Cost Effective NSX Inventory State Management in Cross Cloud

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Computer Science and Application

Submitted By

Saloni Garg

Roll No. 601534014

Under the supervision of:

Ms. Anika

Lecturer, CSE Department

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2017

vmware®

June 9, 2017

Internship Completion Letter

This is to certify that Saloni Garg (352783) has been engaged with us from August 8, 2016 to June 9, 2017 as an Intern.

We wish Saloni Garg in all future endeavors.

For VMware Software India Private Limited,

Subhash Sathyanarayan
Senior Manager, HR Operations
Email: indiahrops@vmware.com

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Cost Effective NSX Inventory State Management In Cross Cloud*", in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Application* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Anika* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Saloni Garg
(Saloni Garg)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Anika 13-July-17
(Ms. Anika)

Lecturer, CSE
Department

ACKNOWLEDGEMENT

First of all I would like to thank the almighty, who has always guided me to work on the right path of the life.

This work would not have been possible without the encouragement and able guidance of my supervisor **Ms. Anika**. I thank my supervisor for their time, patience, discussion and valuable comments. Their enthusiasm and optimism made this experience both rewarding and enjoyable.

I am equally grateful to **Dr. Maninder Singh**, Associate Professor and Head, Computer Science & Engineering Department, a nice person, an excellent teacher and a well – credited researcher, who always encouraged me to keep going with work and always advised me with his invaluable suggestions.

I will be failing in my duty if I don't express my gratitude to **Dr. S.S. Bhatia,** Senior Professor and Dean of Academic Affairs, Thapar University, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Last but not least, I would like to thank my family whom I dearly miss and without whose blessings none of this would have been possible. To my parents, I own thanks for their wonderful love and encouragement. I would also like to thank my brother, since he insisted that I should do so. I would also like to thank my close friends for their constant support.

Date: June, 2017
Place: Thapar University, Patiala

Saloni Garg
(Saloni Garg)

ABSTRACT

With the extreme grown in technology, the amount of data that belong to an organization is many folded. It is practically impossible to persist entire data on premises. So many organizations are turning towards cloud. So the focus is on cross-cloud environment where NSX manager that manages both on premise and public cloud workflow is deployed on premises of an enterprise customer. In such a situation, data must be sent to the public cloud environment. But there are certain fault scenarios, which require whole data to be sent again and again to cloud. The aim is to reduce the amount of data being sent in such situations and find the current cost of maintaining inventory information.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

L2	Switching Vertical
DFW	Distributed Firewall
MP	Management Plane – sometimes used interchangeably with NSX manager Node
MPA	Management Plane Agent on host
RMQ/Message Bus	RabbitMQ, message bus that connects the MPA with MP
Discovery agent	Agent for inventory service running on host connected to RMQ through MPA
VIF	Virtual Interface
vCPU	CPU of the VM
vCenter	Virtual Center
SYNC_INIT	Synchronized INIT
SYNC_INIT_REQ	Synchronized INIT Request
DC	Data Collector
DS	Data Sender
MT_INIT	Message Type INIT
MT_UPDATES	Message Type Updates
MT_COMMAND	Message Type Command
CT_CREATE	Command Type Create
CT_MODIFY	Command Type Modify
CT_DELETE	Command Type Delete
DMT_INIT_START	Discovery Message Type INIT Start
DMT_UPDATE	Discovery Message Type update

CHAPTER 1

INTRODUCTION

1.1 Background

Inventory service provides data to other verticals about the environment; everything in the datacenter that is external to NSX and that NSX needs to interact with. Information for that can be collected through VC or through agent running on hosts and are provided to NSX management plane. Inventory contains items that are either discovered dynamically or pre-existing items that are registered by administrator. Some of the main objects that are identified are hosts, virtual machines and virtual network interfaces.

The inventory service is a part of NSX Manager Node. It is installed along with NSX manager node. The inventory service has an agent on the host that will be installed as a part of NSX bundle. Agent is connected to manager through MPA (management plane agent).

Most of inventory service data is ephemeral since it doesn't need the data and can collect it if anyhow lose it. Host and virtual machine identifiers are disk persisted but their properties are ephemeral.

Compute Virtualization is creating a virtual environment for any program to run on an existing platform as a guest, without interfering or interrupting with the host platform's services or programs. It helps to reduce the cost because less hardware resources are required. Also it adds flexibility to the system. New virtual servers can be added in very less time. So it makes provisioning of resources and applications easier and faster.

1.1.1 VSphere:

It is composed of two main components namely vCenter Server and ESXi host. ESXi is a virtualization platform where virtual machines and virtual appliance can be created and run. VCenter Server acts as administrator of ESXi hosts. It manages resources of multiple hosts. Figure 1.1 shows the vSphere structure.

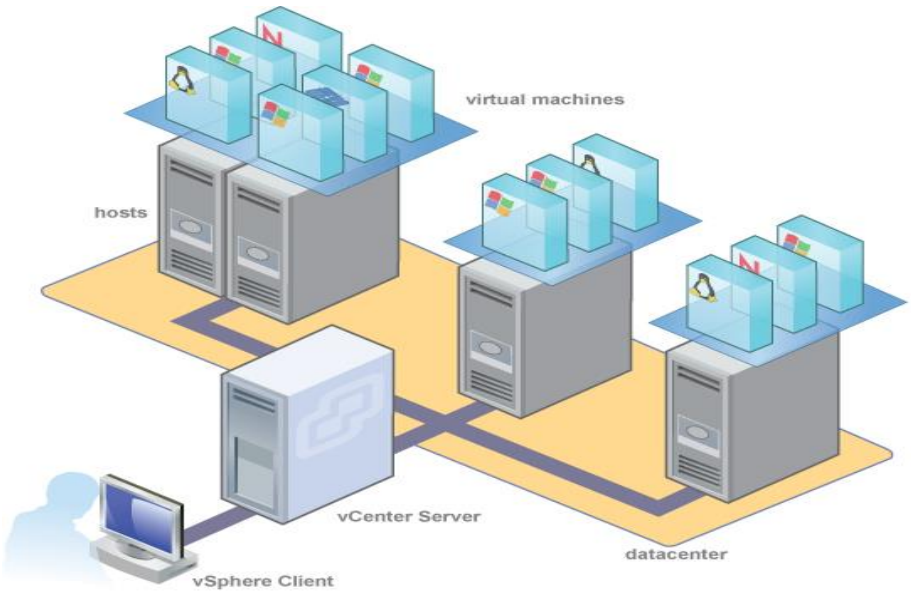

Figure 1.1: vSphere [18]

1.1.2 ESXi:

VMware ESXi is a bare metal embedded hypervisor, which is present on the top of physical hardware. Figure 1.2 shows the ESXi Server structure.

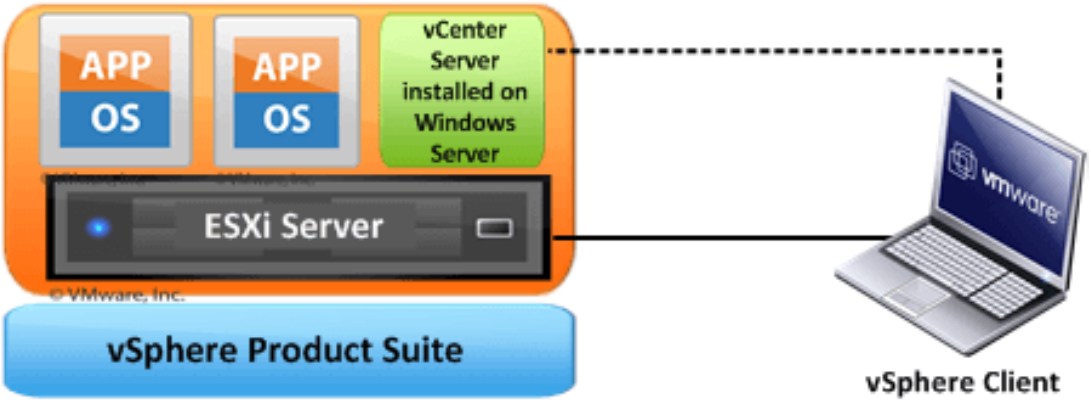

Figure 1.2: ESXi Server [18]

It is a virtualization platform which helps to run more servers on the same physical hardware leading to efficient utilization of the hardware resources such as CPU, memory etc. It allows running virtual machine by providing resources as and when required.

Physical machine's software representation called as virtual machine where applications and operating system can run.

