An Enhanced Tasks Scheduling Based on the Generalized Priority Algorithm in the Cloud Environment

Sanaa Safwat Nooraldaeim Mohamednoor

B.Sc. (Honours) in Computer Science and Information Technology, University of Wad Medani Ahlia (2013)

A Dissertation

Submitted to University of Gezira in Partial Fulfilment of the Requirements for the Degree of Master of Science in Computer Science

Faculty of Mathematical and Computer Sciences

September, 2017
An Enhanced Tasks Scheduling Based on Generalized Priority Algorithm in the Cloud Environment

Sanaa Safwat Nooraldaeim Mohamed Noor

Supervision Committee:

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Murtada Khalfallah Elbashir</td>
<td>Main Supervisor</td>
<td>.............</td>
</tr>
<tr>
<td>Dr. Abdalla Akoud Osman</td>
<td>Co-supervisor</td>
<td>.............</td>
</tr>
</tbody>
</table>

Date: September ,2017
An Enhanced Tasks Scheduling Based on Generalized Priority Algorithm in the Cloud Environment

Sanaa Safwat Nooraldaeim Mohamed Noor

Examination Committee:

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Murtada Khalfallah Elbashir Alfaki</td>
<td>Chair Person</td>
<td></td>
</tr>
<tr>
<td>Dr. Mohammed Babiker Ali Mohamed</td>
<td>External Examine</td>
<td></td>
</tr>
<tr>
<td>Dr. Awadallah Mohammed Ahmed Ali</td>
<td>Internal Examiner</td>
<td></td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

The work presented in this dissertation would have not been possible without the support, advice and encouragement received from a multitude of people to whom I would like to express my most sincere gratitude.

First and foremost, I would like to thank my supervisor, Dr. Murtada Khalafallah Elbashir Al Faki, for giving me the opportunity to pursue this research and for constantly providing me with the encouragement, advice, freedom.

I am also very grateful to Dr. Abdalla akoud Osman for countless discussions and insightful advice that guided my work and helped shaping this dissertation.

Lastly, and most importantly, I am deeply grateful to my family, in particular my mother and father and my brother hamza. They offered me constant encouragement and support throughout all my education and have been a major source of inspiration for all my achievements.
DECLARATION

My parents: I wish to thank for their tremendous contribution and support both morally and financially towards the completion of this project.

My family: Thank you for believing in me; for allowing me to further my studies. Please do not ever doubt my dedication and love for you.

My friends: Thank you to those who helped me in my career to complete this research.
An Enhanced Tasks Scheduling Based On Generalized Priority Algorithm in the Cloud Environment

Sanaa Safwat Nooraldaeim Mohamed Noor

Abstract

Cloud Computing is a new paradigm in IT. It provides pervasive, on demand, Broad Network Access, Resource Pooling. Services provided by the cloud providers are data storage. With the fusion of network technology and traditional computing technology such as distributed computing, parallel computing and grid computing. A cloud computing product is formed. A number of issues are confronted like heavy load or traffic while computation. Cloud provides many facilities due to its vast area such as sharing of resources for different purposes. The primary aim of cloud computing is to give most suitable access to remote scattered resources. A good scheduler fits its scheduling policy according to the varying situation and the type of task. Scheduling problem considered in the existing Shortest Job First algorithm, the long jobs may wait long time to be executed, also the Generalized Priority Algorithm executes tasks according to their size such that one having the biggest size has the highest rank. In Generalized Priority Algorithm the processors are also ranked (prioritized) according to their MIPS (million instruction per second) value such that the one having highest MIPS has the highest rank. Hence, the need to improve it to mitigate the delay in executing tasks with minimum size. This research proposed new hybrid algorithm that combines between Generalized Priority Algorithm and Shortest Job First algorithm, the tasks are divided into two lists according to the condition if the size of task is greater than 10000 then the task will be assigned to big list otherwise will be assign to small list. The big lists contains the task with big or high size sorted according the size of the task in decreasing order and the processor also is ordered according to their MIPS. The small list contains the task with short or small size after that sort the list according to the burst time in increasing order. The hybrid proposed algorithm has the minimum processing time in comparison with Shortest Job First Algorithm and Generalized Priority Algorithm about 5%. The comparison has been done for Twelve, fourteen, fifteen, sixteen, eighteen tasks in terms of processing time, average waiting time and the average turnaround time using five, ten, fifteens processors. This research recommends to consider the response time for further improvement.
جدولة مهام محسنة بناءً على خوارزمية الأولوية المعممة في بيئة الحوسبة السحابية

سناء صفوت نورالدائم محمدنور

ملخص الدراسة

الحوسبة السحابية هي نموذج جديد في تكنولوجيا المعلومات، توفر شبكة واسعة الإنتشار ويتم الوصول للخدمات عندطلب وتسمح أيضاً بمشاركة الموارد لعدد كبير من المستخدمين. وأيضاً، يوفر مووزد الحوسبة السحابية خدمة تخزين البيانات مع مزيج من تكنولوجيا الشبكة وتكنولوجيا الحوسبة التقليدية مثل الحوسبة الموزعة والحوسبة المتوازية والحوسبة الشبكية. يتم تشكيل منتج الحوسبة السحابية وتواجد عدد من القضايا مثل الحمل الزائد. وهي توفر العديد من التسهيلات بسبب مساحتها الكبيرة مثل تقاسم المواد لأغراض مختلفة. والهدف الرئيسي للحوسبة السحابية هو توفير الوصول الأسرع للموارد الموزعة عن بعد، ويتحقق ذلك بالجدولة الجيدة التي تتناسب وفقاً للتغييرات وتوعي المهمة. هناك حاجة أيضاً لتحسين وتعزيز الجدولة لتحسين أداء الخدمة وتقليل الزمن الكلي لتنفيذ المهمة. وتتمثل مشكلة الجدولة في خوارزمية المهمة القصيرة أولاً، قد تستغرق المهام الطويلة فيها زمناً أطول للتنفيذ وأيضاً، هذه الخوارزمية الأولوية المعممة تتنفيذ المهام وفقاً لحجمها مثل المهمة ذات الحجم الكبير تنفذ أولاً، والمعالجة أيضاً يرتب تنفيذها وفقاً لعدد التعليمات في الثانية الواحدة (MIPS) وبالتالي هناك حاجة لتصنيفها والتخفيف من تنفيذ المهام ذات الحجم الصغير. ويهدف البحث لعمل خوارزمية هجينية مقترحة وتمتلك منهجية البحث في تقسيم المهام إلى قائمة وفقاً للشرط إذا كان حجم المهمة أكبر من 10000 وضافت إلى القائمة الكبيرة وإلا إلى القائمة الصغيرة. القوائم الكبيرة تحتوي على المهام ذات الحجم الكبير وترتيب تنفيذها تنزلباً، والمعالجة أيضاً لحجم المهمة وعدد التعليمات التي يعالجها المعالج في الثانية والقائمة الصغيرة تحوي على المهام صغيرة الحجم وترتيب تصاعداً، وفقاً لزمن التنفيذ، الخوارزمية الهجينية المقترحة تقلل من زمن المعالجة مقارنة مع خوارزمية المهمة القصيرة أولاً وخوارزمية الأولوية المعممة حوالى 5%. وقد أجبرت المقارنة لأنثى عشر أربعة عشر، ستة عشر وثمانية عشر مهمة من حيث زمن المعالجة، والمعالج المستخدمة هي متوسط زمن المعالجة، ومتوسط زمن الأنتظار، ومتوسط زمن الإستجابة باستخدام خمسة عشر، وخمسة عشر من المعالجات ويوصى هذا البحث بوضع زمن الإستجابة في الاعتبار.
# TABLE OF CONTENTS

Acknowledgements .............................................................................................................. iv

Declaration .................................................................................................................................

