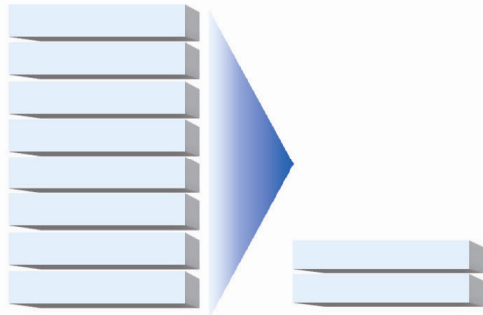# CONSOLIDATION

**CONSOLIDATING
SQL SERVER™ 2000 DATABASES
FROM
DELL™ POWEREDGE™ 2650 SERVERS
TO A
DELL™ POWEREDGE™ R610 SERVER**

REPORT

A Principled Technologies report commissioned by Dell

Principled Technologies®

DELL

Table of contents

Executive summary

Consolidating multiple database servers into a single one may let an organization save hardware, software, and operational costs. To help quantify the potential savings, Principled Technologies examined an example test case: How many Dell PowerEdge 2650s can you consolidate onto a single PowerEdge R610 with two Dell EqualLogic PS5000 iSCSI arrays attached? We defined a consolidation methodology, which we detail later in this report; conducted multiple rounds of testing; and analyzed the results.

As Figure 1 shows, the savings were substantial, with a PowerEdge R610 and two EqualLogic PS5000 iSCSI arrays able to replace 13 legacy PowerEdge 2650s—and deliver the same performance.

Consolidate **THIRTEEN** Dell PowerEdge 2650s into **ONE** Dell PowerEdge R610+ **TWO** Dell EqualLogic PS5000s

1/3 the space

1/3 the power

1/13 the licenses
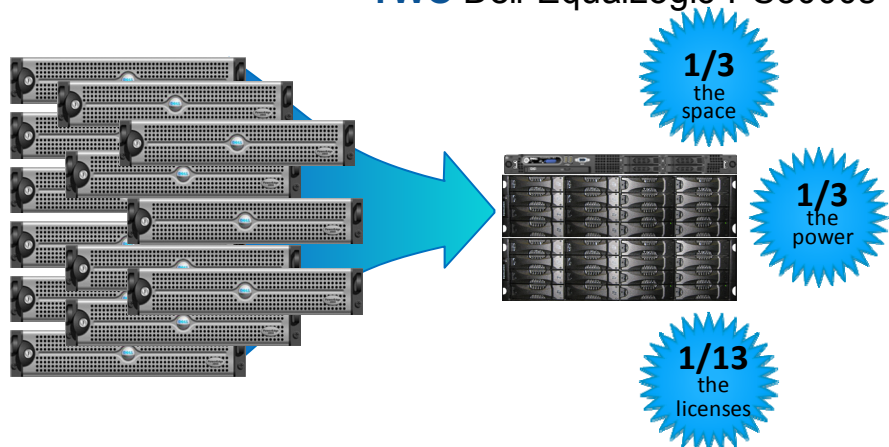
With the **SAME** performance

Figure 1. The benefits of consolidating 13 legacy Dell PowerEdge 2650 servers onto a single Dell PowerEdge R610 with two Dell EqualLogic PS5000 iSCSI arrays.

About the Dell consolidation guides

This consolidation report presents hard performance data that our hands-on benchmarking has yielded. This consolidation report is a companion to the guides "Consolidating multiple SQL Server systems onto Dell PowerEdge R610 servers using Microsoft's

3

Hyper-V"
(http://www.principledtechnologies.com/Clients/Reports/Dell/Cons
olidating_SQLServer_databases_via_Hyper-V_onto_Dell_R610.pdf)
and "Consolidating SQL Server 2000 databases onto Dell
PowerEdge R610 using Microsoft's Hyper-V"
(http://www.principledtechnologies.com/Clients/Reports/Dell/Dell_
R610_Hyper-V_SQL_2000.pdf). Those guides provide concepts,
procedures, and hard performance data to help you successfully
consolidate your Microsoft® SQL Server® 2000 databases from
multiple machines onto a single server system running Windows
Server 2003 SP2 and SQL Server 2000 SP4 in Hyper-V Virtual
Machines (VMs).

In this report, we use the legacy configuration Dell™ PowerEdge™
2650 running SQL Server 2000 as the legacy server and the Dell
PowerEdge R610 as a target consolidation server.

Consolidation overview

In this section, we briefly define **consolidation**, provide reasons
that motivate organizations to consider consolidation, define the
term **consolidation factor**, which we use extensively here, and
explain at a high level our approach to the testing we conducted
for this report.

What is consolidation?

Generally speaking, *consolidation* is the process of combining
multiple things to make a single, more effective unit. In an IT
context, you can consolidate several types of things, including the
following:

Physical servers. After a successful consolidation, all applications
should run on fewer servers than before. Ideally, those applications
should run at least as well as they did previously.
Storage. Depending on your setup, consolidating servers may let
you also consolidate storage by moving data from a number of
servers to a large, locally attached, disk storage subsystem in a
new server.
Space. As you consolidate servers, you might also reduce the
number of racks or even the number of locations that house
servers.

Why consolidate?

An effective server consolidation effort has the potential to yield an environment with more consistent management practices and improved reliability, security, and hardware utilization—all while maintaining the previous level of application performance.

Consolidation can also yield a variety of cost savings:

Hardware savings. Buying, powering, and supporting fewer servers brings obvious savings. Other potential hardware cost savings include a need for fewer racks and network switches: as the number of servers decreases, these costs decrease as well.

Software license savings. Consolidation can save an organization significant money in software licenses. We present a detailed example of potential license savings in our earlier Guide, "Consolidating multiple SQL Server systems onto Dell PowerEdge R610 servers Using Microsoft's Hyper-V" (http://www.principledtechnologies.com/Clients/Reports/Dell/Consolidating_SQLServer_databases_via_Hyper-V_onto_Dell_R610.pdf).

Maintenance and staff savings. A consolidated infrastructure offers many opportunities for maintenance, support, and staffing cost savings. Less hardware and associated equipment means fewer servers that require security patches, monitoring, and other ongoing maintenance.

Reduced support costs. The cost of a given level of support is typically proportional to the size of the installation. By reducing the number of servers, support costs are also likely to go down.

Power and cooling savings. Consolidating servers will usually save power in several ways:

- **Fewer servers.** Obviously, fewer servers consume less power.
- **More efficient servers.** Modern servers are typically more efficient than those of a few years ago, so the power consumption per server is likely to decrease.
- **Less air conditioning.** Fewer, more efficient servers produce less heat and consumer less space. Thus, cooling costs should decrease.

Regulatory compliance savings. With fewer physical devices storing data and more uniform management practices, a consolidated environment can help make the process of complying

with regulatory requirements, such as Sarbanes-Oxley and the Health Insurance Portability and Accountability Act (HIPAA), less expensive, easier, and more secure.

Consolidation factor

How many old servers can a new server replace? The answer to this question is a number we call the ***consolidation factor***. The process of determining this number is what we call ***sizing***.

Our first step in sizing was to quantify the amount of work an old server was performing. Using that information, we defined a standardized workload that was one older server's worth of work.

Our next step was to run multiple copies of the standardized workload against a new server to determine how many concurrent workloads the new server could handle while delivering at least the same performance as the old server on each workload. This number was the consolidation factor.

How we tested: An overview

In this report, we present hard performance data and the consolidation factor from testing one set of specific hardware configurations.

