**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Recent years have witnessed a “gold rush” of online data hosting services (or says cloud storage services) such as Amazon S3, Windows Azure, Google Cloud Storage, Aliyun OSS, and so forth. These services provide customers with reliable, scalable, and low-cost data hosting functionality. More and more enterprises and organizations are hosting all or part of their data into the cloud, in order to reduce the IT maintenance cost (including the hardware, software, and operational cost) and enhance the data reliability. For example, the United States Library of Congress had moved its digitized content to the cloud, followed by the New York Public Library and Biodiversity Heritage Library. Now they only have to pay for exactly how much they have used.

**Heterogeneous clouds -** Existing clouds exhibit great heterogeneities in terms of both working performances and pricing policies. Different cloud vendors build their respective infrastructures and keep upgrading them with newly emerging gears. They also design different system architectures and apply various techniques to make their services competitive. Such system diversity leads to observable performance variations across cloud vendors. Moreover, pricing policies of existing storage services provided by different cloud vendors are distinct in both pricing levels and charging items.

**Vendor lock-in risk** - Facing numerous cloud vendors as well as their heterogeneous performances/policies, customers may be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud and then simply trust to luck. This is subject to the so-called “vendor lock-in risk”, because customers would be confronted with a dilemma if they want to switch to other cloud vendors. The vendor lock-in risk first lies in that data migration inevitably generates considerable expense. Besides, the vendor lock-in risk makes customers suffer from price adjustments of cloud vendors which are not uncommon.

**1.2 OBJECTIVE**

Data deduplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. However, there is only one copy for each file stored in cloud even if such a file is owned by a huge number of users. As a result, deduplication system improves storage utilization while reducing reliability. Furthermore, the challenge of privacy for sensitive data also arises when they are outsourced by users to cloud. Aiming to address the above security challenges, this paper makes the first attempt to formalize the notion of distributed reliable deduplication system. We propose new distributed deduplication systems with higher reliability in which the data chunks are distributed across multiple cloud servers. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems. Security analysis demonstrates that our deduplication systems are secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement the proposed systems and demonstrate that the incurred overhead is very limited in realistic environments.

**1.3 SCOPE**

As a holistic storage system, there are several other factors to be considered, such as cache strategies, geographical data consistency, etc. However, we only focus on the data hosting strategy to minimize monetary cost while meeting flexible availability requirements. Though we have considered the complexity and feasibility when designing this strategy, the system design is out of the scope of this paper, and we put the detailed system design of multi-cloud data hosting into future work. the complexity of this algorithm is mainly the first loop, and the worst case complexity is O(Fn), where Fn is the number of files. In order to reduce the complexity further, we can classify files with similar access patterns into groups, and implement transition in the unit of group. This is out of the scope of this paper.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1. Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai, “Efficient batched synchronization in dropbox-like cloud storage services,” in Proc.ACM/IFIP/USENIX 14th Int. Middleware Conf., 2013, pp. 307–327.**

The author addressed traffic overuse problem, proposed the update-batched delayed synchronization (UDS) mechanism. Acting as a middleware between the user’s file storage system and a cloud storage application, UDS batches updates from clients to significantly reduce the overhead caused by session maintenance traffic, while preserving the rapid file synchronization that users expect from cloud storage services. Furthermore, he extend UDS with a backwards compatible Linux kernel modification that further improves the performance of cloud storage applications by reducing the CPU usage.

**2.2. H. V. Madhyastha, J. C. McCullough, G. Porter, R. Kapoor, S. Savage, A. C. Snoeren, and A. Vahdat, “scc: Cluster storage provisioning informed by application characteristics and SLAs,” in Proc. USENIX Conf. File, Storage Technol., 2012, p. 23.**

Storage for cluster applications is typically provisioned based on rough, qualitative characterizations of applications. Moreover, configurations are often selected based on rules of thumb and are usually homogeneous across a deployment; to handle increased load, the application is simply scaled out across additional machines and storage of the same type. As deployments grow larger and storage options (e.g., disks, SSDs, DRAM) diversify, however, current practices are becoming increasingly inefficient in trading off cost versus performance. To enable more cost-effective deployment of cluster applications, The author developed SCC a Storage Configuration Compiler for cluster applications. SCC automates cluster configuration decisions based on formal specifications of application behaviour and hardware properties. After brief study a range of storage configurations and identify specifications that succinctly capture the tradeoffs offered by different types of hardware, as well as the varying demands of application components. The author applied SCC to three representative applications and find that scc is expressive enough to meet application Service Level Agreements (SLAs) savings in cost on average compared to simple scale-out options. SCC ’s advantage stems mainly from its ability to configure heterogeneous—rather than conventional, homogeneous—cluster architectures to optimize cost.

**2.3. H. B. Ribeiro and E. Anceaume, “Datacube: A P2P persistent data storage architecture based on hybrid redundancy schema,” in Proc. 18th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process., 2010, pp. 302–306.**

This author presented the design of a P2P data persistent platform. Durable access and integrity of the data are ensured despite massive attacks. This platform, named Data Cube, exploits the properties of cluster-based peer-to-peer substrates to implement a compound of full replication and rate less erasure codes. Data Cube guarantees durable access and integrity of data despite adversarial attacks. In particular, the recovery of damaged data is achieved through the retrieval of coded blocks whose integrity is checked on the fly.

**2.4. A. Duminuco and E. W. Biersack, “Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems,” Peer to Peer Netw. Appl., vol. 3, no. 1, pp. 52–66, 2010.**

Redundancy is the basic technique to provide reliability in storage systems consisting of multiple components. A redundancy scheme defines how the redundant data are produced and maintained. The simplest redundancy scheme is replication, which however suffers from storage inefficiency. Another approach is erasure coding, which provides the same level of reliability as replication using a significantly smaller amount of storage. When redundant data are lost, they need to be replaced. While replacing replicated data consists in a simple copy, it becomes a complex operation with erasure codes: new data are produced performing a coding over some other available data. The amount of data to be read and coded is d times larger than the amount of data produced, where d, called repair degree, is larger than 1 and depends on the structure of the code. This implies that coding has a larger computational and I/O cost, which, for distributed storage systems, translates into increased network traffic. Participants of Peer-to-Peer systems often have ample storage and CPU power, but their network bandwidth may be limited. For these reasons existing coding techniques are not suitable for P2P storage. This work explores the design space between replication and the existing erasure codes. The author proposed and evaluate a new class of erasure codes, called Hierarchical Codes, which allows to reduce the network traffic due to maintenance without losing the benefits given by traditional erasure codes.

**2.5. H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, “RACS: A case for cloud storage diversity,” in Proc. 1st ACM Symp. Cloud Comput., 2010, pp. 229–240.**

The increasing popularity of cloud storage is leading organizations to consider moving data out of their own data centers and into the cloud. However, success for cloud storage providers can present a significant risk to customers; namely, it becomes very expensive to switch storage providers. In this paper, we make a case for applying RAID-like techniques used by disks and file systems, but at the cloud storage level. The author argued that striping user data across multiple providers can allow customers to avoid vendor lock-in, reduce the cost of switching providers, and better tolerate provider outages or failures. The introduced RACS, a proxy that transparently spreads the storage load over many providers. RACS can reduce the cost of switching storage vendors for a large organization such as the Internet Archive by seven-fold or more by varying erasure-coding parameters.

**CHAPTER 3**

**SYSTEM ANALYSIS**

System are created to solve problems. One can think of the system approach as an organized way of dealing with a problem. The analysis phase is the second phase of the SDLC and to decide if the project should go ahead with the resources available. This also includes looking at any existing system to see what it is doing for the organization and how well that system is doing its job. The feasibility of the project is also considered and the group has to ask questions such as,

* Will this system significantly improve the organization?
* Can this system be created with the resources we have?
* Does the old system necessarily need to be replaced?

**3.1 OVERALL DESCRIPTION**

**3.1.1 PROBLEM DEFINITION**

* Nevertheless, as for multi-cloud people still encounter the two critical problems:
* How to choose appropriate clouds to minimize monetary cost in the presence of heterogeneous pricing policies?
* How to meet the different availability requirements of different services?
* As to monetary cost, it mainly depends on the data-level usage, particularly storage capacity consumption and network bandwidth consumption.
* As to availability requirement, the major concern lies in which redundancy mechanism (i.e., replication or erasure coding) is more economical based on specific data access patterns. In other words, here the fundamental challenge is:
* How to combine the two mechanisms elegantly so as to greatly reduce monetary cost and meanwhile guarantee required availability?
* Data Hosting and SMS are two important modules in CHARM. Data Hosting decides storage mode and the clouds that the data should be stored in.
* This is a complex integer programming problem demonstrated in the following subsections. Then we illustrate how SMS works in detail in x V, that is, when and how many times should the transition be implemented.

