**CLOUD STORAGE MONITORING SYSTEM**

**ANALYZING THROUGH FILE ACCESS PATTERN**

**ABSTRACT**

Cloud computing is an important technology on current demanding business requirements and it has been emerged as unavoidable technology. The usage of IaaS Service storage for Cloud Computing is being expanding exponential every year. The cloud storages are used by the cloud user due to its economy compared with other storage methods. The replications of files helps user for easy access with high availability which reduces the overall access time of the files, but

at the same time it occupies more storage space and result in high storage cost. The cloud user holds multiple times of the storage than what he is actually needed. It is a dire need of system to find unwanted files in the cloud and also optimize the storage space by evaluating through file access frequency.

This paper propose Cloud Storage Monitoring (CSM) system, which monitor the IaaS storage usage and analyze the file access patterns by various parameters to identify the frequency of

access, size, future access prediction, replication of files in the cloud storage. This allocates a ranking for each file which also predicts future access pattern. This generates a recommendation

dashboard to the user who can decide on the operations such as reorganize, delete or archive the files and eliminate duplicate files in the cloud storage which can increase the space for future use. We implement this project using java technology with deduplication concept. The ranking algorithm technique applied on frequency distribution shows that increase in the storage space upto 10.91% higher than the normal system. It also helps to forecast towards future files usability prediction and prevents the duplicate entries.

**INTRODUCTION**

The data replication services of cloud storage duplicate the files in real time to increase the availability of the files which in turn increase the hardware cost. The data replication service consists of data replication, file replication, cloning infrastructure and remote storage replication.

The cloud storage replication service determine of redundancy which is invaluable on main storage when backup system fails. As the result, replication is used to reach highest availability

at high cost. It is degrading the performance of the service when the cost benefits accrued from the replication. This also increase delay in request and response transaction in cloud environment.

The predictive auto-scaling technique forecast future storage workload of the cloud service and adjusts the cloud storage capacity in order to meet the future needs. The system generates the recommendation dashboard to forecast future files usability and it also eliminate the duplicate entries.

**SYSTEM ANALYSIS**

**EXISTING SYSTEM**

Nowadays most organizations understand the benefits of migrating data to a cloud storage service but at the same time cloud services also having its own risks and drawbacks. In future

cloud storage services will replace the storage network in the data center, mostly due to high sensitive transactional applications, data-intensive, low-response time, and deals with critical data.

Most of use cases are related to organizations and companies having substantial on-premise storage requirements related to cloud storage from various vendors in a Public/Private/Hybrid

model deployment. The organization is making difficult on enforcing cloud storage data management policies and best practices on storage optimization features.

Security on public cloud is not more secure than in-house storage; Most of IT managers aren't comfortable when dealing with sensitive data on public environment. The sensitive data has

been shared to cloud provider which is having multi-tenancy infrastructure which is accessed by public.

**DISADVANTAGES**

* Lot of duplicate data may occupy cloud space due to which customers have to pay for large cloud space.
* Public cloud does not provide much security for data, hence it can be easily hacked.
* Older files which are not in use from long time may occupy much storage space.

**PROPOSED SYSTEM**

A prediction and ranking based system is proposed to handle the de-duplication in cloud storage with the following design objectives.

• Identify the frequency on access pattern

• Provide prediction on file access

• Identify the duplication of files on cloud storage

• Build storage efficient system.

• Increase efficiency of the system.

• Improve search experience

• Block duplication of files in future

The proposed research work, CSM system rank the files based on their popularity and the frequency of access. The system generates ranking dashboard which helps to optimize the storage space and availability. The CMS system reduces the storage space by de-duplication and increase the availability by having the files ready for access.

**ADVANTAGES**

* Prevents duplication of files.
* Files will be displayed based on access pattern ie. Most accesses file will beon top of list.
* Rank is generated for files based on access pattern of file.
* Files are secured by CSM using encryption and compression technique while uploading file.
* Last access date of file is displayed to customer so that unwanted files can be deleted to free storage space.

**SYSTEM REQUIREMENTS SPECIFICATION**

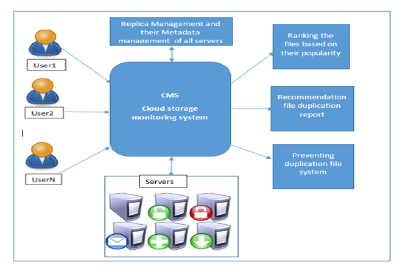
**Hardware requirements:**

* Processor : Any Processor above 1.8 GHz.
* Ram : 2 GB.
* Hard Disk : 20 Gb.
* Input device : Standard Keyboard and Mouse.
* Output device : VGA and High Resolution Monitor