For the legacy server, we used a server typical of those in use roughly three years ago: a Dell PowerEdge 2650 with two single-core Intel Xeon 2.8 GHz processors, running Windows Server 2003 R2 and SQL Server 2000 SP4.

For our target consolidation server, we used the Dell PowerEdge R610 with two Quad-Core Intel Xeon E5520 2.27 GHz processors, running Windows Server 2008 Enterprise Edition SP1 x64. For each PowerEdge 2650 we consolidated onto the PowerEdge R610, we created a Hyper-V VM running Windows Server 2003 SP2 and SQL Server 2000 SP4. Each VM had two logical processors and 3.2 GB of RAM. Like the PowerEdge 2650, the PowerEdge R610 is a dual-socket server. However, the PowerEdge R610 is a 1U server, instead of requiring 2U of rack space like the PowerEdge 2650.

NOTE: We set up the PowerEdge 2650 in a configuration typical at the time of its release. The PowerEdge 2650 had two single-core processors, 8 GB of RAM, two gigabit NICs, and all five internal drive bays occupied. The PowerEdge R610 had two quad-core processors, 48 GB of RAM, four

6

gigabit NICs, and all six internal drive bays occupied. To avoid disk space becoming the limit to consolidation, we gave the PowerEdge R610 extra storage via two EqualLogic PS5000XV iSCSI arrays. Greater detail on these systems' configurations appears in the Defining our environment section.

We used the DVD Store™ Version 2 (DS2) test tool, which is freely available from http://www.delltechcenter.com/page/DVD+Store. DS2 is an open-source simulation of an online e-commerce site. Its main reporting metric is orders per minute, or OPM. Unless we state otherwise, all results in this report are in orders per minute.

To create demand on the servers, we set up 13 workload-generating desktop-class systems on our network, each running a single instance of DS2.

We ran the same workload and used the same settings as in the tests for our previous consolidation guides. Our goal was to represent the consolidation of a server facing a demanding workload, so we made sure the DVD Store test saturated at least one of the major components of the system. We made the standardized workload consume about 80 percent of the CPU capacity of the PowerEdge 2650, so that we would be consolidating a heavily utilized server. (The consolidation factor would only increase for organizations seeking to consolidate servers with lighter workloads.) To avoid any external bottlenecks, we made sure that both the network capacity and the performance of the load-generating machines never limited the performance of the server. To simulate a heavily loaded environment, the load-generating systems ran with no think time, blasting requests as quickly as the server could handle them.

We quickly learned that the bottleneck was moving data to and from the disks. With the data disks maxed out, we were able to get to about 65 percent of the capacity of the CPU between checkpoints. Though short of our original goal, we concluded that this was a reasonable maximum for the PowerEdge 2650 with only internal storage, so we settled on this workload as our standardized one for testing.

The bottom line

When we consolidated the legacy Dell PowerEdge 2650 server onto the Dell PowerEdge R610 server with two EqualLogic PS5000s providing extra storage, the consolidation factor was **13**.

We tested with two different database sizes: 10 GB and 20 GB. In this section, we present our findings for both the 10GB and 20GB scenarios that support the consolidation factor we cite above:

- 13 DS2 instances, each in a separate VM, running against the PowerEdge R610 with two EqualLogic PS5000 iSCSI arrays attached (with 10GB databases)
- 13 DS2 instances, each in a separate VM, running against the PowerEdge R610 with two EqualLogic PS5000 iSCSI arrays attached (with 20GB databases)

Standard scenario

As we note earlier, based on experience and discussions with database administrators, we determined that a common database size for this class of server is 10 GB. On the Dell PowerEdge R610 and EqualLogic PS5000 arrays, we created 13 VMs, each running Windows Server 2003 R2 Enterprise Edition SP2 x86 and each running its own instance of SQL Server 2000 SP4. In each VM, we loaded a database that was 10 GB in size. We set up 13 desktop-class systems on the same network to generate the test workloads. (For details on testing, see Appendices E, F, and G.)

Each of the lines in Figure 2 represents the performance of a single workload. As we discuss in How we tested: An overview, each workload represents the work a single PowerEdge 2650 is capable of performing. As you can see, each of the 13 workloads running on the PowerEdge R610 is performing at least as well as a single workload running on the PowerEdge 2650. This means that you could consolidate 13 PowerEdge 2650 servers onto a single PowerEdge R610 server with two EqualLogic PS5000 arrays, reaping great cost savings, without sacrificing any performance at all.

A single DS2 instance against Dell PowerEdge 2650 vs. 13 DS2 instances against Dell PowerEdge R610 with 2 EqualLogic PS5000 (10GB databases)

Legend:
- 2650
- R610 DS2 1
- R610 DS2 2
- R610 DS2 3
- R610 DS2 4
- R610 DS2 5
- R610 DS2 6
- R610 DS2 7
- R610 DS2 8
- R610 DS2 9
- R610 DS2 10
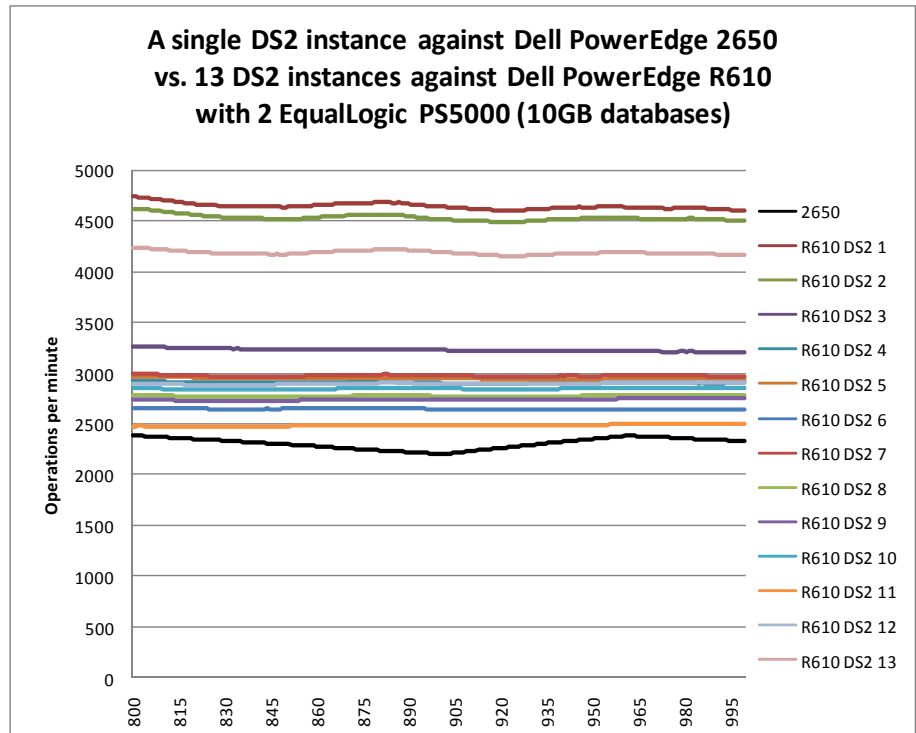- R610 DS2 11
- R610 DS2 12
- R610 DS2 13

Figure 2. A single DS2 instance running on a Dell PowerEdge 2650 vs. 13 DS2 instances running on a Dell PowerEdge R610 with two EqualLogic PS5000s, during a period of steady activity, using 13 DS2 10GB databases, each running in its own VM. Higher numbers of orders per minute are better.

High-demand scenario

While we consider 10 GB to be a common database size for this class server, we also ran a more demanding scenario using 20GB databases to see whether the consolidation factor varied.

It didn't. As Figure 3 shows, the relative performance of the PowerEdge R610 with two EqualLogic PS5000 iSCSI arrays actually improved slightly relative to the PowerEdge 2650 in the 20GB database tests as compared to the 10GB database tests.
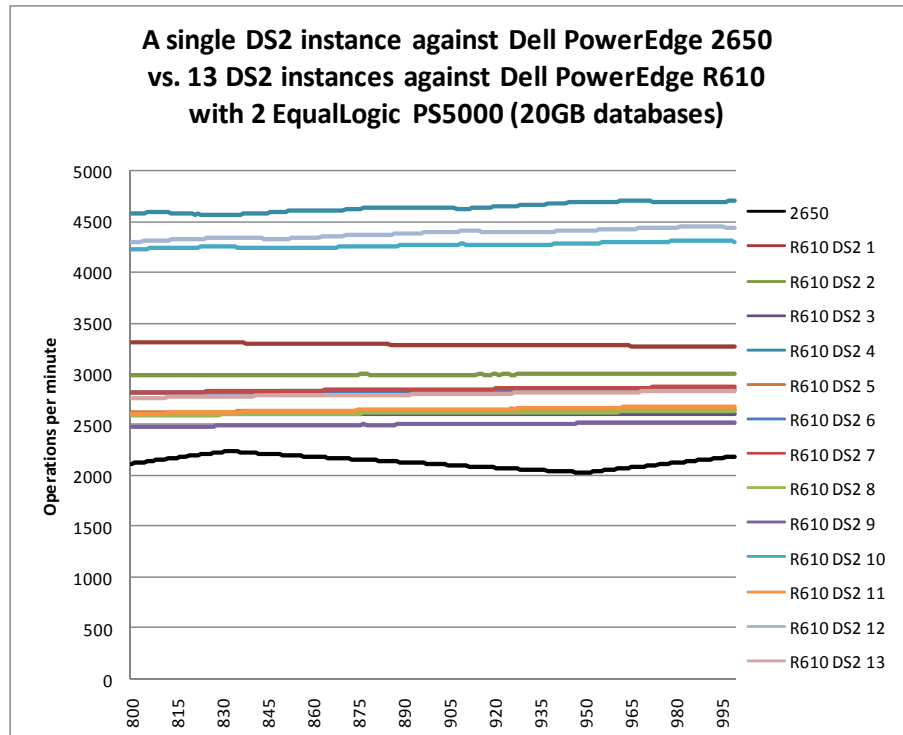
9

**Figure 3. A single DS2 instance running on a Dell PowerEdge 2650
vs. 13 DS2 instances running on a Dell PowerEdge R610 with two
EqualLogic PS5000s, during a period of steady activity, using 13
DS2 20GB databases, each running in its own VM. Higher numbers
of orders per minute are better.**

Power savings

As you might expect, the power savings after consolidation were
also dramatic. As Figure 4 shows, the combined power
requirements of a Dell PowerEdge R610 and two EqualLogic
PS5000 iSCSI arrays are about one-third (35 percent) of those of
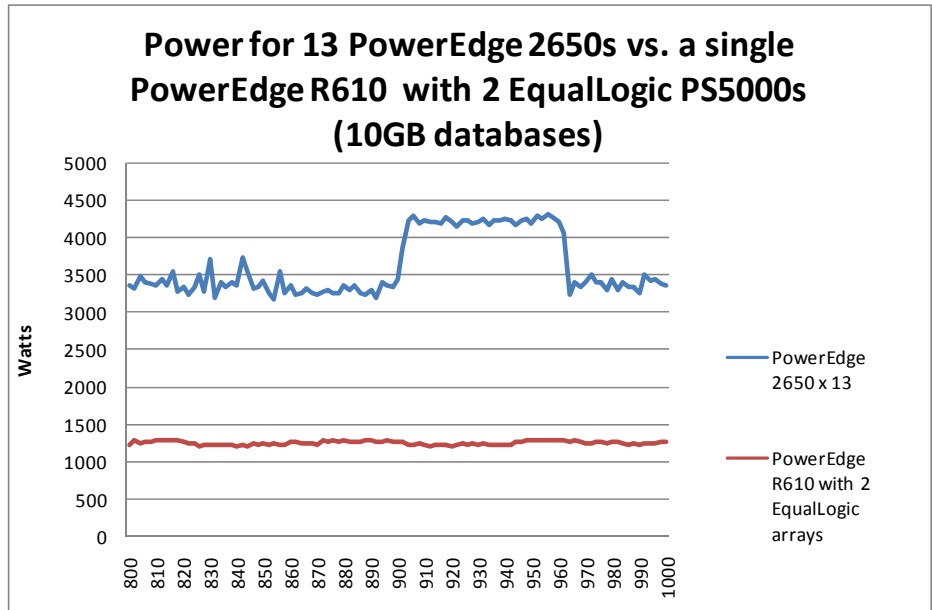the 13 Dell PowerEdge 2650 servers they could replace.

Figure 4. Power usage, in watts, for 13 Dell PowerEdge 2650 servers vs. for a single PowerEdge R610 server with 2 EqualLogic PS5000s when using 13 DS2 10GB databases, each running in its own VM. Lower power usage is better.

NOTE: The pronounced square waveform of the PowerEdge 2650 power consumption is the result of SQL Server "checkpointing," or writing its changes to disk. Checkpointing happens on a per-database basis. As the PowerEdge 2650 was checkpointing, the power decreased while the system wrote changes to disk.

Figure 5 shows that, as you might expect, increasing the database size to 20 GB did not heavily change the relative power consumption of the two servers.
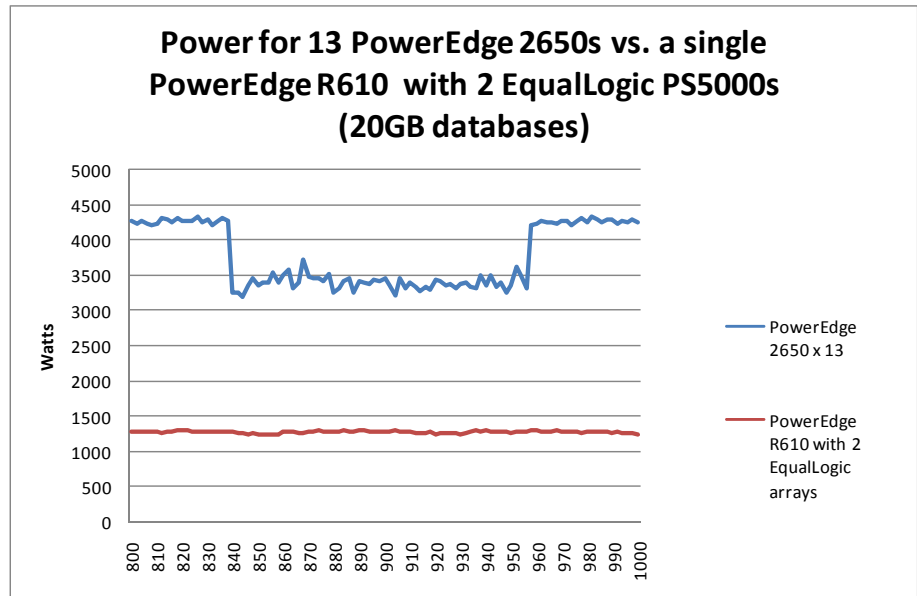
11

Figure 5. Power usage, in watts, for 13 Dell PowerEdge 2650 servers vs. for a single PowerEdge R610 server with 2 EqualLogic PS5000s when using 13 DS2 20GB databases, each running in its own VM. Lower power usage is better.

