International Journal of Information Technology and Knowledge Management
July-December 2012, Volume 5, No. 2, pp. 239-243

IMPLEMENTING MAP-REDUCE TO ENHANCE THE COMPUTATIONAL ABILITIES OF CLOUD COMPUTING PLATFORM

Shivdev Singh[1], Jimmy Singla[2], Gurdev Singh[3] and Raman Goyal[4]

ABSTRACT: The immense computational and storage power that a cloud infrastructure provides, can be aptly used to calculate large sets of data. Map-Reduce is one technique which can help to build such applications that are capable of executing such tasks. Map-Reduce programs are designed to compute large volumes of data in a parallel fashion. This requires dividing the workload across a large number of machines. The computation takes a set of input key/value pairs and produces a set of output key/value pairs. This paper presents a cloud based application that allows users to upload, large amount of data for computations such as indexing, word counts etc. The application performs calculation using asynchronous Map-Reduce requests and similar results will be calculated using simple asynchronous techniques. The application has been implemented using Google App Engine's Python runtime.

1. INTRODUCTION

Map reduce is a software frame work introduced by Googlein 2004 to support distributed computing on large data sets on clusters of computers. Parts of the framework are patentedin some countries. The framework is inspired by the map and reduce functions commonly used infunctional programming, although their purpose in the Map-Reduce framework is not the same as their original forms. Map-Reduce libraries have been written in C++, C#, Erlang, Java, LabVIEW,OCaml, Perl, Python, PHP, Ruby, F#,Rand other programming languages. The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the Map-Reduce library expresses the computation as two functions: Map and Reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The Map-Reduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function.

The Reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory [1]. Map-Reduce has emerged as a popular way to harness the power of large clusters of computers. Map-Reduce allows programmers to think in a data-centric fashion: they focus on applying transformations to sets of data records, and allow the details of distributed execution, network communication, coordination and fault tolerance to be handled by the Map-Reduce framework. The Map-Reduce model is typically applied to large batch-oriented computations that are concerned primarily with time to job completion. The Google Map-Reduce framework [3] and open-source Hadoop system reinforce this usage model through a batch-processing implementation strategy: the entire output of each map and reduce stage is materialized to stable storage before it can be consumed by the next stage, or produce output. Batch materialization allows for a simple and elegant checkpoint/restart fault tolerance mechanism that is critical in large deployments, which have a high probability of slowdowns or failures at worker nodes [2].

2. RELATED WORK

The topic about cloud computing over Map-Reduce has drawn much more attention by the public for the past years. Many researches about the related issues are conducted to evaluate the implementations, as well as to improve the performance. Li *et al*. [13] built the corresponding non-pre-emptive priorityM/G/1 queuing model for differentiated QoS requirements of cloud computing resources. Then, a system cost function was built by considering cloud computing service providers destination to gain the maximum profits. Based on the queuing model and system cost function, the corresponding strategyand algorithm are given to get the approximate optimistic value of service for each job in the queuing model. The analysis and numeric results show that the approach for job scheduling can not only guarantee the QoS requirements of the users' jobs, but also can make the maximum profits for the cloud computing service providers. Maggiani et al. [11] introduced cloud

[1,3,4] Department of Computer Science and Engineering, LLRIET, Moga, India, [1]*E-mail: shivdevsingh1989@gmail.com*, [3]*gurdev23@gmail.com*, [4]*er_raman2004@yahoo.co.in*

[2] Department of Computer Science and Engineering, NWIET, Dhudike, India, *E-mail: jimmysingla285@gmail.com*