1.1.3 vCenter SERVER:

vCenter Server is required to manage vSphere environment. vCenter Server is a platform that manages virtual machines and hosts. It is responsible for management, operation, and provisioning resources to virtual machines and hosts. It can be installed on Microsoft Windows virtual machine or physical server. Figure 1.3 shows the vCenter Server structure.

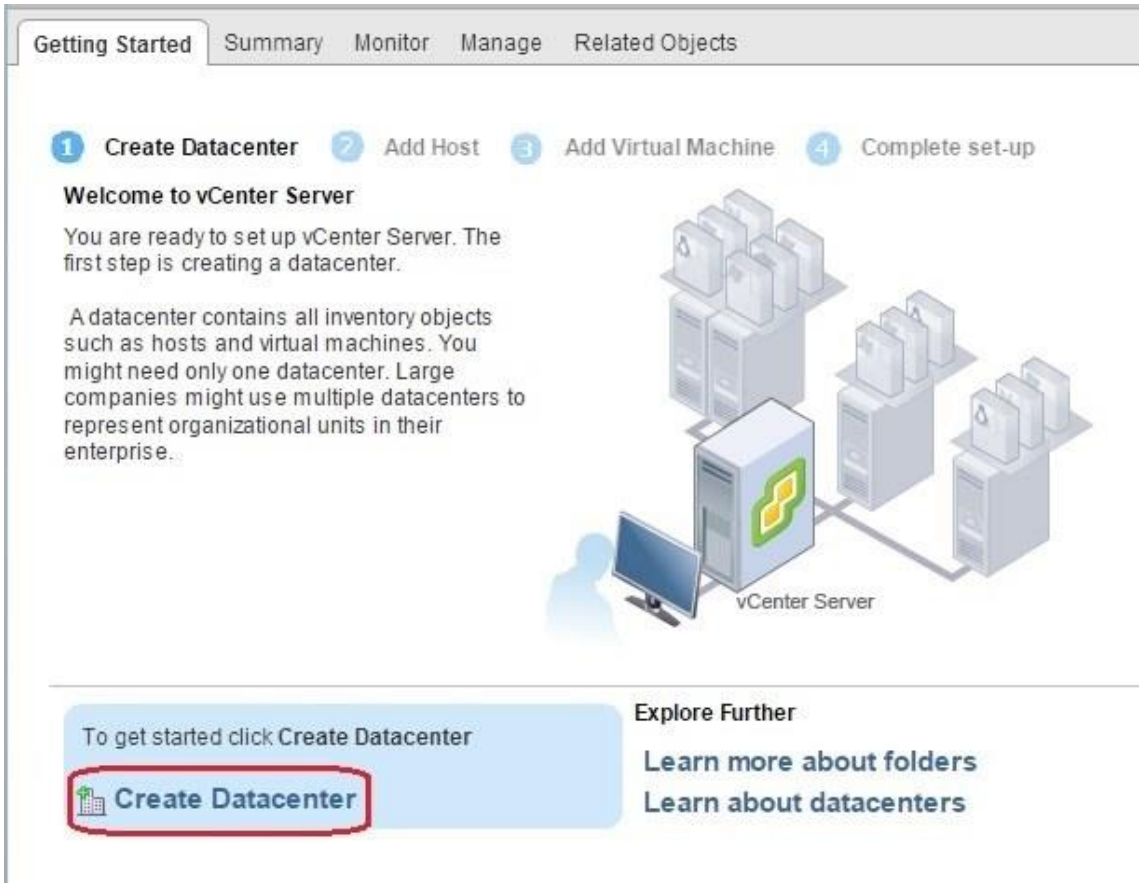

Figure 1.3: vCenter Server [18]

A network virtualization platform where virtual networks are programmatically created, provisioned and managed by utilizing the underlying physical network, called as

NSX. NSX provides logical networking services such as logical switches, logical routers, and logical firewalls, logical load balancers, logical VPN, QoS, and security.

1.1.4 NSX:

A network virtualization platform where virtual networks are programmatically created, provisioned and managed by utilizing the underlying physical network, called as NSX. When a VM is moved to another host, its networking and security services move with it. And when new VMs are created to scale an application, the necessary policies are dynamically applied to those VMs as well. It allows organizations to implement virtual networks without interfering with existing applications and network configurations. With this network can be configured in very less time. Figure 1.4 shows the NSX Architecture.

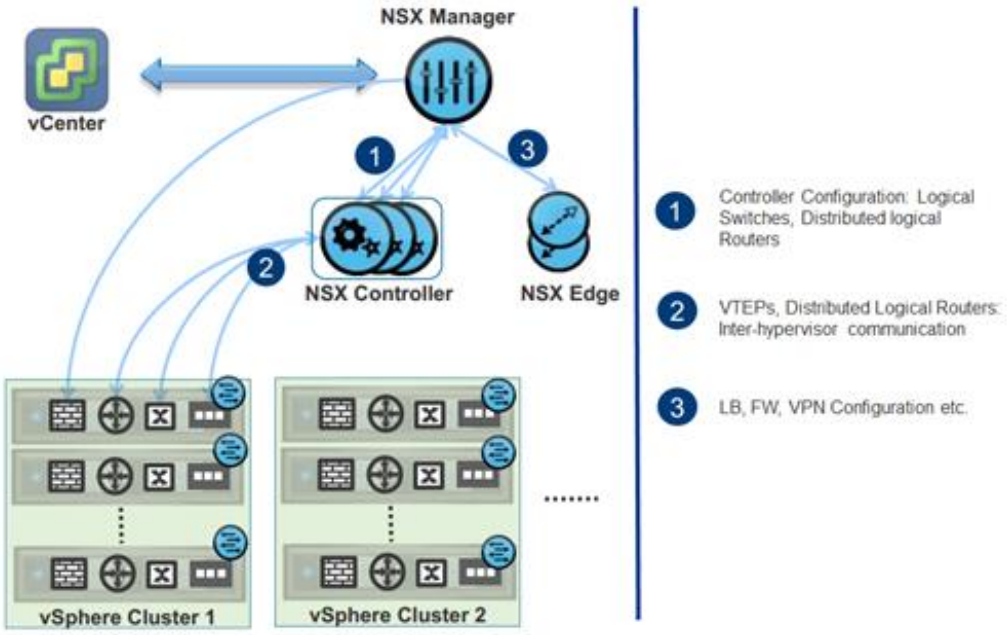

Figure 1.4: NSX Architecture [19]

Virtual network provides logical network services – logical switches, logical routers, logical firewalls, logical load balancers, and logical VPNs to connected workloads. These network and security services are delivered in software and require only IP packet forwarding from the underlying physical network.

In case of NSX deployment, virtual machines and virtual network interface of distributed set of ESXi /KVM host needs to be discovered and persisted. But for on premise deployments, when number of host grow than it is difficult to persist all of the attributes of required set on NSX manager. So the inventory data should move to cloud. Figure 1.5 shows the components of Cloud Controller.

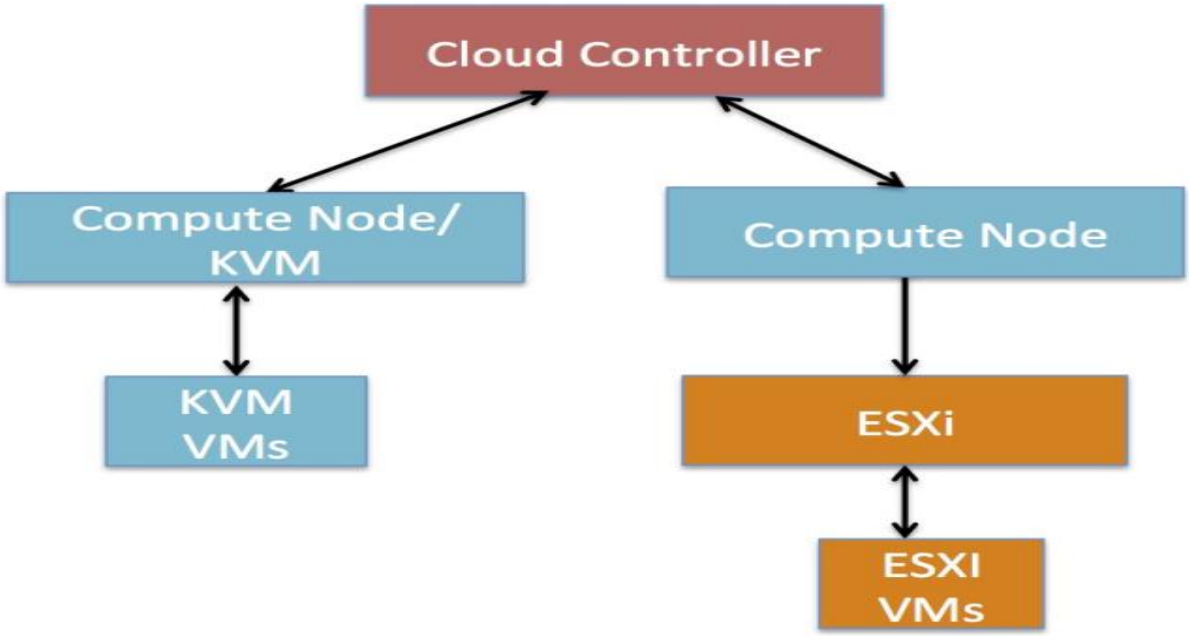

Figure 1.5: ESXi/KVM [20]

There are certain components on premises that together make this transfer of data to cloud possible. The brief description of these components is given below.

