PHYSICS & ASTRONOMY
TEXAS A&M UNIVERSITY

High Performance Computing: Concepts and Best Practices

James Gerity

August 12, 2016

Slides available at http://www.github.com/jgerity/talks

What is HPC?

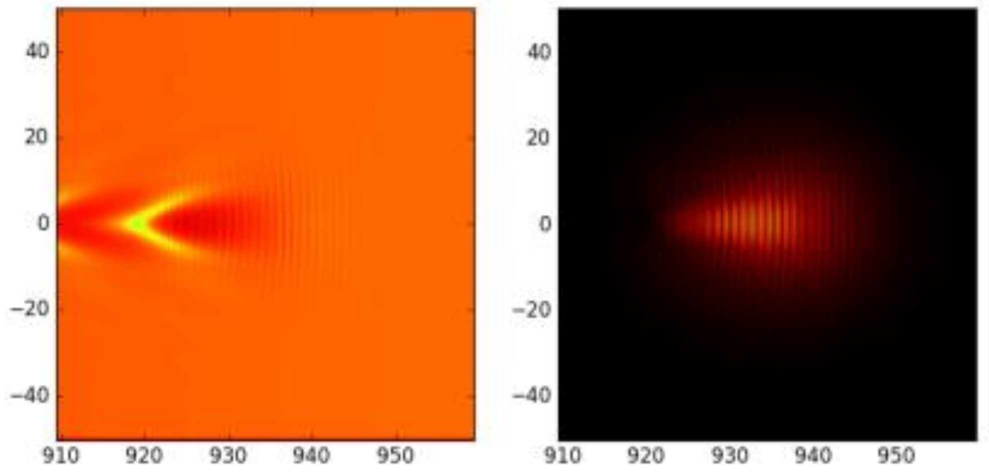
- **High performance computing (HPC)** is a catch-all term for computing at greater scales than what is realized by a typical workstation.
- Some problems would require tens of thousands of hours (or years!) on a single machine
  - With more computing power and memory, perhaps we can do better.

Example problem: Particle-in-cell

- Simulating the time evolution of a plasma is a demanding challenge:
  - N^2 interactions to calculate in the worst case
  - Number density of $10^{25}\ \mathrm{m}^{-3}$ not at all uncommon
  - Small timestep, mesh is necessary
- Linearity of Maxwell’s equations lets us apply the “divide and conquer” strategy

Why use parallel architecture?

- **Recent example:** Laser-wakefield simulation to interpret experiments at LBNL.

