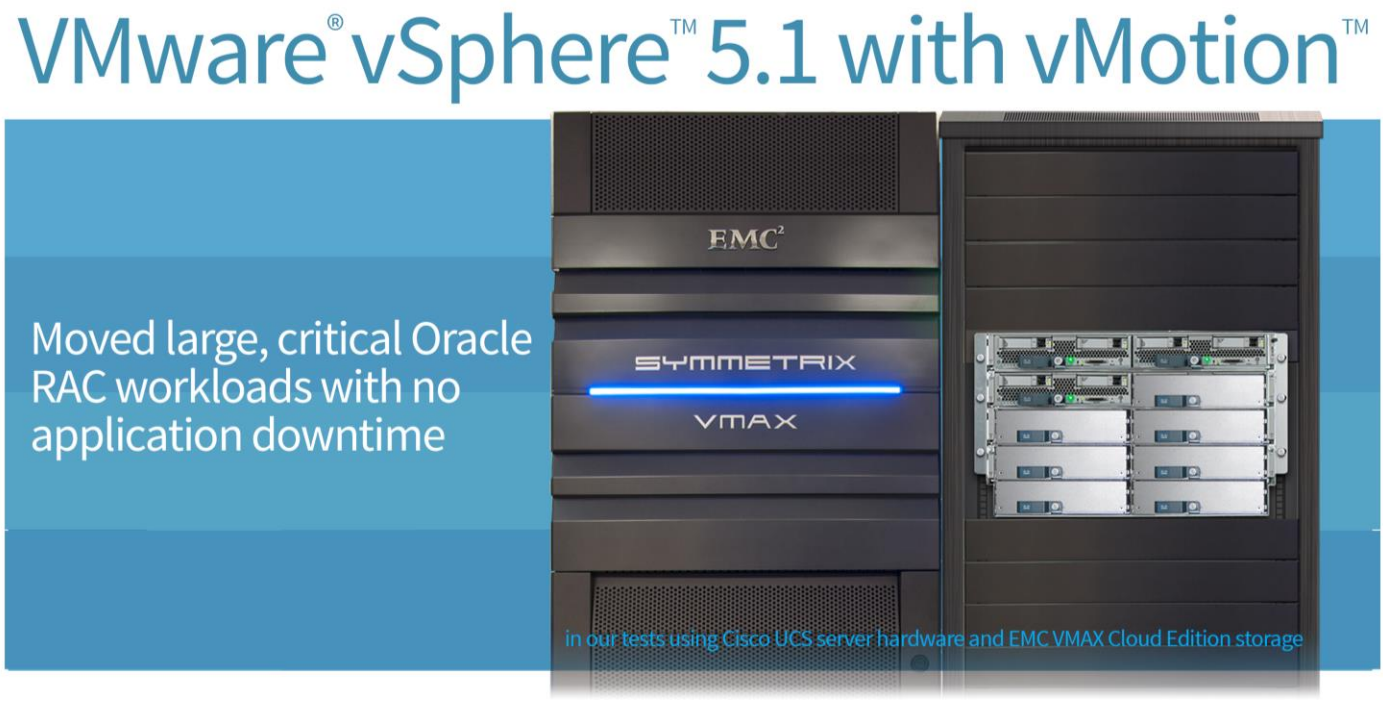
DEMONSTRATING VMOTION CAPABILITIES WITH ORACLE RAC ON VMWARE VSPHERE



Oracle Database products have long been a mainstay of the relational database market, on which many businesses and organizations run their most critical systems. When it comes to this critical application stack layer, these organizations demand the ultimate in uptime and reliability, and have traditionally had to sacrifice some flexibility and agility to achieve these goals. In the last decade, virtualization has led to a convergence where organizations need not choose between uptime and flexibility. Now, with the features and large maximum configurations available with VMware vSphere 5.1, VMware vMotion can distribute and transition workloads of large Oracle RAC configurations seamlessly between the physical nodes in a VMware vSphere cluster.

VMware vMotion can simplify previously time-consuming tasks such as moving VMs between physical servers for maintenance or distributing VMs to optimize processing power during peak workload times. Reducing the migration time required to complete these tasks is critical, along with executing the migration reliably.

In the Principled Technologies labs, we set up a VMware vSphere 5.1 cluster using three Cisco UCS B200 M3 blade servers and EMC VMAX Cloud Edition storage. Each server contained one virtual machine (VM) that was a member of an Oracle Real Application Cluster. The Oracle RAC configuration ran a large Oracle Database 11*g* Release 2 database with 156 GB of memory. Using VMware vMotion technology, we were able to easily transition VMs to other servers in the cluster without downtime.

SEPTEMBER 2013
A PRINCIPLED TECHNOLOGIES TEST REPORT
Commissioned by VMware, Inc.

VMWARE VMOTION FOR VM MIGRATION

VMware vMotion, a feature of VMware vSphere, performs live migrations that transition entire running virtual machines from one physical host to another, without forcing the workload to stop. vMotion transfers the entire execution state of the VMs, by breaking down elements to move the networking and SCSI device connections and the active physical memory of the VM. With VMware vMotion, administrators can fully automate migration and schedule migrations to occur without further intervention. If desired, vSphere can use VMware vSphere Distributed Resource Scheduler (DRS) to dynamically load balance workloads across the hosts within a VMware vSphere cluster in real time using vMotion.

VMware vSphere 5.1 includes multi-NIC vMotion capabilities to move vMotion data over the network as quickly as possible to minimize any impact to migrating workloads. VMware vMotion strives to keep workload transfers imperceptible to end users, which allows you to minimize maintenance windows and maintain service level agreements (SLAs) without hassle.

Quick and successful virtual machine migrations are a key part of running a virtualized infrastructure. When using migration for maintenance purposes, keeping maintenance windows to a minimum requires a reliable migration solution, such as VMware vMotion, to minimize any impact on end users relying on the applicable workload. Migration performance also makes a difference in maintaining SLAs with customers, which indicates the quality of service. VMware vSphere not only transitions VMs from host to host, but also utilizes migration technologies of VMware vSphere Distributed Resource Scheduler (DRS) to balance workloads across hosts dynamically and automatically.

VMware vMotion transitioned database VMs with ease

In our setup, we created a three-server VMware vSphere 5.1 cluster using Cisco UCS servers and EMC VMAX Cloud Edition storage. The configuration used a single VM per ESX host. Each VM was allocated 156 GB of memory with Oracle 11*g* Release 2 (11.2.0.3) being used as the database. For detailed hardware information, see Appendix A. For detailed steps on how tested, see Appendix B.

The initial test included a single vMotion operation, proving that vMotion is capable of transitioning a large Oracle RAC instance without issue. Figure 1 shows how we completed our single vMotion test, in which we migrated the VM from host A to host B. This test mimics a scenario where host A must be taken offline for maintenance, but the large database workload is mission-critical, and must continue to run.

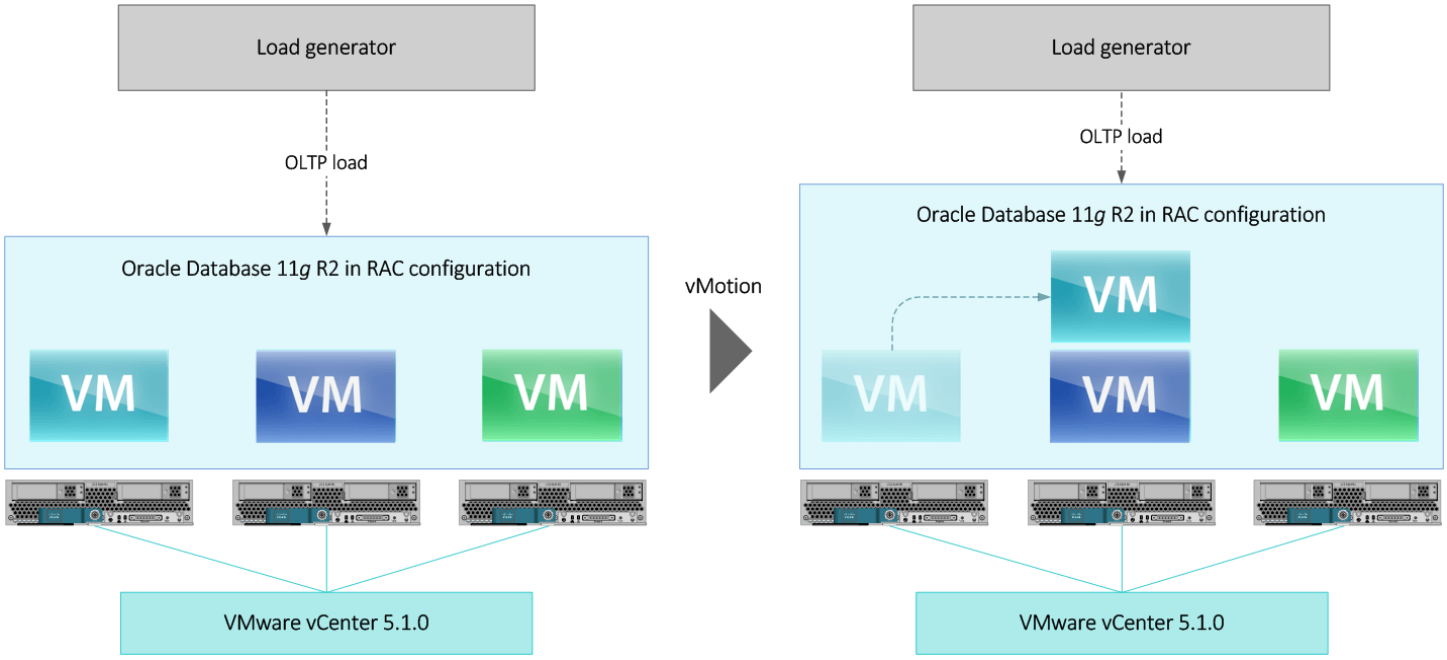


Figure 1: Diagram of our single vMotion test, where we migrated the VM from host A to host B.

To put the vMotion technology and the hardware stack to the test, we then ran two additional tests, doing a double vMotion and then a triple vMotion. While these exact scenarios are less likely to reflect everyday practice, they show the flexibility and durability that VMware vMotion provides when moving large database workloads. Figure 2 shows how we completed our double vMotion test, in which we transitioned the VM from host A to host B and the VM from host B to host A.

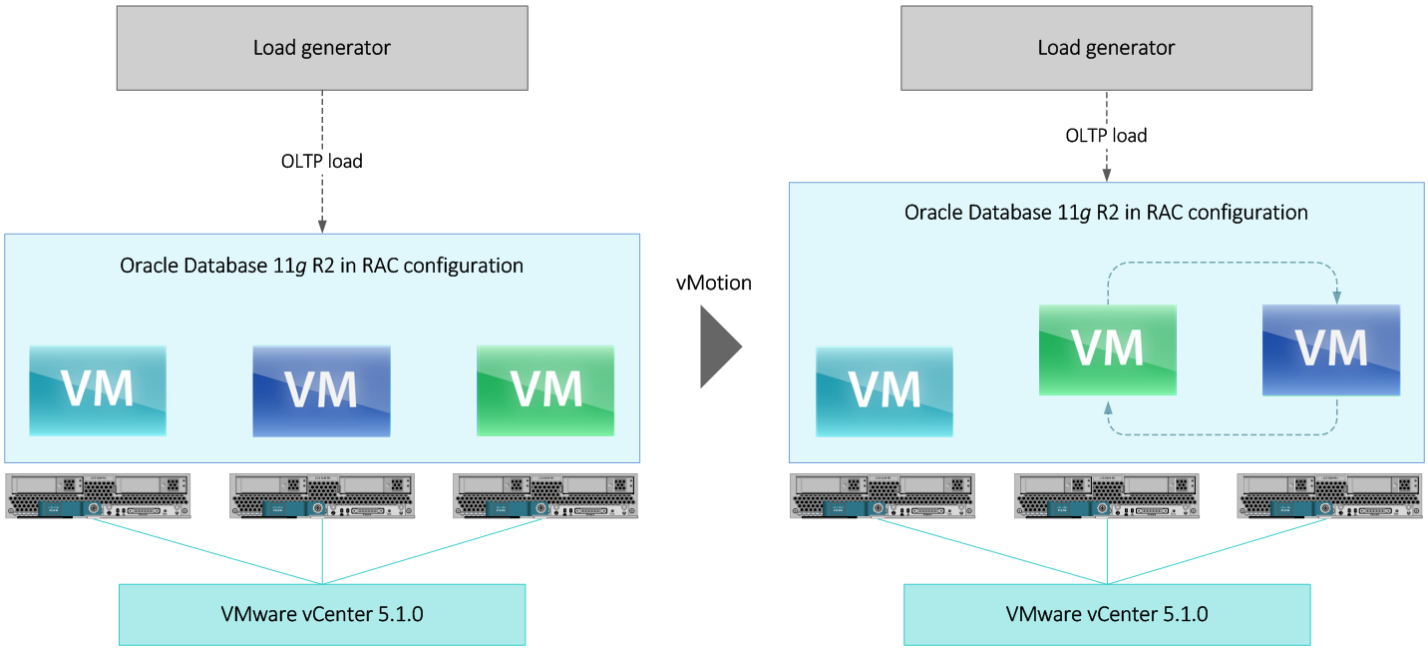


Figure 2: Diagram of our double vMotion test, where we swapped VMs on hosts A and B.

Figure 3 shows how we completed our triple vMotion test, where we migrated VMs from each host to another.

