A SURVEY ON GOOGLE'S OFFLINE COMPUTING ENGINE

SATHYA D
PG Scholar
Erode Sengunthar Engineering College,
Thudupathi.

SIVAKUMAR G
Associate Professor
Erode Sengunthar Engineering College,
Thudupathi.

ABSTRACT

In the Big Data environment, MapReduce is one of the key approaches to meet the increasing demands in computing huge amount of datasets.MapReduce is a programming model which supports distributed and parallel computing on the data intensive applications. Hadoop has been introduced inorder to handle large amount of data from multiple distributed sources.HDFS is a tool to implement Hadoop.In this paper a study is made on the performance attributes of the native mapreduce.High performance is plausible one and only if performance attributes are well defined and they meet the dataset requirements.

Indexterms

Big data, Hadoop, HDFS, Hadoop-On-Demand.

INTRODUCTION

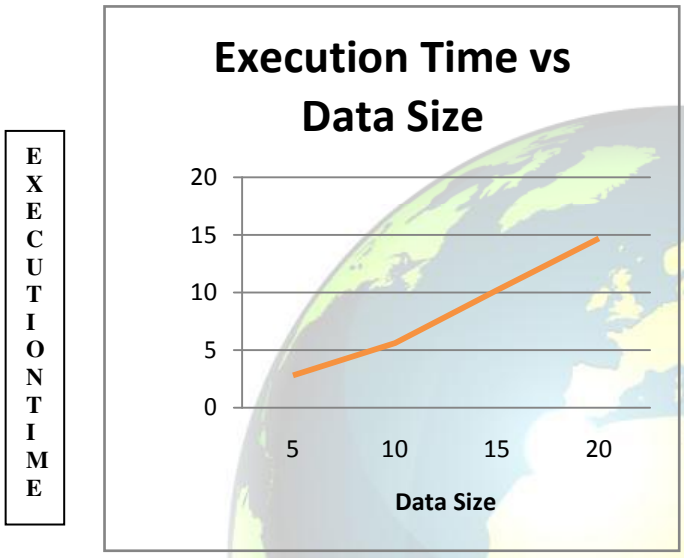
Internet services, such as search engines, on-line portals, e-commerce sites, and social networking sites, have emerged as an important computer applications[1], for delivering contents to more than trillions of users. These services not only deal with huge volumes of data, but also generate a large amount of data that needs to be processed every day. MapReduce is a programming model which supports distributed and parallel processing for large scale data-intensive applications such as machine learning, data mining, and scientific simulation. MapReduce divides input into multiple small chnks and lets them run on diverse machines in parallel. It is highly scalable because thousands of machines can be used as an effective platform for distributed computing. In addition, the MapReduce programming model[2] is designed to be easily reached to the widest possible class of developers; it favors simplicity at the expense of generality by trouncing implementationdetails

and providing simple abstract APIs. Hadoop is an open-source implementation of the MapReduce programming model. It is formerly developed by Yahoo!, but is used by every other companies including Amazon, Facebook, and The NewYork Times due to its high performance, reliability, and availability. Hadoop relies on its own distributed file system called HDFS (Hadoop Distributed File System)[4]. HDFS has a master/slave architecture;an HDFS cluster consists of a single NameNode and a number of DataNodes. NameNode is the master server which manages the namespace of a file system and regulates clients' access to files, while DataNodes are found to manage storage directly attached to each DataNode. HDFS has a coarse-grained policy of replicas.The common placement policy of HDFS is to place replica in the diverse racks scattered. This policy generally improves write performance by cutting down traffic between the racks.One of Hadoop's basic principles is "affecting computation is cheaper than affecting data." This delivers the fact that moving computation to the location of the data is easier fact than switching over the data to the place of the application. This is true in case of the huge data set because the migration[3] of the computation minimizes network traffic and increases the overall throughput of thesystem. When a computation task is located near the data it consumes, we call that the task has good data locality.

