

MEMORY MANAGEMENT TO ANALYSE THE STORAGE OF THE CLOUD

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KOTA MURALI KRISHNA (37110370)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

**(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL- 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of KOTA MURALI KRISHNA (37110370) who carried out the project entitled "**MEMORY MANAGEMENT TO ANALYSE THE STORAGE OF THE CLOUD**" under my supervision from November 2020 to April 2021.

Internal Guide

Mr. R. SETHURAMAN (Ph.D.,)

Head of the Department

Dr.S. VIGNESHWARI M.E., Ph.D.

Submitted for Viva-Voce Examination held on:

Internal Examiner

External Examiner

DECLARATION

I KOTA MURALI KRISHNA (37110370), hereby declare that the Project Report entitled

“MEMORY MANAGEMENT TO ANALYSE THE STORAGE OF THE CLOUD” done

by me under the guidance of **Mr. R. SETHURAMAN (Ph.D.,)** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALAM.E.,Ph.D.**, Dean,SchoolofComputing, **Dr.S.VIGNESHWARIM.E.,Ph.D.**, and **Dr.L.LAKSHMANANM.E., Ph.D.**, Heads Of the Department, Department of Computer Science and Engineering for providing usthenecessarysupportanddetailsattherighttimeduringtheprogressivereviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. R. SETHURAMAN (Ph.D.,)**

for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express our thanks to all Teaching and Non-teaching staff of the **Department of COMPUTER SCIENCE & ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

One of the many definitions of “cloud” is that of a Software (SaaS) system, in which IT infrastructure is deployed in a provider’s data center as virtual machines. With IaaS clouds’ growing popularity, tools and technologies are emerging that can transform an organization’s existing infrastructure into a private or hybrid cloud. OpenNebula is an open source, virtual infrastructure manager that deploys virtualized services on both a local pool of resources and external IaaS clouds. Haizea, a resource lease manager, can act as a scheduling back end for OpenNebula, providing features not found in other cloud software or virtualization-based data center management software.

Existing system has suffered to allocate a memory, based on the users requirement, since the system is not releasing the memory from the existing user which has not been utilized. In the proposed system we are planning to manage the memory that we have and maintain the server within the available storage. To achieve the optimized allocation of memory based on the requisition of new users from the previous users those who are not utilized the allocated memory . This paper summarizes a detailed study about memory utilization and propose an approach which use Random Forest Algorithm for optimized decision. This paper focuses on better memory management for the users

CHAPTER NO.	CHAPTER NAME	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	lx
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	
	1.1 OVERVIEW	1
	1.2 OBJECTIVE	4
	1.3 MOTIVATION	4
	1.4 ORGANIZATION OF THE REPORT	5
2	LITERATURE SURVEY	
	2.1 PROBLEM STATEMENT	11
	2.2 EXISTING SYSTEM	11
	2.3 PROPOSED SYSTEM	11
3	METHODOLOGY	
	3.1 SOFTWARE REQUIREMENTS	15
	3.2 USE CASES	15
	3.3 ARCHITECTURE DIAGRAM	15
	3.4 MODULES	17
	3.4.1 USER LOGIN	17
	3.4.2 REQUESTING FOR MEMORY	17
	3.4.3 ALLOCATING MEMORY	18
	3.5 ALGORITHM	19

4	RESULTS AND DISCUSSION	
	4.1 USER REGISTRATION	23
	4.2 LOGIN PAGE	24
	4.3 MEMORY HOME PAGE	25
	4.4 MEMORY STATUS	26
	4.5 ADMIN ALLOCATION	27
5	CONCLUSION AND FUTURE WORK	
	5.1 CONCLUSION	28
	5.2 FUTURE WORK	28
	REFERENCES	29
	APPENDIX	
	A. SAMPLE CODE	31
	B. SCREENSHOTS	38
	C. PAPER PUBLICATION WITH PLAGARISM REPORT	42

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.2	Architecture Diagram	16
3.4.1	User Login	17
3.4.2	Requesting Memory	17
3.4.3	Allocating Memory	18
3.5	Random Forest Algorithm	21
4.1	User Registration Page	23
4.2	Login Page	24
4.3	Memory Home Page	25
4.4	Memory Status	26
4.5	Admin Allocation Page	27

CHAPTER1

INTRODUCTION

OVERVIEW

Efficient Memory management is one of the hot topics these days in Cloud because the need of integrated data handling and exigency of optimized memory management algorithms. The trending Software as A Service (SaaS), Platform as A Service (PaaS) and Infrastructure as A Service (IaaS) paradigms are in need of smart memory management protocols to be integrated in cloud in order to get rid of the latency and load balancing issues. compute and storage deployments are that all types of computer memory are constrained by an upper limit. No resource on a modern system is perhaps as constrained as memory, which is always needed by operating systems, applications and storage. Without unlimited memory, at some point memory is fully consumed, which leads to system instability or data loss.

Since the beginning of modern IT, the challenge of memory exhaustion has been handled by a diverse set of capabilities, usually grouped under the heading of memory management. Memory management is all about making sure there is as much available memory space as possible for new programs, data and processes to execute. As memory is used by multiple parts of a modern system, memory allocation and memory management can take on different forms. Operating System - Operating systems like Microsoft Windows and Linux, can make use of physical RAM as well as hard drive swap space to manage a total pool of available memory.

Programming Languages - The C programming language requires developers to directly manage memory utilization, while other languages like Java and C#, for example, provide automatic memory management.

Applications - Applications consume and manage memory, but are often limited in memory management capabilities as defined by the underlying language and operating system.

Storage memory management - With new NVME storage drives, operating systems can benefit from faster storage drives to help expand and enable more persistent forms of memory management.

HOW MEMORY MANAGEMENT WORKS

Memory management is all about allocation and optimization of finite physical resources. Memory is not uniform - for example a 2GB RAM DIMM is not used as one large chunk of space. Rather memory allocation techniques are used for segmentation of RAM into usable blocks of memory cache. Memory management strategies within an operating system or application typically involve an understanding of what physical address space is available in RAM and performing memory allocation to properly place, move and remove processes from memory address space.

TYPES OF MEMORY ADDRESS

Static and dynamic memory allocation in an operating system is linked to different types of memory addresses. Fundamentally, there are two core types of memory addresses: Physical addresses - The physical address is the memory location within system RAM and is identified as a set of digits. Logical addresses - Also sometimes referred to as virtual memory, a logical address is what operating systems and applications access to execute code, as an abstraction of physical address space.

HOW DOES MMU CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS?

The Memory Management Unit (MMU) within a computing system is the core hardware component that translates virtual logical address space to physical addresses. The MMU is typically a physical piece of hardware and is sometimes referred to as a Paged Memory Management Unit (PMMU) The process by which the MMU converts a virtual address to a physical address is referred to as virtual address translation and makes use of a Page Directory Pointer Table (PDPT) to convert one address type to another. The process is directly tied to page table allocation, matching and managing one address type to another. To help accelerate virtual address translation there is a caching mechanism known as the Translation Look aside Buffer (TLB) which is also part of the virtual address to physical address translation process

MEMORY ALLOCATION: STATIC VS DYNAMIC LOADING

Applications and data can be loaded into memory in a number of different ways, with the two core approaches being static and dynamic loading.

Static Loading - Code is loaded into memory, before it is executed. Used in structured programming languages including C.

Dynamic Loading - Code is loaded into memory as needed. Used in object-oriented programming languages, such as Java.

