Cloud Computing Project

Virtual Machines and Virtual Clusters

Prepared By

Rishi Raj Sahu (2014H112178P)

Rachit Magon (2014H112175P)

1. Introduction

This document describes the design & implementation of a virtual cluster using virtual machines using QEMU, KVM and Libvirt – API.

2. System Overview

Service Provided:

We are providing a CLI based single system image for a virtual cluster to the user. The following features are configurable by the user

1. The Number of Nodes.
2. Hard Disk size for each Node
3. Main Memory
4. Number of Virtual CPUs
5. Network Interfaces etc.

The OS instances provided in the Nodes for our project were of Ubuntu 14.04 but any Linux/Windows OS image can be installed in the nodes.

The number of Physical Machines is also modifiable through a little configuration. Load Balancing of the physical machines is also provided through round robin / threshold based algorithms.

Distributed applications like MPI can be configured and run on the virtual cluster provided to the user. Check pointing & Migration is also available for the user.

3. Prerequisites

Machine	Software Package	Specification
Physical (Host)	Operating System	Ubuntu 14.04
Physical (Host)	SSH Tool	openssh
Physical (Host)	Bridging Utility	Brctl-utils
Physical (Host)	Hypervisor	KVM + QEMU
Physical (Host)	Virtualization API	Libvirt
Physical (Host)	User Interface	Virt-manager
Guest VM*	All guest VMs should be homogenous to run distributed applications	

* Our code assumes guests to be Ubuntu 14.04 systems with ssh server & client installed.

