CLOUD COMPUTING FOR SPACECRAFT CHECKOUT OPERATIONS

Priyanka T L[1], Rekha Patil[2], Sreedevi. S[3]

[1]*PG Student, Department of CNE, P D A College of Engineering, Kalaburagi, Karnataka, India.*
priyaoff92@gmail.com
[2]*Associate Professor, Department of CSE, PDA College of Engineering, Kalaburagi, Karnataka, India.*
rekha.patilcse@gmail.com
[3]*Sci/Eng 'SD', CSAD/SCG, ISAC, Old Airport Road, Bangalore, India.*
ssrd@isac.gov.in

Abstract

The Spacecraft Checkout deals with testing and validation of all Spacecraft elements and its functionalities before launch. The checkout deals with extensive testing operations on the Spacecraft and the data related to the tests are also voluminous. About 60 percent of the Spacecraft Checkout operations have been automated using a suite of software tools which runs on Checkout servers which are on LINUX platform. The automated operations using this software suite requires a lot of backend effort for setting up configuration files, databases with information on Telemetry, Telecommand and other subsystems, test files which details the operations to be carried out are logged for future reference. A lot of manual effort goes in the checkout preparatory phase where in the databases, test procedures and configuration files are prepared. During the checkout operations, the satellite data acquired has to be distributed online to different agencies for analysis in real time as well as at a later stage. This project aims at exploring cloud platform based solutions for the unified platform where all satellite preparatory data can be kept, support for tools which aids in generating databases and test files based on previous spacecraft data, spacecraft Data Access Portal for External Agencies and support for tools which aids in Remote Checkout operations. A feasible architecture which suits the checkout environment will be the outcome.

Keywords: *Cloud Computing, Hot Redundancy, Hypervisor, Juju, KVM, LAN, MAAS, Open Stack and RAID.*

--***--

1. INTRODUCTION

Spacecraft Checkout activities are automated to a great extent through a set of software products called Checkout Software suit which are housed in the computer called Checkout computer. The data acquired from Checkout Operations by that computer should be stored for the future retrieve by the client. That software suit and data should be made available to the client whenever they need it. Application software, operations and data should never go down at any time as these operations and data are accessed at any time by the client to test and validate the Spacecraft element, which should be done with minimum operation time. Time to do all these operations is a critical performance character.

To optimize the time for testing, we should provide a unique platform for all the clients to make remote access of software, software tools and updated data by all the clients with increased performance speed. To run software and do the test operations with high speed with no down time and also making the updated data copies available by all clients, we need a platform to do so. Such platform which provides all these services should also provide a redundancy of the data and it should be fault tolerant, so that if the data or application software is lost by the client due to any system crash or by any other means it should be able to rebuild the data or that session.

This paper discusses a new platform which meets all the requirements by providing tools to store and retrieve the Checkout Software suit and data easily. It provides backup, which will never let it go down anytime and also provides scalability of hardware and software. It provides high availability and high uptime of software and data. This paper at first gives introduction about Checkout Software Operations and then tells why to use Cloud to implement required platform. Then it deals with the design options to build cloud platform in related work. Then it discusses with implementation of private cloud over the existing system. Then it discusses the results. At last it concludes and states the future enhancement.

2. CHECKOUT SOFTWARE OPERATIONS

Spacecraft rigorously undergoes about 3 to 6 months of testing before launch. Satellite Integration and Testing activities are important milestone in the realization of a spacecraft. The time and effort required to deliver a fault free spacecraft is a very challenging task. Spacecraft checkout operation consists of extensive tests on the spacecraft subsystems and the payloads for the ascertainment of satisfactory performance during various stages of integrated satellite testing before launch. Checkout database schedules test files as per the requirements and it executes them by testing the satellite.

The objective of Spacecraft Checkout is to evaluate the functional performance of the integrated spacecraft and to check all on-orbit operational modes and redundancies. The main tasks of checkout are to provide controlled testing programme to provide means for assessing the overall performance of the integrated spacecraft and to simulate the different on orbit modes and evaluate the performance of various onboard subsystems. Integrated spacecraft testing involves carrying out approved tests on spacecraft during various stages of spacecraft assembly, after each major mechanical operation on spacecraft, for environmental tests, carrying out prelaunch operations and configuring the spacecraft for final lift off and generating timely test reports and initiating corrective actions whenever required. Hence, Checkout software can be defined as a set of software products for automation of spacecraft checkout operations.

Main Checkout functions include automatic sequencing of test procedures, reception of Telemetry data from data acquisition system, processing and display of data, generation of anomaly reports and test reports, storage and retrieval of raw data and processed test results. At the core of checkout, is the Automatic Test Sequence interpreter providing a single point of control for all the spacecraft tests by controlling all the various equipments and controllers (which in turn control various equipments) of the three major elements of Spacecraft Checkout being the Overall Checkout, Special Checkout and payload Checkout.

With the ever increasing number and complexity of spacecrafts, there is a need to quickly, efficiently and accurately test the spacecraft in a diverse set of environments from the disassembled to environmental simulation and through launch preparation. The way to fulfill this need is to improve automation as this is considered crucial to efficient and cost effective checkout operations. Current level of automation includes open system architecture with all the interfaces standardized, well proven Command Language for preparing test schedules in advance, very versatile data processing, presentation and analysis tools. This has enabled us to achieve error free spacecraft testing with minimum human intervention, repetition of tests are assured at various phases of testing, testing time has reduced by more than 50 percent and whenever required, the test files are modified/updated based on the requirement of the new satellite which will reduce cost factor requirements. For all those operations we provide a cloud platform, a single platform to make it more redundant and scalable.

3. WHY CLOUD

In addition to automation, the ease of configuration for various spacecraft projects, maintaining the availability of data and tools, hot redundancy and backup are also important for the success of checkout software. We have many other different options to achieve automation, availability, storage backup and hot redundancy, for example Web applications. Still why should we prefer cloud? Why not web applications though all the cloud applications are web applications?

In cloud computing applications computation happens on cloud servers located somewhere than on clients machine. All the components of a cloud application use sophisticated back-end support to ensure uptime, security and integration with other systems. It supports maximum access methods. Web application program relies on remote server, using browser it is delivered over the Internet or an intranet.

All cloud applications are inherently scalable because they take full advantage of the underlying hardware. Cloud application will not limit the number of users or workload but web applications are limited by scalability. Cloud stores data in multiple replicated data centre, provides standardized applications for all its customers. User can run application on user's system. Cloud uses multi tenancy instead of isolated tenancy to provide high accessibility, access control and authentication. Cloud provides high uptime by mirrored installations of cloud applications so that applications are always available (~100% uptime) [1], making it highly available. So using cloud we can meet the Checkout Operations in a cost efficient and secure way.

What is cloud computing? Cloud computing can be thought of as a network computing approach, where applications run on a server or a group of server which are owned by a provider rather than on the client machine or system [2]. It has deployment models, delivery models, different infrastructures, resources and defining attributes to implement private, public, hybrid or commercial cloud as per the requirement and architecture need by the provider as well as the client.

There are many cloud providers available in the market, using their services Spacecraft organization can meet its hardware and software requirement, which are cost efficient compared to owning cloud. But data and all the Checkout software and their operational result are highly confidential. Placing such confidential data on provider's storage will not be risk free, because of hacking, leakage of data or loss of data, which leads to security threat. Thus we are designing private cloud platform, owned by the organization.

4. RELATED WORK

There are number of ways to create and manage private cloud. One can use RightScale [3] to make their storage system as private cloud. Eucalyptus, OwnCloud, OpenStack, IBM VersaStack and Vmware vSphere like that many sources are there to design. Some are open sources and some are paid. One can also assemble all storage sources by deploying drivers to build private cloud for storing data with all possible security layers, which should be powered always and made available over Internet to access which will be very costly.

The main aim of this project is to provide a cloud platform for the existing system in a cost efficient way. Main challenges in building this cloud platform are backup storage and hot redundancy. The required platform can be designed using the free resources, minimal hardware and software support, which can be used to deploy existing Checkout Software suit, to make it easily accessible at any time with high availability and high uptime. It also provides hot redundancy.

Redundancy can be achieved at software or hardware level. The level of the data backup needed for the system decides the type of the redundancy to be achieved by the platform. There are several different types of redundancy: Network, Hardware, Power, and Geographic. For example, a solid web hosting company will have multiple layers of Redundancy to ensure that data is safe and to maximize the uptime. As our design needs hot redundancy, we can use hardware redundancy. We can achieve hardware redundancy with different architectures, and are called Redundant Array of Independent Disks (RAID) levels.

RAID provides mirroring of data and storing mirrored data in different data in different places on multiple hard disks. Some RAID levels will not provide redundancy. Placing data on multiple disks balance the overlapping input/output (I/O) operations which in turn improve the performance and such redundant data has high fault tolerance as it increase the mean time between failures. Different standard RAID levels are available to achieve redundancy. Combinations of two are more standard RAIDs will result in nested RAID, which can be used to achieve the redundancy of the data in a cost efficient manner for very large data.

5. IMPLEMENTATION

To build private cloud over the LAN network, we used OpenStack [4], which makes it more reliable, efficient and provides good automation. This platform is designed for intranet/LAN (private cloud) which is a big challenge for developers. No external network is allowed to connect and no internet is used except during the implementation of the drivers, OS and software that need Internet.

5.1 Hypervisor

Virtual hardware is very cheap and flexible. Choosing right virtualization technology to provide required environment is too important. Cloud is not all about virtualization but virtualization is a critical concept in building a cloud. Virtual machine should meet and exceed the performance of their physical counterparts, at least in relation to applications within each server. Virtual machine which meets beyond this Benchmark for select cost efficient virtual machine is that it should meet and exceed the performance of the existing hardware. Selection of hypervisor is based on maturity of the software, consolidation ratios, hardware costs, software costs, and some more critical factors which meets the requirements and needs of the client. After discussing all critical factors with the team we selected the right hypervisor for the platform. Here we used Type I hypervisor that is KVM (Kernel Virtual Machine) over which the drivers and OS is deployed [5]. It creates virtual environment for the client by the platform.

5.2 OpenStack

Open Stack is an available free resource to build private cloud platform. This software consist functions and tools to control processing hardware pools, storing, and networking resources throughout a data centre. It manages the cloud platform through a web interface such as dashboard, through command-line tools or through application interface such as RESTful [6]. To implement OpenStack we used MAAS and Juju packages. OpenStack is a free source, whose source code is also available and can be modified according to the requirements. Totally two main servers are used to install these two packages to implement OpenStack.

5.3 MAAS (Metal as a Service)

MAAS [7] is installed over a server, a canonical tool which should be installed before installing Juju. Its major function is to bring language of the cloud to physical servers. Dynamic scaling and configuration of hardware deployed with cloud services are made easy by MAAS. This feature is important in cloud since cloud requirement can fluctuate dynamically. We can use simple web interface we to add, commission, update and recycle servers as per the requirement. As requirements or needs change, we can respond rapidly by adding new nodes and dynamically re-deploying them between services.

5.4 Juju

Then server is installed with the Juju package. Juju [8] uses the concept of an environment. An environment is a particular type of infrastructure used to deploy cloud. Juju supports different types of environments: deploying on top of Joyent, OpenStack, Amazon EC2, HP Public Cloud, Windows Azure, directly on top of hardware (which is called bare metal and MAAS), etc. We installed it directly on top of hardware. Then we deployed software using Juju on top of a private cloud running OpenStack. Therefore, before proceeding to further steps, a proper OpenStack deployment is made available and made sure that it is functioning properly, that means that Keystone, Nova, Neutron and all necessary components are up, healthy and reachable. Following block diagram shows the simple architecture.