MEMORY FRAGMENTATION

When memory is allocated in a system, not all of the available is always consumed in a linear manner, which can lead to fragmentation. There are two core types of memory fragmentation, internal and external

Internal Fragmentation - Memory is allocated to a process or application and isn't used, leaving un-allocated or fragmented memory.

External Fragmentation - As memory is allocated and then de allocated, there can be small spaces of memory leftover, leaving memory holes or "fragments" that aren't suitable for other processes.

PAGING

Within logical address space, virtual memory is divided up using paging, meaning it's divided into fixed units of memory, referred to as pages. Pages can have different sizes, depending on the underlying system architecture and operating system. The process of page table management can be intricate and complex. For more information on how paging is handled in Linux, check out the full kernel.org documentation. On Windows, Microsoft provides details on its paging process [here](#).

SEGMENTATION

Memory segmentation within the primary memory of a system is a complicated process that references specific bits within a memory unit. Each segment within system memory gets its own address, in an effort to improve optimization and memory allocation. Segment registers are the primary mechanism by which modern systems handle memory segmentation.

SWAPPING

Swapping is the process by which additional memory is claimed by an operating system from a storage device. How swapping works is an operating system defines an area of storage that is used as "swap space," that is storage space where memory process will be stored and run as physical and virtual memory space is exhausted, released and reclaimed. The usage of swap space with traditional storage is a sub-optimal way of expanding available memory as it incurs the overhead of transferring to and from physical RAM. Additionally, traditional storage devices run with slower interface speeds than RAM. Swapping however is now being revisited as a way to expand memory with the emergence of faster, PCIe SSDs which offer an interface connect speed of up to 16 Gb/s. In contrast a SATA connected SSD has a maximum connection speed of 6.0 Gb/s.

OBJECTIVE

Memory management is an essential element of all modern computing systems. With the continued use of virtualization and the need to optimize resource utilization, memory is constantly being allocated, removed, segmented, used and re-used. With memory management techniques, memory management errors that can lead to system and application instability and failures can be mitigated. SaaS, IaaS etc are in need of smart memory management protocols to be integrated in Cloud in order to get rid of the latency and load balancing issues. On demand resource allocation is the key in optimizing the data efficiency of the Cloud. There is a huge drainage of resources across the Cloud platform if the resources are allocated and left idle. Continuous checks and monitoring are necessary to get hold of the idle resources.

MOTIVATION

Memory allocation is one of the most ubiquitous operations in computer systems and is a long-studied research field. The efficient use of physical memory has a major impact on the performance of computer programs. A memory management

algorithm allows applications to dynamically capture and release portions of memory of varying sizes many times during their runtime. Further, a memory management algorithm may add several other features to exploit memory usage patterns, memory cache, blowup, false sharing, and memory contention, among others.

ORGANIZATION OF THEREPORT

This report is organized as follows: Chapter 2 consists of related work to allocate the memory efficiently on cloud. Chapter 3 consists of system analysis which defines about the problem definition, various software components and use cases. The chapter 4 describes about the system design which consists of the system architecture for the proposed system and module description for each module. The chapter 5 tells about the system implementation and the various algorithms. The chapter 6 describes about the system testing which consists of different test cases for the various modules. The chapter 7 tells about the output obtained and the explanation about the same. The chapter 8 consists of conclusion and the future work for the project. The appendix 1 consists of the sample code and software implementations. The appendix 2 consists of the screenshots of the various output screens. The appendix 3 consists of the technical paper presentation and the references.

CHAPTER 2

LITERATURE SURVEY

Anwar Al-Yatama et al [1], proposed a paper in which, Memory allocation has a major influence on multiuser systems, cloud-based services, virtual machines, and other computer systems. Memory allocation is a process that assigns physical or virtual memory space to programs and services as efficiently and quickly as possible. Economical memory allocation management needs allocation strategies with minimum wastage. In this paper, we introduce a new memory allocation algorithm based on sequential fits and zoning for on-demand (online) cloud services. The memory is divided into multiple zones, where subgroups of relative request sizes compete in reverse order. We use simulation to compare our new mechanism with existing memory allocation methods that have been deployed using Amazon Elastic Compute Cloud as a test bed. The proposed algorithm is more efficient, and the average saving for the normalized revenue loss is about 7% better than best-fit and 15% better than first-fit memory allocation. In addition, we show that proposed algorithm is robust and faster and has a fairness index that is superior to that of existing techniques.

Dong Oh Son et al [2], proposed a paper in which, Cloud computing has become a mainstream in providing computing resources as services. Recent cloud computing systems are composed of homogeneous hardware resources. Heterogeneous cloud computing can provide better performance and energy efficiency than homogeneous cloud computing. Therefore, heterogeneous cloud computing is expected to be used widely. In general, memory management policy has huge impact on the performance of heterogeneous computing systems. In this paper, we analyze three memory management policies for heterogeneous cloud computing systems. According to our experimental results, dynamic partition memory management policy provides better performance than static partition memory management policy by 4.65% on the average.

Filip Pizlo et al [3], proposed a paper in which, the real-time specification for Java extends the Java platform to support real-time processing and introduces a region- based memory model, called scoped memory, which side-steps the Java garbage collector. While scoped memory succeeds in protecting real-time tasks from execution time jitter, practical experience points to shortcomings. This paper takes stock of the state of the art in memory management for RTSJ programs.

Gauhar Eram Shaikh et al [4], proposed a paper in which, Virtualization technology is drawing a renewed attention from variety of application domain such as data centers, desktop computing and cloud computing etc. Virtualization improves the utilization of hardware resources such as network I/O, processor and memory. Allocating an appropriate amount of main memory resource to virtual machines is challenging task. Static allocation is one approach for managing memory but it leads to inefficient memory utilization. Different application has different memory requirements, even a single application shows varied working set during its execution therefore an optimal memory management technique under a virtualized environment is needed to manage memory dynamically. This paper proposes an efficient memory management technique which includes global allocation technique along with memory releasing and memory growing technique for satisfying the memory demand on run time. The result of this methods leads to better memory utilization, improves the performance of the overall system and maximizes the memory usage.

Pranjali P. Deshmukh et al [5], proposed a paper in which, Cloud computing in these days is the top most area of research because of the major concerns as reducing costs, and also scalability and flexibility of in computing services. The performance of cloud environment is depending upon virtual machines and its memory management. This paper surveys the various memory streaming techniques managed by hypervisor to improve the performance, scalability of virtual machines and optimum utilization of resources and ultimately reducing cost. Researchers are doing research to collect the hypervisor memory management techniques like live migration, cloning, page sharing, read cache and many more to do the memory streaming for virtual machines in cloud

environment. If all the memory management techniques clubbed together then in both public and private cloud environment one can scale virtual infrastructure rapidly with reducing memory requirement and overhead.

Sasmita Parida et al [6], proposed a paper in which, The Cloud Computing is very popular topic in now a day. We are still in early years of cloud computing development which provides lots of scope for researcher. In most of the cloud services the clients follow “Pay and Use” concepts. Cloud services providers charge for the resource use. So, it is important to distribute resources in an efficient way for on demand application of the clients. One of the ways in cloud computing is multi-tenant architecture which provides self-managing of the resources. DDAP2 (Dynamic Distributed Allocation Policy 2) is one of the simplest resource allocation algorithms that is used for CPU allocation. But DDAP2 is not providing a better allocation for memory. In this paper we studied the memory allocation using DDAP2. We modified some of the policies of DDAP2 for memory allocation, which gives better allocation of memory in cloud computing environment. The memory allocation is improved using DDAP2 by adapting a new policy based on CPU allocation using DDAP2. This algorithm dynamically allocates the overall capacity between VMs based on their demand. The modified approach is scalable for better allocation of CPU and memory resources according to the users demand.

