VMware VMotion Performance on the Dell PowerEdge 1955 Blade Server

By Dave Jaffe and Todd Muirhead
Dell Enterprise Technology Center
and
Balasubramanian Chandrasekaran
Dell Virtual Solutions Engineering

Dell | Enterprise Technology Center

dell.com/TechCenter

March 2007

Contents

Executive Summary **3**
Introduction **4**
The Dell PowerEdge 1955 Blade Server **5**
Optimizing ESX Server on the Dell PowerEdge 1955 .... **8**
The Test **10**
Results **11**
Conclusions **13**
Appendix A. VMotion Script **14**

Tables

Table 1: Dell PowerEdge 1955 Blade Features 6
Table 2: Dell/EMC Storage 7
Table 3: VM Parameters 10
Table 4: Results 11

Figures

Figure 1: Dell PowerEdge 1955 Blade Chassis and 10 blades 5
Figure 2: Network Setup 7
Figure 3: ESX NIC Configuration 9
Figure 4: Effect of VMotions on VM Performance 12

Talk Back
Tell us how the Dell Scalable Enterprise Technology Center can help your organization better simplify, utilize, and scale enterprise solutions and platforms. Send your feedback and ideas to **Enterprise_Techcenter@dell.com** or visit us at **dell.com/TechCenter**.

Section 1

Executive Summary

To demonstrate the power of running VMware Infrastructure 3 virtualization software on Dell PowerEdge 1955 blade servers, including use of the VMotion live virtual machine migration facility, a test was run to quantify the impact of VMotions on virtual machine performance. It was found that even very rapid rates of VMotions, far beyond what would be expected in typical VMotion scenarios utilizing VMware's High Availability or Distributed Resource Scheduler features, have negligible impact on virtual machine application performance.

Section 2

Introduction

The Dell PowerEdge 1955 blade server, provided in a chassis supporting up to 10 blades in a 7 rack unit (12.25”) chassis, is a very dense, high-performance, cost effective platform on which to build “scaled-out” server farms. VMware’s Infrastructure 3 (VI3), with ESX Server, software which enables multiple virtual machines (VMs) to be installed on a physical server, is ideally suited for such a server farm. With its VirtualCenter management software to manage the farm of ESX Server hosts, and VMotion to move live VMs from one ESX Server host to another, VMware on PowerEdge 1955 blades provides an excellent platform for hosting virtualized server applications.

The VMotion migration of running VMs from one ESX Server host to another is especially useful in enterprise data centers. The scenarios where VMotion is utilized include dynamic load balancing of VMs across an ESX Server farm with VI3’s Distributed Resource Scheduler feature; failover of VMs off a failing ESX Server host with VI3’s High Availability feature; and movement of VMs off an ESX Server host to bring that host down for routine maintenance. In all cases it is desirable that the performance impact of the VMotion as seen by the end user running applications on the VM being moved be negligible.

Each PowerEdge 1955 blade server ships with two embedded gigabit Ethernet (GE) network interface cards (NICs), which are wired through the blade chassis to two GE switches in the back of the chassis. Additionally, each blade server supports an additional daughter card which can be used for more NICs or for other input/output (I/O) interfaces. To use VMotion on VMware the ESX servers participating in the VMotion pool must share storage. The most typical way to do this, to date, is to populate the daughter card slot with a Fibre Channel (FC) daughter card to connect the blade to an external Dell/EMC storage area network (SAN). In this configuration each PowerEdge 1955 blade server is limited to 2 GE network connections in a VMware with VMotion setup. (A new method to provide shared storage, iSCSI, will be explored in a subsequent paper).

The purpose of this study is to quantify the impact on typical VMware operations of having just two network connections available for each ESX Server host running on a set of PowerEdge 1955 blades. ESX was configured on each blade so that the ESX Service Console and the VMotion traffic use one NIC and the second NIC is used for the network traffic of the VMs running on the blades.

In this test, performed in March, 2007, which updates a previous study by the authors, “VMware VMotion Performance on the Dell PowerEdge 1855 Blade Server” (http://www.dell.com/downloads/global/solutions/vmware_pe1855.pdf) we again run a constant database workload against 20 VMs running on the two blades while we increase the rate at which the VMs are moved back and forth between the two blades. It is shown that the moves have negligible effect on performance of the VMs, even as each VM is moved once every 10 minutes, a rate much higher than any common scenario.

Section 3

The Dell PowerEdge 1955 Blade Server

The Dell PowerEdge 1955 blade server (Figure 1) provides a rack dense server solution based on industry standard Intel Xeon processors. The PowerEdge 1955 blade server resides in a blade chassis that takes up 7 rack units (12.25”) in a standard server rack and holds up to 10 blade servers. The chassis provides power, cooling, network connectivity and, optionally, Fibre Channel or additional network connectivity. This allows for a greatly reduced amount of required cabling and a much simpler deployment of new blades into an existing chassis.

