A Virtual Cluster Manager using a Hierarchical Management Model for Cloud Infrastructure

Pongsakorn U-chupala*, Kohei Ichikawa†, Hirotake Abe†,
Susumu Date†, Shinji Shimojo†
Kasetsart University* Osaka University†

Outline

- Introduction
- Architecture
- Case Study

Introduction

Cluster Computing

- A group of computers linked together create high performance computing environment
- Enable users to archive high performance computing with commodity hardwares

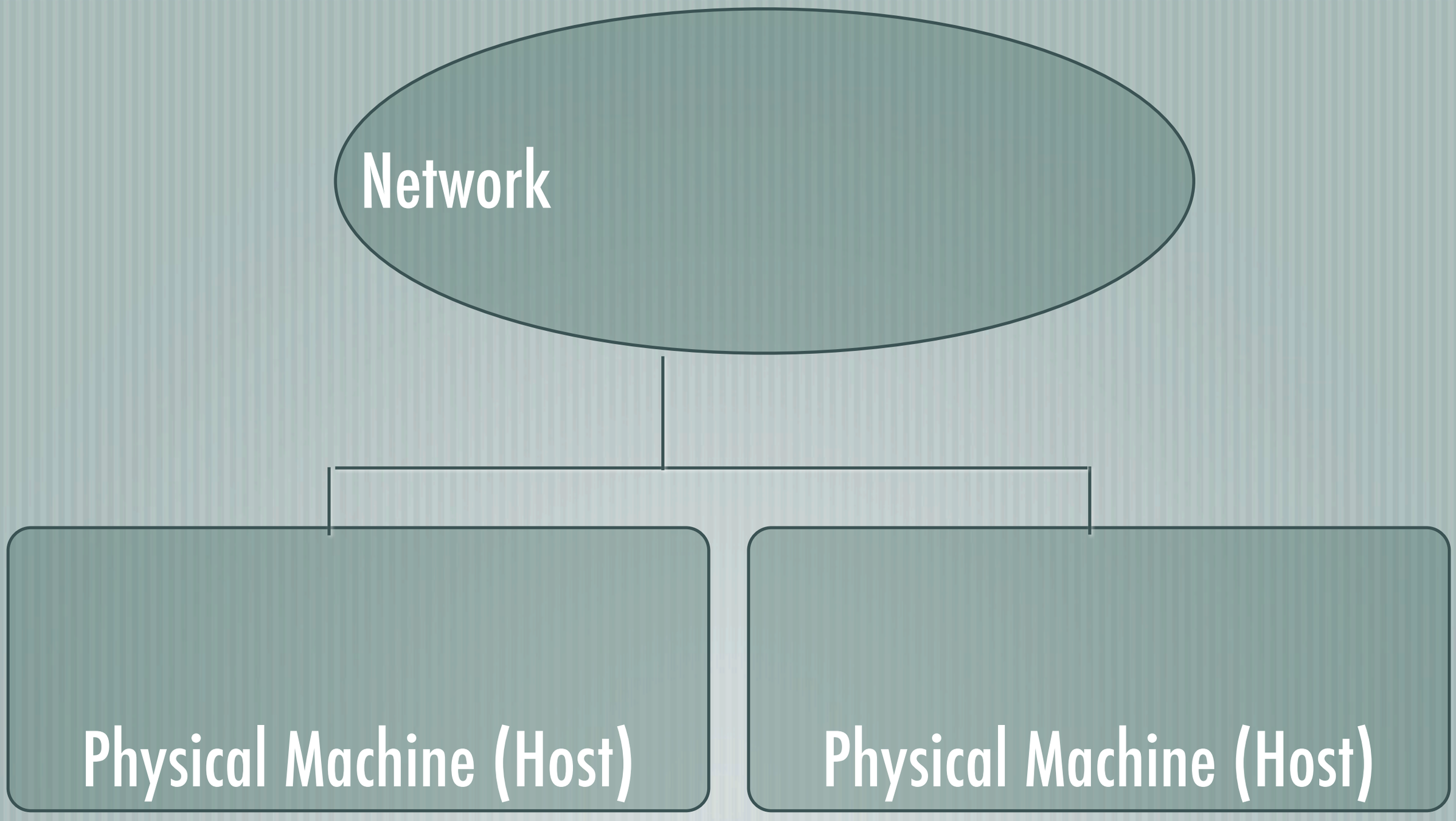
Problems

- We do not always need such a performance all the time
  - Thus, to maintain a cluster system permanently is costly
- Cluster requirements for each application can be different
- Coordinate thousands of machines in a cluster can be a tedious task

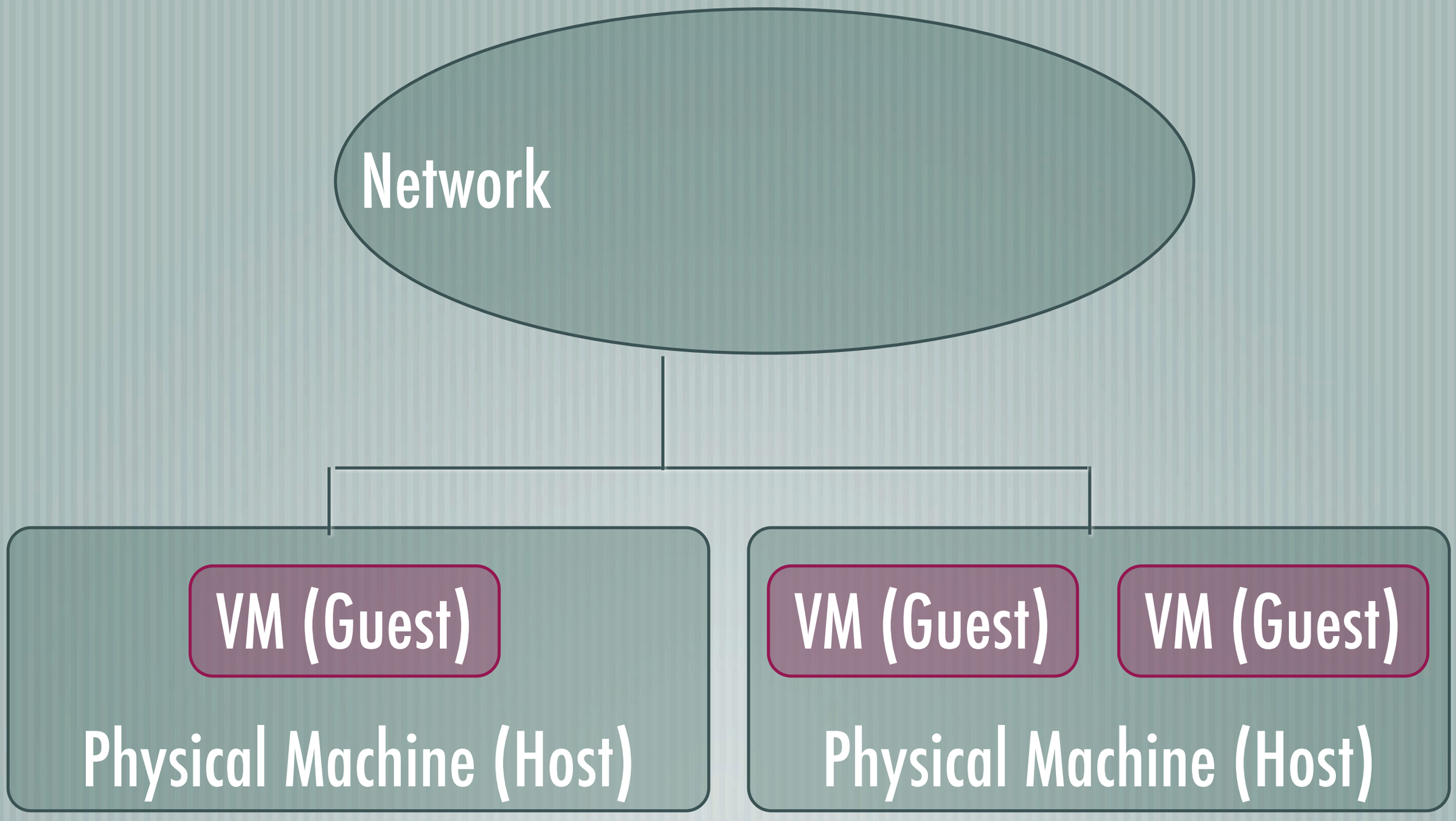
Virtual Cluster (VC)

