Implementing Fault Tolerance
Through VMware Scripting and Dell OpenManage

The VMware® Virtual Infrastructure™ SDK package enables scripts written in popular programming languages to access servers and manipulate the virtual machines (VMs) they host under VMware ESX Server™ software when these hosts are managed by VMware VirtualCenter management software. In this study, the Virtual Infrastructure SDK scripting facility was combined with Dell® OpenManage™ software to demonstrate a powerful fault-tolerance capability. This article explains how Dell engineers developed a script that called the VMware VMotion™ feature to move all the VMs from a failing server to other servers in a server farm. The failing server was then taken down, serviced, brought back online, and repopulated with the VMs it had been hosting—all with little impact on end-user applications.

BY DAVE JAFFE, PH.D., AND TODD MUIRHEAD

For server consolidation, fault tolerance, and ease of administration, many enterprises have moved all or part of their data center applications onto virtual machines (VMs) that reside on multiple Dell PowerEdge™ servers running VMware ESX Server software. Administrators can use VirtualCenter—the VMware central administration program—to manage ESX Server–based servers as a logical pool of resources in a server farm. VirtualCenter lists the physical servers and VMs in the server farm and displays a console showing current information regarding performance, tasks, and events occurring on those servers. VirtualCenter also allows administrators to create, clone, start, and stop VMs, as well as to move live VMs between physical servers running ESX Server software through the use of VMware VMotion technology.¹

VMware provides the VMware Virtual Infrastructure SDK (software developer’s kit) scripting facility, which enables developers to access VirtualCenter data and call VirtualCenter commands through a Web services interface that is accessible using most popular programming languages, including C# and Java.² By enabling developers to access VirtualCenter features through a Web services interface, the VMware Virtual Infrastructure scripting facility extends the functionality of VirtualCenter and enables

¹ These features are discussed in “Introducing VMware ESX Server, VirtualCenter, and VMotion on Dell PowerEdge Servers” by Dave Jaffe, Ph.D.; Todd Muirhead; and Felipe Payet; in Dell Power Solutions, March 2004.
² To download the VMware Virtual Infrastructure SDK package, visit www.vmware.com/support/developer/vc-sdk.
developers to create custom VMware applications that assist them in managing virtual server infrastructures. For example, developers can create a script to clone additional VMs from existing “golden” VM images in a repository, and then tailor each VM as required—for instance, by changing the host name and IP address.

Server administration tools such as Dell OpenManage IT Assistant (ITA) and Dell OpenManage Server Administrator (OMSA) provide other important components needed to manage large server farms running VMs. Administrators can use ITA (the centralized management console) and OMSA (the agent running on each server) to detect and report error conditions, inventory hardware assets, and update server software and firmware. A key feature of ITA is its capability to call administrator-written scripts in response to specific events from servers, such as warnings. This functionality can be combined with VMware scripting capabilities to provide tools that enhance the power and flexibility of a virtual computing infrastructure.

This article describes an example of how the Dell OpenManage infrastructure can be used together with the VMware Virtual Infrastructure SDK scripting facility to load balance a farm of ESX Server–based servers. In December 2004, a Dell test team developed a script to call the VMware VMotion feature to move all the VMs from one ESX Server–based server to another without shutting down the VMs. The script was designed such that, when OMSA detected an error on a server in the server farm, the failing server triggered OMSA to send a warning to ITA, which in turn called the VMware script on the server that ran VirtualCenter administration software. This script, a C# program, used the VMware Virtual Infrastructure script to call the VMware VMotion facility to move VMs from the failing server while the VMs were running. As a result, the VM migration had virtually no impact on end-user applications. Once removed from the server farm, the failed server could be serviced, brought back online, and repopulated with the VMs that had been temporarily moved to other servers.

Using the VMware Virtual Infrastructure scripting facility

The VMware Virtual Infrastructure SDK package enables developers to access VirtualCenter data and call VirtualCenter commands through a Web services interface. Programs communicate with VirtualCenter using the standard Simple Object Access Protocol (SOAP)/XML Web services protocol to access data (from hosts, VMs, performance counters, events, and so on) or to issue commands (such as Clone, Migrate, and Power On/Off). The interface to the VMware VirtualCenter Web service is defined by vma.wsdl, the VMware application programming interface (API) Web Services Description Language (WSDL) file. Programming languages that can issue SOAP/XML Web service requests can be used for the client programs. The SDK supplies examples in C#, Java, Perl, and Visual Basic.

Programming client applications in C# is straightforward when following the examples provided in the SDK. The details of the Web service interface are contained in a C# source file called vmaService_proxy.cs, which must be compiled with the client source code. In the sample C# files included in the SDK and in the program described in this section of this article, the calls to this proxy have a wrapper of client-callable programs inside the client application.

The program used in this example—VMotion All VMs, or vsall—illustrates the use of the VMware Virtual Infrastructure to move the VMs on a given server running ESX Server sequentially to other ESX Server–based hosts in the farm in a load-balanced manner. The program uses the VirtualCenter Web service to list the VMs on the specified host, then check the CPU utilization on the other ESX Server–based hosts in a specified migration pool, and finally, issue the VMotion Migrate command repeatedly to move the VMs off the specified ESX Server–based server.