The discovery agent is a host-based agent that reports the VMs and VIFs available on the host. Figure 1.6 shows the inventory data flow diagram.

It has the following designing goals:

1. Create an agent as part of MPA bundle that is installed on host.
2. The agent will be supported on ESXi and KVM.
3. Create a listener module in MP that gets the data and uses Object CRUD component to insert the data.

4. Entire host data will need to be reconciled in the following cases.
    - NSX cluster recovers from the failure/shutdown and we need to re-discover all the VM/VIF data.
    - Host agent recovers from failure/shutdown.
    - Host agent does not receive acknowledgement for a certain period of time.
    - Host has too many messages in queue and has to discard incoming notifications; it will clear the queue and ask manager for full sync request.

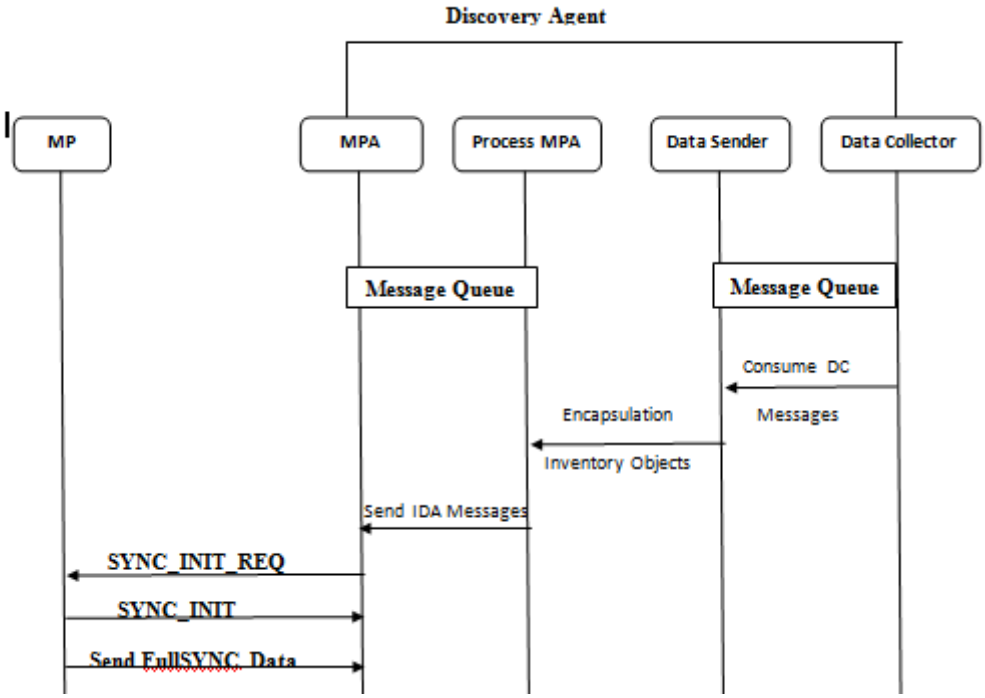

Figure 1.6: Inventory Data flow diagram

5. The agent will be installed as part of MPA bundle.
6. DA communicates with the management plane to send the data. The protocol is explained in later section.
7. IDA message sent from MPA to MP comprises of following information.

- Host contains host id, host entry and interface. Host entry identifies the host properties and their values. Interface specifies interface type (Physical, Virtual), connected_switch etc.
- Virtual Machine contains VmIdentifier and VmEntry. VmIdentifier identifies the VM name (ESX, KVM or HyperV) and VmEntry contains VmPropertyKey and its values.
- Virtual Network Interface contains id, Vni entry, VmIdentifier and IP add Information. Vni entry contains Vni's properties its states.

1.2 Thesis Structure

Remaining analysis of theory is managed in this way:

- The earlier work that is done in the field of NSX inventory management and fault case sceneries are described in chapter 2.
- The problem statement is mentioned in chapter 3. Here gap in existing system within the current inventory management is stressed upon.
- Chapter 4 tells about the existing system.
- Chapter 5 discusses about the implementation part. It describes about the problem solutions and what are all steps performed to solve the problem.
- Results and discussions part describes in chapter 6, it focuses on the results obtained after the proposed algorithm was implemented and their analysis.
- The conclusion part is in Chapter 7, followed by future work of this proposed system.

CHAPTER 2

LITERATURE SURVEY

Every organization requires inventory for proper functioning of its activities. It links production and distribution processes together. In inventories, the investment is the most important part of current assets. Hence it is required to have proper management and control of inventories. The goal of inventory management is to make sure that the materials are available in sufficient quantity and to reduce the investment in inventories.

Shial and Majhi [1] suggested that for similar kind of files De-duplication works fine. Selecting the most suitable chunk size may result in production of maximum number of chunks, which in turn results in higher probability of getting maximum number of similar chunks. So finding best chunk size is a challenge in itself and it differs depending on the type of files being used for synchronization. Chunking method works well over all file synchronization technique. Apart from other file synchronization methods, Intellisync needs to setup client server architecture for the same.

Bellare and Canetti [2] introduced Keying Hash Functions for Message Authentication 1996 [2]. In case of internet applications and protocols, MD5 or SHA types cryptographic hash functions for message authentication has become a common approach. They are very simple to implement but these mechanism supports ad hoc techniques that have less security analysis or have less secure. It has built new schemes of message authentication supported a cryptographic function. There are some message authentication schemes like NMAC and HMAC, tested to be more secure because they have some hash functions which has affordable cryptographic strengths. Moreover it represented, in a very quantitative means, that the schemes retain almost all the security of the underlying hash function. As a black box, they use hash function or its compression function, so that mostly used or likely available library code or hardware will be fixed to implement them in a very simple way and changing the internal hash function is properly supported.

Fildes and Charles explained that "Most companies evaluate the productivity of their inventories through such yardsticks as inventory turn, gross margin return on investment, gross margin return on square foot and the like. These are all valuable tools in assessing inventory productivity, but they are all limited by the fact that they use inventory at cost as the cost basis in their analysis. The true cost of inventory extends far beyond just inventory at cost or the cost of goods sold. The cost of managing and maintaining inventory is a significant expense in its own right, but the true cost of inventory doesn't even stop there. The full cost of inventory, in fact, is actually buried deep within a number of expense items below the gross margin line, almost defying any executive, manager or cost accountant to pull them out, quantify and actually manage them" [3].

Organizations that arrangement with seaward providers sees stock levels in an alternate way. When managing seaward providers, organizations need to consider the travel time. A shipment from China requires 30 - 45 days from the time it leaves the dock in China until the point that it touches base at the dock in the United States. The 30 to 45 days incorporates the time that it takes to get the watercraft over the sea, the time required to dear traditions, and the travel time to convey the item from the port to the end client. Realizing that there is a 30 - 45 day lead time, organizations need to hold in any event that much stock. Sometimes, organizations will hold 60 days of stock in the event that some disastrous occasion happens, similar to the vessel sinks, or the cargo is stolen. Organizations that hold abundance stock consequently experience stock conveying costs. Keeping in mind the end goal to figure stock conveying costs, an organization should first comprehend what is incorporated into the condition. As indicated by Tersine (1982), conveying costs incorporate capital cost, charges, protection, taking care of, capacity, shrinkage, out of date quality, and weakening. The standard scope of yearly holding cost is 20 - 40% of the stock venture [4].

