**A Holistic Approach to Autonomic Self Healing Cloud Computing Architecture**

**Report 2012**

In the partial fulfilment of Final Year BTech Project

By

Abhishek Bhavsar, Ameya More, Chinmay Kulkarni, Dheeraj Oswal

Under the guidance of

Dr. J. V. Aghav (HoD, Department of Computer Engineering and Information Technology)

# Abstract

Ever since the development of modern computers, there has been a constant effort to meet the exponentially growing demands for a robust computing platform that provides better and centralized storage, huge computing capability and greater accessibility. Clustered Computing, Grid Computing have been some of the offshoots of this process. One of the major developments that has caught the eye almost everyone worldwide is the emergence of Cloud Computing. Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). [[1]](#id.41mghml)Cloud Computing today, offers an array of services encompassing Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (Paas) and Software-as-a-Service (SaaS). Although a lot of organizations have turned to Cloud for meeting their computational and data storage needs, Cloud Computing still isn’t at par with other technologies that exist today. Its relative nascent nature provides with excellent opportunities for development. The project focuses efforts to make a cloud computing environment reliable and robust by proposing an autonomic, self healing architecture. It makes use of the ideal state signature of the system architecture to make decisions and do self healing. We adopt a holistic approach to the problem and aim at proposing an architecture that is general enough to be adopted by a wide range of existing systems. Some of the major challenges include selecting the appropriate mechanism for healing and reducing the overhead thus making healing lightweight and transparent, yet effective.

# Table of Contents

**Abstract**

**Table of Contents**

**List of Figures**

**Introduction**

**1.1. Cloud Computing**

1.1.1. Infrastructure as a Service

1.1.2. Platform as a Service

1.1.3. Software as a Service

**Layered Architecture of a Cloud Computing Environment**

**2.1. Supporting (IT) Infrastructure**

**2.2. Cloud Specific Infrastructure**

2.2.1. Cloud (Web) applications

2.2.2. Cloud Software Environment

**2.3. Service Customer**

**Healing**

**3.1. Healing – An Overview**

**3.2. Faults**

**3.3. Fault Tolerance**

3.3.1. Information redundancy

3.3.2. Time redundancy

3.3.3. Physical redundancy

**3.4. Healing**

**3.5. Existing Solutions**

**Problem Statement**

**Literature Survey**

**5.1. Autonomic computing system with model transfer**

**5.2. Architecture for a Self-Healing Computer System**

**5.3. System and Method for Achieving Autonomic Computing Self Healing Utilizing Meta Level Reflection and Reasoning**

**Proposed Solution**

**The Path Ahead**

**Bibliography**

# List of Figures

Figure 1: Approach for Controlling a Managed Resource using an Autonomic System

Figure 2: Block diagram of an Autonomic System

Figure 3: Embodiment of a Self Healing Processor

Figure 4: Embodiment of an Error Mitigation System

Figure 5: System and method for Handling Errors

Figure 6: Block Diagram for Proposed Architecture 17

# Introduction

The advent of computers kicked off a race for development of a unified computing platform that could provide for a centralized computing facility, large and reliable storage and increased accessibility. This race brought about revolutionary technologies like Grid Computing, Clustered Systems, Distributed System and many others. Cloud Computing is the most recent development in this field and shows a lot of promise.

## Cloud Computing

Cloud computing is a framework that revolves around the concept of providing services such as network storage and computational capabilities. One of the major motivations behind the development of cloud computing was to provide the above mentioned capabilities without the need to physically possess the resources. This is important as it makes available the resources even to individuals that may not possess the strength to set up the infrastructure needed to deliver such a computational environment. Users could thus avail services on subscription through service providers such as Google, Amazon etc.

Services provided by the cloud need not be only of the form of software servers, rather individuals could subscribe to an entire computing infrastructure. We illustrate further, the different services provided by the cloud computing environment.

### Infrastructure as a Service

Infrastructure as a Service aims at outsourcing hardware resources such as computational power, network and storage. This means that an individual can access the hardware resources as his own but it is the service provider who assumes the responsibility of setting up, maintaining and housing it. Some of the major Infrastructures available for subscription are

* Virtual Machines
* Servers
* Storage
* Network

It should be noted that though all these infrastructural facilities are provided to the user as a whole, they might be distributed over various areas, even geographically separated.

