International Journal of Advanced Research in
Computer Science Engineering and Information Technology
Volume: 1 Issue: 3 08-Jan-2014,ISSN_NO: 2321-3337

Efficient Data Processing in the cloud Using Transpose-Minify Framework

M. Florence Dayana[1] ,G. Nalini[2]
Head, Dept.of BCA , Bon Secours College for Women, Thanjavur.
Asst.Professor, Dept.of Computer Science, MaruduPandiyar College, Thanjavur.
florencedayana@gmail.com ,nalinigurumurthi@gmail.com

Abstract—*With the advent of the cloud computing technology the user can operate the data and perform the computations anywhere, anytime in the world. Cloud computing provides highly scalable services to be easily consumed over the Internet on an as-needed basis. The interest thing in cloud computing has been motivated by many factors such as the low cost of system hardware, the increase in computing power and storage capacity and the massive growth in data size generated by digital media, Web authoring, scientific instruments, physical simulations,etc. To this end, still the main challenge in the cloud is how to effectively store, query, analyse, and utilize these immense datasets. To provide the solution to this problem in this paper a novel highly decentralized software framework called Transpose-Minify Framework is used for effectively managing the data*.

Keywords— map, reduce, data processing, transpose, minify.

1. INTRODUCTION

Cloud computing has been coined as an umbrella term to describe a category of sophisticated on-demand computing services initially offered by commercial providers, such as Amazon, Google, and Microsoft. It denotes a model on which a computing infrastructure is viewed as a cloud from which businesses and individuals access applications from anywhere in the world on demand. The main principle behind this model is offering computing, storage, and software "as a service. In addition to raw computing and storage, cloud computing providers usually offer a broad range of software services. They also include APIs and development tools that allow developers to build seamlessly scalable applications upon their services. Indeed, the long-held dream of delivering computing as a utility has been realized with the advent of cloud computing. Cloud computing provides software as a service (saas), Platform as a service(paas), Infrastructure as a service(IaaS). Cloud provides feature such as pay for usage (metering and billing), elasticity, self service, and customization. Further cloud provides deployment models such as Private cloud, Public cloud, Hybrid cloud. The important feature offered by the cloud is the users data resides anywhere in the world and which can be operated remotely in unknown machine by the user. In the day-to-day life the organizations produce large amount of data. The interest thing in cloud computing has been motivated by many factors such as the low cost of system hardware, the increase in computing power and storage capacity and the massive growth in data size generated by digital media(images, video, audio), Web authoring, scientific instruments, physical simulations,etc. To this end, still the main challenge in the cloud is how to effectively store, query, analyse, and utilize these immense datasets. To effectively manage the data produced by the organizations in this paper a novel highly decentralized software framework implemented which is called Map Reducing Technology.The main aim of this paper is to combine cloud computing Technologies to efficiently store, query, analyse, and utilize the organizations data.This paper is structured as follows: Section II presentsthe Transpose-Minify is a software framework for data processing in the cloud. In section III contains the existing and proposed system analysis. Further in section IV discussed about main features of transpose-minify framework.In section V implements the sculptor- Serf architecture for the Transpose-Minify Framework. Further in section VI the Transpose-Minify Framework is implemented using the Transpose and minify functions. Section VII discussed about the some of the related implementations for the Transpose-Minify Framework done in the cloud. Finally section VII presents our conclusions and future work.

2. TRANSPOSE-MINIFY MODEL

Transpose-Minify is a software framework for solving many large-scale computing problems.. By using this programming model large set of data sets can be processed. Transpose-Minify has the two main functions Transpose and Minify.

2.1 Transpose Function

This function performs the searching and sorting of the similar data items.

2.2 Minify Function

This procedure performs the summary operation.

The Transpose-Minify provides many useful features such as simplicity, fault tolerance, and scalability. It is the most powerful realization of data-intensive cloud computing programming. It is often advocated as an easier-to-use, efficient and reliable replacement for the traditional data intensive programming model for cloud computing. It is proposed to form the basis of the data-centre software stack. The Transpose-Minify can be applied in many fields such as data and compute-intensive applications, machine learning, graphic programming, multi-core programming.

