ECE 574 – Cluster Computing
Lecture 5

Vince Weaver

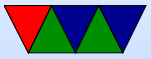
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

15 September 2015

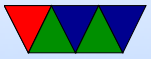
Announcements

- Don't forget about homework
- How is the homework going?



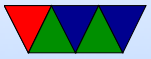
Cluster Layout

Draw diagram. Head/compile nodes, compute nodes.

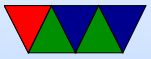


Job Schedulers

- On a big cluster, how do you submit jobs?
- If everyone just logged in to nodes at random, would be a mess
- Batch job scheduling
- Different queues (high priority, long running, etc)
- Resource management (make sure don't over commit, use too much RAM, etc)

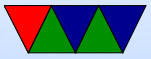


- Notify you when finished?
- Accounting (how much time used per user, who is going to pay?)

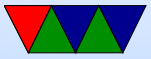


Scheduling

- Different Queues Possible – Low priority? Normal? High priority (paper deadline)? Friends/Family?
- FIFO – first in, first out
- Backfill – bypass the FIFO to try to efficiently use any remaining space
- Resources – how long can run before being killed, how many CPUs, how much RAM, how much power? etc.

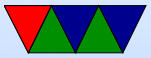


- Heterogeneous Resources – not all nodes have to be same. Some more cores, some older processors, some GPUs, etc.



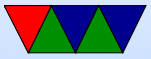
Common Job Schedulers

- PBS (Portable Batch System) – OpenPBS/PBSPro/TORQ
- nbs
- slurm
- moab
- condor
- many others



Slurm

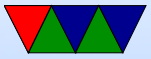
- `http://slurm.schedmd.com/`
- Slurm Workload Manager
  Simple Linux Utility for Resource Management
  Futurama Joke?
- Developed originally at LLNL
- Over 60% of top 500 use it



sinfo

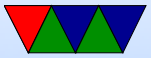
provides info on the cluster

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up   infinite      1   idle haswell
general*     up   infinite      1   idle haswell
```



`srun`

start a job, but interactively

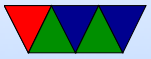


sbatch

submit job to job queue

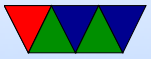
```
#!/bin/bash

#SBATCH -p general        # partition (queue)
#SBATCH -N 1                # number of nodes
#SBATCH -n 8                # number of cores
#SBATCH -t 0-2:00         # time (D-HH:MM)
#SBATCH -o slurm.%N.%j.out # STDOUT
#SBATCH -e slurm.%N.%j.err # STDERR
export OMP_NUM_THREADS=4
./xhpl
```

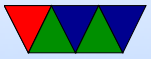


`scancel`

kills job

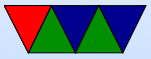


More Computer Architecture Review

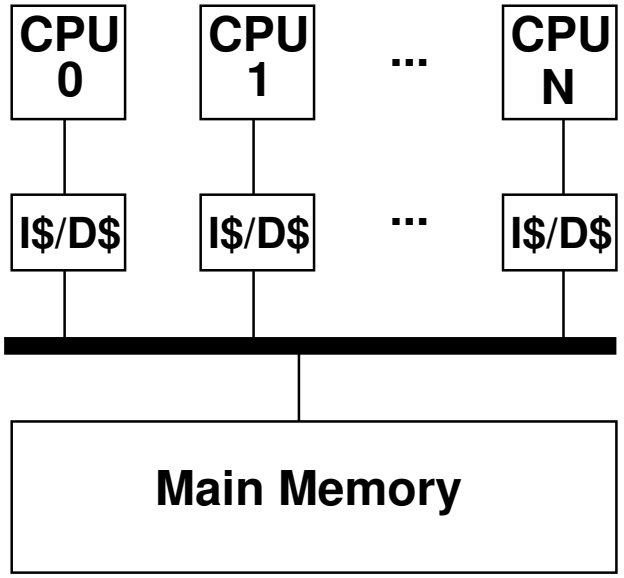


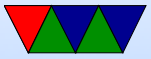
Multicore Systems

- CMP (Chip-multiprocessor) or SMP (Symmetric-multiprocessor)
- Multiple processors in system. More on this later.



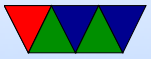
CMP Diagram





Hardware Multi-Threading

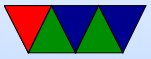
- Idea is to re-use a pipeline to execute multiple threads at once, *without* fully replicating the entire CPU (so less than multicore)
- You will have to replicate some things (program counter for each, etc)
- Usually they appear to the CPU as full separate processors even though they are not.
- Various ways to do this:



- Fine-grained – rotate threads every cycle
- Coarse-grained – rotate threads only if long latency event happens (cache miss)
- Simultaneous – issue from any combination of threads, to maximize use of pipeline (have to be superscalar)

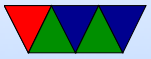
- Why do this? Often on superscalar running only one thread will leave parts idle, try to make use of these.

- Bad side effects?
  Can actually slow down code (especially if both threads

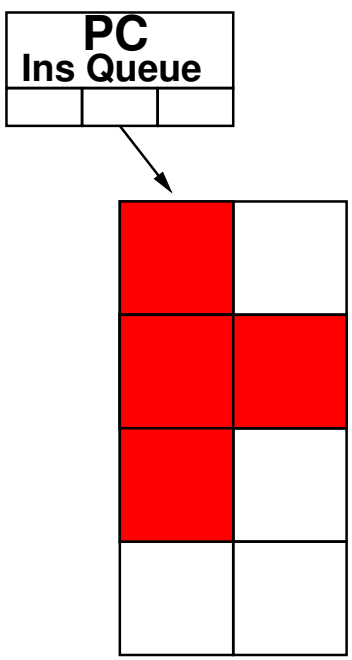


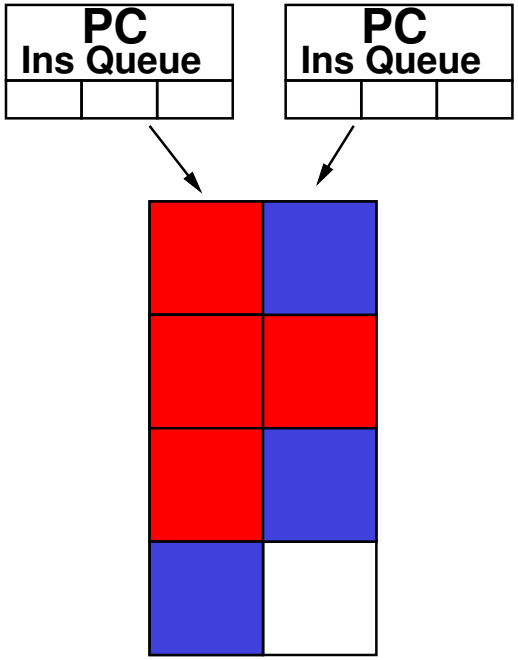
trying to use same functional units, also if both using memory heavily as cache is often shared)

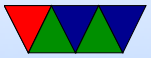
- Sometimes see it talked about as SMT (Simultaneous Multithreading), Intel Hyperthreading is more or less the same thing



SMT Diagram

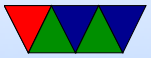






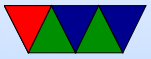
Cache Coherency

- How do you handle data being worked on by multiple processors, each with own cache of main memory?
- Cache coherency protocols.
- Many and varied. MESI is a common one
- Directory vs Snoopy



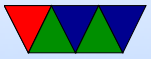
MESI

- Modified, Exclusive, Shared, Invalid



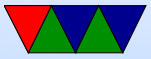
Barriers and Ordering

- On modern out-of-order execution, memory accesses can happen out-of-order
- Sequential consistency – all happen in order
- Strong consistency – stores
- Weak consistency – can be arbitrarily reordered, only barriers protect you
- A memory barrier instruction makes sure all previous



loads/stores finish before moving on

- Most important for things like locks, as well as memory-mapped I/O



Ordering Example

