oVirt

# Getting cross-platform: bringing virtualization management to the PPC world

February 2, 2013

Omer Frenkel
**red**hat

# Who am I ?

- Omer Frenkel
- Software engineer
- oVirt engine maintainer
- Team lead at redhat

# What's going to be here ?

Bringing multiplatform management capability to oVirt,

Initially x86 and PPC64.

- Background
- Problem
- Solution
- What's done
- What's left

# Some Background

oVirt

- The goal
  - Bringing multiplatform management capability to oVirt, Initially x86 and PPC64.
- Why ?
  - KVM on POWER systems announcement
    - http://www-03.ibm.com/press/us/en/pressrelease/41255.wss
  - OpenPOWER Consortium announcement
    - http://www-03.ibm.com/press/us/en/pressrelease/41684.wss
  - Infrastructure for adding support to more platforms

# Some Background

oVirt

- Important credits:

  Fully contributed by developers

  from Eldorado, Brazil.

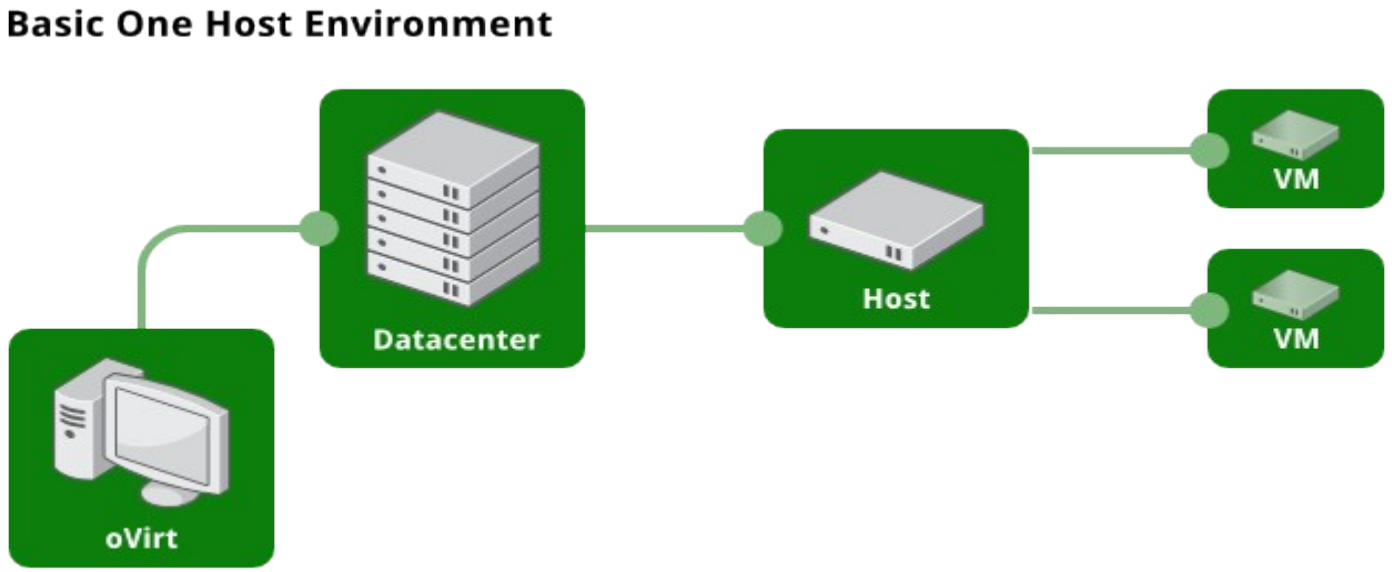
  - Eldorado is a not for profit organization located in Brazil, focused on technology development.

# Some Background

- Contribution process:
  - Design in oVirt wiki
  - Design reviewed by other community members and maintainers
  - Once accepted – implementation
  - Discussions in mail and IRC
  - Working closely with the maintainers for review
  - Became part of the latest oVirt 3.4 release !

