

Virtualizing Postgres

BRUCE MOMJIAN



Draft
February, 2012

Postgres is an ideal database to use in virtualized environment. This presentation explains many of the details of such deployments.

Creative Commons Attribution License

<http://momjian.us/presentations>

Outline

1. Virtualization Primer
2. Postgres On Open-Source Hypervisors
3. Considering Postgres On Proprietary Hypervisors
4. Public Clouds
5. Postgres Customized for Clouds

Virtualization Primer: Why Virtualize?

- ▶ Multiple operating systems
- ▶ Operating system isolation
 - ▶ security
 - ▶ testing
 - ▶ upgrades
- ▶ Plug-In deployment
- ▶ Application migration
 - ▶ hardware utilization
 - ▶ reliability
- ▶ Public hardware, cloud usage

Virtualization Levels

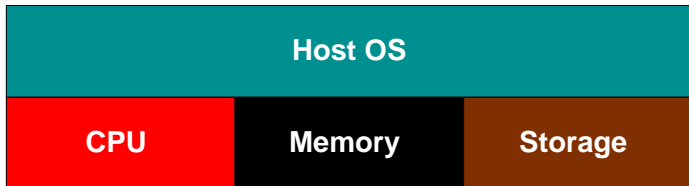
- ▶ CPU instruction emulation (QEMU)
- ▶ Hardware-Assisted virtualization
- ▶ Paravirtualization (modified operating system (compile- or run-time modified))
- ▶ Process/application virtualization (Java VM)
- ▶ API translation (Wine)

<http://www.scribd.com/doc/23757396/Virtualisation-in-Debian-Present-and-Future-Jan-Lubbe-%C2%A8-Overview>

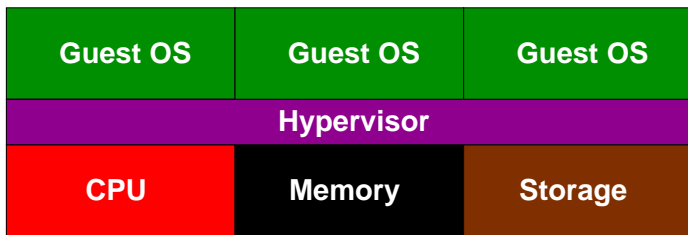
Virtual Machine Manager / Hypervisor

- ▶ Stand-Alone Hypervisor (Robert P. Goldberg type 1)
 - ▶ VMware vSphere (ESXi hypervisor)
- ▶ Host operating system with built-in hypervisor (type 2)
 - ▶ KVM
 - ▶ VirtualBox
 - ▶ Parallels
 - ▶ VMware Fusion (OS X can't legally be run on a hypervisor)
 - ▶ VMware Workstation (developer usage)
 - ▶ VMware Server (discontinued)
- ▶ Hybrid Hypervisor (1.5?)
 - ▶ Xen
 - ▶ Microsoft Hyper-V

Traditional Bare Metal Server



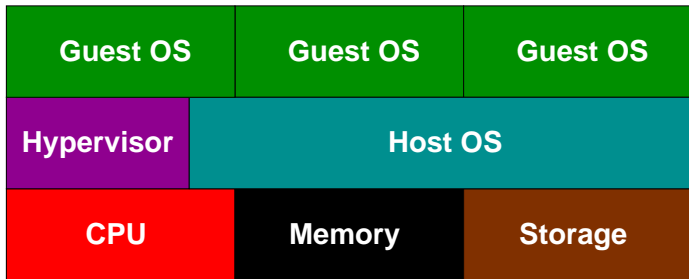
Type 1 Hypervisor



VMware vSphere (ESXi) is a type 1 hypervisor. It requires a 32MB install on bare metal, and creates a dedicated VMFS file system to store the guest OS images. The VMware hypervisor is controlled via a MS Windows machine network connection.