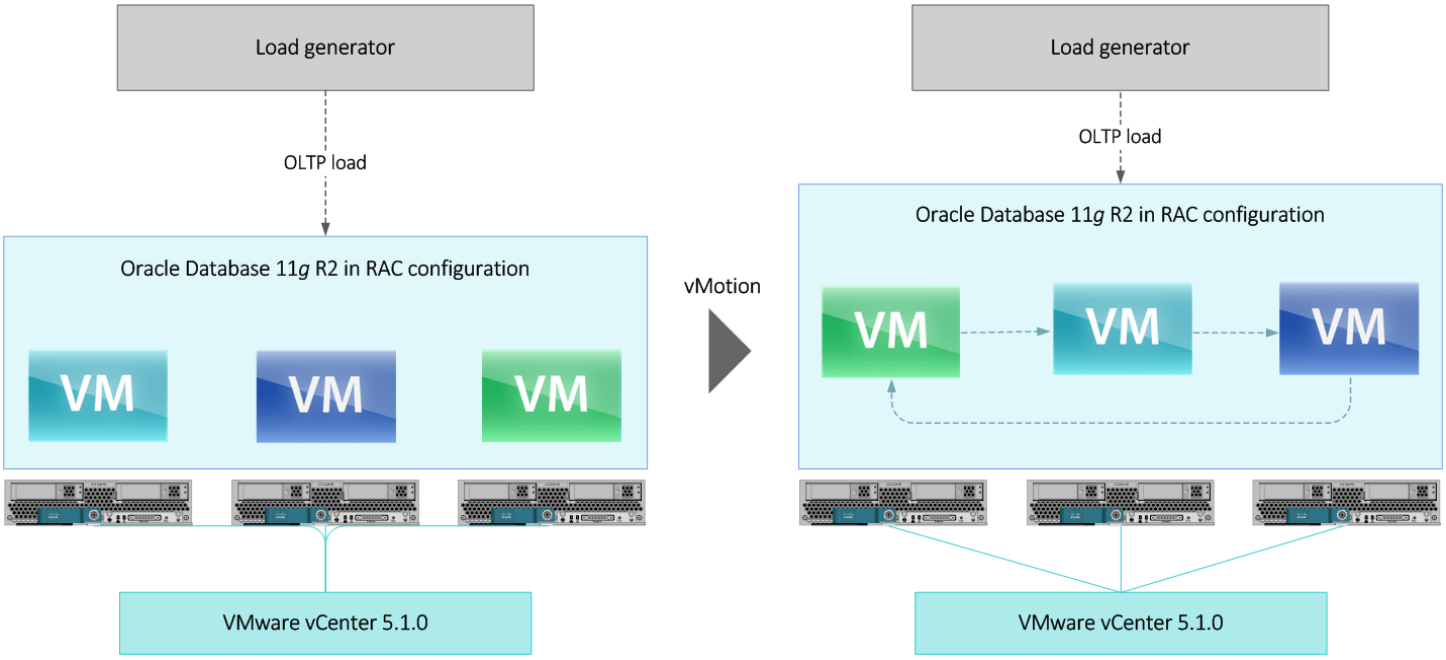


Figure 3: Diagram of our triple vMotion test, where we migrated VMs in a round-robin formation across the three hosts.

WHAT WE FOUND

Our full vMotion test consisted of a 40-minute window. First, we began an active TPC-C like workload, using Benchmark Factory for Databases, to exercise the hardware and software stack. While the TPC-C like workload ran, we performed a single vMotion, the double vMotion, and then the triple vMotion – all while the workload continued to service requests to the client application. Between the single vMotion and the double vMotion event, we reset the VMs onto their original physical hosts.

To show the impact of the moving VMs from host to host, we measured the processor utilization of each host before, during, and after each vMotion occurred (see Figure 4). Each of the before and after CPU utilization percentages are an average taken from a 90-second sample.

Core utilization percentage for each VMware vSphere host					
	State	Time (s)	Host A	Host B	Host C
Single vMotion (Host A -> Host B)	Before (90s)		32.56	31.12	29.71
	During	130	46.89	41.16	26.34
	After (90s)		0.57	60.49	31.51
Double vMotion (Host B -> Host C) (Host C -> Host B)	Before (90s)		34.28	31.55	31.85
	During	155	28.42	53.12	53.15
	After (90s)*		33.89	31.91	31.25
Triple vMotion (Host A -> Host B) (Host B -> Host C) (Host C -> Host A)	Before (90s)		33.67	33.09	31.65
	During	180	45.14	49.96	52.90
	After (90s)*		31.01	31.31	35.32

Figure 4: Processor utilization of each host during our vMotion tests. *Following these vMotion events, we allowed the workload and Oracle distribution to stabilize. The core utilization percentages here refer specifically to the 90-second interval during which the workload returned to pre-vMotion levels.

Figure 5 illustrates the core utilization of the hosts throughout our testing.

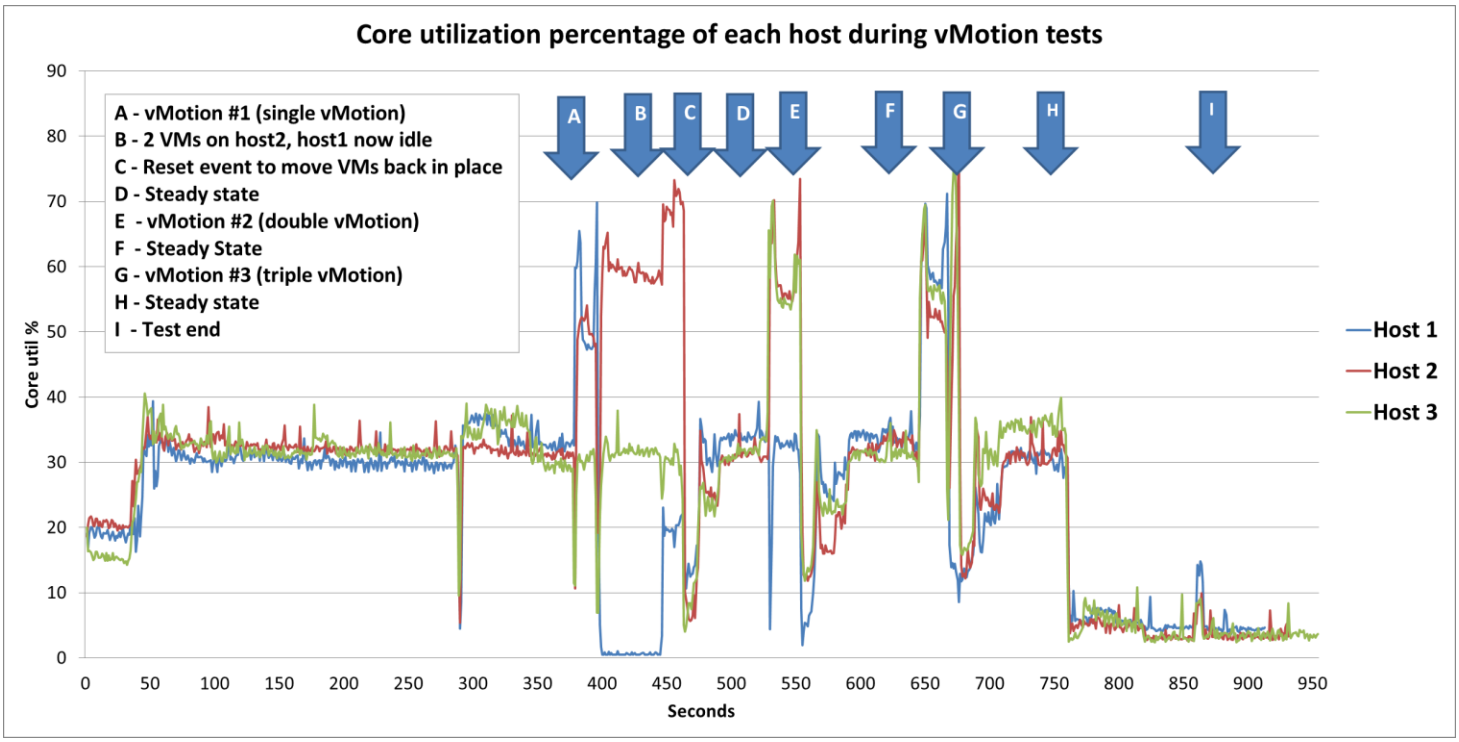


Figure 5: Core utilization percentage of each host during our vMotion tests.

We also measured NIC throughput during each of the vMotion events. Figure 6 shows the resulting networking traffic in each vMotion event. In the single vMotion, Host A mainly sent data to Host B, and Host C's vMotion network remained unaffected. In the double vMotion event, both Host B and Host C sent and received the vMotion data, because the VMs residing on those hosts swapped places. Finally, during the third and final vMotion event, all three hosts' vMotion networks sent and received data.

NIC throughput during vMotion events						
	Host A		Host B		Host C	
(Mbps)	Transmitted	Received	Transmitted	Received	Transmitted	Received
Single vMotion	10,618.19	35.59	35.49	10,668.66	0.14	0.16
Double vMotion	0.14	0.16	9,109.14	10,477.86	9,199.56	10,487.98
Triple vMotion	7,792.56	8,216.34	8,178.27	7,757.70	7,695.30	7,846.09

Figure 6: Throughput during vMotion events.

In the configuration we tested, the transactions per second from the hardware and software stack reached a maximum of 3,654 TPS. As with any live migration event, the performance was affected during each of the vMotions – the single, double, and triple vMotion. For the single vMotion, the TPS returned to pre-migration levels within 160 seconds, including the migration window. For the double vMotion, the TPS returned to pre-migration levels within 345 seconds, including the migration window. For the triple vMotion, the TPS returned to pre-migration levels within 350 seconds, including the migration window.

WHAT WE TESTED

About our database configuration

About our Oracle configuration

We created a new VM, installed Oracle Enterprise Linux on the VM, and configured the VM with 156GB RAM and 16 vCPUs. For the RAM, we reserved 145GB for the memory reservation in the VM settings. Each VM had a 60GB virtual disk, on which we created three logical volumes: root (15GB), swap (20GB), and oracle-software (/u01) (40GB) during the OS installation.

We made the necessary networking adjustments and prepared the VM for Oracle installation prerequisites (see Appendix B). Additionally, we configured other OS settings, such as stopping unneeded services, configuring NTP, setting the tuned profile to enterprise-storage, and setting the system's kernel parameters in sysctl.conf .

After cloning this VM twice, we installed Oracle Grid Infrastructure 11*g* Release 2 (11.2.0.3.0) for Linux x86-64 on the VMs on all nodes. We created additional the Oracle ASM disk groups, and then installed and configured the Oracle database. For

each database instance, we assigned 145GB for the System Global Area (SGA) and 3,686MB for the Process Global Area (PGA) aggregate target (pga_aggregrate_target).

Finally, we used the *Benchmark Factory for Databases* tool to generate the initial database schema, which we modified as we note in Appendix B. We used a benchmark scale of 5,000 in Benchmark Factory's *TPC-C* like workload generator to create approximately 320GB of data, and we configured Benchmark Factory to create the tables and indices. We specified the placement of the tables on tablespace USRS and the indices on tablespace INDX. For greater detail, including schema scripts, see Appendix B.

About Benchmark Factory for Databases

To measure database performance, we used the Benchmark Factory for Databases tool, which lets you conduct database workload replay, industry-standard benchmark testing, and scalability testing. It enables you to deploy changes to your database environment with confidence by reducing the risks associated with patches, upgrades, migrations, and adjustments to virtual machine configurations. With this workload replay and scheduling software, you can eliminate slow SQL database performance and dramatically simplify high-performance database management. We used an OLTP database test in Benchmark Factory to run a TPC-C-like workload.

For more information, visit http://www.quest.com/benchmark-factory/.

About our VMware vSphere and vCenter Server configuration

About VMware vSphere and vCenter Server

VMware vSphere is the flagship platform of virtualized hardware from VMware. vSphere allows companies to virtualize their server, storage, and networking resources, achieving a consolidation ratio in some cases greater than 15:1. Features included in vSphere include automated management and dynamic resource allocation. To learn more about VMware vSphere, visit www.vmware.com/products/vsphere/overview.html.

According to VMware, vCenter Server "provides a centralized platform for managing VMware vSphere environments." vCenter offers visibility into a vSphere infrastructure to analyze and fix issues. Features such as automated load balancing and out-of-the-box automation workflows create automated proactive management. vCenter can work with third-party solutions to extend its virtualization capabilities. For more information, visit www.vmware.com/products/vcenter-server/.

About our VMware vSphere networking configuration

We created two vSphere-distributed switches for cluster networks: the first for VM management, Oracle Application, and RAC interconnects; the second for vMotion traffic. Each had a maximum MTU of 9,000 bytes, and used two uplink ports (physical NICs). We added the three hosts to each distributed vSwitch with physical NICs vmnic0

and vmnic1 as uplinks for the first switch, and NICs vmnic2 and vmnic3 as uplinks for the second switch.

For the vMotion network specifically, we setup multi-NIC vMotion on the vSphere hosts according to VMware best practices[1]. In the Networking inventory area of the vSphere client, we created two VMkernels on the vMotion distributed-vSwitch. We assigned one portgroup on the vMotion subnet to each, and configured teaming and failover. For the first VMkernel, we assigned the first uplink as active and the second as standby. On the second, we assigned the first uplink as standby and the second as active. The vMotion network used Jumbo Frames (MTU 9000).

Our VM virtual NICs used the VMXNET3 type, as it offers the latest paravirtualized benefits for VM NICs.

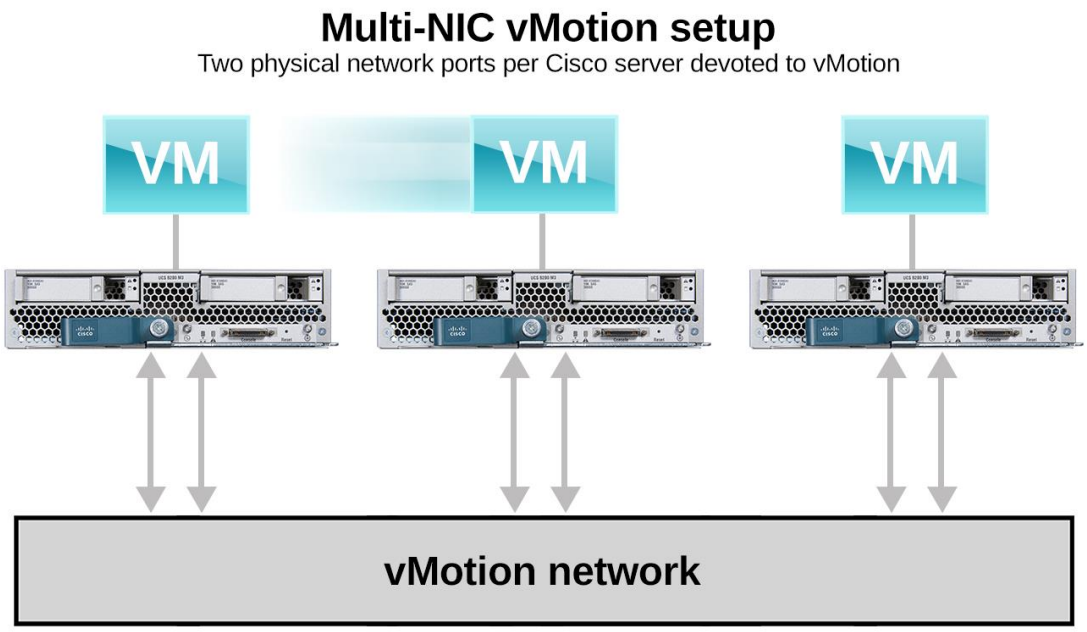


Figure 7: Each server in our test bed had two physical NICs assigned for the vMotion network.

