Violin MEMORY

Technical White Paper

Optimizing OVM With Violin Memory Arrays

Abstract

This document contains guidelines and best practices for solutions integrating Violin Memory Arrays with Oracle OVM. It details what is possible within an OVM Virtual environment and the implications of deployment decisions on the performance.

Table of Contents

Executive Summary

With Oracle's latest release of OVM, enterprises have been able to realize both economic and environmental benefits of server virtualization. When combined with Violin Memory Flash arrays, virtualization and consolidation of applications or database can be achieved while sustaining superb levels of performance.

Violin Memory Arrays are leading the storage industry in delivering new levels of application performance and availability. When combined with virtualization technologies, both high performance and consolidations can be achieved.

This white paper highlights the possible performance and availability Violin Memory Arrays can achieve and how to best design and deploy a solution that will sustain these levels.

Introduction

OVM is being increasingly adopted as the Hypervisor of choice to virtualize Oracle Databases and other application environments for maximum performance and cost benefits. With the introduction of memory arrays as persistent storage in virtual environments, designing solutions to get the best performance has raised technical issues and questions that were previously non-existent.

These testing results and best practice notes are intended to be reviewed.

Audience

This paper is intended for systems administrators; systems architects and other technology experts involved in designing, planning or managing enterprise virtualization solutions with Oracle Virtual Server.

Methodology

Powerful server hardware with a low latency high bandwidth SAN network was deployed in conjunction with a Violin 6616 Memory Array. This was used to measure comparative behaviour of components within the hypervisor to allow key deployment choices to be made. The target was not to find the limits of the hardware but to produce comparative results between different configuration options.

This paper focuses on recommendations for an environment connected by FC, but the same recommendations can apply to ISCSI connectivity to the hypervisor.

Required Configuration Settings

To ensure the correct behavior of the storage in the case of failures in the environment, it is necessary to make adjustments to two areas of each Hypervisors configuration. First, the correct rule must be put in place for the multi-pathing subsystems to function optimally Violin Array. Second, the recommended udev rules should be put in place so that the device entries are created with the correct parameters. These changes should be made on each Oracle Virtual server accessing Violin devices.

Multipath Configuration

The following device entry should be added to the /etc/multipath.conf file on each OVS Hypervisor host.

```
device {
        vendor               "VIOLIN"
        product              "SAN ARRAY"
        path_grouping_policy  group_by_serial
        getuid_callout        "/sbin/scsi_id -p 0x80 -g -u -s /block/%n"
        path_checker          tur
        path_selector         "round-robin 0"
        hardware_handler      "0"
        features              "0"
        failback              immediate
        rr_weight             uniform
        no_path_retry         fail
        rr_min_io             4
        max_fds               8192
}
```

To ensure the new device definitions are in effect after editing the file the following commands can be used.

```
[root@ovs1 etc]# multipath –F
[root@ovs1 etc]# multipath
```

The above commands will remove and then recreate the multipath definitions for all multipath devices.

Finally check that the definitions have bee correctly applied to the violin devices by running multipath –ll as below. Check in the output all the Violin Devices the features entry should read ‘0’, also that all paths for a device appear in a single policy group as the below example.

```
[root@halfserver1 ~]# multipath -ll
mpathw (SVIOLIN_SAN_ARRAY_16968FC5BB38CBD8) dm-1 VIOLIN,SAN ARRAY
size=101G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  |- 2:0:3:1  sda  8:0    active ready running
  |- 3:0:0:1  sde  8:64   active ready running
  |- 4:0:5:1  sdj  8:144  active ready running
  |- 6:0:1:1  sdo  8:224  active ready running
```

"udev" Rules

To ensure that devices have the optimum parameters for peak performance, "udey" is used. Udev is a mechanism to allow persistent device naming and dynamic device configuration in linux kernels 2.6 and higher.