**Software requirements:**

* Operating System : Windows Family.
* Language : Java
* Front End : Java AWT/Swings
* Tools : JDK 8, Netbeans IDE 8.0, Heidisql
* Back End : Mysql 5.0 server

**SYSTEM ARCHITECTURE**



**LITERATURE SURVEY**

# AN IMPROVED DYNAMIC DATA REPLICA SELECTION AND PLACEMENT IN CLOUD

**AUTHORS:** A. Rajalakshmi, D. Vijayakumar, K. G. Srinivasagan

**ABSTRACT**

Cloud computing platform is getting more and more attentions as a new trend of data management. Data replication has been widely used to speed up data access in cloud. Replica selection and placement are the major issues in replication. In this paper we propose an approach for dynamic data replication in cloud. A replica management system allows users to create, and manage replicas and update the replicas if the original datas are modified. The proposed work concentrates on designing an algorithm for suitable optimal replica selection and placement to increase availability of data in the cloud. The method consists of two main phases file application and replication operation. The first phase contains the replica location and creation by using catalog and index. In second phase is used to find whether there is enough space in the destination to store the requested file or not. Replication aims to increase availability of resources, minimum access cost, shared bandwidth consumption and delay time by replicating data. The proposed systems developed under the Eucalyptus cloud environment. The results of proposed replica selection algorithm achieve better accessibility compared with other methods.

# AN AUTONOMIC PREDICTION SUITE FOR CLOUD RESOURCE PROVISIONING

**AUTHORS:** Ali Yadavar Nikravesh, Samuel A. Ajila & Chung-Horng Lung

**ABSTRACT**

One of the challenges of cloud computing is effective resource management due to its auto-scaling feature. Prediction techniques have been proposed for cloud computing to improve cloud resource management. This paper proposes an autonomic prediction suite to improve the prediction accuracy of the auto-scaling system in the cloud computing environment. Towards this end, this paper proposes that the *prediction accuracy of the predictive auto-scaling systems will increase if an appropriate time-series prediction algorithm based on the incoming workload pattern is selected.* To test the proposition, a comprehensive theoretical investigation is provided on different risk minimization principles and their effects on the accuracy of the time-series prediction techniques in the cloud environment. In addition, experiments are conducted to empirically validate the theoretical assessment of the hypothesis. Based on the theoretical and the experimental results, this paper designs a self-adaptive prediction suite. The proposed suite can automatically choose the most suitable prediction algorithm based on the incoming workload pattern.

# CLOUD STORAGE MONITORING SYSTEM ANALYZING THROUGH FILE ACCESS PATTERN

**AUTHORS:** A Augustus Devarajan; T SudalaiMuthu

**ABSTRACT**

Cloud computing is an important technology on current demanding business requirements and it has been emerged as unavoidable technology. The usage of IaaS Service storage for Cloud Computing is being expanding exponential every year. The cloud storages are used by the cloud user due to its economy compared with other storage methods. The replications of files helps user for easy access with high availability which reduces the overall access time of the files, but at the same time it occupies more storage space and result in high storage cost. The cloud user holds multiple times of the storage than what he is actually needed. It is a dire need of system to find unwanted files in the cloud and also optimize the storage space by evaluating through file access frequency.This paper propose Cloud Storage Monitoring (CSM) system, which monitor the IaaS storage usage and analyze the file access patterns by various parameters to identify the frequency of access, size, future access prediction, replication of files in the cloud storage. This allocates a ranking for each file which also predicts future access pattern. This generates a recommendation dashboard to the user who can decide on the operations such as reorganize, delete or archive the files and eliminate duplicate files in the cloud storage which can increase the space for future use. This system is experimented in the CloudSim environment and validate through multiple simulations testing, by using comparison techniques related to file attributes, delta version-hashing, Data de-duplication. The ranking algorithm technique applied on frequency distribution shows that increase in the storage space upto 10.91% higher than the normal system. It also helps to forecast towards future files usability prediction and prevents the duplicate entries.

1. **ATOM: Efficient Tracking, Monitoring, and Orchestration of Cloud Resources**

**AUTHORS:** Min Du, and Feifei Li

**ABSTRACT**

The emergence of Infrastructure as a Service framework brings new opportunities, which also accompanies with new challenges in auto scaling, resource allocation, and security. A fundamental challenge underpinning these problems is the continuous tracking and monitoring of resource usage in the system. In this paper, we present ATOM, an efficient and effective framework to automatically track, monitor, and orchestrate resource usage in an Infrastructure as a Service (IaaS) system that is widely used in cloud infrastructure. We use novel tracking method to continuously track important system usage metrics with low overhead, and develop a Principal Component Analysis (PCA) based approach to continuously monitor and automatically find anomalies based on the approximated tracking results. We show how to dynamically set the tracking threshold based on the detection results, and further, how to adjust tracking algorithm to ensure its optimality under dynamic workloads. Lastly, when potential anomalies are identified, we use introspection tools to perform memory forensics on VMs guided by analyzed results from tracking and monitoring to identify malicious behavior inside a VM. We demonstrate the extensibility of ATOM through virtual machine (VM) clustering. The performance of our framework is evaluated in an open source IaaS system.

