International Journal for Rapid Research in Engineering Technology & Applied Science
ISSN (Online): 2455-4723
Vol III Issue II February 2017

Analysis of Hadoop Word Count Using Map Reducing Method

[1]Neha Chouhan, [2]Dr. Pankaj Dashore
[*1]Research Scholar, Sanghvi Innovative Academy, Indore, M.P, India
[2]Principal,, Sanghvi Innovative Academy, Indore, M.P, India
*Department of Computer Science & Engineering

Abstract
As a result of the rapid development in cloud computing, it's fundamental to investigate the performance of extraordinary Hadoop MapReduce purposes and to realize the performance bottleneck in a cloud cluster that contributes to higher or diminish performance. It is usually primary to research the underlying hardware in cloud cluster servers to permit the optimization of program and hardware to achieve the highest performance feasible. Hadoop is founded on MapReduce, which is among the most popular programming items for huge knowledge analysis in a parallel computing environment. In this paper, we reward a particular efficiency analysis, characterization, and evaluation of Hadoop MapReduce WordCount utility.

Keywords: Performance analysis, cloud computing, Hadoop WordCount.

1. Introduction
We are residing within the era of large data. In these days a tremendous amount of knowledge is generating everywhere as a result of advances within the web and verbal exchange applied sciences and the pursuits of men and women using smartphones, social media, internet of things, sensor contraptions, online offerings and lots of more. In a similar way, in improvements in knowledge applications and broad distribution of application, a couple of govt and commercial organizations such as monetary institutions, healthcare institution, schooling and research division, power sectors, retail sectors, lifestyles sciences and environmental departments are all producing a enormous amount of information everyday. For examples, international data enterprise (IDC) said that 2.8 ZB (zettabytes) knowledge of universe had been saved in the year of 2012 and this may reach up to forty ZB through 2020 [1]. In a similar fashion Facebook processes round 500 TB (terabytes) knowledge per day [2] and Twitter generates eight TB data daily [3]. The huge datasets no longer handiest comprise structured form of knowledge but greater than seventy five% of the dataset includes uncooked, semi-structured and unstructured type of data [4]. This large quantity of information with one of a kind codecs can be viewed as giant information.The derivation of big knowledge is indistinct and there are a lot of definitions on huge data. For examples, Matt Aslett outlined massive knowledge as "tremendous data is now virtually universally understood to refer to the recognition of larger business intelligence through storing, processing, and examining data that was previously ignored because of problem of normal data management applied sciences" [5]. Recently, the term of giant data has got a brilliant momentum from governments, industry and research communities. In [6], significant information is outlined as a term that encompasses using tactics to capture, approach, analyze and visualize potentially significant datasets in a cheap timeframe now not obtainable to usual IT applied sciences.

II. LITERATURE SURVEY

[1] In this paper, author proposed a structure that integrates search-headquartered clusters and semantic media wiki by using relaxation APIs to help the exploration of cloud monitoring data. It utilizes the spark's MapReduce paradigm to establish the cluster in the dataset utilizing k-way clustering approach.

[2] In this paper, the writer offered a distinct performance analysis and analysis for Hadoop WordCount workload utilizing different processors similar to Intel's ATOM D525, Xeon X5690, and AMD's Bobcat E350. The estimation model common error was not up to 5% compared to a measured knowledge line.

[3] In this paper, author recommends Dache, a knowledge-conscious cache framework for big-data functions. A novel cache description scheme and a cache request and reply protocol are designed. We enforce Dache by means of extending Hadoop. Test bed experiment results show that Dache tremendously improves the completion time of MapReduce jobs.

[4] In this work, author presents a novel efficiency analysis framework for answering this question. We observe that the execution of every map (lessen) duties consists of distinctive, good-defined knowledge processing phases. Handiest map and scale back services are customized and their executions are consumer-outlined for exclusive MapReduce jobs.

III. EXISTING SYSTEM

ESTIMATION MODEL USING Amdahl's LAW METHOD

Based on Amdahl's law definition we discussed before, the performance of a given processor can be divided into two parts, the part which increases with the performance improvement and is said to scale is defined as variable *a,* and the other part which does not improve due to the performance enhancement and is said to not scale is defined as variable *b.* The *a* and *b* variables can be derived using the basic definition of Amdahl's law which can be written in the form of:

$$T = T_o + (T_1 - T_o) \times \frac{I_1}{I} \qquad (1)$$

where T1 is the measured execution time at a given input size I1, and To be the non-scale execution time. We can write To in terms of a second measurement T2 at I2:

$$T_o = \frac{T \times I_2 - T_1 \times I_1}{(I_2 - I_1)} \qquad (2)$$

When we substitute Eq(2) for To in Eq(1) we obtain Amdahl's law in terms of two speci_c measurements without reference to:

$$T = a + b \times \frac{1}{I} \qquad (3)$$

$$a = \frac{I_2 \times T_2 - I_1 \times T_1}{(I_2 - I_1)} \qquad (4)$$

$$b = \frac{(I_1 \times I_2)(T_2 - T_1)}{(I_2 - I_1)} \qquad (5)$$

The variables *a* and *b* can be transformed to the performance instead of the time domain by using P D 1/T. This will give us *a* and *b* variables in terms of performance and input size as shown in Eq(6) and Eq(7).

$$a = \frac{P_1 \times I_2 - P_2 \times I_1}{(P_1 \times P_2)(I_2 - I_1)} \qquad (6)$$

$$b = \frac{(I_1 \times I_2)(P_2 - P_1)}{(P_1 \times P_2)(I_2 - I_1)} \qquad (7)$$

V. PROPOSED WORK

Word count is typical examples where Hadoop map reduce developers start their hands on. This sample map reduce is intended to count the no of occurrences of each word in the provided input files.

The word count operation takes place in two stages a mapper phase and a reducer phase. In mapper phase first the test is tokenized into words then we form a key value pair with these words where the key being the word itself and value '1'. For example consider the sentence "tring tring the phone rings"

In map phase the sentence would be split as words and form the initial key value pair as
<tring,1>
<tring,1>
<the,1>
<phone,1>
<rings,1>

In the reduce phase the keys are grouped together and the values for similar keys are added. So here there are only one pair of similar keys 'tring' the values for these keys would be added so the out put key value pairs would be
<tring,2>
<the,1>
<phone,1>
<rings,1>
This would give the number of occurrence of each word in the input. Thus reduce forms an aggregation phase for keys.

The point to be noted here is that first the mapper class executes completely on the entire data set splitting the words and forming the initial key value pairs. Only after this entire process is completed the reducer starts. Say if we have a total of 10 lines in our input files combined together, first the 10 lines are tokenized and key value pairs are formed in parallel, only after this the aggregation/ reducer would start its operation.

The figure below would throw more light to your understanding

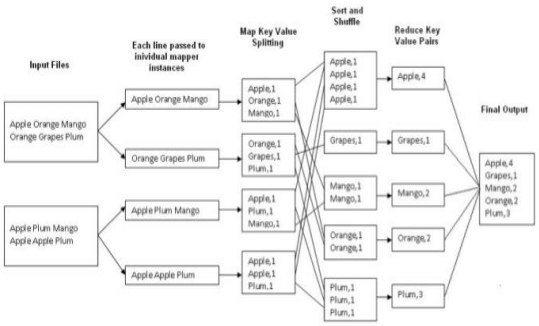

Figure 1. Proposed System using Map Reduce

VI. RESULT ANALYSIS

Existing and proposed system implemented on Ubuntu 14.10 Server edition. First install and configure jdk1.8 on machine. After that install Hadoop 2.7 and configure it. NetBeans 8.0 used as editor and creates Graphical User Interface for project. Compare existing and proposed on the basis of computation time. Below figures show GUI and comparison between both systems.

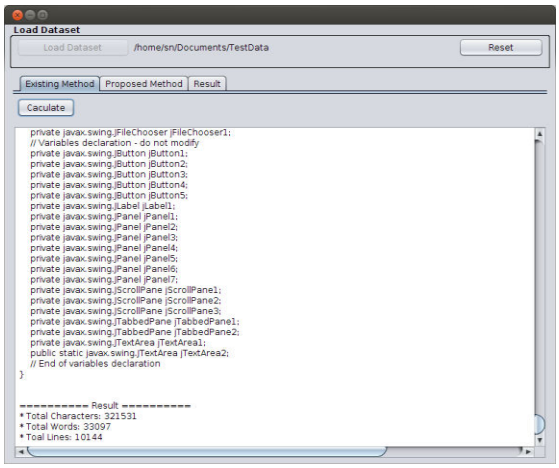

Figure 2. Word Count problem using Simple Java Code

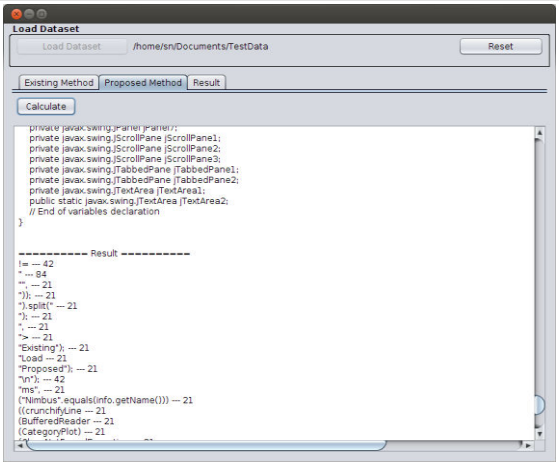

Figure 3. Word Count problem using Map Reduce

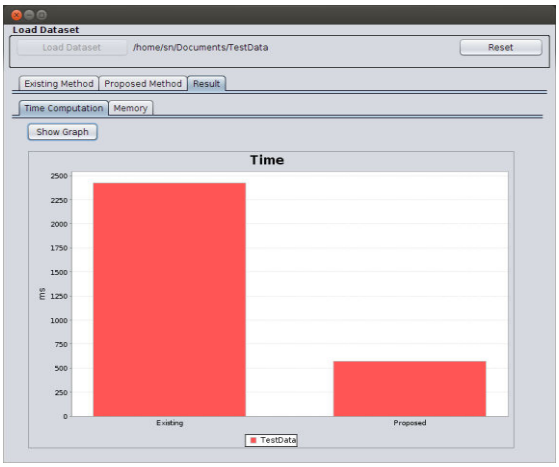

Figure 4. Computation time chart for existing and proposed system.

V. CONCLUSION

Map-Reduce has become an important platform for a variety of data processing applications. Word Count Mechanisms in Map-Reduce frameworks such as Hadoop, suffer from performance degradations in the presence of faults. Word Count Map-Reduce, proposed in this paper provides an online, on-demand and closed-loop solution to managing these faults. The control loop in word count mitigates performance penalties through early detection of anomalous conditions on slave nodes. Anomaly detection is performed through a novel sparse-coding based method that achieves high true positive and true negative rates and

can be trained using only normal class (or anomaly-free) data. The local, decentralized nature of the sparse-coding models ensures minimal computational overhead and enables usage in both homogeneous and heterogeneous Map-Reduce environments.

VII. REFERENCES

[1] Samneet Singh and Yan Liu,"A Cloud Service Architecture for Analyzing Big Monitoring Data",ISSNll1007-0214ll05/10llpp55-70 Volume 21, Number 1, February 2016

[2] JOSEPH A. ISSA, "Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount", Received November 19, 2015, accepted December 12, 2015, date of publication December 18, 2015, date of current version December 29, 2015.

[3] Yaxiong Zhao, Jie Wu, and Cong Liu, "Dache: A Data Aware Caching for Big-Data Applications Using the MapReduce Framework",ISSNll10070214ll05/10llpp39-50 Volume 19, Number 1, February 2014

[4] Zhuoyao Zhang Ludmila Cherkasova, "Benchmarking Approach for Designing a MapReduce Performance Model"**,** *ICPE'13,* April 21-24, 2013

[5] Nikzad Babaii Rizvandi, Albert Y. Zomaya , Ali Javadzadeh Boloori, Javid Taheri1, "On Modeling Dependency between MapReduce Configuration Parameters and Total Execution Time", 2012

[6] Nikzad Babaii Rizvandi, Javid Taheri1, Reza Moraveji, Albert Y. Zomaya, "On Modelling and Prediction of Total CPU Usage for Applications in MapReduce Enviornments", 2011

[7] A. Baratloo, M. Karaul, Z. Kedem, and P.Wyckoff, ``Charlotte: Meta computing on theWeb," in *Proc. 9th Int. Conf. Parallel Distrib. Comput. Syst.*, 1996, pp. 1_13.

[8] J. Bent, D. Thain, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, ``Explicit control in the batch-aware distributed _le system," in *Proc. 1st USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Mar. 2004, pp. 365_378.

[9] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier,``Cluster-based scalable network services," in *Proc. 16th ACMSymp. Oper. Syst. Principles*, Saint-Malo, France, 1997, pp. 78_91.

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung, ``The Google _le system," in *Proc. 19th Symp. Oper. Syst. Principles*, New York, NY, USA, 2003, pp. 29_43.

[11] S. Ibrahim, H. Jin, L. Lu, L. Qi, S.Wu, and X. Shi, ``Evaluating MapReduce on virtual machines: The Hadoop case," in *Proc. Int. Conf. Cloud Comput.*, vol. 5931. 2009, pp. 519_528.

[12] J. Issa and S. Figueira, ``Graphics performance analysis using Amdahl's law: IEEE/SCS SPECTS," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst.*, Ottawa, ON, Canada, 2010, pp. 127_232.