Announcements

• HW#10 was due
• Remember prelim-exam on Tuesday.
• Remember project status update.
HW #7 notes

• Fine grained parallelism
• Running on the Pi-cluster
  ○ Test with np=7, some code failed that worked with 2 9d4b6548fa8c6ff66602ef5993aca90f common seems to be not gathering in the extra lines
  ○ Reading from each core rather than Bcast doesn’t help anything.
  ○ Some analysis of pi-cluster results
    Only scale up to 4.
<table>
<thead>
<tr>
<th>cores</th>
<th>load</th>
<th>bcast</th>
<th>convolve</th>
<th>combine</th>
<th>gather</th>
<th>tail</th>
<th>store</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0</td>
<td>12.8</td>
<td>3.8</td>
<td>0.1</td>
<td>0</td>
<td>3.4</td>
<td>21.2</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.1</td>
<td>6.4</td>
<td>1.9</td>
<td>1.8</td>
<td>0</td>
<td>2.4</td>
<td>13.7</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.3</td>
<td>3.2</td>
<td>0.9</td>
<td>3.0</td>
<td>0</td>
<td>2.4</td>
<td>10.9</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>5.6</td>
<td>1.7</td>
<td>0.5</td>
<td>4.6</td>
<td>0</td>
<td>2.4</td>
<td>15.8</td>
</tr>
<tr>
<td>16</td>
<td>1.0</td>
<td>7.3</td>
<td>0.7</td>
<td>0.2</td>
<td>6.5</td>
<td>0</td>
<td>2.4</td>
<td>18.2</td>
</tr>
<tr>
<td>32</td>
<td>1.0</td>
<td>8.0</td>
<td>0.3</td>
<td>0.1</td>
<td>6.4</td>
<td>0</td>
<td>2.4</td>
<td>18.3</td>
</tr>
<tr>
<td>64</td>
<td>1.0</td>
<td>8.8</td>
<td>0.1</td>
<td>0.06</td>
<td>6.9</td>
<td>0</td>
<td>2.4</td>
<td>19.5</td>
</tr>
</tbody>
</table>
HW #8 notes

- Be careful memory copying, if copying an array of 9 ints need to copy 36 bytes (not 9)
- Also, you can pass in ints as parameters (no need to allocate space and then memcpy in. Or you could, but if you do you would use points and allocate space properly)
- Be sure you are using *unsigned char* for the image data, not signed char.
- Limits and matrix indexing
- My results with a large image
<table>
<thead>
<tr>
<th>Type</th>
<th>Load</th>
<th>Copy</th>
<th>Convolve</th>
<th>Combine</th>
<th>Store</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMPx16</td>
<td>1s</td>
<td>0.882</td>
<td>0.135</td>
<td>0.9</td>
<td>3.12</td>
<td></td>
</tr>
<tr>
<td>MPIx16</td>
<td>1s</td>
<td>0.5+1.4</td>
<td>0.6</td>
<td>0.1</td>
<td>1.0</td>
<td>4.9</td>
</tr>
<tr>
<td>Cuda</td>
<td>1s</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>1.0</td>
<td>3.3</td>
</tr>
<tr>
<td>OpenCL CUDA</td>
<td>1s</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.9</td>
<td>2.9</td>
</tr>
<tr>
<td>OpenCL intel</td>
<td>1s</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.9</td>
<td>3.0</td>
</tr>
<tr>
<td>OpenCL pocl</td>
<td>1s</td>
<td>0.4</td>
<td>0.6</td>
<td>0.7</td>
<td>1.0</td>
<td>3.4</td>
</tr>
</tbody>
</table>
HW #9 notes

• Didn’t get to grade it, but it should have been straightforward.
HW #10 notes

- Nothing exciting
- In general, fastest is also most energy efficient
- This is not always the case, but is on this hardware
Midterm Review

- Cumulative, but concentrating on stuff since last exam
- Speedup/Parallel efficiency
- Tradeoffs. Given code and hardware, would you use MPI, pthreads, CUDA, etc?
- OpenMP: dynamic vs static scheduling. Parallel for MPI
- GPGPU/CUDA: read code, know about cudaMemcpy, cudaMemcpy
- BigData: sizes involved, distributed filesystems
• Reliability. Causes of errors. Tradeoffs of Checkpointing
• Energy, Energy Delay, Time, Performance
Roadmap to Exascale

- Japan, China, France, US
- Goal between 2020 and 2023
- Build a meaningful machine. Can scale up current tech, but it would be insanely power hungry and jobs wouldn’t scale (it would be just a job-sharing cluster)
- US Plan – 2023. DoE. Money to companies (suspected IBM, Cray, Intel, NVIDIA). Same as CORAL 100PFLOP machines (Collaboration of Oak Ridge, Argonne, and
Livermore). Likely IBM Power/NVIDIA GPU/Mellanox InfiniaBand or Intel Xeon Phi/Omni-Path

- Japan – 2020, Likely a Riken K-computer followup
- China – 2020, homegrown, as in 2015 US banned selling supercomputer to china
- France – 2020, Atos/Bull. Likely Intel Xeon/Xeon-Phi
Roadmap to Exascale

- Paul Messina interview (Argonne)
- [https://www.hpcwire.com/2016/06/14/us-carves-path-capable-exascale-computing/](https://www.hpcwire.com/2016/06/14/us-carves-path-capable-exascale-computing/)
- Looking for *Capable Exascale* not *Exaflop* (defined as can calculate 50x faster than 20PFLOP machine and using 20-30MW)
- Broad societal benefits, not just bragging rights
- Interesting graph showing contribution from CMOS scaling leveling off
- Exaflop system 20MW and cost 200 million, then *petaflop*...
McCalpin Talk

- Heterogeneous clusters, some machines with lots of cores, some with lots of memory, can choose.
- Peak FLOPS/socket increasing 50% year
- Memory Bandwidth increasing 23% year
- Memory Latency *increasing* 4% year
- Interconnect Bandwidth increasing 20% year
- Interconnect Latency decreasing 20% year
- Why memory so bad? Emphasis on size. 64-bit wide
internally not increased. Pins cost money.

