Job packing: optimized configuration for job scheduling

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

The Problem

- The INFN Tier-1 batch system (currently IBM/Platform LSF) selects for dispatching the lesser loaded candidate WN.
  - *Candidate*: “have a free slot” & “adequate”
  - *Adequate*: have suitable resources for the job to run.
  - *Lesser loaded*: According to a metric. (system load)

We want to modify the WN selection according to one or more (known) Job properties C (queue|group|<criterion X>).

Motivation

- It is desirable to have some activities executing on the smallest possible set of nodes
  - e.g. exp. Auger
- Keep together “risky jobs”
- Keep together MPI jobs
- Being able to spread (known) I/O intensive jobs

Motivation (2)

- WNoDeS (see www.eu-emi.eu) may exploit these features to achieve:
  - "equal" Virtual wn on the same HV (minimum vwn imageset to copy to the HV)
- Packing or no_packing for jobs requiring common resources

Packing Policies

- PACKING_RELAXED (aggregation):
  - Job J with a given property (C(J) == True) *should* prefer nodes having similar jobs already running on them.
  - No constraints applied on other Jobs.
- PACKING_EXCLUSIVE (concentration):
  - Jobs with property C should prefer nodes as above

Other jobs *must* avoid nodes with C-jobs running in it.

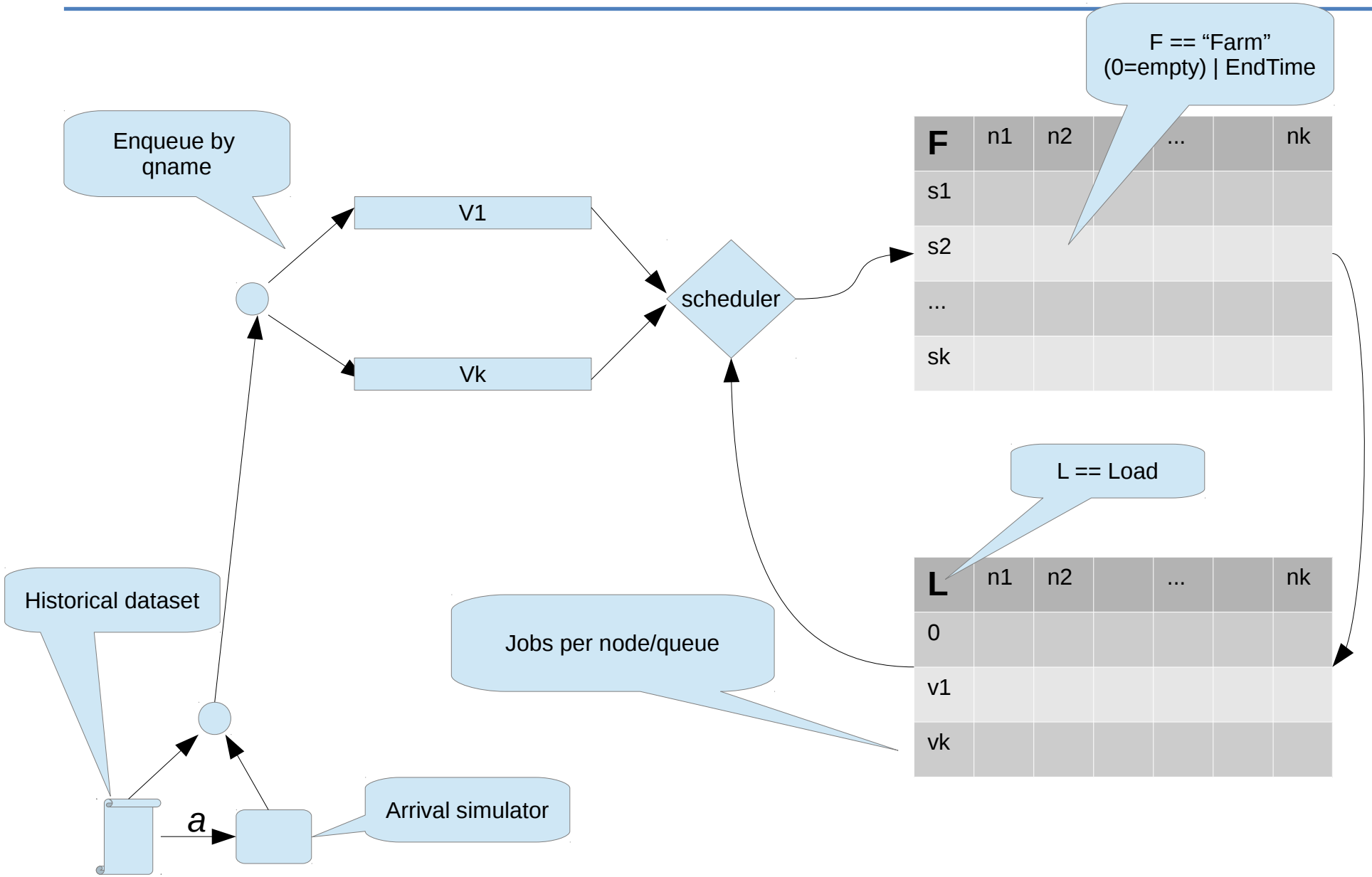
- PACKING_NONE (spreading):

C-jobs should prefer nodes with NO C-jobs running in it.