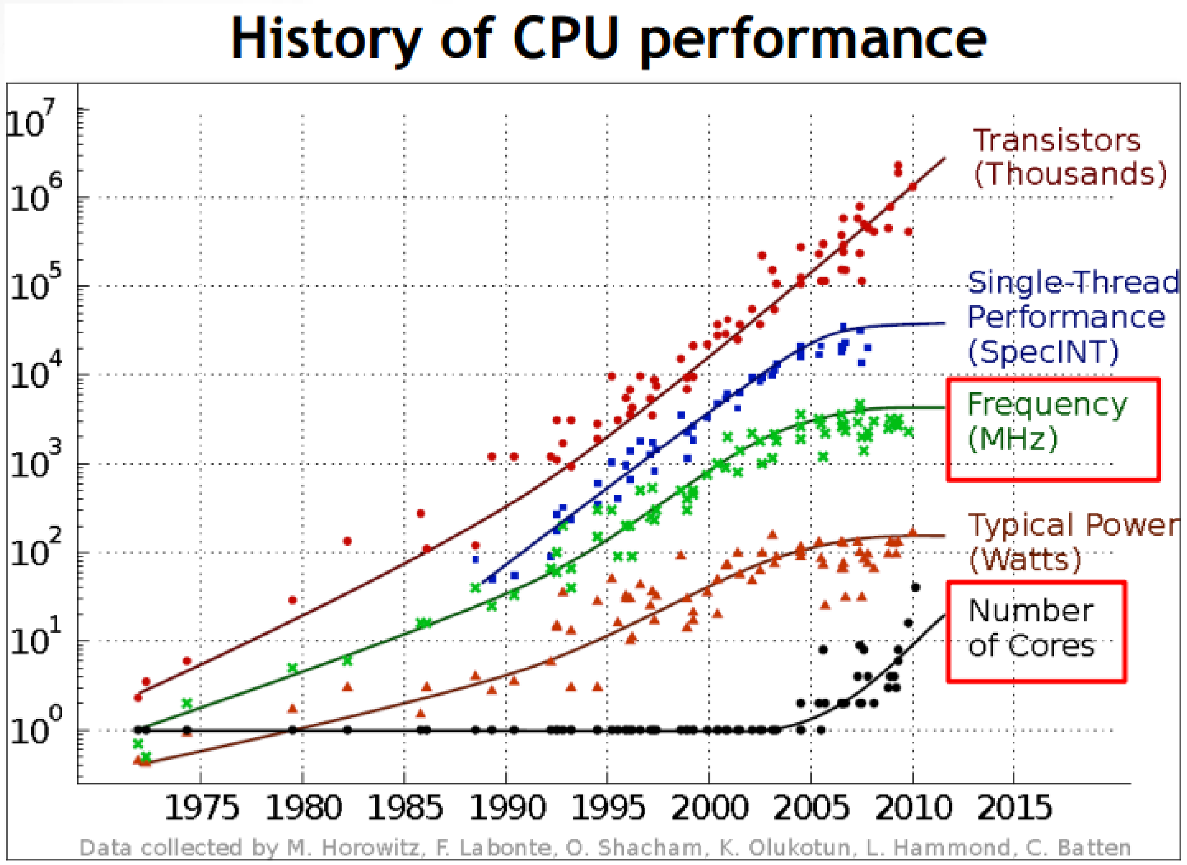


3D grid with 2500 x 200 x 200 grid points
0.2 billion macroparticles
140,000 timesteps (Courant limit!)

~60,000 hours on 1 core = 7 years!
=> Need either <u>faster cores</u> or <u>more cores in parallel</u>

Slide courtesy of Remi Lehe

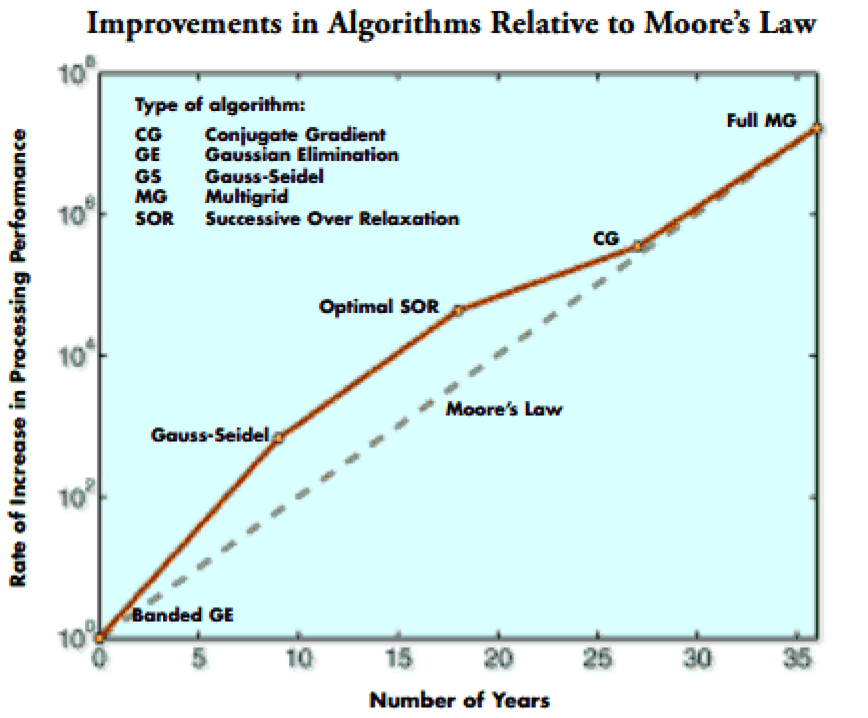
Can we just wait on faster processors?



PHYSICS & ASTRONOMY
TEXAS A&M UNIVERSITY

What problems does HPC *not* solve?

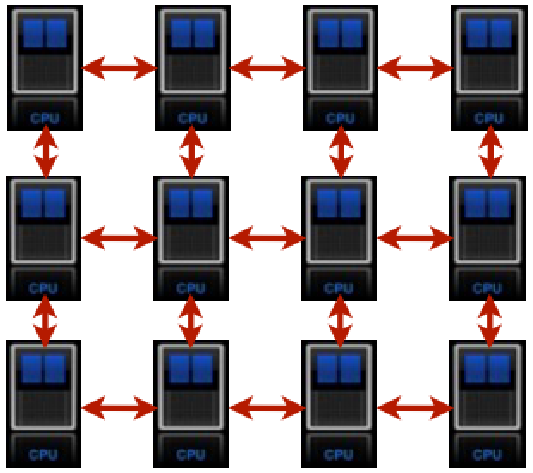
- If the problem requires sharing information between CPUs, we have to communicate, which is slow!
- Better algorithms yield more gain than Moore's Law does



The relative gains in some algorithms for the solution of an electrostatic potential equation on a uniform cubic grid compared to improvements in the hardware (Moore's Law).

