

A Survey on Large Scale Dataset Processing Into Batches Using Map Reduce Query Language (MRQL) Technique

Pragati P. Pachghare¹, Prof. Pravin G. Kulurkar²

M.Tech CSE, Vidarbha Institute of Engineering, Nagpur¹

H.O.D, CSE, Vidarbha Institute of Engineering, Nagpur²

Abstract: Online query processing for large-scale, incremental data analysis on a distributed stream processing engine (DSPE). Our goal is to convert any SQL-like query to an incremental DSPE program automatically. In contrast to other approaches, we derive incremental programs that return accurate results, not approximate answers, by retaining a minimal state during the query evaluation lifetime and by using a novel incremental evaluation technique, which, at each time interval, returns an accurate snapshot answer that depends on the current state and the latest batches of data. Our methods can handle many forms of queries on nested data collections, including iterative and nested queries, group-by with aggregation, and equi-joins. Finally, we report on a prototype implementation of our framework, called MRQL Streaming, running on top of Spark and we experimentally validate the effectiveness of our methods.

Keywords: Outlier detection, Stream data mining, local outlier, Memory efficiency.

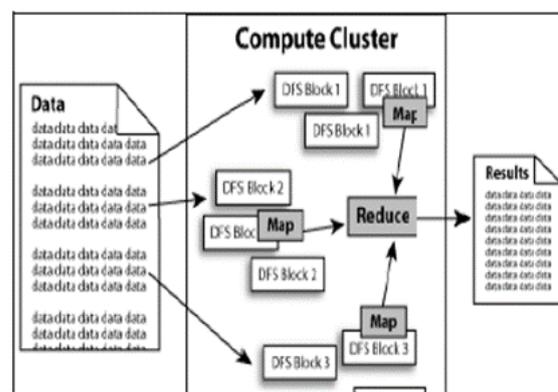
1. INTRODUCTION

We are living in an age when an explosive amount of data is being generated every day. Data from sensors, mobile devices, social networking websites, scientific data & enterprises – all are contributing to this huge explosion in data. This sudden bombardment can be grasped by the fact that we have created a vast volume of data in the last two years. Big Data- as these large chunks of data is generally called- has become one of the hottest research trends today.

Research suggests that tapping the potential of this data can benefit businesses, scientific disciplines and the public sector – contributing to their economic gains as well as development in every sphere. The need is to develop efficient systems that can exploit this potential to the maximum, keeping in mind the current challenges has been a shift in the architecture of data-processing systems today, from the centralized architecture to the distributed associated with its analysis, structure, scale, timeliness and privacy. There architecture. Enterprises face the challenge of processing these huge chunks of data, and have found that none of the existing centralized architectures can efficiently handle this huge volume of data. These are thus utilizing distributed architectures to harness this data. Several solutions to the Big Data problem have emerged which includes the Map Reduce environment championed by Google which is now available open-source in Hadoop. Hadoop's distributed processing; Map Reduce algorithms and overall architecture are a major step towards achieving the promised benefits of Big Data.

Map Reduce & Hadoop are the most widely used models used today for Big Data processing. Hadoop is an open source large-scale data processing framework that

supports distributed processing of large chunks of data using simple programming models. The Apache Hadoop project consists of the HDFS and Hadoop Map Reduce in addition to other modules. The software is modelled to harvest upon the processing power of clustered computing while managing failures at node level. The Map Reduce software framework which was originally introduced by Google in 2004 is a programming model, which now adopted by Apache Hadoop, consists of splitting the large chunks of data, and „Map“ & „Reduce“ phases (Fig. 1). The Map Reduce framework handles task scheduling, monitoring and failures.



2. LITERATURE SURVEY

Incremental data processing can generally achieve better performance and may require less memory than batch processing for many data analysis tasks. It can also be

used for analyzing Big Data incrementally, in batches that can fit in memory. Consequently, incremental data processing can also be useful to stream-based applications that need to process continuous streams of data in real-time with low latency, which is not feasible with existing batch analysis tools. For example, the Map-Reduce framework [9], which was designed for batch processing, is ill-suited for certain Big Data workloads, such as real-time analytics, continuous queries, and iterative algorithms. New alternative frameworks have emerged that address the inherent limitations of the Map-Reduce model and perform better for a wider spectrum of workloads.

