

## **ADVANCED COSHH: A Hadoop Scheduling Algorithm in Cloud Computing**

A Dissertation

Submitted By

Animesh Jha

(11007221)

To

Department of Computer Science

In partial fulfilment of the requirement for the

Award of the degree of

Master of Technology in Computer Science

Under the guidance of

Asst. Prof. Parminder Singh

(May 2015)

## **Abstract**

The high scalability of Cloud computing infrastructure has significantly increased the use of parallel computing. Hadoop is a framework for parallel processing of large datasets in a clustered datacenter. Its work is based on a MapReduce model, for providing efficient processing of Big Data. This solution is being used by large number of Cloud computing companies for performing desired performance level. The increase in the demand of sharing Hadoop clusters among various users leads to an increase in the system heterogeneity. The heterogeneity of the resources causes reduce in efficiency in the Hadoop system. These difference in the type of jobs provided to the Hadoop system increases the challenges to the scheduler in the real Hadoop workload.

The present work on COSHH schedules jobs on the basis of heterogeneity at both the application and the cluster level. The tasks are scheduled on the basis of minimum share satisfaction, fairness and locality with respect to other scheduling algorithms. However, the problem with COSHH algorithm is that when more than one jobs have same priority, the scheduling is done randomly. The ACOHH algorithm focuses on this drawback of COSHH and schedule the jobs with same priority on the Shortest Job First basis. This improves the efficiency of the present system by reducing the completion time of the jobs as jobs with less estimated completion time are executed before the jobs with more completion time.

Moreover, the priority of the jobs are decreased as soon as the jobs burst time is finishes.

This provides protection against resource starvation.

## **Acknowledgements**

I have to thank a lot of peoples for helping me completing this work: family, teachers and friends. This work would not have be possible without your help and support.

I should thank my mentor Mr. Parminder Singh for his support and his regular guidance in helping me in completing my work. I would like to thank him for helping to understand the topic from scratch and then guiding the work in steps.

I would like to extend my special gratitude to Dr. Shweta Mam for her support and for constantly motivating me for completion my work.

I would like to thank my family for being a support while working on this report. I would like to thank all my friends for helping me and sharing all the information related to my topic.

## **Declaration**

I hereby declare that the dissertation proposal entitled, “**ADVANCED COSHH: A Hadoop Scheduling Algorithm in Cloud Computing**” submitted for the M. Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: 3<sup>rd</sup> May, 2015

Animesh Jha

11007221

## **Certificate**

This is to certify that Animesh Jha has prorating M.Tech dissertation proposal “**Advance COSHH: A Hadoop Scheduling Algorithm in cloud computing**” under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma. The dissertation is fit for the submission and the partial fulfillment of the conditions for the award of M.Tech Computer Science & Engineering.

Date: - 3<sup>rd</sup> May, 2015

Signature of Advisor  
Parminder Singh

## Table of Contents

PAC Form .....	ii
Abstract .....	iii
Acknowledgements.....	iv
Declaration .....	v
Certificate .....	vi
CHAPTER 1 .....	1
INTRODUCTION.....	1
1.1 Cloud Computing: Overview .....	1
1.1.1 Virtualization .....	1
1.1.2 Utility Computing .....	2
1.1.3 Service Oriented Architecture (SOA) .....	2
1.2 Cloud Computing Architecture.....	3
1.3 Cloud Delivery models.....	3
1.3.1 Infrastructure as a Service (IAAS).....	4
1.3.2 Platform as a Service (PAAS).....	4
1.3.3 Software as a Service (SAAS) .....	4
1.4 Cloud Deployment model .....	5
1.4.1 Public Cloud.....	5
1.4.2 Private Clouds .....	6
1.4.3 Community cloud.....	6
1.4.4 Hybrid Cloud.....	6
1.5 Hadoop .....	6
1.5.1 Hadoop: Architecture .....	7
1.5.2 Working Process .....	8
CHAPTER 2 .....	10
LITERATURE SURVEY.....	10
CHAPTER 3 .....	13
PRESENT WORK.....	13
CHAPTER 4 .....	21

RESULTS AND DISCUSSIONS.....	21
CHAPTER 5 .....	30
CONCLUSION.....	30
CHAPTER 6 .....	31
REFERENCES .....	31

## List of Figures

1.	Fig 1.1 Cloud Computing.....	3
2.	Fig 1.2 Cloud Delivery Models.....	5
3.	Fig 1.3 Amazon MapReduce working process.....	7
4.	Fig 1.4 Hadoop MapReduce working process.....	9
5.	Fig 3.1 Overview of COSHH Scheduler.....	14
6.	Fig 3.2 COSHH Algorithm.....	17
7.	Fig 3.3 Advanced COSHH Flowchart.....	18
8.	Fig 4.1 Select Scheduler Interface .....	22
9.	Fig 4.2 Database for Job Queuing .....	23
10.	Fig 4.3 FIFO Scheduler Interface .....	24
11.	Fig 4.4 ACOSHH Scheduler Interface .....	26
12.	Fig 4.5 FIFO Vs ACOSHH .....	27
13.	Fig 4.6 COSHH Vs ACOSHH .....	28



# CHAPTER 1

## INTRODUCTION

---

### 1.1 Cloud Computing: Overview

Overview Cloud Computing is the term used for providing computing facility over the internet. The National Institute of Cloud computing [11]. "Defines it as a model for enabling convenient, on –demand network access to a shared pool of configurable computing resources (eg. Network, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The cloud computing is one of the recent evolving technology. It was popularized in the year 2007 with the joint collaboration of IBM and Google in this field. The further interest in this field was added with the development of Blue cloud by IBM. Cloud computing is the result of intense researches conducted in many different underlying technologies. These technologies can be named as: -

- Virtualization
- Utility Computing
- Service oriented Architecture (SOA)

#### 1.1.1 Virtualization

Virtualization is a process or a method of dividing the available resources into multiple execution environments. This is done based on various factors like resource sharing, timesharing, etc. [12]. It is used in datacenters to provide the facility to run multiple applications on the same server and to provide abstraction to the applications from each other. This helps the organization to consolidate multiple servers without sacrificing application isolation, scaling there organization according to the requirement and increasing availability through dynamic provisioning and relocation of critical resources. Thus the concept of virtualization provides a datacenter the advantage of flexibility, availability, scalability, efficient hardware utilization and security. The cloud is a type of parallel and distributed computing system. So clients are dynamically provided resources using the concept of virtualization based on the Service-Level-Agreements (SLA) that has been duly accepted by a series of negotiation between the service provider and the user client.

### **1.1.2 Utility Computing**

Utility computing is a concept that forms another major component for the popularization of Cloud computing. It helps the customers to use the resources provided by the service providers and the pay based on their usage (pay-as-you-go). This offers a number of benefits to both service provider and the users. From the provider's point of view, the actual hardware and software are not configured to be used one user or solution. Instead, the resources can be provided dynamically as the demand changes.

From the User's prospective, the advantage includes low cost and complexities. [4], in his paper explains that the clients do not need to invest heavily on building or maintaining IT Infrastructure. The resources are provisioned dynamically so the client do not have to worry about the under or over usage of the resources.

### **1.1.3 Service Oriented Architecture (SOA)**

In an SOA environment, the client requests for a service from his service provider. This service request can be of a functionality, a quality or a request for Storage space. These service can be requested either in real-time or at the particular time period [5].

It can also be defined as a designing pattern that is based on different types of software. These software provides different functionalities as services to the applications. These functionalities are provided using different types of protocols. The major protocol used are:

- REST (Representational State Transfer)
- SOAP (Simple Object Access Protocol)

The term 'cloud computing' was coined by Eric Schmidt of Google in the year 2006 to explain the type of services providers over the internet. It refers to the collection of applications over the internet and the hardware and system software in the datacenters providing such services. Thus the datacenter software and hardware is what collectively known as cloud. In a distributed computing framework, there's a huge workload shift. Nearby PCs no more need to do all the hard work in terms of running applications. The system of PCs that make up the cloud handles them. Equipment and programming requests on the client's side lessening.

The current service provides for the cloud service include Amazon, Google, Microsoft, etc. Figure 1.1, represents an overview of the cloud environment.

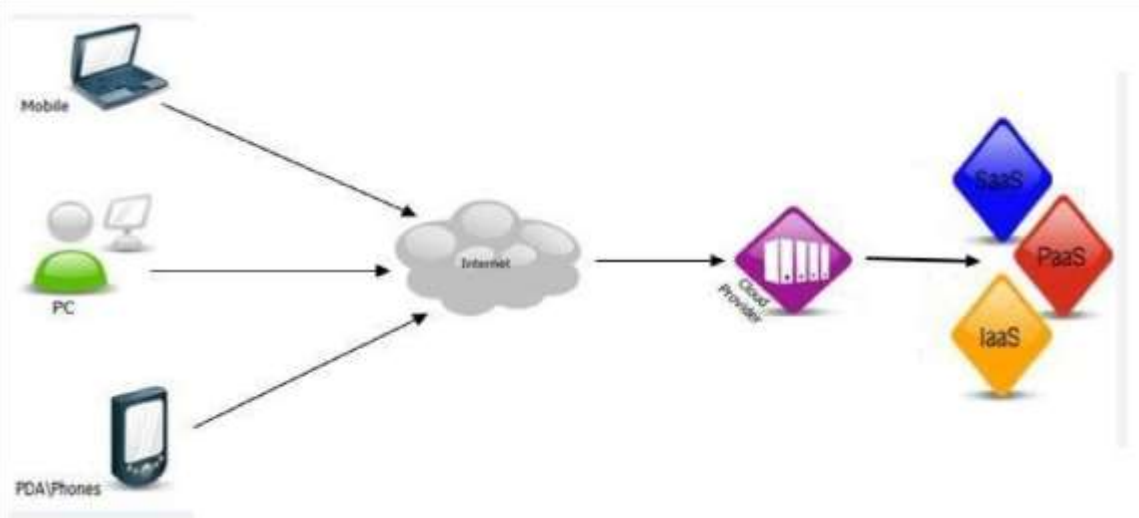


Fig. 1.1 Cloud Computing

## 1.2 Cloud Computing Architecture

In this section, discussed a high level view of the architecture of the cloud computing model. Also discussed about the major concepts and technologies that resulted the development of cloud computing. The basic architecture of cloud computing, consists mostly of infinite resources (storage, compute, etc.), virtualization software, system and application software, network and finally the clients using these services over the internet.