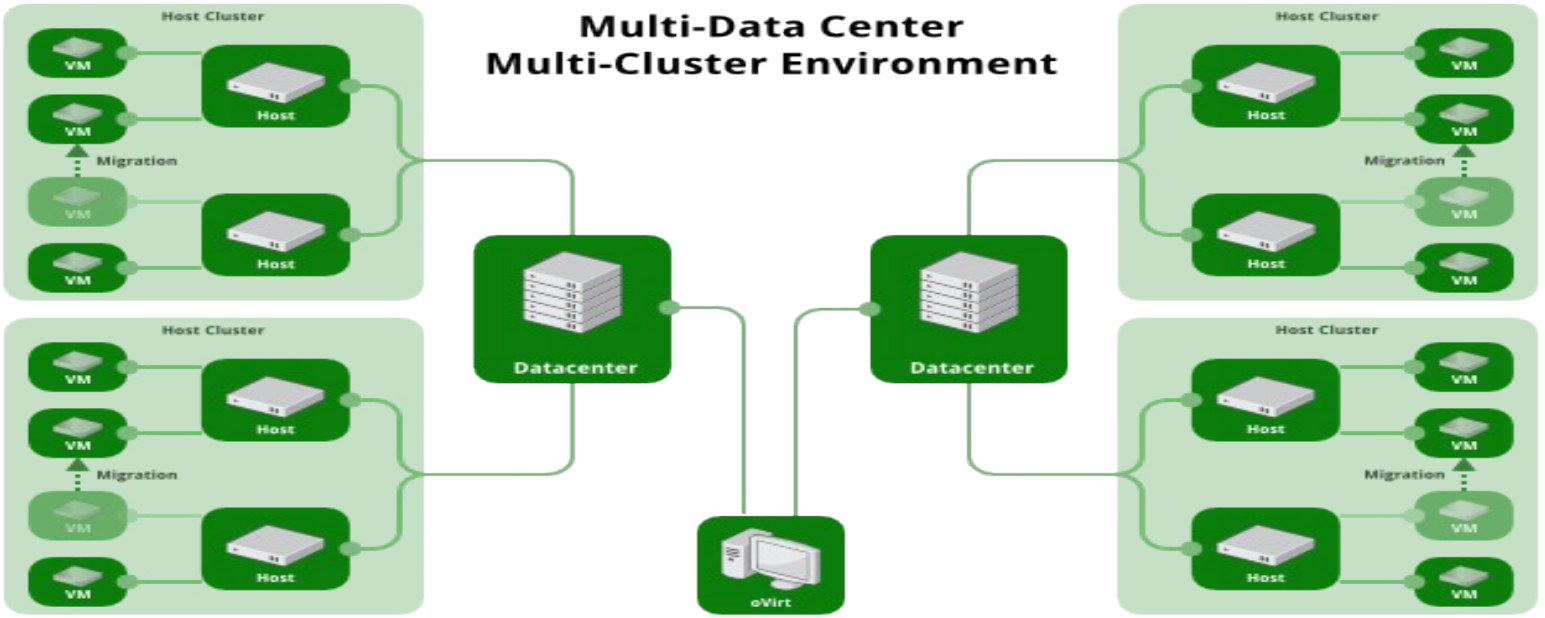
# oVirt

- “Large scale, centralized management for server and desktop virtualization”
- Provide an open source alternative to vCenter/vSphere
- Focus on KVM for best integration/performance
- Focus on ease of use/deployment