From 2005 report *"Computational Science: Ensuring America's Competitiveness"*

What is a cluster?



Individual nodes have several cores (i.e. computing units):

- "Traditional" CPUs: ~10 cores

- Xeon Phi: 68 cores

- GPUs: ~1000 (slow) cores

Cores within one node <u>share memory</u>

- How to use this architecture to make a PIC code faster?
- How to use the two levels of parallelism?
  (within one node and between nodes)

Slide courtesy of Remi Lehe

What is a cluster?

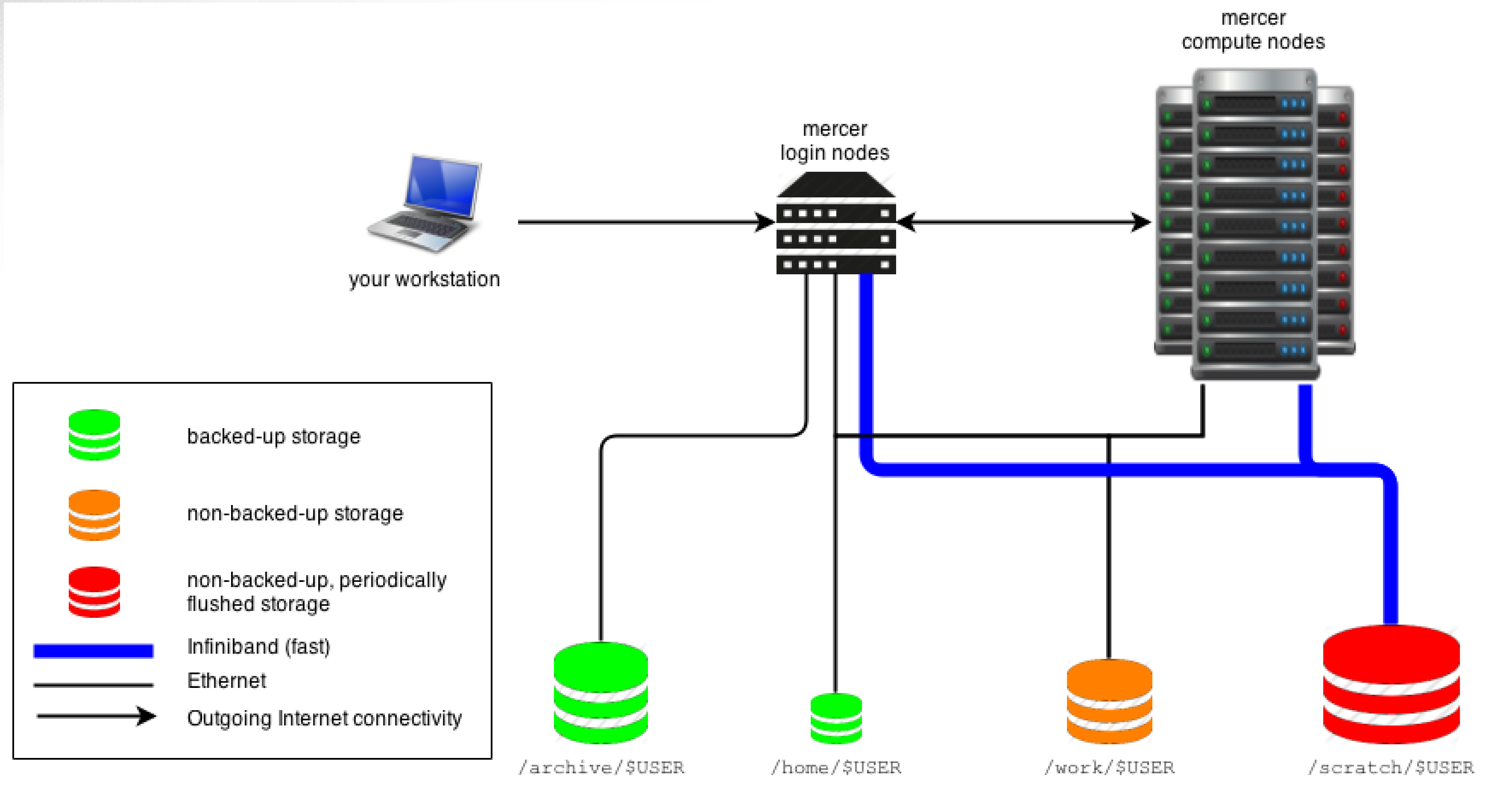


Image modified from https://wikis.nyu.edu/display/NYUHPC/High+Performance+Computing+at+NYU

Cluster layout

- Login nodes
  - Main form of user interaction with the system
  - Allows some development, as well as submitting jobs
- Compute/worker nodes
  - Computational work done here, generally less aware of the broad view of how a job is split up.

Cluster layout

- Storage
  - At large machines, usually have separate facilities that can be accessed from many places
  - Networked file systems like AFS, NFS
  - Important to understand what is available to compute nodes and how fast it is
    - If you bottleneck at filesystem I/O, potentially several orders of magnitude slower!

Schedulers

- From the login node, jobs are submitted to a **scheduler**, software that handles the logistics of balancing many jobs from different users across nodes.
  - Knowing the ins and outs of the scheduler you’re using is an absolute must!
- Examples of schedulers: SLURM, TORQUE, HTCondor

How not to be “that jerk” on a shared system

- It is *your* responsibility to know the local machine’s policy. Read the MOTD and the documentation!
- Make sensible resource requests – does your job actually benefit from tying up several nodes? Benchmark it first!

How not to be "that jerk" on a shared system

- Don't use the login node for heavy computational work!
  - Best case scenario, you waste your own time
  - Worst case scenario, you crash the login node and waste *everybody's* time
- Ideally, develop locally and understand how things scale, so there aren't surprises when you go to bigger jobs.

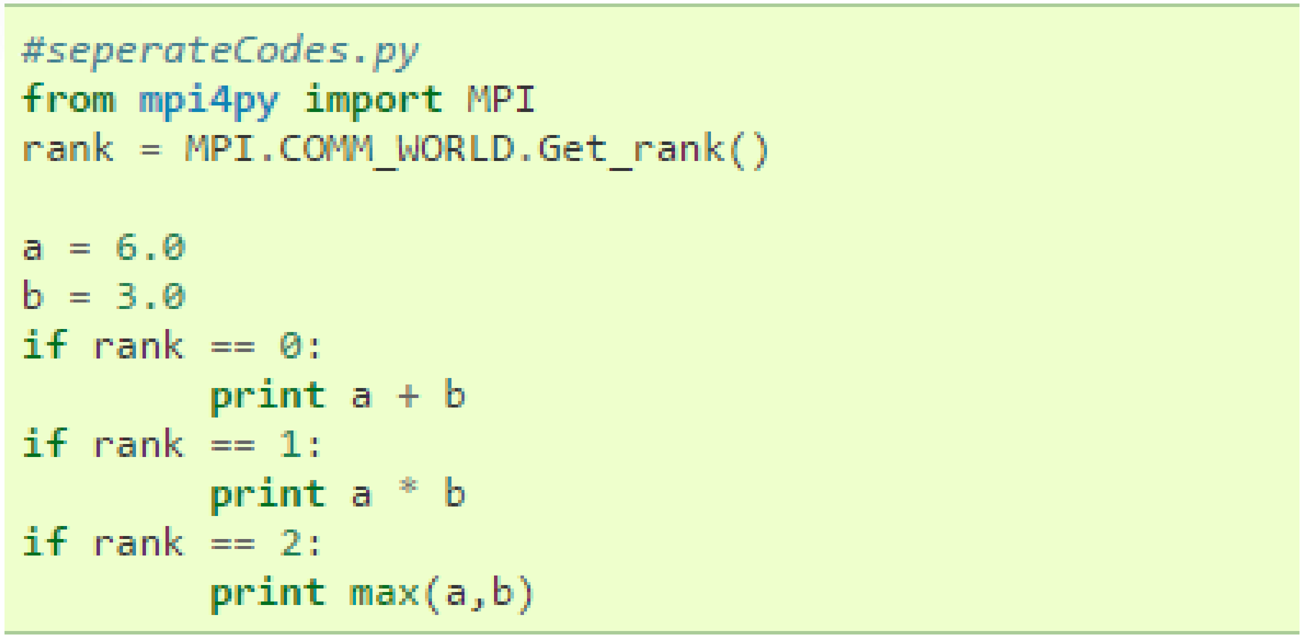
How not to be “that jerk” on a shared system

- Some remarks c/o Alexx Perloff:
- World-wide LHC Computing Grid (WLCG): This is a complicated beast, but we make use of it through a system called CRAB (CMS Remote Analysis Builder).
- There is an assumption that you will be submitting a certain type of job (I.e. Single core not requiring memory over 1.9 GB, and a job that won’t last more than 48 hours). Beyond that there is a priority that takes things into account similar to the LPC computing cluster, but it also takes into account where your requested data is, where the stage-in and stage-out are coming from, how many jobs are running on a particular
  cluster and how many open slots it has.
- Basically it is a world-wide balancing act across >100 computing clusters of various locations, sizes, throughput, and architecture.
  - The value of understanding the “right” way to submit jobs on a system like this cannot be overstated!

