ECE 574 – Cluster Computing
Lecture 14

Vince Weaver

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

vincent.weaver@maine.edu

25 March 2025

Announcements

- HW#6 due Friday.
- Project topics due Thursday.

Remember Project Topics Due

- Send e-mail with topic and group members by March 27th (Thurs)
- Can work alone or in groups of 2 to 3
- Do something interesting parallel computing related
- 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#4 Review – Code

- Usually the difficult part is splitting up the bounds
  This year that wasn't too bad
- The Coarse code people mostly did fine on
- The Fine code caused the most trouble
  - The most trouble was trying to get the paramaters properly into a struct to pass in
  - Each thread *needs* its own struct.
    If you have one argument struct, pass it to a thread, then modify it and try to pass to another, there's a

race condition where all the threads are using the same struct and any changes you make in the main thread will be seen by all

- Generally the way to do this is calloc() or malloc() (I know, low-level C)
- Some people were using VLA (Variable Length Arrays) where you can do something like declare

```
int num_threads=4;
struct thread_arguments[num_threads];
```

This is a relatively new feature of C so I forget it will work (originally array sizes needed to be a constant). Do note if you use this, the array gets allocated when declared, if you later change num_threads it doesn't go back and re-allocate the array size and in that case your code might overrun the buffer.

- You are free to add members to the argument struct. You probably will have to for when you call compare.

HW#4 Review – Deadlock Question

- The example code problem is an example where deadlock can occur
- The problem is both threads at the same time can take lock1/lock2 and then when they get to the next locks lock2/lock1 they both get stuck waiting for the other to be free, but since they are both cross-waiting it will never happen, no progress is made, deadlock
- Why not just re-arrange the locks? In real life lock ordering issues aren't always so simple

this type of thing might happen in an OS

```
thread1                                  thread2

printf("\b"); // (beeps)                 play_sound();
...                                      ....
console_lock(); // to safely             sound_lock();    // get sound
...             // print                 .....            // hw access
...                                      error happens
beep();                                  print_error();
...                                      console_lock();  // to safely
sound_lock();   // to get                                 // print
                // sound hw access
```

Hand back Midterms

Average was 85%

Lower-Level Problems

Soft errors/Radiation

- Chips so small, that radiation can flip bits. Thermal and Power supply noise too.
- Soft errors – excess charge from radiation. Usually not permanent.
- Sometime called SEU (single event upset)

Radiation

- Neutrons: from cosmic rays, can cause "silicon recoil" Can cause Boron (doped silicon) to fission into Li and alpha.
- Alpha particles: from radioactive decay
- Cosmic rays – higher up you are, more faults Denver 3-5x neutron flux than sea level. Denver more than here. Airplanes. Satellites and space probes are radiation-hardened due to this.
- Smaller devices, more likely can flip bit.

Shielding

- Neutrons: 3 feet concrete reduce flux by 50%
- alpha: sheet of paper can block, but problem comes from radioactivity in chips themselves

Case Studies

- "May and Woods Incident" first widely reported problem. Intel 2107 16k DRAM chips, problem traced to ceramics packaging downstream of Uranium mine.

- "Hera Problem" IBM having problem. ^{210}Po contamination from bottle cleaning equipment.

- "Sun e-cache" Ultra-SPARC-II did not have ECC on cache for performance reasons. High failure rate.

Hardware Fixes

- Using doping less susceptible to Boron fission
- Use low-radiation solder
- Silicon-on-Insulator
- Double-gate devices (two gates per transistor)
- Larger transistor sizes
- Circuits that handle glitches better.

Memory Fixes

- ECC code
- spread bits out. Right now can flip adjacent bits, flip too many can't correct.
- Memory scrubbing: going through and periodically reading all mem to find bit flips.

Extreme Testing

- Single event upset characterization of the Pentium MMX and Pentium II microprocessors using proton irradiation", IEEE Transactions on Nuclear Science, 1999.
- Pentium II, took off-shelf chip and irradiated it with protons. Only CPU, rest shielded with lead. Irradiate from bottom to avoid heatsink
- Various errors, freeze to blue screen. no power glitches or "latchup" 85% hangs, 14% cache errors no ALU or FPU errors detected.

Memory Failures

- Give brief review of SRAM vs DRAM
- Memory Errors in Modern Systems
  ASPLOS 2015
- Battling Borked Bits
  IEEE Spectrum December 2015

Intentional Memory Failures?

- Rowhammer
- DRAM is just holding RAM contents in capacitors, which leak away and need to be constantly refreshed
- Need to refresh every 32 to 64ms
- If you access a memory location a lot, it can also make nearby locations drain faster and make them have bit flips
- Interesting project?

Failure and Error Rate – Examples

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-resili 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, Laxmikant V. Kal
  Most of failures do not take down more than one node
  Jaguar/Titan 92% crashes single-node crashes

SSMD/ORNL (2015)

- https://csmd.ornl.gov/highlight/failures-large-scale-systems-long-term-measurement-analysis-and-i
- 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-failur
- 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
- Survival Analysis methods (similar to those used in medical field)

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)

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

- TODO: a little more info on this project

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

BOINC

- Someone asked how distributed computing worked for things like Folding at Home
- Use Berkeley Open Infrastructure for Network Computing
- Sort of like grid computing
- As of 2020 worldwide added up to 41 PFLOPS (would be #5 on top500)
- Researchers upload binaries and datasets, the servers then distribute them to volunteers by client they run

- Server is just really old-fashioned PHP/MySQL LAMP server
- Server also validates results, also hands out workloads multiple times to be sure the answers match
- My friend runs the Machine Learning Comprehension at home project.