ECE 574 – Cluster Computing
Lecture 14

Vince Weaver

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

vincent.weaver@maine.edu

11am, Barrows 133

5 March 2024

Announcements

- HW#6 was posted. Tentatively due Friday.
- Midterm still being graded.
- Academic honesty discussion (please don't share code, especially if it's before the homework deadline)

HW#4 Finally Graded

- Most issues were with splitting up workload
  Some attempts were overly complex, if you have a lot of trouble it might be best to try something simpler
- Another issue was forgetting the top/bottom rows should not be calculated (border)

HW#6 Cluster fairness

- If your job gets stuck, be nice and kill it (scancel)
- The node isn't currently enforcing times. I could set it up to do so but worried I'd break things
- sbatch scripts I give you have 10 minute timeout, you can lower that if you want to be safer

HW#6 – Yet Again

- I know you've heard this before, but maybe as people have tried the code it will make more sense
- This is probably the most difficult assignment

HW#6 – Step #1, Broadcast

- Load jpeg in rank0
- Broadcast xsize, ysize, depth to all
- Non-rank0 set image.xsize, image.ysize, image.depth
- Non-rank0 allocate image.pixels
- Broadcast image data (note: to image.pixels, not image), be sure CHAR not INT
- I added simple checksum you can verify it worked

HW#6 – Step #2, Convolve

- Split up work per-rank
- Simplest way is to have ystart as `rank*(ysize/num_ranks` and yend as `(rank+1)*(ysize/num_ranks`
- This looks like it might overlap, but if in loop use $<$ it should be OK (is it problem if you do overlap?)
- Be sure you handle skipping y=0 and y=(ysize-1) cases. Do that *after* you calculate split or you can miss lines
- Be sure the leftover rows get calculated. Easiest way is just have the last rank have yend=ysize (not balanced

but simple)

HW#6 – Step #3, Gather

- Gather is annoying because it gathers from the beginning of an array, but by default our convolve() routine puts the results at different locations in each rank
- Three ways to handle this:
  - Adjust the convolve routine to subtract off ystart when doing output so the results end up at the start of the array
  - When done the convolve, use memcpy() to copy memory to the start

- In the gather call, instead of just putting sendbuf as newimage, you can have offsets like &newimage[rank*(ysize/ranks)*xsize*3] this is easy to get wrong as it's pointer math

HW#6 – Step #4, Leftover

- Getting to step #3 is most important, which is why this is worth fewer points. This is more for spacestation example rather than butterfinger (which is 320x320 so very divisible)
- Most straightforward way is to replace Gather() with Gatherv().
- You will need to set up two arrays, one with offsets and ones with lengths. This is tricky as these are bytes, so you can re-use your yoffsets from before but you'll need

to multiply by xsize and depth

- For the last rank just have the length have the extra part. This is mildly tricky to calculate, use the % operator to get the remainder.

HW#6 – Common Failures

- Top of image there, rest is black – usually this means you haven't adjusted your data before gathering, and gather is grabbing from the top of your image which is empty on non-rank0
- Top is fine, but weirdly offset and maybe rainbow for rest – this happens if you gather in (ysize*xsize*3)/ranks chunks rather than (ysize/ranks)*xsize*3. Those look like they are the same, but it's an integer divide so truncating means the latter will grab things in a non-

multiple of the rowsize.

- Looks correct but md5sum doesn't match – this is usually because you forgot to handle the top/bottom border, or else your ystart/yend ranges have small gaps in them

Reliability in HPC

Good reference is a class I took a long time ago, CS717 at Cornell:

http://greg.bronevetsky.com/CS717FA2004/Lectures.html

Sources of Failure

- Software Failure
  - Buggy Code
  - System misconfiguration
- Hardware Failure
  - Failed capacitors
  - Loose wires
  - Tin whiskers (lead-free solder)
  - Lightning strike
  - Radiation

- Moving parts wear out
- Malicious Failure
  - Hacker attack
- Environment issues
  - Fire in datacenter
  - Loss of cooling during heat wave

Types of fault

- Permanent Faults – same input will always result in same failure
- Transient Faults – go away, temporary, harder to figure out

What do we do on faults?

- Detect and recover?
- Just fail?
- Can we still get correct results?

Metrics

- MTBF – mean time before failure
- FIT (failure in Time)

  One failure in billion hours. 1000 years MTBF is 114FIT. Zero error rate is 0FIT but infinite MTBF Designers just FIT because additive.
- Nines. Five nines 99.999% uptime (5.25 minutes of downtime a year)

  Four nines, 52 minutes. Six nines 31 seconds.
- Bathtub curve

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
- Low mantissa bits in floating bit numbers
- Colors in graphics shown for only a frame

Things you can do for reliable Hardware

Hardware Replication / Redundancy

- Lock step – Have multiple machines / threads running same code in lock-step Check to see if results match. If not match, problem. If replicated a lot, vote, and say most correct is right result.
- RAID – (redundant array of inexpensive disks)
- Memory checksums – caches, busses
- Power conditioning, surge protection, backup generators, UPS

- Hot-swappable redundant hardware

Lower Level (Inside your Computer)

- Replicate units (ALU, etc)
- Replicate threads or important data wires
- CRCs and parity checks on all busses, caches, and memories

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 proton. 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

- 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