The cloud services provided to the clients can be categorized in many ways based on the abstraction level of resources, flexibility to the clients and type of users to which the cloud is provided. This leads to two major models for cloud service provisioning.

- Cloud Delivery Model
- Cloud Deployment Model

## 1.3 Cloud Delivery Model

The Web services are mostly three tiered. These include the physical Infrastructure, the operating system and finally the software application. In case of cloud computing all these services can be provided over the internet. Thus any service provided over the internet is defined as X-as-a-service (XAAS). These services are as follows:

- Infrastructure as a Service (IAAS)
- Platform as a Service (PAAS)

- Software as a Service (SAAS)

### **1.3.1 Infrastructure as a Service (IAAS)**

Infrastructure as a service is a cloud delivery model in which an organization provides its client with Storage, Compute and Network components. These services are provided to the clients through Virtual Private Servers (Turner 2009).

The Virtual Private Server (VPS) allows abstraction of hardware alone. The clients are provided with a VPS machine image. These image can be provided from different servers located in different geographical locations. It also provides the clients to save the state of running virtual machine servers can be saved at any time. This helps in providing very easy and convenient backup procedure. The Client has the facility to install and run any operating system and applications desired. Companies providing VPS include WebFusion, Amazon, ParaScale, etc.

### **1.3.2 Platform as a Service (PAAS)**

Platform as a Service can be considered as a service between SAAS and IAAS. In case of PAAS, the clients are provided with space on Windows based server. The platform is accessed through a web interface and the clients can install any application that are specific to that platform. This helps in minimize the developer's maintenance and provides a considerable amount of customization and configuration. Some of the cloud PAAS are Microsoft's Windows Azure and Google Application Engine. (Turner 2009).

### **1.3.3 Software as a Service (SAAS)**

The software as a Service provides its users the facility to use a specific software over the internet. The software is installed into the provider's server and it can be used by clients.

The software as a Service provides the client to use a specific software over the internet. A particular software is installed on the server and the clients can only use an instance of that software. The hardware, operating system and everything else is abstracted from the client. This helps the clients to access the applications from anywhere in the world. Some of the major advantages of SAAS also include easy backups, flexibility pricing, portability, etc. The client do not have to worry about the application he is using over the cloud environment. It depends on the application provider to manage the licensing of the cloud and the other major

issues. It saves the clients from the burdens of Installation and licensing of the software being used. Eg of SAAS include Salesforce.com, Microsoft's SharePoint, etc.

Figure 1.2, represents the overview of the components that are shared in each of the three cloud delivery models.

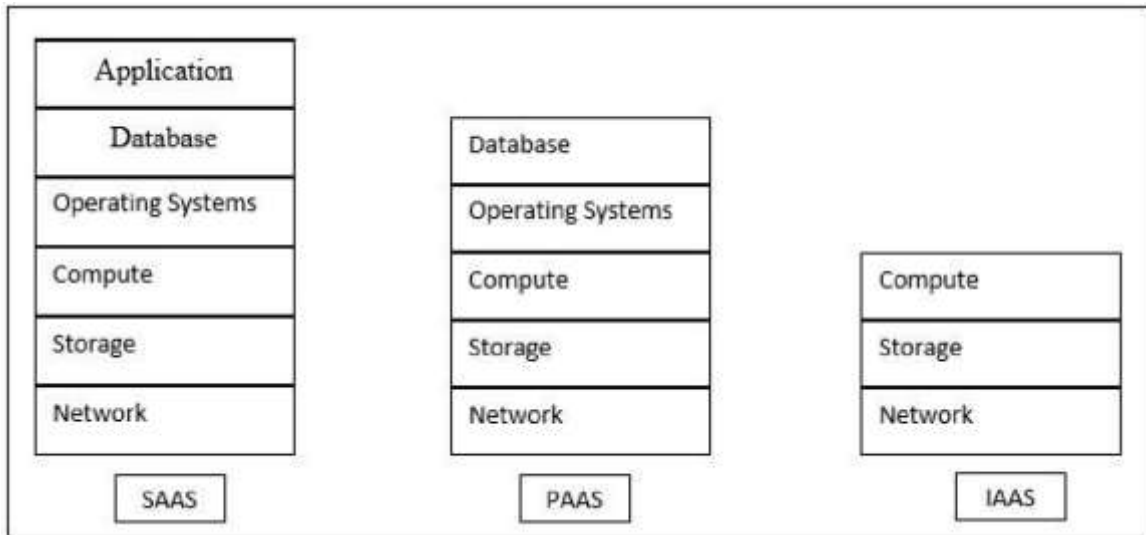


Fig. 1.2 Cloud Delivery Models

#### 1.4 Cloud Deployment model

In cloud Deployment model, the cloud services provided is categorized on the basis of ownership, size and access. Based on these factors, there are four common cloud deployment models.

- Public Clouds
- Private Clouds
- Community Clouds
- Hybrid Clouds

##### 1.4.1 Public Cloud

Public Clouds can be accessed publically by anyone but are owned by a third-party cloud provider. The resources are provisioned on the basis of cloud delivery models. These resources are generally provided to the customers at a cost or are popularized through advertisements. The cloud provider is responsible for the creation and maintenance of the public cloud.

The resources are monitored and has a very limited security. These cloud are mostly off campus and are used by a large number of organizations, individuals and anyone interested to pay for the required services.

#### **1.4.2 Private Clouds**

The private clouds can be on-campus or off-campus. These services are owned by a single organization and it provides the facility to access the centralized resources from different locations or departments of the organization. In case of private cloud, the same organization acts like a cloud provider and consumer. So to differentiate between the provider and the consumer, a separate department in the organization is provided with the responsibility to provisioning the cloud. The peoples in the department requires an access information to the private cloud.

#### **1.4.3 Community cloud**

A community cloud is quite similar to a public cloud except that the access to the cloud environment is restricted to the participating organizations. The cloud environment may be hosted by some third-party vendor or from within one of the organizations in the community.

#### **1.4.4 Hybrid Cloud**

Sometimes the cloud infrastructure is hosted with the properties of two or more type of clouds. These clouds remain unique entities but are also bounded on by standard or underlying technologies. The organizations generally use the hybrid cloud for optimizing the resource requirement and provisioning of resources in an efficient manner.

### **1.5 Hadoop**

In this section, we provide an introduction to Big Data and the problems that were faced in processing them. . We also discuss about the architecture of Hadoop and its working process.

The advancement of cloud computing technology resulted in the enormous increase in data being collected from a large number of different sources. This generation and collection of a large volume of data from a variety of sources like web applications, sensors etc. at a high velocity is termed as Big Data. One of the fastest growing application in the field of research is the processing of Big Data.

The scalability and fault tolerance features of cloud computing provides an important solution for working in these applications. However, the huge size and processing needs of ‘Big Data’ causes major issues in this field. Hadoop can defined as a cross-platform framework that has the capability to support data intensive as well as compute intensive distributed cloud applications with a focus on data analysis.

### 1.5.1 Hadoop: Architecture

The Hadoop is a batch process. The under lying technology used by Hadoop is MapReduce. The Map-Reduce is a programming model as well as a framework that supports the underlying programming model. It is designed to abstract the details of parallel programming and provides its users a facility to focus only on the required data processing strategies. The Map –Reduce model consists of two primitive methods: Map and Reduce.

In the programming model, these Map () and Reduce () are defined by the users. The input to the map function is a list of key-value pair (Say {K, Y}). Each pair is passed to the Map () to produce an intermediate value. The intermediate key value pair is then shuffled and grouped together on the basis of key equality {K, list (V)}. This value is then passed to the reduce function to produce an aggregated results.

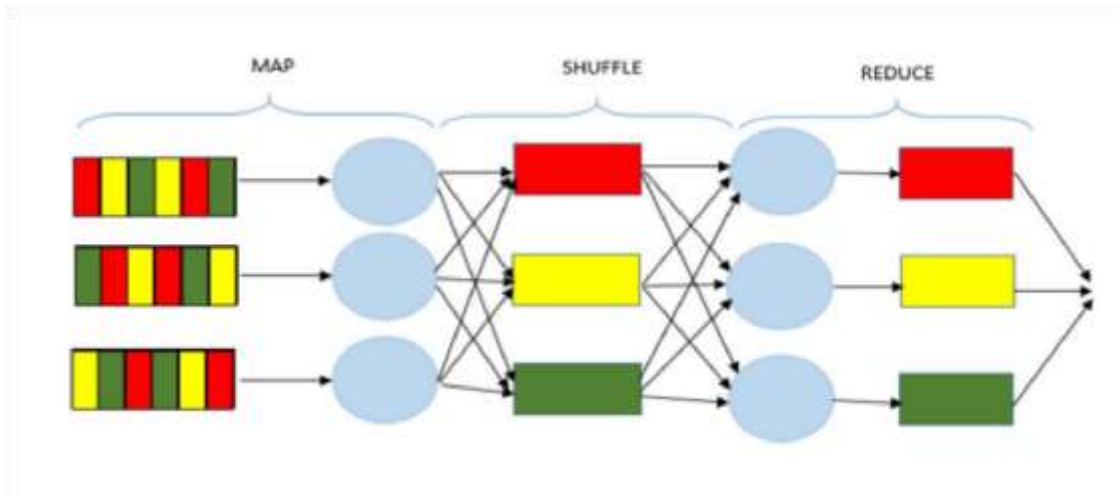


Fig.1.3 Amazon Map Reduce working process

The MapReduce framework stores and reads data from a special type of file System called HDFS. HDFS or Hadoop Distributed File System is used to for storing large datasets in

a distributed fashion. These datasets are broken down into smaller chunks of data before storing them in the file system.

