# Advance Resource Reservation in Cloud Computing using Checkpoint Mechanism


Submitted By

**Parimal Gajre**

**15MCEC12**




**DEPARTMENT OF COMPUTER ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**


**May 2017**

# Advance Resource Reservation in Cloud Computing using Checkpoint Mechanism

**Major Project**

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering


Submitted By

**Parimal Gajre**

**(15MCEC12)**

Guided By

**Asst. Prof. Vivek K. Prasad**




**DEPARTMENT OF COMPUTER ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**


**May 2017**

# Certificate

This is to certify that the major project entitled **"Advance Resource Reservation in Cloud Computing using Checkpoint Mechanism"** submitted by **Parimal Gajre (15MCEC12)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.


Prof. Vivek Kumar Prasad
Guide & Asst. Professor,
Computer Engineering Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Priyanka Sharma
Associate Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
Computer Engineering Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

# Statement of Originality

I, **Parimal Gajre**, Roll. No. **15MCEC12**, give undertaking that the Major Project entitled "**Advance Resource Reservation in Cloud Computing using Checkpoint Mechanism**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Vivek K. Prasad

(Signature of Guide)

# Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Vivek Kumar Prasad**, Assistant Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

**- Parimal Gajre**
**15MCEC12**

# Abstract

Cloud Computing gives the facility of provisioning resources on rent in pay as-you-go fashion, due to which resource demand changes dynamically over time. Such type of dynamic resource demand leads to mainly two types of problems such as Over-Provisioning and Under-Provisioning. The former leads to violations of Service Level Agreements (SLAs) whereas latter leads to wastage of resources, as the system is not being used to full capacity all the time. Also some cloud having limited number of resources cannot satisfy all the requests at a time.

To handle such scenario advance reservation techniques are used, so that the resources available could be used efficiently with minimum possible provisioning cost and at the same time satisfying service level agreements. In the proposed technique, history of resource usage profile of tasks is maintained. For each task submitted to cloud, pattern finding technique is used to find a task with similar resource usage requirement from history of resource usage profile. After this check-pointing mechanism is used to monitor completion of new task based on resource usage profile of task found in historical data. Such type of monitoring helps in order to estimate the amount resources that will be released as per profiled time-line and based on that resources can be reserved in advance as per users request. Hence problem of under as well as over provisioning could be solved up to a great extent at the same time meeting the Service Level Agreements.

# Abbreviations

| | |
|---|---|
| **CSP** | Cloud Service Provider. |
| **DC** | Data-Center |
| **GWP** | Google Wide Profiling. |
| **IaaS** | Infrastructure as a Service |
| **PaaS** | Platform as a Service |
| **PFS** | Parallel File System |
| **QoS** | Quality of Service. |
| **SLA** | Service Level Agreement |
| **SaaS** | Software as a Service |
| **TLC** | Thread-based Live Checkpointing |
| **VM** | Virtual Machine |

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Cloud Computing

Cloud Computing is a new paradigm in todays world of fastest changing technology, which delivers computing as a utility through the Internet. Cloud computing follows pay as you go fashion, in the sense that customers need to pay only for the amount of time the resources are being used by customers. So such type of system is advantageous to organisations as they do not need to worry about investing the capital in setting up the infrastructure and directly use the available services from the Cloud Service Providers. Cloud Computing services are provided CSP usually follows XaaS model, which means X as a Service. Some of the most commonly used services are Infrastructure as service (IaaS), Platform as a service (PaaS) and Software as Service (SaaS), which is provisioned by pay as you go model over Internet.

## 1.2 Cloud Computing Characterstics

- **Broad network access:** Customer can access their service from broad range of devices like Mobile, laptop, tablet, pc and workstation. There is no limitation about geographical area it can be accessible from any device in any area.
- **On-demand self-services:** Customer can get their services as they required like server computation time, storage and latest patches without any human efforts.
- **Resource pooling:** Hardware resources are distributed among different countries,

states, datacenters. This all different resources like Memory, storage, network bandwidth, processing are pooled to gather to provide better and fast services. Customer has no particular information about backend resources but they have some abstract information.

- **Measured Services:** Different cloud services are measured and paid according to type of services like Bandwidth, storage, active user accounts, cpu processing and quantity of services used by customer.

- **Rapid elasticity:** customer can easily expand or shrink their cloud services provided by cloud service provider. As Cloud has characteristics of elasticity any service can be manually or automatically adjust.

## 1.3 Types of Cloud

Type of Clouds classified based on thier access levels are as represented in figure 1.1 and discussed as follows,

- **Public Cloud:** In public cloud pooled of resources are shared with all the public user. Any type of user like academic organization, small scale company, financial organization can use this services and can save cost of hardware and software. This type of services are free for certain amount of use and then it will be paid according to provider but cheaper than other type of cloud. As all the resources are publicly shared then security is major concern. But it can be managed using SLA with the Cloud provider.

- **Private Cloud:** To overcome the security issue from public cloud Private cloud are used. In private cloud all the physical infrastructure are available within the organization and configure and managed by IT technician. As it is in the organization company can set its security policy and managed it according to their use. But for this type of cloud services company need more budget for infrastructure and IT staff.

- **Community Cloud:** Multiple organization share the same Pooled of resources like Infrastructure, policies, security requirement. This type of services managed by those organization itself or by third party CSP. An example of the community

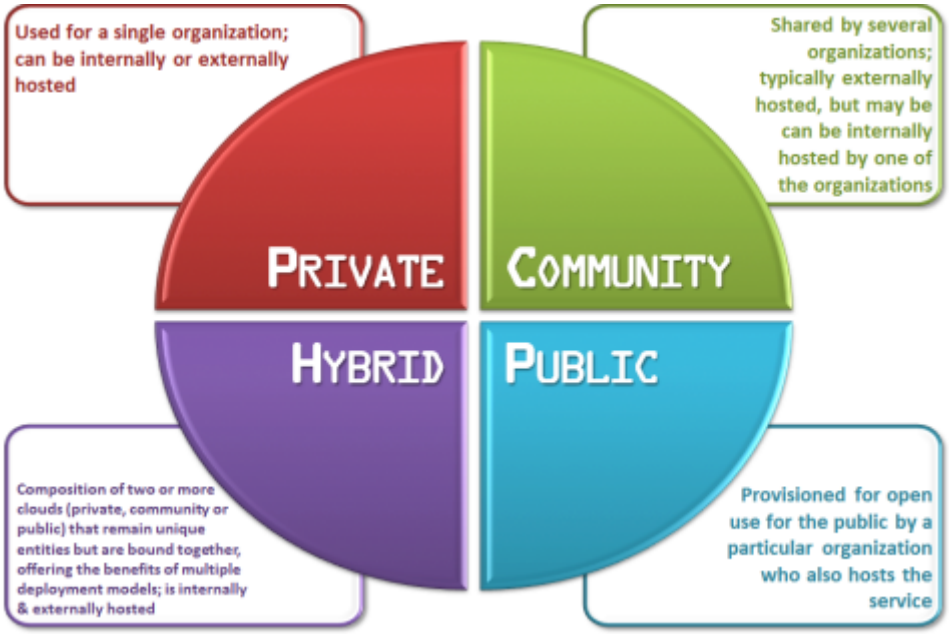



Figure 1.1: Types of Cloud
[7]

cloud can be a cloud shared by some legal firms. A community cloud can be thought of as a targeted public cloud.

- **Hybrid Cloud:** This is the common type of cloud which is widely used nowadays. As Different organizations have different requirement so they need to use both public and private cloud. As common data are shared via public cloud and important data store and shared in private cloud. In this way you only pay for what you use, when you use it, while not having to migrate entirely to the cloud if the business is not yet ready for that step or its not a requirement.

## 1.4 Cloud Services

Features provided by Cloud Service Provider under cloud service model as shown in figure 1.2 are as follows:

- **Infrastructure as a Service (IaaS):**
  This type of cloud service provides storage and compute capacity over the network. Varying workload requirement ranging from small application components to High Performance Applications is being catered by Pool of Servers, storage systems, switches, routers, and other systems. This is the most flexible and configurable

cloud service option, but conversely requires the most effort to manage and support from a consumer perspective. Some examples of giant enterprise that provides IaaS commercially include Google compute engine, Amazon EC2, Microsoft Azure, etc.

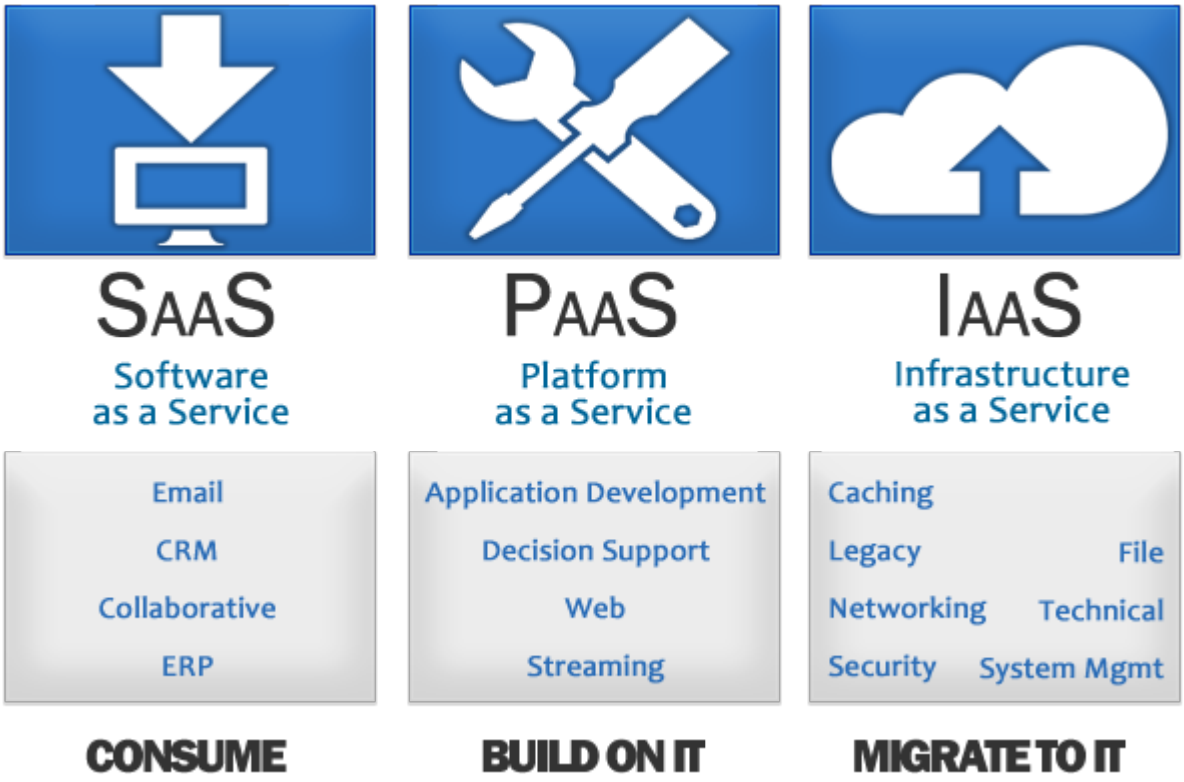



Figure 1.2: Cloud Service Model
[4]

- **Platform as a Service (PaaS):**

  This type of cloud service provides platform by encapsulating software layer, which is used to build higher level software services. So it basically provides the platform or framework for application and other development or customizing an already available application. PaaS is less configurable than IaaS, but provides a fully managed platform such as the Database engine incase of database used as a service, so it requires less management and effort to set-up / configure than IaaS. Some examples of giant enterprise that provides PaaS commercially include Force.com, Google App Engine, etc.

- **Software as a Service (SaaS):**

  This type of cloud service provides complete software application on demand. A single instance of the software runs on the cloud and services multiple end users or client organizations. So users need not to worry about the installation of the software and licensing agreements, it is just needed to start using the software application as and when required. Some examples of giant enterprise that provides SaaS commercially include Salesforce.com, Gmail, Microsoft Office 365, etc.