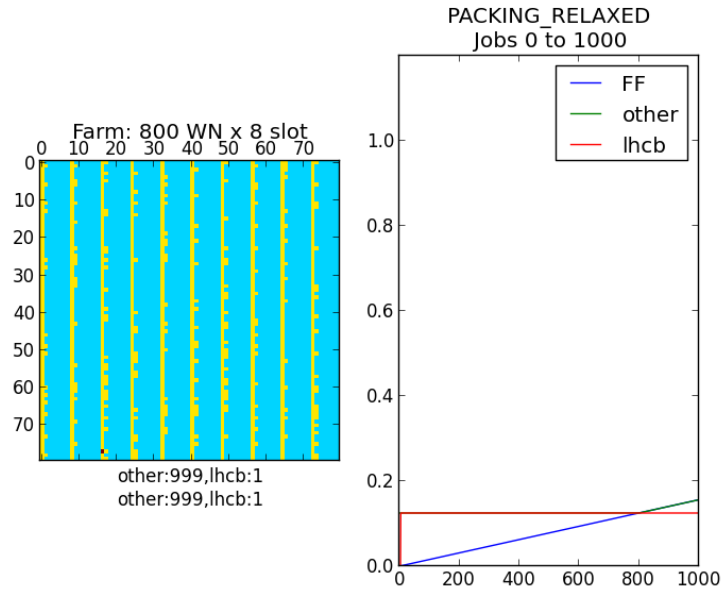
Impact

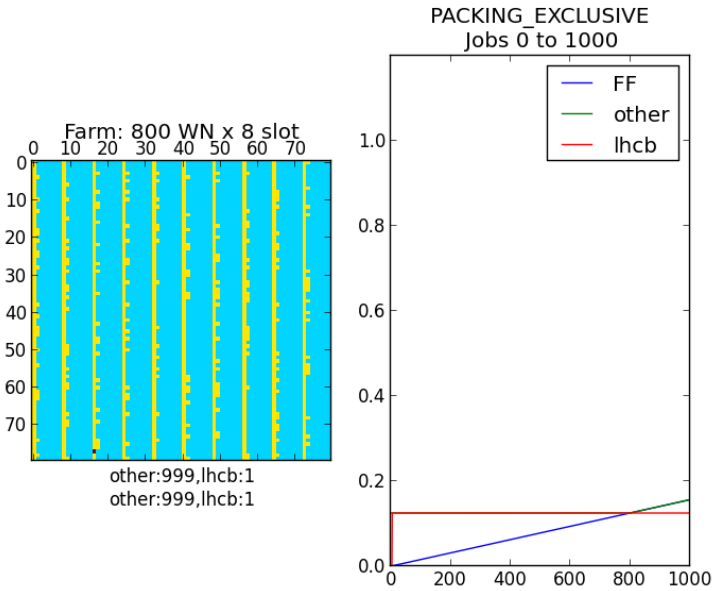
- A simple “farm simulator” has been written to evaluate different packing policies.
- Two synthetic indicators:
  - “Packing Index” (concentration)
    - PI = Needed_nodes / Used_nodes
  - “Fill Factor”: (saturation)
    - FF = Used slots / Available slots
- python, pylab (matplotlib, numpy)
- Real data (Tstart, Tend, queuename; ~ 2.5Y historical dataset, ~15Mrecord) or simulated arrivals (modeled after historical dataset)

Farm simulator



Relaxed vs Exclusive, 1VO



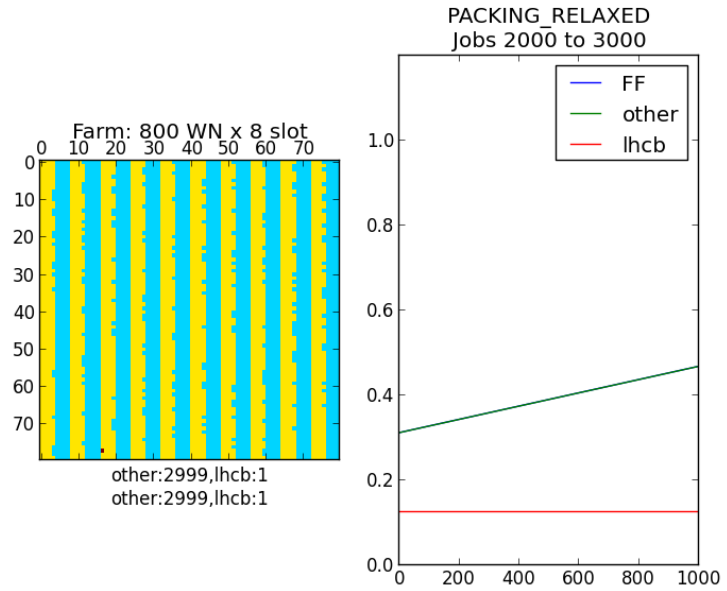


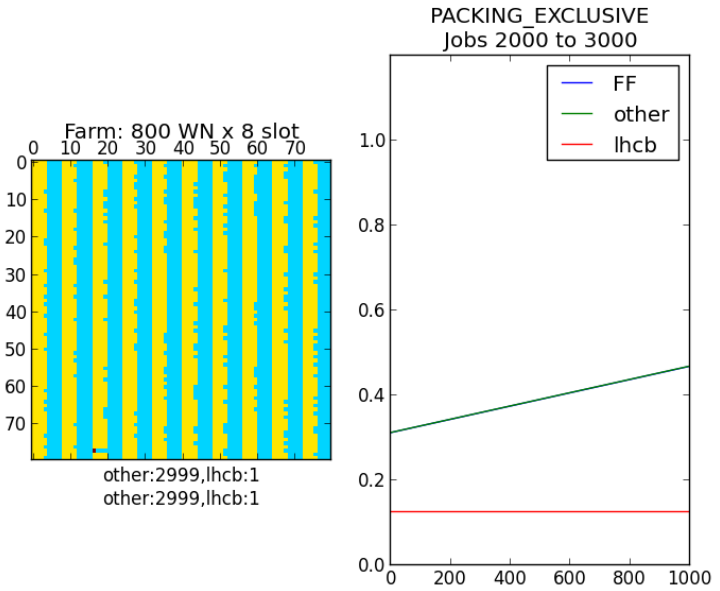
Starting with empty Farm behaviour is initially the same.

