

REDUCE VM MIGRATION IN BANDWIDTH CLOUD DATA CENTRES

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

by

Kundella Venkata Jagadeesh Kumar(37110395)

M S R MohanTeja (37110412)



**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**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade 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 **Kundella Venkata Jagadeesh Kumar(37110395)** and **M S R MohanTeja (37110412)** who carried out the project entitled "**REDUCE VM MIGRATION IN BANDWIDTH CLOUD DATA CENTRES**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Dr.A.Viji Amutha Mary,M.Tech.,Ph.D.,

Head of the Department

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

Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I Kundella Venkata Jagadeesh Kumar(37110395) and M S R MohanTeja (37110412) hereby declare that the Project Report entitled “**REDUCE VM MIGRATION IN BANDWIDTH CLOUD DATA CENTRES**” is done by me under the guidance of Dr.A.Viji Amutha Mary,M.Tech.,Ph.D.,Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my 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. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Viji Amutha Mary,M.Tech.,Ph.D.,Associate Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Cloud Computing is one of the emerging basic necessities in the IT world and has shown immense growth over the years. The applications can be run remotely using other people's servers. This is done with a simple user interface or an application format. In general, Cloud is simply the Internet which in turn is a network of remote servers. 'Pay only for what you use' is the bottom line for computing in cloud environment. The Cloud is a pool of heterogeneous resources.

The storage dealt with cloud computing has opened the gates to Load Balancing. Load Balancing refers to the process of balancing the load efficiently among the servers such that no server is overloaded or under loaded. The prime motive of this project is to minimize the response time thereby increasing the performance.

This is implemented by using the Improved Weighted Round Robin dividing the workload efficiently and successfully, thereby multiplying the scalability and the performance of computing, subtracting the response time.

Ant Colony Optimization is basic foraging behavior of an ant that encouraged them to find the optimal shortest path.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii
CHAPTER	TITLE
No.	PAGE No.
1.	INTRODUCTION
	1
	1.1 INTRODUCTION OF VM
	1
	1.2 RESEARCH AND SIGNIFICANCE
	4
2.	LITERATURE SURVEY
	5
	2.1 REGARDING VARIOUS VM IN SURVEY
	5
3.	AIM AND SCOPE
	9
	3.1 AIM OF PROJECT
	9
	3.2 OBJECTIVE
	9
	3.3 EXISTING SYSTEM
	9
	3.3.1 DISADVANTAGES
	10
	3.4 PROPOSED SYSTEM
	10
	3.4.1 ADVANTAGES
	10
4.	METHODS AND ALGORITHMS
	11
	4.1 INTRODUCTION
	11
	4.2 APPLICATION OF JAVA
	11
	4.3 FEATURES OF JAVA
	11
	4.4 MYSQL
	14
	4.4.1 SQL SERVER STUDIO
	16
	4.4.2 CREATE A NEW DATABASE
	16
	4.5 ALGORITHM USED
	20
	4.6 REQUIREMENT ANALYSIS
	20
	4.6.1 FUNCTIONAL REQUIREMENTS
	20
	4.6.2 NON-FUNCTIONAL REQUIREMENTS
	21

4.7 REQUIREMENT SPECIFICATIONS	22
4.8 MODULES	22

5.	RESULT AND DISCUSSION	25
	5.1 INPUT DESIGN	25
	5.2 OUTPUT DESIGN	26
	5.3 BASICAS OF SOFTWARE TESTING	28
	5.4 TYPES OF TESTING	28
6.	SUMMARY AND CONCLUSIONS	31
	6.1 SUMMARY	31
	6.2 CONCLUSION	31
	REFERENCES	32
	APPENDIX	
	A. PLAGIARISM REPORT	34
	B. JOURNAL PAPER	37
	C. SOURCE CODE	46

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	PLATFORM REGARDING JAVA	12
4.2	COLLECTION FRAMEWORK	14
4.3	GRAPHICAL UI OF FRAMEWORK	15
4.4	SQL SERVER MANAGEMENT	16
4.5	CREATING NEW DATABASE	17
4.6	ARCHITECTURE DIAGRAM	19
5.1	UI OF INPUT SCREEN	25
5.2	INPUT DATA FORM	26
5.3	OUTPUT DESIGN	27
5.4	OUTPUT ANALYSIS	27

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION OF VM

The rapidly increasing demands of modern resource rigorous enterprise and scientific applications have stimulated the conception of large-scale Cloud Data Centers(CDC). Usually, CDCs are over-provisioned to guarantee absolute service reliability and availability. However, on average, 30 % of cloud servers persistently remain idle the majority of time and often employ 10–15 % of their resource capacity.Resource underutilization results in truly phenomenal growth of cloud operational and energy consumption. According to Environmental Protection Agency(EPA) report , it was tentatively estimated that by 2011 CDCs would consume61 TWh of electricity. In 2013, it was roughly estimated that Google data centers devoured 260 million Watts of electricity, which is 0.01 % of the global energy—an ample amount to consistently power 200,000 homes. As today's clouds have adequate processing power, virtualization technology is chosen to consolidate several workloads onto a single physical server to significantly improve cloud resource utilization and power efficiency. Motivated by multitasking and time- sharing systems,virtualization has significantly improved the Return on Investment (ROI) due to optimal resource usage. Furthermore, virtualization technology scales CDC by migrating virtual servers across physical servers to achieve various resource management objectives such as server maintenance, load balancing , power management, and fault tolerance.The CDC embodies a relatively large group of networked servers and storage farms offering highly reliable services to organizations for remote storage, processing, or distribution of fairly large-scale data. CDC provides a foundation for the cloud computing paradigm to offer elastic cloud services to its subscribers.More recently, scientists, developers, and business personnel have begun to efficiently utilize cloud service models encompassing Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) to adeptly meet business

goals. Virtualization technology proficiently manages CDC through highly dynamic resource provisioning, effective resource monitoring, automated security, and systematic change request handling to efficiently utilize underlying resources. Virtualization

runs a software layer between the hardware and operating system, known as a virtual machine monitor (VMM) or hypervisor, to control and map numerous VMs (applications running inside OS) on a single platform. The hypervisor dexterously manages shared hardware resources and entertains OS requests to access hardware resources. Among characteristics that set the hypervisor apart include support for the legacy operating system, fault tolerance, and performance isolation to augment CDC performance. VMM supports the legacy operating system to unify several underutilized server workloads onto a single server. It also guarantees that the failure of one VM does not impact the proper functioning of the entire physical machine. However, co-hosting multiple VMs onto a single physical server is very challenging due to resource contention among co-hosted applications and system performance degradation attributed to server overutilization.

To resolve the issues stated above, the hypervisor carefully selects and migrates VMs from over-utilized to underutilized servers. The role of VM migration is two fold. To achieve energy consumption, it unifies various servers' workloads onto a few physical servers. Then, to improve an application's performance, the hypervisor migrates the workload from a low-performance server to a high-performance server . As additional resources are consumed during the VM migration process, the performance of applications running within migrant VMs is severely affected until VM migration completes. Consequently, the VM migration process must be successfully completed within nominal migration time (to free system resources as soon as possible) while utilizing optimal server and network resources to enhance application performance, bandwidth utilization efficiency, and migration transparency. The hypervisor effectively exploits live or non-live VM migration patterns to move VMs between corresponding servers either using dedicated or shared system resources. A live VM migration pattern continues servicing the application during elapsed VM migration time to achieve seamless connectivity, avoid service level agreement (SLA) violation, and attain optimal resource utilization. Non-live VM migration halts application services during VM migration and offers predictable VM migration and service downtime. For efficient resource management, the VM migration controller either migrates a single VM or a complete VM cluster (multiple VMs) across LAN or WAN links. However, VM

migration within LAN is easy to manage since storage migration is not required owing to the network-attached storage(NAS) integrated CDC architectures.

Furthermore, network management within LAN only requires limited management effort because the IP address remains unchanged for the outside world. Alternatively, VM migration over a WAN link considerably prolongs VM migration duration due to storage migration, IP management, limited available bandwidth, network congestion, and the error-prone nature of WAN links.

To the best of our knowledge, there exist only a few surveys that underline the significance of VM migration within CDC. The authors surveyed performance overhead of virtualization within a physical server, intra-CDC and inter-CDC VM migration. Nevertheless, this survey is lacking in briefly discussing the non-live VM migration methods. Furthermore, the proposed taxonomy does not cover the various vital aspects of VM migration technology, including VM migration pattern, execution resource constraints, and VM migration granularity to highlight the trade-off between resource consumption and application performance during VM migration process. Similarly, a survey covering diverse aspects of process migration, memory migration, and suspend/resume based VM migration schemes have been reported. However, the authors considered only a very few VM migration approaches with inadequate analysis of VM migration approaches. Further, the authors did not consider performance aspects of running applications during VM migration, network optimization, and hybrid VM migration pattern to improve migration processes. In comparison to the above-mentioned studies, our review is more detailed and comprehensive. Our proposed classification and analysis are more diverse and cover various aspects of VM migration, which are missing in reported literature such as optimization applied by pre-copy, post-copy, and hybrid methods to optimize the network, running, and co-hosted application's performance. Moreover, the offline VM migration schemes are also thoroughly investigated to highlight their strengths and weaknesses. The main contribution of this article is to comprehensively review state-of-the-art VM migration schemes while considering several essential VM migration aspects, including highlighting the trade-off between application performance and total migration time, network performance optimization, offline VM migration, pre-copy and post-copy VM migration pattern, and hybrid methods to meet resource management objectives within CDC. We critically reviewed state-of-the-art VM migration schemes and highlighted their strengths, weaknesses, and issues that need further research. A

novel taxonomy for VM migration schemes is proposed to classify existing

literature. Furthermore, we briefly discussed several applications of VM migration process and parameters that affect VM migration process. The highly critical aspects and significant features of the existing VM migration schemes are investigated through qualitative analysis. We derive key parameters from the literature for comparisons of VM migration schemes. The comparison parameters include: (1) Migration Pattern, (2) Bandwidth Utilization Efficiency, (3) App QoS Degradation Duration, (4) Network Link, (5) Migration Granularity, (6) Migration Duration, (7) Downtime, (8) Execution Resource Constraints, (9) Hypervisor Type, and (10) Network Management Policy. Finally, open research issues and challenges in VM migration schemes are discussed that need further investigation to develop optimal VM migration schemes to increase body of knowledge in this domain of study. The rest of the paper is structured as follows. Section 2 presents discussion on CDC, existing virtualization technologies, and VM migration process. Section 3 discusses the taxonomy for the classification of current VM migration schemes, the performance parameters, and VM migration technologies. Section 4 presents detailed review on state-of-the-art VM migration schemes, analysis of existing VM migration schemes, and comparisons of recent VM migration schemes based on selected parameters. Conclusions and future directions are followed by a brief discussion on open research issues and challenges in Sects. 5 and 4, respectively.