About EMC VMAX Cloud Edition

"Built upon the industry-proven scale-out Virtual Matrix Architecture delivered by the EMC Symmetrix VMAX family," the enterprise-grade, storage-as-a-service VMAX Cloud Edition from EMC offers service providers and enterprises a self-service portal for building private, public, and hybrid clouds. VMAX Cloud Edition is the first of its kind to deliver enterprise-class storage for multitenant use with automated delivery levels of service. With two physical appliance servers that make up the core of its architecture running VMware ESXi and protected by VMware High Availability, VMAX Cloud Edition comes pre-configured and pre-engineered for the specific needs of enterprises. For

[1] http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2007467

more information about EMC VMAX Cloud Edition, see www.emc.com/storage/symmetrix-vmax/vmax-cloud-edition.htm.

About our EMC VMAX Cloud Edition configuration

In our labs, EMC engineers configuration the VMAX Cloud Edition array according to best practices. We coordinated with EMC VMAX Cloud Edition engineers on portal access, cabling, service levels, and monitoring.

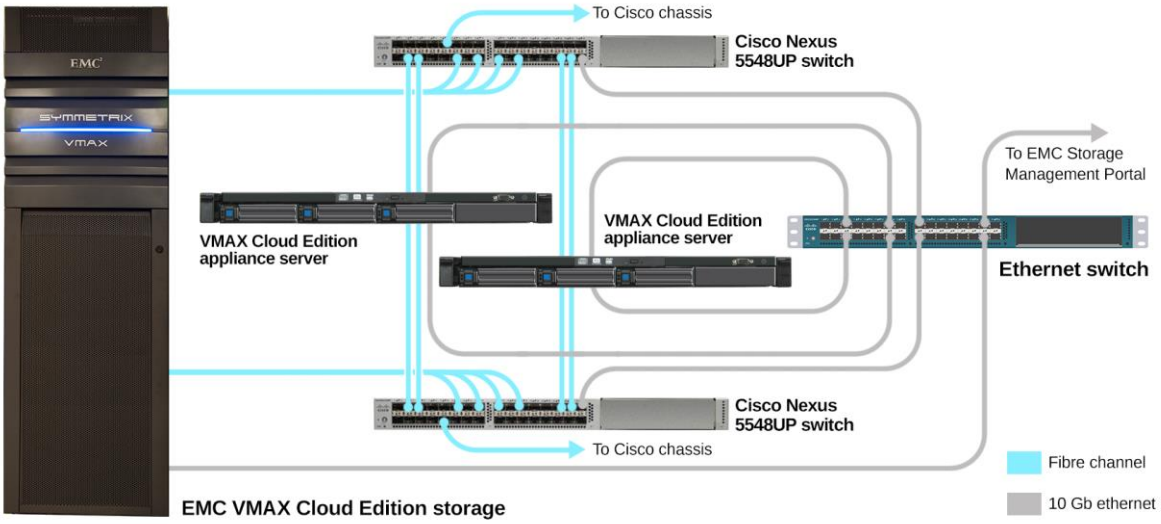


Figure 8: Diagram of the VMAX Cloud Edition

We followed EMC's best practices for allocating high-performance storage volumes for the vSphere datastores associated with the Oracle data and temporary tablespaces. From the VMAX Cloud Edition Web Portal, we created volumes in the DIAMOND-1 level and GOLD-4 service bands. The volume in the DIAMOND-1 service band held the virtual disks containing the Oracle database and Oracle temporary data. The GOLD-4 volumes held virtual disks containing operating system data, as well as the remaining Oracle disks (tablespaces for redo logs, system files, and undo logs). (See Figure 9.)

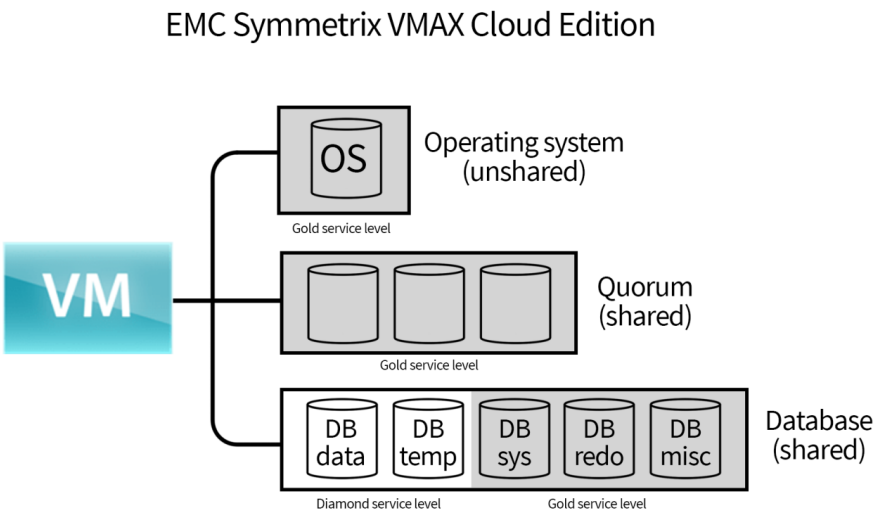


Figure 9: Logical layout of Oracle storage on the EMC VMAX Cloud Edition.

About Cisco UCS infrastructure

The Cisco UCS infrastructure features a converged fabric where all systems management and configuration originates from a pair of redundant layer 2 network switches called Fabric Interconnects (FI). This allows management of large-scale deployments from a single location. The FI is a convergence and distribution point that combines traffic from SAN, management, and data networks into a single converged network that connects directly to the managed compute nodes. As a network scales out and requires multiple layer 2 Fabric Interconnects, Cisco UCS Central consolidates all management systems together into one dashboard. A pair of Fabric Extenders (FEX) connects each Cisco blade chassis to the Fabric Interconnects. With UCS Manager, all firmware versions and updates can apply to existing servers, new nodes, and other hardware with the click of a mouse. For more information, visit www.cisco.com/en/US/products/ps10265/index.html.

In our test bed, we configured the UCS 5108 Blade Server chassis with four cables coming from each FEX (eight in total), going into two UCS 6248UP FI. Each FI was then cabled via four 10Gb Ethernet port and one 8GB FC port to two Cisco Nexus 5548UP switches, with two Ethernet links from each FI connected to each switch, resulting in a fully redundant infrastructure. We aggregated each set of four Ethernet ports into a port channel to ensure maximum bandwidth.

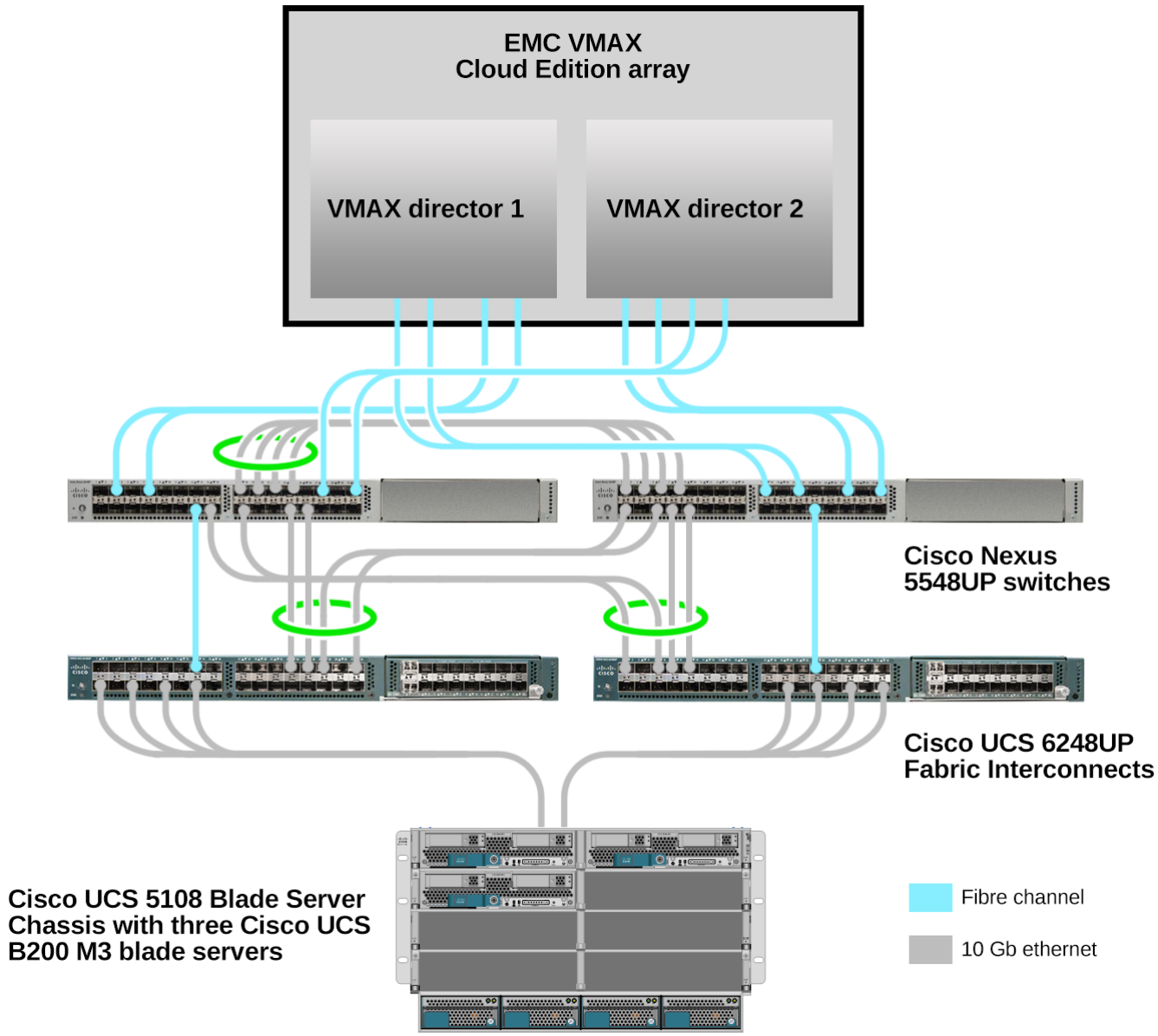


Figure 10: Diagram of the Cisco components in our test bed.

IN CONCLUSION

With today's technologies, you can run large mission-critical databases on virtual machines safely and securely. Creating virtual servers to host databases, instead of running them on physical servers, has the potential to provide a number of time- and cost-saving benefits to your organization, as you can take advantage of the many powerful features and tools that VMware vSphere offers including vMotion, DRS, and much more. We designed this functional stress test to test the complete hardware and software stack, which was comprised of VMware vSphere, Cisco UCS, and EMC VMAX Cloud Edition storage. The EMC VMAX Cloud Edition absorbed the workload's requested storage throughput. The Cisco B200 M3 servers were able to supply the processing power necessary for the workload. The Cisco Nexus switches, despite using only at a fraction of their maximum capacity in this test, allowed for the required network throughput, including the vMotion traffic. This physical infrastructure, used in concert with VMware vSphere, allowed vMotions to occur and the RAC application to remain active despite the migration scenarios we tested, some of which were more extreme than database administrators would typically encounter. The tests experienced no false ejections or RAC cluster fencing operations, showing the mature capabilities of VMware vMotion technology.

As we showed in our tests using Cisco UCS server hardware and EMC VMAX Cloud Edition storage, VMware vSphere with vMotion made it easy to shift large databases and other workloads from one server to another in a cluster, without application downtime. By choosing to run your large databases on VMware vSphere virtual machines, you can reap the benefits of VMware vMotion for the ultimate agility and flexibility in managing your mission-critical database workloads.

APPENDIX A – SYSTEM CONFIGURATION INFORMATION

Figure 11 provides detailed configuration information for the test systems.

System	3x Cisco UCS B200 M3 server
General	
Number of processor packages	2
Number of cores per processor	8
Number of hardware threads per core	2
System power management policy	Default
CPU	
Vendor	Intel
Name	Xeon
Model number	E5-2680
Stepping	7
Socket type	LGA2011
Core frequency (GHz)	2.7
Bus frequency	8.00 GT/s
L1 cache	32KB +32KB
L2 cache	256KB per core
L3 cache	20MB
Platform	
Vendor and model number	Cisco UCS B200 M3
Motherboard model number	Cisco FCH1607GV4
BIOS name and version	Cisco B200M3.2.1.1a.0.121720121447
BIOS settings	Default
Memory modules	
Total RAM in system (GB)	320
Vendor and model number	16x Cisco UCS-MR-1X162RY-A16, 8x Cisco UCS-MR-1X082RY-A
Type	PC3-12800
Speed (MHz)	1,600
Speed running in the system (MHz)	1,333
Size (GB)	(16x) 16, (8x) 8
Number of RAM module(s)	24 (16 + 8)
Chip organization	Double-sided
Rank	Dual
Operating system	
Name	VMware vSphere 5.1.0
Build number	1157734
Language	English
RAID controller	
Vendor and model number	LSI MegaRAID SAS 2004
Firmware version	20.10.1-0100

System	3x Cisco UCS B200 M3 server
Hard drives	
Vendor and model number	Seagate A03-D146GC2
Number of drives	2
Size (GB)	146
RPM	15,000
Type	SAS
Converged I/O Adapters	
Vendor and model number	Cisco UCSB-MLOM-40G-01, Cisco UCS-VIC-M82-8P
Type	mLOM, Mezzanine

Figure 11: Configuration information for the test systems.