LaMacchia (2004) additionally revealed that the cost to store stock for the most part runs 20-40% of the cost of merchandise sold (COGS) every year. Organizations need to comprehend the effects of holding abundance stock and start intends to lessen stock. Having exact stock records can enable associations to better comprehend stock needs, which will take into consideration diminished stock levels.

Williams (2003) announced that a key component that isolates fruitful organizations from organizations that are not as beneficial is the means by which stock levels are resolved. An organization should first pick up a comprehension of where and how much stock is in the framework to viably decide fitting levels of stock expected to support generation and not have overabundance stock. This information can be gotten by breaking down what kind of interest the organization is encountering [8].

Golle and Staddon (2004) presented Secure Conjunctive Keyword Search Over Encrypted Data [7]. A user stores an encrypted documents like e-mails, these types of settings on untrusted servers are done by them. So by this, for retrieving a specific document user needs to satisfy particular search criterion. After successful matching, user gives a server an authority that permits the server to spot those specific documents. Work here has generally fixated on look criteria comprising of a one watchword. On the off chance that the client is really keen on reports containing everything about watchwords (conjunctive catchphrase seek) the client should either offer the server abilities for everything about watchwords independently and depend on a crossing point count (by either the server or the client) to decide the right arrangement of archives, or on the other hand, the client may store additional data on the server to encourage such hunts.

CHAPTER 3

PROBLEM STATEMENT

In a typical NSX deployment, software assets such as virtual machines and virtual network interfaces of a distributed set of ESXi / KVM hosts need to be discovered, collated and persisted.

In traditional on premise deployments, as the scale of these host elements grow, it is impractical to persist all the attributes of the interested asset set on the NSX manager. It is more efficient to hold some of the attributes in memory and fetch them again in case the NSX manager crashes and recovers. The NSX manager itself is deployed in a high availability configuration with shared storage, thereby reducing the risk of loss of data. This product plans to support scenarios where these assumptions of re-populating the entire data set may not be cost effective in future. Take for instance a cross cloud / hybrid cloud deployment scenario, where the NSX Manager is deployed on the premises of an enterprise customer, which is managing SDN use cases for both on premise and public cloud workloads. In such a situation, inventory data must be collected from the public cloud environment and sent over WAN links to the on premise NSX manager. The product must evolve, to minimize the need for re-populating host inventory information as it encounters various faults.

The aim is to find the current cost of maintaining inventory information for a typical NSX deployment and propose a solution to effectively reduce the inventory delta exchange.

CHAPTER 4

EXISTING SYSTEM

In cloud environment whether it is on premise or public cloud there is a need to send inventory data related to the objects that already exists. This is done because it is not possible to save the entire information on premises. For this entire data set, which includes virtual machines and network interface information is sent to the public cloud.

First the connection is established between client and server. Here the client is the ESXi host and server is AWS cloud. DA which sits on ESXi host will send SYNC_INIT_REQ to MP through MPA when there is connection between MP and MPA. When MP is ready to receive full sync, it will send SYNC_INIT. At this point entire host inventory information is sent to MP.

The inventory state information keeps on changing from time to time. Sometimes new objects are added or existing objects are updated. Multiple hosts are responsible for making these changes. Thus the inventory information on cloud needs to be updated from time to time. So again entire data will have to be sent.

Also there may be a situation where host goes down or connection breaks out. In these situations also there is a no way to find out how much data has been already sent. So again entire data has to be sent. This repopulation of entire data set again and again is not cost effective. So there is need to find out a way, which reduce the amount of data being sent across cloud.

Data Collector creates DcMessage and posts it to message queue. Data Sender consumes DcMessages and encapsulates Inventory Object into IdaMessage. Manager Node consumes the IdaMessage sent to it over RabbitMQ. Figure 4.1 shows the working of existing protocol.

IdaMessage contains three things:

- Message Type

- Inventory Object
- Ida command

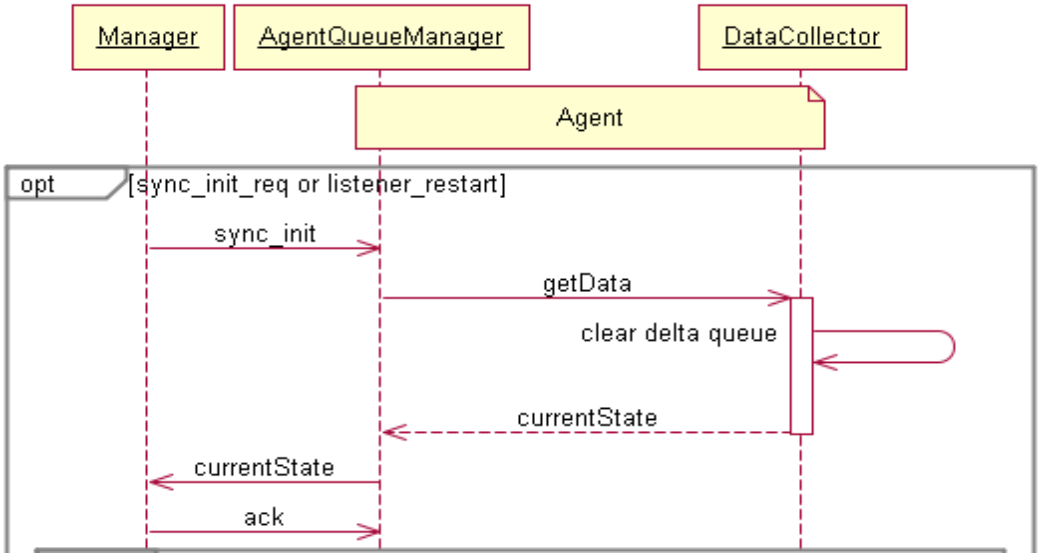

Figure 4.1: Existing system

Message Type can be of some types like:

- MT_INIT
- MT_UPDATES
- MT_COMMAND

Inventory Objects contain object type (Host, VM, VNI), Change type (CT_CREATE, CT_MODIFY, CT_DELETE), Hosts, VMs and VNIs.

Ida command can be:

- SYNC_INIT_REQ: When DA communicates with MP, it sends this message.
- SYNC_INIT: MP reverts back for requesting full sync data
- ACK: After getting full sync data. This ACK is sent by MP to DA.

DCMessages are internal to IDA and not exposed. It contains DcMessage Type and Inventory Objects information.

DcMessageType can be of different types:

- DMT_INIT_START: This means something went wrong because of which full sync needs to be done. So it changes state to WAITING. DS will raise again SYNC_INIT_REQ.
- DMT_INIT
- DMT_INIT_END
- DMT_UPDATE: DS will send only the updates. Here all the updates are sent one by one. Before popping messages it will peek into the queue. If it finds DMT_INIT_START at any point it will break because something went wrong. Otherwise it will go ahead with sending updates to MP.

There are some full Sync scenarios (Fault Scenarios) which trigger re-populating entire information. And the cases are:

1. Hardware and software Refresh:

   As the innovation develops, there is noteworthy change in server's life expectancy, yet even with different choices accessible for expanding the server's life expectancy; there is still need to buy equipment. There comes a point, when server equipment is essentially excessively old, making it impossible to perform and again question is when might you know when it's truly time. To think about the new equipment buy, the key is to distinguish that, when expanded execution, vitality productivity prerequisites will legitimize another equipment buy [3].

2. VM Migration:

   Conversion from physical server to virtual server (VM) is called as physical to virtual migration. For performing physical to virtual machine migration, we need to copy the physical server’s bits to virtual disk, install drivers and modify the guest to support these drives.

   Workload keeps changing in servers. There is always a sudden spike of requests i.e. once in a month. At that time, it is not possible to provision additional servers manually.

3. Unplanned datacenter downtime:
    - Human error and poor infrastructure capacity management.
    - Poor maintenance and lifecycle strategy.
    - Substandard data center site selection and risk mitigation.
    - Hardware/Server failure
    - Servers which are presented in the datacenters are always running and continuously heating up, this is the reason for h/w failure. There are some more reasons for equipment failure, like if temperature increases from its fixed value than it decreases the server's performance and might be reason for datacenter failure.

4. Power Outage:

   The reason for power outage is uninterrupted power supply (UPS). It can happen from battery failure or from the excessive power draw which reaches beyond the predefined limits.

5. Host Goes Down:

   The reason of host breakage is the network traffic between the host servers and SAN (Storage area network which is the location of VM file).

CHAPTER 5

METHODOLOGY

Client Server Architecture has been used to test the proposed solution. It comprises of following components.