1.2 RESEARCH AND SIGNIFICANCE

Processing speed of a system is always highly intended.

- Distributed computing system provides high performance environment that are able to provide huge processing power.
- In distributed computing thousands of processors can be connected either by wide area network or across a large number of systems which consists of cheap and easily available autonomous systems like workstations or PCs.
- The distribution of loads to the processing elements is simply called the load balancing.
- The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become neither overloaded nor idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance

(minimum execution time) of the system.

Chapter-2

LITERATURE SURVEY

2.1 REGARDING VARIOUS VM IN SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Energy-aware resource allocation heuristics for efficient management of data centres for cloud computing

Cloud computing offers utility-oriented IT services to users worldwide. Based on a pay-as-you-go model, it enables hosting of pervasive applications from consumer, scientific, and business domains. However, data centers hosting Cloud applications consume huge amounts of electrical energy, contributing to high operational costs and carbon footprints to the environment. Therefore, we need Green Cloud computing solutions that can not only minimize operational costs but also reduce the environmental impact. In this paper, we define an architectural framework and principles for energy-efficient Cloud computing. Based on this architecture, we present our vision, open research challenges, and resource provisioning and

allocation algorithms for energy-efficient management of Cloud computing environments. The proposed energy-aware allocation heuristics provision data center

resources to client applications in a way that improves energy efficiency of the data center, while delivering the negotiated Quality of Service (QoS). In particular, in this paper we conduct a survey of research in energy-efficient computing and propose: (a) architectural principles for energy-efficient management of Clouds; (b) energy-efficient resource allocation policies and scheduling algorithms considering QoS expectations and power usage characteristics of the devices; and (c) a number of open research challenges, addressing which can bring substantial benefits to both resource providers and consumers. We have validated our approach by conducting a performance evaluation study using the CloudSim toolkit. The results demonstrate that Cloud computing model has immense potential as it offers significant cost savings and demonstrates high potential for the improvement of energy efficiency under dynamic workload scenarios.

Energy-Efficient Management of Virtual Machines in Data Centres for Cloud Computing

Cloud computing has revolutionized the information technology industry by enabling elastic on-demand provisioning of computing resources. The proliferation of Cloud computing has resulted in the establishment of large-scale data centers around the world containing thousands of compute nodes. However, Cloud data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. In 2010, energy consumption by data centers worldwide was estimated to be between 1.1% and 1.5% of the global electricity use and is expected to grow further. This thesis presents novel techniques, models, algorithms, and software for distributed dynamic consolidation of Virtual Machines (VMs) in Cloud data centers. The goal is to improve the utilization of computing resources and reduce energy consumption under workload independent quality of service constraints. Dynamic VM consolidation leverages fine-grained fluctuations in the application workloads and continuously reallocates VMs using live migration to minimize the number of active physical nodes. Energy consumption is reduced by dynamically deactivating and reactivating physical nodes to meet the current resource demand. The proposed approach is distributed, scalable, and efficient in managing the energy- performance trade-off.

Energy efficient resource management in virtualized cloud data centres

Rapid growth of the demand for computational power by scientific, business and web- applications has led to the creation of large-scale data centers consuming enormous amounts of electrical power. We propose an energy efficient resource management system for virtualized Cloud data centers that reduces operational costs and provides required Quality of Service (QoS). Energy savings are achieved by continuous consolidation of VMs according to current utilization of resources, virtual network topologies established between VMs and thermal state of computing nodes. We present first results of simulation-driven evaluation of heuristics for dynamic reallocation of VMs using live migration according to current requirements for CPU performance. The results show that the proposed technique brings substantial energy savings, while ensuring reliable QoS. This justifies further investigation and development of the proposed resource management system.

VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centres

In recent years, the power costs of cloud data centers have become a practical concern and have attracted significant attention from both industry and academia. Most of the early works on data center energy efficiency have focused on the biggest power consumers (i.e., computer servers and cooling systems), yet without taking the networking part into consideration. However, recent studies have revealed that the network elements consume 10–20% of the total power in the data center, which poses a great challenge to effectively reducing network power cost without adversely affecting overall network performance. Based on the analysis on topology characteristics and traffic patterns of data centers, this paper presents a novel approach, called VMPlanner, for network power reduction in the virtualization-based data centers. The basic idea of VMPlanner is to optimize both virtual machine placement and traffic flow routing so as to turn off as many unneeded network elements as possible for power saving. We formulate the optimization problem, analyze its hardness, and solve it by designing VMPlanner as a stepwise optimization approach with three approximation algorithms. VMPlanner is implemented and evaluated in a simulated environment with traffic traces collected from a data center test-bed, and the experiment results illustrate the efficacy and efficiency of this

approach.

A modified algorithm for VM migration to reduce network load in cloud data centres

The gist of cloud computing come from the concept of virtualization. It is a way of materializing a number of guest OS on a single host server such that CPU and memory resources of the host machine are shared with the guest systems. The guest machines are technically called as virtual machines (VMs). Virtualization enables complete utilization of a physical machine in a cost-effective manner. However, varying and continuous load from users may overload a physical machine and can shepherd to serious participation on performance, reliability and other service-level-agreement (SLA) parameters. To deal with this issue, VM migrations are capable which enables the transfer of a virtual machine from an overloaded host to underloaded host, thereby reducing the workload of the source host. In a datacenter load balancing gives a chance to obtain a materially high fault-tolerance, a feature which is very urgent during live application executions in cloud environment. The current research and documents concern is the unwanted time consumption, power utilization, and energy consumption for virtual machine migration in data center. In this paper, we developed a virtual machine migration plan for minimizing total migration time and Energy consumption. In developed virtual machine migration plan, uses monitored technique the load on every host in the data center to decide whether or not to perform VM migration and then we established the criteria to deciding which VM to migrate. An additional host is use to pre-copying processed data and sent to destination host whether VM is migrate, which reduces the total migration time and energy VM migration in datacenter. VM Migration refers to a transparent transfer of an active guest or virtual machine from a source server to a selected destination server.

Chapter–3

AIM AND SCOPE OF THE PRESENT IVESTIGATION

3.1 AIM OF PROJECT

Processing speed of a system is always highly intended.

- Distributed computing system provides high performance environment that are able to provide huge processing power.
- In distributed computing thousands of processors can be connected either by wide area network or across a large number of systems which consists of cheap and easily available autonomous systems like workstations or PCs.
- The distribution of loads to the processing elements is simply called the load balancing.
- The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become neither overloaded nor idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance (minimum execution time) of the system.

3.2 OBJECTIVE

The main objective of this system, Cloud load balancing is a type of load balancing that is performed in cloud computing. Cloud load balancing is the process of distributing workloads across multiple computing resources.

3.3 EXISTING SYSTEM

- ❖ Load balancing is one of the most important factors that has been considered to heighten the working performance of the cloud service provider. The collective system that holds individual nodes are reassigned the total workload by the process of load balancing.
- ❖ The existing systems have worked on making complete utilization of resources and improving the response time of the job, looking into the load balancing process that holds the condition where some of the nodes involved are overloaded while others are not.

- ❖ In other words, the load balancing process in the cloud environment simply means equal and proper distribution of workload across all the nodes involved

in the mechanism. Therefore, load balancing algorithms are chosen appropriately that allows selection of virtual machines or servers.

- ❖ The load balancing mechanism serves to achieve the goals that include maintaining stability of the system considered, creating backups in order to build a system that is fault tolerant, adaptation to certain modifications that must result in the increasing flexibility of the system, thereby providing substantial improvement in performance.
- ❖ The supreme feature of balancing the load also lies in reducing the response time greatly, that leads to maximum throughput, thus satisfying every user operating on the cloud. The tasks that point to a certain system must be treated equally, regardless of the geographical locations from wherever they have arrived.
- ❖ All of the above mentioned factors collaboratively make up an efficient load balanced cloud environment with suitable algorithms that support the features.

3.3.1 DISADVANTAGES OF EXISTING SYSTEM:

- Requires a constant internet connection.
- Doesn't not work well in low speed connections.
- May be slow.
- Limited features.

3.4 PROPOSED SYSTEM:

The Improved Weighted Round Robin algorithm works such that in addition to considering the specification of the servers it also considers the execution time of the tasks such that it assigns the task with the maximum execution time to the server with the maximum weight. This ensures that the load is distributed evenly among the servers thereby minimizing the response time.

3.4.1 ADVANTAGES OF PROPOSED SYSTEM:

Lower software cost.
Improve performance.
Fewer maintenance issue.
Instant software updates.

Increased data safety.

CHAPTER- 4

EXPERIMENTAL OR MATERIALS ,METHODS AND ALGORITHMS

4.1 INTRODUCTION

Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Unlike some other computer languages whose influence has wearied with passage of time, while Java's has grown.

4.2 APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like Linkedin.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

4.3 FEATURES OF JAVA

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are;

1) Simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

Eg: Pointers and Operator Overloading are not there in java but were an important part of C++.

2) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

3) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

4) Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.

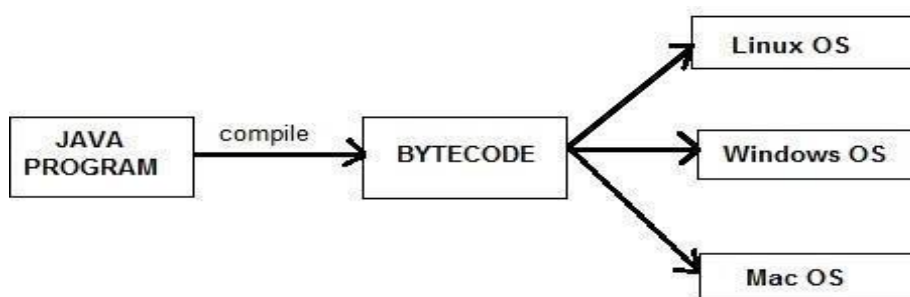


Fig 4.1: PLATFORM REGARDING JAVA

5) Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6) Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

Portable

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

COLLECTION FRAMEWORK

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.

