

BDB Tasks related to UCS

A PROJECT REPORT

submitted by

CB.EN.U4CSE11439

RAHUL E

under the guidance of Mrs. Padmavathi

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING



AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641112

JULY 2015

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, COIMBATORE, 641112



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “**BDB Tasks related to UCS**”
submitted by

CB.EN.U4CSE11439

RAHUL E

In partial fulfilment of the requirements for the award of the degree **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** is a bonafide record of the work carried out under our guidance and supervision at Amrita School of Engineering, Coimbatore and CISCO Systems, Bangalore.

Mr. Shankar Prasath
TAC Engineer, CISCO

Mrs. Padmavathi
Assistant Professor,
Department of CSE

Dr. Latha Parameswaran
Chairperson, Department of CSE

This project report was evaluated by us on

INTERNAL EXAMINER

EXTERNAL EXAMINER



Dated: 29th June 2015

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Rahul E** pursuing his fourth year in Bachelor of Engineering in Computer Science and Engineering at Amrita School of Engineering , Coimbatore, has completed his eighth semester project and internship from 15th January 2015 to 30th June 2015 at Cisco Systems(India) Private Limited with the project titled “**BDB Tasks related to UCS**” under the guidance of **Mr. Shankar Prasath, TAC engineer** and **Mr. Padmanabhan Ramaswamy, Software engineer, Cisco systems Cisco Systems (India) Private Limited.**

Mr. Vipin Thomas
Program Manager,
Cisco TAC

ACKNOWLEDGEMENT

An endeavor over a period can be successful only with the support and advice of well-wishers. It is with hearty gratitude that I take this opportunity to acknowledge their contributions to my project.

Firstly, I would like to thank **Amma** for always being there and blessing us with her unfathomable love and affection.

I express my sincere gratitude to **Brahmachari Abhayamrita Chaitanya**, Pro Chancellor and **Dr.P.Venkat Rangan**, Vice Chancellor and **Dr. Sasangan Ramanathan**, Dean Engineering of Amrita Vishwa Vidyapeetham, for providing me the opportunity to undergo this programme.

I express my sincere thanks to **Prof. Dr.Latha Parameswaran**, Chairperson, Department of Computer Science and Engineering, for her support and encouragement. I am indebted to our university, **Amrita Vishwa Vidyapeetham**, for providing us state-of-the-art facilities, and an impeccable environment.

I express my profound and sincere gratitude to my internal guide, **Mrs. Padmavathi**, Assistant Professor, Dept. of CSE for her guidance and suggestions for the improvement and implementation of the project.

I am very grateful to **Cisco Systems** to have given me the opportunity to intern at their organization and for provision of expertise, and technical support in the implementation. I express my gratitude to my mentors, **Mr. Shankar Prasath**, TAC engineer and **Mr. Padmanabhan Ramaswamy**, Software engineer, Cisco Systems for their encouragement and for the superior knowledge and experience. The project would not have been successful without their help.

Lastly, I would also like to thank my family, friends and all my teachers for encouraging me throughout and providing the valuable suggestions to improve my project.

ABSTRACT

The Cisco Unified Computing System (UCS) is a data center server platform composed of computing hardware, virtualization support, switching fabric, and management software. Management of the system devices is handled by the Cisco UCS Manager software which is accessed by the administrator through a common browser such as Internet Explorer or Firefox. Through the UCSM, we can monitor the devices and also modify the configurations. In addition, we can download tech support files which contain detailed information regarding various components.

However, the size of these tech support files is huge. Manually extracting and inspecting the log files is very time consuming and cumbersome. The aim of this project is to automate this process. We can parse these logs to extract only the relevant information and present it in an easy to read format.

This saves a Cisco engineer or anyone who wants to decipher the log files a lot of time and enables faster troubleshooting of the UCS.

TABLE OF CONTENTS

Chapter	Title	Page No.
	Acknowledgement	4
	Abstract	5
	List of Figures	9
	List of Tables	11
	List of Abbreviations	12
1	Introduction	13
1.1	Overview on UCS	14
1.2	Motivation of automation of Log parser	14
1.3	Problem statement	15
1.4	The current solutions of UCSM logs	15
1.5	Objectives and Scope of the Project	15
1.6	Methodolgy	19
1.7	Organisation of Project Report	17
2	Fundamentals of Cisco UCS	18
2.1	Virtualisation	18
2.1.1	The various levels of Virtualisation	19
2.1.2	Advantages of Virtualisation	20
2.2	Hypervisor	21
2.2.1	Types of Hypervisors	21
2.3	ESXi Hypervisor	23
2.3.1	Architecture of ESXi	23

2.4	VMware vCenter Server	24
3	Introduction of Fabric Interconnect	25
3.1	Cisco Unified Computing System	25
3.2	Cisco Unified Computing System Components	26
3.2.1	Fabric Interconnect	26
3.2.2	Cisco Fabric Extenders Module	27
3.2.3	Cisco UCS Chassis	28
3.2.4	Cisco UCS B200 M1 Blade Server	29
3.2.5	Intel Xeon 5500 Series Processor	30
3.2.6	Cisco UCS M71KR-Q Converged Network Adapter	30
3.2.7	Cisco UCS Virtual Interface Card	31
3.2.8	Cisco UCS C-Series Rack Mount Servers	32
4	Introduction to Service Profiles	33
4.1	Cisco UCS Service Profiles	33
4.1.1	Redefining the Server and the Data Center	33
4.1.2	Virtual Servers: The First Abstraction	35
4.2	Data Center Integration	36
4.3	Cisco Unified Computing System Overview	37
4.4	Service Profiles: Cisco Unified Computing System Foundation technology	38
4.4.1	What is a service profile	38
4.4.2	Service Profiles and Templates	39
4.5	Conclusion	39
5	Fundamentals of SAN and UCS Manager	40
5.1	Storage Area Network	40
5.2	UCS Manager	41
6	Design of Log parser	43
6.1	Project Description and Goals	43

6.2	Technical Specification	45
6.3	Design Approach and Details	46
6.3.1	Design Approach	46
6.4	Algorithm	53
7	Implementation on Cisco BDB	55
7.1	Parse the required information	55
7.2	Using the Script	55
7.3	View Parsed Information	57
7.4	Codes and Standards	57
7.5	Constraints and Tradeoffs	58
7.6	Schedule , Tasks and Milestones	59
8	Results , Discussion and Inference	60
8.1	Project Demonstration	60
8.2	Marketing and Cost Analysis	61
8.2.1	Marketing Analysis	61
8.2.2	Cost Analysis	61
8.3	Outputs	62
8.4	Summary	68
9	Testing and Bug Report	69
10	Conclusion and Future Scope	76
	References	77

LIST OF FIGURES

Figure No.	Title of the Figure	Page No.
2.1	Distinction between the architectural layout of type-1 and type-2 hypervisors	22
2.2	Architecture of ESXi hypervisor	23
2.3	Pictorial description of vCenter Server	24
3.1	Cisco Unified Computing System	26
3.2	Cisco UCS 6120XP 20-Port Fabric Interconnect and Cisco UCS 6140XP 40-Port Fabric Interconnect	27
3.3	Port 1/2/4-Gbps Native Fibre Channel Expansion Module; 4-Port Fibre Channel plus 4-Port 10	27
3.4	Cisco UCS 5108 Blade Server Chassis with two Cisco UCS 2104XP Fabric Extenders	28
3.5	Cisco UCS 2104XP Fabric Extender	28
3.6	Cisco Blade Server Chassis	29
3.7	Cisco UCS B200 M1 Blade Server	29
3.8	Intel Xeon 5500 Series Processor	30
3.9	Cisco USC M71KR-Q Network Adapter	31
3.10	Cisco UCS Virtual Interface Card	31
3.11	Cisco UCS C-Series Rack Mount Server	32
4.1	Cisco Unified Computing System	37
4.2	Major elements of a service profile	38
5.1	Storage area network	41
5.2	UCS Manager	42
6.1	Architecture of UCS	43
6.2	Steps to automate the log parser through BORG engine	50

7.1	CISCO CaseKwery tool	55
7.2	Screenshot of BDB platform	56
7.3	Output of the 'ucs_service_profile' script	57
7.4	Project plan	59
8.1	Screenshot of code snippet	60
8.2	Service profile	63
8.3	ucs-FC-configuration report	65
8.4	ucs-LAN-configuration-report	67

LIST OF TABLES

Table No.	Title of Table	Page No.
6.1	Technical specifications of the UCS Fabric Interconnect	45
8.1	Provides a detailed cost analysis on the required parts for the UCS system	61

LIST OF ABBREVIATIONS

TAC:	Technical Assistance Center
UCS:	Unified Computing System
SAN:	Storage area networking
UCSM:	Unified Computing System Manager
BDB:	Big Data Broker
UCSM:	Unified Computing System Manager
TCP:	Transmission Control Protocol
IP:	Internet Protocol
L3:	Layer 3 of TCP/IP protocol suite
L2:	Layer 2 of TCP/IP protocol suite
MPLS:	Multiprotocol Label Switching
DCI:	Data Center Interconnect
ASIC:	Application Specific Integrated Circuit
NX-OS:	Nexus Operating system
ICMP:	Internet Control Message Protocol
N7K:	Nexus 7000 datacenter switch

CHAPTER 1

INTRODUCTION

Cisco Systems Inc. is the world leader in computer networking technologies. It has a strong hold on the datacenter switching market, and constantly brings about innovation in their datacenter switches. Cisco partners contact TAC if they face any issues in their production network. They also upload UCSM log files to enable the TAC engineer to detect the source of the problem. The TAC engineer then begins troubleshooting by manually inspecting the logs. It is a stressful situation because customers lose money for every second the network is down. The difficulty is compounded by the fact that the logs are huge in size. Automating this process by extracting the relevant parts of the file will help for effective and quick debugging. Analysis of time complexity and space complexity of running scripts will be an advantage to optimize the process further. The first chapter gives more insight into these technologies.

This chapter includes a brief overview on UCS technology. It also includes a section on the motivation behind the project, from which we define the problem statement. A brief discussion on the current solution to the problem is discussed. Finally, objectives and future scope of the project are detailed, along with the methodology which was adopted and the organization of the project report.

1.1 Overview on UCS

The Cisco Unified Computing System consists of a computing component which is available in two versions: the B-Series (a powered chassis and full and/or half slot blade servers), and the C-series (a standalone rack server). Further, the Cisco UCS provides virtualization by supporting several hypervisors including VMware ESX, ESXi, Microsoft Hyper-V and others. Unlike the VMware Workstation software, ESX and ESXi run directly on the system hardware without the need for

any other software (called Bare Metal), and provide the necessary hypervisor functions to host several guest operating systems (such as Windows or Linux) on the physical server. The Cisco 6100 or 6200 Series switch (called a "Fabric Interconnect") provides network connectivity for the chassis, blade servers and rack servers connected to it through 10 Gigabit and Fiber Channel over Ethernet (FCoE). The FCoE component is necessary for connection to SAN storage, since the UCS system blade servers have very little local storage capacity. Management of the system devices is handled by the Cisco UCS Manager software embedded into the 6100/6200 series Fabric Interconnect, which is accessed by the administrator through a common browser such as Internet Explorer or Firefox. The log files are usually downloaded from this software.

1.2 Motivation for automation of log parser:

Troubleshooting thousands of lines is not the way Engineers like it. Automation and user friendly interface is the best thing that can be delivered to a T-shoot engineer. This tool might not be useful for every department of Troubleshooting but is directed to help and assist Server Virtualization Department. To understand what this tool emphasizes on, one should have a decent knowledge of various technologies related to Data Center and Server Virtualization. As the outcome of this project is a tool which is proprietary to Cisco Systems, one need not be a genius to Figure out that this tool can be used only by Cisco Employees. One of the most common trouble tickets that arise are related to FC interfaces and Service Profiles. The problem that the engineers usually face is the complexity of5 searching numerous interfaces from a show tech-support command which shows all other unrelated information along with the FC configuration. So to make the view of such information overloaded output to a more relevant and clear data representation is the aim of this project.

1.3 Problem statement

The problem statement for automation of UCS logs is to automate the process by designing and implementing a script. The objective of this project is to substantially reduce the amount of time and effort needed for TAC engineers to decipher service profile information and fabric channel configuration from extremely lengthy UCSM log files (typically 100MB or more of plain text). UCSM log files usually include all the show tech commands. Unfortunately, this means that a lot of unnecessary information is included in the log files. This project aims to extract only the relevant data and present it in an easy to read format. This would reduce the time taken for the process from about 45 minutes to less than 30 seconds.

1.4 The current solution for UCSM logs

The current process for reading out the log Cisco document which outlines the set of show commands to be verified. However, in most cases, it's based on the source and destination IP addresses, vlan numbers, egress indices etc. This process takes approximately 30-45 minutes, implying that this time adds to the downtime of the datacenter, which would in turn mean a possible loss of millions of dollars for the organization.

1.5 Objectives and Scope of the project

This project aims to provide a complete troubleshooting solution for datacenters based on the cisco nexus 7k switches.

When a network goes down or doesn't function as intended, the critical parameter that is needed to be considered is "TIME". When in such situation it is very important not to waste time searching for a single line that is causing the issue out of thousands of unclear records. This project emphasizes on improving the time needed to troubleshoot the issue by providing the relevant data in a very user friendly manner reducing the complexity of searching.

