ISSN:0975-9646

Dynamic Cluster Configuration Algorithm in MapReduce Cloud

Rahul Prasad Kanu , Shabeera T P , S D Madhu Kumar

Computer Science and Engineering Department,
National Institute of Technology Calicut
Calicut, Kerala- 673601

***Abstract*— With the exponential growth of Data in recent time, industry and academia started looking for an intelligent data analysis tool that would satisfy the need of current requirements in storage and analysis of huge amount of data. The data growth is largely due to the impact of social media, scientific experiments and file logs created by different departments around the globe. MapReduce, proposed by Google in 2004 became popular for doing large scale data analysis. Industry is also concentrating on providing resources and services on demand, cost effectively and with high performance. Implementing MapReduce in cloud requires creation of clusters, where the Map and Reduce operations can be performed. Optimizing the overall resource utilization without compromising with the efficiency of availing services is the need for the hour. Selecting right set of nodes to form cluster plays a major role in improving the performance of the cloud. As a huge amount of data transfer takes place during the data analysis phase, network latency becomes the defining factor in improving the QoS of the cloud. In this paper we propose a novel Cluster Configuration algorithm that selects optimal nodes in a dynamic cloud environment to configure a cluster for running MapReduce jobs. The algorithm is cost optimized, adheres to global resource utilization and provides high performance to the clients. The proposed Algorithm gives a performance benefit of 35% on all reconfiguration based cases and 45 % performance benefit on best cases.**

***Keywords*— MapReduce, Hadoop, Cloud Computing, MapReduce Cloud**

I. INTRODUCTION

In last decade, scientific research trend have become increasingly reliant on processing huge volume of data. The data inflow has come from various fields such as Social Media, Weather Service Centre and Organization such as Newyork Stock Exchange. People send large volume of unstructured data in the form of messages and photographs. People have started creating tools to use and analyze these data. The overall impact of all these can be seen in near future. Since the size of the data is growing at a much higher rate than the rate of accessing the data [18], it became very important to create new system which can handle huge volume of data without deteriorating the performance. New concepts like Cloud computing, MapReduce and its open source implementation Hadoop became widely accepted for doing large scale data analysis. While cloud computing offers raw computing power in the form of storage and other services, there is a requirement of distributed framework to harness the power of cloud easily and efficiently. Google in 2004 introduced a model known as MapReduce [19], which was capable of handling execution of large distributed jobs in cloud infrastructure.

It has been found that operating cost of data centers have doubled in last 5 years and 75% of investment has been on infrastructure and energy consumption [5]. Gartner's survey [8] shows that enterprises invest 39% of their IT budget in Cloud. Improving the global resource utilization can reduce the overall infrastructure cost. Selecting the right set of nodes to form a cluster is one of the prime factors resulting in improved global resource utilization. Network Latency is one of the major factors of energy consumption in a cloud [17]. Hence we propose a Dynamic Cluster Configuration algorithm which reduces the network latency and can improve the utilization of the cloud resources without compromising with the efficiency of the cloud.

The test of the paper is organized as follows: Section 2 discusses on basics of Cloud Computing, MapReduce Framework, MapReduce Cloud and Hadoop. Section 3 highlights the related works that have been done in the field of MapReduce Cloud. Section 4 presents the proposed Dynamic Cluster Configuration algorithm. Section 5 presents the simulation setup used to test the proposed algorithm. Section 6 outlines the Results and analysis and Section 7 concludes the paper.

II. BASICS

A. *Cloud Computing*

Cloud Computing is a utility based computing that provides users with on demand, low cost and flexible services. Services can be both application delivered on internet or the hardware and software installed at the data center. The recent survey of Gartner[8] shows that the spending in cloud computing by various enterprises in IT budget has grown by 39% . Small companies benefit these services via access to resource without having need for prior investment. Cloud computing environment comprises of different models such as Software-as-a-Service(SaaS), Platform-as-a-Service(PaaS) and Infrastructure-as-a-Service(IaaS) [12]. The client sends their requests to the cloud service providers. These providers allocate resources to process the client's requests and earn via pay as you use model. Some of the major Cloud providers are Amazon EC2 [10], Microsoft Azure[11], GoGrid [4] and Rackspace cloud [7].

