Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Evaluation of Stream Log Collection Using HADOOP Distributed File System

N. Ramasubramanian*, Srinivas V.V., Praveen Kumar Yadav
Computer Science and Engineering
National Institute of Technology, Tiruchirappalli, Tamilnadu, India

Abstract— Recently stream logging has been referred to widely by web based and product based companies. Stream logging is one of the most important topic of agenda in business re-engineering. Business re-engineering is done in order to improve the effectiveness and productiveness of a particular product or service. Stream logging is achieved with minimum cost using transaction based model over a distributed environment such as HADOOP distributed file system. HADOOP is a distributed file system that helps to improve performance, scalability and reliability. Here a single master and multiple slave model is employed over HADOOP. The proposed model is based on analytics performed by Google for web pages. Here we present a macroscopic analysis of workload characterized by popularity and arrival process. Though numerous transaction models such as Valor, Ameno have been proposed, this model helps to achieve better utilization and execution time over reduced constrained resource.

Keywords— thread scheduling, multi-core, kernel, BST-tree, block scheduling

I. INTRODUCTION

A. Motivation

In computing, a distributed le system or network le system is any le system that allows access to les from multiple hosts sharing via a computer network [5]. This makes it possible for multiple users on multiple machines to share les and storage resources. Most business applications depend on large amount of data collected [3]. Applications such as marketing, finance, sales etc. which makes use of the Internet or the web pages for selling of their products or services requires analytics of content, and demographics [4].

1. Content analytics - It enables to find the top content in a particular site. It also helps to remove less favourable content.
2. Demographics - It helps us find the type of people using a particular type of service.

B. Evolution of file systems

A file system is a means to organize data [2]. A file system organizes data in an efficient manner and is tuned to the special characteristics of the device. A tight coupling usually exists between the operating system and the le system. File systems have been constantly evolving. The following list shows the evolution.

Disk file systems	Flash file systems
Database file systems	Transactional file systems
Network file systems	Flat file systems
Shared file systems	Distributed file systems

C. Distributed file systems

A distributed le system is a collection of computers containing a separate file systems [6] that appear to the user as a single le system on a single machine. DFS provides location transparency and redundancy to improve data availability in the face of failure or heavy load by allowing shares in multiple different locations to be logically grouped under one folder, or DFS root. There are two ways of implementing DFS on a server namely:

- Standalone DFS namespace allows for a DFS root that exists only on the local computer, and thus does not use Active Directory. A Standalone DFS can only be accessed on the computer on which it is created.
- Domain-based DFS namespace stores the DFS configuration within Active Directory. The DFS namespace root is accessible at domain name/root.

D. HADOOP Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed le system de-signed to run on commodity hardware. It has many similarities with existing distributed le systems. However, the differences from other distributed le systems are

significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to le system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project.

E. Performance aspects in stream log collection

Performance measurement in stream log collection refers to the amount of time needed to perform computation over a distributed le system such as HADOOP. In case of conventional systems, this time consists of a disk-access time and a small amount of CPU-processing time. Whereas in case of a distributed le system such as HADOOP, it is usually the time for remote storage and data computation over remote storage. This includes the time to deliver the request to a peer machine, the time to deliver the response to the client, and for each direction, a CPU overhead of running the communication protocol software. The performance of a distributed le system can be viewed as one dimension of its transparency; to be fully equivalent; it would need to be comparable to that of a local disk.

II. RELATED WORK

In [7] Mandal et al. have proposed a new system which will provide on-demand HADOOP clusters across multiple cloud domains. Open Resource Control Architecture (ORCA) is used in the prototype as a cloud control framework and various provisioning alternative has been examined by evaluating HADOOP benchmarks. The bandwidth for applications on resource topologies kept varying for evaluation.

Weijia Xu et al. have evaluated cost of importing large data in a HADOOP cluster in [8]. A practical evaluation is done of default data importing implementation for HADOOP. For improving the performance direct access of data by Datanodes is proposed during import process.

Seo et al. have proposed pre-fetching and per-shuffling optimization techniques with native HADOOP in [9]. The result shows improvement of 73% while evaluation on Yahoo!Grid. Whereas Mikami et al. in [10] have proposed Gfarm file system instead of HDFS for enhancing performance in HADOOP. The proposed HADOOP-Gfarm plug-in efficiently access file on Gfarm be enabling HADOOP MapReduce.

Grover and Carey have used MapReduce in [11] for sampling massive data set with given predicate. Their model is designed to support incremental job expansion in HADOOP. It supports various policies of job growth rate for both single and multi-user workload.

