IORD Journal of Science & Technology
E-ISSN: 2348-0831
Volume 3, Issue 3 (MAR-APRIL 2017) PP30-36
IMPACT FACTOR 1.719
www.iord.in

Large Data Turns Into Batches Using MRQL (Map Reduce Query Language) Technique

Pragati P. Pachghare[1] , Prof. Pravin G. Kulurkar[2]

[1]M.Tech CSE,Vidarbha Institute of Engineering, Nagpur
pragatipachghare@gmail.com

[2]H.O.D,CSE, Vidarbha Institute of Engineering, Nagpur
pravinkulurkar@gmail.com

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 associated with its analysis, structure, scale,

timeliness and privacy. There has been a shift in the architecture of data-processing systems today, from the centralized architecture to the distributed 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 BigData problem have emerged which includes the Map Reduce environment championed by Google which is nowavailable 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.

IORD Journal of Science & Technology
E-ISSN: 2348-0831
Volume 3, Issue 3 (MAR-APRIL 2017) PP30-36
IMPACT FACTOR 1.719
www.iord.in

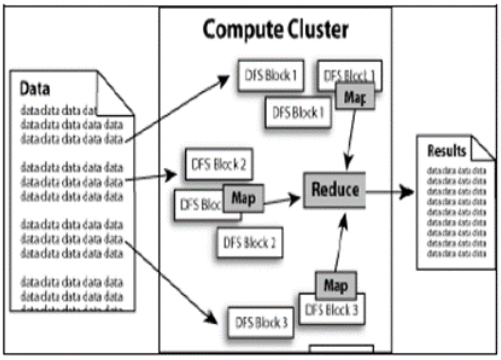


FIG. 1 MAP REDUCE IN HADOOP

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 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 , and Apache Flink , 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 , which includes systems such as Aurora and Telegraph . Currently, among these DSPEs, the most popular platforms are Twitter's (now Apache) Storm , Spark's D-Streams Flink Streaming , Apache S4 , and Apache Samza . The process of the research into complex data basically concerned with the revealing of hidden patterns.

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. Lifesciences 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.

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.A Big Data implementation based on Grid Computing", Grid Computing offered the advantage about thestorage 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 .

Big data analytics define the analysis of large amount of data to get the useful information and uncover the hidden patterns. Big data analytics refers to the Mapreduce Framework which is developed by the Google. Apache Hadoop is the open source platform which is used for the purpose of implementation of Google's Mapreduce Model .

3.PROPOSED METHODOLGY

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 thedata to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

IORD Journal of Science & Technology
E-ISSN: 2348-0831
Volume 3, Issue 3 (MAR-APRIL 2017) PP30-36
IMPACT FACTOR 1.719
www.iord.in

K Means Clustering – Pseudo code

K-Means is a simple learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of n entities in k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized

The k-means algorithm is used for partitioning where each cluster's centre is represented by the mean value of the objects in the cluster

K Means Clustering – Pseudo code

Pseudo code

1.Begin with n clusters, each containing one object and we will number the clusters 1 through n.

2.Compute the between-cluster distance D(r, s) asthe between-object distance of the two objects in r and s respectively, r, s =1, 2, …, n. Let the square matrix D = (D(r, s)). If the objects are represented by vectors, we can use the Euclidean distance.

3.Next, find the most similar pair of clusters r and s,Such that the distance, D(r, s), is minimum among all the pairwise distances.

4.Merge r and s to a new cluster t and compute the between-cluster distance D(t, k) for any existingcluster $k \neq r, s$. Once the distances are obtained, delete the rows and columns corresponding to the old cluster r and s in the D matrix, since r and s do not exist anymore. Then add a new row and column in D corresponding to cluster t.

5.Repeat Step 3 a total of n − 1 times until there is only one cluster left.

ParallelK-Means Algorithm Based on MapReduce

In this section we present the main design for Parallel K-Means (PKMeans) based on MapReduce. Firstly, we give a brief overview of the k-means algorithm and analyze the parallel parts and serial parts in the algorithms. Then we explain how the necessary computations can be formalized as map and reduce operations in detail.

