Chapter 7

Baran:
An Effective MapReduce-Based Solution to Solve Big Data Problems

Mohammadhossein Barkhordari
Information and Communication Technology Research Center, Iran

Mahdi Niamanesh
Information and Communication Technology Research Center, Iran

Parastoo Bakhshmandi
Information and Communication Technology Research Center, Iran

ABSTRACT

The MapReduce method is widely used for big data solutions. This method solves big data problems on distributed hardware platforms. However, MapReduce architectures are inefficient. Data locality, network congestion, and low hardware performance are the main issues. In this chapter, the authors introduce a method that solves these problems. Baran is a method that, if an algorithm can satisfy its conditions, can dramatically improve performance and solve the data locality problem and consequences such as network congestion and low hardware performance. The authors apply this method to previous works on data warehouse, graph, and data mining problems. The results show that applying Baran to an algorithm can solve it on the MapReduce architecture properly.

1. INTRODUCTION

According to data volume growth in information systems, social networks and sensors, it is necessary to design and implement systems that can manage this huge amount of data and be capable to analyze them. Huge data may have other specification too. Velocity can be another property. If data do not process in a specific time, it will not have any value. For example, patient data that are generated by different devices must be processed in pre-determined time. The third property can be variety and it shows that data contain different types like multimedia, text, string, stream etc. The data processing system must be able to manage these types of data. If data has all or some of above features, it is called “Big data”.

DOI: 10.4018/978-1-5225-7214-5.ch007

To solve big data problems, usually traditional algorithms cannot be used. Big data problems are usually solved on the distributed platforms. Distributed platforms have their own problems. One of the main problems is data locality problem. Data locality problem is not existence of the required data on the processor node. Data locality problem causes processor nodes use network to achieve the required data and using network causes following problems:

- Not proper use of node hardware because of node wait to receive data
- Network congestion
- Join received data from other nodes with node local data
- Save intermediate results for iterative problems.

One the most important methods that big data problems are solved by is MapReduce. MapReduce is a programming method that is executed on large hardware clusters (Dean et al., 2008). MapReduce also have above problems and so it is not appropriate for problems like data warehouse, graph and data mining.

In this chapter, some conditions are proposed that if it is possible to apply them on MapReduce problems they can be solved properly. These conditions are called Baran conditions. The proposed conditions are used for different types of problems and the results shows that the proposed conditions solve problems with lower execution time. The solved problems are in different fields like graph, data mining and data warehouse.

The structure of this chapter is as follows. In section 2, related works are discussed. In section 3, Baran conditions are illustrated. The proposed conditions are then evaluated in different fields in comparison with prevalent methods of each field. The final section is the conclusion.

2. RELATED WORKS

In this section related works bout MapReduce optimization, big data tools are investigated

2.1. MapReduce

The two main components of the MapReduce architecture are Mappers and Reducers. The data items in the MapReduce architecture are <Key, Value> pairs. All operations involve these pairs. Every algorithm can utilize this architecture. Each node of Mappers and Reducers takes part in solving a problem. Mappers execute an algorithm and generate intermediate results. These results may be aggregated or filtered according to keys. In the second phase, a Reducer generates the results by combining <Key,Value> pairs on their keys. Combiners are sometimes used to prevent network and process bottlenecks. Combiners act in a similar way to Reducers and improve their performance and speed.

Many attempts have been made to improve the MapReduce architecture. Figure 1 presents five approaches used in (Osman et al., 2013) to optimize the MapReduce architecture.

In Figure 1, the first step in MapReduce optimization tries to improve the job scheduling and the distribution of the tasks over the nodes. MapReduce++ (Zhang et al., 2012) estimates the execution time for each task. In each job, a task with a minimum execution time is then selected. This method improves the overall execution time. The same MapReduce++ method is used in (Polo et al., 2009), but the scheduling is performed by dynamic resource allocation using parameters specified by the user.