International Journal of Research
ISSN: 2348-6848 Vol-3, Special Issue-1
National Conference on Advanced Computing Technologies
Held on 21st July 2015 at Hyderabad city organized by Global Research Academy, Hyderabad, Telangana, India.

# Design & Development of an Efficient and Trustworthy Resource Sharing Platform for Collaborative Cloud Computing

**Salavath Ramesh[1]; B. Madhav Rao[2]& Prof.Dr.G.Manoj Someswar[3]**

[1]M.Tech.(CSE) from Narasimha Reddy Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India

[2]M.Tech. (CSE), Assistant Professor, Department of CSE, Narasimha Reddy Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India

[3]B.Tech., M.S.(USA), M.C.A., Ph.D., Principal & Professor, Department Of CSE, Anwar-ul-uloom College of Engineering & Technology, Affiliated to JNTUH, Vikarabad, Telangana, India

**ABSTRACT**

*Advancements in cloud computing are leading to a promising future for collaborative cloud computing (CCC), where globally-scattered distributed cloud resources belonging to different organizations or individuals (i.e., entities) are collectively used in a cooperative manner to provide services. Due to the autonomous features of entities in CCC, the issues of resource management and reputation management must be jointly addressed in order to ensure the successful deployment of CCC. However, these two issues have typically been addressed separately in previous research efforts, and simply combining the two systems generates doubleoverhead. Also, previous resource and reputation management methods are not sufficiently efficient or effective. By providing a single reputation value for each node, the methods cannot reflect the reputation of a node in providing individual types of resources. By always selecting the highest-reputed nodes, the methods fail to exploit node reputation in resource selection to fully and fairly utilize resources in the system and to meet users' diverse QoS demands. We propose a CCC platform, called Harmony, which integrates resource management and reputation management in a harmonious manner. Harmony incorporates three key innovations: integrated multi-faceted resource/reputation management, multi-QoS-oriented resource selection, and price-assisted resource/reputation control. The trace data we collected from an online trading platform implies the importance of multi-faceted reputation and the drawbacks of highest-reputed node selection. Simulations and trace-driven experiments on the real-world Planet Lab test bed show that Harmony outperforms existing resource management and reputation management systems in terms of QoS, efficiency and effectiveness.*

**Keywords:** Collaboratie Cloud Computing; Cloud Orchestration Policy Engine; Peer-to-Peer Networks; Trust Manager; Data Consumer; Validation Testing; Testing Strategy

## INTRODUCTION

Cloud Computing has become a popular computing paradigm, in which cloud providers offer scalable resources over the Internet to customers. Currently, many clouds, such as Amazon's EC2, Google's AppEngine, IBM's Blue- Cloud, and Microsoft's Azure, provide various services (e.g., storage and computing). For example, Amazon (cloud provider) provides Dropbox (cloud customer) the simple storage service (S3) (cloud service). Cloud customers are charged by the actual usage of computing resources, storage, and bandwidth.

The demand for scalable resources in some applications has been increasing very rapidly. For example, Dropbox currently has five million users, three times the number last year. A single cloud may

not be able to provide sufficient resources for an application (especially during a peak time). Also, researchers may need to build a virtual lab environment connecting multiple clouds for petascale supercomputing capabilities or for fully utilizing idle resources. Indeed, most desktop systems are underutilized in most organizations; they are idle around 95 percent of the time. Thus, advancements in cloud computing are inevitably leading to a promising futurefor collaborative cloud computing (CCC), where globally- scattered distributed cloud resources belonging to different organizations or individuals (i.e., entities) are collectively pooled and used in a cooperative manner to provide services.

As this system CCC platform interconnects physical resources to enable resource sharing between clouds, and provides a virtual view of a tremendous amount of resources to customers. This virtual organization is transparent to cloud customers. When a cloud does not have sufficient resources demanded by its customers, it finds and uses the resources in other clouds.