**3.1.2 EXISTING SYSTEM**

In existing industrial data hosting systems, data availability (and reliability) are usually guaranteed by replication or erasure coding. In the multi-cloud scenario, we also use them to meet different availability requirements, but the implementation is different. For replication, replicas are put into several clouds, and a read access is only served (unless this cloud is unavailable then) by the “cheapest” cloud that charges minimal for out-going bandwidth and GET operation. For erasure coding, data is encoded into n blocks including m data blocks and n x m coding blocks, and these blocks are put into n different clouds. In this case, though data availability can be guaranteed with lower storage space (compared with replication), a read access has to be served by multiple clouds that store the corresponding data blocks. Consequently, erasure coding cannot make full use of the cheapest cloud as what replication does. Still worse, this shortcoming will be amplified in the multi-cloud scenario where bandwidth is generally (much) more expensive than storage space.

**3.1.2.1 DISADVANTAGE**

The existing system uses either replication or erasure coding or both. There are disadvantage during Vendors lock in risk. Data storage should be made on different vendors so that when one vendor is shut then the data will be available in the other. Also there are heterogeneous clouds where the data hosting scheme changes and the cost differs.

**3.1.3 PROPOSED SYSTEM**

In this paper, we propose a novel cost-efficient data hosting scheme with high availability in heterogeneous multi-cloud, named “CHARM”. It intelligently puts data into multiple clouds with minimized monetary cost and guaranteed availability. Specifically, we combine the two widely used redundancy mechanisms, i.e., replication and erasure coding, into a uniform model to meet the required availability in the presence of different data access patterns. Next, we design an efficient heuristic-based algorithm to choose proper data storage modes (involving both clouds and redundancy mechanisms). Moreover, we implement the necessary procedure for storage mode transition (for efficiently re-distributing data) by monitoring the variations of data access patterns and pricing policies. We evaluate the performance of CHARM using both trace driven simulations and prototype experiments. The traces are collected from two online storage systems:, both of which possess hundreds of thousands of users. In the prototype experiments, we replay samples from the two traces for a whole month on top of four mainstream commercial clouds: Amazon S3, Windows Azure, Google Cloud Storage, and Aliyun OSS. Evaluation results show that compared with the major existing schemes which will be elaborated in x VII-B), CHARM not only saves around 20% (more in detail, 7% 44%) of monetary cost.

**3.1.3.1 ADVANTAGE**

Replication mechanism when the file’s size is small. That is why gray level 4 puts its feet into the region of lower read count and smaller file size. This storage mode table only depends on prices of the available clouds and required availability. If the prices change, the table will change accordingly, becoming a different one. Some clouds have minimal cost for downloading data. Mostly deleting data is of zero cost. Cost for upload and download of data may differ from the hosting scheme and the pricing model of the cloud.

**3.2 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.

Three key considerations involved in the feasibility analysis are

* TECHNICAL FEASIBILITY
* ECONOMIC FEASIBILITY
* SOCIAL FEASIBILITY

**3.2.1 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. In the proposed system technical feasibility centers around the hardware and software and to what extent it can support the proposed system. The tools that are used to develop the application are the best tools available in the technological scenario and hence it requires efficient and versatile programmers and programming skills.

**3.2.2 ECONOMIC 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 be 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. In our project all the required facilities, hardware, software to be used are initially may be costly, but when put to use it proves to be much more economical that the existing system. Regarding the maintenance, since the source code will be with us, any small and necessary changes can be done with minimum maintenance cost involved in it. So for sure the proposed system is cost effective than the existing system. Hence the proposed system is economically feasible.

**3.2.3 OPERATIONAL 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. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**3.3 SYSTEM CONFIGURATION**

**3.3.1 HARDWARE REQUIREMENTS**

Hardware - Pentium

Speed - 1.1 GHz

RAM - 1GB

Hard Disk - 20 GB

Floppy Drive - 1.44 MB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

**3.3.2 SOFTWARE REQUIREMENTS**

Operating System : Windows

Technology : Java and J2EE

Web Technologies : Html, JavaScript, CSS

Web Server : Tomcat

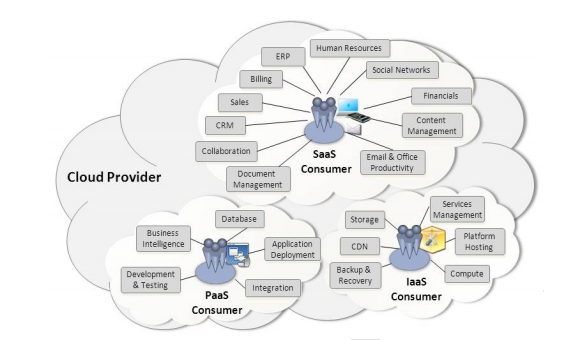
Database : My SQL

Java Version : J2SDK1.5

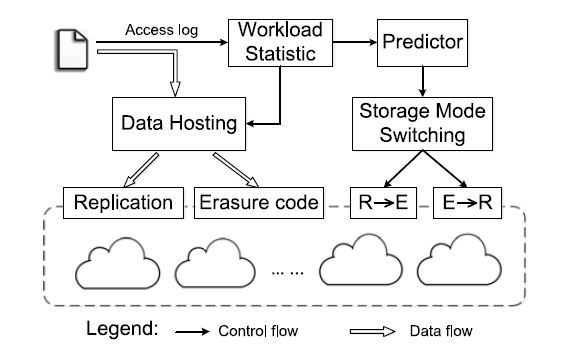
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 DETAILED SYSTEM DESIGN**

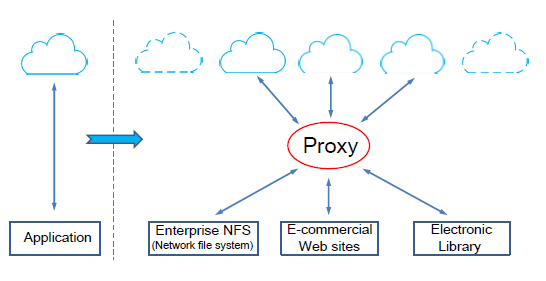
****

**fig 4.1: Services Available to a Cloud Consumer**



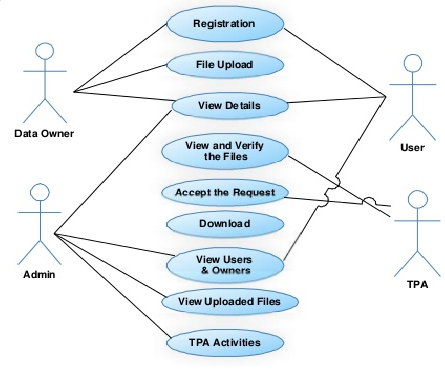
**fig 4.2: Architecture of Charm-'R'-Replication and 'E'-Erasure Coding**

**4.1.1 SYSTEM ARCHITECTURE**

****

**fig 4.1.1: Architecture Diagram**

**4.1.2 USE CASE DIAGRAM**

****

**fig 4.1.2: Use Case Diagram**

**4.1.3 CLASS DIAGRAM**

****

**fig 4.1.3: Class Diagram**

**4.1.4 ACTIVITY DIAGRAM**

****

**fig 4.1.4: Activity Diagram**

**4.1.5 SEQUENCE DIAGRAM**

****

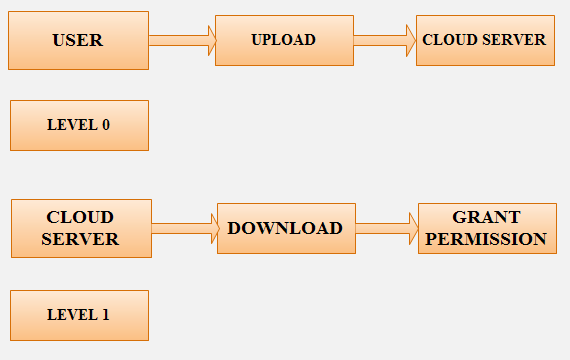
**fig 4.1.5: Sequence Diagram**

**4.1.6 COLLABORATION DIAGRAM**

****

**fig 4.1.6: Collaboration Diagram**

**4.1.7 DATA FLOW DIAGRAM**

****

**fig 4.1.7: Data Flow Diagram**

**4.2 REQUIREMENTS**

**4.2.1 FUNCTIONAL REQUIREMENTS**

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

**4.2.2 NON-FUNCTIONAL REQUIREMENTS**

Describe user-visible aspects of the system that are not directly related with the functional behaviour of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy ((.e. how precise are the systems numerical answers.)

**4.2.3 PSEUDO REQUIREMENTS**

The client that restricts the implementation of the system imposes these requirements. Typical pseudo requirements are the implementation language and the platform on which the system is to be implemented. These have usually no direct effect on the user’s view of the system**.**

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 MODULE DESCRIPTION**

**5.1.1 MULTI-CLOUD**

Lots of data centers are distributed around the world, and one region such as America, Asia, usually has several data centers belonging to the same or different cloud providers. So technically all the data centers can be access by a user in a certain region, but the user would experience different performance. The latency of some data centers is very low while that of some ones may be intolerable high. CHARM chooses clouds for storing data from all the available clouds which meet the performance requirement, that is, they can offer acceptable throughput and latency when they are not in outage. The storage mode transition does not impact the performance of the service. Since it is not a latency-sensitive process, we can decrease the priority of transition operations, and implement the transition in batch when the proxy has low workload.

