Batch Strategies for Maximizing Throughput and Allocation

Richard Gerber

NERSC User Services

NERSC Users' Group Monthly Webinar
October 4, 2012

Scope

I’m assuming you know how us the batch system to run jobs.

I’ll concentrate on Hopper, but most items apply to Carver as well.

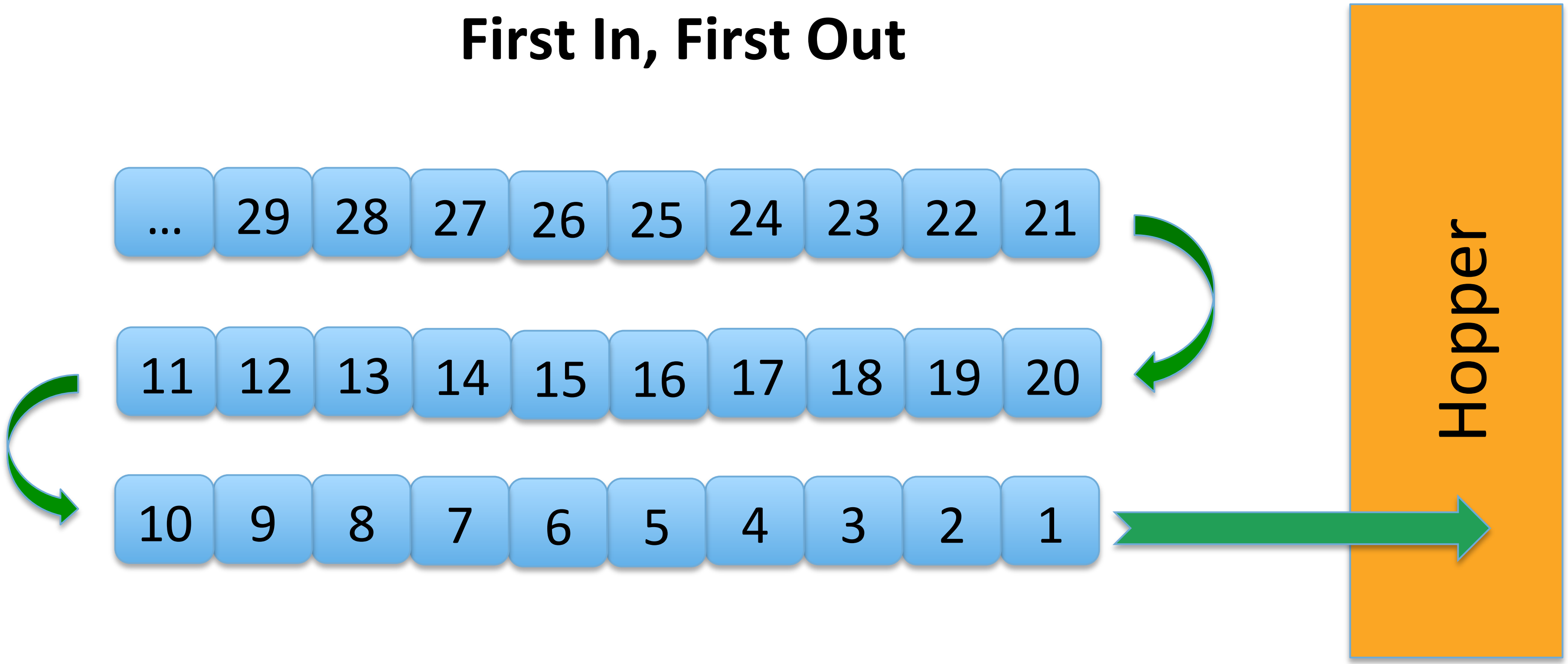
https://www.nersc.gov/users/computational-systems/hopper/