The data locality can be achieved the best, when the data to be used and the computation are placed in the same node.The next best case is that the data is in any other node within the same rack.Hadoop's assumption is that cluster nodes are keen to the MapReduce computation. In Yahoo!,a large experimental cluster called Yahoo!Grid[9] is managed by Hadoop-On-Demand (HOD). HOD is a management system for provisioning virtual Hadoop clusters over a multiple physical clusters. Each engineer at Yahoo! has an authenticated log-on account for the cluster and allocates their own virtual nodes on demand from a pool of physical nodes. Therefore,they are shared by more than one Yahoo!engineers.

Recently there have been a huge number of advances in software frameworks, such as MapReduce, that can be used to address the challenges inherent to the construction of highly parallel, high- performance and highly scalable distributed computing systems in a much plausible way[4]. Although MapReduce has been successful as a distributed computing substrate for simple highly parallel applications like search and data storage,there have beenfew

more complex systems such as scalable data management systems built using this or analogous frameworks.The problem context lashing our information system design is the need for web-scale informationsystems.



MAPREDUCE AND HADOOP

MapReduce is a software framework for dispensating and generating bulky data sets . Users stipulate a map function that splits data into key/value pairs and a reduce function that merges all key/value pairs based on the key. Many low-level tasks are expressible using the mapreduce model including word counting and the Page-rank algorithm.The MapReduce software framework is easily parallelizable for execution on bulk clusters of machines. This enables the edifice of high- performance, highly-scalable applications. One of the more famous MapReduce implementations is Hadoop. Hadoop takes care of the particulars of computing data on compute nodes through the Hadoop Distributed File System (HDFS), scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. Christo Ananth et al. [8] discussed about a method, Optimality results are presented for an end-to-end inference approach to correct(i.e., diagnose and repair) probabilistic network faults at minimum expected cost. One motivating application of using this end-to-end inference approach is an externally managed overlay network, where we cannot directly access and monitor nodes that are independently operated by different administrative domains, but instead we must infer failures via end to-end measurements. We show that first checking the node that is most likely faulty or has the least checking cost does not necessarily minimize the expected cost of correcting all faulty nodes. In view of this, we construct a potential function for identifying the candidate nodes, one of which should be first checked by an optimal strategy. Due to the difficulty of finding the best node from the set of candidate nodes, we propose several efficient heuristics that are suitable for correcting fault nodes in large-scale overlay networks. NameNode creates a restricted access for HDFS. Formerly there were only one data node and the name node.

In case of failure or system crash it was a very difficult task to recover from the disaster.

The next version consisted of name node, data node and a secondary name node. Here a new node named secondary name node has been introduced inorder to handle system failures and crash. Then the future versions are found to have similar advancements in theirdesigns.

CHALLENGES OF SIMPLIFIED DATA PROCESSING:

We know that MapReduce is a master-slave architectural model. The master will ping the slave periodically inorder ensure its presence. The communication between the nodes is carried out through Inter Process Communication(IPC messages). There is communication between the task tracker and the job tracker done through the heart beat message. There will be periodical responses between the name node and the data node. If there is no such responses then the master will consider that the worker had failed. As MapReduce is found to be distributed over multiple nodes, it is resilient tofailures.

In case of master failure two distinct significations are found to occur, current operation dies and a new copy will be started from the last check pointedstate.

IDENTIFICATION OF PERFORMANCE ATTRIBUTES

The performance attributes identified are Hardware, MapReduce, HDFS and Shuffle Tweaks. Increasing hardware utilization helps to improve efficiency, but is challenging to achieve for MIA(Map Reduce with Iterative Analysis) workloads[7]. The key insight is although MIA clusters host huge data volumes, the iterative jobs operate on a small fiction of the data, and thus can be done by a small reserve of dedicated machines. There are three main challenging factors related to workload are namely, tasks within a workload should have very similar resource demands as quantified within class for each workload dimensions. To compute numeric breakpoints that define the boundaries between qualitative coordinates for each workload dimension. Minimize the number of workloads in order to have inter task communication between the nodes, HPC applications, small ratio of computation to communication and the requirement for the specialized interconnects. Prediction of a possible intrusion attack in a network requires continuous collection of traffic data and learning their characteristics. The

recent visualization technique like dimension reduction and data projection can give us an aid in abstract view of the data but not true geometric representation[6]. Unit circle algorithm has been defined in order to track the back attack and the intrusions in the regular and irregular network. The problem here is the network topology which will lead to the delay in data or loss of the data. The missing-data problem is the vital conflict of this algorithm.

