Job Scheduling Explained

More than you ever want to know about how jobs get scheduled on WestGrid systems ...

Martin Siegert, SFU

Cluster Myths

- "There are so many jobs in the queue - it will take ages until my job runs."
  - most of the time those jobs are from users with very low priority, because they run a large no. of jobs.
  - some of the jobs may not be allowed to run (policies, e.g., max. no. of processor-seconds per user)
- "We have an allocation of 100 cores, but my 80p job is not running."
  - cores are not reserved for exclusive use.

⇒ **Cannot tell from the no. of queued jobs how busy a cluster is!**

Scheduling Goals

- fairness
  - distribute resoures fairly (not necessarily equally)
- efficiency, high utilization
  - resources are not reserved for exclusive use

Impossible to accomplish both goals perfectly!

- jobs with short walltimes and/or small no. of processes result in higher utilization and make it easier to distribute resources between users.
- not all problems/workflows can be adapted to that kind of conditions.

Overview

- Resource Manager and Scheduler:
  torque and moab
  - ➤ torque: defines resources (cores, memory, file space), starts jobs, terminates jobs, copies results, and much more.
  - ➤ moab: tells torque which job to start with how many cores on which cpus/nodes; tells torque which job to terminate.
- ***Today's talk is (mostly) about moab***
  - ***– the scheduler***

The Scheduler

- calculates the priority of jobs
- ranks jobs according to their priority
- determines the sequence in which jobs get started
- determines which jobs can be used for backfill

Accounting Groups

- all users belong to (at least) one accounting group (AG)
- determined by PI - a faculty member at a Canadian university; collaborators, postdocs, grad students sponsored by that PI belong to the same AG.
- all members of an accounting group have the same priority (by default - this can be changed on request, see later).