Defining our environment

We used a Windows domain containing a SQL Server 2000 database server, a new consolidation server, 13 workload-generating systems, and an Active Directory server.

To be specific, our example consolidation database server was a Dell PowerEdge R610 running Windows Server 2008 Enterprise Edition SP1 x64. Each VM on the consolidation server ran Windows Server 2003 R2 Enterprise Edition SP2 x86 and SQL Server 2000 Enterprise Edition SP4. Our legacy server was a Dell PowerEdge 2650 running Windows Server 2003 R2 Enterprise Edition SP2 x86 and SQL Server 2000 Enterprise Edition SP4. As we note above, the PowerEdge R610 used two EqualLogic iSCSI arrays to provide extra storage for the testing. Our Active Directory server was a Dell PowerEdge 1950 running Windows Server 2003 R2 Enterprise Edition SP2 x86. We connected all of the components via a gigabit Ethernet switch. Figure 6 illustrates our setup.
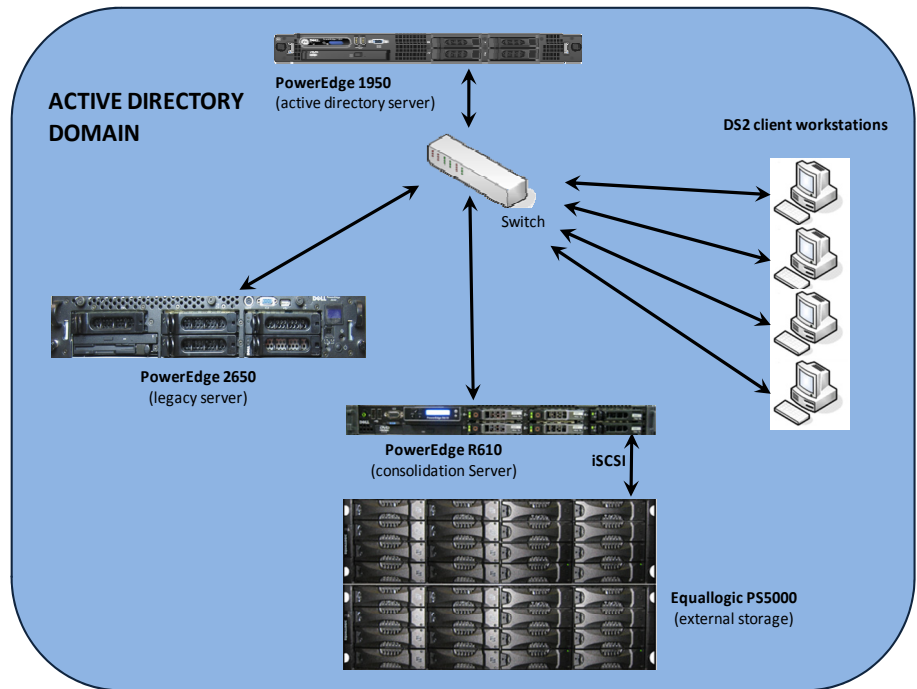
Figure 6. The setup we used in our hands-on testing and research.

Figures 7 and 8 present the hardware and software we used to simulate an Active Directory domain and associated SQL Server 2000 database servers.

Server	Processor	Memory	Disk drives
Dell 1950 (Active Directory server)	2 x Quad-Core E5440 Intel® Xeon® (2.83 GHz)	16 GB	2 x SAS, 15K RPM, 146 GB
Dell PowerEdge 2650 (SQL Server 2000 database server)	2 x single-core Intel Xeon (2.8 GHz)	8 GB	5 x Ultra160 SCSI, 10K RPM, 36 GB
Dell PowerEdge R610 (Hyper-V consolidation server)	2 x Quad-Core E5520 Intel Xeon (2.27 GHz) (2 logical processors per VM)	48 GB (3.2 GB per VM)	6 x SAS, 15K RPM, 72 GB

Figure 7. Servers we used in our hands-on testing and research for this report.

The drive arrays we used were Dell EqualLogic PS5000XV iSCSI arrays, each containing 16 x SAS, 15K RPM, 146GB disk drives, for a total of 32 drives. We dedicated two NICs to iSCSI traffic.

The workload-generating systems were desktop-class machines running Windows Vista 32-bit. All had a minimum of 1 GB of RAM and processors of speed 2.0 GHz or higher. We connected these 13 systems to the network via gigabit switches and verified that their performance was never a bottleneck during the test. We used two 16-port gigabit Ethernet switches for networking.

Server	Server operating system	SQL server version
Dell 1950 (Active Directory server)	Windows Server 2003 Enterprise Edition R2 SP2 x86	N/A (Active Directory server)
Dell PowerEdge 2650 (SQL Server 2000 database server)	Windows Server 2003 Enterprise Edition R2 SP2 x86	SQL Server 2000 Enterprise Edition SP4
Dell PowerEdge R610 (consolidation database server)	**Parent partition**: Windows Server 2008 Enterprise Edition SP1 x64 **Child partitions**: Windows Server 2003 Enterprise Edition R2 SP2 x86	**Parent partition**: N/A **Child partitions**: SQL Server 2000 Enterprise Edition SP4

Figure 8. Software we used in our hands-on testing and research.

BEST PRACTICE: Use the latest tested and validated software, firmware, and driver versions for NICs, storage arrays, and other components. You can find these software components at http://support.dell.com/support/downloads/index.aspx?c=us&l=en&s=gen.

NOTE: Because the PowerEdge 2650 was running 32-bit software, we enabled AWE to make the extra RAM available to SQL Server 2000; in conjunction with using AWE, we set the maximum server memory setting for SQL Server 2000 to 6 GB. There was no need to enable AWE in the VMs of the PowerEdge R610, because each VM only had 3.2 GB of RAM available to it.

Configuring the RAID drives: PowerEdge R610 with EqualLogic PS5000

Although the EqualLogic PS5000s managed the storage as a single volume, the VHDs appeared to Windows Server 2003 as separate disks. So, when creating our virtual disks, we used a basic file layout that followed Microsoft's recommended best practices. We separated the SQL Server transaction logs, tempdb, and the user database files in our drive configuration for the PowerEdge R610.

For a description of Microsoft's best practices, see our earlier Guide, "Consolidating multiple SQL Server systems onto Dell PowerEdge R610 servers Using Microsoft's Hyper- V" (http://www.principledtechnologies.com/Clients/Reports/Dell/Consolidating_SQLServer_databases_via_Hyper-V_onto_Dell_R610.pdf).

Below, we illustrate the exact drive layout we used (see Figure 9) and briefly describe each disk group.

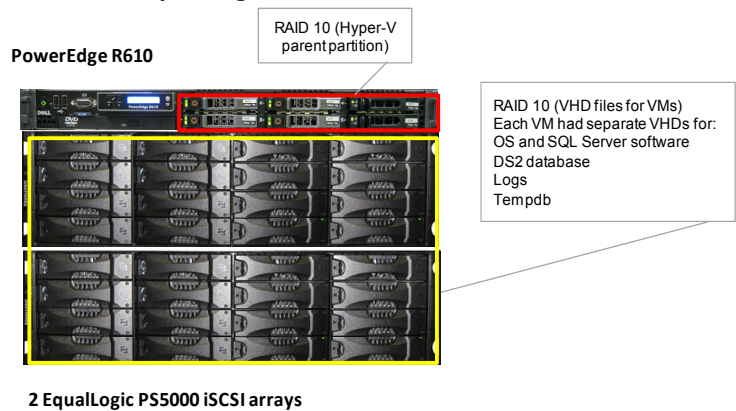**Physical Disk Configuration - PowerEdge R610 with EqualLogic PS5000**

Figure 9. The drive configuration we used in the Dell PowerEdge R610 in our hands-on testing and research.