III. COMPONENT OF HADOOP

HADOOP distributed le system is mainly divided into four divisions namely:

- Name node - It keeps the directory tree of all les in the le system, and tracks where across the cluster the le data is kept.
- Data node - A Data Node stores data in the Hadoop File System.
- Job tracker - The Job Tracker is the service within Hadoop that farms out Map-Reduce tasks to specific nodes in the cluster.
- Task tracker - A Task Tracker is a node in the cluster that accepts tasks

IV. REQUIREMENTS

The entire setup is installed on top of a standalone Ubuntu machine. HADOOP requires java version 1.6 [1] or later for its proper working. HADOOP requires the use of RSA cryptography. This is accomplished with the help of ssh-keygen. The RSA key is generated for every system (i.e. both the master and the slave machines) and the key le is stored on all the other machines, in the location /home/.ssh/authorized keys. This helps, the master and slave communicate with each other without any authentication mechanism.

The setup consists of the front end master node also called as Name node that takes care of the entire working of the HADOOP distributed le system. This master node is connected to 3 slave nodes also called as data nodes. The master node is responsible for allocating jobs to the data nodes. The data nodes perform the operations based on the availability of resources. Finally, the data nodes aggregate the result of the computation of the job and send it to the master node

V. PROPOSED APPROACH

Here, we have stream log collection using HADOOP distributed le system. The stream log collection is performed by collecting data from the browsers history. This is done in order to know the usage statistics of a person. Based on the usage statistics, i.e. the web sites visited and the number of times a particular website has been visited, we can gain better insight into improving the services offered and eliminate redundant content over the internet.

Generally, the history that has to be collected is located in

/home/username/:config/google –chrome/Default/History

The data that is located in the history is in sqlite format. In order to obtain the data in text format, for further processing, we have made use of the commandline sqlite3 database service. The commands for obtaining the history details are given below.

sqlite3 History


```
sqlite >  .mode tabs
sqlite >  .output histoutput
sqlite >  select  *  from urls
sqlite >  .exit
```

Algorithm 1 Stream log collection

Require: History of all the web pages visited
Collect data from History using sqlite3.
Use awk script to segregate data.
Send the collected data to HDFS slaves using namenode.
Find the number of occurrences of a particular web-page.

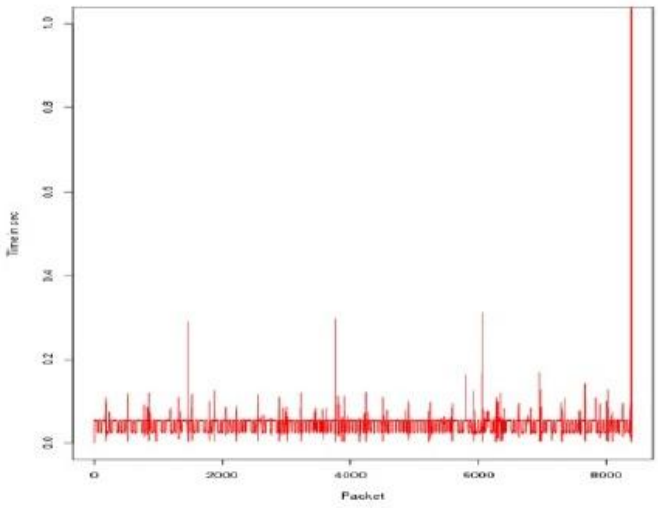

Fig. 1 Packets transmitted for 1 Master and 2 Slave for 1file of data

VI. RESULT OBTAINED

The results in Fig.1 and 2 shows that the number of packets transmitted for 1 le of data over a HDFS le system is much less when compared to the number of packets transmitted for 3 files of data.

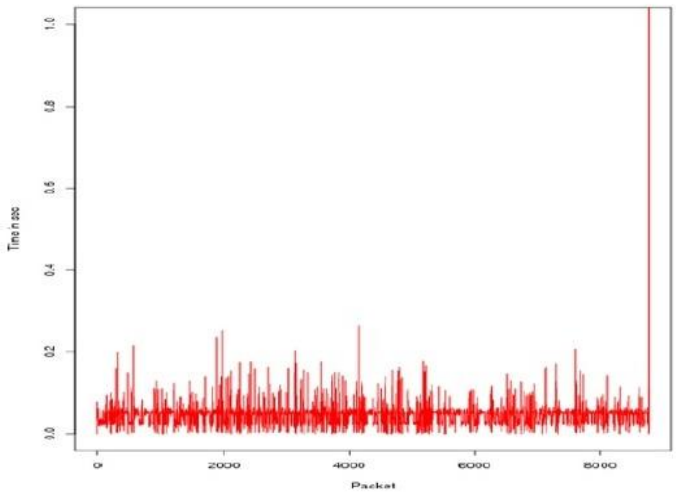

Fig. 2 Packets transmitted for 1 Master and 2 Slave for 3 file of data

Fig. 3 shows that the bandwidth consumed for number of instances 1 is much less compared to the increasing number of instances. Here the number of instances refers to the number of physical computers i.e. masters and slaves connected to each other. Number of instances of 1 refers to the master itself acting as slave. Since the master node acts as slave, the bandwidth consumed is relatively less. In case of a single master connected to multiple slaves, the bandwidth consumed is more.