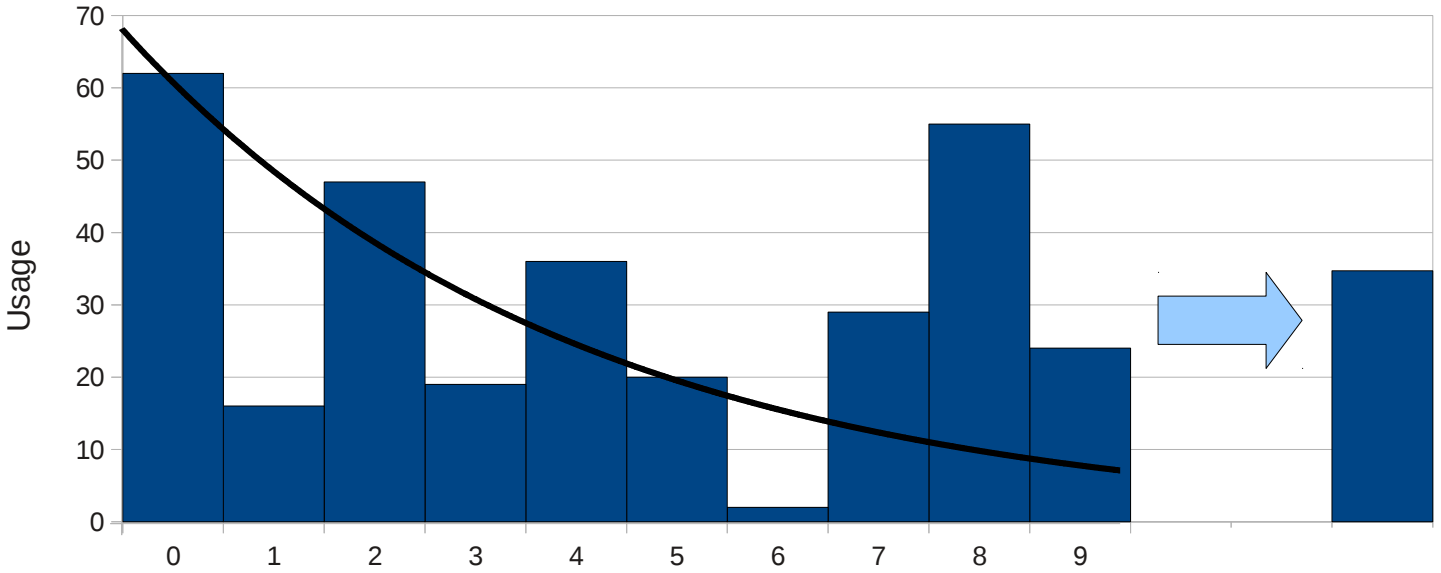
Fairshare Target

- each AG gets a usage target assigned: no. of cores.
  - AG with RAC allocation: *target = allocation*
  - AG without RAC allocation:

    target =
    (no. of cores in cluster – sum of RAC allocations)
    /(no. of AG without RAC allocation)
- this is the usage that you can expect *on average*

Usage

Weighted average over the past 10 – 14 days:



$$usage = \sum_{i=0}^{N-1} (U_i d^i) / \sum_{i=0}^{N-1} d^i \qquad d \simeq 0.8 \ldots 0.9$$

Job Priority

- fairshare priority:

  priority ~ (target – usage)

  - positive priority: ***target > usage***
    - ➔ above average
  - negative priority: ***usage > target***
    - ➔ below average
  - ***average priority = 0***

Job Priority
(cont'd)

No effect on priority:

- waiting time in queue
- order in which jobs were submitted
- number of jobs in the queue

  (there is an effect though: only the first 10 jobs from each user are considered for scheduling)

Fairshare: Target and Usage

Jobs waiting in the queue:
Will their priority rise or fall?

Depends on the number of jobs that are currently running from your AG:

- no. of cores in use > target
  ⇒ priority decreases
- no. of cores in use < target
  ⇒ priority increases

Fairshare Trees

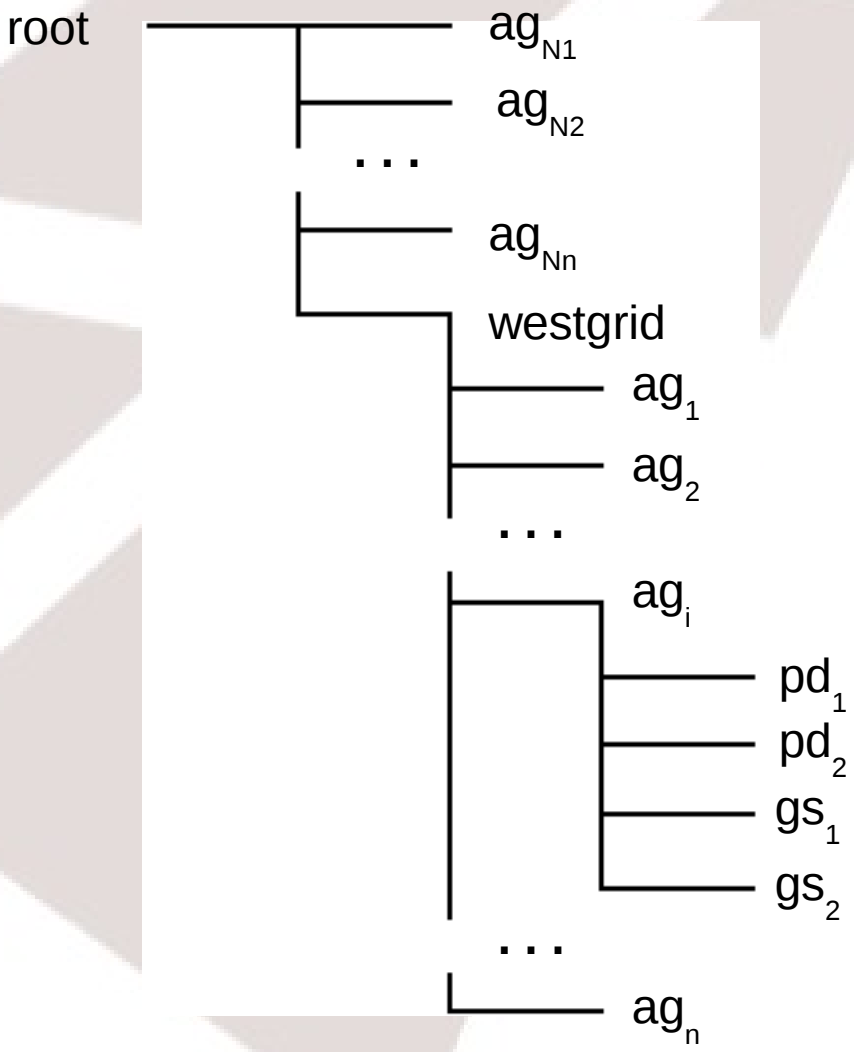
So far: all AG treated the same way.