**5.1.2 DATA HOSTING**

In this section, we elaborate a cost-efficient data hosting model with high availability in heterogeneous multi-cloud, named “CHARM”. The architecture of CHARM is shown in Figure 3. The whole model is located in the proxy in Figure 1. There are four main components in CHARM: Data Hosting, Storage Mode Switching (SMS), Workload Statistic, and Predictor. Workload Statistic keeps collecting and tackling access logs to guide the placement of data. It also sends statistic information to Predictor which guides the action of SMS. Data Hosting stores data using replication or erasure coding, according to the size and access frequency of the data. SMS decides whether the storage mode of certain data should be changed from replication to erasure coding or in reverse, according to the output of Predictor. The implementation of changing storage mode runs in the background, in order not to impact online service. Predictor is used to predict the future access frequency of files. The time interval for prediction is one month, that is, we use the former months to predict access frequency of files in the next month. However, we do not put emphasis on the design of predictor, because there have been lots of good algorithms for prediction. Moreover, a very simple predictor, which uses the weighted moving average approach, works well in our data hosting model. Data Hosting and SMS are two important modules in CHARM. Data Hosting decides storage mode and the clouds that the data should be stored in. This is a complex integer programming problem demonstrated in the following subsections. Then we illustrate how SMS works in detail in x V, that is, when and how many times should the transition be implemented.

**5.1.3 CLOUD STORAGE**

Cloud storage services have become increasingly popular. Because of the importance of privacy, many cloud storage encryption schemes have been proposed to protect data from those who do not have access. All such schemes assumed that cloud storage providers are safe and cannot be hacked; however, in practice, some authorities (i.e., coercers) may force cloud storage providers to reveal user secrets or confidential data on the cloud, thus altogether circumventing storage encryption schemes. In this paper, we present our design for a new cloud storage encryption scheme that enables cloud storage providers to create convincing fake user secrets to protect user privacy. Since coercers cannot tell if obtained secrets are true or not, the cloud storage providers ensure that user privacy is still securely protected. Most of the proposed schemes assume cloud storage service providers or trusted third parties handling key management are trusted and cannot be hacked; however, in practice, some entities may intercept communications between users and cloud storage providers and then compel storage providers to release user secrets by using government power or other means. In this case, encrypted data are assumed to be known and storage providers are requested to release user secrets. we aimed to build an encryption scheme that could help cloud storage providers avoid this predicament. In our approach, we offer cloud storage providers means to create fake user secrets. Given such fake user secrets, outside coercers can only obtained forged data from a user’s stored cipher text. Once coercers think the received secrets are real, they will be satisfied and more importantly cloud storage providers will not have revealed any real secrets. Therefore, user privacy is still protected. This concept comes from a special kind of encryption scheme called deniable encryption.

**5.1.4 OWNER MODULE**

Owner module is to upload their files using some access policy. First they get the public key for particular upload file after getting this public key owner request the secret key for particular upload file. Using that secret key owner upload their file.

**5.1.5 USER MODULE**

This module is used to help the client to search the file using the file id and file name .If the file id and name is incorrect means we do not get the file, otherwise server ask the public key and get the encryption file. If you want the decryption file means user have the secret key.

**5.2 SOFTWARE DESCRIPTION**

## 5.2.1 JAVA TECHNOLOGY

Java technology is both a programming language and a platform.

### 5.2.2 THE JAVA PROGRAMMING LANGUAGE

### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* Simple
  + - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



**fig 5.1: Working of a Java Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



**fig 5.2: Java - Platform Independent**

### 5.2.3 THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

**The Java platform has two components:**

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

We’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



**fig 5.3: Execution of a Java Program**

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are **applets** and **applications**. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications.

Instead of working in browsers, though, Servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeans, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBC)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more.

The following figure depicts what is included in the Java 2 SDK.



## fig 5.4: Features of Java 2 SDK

## How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaProduct Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

### 5.2.4 ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**5.2.5 JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

**5.2.6 MYSQL DESCRIPTION**

## 5.2.6.1 What is Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching, and replicating the data it holds.

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

So now days we use relational database management systems (RDBMS) to store and manager huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

A **Relational Data Base Management System (RDBMS)** is a software that:

* Enables you to implement a database with tables, columns, and indexes.
* Guarantees the Referential Integrity between rows of various tables.
* Updates the indexes automatically.
* Interprets an SQL query and combines information from various tables.

## 5.2.6.2 RDBMS TERMINOLOGY:

Before we proceed to explain MySQL database system, lets revise few definitions related to database.

* **Database:** A database is a collection of tables, with related data.
* **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.
* **Column:** One column (data element) contains data of one and the same kind, for example the column postcode.
* **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
* **Redundancy:** Storing data twice, redundantly to make the system faster.
* **Primary Key:** A primary key is unique. A key value can not occur twice in one table. With a key you can find at most one row.
* **Foreign Key:** A foreign key is the linking pin between two tables.
* **Compound Key:** A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
* **Index:** An index in a database resembles an index at the back of a book.
* **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

## 5.2.6.3 MySQL DATABASE:

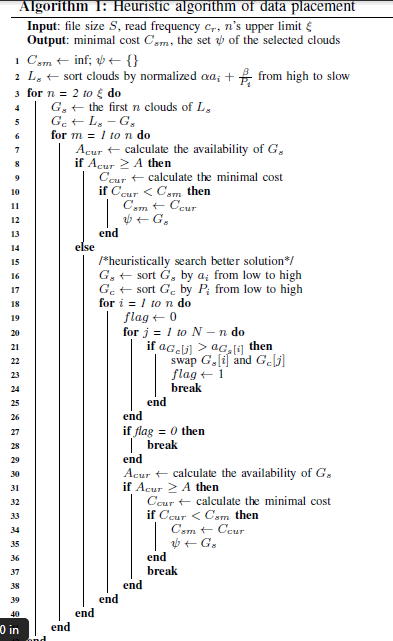
MySQL is a fast, easy-to-use RDBMS used being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons.

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**5.3 ALGORITHM DESCRIPTION**

The key idea of this heuristic algorithm can be described as follows:

We first assign each cloud a value which is calculated based on four factors (i.e., availability, storage, bandwidth, and operation prices) to indicate the preference of a cloud. We choose the most preferred n clouds, and then heuristically exchange the cloud in the preferred set with the cloud in the complementary set to search better solution. This is similar to the idea of Kernighan-Lin heuristic algorithm , which is applied to effectively partition graphs to minimize the sum of the costs on all edges cut. The preference of a cloud is impacted by the four factors, and they have different weights. The availability is the higher the better, and the price is the lower the better.



**CHAPTER 6**

**TESTING**

**6.1 SYSTEM TESTING**

**6.1.1 INTRODUCTION**

After finishing the development of any computer based system the next complicated time consuming process is system testing. During the time of testing only the development company can know that, how far the user requirements have been met out, and so on.

Following are the some of the testing methods applied to this effective project:

# 6.1.2 SOURCE CODE TESTING:

This examines the logic of the system. If we are getting the output that is required by the user, then we can say that the logic is perfect.

**6.1.3 SPECIFICATION TESTING:**

We can set with, what program should do and how it should perform under various condition. This testing is a comparative study of evolution of system performance and system requirements.

**6.1.4 MODULE LEVEL TESTING:**

In this the error will be found at each individual module, it encourages the programmer to find and rectify the errors without affecting the other modules.

**6.1.5 UNIT TESTING:**

Unit testing focuses on verifying the effort on the smallest unit of software-module. The local data structure is examined to ensure that the date stored temporarily maintains its integrity during all steps in the algorithm’s execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

**6.1.6 INTEGRATION TESTING:**

Data can be tested across an interface. One module can have an inadvertent, adverse effect on the other. **Integration testing** is a systematic technique for constructing a program structure while conducting tests to uncover errors associated with interring.

**6.1.7 VALIDATION TESTING:**

It begins after the integration testing is successfully assembled. Validation succeeds when the software functions in a manner that can be reasonably accepted by the client. In this the majority of the validation is done during the data entry operation where there is a maximum possibility of entering wrong data. Other validation will be performed in all process where correct details and data should be entered to get the required results.

**6.1.8 RECOVERY TESTING:**

**Recovery Testing** is a system that forces the software to fail in variety of ways and verifies that the recovery is properly performed. If recovery is automatic, re-initialization, and data recovery are each evaluated for correctness.

**6.1.9 SECURITY TESTING:**

Security testing attempts to verify that protection mechanism built into system will in fact protect it from improper penetration. The tester may attempt to acquire password through external clerical means, may attack the system with custom software design to break down any defences to others, and may purposely cause errors.