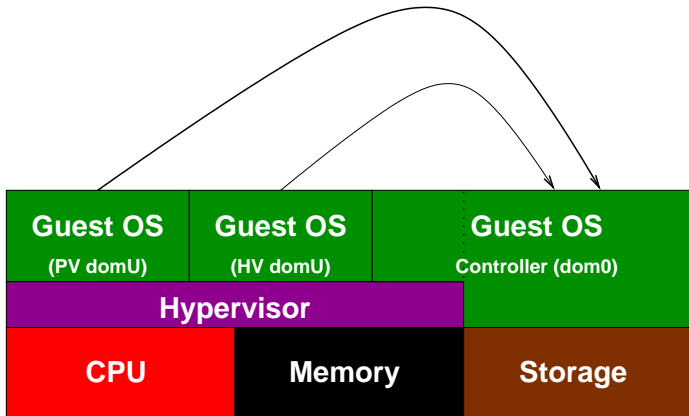
http://en.wikipedia.org/wiki/VMware_VMFS

Type 2 Hypervisor



The hypervisor and guest operating systems are normally controlled by the host operating system.

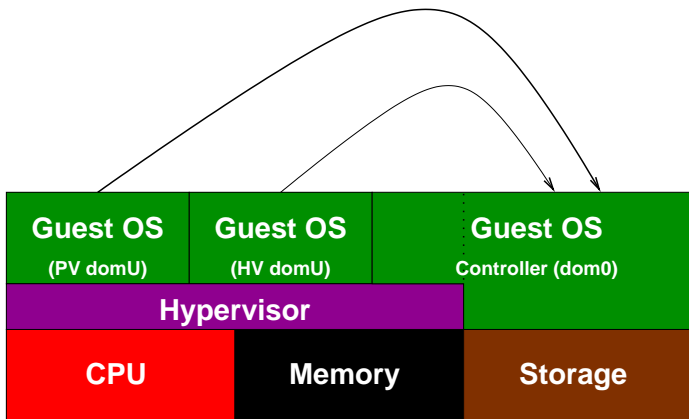
Hybrid Hypervisor (1.5?)



This diagram matches the way Xen handles virtualization. The hypervisor handles CPU and memory for the guests.

http://wiki.xen.org/wiki/Xen_Beginners_Guide

Hybrid Hypervisor (1.5?)



The dom0 guest controls the hypervisor and devices, including storage. Use of paravirtualized (PV) operating systems and device drivers allows data transfer between the domU guests and dom0 to be done via memory buffers, rather than interface daemons.

Guest Virtualization Methods

- ▶ Fully Virtualized
- ▶ Hardware-Assisted Virtual Machine (HVM)
- ▶ Paravirtualized Machine (PVM) (modified to run on a specific hypervisor)

Hardware-Assisted Virtualization

On x86, CPU capabilities Intel VT and AMD-V allow for low-overhead virtualization with:

- ▶ **Page table virtualization** - allows guest virtual memory page tables to map to the hypervisor page tables, effectively allowing two layers of virtual memory addressing
- ▶ **Device virtualization** - allows the guest operating system to control direct-memory access (DMA) and the interrupts of assign devices without hypervisor involvement
- ▶ **Guest state save/restore**