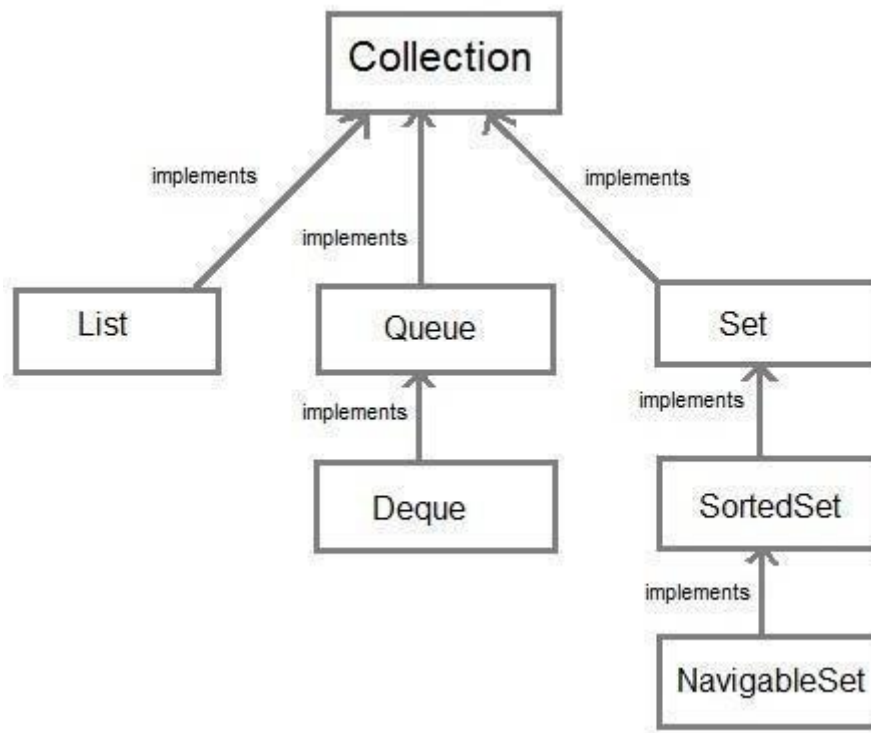


Fig 4.2:-COLLECTION FRAMEWORK

4.4 MYSQL

MySQL, officially, but also called "My Sequel" is the world's most widely used open-source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases, though SQLite probably has more total embedded deployments. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks).

LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Freesoftware-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional

functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used

in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube. Microsoft is the vendor of SQL Server. We have different editions of SQL Server, where SQL Server Express is free to download and use. SQL Server uses T-SQL (Transact-SQL). T-SQL is Microsoft's proprietary extension to SQL. TSQL is very similar to standard SQL, but in addition it supports some extra functionality, built in functions, etc.

T-SQL expands on the SQL standard to include procedural programming, local variables, and various support functions for string processing, date processing, mathematics, etc. SQL Server consists of a Database Engine and a Management Studio (and lots of other stuff which we will not mention here). The Database engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server). The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).

SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

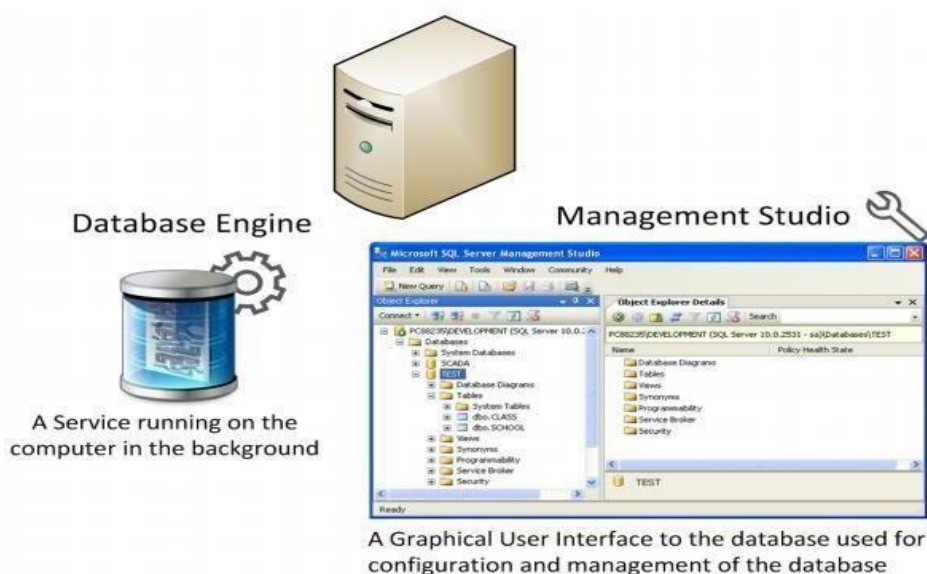


FIG 4.3:GRAPHICAL UI OF DATABASE

4.4.1 SQL SERVER MANAGEMENT STUDIO

SQL Server Management Studio is a GUI tool included with SQL Server for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. As mentioned earlier, version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express.

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

When creating SQL commands and queries, the "Query Editor" (select "New Query" from the Toolbar) is used (shown in the figure above). With SQL and the "Query Editor" we can do almost everything with code, but sometimes it is also a good idea to use the different Designer tools in SQL to help us do the work without coding (so much).

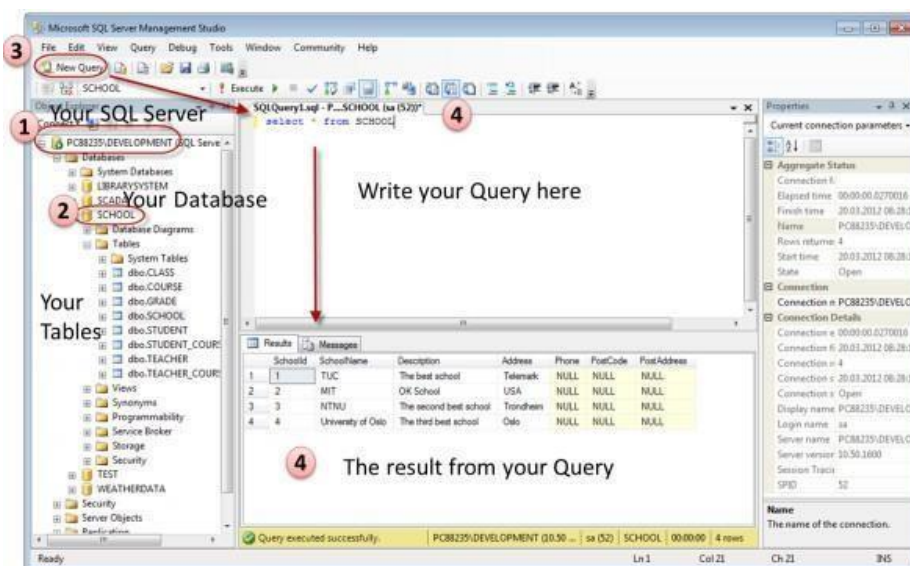


Fig 4.4: SQL SERVER MANAGEMENT

4.4.2 Create a new Database

It is quite simple to create a new database in Microsoft SQL Server. Just right-

click on the "Databases" node and select "New Database..."

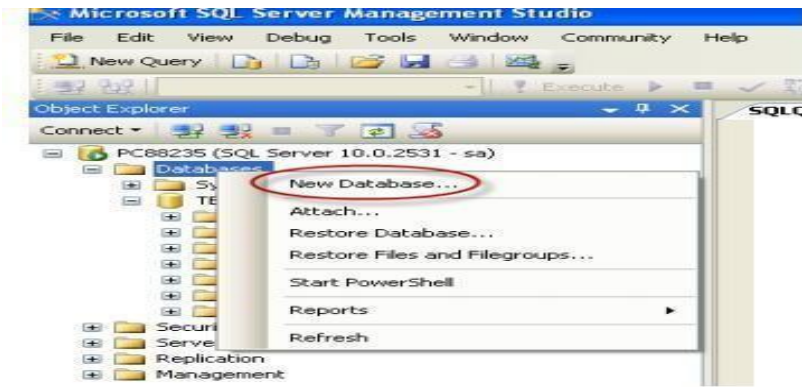


Fig 4.5 CREATING NEW DATABASE

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects, They are:-

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

Table

A database is a collection of data about a specific topic.

Views of Table

We can work with a table in two types,

1. Design View
2. Datasheet View

Design View

To build or modify the structure of a table we work in the table design view.

We can specify what kind of data will be hold.

Datasheet View

To add, edit or analyses the data itself we work in tables datasheet view mode.

Query

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited). Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical

requirements of the system. Any system developed must not have a high demand on

the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

SYSTEM ARCHITECTURE

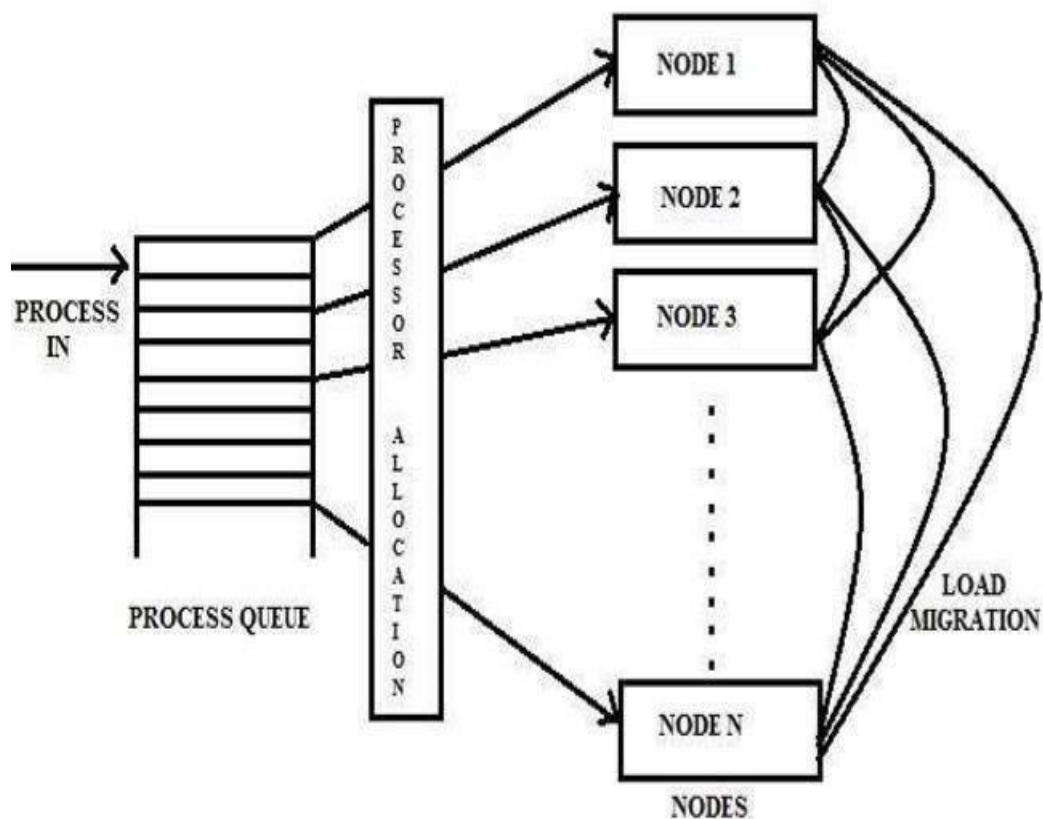


Fig 4.6 : ARCHITECTURE DIAGRAM

4.5 ALGORITHM USED

ROUND ROBIN ALGORITHM

The proposed algorithm is the Improved Weighted Round Robin algorithm. The algorithm takes its origin from the Weighted Round Robin algorithm. The Weighted Round Robin algorithm works by assigning the maximum weight to the server with the best specifications. For e.g. if there are two servers one with the quad core processors and increased processing speed and the other server with dual core processors and comparatively minimum processing speed then the server with the best specification is assigned the maximum weight. In short the Weighted Round Robin algorithm is a special case of the Round Robin algorithm that works well with servers of different specifications.

4.6 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.6.1 FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability:-

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness:-

It refers to a program that performs well not only under ordinary conditions but

also under unusual conditions. It is the ability of the user to cope with errors for

irrelevant queries during execution.

Security:

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability:-

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility:-

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility:-

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety:-

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.6.2 NON- FUNCTIONAL

REQUIREMENTS Portability:-

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance:-

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy:-

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability:-

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means

that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

4.7 REQUIREMENT SPECIFICATION

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the customer's machines already. So the main need is to introduce appropriate equipment for the product. The requirements are split into two categories, namely:-

Software Requirements

The basic software requirements to run the program are:

- i. Operating system : Windows 7.
- ii. Coding Language : Java
- iii. Toolkit : Netbeans
- iv. Database : MySQL

Hardware Requirements

The basic hardware required to run the program are:

- i. Hard disk of 120 GB
- ii. System memory (RAM) of 1 GB
- iii. I5 processor-based computer or higher.

4.8 MODULES

- Static load balancing
 - Optimal static load balancing.
 - Sub optimal static load balancing
- Dynamic load balancing
- Quality of load balancing

Static load balancing

In static algorithm the processes are assigned to the processors at the compile time according to the performance of the nodes.

- Once the processes are assigned, no change or reassignment is possible at the run time.

- Number of jobs in each node is fixed in static load balancing algorithm. Static algorithms do not collect any information about the nodes .
- The assignment of jobs is done to the processing nodes on the basis of the following factors: incoming time, extent of resource needed, mean execution time and inter- process communications.
- Since these factors should be measured before the assignment, this is why static load balance is also called probabilistic algorithm.

Optimal Static Load Balancing Algorithms

- If all the information and resources related to a system are known optimal static load balancing can be done.
- It is possible to increase throughput of a system and to maximize the use of the resources by optimal load balancing algorithm.

Sub-Optimal Static Load Balancing Algorithm

- Sub-optimal load balancing algorithm will be mandatory for some applications when optimal solution is not found.
- The thumb-rule and heuristics methods are important for sub-optimal algorithm.

Dynamic Load Balancing

In dynamic load balancing algorithm assignment of jobs is done at the runtime.

- In DLB jobs are reassigned at the runtime depending upon the situation that is the load will be transferred from heavily loaded nodes to the lightly loaded nodes.
- In this case communication over heads occur and becomes more when number of processors increase.
- In dynamic load balancing no decision is taken until the process gets execution. This strategy collects the information about the system state and about the job information.
- As more information is collected by an algorithm in a short time, potentially the algorithm can make better decision.
- Dynamic load balancing is mostly considered in heterogeneous system because it consists of nodes with different speeds, different communication link

speeds, different memory sizes, and variable external loads due to the multiple.

Quality of a load balancing algorithm

Quality of a load balancing algorithm is dependent on two factors.

- Firstly number of steps that are needed to get a balanced state.
- Secondly the extent of load that moves over the link to which nodes are connected.

CHAPTER-5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

What data should be given as input?

How the data should be arranged or coded?

The dialog to guide the operating personnel in providing input.

Methods for preparing input validations and steps to follow when error occur.

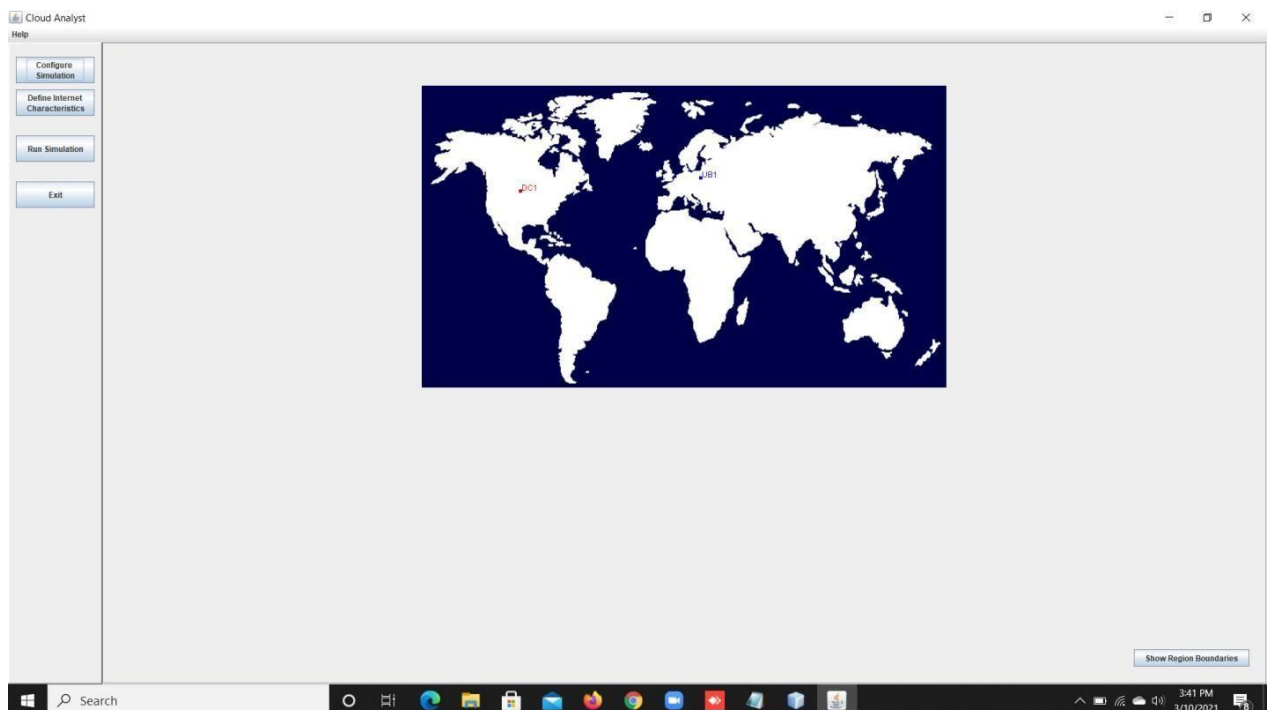


Fig 5.1: UI OF INPUT SCREEN

Configure Simulation

Simulation Duration: 60.0 min

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	21	60	100	3	9	1000	100

Service Broker Policy: Closest Data Center

Application Deployment Configuration:

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000

Buttons: Cancel, Load Configuration, Save Configuration, Done

Fig 5.2 : INPUT DATA FORM

5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.

- Signal important events, opportunities, problems, or warnings.

- Trigger an action.

- Confirm an action.