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-functionThe input dataset is stored on HDFS as a sequence file of<key, value>pairs, each of which represents arecord 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 pseudocode 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) $\longrightarrow$ list(k2, v2)

Reduce (k2, list(v2)) $\longrightarrow$ list(k3, v3)

4. IMPLEMENTATION DETAILS

Distributed data intensive computing To store, manage, access, and process vast amount of data represents a fundamental requirement and an immense challenge in order to satisfy needs to search, analyze, mine, and visualize the data and information. Data intensive computing is intended to address this need.

Google File System The Google File System (GFS) is a proprietary Distributed File System developed by Google. It is

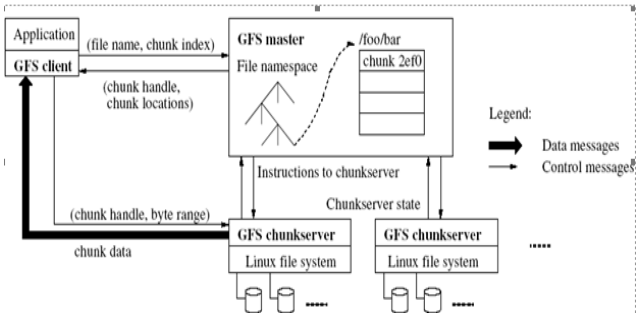
IORD Journal of Science & Technology
E-ISSN: 2348-0831
Volume 3, Issue 3 (MAR-APRIL 2017) PP30-36
IMPACT FACTOR 1.719
www.iord.in

designed (Figure 1) to provide efficient, reliable access to data using large clusters of commodity hardware.

The files are huge and divided into chunks of 64 megabytes . Most files are mutated by appending new data rather than overwriting existing data: once written, the files are only read and often only sequentially. This DFS is best suited for scenarios in which many large files are created once but read many times. The GFS is optimized to run on computing clusters where the nodes are cheap computers.

Hence, there is a need for precautions against the high failure rate of individual nodes and data loss. In the Google file system there can be 100 to 1000 PCs in a cluster can be used.
1) Chunkserver Architecture Server
• Stores 64 MB file chunks on local disk using standard Linux filesystem, each with version number and checksum
• Read/write requests specify chunk handle and byte range
• Chunks replicated on configurable number of chunk servers (default: 3)
• No caching of file data

2) Client
• Issues control (metadata) requests to master server
• Issues data requests directly to chunkservers
• Caches metadata
• Does no caching of data No consistency hence difficulties among clients Streaming reads (read once) and append writes (write once) don't benefit much from caching at Client



Tools/Technology is to be used
Big Data has emerged because we are living in a society which makes increasing use of data intensive technologies.
One current feature of big data is the difficulty working with it using relational databases and desktop statistics/visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". The various challenges faced in large data management include – scalability, unstructured data, accessibility, real time analytics, fault tolerance and many more. In addition to variations in the amount of data stored in different sectors, the types of data generated and stored—i.e., whether the data encodes video, images, audio, or text/numeric information—also differ markedly from industry to industry.
Big data requires exceptional technologies to efficiently process large quantities of data within tolerable elapsed times. Technologies being applied to big data include massively parallel processing (MPP) databases, data mining grids, distributed file systems, distributed databases, cloud computing platforms, the Internet, and scalable storage systems. Real or near-real time information delivery is one of the defining characteristics of Big Data Analytics. Latency is therefore avoided whenever and wherever possible. A wide variety of techniques and technologies has been developed and adapted to aggregate, manipulate, analyze, and visualize big data. These techniques and technologies draw from several fields including statistics, computer science, applied mathematics, and economics. This means that an organization that intends to derive value from big data has to adopt a flexible, multidisciplinary approach.