The HDFS file system is inspired form the earlier Google File System (GFS) [23]. It supports Fault-tolerance by implementing data partitioning and replication. HDFS is blocked structured and is manage by a single Master node.

The input to the MapReduce framework is segregated into a large number of Map tasks and one or more Reduce tasks depending on the available resources and the SLA agreement between the service provider and client.

### **1.5.2 Working Process**

Every MapReduce task is performed in two specific phases: Map phase and Reduce phase. A Hadoop cluster consists of a Namenode multiple worker nodes called Datanodes. The master node consists of:

- JobTracker
- TaskTracker
- NameNode
- Datanodes.

The Hadoop cluster works on the master-slave model. The master node or the Namenode is responsible for providing jobs to the Datanodes for processing. The Job tracker is responsible for breaking the tasks into map and reduce tasks, assign the tasks to the available Datanodes, monitor the progress of TaskTrackers and on completion of all the tasks, provide a report to the user. The Datanodes on the other hand, responsible for processing of the map or the reduce task as assigned by the Namenode.



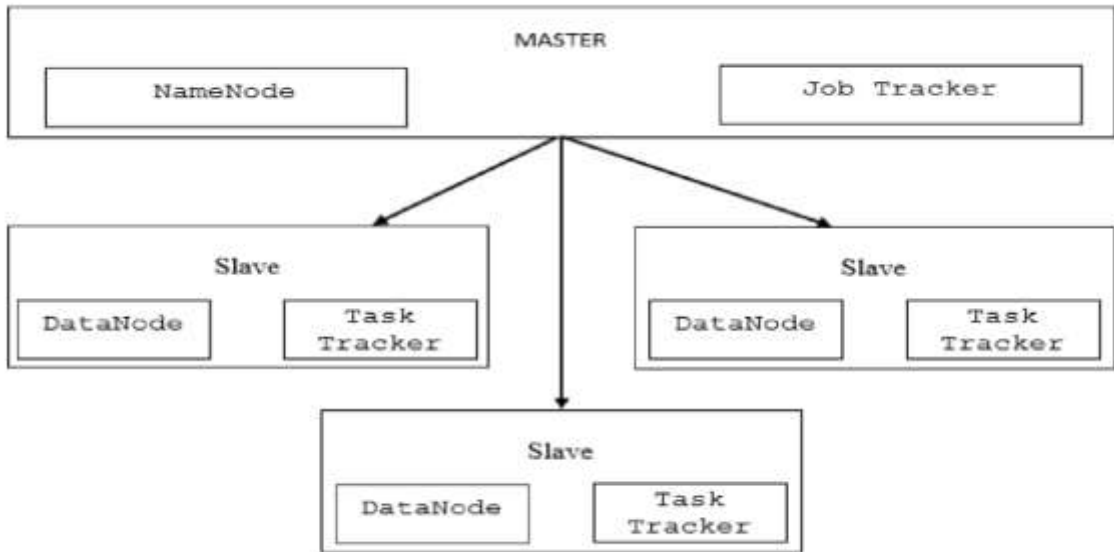


Fig 1.4. Working process of Hadoop Map –Reduce

## CHAPTER 2

### REVIEW OF LITERATURE

---

Due to high usage of Hadoop MapReduce technology, there has been many researches in this field. The paper “Optimizing data shuffling in data-parallel computation by understanding user defined function” provides us with many improved advancements for data shuffling phase in MapReduce process.

The First paper on Hadoop MapReduce was published by Google in the year 2003 and it gives a survey of all the benefits that Hadoop MapReduce provides [8]. It also gives a complete structure of MapReduce operations. The Master Node is responsible for storing and managing the Name Nodes and other data nodes. It keeps several data structures. For each map and reduce task, it stores the information about the working machine. These information consists of the state and the identity of the working machine. The paper provides a basic understanding of structure, implementation and performance related issues. In this particular paper, we study different scheduling algorithms for scheduling of jobs provided to Hadoop – MapReduce Framework. The scheduling of jobs is done for fast completion of jobs and efficient use of resources.

The default Hadoop scheduler is based on FIFO algorithm. Initially, Hadoop was designed for small clusters and a scheduling algorithm like FIFO was able to provide a good performance. However, with the popularization of Hadoop, FIFO caused a severe performance degradation, especially in conditions where data was shared among multiple users.

In order to solve this problem, Hadoop on Demand (HOD) was developed [9]. The Hadoop on Demand addresses this issues by providing private Hadoop clusters on demand. It allowed users to share common files while owning private Hadoop clusters on their allocated nodes. The limitations with this approach was that it compromised the data locality design of the original Map-Reduce scheduler and also resulted in poor resource utilization.

So to overcome the limitations of FIFO, the developers at Facebook developed a Fair Scheduler algorithm [3]. In Fair Scheduler, all the jobs are allocated a minimum share of the cluster capacity over time. The jobs are assigned to pools and each pool is guaranteed a

minimum share of resources. In case of excess resources, the resources are shared evenly among all jobs. When a pool finishes its job, the resources in the pool are distributed evenly among the remaining pools to complete the jobs. The paper also discusses two new techniques, delay scheduling and copy –compute splitting. In delay scheduling, if the node requests a task, and the head-of-line job cannot launch a local task, then we skip the job and search for subsequent jobs. However, if the job is skipped for a long time, then we let the job to launch a non-local task in order to avoid starvation. The next problem is of I/O bound and Compute bound task. This is achieved by providing a separate task process for both copy and compute task.

The problem with Fair Share scheduling is that it does not achieve good performance in case of heterogeneous jobs. So to solve this problem, in paper [4], Yahoo developed a capacity scheduler. It works when the number of users is large and there is a need for a fair allocation of resources among all the users. The algorithm allocates jobs in the form of queues. These queues contain a configurable number of Map and Reduce tasks. The queues have a particular configured capacity. The free capacity of the queue is shared among the other queue. Within a queue, the jobs are scheduled on the modified priority queue basis with a specific user limit. However, in this case, the queue set and queue selecting groups are not carried out automatically. The user needs to know the system information and perform these tasks manually. This is the major bottleneck of this type of algorithm. In this paper, the authors develops an improved weighted round robin algorithm. In this case, the jobs are provided with a definite weight. The weight is calculates as follows:

$$\text{Weight} = \text{Initial weight} * \text{priority factor} \quad (i)$$

The priority factor is predefined and the jobs are processed based on their final weight. The algorithm supports preemption of the jobs after a particular burst time.

Kc, K et al. [14], in his paper, gives priority to the user constraints. These constrains like deadlines are important requirements. The paper extends the real time cluster approach to derive a minimum of map and reduce tasks for performing deadline based job scheduling in Hadoop. The paper leaves certain parameters like the map –reduce task runtime estimation, filter ration estimation, multiple map reduce cycle support, etc. for future works.

Sandholm et al. in his paper, showed that their scheduler works better than the existing Hadoop scheduler in the number of queues [15]. The fair scheduler was not able to handle two queues whereas the capacity scheduler was not able to handle the workload of ten queues.

B. Thirumala Rao in his paper, summarizes the advantages and disadvantages scheduling policies of various Hadoop schedulers presented in different research papers. The paper leaves the task of scheduling in heterogeneous clusters for future work. [16]. H, Chang et al. [18] proposed algorithms that outperforms many other scheduling algorithms like FIFO, SJF and others across different job distributions. In this paper, the authors introduces a theoretical framework for optimal scheduling in MapReduce. He also provide two algorithms for the optimal scheduling of the jobs. The offline version of the algorithm called as OFFA is used to design a 3-approximation algorithm for offline scheduling problem. This algorithm considers that the release time or the arrival time for the job and the completion time of the jobs is known before the arrival of the job. The online version of the algorithm, however considers the real-time scenario in which jobs can arrive at any time. It also considers the job priority which is provided by the weight of the job.

W. Wang et al. in their paper, try to address the problem of balancing between the data locality and load-balancing to expand the throughput and decrease delay in the completion of the job. In this work, a new queueing architecture is presented and a map task scheduling algorithm is proposed that constitutes of two different policies: (a) Shortest Queue policy, and (b) Maximum Weight Policy [19].

Rasooli et al. in the paper discusses a new algorithm COSHH. In COSHH, the scheduler considers the heterogeneity of the jobs under consideration. It uses a classification and optimization technique to restrict the search space. The jobs are categorized based on their requirement. The jobs are placed on different classes. The future work suggests improvement in the COSHH algorithm by separating the data or I/O intensive and compute intensive jobs [20]. Some author discusses the multi-level feedback queue algorithm that is currently implemented on the operating system. It has a number of distinct queues, each assigned a different priority level. It uses priorities to decide which jobs should run at a given time. It uses different rules that helps in separating the data intensive and compute intensive jobs. [21].