Out of the types of cloud services discussed here, the proposed work will have its effect in Infrastructure as a Service layer. As release time of resources currently executing tasks is estimated and based on this estimation, that particular resources are being reserved for next task in queue.

## 1.5 Motivation

In recent years cloud computing has gained an immense growth, especially in the use of public clouds. Giant companies like Microsoft, Google, Amazon and Rackspace have released their public cloud infrastructures such as Microsoft Azure, Google App Engine, Amazon Web Services and Rackspace Cloud Servers. For such huge enterprise software systems provisioning high assurance in terms of Quality of Service (QoS) metrics such as service availability, high throughput and response time is necessary. Customers signs Service Level Agreements (SLA) with Cloud Service Providers (CSP), which specifies the QoS metrics agreed by CSP to satisfy. In case if it fails in fulfilling the QoS mentioned in SLA, it will result in a great loss in income as it will tend to lose its customers base. While at same time maintaining SLA and also keeping costs low is difficult task to achieve for cloud service providers, because number customers to cloud computing system are not constant. [10] [13]

Also the applications that customers need to migrate to cloud have varying resource requirements, so it is necessary to have dynamic mechanism for resource provisioning, in order to satisfy such varying requirements of the applications. This raises the difficult challenge to predict behaviour of applications at run-time in order adjust provisioning of

resources dynamically.[6] [18] Hence workload that is likely to being offered by applications on cloud is required to be predicted, so that resources required for provisioning can be made available in advance so that overhead related to making resources available can be reduced.

## 1.6 Objectives

- To monitor the resource(s) usage of context aware task and creating its profile.
- To reserve the resource(s) using Check-pointing Mechanism.

# Chapter 2

# Existing Scenario

## 2.1 Provisioning of Resources

Provisioning of resources is concerned with allocation of virtual machines with resources such CPU, memory, bandwidth, storage, etc for execution of tasks. But as the cloud computing provides an illusion of infinite resources and resource demands of customers is highly dynamic so provisioning of the available resources is a challenging task.[12]

Resource Provisioning Techniques in Cloud Computing aims to guarantee that the application requirements are attended correctly. Resource provisioning relies heavily on the strong algorithms for allocating properly CPU, RAM, hard disk and other computational resources to the intended applications. To maintain its scalability, the process of provisioning in Cloud Computing must be dynamic.[16]

Mainly there two types problems in provisioning of cloud resources as:

- **Under Provisioning:** Leads to the violations of the service level objectives, often associated with financial penalties.
- **Over Provisioning:** Leads to wastage of the resources, as the system is not being used to full capacity all the time.

## 2.2 Problems in Resource Provisioning

As shown in figure below, there are following possible cases for static resource provisioning of cloud resources,

Figure 2.1 shows the case of over-provisioning when capacity of resources is planned to meet peak load condition, so resources will remain idle most of the time. Here, SLA's with customers will be satisfied but wastage of resources will increase as resources will remain in idle condition when the system is not being used to its full capacity.

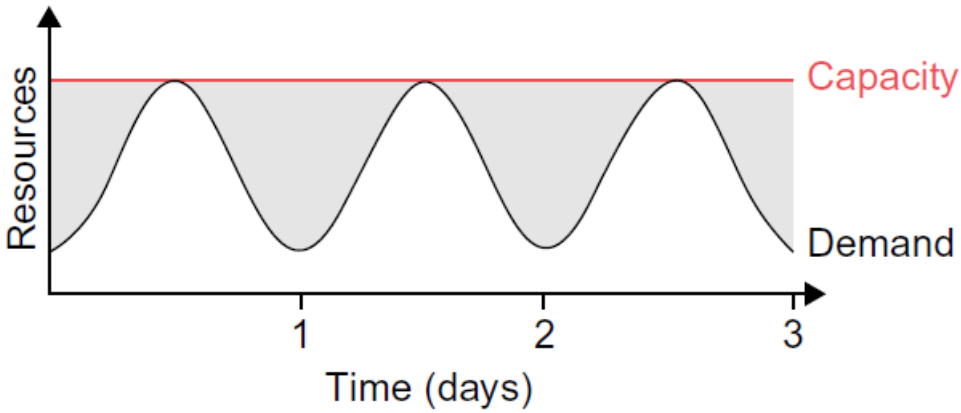



Figure 2.1: Over-Provisioning of Resources
[11]

Figure 2.2 shows the case of under-provisioning when capacity of resources is planned for average load. So in such case SLAs are broken most of the time leading to customers dis-satisfaction. So it would lead to loss of customers base thereby adversely affecting revenue generated by CSP.

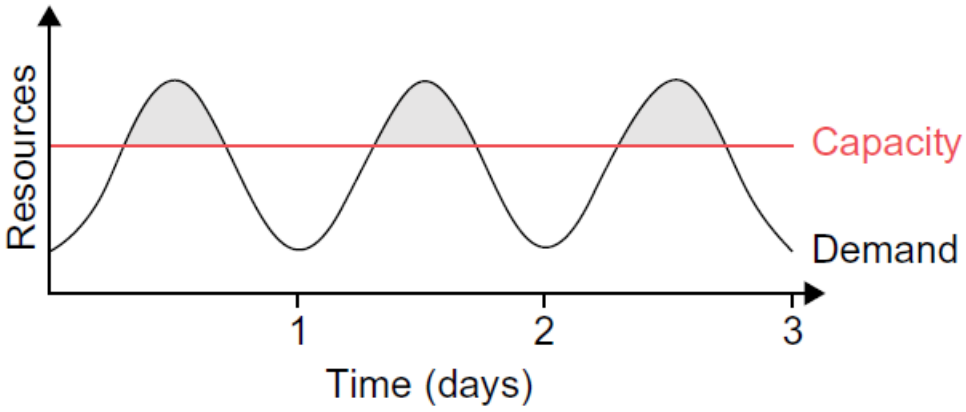



Figure 2.2: Under-Provisioning of Resources
[11]

Figure 2.3 shows the case when users resource demand decreases with time as constant provisioning of resources is not able to satisfy customers resource requirements, thereby further declination in user demand. Such declination of user demand leads to resource wastage and loss in revenue generated by resources to CSP.

## 2.3 Resource Provisioning Types

Based on application type of resource Provisioning is classified as follows,

- **Static Provisioning:** For applications that have predictable and generally unchanging demands/workloads, it is possible to use static provisioning" effectively.

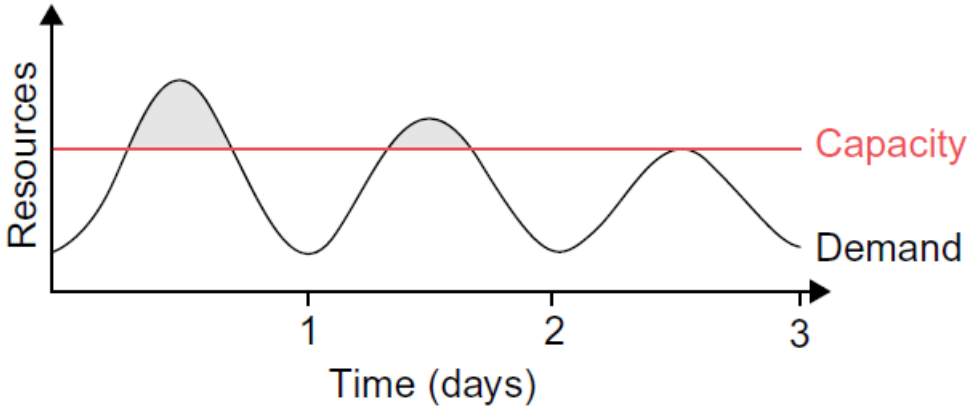



Figure 2.3: Constant-Provisioning of Resources
[2]

With advance provisioning, the customer contracts with the provider for services and the provider prepares the appropriate resources in advance of start of service. The customer is charged a flat fee or is billed on a monthly basis.

- **Dynamic Provisioning:** In cases where demand by applications may change or vary, dynamic provisioning" techniques have been suggested whereby VMs may be migrated on-the-fly to new compute nodes within the cloud. With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not. The customer is billed on a pay-per-use basis. When dynamic provisioning is used to create a hybrid cloud, it is sometimes referred to as cloud bursting.

- **User Self-provisioning:** With user self- provisioning (also known as cloud self-service), the customer purchases resources from the cloud provider through a web form, creating a customer account and paying for resources with a credit card. The provider's resources are available for customer use within hours, if not minutes.[15]

## 2.4 Parameters for Resource Provisioning

Some of important parameters considered for resource provisioning are as follows,

- **Response time:** The resource provisioning algorithm designed must take minimal time to respond when executing the task.

- **Minimize Cost:** From the Cloud user point of view cost should be minimized.
- **Revenue Maximization:** This is to be achieved from the Cloud Service Providers view.
- **Fault tolerant:** The algorithm should continue to provide service in spite of failure of nodes.
- **Reduced SLA Violation:** The algorithm designed must be able to reduce SLA violation.
- **Reduced Power Consumption:** VM placement & migration techniques must lower power consumption.

## 2.5 Resource Provisioning Methods

Following are some commonly used methods for resource provisioning,

### 2.5.1 Demand-Driven Resource Provisioning

In this method of resource provisioning the computing instances are added or removed based on current utilization level of the resources allocated. When an application usage of resource surpasses a threshold value for certain amount of time then according to this scheme resources are increased based on demand. Similarly, if resource usage goes below threshold value for certain amount time then resources being provided are decreased. Such type of auto scale scheme is used in Amazons EC2 platform. This scheme will not perform well in case when workload changes abruptly.

### 2.5.2 Event-Driven Resource Provisioning

In this scheme of resource provisioning resources are allocated or de-allocated based on a specific time event. This type of scheme if resource provisioning works better for seasonal or predicted workload. For example Christmas Eve in west and Lunar New Year in east. This scheme incurs minimal loss of QoS, if events are correctly predicted. Else, it may either result in wastage of resource that do not follow a specific pattern or lead to under provisioning if resource demand is more. One example of event driven resource provisioning is deadline-driven provisioning of resources, it has ablility to efficiently allocate

resources from different sources in order to reduce application execution times. But it is not suitable for HPC-data intensive applications.[21]

### 2.5.3 Popularity-Driven Resource Provisioning

In the said scheme of resource provisioning resources are allocated or de-allocated based on popularity index of the application. Here the popularity index of an application is determined which is based on internet searches for the application. So the increased traffic with increased popularity can be managed. But resources can be wasted if popularity of the application is not determined correctly. So it is lees useful for workloads that do not follow specific patterns.

# Chapter 3

# Literature Review

## 3.1 Profiling-Based Task Scheduling for Factory-Worker Applications in Infrastructure-as-a-Service Cloud

[26]

- **Author:** Rostyslav Zabolotnyi, Philipp Leitner, Schahram Dustdar
- **Year:** 2014
- **Transaction:** 40th Euromicro Conference on Software Engineering and Advanced Applications (IEEE)
- **Summary:**
  - Passive Profiling - Using this technique coarse grain profiling is done, which means profiling happens seamlessly to the application being profiled. In this technique profiling system is being configured for measurements of resource usage at some fixed intervals of time and has no consideration for state of task which being profiled.
  - Active Profiling - Using this technique fine grain profiling, in which task being profiled actively triggers the profiling system in-order to take measurements of resource usage at crucial points during execution of tasks. Hence it is possible to create a context-aware resource usage profile that exposes the actual behavior of tasks, which will be more helpful in making more confident and reliable scheduling actions.

- **Pros:**
  - Passive Profiling - Less impact on application performance as awareness of task is not required for application execution.
  - Active Profiling - More confident and reliable scheduling actions can be taken as it provides resource usage profile which is context-aware, that exposes actual task behaviour. Also profiling system is capable of suspending tasks on profiling points to get more accurate profiles of resource usages of tasks.
- **Cons:**
  - Passive Profiling - It has disadvantage due to its periodic nature as some resource usage spikes are missed. Also there are chances to misinterpret an application profile because of periodical measurements.
  - Active Profiling - More impact on application performance as application execution does require awareness of task profiling and some no. of iteration to achieve required level of granularity.
- **Future Work:** In future work, improvements on profile- based task scheduling can be done in-order to decrease usage complexity and internal knowledge required.