Currently, among them, the most promising frameworks that seem to be good alternatives to Map-Reduce while addressing its drawbacks are Google's Pregel, Apache Spark [6], and Apache Flink [13], which are in-memory distributed computing systems.

There are also quite a few emerging distributed stream processing engines (DSPEs) that realize online, low-latency data processing with a series of batch computations at small time intervals, using a continuous streaming system that processes data as they arrive and emits continuous results. To cope with blocking operations and unbounded memory requirements, some of these systems build on the well-established research on data streaming based on sliding windows and incremental operators [18], which includes systems such as Aurora [17] and Telegraph [19], often yielding approximate answers, rather than accurate results. Currently, among these DSPEs, the most popular platforms are Twitter's (now Apache) Storm [3], Spark's D-Streams [16], Flink Streaming [13], Apache S4 [16], and Apache Samza [16]. The process of the research into complex data basically concerned with the revealing of hidden patterns.

Sagiroglu, S.; Sinanc, D. (20-24 May 2013), "Big Data: A Review" describe the big data content, its scope, methods, samples, advantages and challenges of Data. The critical issue about the Big data is the privacy and security. Big data samples describe the review about the atmosphere, biological science and research. Life sciences etc. By this paper, we can conclude that any organization in any industry having big data can take the benefit from its careful analysis for the problem solving purpose. Using Knowledge Discovery from the Big data easy to get the information from the complicated data sets [1].

The overall Evaluation describe that the data is increasing and becoming complex. The challenge is not only to collect and manage the data also how to extract the useful information from that collected data.

According to the Intel IT Center, there are many challenges related to Big Data which are data growth, data infrastructure, data variety, data visualization, data velocity.

Garlasu, D.; Sandulescu, V.; Halcu, I.; Neculoiu, G. (17-19 Jan. 2013), "A Big Data implementation based on Grid Computing", Grid Computing offered the advantage about the storage capabilities and the processing power and the Hadoop technology is used for the implementation

purpose. Grid Computing provides the concept of distributed computing. The benefit of Grid computing center is the high storage capability and the high processing power. Grid Computing makes the big contributions among the scientific research, help the scientists to analyze and store the large and complex data [4].

3. PROPOSED WORK

Data is conceptually record-oriented in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records. The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system.

Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of moving computation to the data, instead of moving the data to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

PKMeans Based on MapReduce

As the analysis above, PKMeans algorithm needs one kind of MapReduce job. The map function performs the procedure of assigning each sample to the closest center while the reduce function performs the procedure of updating the new centers. In order to decrease the cost of network communication, a combiner function is developed to deal with partial combination of the intermediate values with the same key within the same map task.

Map-function The input dataset is stored on HDFS[11] as a sequence file of <key, value> pairs, each of which represents a record in the dataset. The key is the offset in bytes of this record to the start point of the data file, and the value is a string of the content of this record. The dataset is split and globally broadcast to all mappers. Consequently, the distance computations are parallel executed. For each map task, PKMeans construct a global variant centers which is an array containing the information about centers of the clusters. Given the information, a mapper can compute the closest center point for each sample. The intermediate values are then composed of two parts: the index of the closest center point and the sample information. The pseudo code of map function is shown in Algorithm.

MapReduce Programming Model MapReduce is a software framework proposed by Google, which is a basis computational model of current cloud computing platform. Its main function is to handle massive amounts of data. Because of its simplicity, MapReduce can effectively deal with machine failures and easily expand the number of

system nodes. MapReduce provides a distributed approach to process massive data distributed on a large -scale computer clusters. The input data is stored in the distributed file system (HDFS), MapReduce adopts a divide and conquer method to evenly divided the inputted large data sets into small data sets, and then processed on different node, which has achieved parallelism.

In the MapReduce programming model, data is seen as a series of keyvalue pairs like , as shown in Figure 1, the workflow of MapReduce consists of three phases: Map, Shuffle, and Reduce. Users simply write map and reduce functions. In the Map phase, a map task corresponds to a node in the cluster, as the other word, multiple map tasks are be running in parallel at the same time in a cluster. Each map call is given a key-value pair (k1,v1) and produces a list of (k2,v2) pairs. The output of the map calls is transferred to the reduce nodes (shuffle phase).