IMPACT OF THE RATIO TABLE

In order to handle the failure of the name node, when the HADOOP gets started, Ratio Table gets updated. This in turn simplifies the work of the name node by recording the capacity of the work assigned to each task[9]. Ratio Table records the type of jobs and the computing capacity ratio of each node. The computing capacity depends on the average execution time of a single task in that node according to the heartbeat message of each data node. There are two phases involved in handling the data through the ratio table which means if the records are available then it will divide the input according to their computing capacity, else the data will be equally distributed to all the nodes in the cluster and the name node will add a new record of this type of job in the ratio table[10]. The main result inferred from the survey is that theexecution time is proportional to that of the data size.Common Jobs Blocks Table(CJBT) stores the information about the jobs and the location of the blocks. This creates a serious impact in CPU execution time as of 86% from thewhole.

SCHEDULING MECHANISMS:

There were various scheduling mechanisms prevailing in the MapReduce environment inorder to improve the performance of the large processing clusters. The default scheduler within the HADOOP is FIFO scheduler. The level of performance generally depends on the system latency, memory settings, bandwidth and job parallelization[3].

The scheduling mechanism is broadly classified into two categories namely locality scheduler and resource aware scheduler. Now the serious research is moving on with resource aware scheduler. FIFO scheduler comes under the category of locality scheduler[3], the main demerit of this scheduler is that they give poor response time for the shorter jobs in presence of largerjobs.

Early versions of Hadoop had a very simple approach to scheduling user's jobs. They ran in order of submission, using a FIFO scheduler. Typically, each job would use the whole cluster, so jobs had to wait theirturn.

Although a shared cluster offers great potential for offering large resources to many users, the problem of sharing resources fairly between users requires a better scheduler. Production jobs need to complete in a timely manner, while allowing users who are making smaller ad hoc queries to get results back in a reasonable time. Mapreduce in Hadoop comes with a choice of schedulers. The default in MapReduce 1 is the original FIFO queue-based scheduler,
and there are also multiuser schedulers called the Fair Scheduler and the Capacity Scheduler. MapReduce 2 comes with the Capacity Scheduler, and the FIFOscheduler.

The Fair Scheduler

The Fair Scheduler aims to give every user a fair share of the cluster capacity over time. If a single job is running, it gets all of the cluster. As more jobs are submitted, free task slots are given to the jobs in such a way as to give each user a fair share of the cluster. A short job belonging to one user will complete in a reasonable time even while another user's long job is running, and the long job will still make progress. Jobs are placed in pools, and by default, each user gets their own pool. A user who submits more jobs than a second user will not get any more cluster resources than the second, on average. It is also possible to define custom pools with guaranteed minimum capacities defined in terms of the number of map and reduce slots, and to set weightings for each pool. The Fair Scheduler supports preemption, so if a pool has not received its fair share for a certain period of time, then the scheduler will kill tasks in pools running over capacity in order to give the slots to the pool running under capacity. The Fair Scheduler will work without configuration, but to take full advantage of its features need to configureit.

The Capacity Scheduler

The Capacity Scheduler takes a slightly different approach to multiuser scheduling. A cluster is made up of a number of queues, which may be hierarchical, and each queue has an allocated capacity. This is like the Fair Scheduler, except that within each queue, jobs are scheduled using FIFO scheduling. In effect, the Capacity Scheduler allows users or organizations to stimulate a separate MapReduce cluster with FIFO scheduling for each user or organizations to stimulate a separate MapReduce cluster with FIFO scheduling for each user or organization. The Fair Scheduler, by contrast, enforces fair sharing within each pool, so running jobs share the pool'sresources.

EXPERIMENTAL SETUP

Setting up of the HADOOP infrastructure

Setting up of the Hadoop infrastructure, the steps involved are ,

i.Configure the system with the Ubuntu operatingsystem.

ii.Installation of JDK inUbuntu.

iii.Installation ofHADOOP.