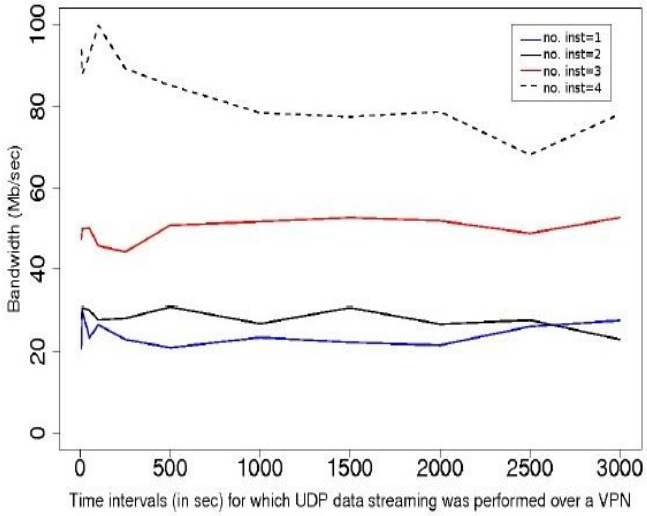

Fig. 3 Bandwidth consumed for number of instances of 1, 2, 3, 4

VII. CONCLUSION AND FUTURE WORK

The stream logging application which is computed over a distributed file system can be used to facilitate applications that involve transaction based computing. Since, the computation is done using normal desktop systems and the number of transactions performed is considerably more, it helps to reduce the number of special hardware (server systems) required for computation. Hence the overall cost of computation is considerably less.

Future work may include, running HDFS on top of a cloud environment such as eucalyptus. Since the cloud environment is distributed and provides server consolidation of resources. Server consolidation of resources is done in order to reduce the cost of the number of hardware resources. The future objective would be to optimize server consolidation for better utilization of resource to enhance the quality of service for the customers.

REFERENCES

[1] D. Borthakur, "*The hadoop distributed file system: Architecture and design*", Hadoop Project Website, 2007.

[2] P.H. Carns, W.B. Ligon III, R.B. Ross, and R. Thakur, "*Pvfs: A parallel file system for linux clusters*", in *Proc.4th Annual Linux Showcase & Confrence*,pp. 28-28, 2000.

[3] R.S. Chin and S.T. Chanson, "*Distributed, object-based programming system*", ACM Computing Survey (CSUR), pp 91-124, 1991.

[4] M. Fisher and A. Sheth, "*Semantic enterprise content management*", Practical Handbook of Internet Computing, pp.1-25, 2004.

[5] J.H. Howard, M. L. Kazar, S. G. Menees, D.A. Nichols, M. Satyanarayanan, R. N. Sidebothem, and M.J. West, *"Scale and performance in a distributed file system"*, ACM Transaction on Computer Systems, pp. 51-81, 1988,

[6] C.A Thekkath, T. Mann and E.K. Lee. Frangipani, "*A scalable distributed file system*", ACM SIGOPS Operating System Review, vol. 31, pp 224-237, 1997

[7] Anirban Mandal, Yufeng Xin, Ilia Baldine, Paul Ruth, Chris Heerman , Jeff Chase, Victor Orlikowski and Aydan Yumerefendi,"*Provisioning and Evaluating Multi-domain Networked Clouds for Hadoop-based Applications*", IEEE 3rd International Conference On Cloude Computing Technology and Science, pp. 690-697, 2011.

[8] Weijia Xu, Wei Luo, Nicholas Woodward, "*Analysis and Optimization of Data Import with Hadoop*" , IEEE 26th International Conference On Parallel and Distributed Processing Symposium Workshop & PhD Forum (IPDPSW), pp. 1058-1066, 2012.

[9] Sangwon Seo, Ingook Jang, Kyungchang Woo, Inkyo Kim, Jin-Soo Kim, Seungryoul Maeng "*HPMR: Prefetching and Pre-shuffling in Shared MapReduce Computation Environment*", Proceedings of 11th IEEE International Conference on Cluster Computing, pp. 1-8, 2009 .

[10] Shunsuke Mikami, Kazuki Ohta, Osamu Tatebe, "*Using the Gfarm File System as a POSIX compatible storage platform for Hadoop MapReduce applications*", IEEE/ACM 12th International Conference on Grid Computing (GRID), pp. 181-189, 2011.

[11] Raman Grover, Michael J. Carey, "*Extending Map-Reduce for Efficient Predicate-Based Sampling*", IEEE 28th International Conference on Data Engineering, pp. 486-497, 2012.