computing clearly in different point of views such as storage, security etc. The components of Software as a Service (SaaS) bring up in this paper. SaaS offers benefits that are simply not available from applications running locally on people's own computer. In cloud computing, people potentially access, from any computer with a browser, to applications that people could never own. For cloud computing, [14] proposed an architecture implemented on InterGrid system which aims to provide an execution environment for running applications on top of the interconnected infrastructure. Experiments show that it can balance load between distributed sites and have validated thata bag-of-tasks application can run on cloud providers by using virtual machines. On the other hand, cloud computing is also applied to the education field, as described in [5]. For example, in University of Westminster (UOW), the concept of cloud computing came out from the fact that 96% of students were setting up their external email accounts due to the outdated campus email service. Hence their received emails were automatically forwarded to their external third party accounts. As a result, the Google Apps (Education Edition) platform became a solution option which can provide the whole campus with a free email service (with a capacity of 7.3GB of disk space for each student), messaging, and shared calendars with no advertising for students or staffs. Besides, a suit of productivity applications (e.g., word processing, spreadsheet, presentation) that support collaboration (i.e., users can share documents remotely) can be potentially useful for students working on group-based assignments by this platform. Jeffrey *et al*. [16] developed an implementation of Map-Reduce that scales to a large cluster comprising thousands of machines which is suitable for use on many of the large computational problems, especially encountered at Google. Overall, the Map-Reduce programming model has been used by Google for different kinds of purposes over recent years. Johnson *et al*. [17] demonstrated that SQL evaluation in a cloud environment. He used a simple library database as the data set and query with SQL (Structure Query Language) clauses, for example select, groupby, having etc. At the same time, he also modified the SQL query as a Map-Reduce programming structure successfully to do the same query. Wei et al. [12] have done a comparison between Hadoop, which is an implementation of Map-Reduce paradigm with a system that was developed earlier by Agrawal's group at OhioState, FREERIDE (Framework for Rapid Implementation of Datamining Engines) by taking three data mining algorithms, which are K - Means Clustering, Apriori Association Mining, and k-Nearest Neighbour search, and a simple data scanning application, Word-Count. Results show that the performance of Hadoop can become better based on the factors like maximum number of concurrent Map per node and number of Reduce. Also, increasing size of dataset can improve the performance of Hadoop. Another Hadoop Map-Reduce implementation was used ina scientific application in [18]. They implemented the storageand analysis phase of an Astrophysics application. Results show that Hadoop enables the in-place analysis of halo data using FOF algorithm, and minimizes the time to move databack and forth between computation and storage resources.Tian *et al.* [19] tried to extract the popular SMS messages in a short period of time based on a Map-Reduce framework torealize the distributed mining. Experimental results show thattheir approach can find the popular messages out at a reasonable cost.

3. MAP-REDUCE MODEL

Computing and data intensive data processing are increasingly prevalent. In the near future, it is expected that the data volumes processed by applications will cross the peta-scale threshold, and increase the computational requirements. Google's Map-Reduce [1] is a programming model and process, resulting in large data sets related to implementation. Map-Reduce is a programming model, it is with the processing production related to implementation of large data sets. Users specify a map function, through the map function handles key/value pairs, and produce a series of intermediate key/value pairs and use the reduce function to combine all of the key values have the same middle button pair The value part. Map-Reduce allows program developers to data-centric way of thinking: a focus on the application data record set conversion, and allows the implementation of the Map-Reduce framework for distributed processing, network communication, coordination and fault-tolerance and other details.

Example: Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code:

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");
reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

The map function emits each word plus an associated count of occurrences (just '1' in this simple example). The reduce function sums together all counts emitted for a particular word.In addition, the user writes code to fill in a *Map-Reduce specification* object with the names of the input and output files, and optional tuning parameters. The user then invokes the Map-Reduce function, passing it the specification object. The user's code is linked together with the Map-Reduce library (implemented in C++) [1]. As popularity of Map-Reduce increases in these recent years, the use of Map-Reduce has become much more common while its abundant APIs enable significantly easier programming [9]. As we know, Map-Reduce is a programming model and an associated implementation for processing large datasets that is amenable to a broad variety of real-world tasks. Users just need to specify the computation in terms of a Map and a Reduce function, and the underlying runtime system automatically parallelizes the computation across large-scaleclusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. [8] Map-Reduce was created by one of the biggest search engine companies - Google for application development on data-centers with thousands of computing nodes. Nowadays, it has been used across a wide range of domains within Google, including [7]

- large-scale machine learning problems
- clustering problems for the Google News and Froogle products
- extracting data to produce reports of popular queries (e.g. Google Zeitgeist and Google Trends)
- extracting properties of web pages for new experiments and products (e.g. extraction of geographical locations from a large corpus of Web pages for localized search)
- processing of satellite imagery data language model processing for statistical machine translation
- large-scale graph computations