1. **A SELF-CONFIGURABLE GEO-REPLICATED CLOUD STORAGE SYSTEM**

**AUTHORS:** Masoud Saeida Ardekani; Douglas B. Terry

**ABSTRACT**

Reconfiguring a cloud storage system can improve its overall service. Tuba is a geo-replicated key-value store that automatically reconfigures its set of replicas while respecting application-defined constraints so that it adapts to changes in clients’ locations or request rates. New replicas may be added, existing replicas moved, replicas upgraded from secondary to primary, and the update propagation between replicas adjusted. Tuba extends a commercial cloudbased service, Microsoft Azure Storage, with broad consistency choices (as in Bayou), consistency-based SLAs (as in Pileus), and a novel replication configuration service. Compared with a system that is statically configured, our evaluation shows that Tuba increases the reads that return strongly consistent data by 63%.

**SOFTWARE DESCRIPTION**

**Java:**

Java was conceived by James Gosling, Patrick Naughton, Chris Wrath, Ed Frank, and Mike Sheridan at Sun Micro system. It is an platform independent programming language that extends it’s features wide over the network.Java2 version introduces an new component called “Swing” – is a set of classes that provides more power & flexible components than are possible with AWT.

- It’s a light weight package, as they are not implemented by platform-specific code.

-Related classes are contained in javax.swing and its sub packages, such as javax.swing.tree.

-Components explained in the Swing have more capabilities than those of AWT

## What Is Java?

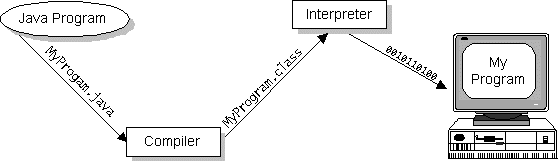
Java is two things: a programming language and a platform.

The Java Programming Language

Java is a high-level programming language that is all of the following:

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

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java byte codes--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how this works.



Java byte codes can be considered as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. The Java program can be compiled 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. For example, the same Java program can run on Windows NT, Solaris, and Macintosh.

### The Java Platform

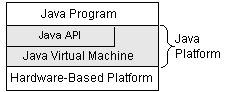
A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

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

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 (packages) of related components.

The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.



As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code withoutthreatening portability.

## What Can Java Do?

Probably the most well-known Java programs are Java applets. An applet is a Java program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, Java is not just for writing cute, entertaining applets for the World Wide Web ("Web"). Java is a general-purpose, high-level programming language and a powerful software platform. Using the generous Java API, we can write many types of programs.

The most common types of programs are probably applets and applications, where a Java application is a standalone program that runs directly on the Java platform.

How does the Java API support all of these kinds of programs?

With packages of software components that provide a wide range of functionality. The core API is the API included in every full implementation of the Java platform. The core API 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 Java applets.
* **Networking**: URLs, TCP and UDP sockets, and IP 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/private key management, access control, and certificates.
* **Software components**: Known as JavaBeans, can plug into existing component architectures such as Microsoft's OLE/COM/Active-X architecture, OpenDoc, and Netscape's Live Connect.
* **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.
* Java not only has a core API, but also standard extensions. The standard extensions define APIs for 3D, servers, collaboration, telephony, speech, animation, and more.

## How Will Java Change My Life?

Java is likely to make your programs better and requires less effort than other languages. We believe that Java will help you do the following:

* **Get started quickly**: Although Java 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 Java can be four times smaller than the same program in C++.
* **Write better code**: The Java language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Java's 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 faster**: 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 with Java and Java is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by following the purity tips mentioned throughout this book and avoiding the use of libraries written in other languages.
* **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 Java feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

We explore the java.net package, which provides support for networking. Its creators have called Java “programming for the Internet.” These networking classes encapsulate the “socket” paradigm pioneered in the Berkeley Software Distribution (BSD) from the University of California at Berkeley.