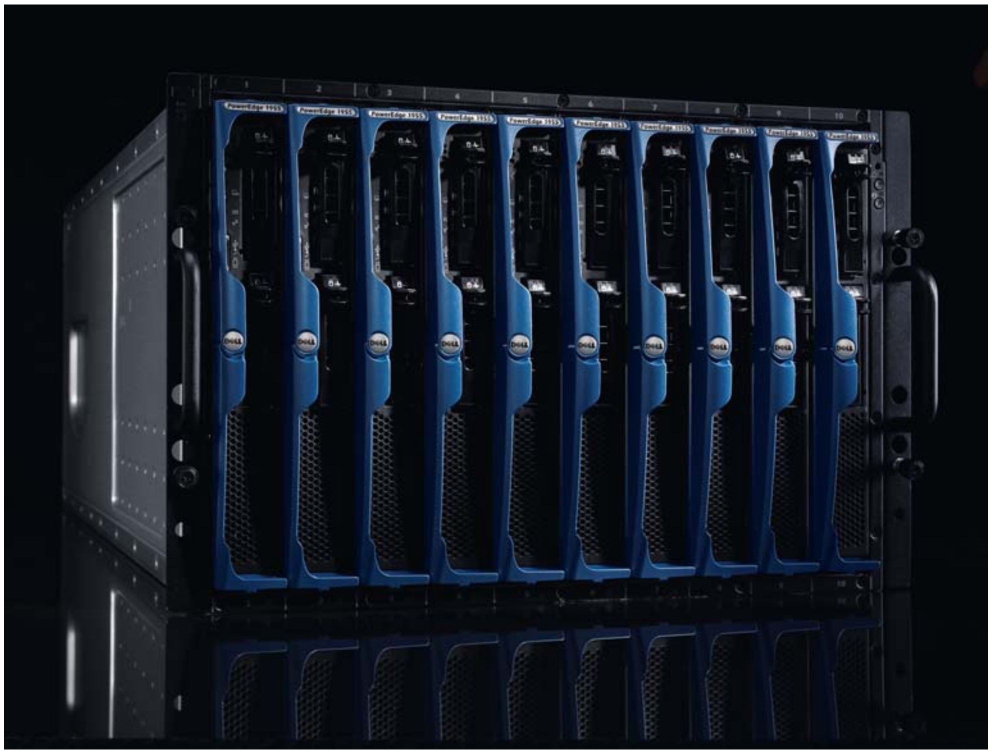


Figure 1: Dell PowerEdge 1955 Blade Chassis and 10 blades

Each individual PowerEdge 1955 blade server supports up to two Intel Xeon quadcore processors and up to 32 GB of fully buffered DIMM memory. Processors currently available for the PowerEdge 1955 range from the dualcore Intel® Xeon® 5050 running at 3.0GHz with a 667MHZ frontside bus (FSB) up to the quadcore Xeon® X5355 running at 2.66GHz with a 1333MHz FSB, which enables an individual blade to have 8 processing cores. The PowerEdge 1955 also has two gigabit Ethernet ports standard and allows for an optional daughter card that can provide two more gigabit Ethernet ports or two Fibre Channel ports. To provide the connectivity in the chassis for the daughter card-provided connections, corresponding Ethernet or Fibre Channel pass-thru modules can be installed or Ethernet or Fibre Channel switch modules can be installed.

For this test the chassis was equipped with two Dell PowerEdge 1955 blade servers, each with VMware ESX Server 3.0.1. The Dell PowerEdge 1955s were configured with two 2.66 GHz Xeon X5355 quadcore processors with 2x4 MB L2 cache, 8GB of fully buffered DIMM memory, and two internal SAS 73GB 10K rpm disks. The detailed system configuration of each blade is shown in

Table 1.

Operating System	ESX 3.0.1 w/ patch ESX-2066306
CPU	Two quad-core Intel Xeon X5355 processors at 2.66 GHz with 2x4MB cache
Memory	8 GB (4x 667 MHz fully buffered 2GB DIMMs)
Internal Disks	2x 73 GB 10K RPM SAS
NICs	2x 10/100/1000[1] Mb/s (Internal)
Disk Controller	Embedded SAS5/ir Integrated Controller
Fiber Channel Daughter Card	QLA 2362

Table 1: Dell PowerEdge 1955 Blade Features

To provide shared storage needed for the live VMotions the two PowerEdge 1955 server blades were connected to a storage area network (SAN) based on a Dell/EMC CX3-80 storage controller with redundant service processors (SPA and SPB), connected through redundant Fibre Channel switches. Each blade was configured with the Dell QLogic 2362 daughter card. The chassis was installed with two McData 4314 Fibre Channel switches which are connected through Inter Switch Links to external Fibre Channel switches, which connect to the CX3-80.