5.Experimental Results

MRQL Algorithm based on MapReduce algorithm extending class is by verifying whether the given object in the dataset D is a core at the specified radius ε. The Algorithm takes a large part of the time spending on the region query of object. When the dataset is very large, the inputted data objects are many, serial MRQL algorithm to determine whether each of object is core object will consume a high I/O overhead. Researchers found that the determination of core object can be parallelized, and we can get a conclusion by analyzing the algorithm that if an object exists in two different classes, and it is a core object, then these two classes can be merged into a new class. Otherwise the object belongs to one of classes, there is no relationship between the two classes.
This paper introduces a new concept - sharing object. sharing object Core objects P and Q belong to different classes, if the object O starting from P, Q are directly density-reachable, then called object O is a sharing object. And if O is a core object, called O is a sharing core object. If there exist a sharing core object in different classes, these classes can be combined into a new class. as shown in Figure 2, object O is a sharing core object: Figure 2. Clustering of Overlapping Objects in Region Obviously we can take advantage of the MapReduce programming model to parallelize the whole process to save clustering time and resources. The basic idea of MRQL algorithm based on MapReduce is divided into four steps:
Step one: The data in the dataset cut into small blocks which are equal size.

IORD Journal of Science & Technology
E-ISSN: 2348-0831
Volume 3, Issue 3 (MAR-APRIL 2017) PP30-36
IMPACT FACTOR 1.719
www.iord.in

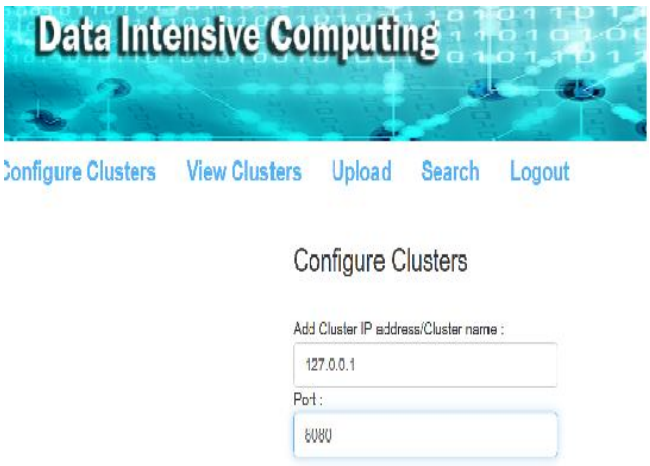


Fig: Configure Clusters

Fig: Available Clusters

Step two: The blocks are distributed to the nodes in the cluster, so that all of nodes in the cluster can run the Map function of Themselves in parallel to calculate and process those blocks.

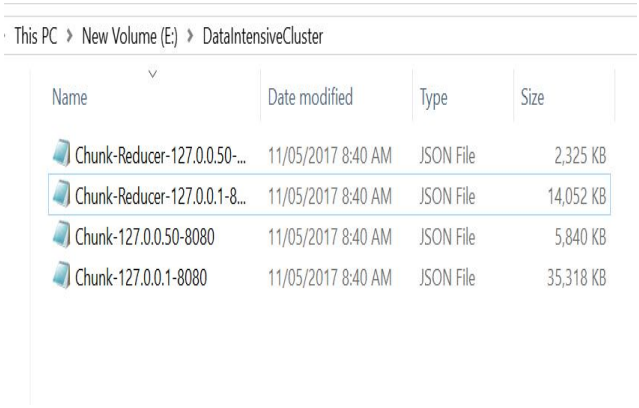


Fig: Divided chunk

(1) Initial Cluster. For any unlabeled object p in the dataset, using Map-Reduce parallel programming model to calculate the number of objects in its region to determine if it is a core object. If is, P and all the objects in the region of P constitute an initial class (C), and marked those objects with the same cluster identifier (Cid). Conversely, if p is not a core object, and there is no other object in its region, marked P as noise. Otherwise, detect whether there has a core object q in the region of non-core object p. If has, given the object p and the objects in the region of q with the same cluster identifier. Repeat until all of the objects in the dataset are identified. After the process is completed, get an initial class clusters and a noise set.
(2) Merge Result. Class merging is to consider these objects which exist in more than two classes. if there is a sharing core object, merging the two classes. Else classify the object as the proximity side. It will be given a new class name if there have different classes are combined. When there is no object exist in different classes, merging initialize class completed.