Violin recommends the creation of the following rule to ensure that all existing and future Violin devices get the optimal parameters by the hypervisor. To achieve this, create a new file within /etc/udev/rules.d called 12-violin.rules, this should be done on each of the OVM hosts accessing Violin devices.

```
[root@ovs1]# cat /etc/udev/rules.d/12-violin.rules

# Set scheduler and queue depth for Violin SCSI devices

KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", SYSFS{model}=="SAN ARRAY*", RUN+="/bin/sh -c
'echo 0 > /sys/$devpath/queue/rotational && echo noop > /sys/$devpath/queue/scheduler && echo 512 >
/sys/$devpath/queue/nr_requests'"

#

# Set rotational, scheduler and queue depth for Violin multipath devices

SUBSYSTEM=="block", KERNEL=="dm-[0-9]*", PROGRAM="/sbin/dmsetup info -c --noheadings -o name -j %M -m %m",
RESULT=="SVIOLIN*", RUN+="/bin/sh -c 'echo 0 > /sys/$devpath/queue/rotational && echo noop >
/sys/$devpath/queue/scheduler && echo 512 > /sys/$devpath/queue/nr_requests'"
```

Before triggering the udev rule you can confirm what settings are in place for your devices. For example if dm-0 is one of your Violin devices identified in the multipath configuration.

```
[root@ovs2 rules.d]# cat /sys/block/dm-0/queue/rotational
 1
[root@ovs2 rules.d]# cat /sys/block/dm-0/queue/scheduler
noop [deadline] cfq
```

To have the new udev rules take effect while the system is running.

```
[root@ovs1 rules.d]# udevcontrol reload_rules
[root@ovs1 rules.d]# udevtrigger
```

Check that the modifications have taken effect.

```
[root@ovs1 rules.d]# cat /sys/block/dm-0/queue/rotational
 0
[root@ovs1 rules.d]# cat /sys/block/dm-0/queue/scheduler
[noop] deadline cfq
```

Depending on the type and version of the OS being deployed similar benefits may be seen by creating a udev rule to set the scheduler parameters within the guest OS.

What performance can be expected?

Performance, as measured in Violin Memory labs with OVS, has been impressive. While not hitting the levels seen using Linux running directly on the same hardware, very high levels of performance have been recorded on a modest dual socket server. The modest dual socket server, contained 4 core Intel Westmere CPUs and the storage performance exceeded the results of the same configuration on much larger Numa systems.

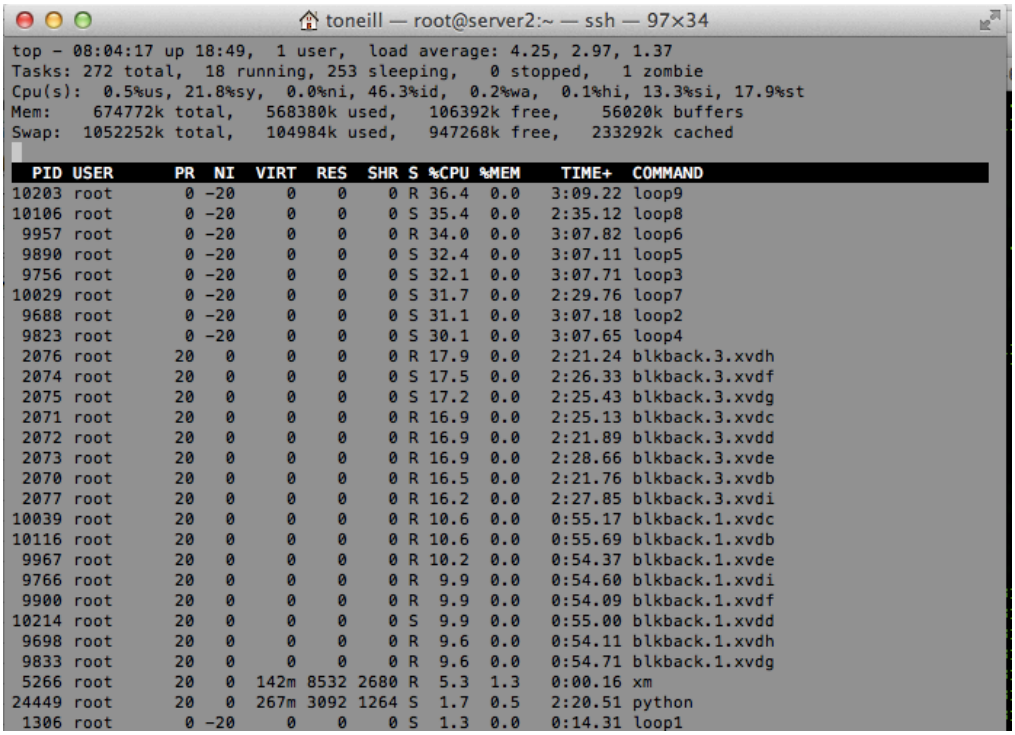
In an optimal environment with a storage-only benchmark, rates of 210,000 disk transactions per second and 3GB per second of throughput have been sustained from a single Virtual Machine.