Internal disk drives

Our Dell PowerEdge R610 server contained six drives. We configured these six drives using RAID 10 for the operating system. We also stored the database backup files there. These backup files were not part of the test. We used these backups to restore the databases after a test.

External disk drives

The EqualLogic PS5000 iSCSI arrays contained a total of 32 drives. We created one RAID 10 volume spanning both arrays. We had two dedicated iSCSI connections to the SAN.

NOTE: While we followed Microsoft best practices and put the data, logs, and tempdb in separate virtual hard drives,

16

we gave the EqualLogic SAN maximum freedom to manage the storage for us.

Configuring the RAID drives: PowerEdge 2650

Microsoft's best practices recommend separating the SQL Server transaction logs, the tempdb system database, and the user database files in our drive configuration. In the internal storage configurations, however, we had only five drives and so faced the same storage constraints that many real-world installations face. We were consequently simply unable to follow all the recommendations.

To balance performance with data redundancy, we used a RAID 1 set along with a RAID 5 set to ensure maximum protection from drive failure, as most real-world database applications require this level of data protection.

Below, we illustrate the exact drive layout we used (see Figure 10) and briefly describe each disk group.
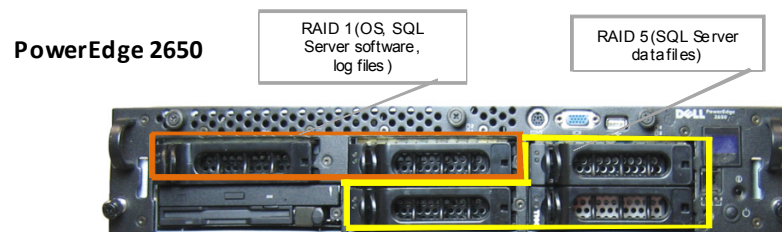
Figure 10. The drive configuration we used in the Dell PowerEdge 2650 server in our hands-on testing and research.

Our PowerEdge 2650 server contained five drives. We configured the first two internal server drives using RAID 1 for the operating system, SQL Server software, and SQL Server system databases. Because our workload did not heavily utilize tempdb, we also put the tempdb on this volume. The RAID 1 volume also held the full-text catalogs.

We configured the remaining three internal drives as one RAID 5 volume and allocated them to user data and indices. Because protecting the data was a priority, RAID 5 was the best available choice.

17

How we tested: The details

In this section, we discuss in depth how we conducted our testing. We review the tool we used and how we set up and ran the test.

Our testing tool

We conducted our testing using the Dell DVD Store application, or DS2, an open-source application with a backend database component, a frontend Web application layer, and a driver layer that operates as the middle tier and actually executes the workload.

Because our goal was to isolate and test a database server, we did not use the Web application layer. Instead, we ran the driver application directly via its command line interface.

DS2 models an online DVD store. Virtual customers log in; browse movies by actor, title, or category; and purchase movies. The workload also creates new customers. Browsing movies involves select operations, some of which use full-text search and some of which do not. The purchase, login, and new customer procedures involve updates and inserts, as well as selects.

For more details about the DS2 stress tool, see
http://www.delltechcenter.com/page/DVD+Store.

Setting up the test

The DS2 stress tool provides options to generate 10MB, 1GB, or 100GB datasets. To get the tool to generate the 10 GB and 20 GB of user data we used in this test, we had to make a few straightforward changes to the source code. We detail these changes and explain how to generate the data in Appendix A.

NOTE: The database files, including indices and other supporting metadata, require approximately 10 GB of storage space. We sized the database files considerably larger than this minimum, however, for the same reason real database administrators would: to ensure that file growth during testing was not an issue.

We built the database schema on SQL Server 2000 using the scripts Dell provided in the DS2 distribution package, though we made a few minor modifications. For details on the database build

18

process, see Appendix B. After loading the data, we built the indices and full-text catalogs. We then performed a full backup of the database. This backup allowed us to restore the server to a pristine state relatively quickly between tests.

As it ships, DS2 accesses only the database named DS2. To run against multiple databases simultaneously, we added the ability to specify the database name on the command line. The details of this change appear in Appendix D.

We also had to make a few other minor modifications to the DVD Store application's scripts. The details of these modifications appear in Appendix E.

Workload-generating systems

We used a set of 13 desktop-class systems, which we connected to our network via gigabit switches, to generate the workload for our tests. Each system contained the DS2 driver application and executed a workload against a single database on the server. An instance of the DS2 driver can access only a single database. Each instance of DS2 spawned 32 threads and ran without think time. Appendix E lists the exact parameters we used.

As we explain in the How we tested: An overview section above, we made sure that the performance of the workload-generating system and the network did not limit the test in any way. As we said, the performance limit came from the storage; the network, the CPU, and other components had capacity to spare.

We used PSEXEC to coordinate the workload-generating systems (see technet.microsoft.com at http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx). One system gathered performance statistics data from the server. Our experiments showed that this function did not affect the performance of the test.

To record each server's power consumption during testing, we used an Extech Instruments Power Analyzer/Datalogger. We used one Extech to measure the power draw of the server, and one Extech for each of the EqualLogic arrays.

In addition to recording power, we used a Digi Watchport sensor to ensure a consistent temperature and humidity level.

19

See [Appendix F](#) for details of how we measured power.

Running the test

Before starting a test, we rebooted the workload-generating systems and the server. We allowed the server to sit idle for at least 15 minutes to ensure it was finished with all aspects of the boot process.

Before starting each test, we recorded power at an idle state for 2 minutes. At the 1-minute mark, we started the data collector sets on all active workload-generating systems to gather performance counters. All workload-generating systems collected the orders per minute metric. Workload-generating system number 1 remotely collected performance data from the server. We waited 2 minutes before starting database activity to ensure that all workload-generating systems' collector sets were functioning.

At the 3-minute mark, we started the DS2 driver application on the first active workload-generating system. Every 30 seconds, we initiated the DS2 driver application on another of the active workload-generating systems, until they were all running. The test period began when all workload-generating systems had joined the test. At the beginning of each DS2 run, the driver connected all threads before it initiated the actual workload. We used 32 threads per workload-generating system in our testing. The connection time for all threads was approximately 15 seconds.

After the DS2 driver application established all connections, it executed the workload. For the purposes of this test, we did not look at results until all workload-generating systems were running. The workload included browsing products, customer logins, new customers, and purchases. We used data in the window 800 to 1,000 seconds after all workload-generating systems were running. See [Appendix E](#) for details on the parameters we selected.

After the workloads completed, we allowed the data collector sets on the relevant workload-generating systems to continue for 1 minute more, and then stopped the performance counters as well as the power measurements.

Summing up

This report shows that you can consolidate 13 legacy configuration Dell PowerEdge 2650 servers on a Dell PowerEdge R610 server

with two EqualLogic PS5000 iSCSI arrays, a move that would let you use less than one-third the rack space, about one-third the power, and one-thirteenth the number of SQL Server licenses, all while enjoying the same performance as before.

For conceptual and how-to instruction on consolidation, see our earlier guides, "Consolidating multiple SQL Server systems onto Dell PowerEdge R610 servers using Microsoft's Hyper-V" and "Consolidating SQL Server 2000 databases onto Dell PowerEdge R610 using Microsoft's Hyper-V" (http://www.principledtechnologies.com/Clients/Reports/Dell/Consolidating_SQLServer_databases_via_Hyper-V_onto_Dell_R610.pdf).