Abstract ....................................................................................................................................... vi

Table of Contents ........................................................................................................................ viii

List of Tables .............................................................................................................................. 10

List of Figures ............................................................................................................................. 11

List of Abbreviations ................................................................................................................. 12

## CHAPTER ONE

1.1 Introduction .......................................................................................................................... 1

1.2 Problem Statement ............................................................................................................. 4

1.3 Research Objectives .......................................................................................................... 4

1.4 Research Methodology ...................................................................................................... 4

1.5 Research Organization ...................................................................................................... 4

## CHAPTER TWO  LITERATURE REVIEW ........................................................................... 5

2.1 Background ......................................................................................................................... 5

2.2 Cloud Computing Characteristics ..................................................................................... 6

2.3 Cloud Service Model ......................................................................................................... 6

2.4 Cloud Deployment Model ................................................................................................... 6

2.5 Task Scheduling Model ..................................................................................................... 9
2.6 Task Scheduling Goals ................................................................. 9
2.7 Task Scheduling Characteristics .............................................. 11
2.8 Task Scheduling Classification ................................................ 12
2.9 Cloud Computing Challenges ................................................ 13

CHAPTER THREE  RESEARCH METHODOLOGY ......................... 18
3.1 Introduction ................................................................................. 18
3.2 The Algorithms ......................................................................... 19
  3.2.1 ShortJob First........................................................................ 19
    3.2.1.1 Drawbacks: ................................................................. 19
  3.2.2 Generalized Priority Algorithm ............................................ 20
    3.2.2.1 Drawbacks: ................................................................. 21
  3.2.3 The Proposed Hybrid Algorithm ........................................... 22
3.3 Experimental setups Model ....................................................... 25.
  3.3.1 Scenarios .............................................................................. 26

CHAPTER FOUR  RESULTS AND DISCUSSION ........................................ 27
4.1 Introduction ................................................................................. 27
4.2 Scenario One .............................................................................. 27
4.3 Scenario Two .............................................................................. 30
4.4 Scenario Three .......................................................................... 33

CHAPTER FIVE  CONCLUSIONS AND RECOMMENDATIONS ........... 37
5.1 Conclusions ............................................................................... 37
5.2 Recommendations ...................................................................... 37
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1 Related Work comparative Table</td>
<td>17</td>
</tr>
<tr>
<td>Table 4.1 Comparative according to processing time and 5 processors</td>
<td>29</td>
</tr>
<tr>
<td>Table 4.2 Comparative according to processing time and 10 processors</td>
<td>30</td>
</tr>
<tr>
<td>Table 4.3 Comparative according to processing time and 15 processors</td>
<td>31</td>
</tr>
<tr>
<td>Table 4.4 Comparative according to Average waiting time and 5 processors</td>
<td>32</td>
</tr>
<tr>
<td>Table 4.5 Comparative according to Average waiting time and 10 processors</td>
<td>33</td>
</tr>
<tr>
<td>Table 4.6 Comparative according to Average waiting time and 15 processors</td>
<td>34</td>
</tr>
<tr>
<td>Table 4.7 Comparative according to Average turnaround time and 5 processors</td>
<td>35</td>
</tr>
<tr>
<td>Table 4.8 Comparative according to Average turnaround time and 10 processors</td>
<td>36</td>
</tr>
<tr>
<td>Table 4.9 Comparative according to Average turnaround time and 15 processors</td>
<td>37</td>
</tr>
<tr>
<td>Figure No</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Cloud computing Services Mode</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Deployment Model for Cloud</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Task Scheduling In Cloud</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Stages of Scheduling</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Shortest Job First Algorithm Flowchart</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Generalized Priority Algorithm Flowchart</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Proposed Hybrid Algorithm Flowchart</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Experimental setups model</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Comparative according to processing time and 5 processors</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Comparative according to processing time and 10 processors</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Comparative according to processing time and 15 processors</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Comparative according to Average waiting time and 5 processors</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Comparative according to Average waiting time and 10 processors</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Comparative according to Average waiting time and 15 processors</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Comparative according to Average turnaround time and 5 processors</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Comparative according to Average turnaround time and 10 processors</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Comparative according to Average turnaround time and 15 processors</td>
</tr>
</tbody>
</table>
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWT</td>
<td>Average waiting Time</td>
</tr>
<tr>
<td>ATT</td>
<td>Average turnaround time</td>
</tr>
<tr>
<td>BMHA</td>
<td>Heuristic scheduling algorithms</td>
</tr>
<tr>
<td>FCFS</td>
<td>First Come First Serve</td>
</tr>
<tr>
<td>GPA</td>
<td>Generalized Priority Algorithm</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>MIPS</td>
<td>Million Instruction Per Second</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PT</td>
<td>Processing Time</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SJF</td>
<td>shortest-Job-First</td>
</tr>
<tr>
<td>VMs</td>
<td>Virtual machines</td>
</tr>
</tbody>
</table>
CHAPTER ONE

1.1 INTRODUCTION

Cloud computing is defined as a concept that is based on the separation of applications and operating systems from hardware, or it is a term that refers to the sources and computer systems that are available upon request. Also, it can provide a number of integrated computing resources without the restriction of the local resources (standard hardware).

As known, the computing moves in the direction of change by moving applications and documents from the desktop to the cloud or what is known as cloud computing. The applications and files are hosted on the cloud which is made up of thousands of computers and servers linked together and can be accessed via the internet. With cloud computing all business will be on the internet rather than on the desktop. We can also have access to all files, documents and programs from any computer connected to the internet.

Cloud computing aims to change the way required to manage the business, by taking the business to anywhere, because the business can be accessed via the web. In addition, the cloud computing facilitates teamwork where all team members have the ability to access the same programs and files from anywhere.

There are many benefits of cloud computed by different researchers which makes it more preferable to be adopted by enterprises. Cloud Computing infrastructure allows enterprises to achieve more efficient use of their IT hardware and software investments. This is achieved by breaking down the physical barrier inherent in separate systems, automating the management of the group of the systems by making it a single entity. Cloud Computing can also be described as ultimately virtualized system and a natural evolution for data centers which offer automated systems management. (Rehan Saleem, 2011).

According to The National Institute of Standards and Technology’s (NIST), cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” (Thomas, G. Krishnalal, 2014).

The core idea behind the cloud computing concept is not a new one. Cloud computing is an evolution of previous computing models starting from grid computing, utility
computing, and finally software as a service. It brings together a set of existing technologies such as virtualization and utility-based pricing, and leverages these existing technologies to meet the technological demand for solving variety of technological problems and issues (Salot, Pinal, 2013).

Cloud computing is an internet based computing that delivers infrastructure as a service (IaaS), platform as a service (PaaS), and software as services (SaaS). In SaaS, software application is made available by the cloud provider. In PaaS an application development platform is provided as a service to the developer to create a web based application. In IaaS computing infrastructure is provided as a service to the requester in the form of Virtual Machine (VM). These services are made available on a subscription basis using pay as-you-use model to customers, regardless of their location. Cloud Computing still under its development stage and has many issues and challenges out of the various issues in cloud scheduling plays very important role in determining the effective execution. (A. Agarwal and S. Jain, 2014)

1.3 PROBLEM STATEMENT

Traditional job scheduling algorithms are not able to provide scheduling in the cloud environments (Salot, Pinal, 2013). This leads to server overflow. Therefore, the resources may not be utilized perfectly. Hence, the execution time will increase. Thus, there is a need for a better and enhanced scheduling technique to enhance the performance of clouds and minimize the total processing time, average waiting time and average turnaround time of the tasks.

