INCREMENTAL MAPREDUCE IN BIG DATA ENVIRONMENT

Kabila

PG Scholar, Department of Computer Science and Engineering, Institute of Road and Transport Technology, Erode – 638316.

Abstract - New data and updates are growing bigger and the results of data mining applications turn out to be stable. In this paper we use Incremental MapReduce and it is the extension of the map reduces. Map reduce have been used widely in mining. Incremental MapReduce process key-value pair level incremental processing. It process only key-value pair level incremental processing somewhat than task level re-computation and it also support sophisticated iterative techniques. In addition to this map reduce we have introduced the K-Nearest Neighbor. This model allows us to bring the output again a large dataset. To map gases will determine the K-n Neighbors in different splits of the data. Incremental Map performs the operation once it get the data from the K-N.

Keywords- Incremental MapReduce, key-value pair, k-Nearest Neighbour, re-computation.

1. INTRODUCTION (Size 11, cambria font)

Today huge amount of digital data is being accumulated in many important areas, including e-commerce, social network, finance, health care, education, and environment. It has become increasingly popular to mine such big data in order to gain insights to help business decisions or to provide better personalized, higher quality services. In recent years, a large number of computing frameworks have been developed for big data analysis. Among these frameworks, MapReduce (with its open-source implementations, such as Hadoop) is the most widely used in production because of its simplicity, generality, and maturity. We focus on improving Map Reduce in this paper. Big data is constantly evolving. As new data and updates are being collected, the input data of a big data mining algorithm will gradually change, and the computed results will become stale and obsolete over time. In many situations, it is desirable to periodically refresh the mining computation in order to keep the mining result sup-to-date.

2. Existing system

The PageRank algorithm computes ranking scores of web pages based on the web graph structure for supporting web search. However, the web graph structure is constantly evolving web pages and hyper-links are created, deleted, and updated. As the underlying web graph evolves, the PageRank ranking results gradually become stale, potentially lowering the quality of web search. Therefore, it is desirable to refresh the PageRank computation regularly. Incremental processing is a promising approach to refreshing mining results. Given the size of the input big data, it is often very expensive to rerun the entire computation from scratch. Incremental processing exploits the fact that the input data of two subsequent computations A and B are similar. Only a very small fraction of the input data has changed. The idea is to save states in computation A, re-use A’s states in computation B, and perform re-computation only for states that are affected by the changed input data. A number of previous studies (including Percolator, CBP) have followed this principle and designed new programming models to support incremental processing. Unfortunately, the new programming models (BigTable observers in Percolator, stateful translate operators in CBP) are drastically different from MapReduce, requiring programmers to completely reimplement their algorithms. Incoop extends MapReduce to support incremental processing. It has two main limitations. First, Incoop supports only task-level incremental processing. That is, it saves and reuses states at the granularity of individual Map and Reduce tasks. Each task typically processes a large number of key-value pairs (kv pairs). If Incoop detects any data changes in the input of a task, it will rerun the entire task. While this approach easily leverages existing MapReduce features for state savings, it may incur a large amount of redundant computation if only a small fraction of kv-pairs have changed in a task. Second, Incoop supports only one-step computation, while important mining algorithms, such as PageRank, require iterative computation. Incoop would treat each iteration as a separate
MapReduce job. However, a small number of input data changes may gradually propagate to affect a large portion of intermediate states after a number of iterations, resulting in expensive global re-computation after wards. The Bulk Synchronous Processing (BSP) model. The computation is broken down into a sequence of super steps. In each super step, a Compute function is invoked on each vertex. It communicates with other vertices by sending and receiving messages and performs computation for the current vertex. This model can efficiently support a large number of iterative graph algorithms. It provides a group wise processing operator Translate that takes state as an explicit input to support incremental processing. But it adopts a new programming model that is very different from Map Reduce. In addition, several research studies support incremental processing by task-level re-computation, but they require users to manipulate the states on their own. In contrast, Incremental MapReduce exploits a fine-grain kv-pair level re-computation that are more advantageous. Incremental processing for iterative application, Proposes a timely data flow paradigm that allows stateful computation and arbitrary nested iterations. To support incremental iterative computation, programmers have to completely rewrite their MapReduce programs. In comparison, extend the widely used MapReduce model for incremental iterative computation. Existing MapReduce programs can be slightly changed to run on incremental MapReduce for incremental processing.

4. Proposed System.

Propose Incremental MapReduce, an extension to MapReduce that supports fine-grain incremental processing for both one step and iterative computation. Compared to previous solutions, Incremental MapReduce incorporates the following three novel features

4.1 Fine-grain incremental processing using MRBG-store

Inc mentsal Map Reduce supports kv-pair level fine-grain incremental processing in order to minimize the amount of re-computation as much as possible. Model the kv-pair level data flow and data dependence in a Map Reduce computation as a bipartite graph, called MRB Graph. A MRB-Store is designed to preserve the fine-grain states in the MRB Graph and support efficient queries to retrieve fine-grain states for incremental processing.