**Importance of Resource and Reputation Management**

CCC operates in a large-scale environment involving thousands or millions of resources across disparate geographically distributed areas, and it is also inherently dynamic as entities may enter or leave the system and resource utilization and availability are continuously changing . This environment makes efficient resource management (resMgt) (i.e., resource location and resource utilization) a non-trivial task. Further, due to the autonomous and individual characteristics of entities in CCC, different nodes provide different quality of service (QoS) in resource provision. A node may provide low QoS because of system problems (e.g., machines break down due to insufficient cooling) or because it is not willing to provide high QoS in order to save costs. Also, nodes may be attacked by viruses and Trojan horse programs.[1] This weakness is revealed in all the cloud platforms built by Google, IBM, and Amazon, and security has been recognized as an important factor in grids (the predecessor of clouds). Thus, resMgt needs reputation management (repMgt) to measure resource provision QoS for guiding resource provider selection. As in eBay and Amazon, a repMgt system computes each node's reputation value based on evaluations from others about its performance in order to provide guidance in selecting trustworthy resources.

To ensure the successful deployment of CCC, the issues of resMgt and repMgt must be jointly addressed for both efficient and trustworthy resource sharing in three tasks:

1. Efficiently locating required trustworthy resources.
2. Choosing resources from the located options.
3. Fully utilizing the resources in the system while avoiding overloading any node.

**Previous Methods and Challenges**

The three tasks must be executed in a distributed manner since centralized methods are not suitable for large-scale CCC. However, though many distributed resMgt and repMgt systems for grids have been proposed previously, and cloud resource orchestration (i.e., resource provision, configuration, utilization and decommission across a distributed set of physical resources in clouds) has been studied in recent years, these two issues have typically been addressed separately. Simply building and combining individual resMgt and repMgt systems in CCC will generate doubled, prohibitively high overhead. Moreover, most previous resMgt and repMgt approaches are not sufficiently efficient or effective in the large-scale and dynamic environment of CCC.[2]

Previous repMgt systems neglect resource heterogeneity by assigning each node one reputation value for providing all of its resources. We claim that node reputation is multi-faceted and should be differentiated across multiple resources (e.g., CPU,

bandwidth, and memory). For example, a person trusts a doctor for giving advice on medical issues but not on financial issues. Similarly, a node that performs well for computing services does not necessarily perform well for storage services. Thus, previous repMgt systems are not effective enough to provide correct guidance for trustworthy individual resource selection. In task (1), RepMgt needs to rely on resMgt for reputation differentiation across multiple resources.

Previous resMgt approaches only assume a single QoS demand of users, such as efficiency or security. Given a number of resource providers (i.e., servers), the efficiency oriented resMgt policy would choose the one with the highest available resource, while the security-oriented repMgt policy would choose the one with the highest reputation. The former may lead to a low service success rate while the atter may overload the node with many resource requests. Thus, uncoordinated deployment of repMgt and resMgt will exhibit contradictory behaviors and significantly affect the effectiveness of both, finally leading to degraded overall performance. The results of the single-QoS-demand assumption and contradictory behaviors pose two challenges.[3]. First, in task (2), how can we jointly consider multiple QoS demands such as reputation, efficiency, and available resources in resource selection? Second, in task (3), how can we enable each node to actively control its reputation and resource supply so that it avoids being overloaded while gaining high reputation and profit.

## Literature Survey

### Declarative Automated Cloud Resource Orchestration

As cloud computing becomes widely deployed, one of the challenges faced involves the ability to orchestrate a highly complex set of subsystems (compute, storage, network resources) that span large geographic areas serving diverse clients. To ease this process, we present COPE (Cloud Orchestration Policy Engine), a distributed platform that allows cloud providers to perform declarative automated cloud resource orchestration. In COPE, cloud providers specify system-wide constraints and goals using COPElog, a declarative policy language geared towards specifying distributed constraint optimizations. [4] COPE takes policy specifications and cloud system states as input and then optimizes compute, storage and network resource allocations within the cloud such that provider operational objectives and customer SLAs can be better met. We describe our proposed integration with a cloud orchestration platform, and present initial evaluation results that demonstrate the viability of COPE using production traces from a large hosting company in the US. We further discuss an orchestration scenario that involves geographically distributed data centers, and concludewith an ongoing status of our work.

