GPU-accelerated systems

סמסטר אביב, תשע''ד
2.7.2014

מרצה: מרק ויילברשטין

חברון סופי

מותר כל חומר כתוב או ממוחשב

אורד המבחנים: שעתיים

כל המ挥发ויות ייכתבו בתופעות המבחנים

שימור הלשון אלון פותחת ייצוגה בימות המורוז העק עד כל שאלות

שימור הלשון במורוזי בימות הבстыון בעב הלשון עד דיیر שלול
Question #1 (30 points)

You will implement a task of computing Euclidean distance between two vectors $A$ and $B$ with 1Mega-elements. Reminder: Euclidean distance between $A$ and $B$ is defined as

$$\sqrt{\sum_{i=0}^{N} (A_i - B_i)^2}$$

1. Write a GPU-accelerated program (CPU and GPU code) that implements this task using CUDA.

Assumptions:

1. Arrays $A$, $B$ and output variable $distance$ are in GPU memory.
2. You may use a library function $iscan()$ that performs parallel inclusive scan. It assumes at least 2 input elements per thread.
3. If necessary, use input arrays as temporary storage.
4. If necessary, add new kernels (in addition to those specified below) or call the same kernel multiple times.

/* computes inclusive scan */
__device__ void iscan(float* _input, float* _output);
__device__ float distance; // global variable to store the output
__global__ void euclidean(float* A, float *B) {

    // GPU code goes here

    cudaDeviceSynchronize();
    /* DONE */
}
2. Does the code YOU wrote have control flow divergence? If yes, mark that part of the code.

Answer:___________________________

3. Does the code YOU wrote perform uncoalesced memory accesses? If yes, mark that part of the code.

Answer:___________________________

4. The code will be invoked on a GPU with the following properties:
   Each SM has 32K Registers, 16K shared memory, can run up to 64 warps, up to 8 thread blocks.
   Assuming each thread consumes 32 registers, what is the GPU occupancy for this kernel. Write a single number

Answer:___________________________

5. Calculate the upper bound (in GFLOPs/s) on the theoretical kernel performance assuming a GPU with 200GB/s bandwidth to its own memory and 4TeraFLOP/s raw ALU performance. Write a single number

Answer:___________________________

6. If you could add new hardware resources to a GPU, which one or more from the list below are going to improve the performance of the Euclidean distance kernel, assuming the kernel and GPU properties as appear in (4) and (5)?

1. twice the size of shared memory
2. twice the number of registers
3. twice the number of ALUs
4. twice the cache size
5. twice faster atomics

Explain in 1 sentence without exceeding the allocated space.

Answer:__________________________________________
__________________________________________
__________________________________________
__________________________________________
Question #2 (5 points)

PTask paper claims that programs that use GPUs suffer from poor performance isolation

Choose one or more correct explanations offered in the paper:
1. GPUs are high-throughput processors
2. GPUs use hardware scheduler that is not under CPU control
3. The scheduler of GPU tasks in the driver is not aware of system-wide scheduling priorities
4. A task running on a CPU cannot run on a GPU, resulting in poor load balancing.

Question #3 (7 points)

Given two GPU threads T1 and T2 which access global variables a, b and c and execute the following code.

```c
__device__ volatile int a,b,c;
Assume that a, b and c are initialized to 0.

if (threadId==T1){
    a=2;
    b=3;
}
else (threadId==T2){
    c=a+b;
}
```

Choose one or more correct answers:
1. c **definitely** equals 5 if T1=T2
2. c **definitely** equals 5 if T1!=T2 and T1 and T2 are in the same warp
3. c **can be** any of 0,2,3,5 if T1 and T2 are in the same thread block
4. c is **definitely** 0 if T1 and T2 are not in the same thread block
5. c is **definitely** 5 if T1 and T2 are in different kernels executed one after another (T1 first, T2 last)
Question #4 (5 points)