Appendix A. Changes we made to the data generation code and how we generated test data

The DS2 readme file encouraged using a Linux system rather than a Windows system when creating large databases. The reason for this is the much larger RAND_MAX constant in Linux. On Windows, RAND_MAX is 32,767. On Linux, it is 2,147,483,647. Because the code uses the C rand() function to generate IDs, the much larger range on Linux is an advantage. We consequently created our test databases on a Linux system.

Code changes to the ds2_create_orders.c module

The module ds2_create_orders.c defines constants that define the bounds of the customer ID and the product ID. The constants for the 10GB and 20GB database size did not exist. We added the constants for both those sizes.

On the command line for the ds2_create_orders module, we specified the size. The available options were S (small), M (medium), and L (large). We added the cases T for using the 10GB databases and W for the 20GB databases. In the switch statement that sets the values for the variables max_cust_id and max_prod_id, we added cases that assigned them the proper values for the 10GB or 20GB database sizes.

We recompiled the ds2_create_orders.c module on Linux, following the instructions in the header comments. We used the following command line: `gcc -o ds2_create_orders ds2_create_orders.c -lm`

Code changes to the ds2_create_cust.c module

We had to make the same changes to the ds2_create_cust.c module that we made to the ds2_create_orders.c module. The module ds2_create_cust.c defines constants that define the bounds of the product ID. The constants for the 10GB and 20GB database size did not exist. We added the constants for both those sizes.

On the command line for the ds2_create_cust module, we specified the size. The available options were S (small), M (medium), and L (large). We added the case T for using the 10GB databases and W for the 20GB databases. In the switch statement that sets the values for the variables max_cust_id and max_prod_id, we added

cases that assigned them the proper values for the 10GB or 20GB database sizes.

We recompiled the ds2_create_cust.c module on Linux, following the instructions in the header comments. We used the following command line: `gcc -o ds2_create_cust ds2_create_cust.c -lm`

The scripts we used to generate the data

We used scripts to run all four executables. The distribution did not include scripts for the 10GB and 20GB sizes. We wrote scripts based on the ds2_create_cust_large.sh and ds2_create_orders_large.sh scripts. The ds2_create_prod and ds2_create_inv executables did not ship with associated scripts, so we created scripts using the instructions in the readme files.

We found two dependencies:

- In the file ds2_create_prod_readme.txt, it says:

  ```
  In ../orders, after creating orderlines files,
  run ds2_create_inv to total up sales by product
  and create inventory load file:
  ds2_create_inv n_prods > ../prod/inv.csv
  ```

  Then, in this directory:

  ```
  ds2_create_prod n_prods > prod.csv
  ```

- The parameter you give to ds2_create_inv.sh must match the number you gave to ds2_create_orders.sh. The inv program uses the product id as an array index. You must have runds2_create_orders.sh first for the lookup to work.

Therefore, we had to run the scripts in a specific order. We ran them in the following order to create the 10GB database:

- ds2_create_orders_10gb.sh
- ds2_create_inv_10gb.sh
- ds2_create_prod_10gb.sh
- ds2_create_cust_10gb.sh

The scripts launch all their tasks in the background. Because there are order dependencies, we had to check and make sure the processes finished before we moved to the next step. On Linux, we used the command `ps -A | grep ds2`. Because the scripts create

only processes with the "ds2 prefix" in the process name, this command displayed the list of currently running processes that the last script we launched had created.

Appendix B. Building the database

We used the scripts Dell provided in the DVD Store distribution package to build the database schema, which includes the file structure, tables, indices, stored procedures, triggers, and so on. We built a master copy of the databases, both the 10GB and 20GB versions, for SQL Server 2000, and then used those master copies to restore our test databases to the relevant target server between each test run.

For more information on the scripts one uses to build the DS2 database, see the Dell Tech Center site at http://www.delltechcenter.com/page/DVD+Store.

We followed these steps to create the database:

1. We created the database and file structure using scripts Dell provided. We made size modifications specific to our 10GB or 20GB database and the appropriate changes to drive letters.
2. We created tables, stored procedures, and objects.
3. We converted the raw text files to use carriage return/line feed combinations that are the standard Windows format. (Because the text files are in Linux format, a line feed indicated a new line; on Windows, a carriage return/line feed indicates a new line.)
4. We set the database recovery model to bulk-logged to prevent excess logging.
5. We loaded the generated data. For data loading, we used a simple SQL Server Integration Services (SSIS) package. This approach provided the flexibility we needed when moving from system to system. The SSIS package used the same options, such as KEEPIDENTITY and TABLOCK options, as the original scripts.
6. We created indices, full-text catalogs, primary keys, and foreign keys.
7. We updated statistics on each table according to Dell scripts, which sample 18 percent of the table data.
8. We created ds2user SQL Server login and user for testing.

We made the following several changes in the build scripts:

- Because we varied the size of the datasets, we sized the files in our scripts to reflect the database size and the number of files per filegroup. We allowed for approximately 40 percent free space in our database

files to ensure that filegrowth activity did not occur during the testing.

- On both the PowerEdge 2650 and PowerEdge R610, we followed Microsoft's recommendation of having 0.25 to 1 file per filegroup per core. On the PowerEdge 2650, which had two cores, we used two files per filegroup. On the PowerEdge R610, which had eight cores, we used four files per filegroup.

- In the Dell DVD Store distribution, the following indices used the INCLUDE keyword, which was not a feature until SQL 2005. We could not use this keyword when building the SQL Server 2000 databases.

> IX_PROD_PRODID ON PRODUCTS
> IX_PROD_PRODID_COMMON_PRODID
> IX_PROD_SPECIAL_CATEGORY_PRODID

As a result, we modified the scripts for these indices to the following form for SQL Server 2000:

```
/******
On this index, the INCLUDE keyword cannot be used in SQL 2000
******/
CREATE INDEX IX_PROD_PRODID ON PRODUCTS
  (
  PROD_ID ASC
  )
  --INCLUDE (TITLE)
  ON DS_IND_FG
GO

/******
On this index, the INCLUDE keyword cannot be used in SQL 2000
******/
CREATE INDEX IX_PROD_PRODID_COMMON_PRODID ON PRODUCTS
  (
  PROD_ID ASC,
  COMMON_PROD_ID ASC
  )
  --INCLUDE (TITLE, ACTOR)
  ON DS_IND_FG
GO

/******
On this index, the INCLUDE keyword cannot be used in SQL 2000
******/
CREATE INDEX IX_PROD_SPECIAL_CATEGORY_PRODID ON PRODUCTS
  (
  SPECIAL ASC,
  CATEGORY ASC,
  PROD_ID ASC
  )
  --INCLUDE (TITLE, ACTOR, PRICE, COMMON_PROD_ID)
  ON DS_IND_FG
GO
```

- We did not use the DBCC PINTABLE command for the CATEGORIES and PRODUCTS tables, because Microsoft recommends against this practice.
- We created a SQL Server login called ds2user and a database user mapped to this login for each database. We made each such user a member of the db_owner fixed database role.

Appendix C. Restoring the database on SQL Server 2000

This appendix details the procedures we used to reinitialize the SQL Server 2000 DS2 databases between test runs.

For speed and simplicity, we use the backup and restore method to move the database files to the target servers. Between each test run, we used the processes we outline below to recreate the DS2 database, as well as to recreate and repopulate the full-text catalog and associated index between each. For the PowerEdge R610, we repeated this process for each VM.