### Gossip-based Reputation Aggregation for Unstructured Peer-to-Peer Networks

*Peer-to-Peer* (P2P) reputation systems are needed to evaluate the trustworthiness of participating peers and to combat selfish and malicious peer behaviors. The reputation system collects locally generated peer feedbacks and aggregates them to yield global reputation scores. Development of decentralized reputation system is in great demand for unstructured P2P networks since most P2P applications on the Internet are unstructured. In the absence of fast hashing and searching mechanisms, how to perform efficient reputation aggregation is a major challenge on unstructured P2P computing.[5] We propose a novel reputation aggregation scheme called *GossipTrust*. This system computes global reputation scores of all nodes concurrently. By resorting to a gossip protocol and leveraging the power nodes, GossipTrust is adapted to peer dynamics and robust to disturbance by malicious peers. Simulation experiments demonstrate the system as scalable, accurate, robust and fault-tolerant. These results prove the claimed advantages in low aggregation overhead, storage efficiency, and scoring accuracy in

unstructured P2P networks. With minor modifications, the system is also applicable to structured P2P systems with projected better performance.

**MAAN: A Multi-Attribute Addressable Network for Grid Information Services**

Recent structured Peer-to-Peer (P2P) systems such as Distributed Hash Tables (DHTs) offer scalable key-based lookup for distributed resources. However, they cannot be simply applied to grid information services because grid resources need to be registered and searched using multiple attributes. This paper proposes a Multi-Attribute Addressable Network (MAAN) which extends Chord to support multi-attribute and range queries. MAAN addresses range queries by mapping attribute values to the Chord identifier space via uniform locality preserving hashing. It uses an iterative or single attribute dominated query routing algorithm to resolve multi-attribute based queries. Each node in MAAN only has O(logN) neighbors for N nodes.[6] The number of routing hops to resolve a multi-attribute range query is O(logN +N ×smin), where smin is the minimum range selectivity on all attributes. When smin = ε, it islogarithmic to the number of nodes, which is scalable to a large number of nodes and attributes. We also measured the performance of our MAAN implementation and the experimental results are consistent with our theoretical analysis.

**Trust and Reputation Model in Peer-to-Peer Networks**

It is important to enable peers to represent and update their trust in other peers in open networks for sharing files, and especially services. In this paper, we propose a Bayesian network-based trust model and a method for building reputation based on recommendations in peer-to peer networks. Since trust is multi-faceted, peers need to develop differentiated trust in different aspects of other peers' capability. The peer's needs are different in different situations. Depending on the situation, a peer may need to consider its trust in a specific aspect of another peer's capability or in multiple aspects. Bayesian networks provide a flexible method to present differentiated trust and combine different aspects of trust. The evaluation of the model using a simulation shows that the system where peers communicate their experiences (recommendations) outperforms the system where peers do not share recommendations with each other and that a differentiated trust adds to the performance in terms of percentage of successful interactions.[7]

**Problem Definition**

The problem of the system is to combine the two systems and identified CCC as an example of large-scale distributed systems for the application area of this combined system. This proposes a new system component called "price-assisted resource/reputation control." We also present new trace analysis results to prove the necessity of the proposed algorithms. We further extensively evaluate the performance of the system through many experiments.