Consider a network server which counts the number of times the word "GPU" appears in all packets. Network bandwidth: 100GB/s, PCIe bandwidth: 16GB/s, GPU memory: 300GB/s, GPU ALUs can perform 300 Giga-comparisons/second, CPU ALUs can perform 100 Giga-comparisons/second

Which system parameter(s) are missing in order to determine whether using GPU is more efficient than using CPU? Specify the range of values for which it is worth using a CPU rather than GPU

<Parameter1> less than/equals/more than <value1>
<Parameter1> less than/equals/more than <value2>
Explain

Answer:

Question #5 (5 points)

In Rhythm the authors suggest to transpose the input data to achieve better performance. SSLShader, on the other hand, does not perform these optimizations. Why? The answer should be 1 sentence and should be phrased as:

"SSLShader X, X, X, X while Rhythm Y, Y, Y, Y"

Answer:

Question #6 (7 points)

List at least three significant problems with the Shredder paper as discussed in the class.

1.
2.
3.
Question #7 (10 points)

Consider a single-GPU system in which a GPU is allowed to read and write data directly from/to a hard disk, bypassing CPU memory entirely. The goal is to rewrite GPUfs to take advantage of this functionality. Assume that we want to support only the following usage pattern: both the CPU and GPU perform writes and reads to the SAME file. However the accesses are not concurrent – they are interleaved. Namely, CPU opens, reads/writes, closes, GPU opens, reads/writes, closes, CPU.. GPU.. etc.

In order to support the close-to-open GPUfs consistency model, write the necessary and sufficient condition using the following basic actions.

(a) GPU open() must invalidate CPU buffer cache
(b) GPU close() must write back to disk and invalidate GPU buffer cache
(c) GPU open() must invalidate GPU buffer cache
(d) CPU open() must invalidate GPU buffer cache
(e) CPU close() must write back to disk and invalidate CPU buffer cache
(f) CPU open() must invalidate CPU buffer cache

Write a single expression using AND and OR. For example, ((a) AND (b)) OR ((c) AND (d))

Answer:

Question #8 (10 points)

Consider a ring buffer for producer - consumer interaction. The following building blocks are given

(a) write value to the array
(b) update 'tail'
(c) update 'head'
(d) read value from the array
(e) check is_empty();
(f) check is_full();

Write produce() and consume() functions using the blocks above while adding the minimum amount of memory fence operations. For example:

(a), MF, (b), MF, (c), MF, etc..

produce(new_data){__________________________________________________________}

new_data consume(){__________________________________________________________}
Question #9 (5 points)

Choose one or more correct answer

PTask system eliminates
1. redundant memory copies
2. redundant GPU invocations
3. redundant GPU preemptions
4. none of the above

Question #10 (5 points)

Two applications P1 and P2 periodically execute GPU kernels G1 and G2, consuming T1 seconds and T2 seconds respectively.

GDEV enables to prevent:
1. starvation of P1 if T1==T2
2. starvation of P1 if T1 << T2
3. starvation of P1 if T1 >> T2
4. GDEV does not prevent starvation for periodic kernel invocations.
Question #11 (13 points)

Assume you are building a system with two GPUs, G1 and G2, using CUDA. Each GPU is used to execute a single node in a data flow graph having two connected nodes. Both G1 and G2 run their respective subtasks in time $T$. Data transfers over PCIe, and CPU-CPU memory copy take time $T$ as well. An application may contain from one to thousands of tasks which pass through the data flow. The application runtime is computed as the time between the moment the first task enters the data flow till the moment the last one leaves it.

A GPU expert claims that using pinned CPU memory as a temporary buffer for moving data between GPUs is not necessary because pinned memory does not affect the system throughput since computations and communications are perfectly overlapped.

Prove him right or wrong.
Hint: compare the runtime of an application with/without pinned memory as a function of the number of tasks in it. Make sure to consider real implementation constraints when using CUDA.