1.6 Methodology

The methodology of the project is given below:

1. To check whether the given file is UCSM show tech support file.
2. To check whether the given archived file contains a C series standalone rack server. To call the program `extract_all` to recursively extract tar files and check it iteratively.
3. To identify various Service Profiles along with their templates.
4. To make a FC and LAN configuration report containing related information about Fabric Interconnect A and Fabric Interconnect B and present them in a GUI which is easy to interpret.

As this project is focused on the presentation of vague show tech-support output to a more precise and clear representation of FC records alone, it requires data parsing and data extraction. The important data that is required would be the output of the show tech-support command from the UCSM which is present on the Fabric Interconnect A & B. Data Extraction is done on the show tech-support output in which the file is parsed to get the FC and Service Profile records individually with name, attributes and values separately. Using that information the proper FC configuration lines are extracted from the running-configuration. Then the UCSM logs are parsed to get the attributes of the client. The information gathered from these two files is then compared and relevant FC attributes that match are separated and then shown in a simple web-based GUI which is in Big Data Broker. This will make the task of an engineer to check if there is anything related to FC and LAN that is causing the issue very simple.

The final output of the tool would be a simple table which shows the relevant FC records alone that are supposed to hit according to the host attributes from UCSM logs and the configured attributes.

1.7 Organization of the project report

Chapter 2 presents a theoretical overview of the Cisco UCS. It gives an understanding of FI's as a concept used in datacenter switches. It gives a thorough understanding of the architecture of the Cisco UCS

Chapter 3 includes a theoretical overview of the Cisco FI.

Chapter 4 includes a theoretical overview of the Cisco Service Profiles and the advantage of it using in it in FI's of UCS.

Chapter 5 includes a theoretical overview of the Cisco SAN architecture and UCS Manager which is basically a GUI to access the FI A and FI B which is secondary to UCS server.

Chapter 6 includes the various design methods adopted for implementing the log parser in BDB platform.

Chapter 7 consists of a section on the typical inputs to the log parser automation tool. It also gives an insight into the operation automation of log parser for BDB.

Chapter 8 covers the output of the log parser automation script which is split into 3 passes, also includes the various results produced according to the logs given by the user in BDB platform.

Chapter 9 finally includes a section on the failure testing and its corresponding results.

Chapter 10 consists of the conclusion and future scope of the project.

CHAPTER 2

FUNDAMENTALS OF CISCO UCS

This chapter presents a theoretical overview of the Cisco UCS. It gives an understanding of FI's as a concept used in datacenter switches. It gives a thorough understanding of the architecture of the Cisco UCS.

As mentioned earlier this project requires a decent knowledge of various technologies related to Server Virtualization. The following are the technologies that are mandatory to understand and use the outcome of the tool.

1. Virtualization
2. Fabric Interconnects and Service Profiles
3. SAN Networking
4. UCS Manager

2.1 Virtualisation Overview:

Definition: Virtualisation is the creation of a virtual (rather than actual) version of something, such as an operating system, a server, a storage device or network resources.

The important concept involved in Visualization is 'Virtual Resource'. Virtual resource is an illusion of resource supported by OS through use of real resource. The underlying physically resource whether it be CPU, memory, disk space or NICs are logically mapped onto the VM created. The resources allocated to the VM are under its complete control. This creates an illusion that VM is using its own resources in fact the resources are being provided to it by the physical server. A virtual machine is provided with virtual resources abstracted from the underlying hardware to act independently and could run on any OS. Since the Virtual Resource is an abstraction, the specifications of the VM could be configured with desired

specifications which could be modified later without affecting the VM operation. Also portability of the VM could be performed easily by allowing the same abstraction on other machine. The VM that are created are also called 'Guests'. The machine on which the VM are deployed is called as 'Host'.

2.1.1 The various levels of Virtualisation are:

1. Server Virtualisation:

Server virtualisation is the masking of server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. The hardware of one physical is divided into multiple isolated virtual environments which are associated to a VM using software called as 'Hypervisor'. This allows the single server to be used various applications. The guest OS could run without making any modifications. The guests can also be deployed with different OS'es because the guest is operating completely independent of the OS of host and is not aware that it is not using real hardware. The greater benefit provided by Server Virtualisation is the Cost and power savings.

2. Desktop Virtualisation:

Desktop Virtualisation is the way of separating the host's desktop from the physical machine. It is considered as an advanced form of server virtualisation where a host's desktop could be accessed not only by directly interfacing to it using keyboard, monitor and mouse but also could be accessed remotely on another desktop computer or any mobile device via a network connection. The host is also able to serve multiple guests by allowing multiple connections to it.

3. Storage Virtualisation:

Storage abstraction is the process of partitioning the physical storage disks into independent Virtual drives. The storage could be considered as array of disks which could be separated from each other and assigned to different guests. This allows the administrator to better manage the storage among various guests. If there are multiple

disks present physically they could be combined into a single logical disk also by performing RAID (Redundant Array of Inexpensive Disks) configuration.

2.1.2 Advantages of Virtualisation:

1. Reduced capital and operational :

Since the same server is used to host multiple guests, each serving a different purpose the work that was traditionally performed by physical server could be fit in by a simple abstraction of the hardware. The investment made on servers could be greatly reduced as we are utilizing the complete hardware. The power consumption would be very less since we are powering on only one server which would also take lesser effort to cool.

2. Application High Availability:

All the virtualized applications come with inbuilt features of High Availability (HA) and Fault Tolerance (FT). If a host fails it could be automatically restarted on another host with minimal downtime and data loss. If a guest fails we also have a provision to restore its previous working state also referred as Snapshot of a Virtual Machine.

3. Improved Responsiveness:

Virtualisation lets us to scale easily to meet the rapidly changing business needs thereby increasing the responsiveness.

4. Increased Productivity:

Managing the VMs can be simplified by automated tools hence reducing the time spent in management of various guests and focus more on innovation.

5. Improved disaster recovery:

Hardware abstraction capability ensures that by removing the dependency on a particular hardware vendor or server model, a disaster recovery site no longer needs to keep identical hardware on hand to match the production environment, and IT can save money by buying cheaper hardware in the disaster recovery site since it is used only in the time of failover.

2.2 Hypervisor:

Definition: A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines. Hypervisor is also called as Virtual Machine Monitor (VMM). As the name itself suggests it is used to create the VM and manage them by allocating resources. It is a software layer that sits between hardware and OSes which access the hardware and provide interfaces to share the available hardware and resources. A computer on which the hypervisor runs is called as 'HOST'. The hypervisor presents the guest OS with virtual platform to operate on and also manages the execution of Guest OS. The underlying hardware is shared among the multiple instances of the guests. The virtual machine that is created on host is called a 'GUEST'. Hypervisor is software that allows the guests with various OS'es to share the single hardware of host. Each guest appears to have its own resources like memory, processor, and storage disk all to itself but they are in fact all the resources are managed by the hypervisor and each OS is allocated what is required for their functioning by the hypervisor. It also ensures that the guest OS do not disrupt each other. The design of hypervisor is specific to particular processor architecture.

2.2.1 Types of Hypervisors:

There are two classifications of hypervisors.

1. Native or Bare-metal hypervisor:

Also referred as type-1 hypervisor, these hypervisors run directly on the host's hardware. They control the hardware and guest OS. The guest runs a process on the host. As the hypervisor runs on direct hardware or bare-metal, they are referred as Bare-metal hypervisors. The type-1 hypervisor provides better performance and flexibility as it runs as a thin layer supporting VM with hardware, this reduces the overhead in running the Hypervisor itself. VMware ESXi, Citrix Xen server, Microsoft hyper-v are some of the leading type-1 hypervisor equivalents.

2. Hosted hypervisor:

These hypervisors run as a program on the host OS just like other computer programmes run. They are also referred as type-2 hypervisors. The guest OS is abstraction of the host OS. As the guest OS is abstracted from host OS, it is always dependant on the host OS. An OS which is used as a type-2 hypervisor has to perform critical operations so it cannot devote 100% of its hardware to the VMs.

Type-1 hypervisors are used in production where the VMs created provide services to users. Type-2 hypervisors are offer a series of different services and are not used often in productions. Typically, a Type 1 hypervisors are more efficient than a Type 2 hypervisor because they devote 100% of the hardware to the VMs, yet they both could be considered similar in functionality because the same kind of VMs could be run on the both types of hypervisors. A VM running on one host server running type-1 hypervisor could be migrated to a host server running type-2 hypervisor and vice versa with some conversion required. VMware workstation and Virtual box are some of the leading type-2 hypervisor equivalents.

The pictorial description (Figure 2.1) of the Type-1 and Type-2 hypervisors helps to distinguish their architecture.

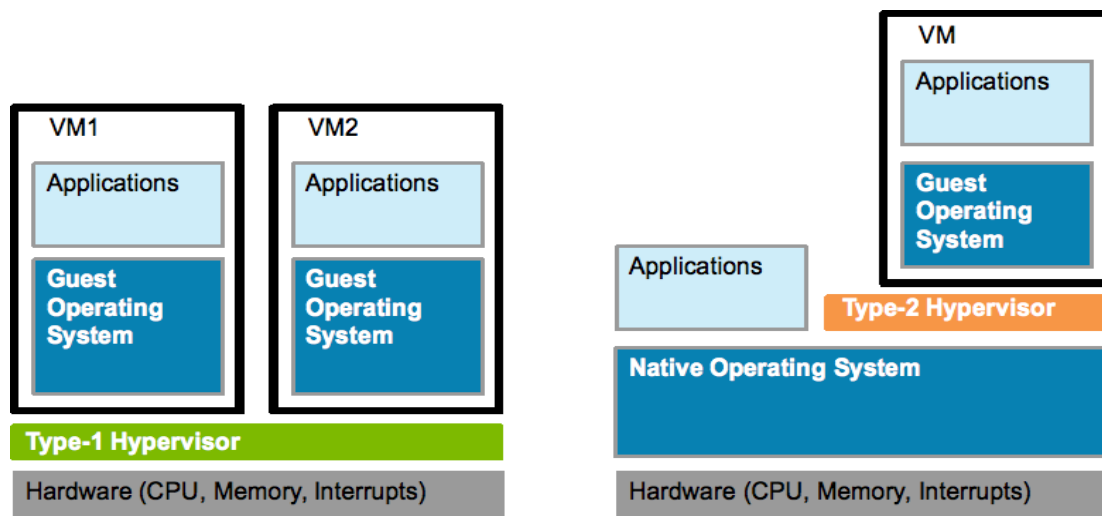


Figure 2.1 –Distinction between the architectural layout of type-1 and type-2 hypervisors.

If type-1 hypervisor is used the server is dedicated to run VMs but the type-2 hypervisor can be used to run VMs and serve other applications also.

2.3 ESXi Hypervisor:

ESXi is a type-1 hypervisor developed VMware for deploying and managing the Virtual Machines (Figure2.2). Versions above 4.1 are called ESXi and versions prior to 4.1 were called ESX. It is a primary component in the VMware infrastructure software suite. Being a type-1 hypervisor it closely manages the underlying hardware and associates it with the VM. It is not merely a software program that runs on the OS but also includes and integrates vital OS components. It includes its own kernel. It helps in virtualizing the servers so that all the application can be consolidated to run on lesser hardware. It eases administration by providing simplified maintenance.

2.3.1 Architecture of ESXi:

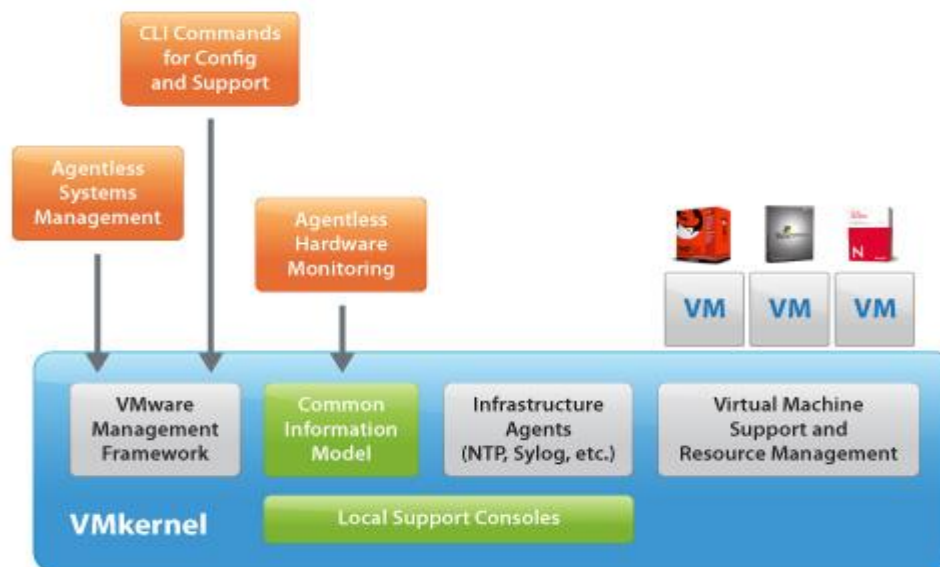


Figure2.2 – Architecture of ESXi hypervisor

2.4 VMware vCenter Server:

vCenter server provides a centralised and extensible environment for managing multiple ESXi hosts, vSphere client environments and their virtual infrastructure (Figure 2.3). vCenter Server manages vSphere® environments, giving IT administrators simple and automated control over the virtual environment to deliver infrastructure with confidence. vCenter helps in analysing and remediating the issues by providing quick and complete visibility into vSphere infrastructure.