### Platform as a Service

Platform as a Service is a service model which provides a solution stack as a service. The service provider provides an individual a set of tools to develop and deploy software. PaaS allows individuals to build multi-tenant applications that can be concurrently accessed by a lot of users. PaaS provides subscription for

* Execution Runtime
* Database
* Web-server
* Development Tools

### Software as a Service

Software as a Service allows the software and its data to be placed on a remote server i.e. the cloud. SaaS allows this software by a subscriber using a remote machine like a desktop terminal via a web browser. Softwares that can be hosted on a cloud are

* Customer Relationship Management
* Management Information Systems
* Human Resource Management
* Enterprise Resource Planning
* Content Management
* Email
* Virtual Desktop

# Layered Architecture of a Cloud Computing Environment

A Cloud Computing Environment can be looked upon from the perspective of a layered architecture as divided into 8 layers each interlinked with the layers above and below it and divided into 3 major parts.

## Supporting (IT) Infrastructure

This layer houses all the actual resources and computing power of the entire system. This layer if of not much importance from the perspective of building autonomic system in a cloud, as it is hardware dependent and a lot of work is already done in this layer.

## Cloud Specific Infrastructure

This is the layer that provides most scope for work and it is the only layer that actually defines the cloud-computing environment.

### Cloud (Web) applications

This layer includes the services and APIs written using programming platforms like Java, PHP, and Silverlight etc. Making changes to this layer is not feasible because it would mean putting additional burden on the programmers to design systems to incorporate autonomic computing and would make existing systems difficult to port. Also the motive behind the project is to provide a transparent mechanism for self-healing and autonomic computing that cannot be provided through this.

### Cloud Software Environment

This is the layer that defines the cloud-computing environment. It makes use of the layers below to provide a unified computing environment in the form of a cloud. This is the layer where the Platform-as-a-Service and Infrastructure-as-a-Service frameworks of the cloud reside.

The cloud software infrastructure layer provides an abstraction level for basic IT resources that are offered as services to higher layers:

* Computational resources (usually VMEs),
* Storage, and
* (Network) communication.

These services can be used individually, as is typically the case with storage services, but they’re often bundled such that servers are delivered with certain network connectivity and (often) access to storage. This bundle, with or without storage, is usually referred to as IaaS.

The cloud software environment layer provides services at the application platform level:

* A development and runtime environment for services and applications written in one or more supported languages;
* Storage services (a database interface rather than file share); and
* Communication infrastructure, such as Microsoft’s Azure service bus.

#### Computational Resources

The cloud-computing environment provides a set of computing resources that can be used to perform work that is desired.

#### Storage

One of the major resources that a cloud provides is storage on demand.

#### Communication

The backbone of the entire cloud-computing environment is the communication network that binds various nodes.

We propose that the solution be planned in regard to managing this layer efficiently to provide a robust, self-healing, autonomic cloud-computing system that is fault tolerant and exhibits capabilities to resist and resolve future failures thus making the system impervious to faults.

## Service Customer

This is the highest layer of abstraction that hides all the details of the lower layer and provides services that makes use of the lower layers to do actual work. The layer houses the front end and other network services involved in delivering the services to the end user and actual transmission of data between different components of a cloud computing environment.

The layer is a very high level abstraction of the layers below it and does not provide opportunity directly for building an autonomic computing environment. At the most, this layer can be used as an interface to provide interaction with the lower level.

# Healing

## Healing – An Overview

Healing is the process of restoring a damaged or diseased system to its original working state, which is free from these problems. Being able to automatically detect and discover faults is of great importance for any healing system. The necessary actions must be taken in order to return to a working and fully functional working state. Cloud computing systems must also incorporate healing in order to ensure efficient working which can quickly, efficiently and accurately recover from a problematic state to its previous working state.

The cloud computing environment can also suffer from a varied range of problems and failures. Some of these problems are

* Security issues at both the client and at the server ends.
* Privacy of the users that have registered themselves with a remote cloud computing system.
* Integrity of the user data that is stored on the cloud servers.
* Theft of the user data stored on cloud servers.
* Loss of the user data stored on cloud servers.
* Applications to be stored on the cloud which are infected or are remotely stored on the cloud with malicious intent.
* The services and resources provided by a cloud computing infrastructure are not limited by geographic boundaries. The cloud computing systems provide these services to their clients through a single interface thus making it very difficult to locate the data of different users on the physical storage at the cloud server end.

All the above-mentioned problems make it absolutely necessary to develop an efficient healing system for the cloud computing system.