## 3.2 Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers

[17]

- **Author:** Gang Ren, Eric Tune, Tipp Moseley, Yixin Shi, Silvius Rus, Robert Hundt
- **Year:** 2010
- **Transaction:** - IEEE Computer Society
- **Summary:** GWP does profiling by performing event based sampling at machine level. GWP gets the data of machines in the fleet and remotely activates profiling on some random machines.

- **Pros:** GWP provides robustness as its collector is implemented as distributed service, thereby also improving availability. To decrease distortion of machines and applications running on them, monitoring of erroneous conditions is carried out and sampling rate of profiling is reduced if failure rate reaches some threshold value.

- **Cons:** Overhead incurred in continuous profiling of such huge data is significant and possibly degrades the performance of the application execution.

- **Future Work:** In future more type of performance events can be collected. Also more exploraration can be done in direction of how the GWP profiles could be used in more areas. Also it can more enhanced advanceddata mining techniques could be used for detecting interesting patterns from profiles.

## 3.3 Profiling-as-a-Service in Multi-Tenant Cloud Computing Environments

[24]

- **Author:** Kuai Xu, Feng Wang, Arizona State University Lin Gu, Hong Kong University of Science and Technology

- **Year:** 2012

- **Transaction:** - 32nd International Conference on Distributed Computing Systems Workshops (IEEE)

- **Summary:** It provides profiling as a service at different levels such as application profiling, instance profiling, hypervisor profiling and network profiling which is useful for making a comprehensive and correlated traffic profiles.

- **Pros:** As profiling is done at different layers it provides an indepth understanding of network traffic in cloud which helps to detect the anomalous patterns leading to attacks such denial of service i.e security monitoring. Also it is useful in customer profiling and application profiling where customer's application's resource usage could be monitored.

- **Cons:** As traffic profiling is done at each layer seperately it would add significant overhead which can degrade the performance of the system. Hence, it is required to calculate the frequency in whuch profiling should be at each layer so as to minimize the over head incurred due to profiling.

- **Future Work:** In future we can examine system utilizations of profiler can be measured at each layer for resources such as CPU and memory usage and speed of generation of traffic profiles at each layer. So network operators can use such information for real-time traffic analysis or attack mitigation.

## 3.4 Optimization of Cloud Task Processing with Checkpoint Restart Mechanism

[9]

- **Author:** Sheng Di, Yves Robert, Frdric Vivien, Derrick Kondo, Cho-Li Wang, Franck Cappello
- **Year:** 2013
- **Transaction:** ACM
- **Summary:** It checkpoints the transient memory of running tasks instead of entire VM state to reduce the overhead of checkpointing. It does not make any assumptions about the probability distribution of failures like Young's Work and optimizes the number of check-points and position of checkpoints.
- **Pros:** As Distributively Managed Network File System (DM-NFS) is used to store the checkpoints, so it provides reliability as well as flexibility as shared memory storage and also it does not has single bottleneck failure. Hence checkpointing costs can be reduced significantly
- **Future Work:** In future method can be improved to better suit applications with high performance computing like MPI programs having extremely large scales.

## 3.5 Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud

[25]

- **Author:** Sangho Yi and Derrick Kondo, Artur Andrzejak
- **Year:** 2010
- **Transaction:** IEEE 3rd International Conference on Cloud Computing

- **Summary:** Amazon Elastic Cloud Compute (EC2) uses checkpointing mechanism to offer high reliability at low cost and volatility of reource provisioning. Various dynamic checkpointing strategies uses real price trace of amazon's spot instance to adapt price of current instance.

- **Pros:** There are various dynamic checkpointing strategies available such as,

  - Hour-boundary Checkpointing
  - Rising edge-driven Checkpointing
  - Checkpointing with Adaptive Decision
  - Checkpointing Combinations

  Selecting an appropriate method of Checkpointing significantly reduce the price as well as the task completion time.

- **Future Work:** Future work includes identification of correlations between current and past prices, between types of instance, and between rising edges. A robust prediction method can also be developed to decrease costs of monetary and completion times.

## 3.6 Coordinated checkpoint versus message log for fault tolerant MPI

[5]

- **Author:** AurelienBouteiller, PierreLemarinier, Geraud Krawezik, Franck Cappello
- **Year:** 2003
- **Transaction:** IEEE International Conference on Cluster Computing (IEEE)
- **Summary:** Comparision is made between two checkpointing method namely,

  - Coordinated Checkpoint - It is mechanism of fault tolerance in which checkpoints for all processes executing are coordinated and stored, so that in case of failure the task can be restarted from the latest checkpoint. Its performance highly depends on the location where check-pointing images are stored locally or remotely on independent server.

- Message Logging associated with Uncoordinated Checkpoint - In this method of fault tolerance all processes executing can make a checkpoint without being coordinated. Execution of process is supposed to be piece-wise deterministic, that means it is governed by its message receptions. So all these communications are stored in stable media so that it is possible to rollback only the crashed processes to a precedent local snapshot and execute the same computation as in initial execution.

Based on evaluation results of both method it is discovered that message logging has higher performance over coordinated checkpointing even with a frequency of one fault per hour.

- **Pros:**
  - Coordinated Checkpoint- It offers a low overhead on fault free execution.
  - Message Logging- It allows to checkpoint individual processes in an uncoordinated manner w.r.t other processes, so snapshot processes are taken and on failure only that particular process is being restarted from the latest checkpoint not all the processes are being restarted from a global checkpoint.
- **Cons:**
  - Coordinated Checkpoint- It offers comparatively high synchronization cost before checkpoint, cost of synchronized checkpoint and cost of restart after fault.
  - Message Logging- It has significant message transfer overhead even if no fault occurs.

## 3.7 DMTCP:Transparent Checkpointing for Cluster Computations and the Desktop

[1]

- **Author:** Jason Ansel, Kapil Arya, Gene Cooperman
- **Year:** 2009

- **Transaction:** IEEE

- **Summary:** DMTCP is a Distributed Multi-Threaded CheckPointing system, that provides checkpointing facility for distributed applications running on clusters as well as for desktop applications. It provides support for critical feature of transparency as there is no requirement of re-compilation and re-linking of user binaries. Also binaries uses no root privileges and can efficiently add the capability of save/restore to an application.

- **Pros:** It uses forked checkpointing in which a child process is forked and does the checkpointing task while execution of applications continues by parent process, leveraging UNIX copy-on-write-semantics. Hence the time required for writing checkpoint to disk is totally eliminated as the process of checkpointing is executed in parallel.

- **Cons:** It has a demerit that as it uses forked checkpointing, using which the compression runs in parallel with the executing user application this may slow down the user process, which may require longer thwn usual.

- **Future Work:** In future, support can be provided for new models of communication such as Remote Direct Memory Access (RDMA) and multicast, as used in networks such as InniBand. Also in future, extended support can be provided to checkpoint X-Windows applications, as it is currently demonstrated by the simple checkpointing of TightVNC.

## 3.8 Hybrid Checkpointing for MPI Jobs in HPC Environments

[22]

- **Author:** Chao Wang, Frank Mueller, Christian Engelmann, Stephen L. Scott

- **Year:** 2010

- **Transaction:** 16th International Conference on Parallel and Distributed Systems (IEEE)

- **Summary:** In high performance computing number of cores used much high. Also frequency is high, which requires checkpointing for fault tolerance. Although a subset of the process image changes between the subsequent checkpoints whole process image is taken. In the proposed hybrid checkpointing technique, alternatively incremental and full checkpointing is carried out.

- **Pros:** It uses an incremental checkpointing technique alternatively with full checkpointing, which requires to capture only the data that changed since last checkpoint captured. Hence it results in reduced checkpoint sizes, which in turn reduces the cost and overhead associated with checkpointing.

- **Cons:** It has demerit that as it uses incremental checkpointing so the number of checkpoints required to restart the process/jobs incase of a failure is total number of incremental checkpoints after the last full checkpoint and also the last full checkpoint, which may increase the restart time of process/job incase of a faillure.

## 3.9 Towards Scalable Application Checkpointing with Parallel File System Delegation

[3]

- **Author:** Dulcardo Arteaga, Ming Zhao

- **Year:** 2011

- **Transaction:** Sixth IEEE International Conference on Networking, Architecture, and Storage(IEEE)

- **Summary:** A new technique named Parallel File System (PFS) is proposed, which delegates the PFS storage space management, used to checkpoint applications. So PFS system is relieved from the load of metadata operations during their checkpointing.

- **Pros:** Parallel File System (PFS) based checkpointing gradually reduces number of metadata operations handled by metadata servers during process of checkpointing, as it provides a view of single logical file to the metadata server.

- **Future Work:** : As in experiments, resources used are only up to 128 clients and 4 servers simultaneously, but the same type of advantage can be obtained by delegation used in PFS, when the count of clients and servers scale up proportionally in larger HPC system. So in future evaluation can be done on real production HPC system which has much larger scale testbed used.

## 3.10 Thread-Based Live Checkpointing of Virtual Machines

[19]

- **Author:** Vasinee Siripoonya, Kasidit Chanchio
- **Year:** 2011
- **Transaction:** IEEE International Symposium on Network Computing and Applications (IEEE)
- **Summary:** Proposed technique of Thread-based Live Checkpointing (TLC) mechanism takes advantage of pre-copy live migration mechanism which introduces a checkpoint thread, that is responsible for the most of the checkpointing activities. In this mechanism the checkpoint thread continues saving the virtual machine state to some persistent storage, while the virtual machine thread is allowed to simultaneously progress with normal execution.
- **Pros:** It is similar to concurrent checkpointing but it has advantage over concurrent checkpointing that it uses incremental checkpointing which periodically copies only the changed pages to a hash table rather than copying an entire page to the buffer on each memory write operation, so the over head of checpointing is reduced.
- **Cons:** For higher memory updates in locality the incremental checkpointing incurrs higher cost of checkpointing.
- **Future Work:** In future support can be provided to implement Thread based Live Checkpointing for multiprocessor VMs, that would automatically switch TLC operation to T-onesave when high memory update locality is detected based on some set of rules.

## 3.11 Predicting Job Start Times on Clusters

[19]

- **Author:** Hui Li, David Groep, Jeff Templon, Lex Wolters
- **Year:** 2004
- **Transaction:** IEEE International Symposium on Cluster Computing and the Grid
- **Summary:** Start time of job is predicted based on statistical analysis of historical job traces and simulation of schedulers at different sites.
- **Pros:**
  - Using such prediction of job start time middleware component such as resource broker can balance workload distributions.
  - Also it can be used to compute price of resources in a grid accounting system.
- **Cons:** It runs slower than the EDG solution because of the statistical predictions and simulations involved. Although we have made the simula- tion event-driven, the simulation time grows linearly as the number of queued jobs increases. Also, the system assumes that the site does not employ dynamic scheduling policies.
- **Future Work:** In future further statistical properties of EDG workloads can be studied and evaluation of more estimators for run time predictions can be done.

## 3.12 Problem Statement

Provisioning of resources in cloud is crucial task as demand of resources varies continuously. Preparing for peak load capacity would lead to wastage of resources as the resources will remain idle when system is not used to its full capacity, whereas preparing for average load would lead to violations of SLAs with customers. In-order to provision resources with low cost for CSP and at the same time maintaining the SLAs with customers, resources provisioning should be done judiciously. To solve this problem of resource provisioning, advance reservation of resources can be used effectively.

In this technique resources are provisioned in such a manner that it benefits customer by providing guaranty of resource availability and at the same time benefits CSP by providing control to service all customers with the available resources efficiently.

