Implementation of Word Count Problem with Map Reduce using Hadoop

Rahul Sharma, Prof. O.P. Sharma
Department of Computer Science Alpine Institute of Technology, Ujjain (MP)
er.rahulsharma005@gmail.com, opsharma.28@gmail.com

ABSTRACT:

In cloud computing, data is moved to a remotely located cloud server. Cloud server faithfully stores the data and return back to the owner whenever needed. Data and computation integrity and security are major concerns for users of cloud computing facilities. Today's clouds typically place centralized, universal trust in all the cloud's nodes.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.

I. INTRODUCTION

Yesteryear decade features seen your rise regarding cloud calculating [1], an arrangement where businesses in addition to individual users utilize hardware, storage space, and software program of 3rd party companies named cloud providers rather than running their very own computing commercial infrastructure. Cloud calculating offers customers the illusion of needing infinite calculating resources, of which they can use all the or less than they have to have, without being forced to concern themselves with exactly how those resources are offered or maintained [2].

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. The figure below would throw more light to your understanding.

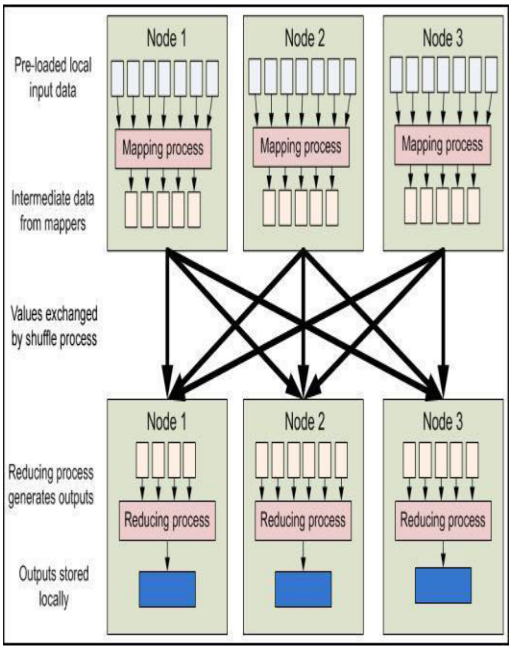


Figure 1. Flow of Map Reduce

II. Map Reduce Problem

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. Below line show about Map Reduce Problem.

- **Map()**
  - Process a key/value pair to generate intermediate key/value pairs
- **Reduce()**

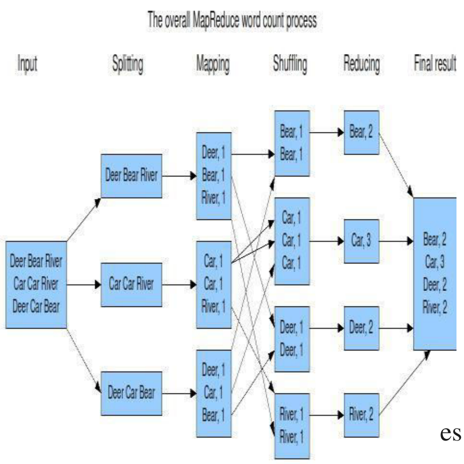
o Merge all intermediate values associated with the same key

Users implement interface of two primary methods:

- **Map:(key1,val1)→(key2,val2)**
- **Reduce:(key2,[val2])→[val3]**
- Map - clause group-by (for Key) of an aggregate function of SQL
- Reduce - aggregate function (e.g., average) that is computed over all the rows with the same group-by attribute (key).

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.

III WORD COUNT PROBLEM WITH MAP REDUCES.



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"tringtring 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.

Algorithm for Word Count using Map-Reduce

```
Mapper<LongWritable,Text,Text,IntWritable> {
private static final IntWritable one = new
IntWritable(1);

private Text word  = new Text();
public static void map(LongWritable key, Text
value, OutputCollector<Text,IntWritable> output,
Reporter reporter) throws IOException {
    String line = value.toString();
StringTokenizer = new StringTokenizer(line);
while(tokenizer.hasNext()) {
word.set(tokenizer.nextToken());
output.collect(word,one);
         }
       }
  }
```

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.

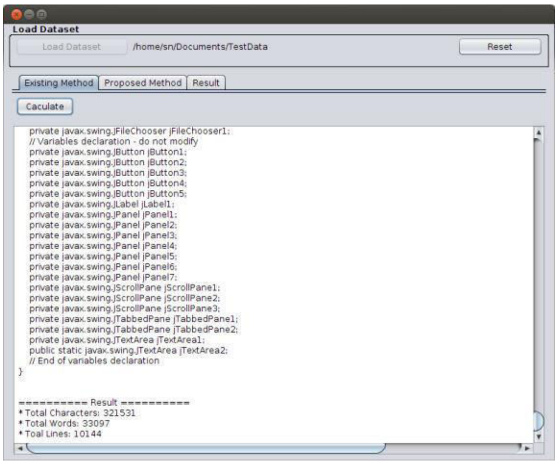


Figure 2. Word Count problem using Simple Java Code

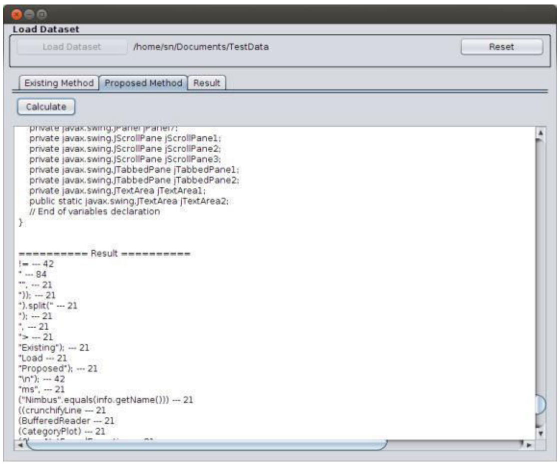


Figure 3. Word Count problem using Map Reduce

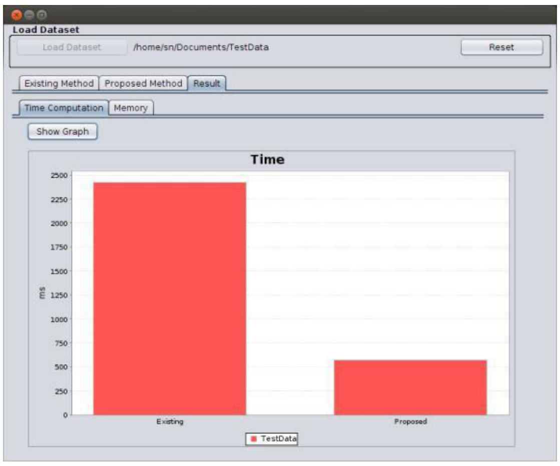


Figure 4. Computation time chart for existing and proposed system.

V. CONCLUSION

Map-Reduce, 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.

VI. REFERENCES

[1] Samneet Singh and Yan Liu,"A Cloud Service Architecture for Analyzing Big Monitoring Data",ISSNl11007-0214ll05/10llpp55-70 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 LudmilaCherkasova, "Benchmarking Approach for Designing a MapReduce Performance Model", ICPE'13, April 21-24, 2013

[5] NikzadBabaiiRizvandi, Albert Y. Zomaya , Ali JavadzadehBoloori, Javid Taheri1, "On Modeling Dependency between MapReduce Configuration Parameters and Total Execution Time", 2012

[6] NikzadBabaiiRizvandi, 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.