All the intermediate records with the same intermediate key (k2) are sent to the same reducer node. At each reduce node, the received intermediate records are sorted and grouped (all the intermediate records with the same key form a single group). Each group is processed in a single reduce call. The data processing [4-6] can be summarized as follows: Map (k1, v1) \rightarrow list(k2, v2)
Reduce (k2, list(v2)) \rightarrow list(k3, v3)

4. CONCLUSION

We propose general, sound methods to transform batch queries to incremental queries. The first step in our approach is to transform a query so that it propagates the join and group-by keys to the query output. This technique is known as lineage tracking .That way, the values in the query output are grouped by a key combination, which corresponds the join and group-by keys used in deriving these values during query evaluation.

If we also group the new data in the same way, then computations on current data can be combined with the computations on the new data by joining the data on these keys. This approach requires that we can combine computations on data that have the same lineage to derive incremental results. In our framework, this task is accomplished by transforming a query to a 'monoid homomorphism' by extracting the non-homomorphic parts of the query outwards, using algebraic transformation rules, and combining them to form an answer function, which is detached from the rest of the query.

We present a general automated method to convert most distributed data-analysis queries to incremental stream processing programs.

- Our methods can handle many forms of queries, including iterative and nested queries, group-by with aggregation, and joins on one-to-many relationships.
- We report on a prototype implementation of our framework using Apache MRQL running on top of Apache Spark Streaming. We show the effectiveness of our method through experiments on four queries:

groupBy, join-groupBy, k-means clustering, and PageRank.

REFERENCES

- [1] O. Boykin, S. Ritchie, I. O'Connell, and J. Lin Summingbird: A Framework for Integrating Batch and Online MapReduce Computations. In International Conference on Very Large Data Bases (VLDB), pages 1441–1451, 2014
- [2] B. Chandramouli, J. Goldstein, M. Barnett, R. DeLine, D. Fisher, J. C. Platt, J. F. Terwilliger, J. Wernsing, Trill: A High-Performance Incremental Query Processor for Diverse Analytics. In International Conference on Very Large Data Bases (VLDB), pages 401–412, 2014.
- [3] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi. Naiad: a Timely Dataflow System. In ACM Symposium on Operating Systems Principles (SOSP), 2013.
- [4] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin. Incoop: Mapreduce for Incremental Computations. In ACM Symposium on Cloud Computing (SoCC), 2011.
- [5] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. Mapreduce Online. In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 10(4), 2010.
- [6] D. Peng and F. Dabek. Large-scale Incremental Processing using Distributed Transactions and Notifications. In Symposium on Operating System Design and Implementation (OSDI), 2010.
- [7] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: a not-so-Foreign Language for Data Processing. In ACM SIGMOD International Conference on Management of Data, pages 1099-1110, 2008
- [8] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with Uncertainty and Lineage. In International Conference on Very Large Data Bases (VLDB), pages 953–964, 2006.
- [9] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Symposium on Operating System Design and Implementation (OSDI), 2004.
- [10] D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An Annotation Management System for Relational Databases. In International Conference on Very Large Data Bases (VLDB), pages 900–911, 2004.
- [11] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A Recursive Model for Graph Mining. In Fourth SIAM International Conference on Data Mining (SDM), pages 442–446, 2004.
- [12] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: Continuous Data flow Processing for an UncertainWorld. In Conference on Innovative Data System Research (CIDR), 2003.
- [13] Apache Hadoop. <http://hadoop.apache.org/>.
- [14] Apache S4 (incubating): A Distributed Stream Computing Platform. <http://incubator.apache.org/s4/>.
- [15] Apache S4 (incubating): A Distributed Stream Computing Platform. <http://incubator.apache.org/s4/>.
- [16] Apache S4 (incubating): A Distributed Stream Computing Platform. <http://incubator.apache.org/s4/>.
- [17] D. J. Abadi, D. Carney, U. Cetintemel, et al. Aurora: A New Model and Architecture for Data Stream Management. In VLDB Journal, 12(2):120–139, 2003.
- [18] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: Continuous Data flow Processing for an UncertainWorld. In Conference on Innovative Data System Research (CIDR), 2003.
- [19] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. In Symposium on Principles of Database Systems (PODS), pages 1–16, 2002.