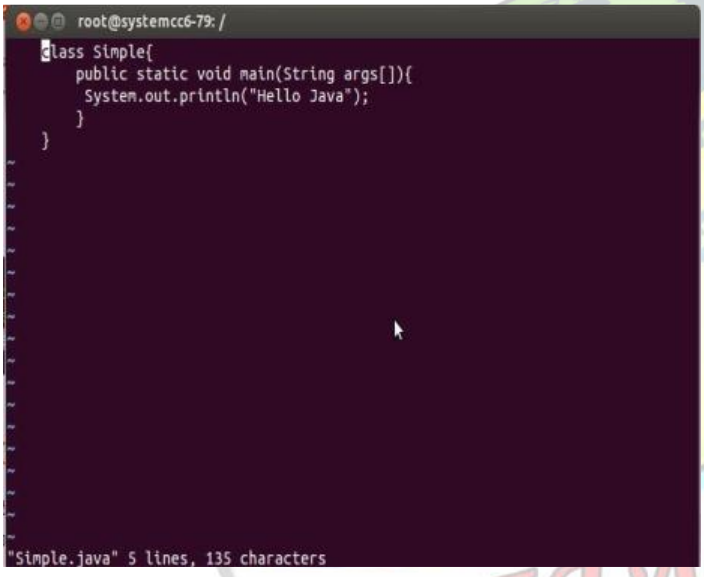


Fig 1.1 Checking out JAVA installation

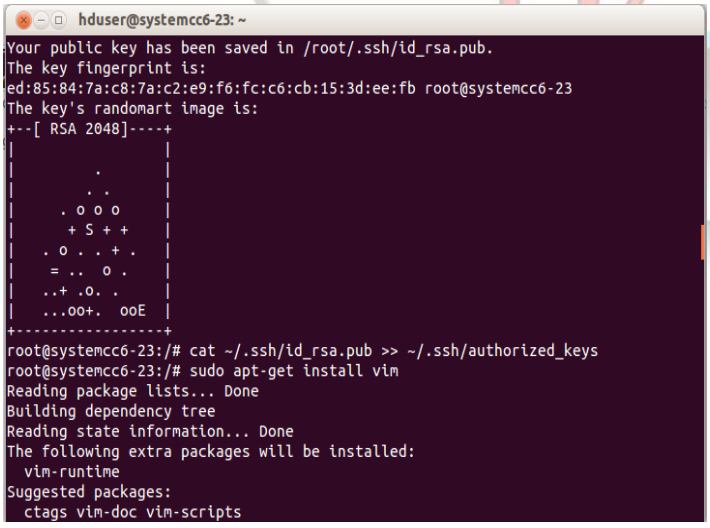


Fig 1.2 Installation of Hadoop

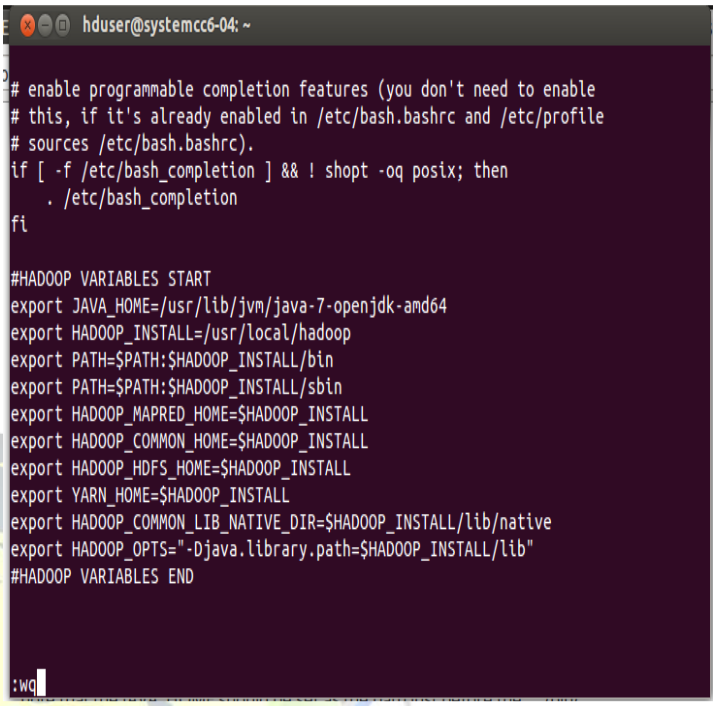
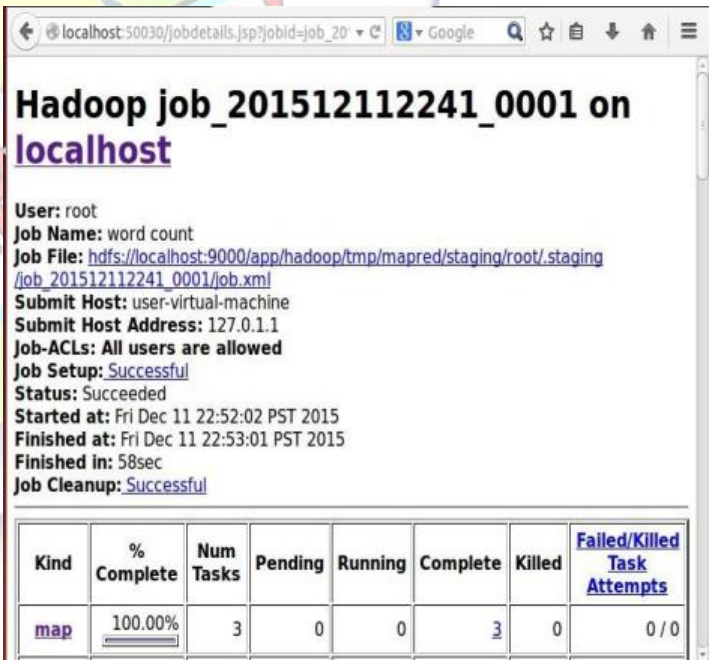


Fig 1.3 Hadoop Configuration



FUTURE WORK

So far various experiments has been conducted by various authors and inferences

were made. The future work that has been planned for implementation is the invention of new scheduling algorithm. This has been planned to bring out new era in performance attribute.

CONCLUSION

This paper has been made on the literature survey as the survey paper based on some of the performance attributes identified. The conclusion obtained from this is that development of new scheduling algorithm that will make drastic change in the performancescenario.

REFERENCES

[1] S.Krishnamoorthy and A.Choudhary, "A scalable distributed shared memory architecture", JPDC, vol. 22, no.3, pp.547-554,1994.

[2] Hadoop: The Definitive Guide, Third Edition, Textbook. Hadoop's Fair Scheduler. [3] E. Medernach, "Workload analysis of a cluster in a grid environment,"in *Job Scheduling Strategies for Parallel Processing*, 2005, pp. 36–61.