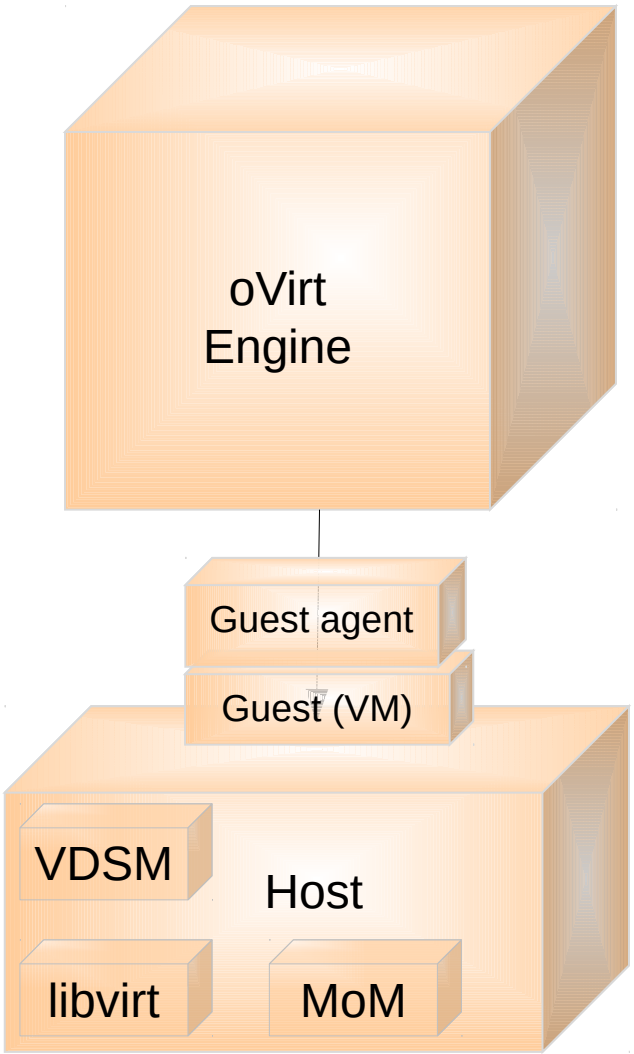


# oVirt

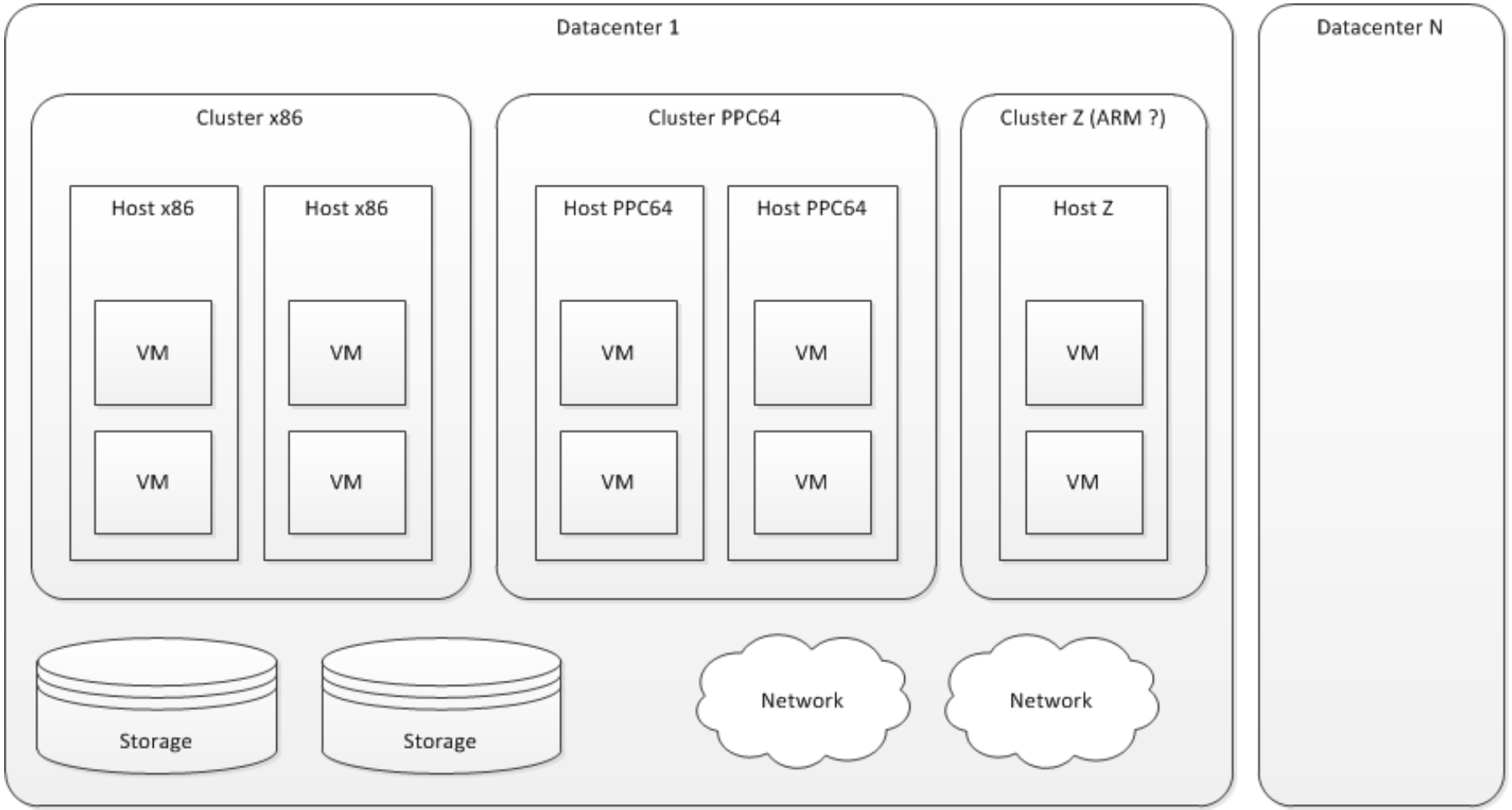
- “Large scale, centralized management for server and desktop virtualization”
- Provide an open source alternative to vCenter/vSphere
- Focus on KVM for best integration/performance
- Focus on ease of use/deployment



# oVirt – High Level Arch



# The Idea




- Goal: Adding multiplatform awareness with minimal changes in UI, architecture and code.

# The Problem

- oVirt designed and developed with single platform in mind.

- No platform specification for vm devices:
  - Network
  - Display
  - Disks
  - ...

    Not all configurations are supported on all platforms.

# The Problem

- For example, supported disk interfaces:

```
public enum DiskInterface {

    IDE("ide"),
    VirtIO_SCSI("scsi"),
    VirtIO("virtio");
}
```

- IDE is not supported by PPC64 architecture, so it need to be filtered by architecture.