FF: Fill Factor

Other: Any job who doesnt require packing

Relaxed vs Exclusive, 1VO lhcb

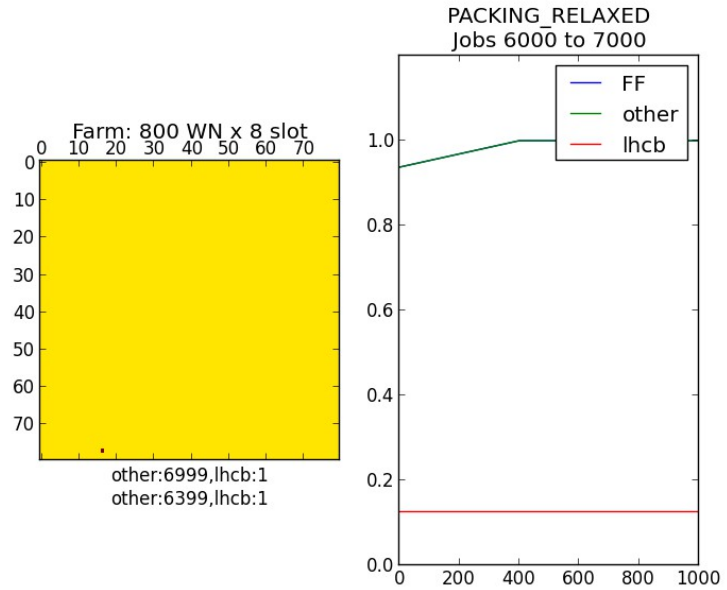


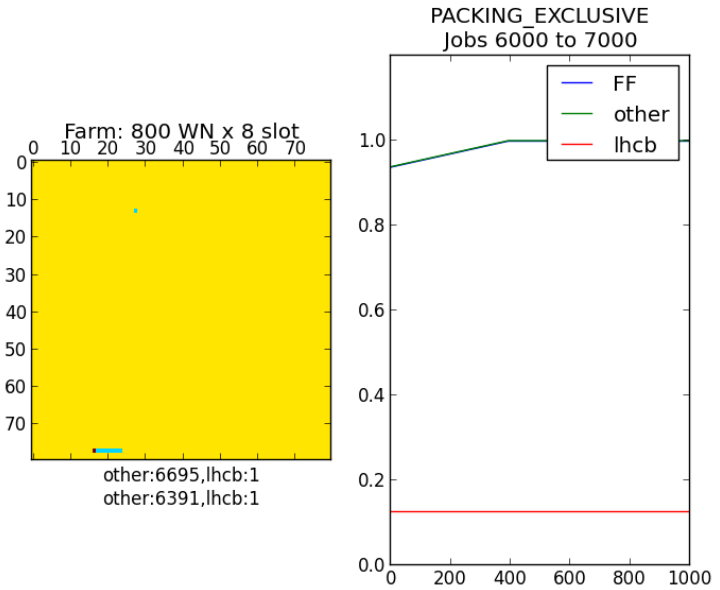


Relaxed: the node with LHCB job gets other non LHCB jobs

Exclusive: the node with 1 job LHCB remains reserved

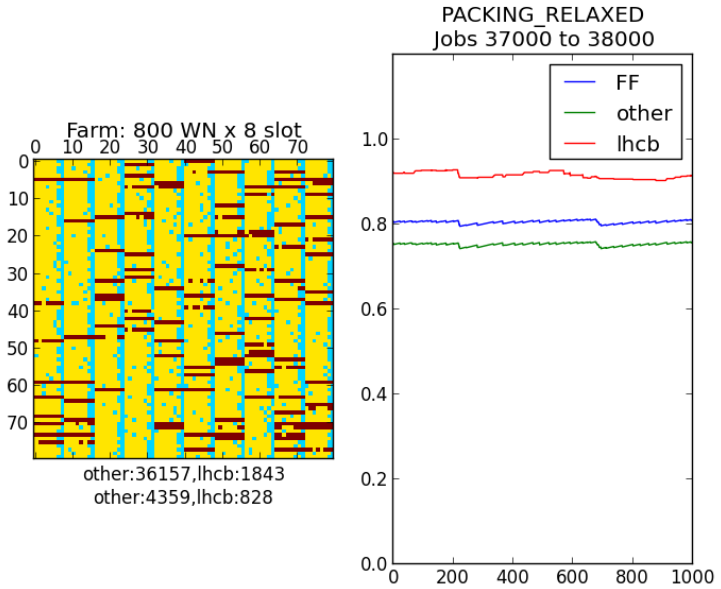
Relaxed vs Exclusive, 1VO lhcb

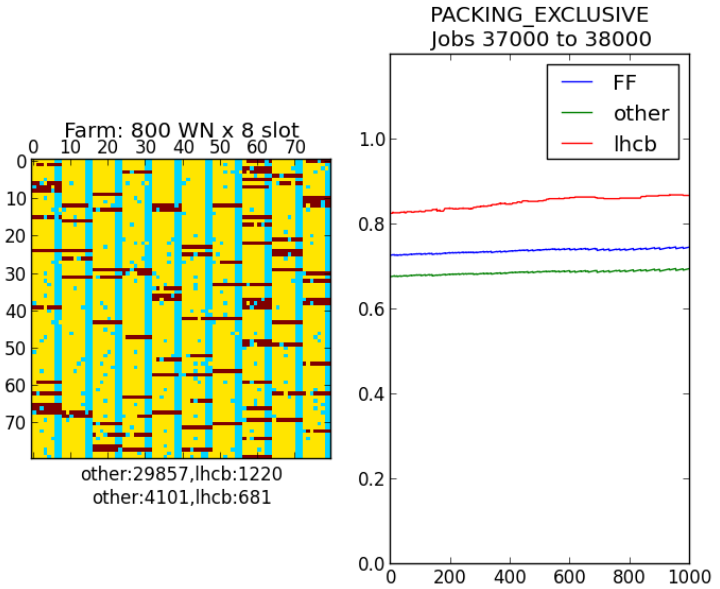




Farm, saturation begins:
Relaxed: All slots are in use, one only JP
Exclusive: the node with 1 JP job remain reserved (we have unused slots)