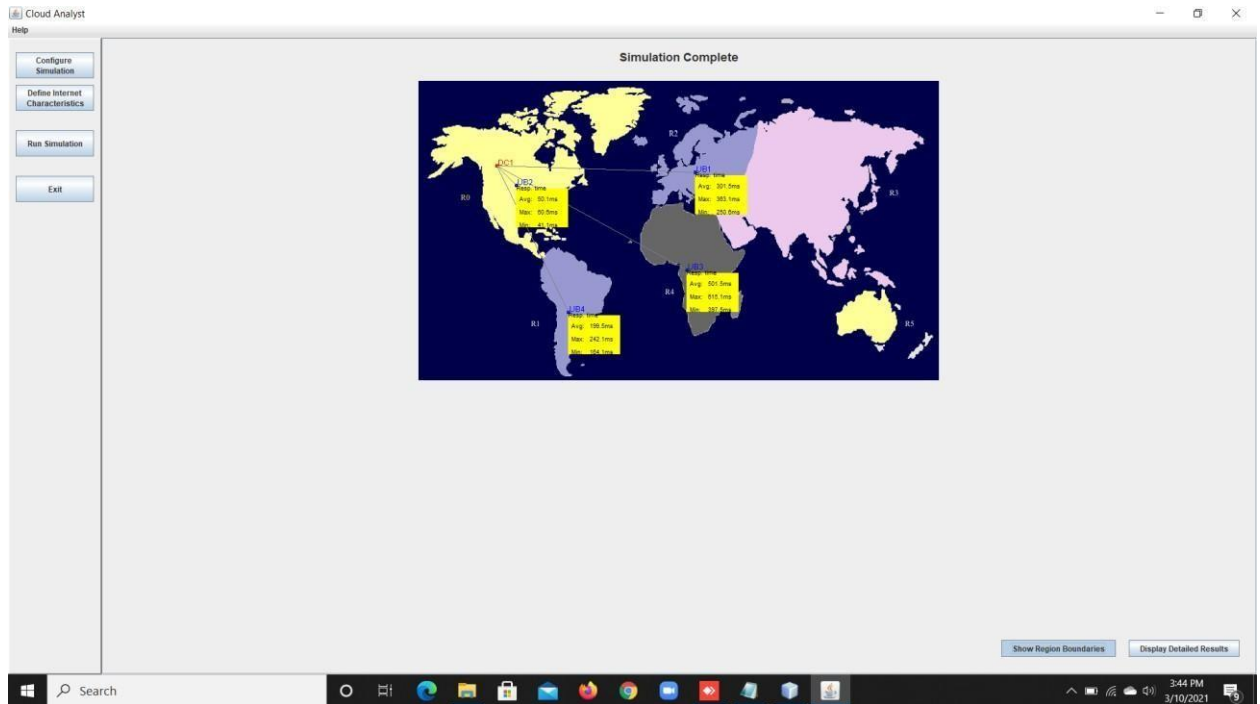


Fig 5.3 : OUTPUT DESIGN

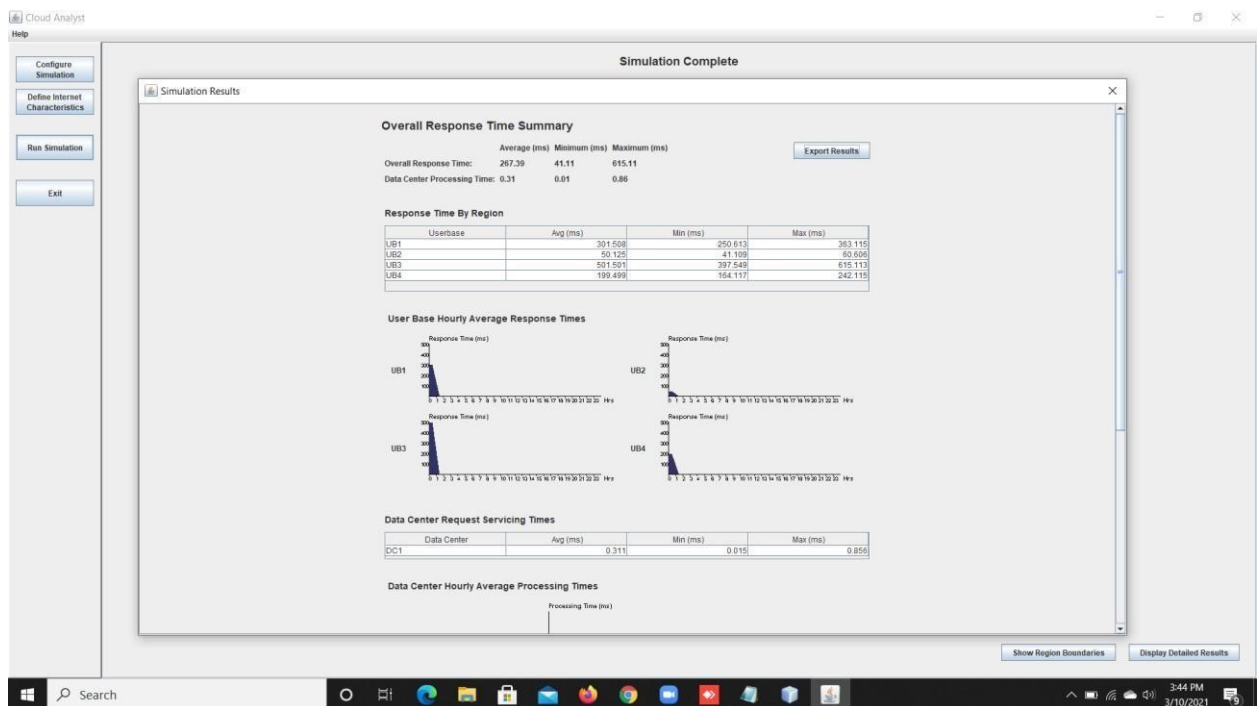


Fig 5.4 : OUTPUT ANALYSIS

Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that

should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

5.3 Basics of software testing

There are two basics of software testing: black box testing and white box testing.

Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

5.4 Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing

- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under

the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

CHAPTER-6

SUMMARY AND CONCLUSIONS

6.1 Summary:-

Cloud Computing is one of the emerging basic necessities in the IT world and has shown immense growth over the years. The applications can be run remotely using other people's servers. This is done with a simple user interface or an application format. In general, Cloud is simply the Internet which in turn is a network of remote servers. 'Pay only for what you use' is the bottom line for computing in cloud environment. The Cloud is a pool of heterogeneous resources.

The storage dealt with cloud computing has opened the gates to Load Balancing. Load Balancing refers to the process of balancing the load efficiently among the servers such that no server is overloaded or under loaded. The prime motive of this project is to minimize the response time thereby increasing the performance.

This is implemented by using the Improved Weighted Round Robin dividing the workload efficiently and successfully, thereby multiplying the scalability and the performance of computing, subtracting the response time.

Ant Colony Optimization is basic foraging behavior of an ant that encouraged them to find the optimal shortest path.

6.2 Conclusion:-

This is implemented by using the Improved Weighted Round Robin dividing the workload efficiently and successfully, thereby multiplying the scalability and the performance of computing, subtracting the response time. Ant Colony Optimization is basic foraging behaviour of an ant that encouraged them to find the optimal shortest path. The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become neither overloaded nor idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance (minimum execution time) of the system.

References:-

- [1] B. Anthon, A. Ilemal, B. Rajkumar, "Energy-aware resource allocation heuristics for efficient management of data centres for cloud computing " Future Generation Computer Systems, ELSEVIER (2011).
- [2] B. Anthon "Energy-Efficient Management of Virtual Machines in Data Centres for Cloud Computing", Ph.D. dissertation, Department of Computing and Information Systems, the university of Melbourne Australia, Feb 2013.
- [3] B. Anthon., B. Rajkumar, "Energy efficient resource management in virtualized cloud data centres", In: IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp.826-831, May 2010.
- [4] V. Mann, A Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A Iyer, "Remedy: Network-aware steady state VM management for data centres," in Proc. IFIP TC 6 Networking Conf., ser. LNCS, 2012, vol. 7289, pp. 190-204.
- [5] Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A, Raz, D., Silvera, E.: A stable network-aware VM placement for cloud systems. In: Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), pp. 498-506. IEEE (2012).
- [6] Fang, W., Liang, X., Li, S., Chiaraviglio, L., Xiong, N.: VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centres. *Comput. Net.* 57(1), 179-196 (2013).
- [7] Shahzad, K.; Iqbal Umer, A; Nazir, B.; Bilal, K.; Kamran; Zaib Alam, M., "A modified algorithm for VM migration to reduce network load in cloud data centres," 1st National Conference on Mathematics and Computer Science (NCMCS'15) January 21-23, 2015.
- [8] Tziritas, N.; Cheng-Zhong Xu; Loukopoulos, T.; Khan, S.U.; Zhibin Yu, "Application-Aware Workload Consolidation to Minimize Both Energy Consumption and Network Load

in Cloud Environments," Parallel Processing (ICPP), 2013 42nd International Conference on , vol., no., pp.449,457, 1-4 Oct. 2013.

[9] Bari, M.F., Boutaba, R., Esteves, R., Podlesny, M., Rabbani, M.G., Zhang, Q., Zhani, M.F.: Data centre network virtualization: a survey. *IEEE Commun.Surv.*15 (2), 909-928(2013).

[10] F.P. Tso, G. Hamilton, K. Oikonomou, D.P , " Implementing Scalable, Network-Aware Virtual Machine Migration for Cloud Data Centres ",ACM 2013.

A.Plagiarism Report:-

REDUCE VM MIGRATION IN BANDWIDTH
OVERSUBSCRIBED CLOUD DATA
CENTRES

ORIGINALITY REPORT

11%

SIMILARITY INDEX

8%

INTERNET SOURCES

9%

PUBLICATIONS

9%

STUDENT PAPERS

PRIMARY SOURCES

1

%

Khyyam Shahzad, Arif Iqbal Umer, Babar Nazir.

6

"Reduce VM migration in bandwidth
oversubscribed cloud data centres", 2015
IEEE 12th International Conference on
Networking, Sensing and Control, 2015

Publication

Marimuthu, K., D. Ganesh Gopal, K.
Sashi Kanth, Srujay Setty, and Kunal
Tainwala.

"Scalable and secure data sharing for
dynamic groups in cloud", 2014 IEEE
International Conference on Advanced
Communications Control and Computing
Technologies, 2014.

Publication

Submitted to Asia Pacific International College

3

Student Paper

1%

www.jpinfotech.org

4

Internet Source

1%

www.slideshare.net

1%

6

www.greendc.eu

Internet Source

1

%

7

docplayer.net

Internet Source

<1%

8

pdfs.semanticscholar.org

Internet Source

<1%

9

psasir.upm.edu.my

Internet Source

<1%

Exclude quotes

On Exclude

Exclude matches

Off

bibliography

On

REDUCE VM MIGRATION IN BANDWIDTH CLOUD DATA CENTRES

Kundella Venkata Jagadeesh Kumar^[1], Maddala Sai Ratna MohanTeja^[2], Dr.A.Viji Amutha Mary^[3] ^[1]B.E.

Computer science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India. ^[2]B.E.

Computer science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India.

^[3]M.Tech.,Ph.d.,Professor,School of Computing, Sathyabama Institute of Science and Technology, Chennai, India.

INDEX TERMS:- Cloud computing, computers and information processing, deep learning, distributed computing, machine learning, serverless architectures.

ABSTRACT

Distributed computing is one of the essential necessities arising in the IT world and has shown huge development throughout the long term. You can utilize others' workers to run applications distantly. This is done through a straightforward UI or application design. Generally, the cloud is only the Internet, which is an organization of distant workers. "Pay just for what you use" is the base measure of PC use in the cloud space. The cloud is an assortment of various assets.

Cloud-based figuring stockpiling makes the way for load adjusting. Burden adjusting alludes to the way toward adjusting the heap between workers in a manner that there is no worker over-burden or download. The reason for this venture is to lessen reaction time and consequently increment proficiency.

This is finished utilizing an improved weighted cooperative technique, which can successfully isolate the heap, accordingly expanding the dispersion and execution of the PC, along these lines disposing of reaction time.

The utilization of insect provinces is an essential technique for eating ants, which can urge ants to locate the briefest course.