Not completely true:

fairshare has a tree structure.

Lower levels dominate priority.

Possible to assign different targets within AG – ask us!



Ranking of Jobs

- ranking is by priority (fairshare)
- jobs are scheduled in priority order

How Jobs Get Scheduled

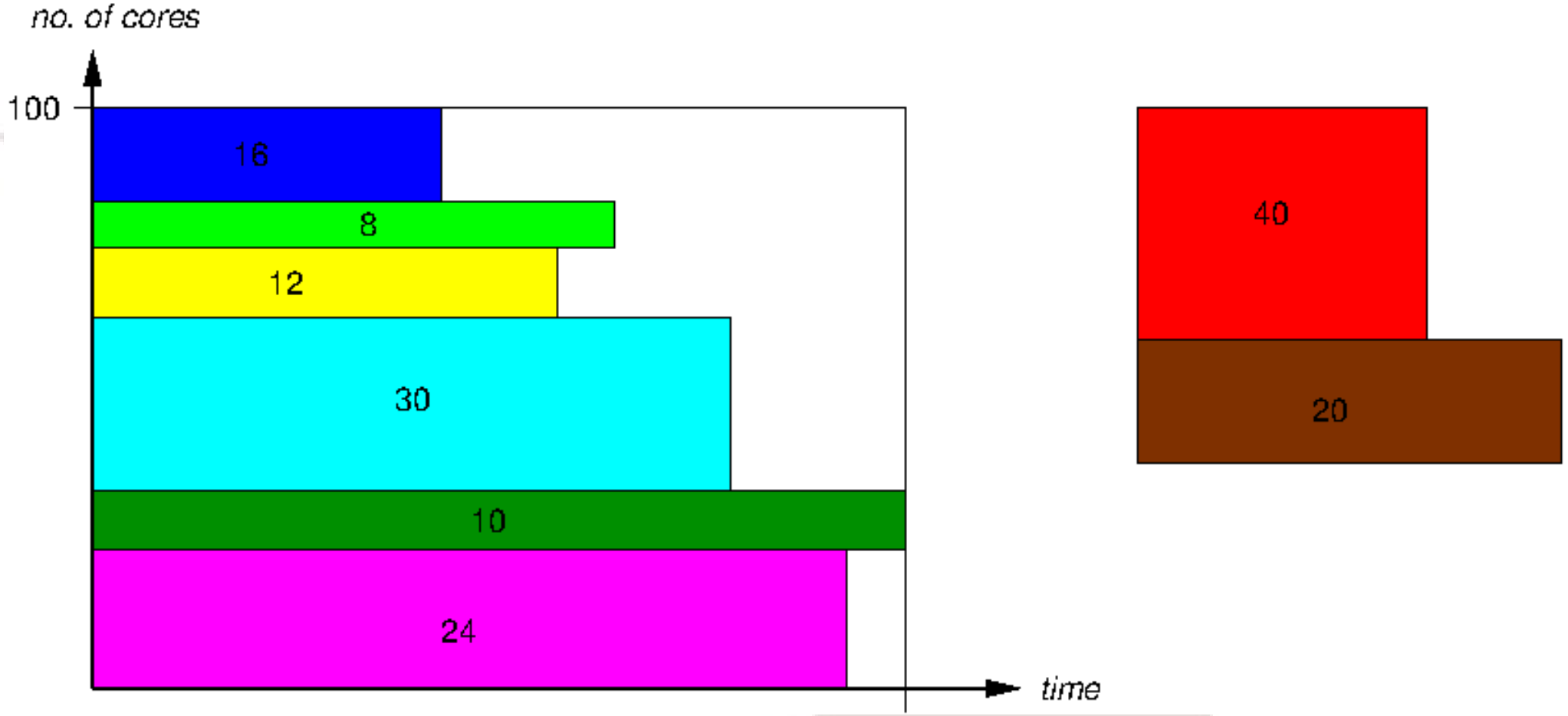
"My job has the highest priority. It should run now!"