5.1File Reader/Writer: Figure 5.1 shows the workflow for performing CRUD operations on a file.

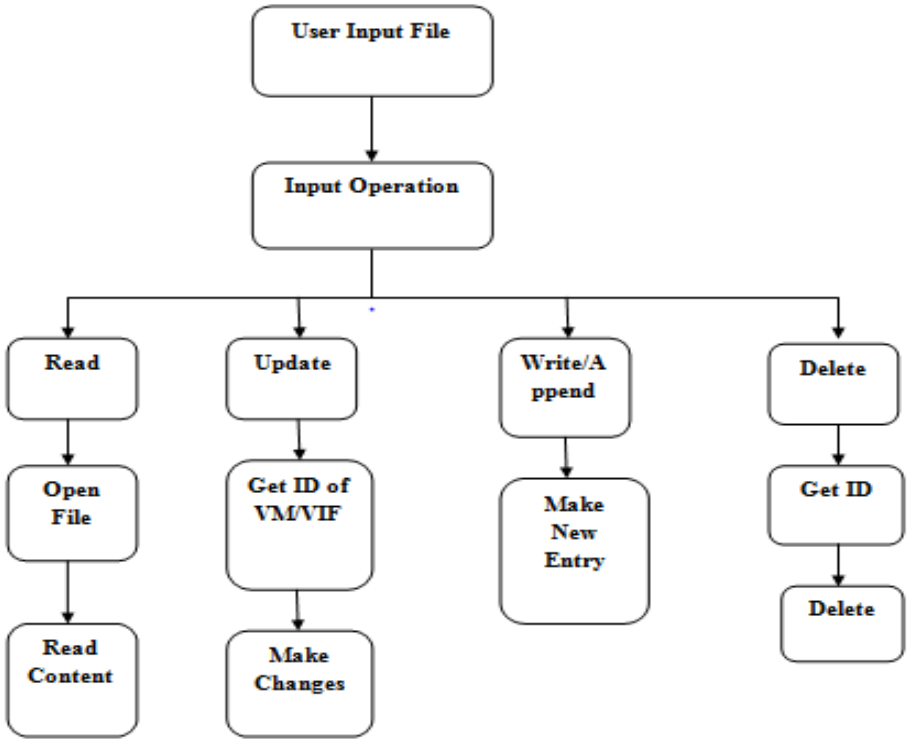

Figure 5.1: Workflow for performing CRUD operations

Various operations can be performed on the file that contains inventory information. The main operations are CRUD (Create, Read, Update, Delete).

- Create: It is basically a write operation which is used to make new entry about VM/VIFs.

- Read: Open an existing file and read the content.
- Update: For performing update operation, get the ID of VM/VIFs and make changes according to that ID. External ID is unique for each VM and VIF.
- Delete: For performing delete operation, get the ID of VM/VIFs and delete the particular entry corresponding to that id.

Once the update is requested the changes are reflected on the original file. But along with that requested updates are saved in separate file so that only those can be sent to the server.

5.2 Checksum

Once the file is updated it is sent to checksum class. MD5 algorithm is used to calculate the checksum of file. This checksum is stored in a file along with the update.

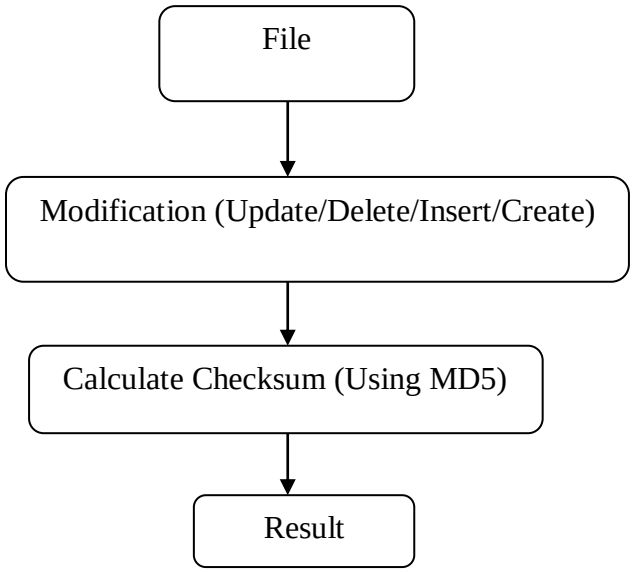

Figure 5.2: Flow Chart for checksum

This is later used to keep track of the data already sent to server. Same algorithm is used on the server side to compare the file present on it.

5.3 Http Client/Server:

Figure 5.3 shows client server system using http protocol.

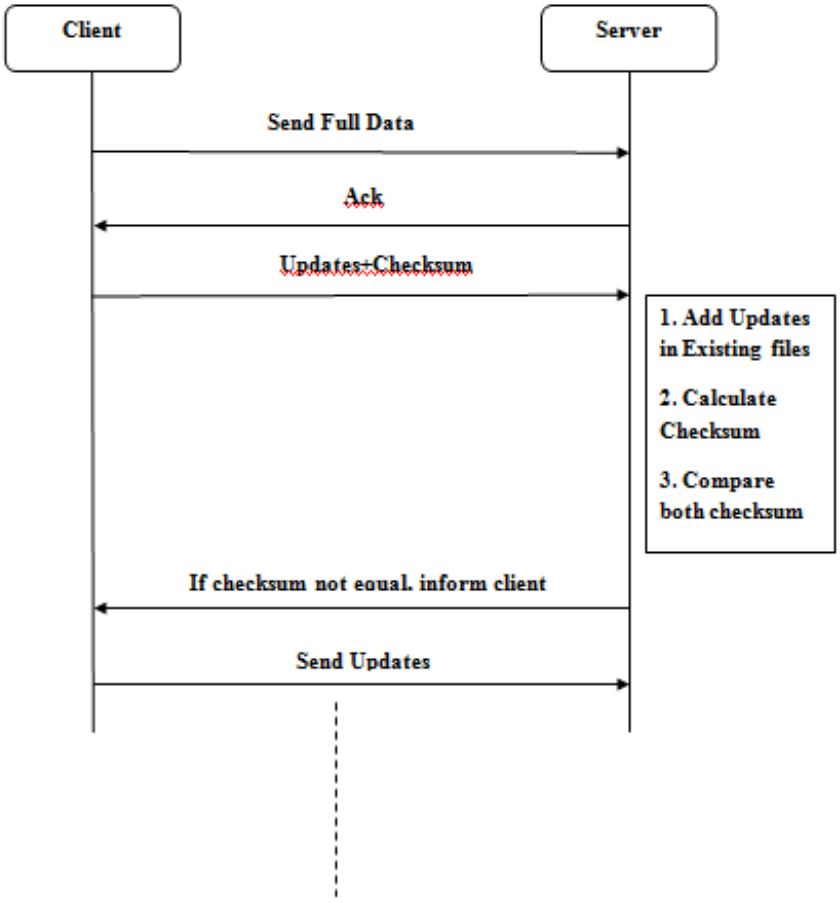

Figure 5.3: Client server diagram

If there is no file on server, send full data from client. After that whenever any update is made on client it will send checksum of latest update performed. On server side calculate checksum of the file present on it. If the checksum received from client matches the calculated value, it implies that server is up to date, there is no need to send any data and server will send success response to client. If the value does not match, server will respond with its checksum. The client maintains a file, which contains all the checksum values in sequential order of the timestamps with corresponding update file. It will start comparing the value sent by server to it and the point where it matches some previous checksum value, client get to know how much data has been successfully received by server and it will send only rest of the updates.

CHAPTER 6

IMPLEMENTATION

6.1 Functional Overview

Inventory services normally depend on compute managers such as VC to collect information about objects in the compute stack. It needs to access some basic objects and properties. The objects that are identified are:

1. Host
2. VM
3. VIF

While the data collection cannot be done through compute managers, the idea is somehow have this data collected and provide this information through the standard inventory interfaces.

Following use-cases have been identified.

1. Admin should be able to add hosts to the inventory using NSX Manager API. They should be able to update or delete the said hosts as well.
2. Inventory should have data about VM and VIFs on the hosts.
3. UI needs information about the Mac address of the VIF, name of the VM and its location (ie. Host). Figure 6.1 shows the inventory services.