APPENDIX B - HOW WE TESTED

For our tests, we set up one Cisco UCS 5108 Blade Server Chassis with two Cisco UCS 6248UP Fabric Interconnects and two Cisco Nexus 5548UP switches. The chassis held three Cisco UCS B200 M3 blade servers, each with two Intel Xeon processors E5-2680 and 384GB RAM. For storage, we used one EMC Symmetrix VMAX Cloud Edition array, provisioned with three levels of storage service, 2,994GB in the DIAMOND-1 level, 18,004 in the GOLD-4 level, and 15.308 in the SILVER-2 level. The storage platform had 48GB of usable cache (96GB raw), and used three types of drives: 16 2TB SATA drives, 106 300GB FC 15K drives, and 24 100GB SSDs. We accessed the storage array from the Cisco Nexus switches with 8 FC cables, running at 8Gbps each.

For our hypervisor, we used VMware vSphere 5.1 and used VMware vCenter Server 5.1 for management. The virtual machines on each server ran the Oracle Enterprise Linux 6.4 x86_64 operating system with the Red Hat kernel. We configured each VM with 16 vCPUs, 156GB RAM, three vNICs, and tuned the profile to Enterprise storage. For database software, we used Oracle Database 11*g* Release 2 for Linux x86-64 (version 11.2.0.3) and Oracle Grid Infrastructure 11*g* Release 2 for Linux x86-64 (version 11.2.0.3).

We used Benchmark Factory for Database 6.9.1 as our test harness, along with Oracle Client for Windows. Figure 12 shows how we set up our test hardware.

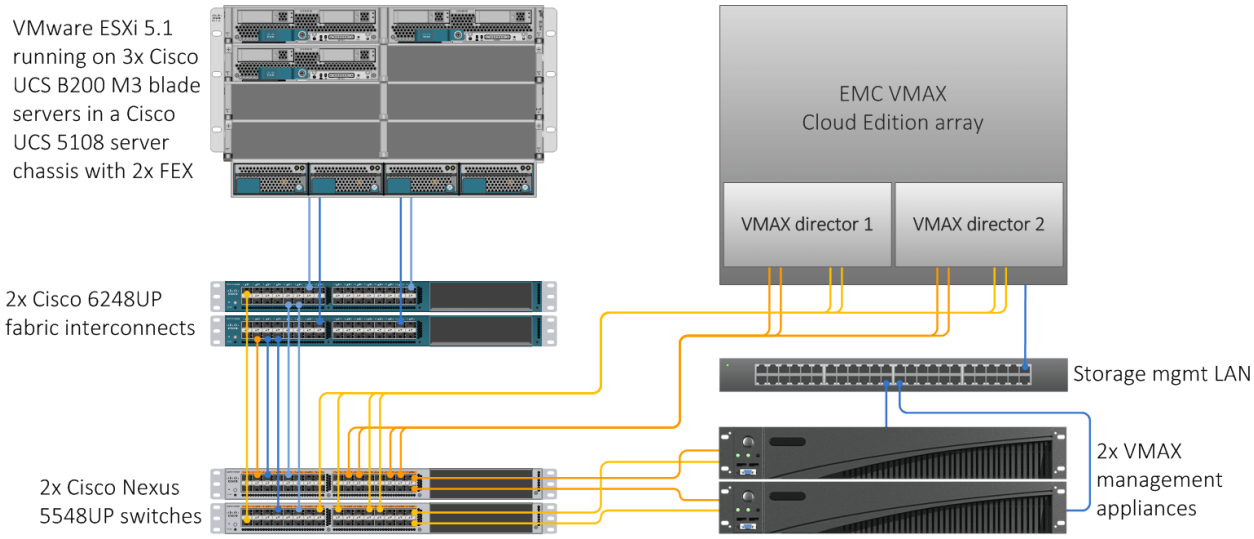


Figure 12: A diagram of our complete testbed.

Storage provisioning on the EMC VMAX Cloud Edition array

We followed EMC's best practices for allocating high-performance storage volumes for the vSphere datastores associated with the Oracle data and temporary tablespaces. From the VMAX Cloud Edition Web Portal, we created a 2,368 GB volume in the DIAMOND-1 level. We created two 4,064GB volumes in the GOLD-4 level. The first held the unshared VM operating-system disks and the Oracle REDO volume. The second held the remaining Oracle disks (tablespaces for redo logs, system files, and undo logs).

We configured the FC zoning so that the three Cisco blades running the hypervisors could access these volumes.

In vSphere, we created one VMware VMFS datastore on each FC volume, naming the datastores vmax-diam-01, vmax-gold-01, and vmax-gold-02.

Configure the VMware cluster

We uploaded the Oracle Linux 6.4 x86_64 installation ISO image to a shared datastore. We configured the hypervisors to use EMC Powerpath/VE to manage the paths to the EMC VMAX Cloud Edition.

We created two distributed vSwitches for cluster networks: the first for VM management, Oracle Application, and RAC interconnects; the second for vMotion traffic. Each had a maximum MTU of 9000 bytes, and used two uplink ports (physical NICs). We added the three hosts to each distributed vSwitch with NICs vmnic0 and vmnic1 as uplinks for the first switch, and NICs vmnic2 and vmnic3 as uplinks for the second switch.

Creating the distributed port groups for the cluster

Create three port groups on the first distributed vSwitch as follows:

1. Log onto the vSphere client with administrator privileges.
2. Press Ctrl-Shift-N to go to the Networking view.
3. Right-click the first distributed vSwitch, and select New Port Group.
4. On the Create Distributed Port Group screen, type `dManagement` for the Name, `VLAN` for the VLAN type, and `VLAN 10` for the VLAN ID. Click Next.
5. To create the port group, click Finish.
6. Right-click the dManagement port group, and select Edit Settings.
7. On the Settings screen, select Teaming and Failover.
8. Set the Load Balancing to Route based on originating virtual port, Network Failover Detection to Link status only, Notify Switches to Yes, and Fallback to Yes.
9. Move dvUplink1 to Active Uplinks and dvUplink2 to Standby Uplinks.
10. Click OK.
11. Repeat Steps 3 through 10 for the dApplication port group with this difference: VLAN ID is 100.
12. Repeat Steps 3 through 10 for the dRAC port group with these differences: VLAN ID is 101, and move dvUplink2 to Active Uplinks and dvUplink1 to Standby Uplinks.

Create two port groups for vMotion on the second distributed vSwitch as follows:

1. Right-click the second distributed vSwitch, and select New Port Group.
2. On the Create Distributed Port Group screen, enter vMo-01 for the Name, VLAN for the VLAN type, and VLAN 100 for the VLAN ID. Click Next.
3. To create the port group, click Finish.
4. Right-click the vMo-01 port group, and select Edit Settings.
5. On the Settings screen, select Teaming and Failover.
6. Set the Load Balancing to Route based on originating virtual port, Network Failover Detection to Link status only, Notify Switches to Yes, and Fallback to Yes.
7. Move dvUplink1 to Active Uplinks and dvUplink2 to Standby Uplinks.
8. Click OK.
9. Repeat Steps 1 through 8 for the vMo-02 port group with these differences: move dvUplink2 to Active Uplinks and dvUplink1 to Standby Uplinks.

Configuring the hypervisors to use the distributed port groups

For each hypervisor, add three VMkernels for VMware management and vMotion as follows:

1. Log onto the vSphere client with administrator privileges.
2. Press Ctrl-Shift-H to go to the Hosts and Clusters view.

3. Select the first host, and click the Configuration tab.
4. Select Networking.
5. Select vSphere Distributed Switch.
6. Select the first distributed vSwitch.
7. Click Manage Virtual Adapters.
8. On the Manage Virtual Adapters screen, click Add.
9. On the Creation Type screen, select New virtual adapter, and click Next.
10. On the Virtual Adapter Types, select VMkernel, and click Next.
11. On the Network Connection screen, select the dManagement port group, select Use this virtual adapter for management traffic, and click Next.
12. Enter the IP Address and Subnet Mask (e.g., 10.4.1.231/255.255.255.0 for the first host, 10.4.1.232/255.255.255.0 for the second, and 10.4.1.233/255.255.255.0 for the third), and click Next.
13. To create the VMware Management VMkernel, click Finish.
14. On the Manage Virtual Adapters screen, click Close.
15. Select the second distributed vSwitch.
16. Click Manage Virtual Adapters.
17. On the Manage Virtual Adapters screen, click Add.
18. On the Creation Type screen, select New virtual adapter, and click Next.
19. On the Virtual Adapter Types, select VMkernel, and click Next.
20. On the Network Connection screen, select the vMo-01 port group, select Use this virtual adapter for vMotion, and click Next.
21. Enter the IP Address and Subnet Mask (e.g., 172.0.0.1/255.255.255.0 for the first host, 172.0.0.2/255.255.255.0 for the second host, and 172.0.0.3/255.255.255.0 for the third), and click Next.
22. To create the VMware Management VMkernel, click Finish.
23. On the Manage Virtual Adapters screen, select the new VMkernel, and click Edit.
24. On the Edit Virtual Adapter screen, enter 9000 for the MTU, and click OK.
25. On the Manage Virtual Adapters screen, click Add.
26. On the Creation Type screen, select New virtual adapter, and click Next.
27. On the Virtual Adapter Types, select VMkernel, and click Next.
28. On the Network Connection screen, select the vMo-02 port group, select Use this virtual adapter for vMotion, and click Next.
29. Enter the IP Address and Subnet Mask (e.g., 172.0.0.9/255.255.255.0 for the first host, 172.0.0.10/255.255.255.0 for the second host, and 172.0.0.11/255.255.255.0 for the third), and click Next.
30. To create the VMware Management VMkernel, click Finish.
31. On the Manage Virtual Adapters screen, select the new VMkernel, and click Edit.
32. On the Edit Virtual Adapter screen, enter 9000 for the MTU, and click OK.
33. On the Manage Virtual Adapters screen, click Close.

Creating the first VM and provisioning VMware disks for Oracle and the operating systems

We created VMware disks for only one VM because we cloned two VMs from the first VM (which duplicated the operating system and application disks), and the VMs share the Oracle disks so that only one copy is needed.

Installing the operating system on the first VM

1. Log onto the vSphere client with administrator privileges.
2. Right-click on the datacenter, and select New Virtual Machine.
3. Select Custom configuration, and click Next.
4. Enter node01 for Name, and click Next.
5. Select one host in the VMware cluster, and click Next.
6. Select the datastore for OS disks created previously (i.e., vmax-gold-01), and click Next.

7. On the Virtual Machine Version screen, keep the defaults, and click Next.
8. Select Linux for the Guest Operating System, and Oracle Linux 4/5/6 as the Version, and click Next.
9. Type `1` for the Number of Virtual sockets and `16` for the Number of cores per virtual socket. Click Next.
10. Type `156GB` for the memory size, and click Next.
11. Select 3 NICs. For each, click Connect at Power On and select VMXNET 3 as the adapter. The first NIC connects to Network dManagement, the second NIC to dApplication, and the third to dRAC. Click Next.
12. Keep the default SCSI controller, and click Next.
13. Type `60GB` for the size of the Virtual disk, to be created as Thick Provision Eager Zeroed, and click Next.
14. Keep the default Virtual Device Node, and click Next.
15. Select Edit the virtual machine settings before completion, and click Continue.
16. Select CD/DVD drive 1, select Datastore ISO file, click Browse, navigate to the datastore with the Oracle Linux installation ISO, select the ISO, and click OK.
17. Select Connect at power on.
18. Select the Resources Tab.
19. Select Memory, and enter 148480MB for the resource reservation.
20. To create the VM, click OK.
21. Power on the new VM.
22. Open a console window.
23. On the Welcome to Oracle Linux Server 6.4 screen, select Install or upgrade an existing system, and press Enter.
24. On the test media screen, select Skip, and press Enter.
25. On the Oracle Linux 6 screen, click Next.
26. Select the installation language (e.g., English), and click Next.
27. Select the installation keyboard style (e.g., U.S. English), and click Next.
28. On the Storage Device screen, select Basic Storage Devices, and click Next.
29. On the Storage Device Warning screen, review the disk information for accuracy, and click Yes, discard any data.
30. Enter node01 for the hostname, and click Next.
31. Select the time zone (e.g., America/New York), select System clock uses UTC, and click Next.
32. Enter the root password (twice), and click Next.
33. On the type of installation screen, select Replace Existing Linux System(s), check Review and modify partitioning layout, and click Next.
34. On the partitioning layout screen, select the lv_root logical volume, and click Edit.
35. On the Edit Logical Volume screen, change the size to 15360MB, and click OK.
36. Select the lv_swap logical volume, and click Edit.
37. Change its size to 20480MB, and click OK.
38. Select the lv_home logical volume, and click Edit.
39. Change the name of the lv_home logical volume to lv_opt. Change its mount point to /u01, and change its size to the maximum available (approximately 25000MB). Click OK.
40. On the partitioning layout screen, click Next.
41. On the Format Warnings screen, click Format.
42. Click Write changes to disk.
43. On the boot-loader screen, accept the defaults, and click Next.
44. Select Basic Server, Customize Later, and click Next.
45. Click Reboot.