- Cluster of virtual machines
- Deployed on infrastructure resources pool
- On-demand cluster computing
  - Create when needed, destroy when finished
  - Can be tailored to suit each user need specifically

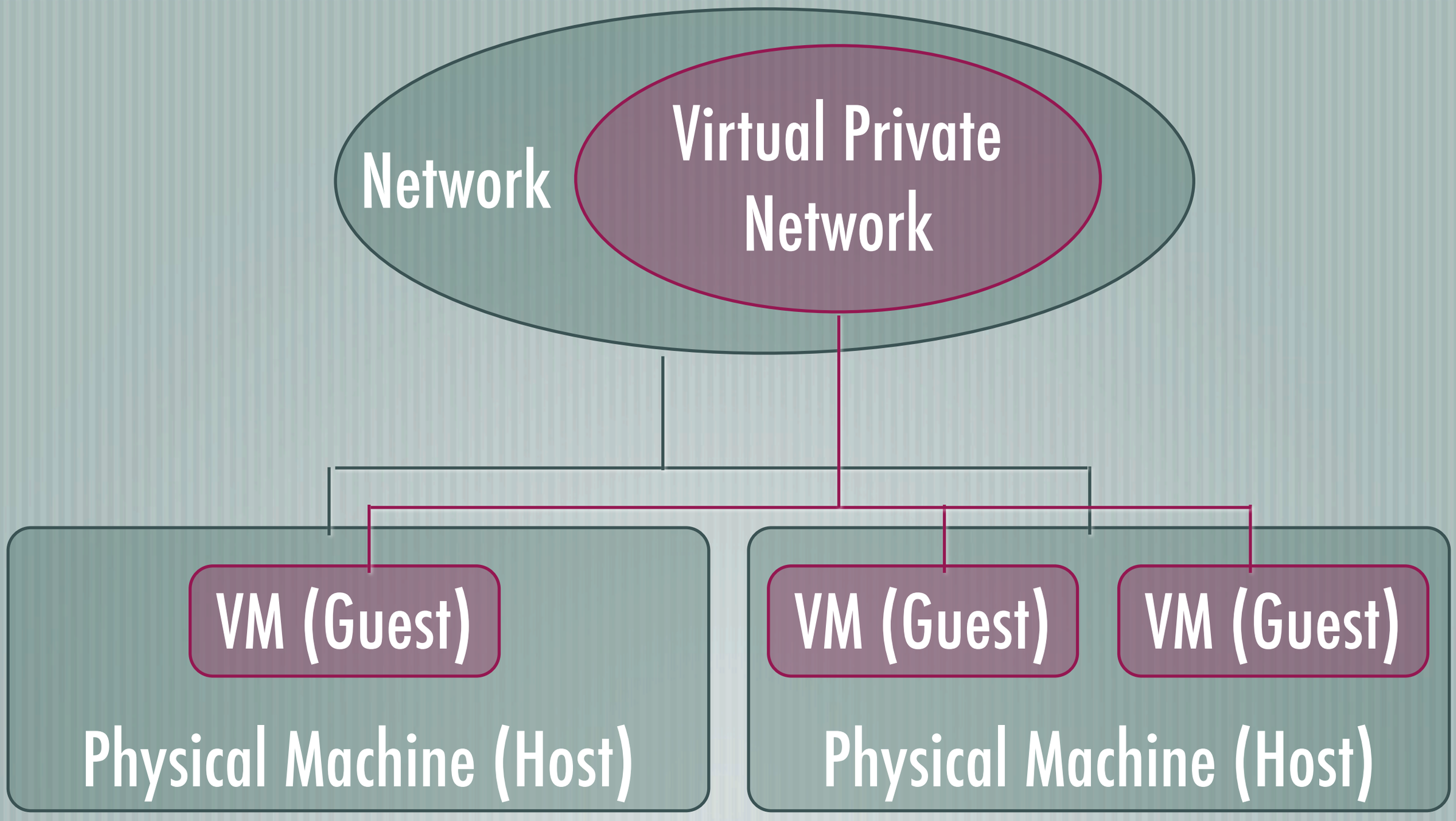
Virtual Cluster (VC)



Virtual Cluster (VC)



Virtual Cluster (VC)



Problems

- ~~We don't always need such a performance all the time~~
  - ~~Thus, to maintain a cluster system permanently is costly~~
- ~~Cluster requirements for each application can be different~~
- Coordinate thousands of machines in a cluster can be a tedious task

Problems

- ~~We don't always need such a performance all the time~~
  - ~~Thus, to maintain a cluster system permanently is costly~~
- ~~Cluster requirements for each application can be different~~
- Coordinate thousands of machines in a cluster can be a tedious task
- Burden of physical infrastructure management

Virtual Infrastructure (VI)