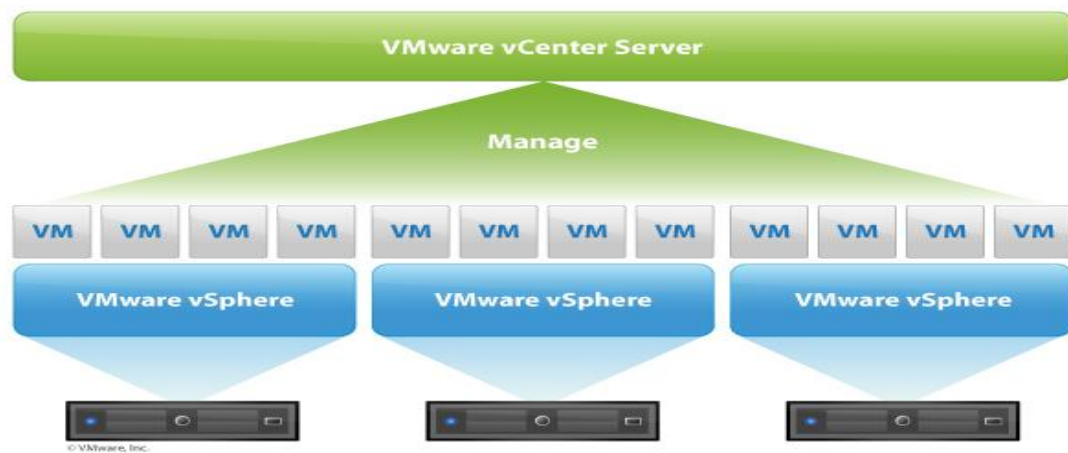


Figure 2.3 – Pictorial description of vCenter Server

CHAPTER 3

INTRODUCTION TO FABRIC INTERCONNECT

This chapter presents a theoretical overview of the Cisco FI. It gives an understanding of FI's as a concept used in datacenter switches. It gives a thorough understanding of the architecture of the Cisco FI.

3.1 Cisco Unified Computing System

The Cisco Unified Computing System is a next-generation data center platform that unites compute, network, storage access, and virtualization into a cohesive system designed to reduce total cost of ownership (TCO) and increase business agility. The system integrates a low-latency, lossless 10 Gigabit Ethernet unified network fabric with enterprise-class, x86-architecture servers. The system is an integrated, scalable, multi-chassis platform in which all resources participate in a unified management domain. Virtualization—the system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtualized environments to better support changing business and IT requirements. Storage access—the system provides consolidated access to both SAN storage and network attached storage (NAS) over the unified fabric. Unifying storage access means that the Cisco Unified Computing System can access storage over Ethernet, Fibre Channel, Fibre Channel over Ethernet (FCoE), and iSCSI, providing customers with choice and investment protection. The Cisco Unified Computing System components illustrated in Figure 3.1 include, from left to right, fabric interconnects, blade server chassis, blade servers, and in the foreground, fabric extenders and network adapters.



Figure 3.1 Cisco Unified Computing System.

3.2 Cisco Unified Computing System Components

3.2.1 Fabric Interconnect

The Cisco® UCS 6100 Series Fabric Interconnects is a core part of the Cisco Unified Computing System, providing both network connectivity and management capabilities for the system (Figure 3.2). The Cisco UCS 6100 Series offers line-rate, low-latency, lossless 10 Gigabit Ethernet and Fibre Channel over Ethernet (FCoE) functions. All chassis, and therefore all blades, attached to the Cisco UCS 6100 Series Fabric Interconnects become part of a single, highly available management domain. The fabric interconnect supports multiple traffic classes over a lossless Ethernet fabric from the blade through the interconnect. Significant TCO savings come from an FCoE-optimized server design in which network interface cards (NICs), host bus adapters (HBAs), cables, and switches can be consolidated.



Figure 3.2 Cisco UCS 6120XP 20-Port Fabric Interconnect (Top) and Cisco UCS 6140XP 40-Port Fabric Interconnect

The Cisco UCS 6100 Series is equipped to support the following module options:

- Fibre Channel plus Ethernet module that provides 4 ports of 10 Gigabit Ethernet using the SFP+ interface; and 4 ports of 1/2/4-Gbps native Fibre Channel connectivity using the SFP interface (Figure 3.3)



Figure 3.3 from left to right: 8-Port 1/2/4-Gbps Native Fibre Channel Expansion Module; 4-Port Fibre Channel plus 4-Port 10

3.2.2 Cisco Fabric Extenders Module

Cisco UCS 2100 Series Fabric Extenders bring the unified fabric into the blade server enclosure, providing 10 Gigabit Ethernet connections between blade servers and the fabric interconnect, simplifying diagnostics, cabling, and management (Figure 3.4). The Cisco UCS 2100 Series extends the I/O fabric between the Cisco UCS 6100 Series Fabric Interconnects and the Cisco UCS 5100 Series Blade Server

Chassis, enabling a lossless and deterministic Fibre Channel over Ethernet (FCoE) fabric to connect all blades and chassis together.

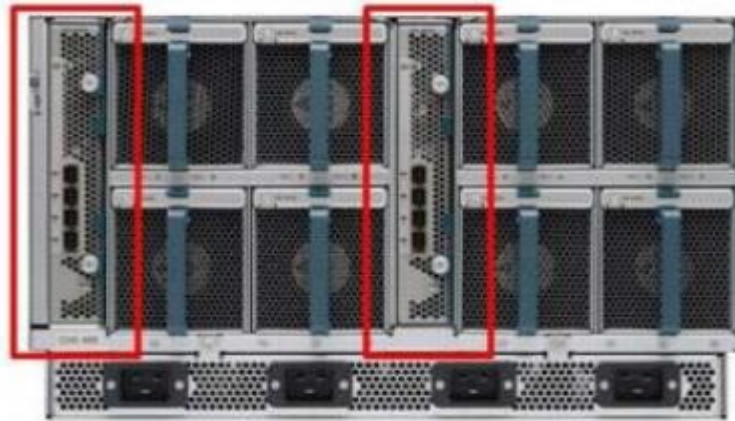


Figure 3.4 Rear view of the Cisco UCS 5108 Blade Server Chassis with two Cisco UCS 2104XP Fabric Extenders.

The Cisco UCS 2104XP Fabric Extender has four 10 Gigabit Ethernet, FCoE-capable, Small Form-Factor Pluggable Plus (SFP+) ports that connect the blade chassis to the fabric interconnect. Each Cisco UCS 2104XP has eight 10 Gigabit Ethernet ports connected through the midplane to each half-width slot in the chassis (Figure 3.5).



Figure 3.5 Cisco UCS 2104XP Fabric Extender

3.2.3 Cisco UCS Chassis

The Cisco UCS 5100 Series Blade Server Chassis is a crucial building block of the Cisco Unified Computing System, delivering a scalable and

flexible blade server chassis for today's and tomorrow's data center while helping reduce TCO. A chassis can house up to eight half-width Cisco UCS Series Blade (Figure 3.6)



Figure 3.6 Cisco Blade Server Chassis (front and back view)

3.2.4 Cisco UCS B200 M1 Blade Server

The Cisco UCS B200 M1 Blade Server (Figure 3.7) is a half-width, two-socket blade server. The system uses two Intel Xeon 5500 Series processors, up to 96 GB of DDR3 memory, two optional hot-swappable small form factor (SFF) serial attached SCSI (SAS) disk drives, and a single mezzanine connector for up to 20 Gbps of I/O throughput.



Figure 3.7 Cisco UCS B200 M1 Blade Server

3.2.5 Intel Xeon 5500 Series Processor

With innovative technologies that boost performance, energy efficiency, and virtualization flexibility, two-processor platforms based on the Intel Xeon processor 5500 series make it easier to deliver more business services within existing data center facilities (Figure 3.8).

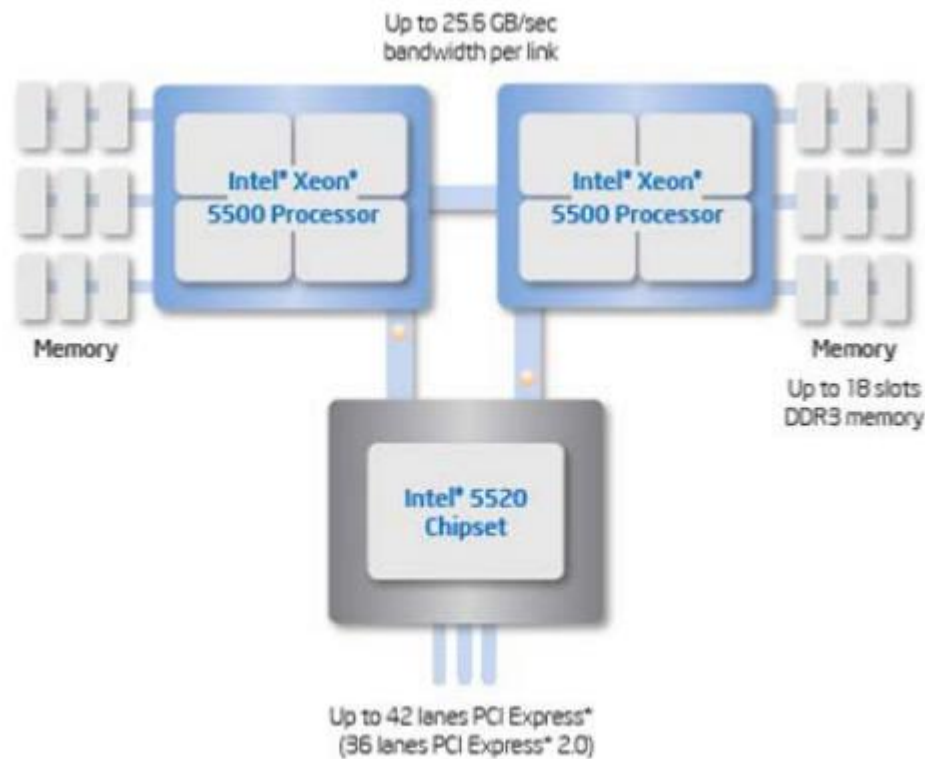


Figure 3.8 Intel Xeon 5500 Series Processor

3.2.6 Cisco UCS M71KR-Q Converged Network Adapter

The Cisco UCS M71KR-Q QLogic Converged Network Adapter (CAN Figure 3.9) is a QLogic-based Fibre Channel over Ethernet (FCoE) mezzanine card that provides connectivity for Cisco UCS B-Series Blade Servers in the Cisco Unified Computing System.



Figure 3.9 Cisco USC M71KR-Q Network Adapter

3.2.7 Cisco UCS Virtual Interface Card (VIC)

Cisco Virtual Interface Cards (Figure 3.10) were developed ground up to provide acceleration for the various new operational modes introduced by server virtualization. The Virtual Interface Cards are highly configurable and self-virtualized adapters that can create up to 128 PCIe endpoints per adapter. These PCIe endpoints are created in the adapter firmware and present fully compliant standard PCIe topology to the host OS or hypervisor.



Figure 3.10 Cisco UCS Virtual Interface Card

3.2.8 Cisco UCS C-Series Rack Mount Servers

Cisco UCS C-Series Rack-Mount Servers (Figure 3.11) extend Cisco Unified Computing System™ innovations to a rack-mount form factor, including a standards-based unified network fabric, Cisco VN-Link virtualization support, and Cisco Extended Memory Technology. Designed to operate both in standalone environments and as part of the Cisco Unified Computing System, these servers enable organizations to deploy systems incrementally—using as many or as few servers as needed—on a schedule that best meets the organization’s timing and budget. Cisco UCS C-Series servers offer investment protection through the capability to deploy them either as standalone servers in hetero-geneous data centers or as part of the Cisco Unified Computing. Although this study was carried out on the Cisco UCS B-Series blade servers, the C-Series Rack-Mount servers extend the same benefits to customers. Future desktop Virtualization studies are planned on this server platform.



Figure 3.11 Cisco UCS C-Series Rack Mount Server

CHAPTER 4

INTRODUCTION TO SERVICE PROFILES

This chapter presents a theoretical overview of the Cisco UCS Service Profiles. It gives an understanding of SP's as a concept used in datacenter switches. It gives a thorough understanding of the architecture of the Cisco's SP.

4.1 Cisco UCS Service Profiles

Cisco Unified Computing System™ technology has redefined the enterprise computing environment. By breaking the traditional model of older data centers and redefining data center infrastructure as pools of virtualized server, storage, and network resources, the Cisco Unified Computing System has enabled a new computing model that both delivers advantages in capital and operational cost, and does so in a manner that improves flexibility, availability, and reduces the amount of time needed for IT to respond to business changes.

To understand how the Cisco Unified Computing System delivers these benefits, it is necessary to understand the concept of a "service profile," the fundamental mechanism by which the Cisco Unified Computing System models the necessary abstractions of server, storage, and networking. This document explores the underlying structure of Cisco Unified Computing System service profiles, explains how they are used to enable Cisco Unified Computing System capabilities, and shows the benefits that accrue from their use.

4.1.1 Redefining the Server and the Data Center

The Old View: Separate Stacks, Lots of Glue

Today's data centers are migrating from the client-server, distributed model of the past toward the more virtualized model of the future. This steady

migration is fueled by the need to conserve space and energy ¹, as well as a desire to overcome the myriad problems that arise from supporting a heterogeneous data center environment:

- Complexity:

Too many software and hardware vendors in the data center leads to too many conflicting components. These incompatibilities hinder new product deployments and upgrades.

