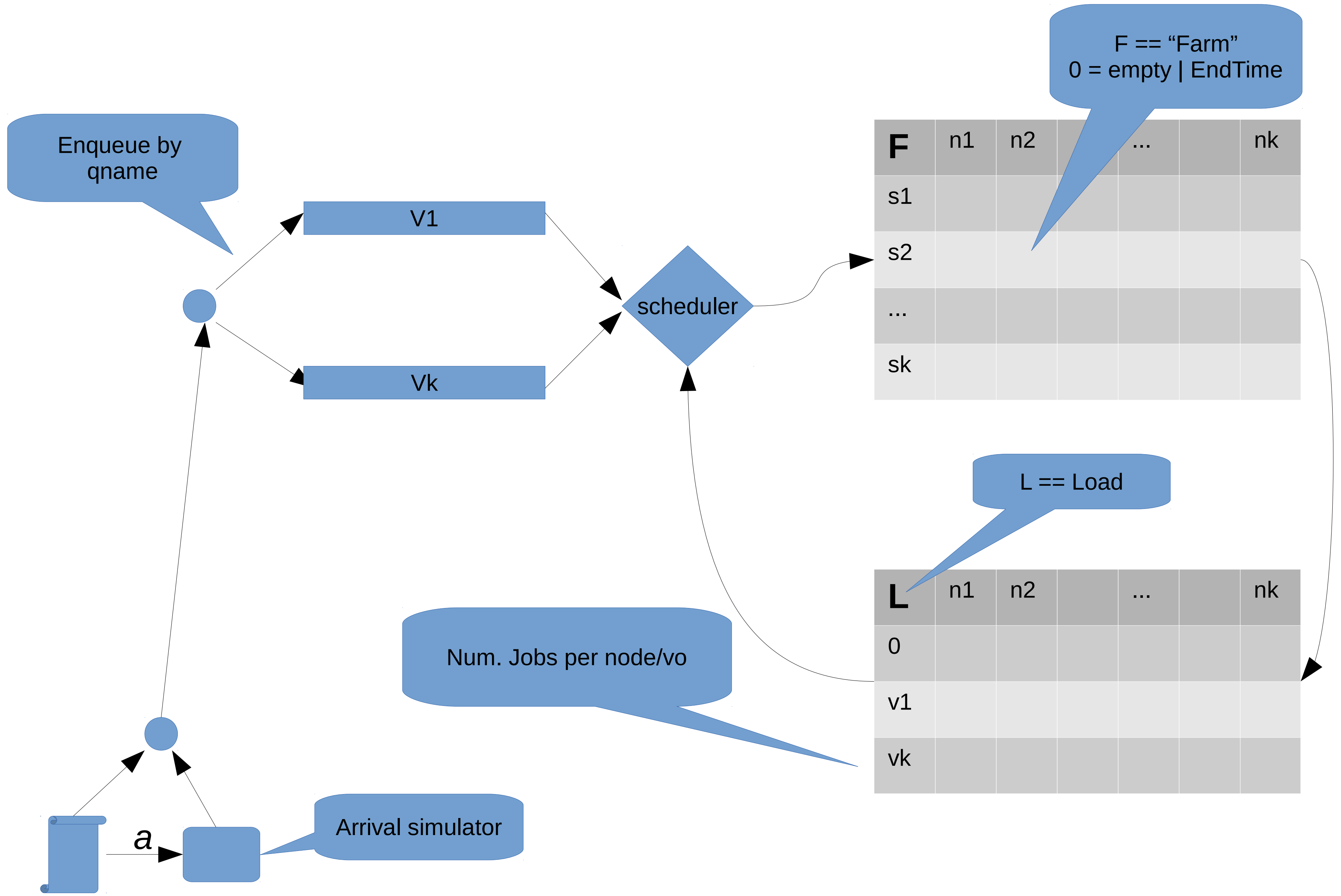
Other Customizations

- Monitor web reporting tool recently rewritten (based on graphite). Collected data available for (internal) accounting
- Dynamic WN update
  - Tool to automate kernel upgrade on WN. Closes the wn, reboots after node drained, open the wn again
- WN failure tracking
  - Offline nodes detected by LSF are notified as down (with explanation), to our HW inventory db (DOCET).
    - Summary email delivered to farm staff. The node is notified as ok when it comes back to production.
    - This provides down/up statistics per single node and HW model.