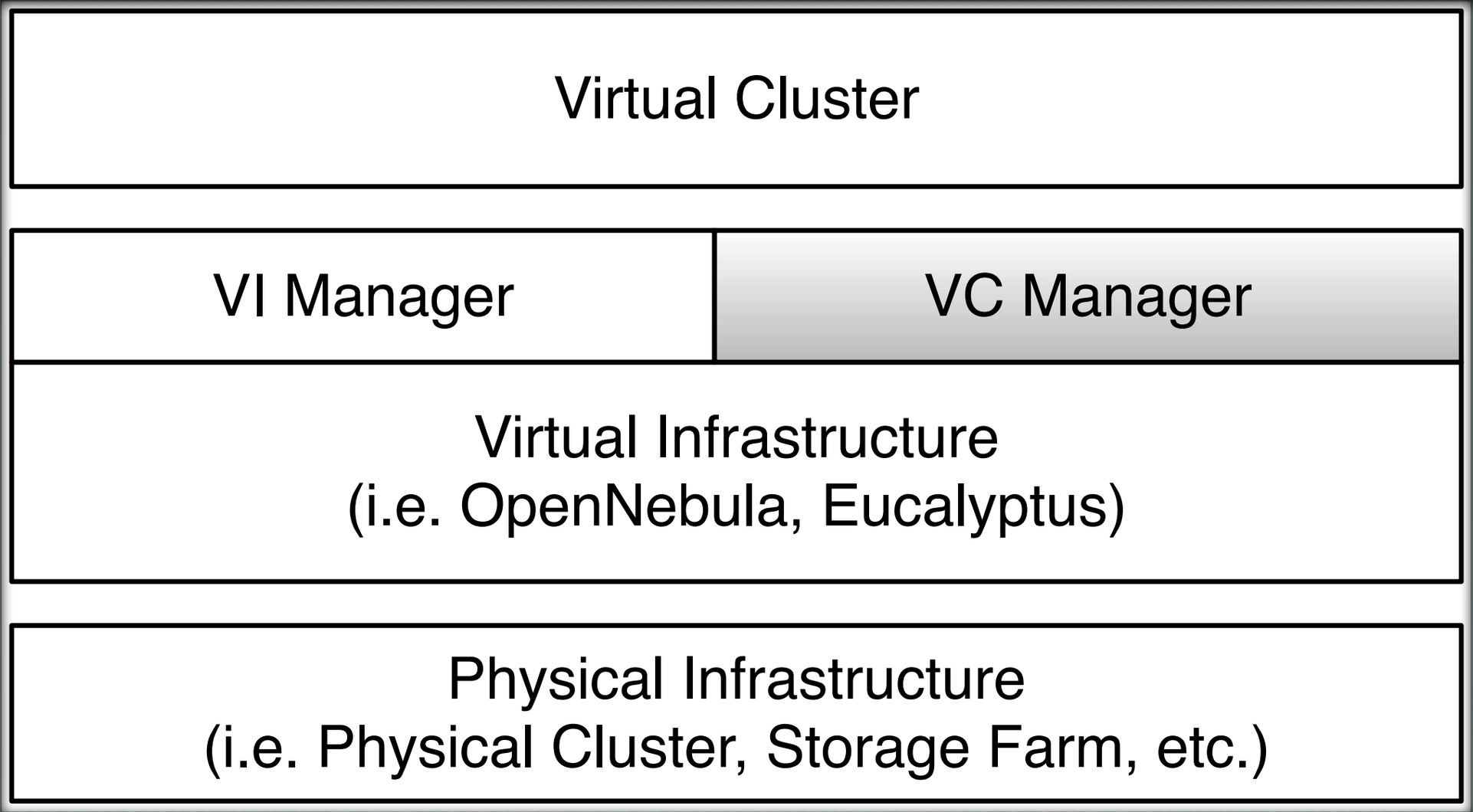
- Pool of virtual resources for Virtual Machines (VM) to use
- A layer over physical resources/infrastructure
- VI toolkits (i.e. OpenNebula, Eucalyptus) come with VI manager
  - A utility that provide methods to manage resources (i.e. deploy VM, allocate disk space, etc.)

Problems

- ~~We don't always need such a performance all the time~~
  - ~~Thus, to maintain a cluster system permanently is costly~~
- ~~Cluster requirements for each application can be different~~
- Coordinate thousands of machines in a cluster can be a tedious task
- ~~Burden of physical infrastructure management~~

VC Manager for VI

- A utility to manage VC
  - By management, we mean deployment, monitoring, etc.
- Work in coordination with VI and its manager



Layer of Abstraction

1. bash

```
Akari:onevc knightbaron$ onevcd start
Akari:onevc knightbaron$ onevc create templates/cluster.one
ID: 1
Akari:onevc knightbaron$ onevc deploy 1
Akari:onevc knightbaron$ onevm list
    ID USER     GROUP    NAME            STAT CPU     MEM         HOSTNAME        TIME
   166 knightba oneadmin Rocks Cluste    runn   0      0K           yozora 00 00:00:43
   167 knightba oneadmin Rocks Cluste    runn   0      0K           yozora 00 00:00:33
   168 knightba oneadmin Rocks Cluste    runn   0      0K             sena 00 00:00:33
Akari:onevc knightbaron$
```

CLI VC Manager

Design Goal

- Single point of operation for these operations of VC
  - Deployment, Stop/Suspension, Deletion, Monitoring
- Flexible enough to manage different kinds of VC
  - Understand complex relationship between each VM in a VC
  - Has the ability to manage VC as a whole as well as each VM in a VC individually

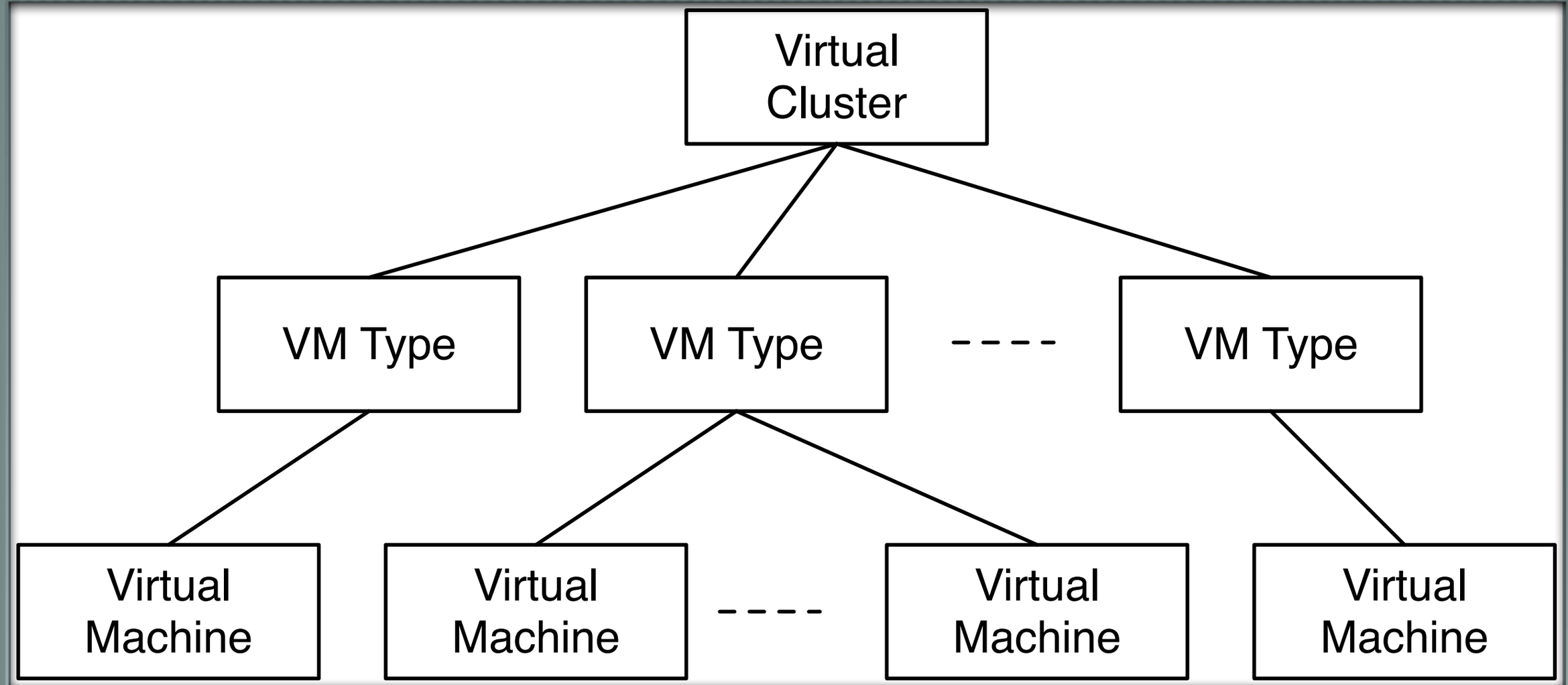
Architecture

VC Representation

- We define VC as a set of VMs
  - Each VM must be grouped into specific VM type
- VC hierarchical model
  - VC hierarchy: represents structure of VC
  - VM type hierarchy: represents dependencies between VM types
- Template system