- Archaic Server Architectures:

While server performance has increased dramatically, the basic architecture of the server has not changed in decades. Each server is essentially its own management island, with local management and settings, some of which, such as MAC addresses, are assumed to be immutable. Multiple generations of management tools have attempted to provide a single management interface to multiple servers to simplify the operating experience for users, but these fundamentally are still layers of tools stacked on top of disjointed management domains. Blade servers as a product class have improved this situation, but they still lack true management integration beyond the chassis level.

- Storage:

As business applications increase in complexity, the need for larger and more reliable storage solutions becomes a data center imperative. Storage was the first significant server resource to be shared, in the form of Network-Attached Storage (NAS) and SANs, and today networked, shared storage is the norm in most organizations.

- Disjointed Network, Server, and Management Domains:

Even more crippling than older server architectures from the perspective of streamlining data center operations has been the evolution of server, storage, and networking solutions into separate technology and management silos.

- Disaster Recovery and Business Continuity:

Data centers must maintain business processes (with limited or no downtime) for the overall business to remain competitive.

- Inefficiency:

Application slowdowns result from server and application incompatibilities, inconsistent server policies, and poor integration of third-party hardware and software.

- Cooling:

Good ventilation saves power and reduces costs. However, dense server racks make it very difficult to keep data centers cool and hold down costs.

- Cabling:

Many of today's data centers have evolved into a complex mass of interconnected cables, further increasing rack density and further reducing data center ventilation.

4.1.2 Virtual Servers: The First Abstraction

The first major breakthrough was the development of enterprise-class hypervisors for x86 systems and the common Linux and Microsoft Windows operating systems. These hypervisors and their associated virtual servers delivered the first building block of a virtualized data center: a server abstraction that was not bound to a physical hardware element. With virtual machines, users could now define

a server in software and deploy it anywhere in the data center without altering any of the underlying hardware.

Instead of thinking of a server as a box in a rack, consider that a server is defined by the combination of the following sets of physical resources:

- Processing Resources:

CPU and memory.

- Storage Bindings:

Connections to external storage, which may contain the boot images, applications, etc. Storage may also include the local disk, but as will be discussed later, this is a less effective choice except in environments in which the boot image is almost invariant.

- Network Interfaces:

Available network connections and Network Interface Cards (NICs), Host Bus Adapters (HBAs), Converged Network Adapters (CNAs), etc.

4.2 Data Center Integration: The Missing Element

Despite the relative lack of success, the vendor community as a whole has not ignored this combined problem and opportunity, and users today have at least 200 products to choose from that purport to offer benefits in managing some aspects of the combined storage, server, and networking environments.

To meet the challenge, Cisco introduced the Cisco Unified Computing System, the first commercially available system that delivers an integrated management model that spans server, storage, and networking as well as hardware designed to take advantage of its innovations. The following sections provide an overview of the basic functions of the Cisco Unified Computing System

along with the details of the Cisco Unified Computing System service profiles, the underlying abstraction model that is central to Cisco Unified Computing System.

4.3 Cisco Unified Computing System Overview

The Cisco Unified Computing System integrates low-latency, unified network fabric with enterprise-class, x86-based servers, creating an integrated, scalable, multichassis platform in which all resources participate in a unified management domain. A single system can scale across multiple chassis and thousands of virtual machines (Figure 4.1).

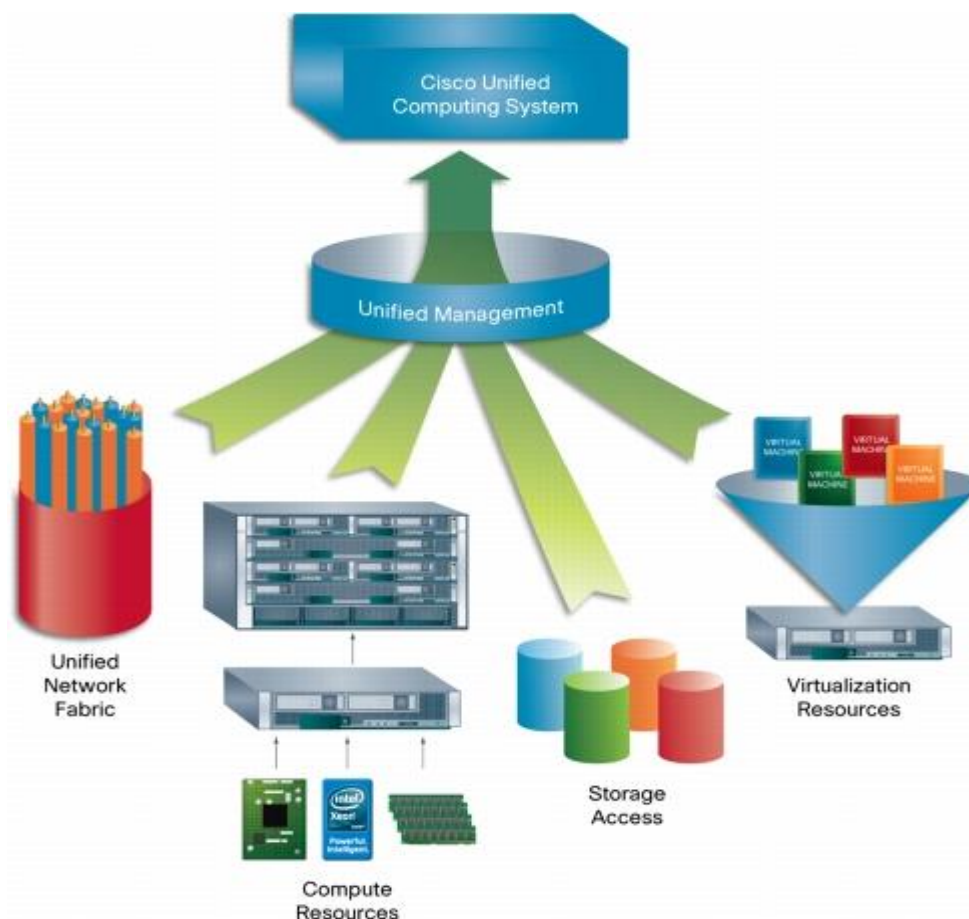


Figure 4.1 The Cisco Unified Computing System Integrates Network, Compute, Storage Access and Virtualization Resources into a Single Cohesive System

4.4 Service Profiles: Cisco Unified Computing System Foundation Technology

4.4.1 What Is a Service Profile?

Conceptually, a service profile (Figure 4.2) is an extension of the virtual machine abstraction applied to physical servers. The definition has been expanded to include elements of the environment that span the entire data center, encapsulating the server identity (LAN and SAN addressing, I/O configurations, firmware versions, boot order, network VLAN, physical port, and quality-of-service [QoS] policies) in logical "service profiles" that can be dynamically created and associated with any physical server in the system within minutes rather than hours or days.

Service profiles also include operational policy information, such as information about firmware versions.

What Is a Service Profile?

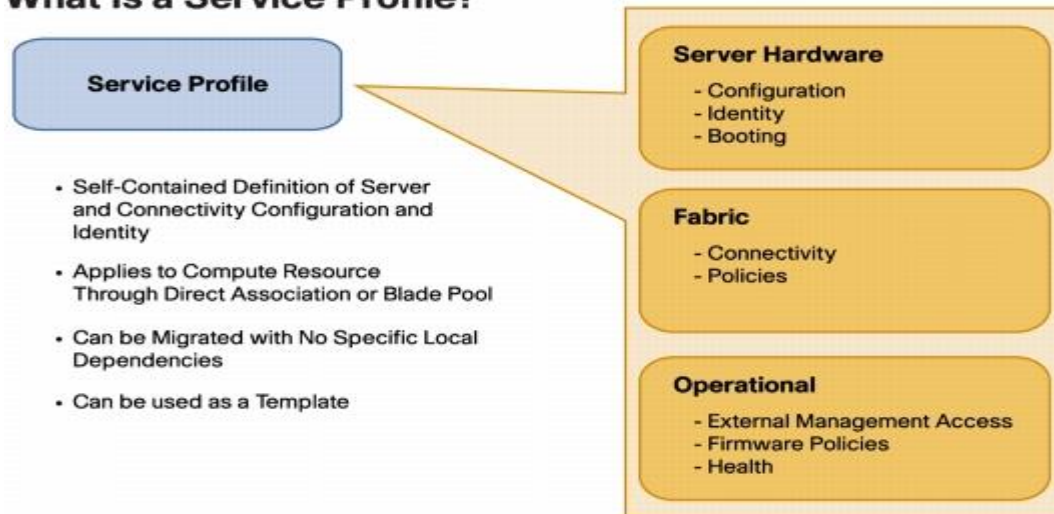


Figure 4.2 shows the major elements of a service profile

4.4.2 Service Profiles and Templates

Cisco Unified Computing System service profiles are a powerful tool for streamlining the management of modern data centers. They provide a mechanism for rapidly provisioning servers and their associated network and storage connections with consistency in all details of the environment, and they can be set up in advance of the physical installation of the servers, which is extremely useful in most organizations. Templates can be categorized into two types: initial and updating.

- Initial Template:

The initial template is used to create a new server from a service profile with UUIDs, but after the server is deployed, there is no linkage between the server and the template, so changes to the template will not propagate to the server, and all changes to items defined by the template must be made individually to each server deployed with the initial template.

- Updating Template:

An updating template maintains a link between the template and the deployed servers, and changes to the template (most likely to be firmware revisions) cascade to the servers deployed with that template on a schedule determined by the administrator.

4.5 Conclusion

Cisco Unified Computing System™ technology has redefined the enterprise computing environment. By breaking the traditional data center model and redefining data center infrastructure as pools of virtualized server, storage, and network resources, the Cisco Unified Computing System has delivered a new computing model with advantages in capital and operational cost, improving flexibility and availability, and reducing the amount of time needed for IT to respond to business changes.

CHAPTER 5

FUNDAMENTALS OF SAN and UCS MANAGER

This chapter presents a theoretical overview of the Cisco's SAN Networks and UCSM. It gives an understanding of SAN as a concept used in datacenter switches. It gives a thorough understanding of the architecture of the Cisco's SAN.

5.1 Storage Area Network:

A storage area network (SAN) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to enhance storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network (LAN) by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

A SAN does not provide file abstraction, only block-level operations. However, file systems built on top of SANs do provide file-level access, and are known as *SAN file systems* or shared disk file systems.

Historically, data centers first created "islands" of SCSI disk arrays as direct-attached storage (DAS), each dedicated to an application, and visible as a number of "virtual hard drives" (i.e. LUNs).^[1] Essentially, a SAN consolidates such storage islands together using a high-speed network.

Operating systems maintain their own file systems on their own dedicated, non-shared LUNs, as though they were local to themselves. If multiple systems were simply to attempt to share a LUN, these would interfere with each other and quickly

corrupt the data. Any planned sharing of data on different computers within a LUN requires advanced solutions, such as SAN file systems or clustered computing.

Despite such issues, SANs help to increase storage capacity utilization, since multiple servers consolidate their private storage space onto the disk arrays.

Common uses of a SAN (Figure 5.1) include provision of transactionally accessed data that require high-speed block-level access to the hard drives such as email servers, databases, and high usage file servers.

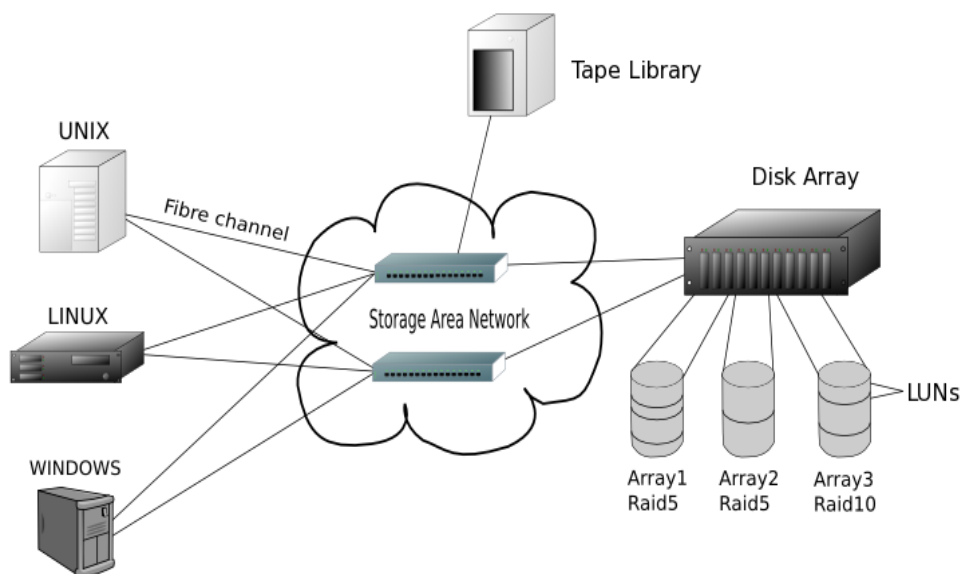


Figure 5.1 Storage Area Network

5.2 UCS Manager:

Cisco UCS (Figure 5.2) was architected with the following design goals:

1. To provide greater administrative and operational control, using fewer individual points of management, thereby allowing increased scalability while reducing complexity