Special usecases

- The **1 job: 1 slot** (0.x cores if using HT) has been our main use-case until now
  - Pro: little or no dedicated WN, no unused cores.
- Multicore: unused core is an unavoidable side-effect.
- Job packing: trying to dispatch jobs with a given property into the same WN.
  - It was investigated months ago (http://goo.gl/n26yxN); this and similar use-cases may be addressed by configuring LSF *External Load Indexes*
  - A brief description follows

Farm simulator



Enqueue by qname

V1

Vk

scheduler

F == “Farm”
0 = empty | EndTime

F	n1	n2		...		nk
s1						
s2						
...						
sk						

L == Load

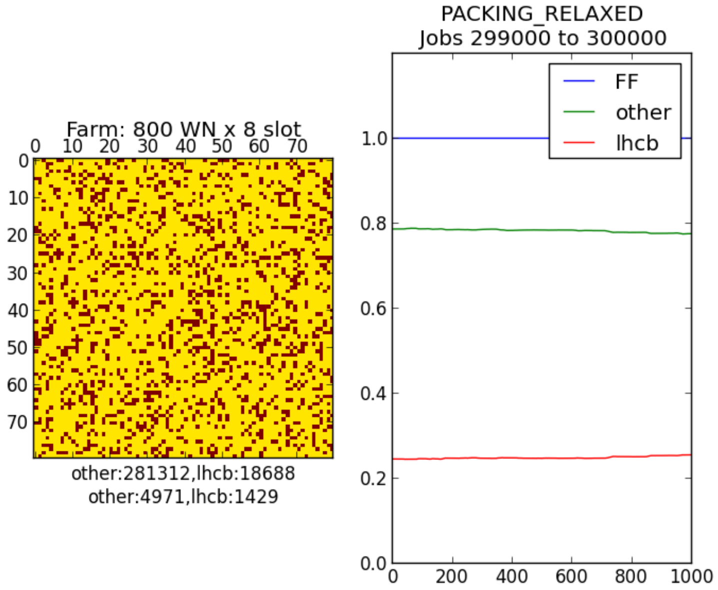
L	n1	n2		...		nk
0						
v1						
vk						

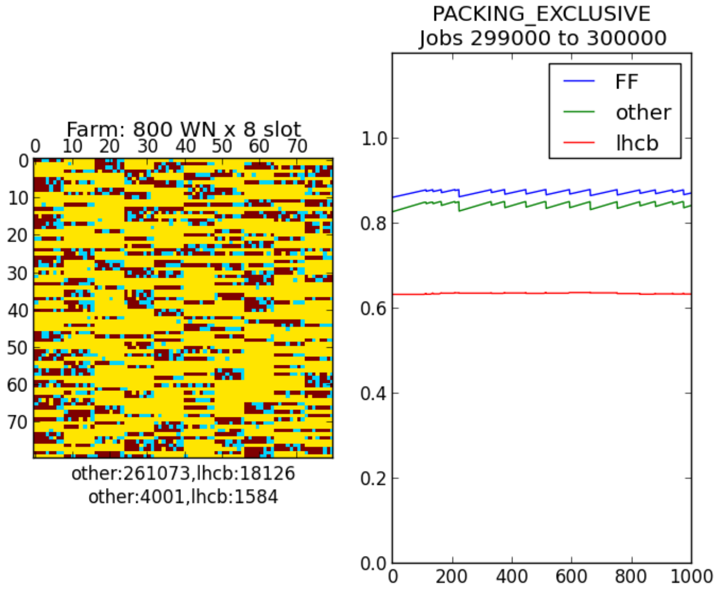
Num. Jobs per node/vo

a

Arrival simulator

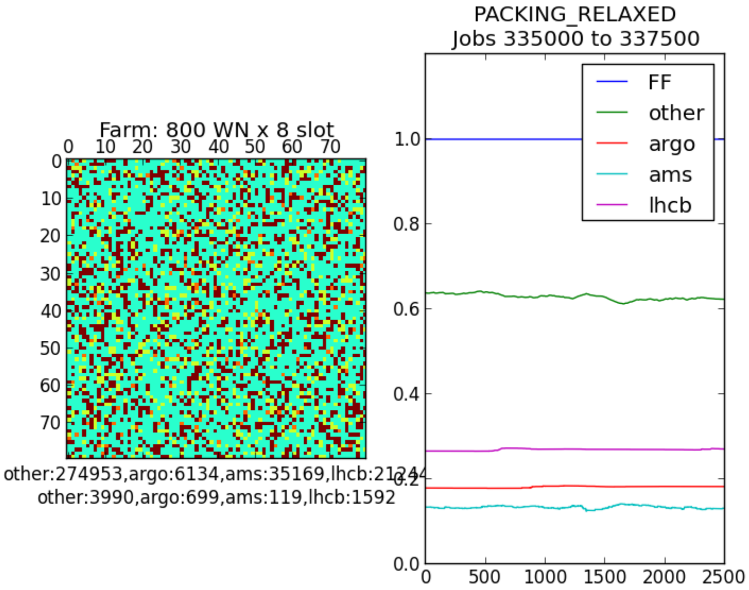
Relaxed vs Exclusive, 1 VO (lhcb)

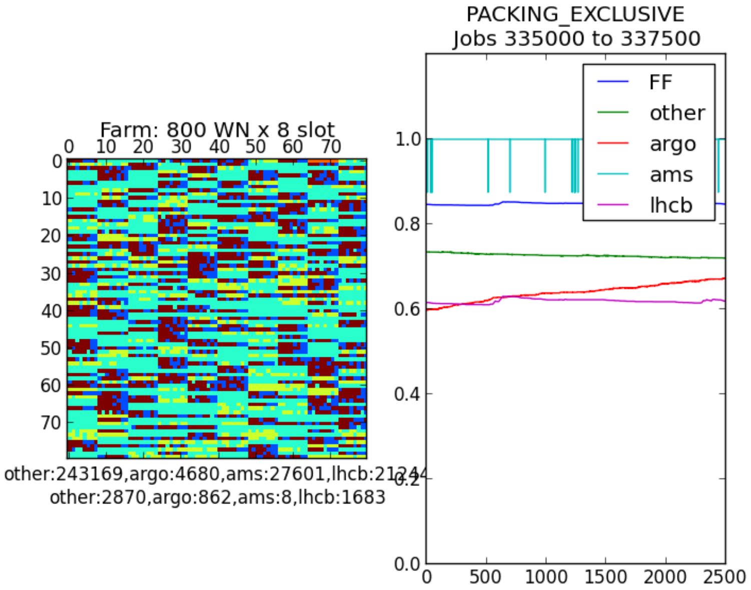




Farm, in the long run:
Relaxed: JP, poor aggregation, no empty slots
Exclusive: good aggregation, at the cost of unused slots.

