ECE 574 – Cluster Computing
Lecture 13

Vince Weaver

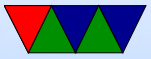
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

16 March 2021

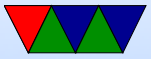
Announcements

- Don't forget HW#6
- Be careful deleting files on Haswell-EP, it's not backed up



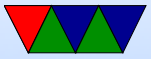
More Feedback on HW#5

- Use of private vars.
  Luckily loop indices aer always treated as private. Not sure how some of the solutions worked without declaring sum to be private.
- Static vs Dynamic speed. Setting up dynamic has a lot of overhead, and since our runs are quick and roughly same size didn't make a difference
- Reduction: you can use a reduction, but only if you are summing up results from a loop. So if you're using a



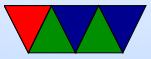
loop to do the sums

- Putting parallelization in inner loops instead of outer. Probably not recommended but seems to work. Maxes out at total number of threads. Possibly some overhead for starting parallel over and over



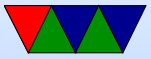
Go Over Midterms

- Average was in the 90s
- Will send e-mails with the results

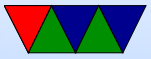


Further HW#6 review

- Fix warnings!
- More code comments too
- Remember, we want from 1 to y-1, not 0 to y
  That is tricky when subtracting off
- If you create a temp buffer, that's fine, but might want to zero-it-out before reusing it or else the edges might have old data in them.
- sizeof operator... what is `sizeof(image.pixels)`
- Calculating the tail-ends of things and copying it in place

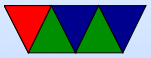


- If you get a staggered/shifted effect. Should probably gather in multiples of xsize, otherwise it will be shifted. Gather in the same size as your ysize limits.

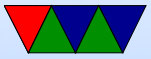


BOINC

- 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

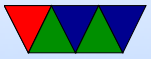


- Server is just really old-fashioned PHP/MySQL LAMP server
- Server also validates results, also hands out workloads multiple times to be sure the answers match
- My friend runs the Machine Learning Comprehension at home project.



Things you can do Software

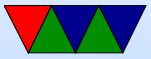
- Note that a lot of reliability bugs are very similar to security bugs
- Programs crash due to out of bounds, memmory overflows, stack smasking
- Hardware is starting to add protections against these types of things (Ryzen3 shadow stack)



Memory Failures

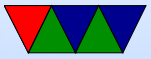
- Memory Errors in Modern Systems
  ASPLOS 2015

- Battling Borked Bits
  IEEE Spectrum December 2015



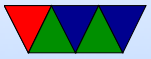
Intentional Memory Failures?

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



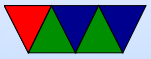
Architectural Vulnerability factor

- Some bit flips matter less
- (branch predictor) others more (caches) some even more (PC)
- Parts of memory that have dead code, unused values



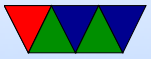
Failure and Error Rates

- Cassini, flight recorders, each with 2.5GB RAM
  Single bit error rate of 280 errors/day
- Google SIGMETRICS 2009 paper
  25-70k errors per billion hours per megabit
  5 single bit errors in 8GB per hour
- ASCI White when came out, MTBF 5hrs, got it to 55hrs
- Sequoia MTBF around 1 day, Blue Waters: 2 per day,

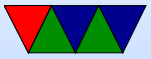


Titan MTGF: less than a day

- 20% of computation is recovering from failures (big energy waste)
- Most of failures do not take down more than one node Jaguar/Titan 92% crashes single-node crashes

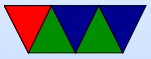


Things you can do Software



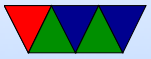
Byzantine Failure

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



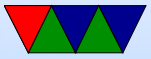
N-version software

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



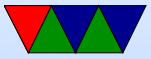
Algorithm Based

- 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



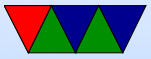
Control Flow Checking

- Knows where code should be allowed to jump to
- If you jump somewhere impossible, checker stops things



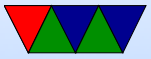
Checking Data Structures

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



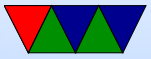
Application Level Checkpointing

- 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.



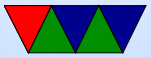
Difficulty? All program state, network connections, RAM contents, disk state, open files, etc. Hard (I've written one). Some support in Linux kernel, need lots of patches as some syscalls are write-only.

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



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