MPI

- **Message Passing Interface** (**MPI**) is a standardized and portable message-passing system
- Run MPI-aware program with “mpiexec” command and –np flag for number of processes to use
  - Tasks can be divided among processes of different “rank,” with a communication framework for exchanging information.

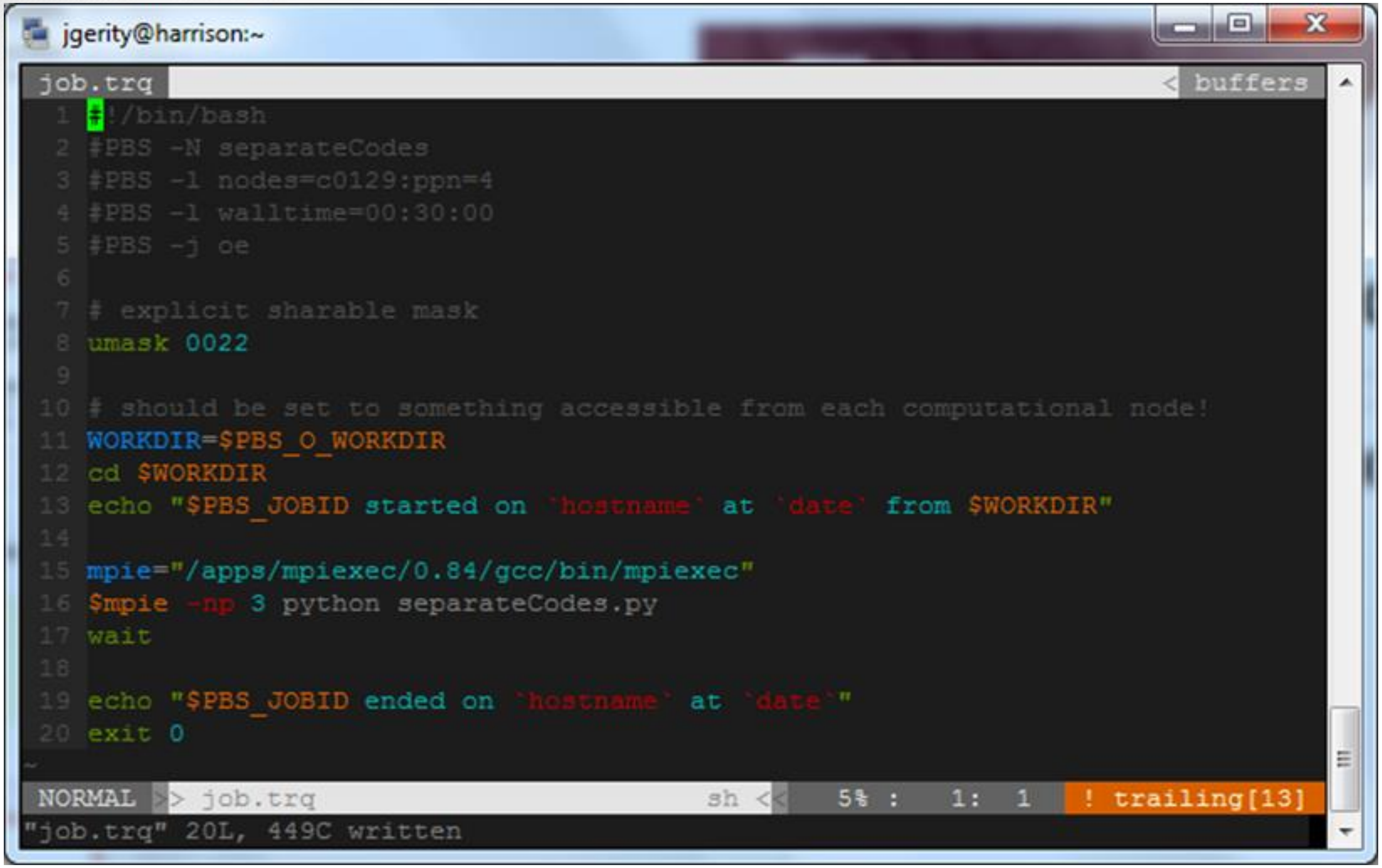
A simple MPI example in Python



```
#seperateCodes.py
from mpi4py import MPI
rank = MPI.COMM_WORLD.Get_rank()

a = 6.0
b = 3.0
if rank == 0:
        print a + b
if rank == 1:
        print a * b
if rank == 2:
        print max(a,b)
```