2. To greatly reduce the time needed to commission new compute resources, and the time needed to deploy new servers (either bare-metal OS or hypervisor-based)
3. To improve service availability through the deepest possible abstraction of hardware state and I/O connectivity
4. To simplify server virtualization through converged physical infrastructure
5. To enable all the physical infrastructure and connectivity to be fully programmable, thereby reducing manual intervention, and enabling automation

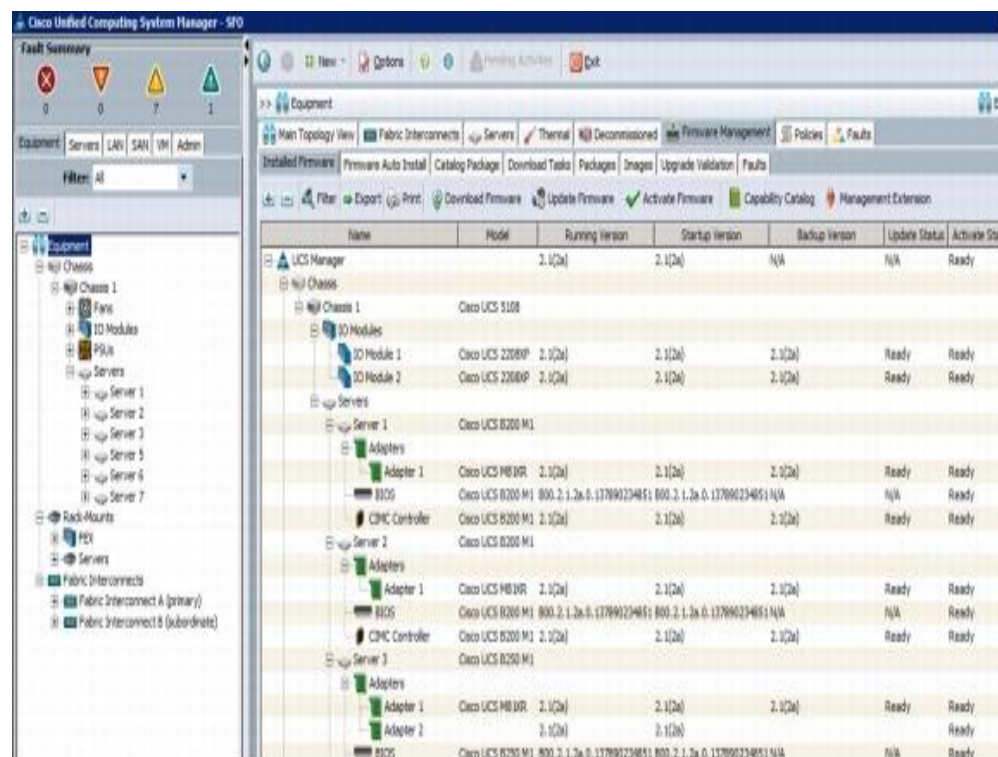


Figure 5.2 UCS Manager

CHAPTER 6

DESIGN OF LOG PARSER

This chapter includes the algorithm and work flow for automation of log parser. It also includes a section on how different modules on the N7k platform are subject to log.

6.1 Project Description and Goals

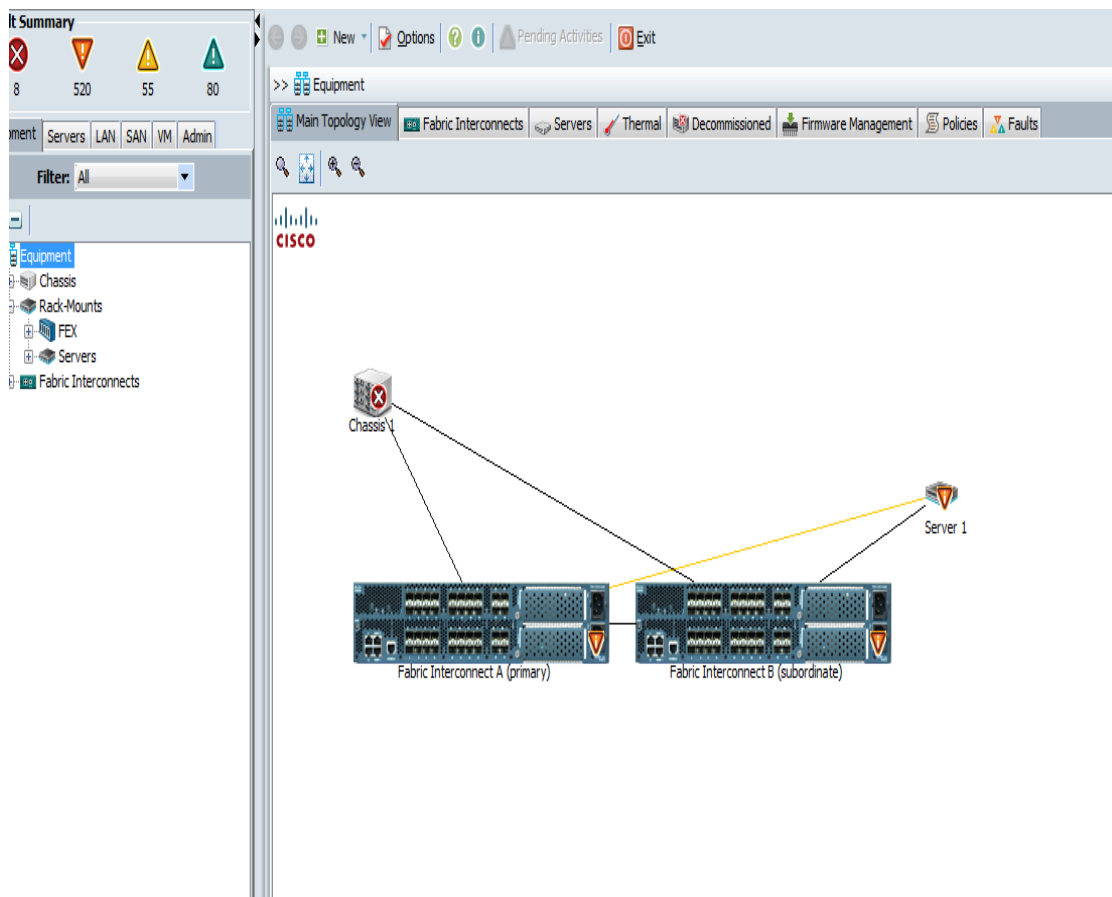


Figure 6.1 – Architecture of UCS

Each Unified Computing System has two Fabric Interconnects (Figure 6.1). One is the primary and the other is the subordinate. The UCSM software and management of the UCS can only be done on the primary. However, data forwarding is done by both the primary and the subordinate. The service profile information, Ethernet and Fiber Channel configurations of the Fabric Interconnects are stored on the flash of the primary Fabric Interconnect.

Service profiles are used to configure the servers. The key benefit of service profiles is the concept of Stateless Computing. Each server blade has no set configuration. MAC addresses, UUIDs, firmware and BIOS settings for example, are all configured on the UCS manager in a service profile and are applied to the servers. This allows for consistent configuration and ease of re-purposing. A new profile can be applied within a matter of minutes.

The goal of this project is to parse the following service profile information:

- Number of service profiles, organizations and templates
- Service profile instance and associated template
- Template and the type

This project also aims to extract the following fabric channel data:

- UCSM and NX-OS versions, FI model
- VSAN, FCoE VLAN information
- Uplink interface configuration (FC: port-channels and FCoE: VFC with bound Ethernet)
- Flogi database

In addition, the Ethernet details given below will also need to be parsed :

- Ethernet mode of operation
- VLANs and relevant ports
- Port-channel configuration

6.2 Technical Specification

While the overall goals, strategies and objectives have been stated, the specifications of the components will be determined as they are identified for their applicability in the project. The UCS system meets the specifications stated in

Table 6.1.

Table 6.1. Technical specifications of the UCS Fabric Interconnect	
Operational Specifications:	
Fabric Interconnects	<p>The uplink ports are used to connect to SAN and LAN</p> <p>The server facing ports are used to connect to the chassis</p>
Power consumption	<ul style="list-style-type: none"> • Maximum DC: 950W • Maximum (Out): 750W
Performance Specifications:	
Humidity	5% to 95% (noncondensing)
Ambient Operating Temperature	32 to 95°F (0 to 35°C)

Weight	1.134 Kg
--------	----------

6.3 Design Approach and Details

6.3.1 Design Approach

Input File Validation

In order to execute the scripts, we need to ensure that the given input file is of the UCSM tech support type. In order to do that we need to check for two UCSM_A_techsupport.tar and UCSM_B_techsupport.tar

```
is_ucsm=False

if tarfile.is_tarfile(filename):

    tar = tarfile.open(filename, "r:*")

    for item in tar.getnames():

        if re.search(r'UCSM_A_TechSupport.tar',item) or
re.search(r'UCSM_B_TechSupport.tar',item):

            is_ucsm=True
```

Identifying the primary Fabric Interconnect

The file that has the relevant data is stored only on the primary Fabric Interconnect and not the subordinate. We make use of the UCS library to check for the primary Fabric Interconnect.

```
self.ucsm = UCS(self)
```

```
        UCSMA =
self.ucsm.get_file_pointer_from_tar(filename,"UCSM_A_TechSupport.tar.gz",
True)

        UCSMB =
self.ucsm.get_file_pointer_from_tar(filename,"UCSM_B_TechSupport.tar.gz",
True)

        sam_tech_pointer =
self.ucsm.get_file_pointer_from_tar(UCSMA,"sam_techsupportinfo",False)

        if not sam_tech_pointer:

            sam_tech_pointer =
self.ucsm.get_file_pointer_from_tar(UCSMB,"sam_techsupportinfo",False)
```

Task 1 and 2:**Problem description**

Currently, program does not validate the given input file is the right type of file for the task or not.

Expected fix

If user provide any file other than "standalone rack server techsupport bundle", the task should exit with informational message to user.

If it matches, then it should proceed with rest of the program.

Informational message

The input file is not rack server techsupport bundle. Please re-run it with valid rack server techsupport bundle.

Task 3:

List Service profiles created in UCS.

- 1) List all service profiles that are created in UCSM (Org, DN and SP name)
- 2) List SP and associated blade
- 3) Get SP type (instance or whether derived from template) . For SP derived from template, display parent template SP name and template type

Task 4:

To extract the information related to FC Configuration :

- UCSM , NX-OS version
- FI model
- FC mode -- End or Switch mode
- VSAN / FCoE VLAN information
- FC MAP value
- Uplink interface configure report
- FC - individual and / or port channel info
- FCoE - vfc and bound Ethernet info
- Storage appliance port --
- Each interface configure, SFP , speed , WWPN, status
- Server interfaces
- vfc #, Server x/y and WWPN , Logged in or not
- uplink pinned interface info

- FLOGI database
- Traffic Counters

Task 5:

- General information
- UCSM, NX-OS version
- FI model
- Ethernet mode - End host or Switch mode
- List of VLANs per FI, number of VLANs, DR information
- VLAN compression enabled or not, VLAN compression groups
- MAC aging timer
- VLAN port count
- Uplink interfaces
- List of interfaces - add port-channel membership if any
- List of VLANs allowed on each uplink
- Interface status
- CDP information for the interface
- Server facing interface
- Chassis facing ports - which port goes to which IOM
- Port-channel or individual
- Server interfaces
- veth #, VLAN list, Server x/y and mac address
- uplink pinned interface info
- Do not consider floating Vethernet interfaces (Veth # >= 32768)
- Error Counters for FI interfaces only
- Last link flap
- input errors, CRC

- output errors
- Free ports information

Task 6:

1) Identify if UCSM techsupport (Figure 6.1) has statsAG core file (refer ucs_reload_info task and "show cores detail" cmd output from sam_techsupportinfo)

If no statsAG core file, exit the task

2) Identify UCSM version is $\geq 2.1.3b$ and less than 2.2.4

3) Identify if statsAG core was created in version of lower than 2.1.3b . If yes, exit the task. If not, issue is due to CSCun07367 and print bug details

Diagram of BORG execution

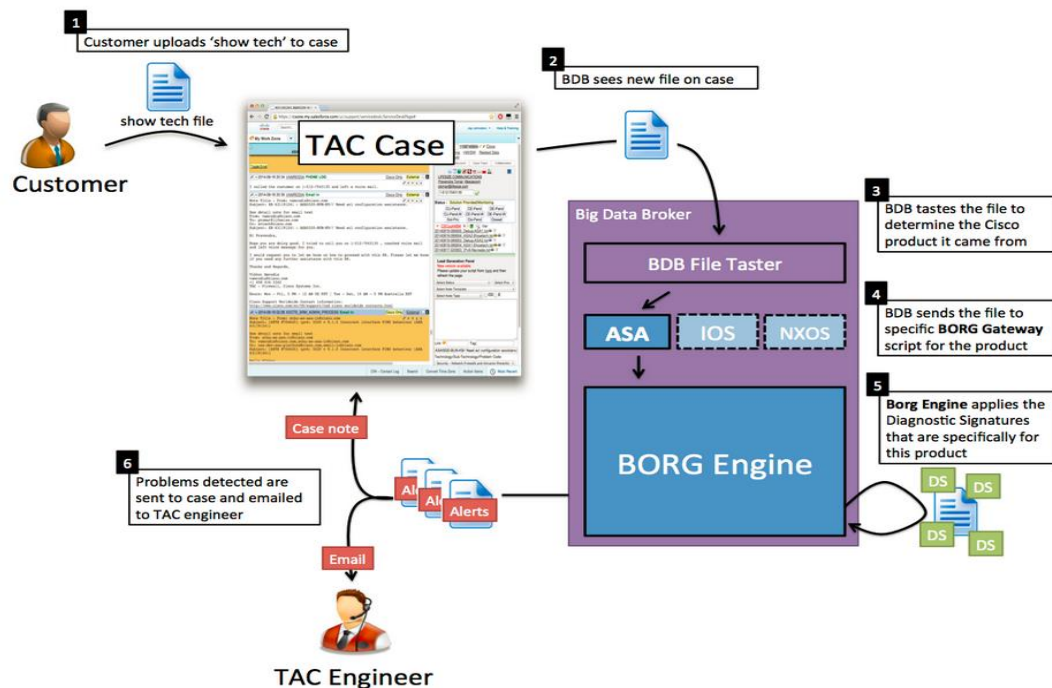


Figure 6.2 Steps to automate the log parser through BORG engine.