Relaxed vs Exclusive: ams, argo, lhcb



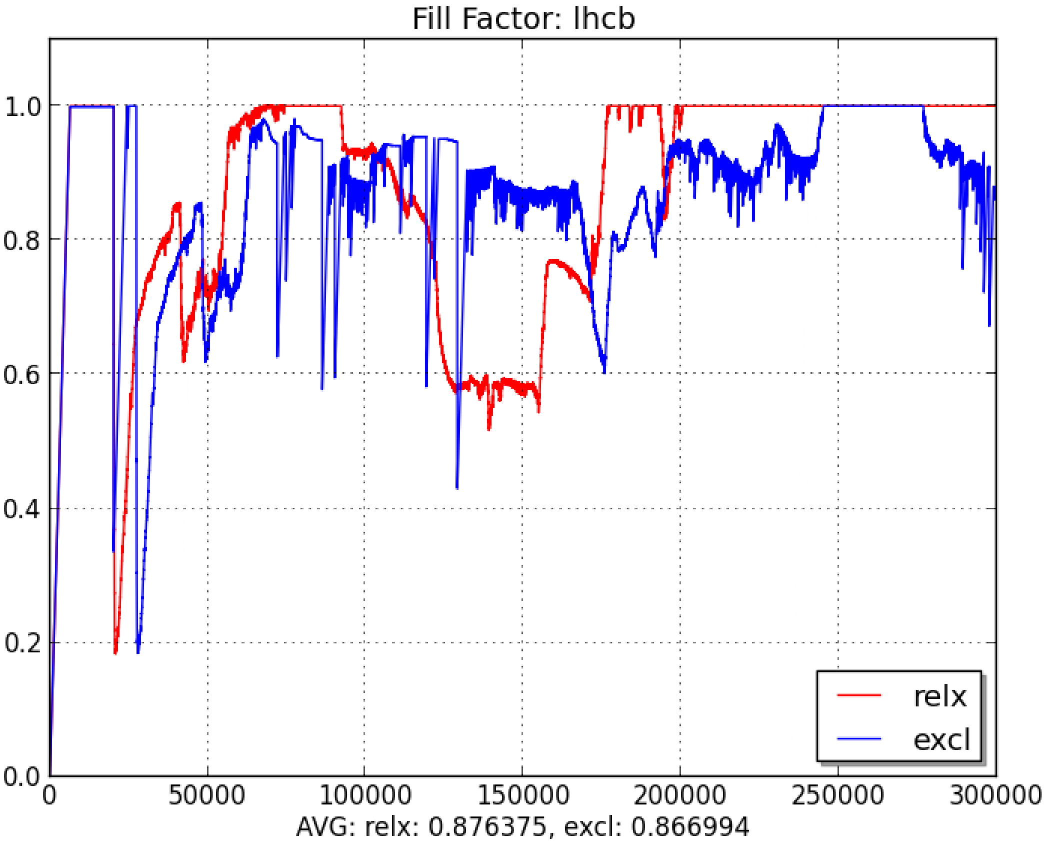


Three packing families:

Relaxed: JP, poor aggregation, no empty slots

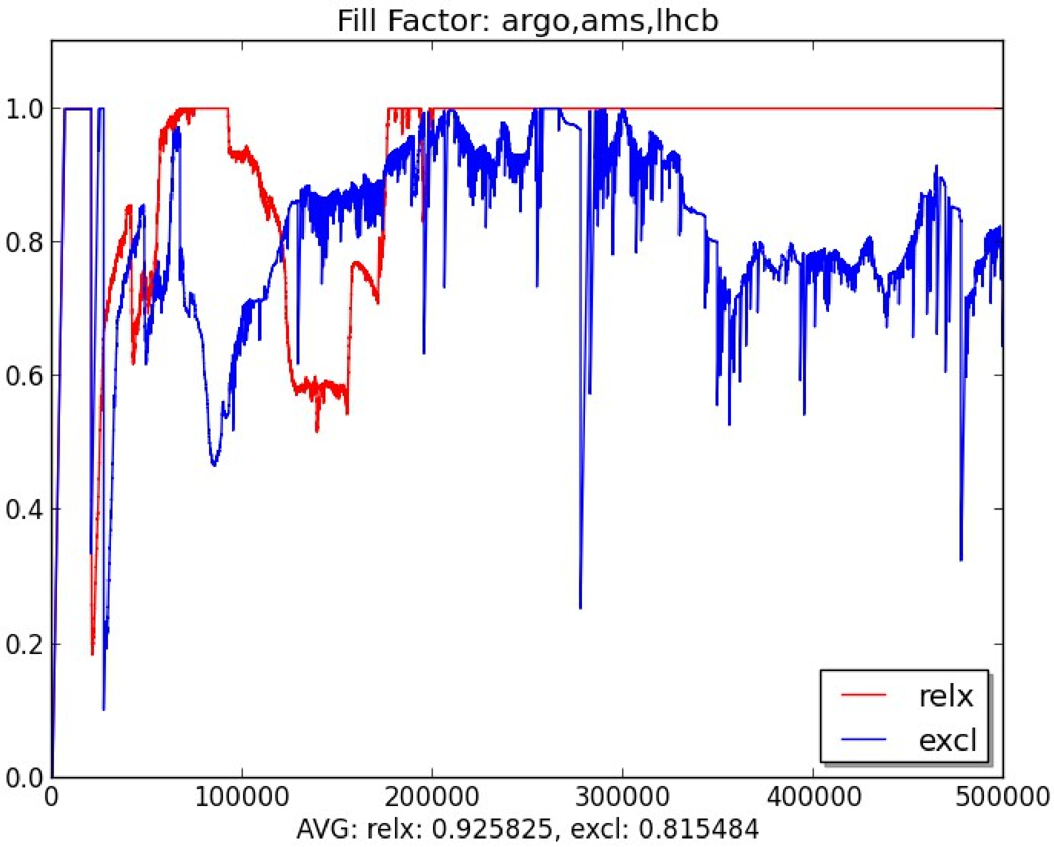
Exclusive: good aggregation, few unused slots.