https://www.nersc.gov/users/computational-systems/carver/

Throughput

NERSC Queues are FIFO

First In, First Out



So Get in Line Early

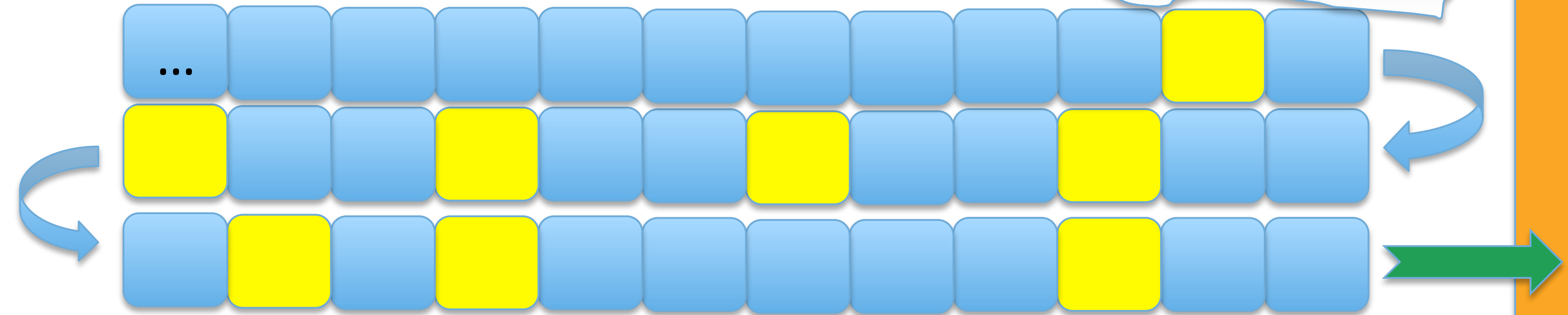
Fill all your eligible queue positions

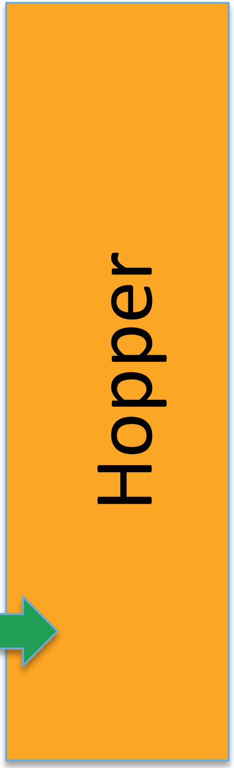
(queue-dependent limits)

Jobs waiting to get in the eligible state (Blocked)

Job will fall into eligible line when one of yours starts running or is deleted.

Jobs waiting in the queue + eligible to run (Limit: 8 max for regular queue)





(Those are your jobs in yellow)

FIFO, But …

NERSC queues are FIFO … but with exceptions …

… which makes things complicated and interesting.

Exception #1: Charge Classes

Premium: Jump ahead in queue for 2X the cost (+2-day boost).

`%qsub –q premium`

(don't let the cost catch you unaware)

Regular: Just what it says.

`%qsub –q regular`

Low: Let others pass you for ½ the cost (-3 days)

`%qsub –q low`

Exception #2: Code Development

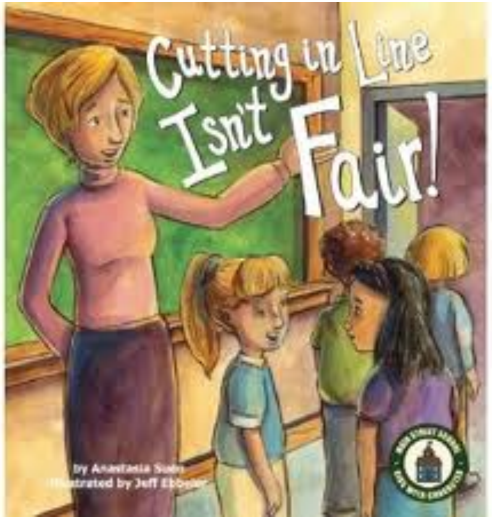
You need fast turnaround for debugging and development.

Interactive: Jump ahead for small short jobs.
30 minute max, 256 node max

```
%qsub –q interactive
```

Debug: Jump ahead for small short jobs.
30 minute max, 512 node max

```
%qsub –q debug
```



Bad things will happen to you if you try to do production runs in these queues!

Exception #3: Big Hopper Jobs

(16K cores)
^
Jobs that use more than 682 nodes get to cut in line. (1-day boost)

~~LONG WAIT~~

They are really hard to schedule otherwise.

No interdependencies, though.

