CS295 – Week 8 Notes (Jess Cobb)

See GPU Optimization Fundamentals slides – Cliff Woodley

Separate into GPU, CPU, and FPGA groups in the coming weeks

GPUs
__global__ tells system to run program on GPU

- for CUDA loop version, array transformed into 1D → array access easier to see when 1D and using CUDA

- threads inside a thread block can share some data between them
- a thread block is mapped to a multiprocessor

- large register file on GPU, small cache

- cache-line → a block of data that is prefetched by the hardware; requesting processor can now be viewed as a thread

- GPUs are memory bound because of the very small amount of storage space

- transpose<<a,b>>(in, out, N); (code for CUDA)
a,b → number of threads running on GPU and how they are organized

- each thread is a copy of another thread

- threadIDs and blockIDs are needed to differentiate threads and blocks from one another since they may be running the exact same code

- there will be N-squared threads running in parallel

- A[i][j] can also be accessed as 1D array → B[i*j]

Roofline
- 1 IPC (instruction count) because only 1 FP (floating point) unit in the Nvidia chip

128 cores/processors x **10 processors** x **1 IPC** x **1.86 cycles/sec** = 238.0 Gflops/s

deviceQuery → Gbytes/sec = 192.18 GB/sec

```
for i
  for k
    for j
```
x-axis = Flops/byte → Flops/byte = C[i][j] += A[i][k] * B[i][k] + B[j][k] * A[j][k]
- 1 double is 8 bytes
- 4 arithmetic ops. / (5*8 bytes)

- cache misses occur when iterating i, k, j

for i, for j, for k → offers performance improvements
a. hardware prefetching
b. vectorization → involves unit stride access

Tiling
- want to maximize use of data currently in cache before replacing it

- divide data space into logical blocks

- optimal tile size is based on size of caches

- tiling is worse than loop permutation because with more loops there is more overhead for the program to keep track of, OR communication cost becomes higher than computation cost

syr2k
1. Naive
2. Loop-permuted
3. Parallelization