B. MapReduce Framework

MapReduce distributed data analysis framework was proposed by Google in 2004. It provides an easy to use programming model that featured fault tolerance, automatic parallelization, scalability and data locality based optimization[6]. Due to its feature of fault tolerance, MapReduce framework is suitable for handling large distributed jobs in dynamic environment such as cloud. In MapReduce framework, the data is broken down into small blocks and distributed on multiple machines called data nodes. MapReduce works with two functions: Map and Reduce. The nodes that perform the Map or Reduce tasks are called compute nodes. The map function takes the input data and generates a key value pair. Figure 2.1 shows the MapReduce execution overview. This key value pair is stored in the local disk as intermediate file. These intermediate key value pairs are shuffled/sorted and passed to the reducer. This assignment is controlled by the master node. The Reduce task then combines the results of all the key-value pairs and then generates the desired output [15].

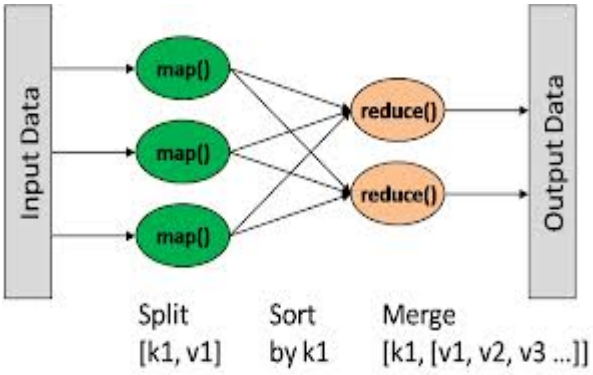

Figure 2.1 MapReduce Execution Overview [4]

C. MapReduce Cloud

MapReduce cloud is a model which can provide MapReduce as a Service to the clients. Once a MapReduce job gets submitted to the cloud it enters the job queue. The master processes the job to know the nature of the job. The scheduling algorithm schedules the job according to the nature of the job. After scheduling the job, a MapReduce Cluster is configured in the cloud that can handle the MapReduce job. The job is processed within the cluster and the result is sent back to the client. For the creation of a cluster, nodes are required that are provided from the pool of resources in the cloud. Since the number of resources can vary, we need to select the right set of resources that can form a cluster and also reduce the network latency while processing MapReduce jobs.

The major challenges of MapReduce Cloud are:

Storage of Data: The performance of any data intensive MapReduce task is highly dependent on the storage location of data. If the storage is not optimized, the network latency will be increased as a result of data transfer taking place between the Map and Reduce phases of processing.

Communication Consistency: This is the result of network sharing. When different nodes are exhibiting data transfer at the same time, communication consistency can become a defining factor.

Reliability: MapReduce can recover from node failure but if the master node fails then the whole system fails. So the reliabilitty of master node is a major challenge.

Consistency of Performance: As the underlying infrastructure may not be same, at extreme load conditions the performance of those nodes can fluctuate which cause the performance to deteriorate [2].

D. Hadoop

Hadoop is an open source implementation of MapReduce framework that can handle the distribution of data in a large set of commodity machines and then use efficient mechanisms to analyze those data. These mechanisms come in the form of Map and Reduce functions which has contributed to the wide use of Hadoop. Hadoop handles its data distribution with its Hadoop Distributed File System. Main components of Hadoop are Hadoop Distributed File System(HDFS) and MapReduce. Hadoop Distributed File System(HDFS) and MapReduce follow Master Slave architecture[14]. The main components of HDFS are NameNode, Secondary NameNode and DataNodes. The main components of MapReduce are JobTracker and TaskTrackers. The request enters the Hadoop cluster through the NameNode. JobTracker keeps track of the jobs and does the distribution of jobs into various TaskTrackers. There is one TaskTracker per node in the cluster. The individual task monitoring is done by the TastTracker and it sends the progress report to the JobTracker. These tasks are generally map or reduce type. So the JobTracker basically tries to divide the tasks such that data locality to complete the task is maintained. Figure 2.2 shows the Hadoop architecture along with the responsibilities of the different components.