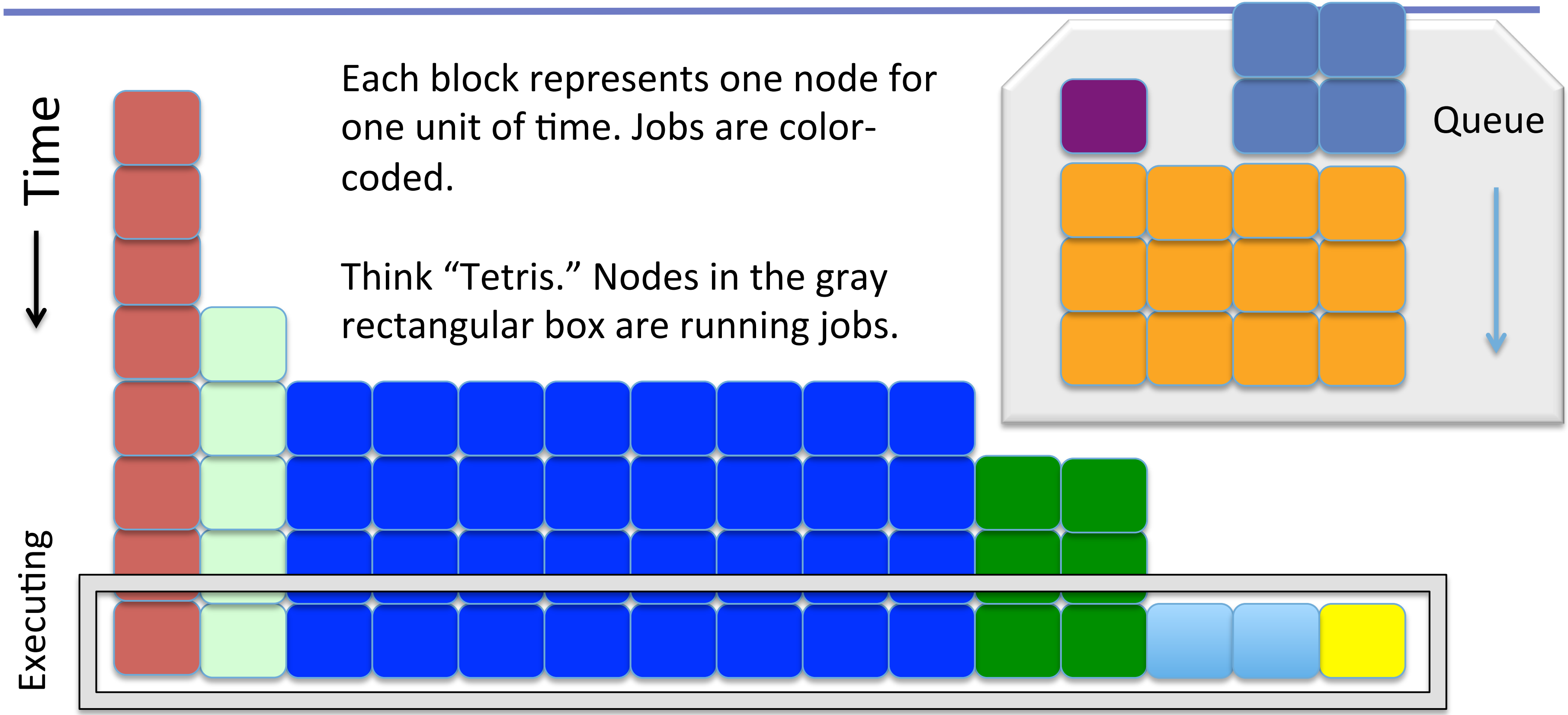
You can bundle similar smaller jobs into a single batch script to take advantage of this.

And get more “jobs” in the queue, too!

