L11: Jacobi, Tools and Project

Administrative Issues

- Office hours today:
  - Begin at 1:30
- Homework 2 graded
  - I'm reviewing, grades will come out later this week
- Project proposals
  - Due 5PM, Friday, March 13 (hard deadline)
- Homework (Lab 2)
  - Due 5PM, Friday, March 6
  - Where are we?

Outline

- Jacobi
  - How to approach
  - Things to take care of
- Programming Tools for CUDA
  - Occupancy Calculator
  - Profiler
- Project
  - Discuss MPM/GIMP
  - Construct list of questions

Jacobi Tiling

Sequential C code:

```
…
#define n 64
float a[n][n][n], b[n][n][n];
 for (i=1; i<n-1; i++)
   for (j=1; j<n-1; j++)
     for (k=1; k<n-1; k++) {
       a[i][j][k]=0.8*(b[i-1][j][k]+b[i+1][j][k]+b[i][j-1][k]  +
                       b[i][j+1][k]+b[i][j][k-1]+b[i][j][k+1]);
     }
```

Analyze Reuse of Data

Iteration space:

for i, 1 <= i <= n-1 for j, 1 <= j <= n-1 for k, 1 <= k <= n-1

Access expressions:

b[i-1][j][k], b[i+1][j][k], b[i][j-1][k], b[i][j+1][k], b[i][j][k-1], b[i][j][k+1]

Reference Pair	Dist.	Reference Pair	Dist.	Reference Pair	Dist.
b[i+1][j][k] b[i-1][j][k]	2,0,0	b[i+1][j][k] b[i][j-1][k]	1,1,0	b[i][j][k-1] b[i][j-1j][k]	0,1,-1
b[i][j-1][k] b[i-1][j][k]	1,-1,0	b[i+1][j][k] b[i][j+1][k]	1,-1,0	b[i][j][k+1] b[i][j-1][k]	0,1,1
b[i][j+1][k] b[i-1][j][k]	1,1,0	b[i+1][j][k] b[i][j][k-1]	1,0,1	b[i][j+1][k] b[i][j][k-1]	0,1,1
b[i][j][k-1] b[i-1][j][k]	1,0,-1	b[i+1][j][k] b[i][j][k+1]	1,0,-1	b[i][j+1][k] b[i][j][k+1]	0,1,-1
b[i][j][k+1] b[i-1][j][k]	1,0,1	b[i][j+1][k] b[i][j-1][k]	0,2,0	b[i][j][k+1] b[i][j][k-1]	0,0,2

Tiled Sequential Code

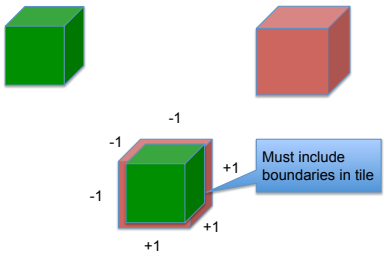
```
…
#define n 64
float a[n][n][n], b[n][n][n];
for (ii=1; ii<n-1; ii+=TI)
  for (jj=1; jj<n-1; jj+=TJ)
    for (kk=1; kk<n-1; kk+=TK)
 for (i=ii; i<min(n-1,ii+TI); i++)
  for (j=jj; j<min(n-1,jj+TJ); j++)
   for (k=kk; k<min(n-1,kk+TK); k++) {
     a[i][j][k]=0.8*(b[i-1][j][k]+b[i+1][j][k]+b[i][j-1][k]  +
                     b[i][j+1][k]+b[i][j][k-1]+b[i][j][k+1]);
   }
```


Relate Tiling Strategy to Computation/Data Partitioning

- Blocks can correspond to tiles (at least in 2 dimensions)
- Threads can correspond to 3-d loop nest
- Dealing with capacity limitations

Correctness: Boundaries

Compute 3-D tile Read Additional Data



Basic Structure of GPU Kernel

```
__shared__ b_l[][][];

// each dimension of a grid corresponds to a tile
// each dimension of a thread corresponds to i,j,k loop

for (portion of third dimension) {
  // copy single element to shared memory
  // copy boundaries to shared memory
  __ syncthreads();
  // compute Jacobi within tile
  __ syncthreads();
} // go to next tile in third dimension
```


Performance Expectations?

- Host-only original
- Global memory
- Shared memory

Tools: Occupancy Calculator

- Assists in calculating how many threads and blocks to use in the computation partitioning
  - Points to resource limitations
  - Points to underutilized resources
- Download from:
  - http://www.nvidia.com/object/cuda_programming_tools.html

Using the Occupancy Calculator

- First, what is the "compute capability" of your device? (see Appendix A of programmer's guide)
  - Most available devices are 1.1
  - GTX and Tesla are 1.3
- Second, compile code with special flag to obtain feedback from compiler
  - --ptxas-options=-v

Example of Compiler Statistics for Occupancy Calculator

```
$ nvcc --ptxas-options=-v \
-I/Developer/CUDA/common/inc \
-L/Developer/CUDA/lib mmul.cu -lcutil
```

Returns:

```
ptxas info    : Compiling entry function
  '__globfunc__Z12mmul_computePfS_S_i'
ptxas info    : Used 9 registers, 2080+1056 bytes smem,
  8 bytes cmem[1]
```


Next Tool: Profiler

- What it does:
  - Provide access to hardware performance monitors
  - Pinpoint performance issues and compare across implementations
- Two interfaces:
  - Text-based:
    - Built-in and included with compiler
  - GUI:
    - Download from http://www.nvidia.com/object/cuda_programming_tools.html

A Look at MMUL

Where are coalesced and non-coalesced loads and stores to global memory?

Another example

- Reverse array from Dr. Dobb's journal
- http://www.ddj.com/architect/207200659
- Reverse_global
  - Copy from global to shared, then back to global in reverse order
- Reverse_shared
  - Copy from global to reverse shared and rewrite in order to global
- Output
  - http://www.ddj.com/architect/209601096?pgno=2

MPM/GIMP Questions from Last Time

- Lab machine set up? Python? Gnuplot?
- Hybrid data structure to deal with updates to grid in some cases and particles in other cases

Some Strategies and Methodologies

- Note that README.txt tells how to run "impact.exe", using small number of cpp files
- Convert to single precision by replacing all "double" with "float"
- Discussion of data structures

  See updateContribList in shape.cpp and patch.h