The vsall program’s source file is called vsall.cs. (To view the vsall.cs source file, visit Dell Power Solutions online at www.dell.com/powersolutions.) As seen in vsall.cs, the vsall program follows the architecture used in the SDK sample files. VmaClient, a class defined in the source file, includes VMware code to create instances of the class, plus code to connect to the Web service as well as code to put a wrapper around vmaService_proxy.cs functions that manipulate VirtualCenter objects (such as ResolvePath, GetContents, and MigrateVM). The administrator-written function get_host_cpu_time is a special function that uses the performance objects exposed by VirtualCenter to return the total CPU time (in milliseconds) used during the previous minute by a server running ESX Server. To increase the accuracy of this sample, administrators can create a performance counter in the VirtualCenter user interface called “Past Hour,” which provides one sample per minute for 60 minutes.

The overall vsall functionality occurs in the administrator-written function Main. Four steps are required to move all VMs off a server:

1. Connect to the VirtualCenter Web service.
2. Check that the specified source server is part of the migration pool of servers running ESX Server.
3. Determine the number of VMs on the source server running ESX Server, and list the other servers running ESX Server in the migration pool.
4. Iterate through the list of VMs and move each VM to the least-loaded server in the migration pool at the time.

Connecting with the VirtualCenter Web service requires the administrator’s credentials (username and password) along with the URL of the Web service. For demonstration purposes, the username and password are hard-coded into the program in plain
Setting up Dell OpenManage to work with VMware scripts

Dell OpenManage infrastructure comprises a set of systems management tools for monitoring, updating, and managing Dell hardware. Specifically, the IT Assistant centralized console and the OMSA server monitoring and management tool are used for monitoring servers. IT Assistant is typically installed on a single server that acts as the monitoring console. OMSA is installed on each Dell PowerEdge server to monitor that server. OMSA uses Simple Network Management Protocol (SNMP) to send information to IT Assistant about hardware status changes such as a disk failure or an overheated processor. Once the information reaches the IT Assistant, an administrator-defined set of rules dictates what actions should be taken. For example, if a disk fails on the database server, IT Assistant can be set up to send an e-mail to the database administrator, and possibly run a script to shut down the database or back it up.

Every Dell PowerEdge server (except for PowerEdge SC servers) ships with Dell OpenManage software, which includes OMSA and IT Assistant. Dell supplies versions of OMSA for Windows, Linux®, and Novell® NetWare® operating systems. Servers running VMware ESX Server use the Linux version of OMSA. Because OMSA is designed to monitor the hardware of the ESX Server–based server, VMs running on ESX Server–based servers do not require their own hardware-monitoring capability.

To set up OMSA to run on an ESX Server–based server:

1. Run /usr/bin/omsetup.sh from the ESX Server console command prompt. When prompted, insert the ESX Server CD and then the OMSA for Linux CD.
2. Add this line to the end of the /etc/snmp/snmpd.conf file:

   trapsink IP address of IT Assistant server public

3. Run the service snmpd restart command.
4. Reboot the server.

The IT Assistant central console can be set up to respond to various types of events, either from specified servers or from all servers. A demonstration can be triggered by intentionally mis-setting a fan speed warning value so that OMSA sends a warning that a fan speed is too slow. This demonstration uses an Event Filter that is triggered by environmental warnings (which include fan speed) that come from ESX Server–based servers. To enable this demonstration in the IT Assistant console, select the Configuration dialog box, the Event Filters dialog box, and then the Add dialog box. In the Add dialog box, configure the following settings, and then click the OK button:

- **Filter Name:** ESXDEMO1
- **Severity Configuration:** Select warning only
- **Select Event Categories/Types:** Select all environmental
- **Select Source Nodes:** Select ESX1, ESX2, and ESX3

---

text and the URL references a non–Secure Sockets Layer (SSL) version of the Web page. In a production scenario, the username and password would be encrypted and the Web service would be accessed via SSL.

The vmal program first creates an instance of the VmaClient object (called vma), and then uses the URL, username, and password to call the Connect method of VmaClient (which puts a wrapper around the vmaService_proxy.cs Login function) to connect to the VirtualCenter Web service.

Next, vmal checks that the source host (the host from which the VMs will be moved) is a member of the migration pool of ESX Server–based servers. For demonstration purposes, the set of hosts that are available for VM migrations is hard-coded into the program in the array migration_pool_hostnames. A production version of this script could determine this set at runtime by querying VirtualCenter.

The vmal program then uses the VirtualCenter Web service to count the VMs on the source server and list the other ESX Server–based hosts in the migration pool. The method ResolvePath(“/host”) is used to obtain a handle to the VirtualCenter host list, which is used by the GetContents VirtualCenter object to capture this list in a Container object. The program then iterates through this list of ESX Server–based hosts. If a listed host matches the source host, the number of VMs currently on the source host is determined. If a listed host is not the source but is contained in the migration_pool_hostnames array, the number of CPUs on that host is stored in the hosts_to_migrate_to_n_cpus array, and the vmal program generates an additional array of host names called hosts_to_migrate_to.