Ye Ding et al [7], proposed a paper in which, As the development in the cloud environment, virtual technology can consolidate multiple servers and abstract the hardware resources to each virtual machine. Memory is a well-known physical hardware resource, which has various limitations because of its high cost and strong power. The size of memory in which virtual machines in the Cloud environment always changes dynamically. Therefore, a flexible virtual memory management is supposed to adopt in Cloud environment so that the system allocates memory reasonably. In this paper, we observe the memory management system in Cloud environment and review the present situation of memory management.

2.1 PROBLEMSTATEMENT

The problem definition of the project is to develop a smart memory allocation and management system to analysis the usage of the memory and to allocate the memory to the cloud users efficiently.

2.2 EXISTING SYSTEM

- Via an Internet service connection, cloud storage works by enabling users' access and to download data on any chosen device, such as a laptop, tablet or Smartphone.
- Cloud storage users **can** also edit documents simultaneously with other users as well, making it easier to work away from theoffice.

2.3 PROPOSEDSYSTEM

- In the proposed system we are planning to manage the memory that we have and maintain the server within the storageavailable.
- As the pervious paper we have planned to allocate thememory.
- The difference from the previous paper is memory that is not used bythe previous users are requested and get it back to theserver.
- Now the server will get some extra memory from the user and that can be allocated to the new user aswell.

CHAPTER 3

METHODOLOGY

The main objective of the project is to develop a smart memory allocation and management system to analysis the usage of the memory and to allocate the memory to the cloud users efficiently.

Java is Object Oriented, Platform Independent, Simple, Secure, Architecture-neutral, Portable, Robust, Multithreaded, Interpreted, High Performance, Distributed, Dynamic. Java is platform independent programming language. Java creates a byte code that can be run on any type of machine with the help of Java Virtual Machine (JVM). Java is secure language because java is not using pointers. Java is also called "write once and run anywhere" programming language. The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications. The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be Write Once, Run Anywhere. The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platform released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private JVM and a few other resources to finish the development of a Java Application. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit (SDK). [citation needed] On 17 November 2006, Sun announced that they would release it under the GNU General Public License (GPL), thus making it free software. Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look

and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element. NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5, and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers. The NetBeans Platform is a framework for simplifying the development of Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again. The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (carries out efficient storage)

- Window management
- Wizard framework (supports step-by-step dialogs)
- Net Beans Visual Library.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, or a proprietary license. Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third-party organizations exist to provide support and services, including MariaDB and Percona. MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

3.1 SOFTWARE REQUIREMENTS

- Operatingsystem: Windows
- Coding Language: Java,Swing.
- FrontEndTool : Net beans.
- Database : MSSql.
- BackEndTool : SQLYog.

3.2USECASES

Scope – Memory management system

Primary Actor – Admin, User

Admin- Admin analysis the memory usage for the users and allocate the memory efficiently.

User- Uses the cloud memory.

Preconditions – user should have an access to the cloud memory.

Success Guarantee – Efficient memory management system is built.

3.3ARCHITECTUREDIAGRAM

The below architecture shows the complete architecture flow of the project. The various users will have the access to the cloud memory. Some of the users will not have a sufficient memory they are in need of extra memory. And some users will have extra memory without using it. So, the Admin analysis the memory usage for every user and borrow the memory from the user who has extra memory and lent the memory who are in need. The users who request for the free memory available can have an efficient use of the cloudmemory.

3.4MODULES

- USERLOGIN
- REQUESTING FOR REQUIREDMEMORY
- ALLOCATING THEMEMORY

USERLOGIN

There are multiple users who are using the cloud memory. To access the cloud the user has to first login to the system, if the user is a new user the user has to first register by entering the required details. And the user has to register the company by entering the details of the company. Based on the entered details the system predicts the required memory for the particular user. The user can also customize the required memory needed for the company.



Fig 3.4.1 User Login

REQUESTING FOR REQUIRED MEMORY

If the registered user need for any extra memory sends a request to the Admin. The Admin accepts the request from the user and the machine learning analysis the history of the users who uses the cloud storage for their company.



Fig 3.4.2 Requesting Memory

ALLOCATING THE MEMORY

After accepting the request from user, system analysis the history of the user who uses the cloud memory for their company. It borrows the unused memory from the other user. Then the Admin allocates the required memory to the requested users. By this way the user can access the cloud memory efficiently.

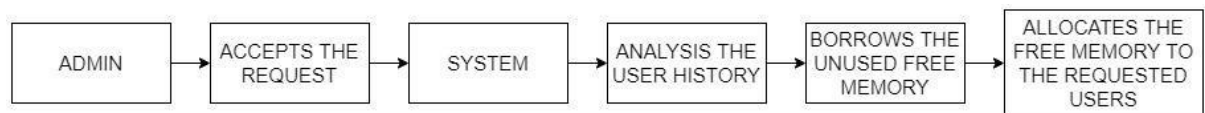


Fig 3.4.3 Allocating Memory

3.5 ALGORITHM

Random Forestalgorithm

Step 1: Create a Bootstrapped Data Set

Bootstrapping is an estimation method used to make predictions on a data set by re-sampling it. To create a bootstrapped data set, we must randomly select samples from the original data set. A point to note here is that we can select the same sample more than once.

Step 2: Creating Decision Trees

Our next task is to build a Decision Tree by using the bootstrapped data set created in the previous step. Since we're making a Random Forest, we will not consider the entire data set that we created, instead we'll only use a random subset of variables at each step.

Step 3: Go back to Step 1 and Repeat

Like I mentioned earlier, Random Forest is a collection of Decision Trees. Each Decision Tree predicts the output class based on the respective predictor variables used in that tree. Finally, the outcome of all the Decision Trees in a Random Forest is recorded and the class with the majority votes is computed as the output class. Having such a variety of Decision Trees in a Random Forest is what makes it more effective than an individual Decision Tree created using all the features and the whole data set.

Step 4: Predicting the outcome of a new data point

Now that we've created a random forest, let's see how it can be used to predict whether a new patient has heart disease or not.

To conclude, we bootstrapped the data and used the aggregate from all the trees to make a decision, this process is known as Bagging.

Step 5: Evaluate the Model

Our final step is to evaluate the Random Forest model. Earlier while we created the bootstrapped data set, we left out one entry/sample since we duplicated another sample. In a real-world problem, about 1/3rd of the original data set is not included in the bootstrapped data set.

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage. The whole process is shown below, and it's easy to understand using the figure.