PHYSICS & ASTRONOMY
TEXAS A&M UNIVERSITY

A simple MPI example in Python

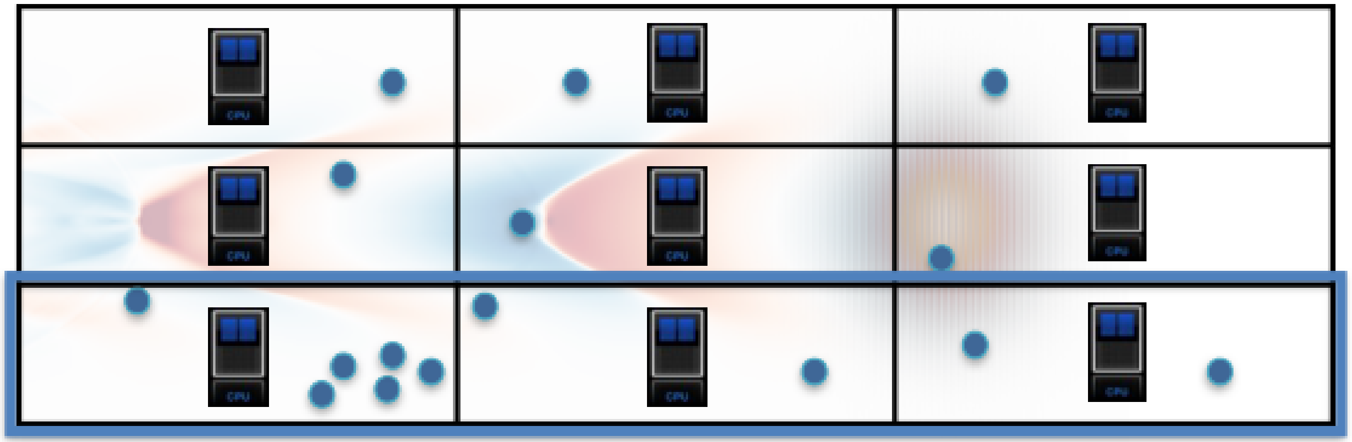


jgerity@harrison:~

job.trq

```
#!/bin/bash
#PBS -N separateCodes
#PBS -l nodes=c0129:ppn=4
#PBS -l walltime=00:30:00
#PBS -j oe

# explicit sharable mask
umask 0022

# should be set to something accessible from each computational node!
WORKDIR=$PBS_O_WORKDIR
cd $WORKDIR
echo "$PBS_JOBID started on `hostname` at `date` from $WORKDIR"

mpie="/apps/mpiexec/0.84/gcc/bin/mpiexec"
$mpie -np 3 python separateCodes.py
wait

echo "$PBS_JOBID ended on `hostname` at `date`"
exit 0
~
NORMAL >> job.trq                                sh <<   5% :   1:  1   ! trailing[13]
"job.trq" 20L, 449C written
```

Problem: load balancing



Example:
particle pusher,
schematic view

Push particle 1	Push particle 1	Push particle 1
Push particle 2	Push particle 2	Push particle 2
Push particle 3	Exchange particles ⟷	Exchange particles
Push particle 4	Wait	...
Push particle 5	Wait	
Push particle 6	Wait	
Exchange particles ⟷	Exchange particles	

The simulation will always progress at the pace of the slowest node (the one doing the most work)
=> Problematic when the particle distribution is very non-uniform