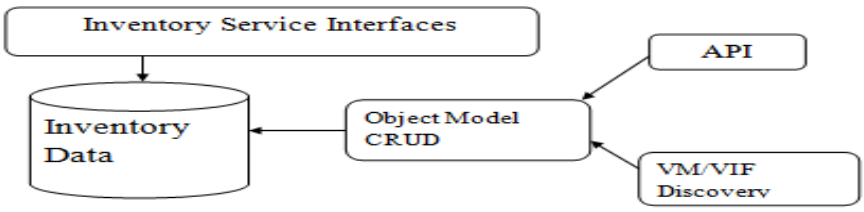

Figure 6.1: Inventory Services

6.2 Object Model

The objects available in the inventory are:

- Hypervisor Info:
    - Id (Copy of External id): Used as an identifier by NSX.
    - External id (Optional): As entered by admin. This is the UUID of the host as per the information on the host. Both ESX and KVM have this property. ESXi and KVM require this.
    - Display name (Optional): As entered by admin.
    - List<IP address of the host> (Ipv4/Ipv6): As entered by admin- used to identify the host in our system.
    - Type: ESXi or KVM.
    - Version: version of the software running on the node such as ESXi version or KVM version.
- Host:
    - Combine information from hypervisor info and Messaging Client Info (a shared object between different message clients).
- VM:
    - Id (generated id): Generated internally by NSX.
    - External Id: This will be the combination of BIOS UUID and path of the VM. This is required to identify the VM in our system. Discovery agent will provide this.
    - Display name (Optional): As discovered by the agent.
    - Type: defaults to regular but edge vertical can mark the VM as an Edge VM.
    - Host id
    - List <VIF.id>
- VIF:
    - Id (generated id): As discovered by agent and same as external id.

- External id: Combination of VM id and device key (ESXi) or device alias (ESXi).
- Vm id
- Host id
- List<MAC address>: as provided by admin or collected from discovery agent.

6.3 Display Name:

For host, this depends on whether user has entered and display name.

For VMs, this is discovered through a host agent.

For VIF, this is the combination of VM name and nic ordinal number or VIF id

6.4 Proposed Solution:

In order to solve the above addressed problem, there has to be some checkpoint which keeps track of the data that has been already sent. The proposed solution is to calculate checksum of data file. Initially the entire data file is sent. After that for all the updates being made on data file checksum is calculated and all the updates are stored in separate files. A separate file is maintained that contains all update files names and their corresponding checksums. Every time some update is made the latest checksum is sent. This checksum is compared with the checksum of file present on cloud. If this checksum matches it means file is up to date. If it does not match the checksum of file present on cloud is sent and it is compared to all the checksums and the point where it matches from then onwards the updates are sent. This eliminates the need to send entire data.

DA (Discovery Agent) which sits on ESXi host will send SYNC_INIT_REQ to MP (management plane) through MPA (management plane agent) when there is connection between MP and MPA. When MP is ready to receive full sync, it will send SYNC_INIT. At this point entire host inventory information is sent to MP with the calculated checksum. Now onwards only checksums are being transferred rather than whole data. If checksum matches that means there is no need to send any data. If not, then cloud's side calculated checksum is being transferred to the MPA for checksum comparison and this

process continues until whole data reaches cloud. Now there is no need to send whole file again and again.

Figure 6.2 shows the working of proposed system.

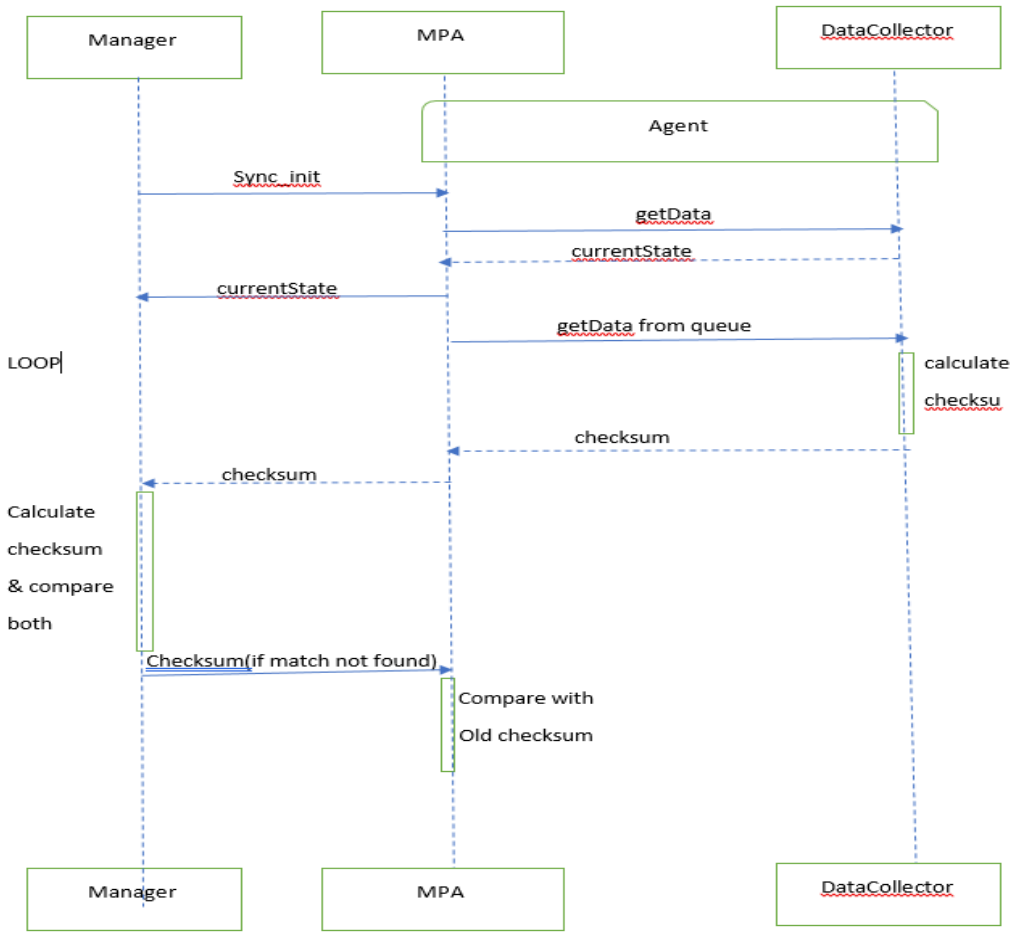

Figure 6.2: Proposed System view

6.5 Experimentation:

1. First make a client-server connection. For sending data, server should be started.
2. Initially there is no file on server so send original file to it. Hence client and server have same file.

   Figure 6.3 shows the vif.json file which needs to send on server side.

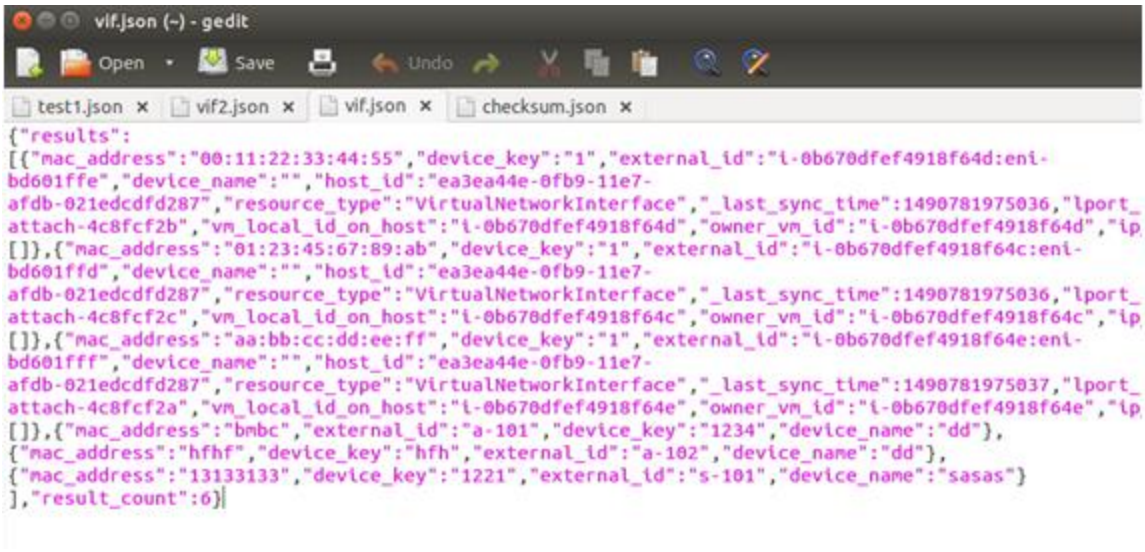

Figure 6.3: Vif.json

3. Whenever any update (Add, Update, Delete) is performed at client side a file is maintained on client side, which contains only the updates.