Four 5-disk RAID 5 LUNs were created on the CX3-80 to be used for the VMs. The VMs used were evenly spread across the four LUNs with five on each LUN. All LUNs and the two blades were assigned to the same storage group allowing for the storage to be shared by both the blades. The storage components used for both servers in the test are shown in

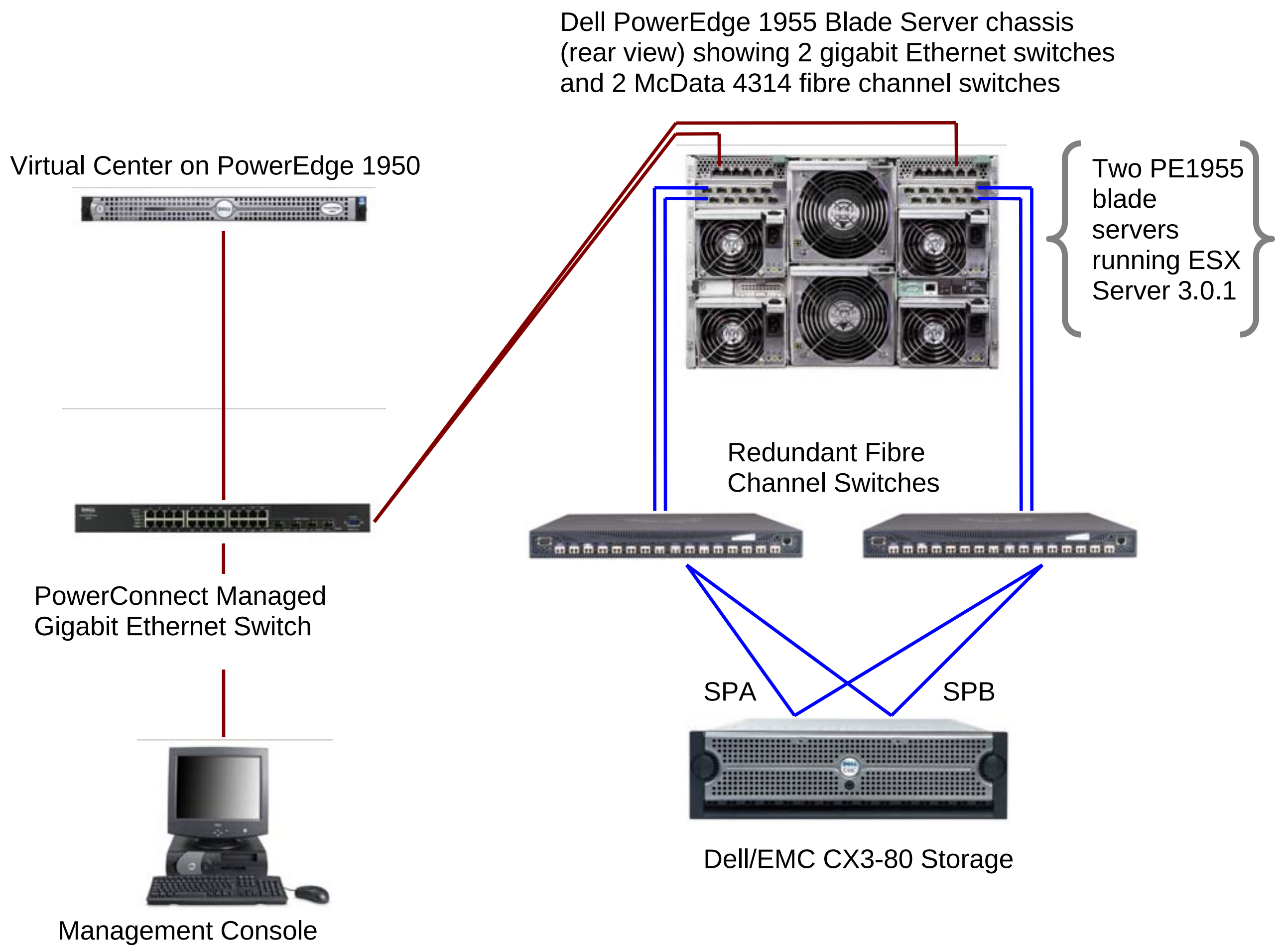
Table 2.

The entire network setup, including the blade chassis, the Fibre Channel SAN, Virtual Center running on a central server, and a management console, is shown in Figure 2.

Controller	1 Dell/EMC CX3-80
Disk Enclosures	2 Dell/EMC DAE4P
Disks	21 x 136 GB/ 15K RPM
LUNs	4 5-Disk RAID 5 1 HotSpare Disk
Software	Navisphere® Manager Access Logix™

Table 2: Dell/EMC Storage

.