## Faults[[2]](#id.2grqrue)

Any system that is working perfectly can be susceptible to a large number of faults. Depending on the tasks executed by that system, these faults may take a varied number of forms. The combined effect of these faults is a decrease in the productivity of the system and hence a drop in the system efficiency. In layman terms, the given system no longer functions as it used to before the occurence of the faults in the system. These faults may occur at different levels in the system architecture.

Furthermore, certain faults can trigger the occurence of subsequent faults and this can be disastrous.

A fault in a system is thus a malfunction, that leads to a ceratin deviation from the expected behavior of the system. If we consider the case of a system of inter-connected computers, faults may occur due to a large number of factors, including hardware failure, software bugs, software failure and network problems.

Three types of faults are observed in a typical distributed system of computers:

* Transient faults
  + These faults occur once and then disappear. For example, a network message doesn't reach its destination but does when the message is retransmitted after some period of time.
* Intermittent faults
  + Intermittent faults fault that are reoccurring. These are the most irritating of faults and occur mainly due to component failures or improper inter-component operation, like a loose connection.
* Permanent faults
  + This type of failure is persistent and it will continue to exist as long as the faulty system component is repaired or even fully replaced in extreme cases. Examples of this fault are disk head crashes, software bugs, and burnt-out power supplies.

It is thus essential for any system to incorporate techniques to resolve these faults as quickly and effectively as possible. In general, a fault tolerent system is what is required.

## Fault Tolerance

The basic approach to building fault tolerant systems is redundancy. Redundancy may be applied at several levels.

### Information redundancy

Information redundancy is used to provide fault tolerance by replicating or coding the data. For example, a Hamming code is used to provide extra bits in the data in order to recover a certain ratio of failed bits. Other important samples used to provide information redundancy are parity memory, ECC (Error Correcting Codes) memory and ECC codes on data blocks.

### Time redundancy

Time redundancy achieves fault tolerance by performing an operation several times. Retransmissions in a reliable point-to-point and the use of timeouts along with group communication are examples of time redundancy. This form of redundancy is extensively useful in the presence of transient or intermittent faults. It is of no use with permanent faults. An example is the retransmission of TCP/TP packets.

### Physical redundancy

Physical redundancydeals with devices rather than data. Extra equipment is added to enable the system to tolerate the loss of some failed components. RAID disks and backup name servers are examples of physical redundancy.

There are many challenges with regard to the implementation of fault tolerant architecute for any distributed system or autonomic computing system. Some of these challenges are:

* Implementation of autonomic fault tolerance techniques is required for multiple instances of any application that is running on the several virtual machines which are provided via the cloud computing infrastructure.
* Fault tolerant techniques must be developed which are integrated with the existing workflow scheduling algorithms that have been implemented for the underlying autonomic system.
* It is important that a great level of reliability and availability of multiple cloud computing providers with independent software stacks be ensured.
* Autonomic fault tolerant methods must react in accurate synchronization among the various interacting clouds, otherwise the solution itself can lead to future faults in the system.

It is quite obvious that the techniques that can be developed for automatic fault tolerance are accompanied by a large number of limitations. In the foresight of various users(clients) registered with a cloud, requesting services, it is thus a great ordeal to ensure efficient provision of services without error. We cannot rely on fault tolerant mechanisms regardless of them being automatic.

It is extremely difficult to synchronise fault repairs amongst the various inter-connected nodes of our autonomic cloud computing system, thus manual intervention in the healing process must be kept to a minimal level. Apart from synchronization, many other aspects must also be considered, including automatic and accurate load balancing in the unfortunate case of a the failure of a node or any component of the cloud server. It is this that has led to the growing demand for an autonomic cloud computing system that is capable of “self-healing”, keeping in mind that the system is not completely fault tolerant and impervious to faults.

## Healing

Ever since development of modern computers it has become very difficult to rectify the system faults and manage recovery from malicious attacks due to the increase in complexity of the systems. All these factors resulted in the study in the field of autonomic computing and have explored the concept of self-healing systems. Autonomic computing is a self-managing computing model named after, and patterned on, the human body's autonomic nervous system. Self-healing in autonomous computing is described as the process to to free people from discovering, recovering, and failures. Self-healing systems are expected to heal themselves at runtime in response to any change in environment or operational circumstances. Thus, the goal of self-healing is to prevent disastrous failure through prompt execution of certain proposed actions. We need a self-healing mechanism which is expected to monitor, diagnose, recover from faults and regain normative performance levels independently. The Self-healing technology enhances the system reliability by removing the need for human operation, as human configuration and maintenance of complicated systems makes the system more vulnerable to errors. Conventional ways to eliminate these errors would include log-based level, model-based level, and component-based level approaches. These approaches do support some parts of the self-healing process but not whole process which includes monitoring, filtering, translation, analysis, diagnosis, decision and healing.