4. Architectural Design

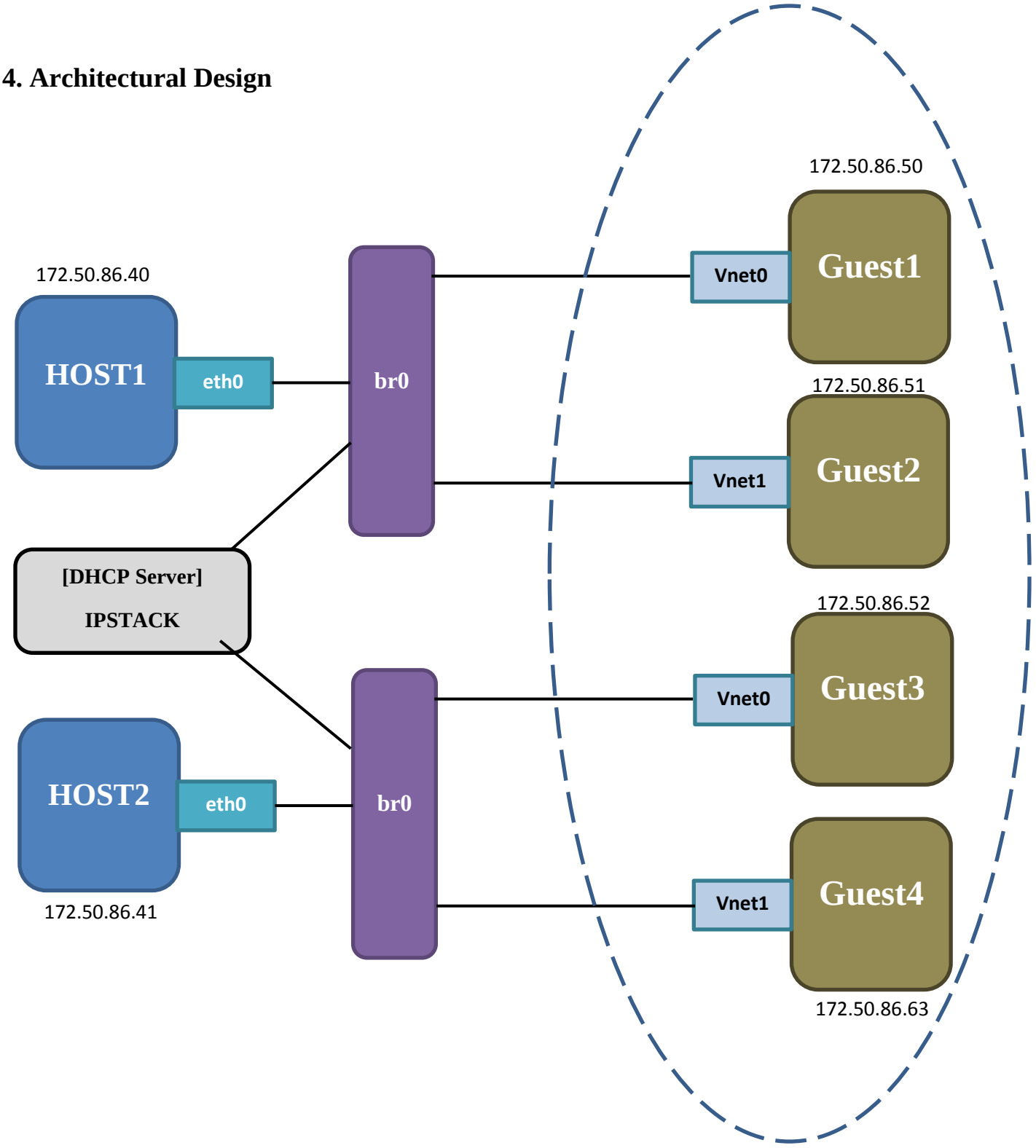

FIGURE 1: Physical View of Virtual Cluster

Figure 1 describes the architectural design of the virtual cluster using bridges. The bridges we have used we configured manually using the brctl utility and the interfaces file of Ubuntu 14.04. The figure 2 (below) shows the logical view of the cluster

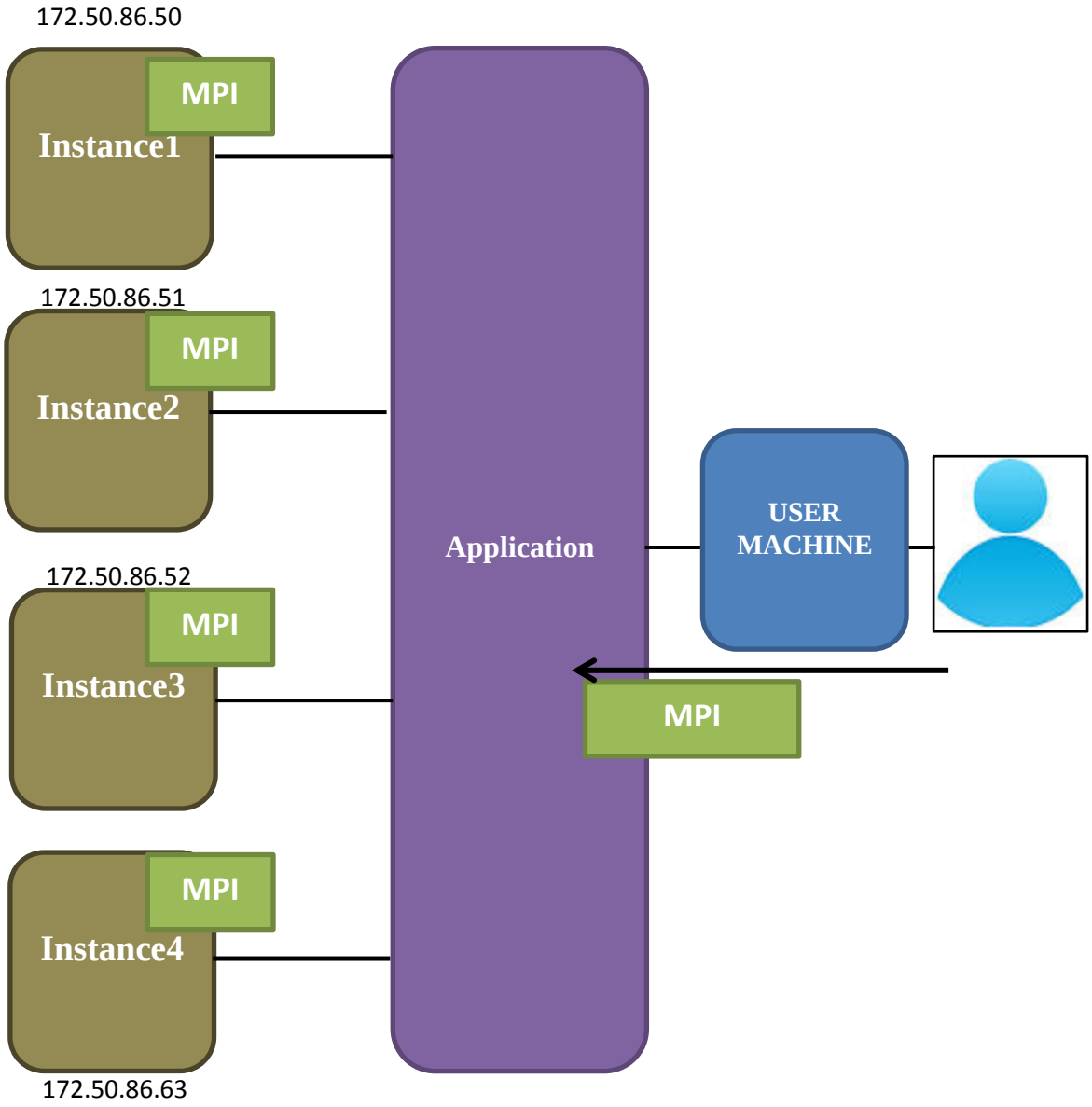

FIGURE 2: Logical View of the Cluster

The user accesses the instances provided to him through our application. For example the user provides the MPI Program to the application to run and the application distributed the program to the instances and gives the user the output.