Here the author firstly shows the Random Forest creation pseudocode:

1. Randomly select “K” features from total “m” features where $k \ll m$
2. Among the “K” features, calculate the node “d” using the best splitpoint
3. Split the node into daughter nodes using the bestsplit
4. Repeat the a to c steps until “l” number of nodes has beenreached
5. Build forest by repeating steps a to d for “n” number times to create “n”
number of trees

Figure shows the process of randomly selecting features:

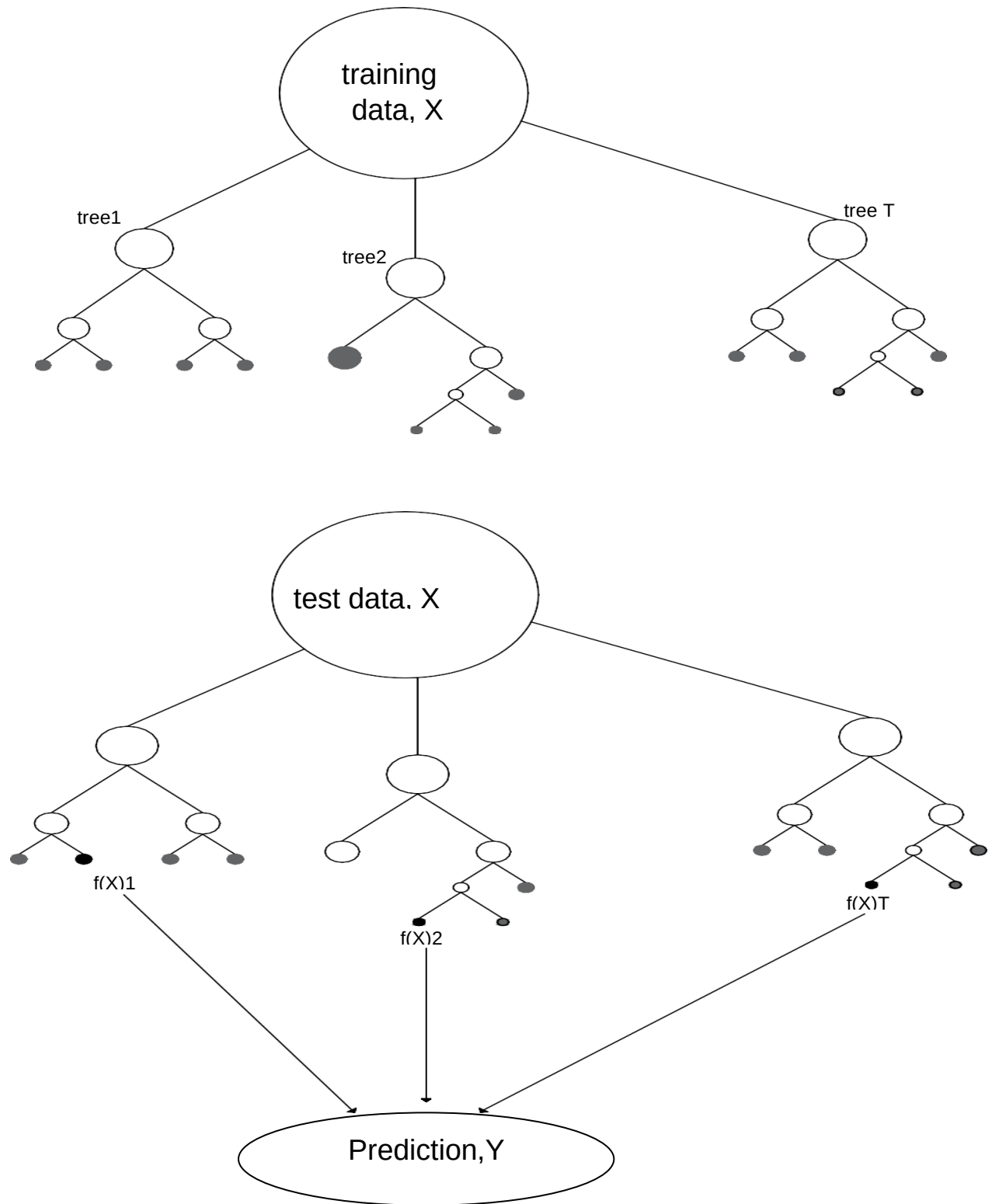


Fig 3.5 Random Forest Algorithm Diagram

In the next stage, with the random forest classifier created, we will make the prediction. The random forest prediction pseudocode is shown below:

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome(target)
2. Calculate the votes for each predictedtarget
3. Consider the high voted predicted target as the final prediction from the random forestalgorithm

Advantages of Random Forest

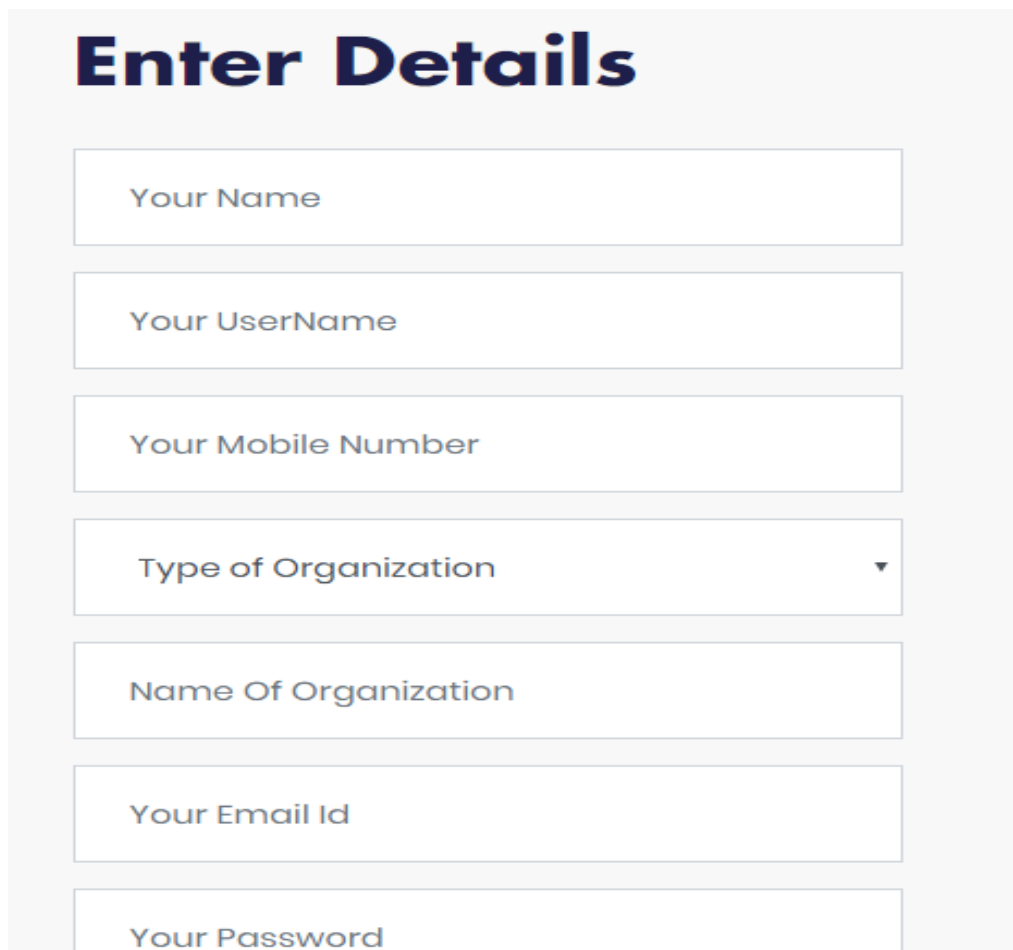
Compared with other classification techniques, there are three advantages as the author mentioned.