## Existing Solutions[[3]](#id.vx1227)

Some of the existing proposals for self-healing system propose a system that encapsulates the self-healing scheme which includes filtering of the internal problems, monitoring the issues, modelling the problem states, analyzing the data, translating the inference, diagnosing anomalies, deciding action states, and healing internal problems. Self-healing systems usually encompass three layers which are the monitoring layer – that monitors states, the Diagnosis and Decision Layer – that checks for anomalies and the Adaptation Layer – that decides the solution. Information such as log context, resource, configuration parameters are monitored by the modules present in the monitoring layer. The monitoring modules detect an irregularity in the behaviour. When the anomalies are detected the diagnosis and decision layer are triggered. The diagnosis and decision layer comprises of various modules that filters, translates, analyzes, diagnoses the problems, and decides its strategy. The last layer in the structured architecture is the adaptation layer which composes modules that execute the strategy selected in diagnosis and decision layer.

Initially the monitoring modules monitor the generation of the log on the fly. Prior to the conversion of the log is converted into the CBE (Common Base Event) format, we perform filtering for the minimization in the size of the memory and disk space used in the conversion of the log. The appropriate adaptation policy is selected using the Rule Model. Here the decision mechanism comes into the picture and it is required to select the appropriate adaptation policy. There are many intelligent sub-modules in the Diagnosis and Decision layer to which this approach to be extended to.

# Problem Statement

*“To provide a robust, self-healing, autonomic cloud-computing system that is fault tolerant and exhibits capabilities to resist and resolve future failures thus making the system impervious to faults.”*

The project aims to come up with an architectural system that provides

* System to detect failures
* System to anticipate failures
* System to take necessary corrective actions once any of the above situation occurs
* Specifying inter-node communication

# Literature Survey

## Autonomic computing system with model transfer[4]

**Patent number**: 7542956

Different data and commands are received from different devices and they are sent to an autonomic manager to produce a single normalized view of the information. The actual state is identified from the normalized view which is compared to the desired state. If there is a difference, configuration adjustment is done to reach the desired state.

**Figure 1: Approach for Controlling a Managed Resource using an Autonomic System**

The technique uses machine learning, dependency processing and action determination logic to decide the action to be taken in case of a mismatch. All the systems are shown in the figure below and are self-explanatory.

**Figure 2: Block diagram of an Autonomic System**

## Architecture for a Self-Healing Computer System[[5]](#id.3fwokq0)

**Patent number**:  2010/0281134 (Patent pending)

The self healing system comprises a self healing processor and an error mitigation system, a code block associated with the operation of a portion of digital logic, dynamic signature analysis circuit. The processor executes the code block. The dynamic signature analysis circuit creates a dynamic signature representing the operation of the portion of digital logic associated with the code block. The error mitigation system receives the dynamic signature from the dynamic signature analysis circuit, and compares it with a static signature to determine if the signatures match. If the signatures do not match then the digital logic associated with the code block has an error. The error mitigation system retries execution of the code block. The error mitigation system stores log information describing the above events.

**Figure 3: Embodiment of a Self Healing Processor**

The Error Mitigation System is the core of the entire architecture. It houses a variety of components that help in finding out the error and correcting it. A lot of techniques like learning and lookup tables are used in this.

**Figure 4: Embodiment of an Error Mitigation System**

## System and Method for Achieving Autonomic Computing Self Healing Utilizing Meta Level Reflection and Reasoning[[6]](#id.1v1yuxt)

**Patent number**:  7260743

In a base level a monitor detects an error in a production environment and provides reification message comprising data about the error to a meta level. A reasoning system in the meta level receives the reification message and analyzes the data using knowledge of computational components in the base level and identifies a self healing action for the error and returns a reversion message comprising a signal to implement the self healing action. Responsive to receiving the signal the base level implements the self healing action.

**Figure 5: System and method for Handling Errors**

# Proposed Solution

After having studied the cloud architecture in detail, we have proposed to carry out self-healing techniques in autonomic cloud computing environments with a unified approach.