Exception #4: Backfill

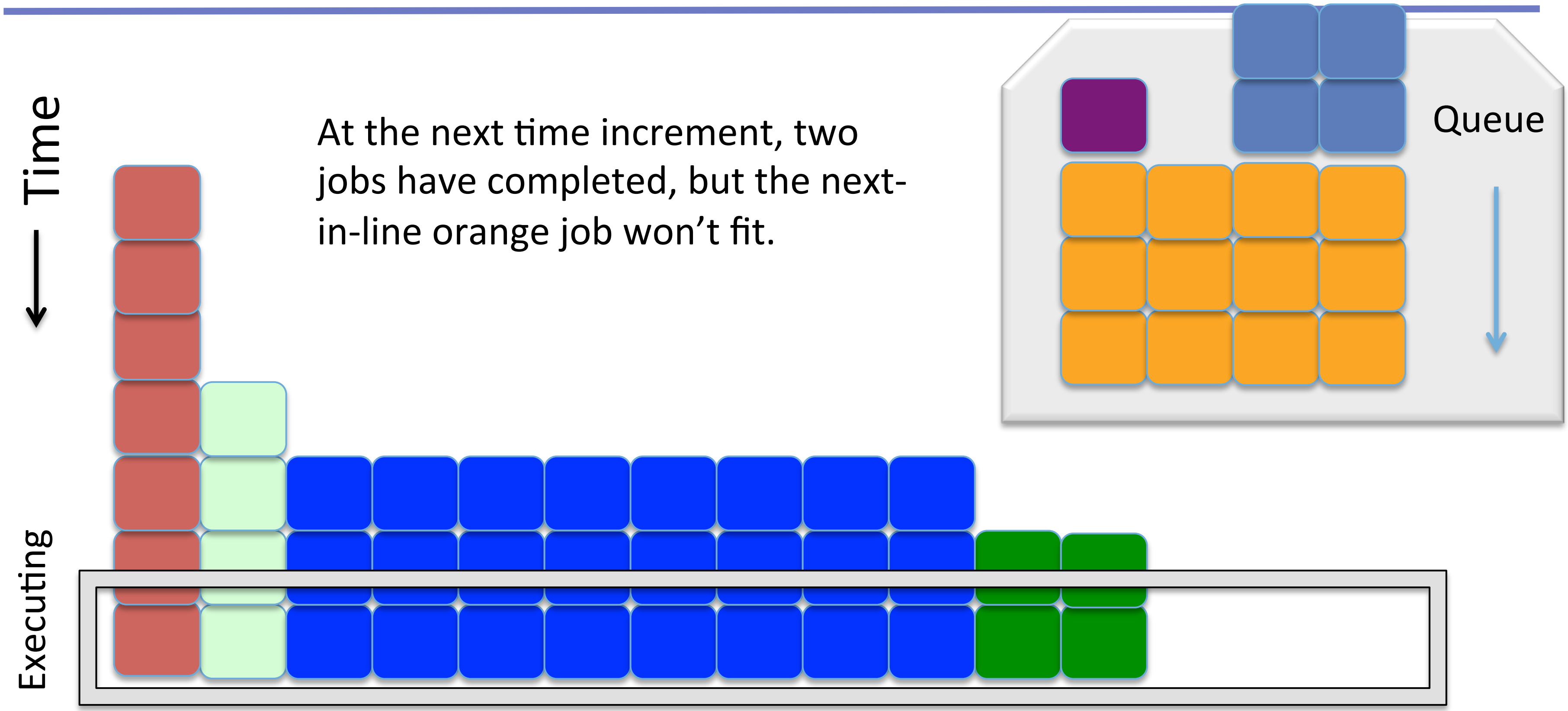
Each block represents one node for one unit of time. Jobs are color-coded.

Think "Tetris." Nodes in the gray rectangular box are running jobs.