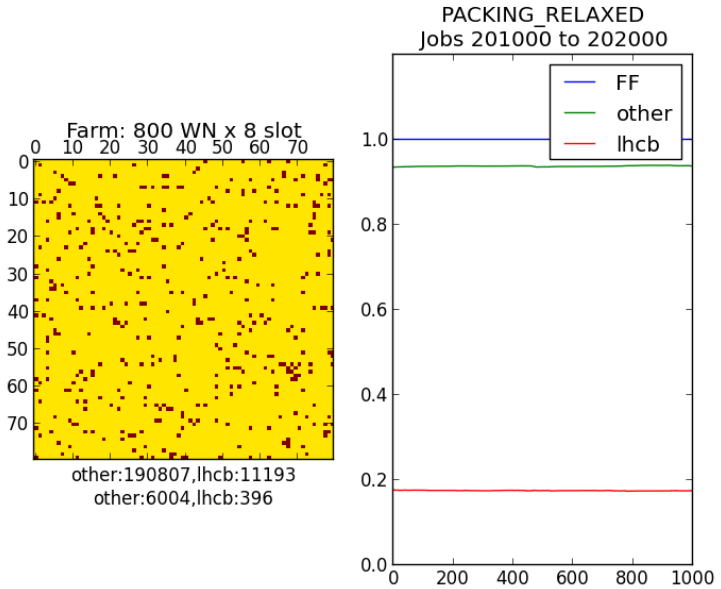
Relaxed vs Exclusive, 1VO lhcb

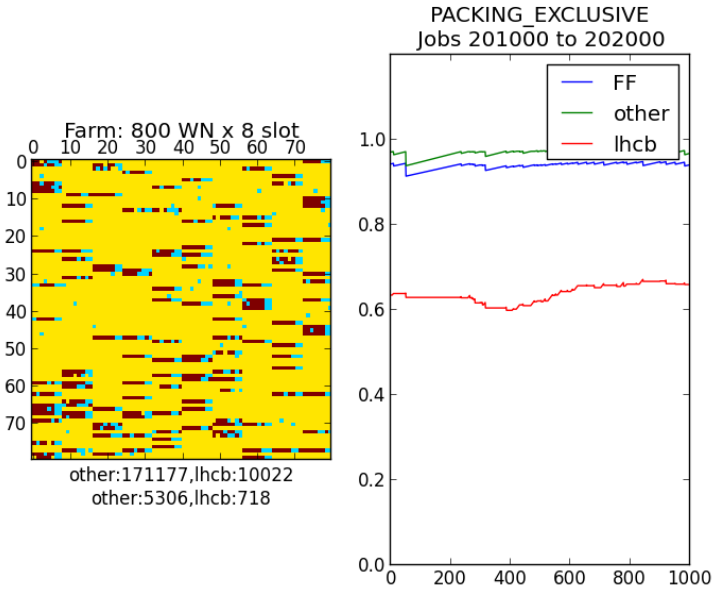




Farm, post saturation:
Relaxed: sligthly better than Exclusive

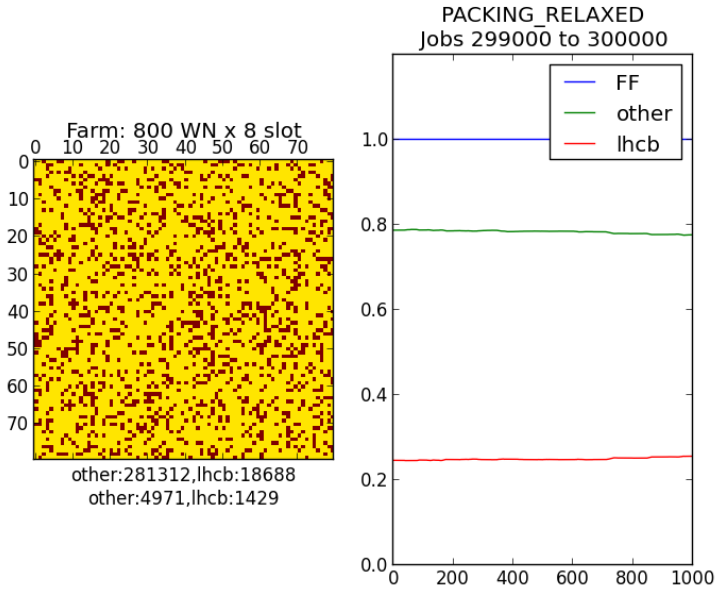
Relaxed vs Exclusive, 1VO lhcb

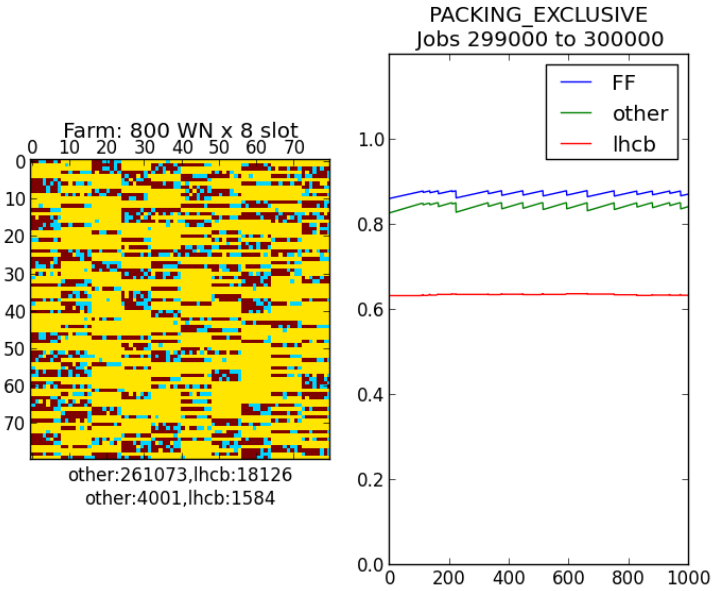




Farm, next saturation:
Relaxed: JP high dispersion, no empty slots
Exclusive: little dispersion, there are unused slots, --> slower Farm.

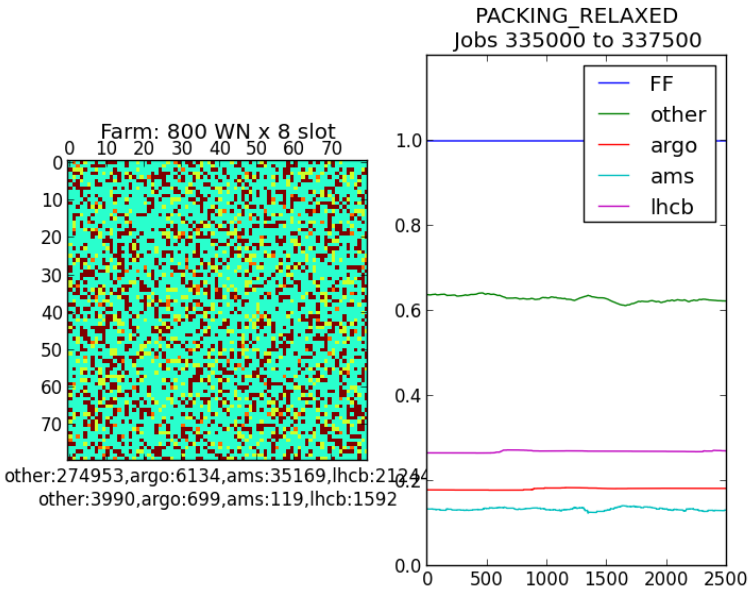
Relaxed vs Exclusive, 1VO lhcb

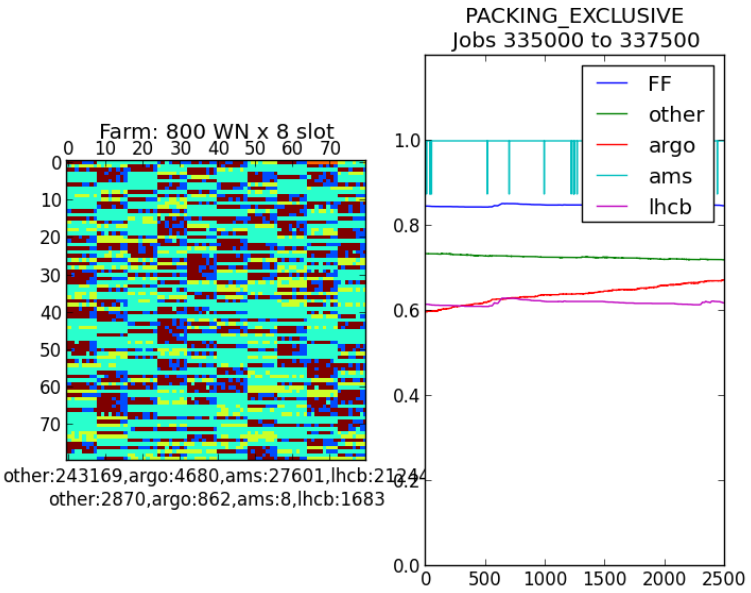




Farm, in the long run:
Relaxed: JP, poor aggregation, no empty slots
Exclusive: good aggregation, at the cost of unused slots.