**Proposed Method**

By identifying and understanding the interdependencies between resMgt and repMgt, we introduce Harmony, a CCC platform with harmoniously integrated resMgt and repMgt. It can achieve enhanced and joint management of resources and reputation across distributed resources in CCC. Different from the previous resMgt and repMgt methods, Harmony enables a node to locate its desired resources and also find the reputation of the located resources, so that a client can choose resource providers not only by resource availability but also by the provider's reputation of providing the resource. In addition, Harmony can deal with the challenges of large scale and dynamism in the complex environment of CCC. The contributions of this work can be summarized as below:

1. Preliminary study on real trace and experimental results. We analyzed the transaction and feedback rating data we collected from an online trading platform (Zol). We found that some sellers have high QoS in providing some merchandise but offer low QoS in others, and buyers tend to buy merchandise from high-reputed sellers. The findings verify the importance of multi-faceted reputation and the drawback of the highest-reputed node selection policy.
2. Integrated multi-faceted resource/reputation management. Relying on a distributed hash table overlay (DHT), Harmony offers multi-faceted reputation evaluation across multiple resources by indexing the resource information and the reputation of each type of resource to the same directory node. In this way, it enables nodes to simultaneously access the information and reputation of available individual resources.
3. Multi-QoS-oriented resource selection. Unlike previous resMgt approaches that assume a single QoS demand of users, Harmony enables a client to perform resource selection with joint consideration of diverse QoS requirements, such as reputation, efficiency, distance, and price, with different priorities.
4. Price-assisted resource/reputation control. In a resource transaction, a resource requester pays a resource provider (in the form of virtual credits) for its resource. The transactions are conducted in a distributed manner in Harmony. Harmony employs a trading model for resource transactions in resource sharing and leverages the resource price to control each node's resource use and reputation. It enables each node to adaptively adjust its resource price to maximize its profit and maintain a high reputation while avoiding being overloaded, in order to fully and fairly utilize resources in the system.

We have conducted extensive trace-driven experiments with PlanetLab and simulations. Experimental results show the superior performance of Harmony in comparison with previous resMgt and repMgt systems, and the effectiveness of its three components. This work is the first to integrate repMgt with resMgt for multi-faceted node reputation evaluation to provide precise guidance for individual resource selection. In addition to CCC, Harmony can also be applied to other areas such as large-scale distributed systems, grids and P2P systems. The preliminary work of this paper was presented in this system. We introduced an integrated resource and reputation management system in this system and an efficient and locality- aware resource management system in this system. In this paper, we combine the two systems and identified CCC as an example of large-scale distributed systems for the application area of this combined system. We propose a new system component called "price-assisted resource/reputation control." We also present new trace analysis results to prove the necessity of the proposed algorithms. We further extensively evaluate the performance of the system through many experiments.

## System Analysis

### Existing System:

An existing system, the system has developed a framework for generic cloud collaboration allows clients and cloud applications to simultaneously use services from and route data among multiple clouds. This framework supports universal and dynamic collaboration in a multi cloud system. It lets clients simultaneously use services from multiple clouds without prior business agreements among cloud providers, and without adopting common standards and specifications.[8]

### Proposed System:

In this research paper, the system proposes a CCC platform, called Harmony, which integrates resource management and reputation management in a harmonious manner. Harmony incorporates three key innovations: integrated multi-faceted resource/reputation management, multi-QoS-oriented

resource selection, and price-assisted resource/reputation control. The trace data we collected from an online trading platform implies the importance of multi-faceted reputation and the drawbacks of highest-reputed node selection. Simulations and trace-driven experiments on the real-world Planet Lab test bed show that Harmony outperforms existing resource management and reputation management systems in terms of QoS, efficiency and effectiveness.

## SYSTEM STUDY

### 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

- ECONOMIC FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

### 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.[9]

### 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.[10]

### SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. 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.[11] 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.

## IMPLEMENTATION

- **Data Owner**

  In this module, the data owner Registers, Login's the owner by registered details, requesting the resources in cloud like Virtual Machine (VM), Memory, Threshold, and uploads their data in the selective cloud server. For the security purpose the Trust manager encrypts the data file and then store in the Cloud Server.

- **Cloud Server**

  The Data Owner sends a request to Clouds to provide services by assigning the VM, Memory, Threshold values to particular cloud server (Amazon, App Engine, Azure, and Blue Cloud). The cloud service provider

manages multiple clouds to provide data storage service via Trust manager. Trust encrypt their data files and store them in the cloud for sharing with data consumers. To access the shared data files, data consumers download encrypted data files of their interest from the specified cloud and then decrypt them and The Cloud server can attack the files in the cloud Server.[12]

- **Data Integrity**

Data Integrity is very important in database operations in particular and Data warehousing and Business intelligence in general. Because Data Integrity ensured that data is of high quality, correct, consistent and accessible.

