ECE 574 – Cluster Computing
Lecture 6

Vince Weaver

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

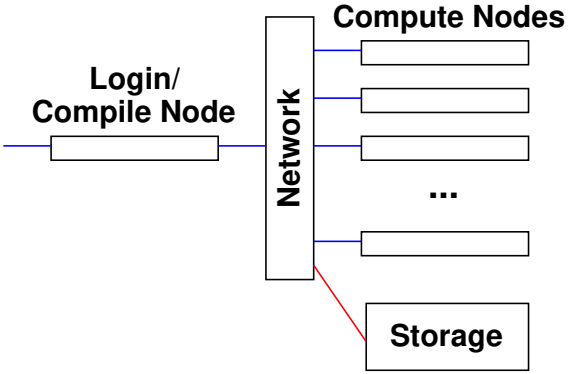
vincent.weaver@maine.edu

16 September 2026

Announcements

- Back from trip?
- Logging into haswell-ep OK?

Commodity Cluster Setup – Hardware



Commodity Cluster Setup – Hardware

- Simple cluster like the pi-cluster, or older ones I've made
- Commodity cluster design is a combo of ECE331/ECE435 more than anything else
- Can be made out of a handful of machines, Ethernet switch, and one machine with two Ethernet ports
- Could it work with wifi instead of wired Ethernet?
- Why have a head node?
- What kind of network? Ethernet? Inifiniband? Something fancier?

Commodity Cluster Setup – Continued

- Operating system? Do all nodes need a copy of the OS? Linux? Windows? None?
- Booting: network boot, local disk boot.
- Network topology? Star? Direct-connect? Cube? Hyper-cube?
- Disk: often shared network filesystem. Why? Simple: NFS (network file system). More advanced cluster filesystems available.
- Don't forget power/cooling

Commodity Cluster Operating System

- Usually Linux these days
- Imagine the cost of getting licenses for a 10k large server
- Setting up user accounts. Not bad if small cluster, a pain to keep synched on large
- Doing system maintenance/updates on large cluster. ssh-agent can help (passwordless login as root).

Commodity Cluster Software

- Do you need to run massively parallel MPI workloads?
- Can you just run many, many single-threaded workloads?
- In any case, how do you launch these jobs?
- Users could pick a node at random to ssh into and run things interactively
  This would be a mess, with some nodes overloaded

Job Schedulers

- 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/TORQUE
- nbs
- slurm
- moab
- condor
- many others

Computer Architecture Review

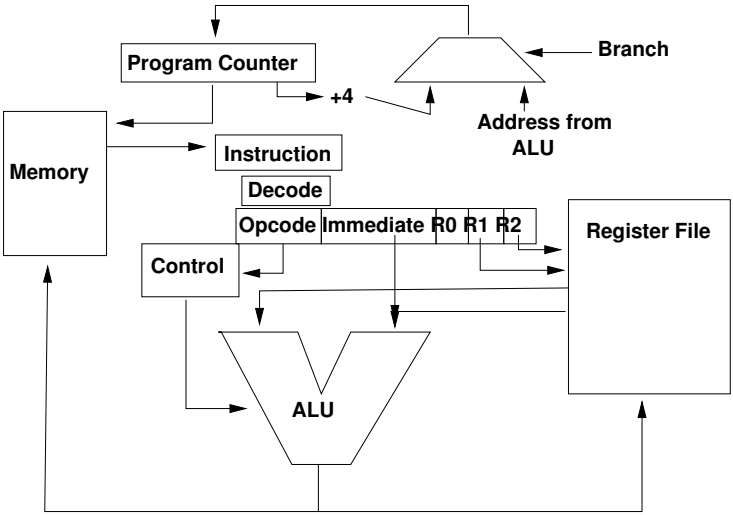
Parallel Computing – Single Core

- Most code written for serial execution:
  one step at a time
- You can re-write it to try to do things in parallel
  (we'll get to that)
- What if the hardware could take your serial code and try to get parallelism out of it for you?

Simple CPUs

- Small, 6502 made of 4500 transistors
- Ran one instruction at a time.
- Could take one or multiple cycles (Instructions per Cycle (IPC) 1.0 or less)
- Example – single instruction take 2-7 cycles?

Simple CPUs Diagram



Code Example – C

```c
int i;
int x[128];
for(i=0;i<128;i++) {
        x[i]=0;
}
```

Can Compiler Provide Speedup?

- First attempt at optimization is to have compiler generate optimal assembly code
- Even today good compilers can often be beaten by skilled assembly language programmers

Code Example – ARM assembly

```
   mov r0,#0       ; i=0 (register allocation)
loop:
   ldr r1,=x       ; point r1 to X array
   lsl r2,r0,#2    ; r2=i*4
   mov r3,#0       ; value to store
   str r3,[r1,r2]  ; X[i]=0
   add r0,r0,#1    ; i=i+1
   cmp r0,#128     ; check if reached 128
   bne loop        ; loop if not equal
.bss
.lcomm x,128,4     ; reserve room for 128 ints
```

Possible Optimizations

- Constant propagation?
- Code hoisting?
- ARM optimizations
  - barrel shifter
  - fancy addressing (auto-increment)
- Loop unrolling?
- Writing 64-bits of zeros rather than 32-bits (then you have to worry about alignment)
- Your OS/libc will provide hand-optimized memset()