1.4 RESEARCH OBJECTIVES

The main objective of this research is to develop optimistic a new algorithm that can minimize processing time of tasks, the detailed objectives are listed as below:

1. To minimizing CPU-intensive.
2. To minimizing waiting and turnaround time.
3. To maximizing CPU utilization.
4. To maximizing throughput.
RESEARCH SCOPE

This research aims to propose a new scheduling algorithm that can minimize the processing time, average waiting time and average turnaround time of the tasks with the valuable use of resources the research is applied with assumptions such as:

1. The main concern in this research is to deal with scheduling algorithms. This type of algorithm minimizes processing time with the valuable use of resources.
2. The research will focus on resource utilization, reducing processing time, average waiting time and average turnaround time of the tasks.
3. The proposed algorithm is a dynamic and distributed scheduling that include in Heuristic algorithms in Batch mode.
4. The performance of the proposed algorithm will be measured using java tool.
5. The proposed algorithm will be compared and evaluates with algorithm depend on processing time, average waiting time and average turnaround time of the tasks.

1.5 Research Methodology and Tools

The proposed hybrid algorithm tasks is divided into two list according to the condition if the size of task greater than 10000. Then the task will assign to big list else assign to small list the big list is contain the task with big or highest size sorted according the size of the task in decreasing order and the processor also is ordered according to their MIPS and small list contains the task with short or small size after that sort the list according to the burst time in increasing order. The performance of the proposed method is tested and experimented, to evaluate the proposed algorithm, it was compared with previous algorithms before hybrid.
1.6 RESEARCH ORGANIZATION

The research was organized as follows:

Chapter two includes a background and literature review of all candidate topics to solve the problem as well as related works.

Chapter three describes the methodology that will be followed to accomplish this research.

Chapter four contains the results and their discussions.

Chapter five contains the conclusion and recommendations of the research.

Finally there are reference list and appendixes.
CHAPTER TWO

Background and Literature review

The development of cloud computing technology is currently at its infancy, with many issues still to be addressed such as scheduling. In this chapter there will be a definition cloud computing and essential characteristics and present its types and services levels. In addition there will be reference scheduling issues in cloud computing and specifically the focus on scheduling Processing Time to minimize the processing time and maximize the utilization of resource. Also discussion some related work done on this research area.

2.1 BACKGROUND

Cloud computing is a construct that allows you to access applications that actually reside at a location other than your computer or other internet-connected device. It has become one of the most talked about technologies in recent times and has got lots of attention from media as well as analysts because of the opportunities it offers. The success of clouds has been driven in part by the use of virtualization as their underlying technology. It is a technology that allows running two or more operating systems side-by-side on just one PC or embedded controller [scheduling]. (R. Kaur and S. Kinger, 2014)

Virtualization greatly helps in effective utilization of resources and builds an effective system. Many applications are having a limited number of concurrent tasks, thus having a number of idle cores. This problem can be solved by using virtualization, allocating a group of cores to an OS (Operating system) that can run it concurrently. It enables the service providers to offer virtual machines for work rather than the physical server machines. Virtual machines (VMs) provide flexibility and mobility through easy migration, which enables dynamic mapping of VMs to available resources (F. Teng, 2012).

2.2 CLOUD COMPUTING CHARACTERISTICS

There are five essential characteristics of Cloud computing which we explain their relation and difference from the traditional computing. (Rehan Saleem, 2011).

1. On-demand-self-service: a consumer can provision or un-provision the services when needed, without the human Interaction with the service provider.

2. Broad Network Access: It has capabilities over the network and accessed through standard mechanism.
3. Resource Pooling: The computing resources of the provider are pooled to serve multiple consumers which are using a multi-tenant model, with various physical and virtual resources dynamically assigned, depending on consumer demand.
4. Rapid Elasticity: Services can be rapidly and elastically provisioned.
5. Measured Service: Cloud Computing systems automatically control and optimize resource usage by providing a metering capability to the type of services (e.g. storage, processing, bandwidth, or active user accounts). (F. Teng, 2012).

2.3 CLOUD SERVICE MODEL

User accesses the cloud services through internet by using Mobile, PC and PDA. These services are:

1. Infrastructure as a Service (IaaS) refers to the sharing of hardware resources for executing services, typically using virtualization technology.
2. Platform as a service (PaaS) approach where offering includes a software execution environment, such as an application server.
3. Software as a Service (SaaS) complete applications are hosted on the Internet so that e.g. your word processing software isn’t installed locally on your PC anymore but runs on a server in the network and is accessed through a web browser (N. Khangahi and R. Ravanmehr, 2013).
2.4 CLOUD DEPLOYMENT MODEL

Cloud is deployed in three forms depending on the access allowed to the users. Clouds systems can be deployed in different fashions. There are four primary cloud deployment models (R. K. Mondal, E. Nandi, and D. Sarddar, 2015).

1. **Public Cloud**

   Standard to cloud computing paradigm, in which a service provider makes resources, such as applications and storage, available to the general public over Internet. Service providers charge on a fine-grained utility computing basis. Examples of public clouds include Amazon Elastic Compute Cloud (EC2), IBM’s Blue Cloud, Sun Cloud, Google App Engine and Windows Azure Services Platform (R. K. Mondal, E. Nandi, and D. Sarddar, 2015).

2. **Private Cloud**

   It looks more like a marketing concept than the traditional mainstream sense. It describes a proprietary computing architecture that provides services to a limited number of people on internal networks. Organizations needing accurate control over their data will prefer private cloud, so they can get all the scalability, metering, and agility benefits of a public cloud without ceding control, security, and recurring costs to a service provider. Both eBay and HP Cloud Start yield private cloud deployments (R. K. Mondal, E. Nandi, and D. Sarddar, 2015).

3. **Hybrid Cloud**

   It uses a combination of public cloud, private cloud and even local infrastructures, which is typical for most IT vendors. Hybrid strategy is proper placement of workloads depending upon cost and operational and compliance factors. Major vendors including HP, IBM, Oracle and VMware create appropriate plans to leverage a mixed environment, with the aim of delivering services to the business. Users can deploy an application hosted on a hybrid infrastructure, in which some nodes are running on real physical hardware and some are running on cloud server instances (R. K. Mondal, E. Nandi, and D. Sarddar, 2015).
2.5 TASK SCHEDULING MODEL:-

Though cloud computing has received considerable attention and has emerged out as a promising approach and a commercial reality in the information technology domain, however there are still many issues that need more attention. Task scheduling is one such fundamental issue that needs to be addressed to enhance the cloud performance. Task scheduling is actually the efficient allocation of resources to the required jobs under the constraint of the Service Level Agreements (SLAs). It is one of the most prominent activities executed in the cloud computing environment to get maximum profit(K. Rana, 2015)
2.6 TASK SCHEDULING GOALS

The job scheduling in Cloud computing is a mapping mechanism from users’ tasks to the appropriate selection of resources and its execution. Specific goals of job scheduling include load balance, quality of service (QoS), economic principle, and optimal operation time and system throughput.

- Load Balancing: It is an important measure in cloud and is closely linked to job scheduling. It helps to distribute all loads between all the nodes and ensures that every computing resource is distributed efficiently and fairly. It helps in preventing bottlenecks of the system which may occur due to load imbalance. It is a relatively new technique that provides high resource utilization and better response time.
- Quality of Service: Cloud mainly aims to provide users with computing and cloud services and resources to the users so as to achieve quality of service.
- Throughput: Optimized performance with task scheduling can be measured by throughput. Task scheduling aims to achieve high throughput (K. Rana, 2015).
2.7 THE TASK SCHEDULING CHARACTERISTICS:

1. Job scheduling is global centralized: - As cloud computing is a computing model which supply the centralized resource by the mirror service to multiple distributed applications, and this mirroring deployment can make heterogeneous procedures executing of interoperate become easier, which used to be difficult to deal with. Therefore, virtualized technology and mirroring services make the task scheduling of cloud computing achieve a global centralized scheduling.