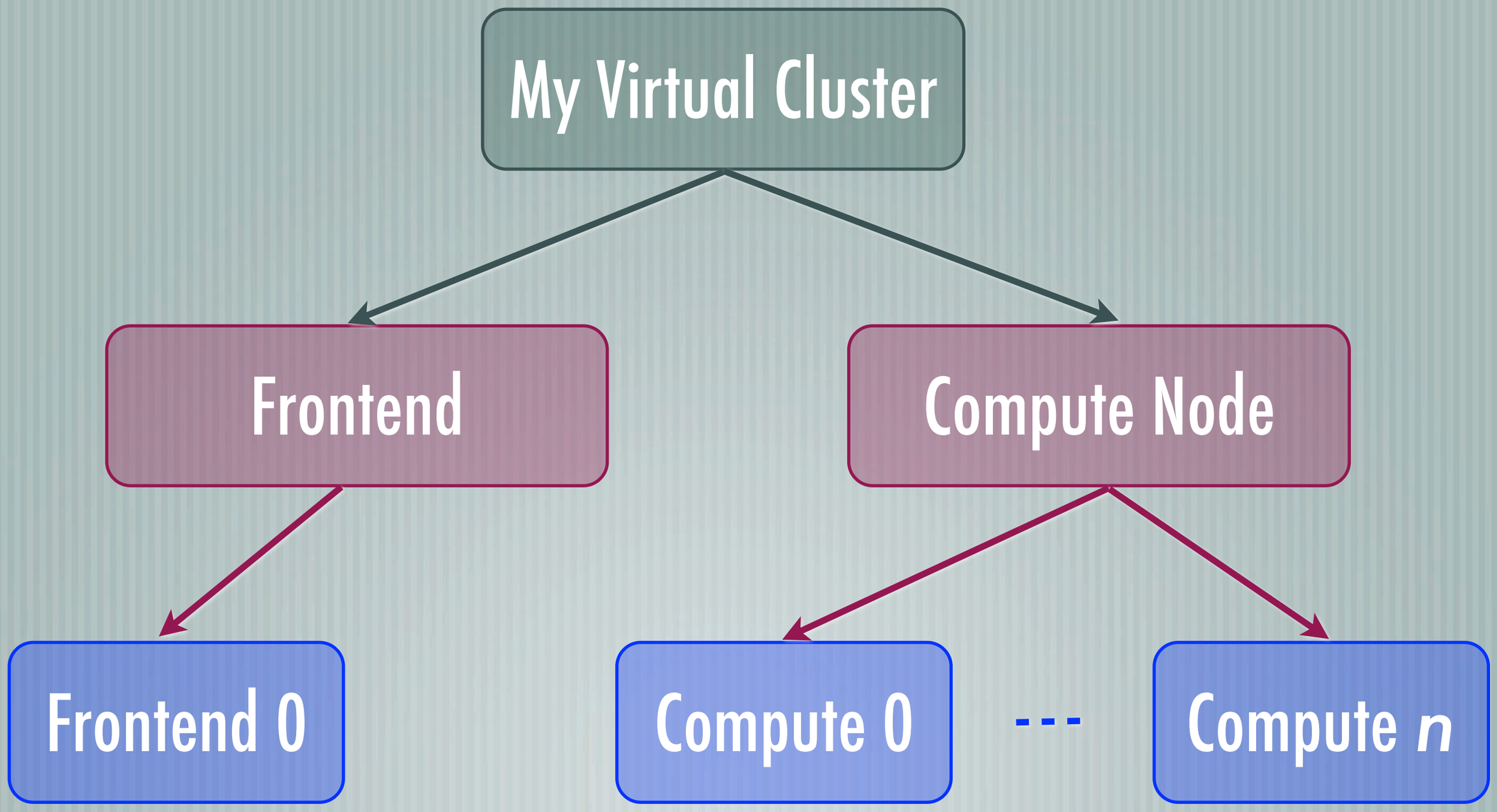
Virtual Cluster Hierarchy

- 3 tiers
  - Virtual Cluster
  - Virtual Machine Type (i.e. Frontend, Compute, etc.)
  - Virtual Machine
- To promote flexibility, reusability of VC configurations
  - Allow configuration override from higher tier
  - Allow manual management of each component in a VC