## CHAPTER 3

### PRESENT WORK

---

In this section, we discuss the problem in the Hadoop scheduler which motivated us to develop a problem definition for our research work. In the later part of the paper, we discuss the objectives as well as the proposed methodology of the presented Hadoop Scheduling Algorithm.

The COSHH scheduler is first designed and implemented in Hadoop to work with heterogeneous workloads. The system information is used to make better scheduling decisions. This lead to improved performance. The COSHH algorithm has several components. These components are used to gather system information. It provides a means to calculate the mean job execution time based on the structure of the job and the number of map and reduce tasks available for each jobs. The main motivation behind COSHH are as follows:

- Scheduling for minimum share required by the jobs, fairness of the resources being shared between jobs and also keeping in mind the heterogeneity of the jobs.
- Decrease in the communication cost of the Hadoop system.
- Decreasing the search overhead for matching jobs and resources.

The present COSHH works on the basis of the heterogeneity of the resources at both the application and cluster level. The algorithm considers the parameters of data locality, resource availability, fair share and User Dissatisfaction.

Rasooli [20] in her paper, shows that the experiments are carried out on the MRSIM and the results show that COSHH is better than most of the previous algorithms when heterogeneity of the resources is considered and the workload on the resources is high. An overview of the architecture of the COSHH is presented in (Fig 3.1) [20]. A Hadoop scheduler typically contains two main messages form the main Hadoop system. Firstly, a message that signals the name node that a new job is submitted by the user and secondly, it contains a heartbeat message from all the free resources. Thus a COSHH algorithm consists of two major processes. These processes are triggered by one of the messages. The scheduler on receiving a

new job performs a queueing process to store the incoming job and assign it to an appropriate queue.

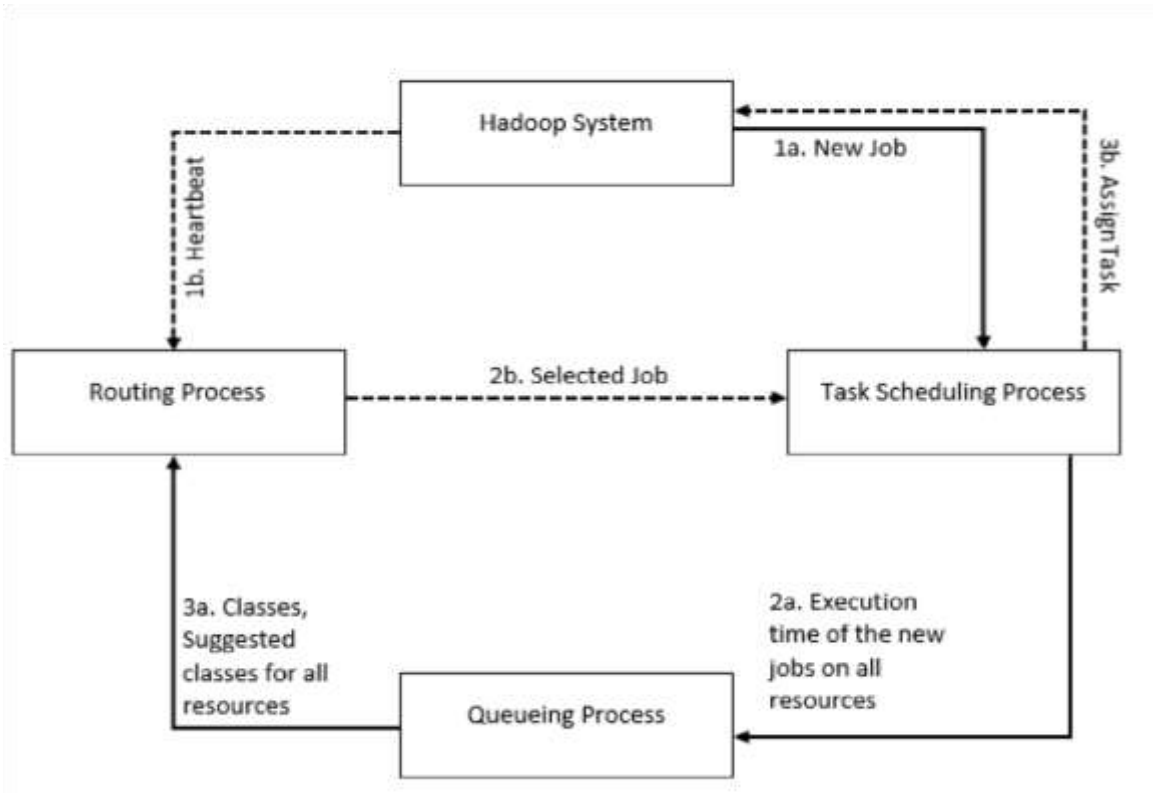


Fig 3.1. Overview of COSHH Scheduling Algorithm

The heartbeat message helps the scheduler to trigger a routing process to assign a job to the available free resources. The new job message and the heartbeat message are represented as 1a and 1b respectively in the Figure 3.1. The high level view of the COSHH represents four major components.

- The Hadoop System.
- The Task scheduling process.
- The Queueing Process
- The Routing Process
- The Hadoop System

The Hadoop System is a collection of cluster. A cluster is a collection of linked resources responsible for making a resource pool. The data in a system are stored in the form

of file in HDFS format. All the jobs submitted to this system consists of a number of tasks. These tasks can be grouped under two categories:

- Map Task
- Reduce Task

This work, assume that there are N number of users and each user submits a set of jobs. Each job submitted by the user, is assigned a priority based on the user's service level agreement (SLA) and the type of Job. Each user is provided with a minimum share of resource. The minimum share is defined as the minimum number of slots that must be provided to each user at any point of time.

Since the job submitted by a user (U) at any time (T) is completely dynamic, so a set of jobs (say  $J_i$ ) submitted by user (U) at time ( $T_1$ ) is completely different from that submitted at time ( $T_2$ ). Moreover, each job submitted by the user consists of a map task and a reduce task. Thus a Job (J) submitted by a user, can be represented as:

$$J_i = \text{Map}_i \cup \text{Reduce}_i \quad (\text{ii})$$

### **Task Scheduling Process**

The task scheduling process is responsible for providing map or reduce tasks to the nodes for execution. In order to do this scheduling, the task tracker also needs to keep an information about the jobs submitted by the user. So, when a job is submitted by a user, an estimation of the mean execution time of that task on all resources is done. This is done with the help of a task duration predictor which estimates the mean execution time of the incoming jobs. This component is a result of research from AMP lab at US Berkeley [1].

In order to define a predictor algorithm, numerous analysis were performed to identify important log signals. The prediction algorithm is then introduced using these log signals and its accuracy is evaluated on workload of Hadoop. It is important for the prediction algorithm to make decisions with microseconds and with fairly high accuracy.

The estimator achieves this goal in two sub-components.

- Chronos - It is a background daemon and is responsible for analyzing Hadoop history logs as well as for monitoring cluster utilization after ever regular intervals. According

to the study by [2], an interval of six hours is able to provide the desired accuracy for Facebook and Yahoo workloads.

- Predictor: - It is the second component and is responsible for on-the-spot decision making on the basis of file the operate on, the input bytes they read, by consulting the lookup table populated by chond and finally returns the mean execution times. Studies by S. Agarwal, show that an 80% of accuracy is achieved by predictor algorithm in estimating the mean execution time for incoming jobs. [1]

### **Queueing Process**

Figure 3.2 shows the queuing process. It uses a classification and optimization based approach to classify jobs into classes. As soon as a new job is submitted to the user, the algorithm uses a classification approach to specify a Job class for the incoming job. The jobs are then assigned to the corresponding queue. If the current jobs does not fit in any of the current queues, then an optimization algorithm is used to find an appropriate class for the job. The results of the queuing process is sent for execution to the routing process on receiving a heartbeat signal from the resource. Rasooli et al. composed the scheduler in view of the way that there are two basic criteria with various levels of significance in a Hadoop framework. The RST measure, forced by the Hadoop supplier, is fulfilling the base shares. The Hadoop suppliers ensure that upon a client's solicitation at whenever, her base offer will be given instantly (in the event that doable). The second measure, imperative to make strides in the general framework execution, is reasonableness. Considering reasonableness forestalls starvation of any client, and partitions the resources among the clients in a reasonable way. Least impart fulfillment has higher criticality than reasonableness. Hence, COSHH has two classification, to consider these issues for RST least impart fulfillment, then for reasonableness. In the essential classification (for least impart fulfillment), just the occupations whose clients have min offer  $> 0$  are classified and in the auxiliary classification (for decency) all of the occupations in the framework are considered. The occupations whose clients have min offer  $> 0$  are considered in both classification. The reason is that when a client requests more than her base offer, RST her base offer is given to her promptly through the essential classification. At that point, additional shares ought to be given to her in a reasonable manner by considering all clients through the auxiliary classification.



The classification and optimization algorithms in the queueing process is used to reduce the search space for finding the appropriate job and related resources. It also helps in reducing the completion time of the jobs. However, these techniques also add up to increase the workload of the scheduling process. So we limit the number of times these steps are performed.

However, in COSHH algorithm, during the routing process, the jobs in the routing queue are submitted for execution randomly. Thus jobs with more execution time or low priority are selected for execution before jobs with less execution time and higher priority.