2. Each node in the cloud is independent: - In cloud computing, the internal scheduling of every cloud node is autonomous, and the schedulers in the cloud will not interfere with the scheduling policy of these nodes.

3. The scalability of job scheduling: - The scale of resources supply from cloud provider may be limited in early stages. With the addition of a variety of computing resources, the size of the abstract virtual resources may become large, and the application demand continues increasing. In the cloud, task scheduling must meet the scalability features, so that the throughput of the task scheduling in the cloud may not be too low.

4. Job scheduling can be dynamically self-adaptive Expanding and shrinking applications in the cloud may be necessary depend on the requirement. The virtual computing resources in cloud system may also expand or shrink at the same time. The resources are
constantly changing, some resources may fail, new resources may join in the clouds or restart.

5. The set of job scheduling - Task scheduling is divided into two parts: one is used as a unified resource pool scheduling, and primarily responsible for the scheduling of applications and cloud API; the other is for the unified port resource scheduling in the Cloud, for example, Map Reduce task scheduling. However, each scheduling consists of two two-way processes that are scheduler leases resource from cloud and scheduler callbacks the requested resources after use. The former process is scheduling strategy and the latter one is call back strategy. The combination of the scheduling and call back resource strategy is the set of task scheduling. (R. Kaur and S. Kinger, 2014).

2.8 TASK SCHEDULING CLASSIFICATION:
Classifying scheduling methods in cloud environment generally into three groups: resource scheduling, workflow scheduling and task scheduling. Task scheduling methods may be centralized or distributed. It can be performed in homogeneous or heterogeneous environment on dependent or independent tasks. Scheduling methods in distributed environment can be of two types: heuristic and hybrid techniques. Heuristic methods are classified into static as well as dynamic scheduling. Static scheduling: all the tasks are known a priori to scheduling and they are statically assigned to virtual resources. Dynamic scheduling: can be performed in: online mode or batch mode (T. Mathew, 2014).

2.8.1 Batch mode:
Jobs are queued and collected into a set when they arrive in the system. The scheduling algorithm will start after a fixed period of time. The main examples of Batch mode heuristic scheduling algorithms (BMHA) based algorithms are:

A. First Come First Serve Algorithm: Job in the queue which come first is served. This algorithm is simple and fast.

B. Round Robin algorithm: In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum. If a process does not complete before its CPU-time expires, the CPU is pre-empted and given to the next process waiting in a queue. The pre-empted process is then placed at the back of the ready list. (S. Mohapatra and K. S. Rekha, 2014).
C. Min–Min algorithm: This algorithm chooses small tasks to be executed firstly, which in turn large task delays for long time (D. Kaur Singh, 2014).

Max–Min algorithm: This algorithm chooses large tasks to be executed firstly, which in turn small task delays for long time (R. M. Singh, 2014).

D. Max–Min algorithm: This algorithm chooses large tasks to be executed firstly, which in turn small task delays for long time (K. Rana, 2016).

E. Most fit task scheduling algorithm: In this algorithm task which fit best in queue are executed first. This algorithm has high failure ratio.

F. Priority scheduling algorithm: The basic idea is straightforward: each process is assigned a priority, and priority is allowed to run. Equal-Priority processes are scheduled in FCFS order (S. Mohapatra and K. S. Rekha, 2013).

The shortest-Job-First (SJF) algorithm is a special case of general priority scheduling algorithm. An SJF algorithm is simply a priority algorithm where the priority is the inverse of the (predicted) next CPU burst. That is, the longer the CPU burst, the lower the priority and vice versa. Priority can be defined either internally or externally. Internally defined priorities use some measurable quantities or qualities to compute priority of a process (Salot, Pinal, 2013).

2.9 CLOUD COMPUTING CHALLENGES:

Job scheduling is an important task in cloud environment. Job Scheduling is used to allocate certain jobs to particular resources in particular time. In cloud computing, job-scheduling problem is a biggest and challenging issue. The main aim of job scheduling algorithm is to improve the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs and reduce the execution cost. An efficient job scheduling strategy must aim to yield less response time so that the execution of submitted jobs takes place within a possible minimum time (F. Mohammadi, S. Jamali, and M. Bekravi, 2014).

Scheduling is a critical problem in Cloud computing, because a cloud provider has to serve many users in Cloud computing system. So scheduling is the major issue in establishing Cloud computing systems. The scheduling algorithms should order the jobs in a way where balance between improving the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs. A good scheduling technique also helps in proper and efficient utilization of the resources (F. Mohammadi, S. Jamali, and M. Bekravi).
2.2 RELATED WORKS

Many research works have been introduced to enhance the performance of the scheduling algorithms in cloud computing and to achieve high throughput. The following section presents some of the most relevant research works.

In 2011 Mousumi Paula, Goutam Sanyalb “Task-Scheduling in Cloud Computing using Credit Based Assignment Problem” defines a new probabilistic measurement or credit of the tasks for allocation of the resources. This measurement depends upon the arrival time and the execution time of the task over the available resource. The consequence of this parameters leads to the minimum completion time. The cost matrix is used for the assigning the tasks to the resources. The task which have high probability to get resource and resource which fits better for the task are allocated (Mousumi Paula, 2011).

In 2012 (Rajput, Ishwari Singh, and Deepa Gupta), developed new scheduling algorithm priority based round robin algorithm and also done its comparison in terms of waiting and turnaround time and CPU Utilization of jobs with algorithm simple round robin. The proposed algorithm performs better than simple round robin for varying waiting and turnaround time and CPU Utilization. The proposed architecture focuses on the drawbacks of simple round robin architecture, which gives equal priority to all the processes, However, the results show an increment in waiting time and response time of the processes, which results in decreasing the system throughput. In addition, the proposed architecture eliminates the defects of implementing simple round robin architecture (Rajput, Ishwari Singh, and Deepa Gupta, 2012).

In December 2012 Pinky Rosemarry1, Ravinder Singh2, Payal Singhal3 and Dilip Sisodia proposed an efficient three scheduling algorithm (Shortest Job First, First Come First Serve, Round Robin) for jobs and send to the queue. We have got some better performance in terms of processing time than job scheduling on the FIFO algorithm. Also they have implemented Shortest Job First algorithm along with all three algorithms for performance analysis. The simulation results have shown that the proposed model is able to achieve the mentioned objectives in grid environment. The comparative study also shows that the proposed hybrid algorithm gives better performance than shortest job first algorithm alone in terms of processing time (Pinky Rosemarry1, Ravinder Singh2, Payal Singhal3 and Dilip Sisodia, 2012)
In 2012 RaghavendraAchar, P. SanthiThilagam “Optimal Scheduling of Computational Task in Cloud using Virtual Machine Tree” proposed novel scheduling mechanism, first priorities the tasks and Virtual Machines and then defines a tree based data structure called Virtual Machine Tree (VMT) in which every nodes of a tree represents a Virtual Machine. The clustering of task is done based on number of leaves nodes in the VMT. The modified DFS algorithm identify the suitable Virtual Machines, for which the assigned tasks be executed. Comparison of this algorithm with the FCFS and priority based algorithm result represent that the proposed algorithm is much efficient than priority based algorithms and FCFS (RaghavendraAchar, P. SanthiThilagam, 2012).