Configuring the guest for Oracle RAC

1. Log onto the guest as root.
2. Replace the content of the file /etc/resolv.conf with the following:

```
nameserver 10.1.1.10
search t1.vvv
```

3. Add the default network gateway to /etc/sysconfig/network, e.g., GATEWAY=10.4.0.1.
4. Disable SELinux, by changing the line `SELINUX=enforcing` to `SELINUX=disabled` in the file `/etc/selinux/config`.
5. Modify the file /etc/sysconfig/network-script/ifcfg-eth0 as follows, being careful to retain the device's MAC address:

```
DEVICE=eth0
## Be sure to use the MAC address for this VM's first NIC
HWADDR=00:50:56:AD:07:12
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
IPADDR=10.4.1.232
PREFIX=24
```

6. Modify the file /etc/sysconfig/network-script/ifcfg-eth1 as follows, being careful to retain the device's MAC address:

```
DEVICE=eth1
## Be sure to use the MAC address for this VM's second NIC
HWADDR=00:50:56:AD:14:41
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
IPADDR=10.1.1.101
PREFIX=24
```

7. Modify the file /etc/sysconfig/network-script/ifcfg-eth2 as follows, being careful to retain the device's MAC address:

```
DEVICE=eth1
## Be sure to use the MAC address for this VM's third NIC
HWADDR=00:50:56:AD:A7:95
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
IPADDR=192.168.1.101
PREFIX=24
MTU=9000
```

8. Create Oracle users and groups by running these shell commands:

```
groupadd -g 1001 oinstall
groupadd -g 1002 dba
groupadd -g 1003 asmadmin
groupadd -g 1004 asmdba
useradd -u 1002 -g oinstall -G dba,asmadmin,asmdba oracle
useradd -u 1003 -g oinstall -G dba,asmadmin,asmdba grid

echo 'ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1' \
  >> ~oracle/.bash_profile
echo 'ORACLE_HOME=/u01/app/11.2.0/grid' >> ~grid/.bash_profile
for i in oracle grid; do
  (
    echo 'PATH=$PATH:$HOME/bin:$ORACLE_HOME/bin'
    echo 'export PATH ORACLE_HOME'
    echo 'umask 0022'
  ) >> /home/$i/.bash_profile
```

```
done

mkdir -p /u01/app/oracle
mkdir /u01/app/grid
chown -R oracle:oinstall /u01/app
chmod -R g+w /u01/app
```

9. Create passwords for the oracle and grid accounts with passwd.
10. The following services are disabled as follows:

```
for i in autofs certmonger cups iptables ip6tables netfs nfs\
     nfslock portreserve postfix rpcbind rpcgssd rpcidmapd; do
   chkconfig $i off
   service $i stop
done
```

11. Enable the following time services:

```
chkconfig ntpd on
chkconfig ntpdate on
```

12. Modify the NTP configuration by adding your local NTP servers to /etc/ntp.conf.
13. Modify the behavior of NTP by appending "-x" to its OPTIONS list in the file `/etc/sysconfig/ntpd`
14. Install the following packages:

```
yum install binutils compat-libstdc++-33 elfutils-libelf \
  elfutils-libelf-devel elfutils-libelf-devel-static gcc gcc-c++ \
  glibc glibc-common glibc-devel glibc-headers ksh libaio \
  libaio-devel libgcc libstdc++ libstdc++-devel make sysstat \
  unixODBC unixODBC-devel compat-libcap1
yum install compat-libstdc++-33.i686 glibc.i686 libaio.i686 \
  libaio-devel.i686 libattr.i686 libcap.i686 libgcc.i686 \
  libstdc++.i686 libtool-ltdl.i686 ncurses-libs.i686 \
  nss-softokn-freebl.i686 readline.i686 unixODBC.i686 unixODBC-devel.i686
yum install libdmx libX11 libX11-common libX11-devel libXau \
  libXau-devel libxcb-devel libXcomposite libXcursor libXdamage \
  libXext libXext-devel libXfixes libXfont libXft libXi libXinerama \
  clibXmu libXrandr libXrender libXScrnSaver libXScrnSaver-devel libXt \
  libXtst libXv libXxf86dga libXxf86misc libXxf86vm \
  xorg-x11-fonts-Type1 xorg-x11-proto-devel xorg-x11-utils xorg-x11-xauth
yum install tuned oracleasm-support kmod-oracleasm kernel-devel
```

15. Install VMware Tools by right clicking on the VM in vSphere and selecting Guest→Install/Upgrade VMware Tools. Click OK.
16. From the login shell, mount /dev/sr0 /mnt.
17. Run tar zxf /mnt/VMwareTools*gz.
18. Run ./vmware-tools-distrib/vmware-install.pl and accept the defaults.
19. Set the tuned profile to enterprise storage:

```
tune-adm profile enterprise-storage
```

20. Disable anonymous hugepages by appending the following lines to /etc/rc.local:

```
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
   echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
if test -f /sys/kernel/mm/transparent_hugepage/defrag; then
   echo never > /sys/kernel/mm/transparent_hugepage/defrag
fi
```

21. Append the following to /etc/security/limits.conf:

```
oracle -  nofile  65536
oracle -  nproc   16384
oracle -  stack   32768
oracle -  memlock 152043520
grid   -  nofile  65536
grid   -  nproc   16384
grid   -  stack   32768
```

22. We modified the system's kernel parameters by appending the following to /etc/sysctl.conf:

```
kernel.shmmax = 83751862272
# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 4294967296
fs.file-max = 6815744
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_max = 1048576
fs.aio-max-nr = 1048576
kernel.sem = 250 32000 100 128
kernel.shmmni = 4096
vm.nr_hugepages = 74251
```

23. Update the RPM packages:

```
yum update -y
```

24. Edit the file /etc/grub.conf, and change the default kernel from the Oracle UEK to the most recent (highest numbered) Red Hat kernel.
25. Shut down the VM:

```
shutdown –h now
```

Cloning the first VM to create the remaining VMs

We created the next two VMs as clones of the first. After cloning in vSphere, we booted the new VMs one at a time to update their hostnames and network configurations as follows:

1. Log onto the VM as root.
2. Change the hostname from node01 to node02 (node03) by altering the HOSTNAME line in the file /etc/sysconfig/network.
3. Record the VM's MAC addresses for each NIC by running the command `ipconfig –a`
4. Update the configuration of the first NIC by altering two lines in the file /etc/sysconfig/network-script/ifcfg-eth0: Replace the HWADDR with the MAC address found in the previous step, and replace the IPADDR with 10.4.1.102 for the second VM (10.4.1.103 for the third).
5. Update the configuration of the second NIC by altering two lines in the file /etc/sysconfig/network-script/ifcfg-eth0: Replace the HWADDR with the MAC address found in step 4, and replace the IPADDR with 10.1.1.102 for the second VM (10.1.1.103 for the third).
6. Update the configuration of the third NIC by altering two lines in the file /etc/sysconfig/network-script/ifcfg-eth0: Replace the HWADDR with the MAC address found in step 4, and replace the IPADDR with 192.168.1.102 for the second VM (192.168.1.103 for the third).
7. Shut down the VM:

```
shutdown –h now
```

Creating and configuring the shared disks for Oracle.

We created the shared disks by editing the configuration of the first VM, and then added the disks to the other VMs.

1. Log onto vSphere with administrator privileges.
2. Right-click on the first VM, and select Edit Settings.
3. On the Virtual Machine Properties screen, click Add to create the first Oracle-Grid Voting disk.
4. On the Add Hardware screen, select Hard Disk, and click Next.
5. Select Create a new virtual disk, and click Next.
6. Enter 20GB for the Disk Size, to be created as Thick Provision Eager Zeroed, and select vmax-gold-02 as its datastore. Click Next.
7. Select SCSI (1:0) for the Virtual device Node. Select Independent Mode and Persistent. Click Next.
8. To add this disk to the creation tasks, click Finish.
9. Select the New SCSI Controller (for bus 1) from the list of Hardware, and click Change Type. On the pop-up screen, select VMware Paravirtual, and click OK.
10. Repeat steps 3 through 8 for the second and third Voting disks (20GB) with these differences: place the second disk on SCSI 1:1, and the third disk on SCSI 1:2.
11. Create the Oracle data disk by repeating steps 3 through 8 with these differences: use datastore vmax-diam-01, size it at 1TB, and place it on SCSI 1:3.
12. Create the Oracle temporary disk by repeating steps 3 through 8 with these differences: use datastore vmax-diam-01, size it at 800GB, and place it on SCSI 1:4
13. Create the Oracle system disk by repeating steps 3 through 8 with these differences: use datastore vmax-gold-02, size it at 100GB, and place it on SCSI 1:5.
14. Create the Oracle REDO disk by repeating steps 3-8 with these differences: use datastore vmax-gold-02, size it at 200, and place it on SCSI 2:0.
15. Select the New SCSI Controller (for bus 2) from the list of Hardware, and click Change Type. On the pop-up screen, select VMware Paravirtual, and click OK.
16. Click OK to close the Virtual Machine Properties screen and create the virtual disks.
17. After the disks are created, right-click the first VM, and select Edit Settings.
18. On the Virtual Machine Properties screen, select the Options Tab.
19. Under Advanced settings, click General.
20. Click Configuration Parameters.
21. On the Configuration Parameters screen, click Add Row. Enter `scsi1:0.sharing` for Name, and `multi-writer` for value.
22. Repeat step 21 for each of the shared disks, changing the Name (`scsi1:1`, …, `scsci1:6`, and `scsi2:0`).
23. Click OK.
24. To close the Virtual Machine Properties screen, click OK.
25. To add the shared disks to the second VM, right-click the first VM, and select Edit Settings.
26. On the Virtual Machine Properties screen, click Add to create the first Oracle-Grid Voting disk.
27. On the Add Hardware screen, select Hard Disk, and click Next.
28. Select Use an existing virtual disk, and click Next.
29. On the Select Existing Disk screen, click Browse.
30. Select the vmax-gold-02 datastore, and click Open.
31. On the Browse Datastores screen, select node01, and click Open.
32. Select disk node01.vmdk, and click OK.
33. On the Select Existing Disk screen, click Next.
34. Select SCSI (1:0) for the Virtual device Node. Select Independent Mode and Persistent. Click Next.
35. To add this disk to the creation tasks, click Finish.

36. Select the New SCSI Controller (for bus 1) from the list of Hardware, and click Change Type. On the pop-up screen, select VMware Paravirtual, and click OK.
37. Repeat steps 26 through 35 for the second Voting disk with these differences: use existing disk node01/node01_1.vmdk on datastore vmax-gold-02 and place it on SCSI 1:1.
38. Repeat steps 26 through 35 for the third Voting disk with these differences: use existing disk node01/node01_2.vmdk on datastore vmax-gold-02 and place it on SCSI 1:2.
39. Repeat steps 26 through 35 for the Oracle data disk with these differences: use existing disk node01/node01.vmdk on datastore vmax-diamond-01 and place it on SCSI 1:3.
40. Repeat steps 26 through 35 for the Oracle temporary disk with these differences: use existing disk node01/node01_1.vmdk on datastore vmax-diamond-01 and place it on SCSI 1:4.
41. Repeat steps 26 through 35 for the Oracle system disk with these differences: use existing disk node01/node01_3.vmdk on datastore vmax-gold-02 and place it on SCSI 1:5.
42. Repeat steps 26-35 for the Oracle miscellaneous disk with these differences: use existing disk node01/node01_4.vmdk on datastore vmax-gold-02 and place it on SCSI 1:6.
43. Repeat steps 26 through 35 for the Oracle REDO disk with these differences: use existing disk node01/node01_1.vmdk on datastore vmax-gold-01 and place it on SCSI 2:0.
44. Select the New SCSI Controller (for bus 2) from the list of Hardware, and click Change Type. On the pop-up screen, select VMware Paravirtual, and click OK.
45. Repeat steps 16 through 24 to complete the addition of the shared disks for the second VM.
46. Repeat steps 25 through 44 to add the shared disks to the third VM.
47. Boot the three VMs.
48. Log onto the first VM as root. The following tasks, shared disk partitioning and ASM disk marking, need be performed only once on a single system. However, as noted below, ASM configuration must be performed on each system.
49. Examine the output of parted to determine which device each virtual disk has been mapped to. For example, /dev/sdb might correspond to the first REDO disk. We assume for the purposes of demonstration that the shared disks correspond to devices /dev/sd[b-i], inclusive.
```
parted -l
```
50. For each of the eight shared disks, create a GPT label, and create one partition.. For example, see the following shell script:
```
for disk in sdb sdc sdd sde sdf sdg sdh sdi; do
  parted /dev/$disk mklabel gpt
  parted /dev/$disk mkpart primary "1 -1"
done
```
51. If desired, label the disk's partition with its Oracle function. For example,
```
parted /dev/sdb name 1 CRS1
parted /dev/sdc name 1 CRS2
parted /dev/sdd name 1 CRS3
parted /dev/sde name 1 DATA
parted /dev/sdf name 1 TEMP
parted /dev/sdg name 1 SYST
parted /dev/sdh name 1 MISC
parted /dev/sdi name 1 REDO
```
52. Initialize Oracle ASM on each server by executing the following commands as root on each node.
```
oracleasm init
oracleasm configure -e -u grid -g oinstall -s y -x sda
```
53. Label each shared disk-partition with an appropriate ASM name. For example, following the OS partition names created above, execute the following commands on one system:
```
oracleasm createdisk CRS1 /dev/sdb1
```

```
oracleasm createdisk CRS2 /dev/sdc1
oracleasm createdisk CRS3 /dev/sdd1
oracleasm createdisk DATA /dev/sde1
oracleasm createdisk TEMP /dev/sdf1
oracleasm createdisk SYST /dev/sdg1
oracleasm createdisk MISC /dev/sdh1
oracleasm createdisk REDO /dev/sdi1
```

54. On each server, scan the disks to make the disks immediately available to Oracle ASM.

```
oracleasm scandisks
oracleasm listdisks
```

Creating the Oracle RAC Database (version 11.2.0.3)

Installing Oracle Grid Infrastructure 11*g* Release 2 for Linux x86-64 on the VMs

We used the following steps to configure the Oracle RAC Database. Figure 13 shows the logical layout of the VM networking, the Oracle cluster network, and the incoming application network.

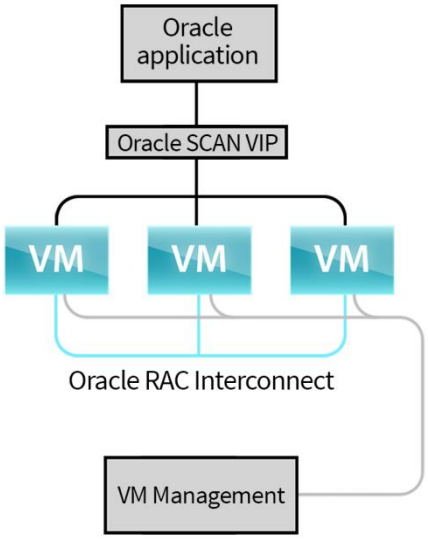


Figure 13: Logical layout of VM networking.