**6.1.10 PERFORMANCE TESTING:**

Performance Testing is used to test runtime performance of software within the context of an integrated system. Performance test are often coupled with stress testing and require both software instrumentation.

**6.1.11 BLACKBOX TESTING:**

**Black-box testing** focuses on functional requirement of software. It enables to derive est. of input conditions that will fully exercise all functional requirements for a program.

Black box testing attempts to find error in the following category:

* Incorrect or missing function
* Interface errors
* Errors in data structures or external database access and performance errors.

**6.1.12 OUTPUT TESTING:**

After performing the validation testing, the next step is output testing of the proposed system since no system would be termed as useful until it does produce the required output in the specified format. **Output format** is considered in two ways, the **screen format** and the **printer format.**

**6.1.13 USER ACCEPTANCE TESTING:**

User Acceptance Testing is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at the time of developing and making changes whenever required.

**6.2 TEST CASES**

**6.2.1 LOGIN CREDENTIALS CHECK**

**Input:** Username and Password

If user provides correct credential, then redirect to home screen. If not then redirect to login screens.

**Output:** Home screen redirect

**6.2.2 FILE UPLOAD**

**Input:** File upload to multi server

If user uploads file to multi-server then it stores the file in the configured server path.

**Output:** File upload successfully.

**6.2.3 FILE DOWNLOAD**

**Input:** File download from multi server

If user provides correct secret key, then it gets downloaded from the server after decrypted.

**Output:** Providers secret key with permission.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

Cloud services are experiencing rapid development and the services based on multi-cloud also become prevailing. One of the most concerns, when moving services into clouds, is capital expenditure. So, in this paper, we design a novel storage scheme CHARM, which guides customers to distribute data among clouds cost-effectively. CHARM makes fine-grained decisions about which storage mode to use and which clouds to place data in. The evaluation proves the efficiency of CHARM.

**APPENDIX 1**

**SAMPLE CODE**

**SOURCE CODE**

/\*

\* Title: CloudSim Toolkit

\* Description: CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation

\* of Clouds

\* Licence: GPL - http://www.gnu.org/copyleft/gpl.html

\*

\* Copyright (c) 2009, The University of Melbourne, Australia

\*/

package examples.org.cloudbus.cloudsim.examples;

import com.app.Database.DatabaseFile;

import java.sql.ResultSet;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.LinkedList;

import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;

import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;

import org.cloudbus.cloudsim.Datacenter;

import org.cloudbus.cloudsim.DatacenterBroker;

import org.cloudbus.cloudsim.DatacenterCharacteristics;

import org.cloudbus.cloudsim.Host;

import org.cloudbus.cloudsim.Log;

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.VmSchedulerTimeShared;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.core.SimEntity;

import org.cloudbus.cloudsim.core.SimEvent;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

/\*\*

\* An example showing how to create simulation entities

\* (a DatacenterBroker in this example) in run-time using

\* a globar manager entity (GlobalBroker).

\*/

public class CloudSimTest {

public CloudSimTest()

{

}

/\*\* The cloudlet list. \*/

private static List<Cloudlet> cloudletList;

/\*\* The vmList. \*/

private static List<Vm> vmList;

private static List<Vm> createVM(int userId, int vms, int idShift,

String vmm, String size1, String ram1, String mips1, String bw1, String pesNumber1) {

//Creates a container to store VMs. This list is passed to the broker later

LinkedList<Vm> list = new LinkedList<Vm>();

//VM Parameters

long size = Long.parseLong(size1); //image size (MB)

int ram = Integer.parseInt(ram1); //vm memory (MB)

int mips = Integer.parseInt(mips1);

long bw = Long.parseLong(bw1);

int pesNumber = Integer.parseInt(pesNumber1); //number of cpus

//create VMs

Vm[] vm = new Vm[vms];

for(int i=0;i<vms;i++){

vm[i] = new Vm(idShift + i, userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());

list.add(vm[i]);

}

return list;

}

private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift,

String name, String len, String size, String out, String pesnumber){

// Creates a container to store Cloudlets

LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

//cloudlet parameters

long length = Long.parseLong(len);

long fileSize = Long.parseLong(size);

long outputSize = Long.parseLong(out);

int pesNumber = Integer.parseInt(pesnumber);

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet[] cloudlet = new Cloudlet[cloudlets];

for(int i=0;i<cloudlets;i++){

cloudlet[i] = new Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

// setting the owner of these Cloudlets

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

}

return list;

}

private static Datacenter createDatacenter(String name){

// Here are the steps needed to create a PowerDatacenter:

// 1. We need to create a list to store one or more

// Machines

List<Host> hostList = new ArrayList<Host>();

// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should

// create a list to store these PEs before creating

// a Machine.

List<Pe> peList1 = new ArrayList<Pe>();

int mips = 1000;

// 3. Create PEs and add these into the list.

//for a quad-core machine, a list of 4 PEs is required:

peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS Rating

peList1.add(new Pe(1, new PeProvisionerSimple(mips)));

peList1.add(new Pe(2, new PeProvisionerSimple(mips)));

peList1.add(new Pe(3, new PeProvisionerSimple(mips)));

//Another list, for a dual-core machine

List<Pe> peList2 = new ArrayList<Pe>();

peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

peList2.add(new Pe(1, new PeProvisionerSimple(mips)));

//4. Create Hosts with its id and list of PEs and add them to the list of machines

int hostId=0;

int ram = 16384; //host memory (MB)

long storage = 1000000; //host storage

int bw = 10000;

hostList.add(

new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList1,

new VmSchedulerTimeShared(peList1)

)

); // This is our first machine

hostId++;

hostList.add(

new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList2,

new VmSchedulerTimeShared(peList2)

)

); // Second 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 = 3.0; // the cost of using processing in this resource

double costPerMem = 0.05; // the cost of using memory in this resource

double costPerStorage = 0.1; // the cost of using storage in this resource

double costPerBw = 0.1; // the cost of using bw in this resource

LinkedList<Storage> storageList = new LinkedList<Storage>(); //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;

}

//We strongly encourage users to develop their own broker policies, to submit vms and cloudlets according

//to the specific rules of the simulated scenario

private static DatacenterBroker createBroker(String name){

DatacenterBroker broker = null;

try {

broker = new DatacenterBroker(name);

} catch (Exception e) {

e.printStackTrace();

return null;

}

return broker;

}

/\*\*

\* Prints the Cloudlet objects

\* @param list list of Cloudlets

\*/

private static void printCloudletList(List<Cloudlet> list) {

int size = list.size();

Cloudlet cloudlet;

String indent = " ";

Log.printLine();

Log.printLine("========== OUTPUT ==========");

Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +

"Data center ID" + indent + "VM ID" + indent + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

DecimalFormat dft = new DecimalFormat("###.##");

for (int i = 0; i < size; i++) {

cloudlet = list.get(i);

Log.print(indent + cloudlet.getCloudletId() + indent + indent);

if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){

Log.print("SUCCESS");

Log.printLine( indent + indent + cloudlet.getResourceId() + indent + indent + indent + cloudlet.getVmId() +

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

indent + indent + dft.format(cloudlet.getExecStartTime())+ indent + indent + indent + dft.format(cloudlet.getFinishTime()));

}

}

}

public void createCloudSim()

{

Log.printLine("Starting CloudSim...");

DatabaseFile objDatabaseFile = new DatabaseFile();

try

{

// First step: Initialize the CloudSim package. It should be called

// before creating any entities.

String sql = " Select count(\*) as val "

+ " FROM consumerdetails ";

ResultSet rs = objDatabaseFile.codeselect(sql);

int num\_user = 0; // number of grid users

while(rs.next())

{

num\_user = rs.getInt("val");

}

Calendar calendar = Calendar.getInstance();

boolean trace\_flag = false; // mean trace events

// Initialize the CloudSim library

CloudSim.init(num\_user, calendar, trace\_flag);

GlobalBroker globalBroker = new GlobalBroker("GlobalBroker");

// Second step: Create Datacenters

//Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation

String sql1 = " SELECT DataCenterId, DataCenterName "

+ " FROM datacenterdetails ";

ResultSet rs5 = objDatabaseFile.codeselect(sql1);

while(rs5.next())

{

@SuppressWarnings("unused")

Datacenter datacenter\_0 = createDatacenter(rs5.getString("DataCenterName"));

}

String sql2 = " SELECT name "

+ " FROM brokerdetails ";

ResultSet rs2 = objDatabaseFile.codeselect(sql2);

DatacenterBroker broker = null;

while(rs2.next())

{

//Third step: Create Broker

broker = createBroker(rs2.getString(1));

int brokerId = broker.getId();

String sql3 = " SELECT VM\_Id, VM\_Name, VM\_Image\_Size, VM\_Ram, VM\_mips, VM\_bw, VM\_pesNumber "

+ " FROM vmdetails ";