VC Hierarchy

Example VC Hierarchy



VM Type Sharing

Virtual Cluster

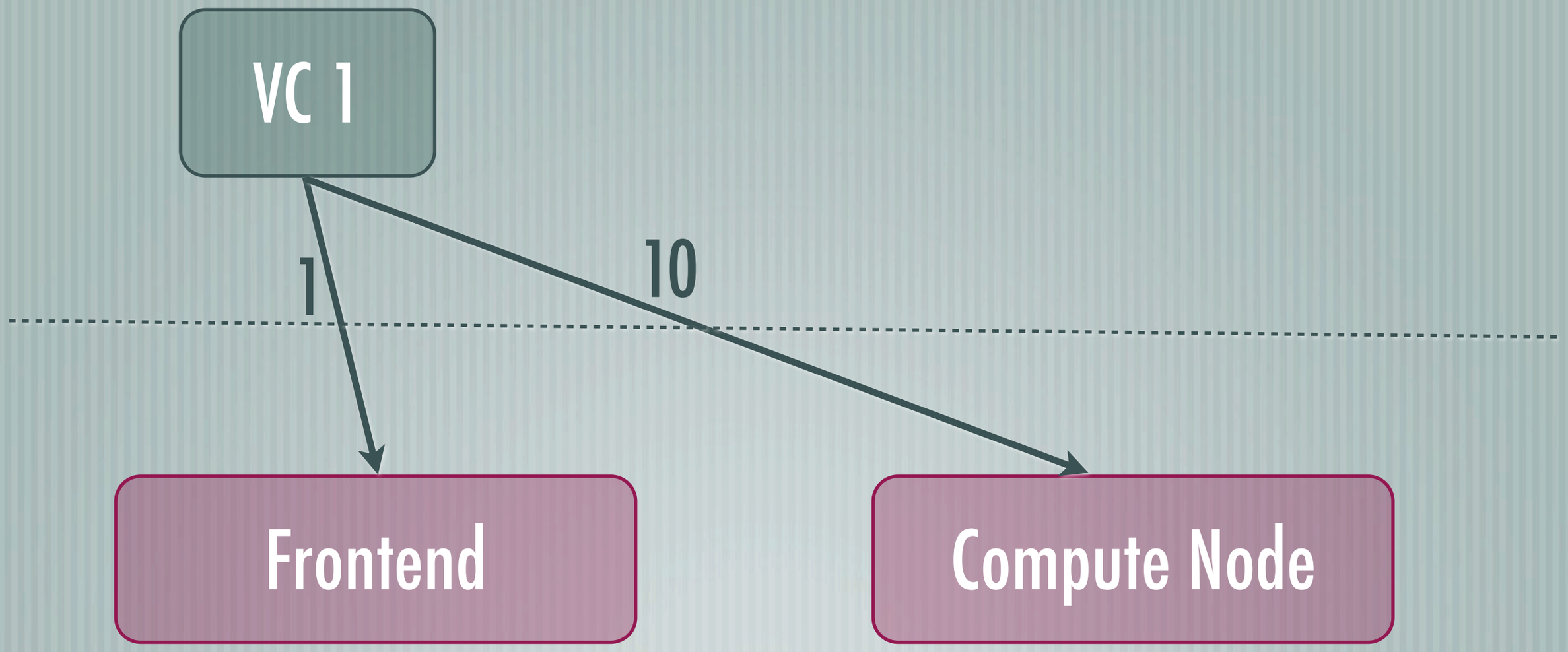
Frontend

Compute Node

Virtual Machine Type

VM Type Sharing

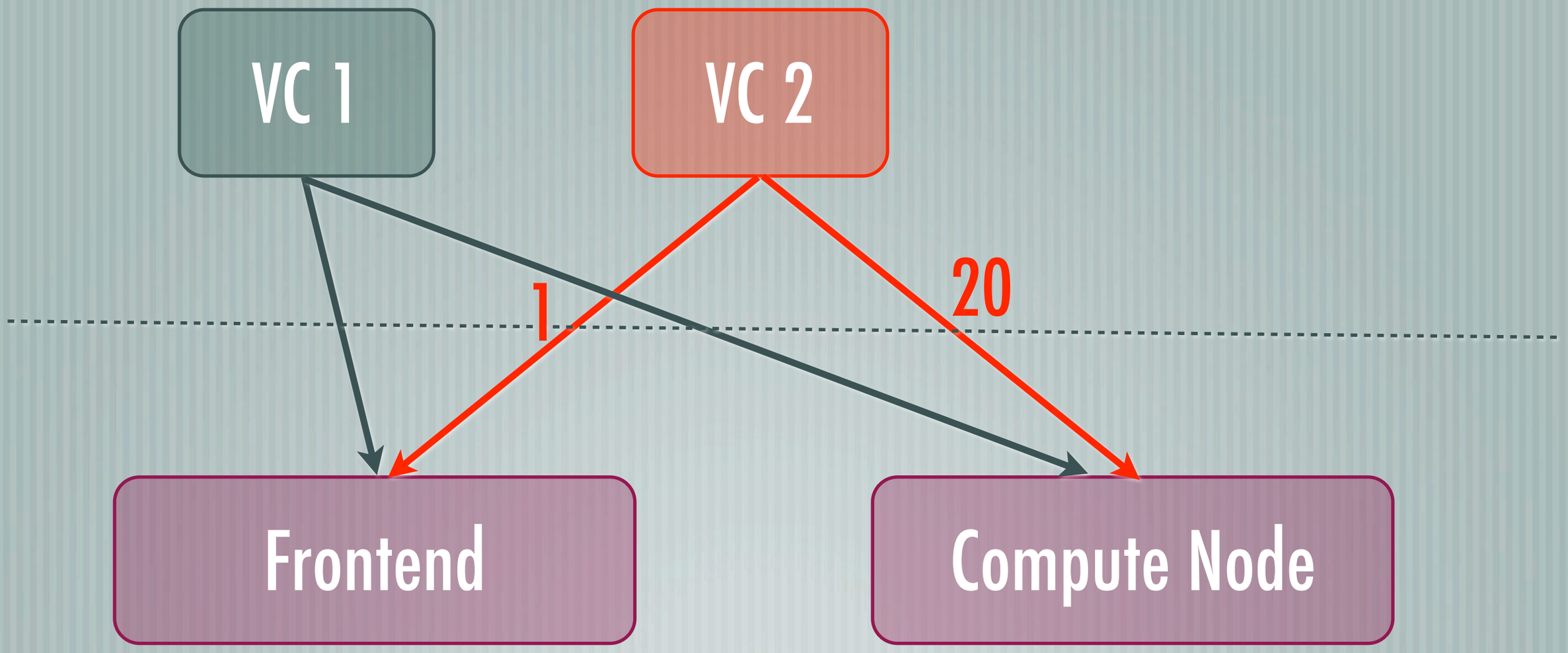
Virtual Cluster



Virtual Machine Type

VM Type Sharing

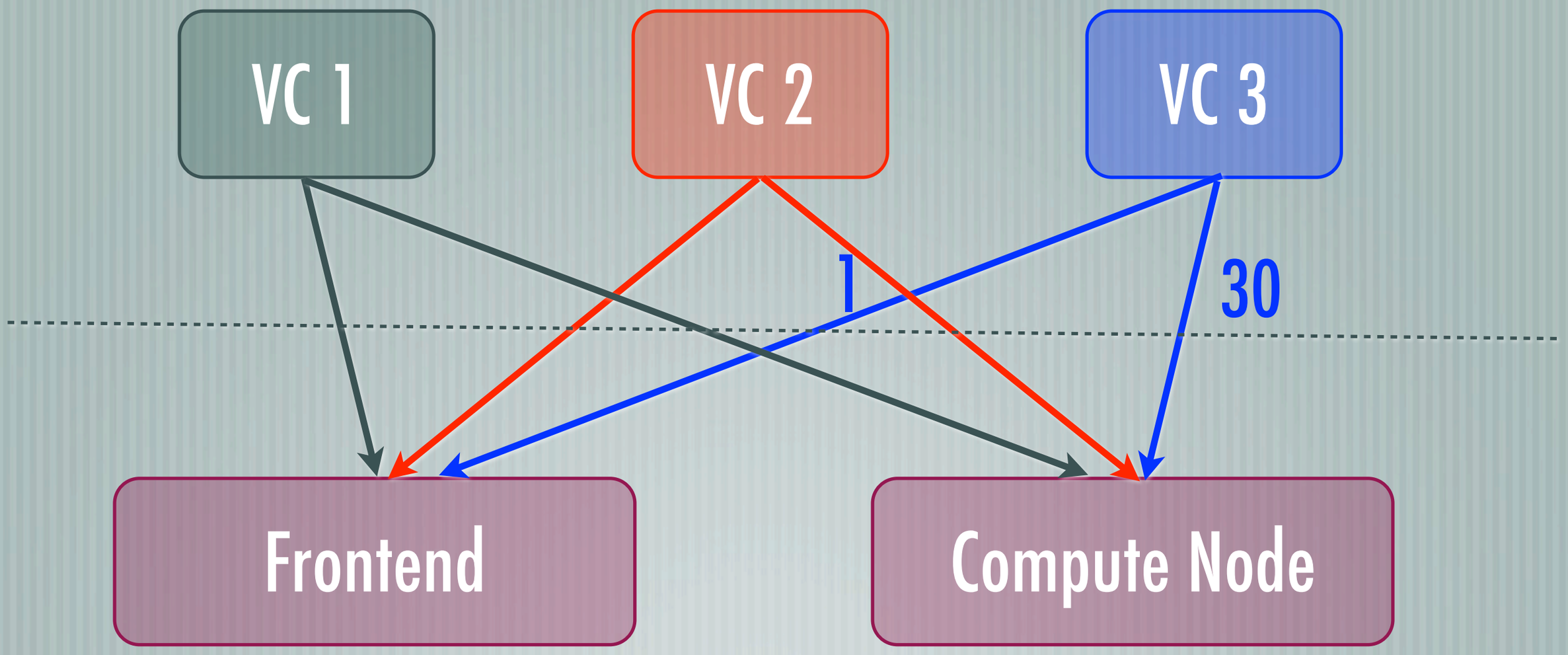
Virtual Cluster



Virtual Machine Type