Fill Factor (lhcb)



avg(ff_relx – ff_excl) = 0.0094 (~ 1%)
0.0094 x 800 * 8 = 60
Exclusive packing. “costs” 60 slot

Fill Factor: ams, argo, lhcb



avg(ff_relx – ff_excl) = 0.110341 (~ 11%)
0.110341 x 800 * 8 = 706
Exclusive packing cost: 700 slot

Defining External Load Index

- lsf.shared:

```
Begin Resource
RESOURCENAME  TYPE INTERVAL INCREASING DESCRIPTION
# two resources to exploit Job Packing
  pkoth      Numeric 15      Y           (no Pack)
  pkone      Numeric 15      N           (Pack)
```

- lsf.cluster.<clustername>:

```
Begin ResourceMap
RESOURCENAME  LOCATION
pkone      [default]
pkoth      [default]
```

- Apply changes: `lsadmin reconfig ; badmin mbdrestart`

External Load Index

- Checking the index

```
[root@lsf ~]# lsload  -I pkone
HOST_NAME       status    pkone
wn-104-03-01-08     ok      0.0
wn-104-03-01-12     ok      0.0
wn-104-03-01-06     ok      1.0
wn-104-03-01-10     ok      0.0
```

Write an Elim

- script executed at WN side

```
[root@wn-xyz ~]# . elim.jp
2 pkone 1 pkoth 0
```

-
  ```
  while True:
    sleep 10
    num1 , num2 = compute(“pkone”,”pkoth”)
     print “2 pkone %d pkoth %d”%(num1,num2)
  ```

- Output must have the form:
  - `n name_1 value_1 … name_n value_n`
- Script name is mandatory: `elim.<name>`
- Must be located under $LSF_SERVERDIR

Write an Elim

- Info retrieved from `/bin/ps`

- First we collect job pids:

  ```
      ps -o pid --ppid `pidof sbatchd`
  ```

- Then we get job groups:

  ```
      ps -o group -p pid1,...,pidn
  ```

- Finally we map and count group names to pkone, pkoth.

Using the index

- Before writing the esub we can check how to use the external index

```
#find nodes with packing jobs
lsload -I pkone -R "select[pkone>0 ||
pkone==0]"

#find nodes without packong jobs
lsload -I pkone -R "select[pkone==0]"
```

