CUDA II: The revenge

2/22/16
HW 4: Parallelizing simulator

• Part 1: Due Thursday morning
  Figure out how to parallelize this allocator
  (which function(s), what technique to use, etc)

• Part 2: Due next Wednesday morning
  Actually do it and submit your code
Final project

• 1-3 people
• Scoped as 2 HWs for 1 person

• Deadlines:
  – Part 1 (Friday 2/26 at 11:59pm)
    Project idea and plan
  – Part 2 (last day of class (Monday 3/7))
    Present project (what you’ve done and what you plan to do)
  – Part 3 (end of finals (Sunday 3/13))
    Final submission (writeup and code (if appropriate))
Project ideas

• Programming environments:
  – CUDA, OpenCL, MPI, MapReduce, Haskell, TBB, Cilk++
  – transactional memory

• New systems:
  – Slimfly, random links, Cloud, Xeon Phi

• Applications:
  – n-body, other parts of simulator, LINPACK, matrix mult, sparse matrices, graph traversal
Next topics

• Read HPC paper for next time and a parallel language after that

• After that (~2 meetings):
  – HPC: Building exascale systems
  – MapReduce/Hadoop
  – Data centers
  – Chapter 8 (linear algebra applications)
Recall: CUDA

• Language for writing code that runs on a graphics card (device) when called from the CPU (host)
• Allows a high degree of SIMD parallelism
• We saw three performance issues
  – Data transfer time
  – Thread divergence
  – Serialization caused by constant memory’s broadcast
Which line below contains an error?

```c
__global__ void kernel(int* arr, int size) {
    int id = threadIdx.x + blockIdx.x*blockDim.x;
    if(id < size)
        arr[id] = id;
}

int main() {
    ...
    int array[5];
    kernel<<<1,5>>>(array, 5);
    ...
}
```
Which line below contains an error?

```c
__global__ void kernel(int* arr, int size) { //A
    int id = threadIdx.x + blockIdx.x*blockDim.x; //B
    if(id < size) //C
        arr[id] = id; //D
}

int main() {
    ...
    int array[5];
    kernel<<<1,5>>>(array, 5); //E
    ...
}
```
Which of the following are reasons to use multiple threads per block?

A. Don’t get parallelism otherwise
B. Threads within a block can communicate via __shared__ memory
C. There is a limit on the number of blocks that can be launched in one statement
D. More than one of the above, but not all of them
E. All of A. through C.
Which of the following are reasons to use multiple threads per block?

A. Don’t get parallelism otherwise
B. Threads within a block can communicate via __shared__ memory
C. There is a limit on the number of blocks that can be launched in one statement
D. More than one of the above, but not all of them
E. All of A. through C.
Which of the following is a correct use of `__syncthreads()`?

I. ```
   for(i=0; i<10; i++) {
       ...
       __syncthreads();
   }
```

II. ```
    if(x[id] == 10) {
        ...
        __syncthreads();
    }
```

III. ```
    if(x[id] == 10) {
        ...
        __syncthreads();
    } else {
        ...
        __syncthreads();
    }
```

A. I only
B. II only
C. III only
D. Two of I, II, and III
E. I, II, and III
Which of the following is a correct use of \_\_syncthreads()?

I. for(i=0; i<10; i++) {
       ...
       \_\_syncthreads();
   }

II. if(x[id] == 10) {
       ...
       \_\_syncthreads();
   }

III. if(x[id] == 10) {
       ...
       \_\_syncthreads();
   } else {
       ...
       \_\_syncthreads();
   }

A. I only
B. II only
C. III only
D. Two of I, II, and III
E. I, II, and III
What scope of variables does CUDA not support?

A. Thread-local
B. Shared within a warp
C. Shared within a block
D. Shared by all kernels in a program
E. CUDA supports all of these
What scope of variables does CUDA not support?

A. Thread-local
B. Shared within a warp
C. Shared within a block
D. Shared by all kernels in a program
E. CUDA supports all of these
Matrix multiplication
When multiplying 2 $n \times n$ matrices, how many loads from memory are needed for the CPU to compute the 1$^{st}$ row of output? Assume no caching

A. $3n$

B. $n^2 + n$

C. $2n^2$

D. $n^3 - n$

E. None of the above
When multiplying 2 n×n matrices, how many loads from memory are needed for the CPU to compute the 1st row of output? Assume no caching

A. 3n
B. \( n^2 + n \)
C. \( 2n^2 \), which implies \( 2n^3 \) for the entire output
D. \( n^3 - n \)
E. None of the above
How many loads to compute the first row if $2n < \text{cache}_\text{size} < n^2$?

A. $3n$
B. $n^2 + n$
C. $2n^2$
D. $n^3 - n$
E. None of the above
How many loads to compute the first row if $2n < \text{cache\_size} < n^2$?

A. $3n$
B. $n^2 + n$, which gives $n^3 + n^2$ for the entire output
C. $2n^2$
D. $n^3 - n$
E. None of the above