ON TRAFFIC-AWARE PARTITION AND AGGREGATION IN MAPREDUCE FOR BIG DATA APPLICATIONS

OBJECTIVE:

The objective of this system to reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme.

ABSTRACT:

The MapReduce programming model simplifies large-scale data processing on commodity cluster by exploiting parallel map tasks and reduce tasks. Although many efforts have been made to improve the performance of MapReduce jobs, they ignore the network traffic generated in the shuffle phase, which plays a critical role in performance enhancement. Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration. In this paper, we study to reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme. Furthermore, we jointly consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. Finally,

extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost under both offline and online cases.

INTRODUCTION:

MapReduce has emerged as the most popular computing framework for big data processing due to its simple programming model and automatic management of parallel execution. MapReduce and its open source implementation Hadoop have been adopted by leading companies, such as Yahoo!, Google and Facebook, for various big data applications, such as machine learning bioinformatics and cyber security MapReduce divides a computation into two main phases, namely map and reduce, which in turn are carried out by several map tasks and reduce tasks, respectively. In the map phase, map tasks are launched in parallel to convert the original input splits into intermediate data in a form of key/value pairs. These key/value pairs are stored on local machine and organized into multiple data partitions, one per reduce task. In the reduce phase, each reduce task fetches its own share of data partitions from all map tasks to generate the final result. There is a shuffle step between map and reduce phase. In this step, the data produced by the map phase are ordered, partitioned and transferred to the appropriate machines executing the reduce phase. The resulting network traffic pattern from all map tasks to all reduce tasks can cause a great volume of network traffic, imposing a serious constraint on the efficiency of data analytic applications.

EXISTING SYSTEM

Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration.