International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

Minimizing Time Span of Big Data Analytics using Hadoop - Map Reduce

D. Christy Sujatha[1]
[1] Department Of Software Engineering, Periyar Maniammai University

D. Selvam[2]
Department Of Computer Science and Engineering , Periyar Maniammai University

A. B. Karthick Anand Babu[3]
Thanjavur, Tamilnadu ,India
Research Scholar, Bharathidasan University, Trichy

Abstract--**Private and public clouds offer a new delivery model with virtually unlimited computing and storage resources. An increasing number of companies are exploiting the MapReduce paradigm and its open-source implementation Hadoop as a platform choice for efficient Big Data processing and advanced analytics over unstructured information. This new style of large data processing enables businesses to extract information and discover novel data insights in a nontraditional and game-changing way. For many companies, their core business depends on a timely analysis and processing of large quantities of new data. The data analysis applications might be of different complexities, resource needs, and data delivery deadlines. This diversity creates competing requirements for program design, job scheduling, and workload management policies in MapReduce environments. In this Paper, Hadoop MapReduce to perform word count is implemented. A *map* function is specified to count the number of words in the distributed nodes that produces intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. We have also conducted a Simulation based estimation. The result obtained from the simulation shows that the wordcount using Map Reduce consumes less amount of time when compared with the result obtained without using map reduce programming .**

Keywords--Map reduce , Work load management policy, data delivery dead lines .

1.INTRODUCTION

Big data refers to large datasets that are challenging to store, search, share, visualize, and analyze. At first glance, the orders of magnitude outstrip conventional data processing and the largest of data warehouses. For example, an airline jet collects 10 terabytes of sensor data for every 30 minutes of flying time. Compare that with conventional high performance computing where New York Stock Exchange collects 1 terabyte of structured trading data per day. Compare again to a conventional structured corporate data warehouse that is sized in terabytes and petabytes. Big Data is sized in peta-, exa-, and soon perhaps, zetta-bytes. And, it's not just about volume, the approach to analysis contends with data content and structure that cannot be anticipated or predicted. These analytics and the science behind them filter low value or low-density data to reveal high value or high-density data. As a result, new and often proprietary analytical techniques are required. Big Data has a broad array of interesting architecture challenges. Big data is big news and so too is analytics on big data. Technologies for analyzing big data are evolving rapidly and there is significant interest in new analytic approaches such as Hadoop MapReduce and Hive, and MapReduce extensions to existing relational DBMSs. Over the past five years, the authors and many others at Google have implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues. As a reaction to this complexity, it is proposed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library.

. In this Paper, Hadoop MapReduce is applied which is a programming model and an associated implementation for processing and generating large data sets. A *map* function is specified to count the number of words in the distributed nodes that produces intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. We have also conducted a Simulation based estimation. The result obtained from the Simulation shows that the wordcount using Map Reduce consumes less amount of time when compared with the result obtained without using map reduce programming . The paper is organized as follows :

Section II denotes the related work focusing on Hadoop Map Reduce Programming . Section III denotes Hadoop Map reduce Architecture . Section IV explains the experimental setup and the Result discussion . Section V gives the conclusion.

2. RELATED WORK

Alberto Abell´o et. all [1] proposed, from the beginning of computerized data management, the possibility of using computers in data analysis has been evident for companies. However, first analysis tools needed the involvement of the IT department to help decision makers to query data [1] . They were not interactive at all and demanded specific knowledge in computer science. By the mid 1980's, executive information systems appeared introducing new graphical, keyboard-free interfaces (like touch screens). However, executives were still tied to IT professionals for the definition of ad hoc queries, and prices of software and hardware requirements where prohibitive for small companies.

In [2] Stavros Harizopoulos et. all , proposed Modern general purpose online transaction processing (OLTP) database systems include a standard suite of features: a collection of on-disk data structures for table storage, including heap files and B-trees, support for multiple concurrent queries via locking- based concurrency control, log based recovery, and an efficient buffer manager. These features were developed to support transaction processing in the 1970's and 1980's ,when an OLTP data base was many times larger than the main memory, and when the computers that run these database cost hundreds of thousands to millions of dollars Today, the situation is quite different.

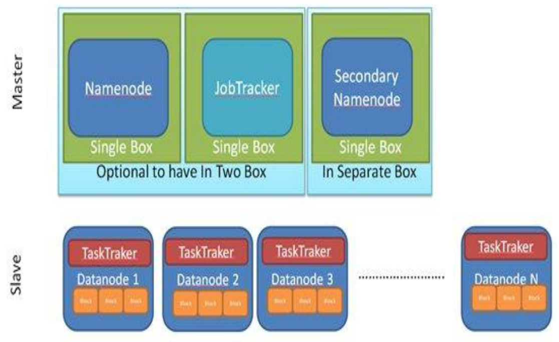
Thus, it was in 1993 that Codd et al., in [3], coined the term OLAP. In that report, the authors defined 12 rules for a tool to be considered OLAP. These rules caused heated controversy, and they did not succeed as Codd's counterpart for Relational Database Management Systems (RDBMS). But, the name OLAP became very popular and broadly used. OLAP is used to extract knowledge from the data warehouse. Another kind of tool used with this purpose are data mining tools (see Data Mining definitional entry). Till now, both research communities have been evolving separately. The former must be interactive, while the latter presents computational complexity problems. However, it seems promising to integrate both kinds of tools so that ones can benefit from the others. In fact, it was already suggested in [4], and some tools like Microsoft Analysis Services already integrate them in some way. Nevertheless, there is much work to do in this field, yet.

In [5] Danica Porobic et. all proposed Legacy multisocket machines, which gained popularity in the 1995s as symmetric multiprocessing servers, had non-uniform memory access (NUMA) latencies. As proposed by Nigel Pendse in [6], OLAP tools should pass the FASMI (Fast Analysis of Shared Multidimensional Information) test. Thus, they should be fast enough to allow interactive queries they should help analysis task by providing flexibility in the usage of statistical tools and what-if studies; they should provide security (both in the sense of confidentiality and integrity) mechanisms to allow sharing data; they should provide a multidimensional view so that the data cube metaphor can be used by users; and, finally, they should also be able to manage large volumes of data (gigabytes can be considered a lower bound for volumes of data in decision support) and metadata.

In [7] Xuepeng Yin et all. Proposed XML format for OLAP system, which overcomes the problem of complexity in integrating fast changing data, physically into a cube which is complex and time-consuming. Processing this data efficiently requires fundamentally new designs for data management systems. First, many database workloads, especially those involving online transaction processing (OLTP) can now fit entirely in main memory. These are the types of databases that run most websites, banks, and other organizations, containing a small set of records for each user or customer. Typical workloads involve many concurrent reads of records and only a few concurrent writes at a time as users buy products, transfer funds, send emails, or perform other operations.

To address these issues, several new database designs have been proposed for main memory OLTP workloads. The most common design involves some form of data partitioning, where each core is responsible for a different subset of the data. Such approaches work well when data partitions cleanly, because each core essentially operates on a separate logical database, resulting in good cache locality and low levels of contention for memory between cores. However, as core counts grow, more and more data partitions are needed when using this approach. As partition become finer-grained, it becomes increasingly hard to partition the database to ensure that each transaction accesses only one partition. The resulting multi-partition transactions require latching entire partitions for access to the records of other cores, and have adverse effects on cache locality. Furthermore, many databases schemas are not trivially partitionable (such as social networks), and state-of-the-art partitioning techniques are very workload dependent [8].

3. SYSTEM ARHITECTURE



3.1 Apache Hadoop

The framework Apache Hadoop is used for distributed processing of huge data sets known as ―Big Data‖ across clusters of computers using a simple programming model . Hadoop permits an application to map, group, and reduce data across a distributed cloud of machines so that applications can process huge data . It can scale up to large number of machines as required for the job; each machine will provide local computation and storage. Apache Hadoop software library itself detects and handles any failures at application layer.

Hadoop Distributed File System - HDFS

A distributed user-level filesystem HDFS Hadoop Distributed File System written in Java stores huge files across machines in a large cluster. Hadoop DFS stores each file as a sequence of blocks, all blocks in a file except the last block are the same size typically 64 MB . Blocks belonging to a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are "write once" and have strictly one writer at any time .

Name-Node

The Name-Node executes file system namespace operations like opening, closing, and renaming files and directories. The Name-Node does not store HDFS data itself, but rather maintains a mapping between HDFS file name, a list of blocks in the file, and the Data Node on which those blocks are stored. The Name-Node makes all decisions regarding replication of blocks.

Secondary Name-Node

HDFS includes a Secondary Name-Node, there is a misconception that secondary Name-Node comes into action after Primary Name-Node (i.e Name-Node) fails. The fact is Secondary Name-Node is continuously connected with Primary Name-Node and takes snapshots of Name-Node's memory structures. These snapshots can be used to recover the failed Name-Node and recent memory structure

Data-Node

A Data-Node stores data in the Hadoop File System. A functional filesystem has more than one Data-Node, with data replicated across them. On startup, a Data-Node connects to the Name-Node; spinning until that service comes up. It then responds to requests from the Name-Node for filesystem operations. Client applications can talk directly to a Data-Node, once the Name-Node has provided the location of the data .

Job-Tracker

Job-Tracker keeps track of which Map-Reduce jobs are executing, schedules individual Maps, Reduces or intermediate merging operations to specific machines, monitors the success and failures of these individual Tasks, and works to complete the entire batch job. The Job-Tracker is a point of failure for the Hadoop Map-Reduce service. If it goes down, all running jobs are halted.

Task-Tracker

A Task-Tracker is a node in the Hadoop cluster that accepts tasks such as Map, Reduce and Shuffle operations from a Job-Tracker. Task-Tracker is set up with set of slots which depicts the number of tasks it can accept. The Task-Tracker spawns a separate JVM processes to do the actual work. The Task-Tracker supervises these spawned processes, capturing the output and exit codes. When the process finishes, successfully or not, the task tracker notifies the Job-Tracker. The Task-Trackers also transmit heartbeat messages to the Job-Tracker, usually every few minutes, to reassure the Job-Tracker that it is still alive. These messages also inform the Job-Tracker of the number of available slots, so the Job-Tracker can stay up to date with where in the cluster work can be assigned .

3.2 Mapping Process

When the user program calls the MapReduce function, the following sequence of actions will be followed .

- The MapReduce library in the user program first splits the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece (controllable by the user via an optional parameter). It then starts up many copies of the program on a cluster of machines.
- One of the copies of the program is special . the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
- A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user defined *Map* function. The intermediate key/value pairs produced by the *Map* function are buffered in memory.
- Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.
- When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same

reduce task. If the amount of intermediate data is too large to fit in memory, an external sort is used.

3.3 Reducing Process

- The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's *Reduce* function. The output of the *Reduce* function is appended to a final output file for this reduce partition.

- When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code. After successful completion, the output of the mapreduce execution is available in the R output files (one per reduce task, with file names as specified by the user). Typically, users do not need to combine these R output files into one file . they often pass these files as input to another MapReduce call, or use them from another distributed application that is able to deal with input that is partitioned into multiple files.

3.4 Word Count Application

The word count that simply reads an input text file containing number of words and count the number of times that each word appears. As Hadoop is build on HDFS and MapReduce, this example of word count will execute as a part of Hadoop application. The input files are needed to be put in HDFS. In some cases, this requires first to format a file system to HDFS. After the system is formatted, the input dictionary files are put into filesystem. Hadoop gives better performance with single larger files rather than smaller files. Short files should be copied to HDFS. This data will be then processed using MapReduce program. The program is Java file that contains Map and reduce algorithms. Here each mapper takes a line as input and breaks it into words. Each mapper then emits <key/value> pairs of word where each emitted<key/value> pairs are then "shuffled" that indicated that the pairs with the same key are grouped and then passed to machine to be reduced by reduce function. For word count example, to count the number of word occurrence, the reduce function will perform the aggregate (sum) of the values of the collection of <key, value> pairs which have the same key.

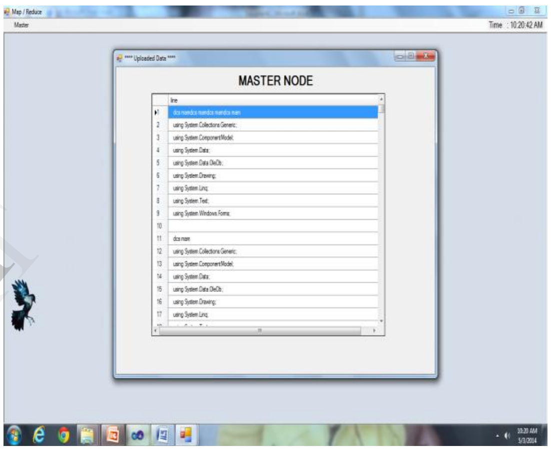
4. SIMULATION MODEL

4.1 SYSTEM REQUIREMENTS

The simulation was applied to the node with dual processor , X86 , memory with 2 to 4 GB per machine and the Operating System used was LINUX. The simulator has been developed to model the implementation of map reduce operation which generates one master node and n number of slave nodes to perform the word count .

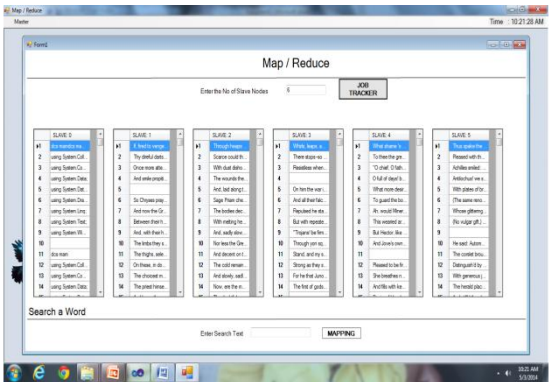
4.2 EXPERIMENTAL RESULT AND DISCUSSION

The simulation has one master node and N number of slave nodes . The number of slave nodes depends on the size of the input data which are created by the job tracker .The MapReduce library in the program first splits the input files into N pieces of typically 64 megabytes (MB) per piece . It then starts up many copies of the program on the slave nodes. The slave node who is assigned a map task reads the contents of the corresponding input split and counts the word occurrence. The intermediate key/value pairs produced by the *Map* function are buffered in memory. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce slave nodes. When a reduce slave node is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers.

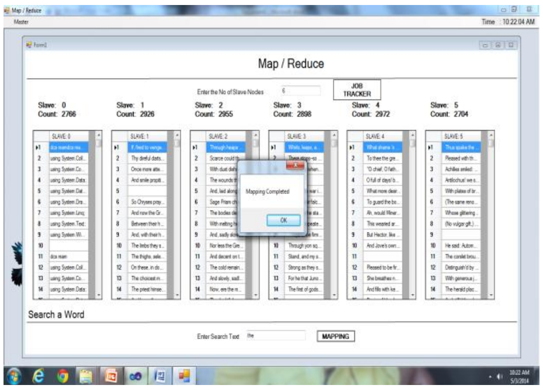


	line
1	dcs mamdcs mamdcs mamdcs mam
2	using System.Collections.Generic;
3	using System.ComponentModel;
4	using System.Data;
5	using System.Data.OleDb;
6	using System.Drawing;
7	using System.Linq;
8	using System.Text;
9	using System.Windows.Forms;
10	
11	dcs mam
12	using System.Collections.Generic;
13	using System.ComponentModel;
14	using System.Data;
15	using System.Data.OleDb;
16	using System.Drawing;
17	using System.Linq;

When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same reduce task. The reduce function will perform the aggregate (sum) of the values of the collection of <key, value> pairs which have the same key.



	SLAVE: 0	SLAVE: 1	SLAVE: 2	SLAVE: 3	SLAVE: 4	SLAVE: 5
1	dcs mamdcs ma...	If, fired to venge...	Through heaps...	Whate, leaps, a...	What shame 's...	Thus spoke the...
2	using System.Col...	Thy direful darts...	Scarce could th...	There stops-so...	To thee the gre...	Pleased with th...
3	using System.Co...	Once more atte...	With dust disho...	Restless when...	"O chief, O fath...	Achilles smiled:...
4	using System.Data;	And smile propiti...	The wounds the...		O full of days! b...	Antilochus! we s...
5	using System.Dat...		And, laid along t...	On him the war i...	What more desir...	With plates of br...
6	using System.Dra...	So Chryses pray...	Sage Priam che...	And all their falc...	To guard the bo...	(The same reno...
7	using System.Linq;	And now the Gr...	The bodies dec...	Repulsed he sta...	Ah, would Miner...	Whose glittering...
8	using System.Text;	Between their h...	With melting he...	But with repeate...	This wearied ar...	(No vulgar gift.)...
9	using System.Wi...	And, with their h...	And, sadly slow...	"Trojans! be firm...	But Hector, like...	
10		The limbs they s...	Nor less the Gre...	Through yon sq...	And Jove's own...	He said: Autom...
11	dcs mam	The thighs, sele...	And decent on t...	Stand, and my s...		The corslet brou...
12	using System.Col...	On these, in do...	The cold remain...	Strong as they s...	Pleased to be fir...	Distinguish'd by...
13	using System.Co...	The choicest m...	And slowly, sadl...	For he that Juno...	She breathes n...	With generous j...
14	using System.Data;	The priest himse...	Now, ere the m...	The first of gods...	And fills with ke...	The herald plac...


Map / Reduce — Master — Time : 10:22:04 AM

Map / Reduce

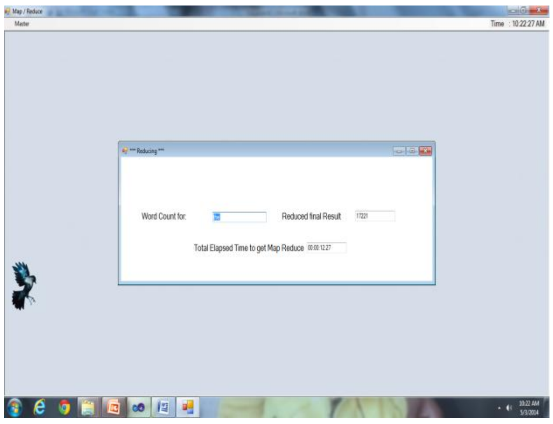
Enter the No of Slave Nodes: 6 — JOB TRACKER

Slave: 0 Count: 2766 — Slave: 1 Count: 2926 — Slave: 2 Count: 2955 — Slave: 3 Count: 2898 — Slave: 4 Count: 2972 — Slave: 5 Count: 2704

Mapping Completed — OK

Search a Word

Enter Search Text: the — MAPPING


Map / Reduce — Master — Time : 10:22:27 AM

*** Reducing ***

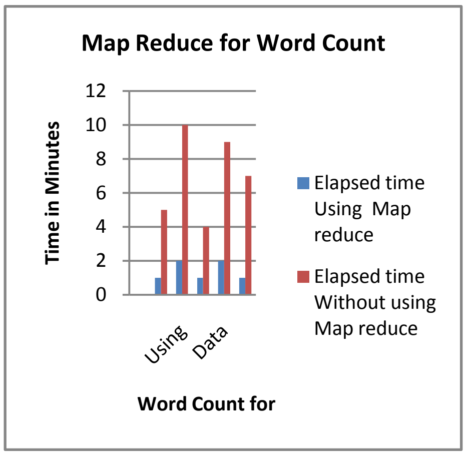
Word Count for: the — Reduced final Result: 17221

Total Elapsed Time to get Map Reduce: 00:00:12.27

The simulation is executed using map reduce algorithm to count the given word and the total elapsed time is calculated . Total elapsed time is calculated for the same word without using map reduce algorithm . The estimated values are plotted in the graph .On the X-axis we have plotted the duration in minutes and on the Y-axis we have plotted the word to be counted . Table 1 shows the time taken to count the words like The, Using ,System, Data & And. The above word occurrences are performed with using map reduce algorithm and without using map reduce algorithm. It is found that the total time taken using mapreduce consumes less amount of time than without using map reduce .

Table 1

Word count for	Elapsed time Using Map reduce (Minutes)	Elapsed time Without using Map reduce (Minutes)
The	1	5
Using	2	10
System	1	4
Data	2	9
And	1	7



5. CONCLUSION

Hadoop MapReduce is an effective programming model and software framework which is used for writing data intensive applications that speedily process vast amounts of data in parallel on low-level compute nodes of clusters. In this paper a *map* function is specified to count the number of words in the distributed nodes that produces intermediate key/value pairs, and a *reduce* function is specified to merge all intermediate values associated with the same intermediate key. A simulation based estimation is conducted to find the time taken to count the number of similar word occurrence .The result obtained from the simulation shows that the wordcount using Map Reduce consumes less amount of time when compared with the result obtained without using map reduce programming .

REFERENCES

[1] Alberto Abell´o and Oscar Romero, UniversitatPolit`ecnica de Catalunya “ON-LINE ANALYTICAL PROCESSING”
[2] Stavros Harizopoulos et al. “ OLTP Through the Looking Glass, and What We Found There” SIGMOD’08, June 9–12, 2008, Vancouver, BC, Canada. Copyright 2008 ACM 978-1-60558-102
[3] E.F. Codd, S.B. Codd and C.T. Salley ,”Providing OLAP to User-Analysts: An IT Mandate” Copyright © 1993 by E. F. Codd & Associates.
[4] Danica Porobic , “ Efficient OLTP Architecture for Future Multisocket Multicore Machines” DIAS, I&C, EPFL EDIC- Research Proposal.
[5] Danica Porobic Ippokratis Pandis Miguel Branco Pınar Anastasia Ailamaki, “OLTP on Hardware Islands”
[6] Nigel Pendse , “Online Analytical Processing Stream Data”
[7] Xuepeng Yin Aalborg University Fredrik Bajers, “Evaluating XML-Extended OLAP Queries Based on a Physical Algebra” November 12–13, 2004, Washington, DC, USA. Copyright 2004 ACM 1-58113-977
[8] Anil Vasudeve et.all “ Next Gen Infrastructure for in data Big Data” , SNIA Education