Testing for x86 Hardware-Assisted Virtualization

Linux test (don't test from inside a virtual machine):

```
# egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

Details:



<http://software.intel.com/en-us/articles/best-practices-for-paravirtualization-enhancements-from-intel-virtu>
(<http://tinyurl.com/7vrxj35>)



http://en.wikipedia.org/wiki/X86_virtualization



http://www.webopedia.com/DidYouKnow/Computer_Science/2007/hardware_assisted_virtualization.asp



<http://www.intel.com/technology/itj/2006/v10i3/2-io/3-vmm-software-architecture.htm>



http://www.hotchips.org/archives/hc17/1_Sun/HC17.T1P2.pdf

Postgres Virtualization Requirements

- ▶ Storage
- ▶ Memory
- ▶ CPU
- ▶ Network

Postgres Virtualization Requirements

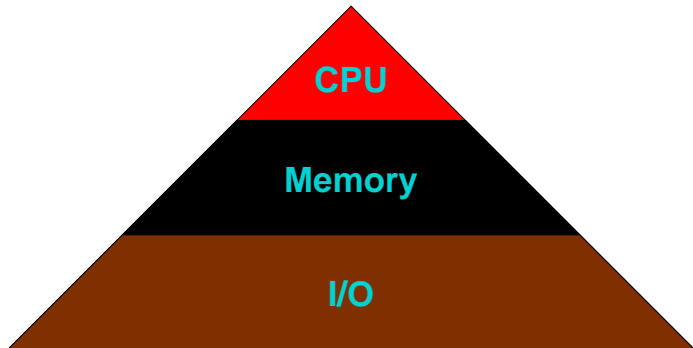
- ▶ Storage
 - ▶ fsync
 - ▶ performance
- ▶ Memory
 - ▶ shared memory
 - ▶ semaphores
- ▶ CPU
 - ▶ test-and-set instructions
- ▶ Network

http://momjian.us/main/presentations/overview.html#hw_selection

Postgres Unnecessary Virtualization

- ▶ ~~Raw devices~~
- ▶ ~~USB~~
- ▶ ~~DVD~~
- ▶ ~~Video~~
- ▶ ~~Audio~~
- ▶ ~~Clipboard~~

Database Hardware Requirements

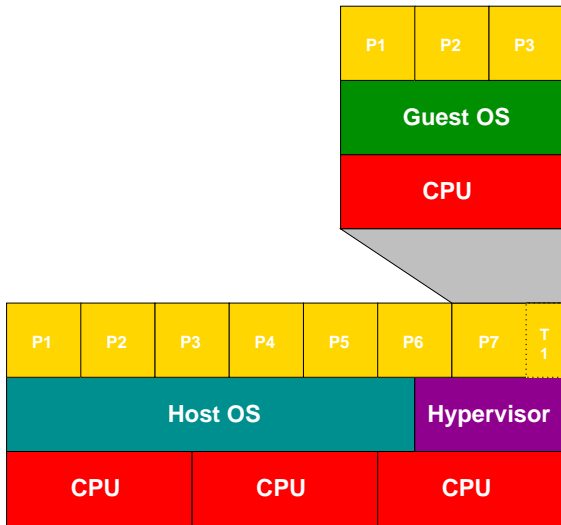


KVM CPU Virtualization

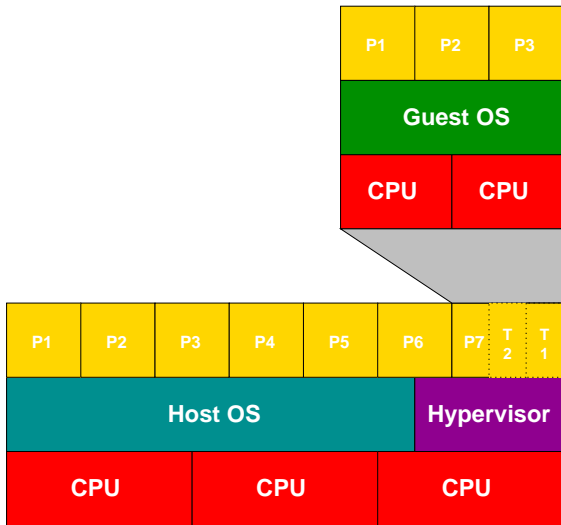
- ▶ Guest Operating System runs as a host operating system process
- ▶ A thread is created for each CPU assigned to the virtual machine

In non-type2 hypervisors (e.g. Xen and vSphere), the hypervisor controls CPU assignment.

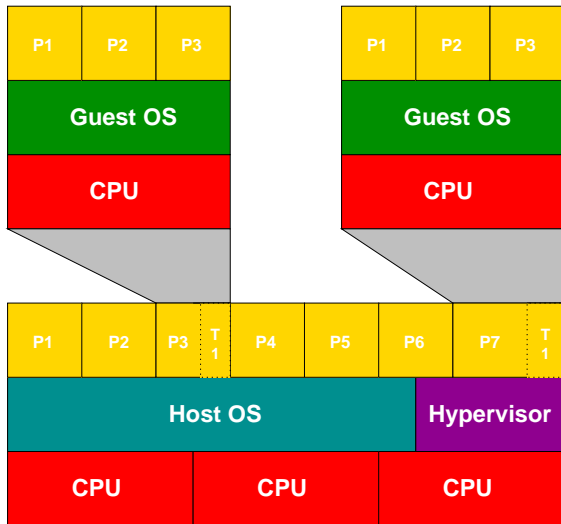