VM Type Sharing

Virtual Cluster

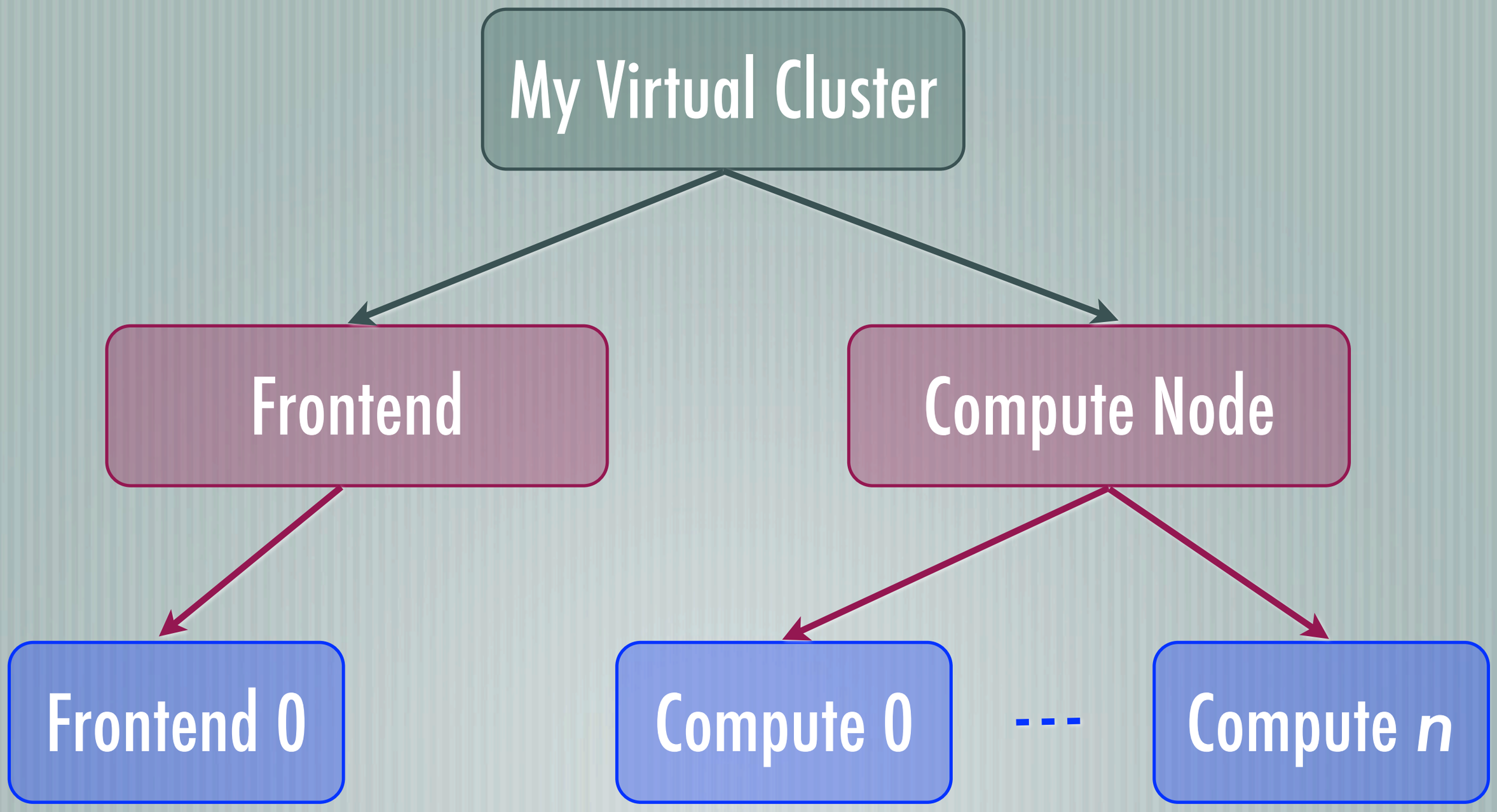


Virtual Machine Type

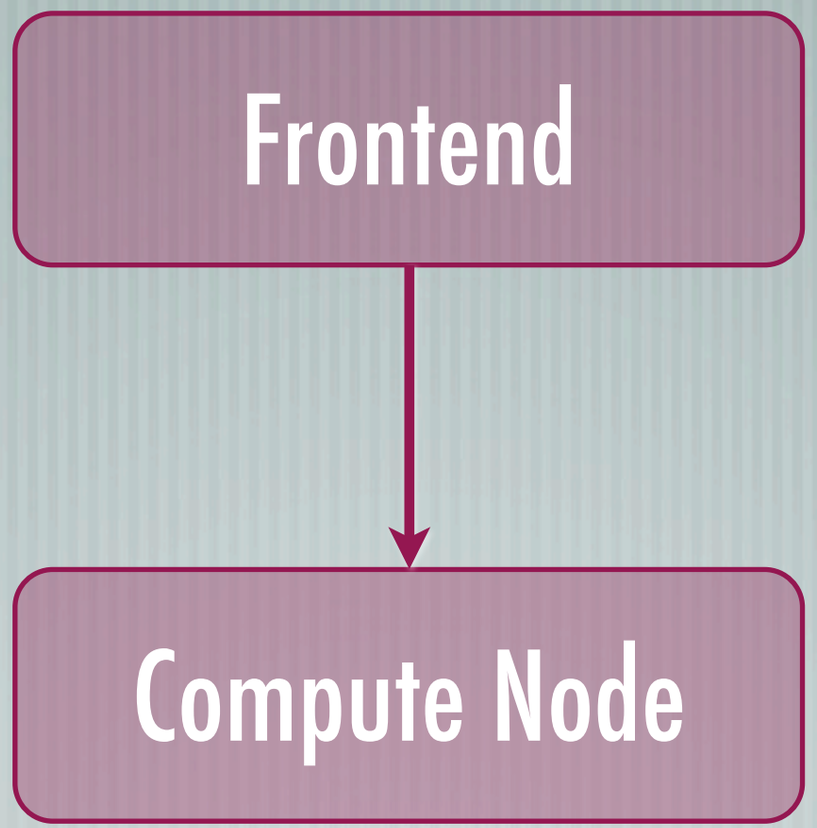
VM Type Hierarchy

- Hierarchy between each VM type in a virtual cluster
- Represents “VM type” dependencies
- Used to determine deployment order of each VM type
  - “Child” VM must be deployed AFTER “Parent” VM