1. For applications in classification problems, Random Forest algorithm will avoid the overfittingproblem
2. For both classification and regression task, the same random forest algorithm can beused
3. The Random Forest algorithm can be used for identifying the most important features from the training dataset, in other words, featureengineering.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 USERREGISTRATION

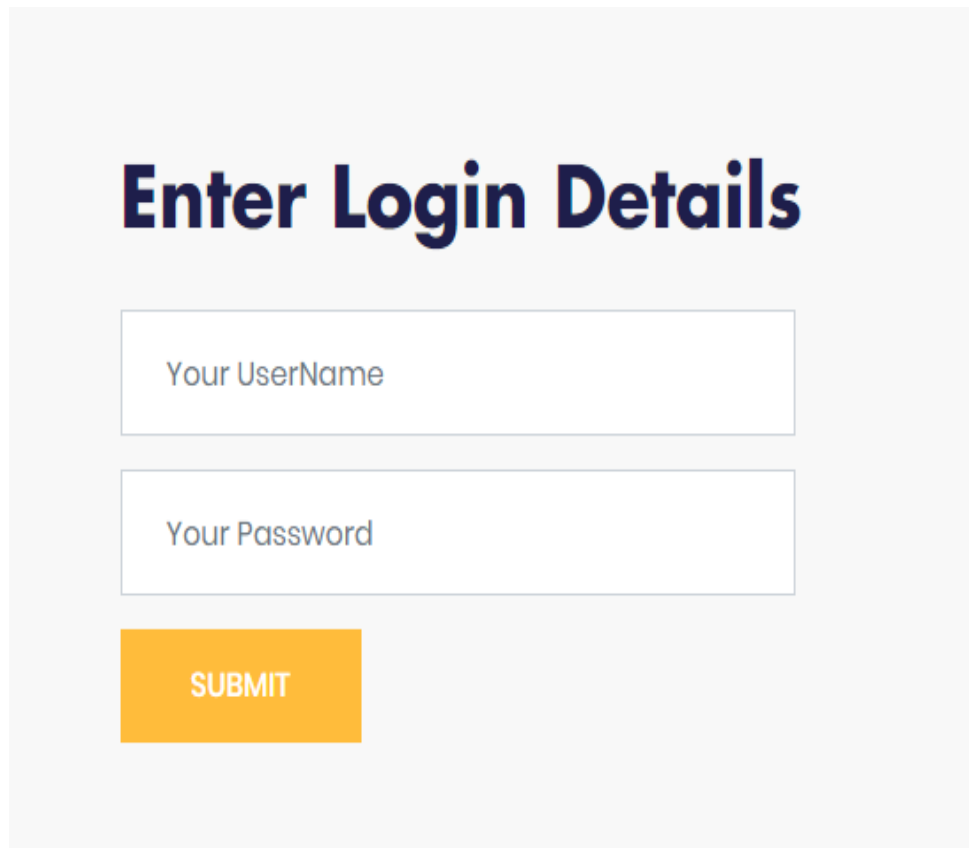


The image shows a user registration form titled "Enter Details". It contains seven input fields arranged vertically: "Your Name", "Your UserName", "Your Mobile Number", "Type of Organization" (a dropdown menu), "Name Of Organization", "Your Email Id", and "Your Password". Each field is a white rectangle with a thin grey border and a light grey placeholder text. The form is set against a light grey background.

Fig 4.1 User Registration Page

In fig 4.1, It shows the page for registration, When the new user comes for requesting the memory allocation from admin, the user should register and provide the details exactly and the user is assigned to create a user name and password for the login page. When the user assigned the type of organization based on that the memory is going to be allocated.

4.2 LOGINPAGE



The image shows a login page with a light gray background. At the top, the text "Enter Login Details" is displayed in a large, bold, dark blue font. Below this, there are two white rectangular input fields with thin gray borders. The first field contains the placeholder text "Your UserName" in a light blue font. The second field contains the placeholder text "Your Password" in a light blue font. Below these fields is a solid orange rectangular button with the word "SUBMIT" written in white, uppercase letters.

Fig 4.2 Login Page

In the fig 4.2, User login page, When the new user comes and automatically the user goes for the registration process, In the registration page the user will be provided for the user name and login details. After registering the user directly goes to the login page and submitting the details of the userdetails.

4.3 MEMORY HOMEPAGE

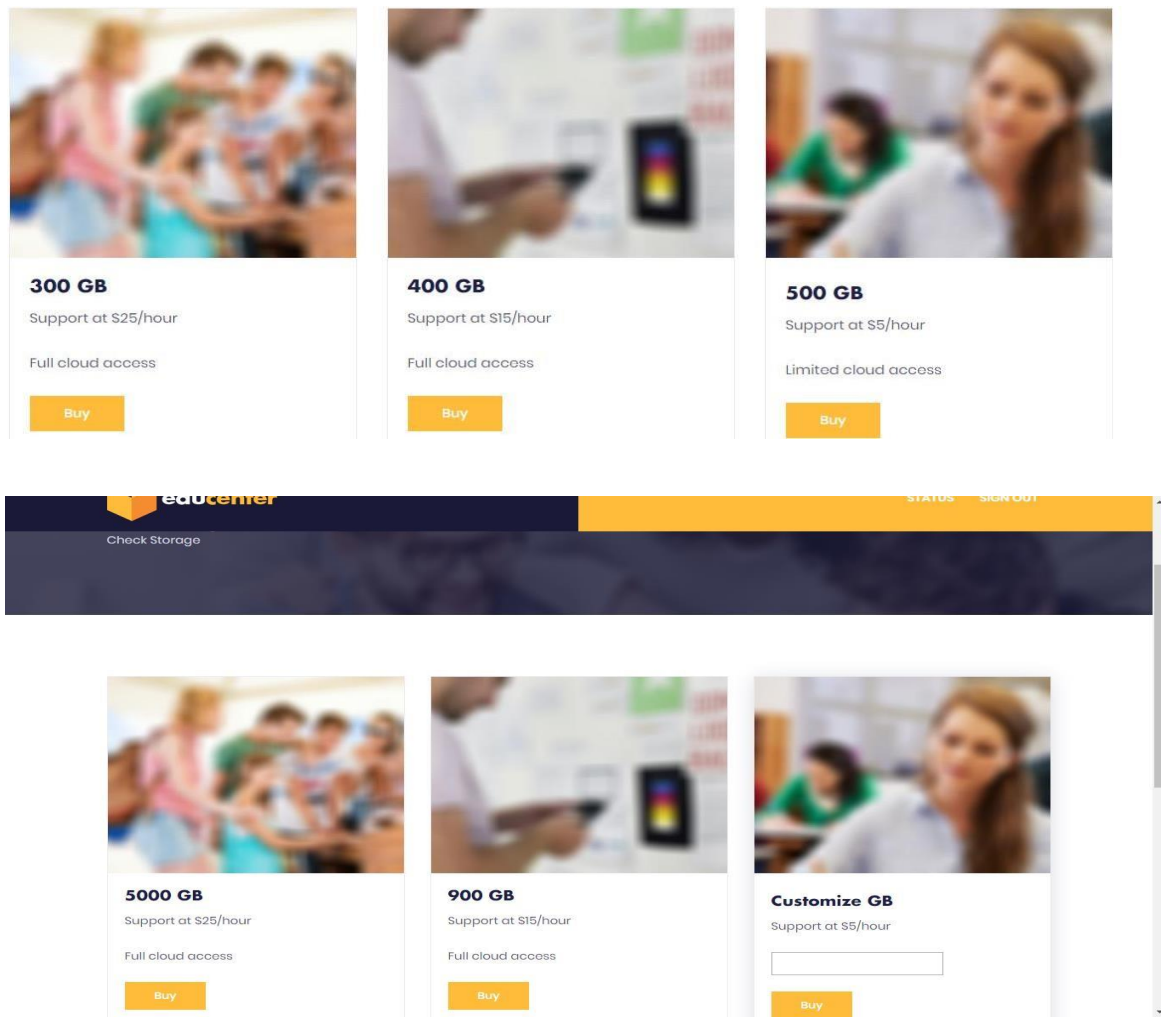


Fig 4.3 Memory Home Page

In Fig 4.3, Storage home page, after user login in the web page it shows the default page starting to check the storages are available, Users checks for memory storage needs for the specific organization and the user can select based on the customize memory storage for the organization.