In 2013 R. Vijayalakshmi, Mrs. Soma Prathibha “A Novel Approach for Task Scheduling in Cloud” focuses on minimizing the makespan to utilization of allocated resources more efficiently. The user’s tasks first got prioritized and then submit to the virtual machine on the basis of the prioritization. The highest prioritized task is scheduled on the largest processing powered virtual machine. Comparison of this algorithm with the FCFS result shows that the proposed algorithm is more efficient than FCFS algorithm (R. Vijayalakshmi, Mrs. Soma Prathibha, 2013).

In 2013 Dr Ajay Jangra, TusharSaini “Scheduling Optimization in Cloud Computing” define computation regarding the user’s tasks is done with the help of various parameters and conditions to be encountered in the simulation of tasks. The user’s tasks are classified on the accordance of data and resources requested by the task and then prioritized. Selection of resources is done on the accordance of its turnaround time and cost using greedy approach. A priority formula is used for the task selection. This way of task selection and resource selection gives better results over sequential scheduling. This method gives a way to more future findings in the scheduling techniques in a cloud environment (Dr Ajay Jangra, TusharSaini, 2013).

In 2014 (Kaur, Rajveer, and SupriyaKinger), presented a comparison between six task scheduling algorithms (FCFS, SJF, Priority Algorithm, Round Robin, Genetic Algorithm) basis of complexity, allocation, waiting time and type of system. The complexity defines which type of algorithm is simple or easy to use in processing. The allocation defines how the jobs are assigned to the resources. Waiting Time defines which of the algorithm
takes more time for processing. The type of system defines which algorithm is suitable for which type of system. The result obtained shows Genetic Algorithm achieves the best performance((Kaur, Rajveer, and SupriyaKinger,2014).

In 2014 Santhosh, Dr. Manjaiah D.H “An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing” modified improved Max-Min algorithm to define two new algorithms based on the average execution time. In this largest task just greater than the average execution time is selected and assigned to the resource which gives minimum completion time. The average execution time is calculated using arithmetic mean for independent tasks and geometric mean for dependent tasks. The experimental comparison of these two algorithms with the Max-Min and Improved Max-Min represents that the proposed algorithms are more efficient for minimization of the makespan(Santhosh, Dr. Manjaiah,2014).

In 2015 (Ranjan Kumar Mondal1, Enakshmi Nandi2 and DebabrataSarddar) , we have compare the result of proposed SJF algorithm with the existing RR, FCFS algorithm using java language. The mechanism of Shortest Job First Algorithm is, to schedule the process with the shortest time to completion first, thus providing high efficiency and low turnaround time. Shortest Job First (SJF) scheduling algorithm can be said to be optimal with an average waiting time is minimal, which helps to improve the system performance (Ranjan Kumar Mondal1, Enakshmi Nandi2 and DebabrataSarddar,2015).

In 2016 Ranjan Kumar Mondal1, Enakshmi Nandi2 and DebabrataSarddar) We have created FCFS, SJF scheduling Algorithm and new proposed Scheduling algorithm is (GPA) generalized .In the proposed strategy, the tasks are initially prioritized according to their size such that one having highest size has highest rank. The Virtual Machines are also ranked (prioritized) according to their MIPS value such that the one having highest MIPS has the highest rank. Thus, the key factor for prioritizing tasks is their size and for VM is their MIPS. The experiment conducted is compared with FCFS and Round SJF. The result shows that the proposed algorithm is more efficient than FCFS and SJF algorithm Ranjan Kumar Mondal1, Enakshmi Nandi2 and DebabrataSardda,2016r.)
<table>
<thead>
<tr>
<th>NO</th>
<th>Year, Author</th>
<th>Research Work</th>
<th>Methodology</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mousumi Paula, GoutamSanyalb, 2011</td>
<td>Task-Scheduling in Cloud Computing using Credit Based Assignment Problem</td>
<td>new probabilistic measurement this measurement depends upon the arrival time and the execution time of the task over the available resource</td>
<td>minimum completion time.</td>
</tr>
<tr>
<td>2</td>
<td>Raghavendra Achar, P. SanthiThilagam, 2012</td>
<td>Optimal Scheduling of Computational Task in Cloud using Virtual Machine Tree</td>
<td>Comparison of DFS algorithm with the FCFS and priority based algorithm</td>
<td>the proposed algorithm is much efficient than priority based algorithms and FCFS.</td>
</tr>
<tr>
<td>3</td>
<td>Pinky Rosemarry, Ravinder Singh, PayalSingha and Dilip Sisodia, 2012</td>
<td>Grouping Based job scheduling algorithm using priority queue and hybrid algorithm in grid computing</td>
<td>Comparative study hybrid three scheduling algorithm in terms of processing time</td>
<td>the proposed hybrid algorithm gives better performance than shortest job first algorithm alone in terms of processing time.</td>
</tr>
<tr>
<td>4</td>
<td>Rajput, Ishwari Singh, and Deepa Gupta, 2012</td>
<td>A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems</td>
<td>comparison in terms of waiting and turnaround time and CPU Utilization of jobs with algorithm simple round robin.</td>
<td>performs better than simple round robin for varying waiting and turnaround time and CPU Utilization</td>
</tr>
<tr>
<td>5</td>
<td>R. Vijayalakshmi, Mrs. Soma Prathibha, 2013</td>
<td>A Novel Approach for Task Scheduling in</td>
<td>The user’s tasks first got prioritized and then submit to</td>
<td>Has minimizing the makespan to utilization</td>
</tr>
<tr>
<td></td>
<td>Cloud</td>
<td>the virtual machine on the basis of the prioritization</td>
<td>Comparison of this algorithm with the FCFS</td>
<td>of allocated resources more efficiently</td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>------------------------------------------------------</td>
<td>--------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>6</td>
<td>Dr Ajay Jangra, Tushar Saini, 2013</td>
<td>Scheduling Optimization in Cloud Computing</td>
<td>Selection of resources is done in accordance of its turnaround time and cost using greedy approach.</td>
<td>This way of task selection and resource selection gives better results over sequential scheduling.</td>
</tr>
<tr>
<td>8</td>
<td>Santhosh, Dr. Manjaiah D.H, 2014</td>
<td>An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing</td>
<td>modified improved Max-Min algorithm to define two new algorithms based on the average execution time</td>
<td>Are more efficient for minimization of the makespan</td>
</tr>
<tr>
<td>9</td>
<td>Ranjan Kumar Mondal, Enakshi Nandi and Debabrata Sarddar, 2015</td>
<td>Load Balancing Scheduling with Shortest Load First</td>
<td>compare the result of proposed SJF algorithm with the existing RR, FCFS algorithms</td>
<td>The average waiting time is minimal, that improve the system performance</td>
</tr>
<tr>
<td>10</td>
<td>M. LawanyaShri, M.B.BenjulaAnbumalar, K. Santhi, Deepa.M, 2016</td>
<td>Task Scheduling Based on efficient optimal algorithm in cloud computing environment</td>
<td>compared three algorithms Time Shared, Space shred and generalizes priority algorithm</td>
<td>proposed algorithm is more efficient than FCFS and SJF algorithm.</td>
</tr>
</tbody>
</table>

**Table 2.1 Related Work comparative Table**
CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction
This chapter presents the steps that will be followed to achieve the research objectives. The proposed hybrid algorithm provides an optimal scheduling method produced by merging the best algorithms (shortest job first algorithm load balancing with Generalized Priority Algorithm(GPA)) in order to mitigate the drawback of both algorithm.