3. SYSTEM ANALYSIS

3.1 Existing System

In the past for managing the large amount data produced by the organization the traditional data intensive system was used which is not suitable for the cloud computing due to the bottleneck of the Internet when transferring large amounts of data to a distant CPU. The drawbacks of the traditional method are it is lack in scalability and it has no enough space to store large amount of data. And also it does not support the query processing.

3.2 Proposed System

In proposed system a novel approach, namely Transpose-Minify software framework is use to effectively manage the large data sets. The main feature of this model is simplicity, fault tolerance, and scalability. In this model the computing and data resources are co-located, thus minimizing the communication cost and benefiting the service providers.

4. MAIN FEATURES OF TRANSPOSE-MINIFY FRAMEWORK

4.1 Simplicity:

The Transpose-Minify runtime is responsible for parallelization and concurrency control, this allows programmers to easily design parallel and distributed applications.

4.2 Manageability:

It provides the two level of management

(a) To manage the input data- prepare the data to execute.

(b) To manage the output data- to get the reduced data.

4.3 Scalability:

When the node increases then the performance of the Transpose-Minify potentially increases.

4.4 Fault Tolerance and Reliability:

The data in the GFS are distributed on clusters with thousands of nodes. Thus any nodes with hardware failures can be handled by simply removing them and installing a new node in their place. Moreover, Transpose-Minify, taking advantage of the replication in GFS, can achieve high reliability by (1) rerunning all the tasks (completed or in progress) when a host node is going off-line, (2) rerunning failed tasks on another node, and (3) launching backup tasks when these tasks are slowing down and causing a bottleneck to the entire job.

5. TRANSPOSE-MINIFY IMPLEMENTATION

The Transpose-Minify framework uses the sculptor- Serf architecture (Fig.1).

5.1Transpose step:

The sculptor node takes the input, divides it into smaller sub-problems, and distributes them to Serf nodes. A Serf node may do this again in turn, leading to a multi-level tree structure. The Serf node processes the smaller problem, and passes the answer back to its sculptor node.

5.2 Minify step:

The sculptor node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

5.3 The Function of sculptor

The sculptor also called master, is responsible for Querying the Name Node for the block locations, Scheduling the tasks on the slave which is hosting the task's blocks, and Monitoring the successes and failures of the tasks.

5.4 The function of Serf

The Serf also called slave, execute the tasks as directed by the master.

6. IMPLEMENTATION OF TRANSPOSE-MINIFY

6.1Steps

- Prepare the input from the user.
- Then the input is send to the sculptor of the architecture.
- The sculptor segments the input and assigns it to all the Serf nodes.
- The Serf nodes process the inputs and produce the relevant output.
- Then the Serf node sends the out file to the sculptor

For doing the above mentioned steps the pivotal - appraisal pair is generated for Transpose and Minify functions

- `For` Transpose `(P1, A1)` → `list (P2, A2)`

- `For reducing (P2, list (A2))→list (A3)`

The Transpose Minify library in the user program first splits the input files into M pieces of typically 16 to 64 megabytes (MB) per piece. It then starts many copies of the program on a cluster. One is the "sculptor" and the rest are "Serf." The sculptor is responsible for scheduling (assigns the Transpose and Minify tasks to th e worker) and monitoring (monitors the task progress and the Serf health).

When Transpose tasks arise, the sculptor assigns the task to an idle Serf, taking into account the data locality. A Serf reads the content of the corresponding input split and emits a pivotal /appraisal pairs to the user-defined Transpose function. The intermediate key/value pairs produced by the Transpose function are first buffered in memory and then periodically written to a local disk, partitioned into R sets by the partitioning function. The sculptor passes the location of these stored pairs to the Serf which reads the buffered data from the Transpose using remote procedure calls (RPC). It then sorts the intermediate keys so that all occurrences of the same key are grouped together. For each key, the worker passes the corresponding intermediate value for its entire occurrence to the Minify function. Finally, the output is available in R output files (one per Minify task).

