International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

An Experimental Evaluation of Data Placement Scheme Using Hadoop Cluster

G. Sasikala[1], N. Meenakshi[2],
[1]*Dept. of computer science, Valliammai Engineering College, TamilNadu, India.*
[2]*Assistant Professor, Dept. of computer science, Valliammai Engineering College, TamilNadu, India.*

Abstract – A network bandwidth is steadily increasing at much slower rate than the corresponding increase in CPU performance. This states that there is a widened gap between CPU and network speed. In this project, we investigate improvements in data compression using the MapReduce framework. The objective of this work is to reduce the amount of data transferred over the network from local file system to Hadoop file system, thereby increasing computation speed. We have conducted experiments for comparing on how various data compression software's like WinZip and WinRar along with MapReduce framework compress the data files. We have found that the MapReduce Framework is producing not so bad results and proves that further tuning the method would provide better results.

KEYWORDS: Big data, Data placement, Data compression, Hadoop, Mapreduce.

1. INTRODUCTION

Distributed Computing is a field of computing science that studies distributed systems. A distributed computing is a software system component placed on networked computers transferred and organizes their action by cursory information. The factors cooperate with each other's in order to achieve a typical goal. There are many options for the message passing method, including RPC like connectors and report queues. Three powerful characteristics of distributed system are: consistency of factors, absence of global clock and independent failure of the segment. A crucial challenge of distributed system is communication transparency.

Distributed system varies from SOA-based systems to densely multiplayer network business to associate utilization; a computer program that runs in distributed system. It is also suggest to the use of a shared system to examine computational issues. In distributed computing, a problem is split into many functions, each other by information passing. A distributed system common goal, such as solving a large problem computational problem. Alternatively, every computer may have its own user with particular needs, and the purposes of the distributed system are to coordinates the use of shared resources or provide communication service to the user.

Properties

- The system has to tolerate failures in included computers.
- The structure of system is not known in advance the system may consist of different kinds of computing and network links and the system may innovation during the execution of a shared program.
- Each computer has only a narrow, partial view of the system.

1.1 Parallel computing

Parallel computing is the model for computing many calculations are carried out together, performing on the foundation that huge problems can often be divided into smaller ones, which are being determined simultaneously. There are several various prototypes of lateral computing: appearance level, the data and responsibility parallelism. Parallel computer programs are more difficult to write then progressive ones, because concurrency hosts several new classes of potential software bugs, of which race conditions are the most common. Transformation and synchronization between the different subtasks are usually some of the greatest difficulties to getting good parallel package performance.

Advantage

- Save time/money
- Solve large problems
- Provide concurrency
- Use of non-local resources
- Limited to serial computer

1.2 Big data

Big data [22] is the term for a collection of testimony sets so large and complex that it becomes difficult to measure using database authority machine or traditional data processing operations. The methods include capture, maturation, repository, research, distribution, relocation, analysis and visualization. The tendency to large datasets is due to the supplementary information available for investigation of a single large set of relevant data, as compared to independent smaller sets with the same total amount of data. Bid data

difficult to work with using most relational database management systems and desktop statistics and visualization packets, prescribed instead huge parallel program running hundreds or even thousands of servers.

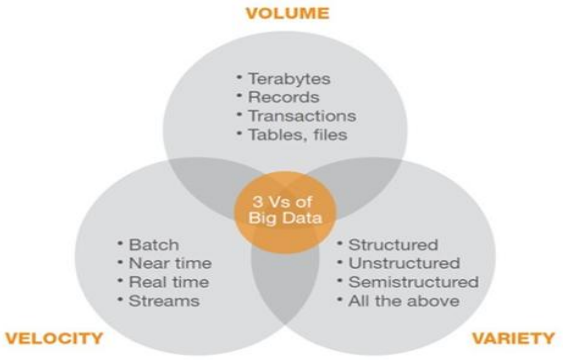


Fig 1: Big data

• Hadoop-Apache Infrastructure

• Mongo DB - MongoDB,Inc

• Splunk - Splunk Inc

1.3 Map Reduce

MapReduce [8] is software framework that allows developers to write program massive amounts of unstructured data in parallel across a distributed cluster of process or stand-alone computers. It is programming model for processing large data set with a parallel, shared procedure on a cluster. It is a shareware scheme for efficiently writing function which practice large number of data on lateral on large clusters of product hardware in a predictable, fault-tolerant manner. Because all nodes in the cluster are typical to report back scientifically to concluded work and quality restore. If a node remains silent for deeper than the regular interruption, name node makes note and assigns the work other nodes.

