Midterm Review and Term Project Overview

Carnegie Mellon University

18-645

Midterm and Final Exam Schedule

- **Midterm Exam – Monday, March 20th** **10%**
  - Pittsburgh 3:30pm ET Wean Hall 4623
  - Silicon Valley 12:30pm PT Bld 23, Room 211
  - China 1:30pm CT TBD
  - ***Modules 1, 2 (multicore) and 3 (manycore) of the course***

- **Final Exam – Monday, May 15th** **30%**
  - Pittsburgh 1-4pm ET TBD
  - Silicon V. 10:00am – 1:00pm PT TBD
  - China TBD CT TBD
  - ***All course content***

Term Project Schedule

- **Term (Team) Project** **20%**
- **Abstract - Friday, March 31st**
  - Describe what you propose to do for this project
  - Baseline performance, expected outcome
  - Email Abstract as PDF
    - **To:** 18645@sv.cmu.edu by 22:00 (PST)
    - **Subject:** Term-Project Abstract:
    - **CC:** all team members (using their <andrewID>@andrew.cmu.edu)
- **Final Report including short video - Monday, May 5th**
  - Final graded report: 5-10 pages
  - Short 5-10minute video (i.e. recorded presentation)
  - Email Final Report as PDF & Video and MP4

Term Project – Report (May 5th)

Apply skills learned in this course to a problem that is important to you

- Technical report describing your work (5-10 pages)
  - ½ page: Overview of the problem you tackled — **Why this problem?**
  - ½ - 1 page: Literature survey, related work — **What have others done?**
  - 1 - 2 pages: Detailed description of techniques — **What did you do?**
  - 1 ½ - 3 pages: Analysis of results — **What are the results?** **What do they mean?**
  - ½ - 1 page: Conclusion — **What is the take home message?**
- Size of report should reflect size of team
- Report could be basis for a conference submission

- Selected past projects will be posted online on March 10th

Homework and Project Schedule

- **Homeworks (infrastructure setup) 10%**
  - Module 2 – Manycore **Wednesday, March 1st**
  - Module 3 – Cluster Computing **Wednesday, March 29th**
- **Mini-Projects (Coding and Evaluation) 30%**
  - Module 2 – Manycore **Friday, March 17rd**
  - Module 3 – Cluster Computing **Friday, April 14th**

Midterm Exam (March 20, 2017)

- 45 minute midterm
  - Pittsburgh 3:30PM to 4:20PM, Wean Hall 4623
  - Silicon Valley 12:30PM to 1:20PM, B23 Room 211
  - China 12:30PM to 1:20PM, B23 Room 211

- Four Parts to the exam:
  - Part 1: Short questions
  - Part 2: OpenMP – code interpretation
  - Part 3: CUDA – code interpretation
  - Part 4: SIMD – code interpretation

Mid-term Exam (March 20, 2017)

- Four Parts to the exam:
  - Question 1: Short questions
  - Question 2: OpenMP – code interpretation
  - Question 3: CUDA – code interpretation
  - Question 4: SIMD – code interpretation

- Questions 1 and 2 compulsory
- Select either Question 3 or Question 4

One side of one page

- You may refer to 1-page of notes that you have individually prepared for this exam. These notes, however, must be attached to the exam on submission.

- If you use extra paper, be sure to mark it with your name, and the number of the corresponding question.

Short questions

- Describe a general problem in which OpenMP **can't** be applied, and why.
- Race conditions and false sharing are common problems in OpenMP code.
  - Write a small code snippet that demonstrates false sharing
  - Write a small code snippet that demonstrates a race condition
- I would like to improve the throughput of a CUDA kernel. What are the three areas I could look at to improve execution throughput? What tools could I use for measuring the performance?
- Any of the questions that you should be able to answer after viewing the course videos

Code Interpretation Questions

- The following code computes 1024 dot products, each of which is calculated from a pair of 256-element vectors. Assume that the code is executed on the GTX460. Use the code to answer the questions that follow.

```
01 #define VECTOR_N 1024
02 #define ELEMENT_N 256
03 const int DATA_N            = VECTOR_N * ELEMENT_N;
04 const int DATA_SZ  = DATA_N * sizeof(float);
05 const int RESULT_SZ         = VECTOR_N * sizeof(float);
. . .
06 float *d_A, *d_B, *d_C;
. . .
07 cudaMalloc((void **)&d_A, DATA_SZ);
08 cudaMalloc((void **)&d_B, DATA_SZ);
09 cudaMalloc((void **)&d_C, RESULT_SZ);
. . .
10 scalarProd<<<VECTOR_N, ELEMENT_N>>>(d_C, d_A, d_B, ELEMENT_N);
```

Code Interpretation Questions

- a) How many threads are there in total?
- b) How many threads are there in a warp?
- c) How many threads are there in a block?

Code Interpretation Questions

```
11
12 __global__ void
13 scalarProd(float *d_C, float *d_A, float *d_B, int ElementN)
14 {
15       __shared__ float accumResult[ELEMENT_N];
16       //Current vectors bases
17       float *A = d_A + ElementN * blockIdx.x;
18       float *B = d_B + ElementN * blockIdx.x;
19       int tx = threadIdx.x;
20
21       accumResult[tx] = A[tx] * B[tx];
22
23       for(int stride = ElementN /2; stride > 0; stride >>= 1)
24       {
25                __syncthreads();
26                if(tx < stride)
27                         accumResult[tx] += accumResult[stride + tx];
28       }
30       d_C[blockIdx.x] = accumResult[0];
31 }
```

Code Interpretation Questions

- a) How many global memory loads and stores are done for each thread?
- b) How many accesses to shared memory are done for **each thread block**?
- c) How many iterations of the for-loop (Line 23) will have branch divergence?
- d) Identify an opportunity to significantly reduce the bandwidth requirement on the global memory.
  - How would you achieve this?
  - How many accesses can you eliminate?