Finally, presents the proposed method for tasks scheduling consider the processing time , average waiting time and the average turnaround time of the tasks. The research methodology steps show firstly, study several literature reviews, secondly, determine the problems, and thirdly, design the proposed schema, then check if the proposed method is suitable or not, if suitable Develop the new Algorithm, then evaluate using java tool and analyse the obtained results. if acceptable to proposed methods must measure the benchmark, else back to up and try again, as show in Figure 3.1

Figure3.1 Research Methodology
3.2 THE ALGORITHMS:-

3.2.1 Shortest Job First

Shortest Job First (SJF) scheduling is a priority and Non-Preemptive scheduling. Non-Preemptive means, when the allotted time a processor then the processor cannot be taken the other, until the process is completed in the execution. Basically Shortest Job First is a dynamic load balancing algorithm which handles the process with priority basis. It determines the priority by checking the size of the process. This algorithm distributes the load randomly by first checking the size of the process and then transferring the load to a Virtual Machine, which is lightly loaded. In that case that process size is lowest. This process will get first priority to execute whether we suppose lowest sized process executes in minimum time.

The mechanism of Shortest Job First Algorithm is to schedule the process with the shortest time to completion first, thus providing high efficiency and low turnaround time. In terms of time spent in the current program (task) began to enter in to the system until the process is finished the system needs a short time. Shortest Job First (SJF) scheduling algorithm can be said to be optimal with an average waiting time is minimal, which helps to improve the system performance.

3.2.1.1 Drawbacks:

In Shorts Job First long jobs may wait longer because it has to wait not only for jobs that are in the system at the time of its arrival, but also for all short jobs that are in the system at the time of its arrival.
3.2.2 Generalized Priority Algorithm:

Customer define the priority according to the user’s demand parameter of cloudlet show be define like size, memory, bandwidth scheduling policy etc. In the Generalized Priority Algorithm, the tasks are initially prioritized according to their size such that one having highest size has highest rank. The processors are also ranked (prioritized) according to their MIPS value such that the one having highest MIPS has the highest rank. Thus, the key factor for prioritizing tasks is their size and for processor is their MIPS.
3.2.2.1 Drawbacks:
The drawback of this algorithm is that, the execution of tasks with highest size first may increase the processing time, and lead to a delay in executing tasks with minimum size if the number of tasks with highest size exceeds that of tasks with minimum size.

Figure 3.2: Generalized Priority Algorithm Flowchart
3.2.3 The Proposed Hybrid Algorithm

**Step1:** Enter number of task, size, burst time, arrival time.

**Step2:** Enter number of processors with MIPS.

The user enters any number of tasks to be executed on processor in proposed hybrid algorithm. also enter size, burst time, arrival time. for each task.

**Step3:** The modified condition If task size >10000 go to step 4 else go to step 12

**Step4:** Add the task with size greater than 10000 to big list

**Step5:** IF un assign task list >0 that mean the un assign list is not empty go to step 6 else go to final step 18.

**Step6:** Sort processors on the basis of MIPS value such that the one having highest MIPS has the highest rank and task on the basis of size in descending order.

**Step7:** i=0 to begin from first task

**Step8:** i++ a very time increase one task until to last task

**Step9:** assign task to processor this task is highest size between task in the list.

**Step10:** calculate the processing time.

**Step11:** IF reach the end of the tasks go to step 17 else some tasks not execute then go to step 8

**Step12:** Add the task not satisfied the condition with the size less than 10000 to small list.

**Step13:** for each task calculate the metrics burst time

**Step14:** Sort the task in ascending order with Short burst time to execute according minimal burst time.
**Step15:** Make average waiting time length of next process.

**Step16:** Start with first process, selection as above as shortest load come first which has minimal average time and other tasks are to be in queue.

**Step17:** calculate the processing time and the average waiting time and the average turnaround time. This process is repeated for each task until reach the end of tasks.

**Step18:** The end of algorithm.

To achieve maximum utilization and user satisfaction it needs to schedule their resources effectively. The proposed hybrid algorithm above, explains how reducing the time of processing, it does not depend on single factor for scheduling. Three important factors are considered for efficient workflows of tasks the processing time, average waiting time and the average turnaround time.
Figure 3.4: Proposed Algorithm Flowchart
3.4 Experimental setups

Figure 3.5 Experimental setups model

3.4.1 Scenarios

In order to evaluate the performance, three scenarios will be implemented with each scenario having four sections according to the number of tasks and also according to the processing time and the average waiting time and the average turnaround time shown in Figure 3.4.

The proposed algorithm is compared to existing task scheduling algorithms (short job first algorithm, load balancing with Generalized Priority Algorithm (GPA)), for this purpose, the following illustrative example is taken.

It is created many processors and tasks with different task sizes. Task sizes range from 10000 to 20000, and the processors have been created which have processing power ranges from 250-1000 MIPS.
**Scenario 1:**

It takes 5, 10 and 15 processors and in each time change tasks with different size at first take 12, 14, 16 and 18 task with 5 processors then take 12, 14, 16 and 18 with 10 processors and finally take 12, 14, 16 and with 15 processors also every time calculate the processing time.

**Scenario 2:**

It takes 5, 10 and 15 processors and in each time change tasks with different size at first take 12, 14, 16 and 18 task with 5 processors then take 12, 14, 16 and 18 with 10 processors and finally take 12, 14, 16 and with 15 processors also every time calculate the average waiting time.

**Scenario 3:**

It takes 5, 10 and 15 processors and in each time change tasks with different size at first take 12, 14, 16 and 18 task with 5 processors then take 12, 14, 16 and 18 with 10 processors and finally take 12, 14, 16 and with 15 processors also every time calculate the average turnaround time.
CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 INTRODUCTION

In order to fully analyze the results of our experiments we classified the results into three scenario based on the implemented in the scenario, in order to verify proposed algorithm, we conducted experiment on Toshiba core(TM) i5 Processor 2.6 GHz, Windows 7 platform and using java Netbeans and set the property of RAM as 512 MB for all processors, and the MIPS as 250, 1000, 250, 500 and 250 respectively.

4.2 SENARIO ONE

4.2.1 processing time according to 5 processors between the proposed algorithm and Short job first load balancing algorithm and the GPA.

![Processing Time-5 processors](image)

**Figure 4.1 Comparison According 5 Processors**

In this Figure 4.1 Comparison According 5 Processors illustrates the comparison processing time between the proposed algorithm and generalized priority algorithm (GPA) and short job first load balancing algorithm. The result observes the processing time. The proposed algorithm performs better than other two algorithms with minimum processing time.
Table 4.1 Comparative the Proposed hybrid Algorithm with the other two existing algorithms according to processing time and 5 processor.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>8.9</td>
<td>9</td>
<td>7.8</td>
<td>8.3</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>9.9</td>
<td>9.3</td>
<td>8.9</td>
<td>11</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>10.3</td>
<td>10.8</td>
<td>9.2</td>
<td>11.7</td>
</tr>
</tbody>
</table>

Processing time according to 10 Processor between the proposed algorithm and the GPA and Short job first load balancing algorithm.

Figure 4.2 Comparison According 10 Processor

In this figure 4.2 illustrates the comparison processing time between the proposed hybrid algorithm and generalized priority algorithm (GPA) and SJF.
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>9.5</td>
<td>9.7</td>
<td>8.5</td>
<td>9</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>1.5</td>
<td>10.8</td>
<td>9.8</td>
<td>12</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>10.9</td>
<td>11.2</td>
<td>10.2</td>
<td>12.6</td>
</tr>
</tbody>
</table>

processing time according 15 Processor between the proposed algorithm and the GPA and SJF.

![Processing Time - 15 Processors](chart)

**Figure 4.3 Comparison According 15 Processors**

By observing the processing time, performance in this Figure 4.2 illustrates the comparison processing time between the proposed algorithm and generalized priority algorithm (GPA) and SJF. When the number of the processors is 15, proposed algorithm performs better than other two algorithms with minimum processing time and increased throughput.