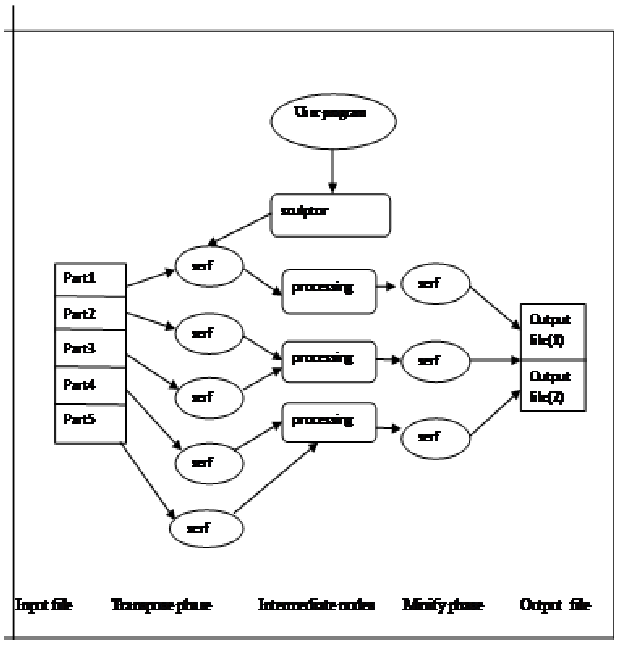

Figure.1 Sculptor- Serf architecture

7. TRANSPOSE-MINIFY IMPLEMENTATIONS FOR THE CLOUD

The Transpose-Minify framework can be implemented in various fields using the cloud computing technology.

7.1Hadoop:

The Hadoop common [7], formerly Hadoop core, includes file System, RPC, and serialization libraries and provides the basic services for building a cloud computing environment with commodity hardware. The two fundamental subprojects are the Transpose-Minify framework and the Hadoop Distributed File System (HDFS).The Hadoop Distributed File System is a distributed file system designed to run on clusters of commodity machines. It is highly fault-tolerant and is appropriate for data-intensive applications as it provides high speed access the application data. The Hadoop

Transpose-Minify framework is highly reliant on its shared file system (i.e., it comes with plug-ins for HDFS, Cloud Store [15], and Amazon Simple Storage Service S3 [16]).

7.2 Disco

Disco is an open-source Transpose-Minify implementation developed by Nokia [21]. The Disco core is written in Erlang, while users of Disco typically write jobs in Python. Disco was started at Nokia Research Center as a lightweight framework for rapid scripting of distributed data processing tasks. Furthermore, Disco has been successfully used, for instance, in parsing and reformatting data, data clustering, probabilistic modeling, data mining, full-text indexing, and log analysis with hundreds of gigabytes of real-world data. Disco is based on the master-slave architecture. When the Disco master receives jobs from clients, it adds them to the job queue, and runs them in the cluster when CPUs become available. On each node there is a Worker supervisor that is responsible for spawning and monitoring all the running Python worker processes within that node. The Python worker runs the assigned tasks and then sends the addresses of the resulting files to the master through their supervisor.

7.3Mapreduce.NET

Mapreduce.NET [22] is a realization of Transpose-Minify for the.NET platform. It aims to provide support for a wider variety of data-intensive and compute intensive applications (e.g., MRPGA is an extension of Transpose-Minify for GA applications based on Transpose-Minify.NET [23]). Transpose-Minify.NET is designed for the Windows platform, with emphasis on reusing as many existing Windows components as possible. The Transpose-Minify.Net runtime library is assisted by several components services from Aneka [24, 25] and runs on WinDFS. Aneka is a.NET-based platform for enterprise and public cloud computing. It supports the development and deployment of.NET-based cloud applications in public cloud environments, such as Amazon EC2.Besides Aneka, Reduce.NET is using WinDFS, a distributed storage service over the.NET platform. WinDFS manages the stored data by providing an object-based interface with a flat name space. Moreover, MapReduce.NETcan also works with the Common Internet File System (CIFS) or NTFS.