Dell PowerEdge 1955 Blade Server chassis (rear view) showing 2 gigabit Ethernet switches and 2 McData 4314 fibre channel switches
Virtual Center on PowerEdge 1950
Two PE1955 blade servers running ESX Server 3.0.1
Redundant Fibre Channel Switches
PowerConnect Managed Gigabit Ethernet Switch
SPA
SPB
Dell/EMC CX3-80 Storage
Management Console

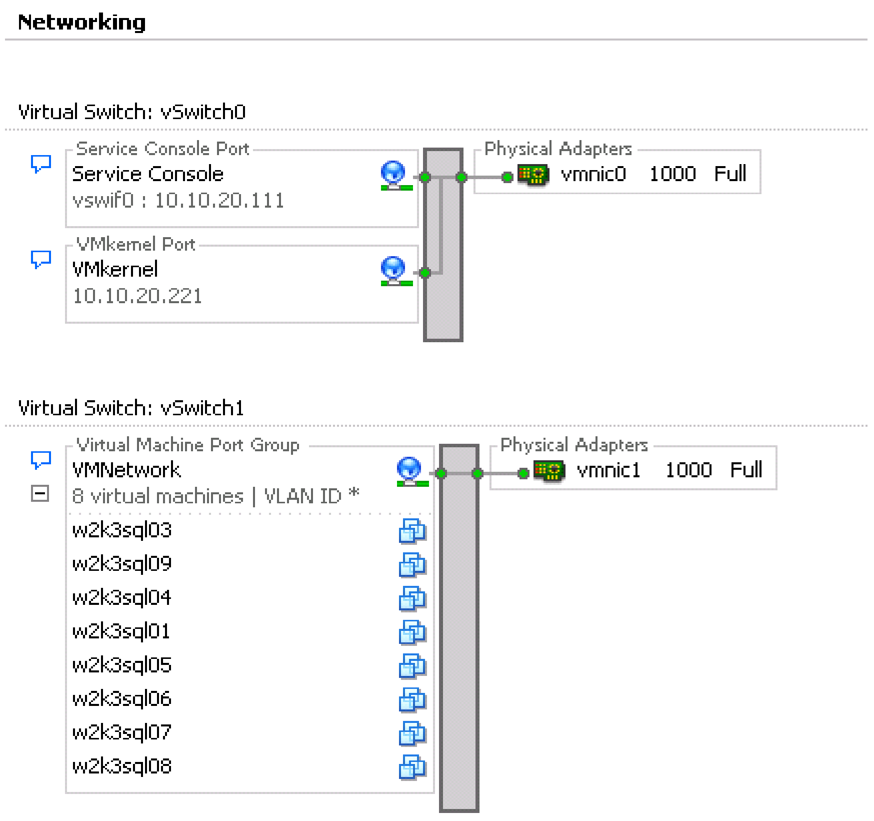
Figure 2: Network Setup

Section 4

Optimizing ESX Server on the Dell PowerEdge 1955

To enable VMotion the virtual machine's virtual disk must reside in storage that is accessible from all the ESX Server hosts which will potentially host that VM. To connect to a Dell/EMC Fibre Channel storage area network for this purpose, each PowerEdge 1955 blade server must be configured with a dual port Fibre Channel daughter card, in addition to the two embedded gigabit Ethernet network interface cards (NIC0 and NIC1). Three services have to be spread over the two NICs: the Service Console, the virtual machine network (VMNetwork) and the VMkernel, which hosts VMotions. In the configuration used in this test, The Service Console and VMkernel run on NIC0 (labeled vmnic0 by ESX) and the VMNetwork runs on NIC1 (vmnic1). The configuration is shown in Figure 3.

VMotion is performed by copying the memory state of the virtual machine from the source physical server to the destination server, through the network fabric. VMotion is a network I/O intensive operation that involves bursty network traffic (in the order of several megabytes to a few gigabytes) for a short duration of time on NIC0. The actual throughput would depend on the size of the allocated memory for the virtual machine. In the following section, we see how the VMotion traffic affects the network traffic of the other VMs.



Networking

Virtual Switch: vSwitch0

Service Console Port
Service Console
vswif0 : 10.10.20.111

VMkernel Port
VMkernel
10.10.20.221

Physical Adapters
vmnic0 1000 Full

Virtual Switch: vSwitch1

Virtual Machine Port Group
VMNetwork
8 virtual machines | VLAN ID *
w2k3sql03
w2k3sql09
w2k3sql04
w2k3sql01
w2k3sql05
w2k3sql06
w2k3sql07
w2k3sql08

Physical Adapters
vmnic1 1000 Full

Figure 3: ESX NIC Configuration

Section 5

The Test

The test used a SQL Server 2005 implementation of Dell's online DVD Store database test application, DS2. On each virtual machine a 1GB database, representing an online DVD store with 100,000 DVD titles, was built. The twenty databases were driven by separate instances of a C# program simulating users logging in to the online store, browsing for DVDs by title, author or category, and finally submitting an order. The driver program measures the number of orders per minute that the database can handle as well as the total response time as seen by the simulated end users.

The complete DVD Store application code is freely available for public use under the GPL license at http://www.linux.dell.com/dvdstore.

Each of the twenty VMs was cloned from the same golden master VM using a VMware Virtual Infrastructure SDK 2.0 script (available at http://www.dell.com/downloads/global/solutions/clone.zip). The parameters of each VM are shown in Table 3.