Finally, vmal generates a list of VMs on the source host by drilling down into the VirtualCenter host object, and then iterates through this list using the MigrateVM object to move each VM to one of the hosts in the hosts_to_migrate_to array. The particular host is chosen by checking CPU utilization and identifying the host with the least-loaded CPUs. This is done by using the get_host_cpu_time method described earlier in this section to obtain the total CPU utilization time of that host for the last minute (in milliseconds). That value is then converted to a utilization percentage by dividing the number of milliseconds in a minute (60,000) and by the number of CPUs in the host and multiplying by 100. Each call to MigrateVM returns a handle to a task that is used to monitor the progress of the VMotion migrations. The VMotion migrations occur sequentially in vmal, but other methods are possible. The vmal program writes the progress of each migration to the Microsoft® Windows® command prompt from which vmal is called. Administrators can also track the progress of the VMotion migrations by viewing the Tasks tab in VirtualCenter. When the migrations are complete, the program disconnects from the Web service and returns control to the command prompt.

---

www.dell.com/powersolutions  Reprinted from Dell Power Solutions, May 2005. Copyright © 2005 Dell Inc. All rights reserved.
This Event Filter also contains a list of actions that will occur when the event is received. In addition to displaying an alert message on the ITA console and writing the event in the Windows event log, the Event Filter creates an action that calls a Visual Basic script called remote2.vbs. The remote2.vbs script remotely calls the vmall.exe program on the server running VirtualCenter and passes it the host name of the ESX Server–based server sending the warning. (To view the remote2.vbs script, visit Dell Power Solutions online at www.dell.com/powersolutions.) Note: The action’s executable file is cscript.exe—the Windows Script Host (WSH) executable—and the $n in the argument list will be replaced at the time the action is called by the host name of the server originating the event. The script remote2.vbs passes along this host name to the vmall program as its only argument.

Combining Dell OpenManage capabilities with VMware scripting

The process used by Dell engineers to integrate Dell OpenManage and VMware software to enable automatic migration of VMs from a failing ESX Server–based server is illustrated in Figure 1.

In the configuration demonstrated by Dell engineers, a server farm consisting of three ESX Server–based servers was managed by VirtualCenter running on a PowerEdge 1850 server. A second PowerEdge 1850 server ran ITA to manage the ESX Server–based servers in the data center (not shown in Figure 1). The two PowerEdge 1850 servers were connected to the three ESX Server–based servers and the rest of the data center through a Gigabit Ethernet switch. These three servers were in turn connected to a storage area network (SAN) using redundant host bus adapters (HBAs) in each server, redundant Fibre Channel switches, and redundant connections to the storage.

In the test scenario, OMSA running on ESX1 detected a low fan speed on that server and sent an event to ITA via SNMP. Using the ESXDEMO1 event filter and associated action described in the “Setting up Dell OpenManage to work with VMware scripts” section in this article, ITA reacted to the warning event by calling a local script, remote2.vbs, which in turn called the VMware script vmall.exe on the VirtualCenter system, passing to it the host name of the system sending the warning. The vmall program then sequentially moved the VMs from the failing ESX Server–based server to other ESX Server–based servers, moving each VM to the least-loaded ESX Server–based server at the time. When this process was completed, the failing server, ESX1, could be taken offline for repairs.

Using scripts to automate tasks and enhance availability

The scenario described in this article demonstrates one of many possible tools that administrators can develop by using the Dell OpenManage infrastructure together with the VMware Virtual Infrastructure SDK scripting facility. The script that the Dell test team developed helped automate fault-tolerance capabilities in a virtual computing infrastructure, and could be used in a real-world scenario to facilitate systems management and enhance application availability for end users. The VMware Virtual Infrastructure SDK package enables administrators to access the features of the VMware VirtualCenter administration program, thereby allowing IT organizations to develop scripts that can automate a wide variety of systems management tasks in addition to the script explored in this article.

Acknowledgments

The authors would like to thank Sudhir Shetty for the Windows Management Instrumentation (WMI) scripts, Mark Clifton for valuable discussions regarding the integration of Dell OpenManage and VMware, and Shivra Ahluwalia at VMware for support on the VMware scripting API.

Dave Jaffe, Ph.D., is a senior consultant on the Dell Global Alliances team who specializes in cross-platform solutions. Previously, he worked in the Dell Server Performance Lab, where he led the team responsible for Transaction Processing Performance Council (TPC) benchmarks and the Dell Technology Showcase. He has a Ph.D. in Chemistry from the University of California, San Diego, and a B.S. in Chemistry from Yale University.

Todd Muirhead is a senior engineering consultant on the Dell Global Alliances team. He specializes in SANs, virtualization, and database systems. Todd has a B.A. in Computer Science from the University of North Texas and is Microsoft Certified Systems Engineer + Internet (MCSE+) certified.