**Networking Basics**

Ken Thompson and Dennis Ritchie developed UNIX in concert with the C language at Bell Telephone Laboratories, Murray Hill, New Jersey, in 1969. In 1978, Bill Joy was leading a project at Cal Berkeley to add many new features to UNIX, such as virtual memory and full-screen display capabilities. By early 1984, just as Bill was leaving to found Sun Microsystems, he shipped 4.2BSD, commonly known as Berkeley UNIX.4.2BSD came with a fast file system, reliable signals, interprocess communication, and, most important, networking. The networking support first found in 4.2 eventually became the de facto standard for the Internet. Berkeley’s implementation of TCP/IP remains the primary standard for communications with the Internet. The socket paradigm for inter process and network communication has also been widely adopted outside of Berkeley.

**Socket Overview**

A *network socket* is a lot like an electrical socket. Various plugs around the network have a standard way of delivering their payload. Anything that understands the standard protocol can “plug in” to the socket and communicate.

*Internet protocol (IP)* is a low-level routing protocol that breaks data into small packets and sends them to an address across a network, which does not guarantee to deliver said packets to the destination.

*Transmission Control Protocol* (TCP) is a higher-level protocol that manages to reliably transmit data. A third protocol, *User Datagram Protocol (UDP)*, sits next to TCP and can be used directly to support fast, connectionless, unreliable transport of packets.

Client/Server

A *server* is anything that has some resource that can be shared. There are *compute servers*, which provide computing power; *print servers*, which manage a collection of printers; *disk servers*, which provide networked disk space; and *web servers*, which store web pages. A *client* is simply any other entity that wants to gain access to a particular server.

In Berkeley sockets, the notion of a socket allows as single computer to serve many different clients at once, as well as serving many different types of information. This feat is managed by the introduction of a *port*, which is a numbered socket on a particular machine. A server process is said to “listen” to a port until a client connects to it. A server is allowed to accept multiple clients connected to the same port number, although each session is unique. To mange multiple client connections, a server process must be multithreaded or have some other means of multiplexing the simultaneous I/O.

**Reserved Sockets**

Once connected, a higher-level protocol ensues, which is dependent on which port you are using. TCP/IP reserves the lower, 1,024 ports for specific protocols. Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for Netnews-and the list goes on. It is up to each protocol to determine how a client should interact with the port.

**Java and the Net**

Java supports TCP/IP both by extending the already established stream I/O interface. Java supports both the TCP and UDP protocol families. TCP is used for reliable stream-based I/O across the network. UDP supports a simpler, hence faster, point-to-point datagram-oriented model.

**InetAddress**

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. We interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. As of Java 2, version 1.4, InetAddress can handle both IPv4 and IPv6 addresses.

Factory Methods

The InetAddress class has no visible constructors. To create an InetAddress object, we use one of the available factory methods. *Factory methods* are merely a convention whereby static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Three commonly used InetAddress factory methods are shown here.

static InetAddress getLocalHost( ) throws UnknownHostException

static InetAddress getByName(String hostName) throws UnknowsHostException

static InetAddress[ ] getAllByName(String hostName)

throws UnknownHostException

The getLocalHost( ) method simply returns the InetAddress object that represents the local host. The getByName( ) method returns an InetAddress for a host name passed to it. If these methods are unable to resolve the host name, they throw an UnknownHostException.

On the internet, it is common for a single name to be used to represent several machines. In the world of web servers, this is one way to provide some degree of scaling. The getAllByName ( ) factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can’t resolve the name to at least one address. Java 2, version 1.4 also includes the factory method getByAddress( ), which takes an IP address and returns an InetAddress object. Either an IPv4 or an IPv6 address can be used.

**Instance Methods**

The InetAddress class also has several other methods, which can be used on the objects returned by the methods just discussed. Here are some of the most commonly used.

Boolean equals (Object other)- Returns true if this object has the same Internet address as other.

byte[ ] getAddress( )- Returns a byte array that represents the object’s Internet address in network byte order.

String getHostAddress( )- Returns a string that represents the host address associated with the InetAddress object.

String getHostName( )- Returns a string that represents the host name associated with the InetAddress object.

boolean isMulticastAddress( )- Returns true if this Internet address is a multicast address. Otherwise, it returns false.

String toString( )- Returns a string that lists the host name and the IP address for conveneince.

Internet addresses are looked up in a series of hierarchically cached servers. That means that your local computer might know a particular name-to-IP-address mapping autocratically, such as for itself and nearby servers. For other names, it may ask a local DNS server for IP address information. If that server doesn’t have a particular address, it can go to a remote site and ask for it. This can continue all the way up to the root server, called InterNIC (internic.net).

**TCP/IP Client Sockets**

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java’s I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The ServerSocket class is designed to be a “listener,” which waits for clients to connect before doing anything. The Socket class is designed to connect to server sockets and initiate protocol exchanges.

The creation of a Socket object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

Socket(String *hostName*, int *port*) Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or anIOException.