Parameter	Description
Memory	512 MB
Hard Disk	10 GB
NIC1	VMNetwork
Virtual CPUs	1
Operating System	Microsoft 2003 Server Enterprise Edition R2

Table 3: VM Parameters

There were two 100 minute parts to the test. In the first run the database workload was driven against all 20 VMs, with 10 VMs on each ESX Server, and no VMotions occurring. Then the same workload was applied for the same length of time, but with VMs being moved from one blade to the other in increasing frequency. In the first 10 minutes 2 VMotions occur, with one VM going from blade1 to blade2, and a different VM being moved from blade 2 to blade 1. In the second 10 minutes, 2 VMotions occur in each direction, and so on until the final 10 minutes in which 10 moves occur in each direction. In other words at the end of the test, each VM is moving on average once every 10 minutes. The VMware script to achieve the VMotions is included in Appendix A.

The two runs – with and without moves occurring – were compared to measure the effect of the moves on VM performance. The results are shown in the next section.

Section 6

Results

As described in the previous section, the performance of 20 VMs running on ESX Server on two PowerEdge 1955 server blades was measured with and without VMotions occurring. In the first test (second column in Table 4 and blue line in Figure 4) the SQL Server 2005 database on each of the 20 VMs was driven with about 4000 orders per minute for a total average of almost 80,000 orders per minute (opm). This workload was selected to drive each ESX Server host to about 60-65% CPU utilization, a loading typical of enterprise virtualization hosts. Then the same workload was applied while VMs were moved randomly between the two ESX Server hosts, with the number of moves increasing as shown in Table 4. To keep a rough balance of work on each server, each move from blade1 to blade2 was followed by a move from blade2 to blade1.

Time Interval	Average Orders Per Minute in Interval (No VMotions)	Average Orders Per Minute in Interval (With VMotions)	Number of VMotions in Interval
First 10 Minutes	76009	76716	2
Second 10 Minutes	76744	76932	4
Third 10 Minutes	77295	77054	6
Fourth10 Minutes	77414	77021	8
Fifth 10 Minutes	77432	76929	10
Sixth 10 Minutes	77485	76876	12
Seventh10 Minutes	77460	76874	14
Eighth 10 Minutes	77444	76840	16
Ninth 10 Minutes	77421	76758	18
Tenth 10 Minutes	77329	76594	20

Table 4: Results

The results of the second test are shown in the 3rd column of Table 4 and as the red line in Figure 4. After some initial noise the order rate with VMotions is slightly less than the order rate without VMotions and then decreases a tiny bit as the number of VMotions is increased. Even in the final 10 minutes, during which all 20 VMs move, the order rate drops less than 1% vs the same time period without moves. Figure 4 also shows how long each VMotion event (shown as yellow diamonds) takes. The time taken to move these VMs, even under heavy load, stays mainly in the 20 – 30 second range throughout the test.