Relaxed vs Exclusive ams,argo,lhcb



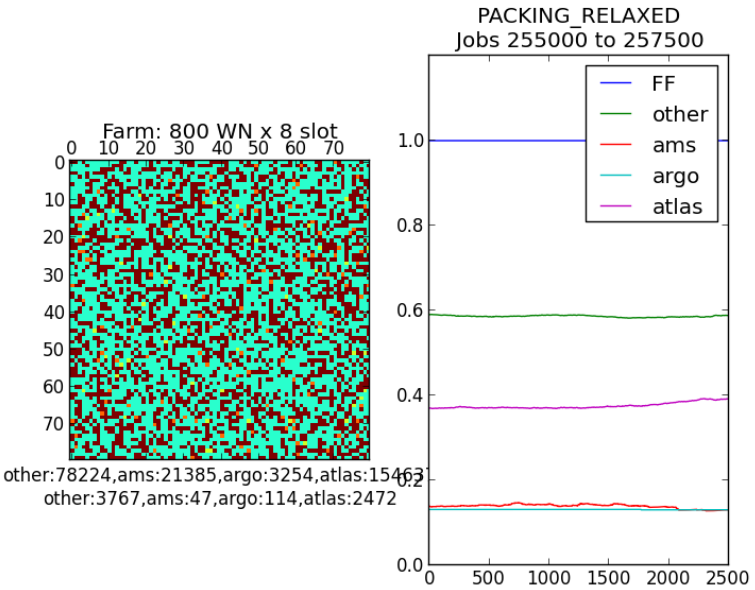


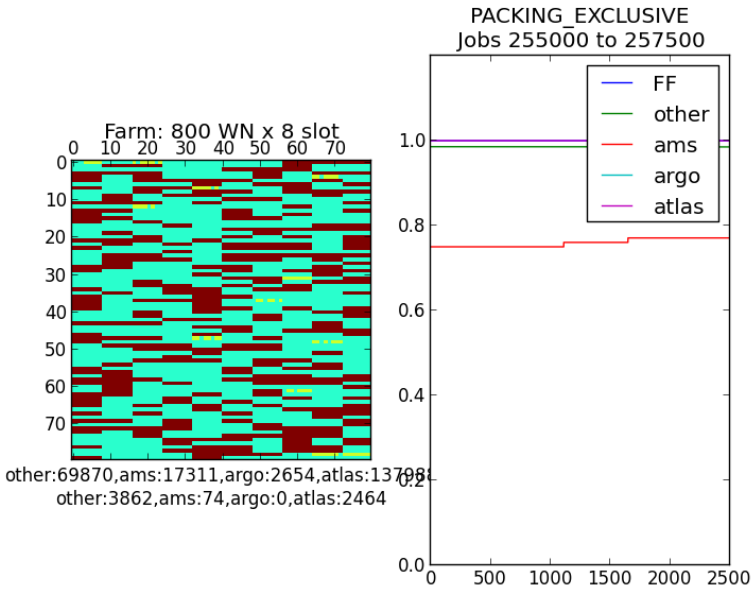
Three packing families:

Relaxed: JP, poor aggregation, no empty slots

Exclusive: good aggregation, few unused slots.