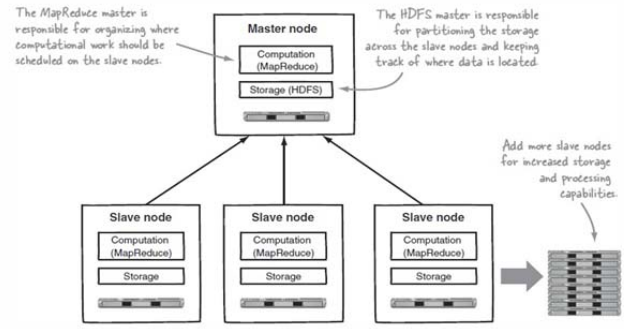

Figure 2.2 Hadoop Architecture [16]

III. RELATED WORKS

MapReduce has been a phenomenon in the dimension of how we perform data analysis. Its excellent feature of fault tolerance in a brittle cloud environment made it suitable for distributed data analysis. Enterprises such as Amazon, Microsoft and Google have created their own MapReduce cloud to provide MapReduce as an on demand service. Amazon Elastic MapReduce (EMR) provides MapReduce service hosted within the Amazon infrastructure[10]. EMR is a hosted Apache Hadoop MapReduce framework which

utilizes Amazon EC2[10] for computing power and Amazon S3[10] for data storage. It allows the users to perform Hadoop MapReduce computation via a web application interface. When a MapReduce job is processed, the data has to be copied from Amazon S3 to Amazon EC2 and the computation is performed on the local copy of Amazon EC2.

AzureMapReduce[6] is a distributed decentralized MapReduce runtime for Windows Azure that was developed using Azure cloud infrastructure services. Azure uses Azure queues for map and reduce task scheduling, Azure tables for metadata and monitoring storage data and Windows Azure compute workers to perform the computations. Google AppEngine MapReduce[6] is an open source library aimed at performing open source MapReduce computations. The AppEngine services perform the MapReduce computations. The experimental release only contains the mapper library while reduce phase is part of planned enhancement.

There have been other significant Models of MapReduce cloud. CURA[1] is a cloud managed model where clients submit jobs and the respective deadlines. It has a profiling and analysis service and a Resource management system. This resource management system consist of Secure instant VM allocation, job scheduler and VM pool manager. Secure instant VM allocation is a scheme where after completion of job Hadoop instance of the cluster is removed but re-initialization of the cluster can done for other jobs. The VM pool Manager understands the current workload and reconfigures the cluster to appropriate size by understanding the type of incoming jobs. The scheduling of jobs takes place via a reservation based policy. Comparatively analyzing above models, CURA doesn't require a separate unit for data storage.

In CURA[1], the cloud resources are divided into pools of clusters. The pools are small size, large size or extra large size pools. These pools have a pre-determined setup of clusters. When the profiler sends the number of nodes to the cluster configuration function, the functions selects the ideal cluster pool and searches for the cluster. If no cluster of required size is found, a reconfiguration based algorithm is run which shrinks larger sized cluster into smaller size.

The disadvantage with CURA's architecture is the application of reconfiguration based approach. Our proposed algorithm will be able to do the cluster configuration without the reconfiguration component. CURA do not create a large sized cluster from existing small sized cluster if there is a demand for large sized cluster and it does not have such a large sized cluster. In such cases CURA is incapable of handling jobs. For example, when there is a request for 4 node cluster and there are two clusters of size 2 available and there is no cluster of size 4 or above, CURA cannot process the request.