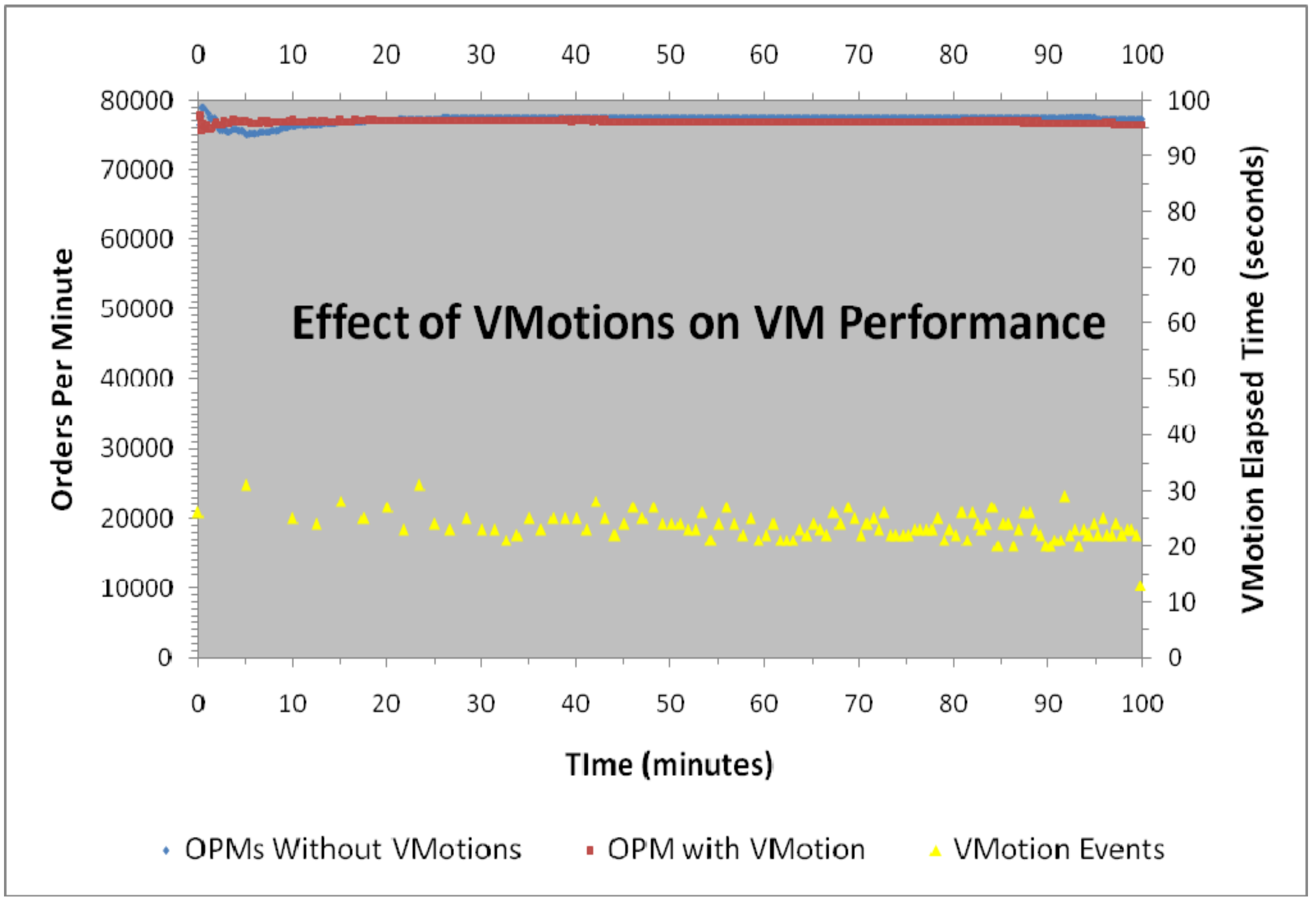


Figure 4: Effect of VMotions on VM Performance

Section 7

Conclusions

PowerEdge 1955 blade servers provide an excellent platform for VMware Infrastructure 3 server farms. While using only two Gigabit Ethernet NICs, one for VM data traffic and one for VMotion and ESX Server console traffic, the test documented in this paper showed that a high level of application load on the VMs can be maintained while also performing a very high rate of VMotions. Specifically, the VMs running on two PowerEdge 1955 blade servers are able to maintain a total rate of almost 80,000 orders per minute or about 40 billion per year, while each of the 20 VMs were moved within a 10 minute period.