Slide courtesy of Remi Lehe

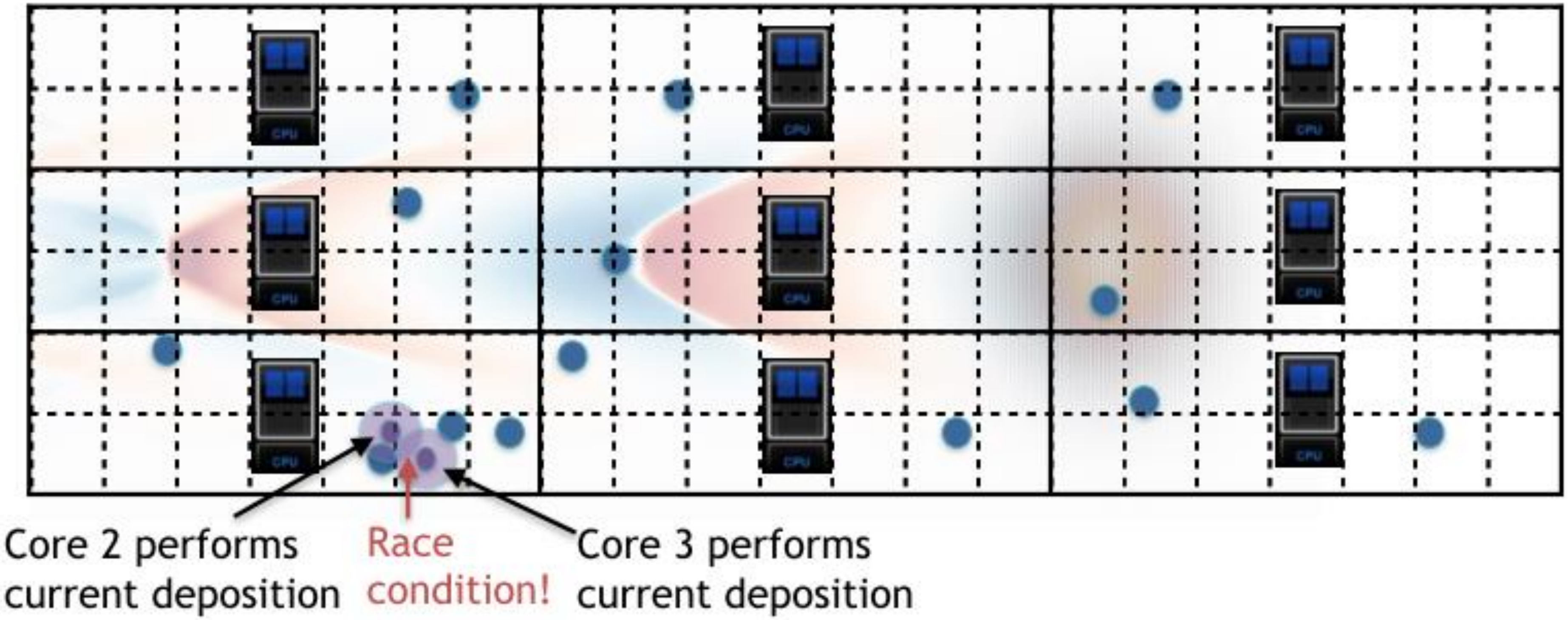
OpenMP

- “OpenMP Application Program Interface (API) is a portable, scalable model that gives parallel programmers a simple and flexible interface for developing portable parallel applications. OpenMP supports multi-platform shared-memory parallel programming in C/C++ and Fortran on all architectures”

OpenMP

- OpenMP allows separate threads to operate as fast as they can…
- …BUT introduces the possibility of a *race condition* in some cases

OpenMP's dangers: race condition


CPU
Core 2 performs current deposition
Race condition!
Core 3 performs current deposition

- The cores do not exchange information via **MPI send/receive**. Instead they **directly** modify the value of the current **in shared memory**, without notifying the other cores.

- Potentially, two cores could **simultaneously** try to modify the value of the current in a given cell (leads to inconsistencies). This can be avoided with proper care (e.g. "atomic operations").

Slide courtesy of Remi Lehe

GPGPU

- A video card, more generally known as a graphics processing unit (GPU), is a collection of *many* processors that share memory.
- Designed with computer graphics in mind, but there is substantial overlap with non-graphics computation!
- Increasingly common to utilize this hardware for computation (future topic, maybe…)