In general there are three different operation models in cloud. The first operational model is a completely customer managed model where each job and its resources are specified by the customer on a per-job basis and the cloud provider only ensures that the requested resources are provisioned upon job arrival. Amazon Elastic Compute Cloud, Amazon Elastic MapReduce uses this model[1]. But this model has a drawback, since there is lack of global optimization across jobs. The second possible model is partly customer-managed and partly cloud-managed model. Customers specify which resources to use for their jobs and the cloud provider has the flexibility to schedule the jobs as long as they begin execution within a specified deadline. The cloud provider takes risk to make sure that all jobs begin execution within their deadlines and as advantage can potentially does better multiplexing of its resources. The third model is a completely cloud managed model where the customers only submit jobs and specify job completion deadlines. The cloud provider takes greater risk and performs a globally optimized resource management to meet the job SLAs for the customers. CURA is a cloud managed model.

Since we aim to globally optimize resource management, we select cloud managed model as our operational approach for our proposed algorithm. We use an Integer Partitioning based dynamic clustering algorithm that will eliminate the concept of VM pools and will still be able to select the right set of nodes for Cluster configuration.

IV. DYNAMIC CLUSTER CONFIGURATION ALGORITHM

A. *Design*

The client submits the job and its deadline to the cloud service provider. The job enters the job queue in the cloud. The job is scheduled by a dynamic scheduling algorithm that uses reservation based policy to schedule the job. Once the job is scheduled, the job passes through a profiling phase where on the basis of the nature of job and the deadline for job completion, optimal number of nodes is generated as output. The output of the profiling phase is the optimal number of nodes that can perform the job. This number of nodes is passed through an Integer partitioning algorithm that generates all possible combination of numbers which result to the output of the profiling phase. The nodes are hence allocated to configure a cluster to handle the job. The proposed Cloud Cluster Configuration algorithm based on Integer Partitioning, makes sure that the nodes chosen for the configuration of cluster follows global resource optimization.

B. *Integer Partitioning Problem*

In number theory and combinatorics, a partition of a positive integer *n*, also called an integer partition, is a way of writing *n* as a sum of positive integers [3]. Two sums that differ only in the order of their summands are considered to be the same partition[12]; if order matters then the sum becomes a composition. For example, 4 can be partitioned in five distinct ways:

: 4
: 3 + 1
: 2 + 2
: 2 + 1 + 1
: 1 + 1 + 1 + 1

The order-dependent composition 1 + 3 is the same partition as 3 + 1, while 1 + 2 + 1 and 1 + 1 + 2 are the same partition as 2 + 1 + 1. There can be *m* ways of

representing *n* as a sum of positive numbers [9]. This paper generates S_k set of integers $\{x_1 ; x_2; x_3::: x_j\}$ the summation of which results to sum *n*. The numbers are generated in descending order to reduce the network latency. The integer partitioning problem is also transformed with a constraint that any number from the number set that is generated is limited to a maximum value,predefined as *max*.

The mathematical representation of the proposed algorithm is as follows:

if $S_k = \{x1 ; x2 ; x3::: xj\}$ is a set $\in$ S where ,

$k \leq m$

such that

$\forall$ xi | $x1 \geq x2 \geq x3 \geq \ldots \geq xj$

and

$$\forall \; Sk \sum_{i=1}^{j} xi = n \text{ where for all } xi \; \varepsilon \; Sk \leq max$$

$1 \leq k \leq m$

C. *Proposed Algorithm*

Once the profiling phase generates the optimal number of nodes to handle the job, the output of the profiling phase is passed through the integer partitioning algorithm. All switches have the information of the number of active nodes inside switch. The partitioning function generates set of possible *activenodes* combination that are matched with the active nodes within a switch. The number generated in the set by the partitioning function is in decreasing order, hence the algorithm ensures that the network latency is minimum. This is because the switch which has maximum number of nodes free will get the precedence to form a cluster. The algorithm traverses through all the switches linearly to find the S1 set of combination as discussed in previous section. The algorithm terminates if a complete match is found otherwise it checks for set S2 that is generated through the partitioning algorithm. The algorithm uses the concept of windowing. There are two windows, left window and right window. Once the first match x_1 from S_k is found the other values from the set are searched within the window range. This concept minimizes the network latency by minimizing the locality. The window is flexible and the cloud service provider can increase or decrease the size of the window.