   Figure 6.4 shows the test1.json file which is the operation that is performed by client.

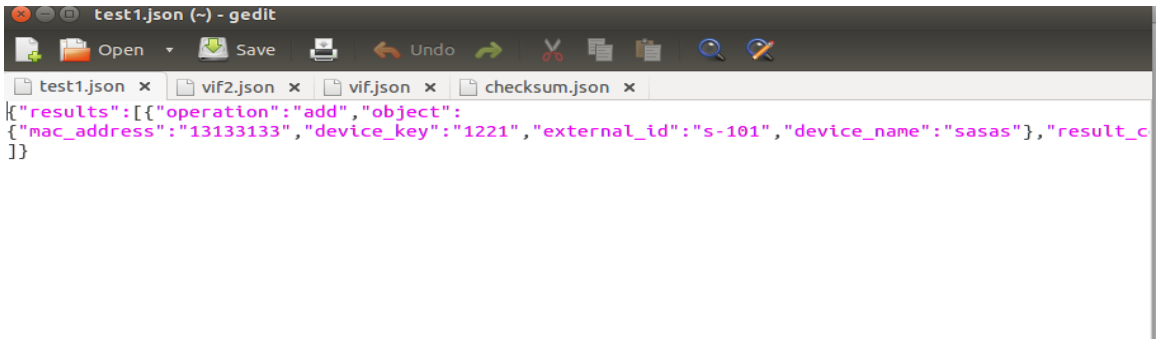

Figure 6.4: test1.json

4. A file is maintained on client side containing checksum of file along with corresponding update file. Here vif.json file is already there on client side and remaining are updated file's checksum. Updated file contains new operation which is performed by clients.

   Figure 6.5 shows the checksum.json file that is the calculated checksums by md5 algorithm.

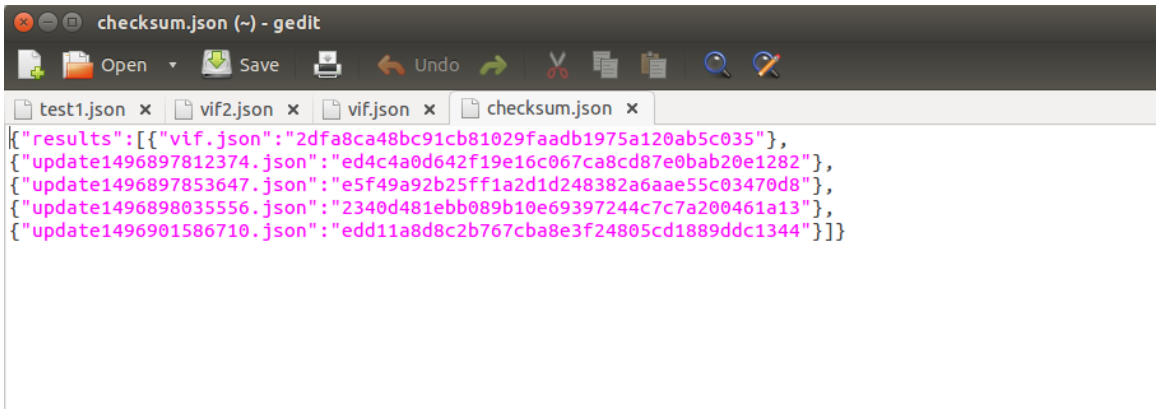

Figure 6.5: checksum.json

5. Client will send the checksum of updated file to server and server will receive that checksum and will match this with the checksum of recent file available on server side. Figure 6.6 shows the client's output after giving new operation's updates.

Figure 6.6: Client's Output

6. If both checksums (Calculated on server and client) are equal than no need to send updated file from client to server. Server will respond with a success message.

7. In case of different checksums, server will inform the client that it has this last updated checksum and want checksums after this one. Client needs to check from file which checksum (presented on client side) it matches, and send the remaining

updates to server side. Those updates are then applied to file present on server side. Figure 6.7 shows the updated file after performing new operations.

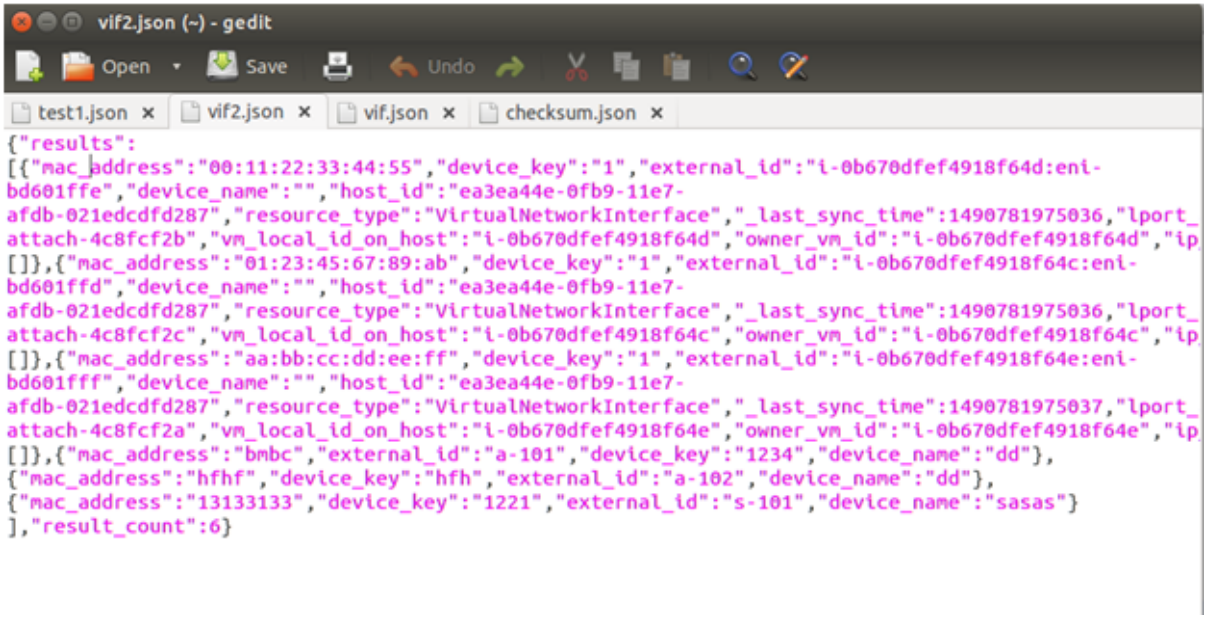

Figure 6.7: Updated File (vif2.json)

These are the solutions for the scenarios where current protocol fails and needs to send full data again and again to the cloud.

- Hardware/Server Refresh:

  Sometimes, a server equipment refresh can unravel execution issues at the part of the cost of another server. Including CPUs or memory can essentially expand server execution. However all the hardware cannot be fixed by up gradation because not all the hardware is upgradable. There is always a need for new purchase.

- VM Migration:

  Virtual machine migration accommodates for changing workloads automatically. It can manage to both additional workload and reduced workload. For the users of servers, a scheduled maintenance normally results in some downtime. Virtual migration helps to migrate virtual machines from one host to another and brought

back to the original server when migration is completed. If there is a case of unscheduled server downtime, virtual machine migration can also migrate from one host to another host. (This is the case of any random server fault). For disaster recovery, virtual machine migration can also be used. This process involves setti0ng up of similar resources in a disaster recovery site with high speed Wide Area Network links and specializes network connectivity equipment. Between primary server and the servers in the data recovery site, the virtual machines and their array Comparative to migration of operating systems and applications, migration of virtual machine is quite effortless process.

With the help of virtual machine migration, it is possible to migrate operating system from older server to newer one easily and without disrupting the services [4].

- Unplanned Datacenter Downtime:

  This can be managed by buying better quality hardware from the known vendors.

- Power Outage:

  Sometimes, a server equipment refresh can unravel execution issues at the part of the cost of another server. Including CPUs or memory can essentially expand server execution. Batteries should be tested at least quarterly for verification of correctness.

  Cyber-attacks may be a reason for power outage. CRAC failure has also become increasing common as densities rise.