# Chapter 4

# Proposed Design

## 4.1 Proposed Working Model

In the given figure 4.1 it describes proposed working model, which include the concept profiling and check-pointing for reservation of resources in the cloud environment. As figure 4.1 shows assuming that, number of virtual machines listed on one host and such multiple hosts in one data center, this altogether forms one cloud. It includes modules such as Profiling resource usage and monitoring task completion, Resource allocator, Task Scheduler, Task Analyzer and Task scheduling algorithm library description of them is provided below.

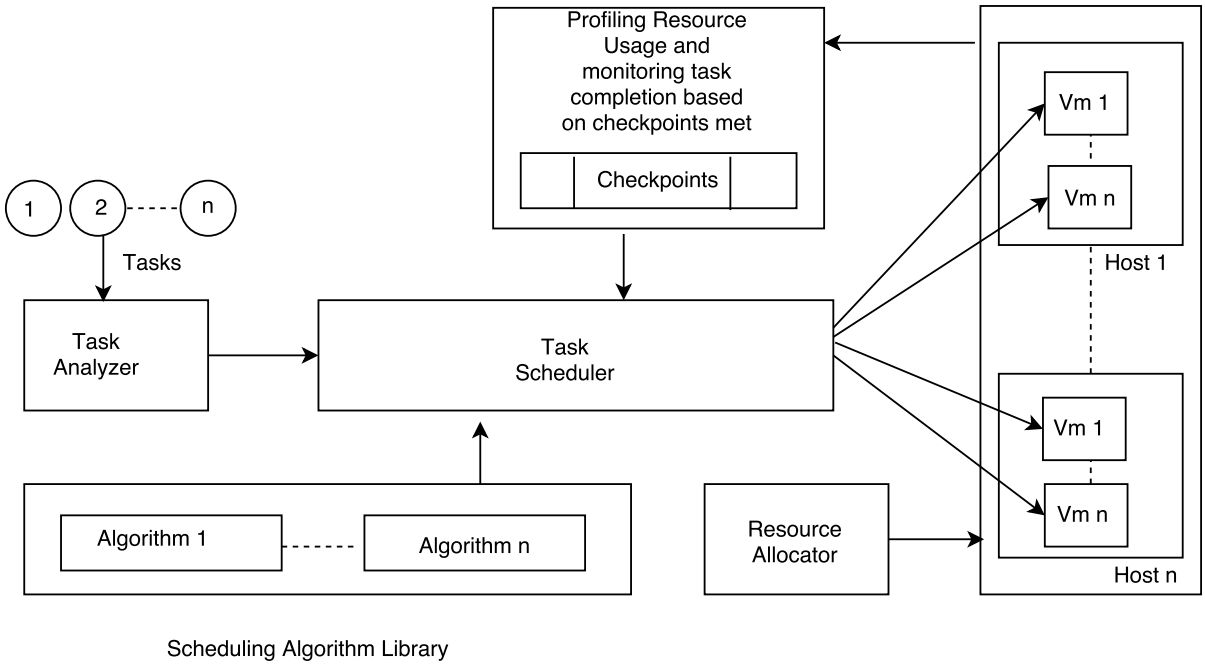



Figure 4.1: Proposed Working Model

Description of modules represented in Proposed Working Model are as follows:

- Profiling and Resource Monitoring: Functionality of this module is to compare new task submitted to cloud with tasks in historical data set of resource usage profile and to find appropriate resource usage pattern. Then monitor the completion of tasks based on checkpoints met according to the matched resource usage profile.

- Task Analyzer and Scheduler: Task Analyzer's functionality is to gather information of new tasks submitted to cloud. It checks the requirements of the task like arrival time of the task, deadline of the task and execution time of the task and so on. Then it passes this information to the scheduler, which schedules the tasks on VM's based on availability of resources.

- Scheduling Algorithm Library: Scheduling Algorithm Library consists of multiple scheduling algorithm from which a scheduling algorithm is chosen according to resource availability.

- Resource allocator: Functionality of Resource Allocator is to allocate VM's to the task based on resource usage requirement of tasks and availability of tasks. If some task is unable to complete its execution within expected amount of time, then new VM will be created as per the tasks requirement by resource allocator. Also if resource allocator find some of the VM's are lightly loaded or may be idle then it consolidates tasks of such VM's thereby efficiently utilizing the available resources.

## 4.2 Proposed System Flow

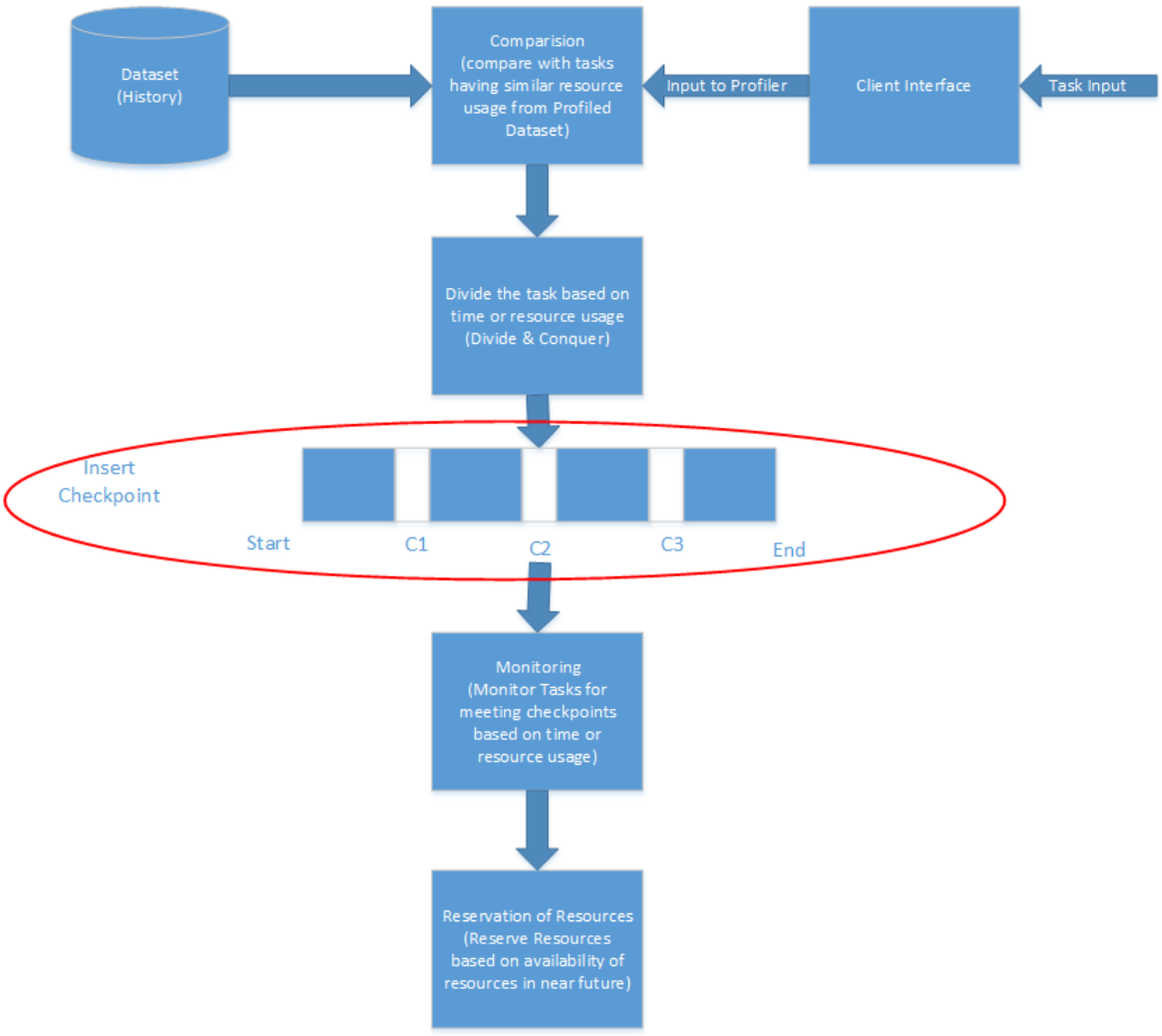



Figure 4.2: Proposed Flow Model

## 4.3 Steps for Implementation

- Take input from user interface for newly arriving tasks.
- Compare resource usage requirement of task submitted to cloud with historical data of resource usage profile to find tasks with similar resource usage requirement(Profiling).
- Determine estimated time required by task submitted to cloud from the tasks in history having similar resource usage profile.
- Divide the new task submitted to cloud based on time and place checkpoints at time when a smallest unit of divided tasks is completed.
- Profile the execution of the newly arrived tasks and also simultaneously monitor the tasks for its partial completion and meeting checkpoints.
- Reserve the resources for next arriving task after the currently executing tasks reaches certain threshold of its execution.

## 4.4 Check Pointing Mechanism

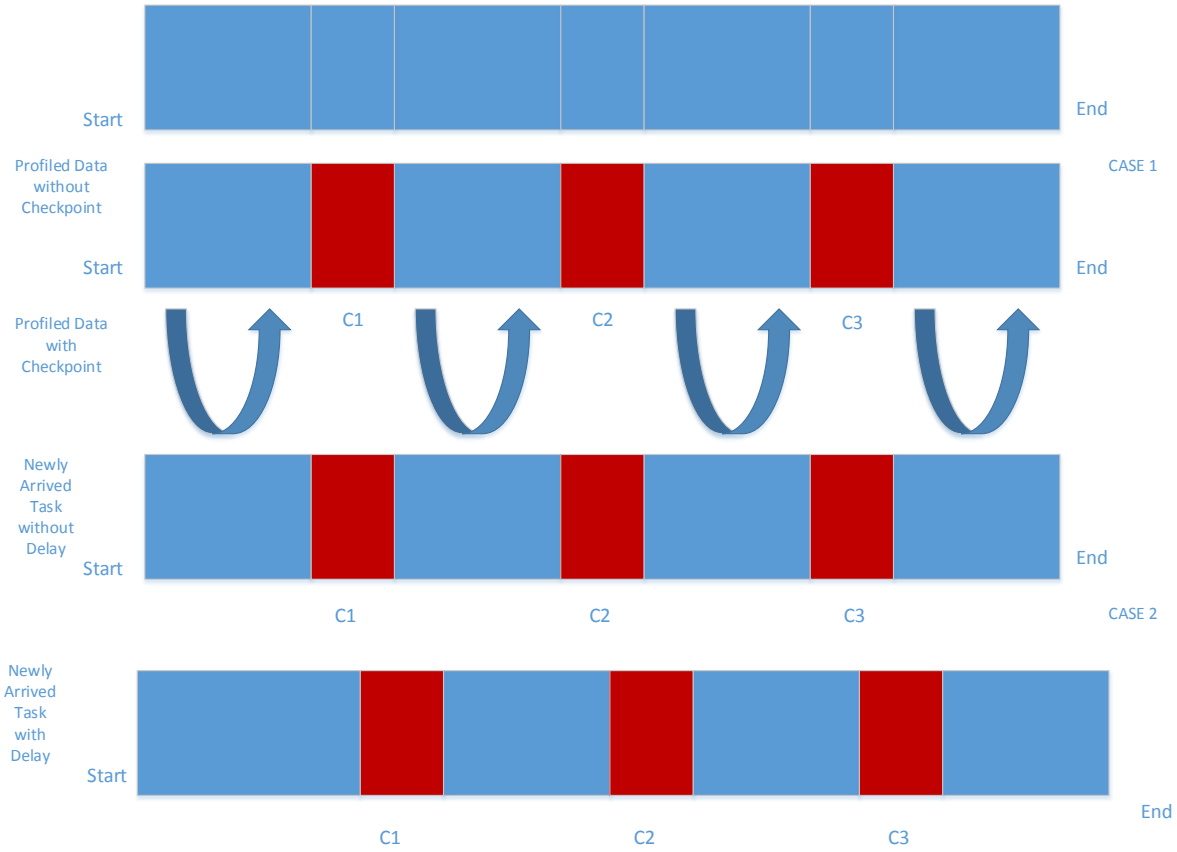



Figure 4.3: Checkpointing Mechanism Model

As shown in Figure 4.2 there are two cases represented which are as follows,

- **Case 1:** Initially checkpoints are placed in profiled data, so that it can be used to compare the completion of new task submitted to the cloud which have similar resource usage requirement.
- **Case 2:** Comparison is being done between profiled task and newly arrived task to get status of completion of newly arrived task, based on the checkpoints met by gradual completion of the task. Hence if a delay is measured in meeting the checkpoints than it can be predicted in advance that completion of task will suffer a delay. Also if checkpoints are met in time then accuracy of prediction about the completion of task gets higher gradually.