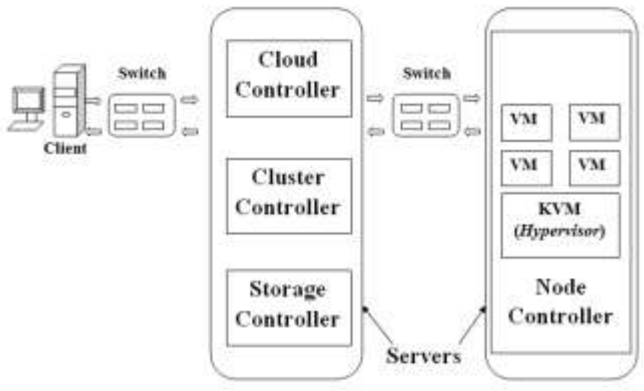

Fig -1: General Architecture Block Diagram.

Cloud Computing is well known for its agility. According to the demand it quickly 'switch on' or 'switch off' cloud services and juju provides this functionality. Juju provides unique functionality do deploy cloud services and are straightforward. Juju shrink or grow the cloud services according to the requirement and demand. Real time modification functionalities are also available in juju.

5.5 Metadata

There are two main issues when deploying a private cloud:

- Providing unique image ids on cloud.
- Restricting external internet access to private cloud

First issue means have to generate image id metadata and make it available by clients. Second issue means all the tools and data needed are mirrored and make it locally accessible on cloud. Juju tools exist to help with generating and validating image and tools metadata.

5.6 Simplestreams

All the metadata is stored as json metadata which are called Simplestreams and are created by using the tools provided by Juju. According to the design we created our own metadata and it can be changed according to the work and need. The Simplestreams format provides structural fashion for all related data items. The general overall workflow is:

- Create metadata of image
- Created image metadata is then copied to search path of metadata somewhere in storage
- Then duplicate tools in the search path of metadata
- Then configure duplicated tool's and/or metadata image's URL path

5.7 Hot Redundancy

Server's hardware piece is a backup for example hard drives can be used as redundant layer, then data will not go offline or lost because RAID arrays are immediately rebuilt [9]. This is called hardware redundancy. Hot redundancy can be achieved by using hardware redundancy. We used a standard RAID 1 for Checkout Storage to achieve hot redundancy.

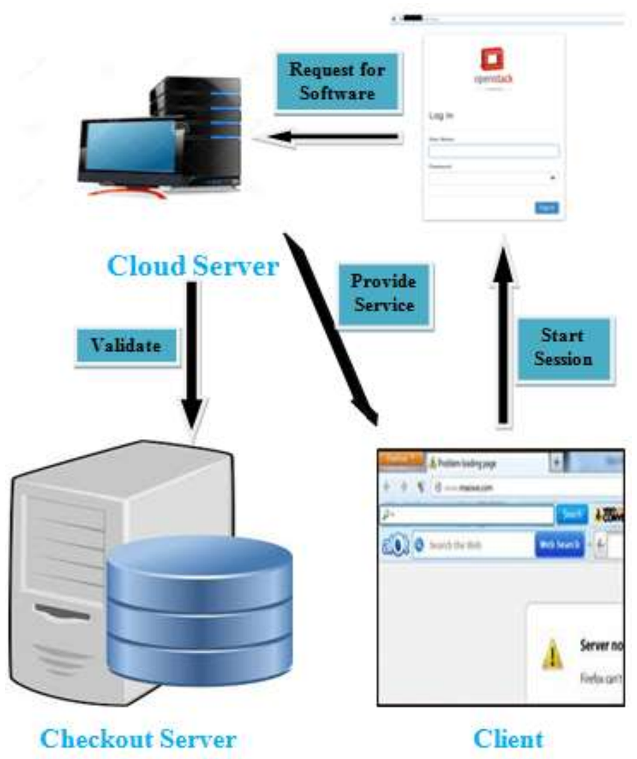

Fig -2: Working of Cloud Infrastructure.