Hadoop [10] is one of the implementations for Map-Reduce. Hadoop is an open-source project attempting to reproduceGoogle's implementation, and is hosted as a sub-project of the Apache software foundation's Lucene search engine library. Hadoop has its own distributed file system, named Hadoop Distributed File System (HDFS) which is similar to Google's implementation based on a distributed file system called Google File System (GFS). Besides, Hadoop is easy to be programmed by user because it is implemented in Java, while HDFS is a pure-java file system. HDFS replicates the data on multiple nodes so that a failure of nodes containing a part of the data will likely not impact the entire computation process. Also, data locality is exploited in the Hadoop implementation. However, HDFS cannot be directly mounted on an existing operating system. Data needs to be copied in and out of HDFS before and after executing a job. This can be time-consuming for large data sets[2].

4. NEED FOR MAP-REDUCE OPTIMIZATION

Today, many commercial and scientific applications require the processing of large amounts of data, and thus, demand compute resources far beyond what can be provided by a single commodity processor. To address this, various high-end computing platforms — including supercomputers, graphics processors, clusters of workstations, and cloud computing environments — have been architected to facilitate parallel processing at different scales. However, efficiently exploiting the hardware parallelism provided by these platforms requires parallel programming skills that are difficult to master by common application developers. Even for well-trained experts, the engineering and debugging of parallel applications can be time consuming. Map-Reduce [4] is an effort that seeks to democratize parallel programing. Originally proposed by Google, Map-Reduce has been used to process massive amounts of data in web search engines on a daily basis. With the maturity of Hadoop, an open-source Map-Reduce implementation, the Map-Reduce programming model has been widely adopted and found effective in many scientific and commercial application areas such as machine learning, bioinformatics, astrophysics and cyber-security [5]. Because of its ease of use and built-in fault tolerance, Map-Reduce is also becoming one of the most effective programming models in cloud computing environments, e.g., Amazon EC2. A Map-Reduce job consists of two user-provided functions: map and reduce. As shown in Figure 1, the run-time system creates concurrent instances of map tasks that split and process the input data. The intermediate results generated by map tasks are then copied to the reduce tasks, which in turn reduce the intermediate results and produce the final output. In a Map-Reduce system, all the above data is stored as key/ value pairs for efficient data indexing and partitioning.

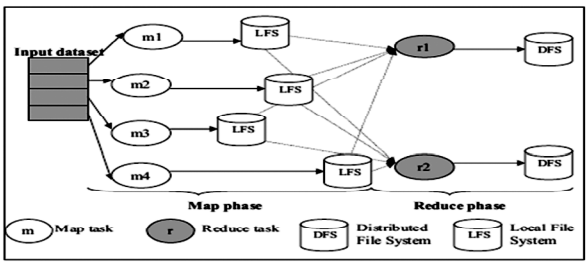

Figure 1: Map-Reduce Framework [6]

Existing Map-Reduce implementations (e.g., Hadoop) enforce a barrier synchronization betwee the map phase and

the reduce phase, i.e., the reduce phase only starts when all map tasks are completed. There are two cases where this barrier synchronization can result in serious resource underutilization. First, on heterogeneous environments, the faster compute nodes will finish their assigned map tasks earlier, but these nodes cannot proceed to the reduce processing until all the map tasks are finished, thus wasting resources. The resource heterogeneity can originate from different hardware configurations or resource sharing through virtualization. Second, even in homogeneous environments, a compute node may not be fully utilized by the map processing because a map task alternates between computation and I/O. Also, there is additional scheduling overhead between the executions of different tasks [6]. In this case, overlapping map and reduce processing can considerably improve job response time, especially when the number of map tasks is relatively large compared to the cluster size. For instance, the number of map tasks of a Map-Reduce job is typically configured to be 100 times the number of compute nodes at Google [4].