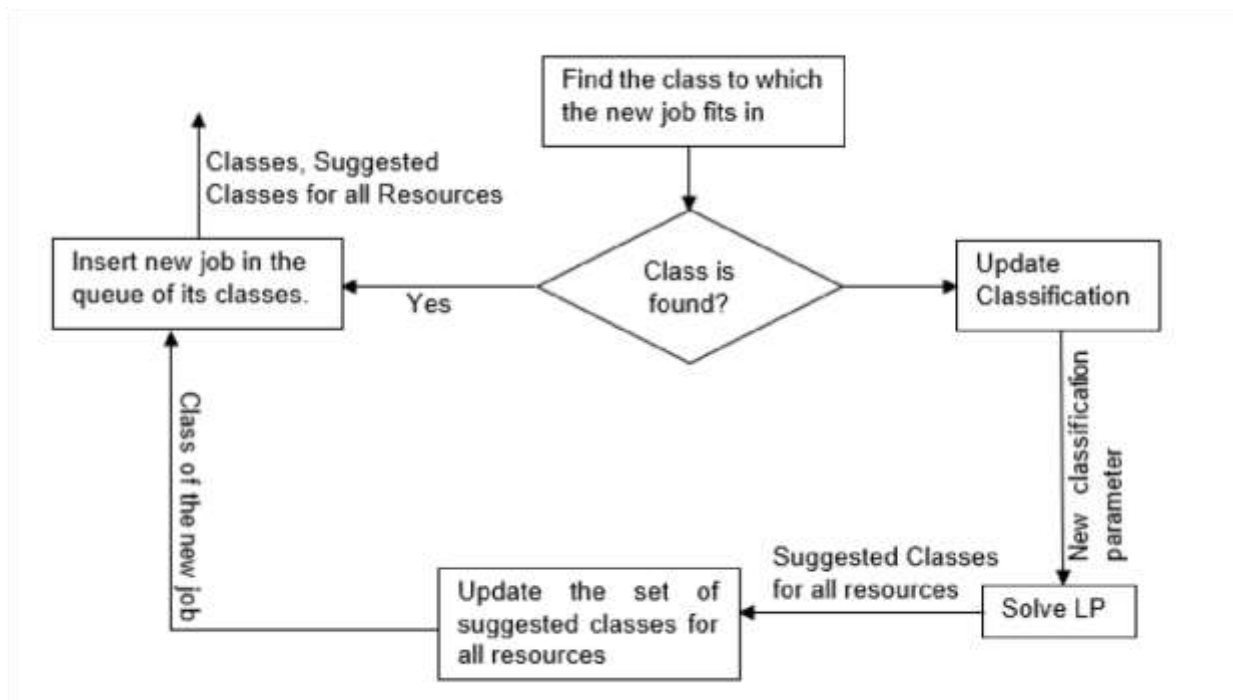


Fig. 3.2. COSHH algorithm

## Objective of Study

The objective of the research report is to improve the scheduling process and efficiently use the resources by separating the data intensive and compute intensive jobs and placing them into separate queues. The specific aims of this research paper includes:

- To understand and analyze the present Hadoop scheduling algorithm in both homogeneous as well as heterogeneous workload.
- To develop an ADVANCED COSHH and implement it in Hadoop Environment.

- To analyze the performance of ADVANCED COSHH with existing COSHH algorithm on performance matrices: minimum share satisfaction, fairness, data locality as well as data type of jobs.

## Methodology

Advanced-COSHH uses the technique of Multi-level feedback queueing on the jobs submitted to the class queues and helps in maintaining the proper resource distribution for the jobs in queue assigned to a class. The algorithm also provides a way to classify the compute intensive and data intensive jobs and apply certain set of rules to overcome the problem of starvation and proper resource allocation.

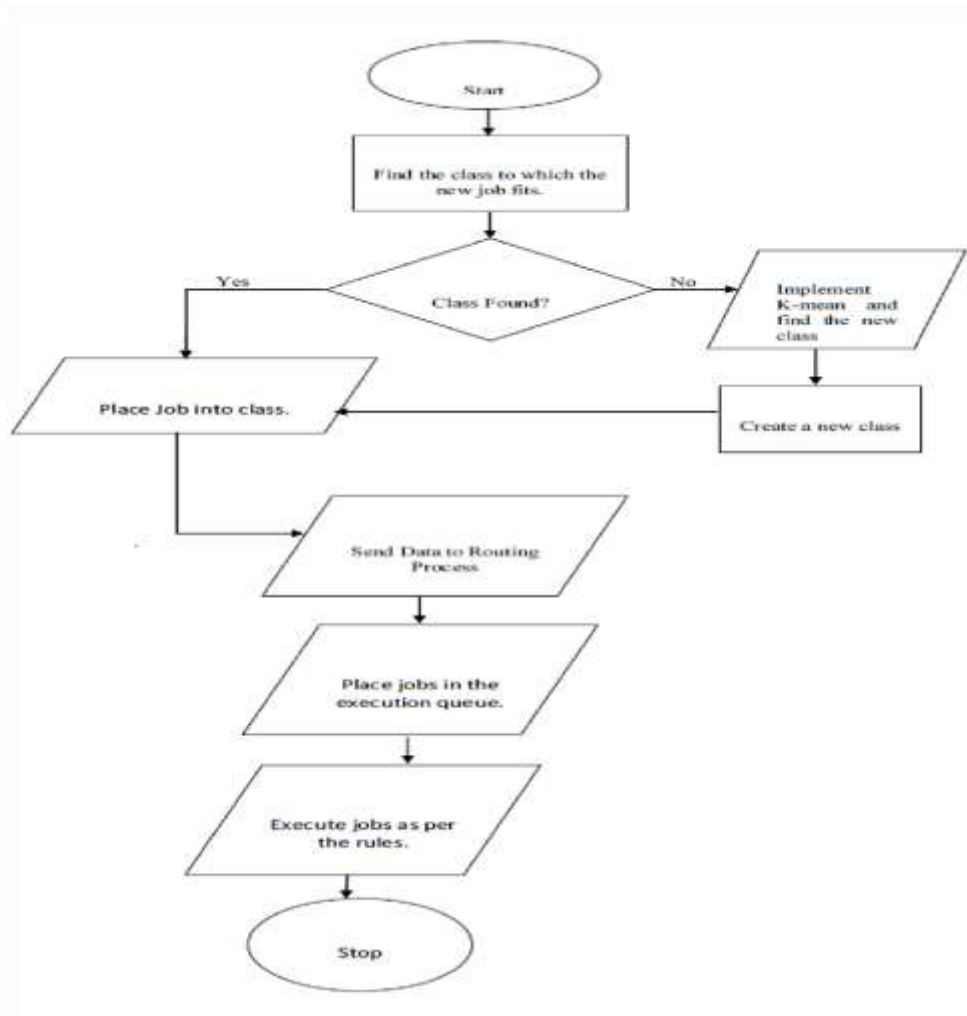


Fig. 3.3. Proposed Advanced COSHH algorithm

In the proposed algorithm, we consider the queueing of the jobs within the class. All the jobs in the same class have same priority. So we create a number of distinct queues within each class, each having a different priority levels. The algorithm works on following rules.

### **Queueing Process Algorithm**

When a new job (say  $J_{new}$ ) is submitted by the user,

Get execution time of  $J_{new}$  from task scheduler

1. If  $J_{new}$  fits in any Class Queue (say  $C_i$ ) then,
2. Add  $J_{new}$  to the queue of  $C_i$
3. Else
4. Implement K-mean clustering algorithm to find a class for  $J_{new}$  and place it in the queue of that class.
5. End if

Submit the job classes (JobClass1, JobClass2, JobClass3.... JobClassN) to the routing process

### **Routing Process Algorithm**

When a heartbeat message is received from a resource ( $R$ ),

SC= Selected Class

If ( $User.minShare - User.CurrentShare > 0$  and Class belongs to SC),

If there is a Job ( $T$ ) which requires the current free resource, then

Add Job ( $T$ ) to the queue of execution process and wait for other resource to be freed.

Execution Process

For all the jobs in the set (Say  $J$ ),

NFS= Number of free slots, EST = Estimated Completion time BST = Burst Completion time.

If (Priority ( $A$ ) > Priority ( $B$ )), then execute A before B

Else

If (Priority ( $A$ ) = Priority ( $B$ )), then

If (EST ( $A$ ) < EST ( $B$ )), then A executes before B

Else

If (EST ( $A$ ) = EST ( $B$ )), then A and B executes in SJF.

If  $(BST(A) > \text{DEFAULT}(BST))$ , then  $Priority(A) = Priority(A) + 1$ .

As soon as a job releases a resource, say R, the resource sends a heartbeat message, this triggers the routing process. In this step, the routing process receives the job set from the queuing and processes it to analyze and send a set of jobs for execution. For all the executable jobs from the routing process and performs one of the following steps.

1. If  $Priority(A) > Priority(B)$ , then
2. Add A to the set of selected jobs
3. If  $Priority(A) = Priority(B)$ , then
4. If  $(EST(A) < EST(B))$ , then add A to the set of selected jobs
5. If  $(EST(A) = EST(B))$ , then A and B runs in FIFO.

## CHAPTER 4

### RESULTS AND DISCUSSIONS

---

In this section we define the parameters that we consider for our job scheduler and also investigate and prove our hypothesis by demonstrating different algorithm under controlled environment. We also describe the experimental environment used for the implementation purpose.

#### **Experimental setup**

The experimental environmental for demonstration of the scheduler is developed in visual studio 2013 windows application. We use C#.net as the language for developing the application. For the storage of data we use SQL Server 2012 as the backend database.

Due to the absence of the supportive algorithm, such as the predictor algorithm, COSHH algorithm and others, we implement them ourselves according to the given algorithms.

Our algorithm considers three specific parameters for selection of the job for execution. These parameters include Priority of Job, Resources required for the job, and Estimated Job Completion time.

In the Hadoop system, each job is spilt into map and reduce tasks. These map and reduce tasks are assigned to each job depending on the Job Priority and the SLA between the Hadoop service provider and the Client. In general, the number of map tasks is always more than number of reduce tasks. We implement the same logic for our demonstration.