Above figure shows the interaction of client, cloud server and Checkout storage server. The software shall start its execution by establishing a connection to the system for which IP address, user name and password are specified in the dashboard. Software shall establish a connection with MySQL database of the connected system to read and load all database tables into memory. The MySQL database contains the configuration database tables- "Configuration" and a template table specifying its fields named as "Configuration Template" so that the software shall read the Checkout Software configuration from configuration table using the configuration template table and then set up the required variables and path and launch the GUI which provides a view of the user area and operational area of the connected system. While fetching the lost data RAID will rebuild it using the backend server.

6. RESULTS

After logging onto the cloud server, we fetch the software or any tool for Checkout operation to test and validate spacecraft element. We can run it over the virtual environment provided by the platform or on client's system.

All the data, software and hardware needed by the client will be made virtual by the cloud server. Whenever client runs any Checkout Software or tool, it will be provided over virtual environment provided by the cloud server. Then operated and updated data will be directed to the Checkout Storage by the cloud server. Cloud server enforces automated creation/editing of test files in a defined format utilizing the information setting from Checkout Operation as well as from the cloud server. Cloud server needs minimum of 32GB RAM, a very high speed processor and memory will be decided by the cloud server. If data is lost over the client machine or system crashes, then Checkout Storage backend will rebuild. Client can access it again using the dashboard on other system.

7. CONCLUSION AND FUTURE WORK

This is a single platform for all the users where every client can easily access, update and store the data. Automation work is made easier and faster by providing tools for easy access and to aid in remote access of Checkout Operations and Software to carryout Spacecraft element testing and validation, then stores the operated data on Checkout Storage. It provides hot redundancy. Stored data can also be retrieved and used in future. If in case any unexpected event occur then execution can be suspended and resumed back using the dashboard and data is rebuilt using the RAID level over Checkout Storage sever and also using the Test Scheduler tools.

Present platform is designed for single satellite. Future enhancement is to make this platform available for all satellites of the organization. This enhancement will increase the data to be stored on Checkout Storage server. To cope up with this change, existing Checkout Storage should be moved to suitable Nested RAID architecture without disturbing existing system. It should also be made highly scalable for all satellites along with high availability and high uptime for data, software and operations.

ACKNOWLEDGEMENT

The authors express thanks to Mrs. Vasantha Kumari U, Group Director, Spacecraft Checkout Group (SCG), Mrs. Muthumani Rajan, Division Head, IMCD, SCG and Mrs. Sheena Jose, Section Head, IMCD, ISRO Satellite Centre, Bangalore, India for the valuable guidance, support and inspiring innovation throughout the realization of this project. This work was supported in part by a grant from ISRO Satellite Centre.

REFERENCES

[1]. Alargam Elrayah Elsanhouri, Mahmoud Ali Ahmed and Abdul Hanan Abdullah, "Cloud Applications Versus Web Applications: A Differential Study"", 2012 IARIA.
[2]. https://en .wikipedia.org/wiki/Cloud_computing
[3]. https://rightscale.com/blog/cloud-management-best.../
[4]. https://www.openstack.org/
[5]. www.globalknowledge.nl/content/.../Citrix-White-Paper-Choosing-Correct-Hypervisor
[6]. Vinay Khedekar, Girish Mane, Siddhi Khanvilkar, Shreedhar Karade and Prof. Atul Yadav, Study of Cloud Setup for College Campus", 2012 IJARCSSE
[7]. www.ubuntu.com/cloud/maas
[8]. www.ubuntu.com/cloud/juju
[9]. http://searchstorage.techtarget.com/definition/RAID