Algorithm 1 is the proposed clustering algorithm that can be used to avoid the reconfiguration stage in CURA. This algorithm uses Integer Partitioning approach. Once the profiler decides the right number of nodes, the algorithm checks for the optimal cluster configuration using the partition function that sends the combination of all possible switch combinations to the cluster setup function.

The states of the switches are made alive until all the match are found. If at any time there is a mismatch, we roll back and change the state of the switches to the inactive state. All the switches of the racks have a variable called *active nodes* on the basis of the number of nodes that are free on the rack. Then through the mapping list which contains the information of the switch status, we use the coloring approach to greedily allocate nodes for handling the jobs. Each color represents active number of nodes on each switch.

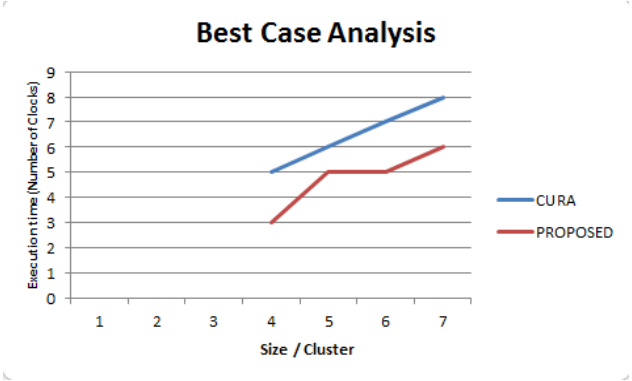

Figure 6.1 Best Case Analysis

Algorithm 1 Find the RightClusterConfiguration(numberOfNodes)

```
Require: SwitchStatus[1..n]=3,
  NodeList[1...n*m]=3,AvailableNodes=n*m,
  allSwitchSelected[1..n]=0
  if numberOfNodes >AvailableNodes then
    print "Cluster cannot be allocated due to insufficient Resource"
  else
    // check for all the permutation of the number of nodes in decreasing order such that we can get the permutation free in the mapping list
    while mappingList!=NULL do
      We check for each of the possibility until first match
      break
      // We get the best switches that can be used for our cluster, now we need to identify the nodes.
      // Decrement the value of each switch by the number of nodes used on that switch, change the color accordingly
      for allSwitchSelected do
        for i=1 to 4 do
          if NodeList[SwitchNumber*4 + i]==0 then
            NodeList[SwitchNumber*4 + i]=1
            We get the node list updated
          end if
        end for
        allSwitchesSelected[next]
      end for
      Call this function for all the request.
    end while
  end if
```

Algorithm 1 first selects the right switch combination, after that it checks for the nodes within those switch to do the cluster configuration. The colors which symbolizes the number of activenodes on the switches are changed accordingly until all the switches change their status to busy. For the implementation, we use *StatusSwitch* Array and *NodeList* Array. The *switch id* are numbered in ascending order of their index in the *StatusSwitch* array. The nodes are also stored in the same way in the *NodeList* array. We have assumed that the switches contain the same number of nodes, although this Algorithm can be extended to handle different number of nodes in a switch.

V. SIMULATION SETUP

We have implemented the proposed system in Java and Swing API. Initially the cloud has no resources. The resources are gradually added to the cloud. These resources are switches and nodes. Once sufficient number of resources gets added, the cloud is ready to serve client requests. The implementation has features where the Cloud Service Provider can monitor and manage all his resources. The Cloud Service Provider has the freedom to add nodes as per his requirement. The system generates report where the Cloud Service Provider can view the information of client and the number of resources allocated to them. The algorithm is scalable and can be extended to any number of nodes. In the current simulation, we have fixed the maximum number to 500.

CURA's experimental setups are identical. But it differs with our proposed algorithm in architecture. CURA needs to take care of management of pools of clusters, once sufficient number of switches and nodes are added, pools of clusters needs to be created. These pools also create an overhead of managing this cluster which deteriorates the performance of the existing CURA's implementation. While the implementation of Reconfiguration based component increases the cluster configuration time as nodes have to shut down to make the cluster small. The performance of the two algorithms is measured on run time of generating the cluster setup. The analysis is done on best cases and reconfiguration based cases. The network latency of the proposed algorithm is calculated taking .01 ms as inter-switch delay between the nodes within the switch and 0.20 ms as delay between switch to switch in message passing.