Socket(InetAddress *ipAddress*, int *port*) Creates a socket using a preexisting InetAddress object and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

InetAddress getInetAddress( )- Returns the InetAddress associated with the Socket object.

int getPort( ) Returns the remote port to which this Socket object is connected.

int getLocalPort( ) Returns the local port to which this Socket object is connected.

Once the Socket object has been created, it can also be examined to gain access to the input and output streams associated with it. Each of these methods can throw an IOException if the sockets have been invalidated by a loss of connection on the Net.

InputStream getInputStream( )Returns the InputStream associated with the invoking socket.

OutputStream getOutputStream( ) Returns the OutputStream associated with the invoking socket.

**TCP/IP Server Sockets**

Java has a different socket class that must be used for creating server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different form normal Sockets.

When we create a ServerSocket, it will register itself with the system as having an interest in client connections. The constructors for ServerSocket reflect the port number that we wish to accept connection on and, optionally, how long we want the queue for said port to be. The queue length tells the system how many client connection it can leave pending before it should simply refuse connections. The default is 50. The constructors might throw an IOException under adverse conditions. Here are the constructors:

ServerSocket(int port) Creates server socket on the specified port with a queue length of 50.

Serversocket(int *port*, int *maxQueue*)-Creates a server socket on the specified *port* with a maximum queue length of *maxQueue*.

ServerSocket(int *port*, int *maxQueue*, InetAddress *localAddress*)-Creates a server socket on the specified *port* with a maximum queue length of *maxQueue*. On a multihomed host, *localAddress* specifies the IP address to which this socket binds.

ServerSocket has a method called accept( ), which is a blocking call that will wait for a client to initiate communications, and then return with a normal Socket that is then used for communication with the client.

**JAVA SWING**

**Swing** is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform.

**Swing overview**

The [Internet Foundation Classes](http://en.wikipedia.org/wiki/Internet_Foundation_Classes) (IFC) were a [graphics library](http://en.wikipedia.org/wiki/Graphics_library) for Java originally developed by [Netscape Communications Corporation](http://en.wikipedia.org/wiki/Netscape_Communications_Corporation) and first released on [December 16](http://en.wikipedia.org/wiki/December_16), [1996](http://en.wikipedia.org/wiki/1996). On [April 2](http://en.wikipedia.org/wiki/April_2), [1997](http://en.wikipedia.org/wiki/1997), [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) and [Netscape Communications Corporation](http://en.wikipedia.org/wiki/Netscape_Communications_Corporation) announced their intention to incorporate IFC with other technologies to form the [Java Foundation Classes](http://en.wikipedia.org/wiki/Java_Foundation_Classes).

Swing introduced a mechanism that allowed the [look and feel](http://en.wikipedia.org/wiki/Look_and_feel) of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a [pluggable look and feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. This feature also makes it easy to make an application written in Swing look very different from native programs if desired.

Originally distributed as a separately downloadable library, Swing has been included as part of the [Java Standard Edition](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition) since release 1.2. The Swing classes and components are contained in the [javax.swing](http://java.sun.com/javase/6/docs/api/javax/swing/package-summary.html) [package](http://en.wikipedia.org/wiki/Java_package) hierarchy.

**Main New Features**

* Lightweight. Not built on native window-system windows.
* Much bigger set of built-in controls. Trees, image buttons, tabbed panes, sliders, toolbars, color choosers, tables, text areas to display HTML or RTF, etc.
* Much more customizable. Can change border, text alignment, or add image to almost any control. Can customize how minor features are drawn. Can separate internal representation from visual appearance.
* "Pluggable" look and feel. Can change look and feel at runtime, or design own look and feel.
* Many miscellaneous new features. Double-buffering built in, tool tips, dockable tool bars, keyboard accelerators, custom cursors, etc.

**Difference between swing and AWT**

* Swings are faster than awt.
* Awt uses heavyweight components for user interfaces.
* Swing uses lightweight components
* Implementations of AWT components use native code which may vary from one machine to another.
* Swing components are pure java code

## Architecture

Swing is a platform-independent, *Model-View-Controller* GUI framework for Java. It follows a single-threaded programming model, and possesses the following traits:

### Foundations

#### Platform independence

Swing is platform independent both in terms of its expression (Java) and its implementation (non-native universal rendering of widgets).

#### Extensibility

Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations. In general, Swing users can extend the framework by extending existing (framework) classes and/or providing alternative implementations of core components.

#### Component-oriented

Swing is a component-based framework. The distinction between objects and components is a fairly subtle point: concisely, a component is a well-behaved object with a known/specified characteristic pattern of behaviour. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to the component.) Specifically, Swing components are Java Beans components, compliant with the Java Beans Component Architecture specifications.

#### Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use these property (settings) to determine the appropriate renderers to use in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

#### Configurable

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at runtime to fundamental changes in its settings. For example, a Swing-based application can change its look and feel at runtime. Further, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

**Lightweight UI**

Swing's configurability is a result of a choice not to use the native host OS's GUI controls for displaying itself. Swing "paints" its controls programmatically through the use of Java 2D APIs, rather than calling into a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics APIs.

However, at its core every Swing component relies on an AWT container, since (Swing's) JComponent extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS agnostic) semantics over the underlying (OS specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to component.paint(), which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

This transposition and decoupling is not merely visual, and extends to Swing's management and application of its own OS-independent semantics for events fired within its component containment hierarchies. Generally speaking, the Swing Architecture delegates the task of mapping the various flavors of OS GUI semantics onto a simple, but generalized, pattern to the AWT container. Building on that generalized platform, it establishes its own rich and complex GUI semantics in the form of the JComponent model. A review of the source of Container java and JComponent java classes is recommended for further insights into the nature of the interface between Swing's lightweight components and AWT's heavyweight widgets.

#### Loosely-Coupled/MVC

The Swing library makes heavy use of the Model/View/Controller software design pattern, which conceptually decouples the data being viewed from the user interface controls through which it is viewed. Because of this, most Swing components have associated *models* (which are specified in terms of Java interfaces), and the programmer can use various default implementations or provide their own. The framework provides default implementations of model interfaces for all of its concrete components.[[1]](http://en.wikipedia.org/wiki/Swing_(Java)#cite_note-0#cite_note-0)

Typically, Swing component model objects are responsible for providing a concise interface defining events fired, and accessible properties for the (conceptual) data model for use by the associated JComponent. Given that the overall MVC pattern is a loosely-coupled collaborative object relationship pattern, the model provides the programmatic means for attaching event listeners to the data model object. Typically, these events are model centric (ex: a "row inserted" event in a table model) and are mapped by the JComponent specialization into a meaningful event for the GUI component.

For example, the JTable has a model called TableModel that describes an interface for how a table would access tabular data. A default implementation of this operates on a two-dimensional array.

The view component of a Swing JComponent is the object used to graphically "represent" the conceptual GUI control. A distinction of Swing, as a GUI framework, is in its reliance on programmatically-rendered GUI controls (as opposed to the use of the native host OS's GUI controls). Prior to Java 6 Update 10, this distinction was a source of complications when mixing AWT controls, which use native controls, with Swing controls in a GUI (see Mixing AWT and Swing components).

Finally, in terms of visual composition and management, Swing favors relative layouts (which specify the positional relationships between components) as opposed to absolute layouts (which specify the exact location and size of components). This bias towards "fluid"' visual ordering is due to its origins in the applet operating environment that framed the design and development of the original Java GUI toolkit. (Conceptually, this view of the layout management is quite similar to that which informs the rendering of HTML content in browsers, and addresses the same set of concerns that motivated the former.)

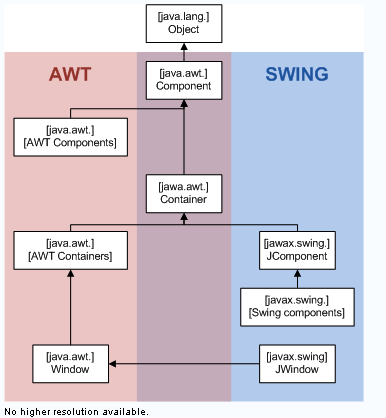
### Look and feel

Swing allows one to specialize the look and feel of widgets, by modifying the default (via runtime parameters), deriving from an existing one, by creating one from scratch, or, beginning with **J2SE 5.0**, by using the skinnable synth Look and Feel (see Synth Look and Feel), which is configured with an XML property file. The look and feel can be changed at runtime, and early demonstrations of Swing frequently provided a way to do this.

### Relationship to AWT

**AWT and Swing class hierarchy**

Since early versions of Java, a portion of the Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system.



By contrast, Swing components are often described as *lightweight* because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as *heavyweight components*.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing (JApplet, JDialog, JFrame, and JWindow) extend an AWT top-level container. However, the use of both lightweight and heavyweight components within the same window is generally discouraged due to Z-order incompatibilities.

The core rendering functionality used by Swing to draw its lightweight components is provided by Java 2D, another part of JFC.

### Relationship to SWT

The Standard Widget Toolkit (SWT) is a competing toolkit originally developed by IBM and now maintained by the Eclipse community. SWT's implementation has more in common with the heavyweight components of AWT. This confers benefits such as more accurate fidelity with the underlying native windowing toolkit, at the cost of an increased exposure to the native platform in the programming model.