# Chapter 5

# Proposed Algorithm

## 5.1 Task Mapper

**Algorithm 1** Task Mapper

```
1: Initialization
2: for each (Task[i] in NewlyArrivedTasks[])
3: if (There is a VM[j] available in AvailableVMCapacity[] then) then
4:     Assign Task[i] to VM[j] for execution and update status of VM[j]
5:     Delete Task[i] from NewlyArrivedTasks[]
6: else if (No VM available in AvailableVMCapacity[]) then
7:     Append NewlyArrivedTasks[] to TaskQueue[]
8:     AdvanceReservation(TaskQueue[])
9: end if
```

As described in algorithm 1 Task Mapper, which maps the newly submitted cloudlets to a particular virtual machine. For each newly submitted tasks it checks if there any virtual machine available according to the resource requirement of tasks. If it finds an virtual machine available then it assigns the task to that virtual machine, else if no virtual machine is available then it invokes the Advance Reservation algorithm.

## 5.2 Advance Reservation

**Algorithm 2** Advance Reservation

```
1: Initialization
2: for each (Task[i] in RunningTaskList[]) do
3: if (! look for task to similar Task[i] found in Resource_Usage_Profile) then
4:    do profiling of Task[i] and append to Resource_Usage_Profile
5:    insert Checkpoints in Task[i] using Logs
6: else
7:    RemainingTime=updateCompletion(Task[i])
8:    if (RemainingTime < (estimatedTaskTime(Task[i])*0.20)) then
9:       Checkpoint 4 Met
10:      notify broker
11:      for each (Task[j] in TaskQueue[])
12:      if (Resource Requirement of next Task[j] in TaskQueue[] <= CheckResourceU-
         tilization(Task[i])) then
13:         //Reserve Resources for Task[j] in TaskQueue[]
14:         Assign Task[j] to VM on which Task[i] is executing
15:         Delete Task[j] from TaskQueue[]
16:         Append Task[j] to ReservedTask[]
17:         break;
18:      else
19:         wait() //wait for some more resources to be
20:      end if
21:   else if (RemainingTime < (EstimatedTaskTime(Task[i])*0.40)) then
22:      Checkpoint 3 Met
23:      notify broker
24:   else if (RemainingTime < (EstimatedTaskTime(Task[i])*0.60)) then
25:      Checkpoint 2 Met
26:      notify broker
27:   else if (RemainingTime < (EstimatedTaskTime(Task[i])*0.80)) then
28:      Checkpoint 1 Met
29:      notify broker
30:   end if
31: end if
```

As described in algorithm 2 Advance Reservation, which reserves resources for the task according to its resource requirements. For each task that is in execution it monitors the completion of task using the checkpoints available in resource usage profile of similar task in history. Based on checkpoints met by tasks in execution, completion time of task when resource will be released can be predicted. So predicting the release time of resource, next task in queue whose resource requirement can be satisfied by the resource that is likely to be released in short duration, is assigned to that particular virtual machine. In this

manner resources can reserved for tasks in queue.

## 5.3 Methods used in Algorithm

**Algorithm 3** updateCompletion

```
1: updateCompletion(Task[i])
2:     return reamainingTime=estimatedTaskTime(Task[i])-(currentTime-startTime)
```

**Algorithm 4** checkResourceUtilization

```
1: checkResourceUtilization(Task[i])
2:     return resource specifications of VM on which Task[i] is executed
```

**Algorithm 5** makeCheckpoint

```
1: makeCheckpoint()
2:     resCheckpoint[cpu,ram,bw,exe_time]
3:     resCheckpoint[0]=cpu_Utilization for Task[i]
4:     resCheckpoint[1]=ram_Utilization for Task[i]
5:     resCheckpoint[2]=remaining_Cloudlet_Length for Task[i]
6:     resCheckpoint[3]=execution_Time for Task[i]
7:     add resCheckpoint to list of checkpoints for Task[i]
```

In algorithm 3 is described Update Completion a method used in Advance Reservation algorithm, which provides status of executing task in terms of the estimated remaining time for the completion of tasks.

In algorithm 4 is described Check Resource Utilization method used in Advance Reservation algorithm, that check gives details about the resource specifications of the virtual machine on the current task is executing.

In algorithm 5 description is provided about the Make Checkpoint method being used in advance Reservation algorithm. This stores resource usage at particular tme instances as checkpoints. Resources considered for making checkpoints at different instances of time are CPU and RAM, also the remaining cloudlet length at that particular time instance is recorded as part of the checkpoint.

## 5.4 Flowchart of Algorithm

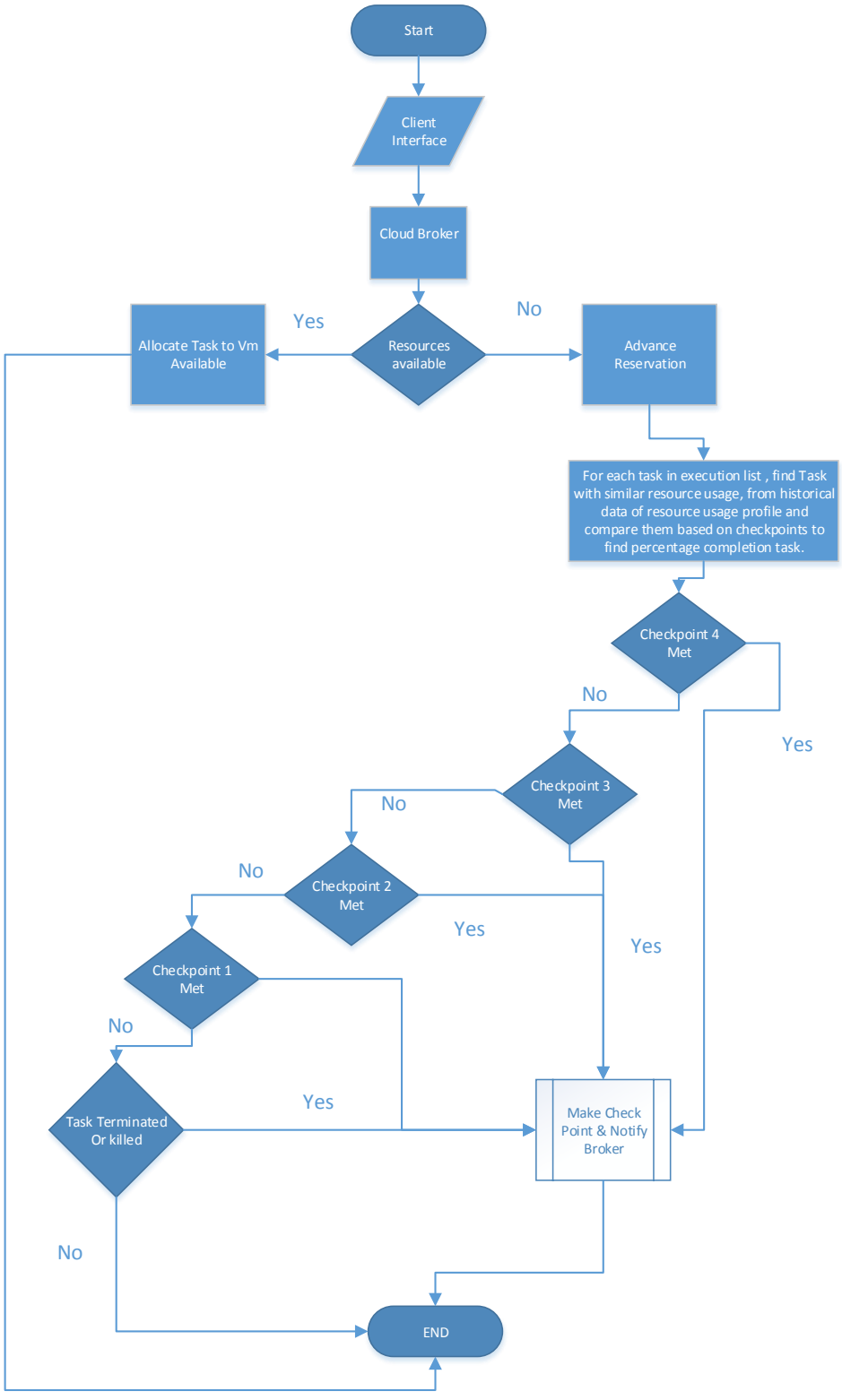



Figure 5.1: Flowchart Diagram based on Algorithm

As shown in figure 5.1 flowchart diagram of algorithm is represented, which describes working flow of the proposed system which is as follows,

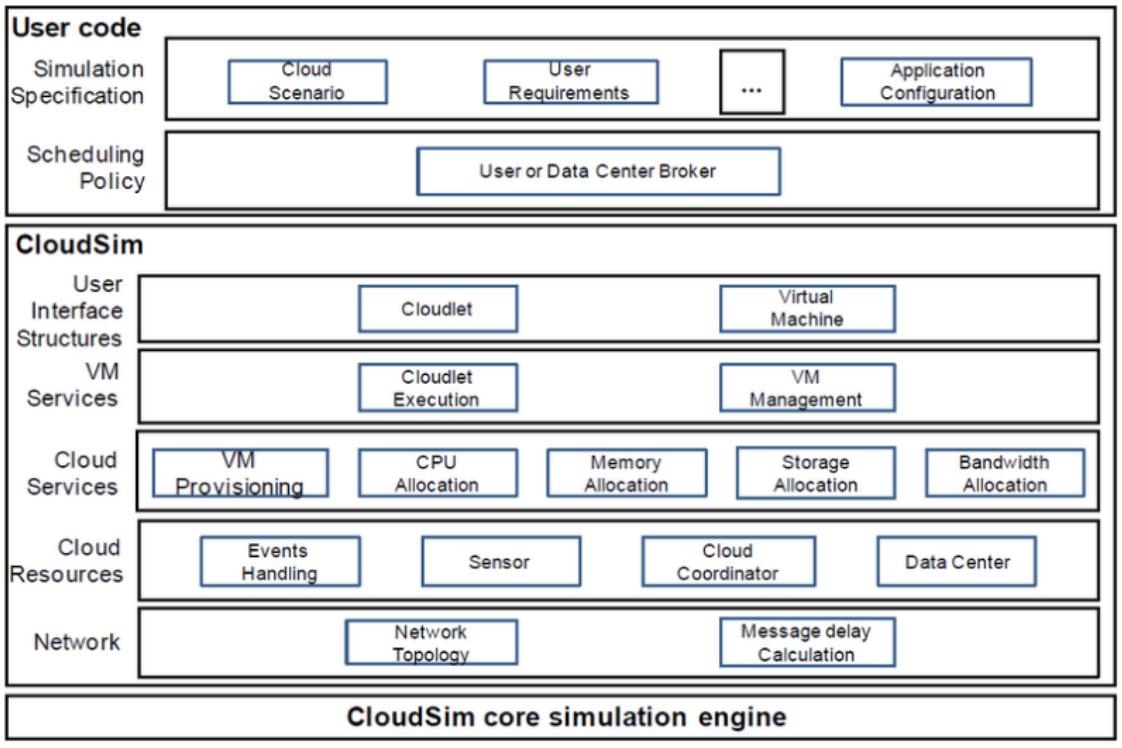
- New task to be executed are submitted to the cloud environment using the client interface.
- Tasks submitted through client interface are received by broker, which has information of both the number of tasks submitted and the number of virtual machines that are available in cloud environment.
- Using the information available broker assigns new tasks submitted to cloud if enough resources are available, else new tasks are queued and the advance reservation algorithm is invoked.
- Advance Reservation algorithm checks for each task in execution list , find Task with similar resource usage from historical data of resource usage profile.
- Then based on checkpoints available in resource usage profile of the similar tasks in history the task in execution is being monitored to find the number of checkpoints met.
- Based on the checkpoints met prediction can be made regarding the remaining time for task completion and time at which resources executing the current task will be released.
- Suppose all previous checkpoints for a task are met on time then it can be predicted that regular execution of task will be completed by the time it meets the last checkpoint. And if there is a delay in meeting any intermediate checkpoint then it is clear that final completion of task will also be delayed by the same amount of time.
- As each checkpoint broker is notified for meeting the checkpoint, so broker has an estimate about completion time of task when resource executing task will be released. Using such an estimate it assigns the new tasks to virtual machine that can satisfy its resource requirements and that are likely to start executing task as soon as possible.