MapReduce job generally separate the input data into independent portion which are handled by the map tasks in a completely parallel manner. Normally both the input and output of the job are gathered in an information system. These method tasks care of appoint function, controlling them and executing the failed tasks. The framework consists of a particular master node and single slave node per cluster. The master is responsible for appointing the job's fundamental nodes on the slaves, monitoring them and running the failed tasks. The slaves execute the tasks as conducted by the master.

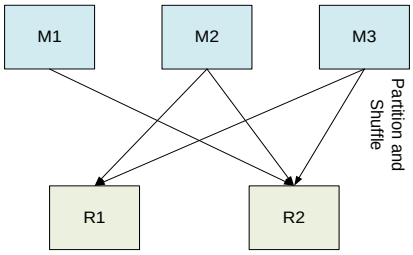
The framework divides into two parts:

- **Map ()** procedure that performs filtering and sorting. A function that package out work to different nodes in the shared cluster.
- **Reduce ()** procedure that performs a summery operation and that assembles the work and decides the results into a single value.

■ **Mapper:** Identity function for value

(v, e) → (v, _)

■ **Reducer:** Identity function (v', _) -> (v', "")



MAPREDUCE DESCRIPTION

Class Mapper

Method Map (nid n1, node N1)

A←N1. PageRank/ N1. AdjacencyList

Emit (nid n1, N1)

for all nodeid m € N.AdjancyList do

Emit (nid m,a)

Class Reducer

Method Reduce (nid m2, [a1, a2, …])

M2←Ø

for all a € counts [a1, a2,…..] do

if IsNode (a) then

M2←a

else

s←s+a

M2.PageRank ← s

Emit (nid m2, node M2)

Minimally, applications examine the input/output location and supply map and reduce function via implementations of relevant interfaces and abstracts collections. These other job

framework, comprise the job configuration. The hadoop job client them submits the job and configuration to the JobTrackers which then assumes the responsibility of sharing the software information to the slaves, development tasks and monitoring them, providing status and diagnostics information to the job-clients.Although hadoop framework is implemented in java, MapReduce application need not be written in java.

MAP REDUCE ARCHITECTURE

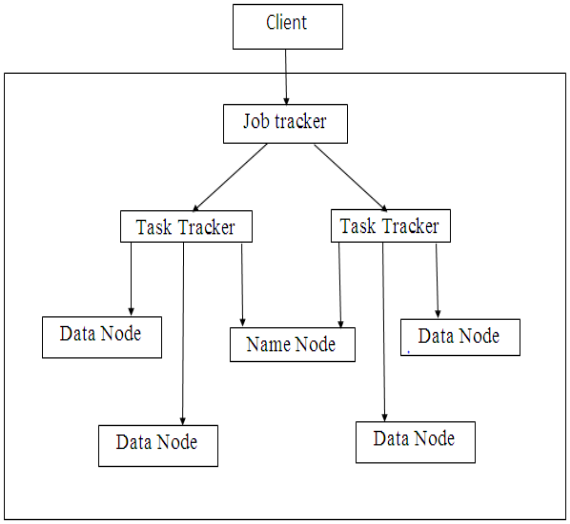


Fig 2: Map Reduce

1.4 Hadoop

Hadoop is a free; Java based prioritizing method that supports the transformation of huge data sets in shared computing surroundings. It is part of Apache project sponsored by the Apache Software Foundation. Hadoop is makes it possible to run operation on system with thousands of task involving thousands of terabytes. These methods lower risk of catastrophic system defeat, even if a huge number of nodes become defective. All the modules in Hadoop are designed with an assumption that hardware failures are common and thus should be naturally handled in software by the framework. Hadoop was developed by Google's MapReduce, a software method in which an application is broken down into various small parts. Any of those parts can be run on any node in the cluster. The Apache Hadoop framework is composed of following modules:

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop YARN

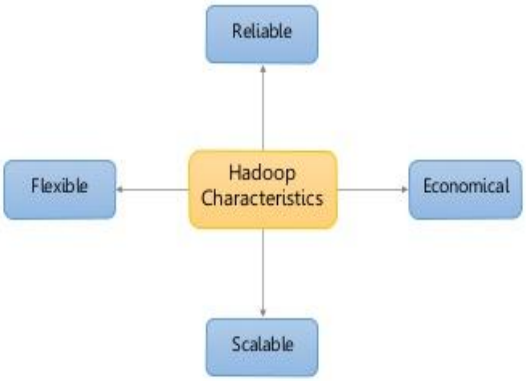


Fig 3: Hadoop Characteristics

2. RELATED WORKS

There are several purpose of data compression [1] in HDFS ADSS Cloud the author proposed to develops a data dependency based on the data placement cloud system, which cluster the relative data as intensively as possible to effectively reduce data movement during the workflows such as MapReduce which relies on the traced divide and data locality, the relevant data desires to be distributed as evenly as possible. Improving MapReduce Performance through data placement [3] the author suggested to takes data locality into account for launching hypothetical MapReduce tasks in heterogeneous locations. They focus on stabilizing data handling load based on network topology and disk space utilization of cluster. In contrast, DRAW focuses on data sharing based on data sets. MARP the author suggested to developing a set of MapReduce APIs for data supplier who may be awake of the successive access organization of the data after being transfer. By determine the access patterns; the data method performance can be achieved. However, it involves utilization developers to specify the data access patterns earlier. Our DRAW [3] captures organizing patterns by runtime device. Dynamic Energy Efficient Data the author suggested addressing the obstacle of energy exchange for huge datacenters that run MapReduce jobs. We also examine that servers are most efficient when used at higher utilization level. In this context, we proposed an energy proficient data assignment and cluster arrangement algorithm that dynamically measures the cluster in agreement with the workload established on it. However, the efficiency of this approach would depend on how quickly a node can be stimulated. RC File a Fast and Space-efficient Data Placement the author proposed to evaluate the row vise, the column vise, and the hybrid PAX store executed in predictable database system. Detailed relationship, analysis, and improvement of these three structures data compression and query performance.

3. SYSTEM MODEL

The Hadoop default random data placement [4] method, the overall data sharing may be balanced, but there is no security that the data accessed as a group is randomly distributed. More specifically, a MapReduce job is divided into many map tasks to action in parallel. Map tasks resolve to be allocated the nodes with the needed data narrowly being stored to achieve high compute storage. Without evenly distributed grouping data, some map tasks are either scheduled with other nodes, which casually access the required data, or they are expected with these data property nodes but have to wait in the queue. These plan responsibility violate the data locality and severely burden down the MapReduce performance.

We establish a new Data placement method (DRAW) that takes into account the data organization property to powerful improve the performance for data comprehensive application with concern locality. Without loss of usual, data placement is designed and implemented as a Hadoop version model. By evaluating with real world genome indexing and astrophysics application, DRAW is able to execute up to 59.8% more local map tasks in comparison with random placement. In addition, DRAW reduces the integration time of map stage by up to 41.7% and the MapReduce task execution time up to 36.4%.

3.1 Data Compression

Existing data compression [1] while performing I/O, we analyzed common compression algorithms and studied their overall I/O performance. Then evaluate the compression algorithm performance to be increased. Our work compressed data have been, reduce the time and increase the I/O performance.

3.2 Data Placement

Existing data parallel scheme, e.g. Hadoop, or Hadoop based clouds, shared data using the random placement [3] method for integrity and load balance. However, we examine that many data comprehensive application model concern localities which only sweeps, part of a big data set. The data generally accessed together result from their grouping elements.

3.2.1 Data Placement Scheme

Develops a data dependency-based data placement

History Data Access Graph (HDAG)

Data Grouping Matrix (DGM)

Optimal Data Placement Algorithm (ODPA).

4. PROPOSED WORK

We examine improvements to I/O performance by utilizing the data Compression [7]. We harness unproductive CPU assets to wrap network traffic, decreasing the amount of data transferred over the network and increasing powerful network bandwidth. Variety of data sets conducted experiments on a high performance computing. We presenter recent analysis of how compressing I/O traffic can affect application I/O throughput [17]. In this new, interconnected bandwidth between nodes and various file systems will be premium. By trading CPU performance for reducing in data size, the Effective network performance can be increased.

Illustrate the data compression business and the integration with different types of information systems. We focus on data compression in high performance computing cluster to reduce transmission latency, increase network frequency and increase the performance of the I/O. Our work focuses on Improving I/O performance in order to reduce cluster power utilization. Improving I/O performance to achieve faster time to science. While data compression methods can decrease the storage impression of experimental data, universal data compression services for file I/O are not readily available. Input/output forwarding the client, server and executes the I/O requests forwarded by the clients. After receiving a request, the server solves and manages the execution of the I/O operation using different apparatus. The state machine executes and transfers with the client to restore any supplementary data required by the I/O requests and communicates the results of the I/O process to the client. We evaluated data compression used in these methods to place different significance on the compression speed; time to compress the data and the compression ratio.