One VCPU With Type 2 Hypervisor



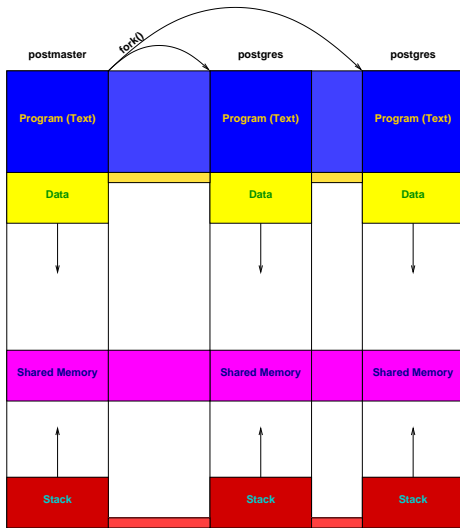
Two VCPUs With Type 2 Hypervisor



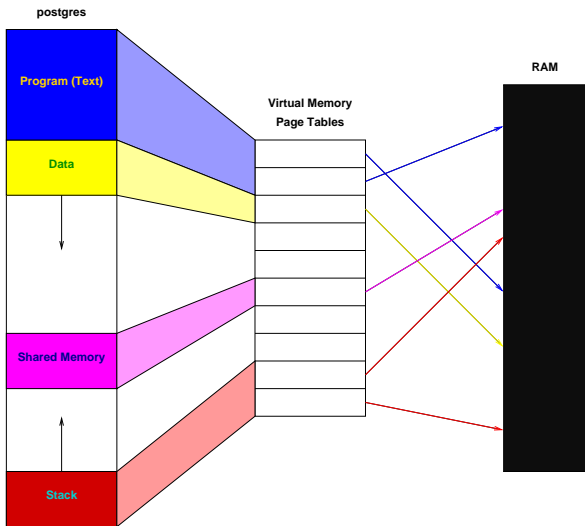
Two Guests With VCPUS



Memory: Postgres Process Address Space

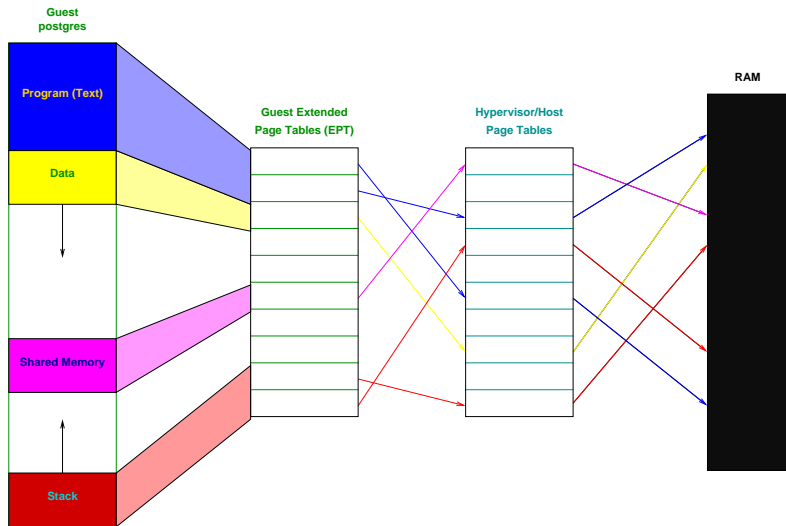


Bare Metal Virtual Page Tables



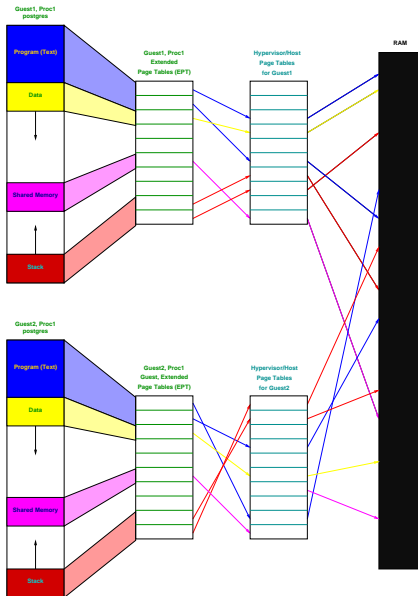
For simplicity, page directories are not shown.

Guest Virtual Memory

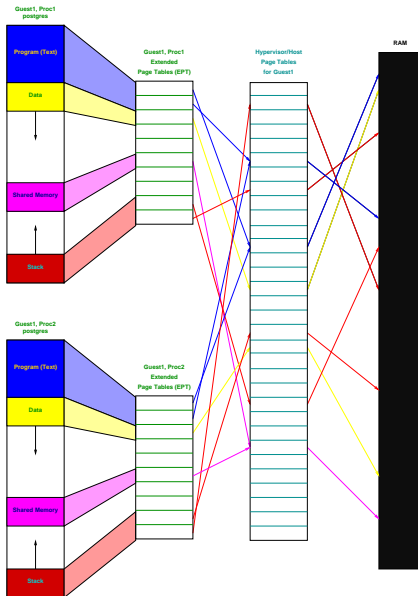


In non-type2 hypervisors (e.g. Xen and vSphere), the hypervisor controls memory allocation.

Two Guests With Virtual Memory



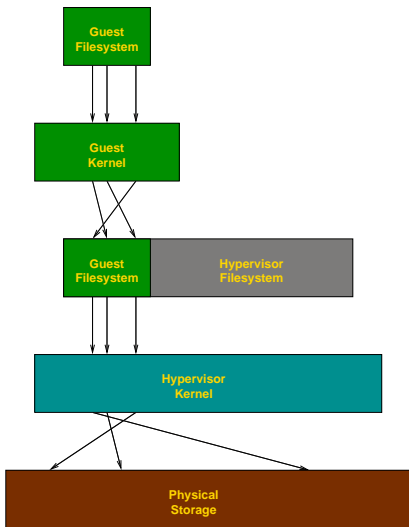
Two Processes in One Guest



Guest Storage Options

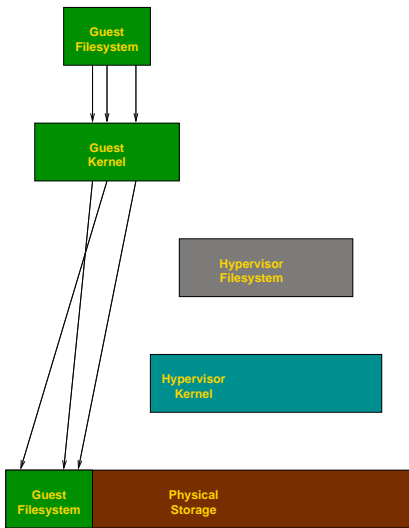
- ▶ Host operating system storage
- ▶ Direct storage
 - ▶ Logical volume manager (LVM)
 - ▶ Physical device partition

Guest Virtual Storage

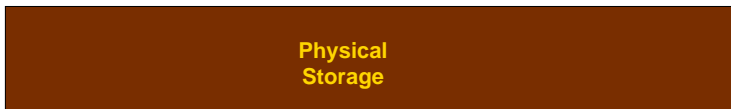
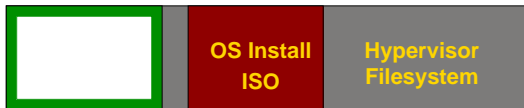


http://en.wikipedia.org/wiki/Inode_pointer_structure

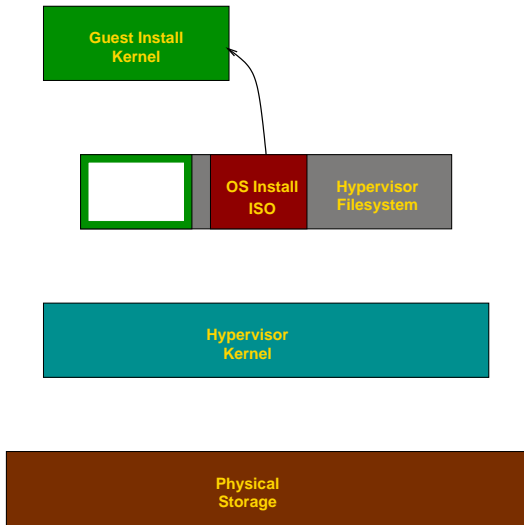
Guest Direct Storage



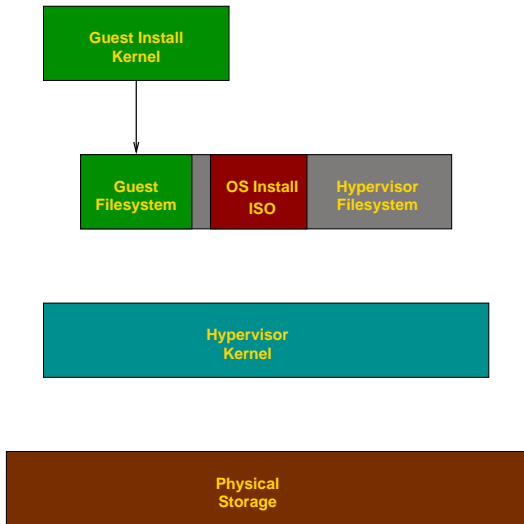
Installing a Guest Operating System



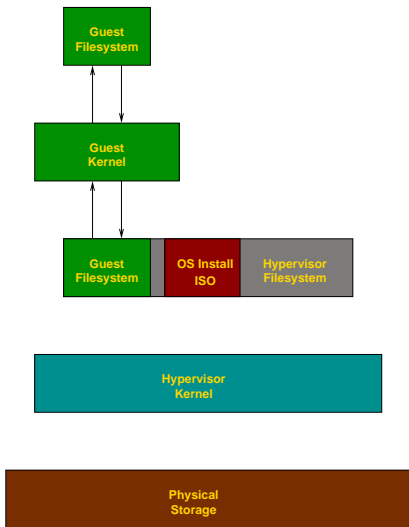
Guest Install Image Loaded from Hypervisor Storage



Guest Operating System Written by Installer



Booting the Guest Operating System



Type 1 and 1.5 Hypervisors