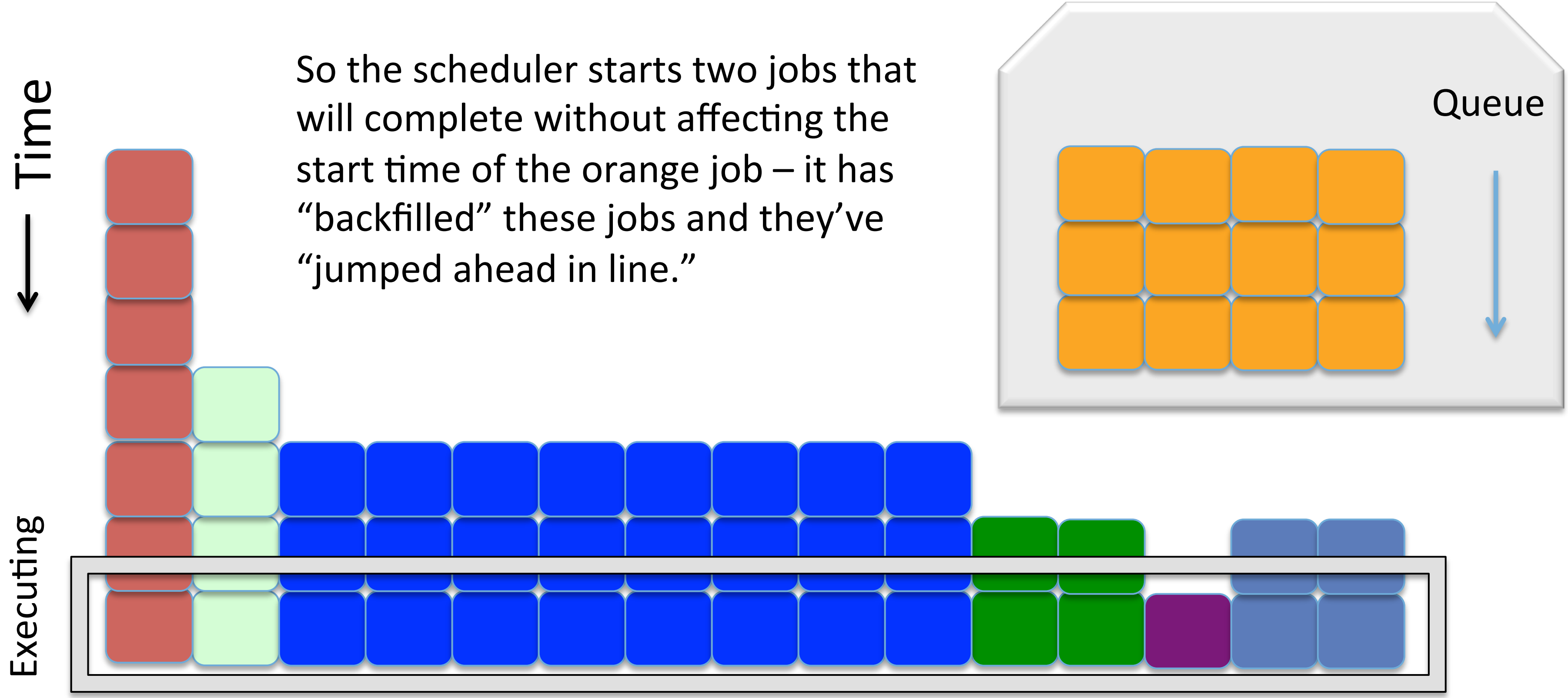
A Little Bit Later

At the next time increment, two jobs have completed, but the next-in-line orange job won't fit.