The advent of SWT has given rise to a great deal of division among Java desktop developers, with many strongly favoring either SWT or Swing. Sun's development on Swing continues to focus on platform look and feel (PLAF) fidelity with each platform's windowing toolkit in the approaching Java SE 7 release (as of December 2006[[update]](http://en.wikipedia.org/w/index.php?title=Swing_(Java)&action=edit)).

There has been significant debate and speculation about the performance of SWT versus Swing; some hinted that SWT's heavy dependence on JNI would make it slower when the GUI component and Java need to communicate data, but faster at rendering when the data model has been loaded into the GUI. A fairly thorough benchmarks show Swing performing better on head to head direct comparisons..

SWT serves the Windows platform very well but is considered by some to be less effective as a technology for cross-platform development. By using the high-level features of each native windowing toolkit, SWT returns to the issues seen in the mid 90's (with toolkits like zApp, Zinc, XVT and IBM/Smalltalk) where toolkits attempted to mask differences in focus behaviour, event triggering and graphical layout. Failure to match behavior on each platform can cause subtle but difficult-to-resolve bugs that impact user interaction and the appearance of the GUI.

### DESIGN

**DATA FLOW DIAGRAM**

**LEVEL ZERO**:

STORAGE NODE

DATABASE

CMS

CLIENT

**LEVEL 1**

CLIENT

DB

STORAGE NODE

CMS

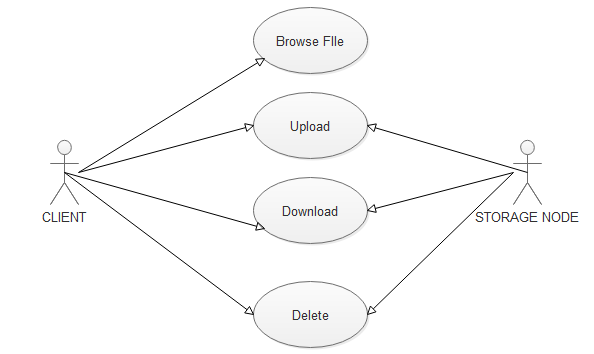
**LEVEL 2**

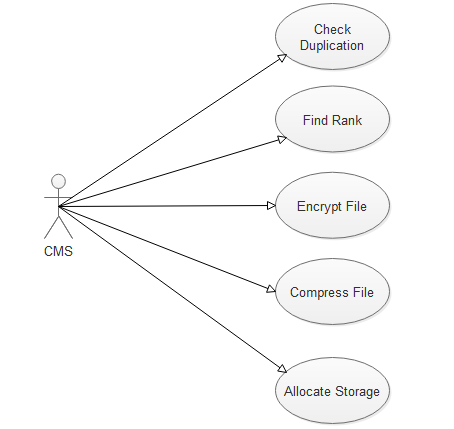
CLIENT

STORAGE NODE

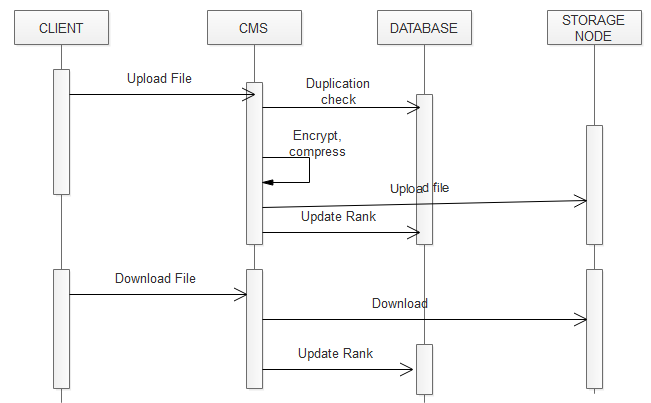
CMS

**USE CASE**



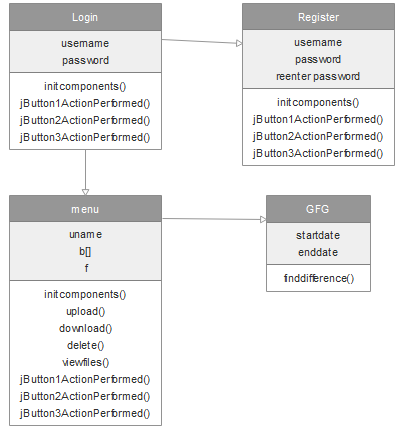


**SEQUENCE DIAGRAM**

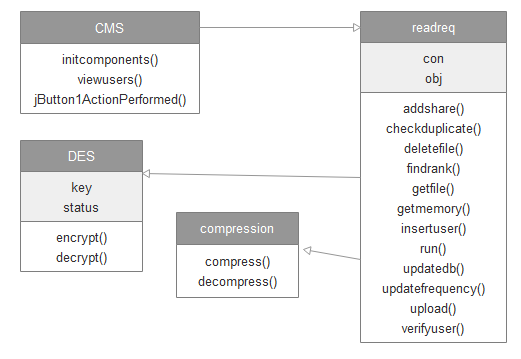
****

**CLASS DIAGRAM**

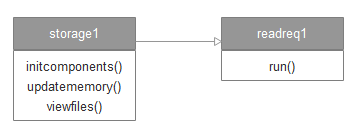
**Client**



**CMS**



**Storage Node**



**IMPLEMENTATION**

Our project implementation consists of three major modules ie. Client, Cloud ManagementS Service (CMS) and Storage Nodes.

The sub modules present in each module are as described below:

**CLIENT**

* **Register**

In this module a new client can enter his username and password and register with CMS.

* **Login**

The registered clients can enter their username and password for login. These details will be verified by CMS and allowed to login on providing valid details.

* **File Upload**

A client can browse a file and upload it to storage node using this module. Upload request is sent to CMS and a response is sent back to client.

* **File Download**

In this module client will get list of files which are uploaded by him. The files are displayed based on rank and also last access details are shown. Client can select a file and download it.

* **Delete**

If client want to delete a file from storage node, he can select the file from list of available files and delete it.

**CMS**

* **Deduplication**

When a file upload request is sent from client to CMS, this module will check for duplication of file based on extension and size of file. If any file which is already available in storage nodes matches with extension and size of file to be uploaded, that file will be matched for duplication. If the file size is less than 100 bytes, whole content will be matched else first and last 50 bytes will be matched. If the file already exist, upload is cancelled and a link will be created to client for downloading existing file.

* **Find Rank**

When a new file is uploaded by client or existing file is downloaded, frequency for that file is updated and rank will be generated for the file based on the frequency of access. Whichever file has more frequency will have 1st rank and viceversa.

