
# PRAGMA Virtual Cluster Sharing Phase 4

U-chupala Pongsakorn, Ichikawa Kohei (NAIST)
Clementi Luca, Williams Nadya, Papadopoulos Philip (UCSD)
Tanaka Yoshio, Ota Akihiko (AIST)
Huang Weicheng (NCHC)

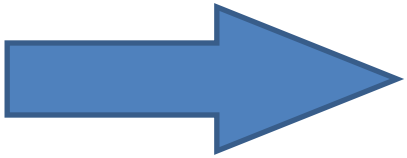
# How it all got started…


- It was a dark and stormy night...

- Kohei Ichikawa (NAIST) was working with some Prime students (from Jason Haga) on DOCK
  - DOCK: large-scale in-silico screening for drug discovery

# How it all got started…


- Nadya Williams was trying to port Lifemaper (Amiee Stewart) on Rocks
  - Lifemapper: builds species diversity map of the world using distributed resources.



Virtual Cluster Sharing

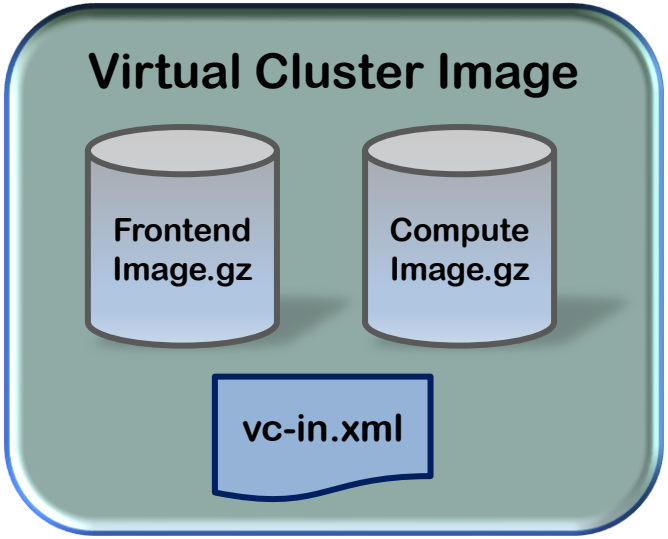
# Virtual Cluster Sharing

We need 2 things:

1. Virtual Cluster Images standard
2. Deployment mechanism

# Virtual Cluster Image

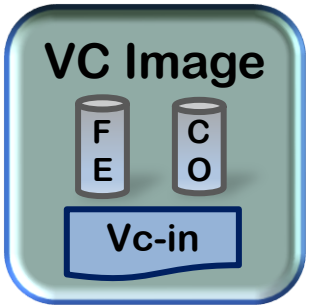
- Define a standard way to share cluster images



# Virtual Cluster Image

Requirements:

- KVM
- Single disk image RAW format
- The first partition is the *root* / partition
- No LVM or RAID!!
- Frontend 2 network interfaces
  - First  private
  - Second public
- Compute 1 network interface
- /root/vc-out.xml for all network configuration



# Virtual Cluster Image

- Rocks Cluster → dynip Roll:
  - https://github.com/rocksclusters/dynip

- Red Hat base systems → vc-out-parse:
  - https://github.com/pragmagrid/vc-out-parser

# Deployment

- Different hosting environments:
  - UCSD uses Rocks Clusters
  - AIST uses OpenNebula
  - …

- How can deploy the Virtual Cluster Image?

pragma_boot:
https://github.com/pragmagrid/pragma_boot

# pragma_boot

- Core application written in python to avoid code replication (where possible)

- “drivers”: it is a set of command line applications which do the work
  - All platform dependent code is in the “drivers”
  - Each driver has a well defined list of arguments
  - Currently we have Rocks and OpenNebula drivers

https://skydrive.live.com/ × Inbox - luca.clementi@gm × california bear - Google Se × pragma_boot/README.rst ×
GitHub, Inc. [US] https://github.com/pragmagrid/pragma_boot/blob/master/README.rst
Award#1148473 - SI... Design of Computer... Software optimizati... coder edition | X86 ... WunderMap® | Inte... Other bookmarks