Vmware and Xen liked to advertise how simple and lightweight a 'bare metal hypervisor' is compared to running a full OS. But in reality, if you want to be able to do everything on a VM hosted environment that you can in a full OS then that makes the hypervisor extremely complicated. As ESX and Xen pile more and more features and gain greater performance and capabilities they are no longer very simple... in fact they essentially are becoming their own entire OS kernels. Not only that they are still dependent on the Linux kernel in order to provide all the facilities they need for hardware emulation, I/O control and management.

drag on Ars Technica

<http://arstechnica.com/civis/viewtopic.php?f=16&t=1138263>

Type 2 Hypervisors

So instead of trying to recreate all the new facilities needed to be competitive with Vmware or Xen... KVM just starts of with a full fledged kernel. What facilities you need to run applications is not very different from what you need to run VMs. This is why KVM is so full of awesome. It did basically everything that Xen/ESX does, but instead of being years in development it only took a few months.

drag on Ars Technica

<http://arstechnica.com/civis/viewtopic.php?f=16&t=1138263>

Part 2: Postgres On Open-Source Hypervisors

Hypervisors reviewed:

- ▶ Xen
- ▶ KVM

Virtualbox was not considered because its strength is desktop virtualization.

Review criteria

- ▶ Administration
- ▶ Features
- ▶ Performance

Test Configuration

- ▶ 8-core Server:
http://momjian.us/main/blogs/pgblog/2012.html#January_20_2012
- ▶ Debian 6.0.4 (Squeeze)
- ▶ Postgres 9.2 source code snapshot of 2012-02-18