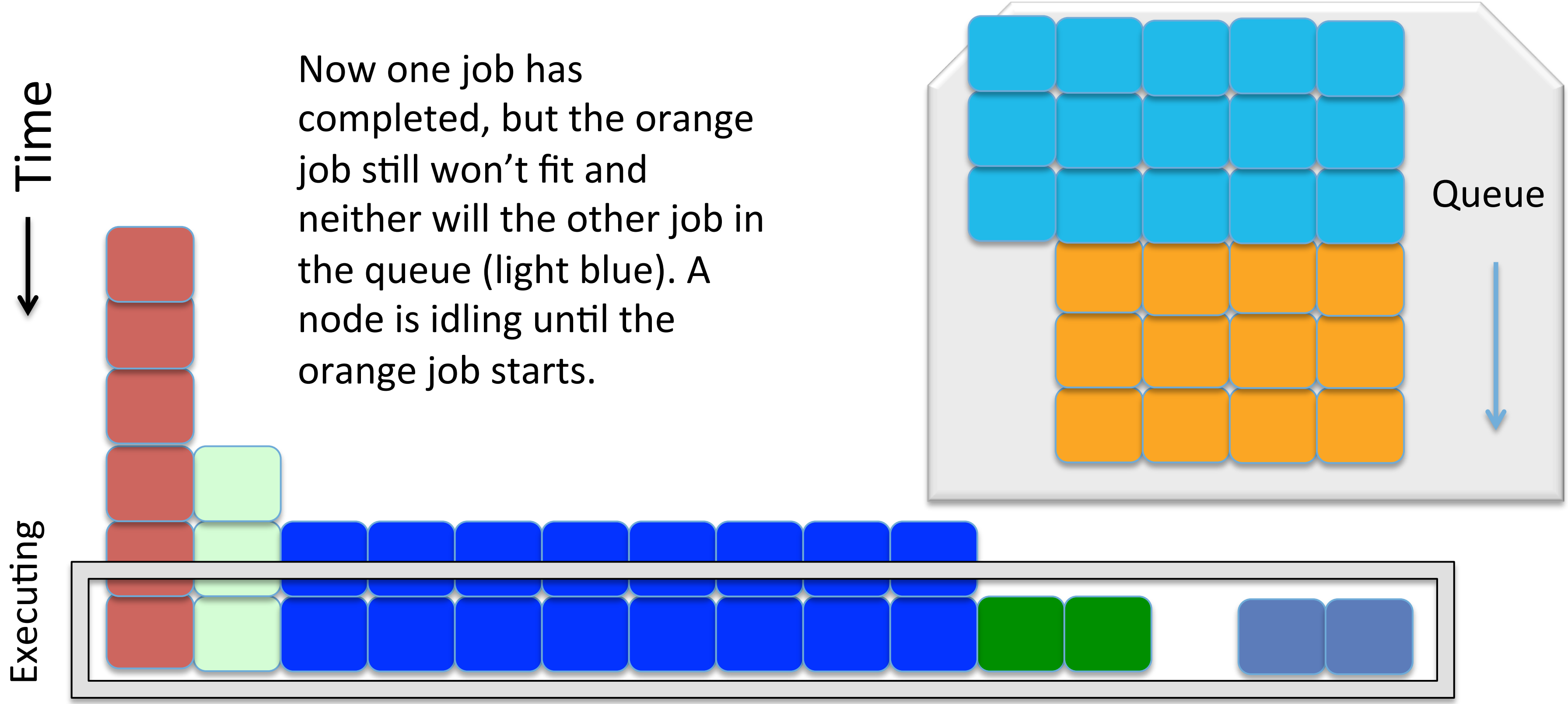
A Little Bit Later - continued

So the scheduler starts two jobs that will complete without affecting the start time of the orange job – it has “backfilled” these jobs and they’ve “jumped ahead in line.”

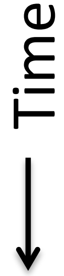


Two Little Bits Later

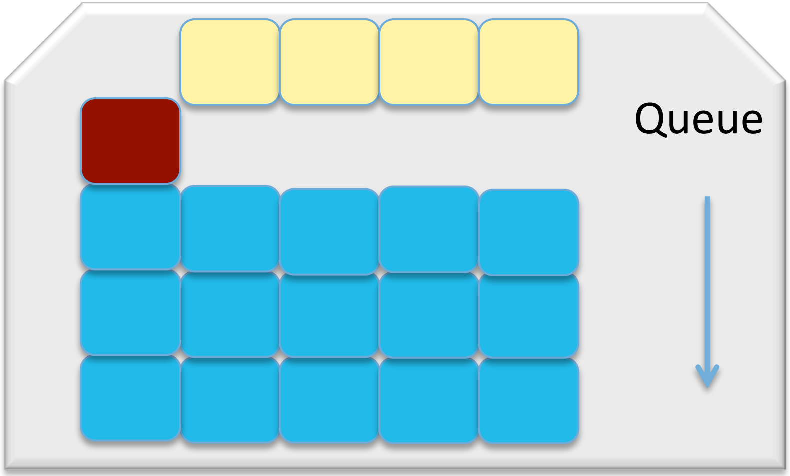
Now one job has completed, but the orange job still won't fit and neither will the other job in the queue (light blue). A node is idling until the orange job starts.