- **Trust Manager(Harmony)**

The Trust Manager allows clients and cloud applications () to simultaneously use services from and route data among multiple clouds. This platform supports universal and dynamic collaboration in amulticloud system is called as Collaborative Cloud Computing (CCC) is also known as Harmony, who is trusted to store verification parameters and offer public query services for these parameters.[13] In our system the Trusted Manager, view the user data and uploaded to the cloud. The Trusted Manager will perform the revocation and un revocation of the remote user if he is the attacker or malicious user over the cloud data.

- **Data Consumer(End User)**

In this module, the user has to get Registered to Trust Manager to access the Cloud services and need to Authenticate the user by Logging in by providing the User Name and auto Generated Password by Trust Manager then the Data Consumer can access the data file with the encrypted key, so if User access the file by wrong Key then the user will consider as malicious users and blocked the User.

## SYSTEM DESIGN

### SYSTEM ARCHITECTURE:

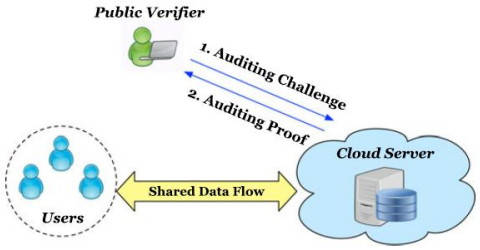


**Figure 1: System Architecture**

### DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into

levels that represent increasing information flow and functional detail.