No.

Usually systems are busy, i.e., all cores are in use.
We cannot run your job instantly.

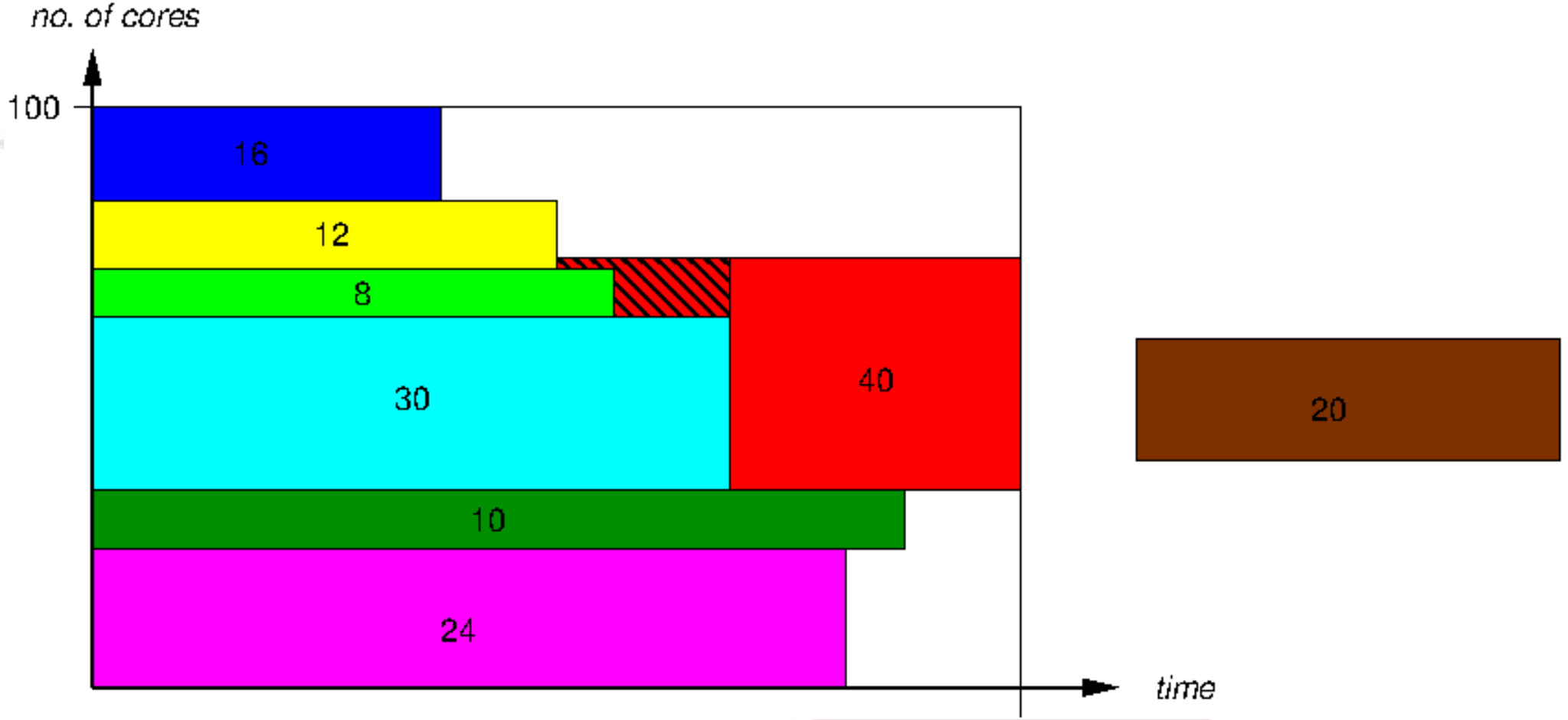
- scheduler knows which jobs will finish next (because of walltime specifications)
- scheduler will reserve those cores for your job until it has enough resources to run your job.

