**IST 659: Server inventory and network management system**

Final Project Report

Instructor: Prof. Chad Harper

Team:

Sharat Sripada

# Introduction

Server inventory and network management at scale has been a pain point for our Engineering Team. With hundreds of servers, switches and switch ports to manage across several production environments it sets a nearly perfect use-case to brace an RDBMS at the backend to store large records of structured data while presenting a simplistic frontend.

Through the knowledge acquired in IST-659 and guidance from Prof. Chad Harper the intent is to design a solution that will help greatly ease the day in a life of a developer.

# Problem Statement

A production (aka prod.) environment typically comprises several servers and upstream switches (CLOS fabric comprising first-hop switches termed LEAF and a SPINE aggregate). Developers are building these prod. environments typically doing the following:

* Add servers to prod. to increase capacity

(OR)

Remove servers from prod. when servicing hardware faults or deprecating servers (due to end of life/expiry of service contracts)

* Configuring upstream switches to facilitate workloads or virtual machines (VM) on prod. servers to communicate with each other across the datacenter
* Setting up NFS mounts etc.
* Shuffling servers between prod. environments

Clearly, this is a cumbersome process proving detrimental to Engineering productivity. The endeavor is to automate the above steps end-to-end offering simplistic workflows via a Graphical User interface (GUI) or Application Programmable Interface (API) and utilizing an RDBMS at the backend.

# Business rules and Entity Relationship Diagram (ERD)

## Abstract

Consider the following narrative explaining the Business rules and scope:

Engineers run their code or features at scale building up prod. environments which would comprise servers and associative switches to facilitate communication.

Each prod. environment will have a dedicated VMWare vCenter managing workloads (Outer: to manage physical servers and Inner: to manage nested hypervisors), Runner virtual-machine (to stage, compile and execute code), an NFS mount-point which includes an ip-address and path, more than ONE dedicated virtual-LAN/VLAN IDs isolating Management, Overlay/data-path and NFS traffic.

Prod. environments are backed by ONE or more servers running VMWare ESX hypervisor and would comprise data related to server-manufacturer, a single ip-address assigned to the server to access it over the Management network, vmkernel ip-address to access NFS, memory and CPU capacities of the server and a parameter indicating whether the server was in-use. If a server were not to be in use it could either be deprecated or temporarily out-of-service servicing a hardware fault/upgrade.

Finally, all servers are managed by TWO upstream Layer-2 switches. The switch would comprise data regarding the manufacturer/vendor (this may also be needed to adjust the switch CLI syntax to automate networking actions), port, port-type, switch ip-address, switch type (Leaf/Spine) and finally more than one virtual-LAN/VLAN IDs isolating Management, Overlay/data-path and NFS traffic mapped to each port.

NOTE

Deprecated servers shall be cleaned up from the production environment correspondingly also cleaning up any network related references to switch or switch port.

## Modeling Entities and Attributes

Based on the abstract following are the entities, attributes & datatypes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Entities** | **Attributes** | **Attribute property** | **Datatype** |
| Production | Prod. Name | Primary Key | varchar |
| NFSIPAddress\* | Required | int |
| NFSShare (path) | Required | varchar |
| OuterVCIPAddress | Required, Unique | int |
| InnerVCIPAddress | Required, Unique | int |
| {VLANs} | Required, Multiple | int |
| Server | Server Name | Primary Key | varchar |
| Manufacturer | Required | varchar |
| IPAddress | Required, Unique | int |
| VMKernelIPAddress | Required, Unique | int |
| MemoryInGB | Required | float |
| CPUInCores | Required | int |
| InUse | Required | varchar |
| {VLANs} | Required, Multiple | int |
| Switch | Switch Name | Primary Key | varchar |
| Manufacturer | Required | varchar |
| Port | Required | varchar |
| PortType | Required | varchar |
| IPAddress | Required, Unique | int |
| SwitchType | Required | varchar |
| FaultTicket | FaultID | Primary Key | varchar |
| Description | Optional | varchar |
| Severity | Optional | varchar |

\*Datatype used for ip-address version 4 is either an unsigned int (4 bytes) or a varbinary(4) (chose int for initial implementation)

## Entity Relationship Diagram (ERD)

The entities, attributes and relationships can be represented using an ERD. The relationships between entities is detailed as:

* **Prod. and Server entities**

A prod. environment will comprise 1 or many servers

(AND)

A server can belong to ONE and ONLY ONE prod. environment, hence a ONE-MANY relationship

* **Server and Switch entities**

A server would have its physical ports on Network Interface Card (NIC) connected to TWO (or multiple) switches.

(AND)

Likewise, a switch can comprise several servers connected to it, hence a MANY-MANY relationship.

A screenshot of a cell phone

Description automatically generated

Figure-1: Entity Relationship Diagram

# Logical Modeling and Enhanced Entity Relationship Diagram (E-ERD)

Translating the ERD in Figure-1 to a logical diagram, we do the following:

* Map entities to tables
* In some cases, prod. name or switch name is too long and sometimes cannot be guaranteed uniqueness so, use surrogate keys instead (int identity datatype)
* All attributes are correspondingly mapped to column names following the best-practices for naming conventions (all lower case & separate with underscore for spaces)
* Map relationships between entities as relationships between tables
* Map the MANY-MANY relationship between Server and Switch entities to a MANY-ONE/ONE-MANY relationship using a bridge or associative table called server\_switch\_info
* Vlans are multi-valued and comprise other non-key attributes like Vlan type indicating Management, Overlay or NFS networks. This is correspondingly mapped to the tables belongs and vlan\_table.

A screenshot of a cell phone

Description automatically generated

Figure-2: Enhanced Entity Relationship Diagram