# Chapter 6

# Tools and Techniques

## 6.1 CloudSim

CloudSim simulator is simulates the cloud environment and provide the faciltiy to get the results similar to applications executing in real cloud. So for implementation of the proposed model CloudSim provides a better option. Architecture of CloudSim simulator is as shown in figure 6.1,



[8]

Figure 6.1: CloudSim Architecture

Some of the important entities of CloudSim architecture are as follows,

- **Data Center:** It is used to model the system level core services of cloud infrastructure. At least one data-center must be created for starting the simulation process. Data-Center consists of set of hosts which represents the physical machines and each host can manage multiple virtual machines based on resources available. Its the functionality of virtual machines to handle the low-level processing.

- **Host:** It is component that actually represents the physical machines in the data-centers. It is used to allocate processing capabilities which is defined in unit of millions of instructions per second, memory and the scheduling policy used in allocating different processing cores to multilple virtual machines which are managed by the particular host.

- **Virtual Machine:** Virtual Machines are responsible execution of the tasks submitted to clouds. Different number of virtual machines are allocated to different hosts, so that the processing cores can be scheduled to virtual machines, which is managed by this component. The configuration depends on particular application, but by default allocation of virtual machines is based on "first-come first-serve" policy.

- **Datacenter broker:** It act as mediator between clients and the Cloud Service Providers (CSP) to map the tasks submitted to the cloud to appropriate virtual machine based on its resource requirements.

- **Cloudlet:** It is responsible for modelling the application service based on computational requirements in CloudSim.

- **CloudCoordinator:** Responsibility of this component is to manage the communication with brokers and other Cloud Coordinators. Also it monitors internal state of data center periodically in terms of simulation time.

The CloudSim library is used for the following operations:Extensive scale distributed computing at server farms ,Virtualised server has with customisable arrangements .Support for demonstrating and reenactment of substantial scale distributed computing server farms. Support for displaying and reenactment of virtualised server has, with customisable approaches for provisioning host assets to VMs .Support for demonstrating and

recreation of vitality mindful computational assets . Bolster for displaying and recreation of server farm organize topologies and message-passing applications. Bolster for displaying and reproduction of combined mists. Bolster for element addition of reproduction components, and ceasing and continuing recreation. Bolster for client defned strategies to apportion hosts to VMs, and approaches for dispensing host assets to VMs. User-defned arrangements for portion of hosts to virtual machines.The real restriction of CloudSim is the absence of a graphical UI (GUI).

## 6.2 WEKA

WEKA is an acronym for Waikato Environment for Knowledge Analysis. It consists of a bundle of machine learning algorithms which are implemented in Java, developed at the University of Waikato, New Zealand. Its library contains a collection of algorithms for data analysis and predictive modelling. It also provides visualization tools which provides graphical user interfaces for easy access to these functions.

WEKA provides support for several data mining tasks like data pre-processing, classification, clustering, regression, attribute/feature selection, and visualization. For dataset WEKA supports file of type .arff, .csv, which describes data-points by a fixed number of attributes. It also provides SQL database by using Java DataBase Connectivity (JDBC), and is able to process the results returned by database query.[23]

## 6.3 K-Means Clustering

For clustering of tasks into groups k-means clustering algorithm which is available in WEKA library under the name "SimpleKmeans" is used. Originally derived from area of signal processing, which popular for analysis of clusters in data mining, k-means clustering is a technique for vector quantization. The technique used in k-means clustering aims to divide n number of observations into k clusters(groups), where k¡n. Here each observation belongs to the cluster with nearest mean value. Mean value of clusteris known as centroid of the cluster which serve as prototype of the cluster.[14]

Initially k random seeds are choosen as centroids for the k clusters and then at each

iteration distance between cluster centroids and observation is measured and based on that observations are assigned to the clusters with nearest mean. Also at each iteration the centroid value is computed based on the values of observation in the particular cluster. Such an iterative process is repeated until either a fixed threshold value for number of iterations is reached or no observation in the clusters formed change the cluster based on new computed value of cluster centroid for at least two iterations.

Distance function used in calculating the distance between the cluster centroid and observations in the dataset is either Euclidean distance or Manhattan distance.

## 6.4 Implementation Details

As discussed above CloudSim is the tool used, which provides simulation of cloud environment and WEKA is the library of machine learning and dat mining algorithms which is used for the purpose clustering of tasks in groups. Details of exactly how and where the use of the above mentioned tools is made is provided in the specifications which are discussed below.

Firstly, specifications of the components like data-centers, hosts and virtual machines which are created in CloudSim simulator which provides cloud environment are discussed. Specifications of Data-Centers (DC) and Virtual Machines created in simulated cloud environment provided by CloudSim are as follows:

Three Data-Centers are created in the simulation environment with four hosts in each Data-Center. Specification of hosts are as follows:

- RAM - 16384 MB
- Storage - 1000000 MB
- Bandwidth - 10000 Kbps
- MIPS - 1000
- cores/processing elements - octa, quad, dual, dual (one octa, one quad core and two dual core machines)

Specification of Virtual Machines created are as follows:

- size - 10000 MB

- RAM - 512 MB
- MIPS - 1000
- Bandwidth - 1000 Kbps

For experimentation purpose data set used is San Diego Supercomputer Center (SDSC) SP2 log, available at URL: "http://www.cs.huji.ac.il/labs/parallel/workload/logs.html". [20] Using the specified data set simulation is done in CloudSim simulator, which provides cloud based environment for execution of tasks. Output of simulation is as shown in figure 6.2.

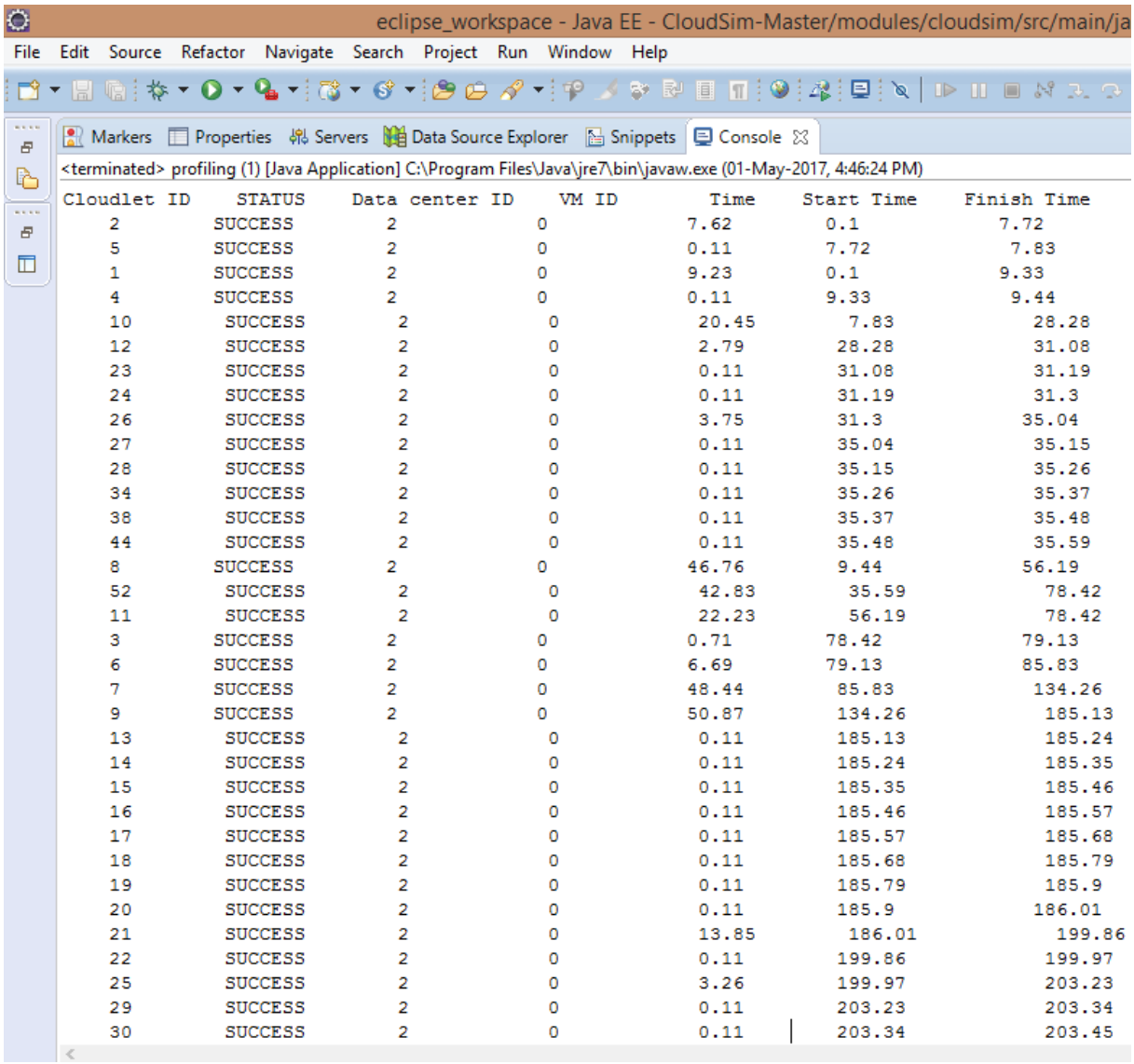



Figure 6.2: Output of Simulation

Initially as new incoming tasks are executed, a profile of resource usage of tasks is generated by monitoring resource utilization at different instances in time. Hence history of resource usage profile is created, which is then used to monitor newly submitted task to the cloud for its completion of execution, based on the checkpoints met while executing newly submitted tasks.

As shown in figure 6.3, graph of resource usage profile is presented, which shows resource utilization of tasks at different instances of time, namely Time Instance 1 to Time Instance 5.