How Jobs Get Scheduled
an example



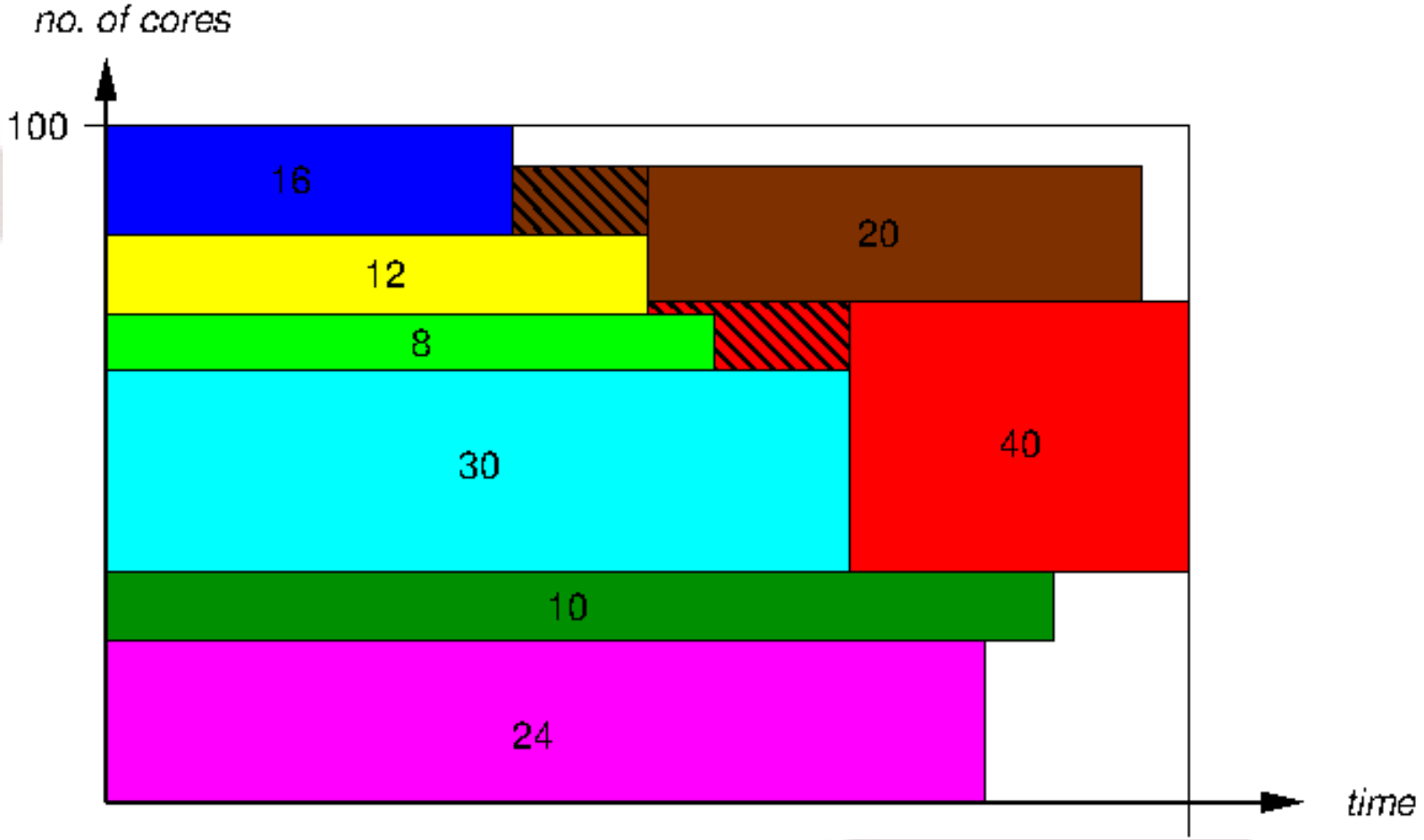
- find the earliest time 40p job can run