On the synthesis cluster, we used the distributed memory; RAM disk in our evaluation narrow file system noise and detached the storage system. Shows that observed transfer rate when reading and writing each of the files representable in the result. Any result showing a read speed higher than the line indicates the compression increased I/O throughput. The graph shows the three types of files can be compressed in the file system. They have,

Mapreduce

Winzib

Winrar.

Advantages

• Network bandwidth to be increased

• Reducing in data size compression the data file size can decrease.

• Improve bandwidth

• Increase I/O performance

• Faster time to access the data

4.1 ARCHITECTURE DIAGRAM

In the result varying the compressed data rate of the selected file system. We are also investigating an adaptive compression approach, in which the best compression algorithm is selected, are automatic. Data compression has been actively studied to explore the feasibility of reducing the data size in the context of parallel processing systems. Design and implement a transparent data compression layer in the distributed file system, so that applications would read and write data.

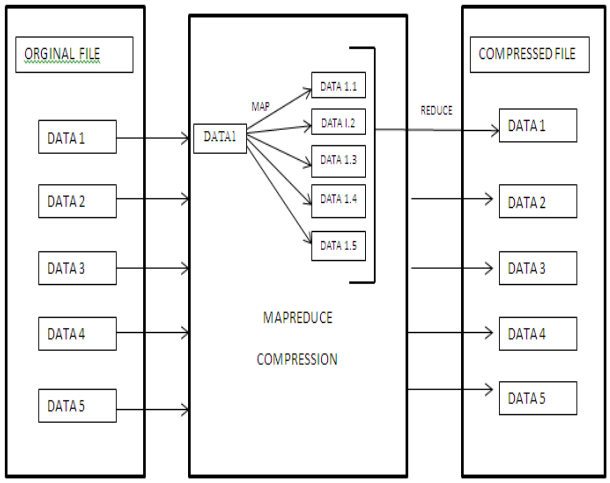


Fig 4: Architecture

5. EXPERIMENT EVALUATION

To evaluate our data compression [7] scheme based on Adaptive data compression algorithm, we generated the real data size can be decrease the actual file size data. The physical hadoop implement the Hadoop 0.19.1. Then Cygin software can be implemented to the hadoop file system.

5.1 System Setup

1. Download hadoop 0.19.1 and place in some folder on your computer such as Java.

2. Open Cygin

3. Execute the command: cd

4. Execute the command to enable your home directory folder to be shown in the Windows

Explorer.

5. Open next Explorer window and operate to the folder that contains the downloaded Hadoop file.

6. Copy the Hadoop file into your home directory folder.

5.2 Algorithm

ODPA ALGORITHM

Consider the sub matrix M[n][n]; where n is number

of nodes

for each row from M[n][n] do

R = the index of current row;

Find the minimum value V in this row;

Put this value and its corresponding column

index C into a set MinSet;

MinSet = C1, V1, C2, V2;

if there is only one tuple (C1,V1) in MinSet then

DP[O][R] = R;

DP[l][R] = C1;

Mark column C1 is invalid (already assigned);

Continue;

end if

for each column Ci, from MinSet do

Calculate Sum[i] = sum(M[*][Ci]);

end for

Choose the largest value from Sum array;

C = the index of the chosen Sum item;

DP[O][R] = R;

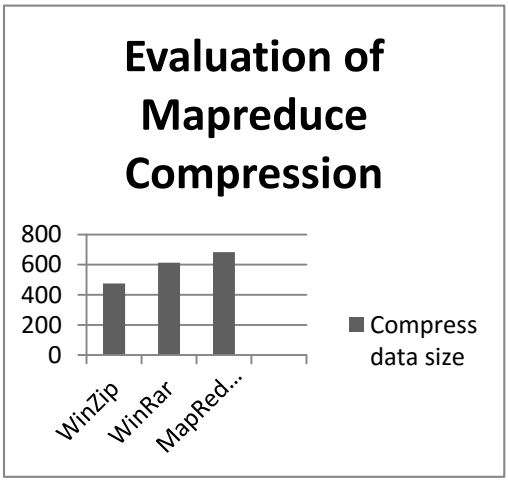
DP[l][R] = C;

Mark column C is invalid (already assigned);

