CS295 - Week 7 Notes (Jess Cobb)

1. GFLOPS calculation : CPU and GPU

115 GFLOPS/sec – theoretical? (CPU), 90 GFLOPS/sec – practical? (CPU)

visit www.cs.colostate.edu/~info/cuda-faq.html

Dimensional Analysis → **GFLOPS/sec** = **flops/cycle** x **cycles/sec** x **cores** x **sockets**

we know: flops/cycle; frequency (cycles/sec); # cores; # sockets

Numerical Application → flops/cycle (operations per cycle) = either 4x4x2 for DP per core or 4x8x2 for SP per core; # cores = 6; # sockets = 1; frequency = 3.6 GHz

Double Precision (DP) → 64 bits, 8 byte (for operations on doubles)

Single Precision (SP) → 32 bits, 4 byte (for operations on integers)

AVX2 256bit data/8 → 32 bytes → DP (4(ALUs per core)x4) / SP(4x8)
FMA 32 bytes → DP(4x4)/SP(4x8)
- should be same number of ops. performed each cycle because of pipelining
- improving performance involves hiding latency of a program

Theoretical = 345.6 GFLOPS/sec

2. Gbytes calculation : CPU and GPU

cat /proc/cpuinfo
/proc/meminfo
dmidecode → all three are options to find out info. computer internals

Memory in fish machines: DDR4 2400; 2400 is frequency (MHz)

- also need to know # occupied RAM slots and bandwidth (i.e. bytes/transfer and == 8)

Gbytes/s = **frequency** x **# slots** x **bandwidth** → 2.4 x 4 x 8 = 76.8 Gbytes/s, then divided by 2 because fish machines have 2 slots filled

Broadwell microarchitecture states max. slot # is 4

Performing Roofline again:
1) Visit www.cs.colostate.edu/~info/cuda-faq.html
2) Use config.titan.ocs.ornl.gov.02
3) Replace CC by g++
4) Add ERT_LDFLAGS -L/usr/local/cuda/lib64 -lcudart
-L for linker

3. OpenMP and Tiling

Hands on: see either CS475 F17 Lab1 or CS475 F16 Lab1 & Lab2

#pragma omp parallel … (in mandelbrot.c) – distributes workload over cores using a for loop, workload is distributed based on N divided by # cores

- this causes the i value of the for loop between each processor to be different → private thread variables
- by default the first loop's variable is private but can specify to be "shared"
- dynamic and static scheduling → no idle processors with dynamic scheduling