* **Update Frequency**

Whenever a client uploads a file or downloads existing file, count will be updated for the file. This count is called as frequency. It is used for finding rank for file.

* **Encryption**

All the files which will be uploaded to storage nodes are encrypted for security purpose. We use DES encryption algorithm for encryption files. It uses 64 bit key.

* **Decryption**

When a client downloads file or a file has to be matched for duplication, that file will be decrypted.

* **Compression**

To reduce the space to be used by file in storage nodes, the files will be compresses and stored. We use zip compression scheme for compressing files.

* **Decompression**
* When a client downloads file or a file has to be matched for duplication, that file will be decompressed and decrypted.
* **Upload**

When a client sends upload request to CMS, the file will be checked for duplication, then encrypted and compressed and finally it is uploaded in storage node. The storage node which has max memory will be selected by CMS for file upload.

* **Download**

When a client sends download request, CMS finds the storage node which conatins file and downloads file from it. Then the file is decompressed and decrypted and sent to client.

* **Delete**

When client sends delete request for a file, CMS checks if the file is used by more than one client. If so, the link for the file which is created for the client will be reased. Hence file will be available for other users. If file is not used by other clients, then the file will be deleted from storage node.

**Storage Node**

* **Upload**

When storage node gets upload request from CMS, it saves the file and calculates available memory and used memory and sends reply back to CMS.

* **Download**

When a download request is sent to storage node, it sends the file content to CMS.

* **Delete**

In this module storage node deletes the file and updates its available memory and used memory.

### SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**Types of Tests**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Integration testing for Server Synchronization:

* Testing the IP Address for to communicate with the other Nodes.
* Check the request is sent from client to CMS.
* Check the request is sent from CMS to storage nodes.
* Check the file is sent from storage node to CMS.
* Check the file is sent from CMS to client.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Testing results are as mentioned in the table below:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MODULE** | **GIVEN INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **RESULT** |
| client | File Upload | Upload request should be sent to CMS | File upload request sent to CMS | OK |
| Client | File Download | File download request should be sent to CMS | File Download request should be sent to CMS and file downloaded | OK |
| CMS | Upload request from client | File shoud be checked for duplication and encrypted, compressed and sent to storage node | File successfully checked for duplication, encrypted, compressed and sent to storage node | OK |
| CMS | Download request from client | Request should be forwarded to storage node | Request successfully sent to storage node | OK |
| Storage node | Upload request | File should be saved and acknowledgement should be sent to CMS | File successfully saved and ACK sent to CMS | OK |
| Storage node | Download request | File content should be sent to CMS | File content successfully sent to CMS | OK |
| Storage node | Delete request | File should be deleted | File successfully deleted | OK |
|  |  |  |  |  |

**SNAPSHOT**