3. Step three: merge the result of each processor.

4. Step four: Output clustering results shows the ideas of MRQL DBSCAN algorithm based on MapReduce, the data exchange format of each stage and the object identifier Oid as the key, value is filled with (core_tag, usedtag, Cid, x, y, z). core_tag indicates whether it is a core object, used_tag identify whether there had been clustering, Cid means class logo.

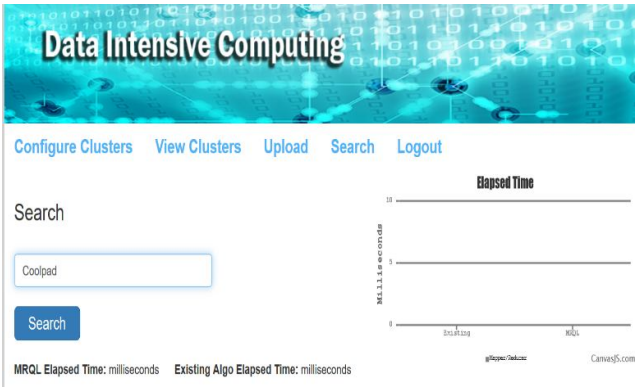


Fig: Search Data

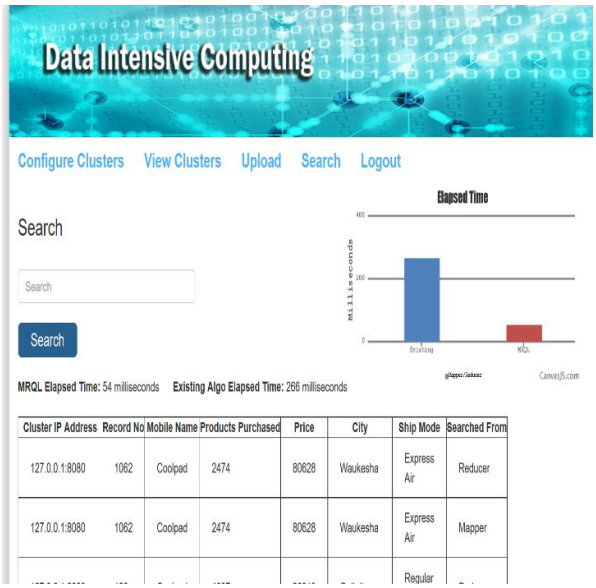


Fig: Final output

6.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.

7.REFERENCES

[1] 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.

[2] 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.

[3] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with Uncertainty and Lineage. In International Confer- ence on Very Large Data Bases (VLDB), pages 953–964, 2006.

[4] 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.

[5] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin. Incoop: Mapreduce for Incremental Computations. In ACM Sym-posium on Cloud Computing (SoCC), 2011.

[6] 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.

[7] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A Recursive Model for Graph Mining. In Fourth SIAM International Conferenceon Data Mining (SDM), pages 442–446, 2004.

[8] 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.

[9] 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.

[10] 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.

[11] Y. Cui and J.Widom. Lineage Tracing for General DataWarehouse Transformations. In International Conference on Very Large Data Bases (VLDB), pages 471–480, 2001.

[12] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Symposium on Operating System Design and Implementation (OSDI), 2004.

[13] P. Desikan, N. Pathak, J. Srivastava, and V. Kumar Incremental PageRank Computation on Evolving Graphs. In International conference on World Wide Web (WWW), pages 1094–1095, 2005.

[14] L. Fegaras. Incremental Stream Processing of Nested-Relational Queries. In International Conference on Database and Expert Systems Applications (DEXA), September 2016. Available at http://lambda.uta.edu/incr-dexa16.pdf.

[15] L. Fegaras, C. Li, U. Gupta, and J. J. Philip. XML Query Optimization in Map-Reduce. In International Workshop on the Web and Databases (WebDB), 2011.

[16] L. Fegaras, C. Li, and U. Gupta. An Optimization Framework for Map-Reduce Queries. In International Conference on Extending Database Technology (EDBT), pages 26–37, 2012.