Example VC Hierarchy

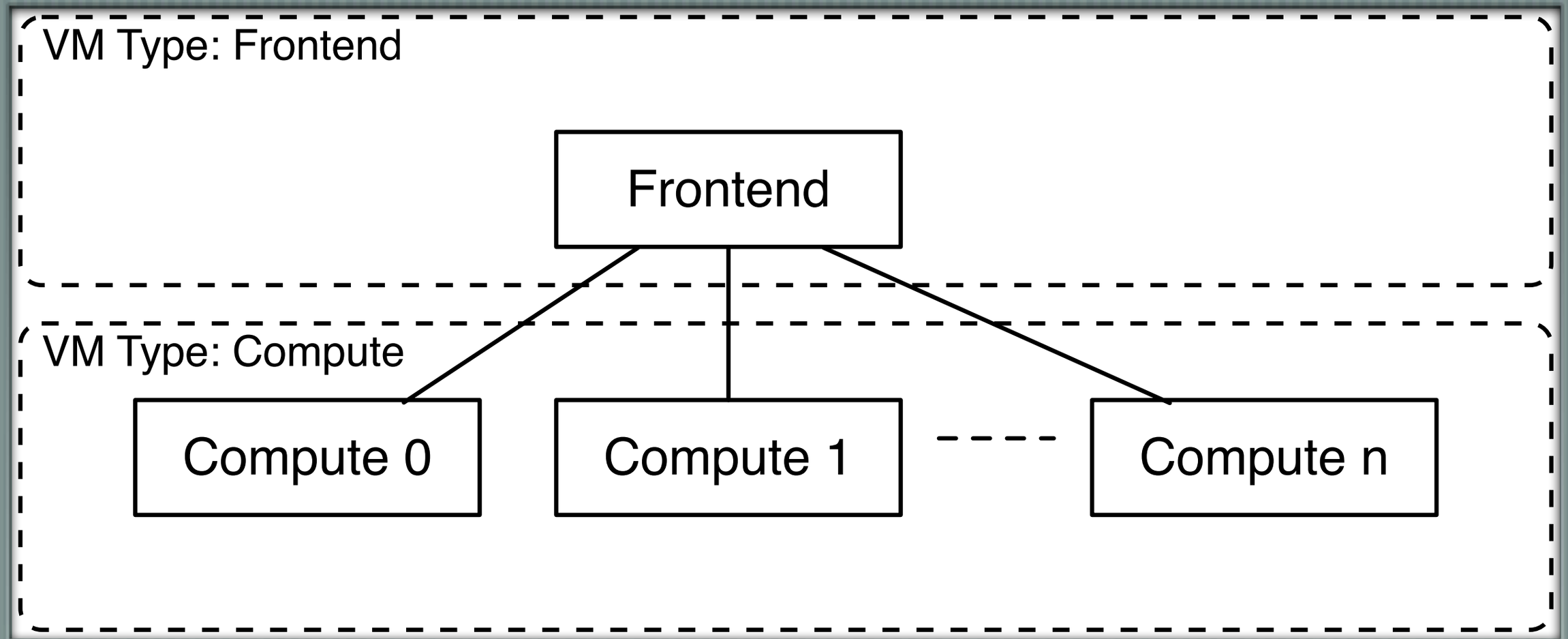


Example VM Type Hierarchy



Template System

- A system to describe virtual clusters, their hierarchies and their configurations
- Divided into 2 types
  - Virtual cluster template
  - VM type template



Example Virtual Cluster

VM Type Configuration File

Name: Frontend
CPU Allocation
Memory Allocation
Virtual Network Configurations
Context Script
Rocks installation Disk Image
Auto-generated Disk Image Specs
Boot Order: HDD, CDROM

VM Type Configuration File

Name: Compute Node
CPU Allocation
Memory Allocation
Virtual Network Configurations
Context Script
Auto-generated Disk Image Specs
Boot Order: PXE, HDD

Example VM Type Template

Virtual Cluster Configuration File

Name: My Cluster
VM Type Configuration Files
- 1 Frontend-typed guest
  - Overridden frontend network configurations

- n Compute-node-typed guests
  - Overridden compute nodes network configurations
  - **Parent: Frontend**