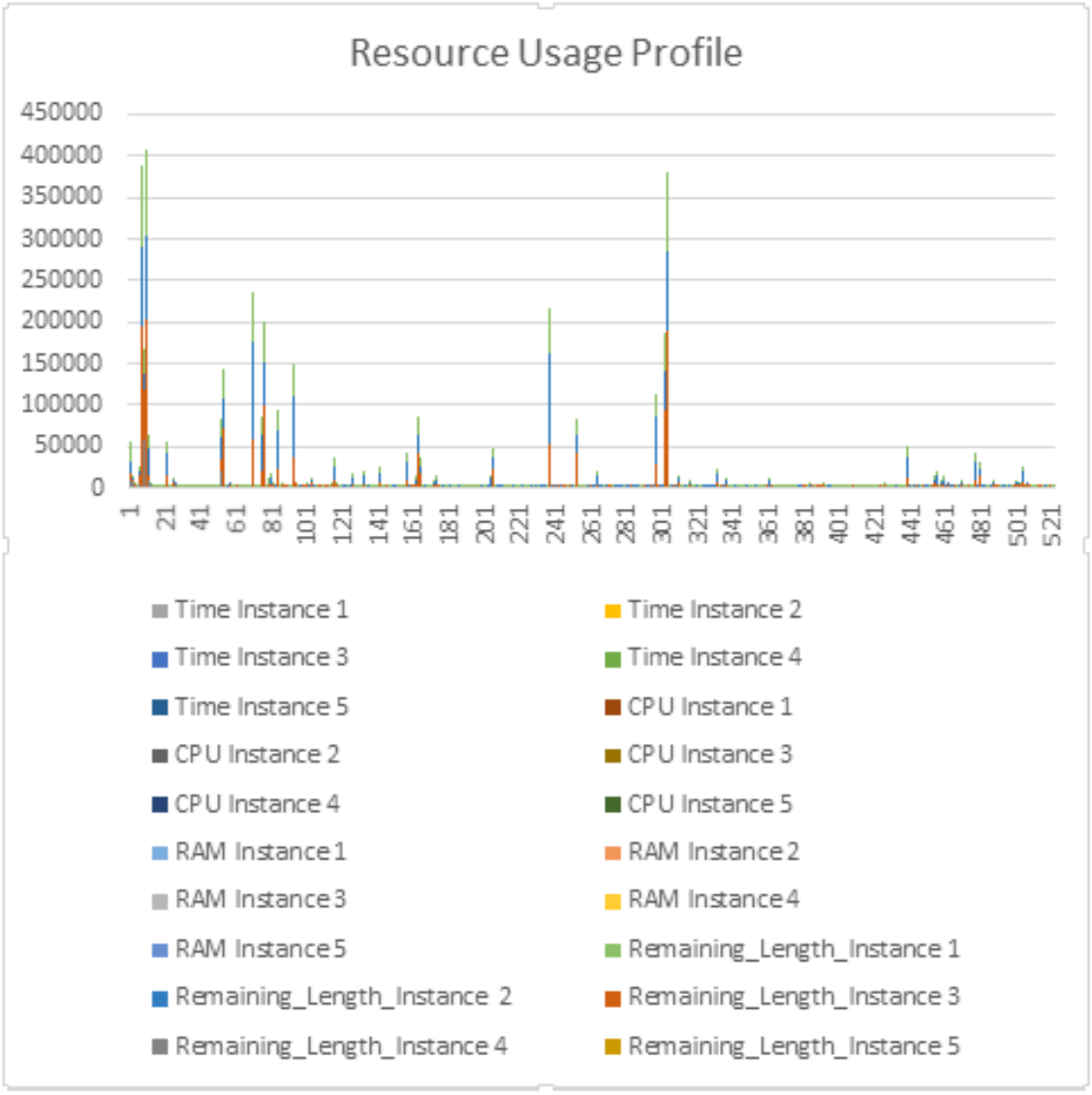



Figure 6.3: Profile of Resource Usage

Here resources considered for resource usage profile are CPU and RAM, which are represented by CPU Instance 1 to CPU Instance 5 and RAM Instance 1 to Ram Instance 5 respectively, corresponding to measures of CPU and RAM at different instances of time from Time Instance 1 to Time Instance 5. Remaining Length Instance 1 to Remaining Length Instance 5 corresponds to amount of tasks completed at different instances of time from time Time Instance 1 to Time Instance 5.

Hence, when a new tasks is submitted to the cloud, a search is made for category of tasks with similar resource requirement. For finding category of newly submitted tasks, K-means clustering algorithm is used. Clusters are formed using historical data of resource usage profile and when a new tasks arrives the model is used to find appropriate cluster to which the tasks belong. As shown in figure 6.4, details about clusters formed using SimpleKmeans algorithm is represented.

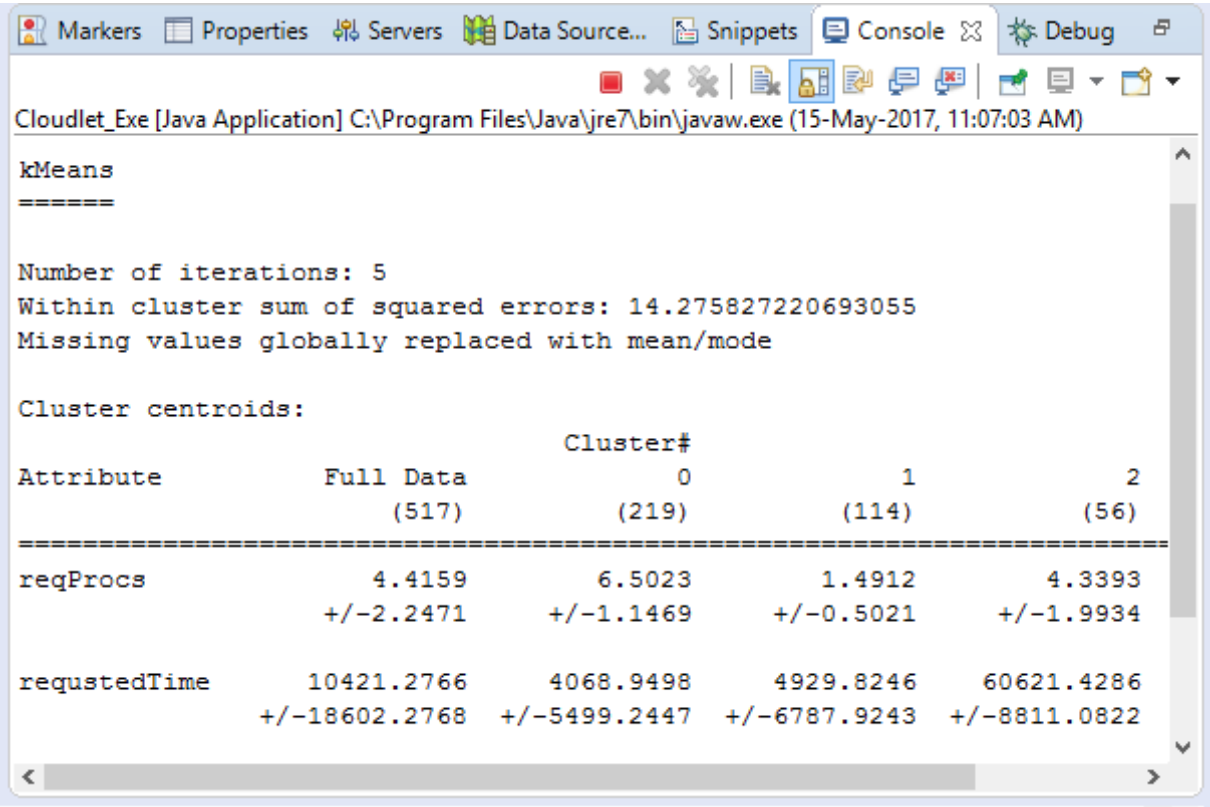



Figure 6.4: Clustering of Tasks using K-means

As it can seen from figure 6.4 number of iterations performed is 5 and within cluster sum of squared errors is approximately 14.2758. Parameters used for the purpose of

clustering are requested processors and requested time, as this two parameters are available in a cloudlet request before execution of cloudlet begins. Hence when new tasks are submitted to the cloud mapping of the task to cluster can be done and then pattern of resource usage is matched with tasks in that particular cluster to find a similar task.

As shown in figure 6.5, graphical user interface of weka tool represents clusters formed using SimpleKmeans algorithm available in weka library. X-axis represents required number of processors and Y-axis represents required time. It can be seen from figure 6.5 that four tasks are being divided into four clusters(groups) based on based onn the selected parameters i.e. requested number of processors and requested time, as this two parameters are available in resource requirement specifications. Hence when new tasks arrives it with its resource requirements specification, it can be associated with one of the cluster based on the parameters of requested number of processors and requested time.


Figure 6.5: Graphical Representation of Clustering of Tasks using K-means

Once clusters are formed using resource usage profile of tasks in training data-set, then as new tasks are submitted to cloud for execution they are being associated to one of the clusters based on parameters used in the cluster formation. After associating the new task with a particular cluster, pattern of their usage is compared with resource usage profiles of the tasks available in the historical data-set. Resource usage at different time instances is compared between the tasks newly arrived and tasks in historical data-set belonging to same clusters. Based on such comparison it is determined that checkpoints placed in resource usage profile of which task in historical data-set is met by the new task in execution.

Hence, such task is determined whose checkpoints coincides or is range of one standard deviation from the checkpoint with new task in execution then it can be further checked if next checkpoint also coincides with this task in execution. So based on such comparison it can estimated that the completion time of similar task in the historical data-set will also match with task in execution. So in this manner completion time of the task in execution can be estimated and which will provide the time for release of resources being used by the task and this can estimated time can be used to reserve the resources for the next task in the queue.

For implementation resource usage profiles of about 400 tasks is used and they are divided into clusters by using k-means clustering. Algorithm for k-means is used from WEKA library of machine learning and data-mining algorithms, which is named as "SimpleKMeans".

For testing purpose about 50 tasks requesting resources are submitted to the CloudSim simulator, which provides cloud environment to the tasks. The output of the execution of these tasks also known as cloudlets in the CloudSim environment are as shown in figure 6.6 and figure 6.7.

As shown in figure 6.6 the output of the execution of the 50 cloudlets in CloudSim environment is shown, when resource usage profiles of similar tasks were available in the historical data-set.

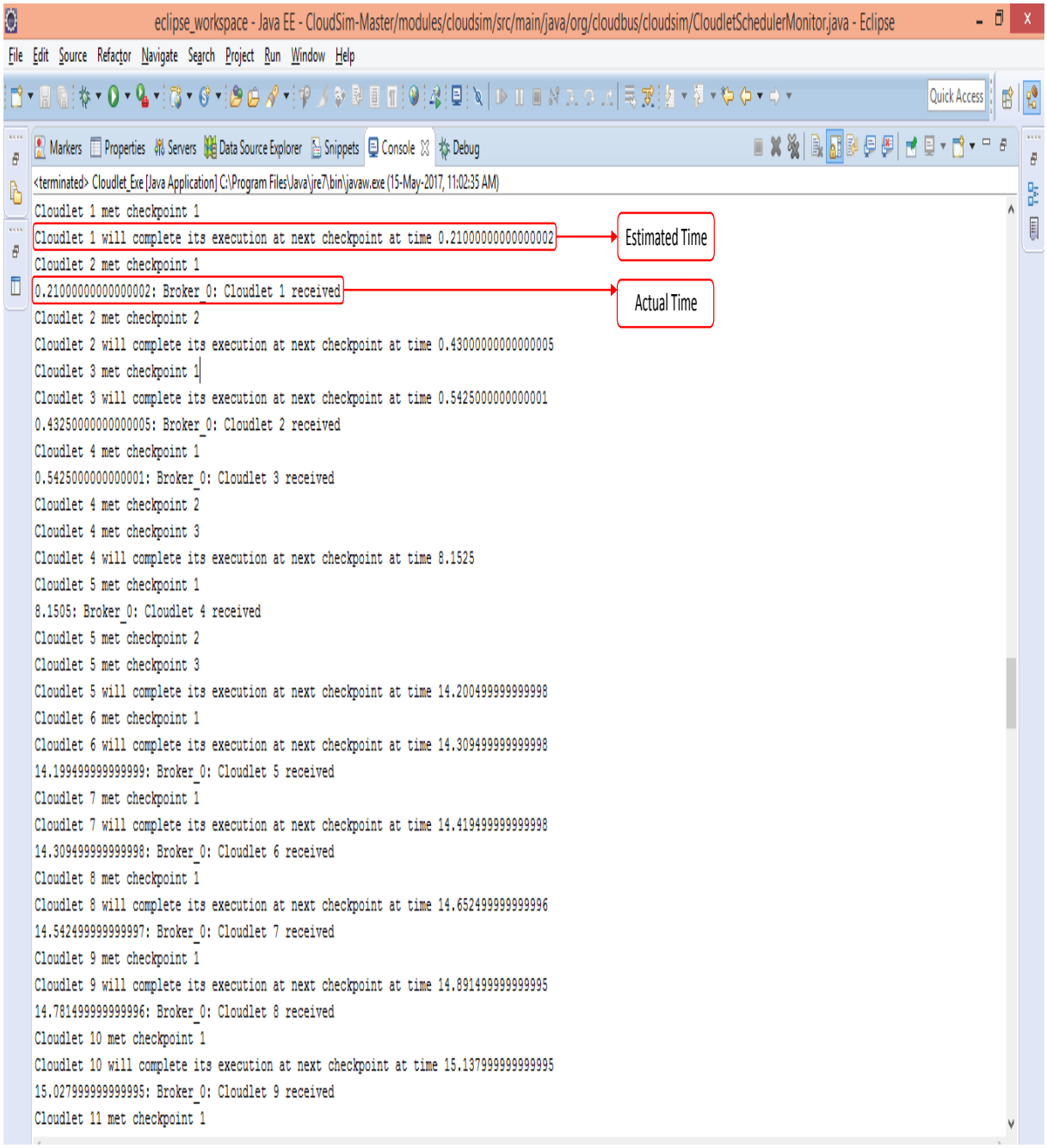



Figure 6.6: Output of execution of Tasks when Similar Resource Usage Profile is available

As it can seen from figure 6.6 that when tasks with similar resource usage profiles are available in the historical data-set then the estimation of the task completion time has good accuracy which is clear from the figure. Also if estimated time is not exactly the same then it is slightly higher than the actual time which ensures that the resources will be released by the estimated time, which can seen for cloudlet with id 4 in figure. For cloudlet 1 the estimated and actual time are almost same which is 1 0.21000000000000002s. While for cloudlet 4 the estimated time is 8.1525s while actual time is 8.1505s, hence estimated

time is slightly more then actual time.

