Parallel & Distributed Processing II: *parallel processing on manycore chips*

Using CUDA

Eric Aubanel
Winter 2010, UNB Fredericton

Use NVIDIA's resources

- In Blackboard
  - Getting Started Guide
  - Programming Guide
  - Best Practices Guide
  - Reference Manual
- CUDA Zone: www.nvidia.com/object/cuda_home.html
- Many other resources on the Web

Compiling and Running

- Set environment variables:

```
export PATH=/usr/local/cuda/bin:$PATH
export DYLD_LIBRARY_PATH=/usr/local/cuda/lib:
  $DYLD_LIBRARY_PATH
```

- Put combined CPU/GPU code in one **`.cu`** file
- Compile

  $ nvcc –o prog prog.cu
  - Can use gcc flags such as –O
    - Optimization of sequential code important for fair comparison
- Run
  - Same as running any CPU executable

  $ prog arg1 arg2 …

Measuring Performance

- Elapsed time of CPU code

```
double wallClockTime() { //time in seconds
  struct timeval tv;
  gettimeofday(&tv, NULL);
  return (1000000*tv.tv_sec+tv.tv_usec)/1.0e6;
}
double time = wallClockTime();
// code to be timed
double time = wallClockTime() - time;
```

Measuring Performance

- Can use CPU timer to time GPU code, but…
  - Some CUDA API functions are asynchronous
    - Kernel calls
    - Memory copies with **`Async`** suffix
  - To avoid this problem use cudaThreadSynchronize() immediately before starting and stopping timer
    - Blocks CPU until all CUDA calls previously issued have completed

Measuring Performance

- CUDA GPU timer
  - Uses GPU clock, so more accurate timings

```
cudaEvent_t start, stop;
float time;
cudaEventCreate(&start);
cudaEventCreate(&stop);

cudaEventRecord( start, 0 );
kernel<<<grid,threads>>> ( p1, p2, ...);
cudaEventRecord( stop, 0 );
cudaEventSynchronize( stop );/* block until event
  actually recorded */

cudaEventElapsedTime( &time, start, stop );
cudaEventDestroy( start ); cudaEventDestroy( stop );
```

Measuring Device Bandwidth

- "Bandwidth is one of the most important gating factors for performance. Almost all changes to code should be made in the context of how they affect bandwidth."

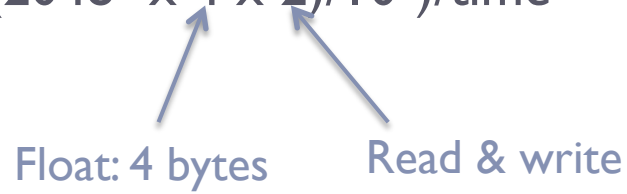
- *CUDA 2.3 Best Practices Guide*

- Compare theoretical and effective bandwidth
  - Theoretical
    - Can calculate (see best practices guide), but running /Developer/GPU Computing/C/bin/darwin/release/bandwidthTest will give actual achievable DRAM bandwidth

Effective bandwidth

- [(bytes read per kernel + bytes written per kernel)/10^9]/time
  - In GB/s
- Eg. Copy of 2048 x 2048 matrix (floats)

  $((2048^2 \times 4 \times 2)/10^9)$/time



Profiler bandwidth

- Not same as effective bandwidth, because:
  - Subset of GPUs multiprocessors used, and results extrapolated
  - Includes transfer of data not used by kernel
- Comparison of effective and profiler bandwidth can indicate how much bandwidth wasted by suboptimal coalescing