5. PIPELINED MAP-REDUCE MODEL

In this section we discuss our extensions to Map-Reduce tosupport pipeline.In general, reduce tasks traditionally issue HTTPrequests to pull their output from each Task-Tracker. This means that map task execution is completely decoupled fromreduce task execution. To support pipelining, we modified the map task to instead push data to reducers as it is produced. To give an intuition for how this works, we begin by describing a straightforward pipelined design, and then discuss the changes we had to make to achieve good performance .In our Pipelined-Map-Reduce implementation, we modified Hadoop to send data directly from map to reduce tasks. When a client submits a new job to Hadoop, the Job-racker assigns the map and reduce tasks associated with the job to the available Task-Tracker slots. For purposes of discussion, we assume that there are enough free slots to assign all the tasks for each job. We modified the Map-Reduce library so that each reduce task contacts every map task upon initiation of the job, and opens a TCP socket which will be used to pipeline the output of the map function. As each map output record is produced, the mapper determines which partition(reduce task) the record should be sent to, and immediately sends it via the appropriate socket. A reduce task accepts the pipelined data it receives from each map task and stores it in an in-memory buffer, spilling sorted runs of the buffer to disk as needed. Once the reduce task learns that every map task has completed, it performs afinal merge of all the sorted runs and applies the user defined reduce function as normal, write the output to the File System.

6. IMPLEMENTATION

In our case we developed an application that accepts an archive file containing some text files for performing the following operations:

- Word Count
- Indexing
- Phrase indexing

The hardware specification of the systems used by Google App Engine are not publically declared, however the performance can be measured against most of the high powered servers used in the cloud infrastructures today.

Figure 2 shows the home page of the application where the user can upload the file and launch any one of the above listed operations on the archived files.

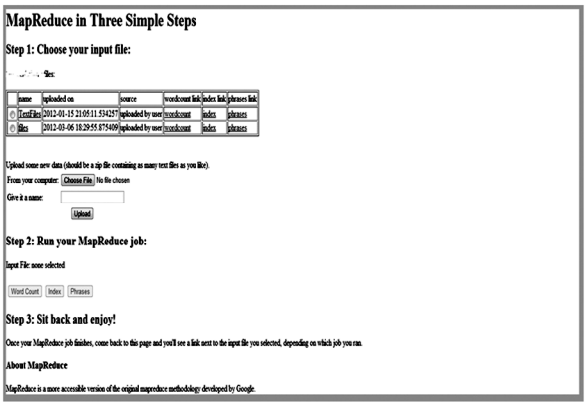

Figure 2: Map-Reduce Application Home Page

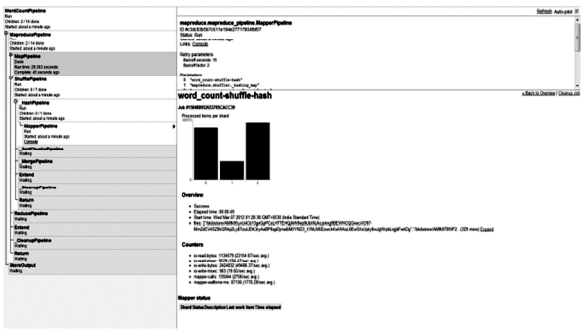

Figure 3: Map-Reduce Job Progress Tracking UI

The next Figure 4 shows the sample output of results generated for word count operation carried out using the application

Similarly the application is able to run simultaneous jobs using Map-Reduce and generate output in a similar fashion.