Table 4.3 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to processing time and 15 processor.
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>8</td>
<td>8.8</td>
<td>7.6</td>
<td>8</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>9.5</td>
<td>9.4</td>
<td>8.6</td>
<td>11</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>9.7</td>
<td>9.9</td>
<td>9</td>
<td>11.5</td>
</tr>
</tbody>
</table>

4.2 SCENARIO TWO

Average waiting time according 5 Processor between the proposed algorithm and the GPA and SJF.

Also when observing the performance of 5 processors. In this Figure 4.4 illustrates the result comparison of the proposed algorithm and generalized priority algorithm (GPA).

![average waiting time-5 processors](image)

**Figure 4.4 Comparison According 5 Processors**
Table 4.4 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to Average waiting time and 5 processor.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>5.9</td>
<td>6.9</td>
<td>8</td>
<td>11.9</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>10.5</td>
<td>11</td>
<td>9.3</td>
<td>13.1</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>14.1</td>
<td>13.3</td>
<td>14.2</td>
<td>15.9</td>
</tr>
</tbody>
</table>

Average waiting time 10 Processor between the proposed algorithm and the GPA and SJF.

![Average waiting time-10 processors](image)

**Figure 4.5** Comparison According 10 Processors

Figure 4.5 Comparison According 10 Processors observes that the proposed algorithm performs better than other two algorithms with minimum Average waiting time and increased throughput.

Table 4.5 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to average waiting and 10 processor.
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>5.9</td>
<td>6.9</td>
<td>8</td>
<td>11.9</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>10.5</td>
<td>11.1</td>
<td>9.7</td>
<td>13.2</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>14.2</td>
<td>13.5</td>
<td>14.6</td>
<td>17</td>
</tr>
</tbody>
</table>

Average waiting time 15 Processor between the proposed algorithm and the GPA and SJF.

![Average waiting time-15 processors](image)

**Figure 4.6  Comparison According 15 Processors**

Also when observing the performance of 15 processors, in this figure 4.6 Comparison According 15 Processors illustrated the result comparison of the proposed algorithm and generalized priority algorithm (GPA). When the parameter is average waiting time.

Table 4.6 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to Average waiting and 15 processor.
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>5.7</td>
<td>7</td>
<td>8.3</td>
<td>11.6</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>10.3</td>
<td>10.8</td>
<td>9.4</td>
<td>12.8</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>14</td>
<td>13</td>
<td>14.2</td>
<td>16.9</td>
</tr>
</tbody>
</table>

### 4.3 SENARIO THREE

Average turnaround time 5 Processors between the proposed algorithm and the GPA and SJF.

![Average Turnaround Time -5 Processors](image)

**Figure 4.7 Comparison According 5 Processors**

Figure 4.7 Comparison According 5 Processors observes that the proposed algorithm performs better than other two algorithms with average waiting time and increased throughput.
Table 4.7 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to Average turnaround time and 5 processor.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>16.7</td>
<td>19.5</td>
<td>18</td>
<td>19.2</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>21.2</td>
<td>26.5</td>
<td>24.4</td>
<td>24.4</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>25.3</td>
<td>28.4</td>
<td>27.3</td>
<td>26.9</td>
</tr>
</tbody>
</table>

Average waiting time 10 Processor between the proposed algorithm and the GPA and SJF.

Also when observing the performance of 10 processors, in this Figure 4.8 illustrates the result comparison of the proposed algorithm and generalized priority algorithm (GPA). When the parameter is average turnaround time.

![Average Turnaround Time -10 Processors](image)

**Figure 4.8 Comparison According 10 Processors**

Table 4.8 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to Average turnaround time and 10 processor.
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>17</td>
<td>19</td>
<td>17.5</td>
<td>18.8</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>20.5</td>
<td>25.5</td>
<td>23.8</td>
<td>23.7</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>24.6</td>
<td>27.8</td>
<td>26.7</td>
<td>26.9</td>
</tr>
</tbody>
</table>

Average waiting time 15 Processors between the proposed algorithm and the GPA and SJF.

![Average Turnaround Time - 15 Processors](image)

**Figure 4.9 Comparison According 15 Processors**

By observing the average turnaround time in this Figure 4.9 Comparison According 15 Processors illustrates the comparison between the proposed algorithm and generalized priority algorithm (GPA) and SJF. When the number of the processors is 15.
Table 4.9 Comparative the Proposed hybrid algorithm with the other two existing algorithms according to Average turnaround time and 15 processor.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>12 Task</th>
<th>14 Task</th>
<th>16 Task</th>
<th>18 Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed hybrid Algorithm</td>
<td>16</td>
<td>17.5</td>
<td>16</td>
<td>17.5</td>
</tr>
<tr>
<td>Generalized Priority Algorithm</td>
<td>19.7</td>
<td>19.4</td>
<td>22.6</td>
<td>23</td>
</tr>
<tr>
<td>Shortest Job First load balancing Algorithm</td>
<td>23.7</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
</tbody>
</table>

4.4 Discussions Result

- (Table 1,2,3) displays processing time of 12,14,16,18 tasks in 5,10 and 15 processors comparison with the two existing algorithms. The results show that the processing time of the hybrid algorithm is less than of the two existing algorithms when the number of processors increase.

- (Table 4,5,6) displays Average Waiting time of 12,14,16, and 18 tasks in comparison with the two existing algorithm in 5,10 and 15 processors, if the number of tasks is increased, this increase has maximized the average waiting of tasks and the opposite is true when the number of processor increases.

- In the last three tables (Table 7,8,9) displays Average turnaround time of 12,14,16 and 18 tasks in comparison with the two existing algorithm in 5,10 and 15 processors when the number of task increase the average turnaround time increase.
CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION
Cloud computing has become an important area of research and one of the major topics of research is dynamic scheduling. Therefore, This research introduced scheduling algorithm which handles the problems of scheduling processing time in an efficient way with minimum process time and also the resource should not be underutilized has been introduced. The proposed hybrid algorithm is based on three parameter. The processing time and the average waiting time and the average turnaround time. The proposed hybrid algorithm has been tested and evaluated using java netbeans tool. The result is minimum processing time in comparison with the two existing algorithms, the first one based on task size and MIPS of processor; the second on short job first according to burst time.

The experiment had been done in order to study the performance of the proposed hybrid algorithm and examine the outputs. The results showed the proposed hybrid algorithm improve the cloud system performance by decreasing the processing time about 5%.

5.2 RECOMMENDATIONS
This research is recommended to:

- Improve the proposed hybrid algorithm by taking the response time
- Apply the proposed hybrid algorithm and improve algorithms on the real cloud computing systems, in order to evaluate these systems performance.
REFERENCES


[6] F. Teng, “Ressource allocation and schelduling models for cloud computing To cite this version ;,” 2012.