# The pragma_boot script

**pragma_boot** is the main program to instantiate Virtual Machine in Pragma. It accepts the following agruments:

- **--list** list the available images
- **--num_cpus N** the number of compute node to start up (default to 0)
- **--vcname vcname** the name of the virtual clutster to start up (the name must be in the database)
- **--base_path path** the base path of the VM database
- **--key path** The ssh key that will be authorized on the frontned of the cluster (default is /root/.ssh/id_rsa.pub)

pragma_boot ivokes the follwing subscripts which will be invoked in the order described below. In the commands below the ve_dirver will be replaced with the local Virtual Engine (VE) driver (the base path used to find all the VE drivers can be configured in the file site_conf.conf) site_conf.conf should be used also to set the path for the temporary_directory used for staging all VM images

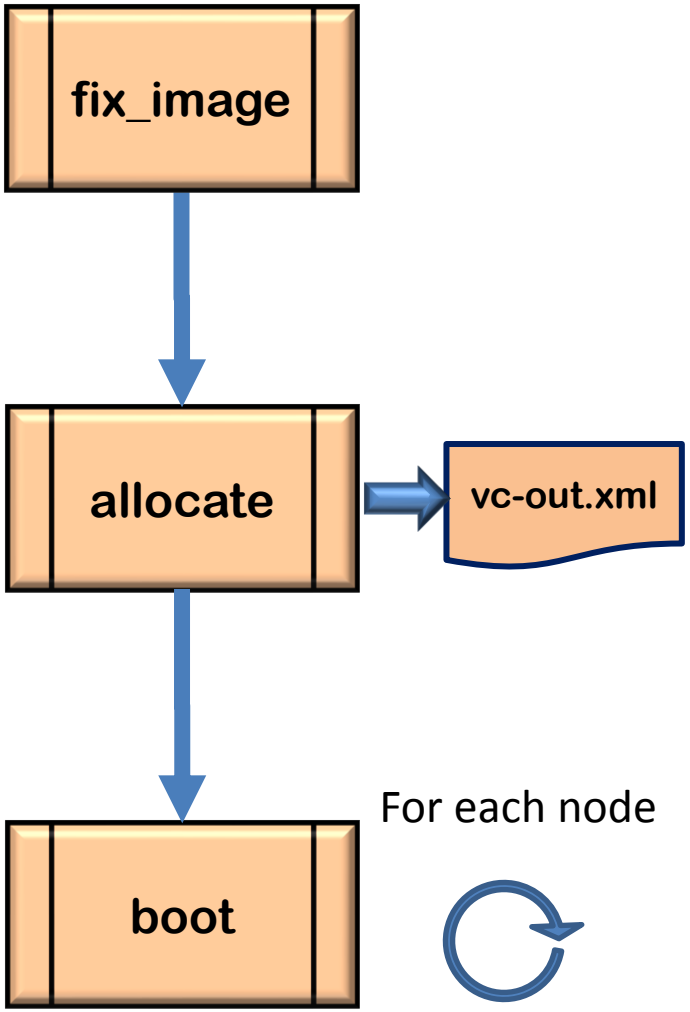
- **ve_driver/fix_images** prepare the given VC images to be run on the current system (fix kernel, drivers, boot options, for current platform, etc.). It's input argumets are (in the following order):
  i. **vc_in_file** the path to the vc-in.xml file of the virtual machine we have to convert
  ii. **temp_directory** the temporary directory used to place all the temporary virtual
  iii. **node_type** a command separated list of node type to be prepared (e.g. "frontend,compute")
- **ve_driver/allocate** this script takes care of verifying that there are enough resoureces to satisfy the user request, if so it will also allocate public IP, private IPs, MAC addresses, and computing resources. If the system can create SMP nodes it can allocate less compute node with multiple cpus in each node. If successful it will write a /root/vc-out.xml file inside the various virtual machines images (see below for more info)
  i. **num_cpus** it specifies the number of CPU requested by the user.
  ii. **vc_in_path** it points to the vc-in.xml of the selected cluster
  iii. **vc_out_path** this should point to the path where the frontend vc-out.xml will be saved

