KVM Evaluation

CC TF meetings, WG1,
The Northeast Asia OSS Promotion Forum

Tomoe Sugihara
VA Linux Systems Japan

Agenda

1. Background/Objectives
2. Overview of the Evaluation
3. Testing Environment
4. Results
5. Conclusion

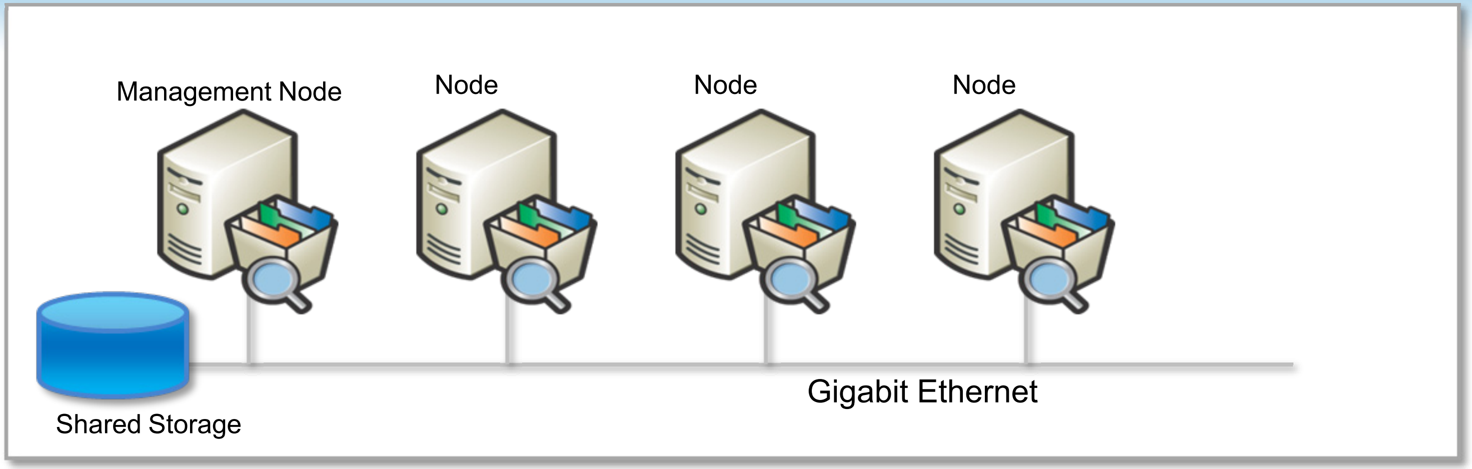
1. Background/Objectives

- ***Background***
  - No comprehensive reports available in public
  - few examples of real system implementations
    - ➡ What is KVM anyway? Is it really usable in an enterprise system?
- ***Objectives***
  - Provide useful information for system architects, integrators, and administrators (not for hackers like …)
    - ➡ Focus on system level testing
    - Clarify cause and workaround of the issues

2. Overview of the Evaluation

- ***I. Basic functionality***
  - VM definition, creation, deletion, migration, …
  - Network/Disk I/O performance
  - Time management in guest OS
- ***II. Fault tolerance***
  - Impact of Host OS and VM failure
  - Impact on filesystem on the guest caused by failures
- ***III. Impact of VM load***
  - CPU, Network, memory, and Disk I/O
  - Load balancing
- ***IV. Scenarios***
  - Migrating VMs
  - Switching back and forth to different systems

3. Testing Environment



- ***Cloud-like environment***
  - Centralized management by management node (libvirt)
  - VM images on Shared storage
  - Heterogeneous CPU archs --- AMD/Intel
- ***Software***
  - CentOS 5.4(x86_64)

Component	Package
Kernel	kernel-2.6.18-164.el5
KVM kernel module	kmod-kvm-83-105.el5_4.9
qemu	kvm-83-105.el5_4.9
libvirt	libvirt-0.6.3-20.1.el5_4

4. Results

Disclaimer:

Performance figures used in the slides are measured on the testbed, which may vary depending on systems to systems

I. Basic functionality

- ***VM management***
  - VM definition, image creation, installation, migration, …
  - Managing resources: definition, dynamic allocations

➡ **Wrote scripts using libvirt-python for VM manipulations**

Enough functions for practical use and no unavoidable critical issue

- ***Network performance***
  - Measured by the iperf benchmarking tool

➡ **Virtio works best (70% of the physical performance)**

- ***Disk I/O performance***
  - Measured I/O performance from a guest OS

➡ **raw/virtio performs best, yet the performance is low (only 40% to physical)**

NFS from guest OS works much better

- ***Time management in Linux guest***
  - Time drift happened regardless of cpu load (by default in the testbed)

➡ **Safer to use NTP both on the host and the guest**

II. Fault tolerance

- ***Impact by Host OS failure and VM failure***
  - Experiments with intentional fault injections

➡ **VM status information (libvirt) cannot always tell failures**

Need monitoring or health checking on VM

- ***Impact on filesystem***
  - Fault injections while writing to ext3 filesystem on the guest