Task 7:

Nexus 1000v show tech parser

We do not support 4.0(4)SV1(1). Check the version/product initially and warn the user if the product is not N1k or if its running 4.0(4)SV1(1).

We support the following:

1. 'show tech svcs' for codes less than 5.2(1)SV3(1.1) - you could look at SV1, SV2 in the version

- You can check if 'show svcs domain', 'show svcs connections', 'show svcs neighbors' are present in the show tech uploaded

2. 'show tech detail' for codes from 5.2(1)SV3(1.1)

- You can check if 'show svcs domain', 'show svcs connections', 'show svcs neighbors', 'show interface' are present in the show tech uploaded

User should be warned if the wrong file is uploaded because many engineers are not aware of the fact that they should start looking at 'show tech detail' from from 5.2(1)SV3(1.1)

VSM details:

`show version` -> System version

`show switchname` -> Switch name

`show clock` -> Current time

`show version` -> Kernel uptime

`show module uptime` -> Module 1 & 2 uptime, start time

`show interface mgmt 0` -> Management IP address

`show module` -> Find which module is active

`show system redundancy status` -> Internal state of 'this supervisor' will show the redundancy state

`show switch edition` -> Switch edition - this will be present only from 4.2(1)SV2(1.1).

SVS details:

`show svs domain` -> for domain ID, control vlan, packet vlan, control mode - if control mode is L3, then need to print L3 control interface also

`show svs connections` -> vCenter IP address, remote port, datacenter name, operational status, vcenter version

`show svs neighbors` -> AIPC interface MAC

VEM details:

`show module` -> module number, ESX version, VEM version, server UUID, host name - print from module 3 onwards and not modules 1 and 2 (these are VSM)

`show module uptime` -> look for module uptime/start time - the current logic of displaying 'last attached time' looks wrong

`show svs neighbors` -> Src mac column would show the DPA MAC - match the node id with the module id. Example node id 0302 is module 3

Port-profiles:

`show port-profile` -> for a list of port-profile - add a new column to the existing script saying if the port-profile is of type Vethernet or Ethernet

`show port-profile` -> assigned interfaces section shows the list of assigned interfaces

`show running-configure` -> would show the configure for each port-profile

Ethernet interfaces:

`show interface brief` -> interface ID, Status, reason (add this to the current script), mode, speed, native vlan, port-channel

`show interface` -> MTU

`show cdp neighbors` -> CDP neighbor and the neighbor interface

`show running-configure` -> configure of the interface

vEthernet interfaces:

`show interface brief` -> interface ID, Status, port mode, VLAN

`show interface` -> VM name, Adapter, MAC address, module

If needed, you can use 'show interface' to check both

`show running-configure` -> configure of the interface

Port-channels:

`show port-channel summary` -> interface ID, status, member ports

You can use 'show interface brief' for interface ID and status (or) 'show interface' for interface ID, status and member ports

`show running-configure` -> configure of the interface and the mode

Others:

`show cores` -> for core files

`show processes log` -> for process restarts

`show startup-configure` -> to show when the startup configure was last saved

`show running-configure` and `show startup-configure` -> for boot statements

`show system internal mts buffers` -> for number of MTS buffers in use

Sample logs: <http://10.76.78.180/misc/ucs-bdb/N1k%20logs/>

Current script: https://scripts.cisco.com/ui/use/n1k_vsm_parser

6.4 Algorithm

1. Begin
2. Parse the show tech-support output file to get show_techsupport and sw_techsupport sections
3. Store the parsed sections in different lists.

4. Parse both files to get various FC and Service Profile Records and it's sub sections
5. Find the matching pattern for each configure
6. Get the result and store it in a dictionary
7. Parse the file to get the attributes of host collected by UCSM
8. Compare the results and store the HTML Records that are relevant
9. Parse the show tech-support output file to get the running-configuration
10. Store the running-configuration in a list
11. Search the running-configuration for dynamic configure and get the actions of each of the polices and store them in another list
12. Compare the Service Profile Records that were compared before and integrate the actions with the policies.
13. Display the combined results of those entries.
14. Stop.

CHAPTER 7

IMPLEMENTATION ON CISCO BDB

7.1 Parse the required information

From the sam_techsupportinfo and sw_techsupportinfo files, we can now parse the service profile and fabric channel information.

7.2 Using the Script

Click on the Grey icon (Figure 7.1) next to the UCSM tar file in CaseKwery

Request Type: Replace my Product

Service Loss: NO

Technology: Data Center Computing - (UCS Blade and Rack Mount Server Systems)

Subtechnology: UCS-B System/Other Issues

Problem Type: Software or Hardware Configuration assistance needed (Including usage of show commands)

SN (product): [SSI183801SP](#) (UCS-SP8-B-FI48)

Node:

Activity: Operate

Hardware: UCS-SP8-B-FI48

Software: N10-MGT UCSM 2.2(3)

R(s) RMA(s) LAB	Attachment(s)	Upload New File(s)				
	20150405-090057687_image005.jpg				944 bytes	05 Apr 2015 14:00 GMT
	20150405-085906835_image006.jpg				912 bytes	05 Apr 2015 13:59 GMT
	20150405-085906834_image004.jpg				944 bytes	05 Apr 2015 13:59 GMT
	20150405-085440539_image006.jpg				912 bytes	05 Apr 2015 13:54 GMT
	20150405-084944004_image006.jpg				912 bytes	05 Apr 2015 13:49 GMT
	20150405-084944004_image007.jpg				5 Kb	05 Apr 2015 13:49 GMT
	20150405-084944003_image002.gif				144 bytes	05 Apr 2015 13:49 GMT
	20150405-084944003_image003.jpg				980 bytes	05 Apr 2015 13:49 GMT
	20150405-084944003_image001.jpg				6 Kb	05 Apr 2015 13:49 GMT
	05_Apr_2015_06_44_46_20150405100017_LAB-GA-UCSIC01_BC1_all tar				14.14 Mb	05 Apr 2015 13:44 GMT
	05_Apr_2015_06_44_35_20150405095517_LAB-GA-UCSIC01_UCSM.ta				51.45 Mb	05 Apr 2015 13:44 GMT
	20150405-084416789_image008.jpg				6 Kb	05 Apr 2015 13:44 GMT
	20150405-084416788_image004.jpg				944 bytes	05 Apr 2015 13:44 GMT
	20150405-083936489_image002.gif				144 bytes	05 Apr 2015 13:39 GMT

:UNDISPATCHE [0m]

:VW-MS [5m]

:ATARR [7m]

:ATARR [4m]

:UNDISPATCHE [0m]

:VW-SV [4m]

Figure 7.1 - CISCO CaseKwery tool

Type the script name (Figure 7.2) – For example, 'ucs_service_profile'

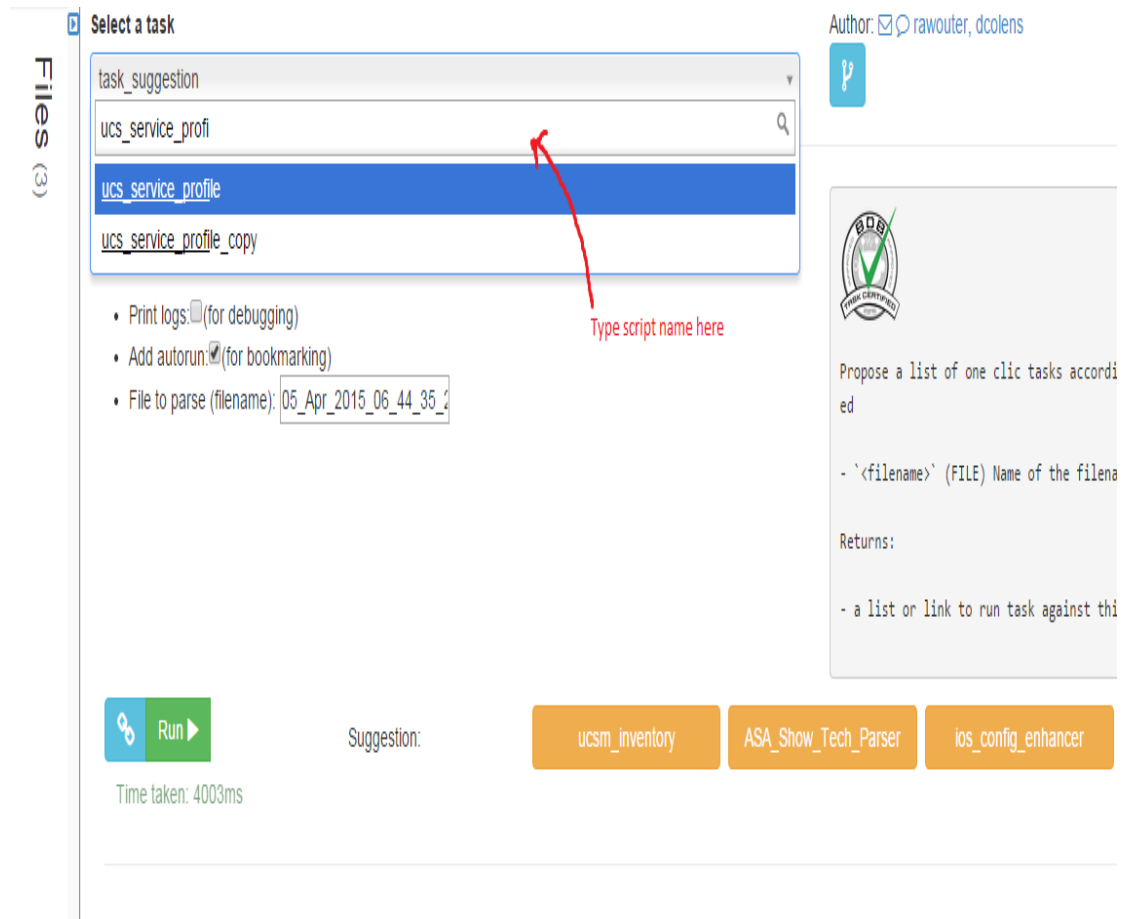


Figure 7.2 – Screenshot of BDB platform

7.3 View parsed information

https://scripts.cisco.com/ui/use/task_suggestion?filename=05_Apr_2015_06_44_35_20150405095517_LAB-GA-UCSIC01_UCSM.tar&cs=csonesr=634363693&autorun=true&url=http%2F%2Fwww-tac.cis

Service profiles under each organization:

```
(root): 0
GA-VCAC: 0
GA-VCAC/GA-ProvSite01: 6
```

Service profiles and associated servers:

#	Service Profile Name	Association Status	Associated Server
1	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX001	Associated	1/1
2	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX002	Associated	1/2
3	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX003	Associated	1/3
4	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX004	Associated	1/4

Service profiles and their source templates:

#	Service Profile Name	Source Template
1	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX001	GA-VCAC-SD-Profile01
2	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX002	GA-VCAC-SD-Profile01
3	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX003	GA-VCAC-SD-Profile01
4	GA-VCAC/GA-ProvSite01/GA-VCAC-ProvSite-ESX004	GA-VCAC-SD-Profile01

Figure 7.3 – Output of the ‘ucs_service_profile’ script

7.4 Codes and Standards

The Fabric Interconnects used for the testing of the script are compliant with the IEEE 802.1p: CoS prioritization, IEEE 802.1Q: VLAN tagging, IEEE 802.1s: Multiple VLAN instances of Spanning Tree Protocol, IEEE 802.1w:

Rapid reconfiguration of Spanning Tree Protocol, IEEE 802.3: Ethernet, IEEE 802.3ad: LACP, and IEEE 802.3ae: 10 Gigabit Ethernet SFP+ support

7.5 Constraints and Tradeoffs

The following constraints were experienced while designing and executing the project.

Design constraints

The script had to be coded in Python to be compatible with BDB (Big Data Broker) which is Cisco's private open source platform for creating scripts that are used to automate the extraction of relevant information.

Performance Constraints

Limited memory and CPU constraints put pressure on the performance of the script. Minimal file handling had to be done to ensure that the program doesn't crash and optimization had to be done to reduce time.

Tradeoffs

Significant tradeoffs were also dealt with while keeping in mind the goals and developments of the project. Two major tradeoffs that were experienced during the design and development are discussed.

Accuracy versus speed

The scripts are designed to be accurate at the expense of time. This is because even in an ideal scenario, the TAC engineer would take considerably more time to gather the same intelligence as this script provides. However, if a TAC engineer depends on this script when solving a live network down situation, a small mistake costs a lot. Therefore higher accuracy is well worth the extra seconds.