Latency vs. Throughput

- ▶ Low latency is the telegraph; throughput limited by finger speed
- ▶ High throughput is a station wagon filled with tapes or disk drives; latency is limited by driving speed

The theoretical capacity of a Boeing 747 filled with Blu-Ray discs is 595,520,000 Gigabytes, resulting in a 245,829 Gbit/s flight from New York to Los Angeles.

<http://en.wikipedia.org/wiki/Sneakernet>

Configuration for Slow (Shared) Storage

- ▶ Is the limitation high latency or low throughput?
- ▶ Latency?
 - ▶ Increase memory to avoid waiting for storage reads
 - ▶ Turn off synchronous commit to avoid waiting for fsyncs
- ▶ Throughput
 - ▶ Turn off `full_page_writes` to reduce WAL traffic

The last two suggestions increase the risk of data loss.

KVM vs. Xen

- ▶ <http://virtually-a-machine.blogspot.com/2009/08/xen-vs-kvm-and-rest-of-world.html>
- ▶ http://www.phoronix.com/scan.php?page=article&item=ubuntu_1110_xenkvm&num=1
- ▶ <http://blog.codemonkey.ws/2008/05/truth-about-kvm-and-xen.html>

KVM Introduction

- ▶ http://www.linux-kvm.org/page/FAQ#Preparing_to_use_KVM
- ▶ <http://www.linuxforu.com/2009/03/kvm-virtualisation-the-linux-way/>
- ▶ <http://adminsgoodies.com/difference-between-kvm-and-qemu/>

KVM Performance

- ▶ <http://www.linux-kvm.org/page/Virtio/Block/Latency>
- ▶ http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/laat/laatbestpractices_pdf.pdf
- ▶ <http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/laav/LPC/LPCKVMSSPV2.1.pdf>

Bare Metal /proc/cpuinfo

