ECE 574 – Cluster Computing
Lecture 22

Vince Weaver

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

vincent.weaver@maine.edu

30 October 2026

Announcements

- HW#7 due today.
- HW#8 will be posted.
- Project topics due next Friday.

Remember Project Topics Due

- Send e-mail with topic and group members by Nov 6th (Friday)
- Can work alone or in groups of 2 to 3
- Do something interesting parallel computing related

Topics

- Can use any operating system and written in any language (asm, C, python, C++, Java, etc.)
- Coding, benchmarking
- Past projects: SIMD, parallelizing code, comparison of C vs Python, parallelizing matlab code, calculating physical constants in parallel, building own cluster, raytracing, GPU shader coding
- Will be a final writeup, and then a 10 minute presentation and demo in front of the class during last week of classes.

HW#8 Preview – Pi clusters

- HW#7 we did MPI, but on a shared memory system (which is inefficient, could have just used OpenMP instead)
- For HW#7 I have you fix up your HW#7 code, then run on an actual cluster
- I don't have any large x86 clusters anymore (big and power hungry) so we'll run on one of my Raspberry Pi clusters which are actual real clusters, but low-power usage

Old Pi2 cluster

- 1 head node (16GB SD card), 24 sub-nodes. One currently seems to be down (reliability!)
- Read up on the cluster here:
  `https://www.mdpi.com/2079-9292/5/4/61/htm`
- 32-bit ARM Cortex A7 with 4-cores each and 1GB RAM
  96 cores and 24GB RAM total
- 15.4 GFLOPS

Old Pi2 Cluster Power Usage

- It's not quite a commodity cluster as it has a fairly complicated power distribution system (ATX power supply to power boards to provide measured 5V to the USB power sockets)
  A bit time consuming to wire up all the cables.
- Power distribution issues
  An ATX power supply runs best when it has a PC-like power draw
  Drawing too much 5V without a 12V load and the 5V line droops low enough that the Pis won't boot.
- Draws 90W at idle, which is 20W for ethernet switch, a few watts for fan/lights, and rest for the boards

Old Pi2 Cluster Power Downsides

- Only 1GB RAM each node
- Very slow ethernet (behind USB adapter), MPI runs are network limited
- Why such old nodes? Each time I bought new board for it, essentially within a week they'd announce better Pis. 1B to 1B+ to 2 to 3 when I gave up

New Pi4 Cluster

- Five Pi4 nodes, four core 64-bit ARM Cortex A72 (one is 8GB, others 4GB)
- Gigabit ethernet (Pi4s have PCIe and can handle)
- Only 5-nodes so manually updating IP addresses and password files
- Also haven't set up ssh-agent yet
- Set up slurm which can be a pain, especially getting a workable configuration file

New Pi4 Cluster Performance / Power Usage

- 50 GFLOPS
- Single node is 13 GFLOPS or so, so scaling reasonably
- Same performance as 10 year old macbook air
- Power over Ethernet
- 33W idle, 64W linpack, 0.863 GFLOPS/W
- Three times as fast as Pi2 cluster while using less power
- Note: in theory they have powerful GPUs too but as far as I know no one has good GPGPU support for them yet.

Is there a Pi-5 cluster coming?

- Increasingly power hungry
- Not really a low-power cluster anymore, also takes more cooling

MPI and slurm

- A bit hard to get working, provided script for sbatch should work
- Use “-n” to specify number of cores
- Alternate use “-N” to specify number of nodes maybe in conjunction with - -tasks-per-node
- Not sure how OpenMP and MPI interact here

Why use slurm?

- Can set account to charge
- Can handle checkpointing
- Can set constraints (run on machine with gpu, certain proc type)
- Contiguous allocations
- CPU freq, power capping
- Licenses avail (things like Matlab etc)
- Memory avail

MPI and Linpack

- Running MPI on your own cluster can be a pain, especially making sure it is properly running on all nodes
- Essentially it uses ssh or similar to log into each node and launch your executable. Need to have a copy in the same place on each node (using NFS or similar helps)
- At MPI init it will set things up and communicate between them using sockets. (Details?)
- If running Linpack, you will need to set the P x Q values in HPL.dat to run on more than one nodei

HW#8 More Preview

- First step will be to get HW#7 running if you haven't already. If you had trouble, drop me an e-mail and I can help debug

HW#8 Fine Frained

- Parallel combine?
  Note if you do this, you can in theory skip the gather steps after sobelx and sobely and only gather after combine
  Alternately you can use GatherAll() after sobelx/sobely which will essentially broadcast the results back to all ranks after a gather
- More advanced MPI functions?
- Scatter at start rather than broadcast?
  Note if you do this, remember that the convolution needs an extra line before and after the rows of interest

HW#8 Run on Pi Cluster

- Finally run on Pi cluster

Cassini Saturn Probe

- Gary M. Swift and Steven M. Guertin. “In-Flight Observations of Multiple-Bit Upset in DRAMs””. Jet Propulsion Laboratory
- Cassini, flight recorders, each with 2.5GB RAM
- Single bit error rate of 280 errors/day

Google Datacenter

- Google SIGMETRICS 2009 paper
- Schroeder, Bianca; Pinheiro, Eduardo; Weber, Wolf-Dietrich (2009). “DRAM Errors in the Wild: A Large-Scale Field Study”. SIGMETRICS/Performance (ACM).
- 25-70k errors per billion hours per megabit
- 5 single bit errors in 8GB per hour

