A Novel Time Interval based Algorithm for Data Fetching on Bigdata

M. Banupriya
PG Scholar
Department of CSE/IT
University College of Engineering Trichy
(BIT Campus)Trichy.

Mrs. K. Uma Maheswari
Assistant Professor
Department of CSE/IT
University College of Engineering Trichy
(BIT Campus)Trichy.

Abstract-Nowadays data mining is most important field for worldwide environmental application and it is having many subfields mainly focusing the temporal or sequential pattern mining. The two interval between relationship is more complex and discovering interval based sequence is a challenging issue. The number of techniques have been developed in recent years for processing sensor data on cloud or big data. Sensor cloud does not provide efficient support on fast detection and locating of errors in big sensor data sets. A novel data error detection approach is developed which exploits the full computation potential of cloud platform. The error detection is based on the scale-free network topology and most of detection operations can be conducted in limited temporal or sequential data blocks instead of a whole big data set. Hence the detection and location tasks can be distributed to cloud platform to fully exploit the computation power and massive storage. The proposed approach can significantly reduce the time for error detection and location in big data set generated by large scale network system with acceptable error detecting accuracy.

Keywords- Data mining; bigdata; cloud computing environment; temporal pattern; classification; Map Reduce concept.

I. INTRODUCTION

In data mining, big data is a research field on both academia and industry. To analyze massive amounts of data and obtain valuable information and knowledge researchers have developed many excellent systems and technologies.
The GFS[1] and Map Reduce[2] developed by Google could process 20PB of web pages per day in 2007.The HDFS[3] and HBase[4] clusters developed by Face book[5] scanned 300 million images daily in 2012, amounting to more than 500TB of data. The search engine system developed by Baidu[6] could handle 100 PB of data per day in 2013.

The development and popularization of e-commerce and social network[7,8] data have been showing increasingly high relevance and coupling degree result in the rapid growth of the scale of structured data to PB level and above.
Now the requirement of Big data pose new challenges for both relational database and big data processing technology.

In 1970, Codd[9] first proposed a new model of the relationship which started the research on the relational method and theory of database. The structured data, relational database is undoubtedly the most classic and popular database system such as Oracle, MYSQL, SQLServer, and DB2.The development of relational database has become widely used in various types of information management systems and business application systems[10] and has become an effective storage and analysis tool for data warehouse[11].The changes with in information technology and architecture of relational database has been improved continuously from centralized database[12], distributed database[13] to parallel database[14] and the storage capacity has been increased from GB level to TB level.
The parallel database based on Massively Parallel Processing (MPP)[15] architecture can manage 100 TB of data. This database system consists of many loosely coupled processing units and each unit has it own private computing and storage resources. The most significant features of MPP database are shared nothing and multiple copies of data.

MapReduce is a programming framework proposed by Google and a typical technology for processing bigdata[2,16] with its super large-scale node scheduling ability and high throughput, MapReduce performs excellently in processing the massive unstructured data[17].Every computing request starts a job in the MapReduce framework.

In order to complete the job, the MapReduce framework needs to perform two kinds of tasks-: map and reduce. First, it splits the input dataset into independent blocks and distributes it to different node. The job manager initializes several map tasks and each map tasks processes one data block and generates an intermediate file after calculation.

Further the MapReduce framework sorts the output file of map tasks and several reduce tasks are initialized that aggregate the sorting results into the final output file. The framework is responsible for scheduling and monitoring the tasks and restarting failed tasks. The MapReduce framework fails to provide sufficient support for interactive query on structured datasets. Furthermore the initiation of map and reduce tasks consumes certain system resources and time. The complex commands involved must be decomposed into multiple sub-operations and communication among the sub-operations is available only through intermediate files[19].The study of structured bigdata is a crossing between the fields of bigdata and relational database[20].By combining HDFS with the splitting and scheduling model, Banian effectively integrates large-scale storage management with interactive query and analysis.

II. RELATED WORKS

The processing of structured bigdata warrants effective integration of massive storage with fast query and analysis. One line of research is incorporating MapReduce on the basis of MPP database such as Greenplum[21] and Teradata[22].These systems can be used to deal with SQL commands and MapReduce tasks simultaneously. Another major research approach is SQL on Hadoop, which provides an SQL interface on the HDFS and MapReduce foundation along with application oriented storage and query optimization[23,25]. Hive is the most typical example of SQL and Hadoop. It is used to map files onto a database table and provide an SQL query interface. Hive can run the data analysis logic reflected by SQL statements on HDFS by converting SQL statements into a series of MapReduce tasks.