4.4 MEMORYSTATUS

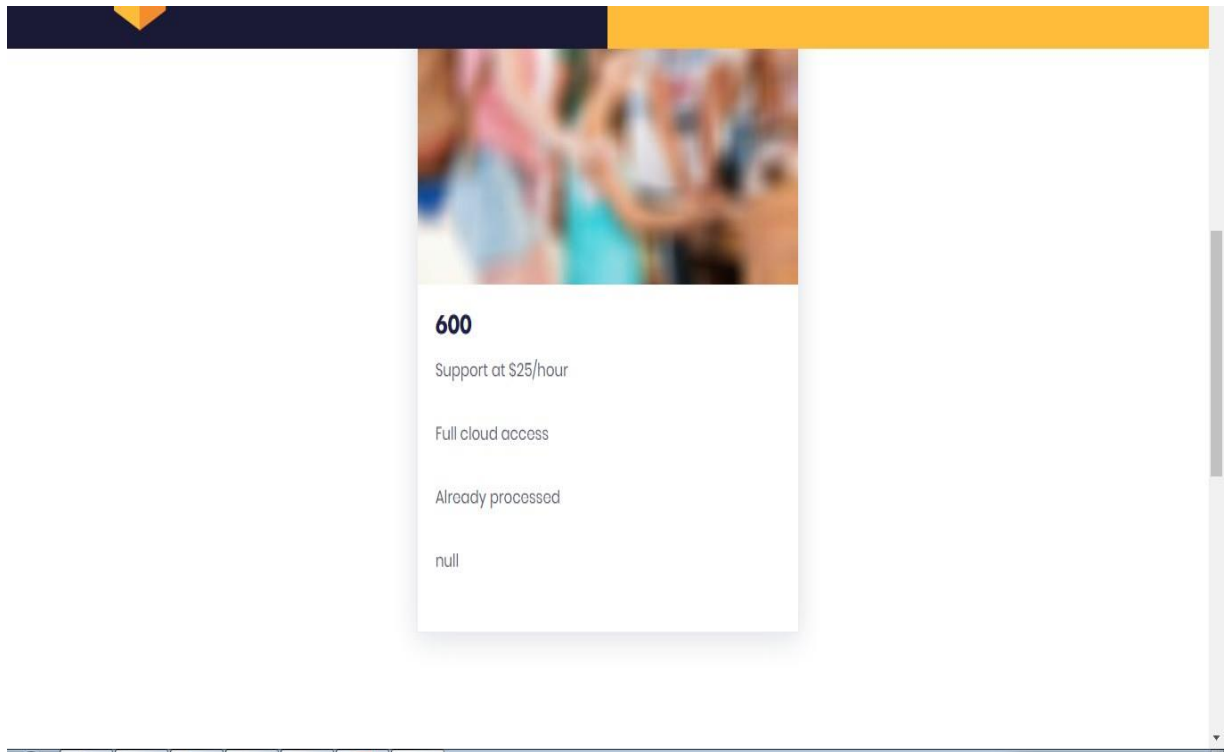


Fig 4.4 Memory Status

In Fig 4.4, It shows the status of the memory storage, User by the storage and the requested goes to admin from the user. How much the user is going to process the admin is visible to see that and the admin is having to accept or decline the process for memory, If the admin is going to allocate the memory the user can able to see the status of the memory request status.

4.5 ADMINALLOCATION

Admin
Check Storage

UserName	Company Type	Amount		
rajsehb	small	200GB	Accept	Decline

Admin
Check Storage

UserName	Company Type	Amount	free Memory
sasi	large	500GB	0GB
praveen	medium	300GB	100GB
kaala	large	500GB	500GB
vinaya	large	500GB	500GB
asv	large	500GB	500GB
uttejb	large	600GB	600GB

Fig 4.5 Admin Allocation Page

In Fig 4.5, The Admin is able to accept and decline for the user has been requested for the memory allocation, In the admin page the admin is going to allocate the memory and can check in the allocation details, How much have been issued the memory for the users and can see the old user data also in the admin page.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The use of efficient and dynamic techniques decreases the use of load balancers and increases the throughput in terms of memory management. To avoid bottleneck scenarios proper management policies must be adopted in terms of a cloud environment. The use of Hyper managers in case of services provided on cloud is necessary. Our approach includes a technique to dynamically change JVM memory limits, an approach to model memory usage and garbage collection cost during query execution, and a memory manager that performs actions on JVMs to reduce total failures and run times. We evaluated our approach in Myria and showed that our approach out-performed static memory allocation both on query failures and execution times.

5.2 FUTURE WORK

In the future, the process will continue on three directions but not limited: to improve and perfect the service-oriented FBDE for CBDM and CPD to add more P2P FBDE services, including pre-P2P and post-P2P services for more heterogeneous CAD systems and to explore information security issues for service-oriented FBDE in CBDM and CPD to apply the OSLC standard and ontologies into CAD data exchange with the research of the OSLC and Factories of the Future(FoF) program to extended the CAD data exchange into the area of manufacturing by researching on the machining features.

REFERENCES

- [1] P. Mell, T. Grance, The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (Final), Tech. rep.(2011).
- [2] M. Armbrust, A. Fox, R. Gri_th, A. Joseph, Above the clouds: A berkeley view of cloud computing, Tech. rep., UC Berkeley Reliable Adaptive Distributed Systems Laboratory(2009).
- [3] E. Elmroth, J. Tordsson, F. Hern´andez, Self-management challenges for multi-cloud architectures, Towards a Service-Based Internet. Lecture Notes in Computer Science 6994(2011)
- [4] M. Pavlovic, Y. Etsion, A. Ramirez, On the memory system requirements of future scientific applications: Four case-studies, in: 2011 IEEE International Symposium on Workload Characterization (IISWC), IEEE, 2011, pp.159–170.
- [5] G. Toye, M. R. Cutkosky, L. J. Leifer, J. M. Tenenbaum, and J. Glicksman, "SHARE: a methodology and environment for collaborative product development,"Int'l J. intelligent and cooperative information systems, vol. 3, pp. 129-153, 1994.
- [6] A. K. Goel, S. Vattam, B. Wiltgen, and M. Helms, "Cognitive, collaborative, conceptual and creative — Four characteristics of the next generation of knowledge-based CAD systems: A study in biologically inspired design,"Computer-Aided Design, vol. 44, pp. 879-900,2012.
- [7] P. Tuli and R. Shankar, "Collaborative and lean new product development approach: a case study in the automotive product design," Int'l J. Production Research, vol. 53, pp. 2457-2471,2015.
- [8] D. Wu, J. L. Thames, D. W. Rosen, and D. Schaefer, "Towards a cloud-based design and manufacturing paradigm: looking backward, looking forward," Proc. ASME.Int'l Conf. Design Engineering Technical and Computers and Information in Engineering, pp.315-328,2012.
- [9] J. Bughin, M. Chui, and J. Manyika, "Clouds, big data, and smart assets: Ten tech-enabled business trends to watch," McKinsey Quarterly, vol. 56, pp. 75-86,2010.