# “drivers”

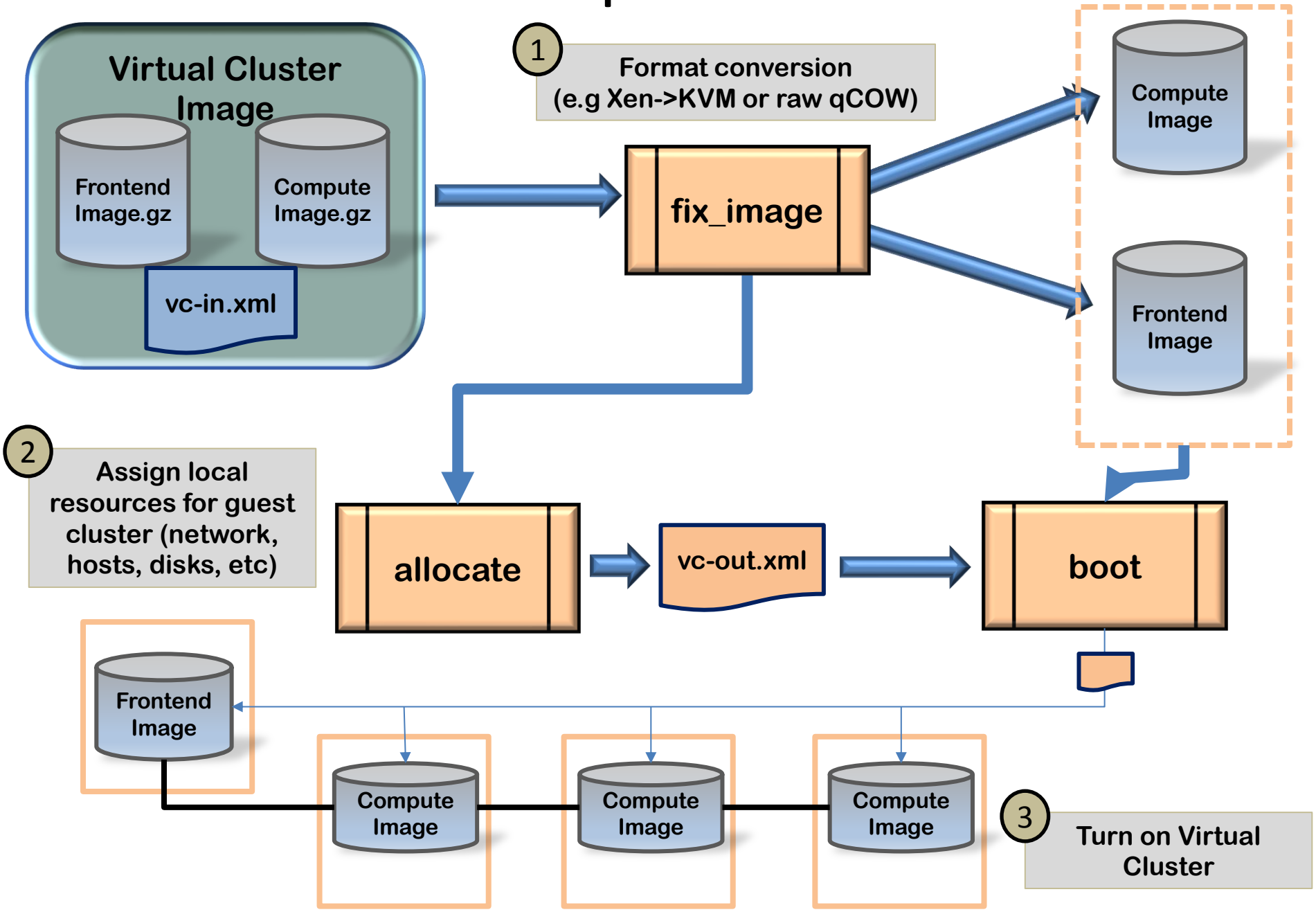
fix virtual frontend and compute node disk images