In the remaining part of this paper, we show the snapshots for the interface and the working of our algorithm and finally the results that we achieve from the execution of the work. Most present Hadoop planning algorithms consider reasonableness and least impart objectives without considering heterogeneity of the employments also, the assets.

One of the benefits of COSHH is that while it addresses the reasonableness and the smaller than usual minimum offer necessities, it does this in a manner that makes efficient assignments, by considering the heterogeneity in the framework.

## Snapshot

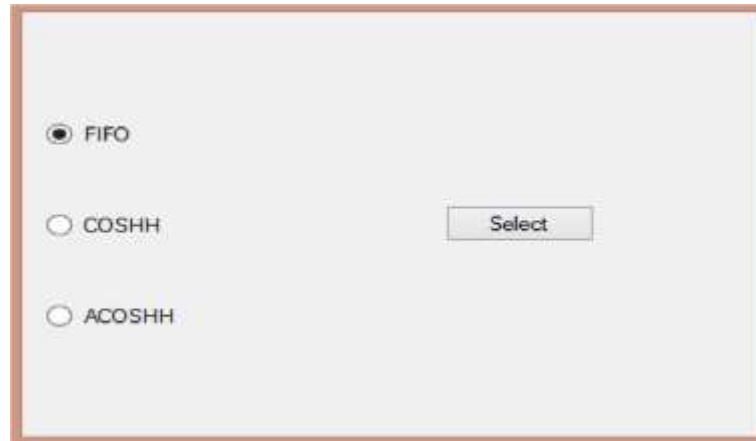


Fig 4.1 Select Scheduler Interface

Fig 4.1 shows the snapshot of the first page of the demonstration. It is used for selecting the particular type of scheduler that we want to execute. We do a comparison between two three different types of algorithms. We store the jobs in the table format in SQL Server databases. The table represents a rack in the Hadoop system. Every time a job is submitted by the client, it is stored in the database with following parameters specific to each job.

- Jobid – It provides a unique ID to each job submitted by the user.
- Job Priority- Provides each job with the priority according to the client's service level agreement with the Hadoop service provider.
- Job Estimated Completion type- It provides the estimated completing time for each job submitted by the user. This is made possible by a predictor algorithm which is provided by AMP lab at US Berkeley and has been explained in details by S. Agarwal in his paper.
- Resource required- Every job submitted to the Hadoop system needs some resources which can be a Storage Space, CPU Time, Network bandwidth or Memory.
- Node Distance – In a multi node Hadoop system, the task scheduler has more than one node for assigning the jobs. So when a job is submitted, it waits for the resources needed for its execution. As soon as the required resource is available, all the jobs requesting for that resource tries to access it. In that case, we need the nodes distance of the job and the resource in order to provide the job to the nearest node.

- Map and Reduce - As explained earlier, each job is a collection of map and reduce tasks. The number of maps and reduce tasks are decided for each job based on the resources it needs and the priority of the job. The map and reduce tasks are responsible for the execution rate for each job. The map task is responsible for splitting a job into smaller tasks.

Jobid	Job_name	Job_priority	Job_Size	Job_NodeDista..	Job_maptask	Job_reducetask	Job_ECT	Job_status	Req_Resource_1	Current_Share	Min_Share
1	J1	2	27	1	9	5	12	Completed	CPU	4	10
2	J1	2	365	2	13	11	18	Completed	DISK	5	10
3	J1	2	252	2	6	3	8	Completed	RAM	3	10
4	J1	2	291	2	8	4	11	Completed	RAM	3	10

Fig 4.2 Database for Job Queue

The database represents the parameters that are used for execution of a job in a Hadoop system. The scheduling of jobs to the task scheduler for execution is done based on these parameters. The FIFO is the most basic type of scheduling algorithm and do not consider the complex parameters. The COSHH and ACOSHH algorithms however, executes the jobs based on these parameters and this helps in decreasing the mean execution time of the jobs in the system.

Figure 4.3 represents the execution of a FIFO based scheduler. In this type of scheduler, the jobs are executed based on first come first serve basis. This is the most basic type of scheduling algorithm. In FIFO, we do not consider the priority of the jobs submitted by the user. The jobs submitted by the user, is queued in a sequence that they are received. Each job is executed in the same sequence as it is received. The FIFO algorithm works well for low workload system but in situations where the work load is high, this algorithm fails to provide the desired result.

The COSHH scheduler is designed to implement scheduling on the basis of fairness, minimum share requirements, job heterogeneity and resources. The Hadoop system specifies the issues of minimum share requirement and fairness. This algorithm is based on messages between the systems. First the jobs are queued in the system and clusters are made based on the K-mean clustering algorithm. The clustering is done based on the estimated completion

time of the job. This is calculated using the predictor algorithm. First the jobs are classified into classes. These class queues form the queueing process of the job. In the second process, the job are provided to the routing process.

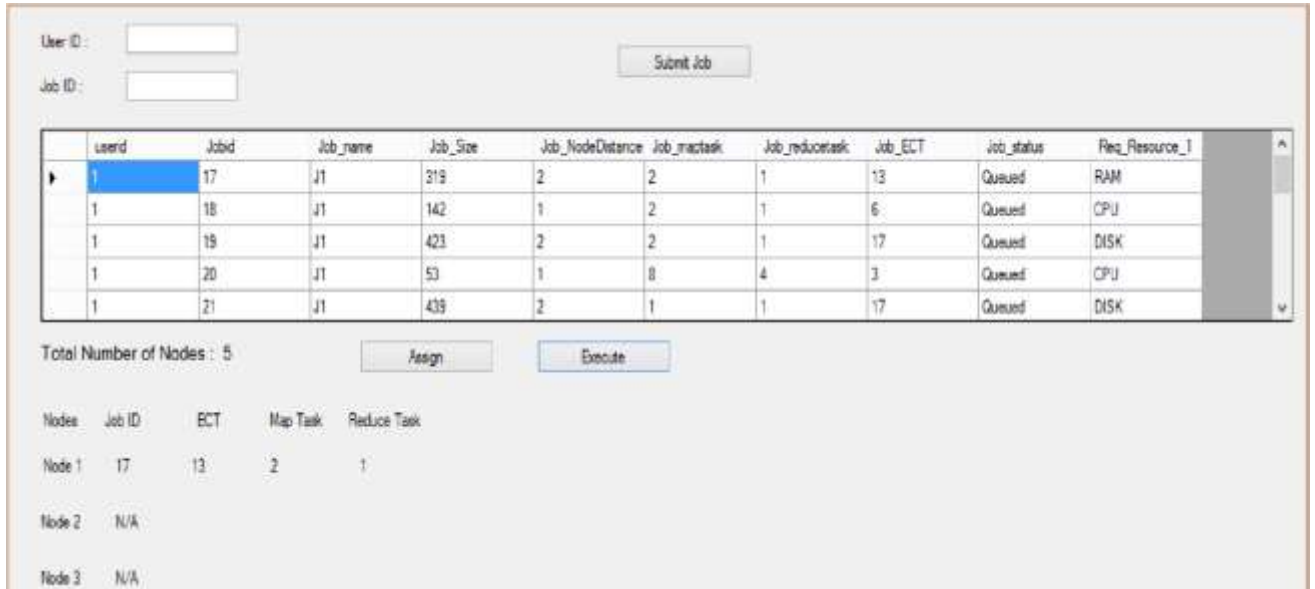


Fig 4.3 FIFO Scheduler Interface

The routing process is triggered by the availability of Heartbeat signal from a free resource. As soon as a heartbeat message is received by the client, this process selects a job for each free available slot of the resource ( $R_j$ ), and sends it to the task scheduler for execution. The job in the routing process are selected for execution in the following ways.

Let there be two  $Job_A$ ,  $Job_B$

- If the Priority ( $A$ ) > Priority ( $B$ ), then  $Job_A$  executes before  $Job_B$
- If the Priority ( $A$ ) < Priority ( $B$ ), then  $Job_A$  executes after  $Job_B$
- However, if Priority ( $A$ ) = Priority ( $B$ ), then the routing process selects a random job for execution from a set of jobs.

This is the problem we try to solve in our process. This means that if there are more than one jobs that have same priority then the task scheduler selects a random job from the execution process. In this case, the completion time of the jobs can vary randomly depending on the selection of the task scheduler.



In this paper, we try to improve the efficiency of the algorithm by adding a sequence in the selection of the jobs in a situation where there are multiple jobs with same priority.

This paper, tries to improve the efficiency of the COSHH algorithm by adding an execution process to select a job from the set of jobs provided by the scheduler. Figure 4.4 shows an ACOSHH algorithm that implements an execution process after the Routing process. The jobs in ACOSHH algorithm are implemented in the following order.

- If Priority (A) > Priority (B), then Execute A before B. □ If Priority (A) < Priority (B), then Execute B before A
- If Priority (A) = Priority (B), then
- If ECT (A) < ECT (B), execute A, else □ Execute B, else if ECT (A) = ECT (B), □ Execute A and B in FIFO.
- If QUEUE\_TIME (A) > BURST\_TIME
- Increment the Priority by 1

The significant highlight of COSHH is that despite the fact that it utilizes modern ways to deal with illuminate the planning problem, it doesn't include impressive overhead. The reason is that first, we constrain the quantity of times needed to do classification, by considering total measures of occupation features (i.e. mean execution time and landing rate). Moreover, since a few occupations in the Hadoop framework are submitted multiple times by clients, these employments don't oblige changing the classification every time that they are submitted.