Spark originated from the cluster computing platform at AMPlab, UC Berkeley. It supports users in performing in-memory computations on large- scale cluster using Resilient Distributed Datasets(RDD). RDDs are capable of dealing with a variety of computing paradigms involving multi- iterative batch processing, data warehouse, flow processing and graph computing.

Hadoop

Hadoop is widely used in big data applications in the industry, for example spam filtering, network searching, social recommendation. In addition, the considerable of academic research is now based on Hadoop. Now the biggest Hadoop cluster has 4000 nodes, but the number of nodes will be increased to 10,000 with the release of Hadoop 2.0.At the same period the facebook announced that their Hadoop cluster can process 100 PB data.

Nowadays many companies provide Hadoop commercial execution and/or support, including Cloudera, IBM,EMC, and Oracle. Bigdata is a trending topic in 2014.Hadoop is an open framework mostly used for big data analytics. Mapreduce is a programming paradigm associated with the Hadoop.

Characteristics of Hadoop

i. Hadoop is a open source system.
ii. It used to handle very large data sets.
iii. It is designed to scale to very large clusters.
iv. It automatically fragments storage over the cluster.

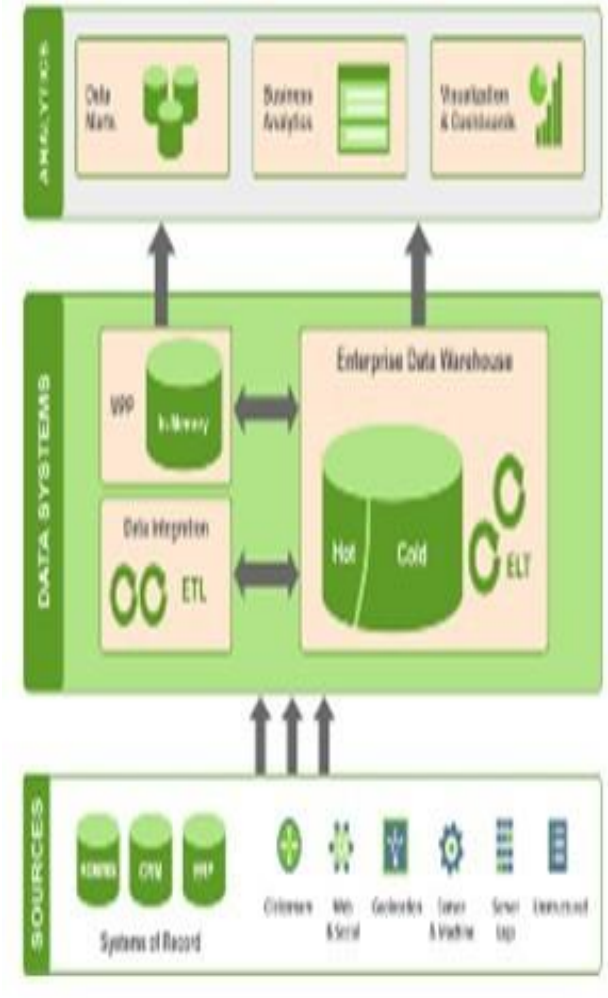

Figure 1. Cloud Environment

Concept of map reduce

Hadoop using HDFS for data storing and map reduce allows for distributed processing of the map or reduction function.

Issues

i. Locality.
ii. Synchronization.
iii. Fairness.

i) Input reader

The input reader divides the given input into appropriate size to splits and the framework assign one split to each map function.