First, we created backup files for both the 10GB and 20GB versions of the database. We stored these on the internal storage of the R610. We put the backup files in a shared folder so that they would be easy to access for the PowerEdge 2650 or any of the VMs running on the R610.

We created a DOS batch file so that we could easily execute the SQL we need from the command line. The command in the batch file was `osql -S %computername% -U sa -P <password> -i RestoreDS2.sql`

The contents of the RestoreDS2.sql file follow below.

> **NOTE:** The backup file shown here is the one for the 10GB database. Also, this SQL depends on the SQL Server 2000 software being installed on the C drive, and the data being on the E drive, in a directory called SQLData.

```
USE [master]
GO
DROP DATABASE DS2
GO

RESTORE DATABASE [DS2] FROM  DISK = N'\\backup-server\share\10GB-DS2-
SQL2000.bak'
WITH  FILE = 1,  MOVE N'primary' TO N'E:\SQLData\DS2.mdf',

MOVE N'cust1' TO N'E:\SQLData\DS2_1.ndf',
MOVE N'cust2' TO N'E:\SQLData\DS2_2.ndf',
MOVE N'cust3' TO N'E:\SQLData\DS2_3.ndf',
MOVE N'cust4' TO N'E:\SQLData\DS2_4.ndf',
```

```sql
MOVE N'ind1' TO N'E:\SQLData\DS2_5.ndf',
MOVE N'ind2' TO N'E:\SQLData\DS2_6.ndf',
MOVE N'ind3' TO N'E:\SQLData\DS2_7.ndf',
MOVE N'ind4' TO N'E:\SQLData\DS2_8.ndf',

MOVE N'ds_misc1' TO N'E:\SQLData\DS2_9.ndf',
MOVE N'ds_misc2' TO N'E:\SQLData\DS2_10.ndf',
MOVE N'ds_misc3' TO N'E:\SQLData\DS2_11.ndf',
MOVE N'ds_misc4' TO N'E:\SQLData\DS2_12.ndf',

MOVE N'orders1' TO N'E:\SQLData\DS2_13.ndf',
MOVE N'orders2' TO N'E:\SQLData\DS2_14.ndf',
MOVE N'orders3' TO N'E:\SQLData\DS2_15.ndf',
MOVE N'orders4' TO N'E:\SQLData\DS2_16.ndf',

MOVE N'ds_log' TO N'C:\Program Files\Microsoft SQL
Server\MSSQL\Data\DS2_17.ldf',

NOUNLOAD,  STATS = 10
GO

USE [DS2]
GO

IF  EXISTS (SELECT * FROM dbo.sysusers WHERE name = N'ds2user')
EXEC dbo.sp_revokedbaccess N'ds2user'
USE [DS2]
GO

EXEC dbo.sp_grantdbaccess @loginame = N'ds2user', @name_in_db =
N'ds2user'
GO

EXEC sp_addrolemember 'db_owner', 'ds2user'
GO

USE [DS2]
GO

EXEC sp_fulltext_database 'enable'
EXEC sp_fulltext_catalog  'FULLTEXTCAT_DSPROD', 'create', 'C:\Program
Files\Microsoft SQL Server\MSSQL\FTData'
```

```
EXEC sp_fulltext_table    'PRODUCTS',                'create',
'FULLTEXTCAT_DSPROD', 'PK_PRODUCTS'
EXEC sp_fulltext_column   'PRODUCTS',                'ACTOR', 'add'
EXEC sp_fulltext_column   'PRODUCTS',                'TITLE', 'add'
EXEC sp_fulltext_table    'PRODUCTS',                'activate'
EXEC sp_fulltext_catalog  'FULLTEXTCAT_DSPROD', 'start_full'
GO
```

The first time you create the database on a server, you will have to create the ds2user login. You only do this once. If you restore the database again, as we did during our repeated tests, you do not have to recreate the SQL Server login. You will, however, need to remap the database user to the SQL Server login upon each database restore. The script we used for this is below.

```
USE [DS2]
GO

IF  EXISTS (SELECT * FROM dbo.sysusers WHERE name = N'ds2user')
EXEC dbo.sp_revokedbaccess N'ds2user'
GO

EXEC dbo.sp_grantdbaccess @loginame = N'ds2user', @name_in_db =
N'ds2user'
GO

EXEC sp_addrolemember N'db_owner', N'ds2user'
GO
```

Appendix D. Code changes to DS2

Changes to the ds2xdriver.cs module

To use the 10GB and 20GB databases we created earlier, we had
to change the following constants:

- In the routine Controller(), we changed the string
  "sizes". We added the T option for the 10GB database
  size and the W option for the 20GB database size. DS2
  uses the sizes string to interpret the db_size_str option.
- In the class Controller, we changed the arrays
  MAX_CUSTOMER and MAX_PRODUCT. To each, we
  added values specifying the bounds for the customer
  and product IDs. The Controller() routine uses these
  arrays.
- We added a command line parameter for the database
  name, "—database_name". As it ships, DS2 accesses
  only the database named DS2. We needed the name to
  be a parameter so we could run against multiple
  databases on the same server.

Changes to the ds2sqlserverfns.cs module

We changed the connection string to increase the number of
available connections and to not use the default administrator
("sa") account. The default version of DS2 capped the number of
connections at 100. We raised the limit to 200 to allow room for
experimentation. As we note above, the default version used the sa
account for its operations. We created a user account called
ds2User and used that account. The connection string is called
sConnectionString.

We also changed the connection string to use a command line
parameter for the database name.

The ds2connect routine defines ConnectionString. We used the
following string; the changes we made appear **in bold**.

```
string sConnectionString = "User
ID=ds2User;Initial Catalog="+dbname+";Max Pool
Size=200;Connection Timeout=120;Data Source=" +
Controller.target;
```

31

Building the ds2sqlserverdriver.exe executable

We recompiled the ds2xdriver.cs and ds2sqlserverfns.cs module on Windows by following the instructions in the header comments. Because the instructions were for compiling from the command line, we used the following steps:

1. We opened a command prompt.
2. We used the `cd` command to change to the directory containing our sources.
3. We ran the batch file C:\Program Files\Microsoft Visual Studio 9.0\Common7\Tools\vsvars32.bat. This set up the environment variables for us.
4. We executed the following command:

```
csc /out:ds2sqlserverdriver.exe
ds2xdriver.cs ds2sqlserverfns.cs
/d:USE_WIN32_TIMER /d:GEN_PERF_CTRS
```

Appendix E. How we ran DS2

Performance statistics

Our workload-generating systems were desktop-class machines running the 32-bit version of Windows Vista. DS2 requires .NET 2.0 on all systems.

The DS2 driver application creates an orders per minute (OPM) performance counter on all systems. We created data collector sets on all systems to collect the OPM statistics once every second.

In addition, we remotely collected server performance statistics via a data collector set in Reliability and Performance Monitor on workload-generating system number 1 for all tests. On that system, we created an extra data collector set to collect performance statistics from the target server (see Figure 11).

Object	Counter
Memory	Available Mbytes
Logical Disk	% Idle time, Disk bytes/sec, Disk Transfers/sec (all drives)
Logical Processor	% Total Run Time (all logical processors and total)
Virtual Network Adapter and Virtual Switch	Bytes/sec
Network Interface	Total Bytes/sec
iSCSI connections	Bytes Received and Bytes sent

Figure 11. Performance statistics we monitored during testing.

NOTE: By default, the Reliability and Performance Monitor stores its results in binary format. Before testing, we specified that the Reliability and Performance Monitor use comma-separated values (CSV) format.