allocate all needed resources

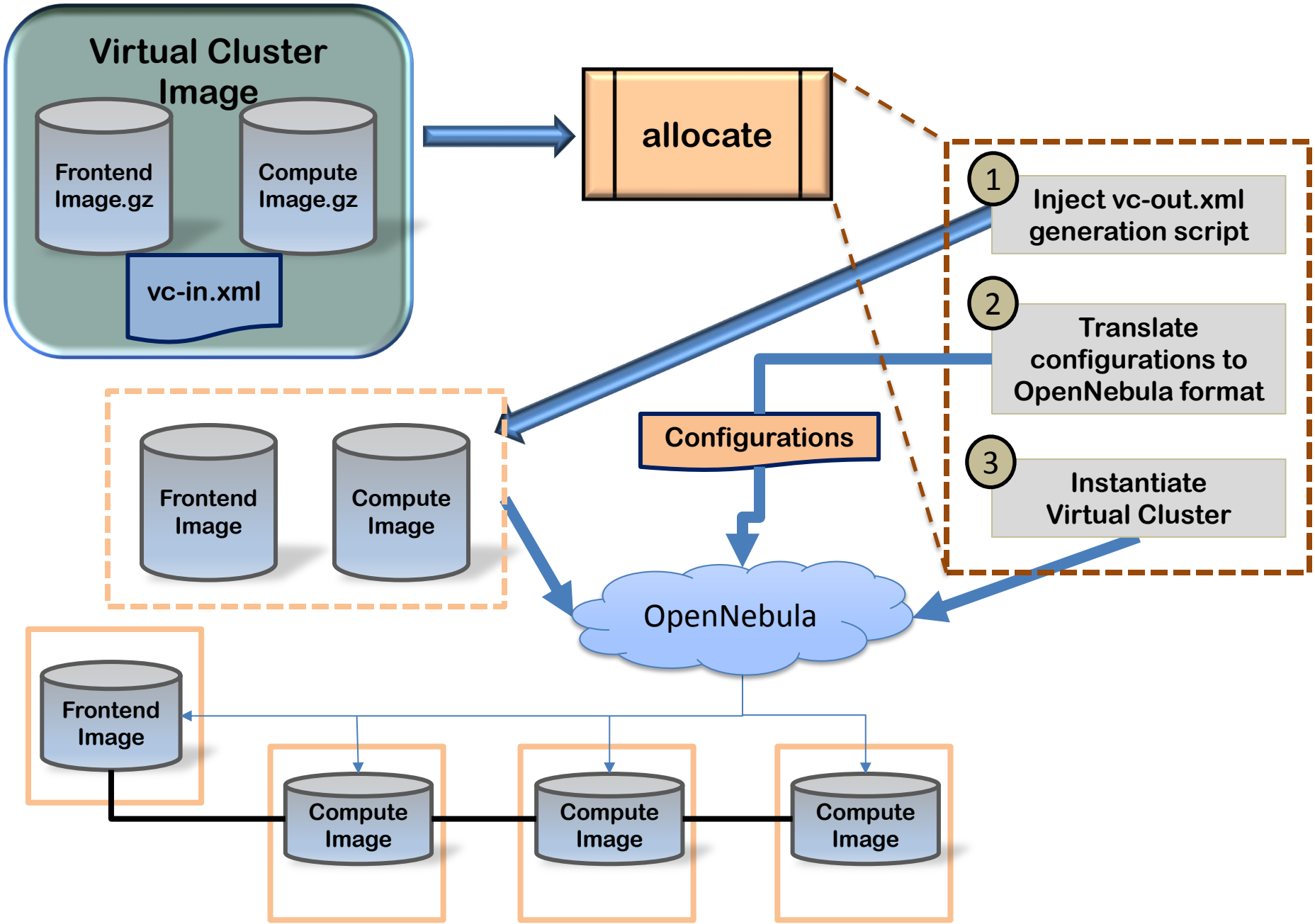
boot 1 node in the cluster



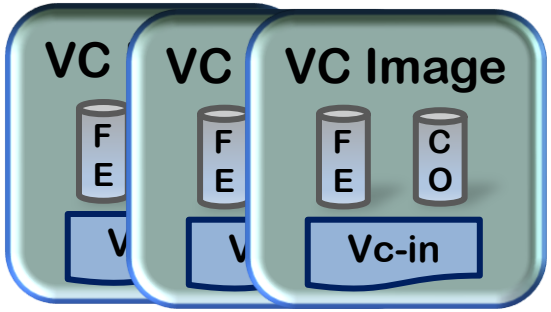
# Rocks Implementation

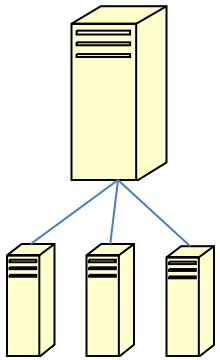


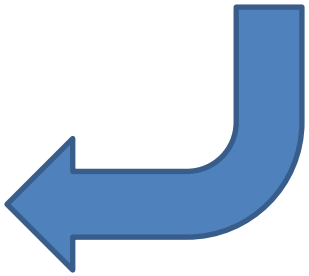
# OpenNebula Implementation

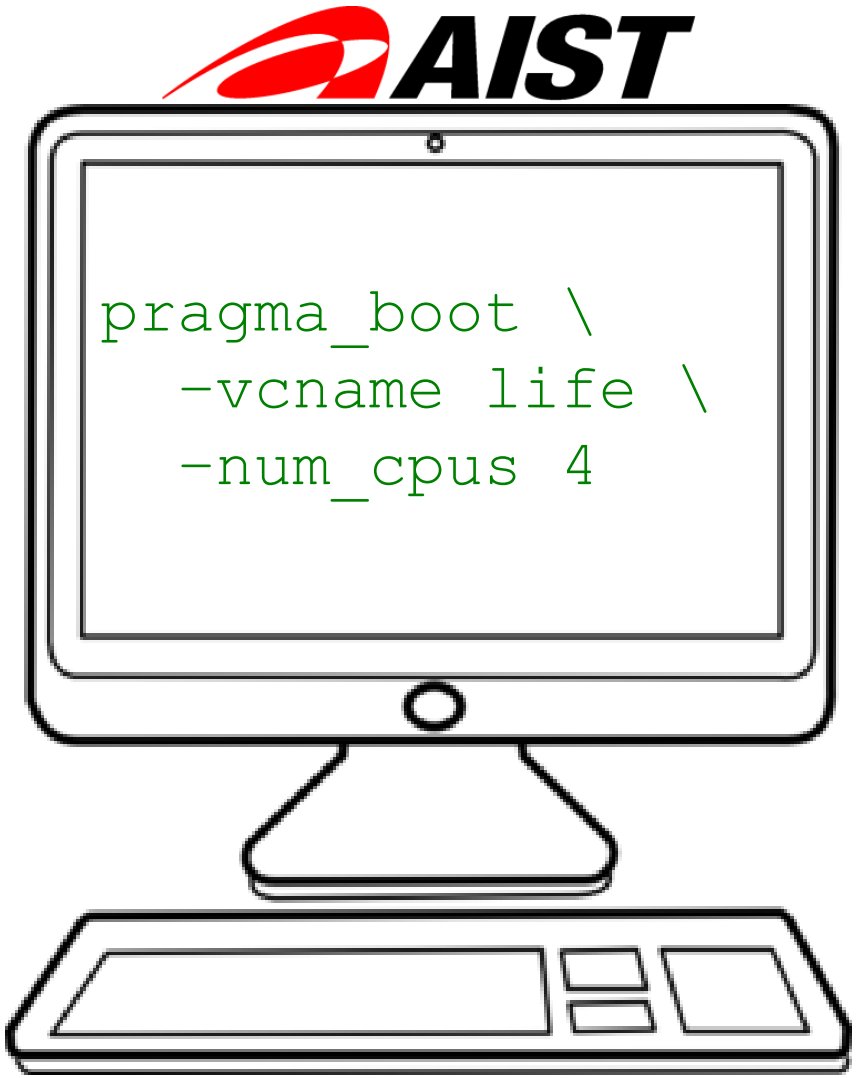


# What is going on?

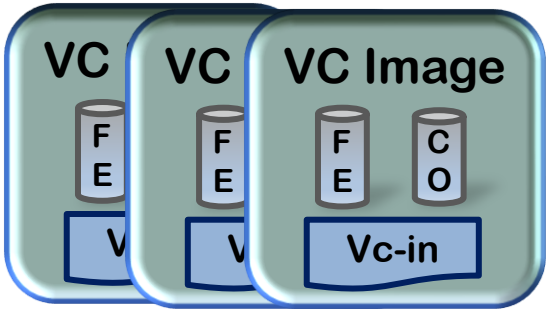



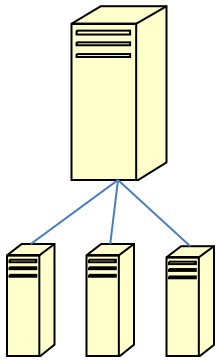


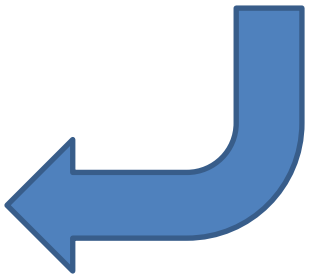


# What is going on?









UC San Diego

```
pragma_boot \
   -vcname dock6 \
   -num_cpus 4
```

# Conclusions

- Do you want to Create Virtual Cluster Image?
  - Rocks cluster:
    - DYNIP: https://github.com/rocksclusters/dynip
  - RedHat:
    - vc-out-parser: https://github.com/pragmagrid/vc-out-parser
- So you want to provide hardware resources:
  - pragma_boot: https://github.com/pragmagrid/pragma_boot

# Thank you!

- Questions?


Contacts:


- lclementi@ucsd.edu
- knightbaron@gmail.com