Appendix A

Package mix;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.Random;

public class mix {

    static List<Task> tasks = new ArrayList<Task>();
    static List<Cpu> cpus = new ArrayList<Cpu>();
    static List<Task> queue = new ArrayList<Task>();

    static List<Task> smallTasks = new ArrayList<Task>();
    static List<Task> bigTasks = new ArrayList<Task>();

    static List<Cpu> smallCpus = new ArrayList<Cpu>();
    static List<Cpu> bigCpus = new ArrayList<Cpu>();

    static List<Task> finishedTask = new ArrayList<Task>();
    static Random rand = new Random();
    static int GlobalTime = 0;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Tasks: =====================");
        System.out.print("Enter Number of Tasks:");
        int numberOfTask = sc.nextInt();
        for (int i = 0; i < numberOfTask; i++) {
        }
System.out.print("Enter arrive time of task number "+(i+1)+":");
int arrive = sc.nextInt();
System.out.print("Enter size of task number "+(i+1)+":");
int size = sc.nextInt();
tasks.add(new Task(i+1,arrive,size));
}
System.out.println("CPUs: =========
============");
System.out.print("Enter Number of CPUs:");
int numberOfCpu = sc.nextInt();
for (int i=0;i<numberOfCpu;i++)
{
System.out.print("Enter mips of cpu number "+(i+1)+":");
intmips = sc.nextInt();
cpus.add(new Cpu(i+1,mips));
}
intAllMips = 0;
for (int i = 0;i <cpus.size();i++)
{
AllMips += cpus.get(i).mips;
}
for (int i = 0;i <cpus.size();i++)
{
int mi = cpus.get(i).mips;
double per = (0.005-(((float)mi / (float)AllMips)*100)*0.00005)/cpus.size();
if (per == 0)
{

per = rand.nextInt(20)+1;
}
//per = per *10;
cpus.get(i).pre = per;
}
for (int i = 0;i <cpus.size();i++)
{
if (cpus.get(i).mips>= 500)
bigCpus.add(cpus.get(i));
}
else
{
smallCpus.add(cpus.get(i));
}
}
System.out.println("Tasks");
printlist(tasks);
while (!tasks.isEmpty() || CpuBusy())
{
GlobalTime++;
    //ds(GlobalTime);
for (Task t : tasks.stream().filter(x -> x.arrive == GlobalTime).toArray(Task[]::new))
{
bigTasks.add(t);
    /*
if (t.size > 10000)
{
bigTasks.add(t);
}
else
{
smallTasks.add(t);
    }*/
    //queue.add(t);
tasks.remove(t);
}
for (int i = 0; i < cpus.size(); i++)
{
if (cpus.get(i).currTask == null) continue;
    Task currTask = cpus.get(i).currTask;
    currTask.remain--;
    if (currTask.remain == 0)
{  
cpus.get(i).currTask.finish = GlobelTime;
cpus.get(i).currTask.wait = cpus.get(i).currTask.finish - cpus.get(i).currTask.start;
cpus.get(i).currTask.realwait = cpus.get(i).currTask.start - cpus.get(i).currTask.arrive;
cpus.get(i).currTask.turn = cpus.get(i).currTask.finish - cpus.get(i).currTask.arrive;
finishedTask.add(cpus.get(i).currTask);
cpus.get(i).EmptyTask();
}

if (!bigTasks.isEmpty())
{
  bigTasks.sort((o1,o2) -> o1.size - o2.size);
  bigCpus.sort((o1,o2) -> o1.mips - o2.mips);
  for (int i=0;i<bigCpus.size();i++)
  {
    if (!bigTasks.iterator().hasNext()) break;
    if (bigCpus.get(i).busy) continue;
    Task temp = bigTasks.iterator().next();
    temp.start = GlobelTime;
    bigCpus.get(i).setCurrTask(temp);
    //bigCpus.get(i).currTask = temp;
    bigTasks.remove(temp);
  }
}

if (!smallTasks.isEmpty())
{
  smallTasks.sort((o1,o2) -> o2.remain - o1.remain);
  smallCpus.sort((o1,o2) -> o2.mips - o1.mips);
  for (int i=0;i<smallCpus.size();i++)
  {
    if (!smallTasks.iterator().hasNext()) break;
    if (smallCpus.get(i).busy) continue;
    Task temp = smallTasks.iterator().next();
  }
}
//bigCpus.get(i).currentJob = temp.id;
//bigCpus.get(i).busy = true;
temp.start = GlobalTime;
smallCpus.get(i).setCurrTask(temp);
    //bigCpus.get(i).currTask = temp;
smallTasks.remove(temp);
    }
    }

System.out.println("smallTasks");
printlist(smallTasks);
System.out.println("bigTasks");
printlist(bigTasks);

System.out.println("finishedTask");
printlist(finishedTask);

System.out.println("cpus");
printCpu(cpus);

int sumProcessing = 0;
for (int i = 0;i <finishedTask.size();i++)
    {
    sumProcessing+= finishedTask.get(i).wait;
    }
System.out.println("Processing:"+((float)sumProcessing/(float)finishedTask.size()));
```java
int sumwait = 0;
for (int i = 0; i < finishedTask.size(); i++)
    sumwait+= finishedTask.get(i).realwait;
System.out.println("Avg Wait time:" + (float)sumwait/(float)finishedTask.size());

int sumturn = 0;
for (int i = 0; i < finishedTask.size(); i++)
    sumturn+= finishedTask.get(i).turn;
System.out.println("Avg Turn time:" + (float)sumturn/(float)finishedTask.size());
}
static void printlist(List<Task> t)
{
for (int i = 0; i < t.size(); i++)
    System.out.println(t.get(i).TaskString());
}
static Task findTask(int id, List<Task> t)
{
for (int i = 0; i < t.size(); i++)
    if (t.get(i).id == id) return t.get(i);
return null;
}
static void printCpu(List<Cpu> c)
{
for (int i = 0; i < c.size(); i++)
```
static void initCpu()
{
    cpus.add(new Cpu(1,1000));
    cpus.add(new Cpu(2,500));
    cpus.add(new Cpu(3,250));
    cpus.add(new Cpu(4,250));
    cpus.add(new Cpu(5,250));
    intAllMips = 0;
    for (int i = 0;i <cpus.size();i++)
    {
        AllMips += cpus.get(i).mips;
    }
    for (int i = 0;i <cpus.size();i++)
    {
        int mi = cpus.get(i).mips;
        double per = (0.005-(((float)mi / (float)AllMips)*100)*0.00005))/cpus.size();
        cpus.get(i).pre = per;
    }
}
static void initTasks()
{
    tasks.add(new Task(1,10,22000));
    tasks.add(new Task(2,1,10000));
    tasks.add(new Task(3,4,20000));
    tasks.add(new Task(4,6,10000));
    tasks.add(new Task(5,8,10000));
    tasks.add(new Task(6,9,20000));
    tasks.add(new Task(7,11,10000));
    tasks.add(new Task(8,13,20000));
    tasks.add(new Task(9,15,10000));
    tasks.add(new Task(10,14,10000));
tasks.add(new Task(11,17,20000));
tasks.add(new Task(12,19,10000));

}

static boolean CpuBusy()
{
    for (int i = 0; i < cpus.size(); i++)
    {
        if (cpus.get(i).busy == true) return true;
    }
    return false;
}

static void ds(int gg){
    System.out.println("GlobalTime" + gg);
    System.out.println("smallTasks");
    printlist(smallTasks);
    System.out.println("bigTasks");
    printlist(bigTasks);
    System.out.println("finishedTask");
    printlist(finishedTask);
    System.out.println("cpus");
    printCpu(cpus);
public class Cpu {
    int mips, id, currentJob;
    Task currTask;
    double pre;

    boolean busy = false;

    Cpu(int id, int mips)
    {
        this.id = id;
        this.mips = mips;
    }

    String CpuString()
    {
        return "id:"+id+", mips:"+mips+", current Job:"+currentJob+", pre:"+pre;
    }

    void setCurrTask(Task t)
    {
        currentJob = t.id;
        currTask = t;
        currTask.remain = (int)(currTask.remain * pre) ;
        busy = true;
    }

    void EmptyTask()
    {

public class Task {
    int id, arrive, size, priority, start, wait, remain, finish, turn, realwait;

    Task(int id, int arrive, int size) {
        this.id = id;
        this.arrive = arrive;
        this.size = size;
        remain = size;
    }

    public String TaskString() {
        return "id: " + id + ", arrive: " + arrive + ", size: " + size + ", start: " + start + ", Processing
    }
}

currTask = null;
busy = false;