- Coherency can dominate performance.
- Power/Energy a concern
- Haswell can only run LINPACK on half the cores before it has to downthrottle
- Power does not matter in operation cost? $2500/\text{socket}, 100W, 40W\text{cooling}, .10/kWh, 5\%\text{ of purchase price per year}$
- This may change with mobile chips
Exascale

- Exascale computing – Exaflop = 1000 Petaflops / 1M Teraflops
- Petascale in 2008, estimated Exascale was in 2018 - 2020 but keeps being pushed back.
- Current fastest roughly 40 Petaflops
- Envision as having 100k nodes, each with 10Tflop; modern high-end GPUs only about 3Tflop (double precision)
- Many challenges
DOE

- US Department of Energy’s objectives and requirements for exascale systems
- Had series of workshops 2008 - 2012 to discuss what is needed
Power

• Biggest challenge going forward

• Power costs of largest system $5-10 million

• Exascale with current tech would need 350 megawatts ($250 million/year)

• To be feasible really need to cap at 20 megawatts

• Data movement – historically 1 byte/flop considered reasonable
But for current 2Petaflops system that would take 1.25MW. Even if reduced to bare minimum (0.2 byte/flop) would be 50MW for exascale. Proposed: more energy-efficient hardware, Si-photonic communication, power-aware algorithms.
Concurrency

• Already can’t keep cores busy to mask long-latency (usually memory) events

• Flattening of CPU clock frequency is keeping things from getting worse, but having more cores making requests is not helping

• With exascale, costs more energy to transport data than to compute it.
Fault Tolerance

- Mean Time To Interrupt (MTTI)

- Improve MTTI so applications can run for hours without faults
RAM

- Current power levels unsustainable
- Slowing technology growth, from 4-times per 3 years to 2-times per three years
- Limiting factor in most applications
- Need 4TBpbs bandwidth and 1TB per node
  Current DIMMs have single-digit number of channels with 10s of GB/s
DRAM Performance Metrics

- Energy per bit
- Aggregate bandwidth per socket
- Memory capacity per socket
- FIT rate per node
- Error detection
- Processing in Memory
- Programmability
Programmability

- Three stages: algorithm capture, correctness debugging, performance optimization
- Parallelism – anticipated that 10-billion-way concurrency needed
- Distributed Resource Allocation – need to spread out to parallel, but also need to keep close for low-latency
- Latency Hiding – overlap communication with computation
• Hardware idiosyncrasies – allow using fast novel hardware without burdening programmer too much with the details

• Portability – use software across machine types

• Synchronization – barriers are expensive operations, replace by lightweight (transactional memory?)
CPU/Network

Not really worried about CPU or Network?
AMD: Achieving Exascale Capabilities Through Heterogeneous Computing

- APU (CPU combined with GPU), 3D-RAM, connected to off-core NVRAM
- CPU handles serial sections, GPUs parallel sections
- APU – exascale heterogeneous processor (EHP) Supports HSA (Heterogeneous System Architecture) – CPU and GPU have same shared memory space, CPU and GPU can trade pointers w/o going over PCIe bus
CPU – 32 cores (ARM or x86)
CPU only possible, but probably not have high enough perf/W why integrated? lower overhead. Also higher FLOPS/volume (meters cubed)

- 2.5D interposer-based stacking vs 3d? (3d has through vias CPU to DRAM, 2.5d the dram stack is next to CPU with interposer board to connect)

- QuickRelease and HRF (heterogeneous-race-free) – need complex setup to get cache coherency between GPU and
CPU

- JEDEC high-bandwidth memory (HBM) standard 128 GBps per DRAM stack. With eight stacks, TBps with current tech.

- Three levels of memory (fast, NVRAM, flash?)

- How to use memory? Transparent like current, or expose to user?

- DRAM power – even if reduce from current 60pJ/bit of DDR3 to 2pJ/bit, 4TBps could consume half of the
power of entire cluster

- Processor in memory (PIM) can maybe provide better energy efficiency

- How to program?

- Reliability? GPUs not typically as reliable as CPUs. Corruption in GPU output not considered as critical as in CPU
Intel’s Exascale (March 2019)

- Aurora system
- Exaflop by 2021 (slip from 2019 or 2020)
- Mix of Xeon, Optane, Xe GPUs
- No Xeon Phi?
- Department of Energy
Fujitsu Post-K Computer

- [https://www.theregister.co.uk/2019/04/16/fujitsu_to_start_selling_postk_derivatives_within_12_months/](https://www.theregister.co.uk/2019/04/16/fujitsu_to_start_selling_postk_derivatives_within_12_months/)
- $910 million, between 2021-2022
- Exascale
- One exaflop (current fastest 200 Petaflops)
- Collaboration with Fujitsu and Riken in Japan
- HBM2 memory – 3D stacked, with interposer?
- Tofu interconnect
6 dimensional hypertoroid mixed up with a 4 dimensional hypercube

- ARMv8 with special new vector extensions (not neon)
- ARM scalable vector instructions (SVE)
  vectors from 128 bits to 2048 bits
Vector-Length-agnostic programming (VLA)