The burst time of the job depends on the random value generated by the Hadoop scheduler. If the time of the jobs queued in the system exceeds the burst time, then the priority of the job is decreased by one. This prevents the jobs to suffer from resource starvation as everything the jobs exceeds its burst time, its priority is decreased and jobs with lowest priority is executed before jobs with higher priority.

The ACOSHH algorithm thus follows the rules of fairness, minimum share, priority which improves the execution rate of the jobs. The algorithm considers the Estimated Completion time of the task with same priority and the minimum share requirement for the jobs in the queue.

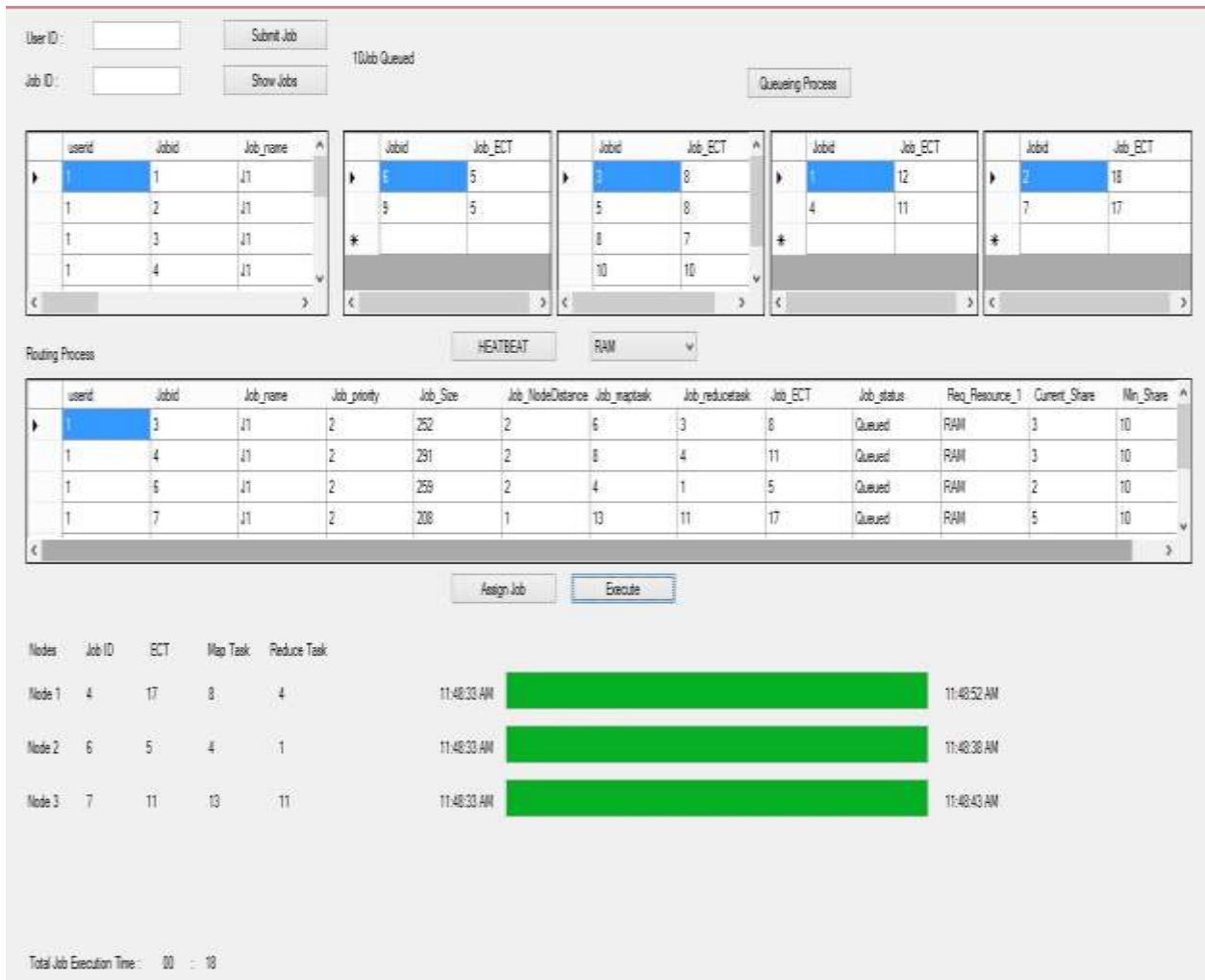


Fig 4.4 ACOSHH Scheduler Interface

## Results

Now, we show that graphical results in order to support our hypothesis. We compare our scheduler with FIFO (in Fig 4.5) and with COSHH (in Fig 4.6). The result in Figure 4.5 show that since priority is not considered in case of FIFO, the jobs are executed as they arrive so the jobs with lower priority is executed before the jobs with higher priority. This causes the jobs to be starved in the queue for a long time before they are executed.

The FIFO algorithm works well in case in case of lower workloads. However, as the workload increases, the execution time of the jobs. The COSHH algorithm solves these problems and according to the work by A. Rasooli, it improves the efficiency of the work by

40% on the original workload. According to her work in the paper, COSHH – A Classification and optimization based scheduler for Heterogeneous Hadoop System the results show that the average completion time is highly improved as compare to FIFO and FAIR share scheduler.

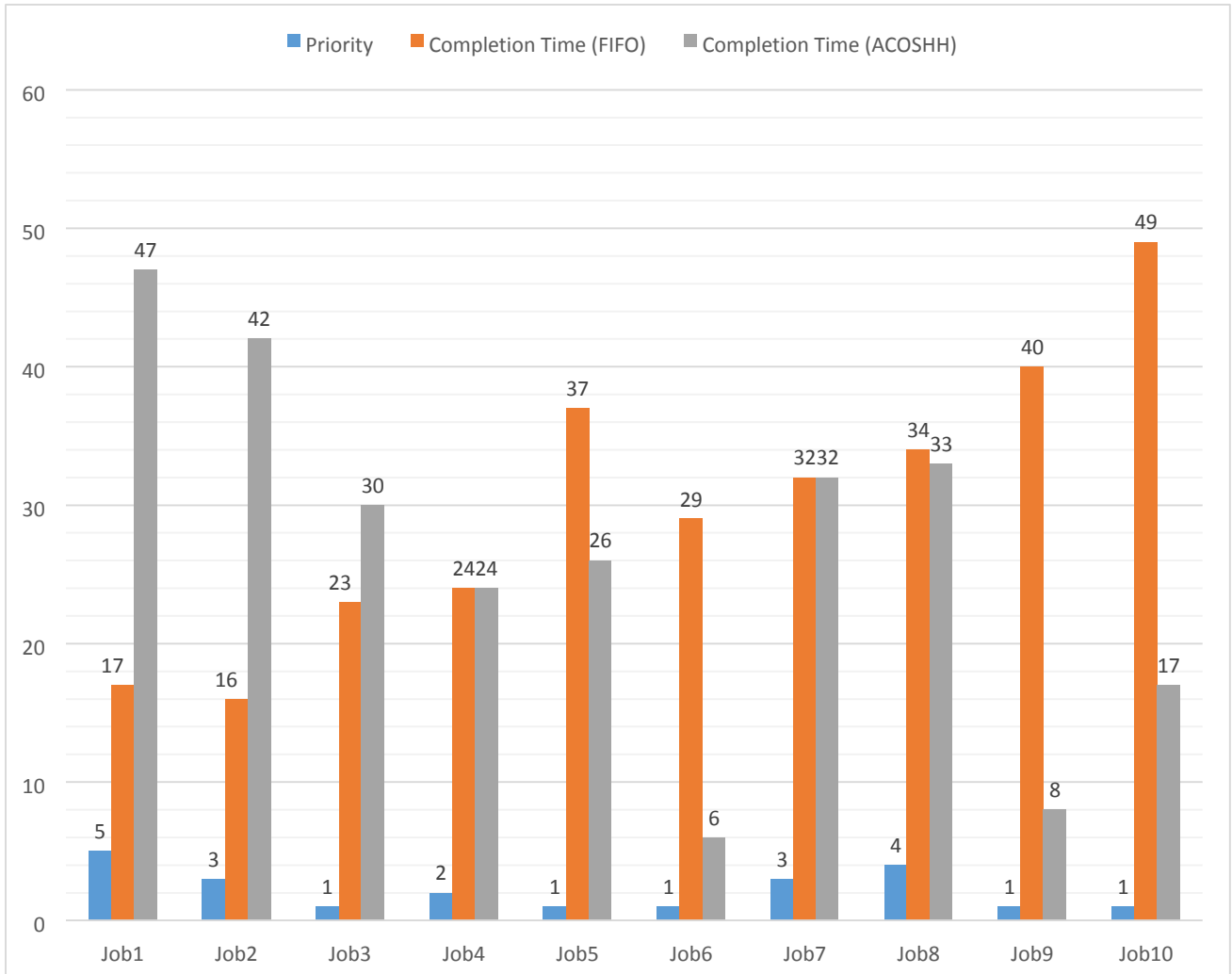


Fig 4.5 Job Completion of FIFO and ACOSHH with respect to Priority

We develop a demonstration of the working of FIFO algorithm in C# and compare the time of execution of the task in case of FIFO and ACOSHH. We capture the results for the execution of ten tasks in FIFO and ACOSHH, Figure 4.5 shows the graphical representation of the tasks in FIFO and ACOSHH scheduling. The graph shows that the FIFO scheduler executes tasks on the basis of first come first server basis. It do not take into account the heterogeneity or

priority of the tasks into consideration. So the tasks are executed in the same order as they arrive. The ACOSHH algorithm executes jobs on the basis of priority and so jobs with higher priority are executed before the jobs with lower priority. The priority of the job is decided based on the SLA of the Client with the Hadoop service provider.