7.3 Skynet

Skynet [17, 26] is a Ruby implementation of Transpose-Minify, created by Geni.Skynet is“ an adaptive, self-upgrading, fault-tolerant ,and fully distributed system with no single point of failure” [17]. At the heart of Skynet is plug-in based message queue architecture, with the message queuing allowing workers to watch out for each other. If a worker fails, another worker will notice and pickup that task. Currently, there are two message queue implementations available: one built on Rinda that uses Tuplespace and one built on MySQL.Skynet works by putting “tasks” on a message queue that are picked up by skynet workers. Skynet workers execute the tasks after loading the code at startup; Skynet tells the worker where all the needed code is. The workers put their results back on the message queue.

7.5Grid Gain

Grid Gain is an open cloud platform, developed in Java, for Java. Grid Gain enables users to develop and run applications on private or public clouds. The Transpose-Minify paradigm is at core of what Grid Gain does. It defines the process of splitting an initial task into multiple subtasks, executing these subtasks in parallel and aggregating (reducing) results back to one final result. New features have been added in the Grid Gain Transpose-Minify implementation such as: distributed task session, checkpoints for long running tasks, early and late load balancing, and affinity co-location with data grids.

8. CONCLUSION AND FUTUREWORK

In this paper introduces the Transpose-Minify Framework Which is important programming model for next-generation distributed systems, namely cloud computing. The Transpose-Minify implementations for cloud computing, especially data- and compute-intensive cloud computing owned by different organizations. In this paper presented the different impacts of the Transpose-Minify model in the computer science discipline, along with different efforts around the world. It can be observed that while there has been a lot of effort in the development of different implementations of Transpose-Minify, there is still more to be achieved in terms of Transpose-Minify optimizations and implementing this simple model indifferent areas. The future work of this paper is performing the optimizations using the Transpose-Minify Frame work.

REFERENCES

[1]. I. Foster, Yong Zhao, I. Raicu and S. Lu, Cloud computing and grid computing 360-degree compared, in Proceedings of the Grid Computing Environments Workshop (GCE ’08), 2008, pp.
[2].A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. The Importance of Data Locality in Distributed Computing Applications, in Proceedings of NSF Workflow,2006.
[3]. A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. Slutz, and R. J. Brunner, Designing and mining multi-terabyte astronomy archives: The Sloan Digital Sky Survey, in Proceedings of the SIGMOD International Conference on Management of Data, 2000, pp. 451_462.
[4]. J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM, 51(1):107_113, 2008.
[5]. S. Ghemawat, H. Gobioff, and S. T. Leung, The Google File System, in Proceedings of the 19th ACM Symposium on Operating Systems Principles, LakeGeorge, NY, October, 2003, pp.
[6]. D. A. Patterson, Technical perspective: The data center is the computer, Communications
of the ACM, 51(1):105, 2008, pp. 105_105.
[7]. Hadoop: http://lucene.apache.org/
[8].Yahoo!, Yahoo! Developer Network, http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html (accessed September 2009).
[9].Hadoop, Applications powered by Hadoop: http://wiki.apache.org/hadoop/ PoweredB