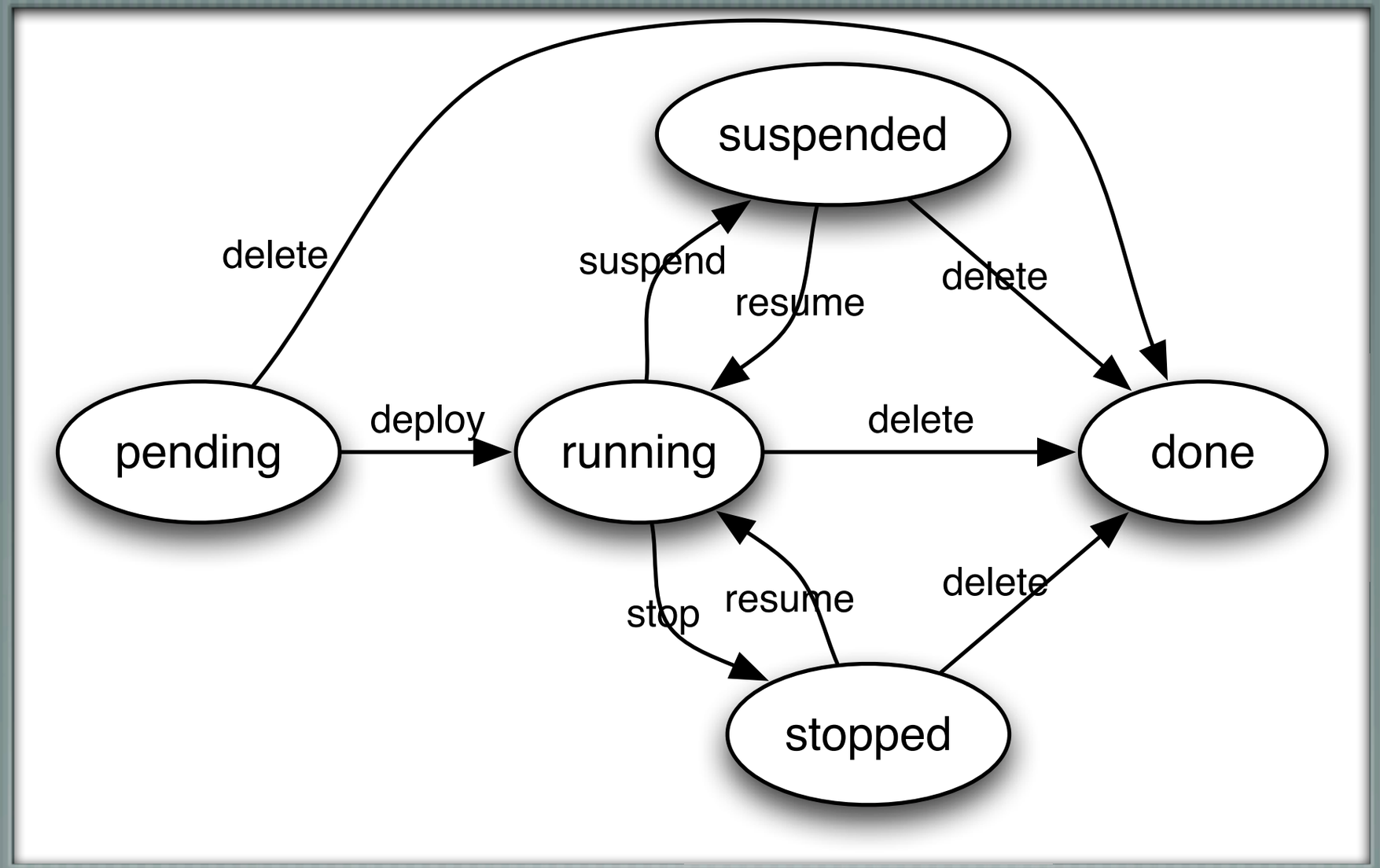
Example VC template

Tracking VC State

- Keep state of each VM type in VCs separately
- Virtual cluster state
  - **If** all VM types in a VC is in the same state (A) **then** VC is in state (A) as well
  - **Otherwise** VC is in “Shifting” (intermediate) state
- Each operation performed on each VCs or VM types transit their states to new one

Benefits

- To make it simpler for manager to handle VC with diverse VM type states
- Allow manual VM type management by user



VM Type State Diagram

Case Study

Case Study

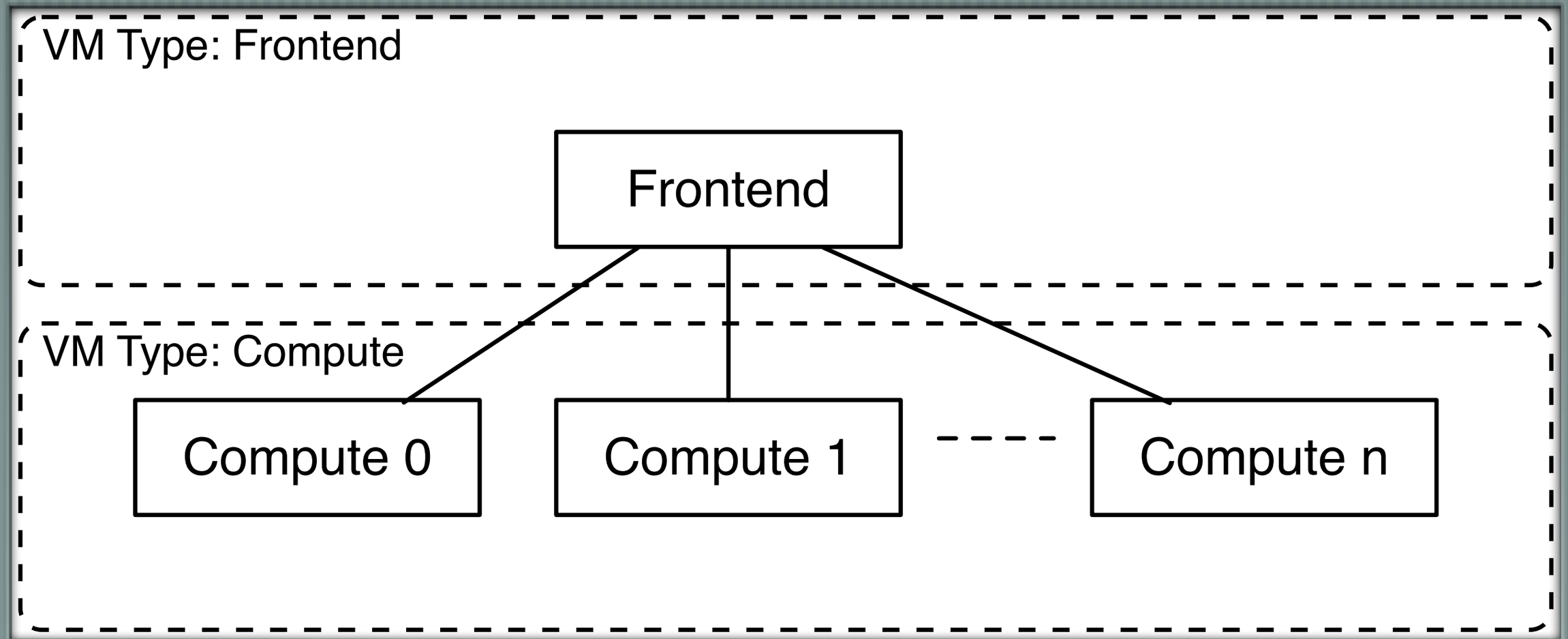
- We have been implementing our VC manager as CLI utility called “onevc” using OpenNebula (VI toolkit) as a building block
- Then did a case study with
  - OpenNebula-provided CLI utilities (represents traditional VC management method)
  - Implemented “onevc” utility

Point of interests

- Ease of use
- Template reusability

Ease of Use (1/3)

- Deploy an example virtual cluster
- Measure “ease of use” with number of commands invoked to deploy a virtual cluster



Example Virtual Cluster

Ease of Use (2/3)

- Traditional method
  - 2 templates used
  - 3+*n* CLI commands invoked
  - *n* is a number of compute node
- "onevc"
  - 3 templates used
  - 2 CLI commands invoked

Ease of Use (3/3)

- Although “onevc” requires 1 more template, a number of commands invoked is a lot less than traditional method especially at higher number of compute node

Template Reusability (1/3)

- Deploy *m* example virtual clusters
  - Each virtual cluster use the **same configurations** for Frontend and Compute node except minor network configurations but has **different number of compute node**
- Measure with total number of templates used to to describe all of the clusters

Template Reusability (2/3)

- Traditional method
  - *m* Frontend templates
  - *m* Compute templates
  - 2*m* in total
- "onevc"
  - 1 Frontend VM type template
  - 1 Compute VM type template
  - *m* VC template
  - 2+*m* in total

Template Reusability (3/3)

- Number of templates required by “onevc” is a lot less (in a different order of magnitude) compared to traditional method

Conclusion

- "VC manager" is our approach to simplify virtual cluster management
- Hierarchical model offer flexibility we want for the manager
- 2 case studies we have shown show that our approach is better than current VC management method

Future Work

- Template management solution
- Deeper integration of VC manager and VI manager
- Extends support for multi-site VC

Acknowledgement

This research is partly supported by the joint research project titled "Research on structuring method of multiple overlay networks leveraging NGN", between Osaka University and **NTT Cyber Solutions Laboratories**. The authors would like to thank **FrontierLab@OsakaU** program for the opportunity.

Thank You