Appendix A. VMotion Script

```
/*
 * vmotiontest.java - - VMware VMotion Program - move vms between 2 servers with
 *   increasing frequency
 *
 * Includes code from VMware SDK 2.0 where noted
 *
 * Author: Dave Jaffe
 * Last modified: 2/7/07
 * Copyright (c) Dell, Inc. 2007
 *
 * syntax vmotiontest host1 host2
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated
 * documentation files (the "Software"), to deal in the Software without restriction,
including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software,
 * and to permit persons to whom the Software is furnished to do so, subject to the
following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all copies
or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED
 * TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF
 * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE
OR THE USE OR OTHER DEALINGS
 * IN THE SOFTWARE.
 *
 */

import java.net.URL;
import java.util.*;
import javax.xml.rpc.Stub;

// the VirtualInfrastructureManagement Client side stubs
import com.vmware.vim.*;

public class vmotiontest {
   // code from VMware SDK 2.0
   protected VimServiceLocator _locator;
   protected VimPortType _service;
   protected int _svcState;
   protected ServiceContent _sic;
   protected ManagedObjectReference _svcRef;

   protected ManagedObjectReference _propCol;
   protected ManagedObjectReference _rootFolder;

   /**
    *  Create the Managed Object Reference for the service
    *
    *  @param svcRefVal to define service reference for HostAgent or VPX
    */
   public void createServiceRef() throws Exception {  // code from VMware SDK 2.0
      _svcRef = new ManagedObjectReference();
      _svcRef.setType("ServiceInstance");

      // could be ServiceInstance for "HostAgent" and "VPX" for VPXd
      _svcRef.set_value("ServiceInstance");
   }
   public void connect(String urlStr, String username, String password) throws Exception
{
      if (_service != null) {   // code from VMware SDK 2.0
         disconnect();
      }

      _locator = new VimServiceLocator();
      _locator.setMaintainSession(true);
      _service = _locator.getVimPort(new URL(urlStr));

      _sic = _service.retrieveServiceContent(_svcRef);
      _propCol = _sic.getPropertyCollector();
      _rootFolder = _sic.getRootFolder();

      if (_sic.getSessionManager() != null) {
         _service.login(_sic.getSessionManager(), username, password, null);
      }
   }

   /**
    *  Log's out and Disconnects from the WebService
    */
   public void disconnect()throws Exception {        // code from VMware SDK 2.0
      if (_service != null) {
         // does not work at this time
         //_service.logout(_svcRef);

         _service = null;
         _sic = null;
      }
   }

  public static void main(String[] args)
    {
    if (args == null || args.length != 2)
      {
      System.out.println( "Usage: vmotiontest host1 host2");
      return;
      }

    int i, j, move_ind;
    String hostname1 = args[0];
    String hostname2 = args[1];
    String[] host1vmarray = null;
    String[] host2vmarray = null;
    Boolean[] already_moved1 = new Boolean[10];
    Boolean[] already_moved2 = new Boolean[10];

    Random generator = new Random(System.currentTimeMillis());

    vmotiontest vmt = new vmotiontest();

    try
      {
      // Create the Service Managed Object Reference
      vmt.createServiceRef();

      // Connect to the Service
      vmt.connect("http://localhost/sdk", "vcadmin", "password");

      // Do 10 10-minute intervals, increasing the number of move pairs by 1 each time
      for (int n_move_pairs=1; n_move_pairs<11; n_move_pairs++)
        {
        // Stage n_move_pairs moves over 10 min
        int move_interval = (new Double(Math.floor(600000/(2*n_move_pairs)))).intValue();
        System.out.println("Interval " + n_move_pairs + ": moving " + 2*n_move_pairs + "
VMs at " +
          move_interval + " msec intervals");

        // Get the vms on host 1 - give last vmove from previous interval time to finish
        do
          {
          host1vmarray = vmt.list_vms_on_host(hostname1);
          } while (host1vmarray.length < 10);
        System.out.println(host1vmarray.length + " vms found on host1:");
        for (i=0; i<host1vmarray.length; i++) System.out.println(host1vmarray[i]);

        // Get the vms on host 2 - give last vmove from previous interval time to finish
        do
          {
          host2vmarray = vmt.list_vms_on_host(hostname2);
          } while (host2vmarray.length < 10);
        System.out.println(host2vmarray.length + " vms found on host2:");
        for (i=0; i<host2vmarray.length; i++) System.out.println(host2vmarray[i]);

        // Move n_move_pairs of vms
        // Don't pick the same one twice in each interval

        for (j=0; j<10; j++) {already_moved1[j] = false; already_moved2[j] = false;}

        for (i=0; i<n_move_pairs; i++)
          {
          do {move_ind = generator.nextInt(10);} while (already_moved1[move_ind]);
          System.out.println("Move  " + host1vmarray[move_ind] + " to " + hostname2);
          already_moved1[move_ind] = Boolean.TRUE;
          vmt.movevm(host1vmarray[move_ind], hostname2);
          System.out.println("Sleep " + move_interval);
          Thread.sleep(move_interval);

          do {move_ind = generator.nextInt(10);} while (already_moved2[move_ind]);
          System.out.println("Move  " + host2vmarray[move_ind] + " to " + hostname1);
          already_moved2[move_ind] = Boolean.TRUE;
          vmt.movevm(host2vmarray[move_ind], hostname1);
          System.out.println("Sleep " + move_interval);
          Thread.sleep(move_interval);
          }  // End for (i=0; i<n_move_pairs; i++)
        } // End for (int n_move_pairs=1; n_move_pairs<11; n_move_pairs++)


      // Disconnect from the Service
      vmt.disconnect();
      }
    catch (Exception e)
      {
      System.out.println("Caught Exception: " + " Name: " + e.getClass().getName() +
                         " Message: " + e.getMessage() + " Trace: ");
      e.printStackTrace();
      }
    }

  public String[] list_vms_on_host(String host_name) throws Exception