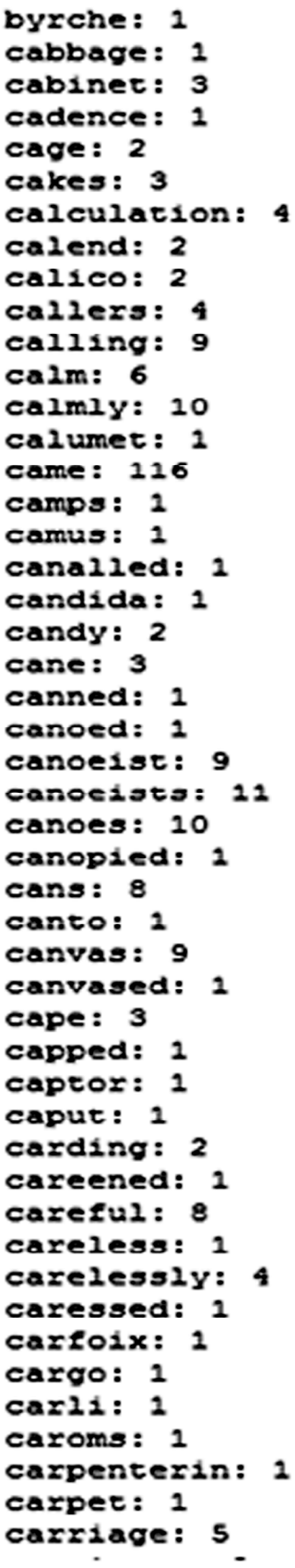

Figure 4: Map-Reduce Job (Word Count) Generated Output

7. CONCLUSION

In this paper we presented an idea about using an enhanced version of Map-Reduce library for data computation tasks and generated results suggest that it performs better than the usual batch mode operation. With help of pipelining and asynchronous requests we are able to process the data in much lesser time and by utilizing lesser resources. The current implementation was performed using Google App Engine, in future the same API can be tested against other cloud platforms and a detailed analysis may be performed.

REFERENCES

[1] "Map-Reduce: Simplified Data Processing on Large Clusters", by Jeffrey Dean and Sanjay Ghemawat; from Google Research

[2] "Map-Reduce Online"; Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein UC Berkeley, Khaled Elmeleegy, Russell Sears Yahoo! Research

[3] Balazinska, M., Balakrishnan, H., Madden, S., and Stonebraker, M. "Fault-tolerance in the Borealis Distributed Stream Processing System". In SIGMOD (2005)".

[4] Jeffrey Dean and Sanjay Ghemawat. Map-Reduce: Simplified Data Processing on Large Clusters. In 6th Symposium on Operating Systems, Design and Implementation, OSDI, 2004.

[5] M. Grant, S. Sehrish, J. Bent, and J. Wang. "Introducing Map-reduce to High End Computing". 3rd Petascale Data Storage Workshop, Nov 2008.

[6] "Enhancing Map-Reduce via Asynchronous Data Processing", Marwa Elteir, Heshan Lin and Wu-chun Feng Department of Computer Science Virginia Tech, 2010 16th International Conference on Parallel and Distributed Systems.

[7] Y. Hung-Chih, D. Ali, H. Ruey-Lung, and D.S. Parker, "Map-Reduce-Merge: Simplified Relational Data Processing On Large Clusters", in Proceedings of the 2007 ACM Sigmod International Conference on Management of Data Beijing", China: acm, 2007.

[8] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing Map-reduce to High End Computing", in Petascale Data Storage Workshop", 2008. pdsw '08. 3rd, 2008, pp. 1-6.

[9] "Comparison of Map-Reduce and SQL on Large-scale Data Processing", Jenq-Shiou Leu et.al National Taiwan University of Science and Technology Taipei, Taiwan.

[10] "Apache, Welcome To Hadoop!", http://hadoop.apache.org, 2009.

[11] R. Maggiani, "Cloud Computing is Changing How We Communicate", in 2009 IEEE International Professional Communication Conference, Ipcc 2009, Waikiki, Hi, United States, 2009.

[12] J. Wei, V.T. Ravi, and G. Agrawal, "Comparing Map-Reduce and Freeride for Data-Intensive Applications", in Cluster Computing and Workshops", 2009. Cluster '09. IEEE International Conference on, 2009, pp. 1-10.

[13] L. LI, "An Optimistic Differentiated Service JobScheduling System for Cloud ComputingService Users and Providers", QINGDAO, China, 2009, pp. 295-299.

[14] C. Alexandre Di, A. Marcos Dias De, and B. Rajkumar, "Harnessing Cloud Technologies Fora Virtualized Distributed ComputingInfrastructure", *IEEE Internet Computing*, **13**, pp. 24-33, 2009.

[15] N. Sultan, "Cloud Computing for Education: A New Dawn?", *International Journal of Information Management*, **30**, pp. 109-116.

[16] D. Jeffrey and G. Sanjay, "Map-Reduce: Simplified Data Processing on Large Clusters", *Commun. ACM*, **51**, pp. 107-113, 2008.

[17] J.L. Johnson, "SQL in the Clouds", *Computing in Science & Engineering*, **11**, pp. 12-28, 2009.

[18] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing Map-Reduceto High End Computing", *in Petascale Data Storage Workshop*, 2008. PDSW '08. 3rd, 2008, pp. 1-6.

[19] X. Tian, "Large-Scale SMS Messages Mining Base donmap-reduce", *In Computational Intelligence and Design*, 2008. ISCID '08 International Symposium on, 2008, pp. 7-12.