The approach to be followed is based on one basic principle i.e. to compare the current state with the expected states. This can be done using a comparison scheme and defining the new set of data to be collected and monitored.

The comparison scheme described above, works with the other elements of the cloud software environment layer. This is the layer that actually defines the cloud and manages the nodes, the computational resources and storage facilities in the cloud. It is at this layer the communication between the nodes happen. This layer thus provides excellent opportunities for developing reliable, self-healing, robust, autonomic computing architecture in the cloud.

The solution would involve coming up with a architecture that communicates and manages the nodes and efficiently involves the self-healing concept of fault recovery. A high level survey has pointed out the need of further survey in inter-node communication and message passing interfaces in the cloud.

As shown in the above figure, initially the architecture is in a stable state and all the components are functioning normally. The arrows connecting the nodes represent the interactions taking place between the nodes. There is a comparison scheme that is constantly monitoring the state of the architecture to ensure that there are no anomalies. This comparison scheme compares the instantaneous state of the architecture with the expected state i.e. the state which the architecture should ideally be in when it is functioning flawlessly without any anomalies.

The expected state is evaluated using the data collected over a period of time as the architecture functions without failure. As the architecture is constantly monitored under normal operation, a signature of the state can be created that defines how the architecture appears when there are no significant anomalies. Once the ideal state signature of the architecture has been constructed, the instantaneous state of the architecture can be constantly compared with the ideal state. While doing so, the healing architecture constantly collects data and stores it for analysis. Analysis of this stored data allows the healing mechanism to identify anomalies if any. This accomplishes the first phase of the healing process, i.e. detection of failure.

The comparison between the instantaneous state signature and the ideal state signature allows us to determine if there is any discrepancy in the working of the architecture. If there is a discrepancy it indicates a potential problem. If it does not reveal any discrepancy it indicates that there are no problems with the working of the architecture and hence the architecture is in a stable state i.e. there is a match between the current state signature and the ideal state signature.

Incase of a mismatch between the ideal state signature and the instantaneous state signature corrective action must be taken to resolve the anomaly and bring the architecture back into the ideal state. Corrective action can be decided by making use of the following policies:

* Generate and Test method
  + This is a very simple solution that generates a possible solution to the problem and applies it to test if the solution solves the problem, if not the architecture will generate another solution and continue until a working solution is found.
* Rule based correction
  + This appreach makes use of a static rule policy to decide the corrective action. Though simple it is not feasible for large and complicated architectures.
* Learning based correction
  + This approach appies a solution and dynamically keeps learning over time and uses guessing to come up with a good solution to restore the architecture back to the normal state.

These solution methods help us develop a robust, self-healing, autonomic cloud-computing system architecture that is fault tolerant and exhibits capabilities to resist and resolve future failures thus making the system impervious to faults.

# The Path Ahead

There is also scope for automatic learning and finding out patterns that might lead to a failure and prevent failures before hand when such patterns are seen. Focus must be directed to anticipating failures before they occur. Efficient data and statistics about nodes must be generated so as to reduce the overhead of the healing architecture and make it as transparent as possible. The proposed solution still needs formalization and detailed work must be done to formalize it into architecture.

# Bibliography

|  |  |
| --- | --- |
| [1] | Wikipedia. [Online]. <http://en.wikipedia.org/wiki/Cloud_computing> |
| [2] | Inderveer Chana Anju Bala, "Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 1, January 2012. |
| [3] | Jinsoo Jung, Shunshan Piao, Eunseok Lee Jeongmin Park, "Self-healing Mechanism for Reliable Computing," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 3, January 2008. |
| [4] | Barry J. Menich John C. Strassner, "Autonomic computing method and apparatus," US 7542956, June 2006. |
| [5] | Nader W. Moussa, Makoto Honda, Hideo Ikai, Kozo Kato Roger D. Melen, "ARCHITECTURE FOR A SELF-HEALING COMPUTER SYSTEM," 2010/0281134, March 31, 2009. |
| [6] | Carl Phillip Gusler, Rick Allen Hamilton, II, Mamdouh Ibrahim Craig William Fellenstein, "SYSTEM AND METHOD FOR ACHIEVING AUTONOMIC COMPUTING SELF HEALING UTILIZING META LEVEL REFLECTION AND REASONING ," US 7260743, January 13, 2004. |