Final, Parallel Programming (EN 620.320/420/620), Fall 2020

December 14, 2020 @ 8 am – December 18, 2020 @ 5 pm.

This is an untimed exam. It is expected to take 2-3 hours to complete. You may use any static online or written resources to research the topics raised in the questions. The answers that you give must be **solely your own**, written and prepared by you individually. You may not discuss your answers to the exam with anyone, including other students in the class, past students, and in online forums. Because this is an open book untimed exam, the questions are designed to require thoughtful answers. In some cases, the answers will not be obvious. Some questions may require additional research.

Each question mark indicates that a response is required. Look for directives such as **Explain**, **Define**, or **Give** and follow these instructions. Please keep your answers as brief as possible. Answers that include extraneous facts and irrelevant details will lose credit even if a correct answer is included in the response.

The assignment should be submitted to GradeScope within the specified time. ‘

1. (*15 pts*) On the implementation of Spark reduceByKey(). Consider the following two lines of code that compute the per key aggregate:

```
rdd.reduceByKey((a, b) => a + b)

rdd.groupByKey().map((x,y) => (x,sum(y)))
```

   (a) (*10 pts*) Would you expect the reduceByKey or groupByKey implementation to be more efficient? Explain why. (Your explanation will likely need to draw upon external source material.)

   (b) (*5 pts*) How is this analogous to the concept of a combiner in map/reduce?

2. (*20 pts*) Read the article on deadlock on the Mars pathfinder `http://www.cs.cornell.edu/courses/cs614/1999sp/papers/pathfinder.html`. Explain why this situation meets all the criteria for a deadlock. You should identify the specific processes, resources, and actions that fulfill the properties for deadlock in this specific example.

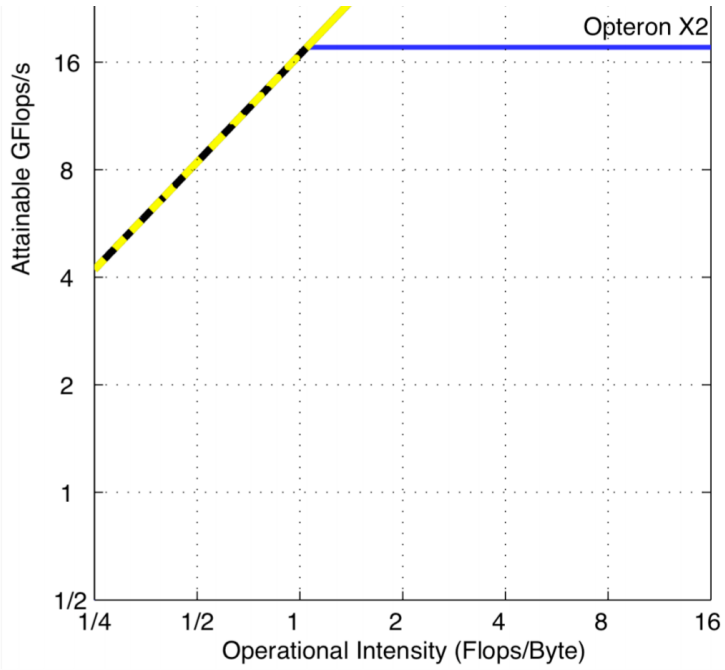
   (a) Mutual exclusion.

   (b) Circular dependency.

   (c) No preemption.

   (d) Hold and wait.

3. (*15 pts*) On the following roofline plot. The roofline is the dashed line on the diagonal that turns into the horizontal line. Give brief answers to the questions, using one or two sentences.



(a) Place the lines for two computational kernels, one that is bound by the off-chip memory throughput and one that is processor bound.

(b) What does the corner of the roofline represent?

(c) Why is it desirable to have computational kernels as close to that corner as possible?

4. (*10 pts*) Understanding MPI. The following questions refer to the LLNL MPI tutorial (`https://computing.llnl.gov/tutorials/mpi/`).

   (a) (*5 pts*) In the context of running on multicore, the tutorial states "The programming model clearly remains a distributed memory model however, regardless of the underlying physical architecture of the machine." Why is MPI unable to take advantage of shared memory?

   (b) (*5 pts*) Read the section entitled "Groups vs. Communicators". Give an example of a program that would desire to use multiple communicators within the same application. Our simple examples in class used only MPI_Comm_World.

5. (*20 pts*) The iterative step to my solution to k-means in Spark (project 5) looked like.

```
1. for i in range(iterations):
2.     clusters = points.map(lambda x: assign_class(x, centroids))
3.     ptstriples = clusters.zip(points)
4.     means = ptstriples.groupByKey().map(lambda x: (x[0], takemean(x[1])))
5.     meansit = means.sortByKey().map(lambda x: x[1]).collect()
6.     centroids = np.array(list(meansit))
```

   Spark (like dask) builds an execution pipeline that performs lazy evaluation and can pipeline execution. So, the steps are not run one after another with a barrier in between (see the animation in Lecture 11). However, Spark and dask will implement barriers based on data dependencies, i.e. all partitions in a subsequent step depend on the completion of all partitions in the previous step. We only covered barriers in passing in class. Wikipedia gives an adequate description `https://en.wikipedia.org/wiki/Barrier_(computer_science)`.

   (a) (*5 pt*) The computation of centroids in the last line must be completed prior to its use in assign_class. What is the datatype of centroids? Is it parallel?

   (b) (*5 pts*) Centroids needs to be sent to all partitions of the points RDD. What kind of collective communication operation is this? Explain briefly. Refer to the collective communication patterns in `https://computing.llnl.gov/tutorials/mpi/#Collective_Communication_Routines`.

   (c) (*10 pts*) In lines 2-4, there is a barrier. Spark must complete all partitions of a previous computation before starting the next function. What two steps are these? Why is it necessary to complete all of the previous step complete before the next step begins?