COMPUTAÇÃO DE ALTO DESEMPENHO

2017/2018

Exam

26/06/2018

1. Consider the problem of finding the index where a given array as its minimum value, usually called the argmin operation:

   argmin(array) = index i where array[i] is minimum

   a. Provide the pseudo-code of a parallel algorithm for computing argmin. Assume the existence of the constructs in appendix A.
   b. What is the expected maximum speedup of your solution?

2.
   a. Provide the pseudo-code of a parallel algorithm for counting the number of primes numbers up to a given number N. Assume the existence of the constructs in appendix A and of a function isPrime(n) that checks if number n is prime.
   b. Your solution requires a dynamic mapping of the parallel tasks to the available parallel hardware, or can this mapping be computed beforehand, like in a parallel for loop? Justify your answer.

3. Implement in CUDA a kernel that, given an array of ints `arr` and a threshold `t`, sets to `t` all the values of `arr` lower than `t`. For example, applying the kernel to array {2, 3, 5, 7, 8, 9, 0, 2, 3} and threshold 4, outputs array {4, 4, 5, 7, 8, 9, 4, 4, 4}. Assume the existence of the constructs in Appendix B.

4.
   a. Implement in CUDA a kernel to compute the argmin of a given array (assume the existence of the constructs in Appendix B).
   b. Supply the host code for calling the kernel for a given array `my_array`, as well as all the required memory transfer operations.

5. Explain how a grid of threads running a CUDA computation are scheduled by a GPU into its hardware elements.

6. What is a distributed memory architecture? Is it possible to have a shared memory programming model (such as OpenMP) on top of such type of architecture? If so, how, and which kind of performance problems may arise?

7. In the context of the Spark distributed parallel computing framework, implement a function that, given two RDDs of integers, returns an RDD with all elements that appear in both input RDDs. For example, applying the function to RDDs with contents {1, 2, 3, 4, 5} and {3, 4, 5, 6, 7} returns an RDD with content {3, 4, 5}.

8. In the context of the Spark distributed parallel computing framework:
   a. Implement a function that computes the transitive closure of a directed graph. The graph is represented by an RDD of edges — pairs (Integer, Integer) that represent edges between the two nodes in the pair.
   The transitive closure must produce an RDD with pairs for all nodes (A, B) for which there is a path from A to B. For example, applying function to {(1, 2), (2, 3), (3, 4)} must output
   {(1, 2), (2, 3), (3, 4), (1,3), (2, 4), (1, 4)}
   Note: Clearly indicate all constructs that you assumed to exist.
   b. In your solution, how many tasks are executed by each worker? Could your solution be optimized to reduce such number of tasks? Justify your answer.

APPENDIX A

`spawn(function_name, arguments)`

creates a new task to run `function_name(arguments)`

```
parallel for (initialization; booleanExpression; update) { codeblock }
```

acts like a regular for loop, but the iteration space is divided among a predefined number (p) of processors

`barrier()`

synchronizes all the workers currently running a parallel section, e.g. a parallel for

APPENDIX B

shared a

allocates variable a in shared (thread block local) memory

`syncthreads()`

synchronizes all threads in the current thread block

```
kernel_function<<<NUMBER_BLOCKS, BLOCK_SIZE, LOCAL_MEMORY, STREAM>>>(arguments)
```

signature of kernel launching in CUDA

```
cudaError_t cudaMemcpy(void* dst, const void* src,
                       size_t count, cudaMemcpyKind kind, )
```

asynchronous data transfer between host and GPU (or vice-versa)