ResultSet rs3 = objDatabaseFile.codeselect(sql3);

while(rs3.next())

{

//Fourth step: Create VMs and Cloudlets and send them to broker

vmList = createVM(

brokerId,

rs3.getInt("VM\_Id"),

rs3.getInt("VM\_Id")-1,

rs3.getString("VM\_Name"),

rs3.getString("VM\_Image\_Size"),

rs3.getString("VM\_Ram"),

rs3.getString("VM\_mips"),

rs3.getString("VM\_bw"),

rs3.getString("VM\_pesNumber")

); //creating 5 vms

}

String sql6 = " SELECT Cloudlets\_Id, Cloudlets\_Name, Cloudlets\_Length, Cloudlets\_FileSize, Cloudlets\_OutputSize, Cloudlets\_pesNumber "

+ " FROM cloudlets ";

ResultSet rs6 = objDatabaseFile.codeselect(sql6);

while(rs6.next())

{

cloudletList = createCloudlet(

brokerId,

rs6.getInt("Cloudlets\_Id"),

rs6.getInt("Cloudlets\_Id")-2,

rs6.getString("Cloudlets\_Name"),

rs6.getString("Cloudlets\_Length"),

rs6.getString("Cloudlets\_FileSize"),

rs6.getString("Cloudlets\_OutputSize"),

rs6.getString("Cloudlets\_pesNumber")

); // creating 10 cloudlets

}

broker.submitVmList(vmList);

broker.submitCloudletList(cloudletList);

}

// Fifth step: Starts the simulation

CloudSim.startSimulation();

// Final step: Print results when simulation is over

List<Cloudlet> newList = broker.getCloudletReceivedList();

newList.addAll(globalBroker.getBroker().getCloudletReceivedList());

CloudSim.stopSimulation();

printCloudletList(newList);

Log.printLine("CloudSim finished!");

}

catch (Exception e)

{

e.printStackTrace();

Log.printLine("The simulation has been terminated due to an unexpected error");

}

Log.printLine("------------------------------------------------------------------------------------------------------------------------------------------------------------------------------");

}

public static class GlobalBroker extends SimEntity {

private static final int CREATE\_BROKER = 0;

private List<Vm> vmList;

private List<Cloudlet> cloudletList;

private DatacenterBroker broker;

public GlobalBroker(String name) {

super(name);

}

@Override

public void processEvent(SimEvent ev) {

switch (ev.getTag()) {

case CREATE\_BROKER:

setBroker(createBroker(super.getName()+"\_"));

try

{

String sql3 = " SELECT VM\_Id, VM\_Name, VM\_Image\_Size, VM\_Ram, VM\_mips, VM\_bw, VM\_pesNumber "

+ " FROM vmdetails ";

DatabaseFile objDatabaseFile = new DatabaseFile();

ResultSet rs4 = objDatabaseFile.codeselect(sql3);

while(rs4.next())

{

//Create VMs and Cloudlets and send them to broker

setVmList(createVM( getBroker().getId(),

rs4.getInt("VM\_Id"),

rs4.getInt("VM\_Id")-1,

rs4.getString("VM\_Name"),

rs4.getString("VM\_Image\_Size"),

rs4.getString("VM\_Ram"),

rs4.getString("VM\_mips"),

rs4.getString("VM\_bw"),

rs4.getString("VM\_pesNumber")

)

); //creating 5 vms

}

String sql6 = " SELECT Cloudlets\_Id, Cloudlets\_Name, Cloudlets\_Length, Cloudlets\_FileSize, Cloudlets\_OutputSize, Cloudlets\_pesNumber "

+ " FROM cloudlets ";

ResultSet rs6 = objDatabaseFile.codeselect(sql6);

while(rs6.next())

{

setCloudletList(createCloudlet(

getBroker().getId(),

rs6.getInt("Cloudlets\_Id"),

rs6.getInt("Cloudlets\_Id")-2,

rs6.getString("Cloudlets\_Name"),

rs6.getString("Cloudlets\_Length"),

rs6.getString("Cloudlets\_FileSize"),

rs6.getString("Cloudlets\_OutputSize"),

rs6.getString("Cloudlets\_pesNumber")

)

); // creating 10 cloudlets

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

broker.submitVmList(getVmList());

broker.submitCloudletList(getCloudletList());

CloudSim.resumeSimulation();

break;

default:

Log.printLine(getName() + ": unknown event type");

break;

}

}

@Override

public void startEntity() {

Log.printLine(super.getName()+" is starting...");

schedule(getId(), 200, CREATE\_BROKER);

}

@Override

public void shutdownEntity() {

}

public List<Vm> getVmList() {

return vmList;

}

protected void setVmList(List<Vm> vmList) {

this.vmList = vmList;

}

public List<Cloudlet> getCloudletList() {

return cloudletList;

}

protected void setCloudletList(List<Cloudlet> cloudletList) {

this.cloudletList = cloudletList;

}

public DatacenterBroker getBroker() {

return broker;

}

protected void setBroker(DatacenterBroker broker) {

this.broker = broker;