The input reader reads data from standard storage and generates key or value pairs.

ii) Map function

The map function is used to process a key or value pair to generate another key or value pair. The number of such key or value pair is to map functions running in parallel on the data that is partitioned, across the cluster to produce a set of intermediate key or value pairs.

$$Map\ (k, v) \rightarrow <k', v'>¿$$

iii) Compare function

The input for each reduces is pulled from the machine where the map run and store using the application comparison function.

$$Compute\ (k', v') \rightarrow <k', v'>¿$$

iv) Partition function

The partition function is given the key and the number of reduced and returns the indexes of desired reduce. Here, it is important to pick a partition function that gives an approximately different or uniform distribution of data reducers assign more , than their share of data for load balancing operation to finish.

v) Reduce function

The reduce function , it can merge all the intermediate values, that are associated with the same intermediate key.

$$Reduce(k', v') \rightarrow <k', v'>¿$$

vi) Output writer

The reduce function then merge all intermediate values that are associated with the same intermediate key.

$$Reduce(k', v') \rightarrow <k', v'>¿$$

Map reduce allows developers to write and display code that runs directly on each data node server in the cluster.

The code that can understands the format of the data stored in each block in the file and it can be implement with simple algorithm and much more complex ones. It can be used to process vast or massive amount of data in parallel on large clusters in a reliable and fault-tolerant fashion.

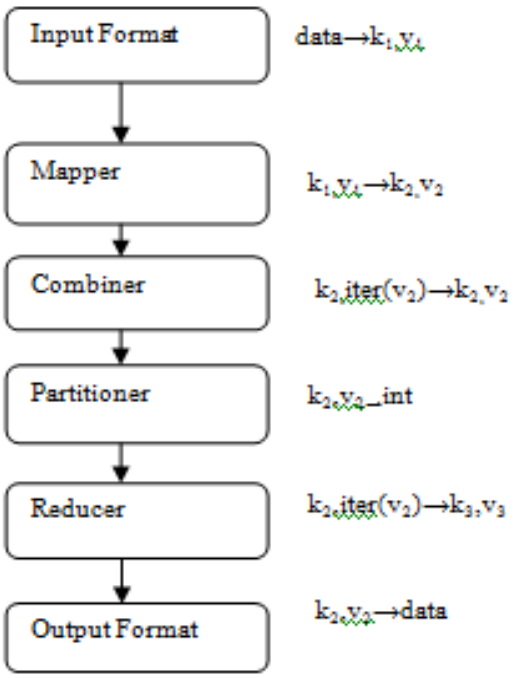

Figure 2. MapReduce steps

III. PROPOSED REPRESENTATION

The architecture of Banian which is divided into three main layers according to logic functions

i)Storage layer
ii)Scheduling and Execution layer
iii)Application layer

These layers are packed into middleware, which can be work on other system with minimal changes.

The storage layer is constructed using HDFS(Hadoop Distributed File System).It stores PB level data with features such as high scalability, compatibility, and fault-tolerance. In the Banian architecture, the storage layer contains three important interface as

i)the interface used for providing the data block distribution information of the file to

the scheduler module through name node

ii)Read/write interface of local data to the query engine module

iii)The red/write interface of HDFS to the ETL module.

The other modules read/write data from/to HDFS actively by calling these API functions. Here Banian can run on newer version of HDFS without requiring changes to its code.

The scheduling and execution layer is the core component of Banian. It contains three
Modules

i)Scheduler
ii)Query Engine
iii)Meta data

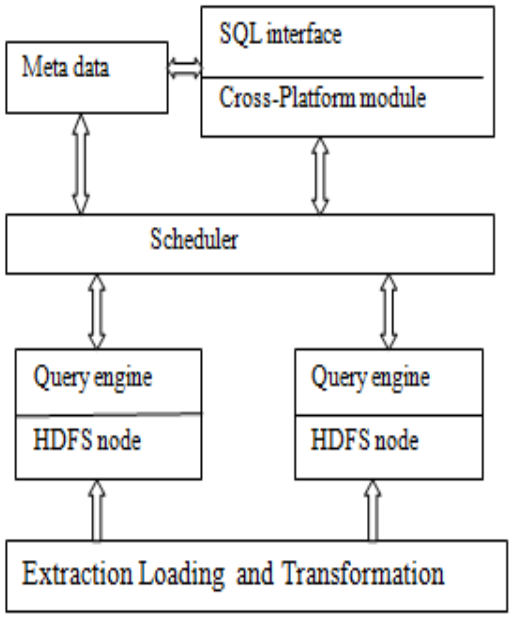

Figure 3. Banian Architecture

The scheduler receives SQL commands from the application layer. Then it adopting the splitting and scheduling technology of parallel database here the commands are split and scheduled to the sub node for concurrent execution. To ensure consistent scheduling of the commands and data, the database table information through the meta data server. After ,That the file information is further analyzed to the position information of data blocks by HDFS name node. The scheduler generate an operation list for local execution at each sub-node.