    {
    ObjectContent[] ocary, ocary2 = null;
    ObjectContent oc = null, oc2 = null;
    DynamicProperty dp = null;
    ManagedObjectReference mor = null, hsmor = null, vmmor = null;
    ArrayOfManagedObjectReference amor;

    ManagedObjectReference crmor =
      _service.findByInventoryPath(_sic.getSearchIndex(), "/TechCenter/host/" +
host_name);
    //System.out.println("list_vms_on_host: crmor: Type= " + crmor.getType() + " Val= " +
crmor.get_value());
    ManagedObjectReference pCollector = _sic.getPropertyCollector();

    PropertySpec pSpec = new PropertySpec();
    pSpec.setType("ComputeResource");
    pSpec.setPathSet(new String[] {"host"});
    ObjectSpec oSpec = new ObjectSpec();
    oSpec.setObj(crmor);
    oSpec.setSkip(Boolean.FALSE);
    PropertyFilterSpec pfSpec = new PropertyFilterSpec();
    pfSpec.setObjectSet(new ObjectSpec[] {oSpec});
    pfSpec.setPropSet(new PropertySpec[] {pSpec});
    ocary = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec });
    oc = ocary[0];
    dp = (oc.getPropSet())[0];
    //System.out.println("    Property Name: " + dp.getName());
    //System.out.println("    Property Value: " + dp.getVal());
    amor = (ArrayOfManagedObjectReference) dp.getVal();
    hsmor = amor.getManagedObjectReference(0);
    //System.out.println("      Object Type: " + hsmor.getType());
    //System.out.println("      Reference Value: " + hsmor.get_value());
    oSpec.setObj(hsmor);
    pfSpec.setObjectSet(new ObjectSpec[] {oSpec});
    pSpec.setType("HostSystem");
    pSpec.setPathSet(new String[] {"vm"});
    pfSpec.setPropSet(new PropertySpec[] {pSpec});
    ocary = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec });
    oc = ocary[0];
    dp = oc.getPropSet()[0];
    //System.out.println("    Property Name: " + dp.getName());
    //System.out.println("    Property Value: " + dp.getVal());
    amor = (ArrayOfManagedObjectReference) dp.getVal();
    String [] vmarray = new String[amor.getManagedObjectReference().length];
    //System.out.println("VMs on host " + host_name + ":");
    for (int ai = 0; ai < amor.getManagedObjectReference().length; ai++)
      {
      vmmor = amor.getManagedObjectReference(ai);
      //System.out.println("      Object Type: " + vmmor.getType());
      //System.out.println("      Reference Value: " + vmmor.get_value());
      oSpec.setObj(vmmor);
      pfSpec.setObjectSet(new ObjectSpec[] {oSpec});
      pSpec.setType("VirtualMachine");
      pSpec.setPathSet(new String[] {"name"});
      pfSpec.setPropSet(new PropertySpec[] {pSpec});
      ocary2 = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec
});
      oc2 = ocary2[0];
      //System.out.println("  " + (oc2.getPropSet())[0].getVal());
      vmarray[ai] = (String) (oc2.getPropSet())[0].getVal();
      }
    return vmarray;
    }

  public void movevm(String vm_name, String target_hostname) throws Exception
    {
    ObjectSpec oSpec = null;
    PropertySpec pSpec = null;
    PropertyFilterSpec pfSpec = null;
    ObjectContent[] ocary = null;
    ObjectContent oc = null;
    DynamicProperty dp = null;
    ManagedObjectReference pCollector= null, mor = null, hsmor = null, vmmor = null,
crmor = null, rpmor;

    pCollector = _sic.getPropertyCollector();

    // Find VirtualMachine MOR associated with vm to be moved, use to determine
PowerState of VM
    vmmor = _service.findByInventoryPath(_sic.getSearchIndex(), "/TechCenter/vm/" +
vm_name);
    //System.out.println("vmmor: Object Type: " + vmmor.getType() + " Reference Value= "
+ vmmor.get_value());
    pSpec = new PropertySpec();
    pSpec.setType("VirtualMachine");
    pSpec.setPathSet(new String[] {"runtime.powerState"});
    oSpec = new ObjectSpec();
    oSpec.setObj(vmmor);
    oSpec.setSkip(Boolean.FALSE);
    pfSpec = new PropertyFilterSpec();
    pfSpec.setObjectSet(new ObjectSpec[] {oSpec});
    pfSpec.setPropSet(new PropertySpec[] {pSpec});
    ocary = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec });
    oc = ocary[0];
    dp = (oc.getPropSet())[0];
    VirtualMachinePowerState vmps = (VirtualMachinePowerState) dp.getVal();
    //System.out.println("    Property Name: " + dp.getName() + " Property Value: " +
dp.getVal());

    // Find ComputeResource associated with target host, use it to find HostSystem MOR
and ResourcePool MOR of host
    crmor = _service.findByInventoryPath(_sic.getSearchIndex(), "/TechCenter/host/" +
target_hostname);
    //System.out.println("crmor: Object Type: " + crmor.getType() + " Reference Value= "
+ crmor.get_value());
    pSpec = new PropertySpec();
    pSpec.setType("ComputeResource");
    pSpec.setPathSet(new String[] {"host"});
    oSpec = new ObjectSpec();
    oSpec.setObj(crmor);
    oSpec.setSkip(Boolean.FALSE);
    pfSpec = new PropertyFilterSpec();
    pfSpec.setObjectSet(new ObjectSpec[] {oSpec});
    pfSpec.setPropSet(new PropertySpec[] {pSpec});
    ocary = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec });
    oc = ocary[0];
    dp = (oc.getPropSet())[0];
    //System.out.println("    Property Name: " + dp.getName());
    //System.out.println("    Property Value: " + dp.getVal());
    ArrayOfManagedObjectReference amor = (ArrayOfManagedObjectReference) dp.getVal();
    hsmor = amor.getManagedObjectReference(0);
    //System.out.println("hsmor: Object Type: " + hsmor.getType() + " Reference Value: "
+ hsmor.get_value());

    pSpec.setPathSet(new String[] {"resourcePool"});
    pfSpec.setPropSet(new PropertySpec[] {pSpec});
    ocary = _service.retrieveProperties(pCollector, new PropertyFilterSpec[] { pfSpec });
    oc = ocary[0];
    dp = (oc.getPropSet())[0];
    //System.out.println("    Property Name: " + dp.getName());
    //System.out.println("    Property Value: " + dp.getVal());
    rpmor = (ManagedObjectReference) dp.getVal();
    //System.out.println("rpmor: Object Type: " + rpmor.getType() + " Reference Value: "
+ rpmor.get_value());

    //System.out.println("Moving vm " + vm_name + " to host " + target_hostname);
    ManagedObjectReference taskRef = _service.migrateVM_Task(vmmor, rpmor, hsmor,
VirtualMachineMovePriority.highPriority,
      vmps);
    }
  }
```