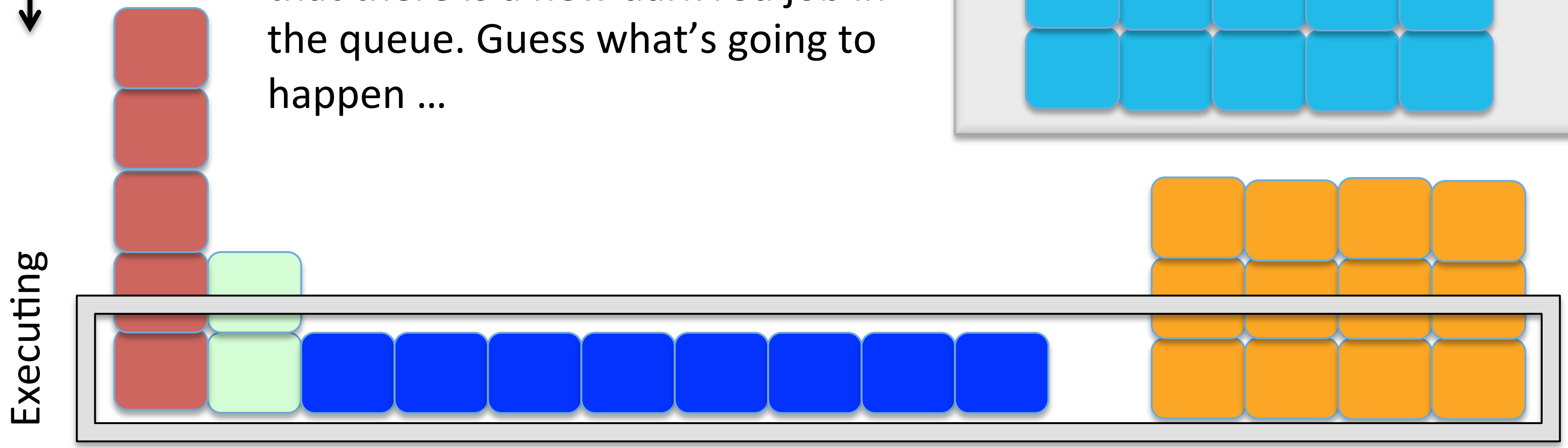


And Three Little Bits Later

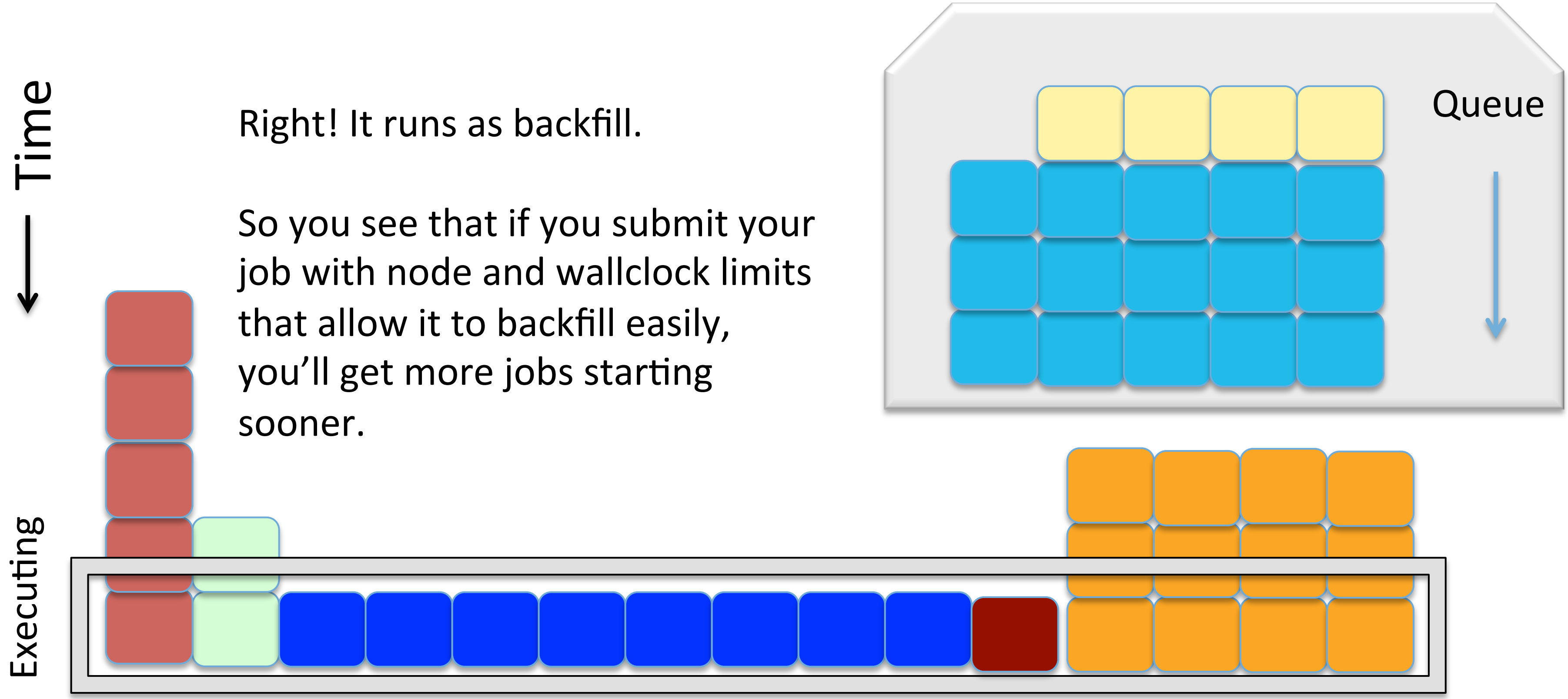


Now two jobs have finished and the orange one is running. Notice that there is a new dark red job in the queue. Guess what's going to happen …





And a Couple of Little Bits Later Still



Right! It runs as backfill.

So you see that if you submit your job with node and wallclock limits that allow it to backfill easily, you'll get more jobs starting sooner.

Exception #5: Special Priority Boosts

NERSC

Jobs deemed high priority by DOE

Projects with urgent needs
(let us know if this is you!)

System testing

These are all pretty rare

Wallclock Request Time is Important

- The shorter your wallclock request, the more likely you'll be eligible for backfill.
- If you request a wallclock much longer than your actual, it can cause scheduling havoc and great angst for other users.
- If there is a scheduled downtime sooner than your job would end based on requested walltime, your job will not start until after the maintenance completes.

A Few Throughput Tips

- Maintain the max number of schedulable jobs in the queue.
- Estimate your job's wallclock time accurately (with an adequate buffer, say + 10%).
- Run "big" batch jobs (bundle jobs if possible).
- Take advantage of backfill opportunities.
- Requests for very long wallclock times may be convenient, but may limit your throughput.
- Debug your batch scripts! (script errors kill your throughput)
- Putting your jobs on user hold does not help you.

“High-Throughput” Jobs

