Maria Mahbub

Big Data Analytics CS690 (001), Fall 2018

University of Tennessee, Knoxville

The **main purpose** of the paper- "MapReduce: Simplified Data Processing on Large Clusters" is to introduce Map Reduce programming model and how it works easily with large sets of data on a large cluster of machines with high performance.

The **key question** the authors are addressing is how to process huge amount of raw data in order to compute different kinds of derived data in a simplified manner. As processing this large amount of data needs hundreds of special computations, this raises complexities in codes. To alleviate these complexities a simplified approach is needed.

The **method** used to answer this key question is the MapReduce programming model which deals with the aforementioned complexities by employing a new abstraction inspired by map and reduce. This model expresses only the simple computations by hiding the complexities in a library. The computation is expressed as two functions: Map and Reduce. The Map function produces a set of intermediate key-value pairs from a set of input key-value pairs written by the user, and then the MapReduce library groups the values together depending on the same key. After that map passes it to reduce where the values get merged to the possible smallest one. However, the execution of this programming model takes place over a distributed cluster of machines, classified as master and workers. In this paper, the authors have also mentioned two different approaches to substantiate the performance of the method. One of them is to look for a particular pattern by searching around one terabyte of data and the other one is to sort that amount of data.

The **testing approach** used to prove the effectiveness of the method is the performance analysis of two computations. Each of these have been designed with

two 2GHz Intel Xeon processors with Hyper-Threading enabled, 4GB of memory, two 160GB IDE disks, and a gigabit Ethernet link. The first example program is grepping. For a particular rare pattern, this program grepped across 10^{10} 100-byte records and has only 92,337 hits. Likewise, a large sort benchmark, modeled after the TeraSort benchmark sorted 10^{10} 100-byte records. The grep was simply looking at the total time that it took to complete the entire MapReduce process, whereas the sort was also looking at the effect of running duplicate tasks towards the end of the MapReduce computation as well as looking at the effect of killing several of the worker nodes during the process. The authors have used some graphs to explain the overall performance of these two computations. In grep, 15000 input files were reduced to only 1 input files. The computation peaked when 1764 machines were in use and at that point, the transfer speed was 30GB per second. The whole computation took only 150 seconds but more than a minute of that was accounted for opening Google file system. On the other hand, the authors have presented a more interesting analysis of the larger sorting experiment; specifically, they have looked at custom partitioning and also observed two special cases: 1) performance of the model when backups were turned off to avoid stragglers and 2) when 200 machine failures were induced. The authors have shown that at the time of normal execution, the entire application took only about 850 seconds which increased up to 1283 seconds without any backup tasks and with 200 induced machine failures the entire computation was completed only in 933 seconds. To summarize the findings from these graphs it can be said that while the backups were turned off, the execution pattern was very similar to the normal execution pattern, although the last five tasks in the former one took almost 30% of the total execution time.

The **main conclusions** in this paper are as follows: MapReduce is a restricted programming model allowing it to automatically provide fault tolerance, transparent parallelization and load balancing to the user. This is also easy to use, applicable to a wide variety of cloud-based application and distributed applications, scalable as evidenced by its use in the large datasets at Google. The authors have been able to show that parallelization and distributed computations are easier when a restricted programming model is used. Likewise, they have illustrated that network bandwidth is an extremely scarce resource and also redundant execution can alleviate machine failure or straggler problems.

The **implications** of the authors are: MapReduce programming model not only simplifies the coding process but also reduce the amount of time taken to create analytical routines. Its highly scalable nature enables to run applications using thousands of terabytes of data. It has also reduced the cost of storage and data processing. Last but not least, the model can skim through terabytes of unstructured data in a matter of minutes.

However, while reading this paper some questions came into my mind which I would like to address here: 1) MapReduce can be unsuitable for real-time processing and it’s always not very easy to implement each and everything as MapReduce program. 2) Also, it is not an appropriate choice when the intermediate processes need to communicate with each other, or 3) streaming data need to be handled ,or 4) the processing requires a lot of data to be shuffled over the network.

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?			NA
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, it addresses an issue at the very beginning stating the complexity of an existing method and gradually goes deeper into the analysis resolving it and substantiating with proper experimental examples. Second, it describes the programming model in a fairly understandable way with some very basic key concepts, adds some examples of some computations where the model can be used and gives a clearer view of the model execution. Third, the paper compares two computational experiments to show the effectiveness of the programming model. It also states some works that have already been using this type of programming model. Finally, the paper has drawn its conclusion by showing some practical use of the programming model at Google along with some well-grounded statements of the effectiveness and usefulness of this model. In a nutshell, the paper is well written and gives a good overview of the issue. I believe that this will be very useful to anyone who is relatively new to MapReduce programming model. This paper can be a good start for them.