1. What is the key need for this algorithm/approach/system? What are some short comings of existing approaches?

   MapReduce and its variants are very popular to implement large-scale data-intensive applications. Nevertheless, the system had been built before around acyclic data flow model that was not very useful for other common applications. Therefore, the new framework, Spark, shows up. Spark can support iterative machine learning algorithms and interactive data analysis tools while having scalability and fault tolerance to accomplish large-scale data-intensive applications.

2. What is the key object (term) that is the solution to this need? (e.g. tweet, RDD, Tensor) Describe this object in a few sentences. This is the fundamental concept or "thing" proposed to solve the need addressed in question #1.

   Resilient Distributed Dataset (RDD) and two restricted types of shared variables: broadcast variables and accumulators are the key object in Spark. RDD is fault tolerance such that it could compute missing or damaged partitions because of node failures. RDD can also be distributed with data on multiple nodes in a cluster. Due to the attributes of RDD, it can implement parallel computing and partition records in order to make them distribute across nodes in a cluster.

3. What has the author identified as a weakness or limitation of the proposed algorithm / approach / system? Or what has the author proposed as next steps? If the author does not provide this information, what do you think could be improved?

   The author mentions that the RDD abstraction cannot allow users to trade between storage cost and re-construction cost. Moreover, in-memory capacity would be very important issue since keep all data into memory would be very expensive. Spark requires lots of RAM to run in-memory. Therefore, the cost of Spark is very high. Lastly, Spark does not have file management system, thus it should rely on other platforms like Hadoop.

4. What is something interesting your learned from this paper, or your thoughts about its strengths and/or weaknesses. Is there anything else interesting about this paper, or your interpretation of it that you want to share?

   I learned lots of knowledge about spark such as lazy evaluation. Lazy evaluation will wait for the instructions before providing final result, and it saves significant time. That is, the data inside RDD does not allow to do any instructions or be transformed until the action, that triggers the execution, is executed. Spark also have dedicated tools

for streaming data, i.e. spark streaming. It can satisfy the need of the real-time processing of data continuously and concurrently. However, the consumption of memory will be the weakness of Spark.

5. Read an additional related paper. Provide a citation for this paper. How does this paper relate to the assigned one? What new information did you learn by reading it? This answer should be a decent sized paragraph describing the content of this paper (be specific) and how it relates to the assigned paper.

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica

University of California, Berkeley, 2012

In this paper, it introduces how RDD works and what features it has. It really helps me understand how to use RDD in Spark. What transformations and actions it has and the example in the paper provide me with the illustrations to build PageRank. Besides, the authors also discussed what types of applications are not suitable for RDDs. Therefore, when choosing the abstraction, I will understand what should avoid using RDDs. The usage of programming interface and operations is also very comprehensive in the paper. Because the RDD is the key object in Spark, this paper gives the whole picture of this abstraction and teach me how to use it into big data applications on Spark. Thus, I learn further knowledge from this paper and understand the general ideas from previous paper, Spark.