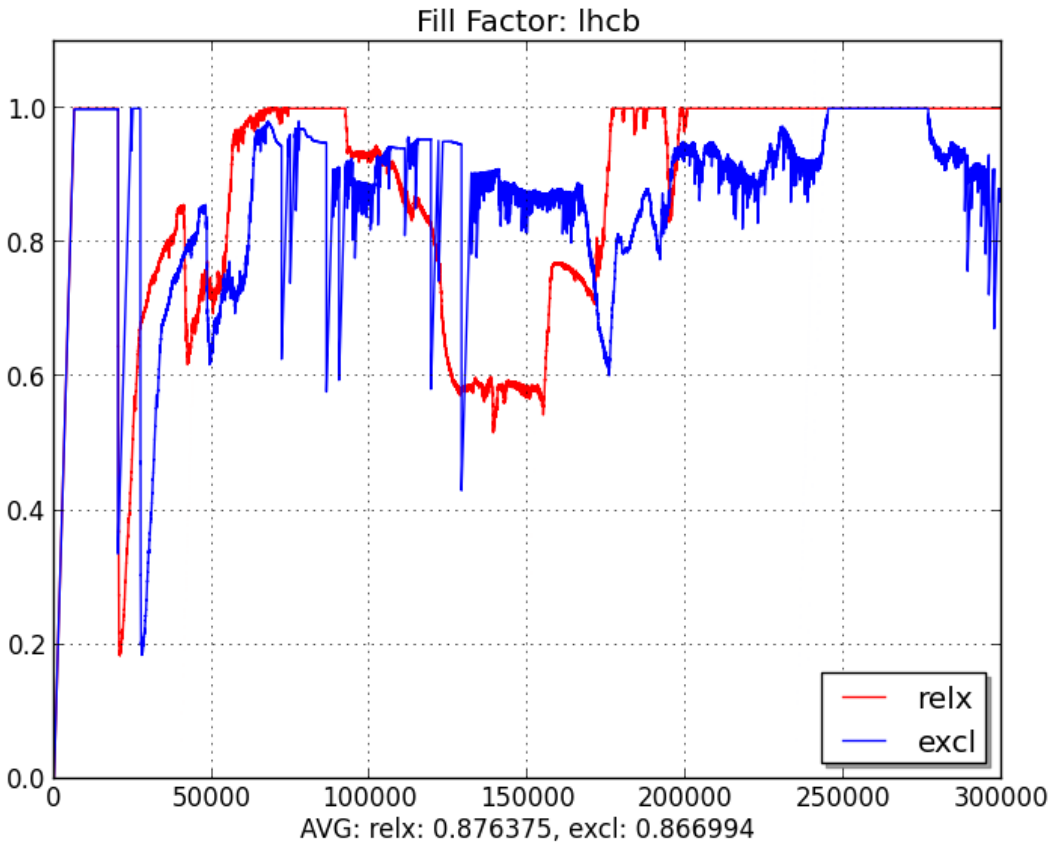
Relaxed vs Exclusive ams,argo,atlas





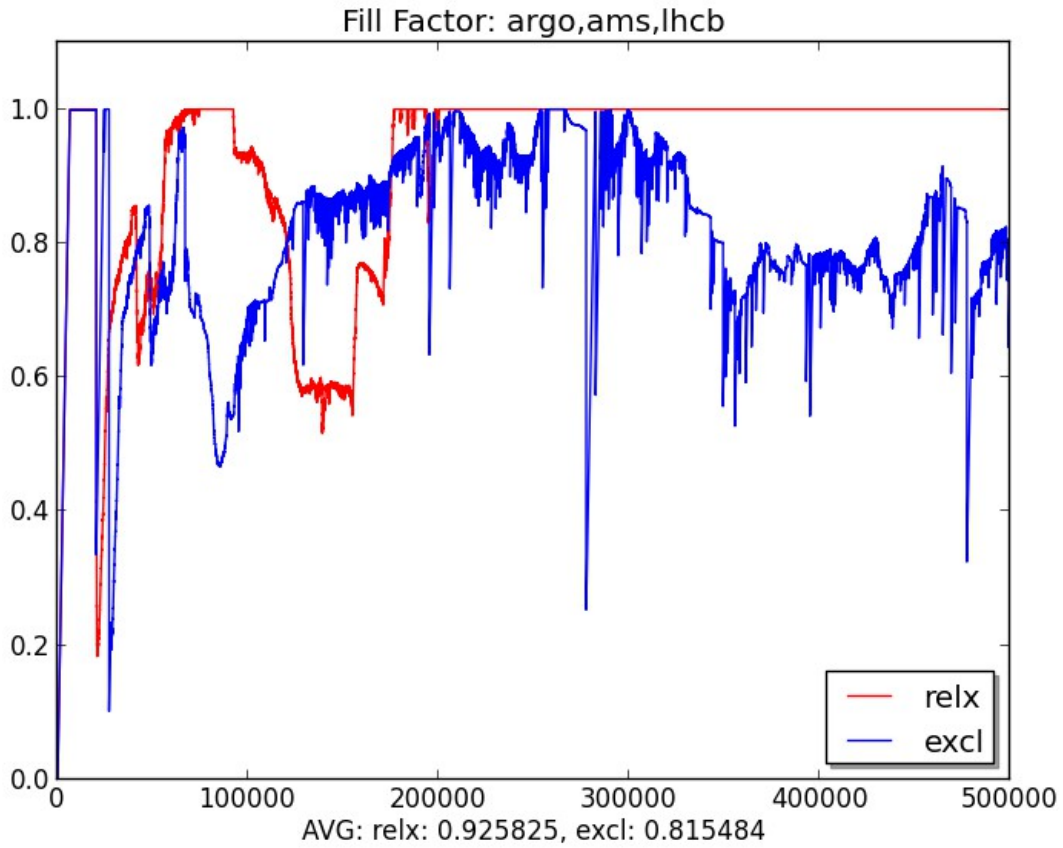
- Relaxed: Farm worked out more jobs.
- Exclusive: Non packing Job have longer queue time.

Fill Factor, lhcb



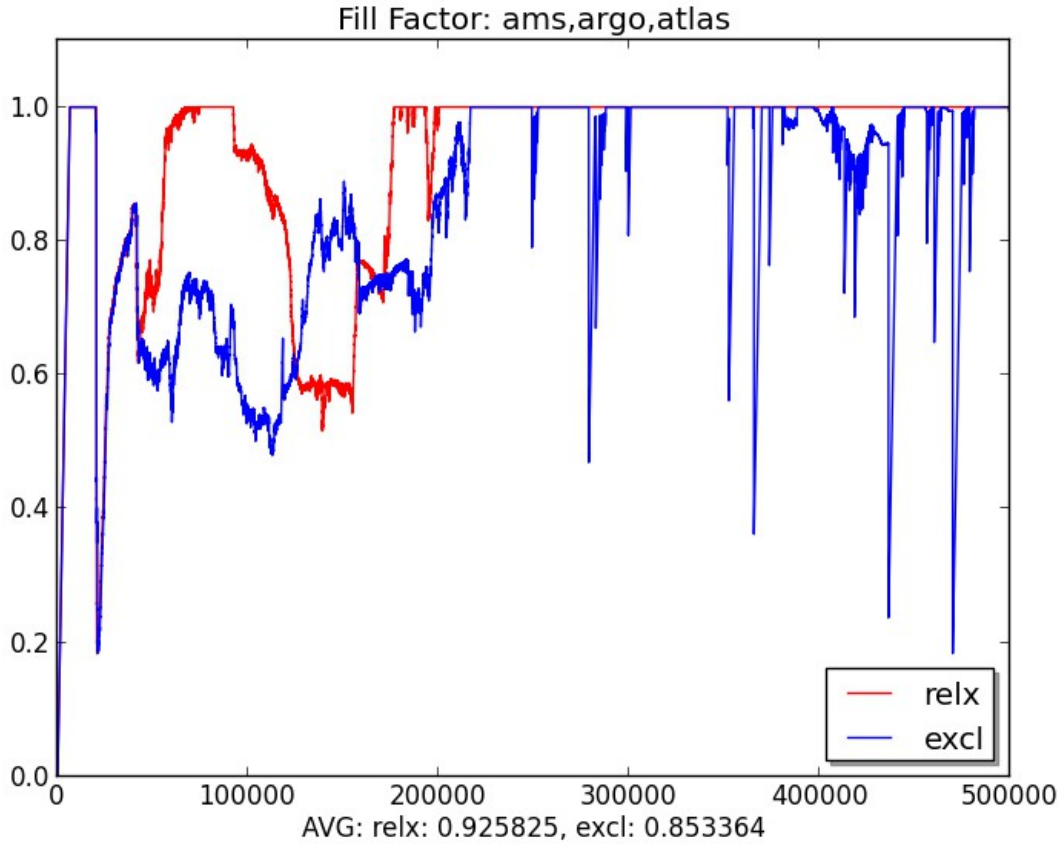
avg(ff_relx – ff_excl) = 0.0094 (~ 1%)
0.0094 x 800 * 8 = 60
Exclusive packing. “costs” 60 slot