INTRODUCTION

a heap adjusting

Processing speed of a system is always highly intended.

- The dispersed PC framework gives an elite climate that can give ground-breaking preparing capacities.
- On a dispersed PC, a great many processors can be associated through a wide region organization or countless frameworks (like workstations or PCs) that contain reasonable self-governing frameworks and are not difficult to utilize.
- Load circulation on preparing hardware is alluded to as burden adjusting for short.
- The motivation behind the heap adjusting calculation is to keep up the heap on each preparing component so that all handling materials won't be over-burden or inactive, which implies that at some random time during framework activity, each handling thing should have a similar burden (Minimum preparing time).

OBJECTIVE

The fundamental motivation behind this program is that distributed computing is

performed on distributed computing. Cloud stacking is the way toward dispersing responsibility across numerous PC assets.

system considered, creating

EXISTING SYSTEM

- ❖ Bal load adjusting is perhaps the main components to improve the exhibition of cloud specialist organizations. The collective system that holds individual nodes are reassigned the total workload by the process of load balancing.
- ❖ The existing systems have worked on making complete utilization of resources and improving the response time of the job, looking into the load balancing process that holds the condition where some of the nodes involved are overloaded while others are not.
- ❖ In other words, the load balancing process in the cloud environment simply means equal and proper distribution of workload across all the nodes involved in the mechanism. Therefore, load balancing algorithms are chosen appropriately that allows selection of virtual machines or servers.
- ❖ The load balancing mechanism serves to achieve the goals that include maintaining stability of the

backups in order to build a system that is fault tolerant, adaptation to certain modifications that must result in the increasing flexibility of the system, thereby providing substantial improvement in performance.

- ❖ The supreme feature of balancing the load also lies in reducing the response time greatly, that leads to maximum throughput, thus satisfying every user operating on the cloud. The tasks that point to a certain system must be treated equally, regardless of the geographical locations from wherever they have arrived.
- ❖ All of the above-mentioned factors collaboratively make up an efficient load balanced cloud environment with suitable algorithms that support the features.

DISADVANTAGES OF EXISTING SYSTEM:

- Requires a constant internet connection.
- Doesn't not work well in low speed connections.
- May be slow.
- Limited features.

PROPOSED SYSTEM:

The Weighted Round Robin algorithm can check the worker particulars, yet additionally check an opportunity to execute the assignment, with the goal that the errand has more opportunity to execute on the heavier worker. This guarantees that the heap is uniformly circulated among the workers, along these lines diminishing reaction time.

ADVANTAGES OF PROPOSED SYSTEM:

- Lower software cost.
- Improve performance.
- Fewer maintenance issue.
- Instant software updates.
- Increased data safety.

LITERATURE REVIEW

Cloud computing gives IT benefits that are utilized by worldwide clients. Contingent upon the pay-more only as costs arise model, it can have applications from buyers, science and business. Nonetheless, server farms that host cloud frameworks utilize a great deal of force, bringing about higher working expenses and carbon impression in the climate. Accordingly, we need a green PC arrangement that can decrease working expenses, yet additionally lessen the effect on the climate. In this article, we will layout the

designs and frameworks of PC produced energy saving frameworks. In view of this ability, we focus on our vision, open exploration difficulties, and asset allotment and energy-saving designation calculations in distributed computing conditions. The proposed power designation framework gives information assets to client applications such that upgrades the effectiveness of the server farm limit, while giving arranged nature of administration (QoS). Specifically, in this article, we examine PC helped energy investigate and propose: (a) Principles for the advancement of energy proficiency the board; (b) Strategy to save energy-saving assets and arranging considering QoS assumptions for gadget and force utilization Algorithm Features; (c) Addressing explicit open-finished examination challenges, which could carry critical advantages to asset suppliers and buyers. We did execution examination exercises utilizing the CloudSim tool stash to approve our methodology. The outcomes show that the distributed computing model is amazing in light of the fact that it can set aside a ton of cash and can possibly improve energy productivity in unique burden conditions.

Energy-Efficient Management of Virtual Machines in Data Centres for Cloud Computing

Cloud computing has changed the data innovation industry by conveying on-request and requested PC administrations. The multiplication of distributed computing has prompted the formation of huge server farms containing a huge number of registering hubs around the planet. Nonetheless, cloud server farms utilize greater power, bringing about higher working expenses and carbon dioxide outflows. In 2010, the force utilization of worldwide server farms is assessed to represent 1.1% and 1.5% of worldwide power utilization, and is relied upon to develop consistently. This paper presents new advances, models, calculations, and programming for the joining of virtual computer-generated simulation (VM) into cloud server farms. The point is to improve the utilization of PC assets and to diminish the utilization of force under free work hindrances. Dynamic VM reconciliation profits by unobtrusive varieties in framework working burdens and constantly conveys VMs with live relocation to diminish the quantity of dynamic actual hubs. By effectively shutting and recovering body hubs to meet current asset necessities, power utilization is decreased. The proposed approach is as yet boundless, far and wide, and viable in overseeing energy

exchanges.

Energy efficient resource management in virtualized cloud data centres

The quick expansion in the interest for PC power in logical, business, and web applications has prompted the making of huge server farms that utilization a lot of power. We propose an asset proficient asset the board framework for cloud-based server farms that can lessen working expenses and offer genuinely necessary assistance (QoS). With the current utilization of assets, the geography of the virtual organization set up among VMs and the hot condition of figuring hubs, energy saving is accomplished by ceaselessly coordinating VMs. In accordance with current CPU execution prerequisites, we have proposed the initially set of reenacted exploratory outcomes utilizing a virtual reallocation framework for virtual machines utilizing live relocation. The outcomes show that the proposed innovation brings huge energy investment funds while guaranteeing dependable QoS. This demonstrates the affectability of the continuous examination and advancement of the proposed asset the board system.

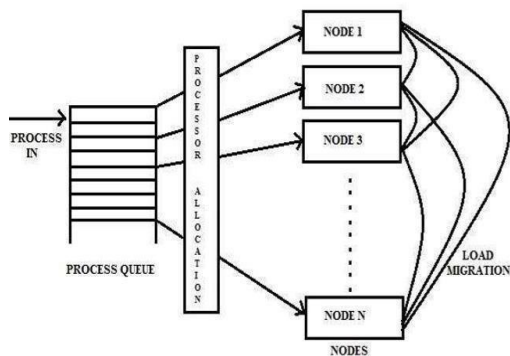
VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centres

As of late, the energy cost of cloud server farms has gotten increasingly more constrained and has pulled in extraordinary consideration from the business and the scholarly world. The greater part of the underlying elements of server farm execution are centered around the primary force clients (ie, PC workers and cooling frameworks) without thinking about correspondence segments. In any case, late examinations have shown that network components devour 10% to 20% of the all out force of the server farm, which represents a significant test for viably lessening network power costs without annihilating generally speaking organization execution. Based on breaking down the topic attributes and street examples of server farms, this paper proposes a novel technique called VMPlanner to decrease the organization limit of information based server farms. The fundamental thought of VMPlanner is to expand the genuine machine arrangement and traffic, to dispose of however many organization pointless things as could be expected under the circumstances to save energy. We proposed the advancement issue, examined its intricacy, and tackled the issue by naming VMPlanner as a sluggish convenience utilizing a three-dimensional

calculation. VMPlanner is utilized and tried in a street based climate gathered in a

server farm proving ground, and the test outcomes show the viability of the technique.

SYSTEM ARCHITECTURE



SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

System : Pentium Dual Core.
 Hard Disk : 120 GB.
 Monitor : 15"LED
 Input Devices : Keyboard, Mouse
 Ram : 1GB.

SOFTWARE REQUIREMENTS:

Operating system : Windows 7.
 Coding Language : Java Toolkit
 :Netbeans
 DATABASE : MySQL

ALGORITHM USED

ROUND ROBIN ALGORITHM

The proposed algorithm is the Improved Weighted Round Robin algorithm. The algorithm takes its origin from the Weighted Round Robin algorithm. The Weighted Round Robin algorithm works by assigning the maximum weight to the server with the best specifications. For e.g. if there are two servers one with the quad core processors and increased processing speed and the other server with dual core processors and comparatively minimum processing speed then the server with the best specification is assigned the maximum weight. In short the Weighted Round Robin algorithm is a special case of the Round Robin algorithm that works well with servers of different specifications.

MODULES

- Static load balancing
 - Optimal static load balancing.
 - Sub optimal static load balancing
- Dynamic load balancing
- Quality of load balancing

Static load balancing

balancing algorithm.

In static algorithm the processes are assigned to the processors at the compile time according to the performance of the nodes.

- Once the processes are assigned, no change or reassignment is possible at the run time.
- Number of jobs in each node is fixed in static load balancing algorithm. Static algorithms do not collect any information about the nodes .
- The assignment of jobs is done to the processing nodes on the basis of the following factors: incoming time, extent of resource needed, mean execution time and inter-process communications.
- Since these factors should be measured before the assignment, this is why static load balance is also called probabilistic algorithm.

Optimal Static Load Balancing Algorithms

- If all the information and resources related to a system are known optimal static load balancing can be done.
- It is possible to increase throughput of a system and to maximize the use of the resources by optimal load

Sub-Optimal Static Load Balancing Algorithm

- Sub-optimal load balancing algorithm will be mandatory for some applications when optimal solution is not found.
- The thumb-rule and heuristics methods are important for sub-optimal algorithm.

- Dynamic load balancing is mostly considered in heterogeneous system

Dynamic Load Balancing

In dynamic load balancing algorithm assignment of jobs is done at the runtime.

- In DLB jobs are reassigned at the runtime depending upon the situation that is the load will be transferred from heavily loaded nodes to the lightly loaded nodes.
- In this case communication overheads occur and become more when number of processors increase.
- In dynamic load balancing no decision is taken until the process gets execution.

This strategy collects the information about the system state and about the job information.

- As more information is collected by an algorithm in a short time, potentially the algorithm can make better decision.