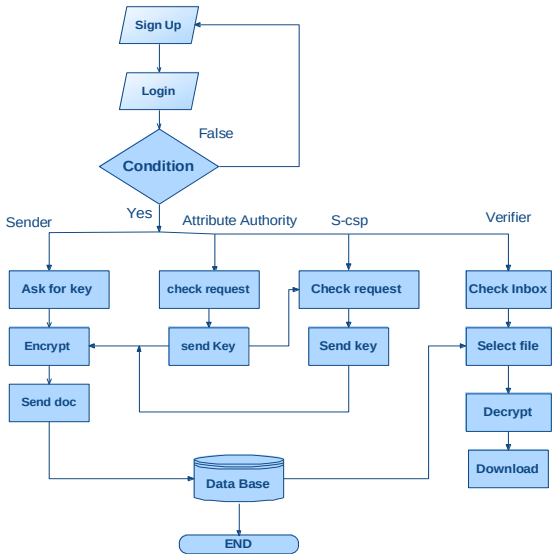


**Figure 2: Data Flow Diagram**

## UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

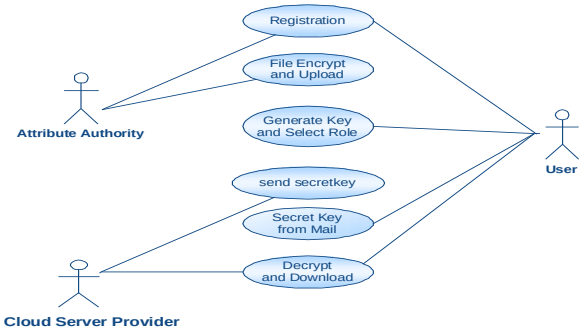


**Figure 3: Use Case Diagram**

## CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

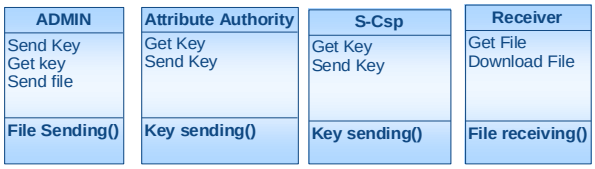


**Figure 4: Class Diagram**

## SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

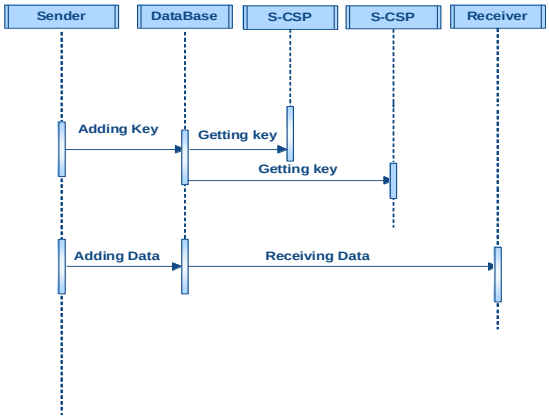


**Figure 5: Sequence Diagram**

## ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

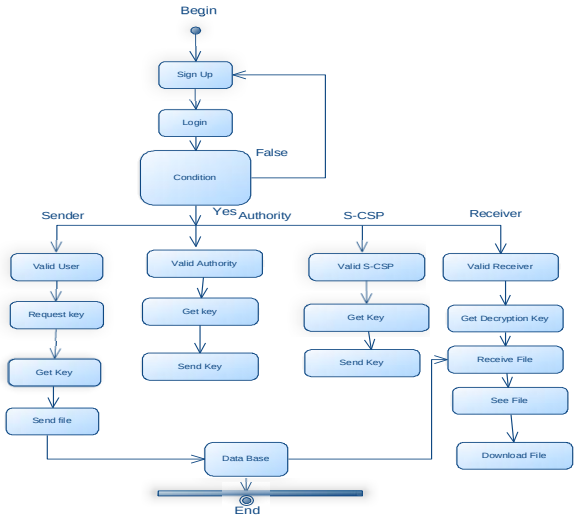


**Figure 6: ActivityDiagram**

## SYSTEM ANALYSIS

## EXISTING SYSTEM:

- Many mechanisms have been proposed to allow not only a data owner itself but also a public verifier to efficiently perform integrity checking without downloading the entire data from the cloud, which is referred to as public auditing . In these mechanisms, data is divided into many small blocks, where each block is independently signed by the owner; and a random combination of all the blocks instead of the whole data is retrieved during integrity checking . A public verifier could be a data user (e.g., researcher) who would like to utilize the owner's data via the cloud or a third-party auditor (TPA) who can provide expert integrity checking services.
- Moving a step forward, Wang et al. designed an advanced auditing mechanism .so that during public auditing on cloud data, the content of private data belonging to a personal user is not

disclosed to any public verifiers. Unfortunately, current public auditing solutions mentioned above only focus on personal data in the cloud .We believe that sharing data among multiple users is perhaps one of the most engaging features that motivates cloud storage. Therefore, it is also necessary to ensure the integrity of shared data in the cloud is correct.

- Existing public auditing mechanisms can actually be extended to verify shared data integrity. However, a new significant privacy issue introduced in the case
shared data with the use of existing mechanisms is the leakage of identity privacy to public verifiers.

**DISADVANTAGES OF EXISTING SYSTEM:**

1. Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information to public verifiers.

2. Protect these confidential information is essential and critical to preserve identity privacy from public verifiers during public auditing.

**PROPOSED SYSTEM:**

- In this research paper, to solve the above privacy issue on shared data, we propose Oruta, a novel privacy-preserving public auditing mechanism.
- More specifically, we utilize ring signatures to construct homomorphic authenticators in Oruta, so that a public verifier is able to verify the integrity of shared data without retrieving the entire data while the identity of the signer on each block in shared data is kept private from the public verifier.
- In addition, we further extend our mechanism to support batch auditing, which can perform multiple auditing tasks simultaneously and improve the efficiency of verification for multiple auditing tasks.
- Meanwhile, Oruta is compatible with random masking, which has been utilized in WWRL and can preserve data privacy from public verifiers. Moreover, we also leverage index hash tables from a previous public auditing solution to support dynamic data. A high-level comparison among Oruta and existing mechanisms is presented.