5. Internal Application Design

The application is a C program which interacts with the libvirt API to manage virtual machines. The internal architecture of the application is shown below:

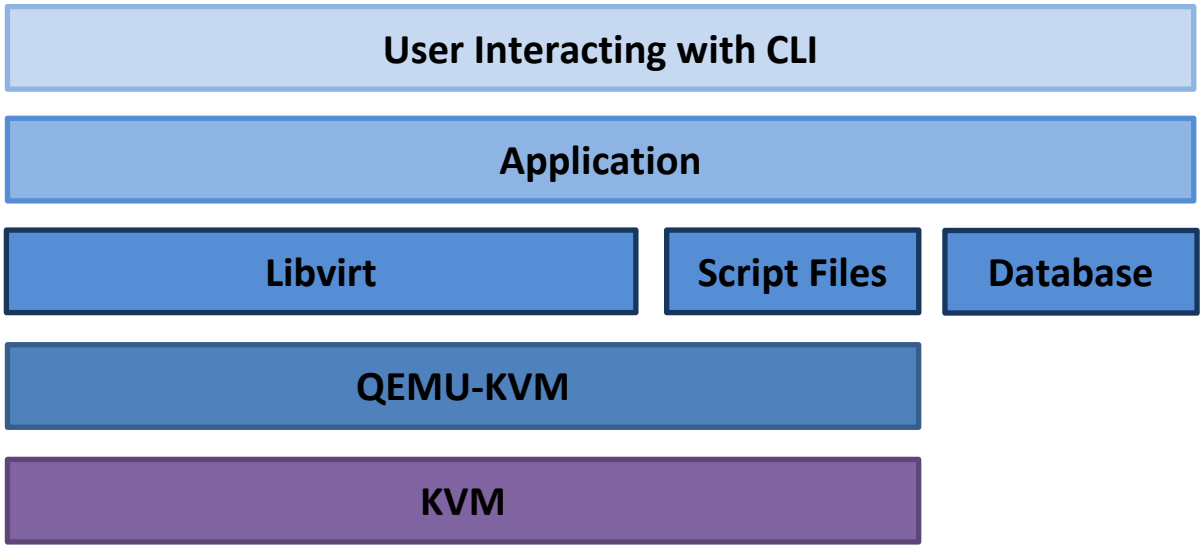

FIGURE 3: Internal Application Architecture

6. Project Structure

Source Code File – init.c

Databases – db-users, txt-login, txt-mac, txt-machine

Scripts – clone, cpu-usage, init-demo, resize, ssh_sendfile, test, vm-createcp, vm-migrate

7. Features of the Application

User Management

The application offers users to sign up and login using their ID and password. Users can only access those machines which are created by them using their credentials. Users cannot log in with invalid credentials. We have provided this feature using the txt-login database, which has the following schema:

id	username	password

```
Virtual Cluster Creator
Select Option to continue
1. Log In
2. Sign Up

1
Enter ID
5
Enter Password [8 char max]
p
Congratulations p
---------YOUR ID IS 5----------
Type OK and Press to Continue
        ---------WECLOME--------------
1.      Print Hostname
2.      CLUSTER Get Cluster Details For User ID=5
3.      CLUSTER Create Cluster Using First Fit Algorithm
4.      CLUSTER Add Using First Fit Algorithm
5.      CLUSTER Create Cluster Using Round Robin
6.      CLUSTER Add Using Round Robin
7.      DOMAIN  Resize an Existing Domain
8.      DOMAIN  Change RAM of an Existing Domain
9.      APPLICATION     mpiRun
10.     MANAGE  Get Physical Address
11.     SERVICES        Create snapshot
12.     SERVICES        Revert snapshot
14.     SERVICES        Migrate Virtual Machine
13.     SERVICES        LogOut
```

Creation of Virtual Cluster

A user can create his own cluster using Round Robin or Threshold Based allocation algorithm. Users also have a feature to add new virtual machines to their existing cluster.

Modification of Virtual Machines

A logged in user can modify the Hard Disk Space, Main Memory and number of Virtual CPUs of the given virtual machine. This is achieved by using the libvirt api for management of a domain.

```
conn = getConnection(IP);
dom = virDomainLookupByName(conn,vmname);

virDomainSetMemory(dom,mem*1024*1024);
```

Running an MPI Program

User can login to the application and select the file he/she wants to execute. The application sends the file to all the machines in the user's own cluster and executes the mpi programs.

Check pointing Virtual Machines

The user can create checkpoints of the running virtual machine and revert back the snapshots in case of failure.

Migrating Virtual Machines

The Virtual Machines can be migrated to any given physical machine. This is a useful feature for load balancing.