Figure 4.6 shows that difference in the execution rate of the COSHH algorithm with the ACOSHH algorithm. In case of COSHH, the jobs at the routing process are provided based

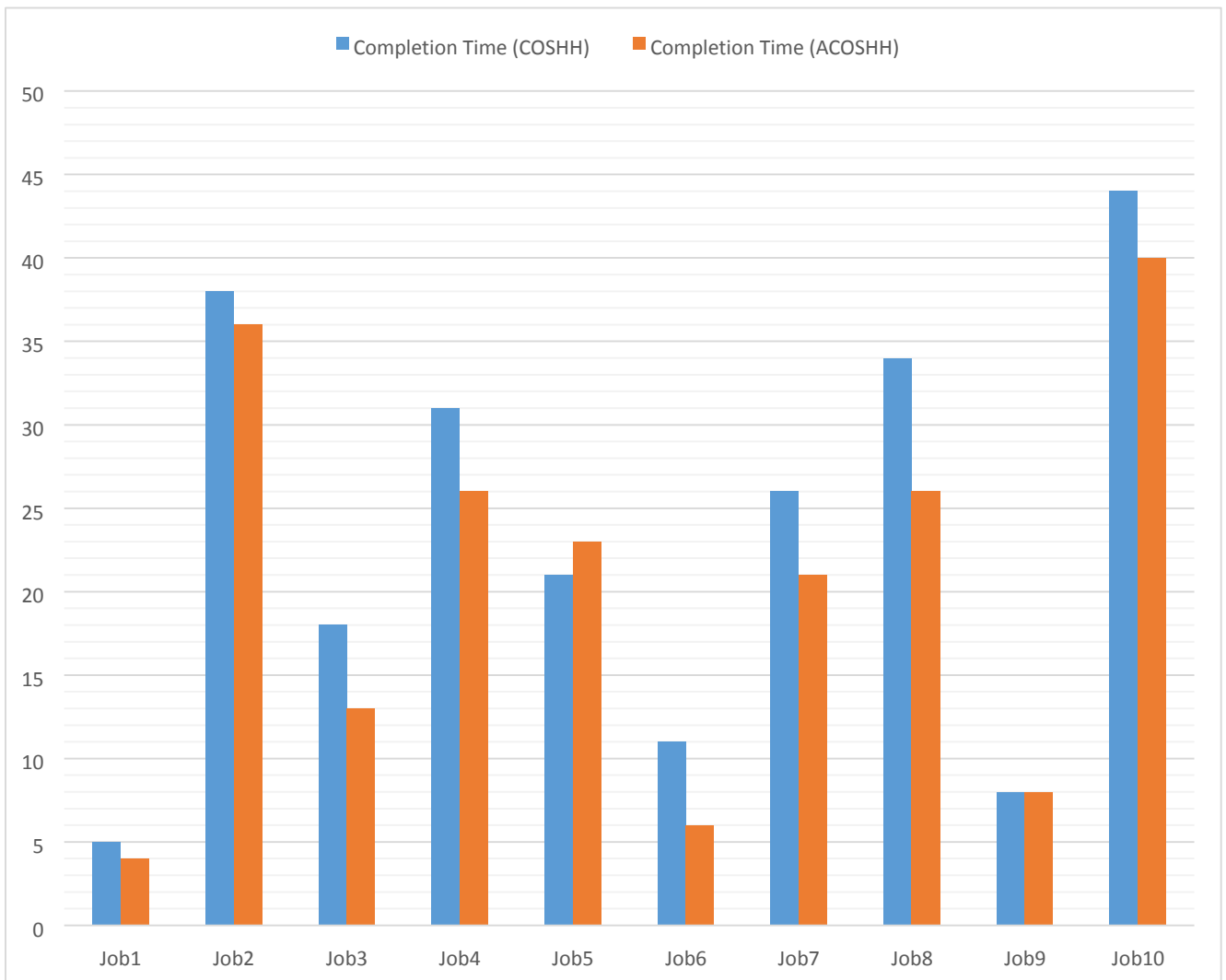


Fig 4.6 Job Completion of COSHH and ACOSHH with respect to Priority

on the job priority and the other factors like minimum satisfaction, number of map and reduce tasks available. However, when the priority of the jobs are same in the queue, the routing process selects a job randomly from the list of jobs. This creates a bottleneck in the working

of COSHH as the jobs with low mean execution time but of same priority has to wait for a long time period for completion. We develop a scheduler that keeps into concentration this bottleneck of the COSHH scheduler and use the estimated completion time provided by the predictor algorithm to schedule the jobs to the task scheduler. Since we know that the estimated time provides an accuracy of 80%. So the tasks with same priority are scheduler on the basis of minimum time needed for execution of task. The compression of results show that the working of ACOSHH scheduler is better COSHH scheduler when there are lots of jobs with the same priority. However when the jobs of different priority are available, the results are same as in COSHH.

## **CHAPTER 5**

### **CONCLUSION**

---

The report presented, introduces in the details about the cloud computing and Hadoop framework. It also explains the architecture of the cloud computing and the components that make up the cloud environment. The various types of Cloud delivery model and cloud deployment models are also summarized.

The report then provides a detailed information of the architecture and working process of the Hadoop –MapReduce. It then discusses various scheduling algorithm along with the problems faced in them and the improvements that were made to solve the drawback.

The report under consideration, then presents a detailed account of the COSHH algorithm. It explains in details the scheduling process of the COSHH with a consideration of its queuing process and the routing process.

The work finally proposes a new algorithm for the improved performance of the COHH. It considers the jobs which have same priority and tries to execute it on the basis of mean estimated completion time of the tasks which has a fairly high accuracy.

## CHAPTER 6

### REFERENCES

---

- [1] I. S. S. Agarwal, "Chronos: A predictive task scheduler for MapReduce," University of California, 2010.
- [2] G. F. K. M. L. K. P. C. S. M. S. S. W. C. Mair, "An Investigation of Machine Learning based Prediction Systems,," *Journal of System and Software*, pp. 23-29, 2000.
- [3] C. C. o. Wikipedia, "en.wikipedia.org/wiki/CloudComputing," [Online].
- [4] C. S. M. D. d. A. J. Y. A. S. S. V. M. P. a. R. B. Yeo, "Utility computing and global grids," 2006.
- [5] M. A. Vouk, "Cloud computing- Issues, Research and Implementation," *Jorunal of computing and Information Technology - CIT 16*, pp. 235-246, 4 2008.
- [6] N. Turner, "Cloud Computing: A Brief Summary." Lucid Communication Limited, 2009.
- [7] D. G. D. Aysan Rasooli, "COSHH: Classification and Optimization based Scheduler for Heterogeneous Hadoop System," *IEEE*, pp. 1-15, 2014.
- [8] J. a. S. G. Dean, "MapReduce: a flexible data processing tool," *ACM*, pp. 72-77, 2010.
- [9] "Apache, Hadoop on demand documentation," (2007). [Online]. Available:  
  
<http://hadoop.apache.org/common/docs/r0.17.2/hod.html>. [Accessed 30 November 2010].
- [10] D. C. J. & Z. W. Wang, "A Task Scheduling Algorithm for Hadoop Platform," *Journal of Computers*, 8(4), pp. 929-936, 2013.

- [11] N. d. o. C. c. NIST, "NIST, NIST definition of Cloud computing," [Online]. Available: [csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc](http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc).
- [12] G. N. ., R. A. L. F. C. M. A. V. A. S. M. B. A. K. F. H. L. L. S. Rich Uhlig, "Intel Virtualization Technology," *IEEE Computer Society*, 2010.
- [13] M. B. D. S. J. S. E. K. S. S. & S. I. Zaharia, "Job scheduling for multi-user mapreduce clusters.," EECS Department, University of California, Berkeley., 2009.
- [14] K. & A. K. Kc, "Scheduling Hadoop jobs to meet deadlines," *IEEE*, pp. 388-392, 2010.
- [15] T. & L. K. Sandholm, "Dynamic proportional Share scheduling in Hadoop," *Springer*, pp. 110-131, 2010 January.
- [16] L. S. S. R. B. Thirumala Rao, "Survey on improved scheduling in Hadoop MapReduce in Cloud Environemnts.," *arXiv:1207 :0780*, 2012.
- [18] H. K. M. K. R. R. L. T. V. L. M. & M. S. Chang, "Scheduling in mapreduce-like systems for fast completion time. In INFOCOM," *IEEE*, pp. 3074-3082, 2011.
- [19] W. Z. K. Y. L. T. J. & Z. L. Wang, "Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality.," *IEEE*, pp. 1609-1617, 2013.
- [20] A. & D. D. G. (. Rasooli, "COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems.," *Future Generation Computer Systems*, pp. 1-15, 2014.
- [21] A. Rasooli, "Improved scheduling in heterogeneous Grid and Hadoop system.," McMaster University, Hamilton, Canada , 2013.
- [22] Y. G. A. G. R. & K. R. Chen, "The case for evaluating MapReduce performance using workload suites. In Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Sympo," *IEEE*, pp. 390-399, 2011 July.



- [23] Y.-J. L. C. Y. D. C. a. B. M. Kyong-Ha Lee, "Parallel Data processing with MapReduce : A Survey," *ACM SIGMoD Record* 40, pp. 11-20, 2012.
- [24] "pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf," 2014. [Online].

## List of Abbreviations

1. SOA ..... Service oriented architecture
2. IAAS ..... Infrastructure as a Service
3. PAAS ..... Platform as a Service
4. GFS..... Google File System
5. HDFS ..... Hadoop Distributed File System
6. SJF ..... Shortest Job First