Number of VMs per Physical Host

OVS is a very efficient server Virtualisation Hypervisor. It has been seen in testing that the maximum IO capabilities of a physical host can be driven from a single VM, or can be shared equally between a number of VMs on a host. As the number of VMs on a single host increases the load on the Dom-0 or Hypervisor partition, will increase significantly and should be monitored.

Specifically the processes associated with IO traffic on Dom-0 can consume significant proportions of available resources. If CPU on Dom-0 becomes a bottleneck then assigning more cores to Dom-0 or adding more hosts to the cluster should be considered.

Monitoring Dom-0



In this screen shot the Dom-0 partition is being kept busy by the blkback and loop processes associated with data traffic. Limits can be identified by either individual process consuming close to a 100%, saturating a single core, or the percent idle number approaching zero for the entire Dom-0.

Storage Design Considerations

Recommendations for Creating LUNs on Violin Memory Arrays for OVM

Sector Size for LUN presentation

Violin memory arrays provide the ability to present LUNs using the Advanced Format sector size of 4k or the traditional sector size of 512 bytes. 4k Devices were tested as repository devices however stability issues were seen when trying to boot VMs from these repositories. 4k sector devices were used successfully as pass through, physical disk devices.

Be aware that not all operating system support booting from devices with 4k Sectors. For example RHEL6 supports attaching the devices for data but not as the boot volume.

Best practice recommendation – Sector Size

If using OCFS2 repositories only 512 byte sector devices should be used. Care should be taken to check for correct alignment otherwise longer latencies would be seen. The "vcounts" utility run on the gateway can be used check for miss aligned IO.

If using "Pass Through" devices to VMs consider using 4k devices to eliminate the possibility of misaligned or partial IO. However care should be taken as not all applications and OS versions utilities fully support 4k advanced sector devices.

IGroup creation and management

The vMOS operating system allows the grouping of WWPNs into igroups to simplify the presentation of luns to hosts. Nested igroups are also permitted to provide an optimal structure for presenting volumes to a clustered environment. These should be used as they simplify the addition of a node to an existing cluster. As a new node is added to an igroup, existing luns will automatically be exported.

Best practice recommendation – Igroups

Create a nested igroup structure for the OVM cluster. This allows mapping to individual nodes or to the entire cluster as required.

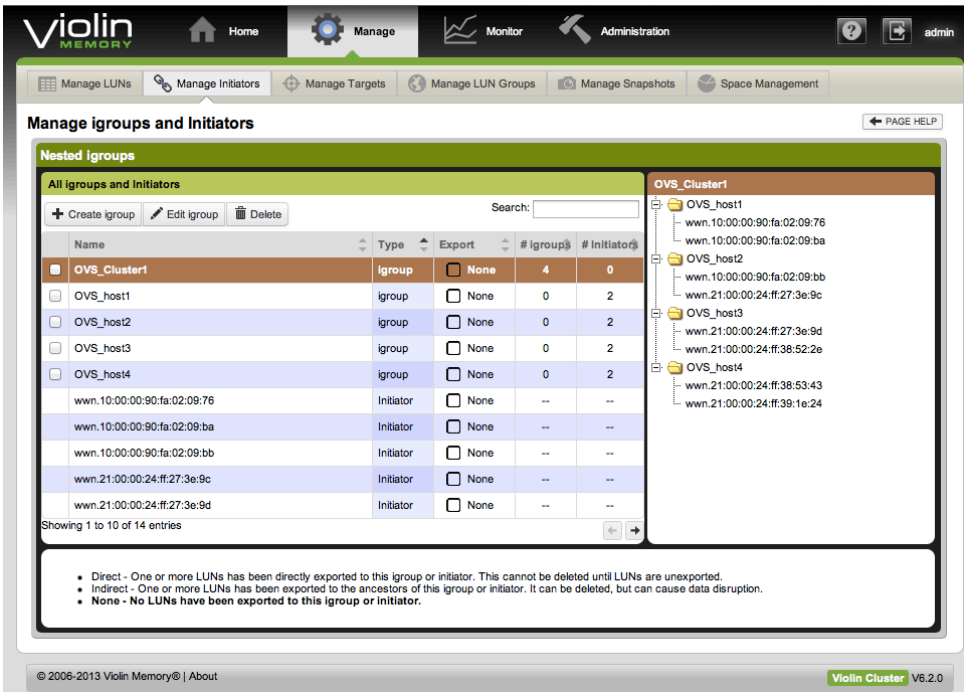


Image showing an optimal configuration of nested igroups from an OVM cluster.