Test timing

We restarted all physical machines, both servers and workload-generating systems, prior to each test and allowed them to rest for a minimum of 15 minutes before we began the test. Figure 12 provides the order of events for the testing process.

Event
We started the power measurements.
We used PSExec to remotely start the logman utility to initiate the data collector sets on all the active workload-generating systems.
We used PSExec to remotely start the DS2 workload on the first of the active workload-generating systems. We will start DS2 on another workload-generating system every 30 seconds until all workload-generating systems have connected.
All threads are connected for all workload-generating systems, and the test period begins.
Each workload-generating system runs for 30 minutes before exiting the DS2 application. We chose to run our tests for 30-minute periods because the workload had stabilized sufficiently.
The DS2 workload ends, and the threads exit.
We used PSExec to remotely halt the data collector sets on all the active workload-generating systems. We stopped the power measurements and archived the data extracts.

Figure 12. The order of testing.

Test parameters

The DS2 driver application allows for multiple parameters, all of which Dell documents at their Tech Center site at http://www.delltechcenter.com/page/DVD+Store. Figure 13 provides the parameters we used on all the workload-generating systems.

All other parameters were at their default values.

To show how we call the DS2 driver application on a workload-generating system, the following sample code shows that call using a 20GB database.

```
c:\ds2run\ds2sqlserverdriver --target=<server or VM to
test> --ramp_rate=10 --run_time=30 --n_threads=32 --
db_size_str=W --think_time=0 --database_name=DS2
```

DS2 parameter	Value we used
--target	Either 2650-SQL, the machine name for the PowerEdge 2650, or the IP address of the VM a particular client was testing.
--ramp_rate	10
--run_time	30
--n_threads	32
--db_size_str	T to use the 10GB database or W to use the 20GB database. These databases are not part of the default distribution. See Appendix D for details.
--think_time	0
--database_name	We added this parameter into the DS2 driver application. It allows us to test databases with different names. Although we did not use this feature for this test, we left the parameter in for clarity. See Appendix D for details.

Figure 13. Parameters we used on all the workload-generating systems.

Appendix F. Power measurement

To record each system's power consumption during testing, we used an Extech Instruments (www.extech.com) 380803 Power Analyzer/Datalogger. Because the PowerEdge 2650 had two power supplies, we measured the power draw of that server by using a single Extech with a splitter cable.

When we were using the EqualLogic PS5000 arrays, we used a second and third Extech to measure the power draw of each drive array. The EqualLogic PS5000 also had dual power supplies, so we used a splitter cable to measure the power draw through a single meter.

We connected all the Extech Power Analyzers to one monitoring system to record the power draw of the systems. We used the Power Analyzer's Data Acquisition Software (version 2.11) to capture all the recordings. We installed the software on a separate PC, to which we connected all the Power Analyzers via a separate RS-232 cable for each one.

We captured power consumption at 1-second intervals.

To gauge the idle power usage, we recorded the power usage for 1 minute while each server was running the operating system but otherwise idle.

If we were using a second Extech for the test, we summed the wattage of both meters. We averaged the power usage during the period the server was running the benchmark. We call this time the *power measurement interval*.

While recording power, we used a Digi Watchport sensor and the Watchport Manager software (version 1.19.07) to monitor the temperature and humidity levels.

1. We started the Power Analyzer software on the power monitoring PC. If we were using the two EqualLogic PS5000s in the test, we started separate instances of the Power Analyzer software to monitor the second and third Extech meter.
2. Before beginning the test, we recorded the power usage for 2 minutes while each server was running the operating system but otherwise idle.
3. We started the test and recorded power during the run.

4. We recorded for 1 minute past the end of the test.
5. Power Analyzer saves its data in a comma-separated format. We imported the CSV files into Microsoft Excel®.
6. If we used more than one meter, we summed the power draw for all meters at all data points.

Appendix G. How we report results

Identifying a period of heavy activity

For all of our metrics, we consider the start of the test to be the point when the controller has started all of the clients. We use the results of the time period from 800 to 1,000 seconds after all clients have started. This is a period of steady activity and heavy load, and it suffers from neither ramp-up nor ramp-down effects.

During ramp-up, the server is adjusting to a sudden and dramatic increase in demands on SQL Server. Until the system stabilizes, the results do not represent the true capacities of the server. One particular concern is the possibility of some threads returning highly inflated rates. These inflated rates are not representative of the system under full load and including them can distort the results.

During the ramp-down period, overall performance eventually drops as more and more threads terminate. Including data from the period after threads start terminating can consequently distort results.

In both the ramp-up and the ramp-down periods, power consumption can be lower than when the system is under full load.

Of course, stable does not necessarily mean flat. A number of factors, such as SQL Server's checkpointing behavior, can create some fluctuation during the stable period.

Based on our observations as we tested, we determined that the period from 800 seconds to 1,000 seconds represented stable system behavior.

Determining the median run

For each configuration, we ran three runs. We report the median run, which we determined as follows:

1. We copied the results for all three runs from the workload-generating systems to a single results-processing system.
2. We imported the results into Microsoft Excel.

> **NOTE:** By default, the Reliability and Performance Monitor stores its results in binary format. Before testing, we specified that the Reliability and Performance Monitor use comma-separated values (CSV) format.

3. For each of the three runs, we determined the average OPM rate during the period from 800 seconds to 1,000 seconds into the test by taking the OPM at 1-second intervals and calculating the arithmetic mean.
4. We looked at the three averages and identified the median; we reported the results of this run.

Reporting DS2 results

To report the DS2 OPM data, we followed these steps:

1. For the median run of each configuration, we copied the OPM rates for each workload-generating system in the test into a single Excel worksheet. For the Dell PowerEdge 2650, there was always only 1 workload-generating system. For the Dell PowerEdge R610, there were 13 workload-generating systems.
2. We looked at only the range of data from 800 to 1,000 seconds into the test, i.e., the same data we used to compute the averages.
3. We plotted the results on a single chart. The charts had 14 lines, with a single black line representing the PowerEdge 2650 results, and 13 colored lines representing the PowerEdge R610 results.

Reporting the power results

To report the power usage of each test configuration, we followed these steps:

1. For the median run of each configuration, we copied the power draw, in watts, to an Excel worksheet.
2. Because the PowerEdge R610 configuration used the EqualLogic arrays, we summed the power for the server and the two arrays. As we explain in the Power measurement section, we used separate Extech meters for the server and for each drive array.

> **NOTE:** The output from the Power Analyzer software reports the power with a 'W' suffix, as in "217W". We used the Excel replace feature to delete the Ws.

3. We looked at only the range of data from 800 to 1,000 seconds into the test, i.e., the same time period over which we computed the OPM averages.

4. We plotted the results on a single chart. In this case, there were always only two lines: one for the Dell PowerEdge 2650 and one for the PowerEdge R610 with its two arrays.
5. Additionally, we computed the average power draw during the period from 800 to 1,000 seconds into the test. We computed the average of the combined power draw of the server and the storage.
6. We multiplied the power draw of the PowerEdge 2650 by the consolidation factor for this set of results (13).
7. We then computed the power draw of the PowerEdge R610 with its EqualLogic arrays as a percentage of the consolidated power for the number of PowerEdge 2650s it could replace. We refer to these percentages in the report.

About Principled Technologies

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 you assess how it will fare against its competition, its performance, whether it's ready to go to market, 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.

Principled Technologies, Inc.
1007 Slater Road, Suite 250
Durham, NC, 27703
www.principledtechnologies.com

Principled Technologies is a registered trademark of Principled Technologies, Inc.
All other product names are the trademarks of their respective owners

Dell AD # 810003687