Similarly figure 6.7 shows the output of the execution of the 50 cloudlets in CloudSim environment, but in case when no similar resource usage profiles ofvtasks were available in the historical data-set.

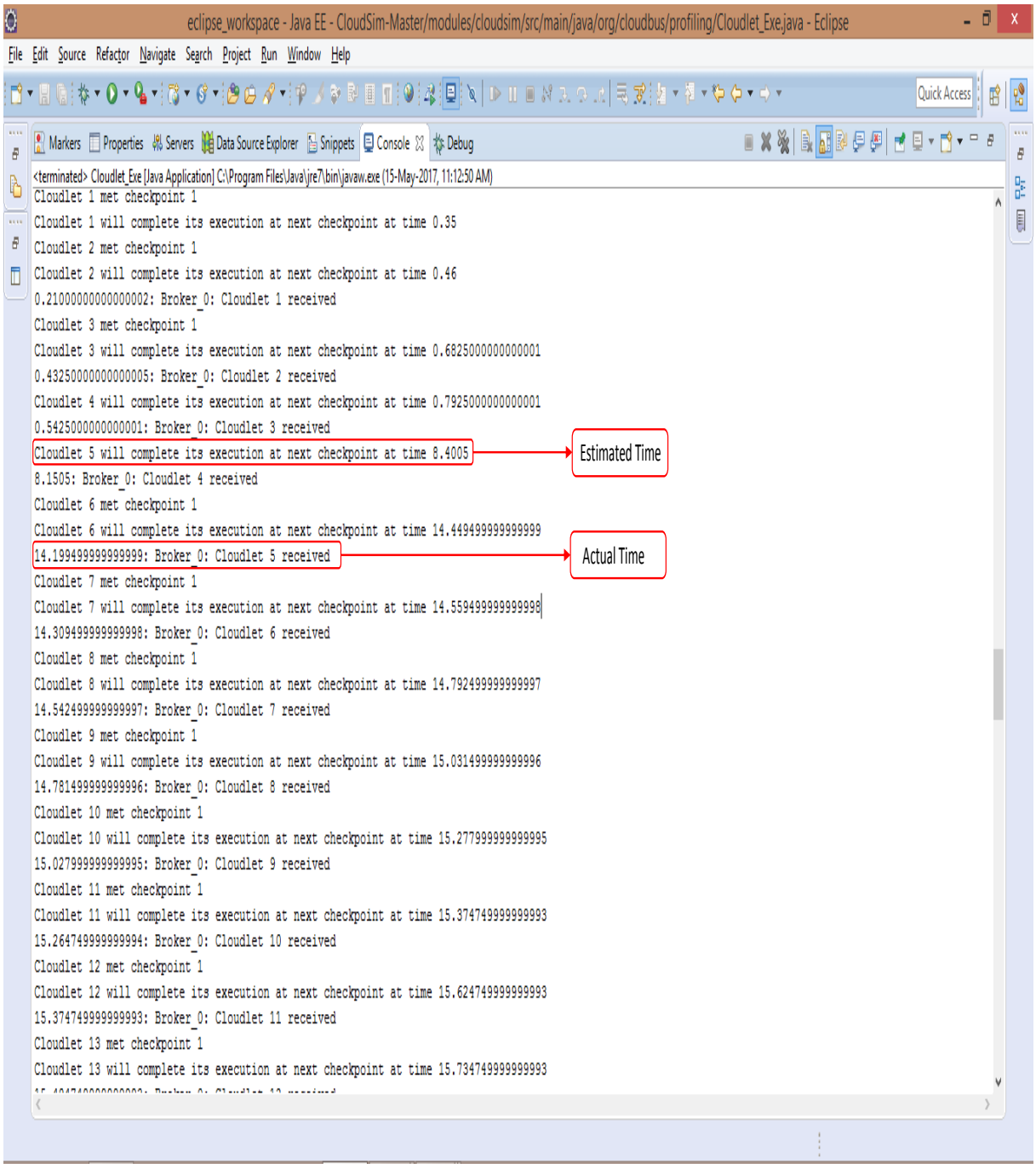



Figure 6.7: Output of execution of Tasks when Similar Resource Usage Profile is not available

As it can seen from figure 6.7 that when tasks with similar resource usage profiles are not available in the historical data-set then the estimation of the task completion

time does not have that good accuracy which is clear from the figure. Also in this case it is possible that estimated time may be less than the actual time which fails to ensure that the resources will be released by the estimated time, which can seen for cloudlet with id 5 in figure. For cloudlet 5 the estimated time is 8.4005s while actual time is 14.199499999999999s, hence estimated time is less than actual time which fails to ensure that resources will be released by the estimated time.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

Provisioning of resources is a difficult task, as planning for peak load leads to overprovisioning and planning for average load leads to under provisioning. Advance Reservation discussed here mitigates these problems of under-provisioning and over-provisioning by reserving resources for tasks. In technique discussed here advance reservation is used by estimating time required for release of resources that are currently executing the ongoing tasks and based on this, reserve the resources for new incoming tasks. Hence using such a reservation technique for resource provisioning, latency of tasks execution will be reduced.

## 7.2 Future Work

In future, focus will be on developing a system that is capable of incorporating tasks of new type which are currently not present in resource usage profile. Hence, when similar type of tasks arrives in future, resource usage profile of this task will be fetched from history and can be used to monitor completion of new tasks submitted to cloud as per the method discussed here. As well as testing can be done using different synthetic workload generators and its comparison with real time workload can be done.

# Chapter 8

# Related Publication

## 8.1 Published Paper

| | |
|---|---|
| Title | "Advance Resource Reservation based on Context Aware Workload" |
| Author(s) | Parimal Gajre, Vivek K. Prasad, Dr. Madhuri Bhavsar |
| Journal Name | International Journal of Advanced Research in Computer Science |
| Publisher Name | IJARCS |
| ISSN | 0976-5697 |
| Date | $26^{th}$ April, 2017 |
| SJIF | 5.845 |

## 8.2 Submitted Paper

| | |
|---|---|
| Title | "Capacity planning through monitoring the resources at IaaS in Cloud Computing" |
| Author(s) | Harshil Mehta, Parimal Gajre, Vidhi Sutaria, Vivek K Prasad, Dr. Madhuri Bhavsar |
| Conference Name | International Conference on Future Internet Technologies and Trends |
| Publisher Name | Springer |
| Status | Acceptence Awaited |
| Date | $10^{th}$ June, 2017 |
| Venue | SVNIT University, Surat |

# Bibliography


[1] Jason Ansel, Kapil Arya, and Gene Cooperman. "DMTCP: Transparent checkpointing for cluster computations and the desktop". In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–12.

[2] Michael Armbrust et al. "A view of cloud computing". In: *Communications of the ACM* 53.4 (2010), pp. 50–58.

[3] Dulcardo Arteaga and Ming Zhao. "Towards Scalable Application Checkpointing with Parallel File System Delegation". In: *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*. IEEE. 2011, pp. 130–139.

[4] *Basics of cloud*. URL: `https://www.google.co.in/search?q=basics+of+cloud&safe=strict&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi576P-05PPAhUO52MKHdgnA2wQ_AUICSgC&biw=1366&bih=643#imgrc=3TMPX5OrYwcjwM`.

[5] Aurélien Bouteiller et al. "Coordinated checkpoint versus message log for fault tolerant MPI". In: *Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on*. IEEE. 2003, pp. 242–250.

[6] Eddy Caron, Frederic Desprez, and Adrian Muresan. "Forecasting for grid and cloud computing on-demand resources based on pattern matching". In: *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE. 2010, pp. 456–463.

[7] *Cloud Computing Ohio Electronics Record Committee*. URL: `http://ohioerc.org/?page_id=187`.

[8] *cloud-simulation-frameworks - home*. URL: `http://cloud-simulation-frameworks.wikispaces.asu.edu/`.

[9] Sheng Di et al. "Optimization of cloud task processing with checkpoint-restart mechanism". In: *2013 SC-International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE. 2013, pp. 1–12.

[10] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. "Press: Predictive elastic resource scaling for cloud systems". In: *Network and Service Management (CNSM), 2010 International Conference on*. Ieee. 2010, pp. 9–16.

[11] K. Hwang, J. Dongarra, and G.C. Fox. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Elsevier Science, 2013. ISBN: 9780128002049. URL: https://books.google.co.in/books?id=IjgVAgAAQBAJ.

[12] Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters". In: *Proceedings of the 6th international conference on Autonomic computing*. ACM. 2009, pp. 117–126.

[13] Arijit Khan et al. "Workload characterization and prediction in the cloud: A multiple time series approach". In: *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE. 2012, pp. 1287–1294.

[14] *K-means clustering*. URL: https://en.wikipedia.org/wiki/K-means_clustering.

[15] Bhavani B Nagesh et al. "Resource Provisioning Techniques in Cloud Computing Environment-A Survey". In: *IJRCCT* 3.3 (2014), pp. 395–401.

[16] Shivangi Nigam and Abhishek Bajpai. "An Optimal Resource Provisioning Algorithm for Cloud Computing Environment". In: ().

[17] Gang Ren et al. "Google-wide profiling: A continuous profiling infrastructure for data centers". In: (2010).

[18] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. "Efficient autoscaling in the cloud using predictive models for workload forecasting". In: *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE. 2011, pp. 500–507.

[19] Vasinee Siripoonya and Kasidit Chanchio. "Thread-based live checkpointing of virtual machines". In: *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*. IEEE. 2011, pp. 155–162.

[20] D. Tsafrir. *Parallel Workloads Archive: Logs*. URL: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

[21] Christian Vecchiola et al. "Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka". In: *Future Generation Computer Systems* 28.1 (2012), pp. 58–65.

[22] Chao Wang et al. "Hybrid checkpointing for MPI jobs in HPC environments". In: *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*. IEEE. 2010, pp. 524–533.

[23] *Weka (machine learning)*. URL: https://en.wikipedia.org/wiki/Weka_(machine_learning).

[24] Kuai Xu, Feng Wang, and Lin Gu. "Profiling-as-a-service in multi-tenant cloud computing environments". In: *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE. 2012, pp. 461–465.

[25] Sangho Yi, Derrick Kondo, and Artur Andrzejak. "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud". In: *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE. 2010, pp. 236–243.

[26] Rostyslav Zabolotnyi, Philipp Leitner, and Schahram Dustdar. "Profiling-Based Task Scheduling for Factory-Worker Applications in Infrastructure-as-a-Service Clouds". In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE. 2014, pp. 119–126.