How Jobs Get Scheduled
an example (cont'd)



- reserve resources for highest priority job while minimizing waste

How Jobs Get Scheduled
an example (cont'd)

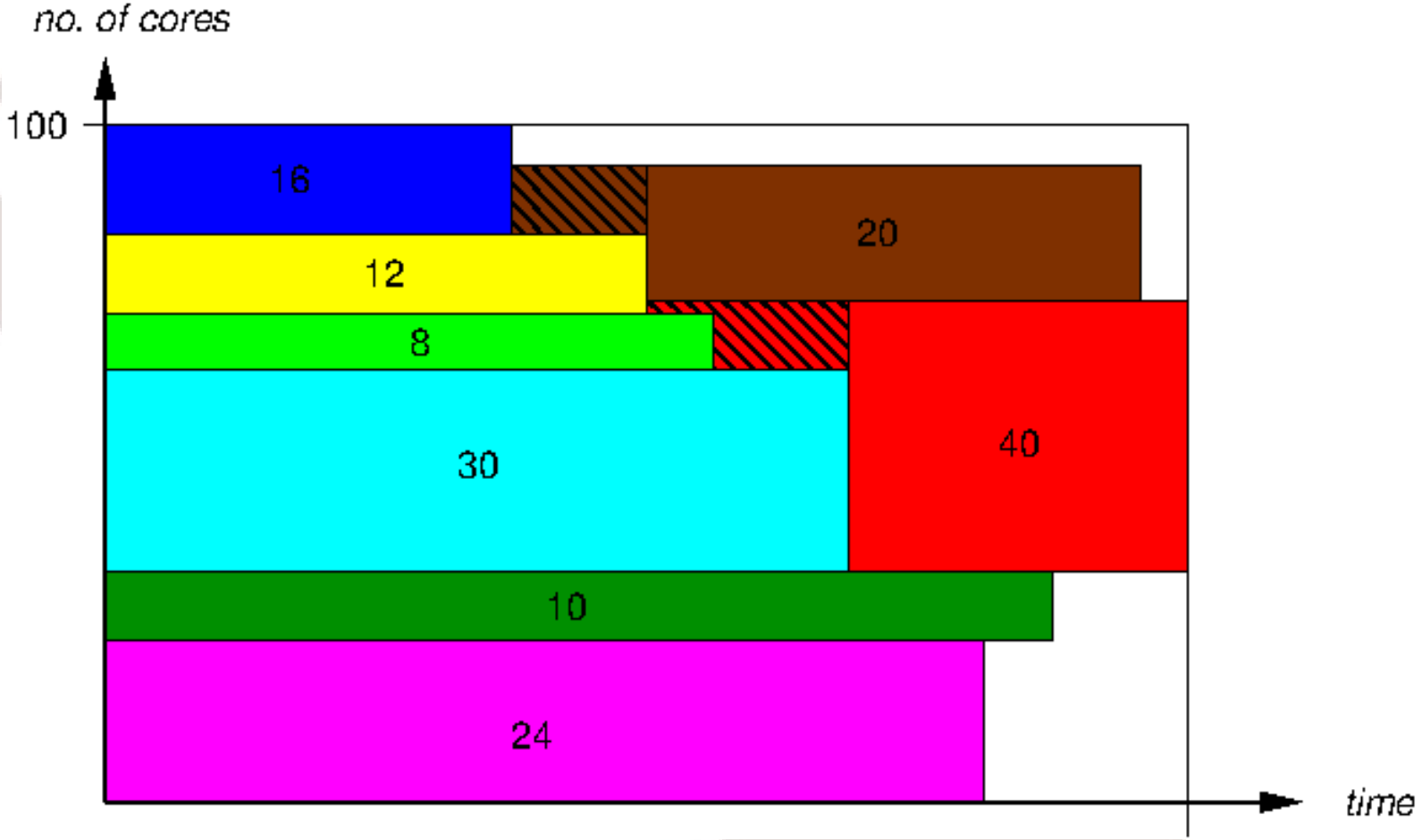


- continue with other jobs in priority order

How Jobs Get Scheduled
an example (cont'd)

- the whole procedure is redone in every iteration (every minute or so)
- a job can lose its reservation, if a higher priority job gets submitted in the meantime
- a job can start earlier, if a job ends earlier than specified
- the “showstart” command gives a startime *estimate* obtained in the last iteration

Backfill



- done after scheduling of high-priority jobs
- first fit

Backfill
showbf

```
siegert@bugaboo:~> showbf
Partition     Tasks  Nodes      Duration   StartOffset       StartDate
---------     -----  -----  ------------  ------------  --------------
ALL              86     50      22:25:56      00:00:00  15:19:40_11/13
ALL              47     18      22:56:41      00:00:00  15:19:40_11/13
ALL               4      1   1:17:29:53      00:00:00  15:19:40_11/13
siegert@bugaboo:~>
```

- shows what kind of jobs can start immediately
- e.g., a job that uses (at most) 86p for at most 22:25:56 can start immediately (those 86p will get distributed over 50 nodes).
- applicable to `procs` requests

Getting More out of the System
jobs in the queue

- have at least one job in the queue - always
- priority increases only so much – difficult to catch up
- “stuffing” the queue does not give an advantage
- if member of AG with RAC allocation: submit jobs on the system with allocation (predominantly)

Getting More out of the System
eliminate constraints