```
     y1=0
     y2=0
     y1=3
     y2=4
Another core
     x1=y1
     x2=y2
```

What values of x1 and x2 can you get?

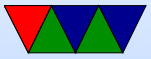
Strong:

x1=0,x2=0

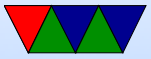
x1=3,x2=0

x1=3,x2=4

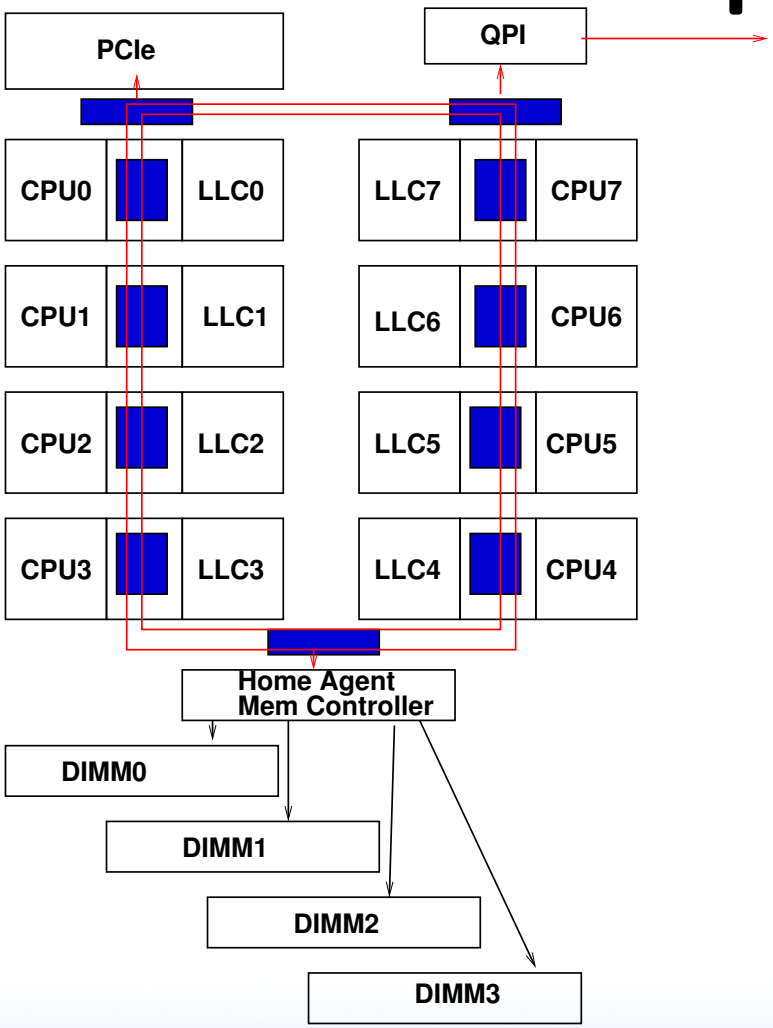
Weak:

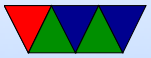


x1=0,x2=4



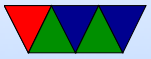
Haswell EP Setup





NUMA

Non-uniform memory access – some accesses will have to cross to other processors, causing extra delay. How can you optimize this?



Traditional NUMA Layout