ECE 574 – Cluster Computing
Lecture 3

Vince Weaver

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

vincent.weaver@maine.edu

26 January 2017

Announcements

- HW#1 graded. Everyone got full credit, won't send out actual grades. Liked the extra details on the clusters
- Handing out accounts on Haswell-EP for HW#3. Please use the machine responsibly.

HW#2 Review

- Sunway Top#1 Article
  - One way they save power? Low-power DDR3 RAM
  - Linpack: 74% efficient, HPCP: 0.3% efficient
  - US exaflop? Early 2020s
- Next Gen Computer Article
  - Does calculation use most energy? No, data movement.
  - Did we hit DARPA exascale in 2015 goal? No
  - When will we hit exascale? 2023

- Reliability Article
  - Jaguar: 350 errors/min
  - BGQ: problems from radioactive lead in solder
  - Power gating: reduces life of chip, can cause surges

Average Machine Speeds

- Look up my top40 list. Green and regular. Compare with top and bottom of top500. Also Pi-cluster

- Computers we might use in class:
  haswell-ep server 436 GFLOPS, 16/32 cores, 80GB, 2.13GFLOP/W
  power8 machine 195 gflops, 8/64 cores, 32GB, ??
  pi-cluster, 15.4 GFLOPS, 96 cores, 24GB RAM, 0.166 GFLOP/W
  pi-3B 3.62 GFLOPS, 4 cores, 1GB RAM, 0.813

GFLOP/W (higher possible)

Reminder, top machine, 93 PFLOPS, sunway, 6GFLOPS/W (top 10 3-9 GFLOPS/W)

- First list, June 1993. Top machine 1024 cores, 60 GFLOPS, 131kW

  Pi cluster would have been #7

Review of Weak/Strong Scaling

Where Performance Info Comes From

- User Level (instrumentation)
- Kernel Level (kernel metrics)
- Hardware Level (performance counters)

Types of Performance Info

- Aggregate counts – total counts of events that happen
- Profiles – periodic snapshots of program behavior, often providing statistical representations of where program hotspots are
- Traces – detailed logs of program behavior over time

Gathering Aggregate Counts

Measuring runtime – using `time`

```
$ time ./dgemm_naive 200
Will need 1280000 bytes of memory, Iterating 10 times

real        0m7.360s
user        0m7.330s
sys        0m0.000s
```

- Real – wallclock time
- User – time the program is actually running (how calculated)
- Sys – time spent in the kernel

- Must USER+SYS = REAL? Not necessarily (what if other things using the kenrel)

- Can USER be greater than REAL? yes, if multiprocessor

- Is the time command deterministic?
  No. Lots of noise in a system. Can write whole papers on why.

- Which do you use in speedup calculations?

perf tool

```
$ perf stat ./dgemm_naive 200
Will need 1280000 bytes of memory, Iterating 10 times

 Performance counter stats for './dgemm_naive 200':

       7239.152263      task-clock (msec)         #    0.992 CPUs utilized
               116      context-switches          #    0.016 K/sec
                 0      cpu-migrations            #    0.000 K/sec
               357      page-faults               #    0.049 K/sec
     6,513,184,942      cycles                    #    0.900 GHz
   <not supported>      stalled-cycles-frontend
   <not supported>      stalled-cycles-backend
     2,592,685,475      instructions              #    0.40  insns per cyc
        91,797,411      branches                  #   12.681 M/sec
           974,817      branch-misses             #    1.06% of all branch

       7.299463710 seconds time elapsed
```

- Many options. Can select events with `-e`
- Use `perf list` to list all available events
- Hundreds of events available on x86, not quite so many on ARM.
- Understanding the results often requires a certain knowledge of computer architecture.

Profiling

- Records summary information during execution
- Usually Low Overhead
- Implemented via **Sampling** (execution periodically interrupted and measures what is happening) or **Measurement** (extra code inserted to take readings)

Profiling Tools

- Low Overhead – Using hardware counters, such as perf
- Small Overhead – Using static instrumentation, such as gprof
- Large Overhead – Using dynamic binary instrumentation, such as valgrind callgrind

Compiler Profiling

- gprof
- gcc -pg
- Adds code to each function to track time spent in each function.
- Run program, gmon.out created. Run “gprof executable” on it.
- Adds overhead, not necessarily fine-tuned, only does time based measurements.
- Pro: available wherever gcc is.

Perf Profiling

Automatically interrupts program and takes sample every X instructions.

- `perf record`
- `perf annotate`

Skid

- Beware of “skid” in sampled results

- This is what happens when a complex processor cannot stop immediately, so the reported instruction might be off by a few instructions.

- Some processors do not have this problem. Recent Intel processors have special events that can compensate for this.

Tracing

- When and where events of interest took place
- Shows when/where messages sent/received
- Records information on significant events
- Provides timestamps for events
- Trace files are typically *huge*
- When doing multi-processor or multi-machine tracing, hard to line up timestamps

Performance Data Analysis

Manual Analysis

- Visualization, Interactive Exploration, Statistical Analysis
- Examples: TAU, Vampir

Automatic Analysis

- Try to cope with huge amounts of data by automatic analysis
- Examples: Paradyn, KOJAK, Scalasca, Perf-expert