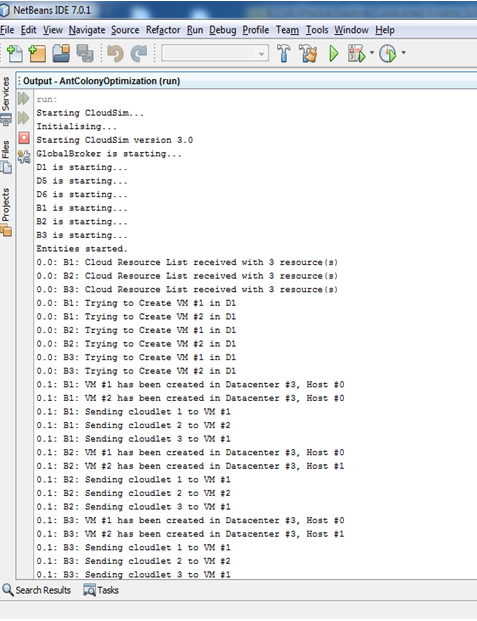
}

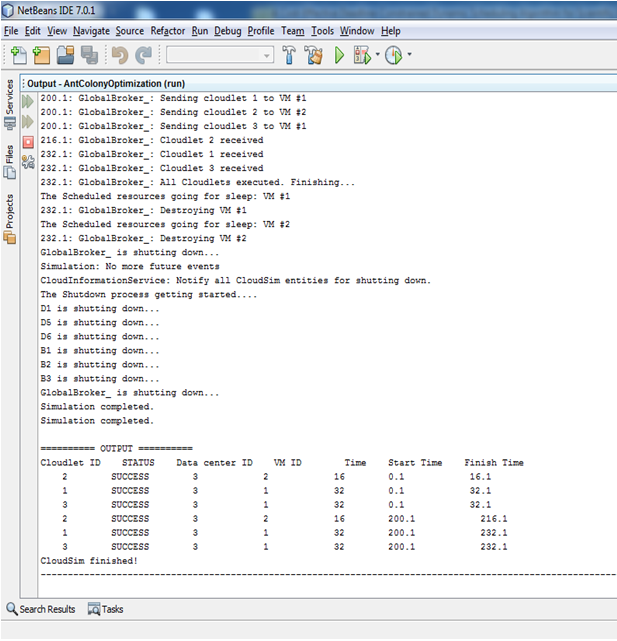
}

}

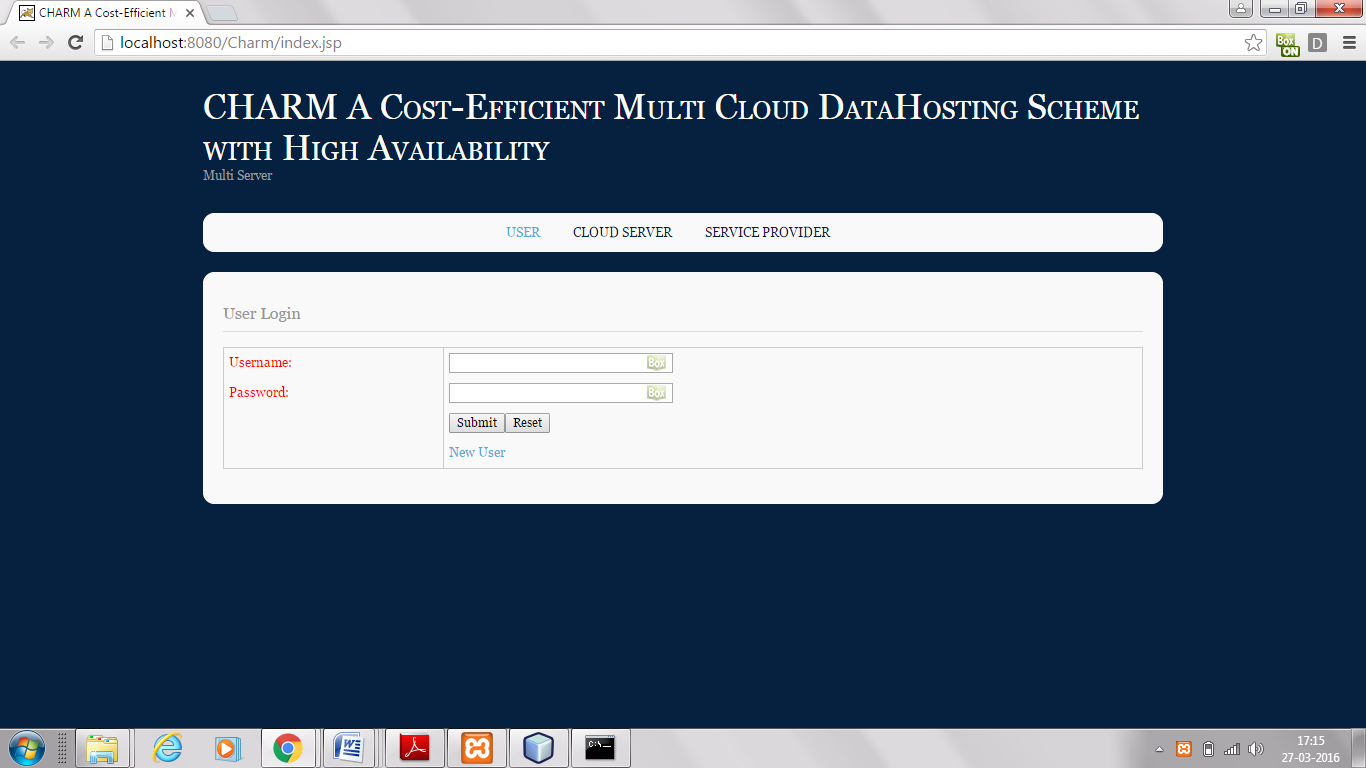
**APPENDIX 2**

**SCREEN SHOTS**

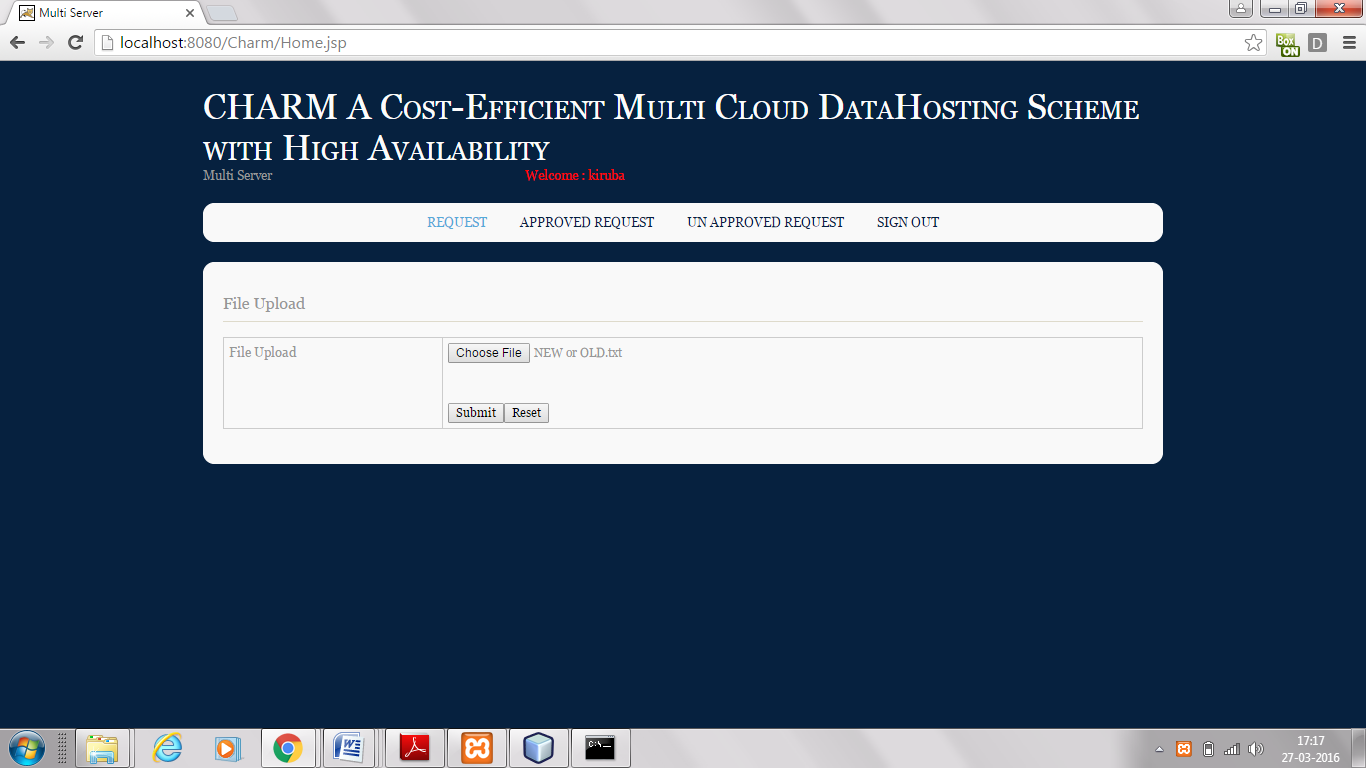
****

****

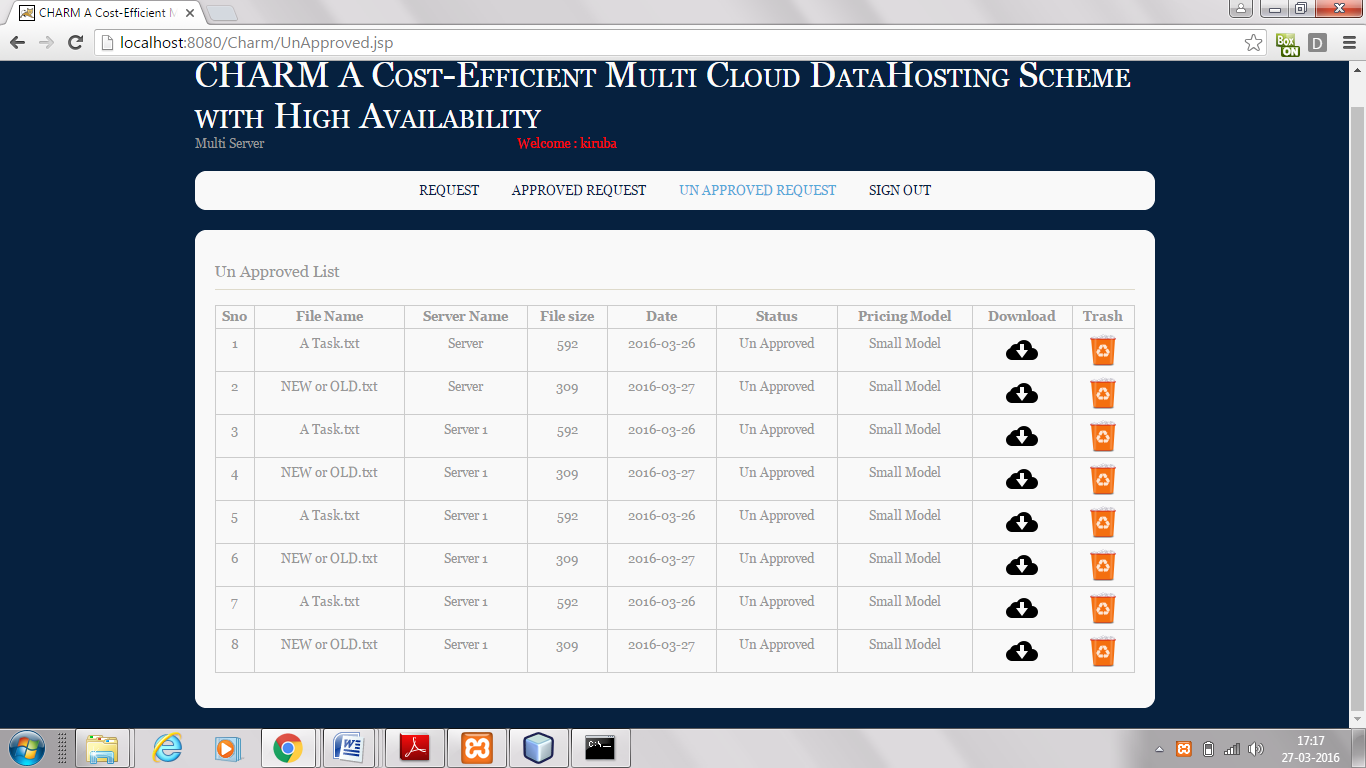
**USER LOGIN PAGE:**

****

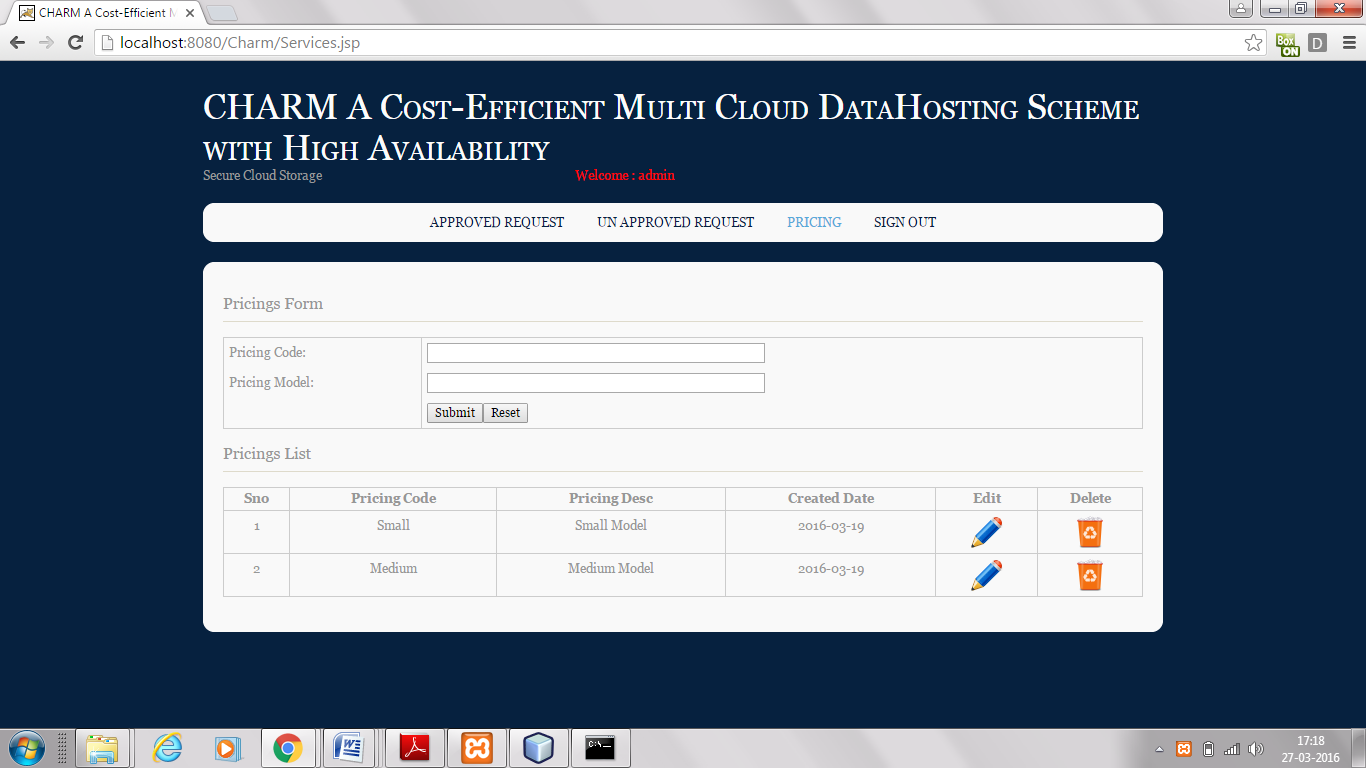