The application layer can send user's query to any node. The scheduler daemon on this node become the worker node for this request and it is responsible for command split, resource allocation, and result convergence.

The metadata server preserves metadata related to the system. The most important data is the database table structure. To ensure system scalability, all related works are completed using the metadata server, while HDFS is excluded from the construction and maintenance of database tables. when a file is loaded into database table, the corresponding relationship between the file and the table is recorded. The metadata server will inform the scheduler about the files needed to be queried and the method for parsing the file content through the table structure when splitting SQL commands. To speed up command distribution, the metadata server maintains a fast lookup table for caching data block information. According to the normal workflow, the scheduler needs to query twice, once each to metadata sever and the HDFS namenode, to obtain the file information and data block information. The fast lookup table, the scheduler can directly send the operation list to the query engine on the corresponding sub-node in the case of cache hit.

The query engine is deployed on each sub-node. It is responsible for receiving and executing the operation list allocated by the scheduler. Guaranteed by a consistent scheduling strategy, the query engine reads local data directly during execution. Such a design is conducive for optimizing concurrent tasks, reducing data transmission among sub-nodes, and alleviating network pressure. Given that only a local data queue needs to be maintained the system cost is reduced greatly. During execution intermediate results are stored in the memory. After completion of the operation list, the query engine sends the final results to worker node. This is an important distinction with Mapreduce. Furthermore the query engine needs to maintain a regular heartbeat connection with the worker node will restart the task of the failed node on another node that has a copy of the relevant data blocks.

In the era of bigdata business applications request execution of different regions. In terms of transverse compatibility and scalability, Banian provides a unified cross-platform query interface in the application layer. To achieves cross-platform join query Banian allocates a data structure called location for each platform, which is stored in the global table.

Banian provides a distributed structure-oriented ETL interface. The ETL interface offers multi-dimensional strategy choices to support the dynamic balancing of upper applications in terms of ETL cost, storage efficiency, and analysis performance.

Splitting and Scheduling

To improve query performance, the scheduling and execution layer first split SQL commands into subtasks as much as possible and schedule the to different sub-node for concurrent execution.

The scheduling and execution layer in processing SQL commands.

i) Grammatical and lexical analysis is performed by the execution and analysis units to generate the task tree after receiving SQL commands.

ii) It traverse each entry on the task tree, query metadata server according to table information, and retrieving corresponding file information.

iii) Transform tasks into file operations.

iv) Traverse each entry on the operation tree, query HDFS name node according to file information and obtain the corresponding data block position.

v) In data block position all entries in the operation tree at the same sub-node are integrated into an operation list. The coordinator unit sends the operation list to query engine on the corresponding sub- node.

vi) The query engine initiates the workflow after receiving the operation list and directly reads local data for future execution.

vii) Next, After completing all commands in the operation list, query engine sends the results to the aggregation unit.

viii) The aggregation unit collects all results and sends them to the application layer.

The above steps defines the scheduler is the key path for the execution of SQL commands and the core module of Banian. The scheduler is a logical unit as opposed to a physical module. It is composed of the scheduler daemons on each physical node. It define different from the metadata server and HDFS name node, the scheduler has no central node and all physical nodes have equal status. Any scheduler daemon can receive an SQL command and become the worker node for task splitting and scheduling, and collecting results for said command.

The splitting and scheduling process operations is scalability and reliability, because there is no central node, the cluster scale can be extended infinitely number of nodes is limited by the HDFS system and the computing ability will maintain a linear growth rate with increasing cluster size. The model of multiple worker nodes improves query concurrency significantly.