[10]. Presentation by Randal E. Bryant, Presented in conjunction with the 2007 Federated Computing Research Conference, http://www.cs.cmu.edu/~bryant/ presentations/DISC-FCRC07.ppt.
[11]. L. Barroso, J. Dean, and U. Holzle, Web search for a planet: The Google cluster architecture, IEEE Micro, 23(2), 2003, pp. 22_28.
[12].MapReduce in Wikipedia, http://en.wikipedia.org/wiki/MapReduce (accessed September2009).
[13].Hadoop in Wikipedia, http://en.wikipedia.org/wiki/Hadoop (accessed September 2009).
[14].CNET news, http://news.cnet.com/8301-13505_3-10196871-16.html (accessed September 2009).
[15].CloudStore (Formerly Kosmos File System), http://kosmosfs.sourceforge.net/
[16].Amazon Simple Storage Service, http://aws.amazon.com/s3/
[17].Ruby MapReduce Implementation, http://en.oreilly.com/rails2008/public/ schedule/detail/2022 (accessed September 2009).
[18]. Cloudera Homepage, http://www.cloudera.com/
[19].OpensolarisHadoop Live CD, http://opensolaris.org/os/project/livehadoop/
[20].Amazon Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/
[21]. Disco Project Homepage, http://discoproject.org/
[22]. C. Jin and R. Buyya, MapReduce programming model for.NET-based cloud computing, in Proceedings of the 15th International European Parallel Computing Conference (EuroPar 2009), Delft, The Netherlands, August 25_28, 2009, pp. 417_428.
[23]. C. Jin, C. Vecchiola, and R. Buyya, MRPGA: An Extension
[24]. X. Chu, K. Nadiminti, J. Chao, S. Venugopal, and R. Buyya, Aneka: Next- Generation Enterprise Grid Platform for e-Science and e-Business Applications, Proceedings of the 3rd IEEE International Conference and Grid Computing,Bangalore, India, December 10_13, 2007, pp. 151_159.
[25].Manjrasoft Products, http://www.manjrasoft.com/products.html (accessed September
[26]. Skynet, http://skynet.rubyforge.org/
[27].Rinda Doc page, http://www.ruby-doc.org/stdlib/libdoc/rinda/rdoc/index.html
[28].Tuplespace in Wikipedia, http://en.wikipedia.org/wiki/Tuple_space (accessed September 2009).
[29]. GridGain, http://www.gridgain.com/
[30].QT concurrent Page, http://labs.trolltech.com/page/Projects/Threads/QtConcurrent
[31]. C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, Evaluating MapReduce for Multi-core and Multiprocessor Systems, in Proceedings of the 13th International Symposium on High-Performance Computer Architecture (HPCA), Phoenix, AZ, February 2007, pp. 13_24.
[32]. M. Kruijf and K. Sankaralingam, MapReduce for the Cell B. E. Architecture, TR1625, Technical Report, Department of Computer Sciences, the University of Wisconsin—Madison, 2007.
[33]. B. S. He, W. B. Fang, Q. Luo, N. K. Govindaraju, and T. Y. Wang, Mars: A MapReduce framework on graphics processors, in Proceedings of the 17th

International Conference on Parallel Architectures and Compilation Techniques, Toronto, Ontario, Canada, 2008, pp. 260_269.
[34]. H. C. Yang, A. Dasdan, R. L. Hsiao, and D. S. P. Jr, Map-reduce-merge: Simplified relational data processing on large clusters, in Proceedings of SIGMOD, 2007, pp. 1029_1040.
[35]. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, Dryad: Distributed dataparallel programs from sequential building blocks, in Proceedings of the European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March, 2007, pp. 59_72.
[36]. S. Chen, and S. W. Schlosser, Map-Reduce Meets Wider Varieties of Applications, IRP-TR-08-05, Technical Report, Intel Research Pittsburgh, May 2008.
[37]. R. E. Bryant, Data-Intensive Supercomputing: The Case for DISC, CMU-CS-07-128, Technical Report, Department of Computer Science, Carnegie Mellon University, May 2007.
[38]. A. W. McNabb, C. K.Monson, andK. D. Seppi, Parallel PSOUsingMapReduce, in Proceedings of the Congress on Evolutionary Computation, Singapore, 2007, pp. 7_14.
[39]. Presentations by Steve Schlosser and Jimmy Lin at the 2008 Hadoop Summit, http://developer.yahoo.com/hadoop/summit/ (accessed September 2009).