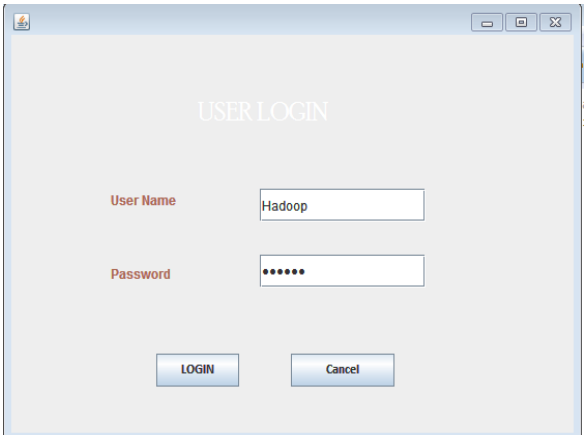
end for

6. RESULT AND GRAPH

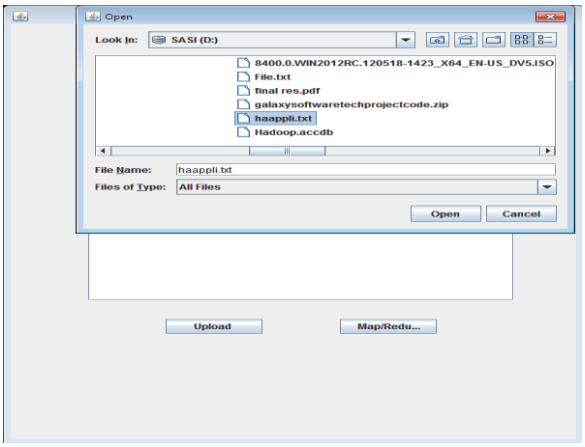
In the experiment, we evaluated the map [8] phase compression time, according to completion rate for our data compression scheme based Adaptive data compression algorithm (ADC). Performance evaluations show that our data placement scheme reduces the file size. Compressed the various data files of the, we can differ from the file size. The file can be applied to the Mapreduce performance, the ordinary file size decreased from the Mapreduce file size. An example of the evaluating the Mapreduce 50 KB of file can be performed to the data compression algorithm after the result reduced size is 24.08 KB.