- Host Goes Down:

  The suggested solution for host failure is VMware high availability (HA). It helps by gathering all the VMs and hosts on which VMs present. HA first checks for failed host and then it can restart a VM which is running on a failed host. HA can also check by checking the VMware tools which are running on it. When any ESX/ESXi host (Host on which VMs are running) fails for any reason than VMs can also fail. VMware HA ensures that VMs which are running on failed host can

run on another host (Means they should be capable of being restarted). ESX/ESXi host failure is primarily dealt by the VMware high availability and what happens with the virtual machines that are running on the host. HA doesn't mean fault tolerance. These are two different terms, HA checks availability for VMs which failed because of host failed and fault tolerance focuses on making system more fault tolerance.

6.6 Scale Targets:

Scale targets for Avalanche are:

- 25K VMs
- 25-50K VIFs
- 5K Hosts

6.7 Backup and Restore:

There is no impact of backup/restore on inventory for the following reasons.

- Inventory data is ephemeral. When the cluster restarts, the agents will connect and give the latest data to the management plane as part of sync operation.
- Inventory agent uses MPA to connect to management plane. Any changes on management plane that was lost as part of restoration should not affect inventory agent directly. They might affect MPA.
- The persistent data of inventory will be backed up and restored as part of the regular backup and restore operation, and what is lost is obviously lost same as other verticals. There is no special case for inventory identified.

CHAPTER 7

RESULTS AND DISCUSSION

A client server application was developed. The client side performs all the CRUD operations and stores the checksum of files. Whenever some update is performed the data goes from client to server. The server side will store the data sent by the client. The size of the data being sent from client to server was calculated. The results were calculated for sending the entire data and for sending only the updates. This result was calculated for 512 hosts having 10000 virtual machines and 50000 network interfaces.

The size is calculated using formula:

$$Size = ((((VIFs*VIFs\ size)+VM\ size)*VMs)+Host\ size)*Hosts)/1024$$

The results are described below.

7.1 Existing System:

Table 1: Size of message for existing system

Object	**Number of objects**	**Size/object(in KB)**
VM	10000	0.24
VIF	50000	0.38
HOST	512	35

Total size of message is 17958 KB (Approx.)

7.2 Proposed System:

Size of message for a proposed system is calculated here.

Table7.2: Size of message for proposed system

Object	Number of objects	Size/object(in KB)
VM	10000	0.02
VIF	50000	0.04
HOST	512	3

Total size of message is 1536 KB (Approx.)

The size of message reduced by 99.9 %. This result is very significant. As the amount of updates increase the size may increase a bit but still it is very effective. It will reduce the bandwidth cost of moving the data to the cloud. So there would not be a need to pay high expenses of sending large amount of data over cloud.

7.3 Discussion:

In existing system, whole data needs to be sent again and again whenever any crash happens because of the hardware failure, power outage etc. This increases the cost of whole data transfer. Whereas in proposed system, only updates are being sent that reduces the cost of whole transfer and amount of data. This system only saves updated file's checksum instead of whole data file and that checksum needs to be maintained on both sides i.e. Server and Client side. In case of proposed system, whenever any crash happens, server and client both already have the last checksum details stored in a file. That forms the reason of sending updates only instead of full sync.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In this report, inventory system was analyzed. The main problem found was repopulation of the inventory data. So various fault scenarios were identified which lead to full data sync.

For performing experiment client server application was developed that was similar to the actual cross-cloud structure. The size of message has been calculated for sending the data in cross-cloud environment. When updates have been sent to the server, there is significant difference in the amount of data transfer according to proposed system. It can further reduce the cost of sending the data to the cloud.

8.1 Limitations:

The proposed system has the following limitations:

- Only a limited set of objects and properties are supported in Avalanche.
- The data from host may not be immediately available after a cluster restart, since at max scale, we would access data from hosts in batches to avoid flooding the message bus.
- A newly added host may not have all data discovered immediately since other hosts might be queued up already for the first sync.

8.2 Future Scope:

While doing this work, MD5 algorithm was used to calculate checksum of files. The main aim was to reduce the amount the data, so simple algorithm was used to find checksum. This work can be further improved by using more efficient algorithm to do calculation.

This solution works for same checksum algorithm used on both the sides that is the client and the server. This makes system hardly coupled. In future it can be extended for

different development platforms having no condition of using same algorithm on both the sides.

The information that was sent to server was stored in similar way as it was present on client. Therefore it was easy to apply updates. But there may be situation where data on client and server side cannot be stored in same format. So in this case data can be serialized on both the sides.

REFERENCES

[1] S. K. Majhi and S. K. Dhal. " Placement of Security Devices in Cloud Data Centre Network: Analysis and Implementation." *Procedia Computer Science*, vol. 78, pp. 33-39, 2016.

[2] H. Krawczyk, R. Canetti and M. Bellare. "HMAC: Keyed-hashing for message authentication.", *RFC 2104*, 1997.

[3] R. Fildes and C. Beard. "Forecasting systems for production and inventory control." *International Journal of Operations & Production Management* vol. 12.5, pp. 4-27, 1992.

[4] Y. Li, X. Xu, and F. Ye. "Research on supply chain inventory optimization and benefit coordination with controllable lead time." *Wireless Communications, Networking and Mobile Computing*, 2007.

[5] R.O. Gupta and T. Champaneria. "A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment." *International Journal of Advanced Research in Computer Science and Software Engineering*, 2013.

[6] S. Zhang and X. Chen. "Cloud computing research and development trend." *Future Networks, 2010. Second International Conference on* IEEE, 2010.

[7] P. Golle, J. Staddon and B. Waters. "Secure conjunctive keyword search over encrypted data." ACNS, vol. 4, 2004.

[8] A. Gunasekaran, H. J. Williams and R. E. McGaughey. "Performance measurement and costing system in new enterprise." *Technovation* vol. 25.5, pp. 523-533, 2005.

[9] C. Gong, J Liu, Q. Zhang and H. Chen. "The characteristics of cloud computing." *Parallel Processing Workshops (ICPPW), 39th International*

Conference on IEEE, 2010.

[10] F. Chen and A. X. Liu. “Privacy and Integrity Preserving Range Queries in Sensor Networks”. *Networking, IEEE/ACM Transactions on vol. 20, 2010.*

[11] S. L. Anil, and R. Thanka. "A survey on security of data outsourcing in cloud." *International Journal of Scientific and Research Publications on vol.* 3, 2013.

[12] R. Fildes, and C. Beard. "Forecasting systems for production and inventory control." *International Journal of Operations & Production Management* vol. 12.5 pp. 4-27, 1992

[13] D. S. A. Elminaam and H. M. Abdual-Kader and M. M. Hadhoud. "Evaluating the performance of symmetric encryption algorithms." *IJ Network Security* vol. 10.3, pp. 216-222, 2010.

[14] W. Stallings and M. P. Tahiliani. *Cryptography and network security: principles and practice*. vol. 6, 2014.

[15] M. Mathur and A. Kesarwani. "Comparison between Des, 3des, Rc2, Rc6, Blowfish And Aes." *Proceedings of National Conference on New Horizons in IT-NCNHIT*. vol. 3, 2013.

[16] M. L. Rupley Jr "Introduction to query processing and optimization." *Indiana University*, 2008.

[17] M. Alhanjouri, and A. M. Al Derawi. "A New Method of Query over Encrypted Data in Database using Hash Map." *International Journal of Computer Applications* vol. 41.4, 2012.

[18] Mrs.S.Selvarani, Dr.G.Sudha Sadhasivam. “Improved cost-based algorithm for task scheduling in Cloud computing”, *Computational Intelligence and Computing Research, international conference on* IEEE, 2010.

[19] A. Chatterjee. "Concurrent error detection and fault-tolerance in linear analog

circuits using continuous checksums." *IEEE Transactions on very large scale Integration (VLSI) Systems* vol. 1.2 pp. 138-150, 1993.

[20] R. T. Braden, D. A. Borman, and C. Partridge. "Computing the internet checksum", *RFC 1071*, 1988.

[21] Cui Lin and Shiyong Lu. "Scheduling Scientific Workflows Elastically for Cloud Computing", *4th International Conference on Cloud Computing on IEEE, 2011.*

[22] H. Hacigümüş, B. lyer and C. Li. "Executing SQL over encrypted data in the database-service-provider model." *ACM SIGMOD International Conference on Management of Data.* ACM, 2002.

PUBLICATION

S. Garg, A. “Finding Cost Effective NSX Inventory State Management in Cross.” *International Conference on Industrial Electronics and Computer Science*, 2017. (**Communicated**).

VIDEO PRESENTATION LINK

- https://youtu.be/N7qon_hWV28