1. Log into the server as the grid account.
2. Set the X Window DISPLAY variable as appropriate for your configuration.
3. Unzip the Oracle Grid Infrastructure software distribution file.
   ```
   unzip p10404530_112030_Linux-x86-64_3of7.
   ```
4. Run the Grid Infrastructure GUI installer.
   ```
   ./grid/runInstaller
   ```
5. On the Download Software Updates screen, select Skip software updates, and click Next.
6. On the Select Installation Option screen, select Install and Configure Grid Infrastructure for a Cluster, and click Next.
7. On the Select Installation Type screen, select Advanced Installation and click Next.
8. On the Select Product Languages screen, keep the default (English), and click Next.
9. On the Grid Plug and Play Information screen, deselect Configure GNS, enter clust00 for the Cluster Name, clust00-scan.t1.vvv for the SCAN Name, and click Next.
10. On the Cluster Node Information screen, click Add.

11. On the Add Cluster Node Information pop-up screen, enter node02.t1.vvv, and node02-vip.t1.vv for the Public and Virtual Hostnames, respectively, and click OK.
12. Repeat step 11 for the third node.
13. Click SSH Connectivity.
14. Enter the OS password for the grid user account, click Next.
15. On the Specify Network Interface Usage screen, set eth0 to Do Not Use, eth1 to Public, and eth2 to Private, and click Next.
16. On the Storage Option Information screen, select Oracle Automatic Storage Management (Oracle ASM), and click Next.
17. On the Create ASM Disk Group screen, click Change Discovery Path.
18. On the Change Discovery Path pop-up screen, enter /dev/oracleasm/disks/*, and click OK.
19. Change the Disk Group Name to OCRV.
20. Select Normal Redundancy.
21. Select disks CRS1, CRS2, and CRS3, and click Next.
22. On the Specify ASM Password, select Use same password for these accounts, and enter the ASMSNMP password (twice), and click Next.
23. On the Failure Isolation Support screen, select Do not use Intelligent Platform Management Interface (IPMI), and click Next.
24. On the Privileged Operating System Groups screen, keep the defaults, and click Next.
25. On the specify Installation location screen, enter /u01/app/grid for Oracle Base, /u01/app/11.2.0/grid for Software Location, and Click Next.
26. On the Create Inventory screen, enter /u01/app/oraInventory for the Inventory Directory, and click Next.
27. On the Perform Prerequisite Checks screen, click Fix and Check Again (to install the missing cvuqdisk package).
28. On the Execute Fixup Scripts pop-up screen, follow the instructions for running the fix-up script on each node. Click OK when done.
29. Click Next and after the check is finished.
30. On the Summary screen, click Install.
31. When the Execute Configuration scripts pop-up screen appears, follow the instructions for running the scripts, and click OK when finished.
32. Click Close to end the installation.

Creating Oracle ASM disk groups for the database

1. Log into the system as the grid user.
2. Set the X Window DISPLAY variable as appropriate for your configuration.
3. Start the ASM configuration assistant, asmca.
4. On the Disk Groups tab, click Create.
5. On the Create Disk Group pop-up screen, enter DATA for the Disk Group Name.
6. Select External (None) for Redundancy.
7. Select/dev/oracleasm/disks/ DATA for the Disk Path.
8. Click Show Advanced Options, and change the Allocation Unit Size (ALU) to 4MB.
9. Click OK to create the DATA disk group, and click OK on the completed-task pop-up screen
10. Repeat steps 4-9 to create disk group TEMP from disk /dev/oracleasm/TEMP.
11. Repeat steps 4-9 to create disk group MISC from disk /dev/oracleasm/MISC.
12. Repeat steps 4-9, except that the ALU in step 8 is the default 1MB, to create disk group SYST from disk /dev/oracleasm/SYST.
13. Repeat steps 4-9, except that the ALU in step 8 is the default 1MB, to create disk group REDO from disk /dev/oracleasm/REDO.
14. Click Exit to close the ASM configuration assistant.

Installing the software for Oracle Database 11*g* Release 2 for Linux x86-64 on the VMs

1. Log into the system as the oracle user.
2. Set the X Window DISPLAY variable as appropriate for your configuration.
3. Unzip the Oracle Database software distribution files.
```
unzip p10404530_112030_Linux-x86-64_1of7.zip
unzip p10404530_112030_Linux-x86-64_2of7.zip
```
1. Run the Database GUI installer.
```
./database/runInstaller
```
2. On the Configure Security Updates screen, unselect I wish to receive security updates via My Oracle Support, and click Next.
3. On the warning pop-up screen, click Yes.
4. On the Download Software Updates screen, select Skip software updates, and click Next.
5. On the Select Installation Option screen, select Install database software only, and click Next.
6. On the Grid Installation Options screen, select Oracle Real Application Clusters database installation, select the nodes in the cluster, and click SSH Connectivity.
7. On the Select Product Languages screen, keep the default (e.g., English) and click Next.
8. On the Select Database Edition screen, select Enterprise Edition and click Next.
9. On the Specify Installation Location screen, enter /u01/app/oracle for Oracle Base and /u01/app/oracle/product/11.2.0/dbhome_1 for Software Location and click Next.
10. On the Privileged Operating System Groups screen, keep the defaults and click Next.
11. On the Perform Prerequisite Checks screen, click Next.
12. On the Summary screen, click Install.
13. When the Execute Configurations scripts screen appears, follow the instructions and clock OK.
14. On the final screen, click Close to end the installation.

Creating the database

1. Log into the system as the oracle user.
2. Set the X Window DISPLAY variable as appropriate for your configuration.
3. Start the database configuration assistant, dbca.
4. On the Welcome screen, select Oracle Real Application Clusters (RAC) database, and click Next
5. On the Operations screen, select Create a Database and click Next.
6. On the Database Templates screen, select General Purpose or Transaction Processing, and click Next.
7. On the Database Identification screen, select Admin-Managed, enter oltp.t1.vvv for Global Database Name, oltp for the SID prefix, and select all three nodes for the cluster. Click Next
8. On the Management Options screen, keep the defaults and click Next.
9. On the Database Credentials screen, select Use the Same Administrative Password for All Accounts, enter the password twice, and click Next.
10. On the Database File Locations screen, select ASM for Storage Type, select Use Common Location for All Database Files, enter +SYST for Database Files Locations, and click Next.
11. Enter the ASMSNMP password on the ASM Credentials pop-up screen, and click OK.
12. On the Recovery Configuration screen, keep the defaults, and click Next.
13. On the Database Content screen, keep the defaults and click Next.
14. On the Initialization Parameters screen, click Custom, enter 148480MB for SGA Size, enter 3686MB for PGA Size, and click Next.
15. On the Database Storage screen, click Redo Log Groups. Modify the configuration so that each of the six groups has two 5,120,000KB log files on disk-group +REDO. To add the second log-file, left click on the leftmost rectangle on an empty row in the members-grid, and click Add in the small pop-up window. Click Next when done.
16. On the Creation Options screen, select Create Database, and click Finish.

17. On the Summary pop-up screen, click OK.
18. Click Exit on the final pop-up screen to close the configuration assistant.

Configuring the Benchmark Factory workload generator

We installed the 64-bit builds of Quest Software's Benchmark Factory for Database, version 6.9.1, and Oracle Client 11.2.0 on Microsoft Windows Server 2012. We replaced the contents of the TNSNAMES.ora and SQLNET.ora files in folder network\admin in the Oracle Client directory with the following:

```
# tnsnames.ora
OLTP=
  (DESCRIPTION=
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=clust00-scan.t1.vvv)
      (PORT=1521)
    )
    (CONNECT_DATA=
      (SERVER=dedicated)
      (SERVICE_NAME=oltp.t1.vvv)
    )
  )
# end of tnsnames.ora
# sqlnet.ora
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
# end of sqlnet.ora
```

We used the Load Scenario Wizard in Benchmark Factory to create the components for creating the TCP-C like database objects and the template for the workload generation. We used a benchmark scale of 5,000 to create approximately 320GB of data. We configured Benchmark Factory to create the tables and indices with up to three times the default degree of parallelism. We specified the placement of the tables on tablespace USRS and the indices on tablespace INDX (see next section). The workload ran for 40 minutes, 20 minutes with 90 users, and 20 minutes with 100. We reduced the transaction latencies by a factor of 1,000 from their default values (to about 10ms).

Creating the Oracle RAC Database for TPC-C like workloads

We created the tablespaces for the data and indices on datastores using the DIAMOND-1 storage. Every other object was backed by GOLD-4 storage.

We used the following SQL commands to create the tablespaces and users.

```
CREATE TABLESPACE "USRS" DATAFILE '+DATA/oltp/d16.dbf' SIZE 32767M REUSE,
'+DATA/oltp/d19.dbf' SIZE 32767M REUSE, '+DATA/oltp/d05.dbf' SIZE 32767M
REUSE, '+DATA/oltp/d07.dbf' SIZE 32767M REUSE, '+DATA/oltp/d14.dbf' SIZE
32767M REUSE, '+DATA/oltp/d15.dbf' SIZE 32767M REUSE, '+DATA/oltp/d20.dbf'
SIZE 32767M REUSE, '+DATA/oltp/d08.dbf' SIZE 32767M REUSE,
'+DATA/oltp/d18.dbf' SIZE 32767M REUSE, '+DATA/oltp/d17.dbf' SIZE 32767M
REUSE, '+DATA/oltp/d02.dbf' SIZE 32767M REUSE, '+DATA/oltp/d11.dbf' SIZE
32767M REUSE, '+DATA/oltp/d12.dbf' SIZE 32767M REUSE, '+DATA/oltp/d01.dbf'
SIZE 32767M REUSE, '+DATA/oltp/d13.dbf' SIZE 32767M REUSE,
'+DATA/oltp/d03.dbf' SIZE 32767M REUSE, '+DATA/oltp/d04.dbf' SIZE 32767M
REUSE, '+DATA/oltp/d06.dbf' SIZE 32767M REUSE, '+DATA/oltp/d09.dbf' SIZE
```

```
32767M REUSE, '+DATA/oltp/d10.dbf' SIZE 32767M REUSE LOGGING EXTENT
MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE "MISC" DATAFILE '+MISC/oltp/misc14.dbf' SIZE 32767M REUSE,
'+MISC/oltp/misc05.dbf' SIZE 32767M REUSE, '+MISC/oltp/misc06.dbf' SIZE
32767M REUSE, '+MISC/oltp/misc07.dbf' SIZE 32767M REUSE,
'+MISC/oltp/misc10.dbf' SIZE 32767M REUSE, '+MISC/oltp/misc09.dbf' SIZE
32767M REUSE, '+MISC/oltp/misc11.dbf' SIZE 32767M REUSE,
'+MISC/oltp/misc01.dbf' SIZE 32767M REUSE, '+MISC/oltp/misc08.dbf' SIZE
32767M REUSE, '+MISC/oltp/misc13.dbf' SIZE 32767M REUSE,
'+MISC/oltp/misc15.dbf' SIZE 32767M REUSE, '+MISC/oltp/misc03.dbf' SIZE
32767M REUSE, '+MISC/oltp/misc04.dbf' SIZE 32767M REUSE,
'+MISC/oltp/misc12.dbf' SIZE 32767M REUSE, '+MISC/oltp/misc02.dbf' SIZE
32767M REUSE LOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO

CREATE TEMPORARY TABLESPACE "TEMP" TEMPFILE '+TEMP/oltp/temp01.dbf' SIZE
32767M REUSE, '+TEMP/oltp/temp06.dbf' SIZE 32767M REUSE,
'+TEMP/oltp/temp05.bdf' SIZE 32767M REUSE, '+TEMP/oltp/temp04.dbf' SIZE
32767M REUSE, '+TEMP/oltp/temp03.dbf' SIZE 32767M REUSE,
'+TEMP/oltp/temp02.dbf' SIZE 32767M REUSE EXTENT MANAGEMENT LOCAL UNIFORM
SIZE 1024K;

ALTER TABLESPACE TEMP TABLESPACE GROUP TMP;
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TMP;

CREATE USER "OLTP" IDENTIFIED BY "scott" DEFAULT TABLESPACE "USRS" TEMPORARY
TABLESPACE "TMP";
GRANT ALTER SESSION TO "OLTP";
GRANT CREATE MATERIALIZED VIEW TO "OLTP";
GRANT CREATE VIEW TO "OLTP";
GRANT QUERY REWRITE TO "OLTP";
GRANT UNLIMITED TABLESPACE TO "OLTP";
GRANT EXECUTE ON "SYS"."DBMS_LOCK" TO "OLTP";
GRANT "CONNECT" TO "OLTP";
GRANT "RESOURCE" TO "OLTP";

CREATE USER "HOLDEM" IDENTIFIED BY “scott” DEFAULT TABLESPACE "MISC"
TEMPORARY TABLESPACE "TMP";
GRANT ANALYZE ANY TO "HOLDEM";
GRANT ANALYZE ANY DICTIONARY TO "HOLDEM";
GRANT UNLIMITED TABLESPACE TO "HOLDEM";
GRANT "CONNECT" TO "HOLDEM";
GRANT "RESOURCE" TO "HOLDEM";

alter system set cursor_sharing=force scope=both sid=”*”;
alter system set recyclebin=off scope=both sid=”*”;
alter system set max_dump_file_size=20M scope=both sid=”*”;
alter system set log_checkpoints_to_alert=true scope=both sid=”*”;
alter system set audit_trail=none scope=both sid=”*”;
alter system set trace_enabled=false scope=both sid=”*”;
alter system set star_transformation_enabled=true scope=both sid=”*”;
alter system set filesystemio_options=setall scope=both sid=”*”;\
```

```
alter system set fast_start_mttr_target=3600 scope=both sid="*";
alter system set shared_pool_size=16G scope=both sid="*";
alter system set large_pool_size=512M scope=both sid="*";
alter system set shared_pool_reserved_size=896M scope=both sid="*";
alter system set db_cache_size=125G scope=both sid="*";
alter system set java_pool_size=512M scope=both sid="*";
```