- using -l procs=N is better than -l nodes=n:ppn=m
- many small jobs are easier to schedule than one big large job
- use checkpoint/restart: take advantage of shorter walltimes
- do not hardcode no. of processors
- use the system that is dedicated to your type of jobs
- MPI jobs easier to schedule than OpenMP
- be flexible

Getting More out of the System
fairshare, priority

- fairshare priority: `jobinfo -f`

```
siegert@bugaboo:~> jobinfo -f
Share Tree Overview for partition 'ALL'
Name                    Usage    Target                  (FSFACTOR)
----                    -----    ------                 ------------
- westgrid            2994.22   3216.00  of  3376.00 (node: 1463032152.24) (0.00)
  - sie-tub-000-      101.81     64.00  of  3216.00 (acct: 46334629.40) (2043.53)
    - user1            23.15     12.50  of   100.00 (user: 10728184.93) (2043.35)
    - user2             0.79     12.50  of   100.00 (user: 365404.66) (2043.68)
    - user3             0.00     12.50  of   100.00 (user: 0.00) (2043.69)
    - siegert           0.00     12.50  of   100.00 (user: 0.00) (2043.69)
    - user5             1.17     12.50  of   100.00 (user: 540688.58) (2043.67)
    - user6            28.49     12.50  of   100.00 (user: 13201038.01) (2043.27)
    - user7             4.86     12.50  of   100.00 (user: 2253318.48) (2043.62)
    - user8            41.55     12.50  of   100.00 (user: 19245994.74) (2043.07)
siegert@bugaboo:~>
```

- rank in input queue: `showq -i`

Getting More out of the System
resource specification

Specify what you really need.
Benefits overall efficiency and fairness.

- walltime: `-l walltime=2928:00:00`

  Do you really need that much??

