4-6-16
HW4
Rich von der Schmidt
CS516

PART 1:

1)

app-20160406183908-0030

cores: 10

memory per node: 6.0 GB

duration: 3.4 min

2)

A) See screen shots attached in folder.

B) The sort by key took longest. This makes sense in principle because sorting anything is more costly computationally than say just changing values iteratively.

C) The application is divided into multiple jobs because this is precisely what makes this process efficient. Rather than having a line of nodes waiting for the node ahead to finish, each can work on "cooking his own meal" so to speak.

Division of labor as they would say in Economics.

3)

A) See screenshots attached

B) Again, if you can divide labor further, this is better. It resembles Henry Ford's assembly line where lots of workers work on one tiny part of the big machine that produces cars. Being able to discretize the work that needs to be done ensures a continuity of work flow- there are no blocking calls, and no sockets that listen indefinitely. (Or, that's what we would like to accomplish)

C) Skipped stages are from cached or persisted values that don't need to be recomputed. Hooray for memory!

4)

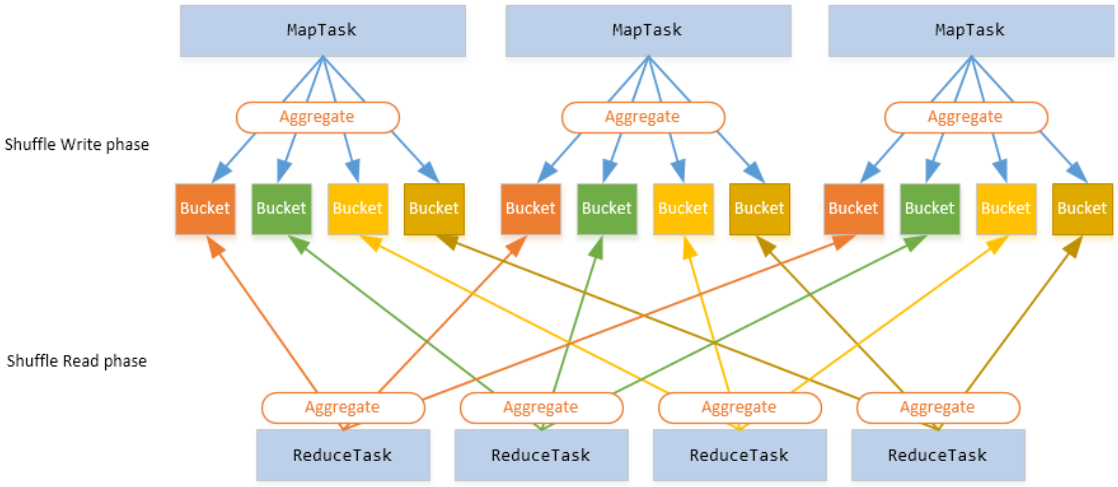
A, B) See attached screenshots

C) Input size/records of task

D) Shuffle read size /records of the task

E)Shuffle write size/ records of the task

We are taking in a certain amount of data, like water into a reservoir. So naturally, the greater the input size, the more work that will be need to be done in a system. The shuffle read and write sizes are indicators of which executor is doing which kinds of work. Pictures speak a thousand words.



(This was for the record, a pretty amazing concept, and this picture distills what is being done at all levels in our programs. We discretize data, only to compress it later based on the boundaries we have created. It is a process of simultaneous creation and destruction that allows for an enormously efficient flow of data.)

5)

The work given to the executors is almost uniform, with the exception of a small handful of executors that handle a much greater load of the work. (27 mb workers vs 161.9 mb) and this division of labor sort of resembles ancient armies, where you have foot soldiers and cavalry. You have a few of the soldiers who lead in battle on horse and are able to do more damage, and then foot soldiers, without the benefit of a beast of burden, will naturally be less effective individually, but will constitute a greater mass when they are summed up beyond their parts.

(This is the best learning experience I have had here at Duke. Sorry I could not get through the graphs, I wanted to make sure I was getting the class material bit as much as I could since I also get tested on that. I will cherish this code for quite some time.)