The comparative performance of the experiments is evaluated by maintaining a clock that calculates the runtime of both the experiments. The experiments were run on a single CPU (64 bit, Intel Pentium(R) i3 CPU 2.1GHz) with 4 GB RAM.

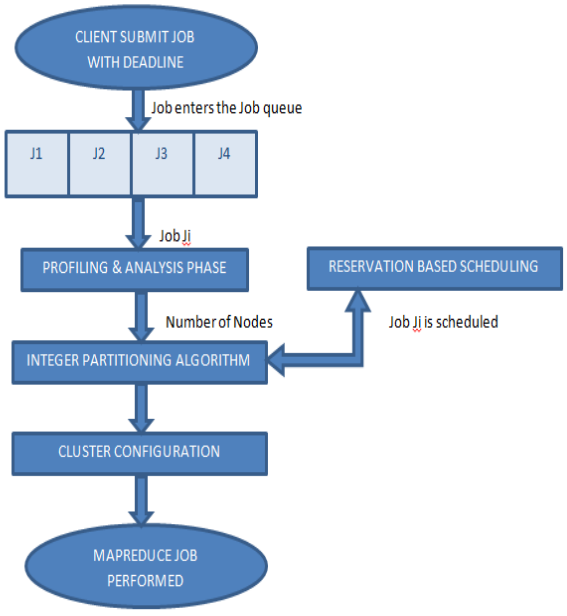

Figure Flowchart of Proposed Algorithm

VI. RESULT AND ANALYSIS

We found that the Integer Partitioning based Clustering Algorithm gives a better performance than the CURA clustering approach. The proposed algorithm has a performance improvement of over 35% on all reconfiguration based cases where there is a reconfiguration function applied on existing architecture. For the best case the performance benefit is over 45%. When there is no reconfiguration applied, both the algorithms gives comparable results. The proposed work has no overhead of managing Pools of cluster, which simplifies the management of cloud resources. All resources are identical as per our algorithm. CURA fails when there is an availability of two small clusters of size 2 and there is a requirement of size 4 cluster. It is not structured to integrate two small sized clusters, while the proposed algorithm has no such issues. The best case analysis result is obtained by considering identical cloud environments where there exists a cluster of same configuration. Once the client sends a request, both the algorithms run on the setup and the clock count is monitored. The clock count gives the effective time that the algorithm takes to execute. Through the analysis, we find that the execution times for 4, 5, 6 and 7 sized clusters are linearly increasing in case of CURA but in case of our algorithm the time required are found to be less. This is because our algorithm doesn't consider pool management and hence saves execution time. The line charts in figure 6.1 and 6.2 shows the comparison.

In Reconfiguration based cases, the proposed algorithm outperforms the existing cluster configuration model. The experiment is repeated where 1, 2 or 3 nodes had to be shut to configure a cluster. In all the cases, proposed algorithm performed better. This is because the reconfiguration components shut down computer. The more the number of computers it needs to shut down, more will be the execution time. The line chart shows the comparison. On average cases the results are identical and not much deviation was found.