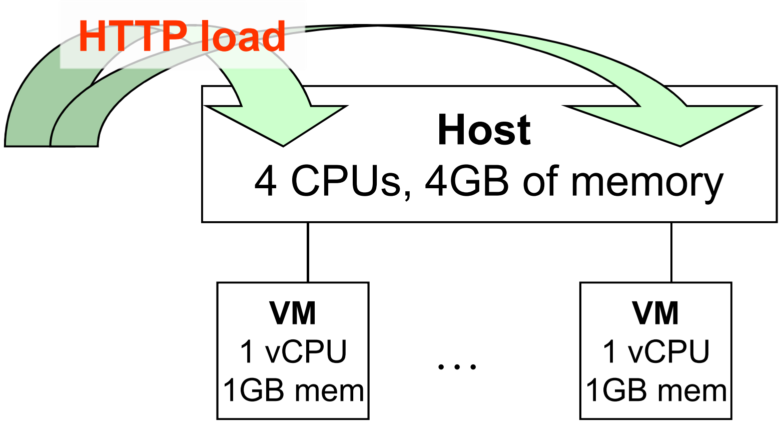
➡ **No corruption or no disadvantages to physical machines**

- ***Example system***
  - Experiment on a web server system with HA software (heartbeat)
    - Examined failure detection, failover'ablity

➡ **Worked just as fine as a system with physical servers**

III. Impact of VM load

- ***CPU, Network, Memory, Disk I/O***
  - **Equally distributed over VMs, except for network-receive**
  - **Can be worked around by packet scheduler on the host OS**
- ***On an example scale-out type web server system***



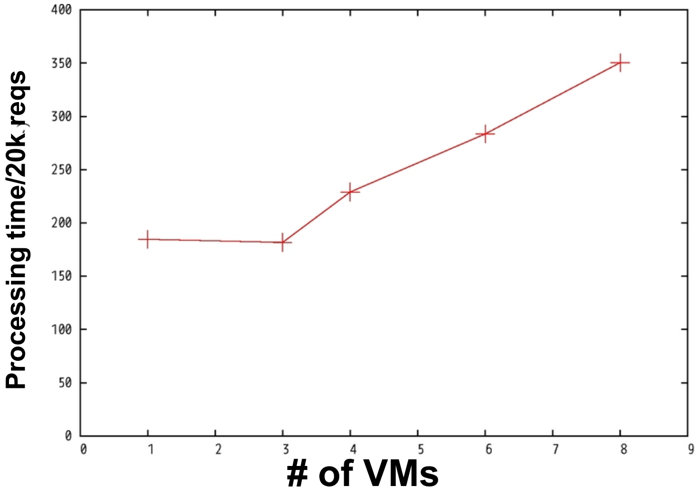
HTTP load

Host
4 CPUs, 4GB of memory

VM
1 vCPU
1GB mem

…

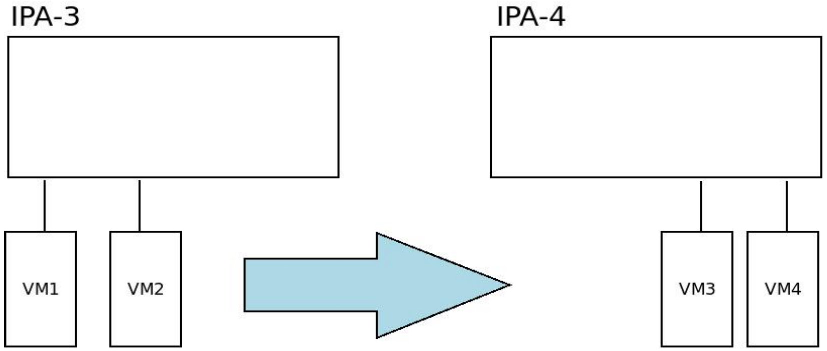
VM
1 vCPU
1GB mem



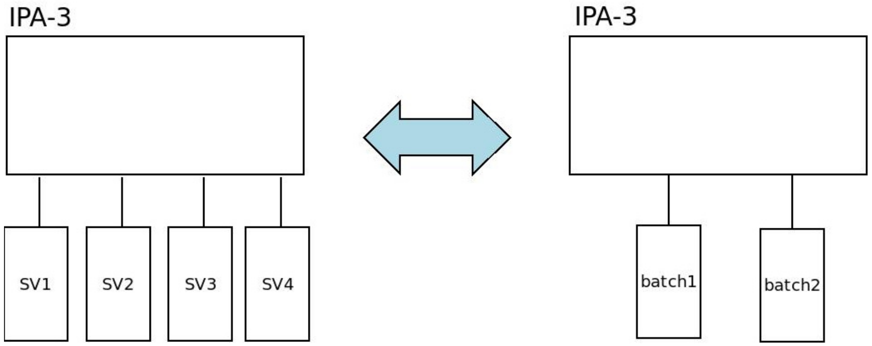
- **Need to take CPU consumption for I/Os into account**
  - **A single VM with 1 vCPU was consuming up to 130%**
- **Need to monitor performance to detect possible performance degradation**

IV. Scenarios

- ***Scheduled migration***
  - Moves VMs to another host with VM live migration



- ***Switching systems***
  - Runs different systems in different times (day/night) with VM stop/start (instead of suspend/resume)



➡ ***Perfectly achievable with our libvirt-python scripts***

5. Conclusion

- ***Summary***
  - **Comprehensive evaluation and objective analysis**
  - **Usable scripts and re-producible setup procedures of the testbed**
  - **Documents:**
    - http://ossipedia.ipa.go.jp/doc/207
- ***Conclusion***
  - **Ready for enterprise use PROVIDED with cares for the pitfalls we found**
  - **Performance improvements, more sophisticated management tools are desirable**

謝謝,감사합니다, ありがとうございました