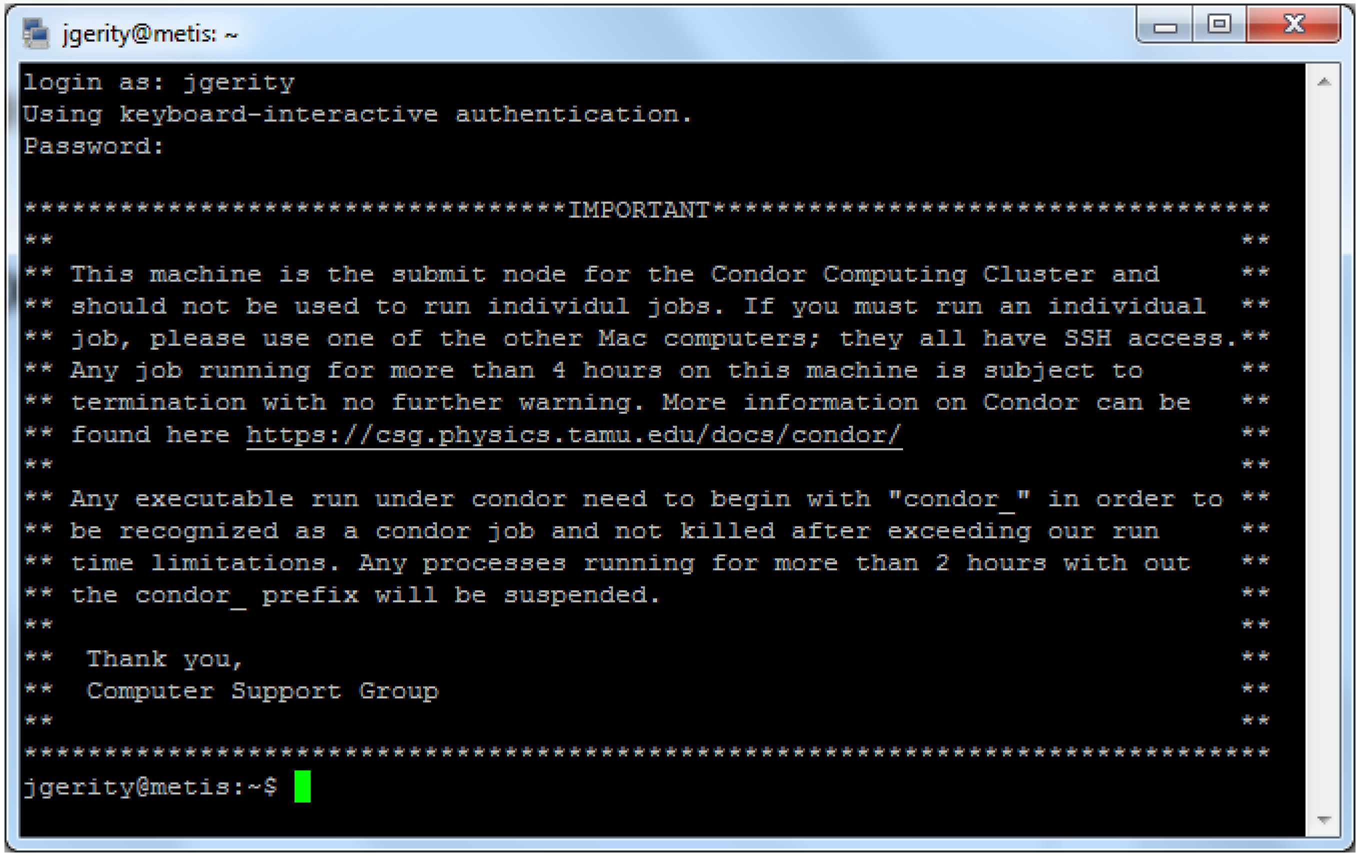
We have a small computational cluster!

- The 20 iMac workstations in MPHY 330 are networked in an HTCondor cluster, giving you access to 20 i5 quad-cores and 80 GB of RAM!
- Some more information available from https://csg.physics.tamu.edu/docs/condor/

Interactive Condor computing session

- Let’s log into the Condor cluster.
- Use your ssh client to log into metis.physics.tamu.edu with your department credentials
  - On Mac/Linux:
    - ssh username@metis.physics.tamu.edu
  - On Windows:
    - Use a terminal emulator like PuTTY

Interactive Condor computing session


jgerity@metis: ~

```
login as: jgerity
Using keyboard-interactive authentication.
Password:

*********************************IMPORTANT*************************************
**                                                                           **
** This machine is the submit node for the Condor Computing Cluster and      **
** should not be used to run individul jobs. If you must run an individual  **
** job, please use one of the other Mac computers; they all have SSH access.**
** Any job running for more than 4 hours on this machine is subject to      **
** termination with no further warning. More information on Condor can be   **
** found here https://csg.physics.tamu.edu/docs/condor/                      **
**                                                                           **
** Any executable run under condor need to begin with "condor_" in order to **
** be recognized as a condor job and not killed after exceeding our run      **
** time limitations. Any processes running for more than 2 hours with out    **
** the condor_ prefix will be suspended.                                     **
**                                                                           **
**  Thank you,                                                               **
**  Computer Support Group                                                   **
**                                                                           **
*******************************************************************************
jgerity@metis:~$
```

Submitting Condor jobs

- To run a job on metis, follow the CSG instructions:
  1. Create a working directory called **Condor** in your home directory
  2. Create a job file describing your task. See **fourier.job** for an example
  3. Submit the job using the **condor_submit** command

Thank you!