Recommendations for Connecting to an OVM Environment

Number of FC ports to use

Violin arrays are able to sustain higher level of IOPS and Bandwidth than traditional storage systems. They are able to fully saturate FC ports for indefinite periods. This can lead to the FC connectivity becoming a bottleneck to the storage. As a reference point a single 8GB FC port can sustain around 150,000 IOPS or 800 MB per sec.

> ***Best practice recommendation – FC Connections***
> Ensure that a dual fabric FC structure is deployed to provide resiliency. Connect as many storage ports and server ports per host as needed to sustain the performance required.

Zoning Considerations

It is important for stability to follow best practice when configuring zones to ensure manageability and stability. Optimal “Single Initiator Single target” zoning may cause zone sets to become too large to manage efficiently in virtual clusters. So a design using “Single Initiator Multiple Targets” ports per zone may be adopted. Care should be taken with NPIV virtualization so that active and inactive WWPNs are zoned and mapped.

Best practice recommendation – Zoning and masking
Build zones, where each containing a “Single Initiator and Multiple Targets”

Recommendations to be used Configuring Storage within OVS to VMs

PVM or HVM virtual machine templates

The decision to deploy PVM or HVM virtual machines very much depends on the application and the operating system version that will be used. Generally PVM configurations will perform better, however this is not always the case. For detailed information on this discussion please see Oracle Doc ID 757719.1.

Hypervisor CPU assignment

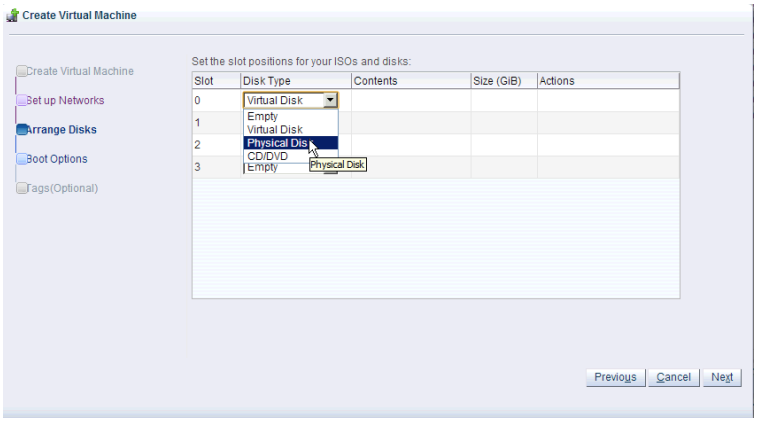
Testing has shown a significant benefit to the peak IOPS achievable may be reached by locking CPU resources assigned to the Hypervisor. Results will vary depending on the workload running on the guests and the hardware on the server. However a good starting point is to match the number of cores assigned to dom-o to the number of cores in one of the CPU sockets. Then pin the hypervisor to use just those cores.

This can be achieved by adding the **dom0_vcpus_pin dom0_max_vcpus=X** parameters in grub.conf

```
# cat /boot/grub/grub.conf
# extra comments removed for brevity
default=0
timeout=5
title Oracle VM Server-ovs (xen-4.1.3 2.6.39-300.22.2.el5uek)
        root (hd0,0)
        kernel /xen.gz dom0_mem=752M dom0_vcpus_pin dom0_max_vcpus=8
        module /vmlinuz-2.6.39-300.22.2.el5uek ro root=UUID=ab6cf316-74f9-45a1-b511-ed82ec4aff20
        module /initrd-2.6.39-300.22.2.el5uek.img
title Oracle VM Server-ovs serial console (xen-4.1.3 2.6.39-300.22.2.el5uek)
        root (hd0,0)
        kernel /xen.gz console=com1,vga com1=57600,8n1 dom0_mem=752M
        module /vmlinuz-2.6.39-300.22.2.el5uek ro root=UUID=ab6cf316-74f9-45a1-b511-ed82ec4aff20  console=hvc0
        module /initrd-2.6.39-300.22.2.el5uek.img
```