- The term “High-Throughput” has come to be associated with workflows that use serial or very low concurrency codes, often running for days, weeks, or longer.
- NERSC has special queues on Hopper and Carver to accommodate these jobs: please see the NERSC website or contact the consultants (consult@nersc.gov) for more information.

Stretching Your Allocation

Your Best Value: Big Jobs on Hopper

NERSC

Hopper jobs that use > 682 nodes (16K cores) get charged 0.6 the regular rate.

"We got a 14.5 Million Hour bonus in 2012 by running big jobs!"

-Repo m1383

Use Low Priority Queue

Low priority jobs are charged at ½ the regular rate.

Wait times can be very long for a single job, but if you keep a steady stream of jobs queued, you win!

Testimonials

"I 've run 310 12K-core jobs through low this year, saving me 3.7 Million Hours in charges!"

-Unnamed NERSC user

Run Lean (Optimize Your Codes)

Try different compilers (improve performance and shorten your run time)

CRAY

intel

GCC

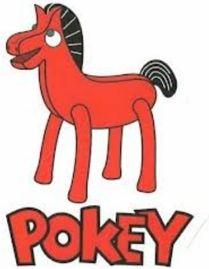
You might gain 25% by switching compilers

Read/write files in $SCRATCH

Use math and I/O libraries

Home directories

Numerical Recipes



Notes on Charging

- **You are charged for all nodes allocated to your job for the full wall time, whether you use them all or not**
- **You are charged for all cores on a node, whether you use them or not.**
  - Exception: Serial queue on Carver
  - Don't do long (serial) builds, file xfers in batch
- **Carver jobs are charged at a rate 1.5X Hopper jobs**

Summary

- Run big jobs on Hopper
- Use the low queue if you can
- Optimize your code's performance
- Don't run serial work in a parallel batch script