  (overestimating a little bit is recommended, but specifying three times as much is not)

- memory: `-l pmem=2500mb` or `-l mem=160gb`

  Specify this! We are losing too many jobs because of users not specifying their memory requirements! Particularly important when using `-l procs=...`

Know Your Job Requirements

checkjob -v

```
siegert@bugaboo:~> checkjob -v 4703493
job 4703493 (RM job '4703493.b0')

AName: badexample
State: Running
Creds:  user:siegert  group:siegert  account:sie-tub-000-aa  class:qs  qos:qoss
WallTime:   1:03:18:13 of 7:00:00:00
SubmitTime: Sun Nov 13 15:10:19
  (Time Queued  Total: 1:20:50  Eligible: 1:19:47)

StartTime: Sun Nov 13 16:31:09
NodeMatchPolicy: EXACTNODE
Total Requested Tasks: 32
Total Requested Nodes: 26

Req[0]  TaskCount: 32  Partition: base
Memory >= 256M  Disk >= 0  Swap >= 0
Dedicated Resources Per Task: PROCS: 1  MEM: 256M
Utilized Resources Per Task:  PROCS: 0.99  MEM: 1621M  SWAP: 58G
Avg Util Resources Per Task:  PROCS: 0.99
Max Util Resources Per Task:  PROCS: 0.99  MEM: 1621M  SWAP: 58G
Average Utilized Memory: 1588.58 MB
Average Utilized Procs: 31.31
NodeCount:  26
...

siegert@bugaboo:~>
```

Getting More out of the System

know what is available: `jobinfo -n`

```
siegert@bugaboo:~> jobinfo -n
compute node summary
Name                      State   Procs      Memory          Opsys

...
b196                       Busy    0:12     1976:24096       linux
b197                       Busy    0:12    12756:24096       linux
b198                    Running    1:12       84:24096       linux
b199                       Busy    0:12     4472:24096       linux
b200                    Running    7:12     3000:24096       linux
b201                    Running    1:12     2856:24096       linux
b202                    Running    6:12    17952:24096       linux
b203                    Running    2:12     1768:24096       linux
b204                    Running    6:12    17952:24096       linux
b205                    Running    4:12      768:24096       linux
b206                    Running    1:12      248:24096       linux
b207                       Busy    0:12    12448:24096       linux
b208                       Busy    0:12     1976:24096       linux
...
```

Getting More out of the System

know what is available: `jobinfo -n`

```
siegert@bugaboo:~> jobinfo -n | awk '$3 !~ /^0:/ {print $0}'
compute node summary
Name                      State   Procs      Memory         Opsys

b38                     Running    4:8        49:16049      linux
b157                    Running    1:8     10501:16049      linux
b174                    Running    1:12       72:24096      linux
b176                    Running    1:12       84:24096      linux
b177                    Running    1:12       84:24096      linux
b181                    Running    4:12        0:24096      linux
b183                    Running    4:12        0:24096      linux
b190                    Running    4:12        0:24096      linux
b198                    Running    1:12       84:24096      linux
b205                    Running    1:12        0:24096      linux
b206                    Running    1:12      248:24096      linux
b226                    Running    1:12       84:24096      linux
b258                    Running    3:12       20:24096      linux
-----                       ---   27:3376 3576790:6776638     -----

Total Nodes: 334  (Active: 319  Idle: 0  Down: 15)

siegert@bugaboo:~>
```

Problems
blocked jobs

"Why is my job not running?"

- Do you have blocked jobs? The scheduler does not look at blocked jobs.
  - `showq -b` lists blocked jobs
  - `checkjob -v <jobid>` indicates the problem of the blocked job
  - "violates idle HARD MAXIJOB limit of 10" is harmless
- requesting whole nodes `-l nodes=8:ppn=12` can increase waiting times dramatically; if you can use `-l procs=96` instead.

Feedback

The scheduling software is a ***very*** complicated piece of software: many, many parameters, many, many different options.

Please, email us your suggestions, criticism!!

<support@westgrid.ca>

- Are your jobs not well served with current policies?
- Do you believe that you are not getting your fair share?
- Recommend a different policy?
- ...

Thank You!

Questions?