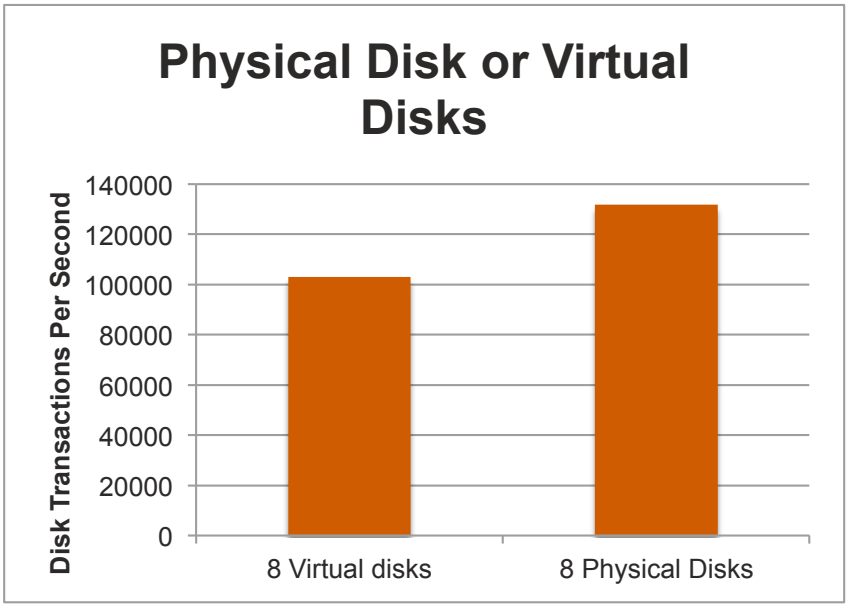
Virtual disks or Physical disk pass through

When adding disks to OVS guests a number of options on how to present the capacity are possible. One of the most important decisions is to either use virtual disks within a repository or Physical Device pass through.



There are advantages and different flexibility options provided by either mechanism.

However, if you are looking to achieve maximum storage performance from your configuration a measurable advantage can be seen using Physical Disks.



Disk transactions tested are 100% random and a mix of reads and writes to simulate a real world environment

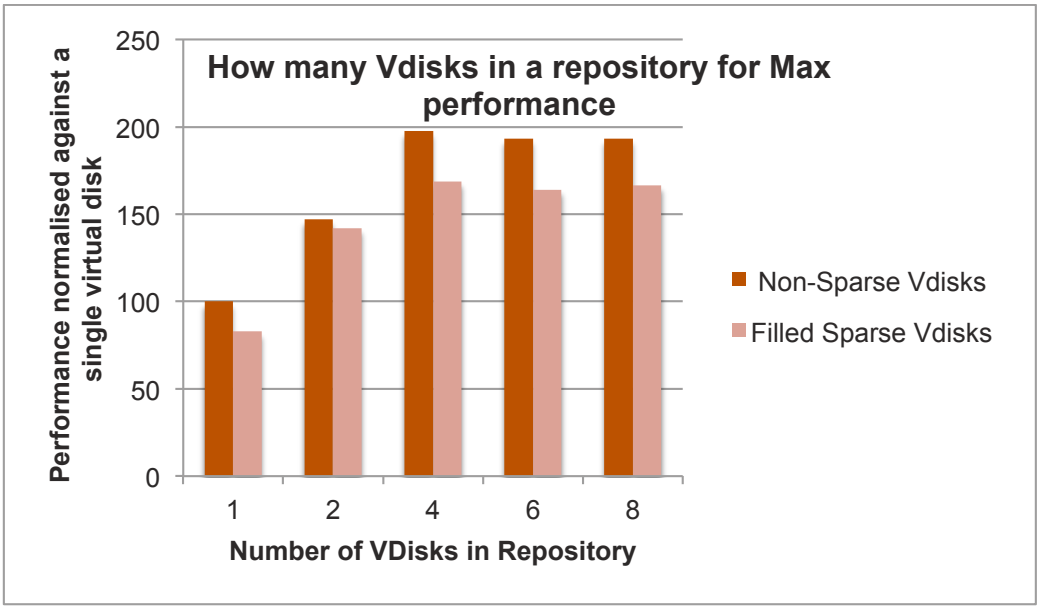
Best practice recommendation – Virtual disk or Physical Disk pass through
If you do not require the extra flexibility of Virtual disks, then use Physical disk configuration to gain the maximum performance.