# The Problem

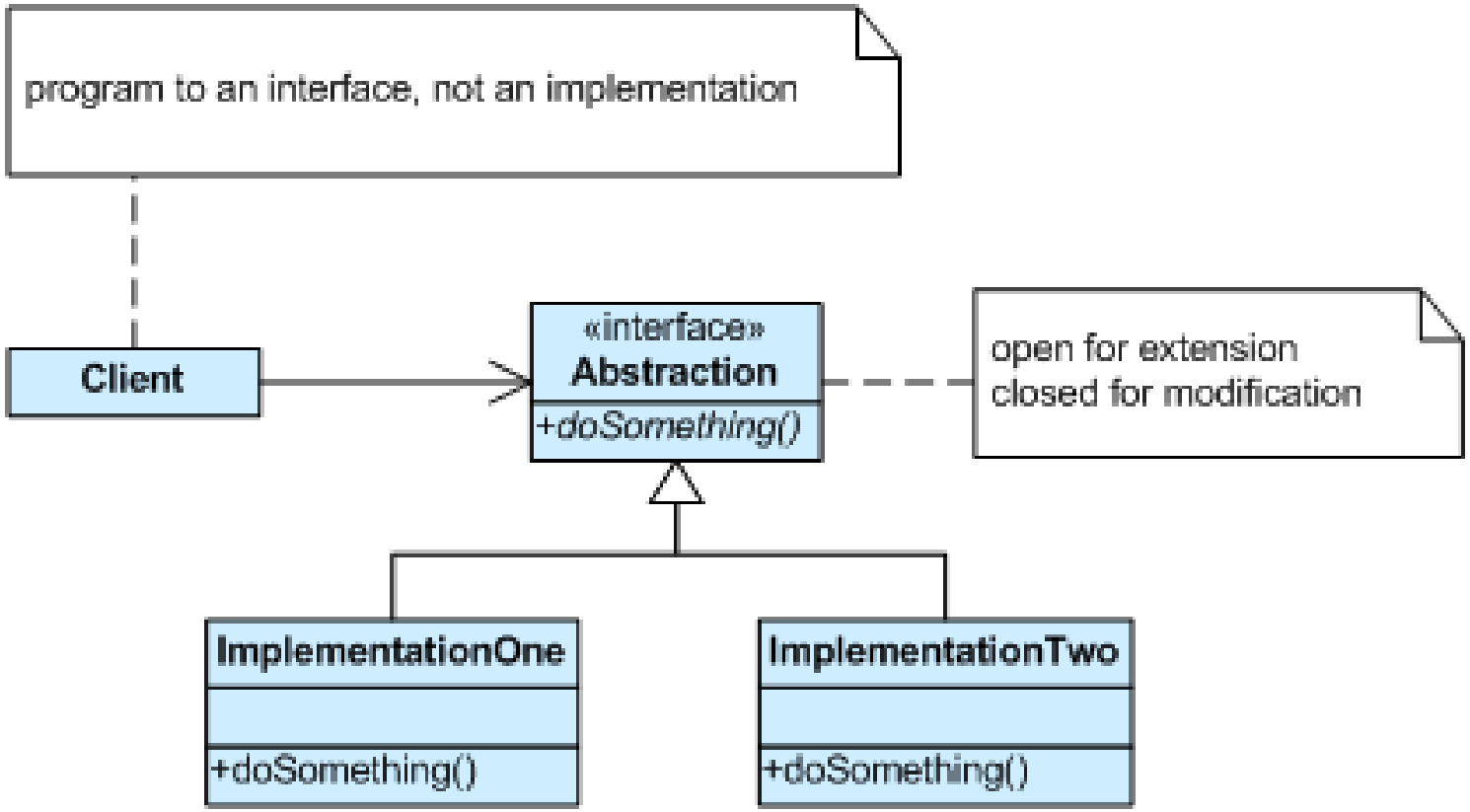
oVirt

- Assumptions that are correct for specific architecture
- Examples:
  - CD PCI addressing is different for PPC64, than what is used today for x86, and may be different for other architectures.
  - VM Live Migration still not supported for PPC64
- The code must change behavior according to the specific architecture.

# The Solution

- Using Strategy design pattern in oVirt,

  to be able to add support for other architectures.



# The Solution

oVirt

- Benefits:
  - Selects a behavior at runtime.
  - Defines a family of algorithms – encapsulates each one.
  - Avoids "if" to switch on architecture behavior.
  - Easy identification of architecture specific code.
  - Easy way to add another architecture and new architecture specific functionality.

# The Solution

oVirt

- Architecture defined in the cluster level
- CPU type reported by host
- Configuration specification
  - Values set per architecture
- Integration with and usage of OSInfo
  - Configurations per guest OS and architecture

# What's done

- Moved x86_64 specific code.
- Application configuration.
- PPC64 code specific development.

# What's done

- Moving x86_64 specific code
  - Architecture field for Cluster and supported CPUs
  - Implementation of Strategy Design Pattern.
  - All the x86_64 specific code was encapsulated in a Strategy.

# What's done

- Configuration w/ config files
  - OSInfo configuration file:

    Settings are defined per OS and per architecture.

  - Benefits: Flexibility
    - Assignment of Lan/Video/Disk/CD for each OS.
    - Filter items in the frontend.
    - Compatibility check.
    - Minimizes architecture specific code.