We created the initial OLTP tables with the Benchmark Factory job, created in the previous section, and then copied it to the HOLDEM schema. After each workload-generation run, we re-initialized the OLTP tables from the HOLDEM data, instead of using Benchmark Factory. The following scripts, adapted from those created found at this Quest website http://www.toadworld.com/products/benchmark-factory/f/45/p/1148/202.aspx.

```
--- filename - oltp_init.sql

connect oltp/scott

grant select on c_customer   to holdem;
grant select on c_district   to holdem;
grant select on c_history    to holdem;
grant select on c_item       to holdem;
grant select on c_new_order  to holdem;
grant select on c_order      to holdem;
grant select on c_order_line to holdem;
grant select on c_stock      to holdem;
grant select on c_warehouse  to holdem;

connect holdem/scott

drop table c_customer   cascade constraints purge;
drop table c_district   cascade constraints purge;
drop table c_history    cascade constraints purge;
drop table c_item       cascade constraints purge;
drop table c_new_order  cascade constraints purge;
drop table c_order      cascade constraints purge;
drop table c_order_line cascade constraints purge;
drop table c_stock      cascade constraints purge;
drop table c_warehouse  cascade constraints purge;

create table c_customer   tablespace misc parallel (degree 8) as select *
from oltp.c_customer;
create table c_district   tablespace misc parallel (degree 8) as select *
from oltp.c_district;
create table c_history    tablespace misc parallel (degree 8) as select *
from oltp.c_history;
create table c_item       tablespace misc parallel (degree 8) as select *
from oltp.c_item;
create table c_new_order  tablespace misc parallel (degree 8) as select *
from oltp.c_new_order;
```

```
create table c_order      tablespace misc parallel (degree 8) as select *
from oltp.c_order;
create table c_order_line tablespace misc parallel (degree 8) as select *
from oltp.c_order_line;
create table c_stock      tablespace misc parallel (degree 8) as select *
from oltp.c_stock;
create table c_warehouse  tablespace misc parallel (degree 8) as select *
from oltp.c_warehouse;

grant select on c_customer   to oltp;
grant select on c_district   to oltp;
grant select on c_history    to oltp;
grant select on c_item       to oltp;
grant select on c_new_order  to oltp;
grant select on c_order      to oltp;
grant select on c_order_line to oltp;
grant select on c_stock      to oltp;
grant select on c_warehouse  to oltp;

begin
  DBMS_STATS.GATHER_SCHEMA_STATS
  (
    OwnName             => 'HOLDEM'
    ,Estimate_Percent   => SYS.DBMS_STATS.AUTO_SAMPLE_SIZE
    ,Block_sample       => TRUE
    ,Method_Opt         => 'FOR ALL INDEXED COLUMNS SIZE AUTO '
    ,Degree             => 8
    ,Cascade            => TRUE
    ,No_Invalidate      => TRUE
  );
end;
/

exec dbms_stats.gather_dictionary_stats;
exec dbms_stats.gather_fixed_objects_stats;

--- filename - tpcc_reset.sql

connect oltp/scott

drop table c_customer   cascade constraints purge;
drop table c_district   cascade constraints purge;
drop table c_history    cascade constraints purge;
drop table c_item       cascade constraints purge;
drop table c_new_order  cascade constraints purge;
drop table c_order      cascade constraints purge;
drop table c_order_line cascade constraints purge;
drop table c_stock      cascade constraints purge;
drop table c_warehouse  cascade constraints purge;

create table c_customer   tablespace usrs parallel (degree  9) as select *
from holdem.c_customer;
```

```
create table c_district   tablespace usrs parallel (degree  3) as select *
from holdem.c_district;
create table c_history    tablespace usrs parallel (degree  9) as select *
from holdem.c_history;
create table c_item       tablespace usrs parallel (degree  3) as select *
from holdem.c_item;
create table c_new_order  tablespace usrs parallel (degree  6) as select *
from holdem.c_new_order;
create table c_order      tablespace usrs parallel (degree  9) as select *
from holdem.c_order;
create table c_order_line tablespace usrs parallel (degree 12) as select *
from holdem.c_order_line;
create table c_stock      tablespace usrs parallel (degree 12) as select *
from holdem.c_stock;
create table c_warehouse  tablespace usrs parallel (degree  3) as select *
from holdem.c_warehouse;

alter table c_customer   NOPARALLEL;
alter table c_district   NOPARALLEL;
alter table c_history    NOPARALLEL;
alter table c_item       NOPARALLEL;
alter table c_new_order  NOPARALLEL;
alter table c_order      NOPARALLEL;
alter table c_order_line NOPARALLEL;
alter table c_stock      NOPARALLEL;
alter table c_warehouse  NOPARALLEL;

CREATE UNIQUE INDEX C_CUSTOMER_I1 ON OLTP.C_CUSTOMER
(C_W_ID, C_D_ID, C_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE INDEX C_CUSTOMER_I2 ON OLTP.C_CUSTOMER
(C_LAST, C_W_ID, C_D_ID, C_FIRST)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_DISTRICT_I1 ON OLTP.C_DISTRICT
(D_W_ID, D_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_ITEM_I1 ON OLTP.C_ITEM
(I_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_NEW_ORDER_I1 ON OLTP.C_NEW_ORDER
```

```
(NO_W_ID, NO_D_ID, NO_O_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_ORDER_I1 ON OLTP.C_ORDER
(O_ID, O_W_ID, O_D_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_ORDER_LINE_I1 ON OLTP.C_ORDER_LINE
(OL_O_ID, OL_W_ID, OL_D_ID, OL_NUMBER)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_STOCK_I1 ON OLTP.C_STOCK
(S_I_ID, S_W_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

CREATE UNIQUE INDEX C_WAREHOUSE_I1 ON OLTP.C_WAREHOUSE
(W_ID)
NOLOGGING
TABLESPACE INDX
PARALLEL (degree 4);

/* Formatted on 2007/02/08 14:25 (Formatter Plus v4.8.8) */
CREATE OR REPLACE PROCEDURE c_sp_delivery (
   ware_id               NUMBER,
   carrier_id            NUMBER,
   order_1      IN OUT   NUMBER,
   order_2      IN OUT   NUMBER,
   order_3      IN OUT   NUMBER,
   order_4      IN OUT   NUMBER,
   order_5      IN OUT   NUMBER,
   order_6      IN OUT   NUMBER,
   order_7      IN OUT   NUMBER,
   order_8      IN OUT   NUMBER,
   order_9      IN OUT   NUMBER,
   order_10     IN OUT   NUMBER,
   retry        IN OUT   NUMBER,
   cur_date     IN       DATE
)
AS
   TYPE intarray IS TABLE OF INTEGER
      INDEX BY BINARY_INTEGER;

   order_id          intarray;
   dist_id           INTEGER;
```

```
   cust_id            INTEGER;
   amount_sum         NUMBER;
   no_rowid           ROWID;
   not_serializable   EXCEPTION;
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);

   CURSOR o_cur
   IS
      SELECT   no_o_id, ROWID
          FROM c_new_order
         WHERE no_w_id = ware_id AND no_d_id = dist_id
      ORDER BY no_w_id, no_d_id, no_o_id;
BEGIN
   FOR i IN 1 .. 10
   LOOP
      dist_id := i;

      LOOP
         BEGIN
            OPEN o_cur;

            FETCH o_cur
             INTO order_id (i), no_rowid;

            IF (o_cur%NOTFOUND)
            THEN
               CLOSE o_cur;

               COMMIT;
               order_id (i) := 0;
               EXIT;
            END IF;

            CLOSE o_cur;

            DELETE FROM c_new_order
                  WHERE ROWID = no_rowid;

            UPDATE c_order
               SET o_carrier_id = carrier_id
             WHERE o_d_id = dist_id AND o_w_id = ware_id
                   AND o_id = order_id (i);

            SELECT o_c_id
              INTO cust_id
              FROM c_order
             WHERE o_d_id = dist_id AND o_w_id = ware_id
                   AND o_id = order_id (i);
```

```
            UPDATE c_order_line
               SET ol_delivery_d = cur_date
             WHERE ol_d_id = dist_id
               AND ol_w_id = ware_id
               AND ol_o_id = order_id (i);

            SELECT SUM (ol_amount)
              INTO amount_sum
              FROM c_order_line
             WHERE ol_d_id = dist_id
               AND ol_w_id = ware_id
               AND ol_o_id = order_id (i);

            UPDATE c_customer
               SET c_balance = c_balance + amount_sum,
                   c_delivery_cnt = c_delivery_cnt + 1
             WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

            COMMIT;
            EXIT;
         EXCEPTION
            WHEN not_serializable OR deadlock OR snapshot_too_old
            THEN
               ROLLBACK;
               retry := retry + 1;
         END;
      END LOOP;
   END LOOP;

   order_1 := order_id (1);
   order_2 := order_id (2);
   order_3 := order_id (3);
   order_4 := order_id (4);
   order_5 := order_id (5);
   order_6 := order_id (6);
   order_7 := order_id (7);
   order_8 := order_id (8);
   order_9 := order_id (9);
   order_10 := order_id (10);
END;
/

CREATE OR REPLACE PROCEDURE c_sp_new_order (
   ware_id                 NUMBER,
   dist_id                 NUMBER,
   cust_id                 NUMBER,
   ord_ol_cnt              NUMBER,
   ord_all_local           NUMBER,
   cust_discount   OUT     NUMBER,
   cust_last       OUT     VARCHAR2,
   cust_credit     OUT     VARCHAR2,
```

```
   dist_tax        OUT      NUMBER,
   ware_tax        OUT      NUMBER,
   ord_id          IN OUT   NUMBER,
   retry           IN OUT   NUMBER,
   cur_date        IN       DATE
)
AS
   dist_rowid         ROWID;
   not_serializable   EXCEPTION;
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);
   integrity_viol     EXCEPTION;
   PRAGMA EXCEPTION_INIT (integrity_viol, -1);
BEGIN
   LOOP
      BEGIN
         SELECT c_district.ROWID, d_tax, d_next_o_id, w_tax
           INTO dist_rowid, dist_tax, ord_id, ware_tax
           FROM c_district, c_warehouse
          WHERE d_id = dist_id AND d_w_id = w_id AND w_id = ware_id;

         UPDATE c_district
            SET d_next_o_id = ord_id + 1
          WHERE ROWID = dist_rowid;

         SELECT c_discount, c_last, c_credit
           INTO cust_discount, cust_last, cust_credit
           FROM c_customer
          WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

         INSERT INTO c_new_order
              VALUES (ord_id, dist_id, ware_id);

         INSERT INTO c_order
              VALUES (ord_id, dist_id, ware_id, cust_id, cur_date, 11,
                      ord_ol_cnt, ord_all_local);

         EXIT;
      EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old OR
integrity_viol
         THEN
            ROLLBACK;
            retry := retry + 1;
      END;
   END LOOP;
END;
/
```

```
CREATE OR REPLACE PROCEDURE c_sp_order_status_id (
   ware_id                   NUMBER,
   dist_id                   NUMBER,
   cust_id          IN OUT   NUMBER,
   bylastname                NUMBER,
   cust_last        IN OUT   VARCHAR2,
   cust_first       OUT      VARCHAR2,
   cust_middle      OUT      VARCHAR2,
   cust_balance     OUT      NUMBER,
   ord_id           IN OUT   NUMBER,
   ord_entry_d      OUT      DATE,
   ord_carrier_id   OUT      NUMBER,
   ord_ol_cnt       OUT      NUMBER
)
IS
   TYPE rowidarray IS TABLE OF ROWID
      INDEX BY BINARY_INTEGER;

   cust_rowid         ROWID;
   ol                 BINARY_INTEGER;
   c_num              BINARY_INTEGER;
   row_id             rowidarray;
   not_serializable   EXCEPTION;
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);

   CURSOR mo_cur
   IS
      SELECT   o_id, o_entry_d, o_carrier_id, o_ol_cnt
          FROM c_order
         WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
      ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC;
BEGIN
   LOOP
      BEGIN
         SELECT c_balance, c_first, c_middle, c_last
           INTO cust_balance, cust_first, cust_middle, cust_last
           FROM c_customer
          WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

         OPEN mo_cur;

         FETCH mo_cur
          INTO ord_id, ord_entry_d, ord_carrier_id, ord_ol_cnt;

         CLOSE mo_cur;

         EXIT;
      EXCEPTION
```

```
         WHEN not_serializable OR deadlock OR snapshot_too_old
         THEN
            ROLLBACK;
      END;
   END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE c_sp_order_status_name (
   ware_id                   NUMBER,
   dist_id                   NUMBER,
   cust_id          IN OUT   NUMBER,
   bylastname                NUMBER,
   cust_last        IN OUT   VARCHAR2,
   cust_first       OUT      VARCHAR2,
   cust_middle      OUT      VARCHAR2,
   cust_balance     OUT      NUMBER,
   ord_id           IN OUT   NUMBER,
   ord_entry_d      OUT      DATE,
   ord_carrier_id   OUT      NUMBER,
   ord_ol_cnt       OUT      NUMBER
)
IS
   TYPE rowidarray IS TABLE OF ROWID
      INDEX BY BINARY_INTEGER;

   cust_rowid         ROWID;
   ol                 BINARY_INTEGER;
   c_num              BINARY_INTEGER;
   row_id             rowidarray;
   not_serializable   EXCEPTION;
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);

   CURSOR c_cur
   IS
      SELECT   ROWID
          FROM c_customer
         WHERE c_d_id = dist_id AND c_w_id = ware_id AND c_last = cust_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;

   CURSOR mo_cur
   IS
      SELECT   o_id, o_entry_d, o_carrier_id, o_ol_cnt
          FROM c_order
         WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
      ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC;
BEGIN
   LOOP
```

```
      BEGIN
         c_num := 0;

         FOR c_id_rec IN c_cur
         LOOP
            c_num := c_num + 1;
            row_id (c_num) := c_id_rec.ROWID;
         END LOOP;

         cust_rowid := row_id ((c_num + 1) / 2);

         SELECT c_id, c_balance, c_first, c_middle, c_last
           INTO cust_id, cust_balance, cust_first, cust_middle, cust_last
           FROM c_customer
          WHERE ROWID = cust_rowid;

         OPEN mo_cur;

         FETCH mo_cur
          INTO ord_id, ord_entry_d, ord_carrier_id, ord_ol_cnt;

         CLOSE mo_cur;

         EXIT;
      EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old
         THEN
            ROLLBACK;
      END;
   END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE c_sp_payment_id (
   ware_id                    NUMBER,
   dist_id                    NUMBER,
   cust_w_id                  NUMBER,
   cust_d_id                  NUMBER,
   cust_id           IN OUT   NUMBER,
   bylastname                 NUMBER,
   hist_amount                NUMBER,
   cust_last         IN OUT   VARCHAR2,
   ware_street_1     OUT      VARCHAR2,
   ware_street_2     OUT      VARCHAR2,
   ware_city         OUT      VARCHAR2,
   ware_state        OUT      VARCHAR2,
   ware_zip          OUT      VARCHAR2,
   dist_street_1     OUT      VARCHAR2,
   dist_street_2     OUT      VARCHAR2,
   dist_city         OUT      VARCHAR2,
   dist_state        OUT      VARCHAR2,
   dist_zip          OUT      VARCHAR2,
```

```
    cust_first         OUT      VARCHAR2,
    cust_middle        OUT      VARCHAR2,
    cust_street_1      OUT      VARCHAR2,
    cust_street_2      OUT      VARCHAR2,
    cust_city          OUT      VARCHAR2,
    cust_state         OUT      VARCHAR2,
    cust_zip           OUT      VARCHAR2,
    cust_phone         OUT      VARCHAR2,
    cust_since         OUT      DATE,
    cust_credit        IN OUT   VARCHAR2,
    cust_credit_lim    OUT      NUMBER,
    cust_discount      OUT      NUMBER,
    cust_balance       IN OUT   NUMBER,
    cust_data          OUT      VARCHAR2,
    retry              IN OUT   NUMBER,
    cur_date           IN       DATE
)
AS
    TYPE rowidarray IS TABLE OF ROWID
       INDEX BY BINARY_INTEGER;

    cust_rowid          ROWID;
    ware_rowid          ROWID;
    dist_ytd            NUMBER;
    dist_name           VARCHAR2 (11);
    ware_ytd            NUMBER;
    ware_name           VARCHAR2 (11);
    c_num               BINARY_INTEGER;
    row_id              rowidarray;
    cust_payments       PLS_INTEGER;
    cust_ytd            NUMBER;
    cust_data_temp      VARCHAR2 (500);
    not_serializable    EXCEPTION;
    PRAGMA EXCEPTION_INIT (not_serializable, -8177);
    deadlock            EXCEPTION;
    PRAGMA EXCEPTION_INIT (deadlock, -60);
    snapshot_too_old    EXCEPTION;
    PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);
BEGIN
    LOOP
       BEGIN
          SELECT ROWID, c_first, c_middle, c_last,
                 c_street_1, c_street_2, c_city, c_state,
                 c_zip, c_phone, c_since, c_credit,
                 c_credit_lim, c_discount, c_balance - hist_amount,
                 c_payment_cnt, c_ytd_payment + hist_amount,
                 DECODE (c_credit, 'BC', c_data, ' ')
            INTO cust_rowid, cust_first, cust_middle, cust_last,
                 cust_street_1, cust_street_2, cust_city, cust_state,
                 cust_zip, cust_phone, cust_since, cust_credit,
                 cust_credit_lim, cust_discount, cust_balance,
                 cust_payments, cust_ytd,
```

```
                cust_data_temp
            FROM c_customer
           WHERE c_id = cust_id AND c_d_id = cust_d_id AND c_w_id =
cust_w_id;

         cust_payments := cust_payments + 1;

         IF cust_credit = 'BC'
         THEN
            cust_data_temp :=
               SUBSTR (   (   TO_CHAR (cust_id)
                           || ' '
                           || TO_CHAR (cust_d_id)
                           || ' '
                           || TO_CHAR (cust_w_id)
                           || ' '
                           || TO_CHAR (dist_id)
                           || ' '
                           || TO_CHAR (ware_id)
                           || ' '
                           || TO_CHAR (hist_amount, '9999.99')
                           || '|'
                          )
                        || cust_data_temp,
                        1,
                        500
                       );

            UPDATE c_customer
               SET c_balance = cust_balance,
                   c_ytd_payment = cust_ytd,
                   c_payment_cnt = cust_payments,
                   c_data = cust_data_temp
             WHERE ROWID = cust_rowid;

            cust_data := SUBSTR (cust_data_temp, 1, 200);
         ELSE
            UPDATE c_customer
               SET c_balance = cust_balance,
                   c_ytd_payment = cust_ytd,
                   c_payment_cnt = cust_payments
             WHERE ROWID = cust_rowid;

            cust_data := cust_data_temp;
         END IF;

         SELECT c_district.ROWID, d_name, d_street_1, d_street_2,
                d_city, d_state, d_zip, d_ytd + hist_amount,
                c_warehouse.ROWID, w_name, w_street_1, w_street_2,
                w_city, w_state, w_zip, w_ytd + hist_amount
           INTO cust_rowid, dist_name, dist_street_1, dist_street_2,
                dist_city, dist_state, dist_zip, dist_ytd,
```

```
                ware_rowid, ware_name, ware_street_1, ware_street_2,
                ware_city, ware_state, ware_zip, ware_ytd
           FROM c_district, c_warehouse
          WHERE d_id = dist_id AND d_w_id = w_id AND w_id = ware_id;

         UPDATE c_district
            SET d_ytd = dist_ytd
          WHERE ROWID = cust_rowid;

         UPDATE c_warehouse
            SET w_ytd = ware_ytd
          WHERE ROWID = ware_rowid;

         INSERT INTO c_history
              VALUES (cust_id, cust_d_id, cust_w_id, dist_id, ware_id,
                      cur_date, hist_amount, ware_name || '    ' ||
dist_name);

         COMMIT;
         EXIT;
      EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old
         THEN
            ROLLBACK;
            retry := retry + 1;
      END;
   END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE c_sp_payment_name (
   ware_id                    NUMBER,
   dist_id                    NUMBER,
   cust_w_id                  NUMBER,
   cust_d_id                  NUMBER,
   cust_id           IN OUT   NUMBER,
   bylastname                 NUMBER,
   hist_amount                NUMBER,
   cust_last         IN OUT   VARCHAR2,
   ware_street_1     OUT      VARCHAR2,
   ware_street_2     OUT      VARCHAR2,
   ware_city         OUT      VARCHAR2,
   ware_state        OUT      VARCHAR2,
   ware_zip          OUT      VARCHAR2,
   dist_street_1     OUT      VARCHAR2,
   dist_street_2     OUT      VARCHAR2,
   dist_city         OUT      VARCHAR2,
   dist_state        OUT      VARCHAR2,
   dist_zip          OUT      VARCHAR2,
   cust_first        OUT      VARCHAR2,
   cust_middle       OUT      VARCHAR2,
   cust_street_1     OUT      VARCHAR2,
```

```
   cust_street_2     OUT      VARCHAR2,
   cust_city         OUT      VARCHAR2,
   cust_state        OUT      VARCHAR2,
   cust_zip          OUT      VARCHAR2,
   cust_phone        OUT      VARCHAR2,
   cust_since        OUT      DATE,
   cust_credit       IN OUT   VARCHAR2,
   cust_credit_lim   OUT      NUMBER,
   cust_discount     OUT      NUMBER,
   cust_balance      IN OUT   NUMBER,
   cust_data         OUT      VARCHAR2,
   retry             IN OUT   NUMBER,
   cur_date          IN       DATE
)
AS
   TYPE rowidarray IS TABLE OF ROWID
      INDEX BY BINARY_INTEGER;

   cust_rowid         ROWID;
   ware_rowid         ROWID;
   dist_ytd           NUMBER;
   dist_name          VARCHAR2 (11);
   ware_ytd           NUMBER;
   ware_name          VARCHAR2 (11);
   c_num              BINARY_INTEGER;
   row_id             rowidarray;
   cust_payments      PLS_INTEGER;
   cust_ytd           NUMBER;
   cust_data_temp     VARCHAR2 (500);
   not_serializable   EXCEPTION;
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);

   CURSOR c_cur
   IS
      SELECT   ROWID
          FROM c_customer
         WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id
               AND c_last = cust_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
BEGIN
   LOOP
      BEGIN
         c_num := 0;

         FOR c_id_rec IN c_cur
         LOOP
            c_num := c_num + 1;
            row_id (c_num) := c_id_rec.ROWID;
```

```
END LOOP;

cust_rowid := row_id ((c_num + 1) / 2);

SELECT c_id, c_first, c_middle, c_last, c_street_1,
       c_street_2, c_city, c_state, c_zip, c_phone,
       c_since, c_credit, c_credit_lim, c_discount,
       c_balance - hist_amount, c_payment_cnt,
       c_ytd_payment + hist_amount,
       DECODE (c_credit, 'BC', c_data, ' ')
  INTO cust_id, cust_first, cust_middle, cust_last, cust_street_1,
       cust_street_2, cust_city, cust_state, cust_zip, cust_phone,
       cust_since, cust_credit, cust_credit_lim, cust_discount,
       cust_balance, cust_payments,
       cust_ytd,
       cust_data_temp
  FROM c_customer
 WHERE ROWID = cust_rowid;

cust_payments := cust_payments + 1;

IF cust_credit = 'BC'
THEN
   cust_data_temp :=
      SUBSTR (   (   TO_CHAR (cust_id)
                  || ' '
                  || TO_CHAR (cust_d_id)
                  || ' '
                  || TO_CHAR (cust_w_id)
                  || ' '
                  || TO_CHAR (dist_id)
                  || ' '
                  || TO_CHAR (ware_id)
                  || ' '
                  || TO_CHAR (hist_amount, '9999.99')
                  || '|'
                 )
              || cust_data_temp,
              1,
              500
             );

   UPDATE c_customer
      SET c_balance = cust_balance,
          c_ytd_payment = cust_ytd,
          c_payment_cnt = cust_payments,
          c_data = cust_data_temp
    WHERE ROWID = cust_rowid;

   cust_data := SUBSTR (cust_data_temp, 1, 200);
ELSE
   UPDATE c_customer
```

```
                SET c_balance = cust_balance,
                    c_ytd_payment = cust_ytd,
                    c_payment_cnt = cust_payments
              WHERE ROWID = cust_rowid;

            cust_data := cust_data_temp;
         END IF;

         SELECT c_district.ROWID, d_name, d_street_1, d_street_2,
                d_city, d_state, d_zip, d_ytd + hist_amount,
                c_warehouse.ROWID, w_name, w_street_1, w_street_2,
                w_city, w_state, w_zip, w_ytd + hist_amount
           INTO cust_rowid, dist_name, dist_street_1, dist_street_2,
                dist_city, dist_state, dist_zip, dist_ytd,
                ware_rowid, ware_name, ware_street_1, ware_street_2,
                ware_city, ware_state, ware_zip, ware_ytd
           FROM c_district, c_warehouse
          WHERE d_id = dist_id AND d_w_id = w_id AND w_id = ware_id;

         UPDATE c_district
            SET d_ytd = dist_ytd
          WHERE ROWID = cust_rowid;

         UPDATE c_warehouse
            SET w_ytd = ware_ytd
          WHERE ROWID = ware_rowid;

         INSERT INTO c_history
              VALUES (cust_id, cust_d_id, cust_w_id, dist_id, ware_id,
                      cur_date, hist_amount, ware_name || '    ' ||
dist_name);

         COMMIT;
         EXIT;
      EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old
         THEN
            ROLLBACK;
            retry := retry + 1;
      END;
   END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE c_sp_stock_level (
   ware_id           NUMBER,
   dist_id           NUMBER,
   threshold         NUMBER,
   low_stock   OUT   NUMBER
)
IS
   not_serializable   EXCEPTION;
```

```
   PRAGMA EXCEPTION_INIT (not_serializable, -8177);
   deadlock           EXCEPTION;
   PRAGMA EXCEPTION_INIT (deadlock, -60);
   snapshot_too_old   EXCEPTION;
   PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);
BEGIN
   LOOP
      BEGIN
         SELECT COUNT (DISTINCT s_i_id)
           INTO low_stock
           FROM c_order_line, c_stock, c_district
          WHERE d_id = dist_id
            AND d_w_id = ware_id
            AND d_id = ol_d_id
            AND d_w_id = ol_w_id
            AND ol_i_id = s_i_id
            AND ol_w_id = s_w_id
            AND s_quantity
            AND ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1);

         COMMIT;
         EXIT;
      EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old
         THEN
            ROLLBACK;
      END;
   END LOOP;
END;
/

begin
  DBMS_STATS.GATHER_SCHEMA_STATS
  (
    OwnName             => 'OLTP'
    ,Estimate_Percent   => SYS.DBMS_STATS.AUTO_SAMPLE_SIZE
    ,Block_sample       => TRUE
    ,Method_Opt         => 'FOR ALL INDEXED COLUMNS SIZE AUTO '
    ,Degree             => 9
    ,Cascade            => TRUE
    ,No_Invalidate      => TRUE
  );
end;
/

exit
```