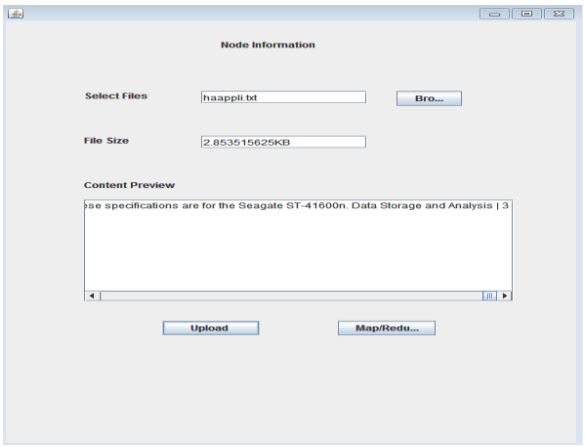


1. User login

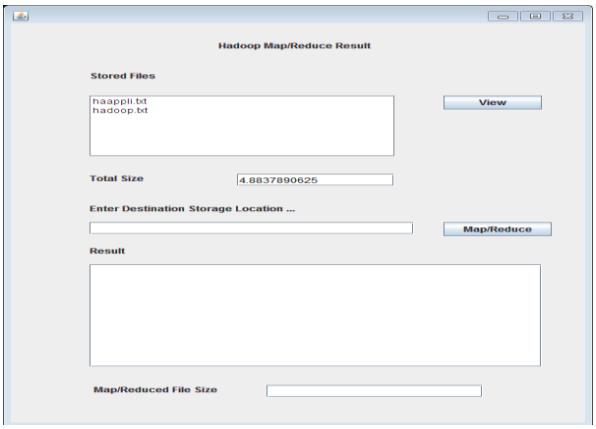


2. Select File

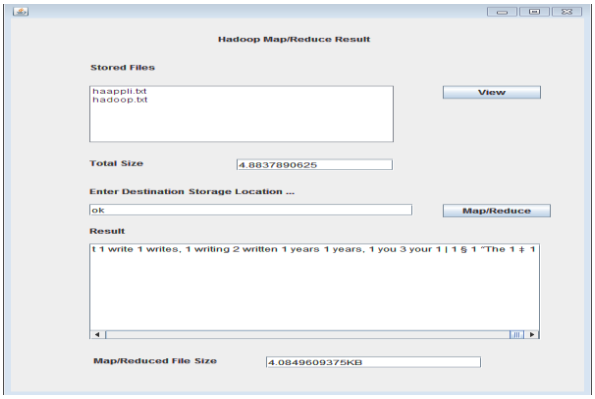




3. View file size



4. Map reduce file size



CONCLUSION

In this paper, we proposed a feasibility of Adaptive data compression while performing I/O. We analyzed common compression algorithms and effect on the overall I/O performance. To evaluate compression algorithm performance on I/O, we selected experimental testimony gather from multiple fields of measurement. For certain type of data, we observed significant bandwidth improvements. Our work shows that the benefits of compression data prior to transmission are highly dependent on the data being transferred. We are investigating an Adaptive compression approach, in which the best compression Algorithm is selected automatically. In different location compressed the data into vary to the Mapreduce result. If the data compression time can be reduced. We can improve the I/O Performance in data compression. Input file size have been 52KB, then mapreduce size is 23KB, Winrar is 15KB and Winzip is 14.08KB.

REFERENCES

[1] .R.Alameldeen and D.A.Wood, "Adaptive cache compression for high-performance processors", in ISCA 04: Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004, pp 212.

[2] Ali.N, P.Carns, K.Iskra, D.Kimpe, S.Lang, R.Latham, R.Ross, L.Ward, and P.Sadayappan, "Scalable I/O forwarding framework for high-performance computing systems", in IEEE International Conference on Cluster Computing, 2009.

[3] Dong Yuan, Yuan Yong, Xiao Liu and Jinjun Chen. "A Data placement strategy in scientific cloud workflows", Future Generation Computer System, 26:1200-1214, October 2010.

[4] Dongfang Zhao, Chen Shou, Zhao Zhang, Iman Sadooghi, Xiaobing Zhou, Tonglin Li and loan Raicu. FusionFS. "A distributed file system for large scale data-intensive computing", 2nd Greater Chicago Area System Research Workshop, 2013.

[5] Alameldeen. A and D.A.Wood, "Adaptive cache compression for high-performance processors", in ISCA '04: Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004, pp. 212.

[6] Bicer.T, J.Yin, D.Chiu, G.Agarwal and K.Schuchardt. "Integrating Online Compression to Accelerate Large-Scale Analysis Applications", IEEE International Parallel and Distributed Processing Symposium, 2013.

[7] Barr.K.C and K.Asanovi´c, "Energy-aware lossless data compression", ACM Trans. Comput. Syst., vol. 24, no. 3, pp. 250–291, 2006.

[8] Chen.Y, A.Ganapathi, and R.Katz, "To compress or not to compress compute vs. IO tradeoffs for MapReduce energy efficiency", in Proceedings of the First ACM SIGCOMM Workshop on Green Networking. ACM, 2010, pp. 23–28.

[9] Canal.R, A.Gonz´alez, and J.E.Smith, "Very low power pipelines using significance compression", in MICRO 33: Proceedings of the 33rd annual ACM/IEEE International Symposium on Microarchitecture, 2000, pp. 181–190.

[10] Dong.W, X.Chen, S.Xu, W.Wang, and G.Wei, "Proxy based object packaging and compression: A web acceleration scheme for UMTS", in WiCOM'09: Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009, pp. 4965–4969.

[11] Dean.J and S.Ghemawat, "MapReduce:Simplified Data Processing on Large Cluster", Proc. Sixth Conf. Symp. Operating Systems Design and Implementation(OSDI), 2004.pp.113-125

[12] Das.R, A.Mishra, C.Nicopoulos, D.Park, V.Narayanan, R.Iyer, M.Yousif, and C.Das, "Performance and power optimization through data compression in network-on-chip architectures", in IEEE 14th International Symposium on High Performance Computer Architecture, 2008., 2008, pp. 215–225.

[13] Kothiyal.R, V.Tarasov, P.Sehgal, and E.Zadok, "Energy and performance evaluation of lossless file data compression on server systems", in SYSTOR '09: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, 2009.

[14] Mogul.J.C, F.Douglis, A.Feldmann, and B.Krishnamurthy, "Potential benefits of delta encoding and data compression for HTTP", in SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 1997, pp. 181–194.

[15] Ma.M.J and R.Bartos, "Analysis of transport optimization techniques", in ICWS '06: Proceedings of the IEEE International Conference on Web Services, 2006, pp. 611–620.