User logs in with username, password and uploads a file for example "NEW or OLD.txt". click Submit to upload.



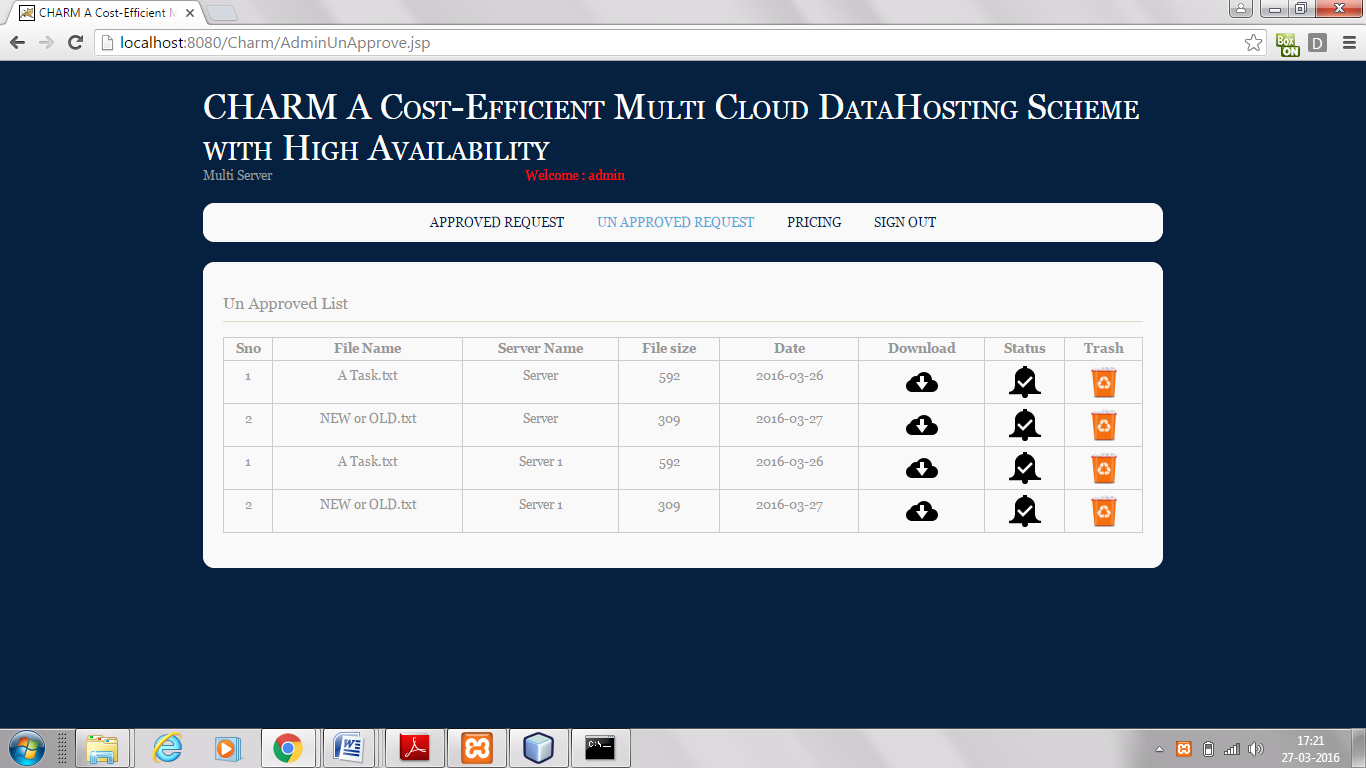
The Uploaded file is still in Unapproved Request.



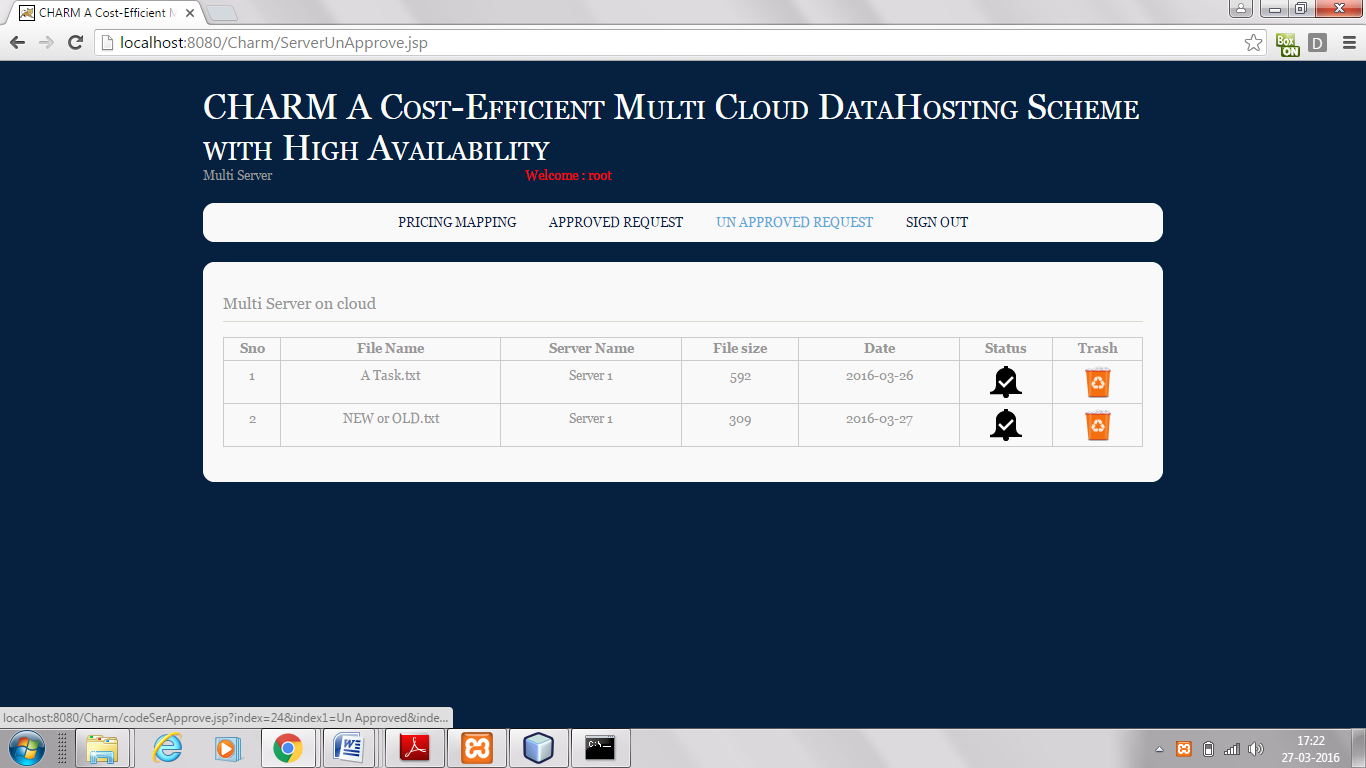
In the Service Provider there are pricing policies which classifies the file based on that strategy. Pricing policy may be small, medium or large.