Fill Factor, ams,argo,lhcb



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

Fill Factor, ams,argo,atlas



Note: in the second half the cost decreases to 0.0438 x 800 * 8 = 280

Comments

- Arrival order matters.
  - Exclusive: may reduce FF, while providing good PI.
    - Very long Jobs may decrease the FF
  - Relaxed: PI may degrade a lot, no impact FF.
- We can balance between Relaxed and Exclusive by considering the introduction of a “Time To Live” constraint.
  - (remove the “exclusive” constraint on nodes after some time without new jobs dispatched to it).
- Relaxed: TTL = 0 ; Exclusive: TTL $\rightarrow \infty$
  - Exclusive packing without TTL may be risky! (e.g. on worst case arrival order)

Implementation (LSF 7.06)

Example for Exclusive packing

1. Add new custom dynamic resources

   (Configure External Load Indexes)

2. Each node cyclically reports their value.

   (write/deploy an elim script)

3. At job submission, specify/add Conditions (resources Requirements) based on those indexes.

   (write/configure an esub script)

- These conditions are then evaluated by the scheduler at dispatch time.

Define 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`

Write an Elim

- script executed at WN side. Counts the number of running "packing jobs". Non packing ones are counted as "other".

```
[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 has a mandatory prefix: `elim.<name>`
- Must be located under $LSF_SERVERDIR
  ```

Write an Elim

- We detect a packing job by its group id
  - (The group id identifies the queue)
- Info retrieved from `/bin/ps`

1. collect job pids:

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

2. get job groups:

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

3. map and count group names to pkone, pkoth.

Check your 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 packing 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 "select[pkone == 0]" sleep 3600
```

Check your index


```
[root@lsf ~]# lsload  -I pkone:pkoth
HOST_NAME       status      pkone   pkoth
wn-104-03-01-08     ok        0.0     1.0
wn-104-03-01-10     ok        0.0     0.0
wn-104-03-01-12     ok        0.0     0.0
wn-104-03-01-06     ok        2.0     0.0
```

Write an esub

- script executed in the Master, at bsub invocation

```
#!/bin/sh
if test "$LSB_SUB_PARM_FILE" != ""
then
    . $LSB_SUB_PARM_FILE
    if [ $LSB_SUB_QUEUE  = "pk1" ] ; then
       eval  echo 'LSB_SUB_RES_REQ=\
"\"select[pkone > 0 || pkone == 0]\"" ' > $LSB_SUB_MODIFY_FILE
    else
       eval  echo 'LSB_SUB_RES_REQ=\
"\"select[pkone == 0]\"" ' > $LSB_SUB_MODIFY_FILE
    fi
fi
```

- This implements Exclusive packing for the pk1 queue
- Role: automatically adds the proper `-R "..."` option.

Activate esub

- Save as `$LSF_SERVERDIR/esub.jp`
- Set in lsf.conf: `LSB_ESUB_METHOD=jp`
- `lsadmin reconfig;`
- `badmin mbdrestart`

About TTL

- Modify elim to force `pkone=0` after some time without new jobs in the node

Exploiting LSF API

A different approach

- Customize an external scheduler plugin
  - Sample `C` code in `lsf/7.0/misc/examples/external_plugin/`
- The plugin can be invoked with `bsub -extsched -R "EXTSCHED_OPT=..."`
- The code has access to a list of candidate hosts selected by LSF for dispatching.
- References in `lsf_programmer.pdf`

Exploiting LSF API

- In the plugin it is possible to:
  - Filter theCandidate Host List
  - Adjust Host order in the list
  - Allocate jobs to hosts
- The Idea:
  - move up in the CandidateHostList those having pk jobs
  - This can be done either using the external index or directly checking for pk jobs presence in the candidate host.

Exploiting LSF API

- An example program (early test):

```
[root@lsf pkhost]# ./pkhost pk1
queue pk1 is valid
lsb_hostinfo: 2 pk hosts found...
wn-104-03-01-08-a --> OK: 8 slots, 3 jobs
wn-104-03-01-06-a --> OK: 8 slots, 5 jobs
Selected host: wn-104-03-01-06-a;
```

- Validates queuename (`lsb_queueinfo`)
- get running pk1 jobs (`lsb_openjobinfo`) and counts them by hostname
- Identifies the free (`lsb_hostinfo`) Host with the higher number of pk1 jobs.

The End

Thanks!