The evaluation results of processing procedure cannot be executed in parallel only for 2%- 4% of the total query time, it says the Banian can be very effective in reducing the query response time by increasing the number of nodes

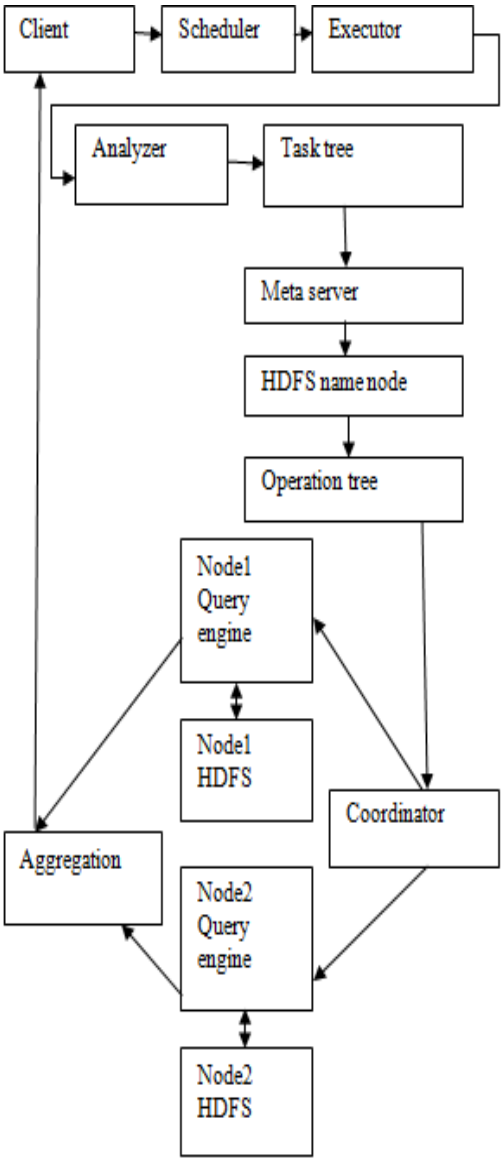

Figure 4. Workflow of splitting and scheduling

Failure detection is important in a large scale distributed system. During the running process the Banian, first the scheduler is responsible for monitoring the health and task completion condition of each node.

Cross-platform query

Bigdata applications often need to access datasets on different platforms that my even be cross-

domain. The structured data of time cost of data extraction and loading cannot meet the real-time requirement.

The cross platform query interface contains three main components SQL interface, cross-platform module, and global table. The SQL interface provides a command shell for users and forwards query commands to the cross platform module.

In case the request command involves several datasets on different platforms the cross platform module queries the global table and gets the information of location. Next it splits the command according to the variable tag name of location sends the subcommand to the slave platform as master, and receives the result.

The global table stores the configuration information of all platforms using a data structure called location.

```
structLocation
{
char*tagname;
char*best;
int port;
int authority;
char*username;
char*password;
}
```

IV. PROPOSED ALGORITHM

Error Detection and Location

In this algorithm the error detection and localization are not so ideal and it is hard to directly use Mapreduce to solve perfectly.

1.Originalalgorithm->Map()/Reduce()

2.Partitio the task flow of algorithm->Identify which part of the task flow to generate a Map Reduce job->Map Reduce generated result returns back to the flow.

3.Complete Map Reduce design->Control flow parallelization/data parallelization.

Map pseudo code

```
DefineMap(Node_inputN,detection_inputD)Correcti ng Set=[];
for each Node in Node_input N if detection flag D(i)==1,then
CorrectingSet=[Node((i-2)%N)];
Node((i-1)%N);
Node(i);
Node((i+1)%N);
Node((i+2)%N);
Error_Location=i;
Endif
Endfor
Return Reduce(Error_Location,CorrectingSet);
```

Reduce pseudo code

```
Define Reduce(Error_Location key, Correcting Servalue):
New_node=[];
New_node.x =xor(Node((i-2)%N).s,Node( (i-2)%N).y);
New_node.y =xor(Node((i+1)%N).x,Node( (i-1)%N).s);
New_node.s=xor(Node((i+2)%N).x);
Node(i)=New_node;
return(Error_Location,New_node);
```

V. CONCLUSION

In order to detect error in bigdata sets from sensor network system a novel approach is developed with cloud computing. First, error classification for bigdata sets is presented. Second, the correlation between sensor network systems and scale-free complex networks are defined. According to each error type and the features from scale-free networks, proposed a time efficient strategy for detecting error and locating error in bigdata sets on cloud. From the experiment results of cloud computing environment, it is demonstrated that, the proposed scale-free error detecting approach can significantly reduce the time for fast error detection in numeric bigdata sets and it also similar for non scale-free network error.