# What's done

oVirt

- Configuration w/ config files

```
# "Other OS" type to the ppc64 architecture
os.other_ppc64.id.value = 1001
os.other_ppc64.name.value = Other OS
os.other_ppc64.derivedFrom.value = other
os.other_ppc64.cpuArchitecture.value = ppc64
os.other_ppc64.bus.value = 64
os.other_ppc64.devices.network.value = pv, spaprVlan, e1000, rtl8139
os.other_ppc64.devices.cdInterface.value = scsi
os.other_ppc64.devices.diskInterfaces.value.3.3 = VirtIO, VirtIO_SCSI, SPAPR_VSCSI
os.other_ppc64.devices.diskInterfaces.value.3.4 = VirtIO, VirtIO_SCSI, SPAPR_VSCSI
os.other_ppc64.devices.disk.hotpluggableInterfaces.value.3.3 = VirtIO_SCSI, SPAPR_VSCSI
os.other_ppc64.devices.disk.hotpluggableInterfaces.value.3.4 = VirtIO_SCSI, SPAPR_VSCSI
os.other_ppc64.devices.network.hotplugSupport.value.3.3 = false
os.other_ppc64.devices.network.hotplugSupport.value.3.4 = false
os.other_ppc64.devices.display.protocols.value = vnc/vga
os.other_ppc64.devices.watchdog.models.value = i6300esb
# In the ppc64 architecture there are only three devices occupying
# virtual PCI slots in a newly created VM, the USB controller,
# the VirtIO balloon and the VirtIO serial channel
os.other_ppc64.devices.maxPciDevices.value = 29
```

# What's done

- Configuration w/ config files

```
os.other_ppc64.devices.display.protocols.value = vnc/vga
```

```
/**
 * @return The supported display types for the given OS and cluster compatbility version
 */
public List<DisplayType> getDisplayTypes(int osId, Version version);
```

# What's done

- PPC64 code specific development
  - Engine:
    - Addressing Disk and CD
    - SPAPR VLAN and VSCSI (PPC64 specific)
    - Front-end adjustments (UI and REST)
    - Blocking unsupported features
  - VDSM:
    - Topology.
    - Processor name.
    - Hardware information.

# What's done

- Strategy design pattern - Before

```
protected void buildVmDrives() {...
  case VirtIO_SCSI:
    struct.put(VdsProperties.INTERFACE, VdsProperties.Scsi);
    if (disk.getDiskStorageType() == DiskStorageType.LUN) {
      struct.put(VdsProperties.Device, VmDeviceType.LUN.getName());
      struct.put(VdsProperties.Sgio, disk.getSgio().toString().toLowerCase());
    }...
}
```

# What's done

oVirt

- Strategy design pattern - after

```
protected void buildVmDrives() {...
  case VirtIO_SCSI:
    struct.put(VdsProperties.INTERFACE, VdsProperties.Scsi);
    if (disk.getDiskStorageType() == DiskStorageType.LUN) {
      struct.put(VdsProperties.Device, VmDeviceType.LUN.getName());
      struct.put(VdsProperties.Sgio, disk.getSgio().toString().toLowerCase());
    }
    if (StringUtils.isEmpty(vmDevice.getAddress())) {
      ArchStrategyFactory.getStrategy(vm.getArchitecture())
            run(new AssignSCSIAddress(struct, maxUsedLunByController,  disk.getDiskInterface()));
  }
  break;
  case SPAPR_VSCSI:
    ...
  }
```

# What's done

- Visitor design pattern
  - Strategy receives an object and runs the architecture specific code.
  - Visitor class is located in the subproject.
  - Easy to add new architecture specific code.

  - Interface

```
public interface ArchCommand {
  void runForX86_64();
  void runForPPC64();
}
```

# What's done

- Visitor design pattern
  - Implementation:

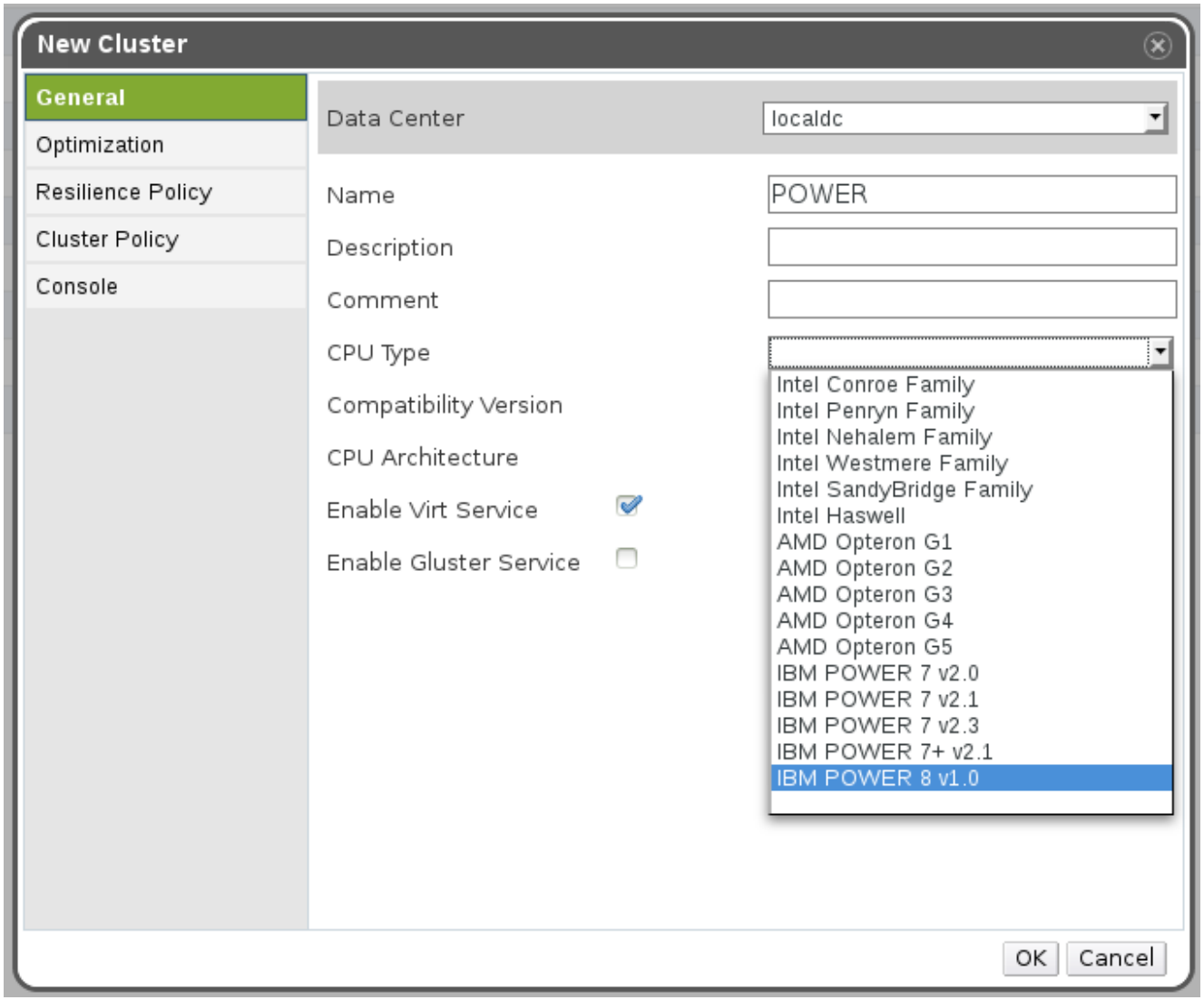
```
public class AssignSCSIAddress implements ArchCommand {...
  @Override
  public void runForX86_64() {
    // In the x86_64 there is only one VirtIO-SCSI controller present.
    // The default address given by libvirt works fine
  }
  @Override
  public void runForPPC64() {
    if (diskInterface == DiskInterface.VirtIO_SCSI) {
      SCSIAddressingUtils.dynamicAddressing(device, maxUsedLunByController, 1);
    } else if (diskInterface == DiskInterface.SPAPR_VSCSI) {
      SCSIAddressingUtils.dynamicAddressing(device, maxUsedLunByController, 0);
    }
  }
}
```

# What's done

- Features ready:
  - Create Clusters, VMs, Templates and Pools.
  - Import/Export VMs and Templates.
  - Attach disks to VMs.
  - Search VMs by architecture.
  - Manage VMs.