Virtual Disks Configurations

If you have chosen to use virtual disk storage presentation, then careful consideration should be made to the number of virtual disks and repositories used.

The storage repositories used within OVM to host virtual disks are formatted with the OCFS2 filesystem. OCFS2 is extremely scalable and the full storage capabilities of a single host can be typically driven from a single Repository.

However performance can be more complicated to troubleshoot if multiple high workload VMs, that share a repository are running on separate physical hosts.



Disk transactions tested are 100% random and a mix of reads and writes to simulate a real world environment

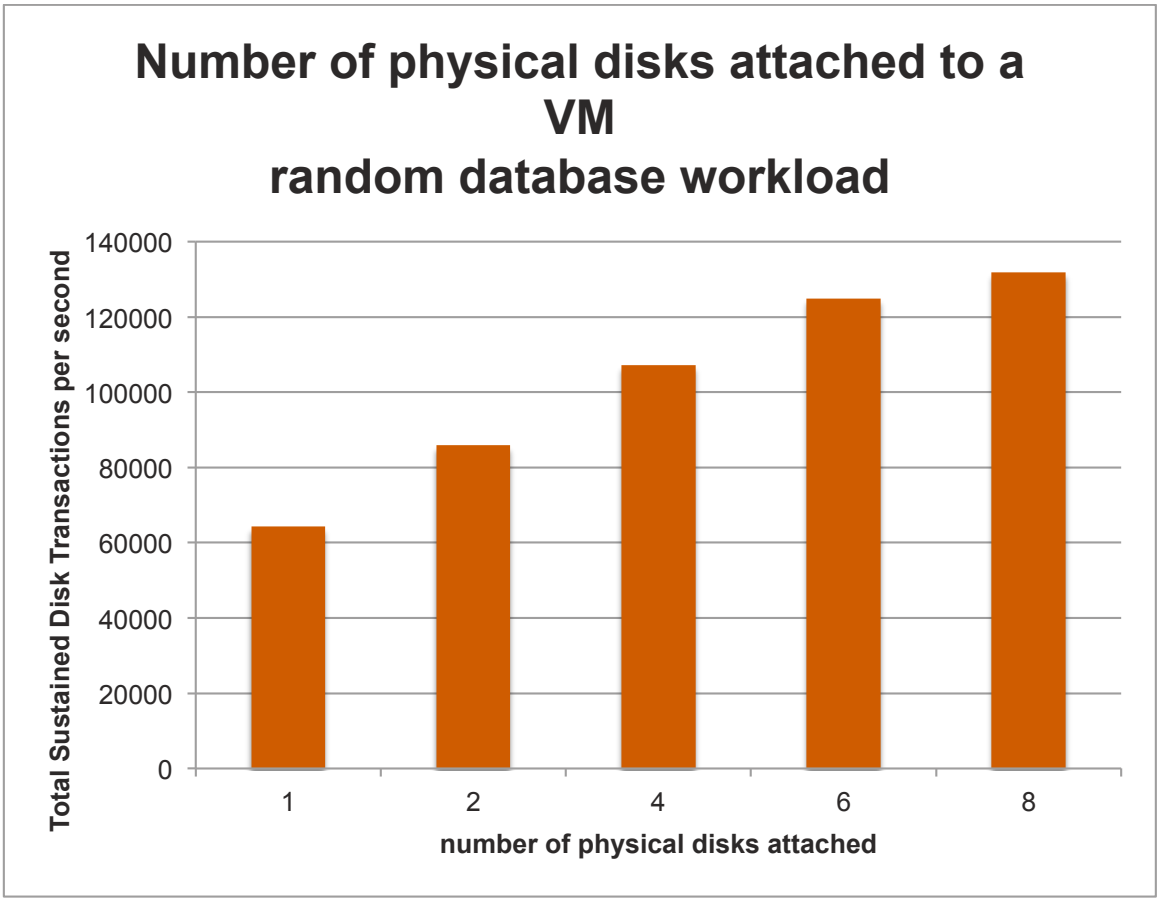
Oracle Virtual Server provides the ability to create virtual disks in either pre allocated or sparse allocated format. The advantage of the sparse format is that on space is only consumed in the repository after data is written. However as can be seen from the above chart, there is a performance overhead even after the space is filled.