Time versus Features per Script

If the script contains more modules than usual, the time taken to execute the script will increase. By mutual agreement with our mentors and other TAC engineers, we have decided that speed is more important. One can run multiple scripts to extract different kinds of information if required. Integrating them will provide sometimes irrelevant or unrelated information and will take long to run too. Thus we have decided to make one script per feature.

7.6 Schedule, Tasks and Milestones

There are three major milestones (Figure 7.4) as well as several smaller tasks that must be achieved in order to complete the project. The milestones are given below:

- Service Profile Task
- Fabric Channel Script
- LAN Report

The project plan is represented in the diagram given on the next page.

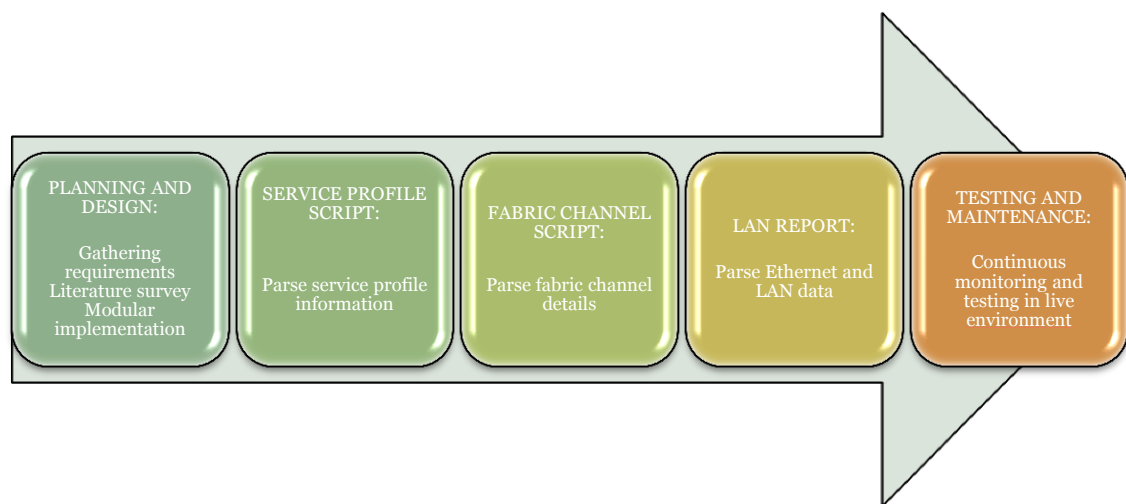


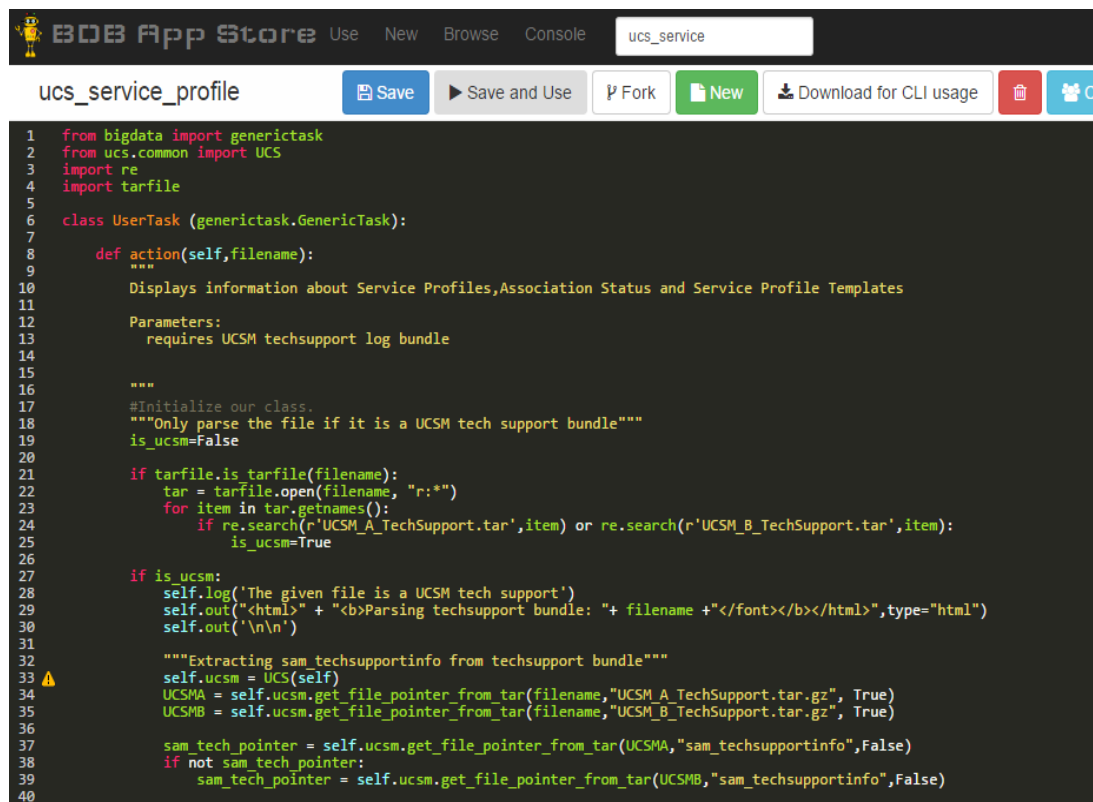
Figure 7.4 – Project plan

CHAPTER 8

RESULTS, DISCUSSION AND INFERENCE

8.1 Project Demonstration

The project aims to parse out certain details from the UCSM tar files that customer uploads onto the CaseKwery tool. To verify the correctness of the script, the output can be matched with the content in the log files. These scripts are Cisco proprietary and require a user to be connected to the Cisco server to gain access. Therefore due to confidentiality and access constraints, a live demonstration is not possible. Instead, the scripts will be run prior to the review and the output will be shown.



```

1  from bigdata import generictask
2  from ucs.common import UCS
3  import re
4  import tarfile
5
6  class UserTask (generictask.GenericTask):
7
8      def action(self,filename):
9          """
10             Displays information about Service Profiles,Association Status and Service Profile Templates
11
12             Parameters:
13                 requires UCSM techsupport log bundle
14
15             """
16
17             #Initialize our class.
18             """Only parse the file if it is a UCSM tech support bundle"""
19             is_ucsm=False
20
21             if tarfile.is_tarfile(filename):
22                 tar = tarfile.open(filename, "r:")
23                 for item in tar.getnames():
24                     if re.search(r'UCSM_A_TechSupport.tar',item) or re.search(r'UCSM_B_TechSupport.tar',item):
25                         is_ucsm=True
26
27             if is_ucsm:
28                 self.log('The given file is a UCSM tech support')
29                 self.out('<html>' + "<b>Parsing techsupport bundle: " + filename + "</font></b></html>",type="html")
30                 self.out('\n\n')
31
32                 """Extracting sam_techsupportinfo from techsupport bundle"""
33                 self.ucsm = UCS(self)
34                 UCSMA = self.ucsm.get_file_pointer_from_tar(filename,"UCSM A_TechSupport.tar.gz", True)
35                 UCSMB = self.ucsm.get_file_pointer_from_tar(filename,"UCSM B_TechSupport.tar.gz", True)
36
37                 sam_tech_pointer = self.ucsm.get_file_pointer_from_tar(UCSMA,"sam_techsupportinfo",False)
38                 if not sam_tech_pointer:
39                     sam_tech_pointer = self.ucsm.get_file_pointer_from_tar(UCSMB,"sam_techsupportinfo",False)
40

```

Figure 8.1 – Screenshot of code snippet

8.2 Marketing and Cost Analysis

8.2.1 Marketing Analysis

There are other scripts that parse UCS logs on BDB but those are limited to parsing just the hardware details of the Fabric Interconnects. None of them can extract the configuration. Moreover, those were developed for older software versions. As such, our scripts are the only ones that cater to the latest software releases and can extract the running configuration.

Further, the outputs of our scripts have easy to read formats. We have made extensive use of HTML tables, fonts and text formats to deliver the results. In addition, our scripts are far more accurate and robust than most other programs due to concentrated testing and maintenance. We will continue to check into the status every week to ensure that our scripts remain top notch.

8.2.2 Cost Analysis

Table 8.1. Cost analysis for Cisco's Unified Computing System	
Parts	
Chassis (1)	\$ 2639.56
B server blade (1)	\$ 1405.10
Fabric Interconnect (2)	\$ 28,339.20
10G SFP (4)	\$ 3980.00
Cables (3)	\$ 7.80
Total Cost per Unit	\$ 36,371.66

Table 8.1 provides a detailed cost analysis on the required parts for the UCS system.

8.3 Outputs:

Parsing techsupport bundle: 19_Nov_2014_07_34_57_20141119150330_NECC2014_UCSM.tar

Service Profile Summary:

Service profiles: 38
Templates: 5
Organizations: 2

Service profiles under each organization:

(root): 0
2ktelecom: 43

Service profiles and associated servers:

#	Service Profile Name	Association Status	Associated Server
1	2ktelecom/Blade-01.1	Associated	1/1
2	2ktelecom/Blade-01.2	Associated	1/2
3	2ktelecom/Blade-01.3	Associated	1/3
4	2ktelecom/Blade-01.4	Associated	1/4
5	2ktelecom/Blade-01.5	Associated	1/5
6	2ktelecom/Blade-01.6	Associated	1/6
7	2ktelecom/Blade-01.7	Associated	1/7
8	2ktelecom/Blade-01.8	Associated	1/8
9	2ktelecom/Blade-02.1	Associated	2/1
10	2ktelecom/Blade-02.2	Associated	2/2

Service profiles and their source templates:

#	Service Profile Name	Source Template
1	2ktelecom/Blade-01.1	Sasiu-1
2	2ktelecom/Blade-01.2	Sasiu-1
3	2ktelecom/Blade-01.3	Sasiu-1
4	2ktelecom/Blade-01.4	Sasiu-1
5	2ktelecom/Blade-01.5	Sasiu-1
6	2ktelecom/Blade-01.6	Sasiu-1
7	2ktelecom/Blade-01.7	Sasiu-1
8	2ktelecom/Blade-01.8	Sasiu-1
9	2ktelecom/Blade-02.1	Sasiu-2

Template information:

#	Service Profile Template Name	Template Type
1	2ktelecom/Sasiu-1	Initial Template
2	2ktelecom/Sasiu-2	Initial Template
3	2ktelecom/Sasiu-3	Initial Template
4	2ktelecom/Sasiu-4	Initial Template
5	2ktelecom/Sasiu-5	Initial Template

Figure 8.2 - Service profile

File name :

SR632746105-20150113155200_mgmt-ucs01_UCSM (2).tar

FI Model :

UCS FI 6120

UCSM Version :

2.2(1d)A

NX-OS Version :

FI	Kernel	System
A	5.2(3)N2(2.21d)	5.2(3)N2(2.21d)
B	5.2(3)N2(2.21d)	5.2(3)N2(2.21d)

FC Operational Mode :

End Host

VSAN / FCoE VLAN :

Name	VSAN	FCoE VLAN	Fabric
Production	301	301	A
TestDev	311	311	A
Production	302	302	B
TestDev	312	312	B
default	1	4048	Dual

FABRIC INTERCONNECT A

Uplink interface configuration summary :

FC Interfaces :

Interface	Speed	Type	Status	Trunking	Port channel
2/1	4	NP	trunking	on	215
2/2	4	NP	trunking	on	215
2/3	---	NP	down	on	--
2/4	---	NP	down	on	--
2/5	---	NP	sfpAbsent	on	--
2/6	---	NP	sfpAbsent	on	--

SAN Port Channel :

Interface	Vsan	Admin Trunk Mode	Status	Oper Mode	Oper Speed
san-port-channel 215	1	on	trunking	TNP	8

FCoE Interfaces :

FCoE interfaces not found

Port Channel Interfaces :

Interface	VLAN	Mode	Status	Reason	Protocol
Po205	1	trunk	down	Administratively	10G(D)

VFC and bound ethernet information :

Interface	Vsan	Admin mode	Admin Trunk Mode	Status	Bind Info	Oper mode
vfc1156	1	NP	on	down	port-channel205	--

FLOGI Table :

BUG !!! 'show npv flogi-table' command not available

FC Error Counters :

Note : Interfaces with only non zero counters are included.

Interface	Input Discards	Input errors	CRC	Output Discards	Output errors	Input OLS	Input NOS	Output OLS	Output NOS
2/1	12	0	0	12	0	3	3	3	3
2/2	0	0	0	0	0	14	9	14	9
2/3	2	0	0	2	0	1	1	1	1
2/4	0	0	0	0	0	5	2	5	2

B2B Credits :

Note : Interfaces with only non zero counters are included.

Interface	Transmit transitions	Receive transitions	Receive remaining	Transmit remaining
2/1	0	0	16	16
2/2	0	0	16	0
2/3	0	0	16	16
2/4	0	0	16	0
2/5	0	0	16	16
2/6	0	0	16	0

FCoE error counters :

Note : Interfaces with only non zero counters are included.

No FCoE error counters found

Figure 8.3 – ucs-FC-configuration report

File name :

SR632746105-20150113155200_mgmt-ucs01_UCSM (2).tar

UCSM Version :

2.2(1d)A

NX-OS Version :

FI	Kernel	System
A	5.2(3)N2(2.21d)	5.2(3)N2(2.21d)
B	5.2(3)N2(2.21d)	5.2(3)N2(2.21d)

FI Model :

UCS FI 6120

Ethernet Operational Mode :

End Host

MAC table aging time :

Mode Default

VLAN port count optimisation :

Disabled

VLAN compression groups :

VLAN Compression groups does not exist

Note: VLAN compression (Port count optimization) is not relevant for 6100 series FIs and works only on UCS 2.1 or later (1.3, 1.4, 2.0 wont have these).