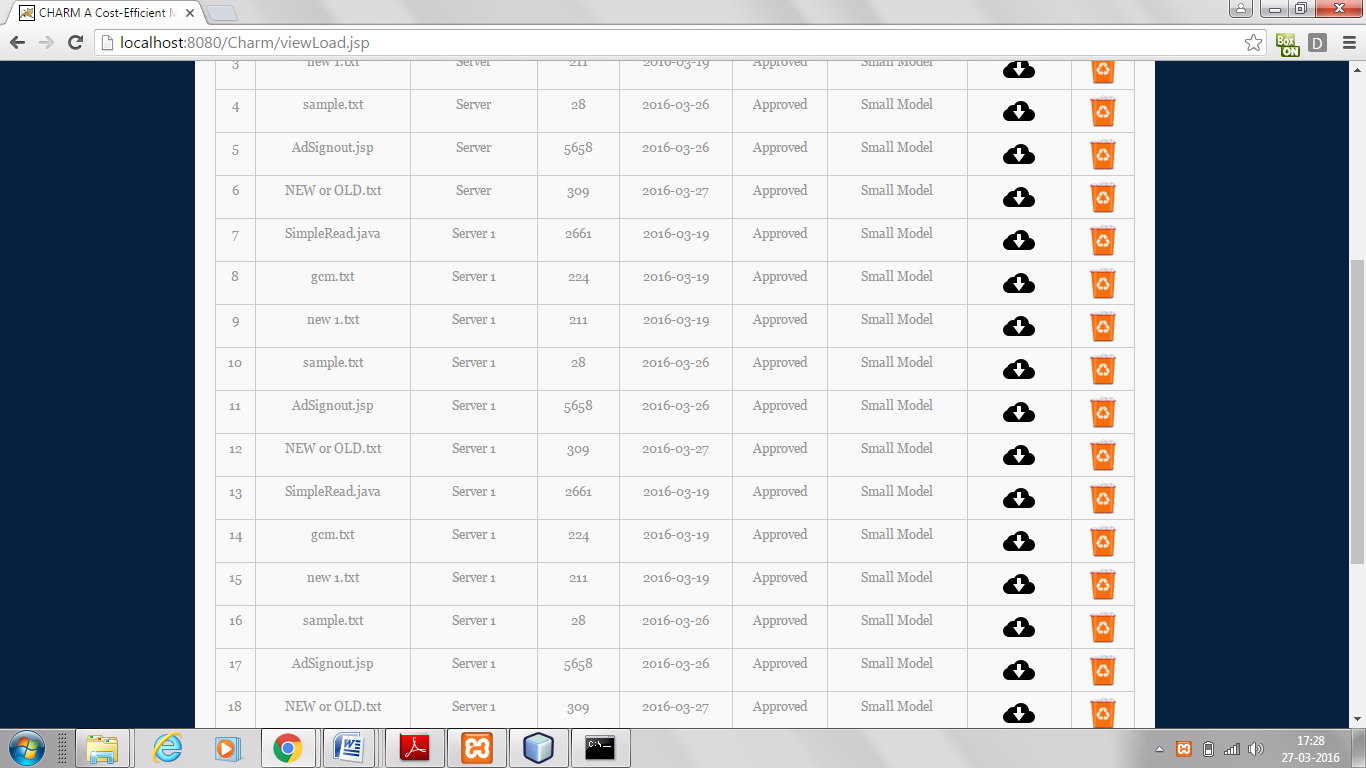
In the Unapproved Request the uploaded file exists. As the file is approved in the cloud server it gets automatically updated in the service provider.



In the cloud server the unapproved list contains the file uploaded by the user "NEW or OLD.txt". The status of the file must be chosen so that the file is approved by the server.



As the server approves the file it is sent to the approved list of the server. The server can also unapprove the file. The user will get the file in the approved list such that the pricing model is efficient enough. The user can download the file whenever needed from the list such that the hosting scheme is effective with minimal monetary cost. If the file is not needed by the user they can delete it by putting it into the trash.

  
There are 2 servers in this method representing replication and multi-cloud concept to avoid data loss and to maximize data reliability. Also to avoid vendors lock in risk opt different cloud like Google Cloud Storage, Amazon S3, Windows Azure, etc.,

**REFERENCES**

[1] Aliyun OSS (Open Storage Service). [Online]. Available: http://

www.aliyun.com/product/oss, 2014.

[2] Gartner: Top 10 cloud storage providers. [Online]. Available:

http://www.networkworld.com/news/2013/010313-gartnercloud-

storage-265459.html?page=1, 2013.

[3] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, and

Z.-L. Zhang, “Towards network-level efficiency for cloud storage

services,” in Proc. ACM SIGCOMM Internet Meas. Conf., 2014,

pp. 115–128.

[4] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Z.-L. Zhang,

and Y. Dai, “Efficient batched synchronization in dropbox-like

cloud storage services,” in Proc.ACM/IFIP/USENIX 14th Int. Middleware

Conf., 2013, pp. 307–327.

[5] C. M. M. Erin Allen, “Library of congress and duracloud launch

pilot program using cloud technologies to test perpetual access to

digital content,” Library Cong., News Releases. [Online]. Available:

http://www.loc.gov/today/pr/2009/09-140.html, 2009.

[6] A. Li, X. Yang, S. Kandula, and M. Zhang, “CloudCmp: Comparing

public cloud providers,” in Proc. ACM SIGCOMM Internet

Meas. Conf., 2010, pp. 1–14.

[7] Windows Azure Pricing Updates. [Online]. Available: http://

azure.microsoft.com/en-us/updates/azure-pricing-updates/,

2014.

[8] Google Cloud Platform Pricing Updates. [Online]. Available:

http://googlecloudplatform.blogspot.com/2014/03/googlecloud-

platform-live-blending-iaas-and-paas-moores-law-for-thecloud.

html, 2014.

[9] It’s Official, the Nirvanix Cloud Storage Service is Shutting Down.

[Online]. Available: http://techcrunch.com/2013/09/27/its-official-

the-nirvanix-cloud-storage-service-is-shutting-down/, 2013.

[10] Shutting Down Ubuntu One File Services. [Online]. Available:

http://blog.canonical.com/2014/04/02/shutting-down-ubuntuone-

file-services/, 2014.

[11] Nirvanix Provides Cautionary Tale for Cloud Storage. [Online].

Available: http://www.forbes.com/sites/tomcoughlin/2013/09/

30/nirvanix-provides-cautionary-tail-for-cloud-storage/, 2013.

[12] Google Outages Damage Cloud Credibility. [Online]. Available:

https://www.networkworld.com/news/2009/092409-googleoutages-

damage-cloud.html, 2009.

[13] Rackspace to issue as much as $3.5M in customer credits after outage.

[Online]. Available: http://www.networkworld.com/news/

2009/070609-rackspace-outage.html, 2009.

[14] Summary of the amazon EC2 and amazon RDS service disruption

in the US east region. [Online]. Available: http://aws.amazon.

com/cn/message/65648/, 2011.

[15] A. Bessani, M. Correia, B. Quaresma, F. Andr\_e, and P. Sousa,

“DepSky: Dependable and secure storage in a cloud-of-clouds,”

in Proc. 6th Conf. Comput. Syst., 2013, pp. 31–46.

[16] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, “RACS: A

case for cloud storage diversity,” in Proc. 1st ACM Symp. Cloud

Comput., 2010, pp. 229–240.

[17] DuraCloud. [Online]. Available: http://www.duracloud.org/,

2014.

[18] Cloud Foundry. [Online]. Available: http://www.cloudfoundry.

org/, 2014.