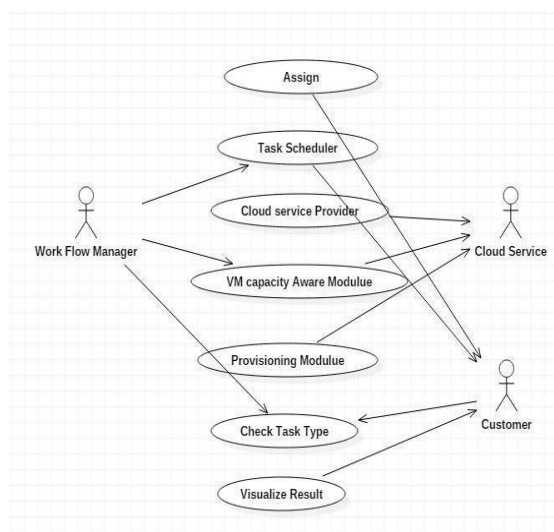
because it consists of nodes with different speeds, different communication link speeds, different memory sizes, and variable external loads due to the multiple.

Quality of a load balancing algorithm

Quality of a load balancing algorithm is dependent on two factors.

- Firstly number of steps that are needed to get a balanced state.
- Secondly the extent of load that moves over the link to which nodes are connected.

USECASE DIAGRAM



CONCLUSION

This is implemented by using the Improved Weighted Round Robin dividing the workload efficiently and successfully, thereby multiplying the scalability and the performance of computing, subtracting the

response time. Ant Colony Optimization is basic foraging behaviour of an ant that encouraged them to find the optimal shortest path. The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become neither overloaded nor idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance (minimum execution time) of the system.

pp.826-831, May 2010.

REFERENCES

- [1]B.Anthon,A.lemal,B.Rajkumar,"Energy-aware resource allocation heuristics for efficient management of data centres for cloud computing " Future Generation Computer Systems,ELSEVIER (2011).
- [2] B.Anthon
"Energy-Efficient Management of Virtual Machines in Data Centres for Cloud Computing", Ph.D. dissertation,Department of Computing and Information Systems, the university of melbourneAustralia,Feb 2013.
- [3] B.Anthon., B.Rajkumar ,"Energy efficient resource management in virtualized cloud data centres", In: IEEEI ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid),

[4] V. Mann, A Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A Iyer, "Remedy: Network-aware steady state VM management for data centres," in Proc. IFIP TC 6 Networking Conf., ser. LNCS, 2012, vol. 7289, pp. 190-204.

[5] Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A, Raz, D., Silvera, E.: A stable network-aware VM placement for cloud systems. In: Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), pp. 498-506. IEEE (2012).

[6] Fang, W., Liang, X., Li, S., Chiaraviglio, L., Xiong, N.: VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centres. Comput. Net. 57(1), 179-196 (2013).

[7] Shahzad, K.; Iqbal Umer, A; Nazir, B.; Bilal, K.; Kamran; Zaib Alam, M., " A modified algorithm for VM migration to reduce network load in cloud data centres," 1st National Conference on Mathematics and Computer Science (NCMCS'15) January 21-23, 2015

[8] Tziritas, N.; Cheng-Zhong Xu; Loukopoulos, T.; Khan, S.U.; Zhibin Yu, "Application-Aware Workload Consolidation to Minimize Both Energy Consumption and Network Load in Cloud

(ICPP), 2013 42nd International Conference on , vol., no., pp.449,457, 1-4 Oct. 2013.

[9] Bari, M.F., Boutaba, R., Esteves, R., Podlesny, M., Rabbani, M.G., Zhang, Q., Zhani, M.F.: Data centre network virtualization: a survey. IEEE Commun.Surv.15 (2), 909-928(2013).

[10] F.P. Tso, G. Hamilton, K. Oikonomou, D.P. , " Implementing Scalable, Network-Aware Virtual Machine Migration for Cloud Data Centres ",ACM 2013.

C. SOURCE CODE

```
package rallocloud.main;

import

java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.PrintWriter;

import

org.cloudbus.cloudsim.*;

import

java.text.DecimalFormat;

import java.util.ArrayList;

import

java.util.Calendar;

import

java.util.HashMap;

import

java.util.HashSet;

import

java.util.LinkedList;

import java.util.List;

import java.util.Map;

import

java.util.logging.Level;

import

java.util.logging.Logger;

import org.apache.commons.math3.distribution.PoissonDistribution;

import

org.apache.commons.math3.distribution.UniformRealDistribution;

import org.cloudbus.cloudsim.Cloudlet;
```

```
import
org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Pe;
import
org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import
org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
```

```

import org.cloudbus.cloudsim.core.CloudSim;

import
org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import org.cloudbus.cloudsim.NetworkTopologyPublic;
import org.cloudbus.cloudsim.provisioners.BwProvisionerNetworked;

/**
 *
 * @author Atakan
 */
public class RalloCloud
{
    private static int
    vmid = 0;

    private static int cloudletid = 0;

    private static final HashSet<DatacenterBrokerStrategy> brokerSet = new
    HashSet<>(); public static PrintWriter out = null;

    public static String
    strategy; private static
    int vmRAM; private
    static int vmBW; private
    static int vmNUM;

    private static int vmNW;

    private enum
    topologyType {
        LINEAR, CIRCULAR, COMPLETE, STAR    }

    public static void main(String[]
    args) { try {

        boolean printList = true; //Human

```

readable? vmRAM = 4;

vmBW = 4;

vmNUM = 4;

vmNW = 2;

strategy = "SNW";

```

if (args.length >
0) { printList
= false;
if (args.length > 1) {
    vmRAM = Integer.parseInt(args[1]);
}
if (args.length > 2) {
    vmBW = Integer.parseInt(args[2]);
}
if (args.length > 3) {
    vmNUM = Integer.parseInt(args[3]);
}
if (args.length > 4) {
    vmNW = Integer.parseInt(args[4]);
}
strategy = args[0];

    out = new PrintWriter(new BufferedWriter(new FileWriter("dist/out/" + vmRAM +
 "-" + vmBW + "-" + vmNUM + "-" + vmNW + ".txt", true)));

    out.println(strategy);
}      int num_user = 15;
Calendar calendar =
Calendar.getInstance(); boolean trace_flag
= false; CloudSim.init(num_user, calendar,
trace_flag); ArrayList<String> labels = new
ArrayList<>(); labels.add("GARR");
labels.add("DFN");
labels.add("CESNE
T");
labels.add("PSNC")
;

```

```
labels.add("FCCN")  
;  
labels.add("GRNET  
");
```

```

labels.add("HEANE
T");
labels.add("I2CAT")
;
labels.add("ICCS");
labels.add("KTH");
labels.add("NIIF");
labels.add("PSNC-2");
labels.add("RedIRIS");
labels.add("SWITCH");
labels.add("NORDUNET");

int[] numVnodes = {2, 2, 2, 3, 1, 1, 2, 2, 1, 2, 1, 3, 1, 1};

NetworkTopologyPublic.buildNetworkTopology("D:\\cloud based
framework\\Rallo
cloud\\RalloCloud-master\\RalloCloud-master\\RalloCloud\\data\\federica.brite");
NetworkTopologyPublic.setNextIdx(NetworkTopologyPublic.getBwMatrix().length);

ArrayList<Datacenter> dcList = new
ArrayList<>(); for (int i = 0; i < 14; i++) {
    int n = numVnodes[i];

    Datacenter dc = createDatacenter(labels.get(i), 1538 * 4 * n, 65536 * n, 4000000 *
4000);

                                dcList.add(
                                dc);

                                NetworkTo
                                pologyPubl
                                ic.mapNod
                                es(dc, i);

                                }      Datacenter dc =
                                createDatacenter(labels.get(14), 0, 0, 0, 4000);
                                //Empty

datacenter for nordunet

    dcList.add(dc);

    NetworkTopologyPublic.mapNodes(dc,

```

```
14); int i = 0;
for (Datacenter d : dcList) {
    String name = "BROKER"
    + i; i++;
    labels.add(name);
    createBroker(dcList, name, d.getId(), printList);    }
```

```

for (DatacenterBrokerStrategy bs :
    brokerSet) { int count = (bs.getPopulation()
    * vmNUM) / 10; count = count == 0 ? 1 :
    count;
    for (i = 0; i < count; i++) {
        createVmGroup(bs, 3, 200, topologyType.LINEAR);
        createVmGroup(bs, 2, 200, topologyType.COMPLETE);
    }
}
    NetworkTopologyPublic.setBrokerSet(brokerSet);
//Visualizer.emptyTopology(MyNetworkTopology.getBwMatrix(), labels);
//START        CloudSim.startSimulation();
List<Cloudlet> clList = new ArrayList<>();
ArrayList<List<Cloudlet>> clSepList = new
ArrayList<>();
HashMap<Integer, Integer> VmsToDatacentersMap = new
HashMap<>(); for (DatacenterBrokerStrategy bs : brokerSet) {
    if (bs.getCloudletSubmittedList().isEmpty()) {
        continue;
    }
    clList.addAll(bs.getCloudletSubmittedList());
    clSepList.add(bs.getCloudletSubmittedList());
    VmsToDatacentersMap.putAll(bs.getVmsToDatacentersMa
    p());
}
    CloudSim.stopSimulation();
//STOP
System.out.println
("");
printCloudletList(clList,
printList); if (printList) {
    DecimalFormat dft = new DecimalFormat("###.##");

```

```
System.out.println("Distribution Factor (DSF)\t:  
\t" + dft.format(Statistician.getDSF(clSepList)));
```

```

dcList)));

System.out.println("Load Balance (LDB)\t\t: \t"
+ dft.format(Statistician.getLDB(clList,

printVmList(VmsToDatacentersMap, labels);

} else {

    out.println(Statistician.getDSF(clSepList));
    out.println(Statistician.getLDB(clList, dcList));
    out.println("");
    out.close();
}

System.out.println("");
System.out.println(Statistician.getEndTime());
System.out.println("");
} catch (Exception e) {
    //e.printStackTrace();

    System.out.println("The simulation has been terminated due to an unexpected error");
} } private static Double[][] createVmGroup(DatacenterBrokerStrategy broker, int
count, double time, topologyType type) {
    int brokerId = broker.getId();
    PoissonDistribution pd = new PoissonDistribution(count); //for VM
    count count = pd.sample();
    UniformRealDistribution urd = new UniformRealDistribution(0, time); //For request
    time time = urd.sample();
    if (count ==
        0) {
        return
        null;
    }    ArrayList<Integer> group = new
    ArrayList<>(); for (int i = 0; i < count; i++) {
        int mips = 50;
        long size = 10000; //image size (MB)