4.2 General-Purpose Iterative Computation

With modest extension to MapReduce API, previous work proposed Incremental MapReduce to efficiently support iterative computation on the MapReduce platform. However, it targets types of iterative computation where there is a one-to-one/all-to-one correspondence from Reduce output to Map input. In comparison, our current proposal provides general-purpose support, including not only one-to-one, but also one-to-many, many-to-one, and many-to-many correspondence. Enhance the Map API to allow users to easily express loop-invariant structure data, and propose a Project API function to express the correspondence from Reduce to Map. While users need to slightly modify their algorithms in order to take full advantage of Incremental MapReduce, such modifications are modest compared to the effort to re-implement algorithms on a completely different programming paradigm.

4.3 Incremental Processing For Computation.

Incremental iterative processing is substantially more challenging than incremental one-step processing because even a small number of updates may propagate to affect a large portion of intermediate states after a number of iterations. To address this problem, propose to reuse the converged state from the previous computation and employ a change propagation control (CPC) mechanism. We also enhance the MRBG-Store to better support the access patterns in incremental iterative processing. To our knowledge, IncrementalMapReduce is the first MapReduce-based solution that efficiently supports incremental iterative computation.

5 System Architecture Diagram.

Fig -1: System Architecture Diagram.

6. Module Description

6.1 Collect data blocks Module

Huge amount of digital data is being accumulated in e-commerce, social network, finance, health care, education environment. It has become increasingly popular to mine such big data in order to gain insights to help business decisions or to provide better personalized, higher quality
services. In recent years, a large number of computing frameworks have been developed for big data analysis. As new data and updates are being collected, the input data of a big data mining algorithm will gradually change, and the computed results will become stale and obsolete over time. Given the size of the input big data, it is often very expensive to rerun the entire computation from scratch. Divides a file into equal-sized blocks and stores the blocks across a cluster of machines. The MapReduce system runs a Job Tracker process on a master node to monitor the job progress, and a set of Task Tracker processes on worker nodes to perform the actual Map and Reduce tasks.

6.2 Iterative computation Module

Iterative computation is substantially more challenging than incremental one-step processing because even a small number of updates may propagate to affect a large portion of intermediate states after a number of iterations.

To reuse the converged state from the previous computation and employ a change propagation control (CPC) mechanism. Also enhance the MRB-Store to better support the access patterns in incremental iterative processing. Incremental MapReduce is the first MapReduce-based solution that efficiently supports incremental iterative computation. Including not only one-to-one, but also one-to-many, many-to-one, and many-to-many correspondence. Enhance the Map API to allow users to easily express loop-invariant structure data, and API function to express the correspondence from Reduce to Map. While users need to slightly modify their algorithms in order to take full advantage of Incremental MapReduce, such modification is modest compared to the effort to re-implement algorithms on a completely different programming paradigm.

6.2 Fine-Grain Incremental Processing Module

Incremental MapReduce supports kv-pair level fine-grain incremental processing in order to minimize the amount of re-computation as much as possible. The kv-pair level data flow and data dependence in a MapReduce computation as a bipartite graph. A MRB-Store is designed to preserve the fine-grain states in the MRBGraph and support efficient queries to retrieve fine-grain states for incremental processing. The Map function takes a kv-pair (K1; V 1) as input and computes zero or more intermediate kv-pairs (K2; V 2). Then all (K2; V 2) is grouped by K2. The Reduce function takes a K2 and a list of V 2 as input and computes the final output kv-pairs (K3; 3). The fine-grain incremental processing engine with an example application, which computes the sum of in-edge weights for each vertex in a graph. The Map input is the adjacency matrix of the graph. Every record corresponds to a vertex in the graph. K1

7 Implementation

7.1 Performance and result for mapReduce
Fig 2: Performance of MapReduce

Fig 3: Result of map and Reduce
Fig 4: Analytic Report for MapReduce

Fig 5: Analytic Report for MapReduce
8. CONCLUSIONS

In this work, we focus on the problem of stale data when data are frequently arriving. To avoid this, we use incremental MapReduce along with k-nearest neighbour. MapReduce combines a fine-grain incremental engine, a general-purpose iterative model, and a set of effective techniques for incremental iterative computation. Real-machine experiments show that MapReduce can significantly reduce the run time for refreshing big data mining results compared to re-computation on both plain and iterative MapReduce.

REFERENCES