Uplink interfaces :

Ethernet interface	VLAN	Mode	Status	Reason	Port Channel	CDP Neighbor	Port id	Allowed Vlans
1/1	1	trunk	up	none	11	---	---	1-4,121,125-126,129,139,144-148,152-155,252,720,901,3901
1/2	1	trunk	up	none	11	---	---	1-4,121,125-126,129,139,144-148,152-155,252,720,901,3901

Port channel interfaces :

Interface	VLAN	Mode	Status	Reason	Protocol
11	1	trunk	up	none	lacp

FABRIC INTERCONNECT A

VLAN port count :

VLAN-Port Limit	Access VLAN	Border VLAN	Alloc Status
14000	206	22	Available

List of VLANs :

ID	Vlan Name	DR ports
1	default	Po11
2	VLAN0002	Po11
3	VLAN0003	Po11
4	VLAN0004	Po11
121	VLAN0121	Po11

Uplink interfaces :

Ethernet interface	VLAN	Mode	Status	Reason	Port Channel	CDP Neighbor	Port id	Allowed Vlans
1/1	1	trunk	up	none	11	---	---	1-4, 121, 125-126, 129, 139, 144-148, 152-155, 252, 720, 901, 3901
1/2	1	trunk	up	none	11	---	---	1-4, 121, 125-126, 129, 139, 144-148, 152-155, 252, 720, 901, 3901

Port channel interfaces :

Interface	VLAN	Mode	Status	Reason	Protocol
11	1	trunk	up	none	lacp

Server facing interfaces :

Fabric port	FEX ID	Fabric port state	FEX uplink	FEX Model	FEX serial	Port ch id
1/9	1	Active	1	N20-I6584	QCI1607A2R7	--
1/10	1	Active	2	N20-I6584	QCI1607A2R7	--
1/11	2	Active	1	N20-I6584	QCI1543A0QI	--
1/12	2	Active	2	N20-I6584	QCI1543A0QI	--
1/13	3	Active	1	N20-I6584	QCI1602A6EY	--
1/14	3	Active	2	N20-I6584	QCI1602A6EY	--
1/15	4	Active	1	N20-I6584	QCI1607A2P2	--
1/16	4	Active	2	N20-I6584	QCI1607A2P2	--

vEthernet Interfaces :

vEthernet	Server	vNIC name	MAC address	VLAN	Status	Reason	Border Interface	Pinned Duration	Allowed VLANs
1088	1/1	vNIC1_vKernel_A	547f.ee55.e240	152	up	none	Po11	10d 53:41:3	152
1090	1/1	vNIC3_VMs_A	547f.ee55.e240	1	up	none	Po11	10d 53:41:4	2-4, 121, 125-126, 129, 139, 145-148, 153-155, 720
1092	1/1	vNIC5_vMotion_A	547f.ee55.e240	252	up	none	Po11	10d 53:41:4	252
9286	1/1	vHBA1_Fab1_FI_A	547f.ee55.e240	311	up	none	-	-	---
1048	1/2	vNIC1_vKernel_A	547f.ee55.e240	152	up	none	Po11	10d 53:4:57	152
1050	1/2	vNIC3_VMs_A	547f.ee55.e240	1	up	none	Po11	10d 53:5:17	2-4, 121, 125-126, 129, 139, 145-148, 153-155, 720
1052	1/2	vNIC5_vMotion_A	547f.ee55.e240	252	up	none	Po11	10d 53:5:17	252

Free ports :

Slot	Free ports
1	17, 18, 3, 4, 5, 6, 7, 8

Figure 8.4 – ucs-LAN-configuration-report

8.4 Summary

Currently, the project is complete and it is public to all Cisco employees. We noticed that our scripts attracted considerable amount of traffic and were reasonably popular. TAC Engineers across the globe are making use of these scripts and we have received positive feedback. Although extensive testing was done, we still encountered two bugs when the project was run in the live environment. Those bugs are fixed and there are no more issues as of now. We will continue to monitor the scripts and make changes as required.

CHAPTER 9

TESTING AND BUG REPORT

Various logs used for testing:

FC mode, Traffic / error counters

632711357

FCoE uplink interface

632746105

FC port-channel, interface status

632486557

And we can take logs attached (from SRs) to defect CSCus11782

UCS FI with no Fc or FCoE interfaces (uplink)

633199693

FC end host mode connected to brocade

633570693

MDS- FI port channel issues

633601741

FC port errors

633479893

FC ports - speed difference

633659853

FI - MDS connectivity / configure issue

633645525

FC interface down

633637397

HBA paths

633672325

FC ports down after upgrade

633671845

FI - Brocade

633673833

(These logs are taken from web monitor and some of the logs does not contain UCSM.tar file.)

Service Profile:

Bug is found in the SR 634363693 and it is rectified now.

634363693 Bug Found

All bugs found in FC and LAN Configure report the same error.

BUG:

An unhandled exception occurred inside UserTask

Traceback (most recent call last):

File "/var/bdb/task_libs/python/bigdata/generictask.py", line 233, in run

self.action(**self.arguments)

File "/var/bdb/tasks/task_ucs_fc_configure_report.py", line 103, in action

portXml = self.ucsm.convertUCSMxmltoDict(samPorts)

File "/var/bdb/task_libs_contrib/lib/python/ucs/common.py", line 648, in
convertUCSMxmltoDict

for item in Array:

TypeError: 'bool' object is not iterable

BUG Tracker:**BUG 1:**

On 2/2/2015 7:01AM, bigdatabroker@cisco.com wrote:

raethira tried to run ucs_hw_info with the following inputs:

```
{"filename":"20150127161214_beucsk140fi_BC_CIMC12.tar"}
```

the execution failed with the following response:

```
{  
  
  "code": 401,  
  
  "data": {
```

```
"exception": "Traceback (most recent call last):\n File\n \"/var/bdb/task_libs/python/bigdata/generictask.py", line 171, in run\n self.action(**self.arguments)\n File \"/var/bdb/tasks/task_ucs_hw_info.py", line 58,\n in action\n   for line in sam_tech_pointer:\nTypeError: 'bool' object is not iterable\n",\n\n"printables": [],\n\n"variables": { }\n\n},\n\n"hostname": "bdb-wrk-alln-2-docker",\n\n"message": "An unhandled exception occurred inside UserTask",\n\n"task_name": "ucs_hw_info",\n\n"worker_duration": 0.041116952896118164 } sr id:null
```

Status:

Fixed. The input should be of valid ucsn file.

BUG 2:

Bug in UCSM software

A template cannot have source template. However, in this case it has it.

----- Update task -----

Task did not capture this info. If the SP is template type, verify it has any source template. If it has, print it out.

http://www.tac.cisco.com/Teams/ks/c3/getLargeFile.php?srId=627805569&fileName=AFTER-UPGRADE-20150210041904_PIL-FI_UCSM.tar


```
enter service-profile Rossel-VDN-SPT-RHEL-Wbl-Prod-C1 updating-template
```

```
    associate server-pool Rossel-VDN-Servers-C1-Menlo ""
```

```
    enter vcon-assign ethernet vNIC-A any 3
```

```
    set admin-vcon any
```

```
    set order 3
```

```
set src-templ-name Rossel-VDN-SPT-RHEL-C1
```

```
    set stats-policy default
```

```
    set user-label ""
```

Status:

The bug was found in in-built function and it is fixed by rewriting the sorting code using user defined function with the help of dictionaries.

BUG 3:

Log file: [http://www-](http://www-tac.cisco.com/Teams/ks/c3/getLargeFile.php?srId=627805569&fileName=AFTER-UPGRADE-20150210041904_PIL-FI_UCSM.tar)

[tac.cisco.com/Teams/ks/c3/getLargeFile.php?srId=627805569&fileName=AFTER-UPGRADE-20150210041904_PIL-FI_UCSM.tar](http://www-tac.cisco.com/Teams/ks/c3/getLargeFile.php?srId=627805569&fileName=AFTER-UPGRADE-20150210041904_PIL-FI_UCSM.tar)

Your task (**INCORRECT**) output

[http://www-](http://www-tac.cisco.com/Teams/ks/c3/casekwery.php?Case=627805569+#note_621161822)

[tac.cisco.com/Teams/ks/c3/casekwery.php?Case=627805569+#note_621161822](http://www-tac.cisco.com/Teams/ks/c3/casekwery.php?Case=627805569+#note_621161822)

service profile

bsv-6120-1-A /org/service-profile # where

Mode: /org/service-profile

Mode Data:

scope org

```
enter org Rossel-VDN
```

```
enter service-profile Rossel-VDN-SP-RH-Wbl-Prod-C2-1 instance
```

```
bsv-6120-1-A /org/service-profile # show detail | egrep 'Service Profile
```

```
Name|Type|Source Template'
```

Service Profile Name: Rossel-VDN/Rossel-VDN-SP-RH-Wbl-Prod-C2-1

Type: Instance

Source Template: Rossel-VDN-SPT-RHEL-Wbl-Prod-C2

Oper Source Template: org-root/org-Rossel-VDN/ls-Rossel-VDN-SPT-RHEL-Wbl-Prod-C2

```
bsv-6120-1-A /org/service-profile #
```

===== SPT of SP =====

```
bsv-6120-1-A /org/service-profile # where
```

Mode: /org/service-profile

Mode Data:

```
scope org
```

```
enter org Rossel-VDN
```

```
enter service-profile Rossel-VDN-SPT-RHEL-Wbl-Prod-C2 updating-template
```

```
bsv-6120-1-A /org/service-profile # show detail | egrep 'Service Profile
```

```
Name|Type|Source Template'
```

Service Profile Name: Rossel-VDN/Rossel-VDN-SPT-RHEL-Wbl-Prod-C2

Type: Updating Template

Source Template: Rossel-VDN-SPT-RHEL-C2

Oper Source Template: org-root/org-Rossel-VDN/ls-Rossel-VDN-SPT-RHEL-C2

Status:

Bug is identified to be present in the log file itself and it can only be avoided by throwing a exception while the program run. An informational message is given to user, if the bug is found in the log file.

BUG 4:

http://www-tac.cisco.com/Teams/ks/c3/getLargeFile.php?srId=633799045&fileName=sr633799045.20150224021519_akrpcrf3fix1_UCSM.tar

An unhandled exception occurred inside UserTask

Traceback (most recent call last):

File "/var/bdb/task_libs/python/bigdata/generictask.py", line 228, in run

self.action(**self.arguments)

File "/var/bdb/tasks/task_ucs_fc_configure_report.py", line 872, in action

oper_mode=fc_mode(self)

File "/var/bdb/tasks/task_ucs_fc_configure_report.py", line 585, in fc_mode

return oper_mode

UnboundLocalError: local variable 'oper_mode' referenced before assignment

Status:

The calling function was called statically and could not parse the log file correctly.

The function is now called dynamically and the bug is fixed now

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

In conclusion, the project deals with the automation of log parser for BDB on the CISCO platform, which brings about hardware (ASIC) information, focusing on the given line cards in the logs. It saves the engineer an hour's worth of manual troubleshooting and has already been widely appreciated by the engineers involved in the datacenter switching domain. Before they had to manually analyze and trigger each pass of the log, now they just have to load the scripts onto the box at both the encap and decap ends and provide a couple of inputs to bring about the same effect. The script runs and finds the problem, if present in each pass, thereby bringing about hardware configuration of the ASICs. The bugs and suggestions are notified to me by the engineers using it. The configuration across nexus switches can be dynamic and the script has been designed very efficiently to handle all the scenarios and issues which can arise because of variations.

Future work includes designing a similar script for automating log parser for BDB on the other line cards available for the Nexus 7k platform which are M2, F1, F2 and F3. The goal on the long term is to develop a comprehensive Cisco proprietary tool to bring about hardware verification for BDB for the different ASICs used in the different line cards on the Nexus 7k datacenter switch to provide a complete troubleshooting solution.

REFERENCES

1. Cisco systems Inc. (2013).UCS techzone documentation(1st ed) (Online).
Available:<https://techzone.cisco.com/t5/Layer-2-Features-Services/UCS-elam/ta-p/627034>
2. Cisco systems Inc. (2012).Cisco Nexus 7000 datasheet (1st ed) (Online)
Available: <http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000s-series>.
3. Bill Lubanovic, “Introducing Python”, New York: Shroff/O'Reilly; First Edition (2014),pp 1-300.
4. Wightman's and Adam's Nexus 7K ELAM Page - Eureka, Lamira (Online)
Available: <http://tac-wiki/Nexus7000ElamExample>
5. Nexus 7000 Metro ELAM (Online)
Available: <http://wikicentral.cisco.com/display/DCSTGSW/Metro+ELAM>
- 6.Lamira Exceptions Explained (Online)
Available: <https://techzone.cisco.com/t5/Layer-2-Features-services/ElamException-Flags-Explained/ta-p/484867>
7. How to capture mpls packet with Eureka & Lamira elam on n7k - M2/Earl8
Available: <https://techzone.cisco.com/t5/Layer-3-Services-Features/How-to-capture-mpls-packet-with-Eureka-amp-Lamira-elam-on-n7k/tap/574362>