Maria Mahbub

Big Data Analytics CS690 (001), Fall 2018

University of Tennessee, Knoxville

The **main purpose** of the paper- "Spark: Cluster Computing with Working Sets" is to provide an enriched cluster computing framework named Spark, which can efficiently be used for some of the applications that cannot be easily run with programming models such as: MapReduce and Dryad.

The **key question** the author is addressing is efficiency in reusing same datasets over and over again. MapReduce and its variants are quite successful in cluster computing using larger datasets with their fault tolerance and scalability. However, when an iterative or interactive computation is to perform, the time needed for fetching data from the disk for each computation causes significant amount of latency, resulting into their low performance in multiple parallel computations where data need to be reused.

The **method** used to answer this key question is Spark's programming model which resolves the aforementioned issue while retaining two important properties of MapReduce: scalability and fault tolerance. It uses a read only collection of objects- termed as Resilient Distributed Datasets (RDD), which are distributed across the clustering nodes, have the properties to be cached explicitly and can be rebuilt if needed. In the paper, the author describes four ways to construct RDDs. Once these datasets are defined, a set of parallel operations are performed on them. Among the several operations, reduce, collect and foreach are mentioned in the paper with fair description. Spark also supports two restricted types of shared variables: Broadcast variables and Accumulators. The author has also mentioned the ease of using Spark by means of its integration into Scala allowing the user to use Spark from Scala interpreter.

The **testing approach** used to prove the effectiveness of the method is the performance analysis of three sample Spark programs provided in the paper: text

search, logistic regression, and alternating least squares. In text searching example, the author describes a program to count the lines of errors in a big set of logs in a file system. This program creates distributed datasets containing collection of lines and then transform them to find only the error messages in it. However, the author asserts that for speeding up any subsequent operations on these RDDs, they best way is to cache them into memory. The next example program implements logistic regression -an iterative classification algorithm in machine learning in order to find the best fitted hyperplane separating two sets of points. In this program the author shows that as this algorithm uses sum of the hyperplane defining function iteratively to predict the best fit, performing this over the cached data and by using accumulator variable will provide much efficiency to the computation. Finally, the author explains the importance of parallelizing and broadcast variables by providing an instance of Alternating Least Squares (ALS), which is used for collaborative filtering problems. However, while predicting unknown matrix elements from the known ones, all the steps involved in computation use a common variable. By using Spark, the author shows that this variable can be assigned as the broadcast variable and thus the time required to resend it to each node at each step gets saved. The author also provides some codes and a graphical representation to explain Spark's efficiency over Hadoop. While performing logistic regression Hadoop takes 127s for each iteration because of task independency, Spark takes 174s for the first one and 6s for subsequent ones. The author evidently states that because of reusing cached memory Spark performed 10 times faster than Hadoop. He also illustrates the efficiency due to the interactive property of Spark with an example of loading a 39GB dump of Wikipedia in memory across 15 clustering nodes.

The **main conclusion** in this paper is whereas MapReduce, Dryad fail to provide efficiency in iterative and interactive computations, Spark thrives in these areas despite having limitations in its model abstractions. The author also includes that the idea behind reconstructing RDDs may evolve into some more useful abstractions for the future programming clusters. Last but not least, the author concludes the paper with some brief discussions on four areas where further focus will be provided.

The **implication** of the author is: Spark performs comparatively slower than other frameworks such as Hadoop for the first iteration due to Scala execution speed. Nevertheless the performance of subsequent iterations of Spark is 10 times faster than other frameworks because of its cache mechanism.
However, while reading this paper I felt that there is some lacking in information on how recovery of RDDs works, how fast can it work and what percentage of it can be retrieved from the remaining ones. It would be better if the author included more discussions about them.

Complete the evaluating reasoning below:

	Elements of reasoning	Yes	No	NA
1	**Purpose:** What is the purpose of the reasoner? Is the purpose stated clearly or clearly implied? Is it justifiable?	yes		
2	**Question:** Is the question at issue well stated? Is it clear and unbiased? Does the expression of the question do justice to the complexity of the matter at issue? Are the question and purpose directly relevant to each other?	yes		
3	**Information:** Does the writer cite relevant evidence, experiences, and/or information essential to the issue? Is the information accurate? Does the writer address the complexities of the issue?	yes		
4	**Concepts:** Does the writer clarify key concepts when necessary? Are the concepts used justifiably?	yes		
5	**Assumptions:** Does the writer show sensitivity to what he or she is taking for granted or assuming (insofar as those assumptions might reasonably be questioned)? Does the writer use questionable assumptions without addressing problems that might be inherent in those assumptions?	yes		
6	**Inferences:** Does the writer develop a line of reasoning explaining well how s/he is arriving at her or his main conclusions?	yes		
7	**Point of View:** Does the writer show sensitivity to alternative relevant points of view or lines of reasoning? Does s/he consider and respond to objections framed from other relevant points of view?	yes		
8	**Implications:** Does the writer show sensitivity to the implications and consequences of the position s/he is taking?	yes		

Overall, is this a well-written paper? Briefly substantiate your answer.

In my view, this is a well written paper mainly because of the following reasons. First, there is a clear statement in the beginning of the paper stating and explaining the problem which the paper seeks to solve. Second, it contains clearly stated analytical framework with some basic concepts necessary for understanding the paper. Third, the paper articulates the type of research design with some detailed examples by referencing to earlier related studies and works. Fourth, the paper contains sufficient information for evaluation by providing evidencing comparison with some relevant works. Finally, at the end, the paper concludes with some related previous works and some future possible works, along with an ending remarks of the paper contents. In a nutshell, the task of reading and summarizing it was easy because of the content flow of the paper, which inevitably makes it a well-written paper.