ABOUT PRINCIPLED TECHNOLOGIES

Principled Technologies, Inc.
1007 Slater Road, Suite 300
Durham, NC, 27703
www.principledtechnologies.com

We provide industry-leading technology assessment and fact-based marketing services. We bring to every assignment extensive experience with and expertise in all aspects of technology testing and analysis, from researching new technologies, to developing new methodologies, to testing with existing and new tools.

When the assessment is complete, we know how to present the results to a broad range of target audiences. We provide our clients with the materials they need, from market-focused data to use in their own collateral to custom sales aids, such as test reports, performance assessments, and white papers. Every document reflects the results of our trusted independent analysis.

We provide customized services that focus on our clients' individual requirements. Whether the technology involves hardware, software, Web sites, or services, we offer the experience, expertise, and tools to help our clients assess how it will fare against its competition, its performance, its market readiness, and its quality and reliability.

Our founders, Mark L. Van Name and Bill Catchings, have worked together in technology assessment for over 20 years. As journalists, they published over a thousand articles on a wide array of technology subjects. They created and led the Ziff-Davis Benchmark Operation, which developed such industry-standard benchmarks as Ziff Davis Media's Winstone and WebBench. They founded and led eTesting Labs, and after the acquisition of that company by Lionbridge Technologies were the head and CTO of VeriTest.