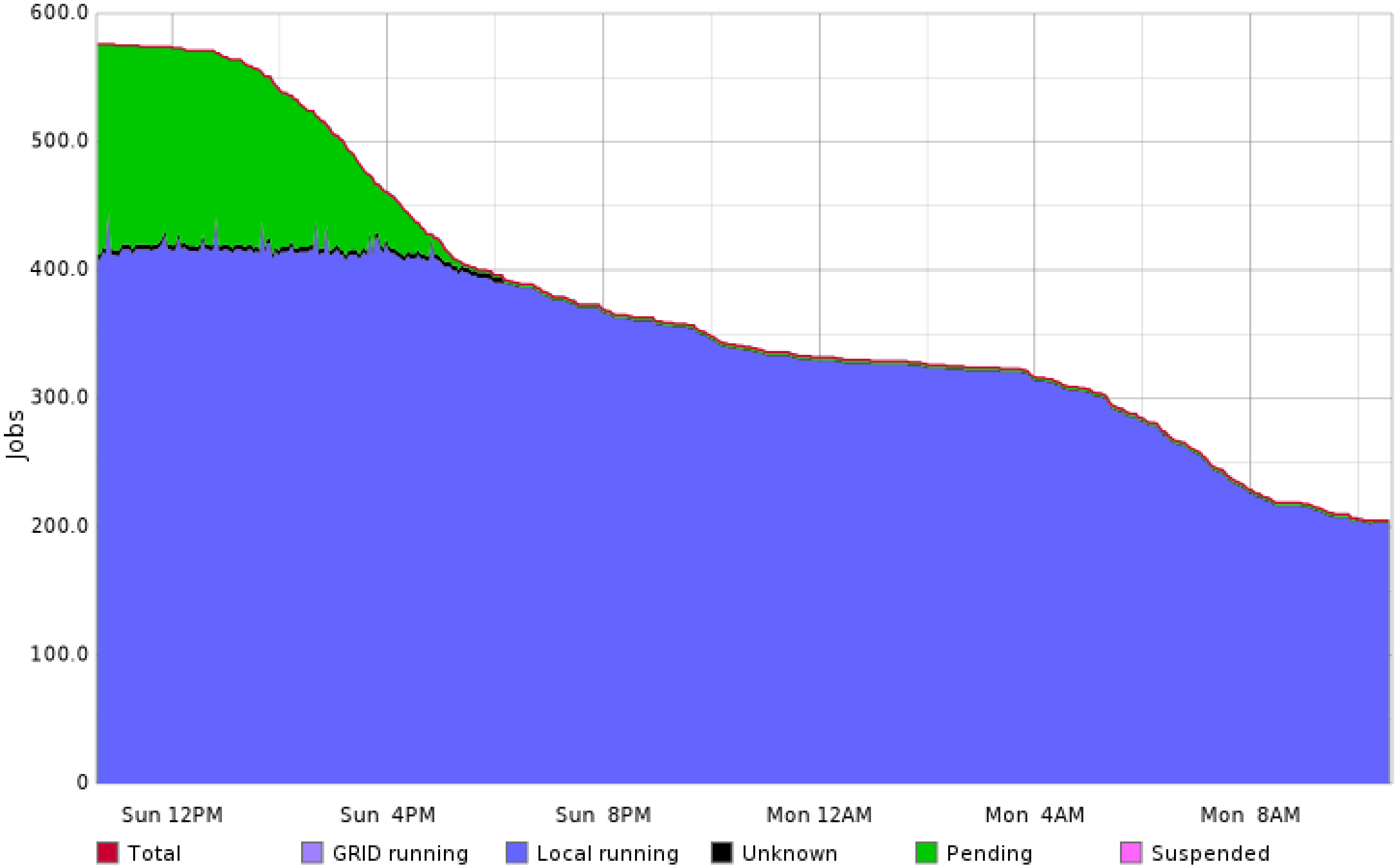
- Submit packing (pk1) and non packing (pk2) jobs

```
bsub -q pk1 -R "select[pkone > 0 || pkone ==
0]”\           sleep 3600
bsub -q pk2 -R "(pkone == 0) sleep 3600
```

LSF & External Index

- WnoDeS internals have been rewritten to use “external Indexes”:
  - At submission time an esub script runs on the submitting node. This defines a vmid value (uuid: 0xabcd123) and a resource request for that value
    - `-R "select[vmid==0 || vmid==0xabc123], order(-vmid)"`
  - LSF dispatches the job to a wnodes hypervisor (it has vmid==0). pre_exec triggers the creation of a vwn publishing the requested vmid value. When ready, the pre_exec fails
  - LSF dispatches the job (only) to the vwn with vmid==0xabcd123

Wnodes cluster



Multicore

- Ce0*x*-lcg.cr.cnaf.infn.it:8443/cream-lsf-mcore
- 1 LSF queue dedicated, 10 WNs, 8 cores
- Atlas and cms enabled
- Resources provided “as is” up to now

Multicore

- We plan to start testing the following configuration:
  - Mcore hostgroup available to multi-core and single-core jobs
  - Elim on the Mcore hosts publishing an External resource “mc == <number_of_mcore_jobs>”
  - Esub adds resource request for jobs:
    - MC==0 for single-core jobs
    - MC>0 || MC==0 for multi-core jobs

Multicore

- The desired effect is to have a dynamic set of nodes dedicated to m-core, enlarging or shrinking as needed
- If 0 nodes are running m-core jobs, first ones should be selected by LSF with advanced reservation
- Nodes running or finishing m-core jobs, remain dedicated to m-core until timeout.
- The reason would be to reduce to a minimum the number of “WN drain”, which happens when reserving a node.

Multicore

- We expect this to be effective if
  - There is little variety on the number of requested cores (e.g. dealing with 8-core jobs only)
  - Nodes with 8, 16, …, k * 8 cores
  - A steady flow of multicore jobs is provided.

Conclusions

- Our overall experience with LSF is definitely positive
  - Stable, robust, resilient, reliable
  - Many usecases are quite straightforward to configure
  - Need to gather experience with multicore
- Scalability not much a problem yet

Acknowledgments

Thanks to the Multicore TF for their activity and support to requests.