- [10] A. Rappoport, S. Spitz, and M. Etzion, "Two-dimensional selections for feature-based data exchange," *Proc. Geometric Modeling and Processing (GMP'06)*, pp. 325-342, 2006.
- [11] Youl, Philippa H., et al. "Can skin cancer prevention and early detection be improved via mobile phone text messaging? A randomized, attention control trial." *Preventive medicine* 71 (2015):50-56.

APPENDIX

A. SOURCECODE

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.swing.JOptionPane;

public class Allocate extends HttpServlet {

    throws IOException if an I/O error occurs
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException{
        response.setContentType("text/html;charset=UTF-8");
        Stringu=request.getParameter("uname");
        String ureq=request.getParameter("amount");
        String ctype=request.getParameter("ctype");
        System.out.println("name==" +u);
        System.out.println("ureq==" +ureq);
        int total=0;
        int iureq=Integer.parseInt(ureq);
        int old=0;
```

```

try (PrintWriter out = response.getWriter()) {
    /* TODO output your page here. You may use following sample code. */
    Class.forName("com.mysql.jdbc.Driver");
    Connection
con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/server","r
oot","root");
    PreparedStatement ps4=con.prepareStatement("select * from admin");
    ResultSet rs1;
    rs1=ps4.executeQuery();
    while(rs1.next()){
        total=rs1.getInt("totalMemory");
    }
    if(total==0 || total <iureq){
        JOptionPane.showMessageDialog(null, "Storage Not available");
        RequestDispatcher rd=request.getRequestDispatcher("admin.jsp");
        rd.include(request, response);
    }else{
        PreparedStatement ps=con.prepareStatement("select * from oldusers");
        ResultSet rs;
        rs=ps.executeQuery();
        while(rs.next() && iureq > 0){
            String tu= rs.getString("user");
            String temp=rs.getString("freeMemory");
            String days=rs.getString("totalDaysUsed");
            int itemp=Integer.parseInt(temp);
            int idays=Integer.parseInt(days);
            if(idays>365){
                iureq-=itemp;
                old=old+itemp;
                PreparedStatement ps7=con.prepareStatement("update oldusers set
freeMemory=? whereuser=?");
                ps7.setInt(1, 0);
                ps7.setString(2, tu);
                ps7.executeUpdate();
            }
        }
        if(iureq<=0){
            iureq=0;
        }
        System.out.println("iureq==" +iureq);
        total-=iureq;
        System.out.println("total==" +total);
    }
}

```

```

        if(total<0){
            total=0;
        }
        PreparedStatement ps5=con.prepareStatement("update admin set
totalMemory=?");
        ps5.setInt(1, total);
        ps5.executeUpdate();
        PreparedStatement ps1=con.prepareStatement("insert into allocation
(name,oldUser,admin) values(?,?,?)");
        ps1.setString(1, u);
        ps1.setInt(2, old);
        ps1.setInt(3, iureq);
        inti=0;
        i=ps1.executeUpdate();
        if(i!=0){
            JOptionPane.showMessageDialog(null, "Allocated");
            PreparedStatement ps2=con.prepareStatement("update req set status=1
where name=?");
            ps2.setString(1, u);
            ps2.executeUpdate();
            PreparedStatement ps3=con.prepareStatement("insert into oldusers
(user,companyType,memory,freeMemory,totalDaysUsed)values(?,?,?,?,?)");
            ps3.setString(1, u);
            ps3.setString(2, ctype);
            ps3.setString(3, ureq);
            ps3.setString(4, ureq);
            ps3.setString(5, "0");
            ps3.executeUpdate();
            RequestDispatcher rd=request.getRequestDispatcher("admin.jsp");
            rd.include(request, response);
        }else{
            JOptionPane.showMessageDialog(null, "update failed");
            RequestDispatcher rd=request.getRequestDispatcher("admin.jsp");
            rd.include(request, response);
        }
    }catch(Exception e){
        System.out.println(e);
    }
}
}

```

```

import java.io.*;
import java.sql.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import javax.swing.JOptionPane;

public class register extends HttpServlet {
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        String user = request.getParameter("user");
        String mobile =request.getParameter("mobile");
        String type =request.getParameter("type");
        String company = request.getParameter("company");
        String mail =request.getParameter("mail");
        String pass = request.getParameter("pass");
        String cpass = request.getParameter("cpass");

        if(pass.equals(cpass)){
            try{
                Class.forName("com.mysql.jdbc.Driver");
                Connection con=DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/server","root","root");

                PreparedStatement ps=con.prepareStatement(
                    "INSERT INTO
                    `reg` (name,username,mobile,type,company,mail,pass)VALUES(?,?,?,?,?,?,?)");
                ps.setString(1,name);
                ps.setString(2, user);
                ps.setString(3, mobile);
                ps.setString(4, type);
                ps.setString(5, company);
            }
        }
    }
}

```

```

ps.setString(6, mail);
ps.setString(7, pass);
int i=ps.executeUpdate();
if(i>0) {
    RequestDispatcher rd=request.getRequestDispatcher("login.html");
    rd.forward(request, response);
}
else{
    RequestDispatcher rd=request.getRequestDispatcher("register.html");
    rd.forward(request, response);
}
}catch (Exception e2) {System.out.println(e2);
RequestDispatcher rd=request.getRequestDispatcher("register.html");
    rd.forward(request, response);}

out.close();
}else{
    JOptionPane.showMessageDialog(null,"Incorrect Details");
    RequestDispatcher rd=request.getRequestDispatcher("register.html");
    rd.forward(request, response);
}

}}

```

```

import java.io.File;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.core.Instances;
import weka.core.converters.ArffLoader;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
import java.io.IOException;
import weka.classifiers.trees.RandomForest;

public class RandomForestDemo {

    public static final String TRAINING_DATA_SET_FILENAME="decision-train.arff";
    public static final String TESTING_DATA_SET_FILENAME="decision-test.arff";

```

```

public static Instances getDataSet(String fileName) throws IOException {

    int classIdx = 1;
    ArffLoader loader = new ArffLoader();
    loader.setSource(RandomForestDemo.class.getResourceAsStream("'" +
fileName));

    //loader.setFile(new File(fileName));
    Instances dataSet = loader.getDataSet();
    /** set the index based on the data given in the arff files */
    dataSet.setClassIndex(classIdx);
    return dataSet;
}

public static void process() throws Exception {

    Instances trainingDataSet = getDataSet(TRAINING_DATA_SET_FILENAME);
    Instances testingDataSet = getDataSet(TESTING_DATA_SET_FILENAME);

    RandomForest forest=new RandomForest();
    forest.setNumTrees(10);

    forest.buildClassifier(trainingDataSet);

    Evaluation eval = new Evaluation(trainingDataSet);
    eval.evaluateModel(forest, testingDataSet);

    System.out.println("*** Decision Tress Evaluation with Datasets ***");
    System.out.println(eval.toSummaryString());
    System.out.print(" the expression for the input data as per alogorithm is ");
    System.out.println(forest);
    System.out.println(eval.toMatrixString());
    System.out.println(eval.toClassDetailsString());