Caution if using either sparse on non-sparse virtual disks increased write latency can be seen on the first write to each block of capacity. The results detailed in the above chart are created after the virtual disks have been written once.

Best practice recommendation – Storage Repositories

Place virtual disks for different VMs on different storage repositories where the VMs are expected to drive a heavy workload. Use multiple virtual disks per VM so that each VM has full access to the storage performance.

Using multiple physical disks to get maximum transactions per second



Disk transactions tested are 100% random and a mix of reads and writes to simulate a real world environment

When using physical disks presented to a VM, a significant performance improvement is available using multiple physical disks. This is because of the blkback/blkfront storage architecture within the Xen based Hypervisor. While it may not be practical in many situations, some further benefits have been by scaling to 16 or more physical disks attached to a VM.

For database workloads such as Oracle with it's ASM storage layout it is simple to make use of multiple devices and ensure that the workload is spread evenly. For other workloads striping within the guest OS using LVM could be considered, however care should be taken to ensure that the Logical volume created is aligned to 4096 byte sector boundaries.

Conclusion

Violin have performed extensive testing in lab environments and have deployed OVM with Violin for customer workloads in production environments. Oracle Virtual Server has been seen to be a capable hypervisor able to run multiple workloads effectively, providing both consolidation and high availability benefits.

When combined with the levels of performance available from Violin Memory arrays the virtualized workloads can deliver levels of storage performance that exceed those possible on legacy storage devices from physical servers.

Glossary

- **Bandwidth** – The amount of data read or written per sec to the persistent data store.
- **DomU** – Domain User, Domains or Virtual machines running within another operating system
- **Dom0** – The most privileged virtual machine or domain on a host. This is the domain that can access the hardware directly; the hypervisor can be managed and unprivileged domains launched.
- **FC** – Fibre Channel - A storage networking protocol used to present storage capacity (typically over Fibre Optic connections) to servers
- **IOPS** – A term commonly used to describe a count of **I**nput **O**utput operations **P**er **S**econd.
- **Host** – A host is a server that is running a hypervisor operating system and can host multiple guest Virtual Machines.
- **Hypervisor** – A hypervisor is a piece of computer software, firmware or hardware that creates and runs virtual machines.
- **LUN** – A name commonly used for a volume created on a storage array and presented to a physical server.
- **Latency** – The time taken for any read or write operation to complete, normally measured in milliseconds.
- **OCFS2** – A clustered file system used on a disk in the Hypervisor to create a Storage Repository.
- **Pass Through Devices** – A way of presenting devices to a VM or Guest where an entire storage volume is “Passed Through” to a guest VM.
- **PVM** –Paravirtual machine, a VM where the kernel of the guest operating system has been recompiled to be made aware of the Virtual environment.
- **HVM** –Hardware assisted Virtual Machine. A VM who runs unmodified from an operating system on Physical hardware using Virtualization aware CPU functionality such as Intel VT or AMD-V
- **Raw Disk** – A device pass through the Hypervisor directly to the guest OS without the use of a clustered file system in the hypervisor.
- **Storage Repository** – A storage pool built on the Hypervisor to contain one or more Virtual disks to present to a Virtual Machine.
- **Thin Provisioning** – A functionality of data storage system
- **VM - Virtual Machine** – A computer operating system running within another operating system
- **Zone** – A Fibre Channel networking construct, that defines which devices have visibility of each other.

About Violin Memory

Violin Memory is pioneering a new class of high-performance flash-based storage systems that are designed to bring storage performance in-line with high-speed applications, servers and networks. Violin Flash Memory Arrays are specifically designed at each level of the system architecture starting with memory and optimized through the array to leverage the inherent capabilities of flash memory and meet the sustained high-performance requirements of business critical applications, virtualized environments and Big Data solutions in enterprise data centers. Specifically designed for sustained performance with high reliability, Violin's Flash Memory Arrays can scale to hundreds of terabytes and millions of IOPS with low, predictable latency. For more information about Violin Memory products, visit www.vmem.com.