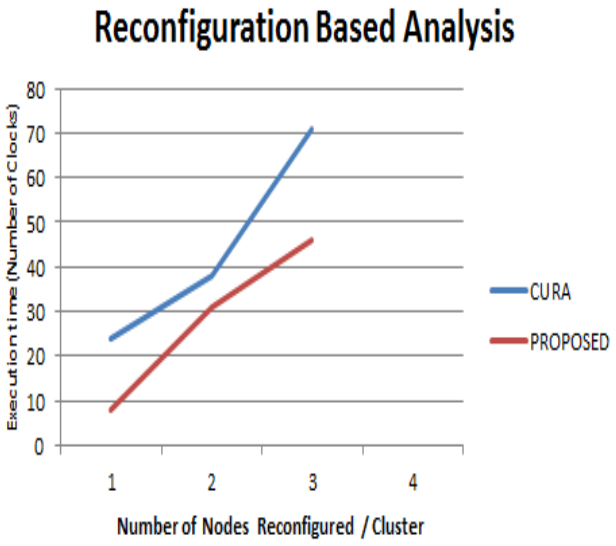

Figure 6.2 Reconfiguration Based Analysis

VII. CONCLUSION AND FUTURE WORK

Cluster Configuration plays a vital role in improving the performance of a cloud. Especially while considering a MapReduce cloud where a lot of data transfer takes place during Map and Reduce phases. Inefficient cluster configuration can drastically increase network latency which results in deteriorated performance. Cloud Managed model improves the performance of the cloud by considering global optimization of resources. After implementation of the algorithm proposed in CURA and relatively comparing with our algorithm, we find that our algorithm reduces the complexity of the cloud by avoiding segregation of the cloud environment into various sized pools of clusters. This reduces the extra work of managing those clusters. The concept of reconfiguration of these cluster pools is also avoided as the proposed algorithm doesn't use any reconfiguration of existing cluster and thus further improves the performance in dual respect.

The algorithm guarantees to provide optimal selection of resources with minimum network latency. It ensures global resource utilization in cloud. The proposed algorithm can be further improvised by automating the concept of windowing and incorporating it within the cluster configuration method. The algorithm has been tested on a homogeneous cloud. It can be extended to heterogeneous cloud also. The Cluster configuration algorithm can also be extended to work on any distributed cloud where a set of nodes are requested to process a particular task.

REFERENCES

[1] Balaji Palanisamy, Aameek Singh, Ling Liu, and Bryan Langston. Cura: A cost-optimized model for mapreduce in a cloud. In Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, pages 1275{1286. IEEE, 2013.

[2] Novia Nurain, Hasan Sarwar, Md Sajjad, Moin Mostakim, et al. An in- depth study of map reduce in cloud environment. In Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on, pages 263{268. IEEE, 2012.

[3] Wikipedia, http://en.wikipedia.org/wiki/Partition_%28number_theory%29, Accessed on 21st May, 2014

[4] GoGrid , http://www.gogrid.com/.

[5] Jiankang Dong, Xing Jin, Hongbo Wang, Yangyang Li, Peng Zhang, and Shiduan Cheng. Energy-saving virtual machine placement in cloud data centers. In Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE /ACM International Symposium on, pages 618{624. IEEE, 2013.

[6] Thilina Gunarathne, Tak-Lon Wu, Judy Qiu, and Geo_rey Fox. Mapreduce in the clouds for science. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pages 565{572. IEEE, 2010.

[7] Rackspace: The Open Cloud Company, http://www.rackspace.com/.

[8] Igou. User survey analysis: Cloud-computing budgets are growing and shifting; traditional it services providers must prepare or perish. Gartner Report, 2010.

[9] *University of Pennsylvania*, Joseph Laurend. Partitions of integers. 2005.

[10] An Amazon Company, http://aws.amazon.com/ 2014

[11] Microsoft, http://azure.microsoft.com/en-us/ ,2014

[12] Wikipedia, http:..en.wikipedia.org/wiki.Cloudcomputing Accessed On 24th May, 2014

[13] Pragmatic Programming Techniques: http://horicky.blogspot.in/2008/archive.html, Accessed on March 2014

[14] Yahoo Developer Networks, https://developer.yahoo.com/hadoop/tutorial/.

[15] Yahoo Developer Networks https://developer.yahoo.com/hadoop/tutorial/module4.html.

[16] Hadoop IN PRACTICE by Alex Holmes, http://www.sappers.co.in/category/hadoop/

[17] Balaji Palanisamy, Aameek Singh, Ling Liu, and Bhushan Jain. Purlieus: locality-aware resource allocation for mapreduce in a cloud. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, page 58. ACM, 2011.

[18] O'Reilly; Third edition, Tom White. Hadoop: A definitive guide.2012

[19] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1(2008): 107-113.