**ADVANTAGES OF PROPOSED SYSTEM:**

A public verifier is able to correctly verify shared data integrity.

1. A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing.

The ring signatures generated for not only able to preserve identity privacy but also able to support blockless verifiability.

**SYSTEM TESTING**

**TESTING METHODOLOGIES**

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.[14]

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1. Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.[15]

**User Acceptance Testing**

User Acceptance of a system 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 the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text

fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**RESULTS & CONCLUSION**

In this research paper, we propose an integrated resource/reputation management platform, called Harmony, for collaborative cloud computing. Recognizing the interdependencies between resource management and reputation management, Harmony incorporates three innovative components to enhance their mutual interactions for efficient and trustworthy resource sharing among clouds. The integrated resource/reputation management component efficiently and effectively collects and provides information about available resources and reputations of providers for providing the types of resources. The multi-QoS-oriented resource selection component helps requesters choose resource providers that offer the highest QoS measured by the requesters' priority consideration of multiple QoS attributes. The price-assisted resource/reputation control component provides incentives for nodes to offer high QoS in providing resources. Also, it helps providers keep their high reputations and avoid being overloaded while maximizing incomes. The components collaborate to enhance the efficiency and reliability of sharing globally-scattered distributed resources in CCC. Simulations and trace-driven experiments on Planet Lab verify the effectiveness of the different Harmony components and the superior performance of Harmony in comparison to previous resource and reputation management systems. The experimental results also show that Harmony achieves high scalability, balanced load distribution, locality-awareness, and dynamism-resilience in the large-scale and dynamic CCC environment. In our future work, we will investigate the optimal time period for neural network training and load factor calculation. We will investigate the challenges of deploying the Harmony system for the real-world applications which involve cooperation between cloud providers.

**REFERENCES**

[1] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com,2013.

[2] Dropbox, www.dropbox.com, 2013.

[3] P. Suresh Kumar, P. Sateesh Kumar, and S. Ramachandram,"Recent Trust Models In Grid," J. Theoretical and Applied Information Technology, vol. 26, pp. 64-68, 2011.

[4] J. Li, B. Li, Z. Du, and L. Meng, "CloudVO: Building a Secure Virtual Organization for Multiple Clouds Collaboration," Proc. 11th ACIS Int'l Conf. Software Eng. Artificial Intelligence Networking andParallel/Distributed Computing (SNPD), 2010.

[5] C. Liu, B.T. Loo, and Y. Mao, "Declarative Automated Cloud Resource Orchestration," Proc. Second ACM Symp. Cloud Computing (SOCC '11), 2011.

[6] C. Liu, Y. Mao, J.E. Van der Merwe, and M.F. Fernandez, "Cloud Resource Orchestration: A Data Centric Approach," Proc. Conf.Innovative Data Systems Research (CIDR) , 2011.Fig. 13. Effectiveness of price-assisted control algorithm.

[7] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Management," Proc. IEEE Int'l Conf. Dependable, Autonomic and Secure Computing (DASC), 2009.
[8] IBM Red Boo. Fundamentals of Grid Computing, Technical Report REDP-3613-00 2000.

[9] L. Xiong and L. Liu, "Peertrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," IEEE Trans. Knowledge and Data Eng., vol. 16, no. 7, pp. 843-857, July 2004.

[10] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks," Proc. World Wide Web Conf., 2005.

[11] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 4, pp. 460- 473, 2008.

[12] R. Zhou and K. Hwang, "Gossip-Based Reputation Management for Unstructured Peer-to-Peer Networks," IEEE Trans. Knowledge and Data Eng., 2007.

[13] Zol, http://www.zol.com.cn/, 2013. [14] PlanetLab, http://www.planet-lab.org/, 2013.

[15] H. Shen and G. Liu, "Harmony: Integrated Resource and Reputation Management for Large-Scale Distributed Systems," Proc. 20th Int'l Conf. Computer Comm. and Networks (ICCCN), 2011.