In accordance with error detection for bigdata sets from sensor network systems on cloud the issues such as error correction, bigdata cleaning and recovery, with suitable application will be further explored in future.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.T. Leung, The Google File System, ACM SIGOPS Operating System Review, vol. 37, No.5, pp. 29-43, 2003.

[2] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Cluster, Communication of ACM, vol. 51, No.1, pp. 107-113, 2008.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, The Hadoop Distributed File System, In Processing of IEEE Conference on Mass Storage Systems and Technologies(MSST), pp. 1-10, 2010.

[4] HBase Project, http://hbase.apache.org/, 2014.

[5] D. Borthakur, J. Grap, J.S Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash, et al., Apache Hadoop Goes Realtime at Facebook, in Processing of the 2011 ACM SIGMOD International Conference on Management of Data, New York, NY,USA, pp.1071-1080, 2011.

[6] K. Yu, Large-scale deep learning at Baidu, in Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 2211–2212, 2013.

[7] C. Budak, D. Agrawal, and A. El Abbadi, Structural trend analysis for online social networks, in Proceedings of the VLDB Endowment, vol. 4, no. 10, pp. 646–656, 2011.

[8] L. Pu, J. Xu, B. Yu and J. Zhang, Smart cafe: A mobile local computing system based on indoor virtual cloud, China Communications, vol. 11, no. 4, pp. 38–49, 2014.

[9] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM, vol. 13, no. 6,pp. 377–387, 1970.
[10] L. Bellatreche and K. Y. Woameno, Dimension tabledriven approach to referential partition relational data warehouses, in Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP,New York, NY, USA, 2009, pp. 9–16.
[11] J. Han, J. Y. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng,W. Gong, M. Kamber, K. Koperski, G. Liu, et al.,DBMiner: A system for data mining in relational databasesand data warehouses, in Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, 1997, pp. 326–336.
[12] Y. C. Tay, N. Goodman, and R. Suri, Locking performance in centralized databases, ACM Transactions on Database Systems (TODS), vol. 10, no. 4, pp. 415–462, 1985.
[13] D. Bell and J. Grimson, Distributed Database Systems. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.
[14] D. DeWitt and J. Gray, Parallel database systems:The future of high performance database systems,Communications of the ACM, vol. 35, no. 6, pp. 85–98,1992.
[15] L. Antova, A. El-Helw, M. A. Soliman, Z. Gu, M. Petropoulos, and F. Waas, Optimizing queries over partitioned tables in MPP systems, in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 373–384.
[16] Y. Chen, S. Alspaugh, D. Borthakur, and R. H. Katz, Energy efficiency for large-scale mapreduce workloads with significant interactive analysis, in Proceedings of the 7th ACM European Conference on Computer Systems, 2012, pp. 43–56.
[17] Y. Meng, Z. Luan, and D. Qian, Differentiating data collection for cloud environment monitoring, China Communications, vol. 11, no. 4, pp. 13–24,2014.
[18] M. Zaharia, D. BorthaKur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, Job scheduling for multi-user mapreduce clusters, Technical Report UCB/EECS-2009 55, EECS Department, University of California, Berkeley, USA, April 2009.
[19] I. Elghandour and A. Aboulnaga, ReStore: Reusing results of MapReduce jobs in pig, in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 701–704.
[20] M. Stonebraker, D. Abadi, D. J. DeWitt, S.Madden, E. Paulson, A. Pavlo, and A. Rasin, MapReduce and parallel DBMSs: Friends or foes? Communications of the ACM, vol. 53, no. 1, pp. 64–71, 2010.
[21] Greenplum Inc., Greenplum Database: Powering the data driven enterprise, http://www.greenplum.com/resources,2014.
[22] Y. Xu, P. Kostamaa, and L. Gao, Integrating hadoop and parallel DBMs, in Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, 2010, pp. 969–974.
[23] F. N. Afrati and J. D. Ullman, Optimizing multiway joins in a map-reduce environment, IEEE Transactions on Knowledge & Data Engineering, vol. 23, no. 9, pp. 1282–1298, 2011.
[24] H. Herodotou and S. Babu, Profiling, what-if analysis, and cost-based optimization of MapReduce programs, in PVLDB, 2011, pp. 1111–1122.
[25] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads, iPVLDB,2009, pp. 922–933.