```
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model        : 44
model name    : Intel(R) Xeon(R) CPU           E5620  @ 2.40GHz
stepping     : 2
cpu MHz      : 1600.000
cache size   : 12288 KB
physical id  : 0
siblings     : 8
core id      : 0
cpu cores    : 4
apicid       : 0
initial apicid : 0
fpu          : yes
fpu_exception : yes
cpuid level  : 11
wp           : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_
cx16 xtpr pdcml dca sse4_1 sse4_2 popcnt lahf_lm ida arat tpr_shadow vnmi flexpriori
bogomips     : 4787.90
clflush size : 64
cache_alignm : 64
address sizes : 40 bits physical, 48 bits virtual
```

KVM /proc/cpuinfo

```
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 2
model name    : QEMU Virtual CPU version 0.12.5
stepping     : 3
cpu MHz       : 2493.796
cache size   : 4096 KB
fpu          : yes
fpu_exception : yes
cpuid level  : 4
wp           : yes
flags        : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 c
bogomips     : 4987.59
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:
```

Xen /proc/cpuinfo

```
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 44
model name    : Intel(R) Xeon(R) CPU           X5650  @ 2.67GHz
stepping      : 2
cpu MHz       : 2666.760
cache size    : 12288 KB
fpu           : yes
fpu_exception : yes
cpuid level   : 11
wp            : yes
flags         : fpu tsc msr pae cx8 cmov pat clflush mmx fxsr sse sse2 ss ht sysc
bogomips     : 5333.52
clflush size  : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:
```

Part 3: Considering Postgres On Proprietary Hypervisors

Considered hypervisors:

- ▶ ESXi on vSphere (VMware)
- ▶ Hyper-V (MS Windows)

Add Postgres to VMware

- ✓ VMware → VMware & Postgres
 - ▶ BSD license
 - ▶ VMware VM-specific modifications for scaling and failover

Add VMware to Postgres

✗ Postgres → Postgres & VMware

- ▶ Windows-specific administration (web client in vSphere 5)
- ▶ Cost
- ▶ Lack of integration with Unix tools (Linux console in vSphere 5)
- ▶ Cannot publish benchmarks without prior approval

Windows Hyper-V virtualization has similar drawbacks.

<http://www.vmware.com/files/pdf/VMware-vSphere-Competitive-Reviewers-guide-WP-EN.pdf>

<http://register.vmware.com/content/eula.html>

VMware and Hyper-V

Microsoft Windows and VMware provides complete solutions with many advanced features. However, their “full solutions” limit flexibility. KVM or Xen have fewer built-in features, but more flexible. Organizations have to balance ease-of-use, features, cost, and flexibility in choosing a virtualization solution.

Part 4: Public Clouds

- ▶ GoGrid
- ▶ Amazon Web Services

Both use Xen for virtualization.

Part 5: Postgres Customized for Clouds

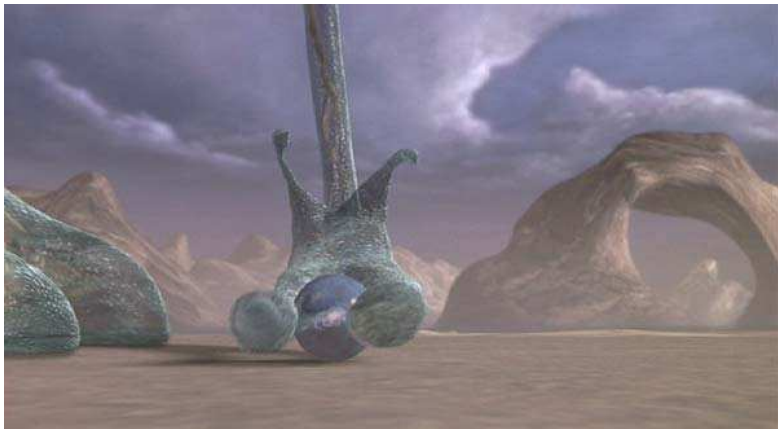
- ▶ Heroku Postgres (PaaS)
- ▶ EnterpriseDB Cloud Database

EnterpriseDB Cloud Database

- ▶ Auto-Failover
- ▶ Scaling based on storage and number of connection
- ▶ Cannot publish benchmarks without prior approval

<http://www.enterprisedb.com/cloud-database/terms-of-use>

Conclusion



<http://momjian.us/presentations>

Men In Black (1997)