}
}

public String[] product(String img)
{
    Stringarr[]={"", ""};

```

```

        if(img.contains("bbb"))
        {
            arr[0]="black";
            arr[1]="boy";
        }
        return arr;
    }
    public String predict(){
        String u=null;
        String u1=null;

try{

            Class.forName("com.mysql.jdbc.Driver");
            Connection
con=(Connection)DriverManager.getConnection("jdbc:mysql://localhost:3306/server","root","root");

            PreparedStatement ps=con.prepareStatement("select olduser from registereduser
where likeCount= (select Max(likeCount) from registereduser)");
            ResultSet rs=ps.executeQuery();
            while(rs.next()){
                u=rs.getString("UserName");
            }

            PreparedStatement ps1=con.prepareStatement("select UserName from
registereduser where commentCount= (select Max(commentCount) from registereduser)");
            ResultSet rs1=ps1.executeQuery();
            while(rs1.next()){
                u1=rs1.getString("UserName");
            }
            u=u+","+u1;
            return u;
        }catch(Exception e){
        }
        return u;
    }

}

```

B. SNAPSHOTS

Enter Details

Your Name
Your UserName
Your Mobile Number
Type of Organization ▼
Name Of Organization
Your Email Id
Your Password

Registration Page

Enter Login Details

Your UserName
Your Password
SUBMIT

Login Page



300 GB
 Support at \$25/hour
 Full cloud access
[Buy](#)



400 GB
 Support at \$15/hour
 Full cloud access
[Buy](#)



500 GB
 Support at \$5/hour
 Limited cloud access
[Buy](#)



STATUS SIGN OUT

Check Storage



5000 GB
 Support at \$25/hour
 Full cloud access
[Buy](#)



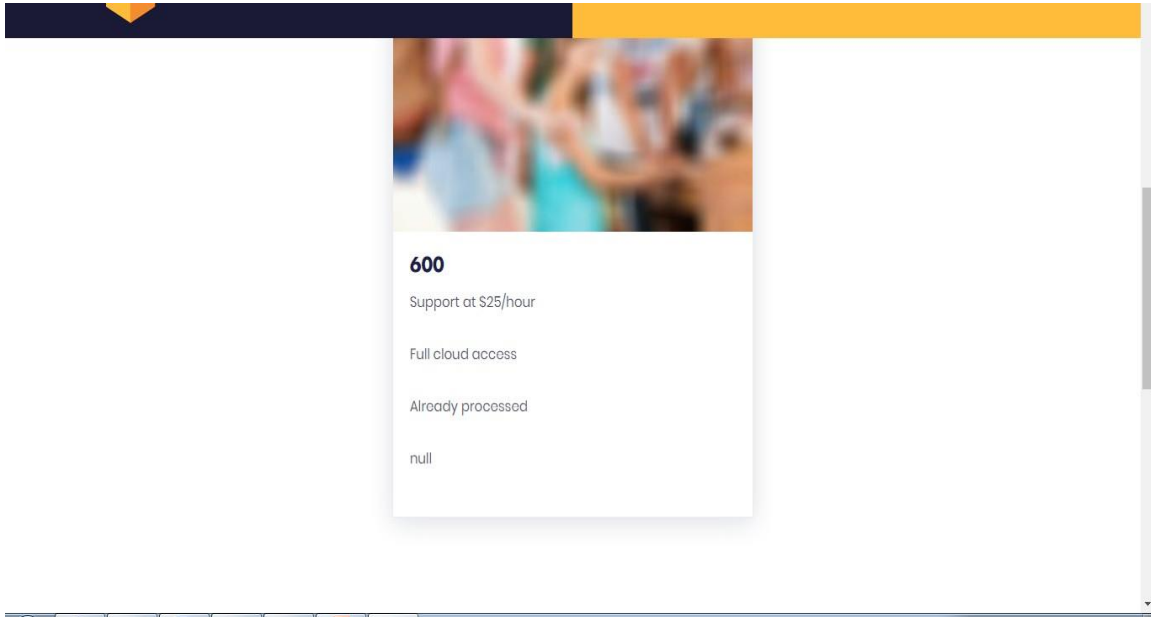
900 GB
 Support at \$15/hour
 Full cloud access
[Buy](#)



Customize GB
 Support at \$5/hour

[Buy](#)

Memory Home Page



Requested Memory Status

Admin
Check Storage

UserName	Company Type	Amount		
rajsehb	small	200GB	Accept	Decline

Admin Request Page

Admin

Check Storage

UserName	Company Type	Amount	free Memory
sasi	large	500GB	0GB
praveen	medium	300GB	100GB
kaala	large	500GB	500GB
vinaya	large	500GB	500GB
asv	large	500GB	500GB
uttejib	large	600GB	600GB

Admin Old User Data Page

Admin

Check Storage

UserName	Memory taken from old User	Memory taken from Admin Server
kaala	300GB	200GB
vinaya	0GB	500GB
asv	100GB	500GB
uttejib	0GB	600GB
rajsehb	0GB	200GB
sureshb	0GB	100GB

Admin Allocation Details Page

Turnitin Originality Report

Processed on: 29-Apr-2020 11:26 AM EDT

ID: 1311162158

Word Count: 3438

Submitted: 1

Similarity Index

3%

Similarity by Source

Internet Sources: 1%
Publications: 0%
Student Papers: 3%

12 By 12 12

2% match (student papers from 13-Sep-2009)

[Submitted to New Hope-Solebury High School on 2009-09-13](#)

1% match (student papers from 14-Mar-2020)

[Submitted to Humble Independent School District on 2020-03-14](#)

< 1% match (Internet from 28-Oct-2014)

<http://ftp.cbi.pku.edu.cn/pub/database/hssp/hssp/1dy7.hssp>

< 1% match (Internet from 23-Jun-2015)

http://www.thinkmind.org/download_full.php?instance=CLOUD+COMPUTING+2014

< 1% match (student papers from 10-Jun-2019)

[Submitted to Uxbridge College, Middlesex on 2019-06-10](#)

< 1% match (student papers from 04-Jul-2019)

[Submitted to Far Eastern University on 2019-07-04](#)

MEMORY MANAGEMENT USING MACHINE LEARNING A SRI VINAY[1],B
UTTEJ[2], M MAHESWARI [3].[1][2] [UG Student, Dept. of Computer
Science, Sathyabama University, Chennai, India](#) [3] [Assistant professor,
Dept. of Computer Science, Sathyabama University, Chennai, India](#)

Abstract: - We build up a partner approach for the computerized and versatile administration of memory in shared bunches execution information investigation applications Our methodology known as ElasticMem contains a strategy for progressively ever Switching memory confines in Java virtual machines models to anticipate memory utilization and trash pickup cost and an arranging recipe that powerfully Reallocates memory between requests Investigations with our model execution show that our methodology outflanks static memory designation bringing about less inquiry disappointments once memory is rare up to eightieth lower junk assortment overheads and up to half hour lower question times once memory is voluminous I INTRODUCTION Analyzing large databases is a critical concern and a lot of To facilitate this function wide data structures are available [2, 29, 33, 36, 48, 53]. Such structures run in shared-nothing clusters to manage giant sizes of information. whether or not public or non-public, clusters unit generally shared by several queries (also referred to as "applications")1 and even several systems death penalty within the identical cluster at constant time. A resource manager [25, 47] is responsible for the resource sharing between systems and applications within these common clusters. Modern resource managers think of containers (e.g., YARN[47], iDocker[3], or iKubernetes[5]), those separate programs use the same computer