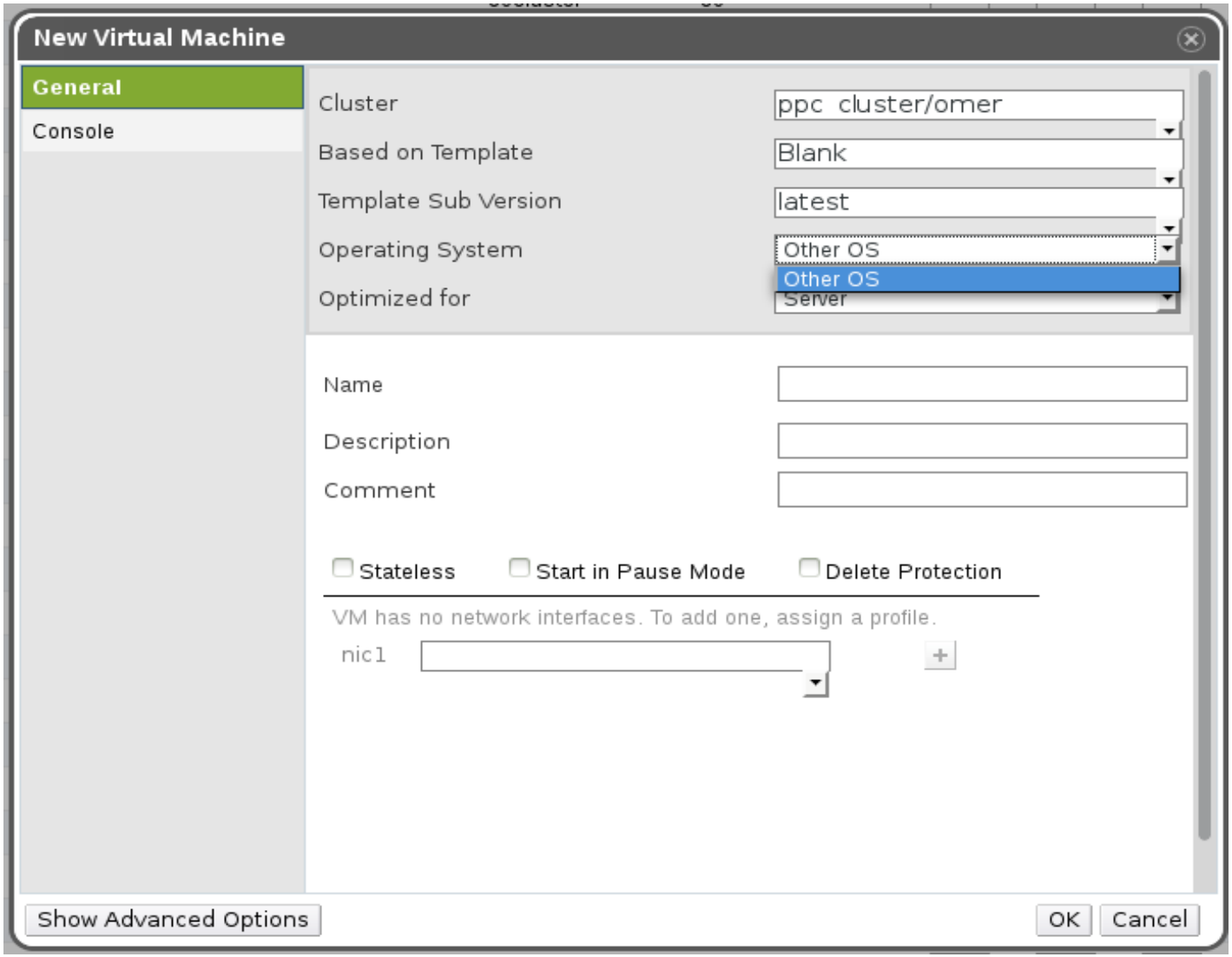
# What's done

- Power CPU type for Cluster



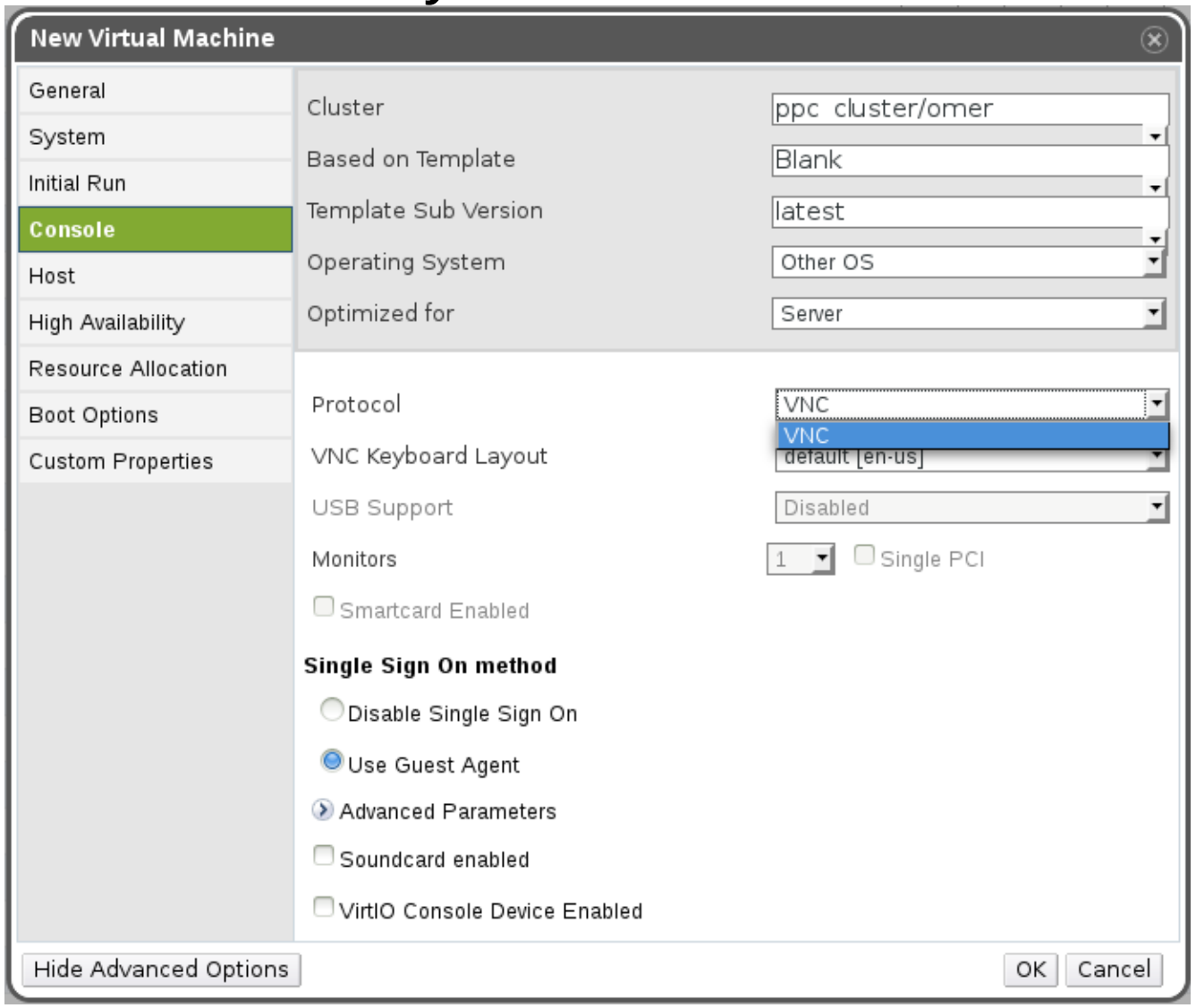
# What's done

- Filtered OS list by architecture



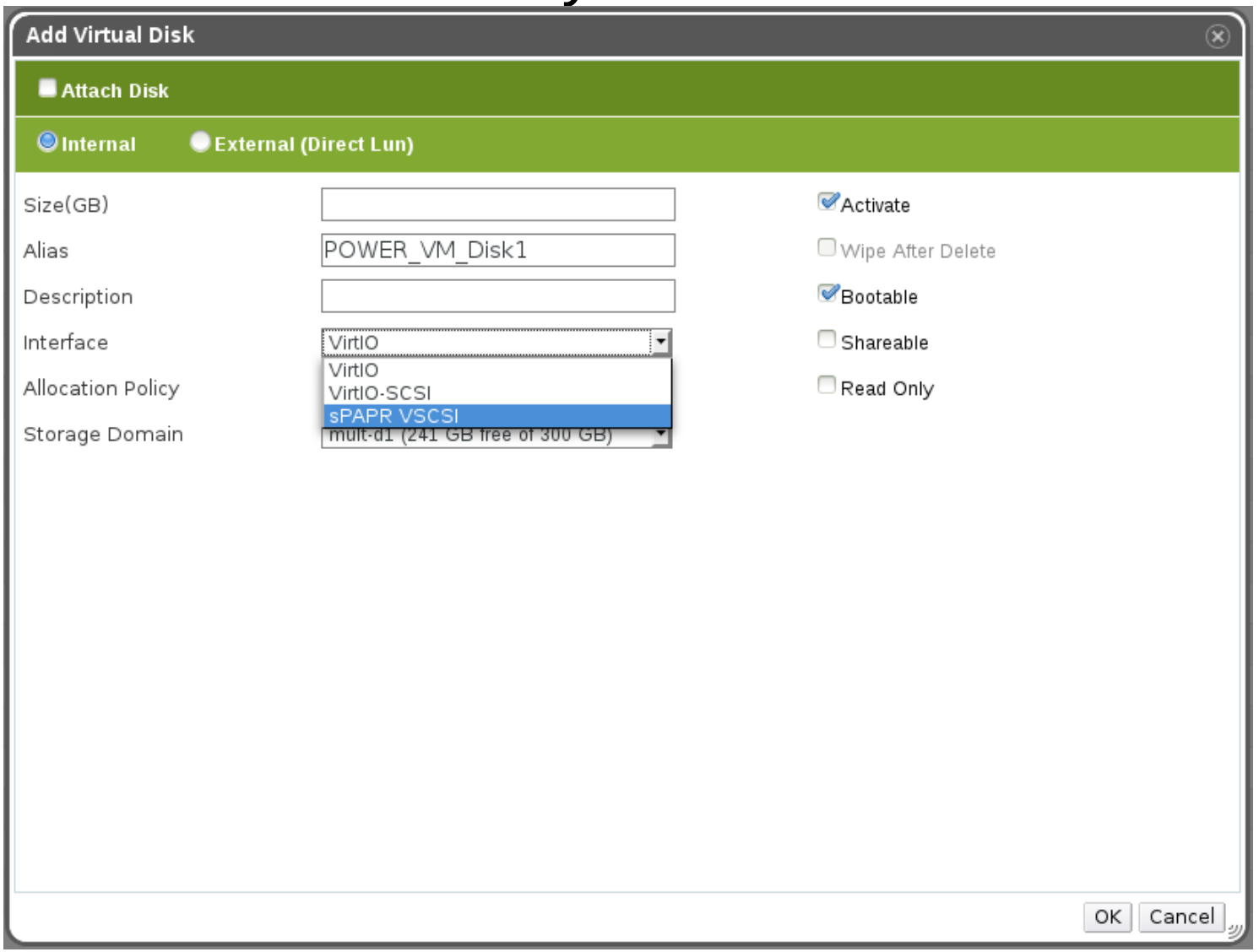
# What's done

- Filtered console list by architecture



# What's done

- Filtered disk interfaces by architecture



# What's left

- Missing features
  - Network booting
- Blocked features based on architecture
  - Migration
  - Snapshotting
  - Hotplugging

# Summing it up

oVirt

- oVirt engine is now multiplatform ready,

  currently supporting x86_64 and PPC64.
- With infrastructure ready to add other architectures easily.

- More info:
  - Website
    - http://www.ovirt.org/Community
  - Wiki
    - http://www.ovirt.org/Features/Engine_support_for_PPC64
    - http://www.ovirt.org/Features/Vdsm_for_PPC64

Questions?

# THANK YOU !

ofrenkel@redhat.com
engine-devel@ovirt.org

ofrenkel on #ovirt irc.oftc.net