Various Supercomputer

- http://www.computerworld.com/article/2493336/computer-hardware/supercomputers-face-growing-resilience-problems.html

  ASCI White when came out, MTBF 5hrs, got it to 55hrs
- "Analysis of the Tradeoffs between Energy and Run Time for Multilevel Checkpointing" PMBS 2014
  - Sequoia MTBF around 1 day
  - Blue Waters: 2 per day,
  - Titan MTBF: less than a day
  - 20% of computation is recovering from failures (big energy waste)
- Scalable In-memory Checkpoint with Automatic Restart on Failures. Xiang Ni, Esteban Meneses,

SSMD/ORNL (2015)

- https://csmd.ornl.gov/highlight/failures-large-scale-systems-long-term-measurement-analysis-and-implications
- 1.2 billion hours Jaguar/Titan/EOS
- MTBF can change over time, optimal checkpoint might depend on this
- Temporal and Spatial Locality
  - Temporal, fail at same time. Things like voltage surge, OS panic, filesystem error
  - Spatial, failure in same location. Cooling/overheat issues

Frontier Supercomputer

- 10 oct 2022
- https://www.datacenterdynamics.com/en/news/frontier-supercomputer-suffering-daily-hardware-failures-during-testing/
- This also mentioned in the SC top500 video
- Problems with MPI Cray fabric
- Possibly also GPU issue under load
- MTBF hours, not days. “one day MTBF would be amazing”

GPU Lifetimes

- "GPU Lifetimes on Titan Supercomputer: Survival Analysis and Reliability" by Ostrouchov et al
- https://www.christian-engelmann.info/publications/ostrouchov20gpu.pdf
- 18,688 GPUs do most of computing
- There have been three re-works
- Two to fix chassis issue
- One to fix resistors failing due to silver sulfide corrosion
- DBE (double bit errors) and OTB (off the bus errors, i.e. GPU stop responding)
- MTBF for first batch around 3 years, but some fail more quickly

GPU Lifetimes

- Survival Analysis methods (similar to those used in medical field)
- How resources for this? A single GPU fail, you maybe send for warranty. If you are buying 20k of them you get some extra attention from the company.

What can Software do to avoid HW Problems

- Note that a lot of reliability bugs are very similar to security bugs
- Programs crash due to out of bounds, memory overflows, stack smashing
- Hardware is starting to add protections against these types of things (Ryzen3 shadow stack)
- Shadow Stacks
- Control Flow Validation

Byzantine Faults

- This is a long-standing issue in distributed system research
- Can you have code that runs correctly even if the underlying hardware is unreliable
- A system where you can't trust the hardware is often said to have Byzantine Faults

Byzantine Failure

- Byzantine General Problem, Lamport et al
  - Generals surround a city.
  - Want to all attack or all retreat; doing part way will fail
  - Might be traitorous generals with complex motives
  - (split their vote, if 5R 4A, tell the 5A and 4R).
  - Unreliable messengers

N-version software

- Implement same code many different ways
- Vote on result.
- Need a tight spec to make sure results will all match.

Algorithm Based

- Parity checks, CRC
- Spread out work so that if one gives wrong result it can be checked. Overlap work.
- Add some extra values to calculation that can be checked, can tell if something went wrong

Data Structure Based

- Extra state in data structure or checksum so can tell if it gets corrupted.

Control Flow Checking

- Knows where code should be allowed to jump to
- If you jump somewhere impossible, checker stops things
- Hardware these days can help with this
- For example: compiler knows all callers of function. Return from function should always return back to one of these locations.

Application Level Checkpointing

- Checkpoint your program state periodically.
- If a failure takes down a program or hardware node, you can restore to last checkpoint rather than starting from scratch.
- Two kinds
  - manual — (you save out your state manually and have to write code to restart from arbitrary point)
  - Automatic – kernel stores everything possible about your state and can restart a program from a snapshot.

Checkpointing Difficulty

- Must save all program state, network connections, RAM contents, disk state, open files, etc.
- Hard to do (I've written one). Some support in Linux kernel, need lots of patches as some syscalls are write-only.

Checkpointing Overhead

- Checkpoints have high overhead. Have to stop while taking them? Write GB to disk?
- Multilevel checkpoint – big checkpoint occasionally and smaller subcheckpoints

Linux Checkpointing Support

- `crui.org`
- CRIU: Checkpoint/Restore in Userspace

Saving Checkpoint

- Use proc to gather state. Open file descriptors, etc
- Insert “parasite” code using ptrace that runs and saves all memory, then removes itself

Restoring Checkpoint

- Re-allocate shared resources
- Fork processes
- Restore memory
- Needs special help from kernel, things like `TCP_REPAIR` which lets you put tcp/ip connections in exact previous state

Crash Only Software

- Crash-only software – crashing and restarting can take less time than clean reboot.
- So why write code to cleanly shutdown? Instead write your code so it can handle crashes cleanly. That way your cleanup code is tested every exit, rather than rarely on a crash.

Approximate Computing

- Approximate Computing – some algorithms don't necessarily need the "right" value
- Video rendering, voice recognition, web search, robotics, GPS, image processing