Austin Bomhold
Simarpal Singh
Steven Golob
TCSS 562 Project Proposal
Oct 25, 2024

I. Description of the proposed cloud application

In this project, we will study the effects of implementing MapReduce under different architectures. We will use MapReduce to perform several dissimilar computations that can be *embarrassingly parallelizable*, and evaluate how parallelizing the computation tasks under the different MapReduce designs impacts performance.

Specifically, is it quicker to split the jobs across 20 serverless functions or 1000? At what point does the overhead of spawning a new sandbox and MapReduce instance outweigh the gains of more concurrency? Additionally, is splitting the job across 1000 serverless functions more efficient than 20 containers? What about 1000 containers versus 20 serverless functions? Can we graph how the performance changes with the number of MapReduce threads? In doing so, can we measure the overhead of MapReduce or the overhead of launching AWS services under this evaluation framework? And, even if one design might be more efficient, is it cost effective? That is, does launching *N* times more resources result in the job running exactly *N* times quicker?

The choice of which computations to parallelize is discretionary, and only serves the purpose of stressing the resources. For example, as our parallelizable computation, we could do many different types of tasks, e.g.:

a. Use an image classifier to classify images.
b. Count the occurrences of a word in a large text file.
c. etc.

We will choose two or three of these tasks, and perform each of them under all of our MapReduce designs, and compare how they perform. It would be interesting to see one MapReduce design work better for one task, and another MapReduce design work better for another. The goal then would be to determine why.

With learning as our primary goal, we will also have data that the parallelizable tasks use stored in S3, and have our MapReduce designs interact with S3 to get the data.

II. Design-tradeoffs

Here we describe the different MapReduce designs we are interested in evaluating.

1. From a serverless function, simply implement MapReduce instances as serverless function instances.
2. From a serverless function, instantiate several containers to run MapReduce. (To use the additional space in a container, can we parallelize MapReduce in separate threads in one container.)

Other design trade-offs:
1. Vary the size of the microservices and the containers.
2. Vary the maximum number of times we split the task in MapReduce.

III. Proposed evaluation metrics

For the various MapReduce designs, we can evaluate:
1. The total cost in AWS credits
2. The overall Runtime
3. Memory usage
4. Overhead runtime of mapreduce mechanisms vs running it in a single-thread paradigm

IV. Work plan

Week 1: Oct. 27 - Nov. 2	● Complete Tutorial 3 as a group so that we all are comfortable with creating EC2 instances, and so that we can collaborate on designing the details of our architecture. ● Revise the project proposal based on the instructor's feedback.
Week 2: Nov. 3 - 9	● Survey related works to incorporate in project. ● Complete tutorial 4 to become comfortable with AWS Lambda, creating our own serverless functions, and using the Java/Python evaluation tool.

Week 3: Nov. 10 - 16	● Implement all tasks in the MapReduce paradigm. ● Get the serverless endpoint live.
Week 4: Nov. 17 - 23	● Get the MapReduced task working across N containers. ● Begin evaluations.
Week 5: Nov. 24 - 30	● Continue experiments. ● Begin writing paper. ● Revise and rerun experiments as needed.
Week 6: Dec. 1 - 7	● Complete final experiments. ● Work on paper.
Week 7: Dec. 8 - 14	● Prepare short presentation. Deliver presentation on Dec. 12. ● Finish writing, revise, edit, and build graphs for project paper.

V. References