Elliot Smith
Research 4 - Spark

1. The key need for this algorithm approach is that the current, popular solution to performing large-scale computation and data aggregation, MapReduce, is limited in that it is built to function in an acyclic data flow, and as such, many popular algorithms that rely on the intermediate results of iterations cannot be utilized. For example, with many machine learning algorithms, gradient descent is employed to iteratively find the minimum point on a convex curve, the point that minimizes a particular loss function. MapReduce is not able to be employed to learn via gradient descent in an optimal manner because the data currently being iterated upon cannot be reused across multiple operations in the way that Spark is able to.

2. The key object that is the solution to this need is called the RDD, or resilient distributed dataset. An RDD is a read-only collection of objects that can be divided up and distributed to several different machines for parallel processing. The key feature of RDDs is that a user may cache an RDD in memory across several machines for use in multiple parallel operations. The power here is that once cached, several different machines that may be doing very unique tasks are able to access this cached dataset at will; this will allow for massive improvement in time to completion for tasks that need to iteratively update data and be able to access the updated results frequently.

3. The authors have proposed the following next steps:

    a. Formally document the purpose and properties of RDDs and Spark’s other abstractions in regards to tasks that they may solve
    b. Expand upon RDDs abilities to trade between storage costs and re-construction costs
    c. Define new operations to transform RDDs, including a “shuffle” operation that allows for partitioning an RDD in a user-defined way
    d. Provide methods through which other programs may interact with the Spark interpreter, such as through SQL and R

4. Something interesting that I learned from the paper is really just how Spark itself works and how it is constructed to take advantage of the ability to cache datasets. Particularly enlightening to me was to see how very often, the first iteration using MapReduce and the first iteration using Spark are often very similar in terms of time; they tend to take about the same amount of time. However, subsequent runs for Spark are substantially faster because of this ability to cache data and share it across multiple machines. To me this is the most interesting and powerful component of the Spark model, the ability to perform multiple parallel processes is very powerful; however, the ability to access a

dataset between multiple machines that are working on the same task is even more powerful and a truly incredible advancement in computing technology.

5. Venkataraman, Shivaram; Yang, Zongheng; Liu, Davies; Liang, Eric; Falaki, Hossein; Meng, Xiangrui; Xin, Reynold; Ghodsi, Ali; Franklin, Michael; Stoica, Ion; Zaharia, Matei. SparkR: Scaling R Programs with Spark. ACM. SIGMOD. July 2016.

   My paper relates to this one in that they have taken the Spark infrastructure and created an R package that takes advantage of the Spark distributed computing engine to perform large-scale data analysis and aggregation. While R, a primarily statistical programming language, is single-threaded and can only process data that fits into the machines memory, this package will allow R users to circumvent this shortcoming and perform large-scale data processing. The new information I learned from reading is that his package is available for use! As a Statistics student here at Rice, I spend a lot of my time working in R; however, computation in R is often unbearably slow for memory-intensive tasks. I plan on taking some time to play around with SparkR now that I have learned of its existence and leverage it when I work on some future projects!