[3] E. Medernach, "Workload analysis of a cluster in a grid environment,"in *Job Scheduling Strategies for Parallel Processing*, 2005, pp. 36–61.

[4] Hung-Chih Yang, Ali Dasdan, Ruey- Lung Hsiao, and D.Stott Parker from Yahoo and UCLA, "Map-Reduce-Merge: Simplified Data Processing on Large Clusters",paper published in Proc. of ACM SIGMOD, pp. 1029–1040, 2007.

[5] M. Zaharia, A. Kowinski, A. Joseph, R. Katz and I. Stoica,"Improving mapreduce performance in heterogeneous environments". USENK OSDI.2008.

[6] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun ACM, 51(1), pp. 107-113, 2008.

[7] G. Mackey, S. Sehrish, and J. Wang, "Improving metadata management for small files in HDFS," in *CLUSTER*, 2009, pp. 1– 4.

[8] Christo Ananth, Mona, Kamali, Kausalya, Muthulakshmi, P.Arthy, "Efficient Cost Correction of Faulty Overlay nodes", International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE), Volume 1,Issue 1, August 2015,pp:26-28

[9] Y. Chen, S. Alspaugh, D. Borthakur, and R. H. Katz, "Energy efficiency for large- scale mapreduce workloads with significant interactive analysis," in *EuroSys*. ACM, 2012, pp. 43–56.

[10] F. Li, B. C. Ooi, M. T. Özsu and S. Wu, "Distributed data management using MapReduce," ACM Computing Surveys, 46(3), pp. 1-42,2014.