```

```
int ram = 1024 * vmRAM; //vm memory (MB)
```

```

    long bw = 50 * vmBW;

    int pesNumber = 1; //number of
    cpus String vmm = "Xen"; //VMM
    name

    Vm virtualMachine = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());

    group.add(vmid);

    broker.getVmList().add(virtualMachine);

    broker.getAllVmList().add(virtualMachin
e); long length = 1500;

    long fileSize = 250 * vmNW;

    long outputSize = 250 *
vmNW;

    UtilizationModel utilizationModel = new UtilizationModelFull();

    Cloudlet application = new Cloudlet(cloudletid, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);

    application.setUserId(brokerId);

    broker.getCloudletList().add(application);

    broker.bindCloudletToVm(application.getCloudletId(), virtualMachine.getId());

    vmid++;

    cloudletid++;

}

Double[][] topology = new
Double[count][count]; for (int i = 0; i < count;
i++) {
    for (int j = 0; j < count;
        j++) { topology[i][j] =
            0.0;

        }
    }
}

```

```
}    if (type ==  
    topologyType.LINEAR) { for (int i =  
    0; i < count - 1; i++) {  
        topology[i][i + 1] =  
        1.0; topology[i +  
        1][i] = 1.0;  
    }
```

```

    } else if (type ==
        topologyType.COMPLETE) { for (int i =
            0; i < count; i++) {
                for (int j = 0; j < count;
                    j++) { if (i != j) {
                        topology[i][j] = 1.0;
                    }
                }
            }
        }
    } else if (type ==
        topologyType.CIRCULAR) { for (int i =
            0; i < count - 1; i++) {
                topology[i][i + 1] =
                1.0; topology[i +
                1][i] = 1.0;
            }
        topology[count - 1][0] = 1.0;
        topology[0][count - 1] = 1.0;
    } else if (type ==
        topologyType.STAR) { for (int i =
            1; i < count; i++) {
                topology[i][0] = 1.0;
                topology[0][i] = 1.0;
            }
        }
    broker.getVmGroups().put(group,
        topology); broker.getGroupTimes().put(group, time);
    return topology;
}
private static Datacenter createDatacenter(String name, int mips, int ram, long
storage, int bw) {
    // Here are the steps needed to create a PowerDatacenter:
    // 1. We need to create a list to store

```

```
// our machine  
List<Host> hostList = new ArrayList<>();  
// 2. A Machine contains one or more PEs or CPUs/Cores.
```

```

// In this example, it will have only one
core. List<Pe> peList = new
ArrayList<>();
for (int i = 0; i < 64; i++) {
    peList.add(new Pe(i, new PeProvisionerSimple(mips)));
}    //4. Create Host with itsid and list of PEs and add them to the list of
machines int hostId = 0;
hostList.add(
    new Host(
        hostId,
        new RamProvisionerSimple(ram),
        new BwProvisionerNetworked(bw,
            -1), storage,
        peList,
        new VmSchedulerSpaceShared(peList)    )
); // This is our machine

// 5. Create a DatacenterCharacteristics object that stores the
// properties of a data center: architecture, OS, list of
// Machines, allocation policy: time- or space-shared, time zone
// and its price (G$/Pe time unit).
String arch = "x86"; // system
architecture String os = "Linux"; //
operating system String vmm = "Xen";
double time_zone = 10.0; // time zone this resource located
double cost = 1; // the cost of using processing in this
resource
double costPerMem = 0.05; // the cost of using memory in this
resource double costPerStorage = 0.001; // the cost of using storage in this

```

resource

double costPerBw = 0.0;

// the cost of using bw in this resource

```
LinkedList<Storage> storageList = new LinkedList<>();           //we are not adding
SAN devices by now
```

```
DatacenterCharacteristics characteristics = new
    DatacenterCharacteristics( arch, os, vmm, hostList, time_zone,
    cost, costPerMem, costPerStorage, costPerBw);
```

```
// 6. Finally, we need to create a PowerDatacenter
object. Datacenter datacenter = null;
try {
    datacenter = new Datacenter(name, characteristics,
new VmAllocationPolicySimple(hostList), storageList, 0);
    } catch (Exception e) {
        //e.printStackTrace();    } return datacenter;
} /**
```

```
* Prints the Cloudlet objects
```

```
*
```

```
* @param clList list of Cloudlets
```

```
*/
```

```
private static void printCloudletList(List<Cloudlet> clList, boolean
```

```
list) { int size = clList.size();
```

```
Cloudlet
```

```
cloudlet; String
```

```
indent = "\t\t"; if
```

```
(list) {
```

```
    System.out.println("\n===== CLOUDLETS
```

```
====="); System.out.println("CL ID" + indent +
```

```
"STATUS" + indent
```

```
    + "DC Name" + indent + "DC ID" + indent + "VM ID" + indent + "Durat" + indent
    +
```

```
"Time"
```

```
    + "Cost");
```

+ indent + "Start" + indent + "Finish" + indent
+ "Broker" + indent + "Group" + indent

```

    }    double AUL
    = 0; double MUL
    = 0; double JRT =
    0; double JCT = 0;
    double CST = 0;
    double AVC = 0;

    DecimalFormat dft = new
    DecimalFormat("###.##"); for (int i = 0; i < size;
    i++) {

        cloudlet = clList.get(i);

        DatacenterBrokerStrategy broker = null;

        for (DatacenterBrokerStrategy b :
            brokerSet) { if (b.getId() ==
            cloudlet.getUserId()) {

                broker
                = b;

                break;

            }    }    List<Integer> group = null;

        for (List<Integer> l :
            broker.getVmGroups().keySet()) { if
            (l.contains(cloudlet.getVmId())) {

                group = l;    }

        }    double time =

        broker.getGroupTimes().get(group); if (list) {

            System.out.print(cloudlet.getCloudletId() + indent);

            System.out.print(cloudlet.getCloudletStatus() == Cloudlet.SUCCESS ?
            "SUCCESS" : "OTHER");

            System.out.println(indent + cloudlet.getResourceName(cloudlet.getResourceId()) +
            indent + cloudlet.getResourceId() + indent + cloudlet.getVmId()

                + indent + dft.format(cloudlet.getActualCPUTime()) + indent +
            dft.format(time) + indent + dft.format(cloudlet.getExecStartTime())

```

```
        + indent + dft.format(cloudlet.getFinishTime()) + indent +  
cloudlet.getUserId() + indent + group + indent + cloudlet.getCostPerSec());  
    }
```

```

AUL += (cloudlet.getExecStartTime() -
time); JRT +=
cloudlet.getActualCPUTime();
JCT += (cloudlet.getFinishTime() - time);
CST += cloudlet.getCostPerSec() *
cloudlet.getActualCPUTime(); AVC +=
cloudlet.getCostPerSec();
if (cloudlet.getExecStartTime() >
    MUL) { MUL =
        cloudlet.getExecStartTime();
    }
}
if (list) {
    System.out.println("\n===== METRICS =====");
    System.out.println("Average User Latency (AUL)\t: \t" + dft.format(AUL / size) +
"s"); System.out.println("Maximum User Latency (MUL)\t: \t" + dft.format(MUL)
+ "s");
    System.out.println("Average Inter-DC Latency (ADL)\t: \t" +
dft.format(Statistician.getADL()) + "s");
    System.out.println("Maximum Inter-DC Latency (MDL)\t:
\t" + dft.format(Statistician.getMDL()) + "s");
    System.out.println("Job Run Time (JRT)\t\t: \t" + dft.format(JRT / size) + "s");
    System.out.println("Job Completion Time (JCT)\t: \t" + dft.format(JCT / size) + "s");
    System.out.println("Throughput (TRP)\t\t: \t" + dft.format(Statistician.getTRP()) + "
MIPS");
    System.out.println("Rejection Rate (RJR)\t\t: \t" + dft.format(Statistician.getRJR() *
100)
+ "%");
    System.out.println("Total Cost (CST)\t\t: \t" +
dft.format(CST)); System.out.println("Average
Cost (AVC)\t\t: \t" + dft.format(AVC / size));
    System.out.println("Algorithm Calculation Time

```

```
(ACT): \t" + Statistician.getACT() + "ns");
```

```
    } else {
```

```
        out.println(AUL / size);
```

```
        out.println(MUL);
```

```
        out.println(Statistician.getAD
```

```
        L());
```

```

        out.println(Statistician.getMD
L()); out.println(JRT / size);
        out.println(JCT / size);
        out.println(Statistician.getTR
P());
        out.println(Statistician.getRJR() * 100);
        out.println(CST);
        out.println(AVC / size);
    } } private static void printVmList(Map<Integer, Integer> m,
ArrayList<String> l) { String indent = "\t\t";
    System.out.println();
    System.out.println("===== VMs =====");
    System.out.println("VM ID" + indent + "DC Name" + indent + "DC
ID"); for (int vmId : m.keySet()) {
        int dcId = m.get(vmId);
        System.out.println(vmId + indent + l.get(dcId - 2) + indent + dcId);
    } } private static DatacenterBrokerStrategy createBroker(ArrayList<Datacenter>
dcList, String name, int dcId, Boolean log) {
    try {
        DatacenterBrokerStrategy broker =
        null; switch (strategy) {
            case "AFF":
                broker = new
                AFFDatacenterBroker(name); break;
            case "ANF":
                broker = new
                ANFDatacenterBroker(name); break;
            case "LBG":
                broker = new
                LBGDatacenterBroker(name); break;

```

```

    case "LFF":
        broker = new
            LFFDatacenterBroker(name); break;
    case "LNF":
        broker = new
            LNFDatacenterBroker(name); break;
    case "RAN":
        broker = new
            RANDatacenterBroker(name); break;
    case "SNW":
        broker = new
            TBFDatcenterBroker(name); break;
    case "TBF":
        broker = new
            TBFDatcenterBroker(name); break;
    default:
        System.exit
            t(1); break;

    }

    int[] pops = {-1, -1, 61, 81, 11, 30, 10, 6, 5, 23, 5, 10, 10, 9, 23, 8, 6};
    if (!log) {
        broker.disableL
        og();
    }
    broker.setDatacenterList(dcList);
    NetworkTopologyPublic.addLink(dcId, broker.getId(), 10.0,
    0.1); broker.setPopulation(pops[dcId]);
    brokerSet.add(brok

```

```
er); return broker;  
} catch (Exception ex) {  
    Logger.getLogger(RalloCloud.class.getName()).log(Level.SEVERE, null, ex);
```

```
        }    return null;
    }
}
```