

# Performance Characteristics of PERC 3/QC Release 1.2 RAID Controller

By Mike Molloy, Ph.D. and Serdar Acir

This article reviews key features of the Dell® PERC 3/QC Release 1.2 RAID controller, presents results of performance tests, and demonstrates how disk latency depends on disk subsystem design. The performance analysis uses I/O size, workload type, number of disks, cache enabling and disabling, and RAID level as input parameters.

The Dell® PowerEdge® Expandable RAID Controller, version 3, Quad Channel (PERC 3/QC) Release 1.2 is a peripheral component interconnect (PCI) adapter card that provides a host-based RAID (redundant array of independent disks) solution. In a previous *Power Solutions* article, Dell compared the performance of PERC 3/QC Release 1.2 to PERC 2/QC and measured PERC 3/QC Release 1.2 performance with different RAID configurations and disk speeds.<sup>1</sup> This article further examines factors affecting the performance of PERC 3/QC Release 1.2 and disk subsystems and presents the results of additional testing.

## Features of PERC 3/QC Release 1.2

PERC 3/QC Release 1.2 provides four SCSI channels on a high-performance, 64-bit, 66 MHz PCI bus. Each channel supports up to 12 Ultra3 low-voltage differential (LVD) SCSI drives in a Dell PowerVault® 210S or 211S enclosure. PERC 3/QC Release 1.2 allows up to 40 logical drives with a maximum of 32 physical drives per logical drive. The controller supports RAID levels 0, 1, 5, 10, and 50.

### Onboard processor and cache

PERC 3/QC Release 1.2 employs a dedicated 100 MHz Intel® i960® core processor to perform logical I/O operations. This onboard processor off-loads processing tasks from the host CPU

and improves I/O and overall system performance, especially in I/O-intensive environments. To improve RAID-5 performance, the processor includes a hardware XOR engine for RAID-5 parity calculations.

PERC 3/QC Release 1.2 is also equipped with a removable 64 MB or 128 MB ECC (error checking and correction) DIMM (dual in-line memory module) cache. A battery backup preserves cache contents up to 72 hours in the event of a power outage.

### Throughput

Incorporating Ultra3 (Ultra160) LVD SCSI technology, PERC 3/QC Release 1.2 can support a maximum throughput of 160 MB/sec per channel in burst mode, which is twice the burst rate offered by Ultra2 technology. Using all four channels, PERC 3/QC Release 1.2 has a theoretical maximum throughput of 640 MB/sec, but in practice, throughput is limited by external factors such as the disk subsystem architecture, the application environment, and the technology of intermediate hardware components. For example, if the controller is connected to a PCI bus with a maximum throughput of 528 MB/sec, its throughput will automatically degrade to this level. The tests described later in this article use megabyte-per-second throughput as a measure of performance.

<sup>1</sup>Molloy, Mike, Ph.D. and Serdar Acir. "PERC 3/QC Array Storage Controller." Dell *Power Solutions*, Issue 3, 2001.

## Factors affecting disk subsystem performance

To maximize the throughput of a controller, disk subsystem latency must be minimized. Some of the most important factors that affect disk subsystem performance are workload type, controller caching, disk scaling, and disk caching.

### Workload type and the controller cache

To minimize I/O delay, the disk subsystem must be tuned according to the type of I/O that dominates the workload. Many applications perform random reads and writes, such as Web servers, file servers, and database management systems (DBMS). Other applications, such as streaming media servers, are sequential read/write intensive.

The onboard cache of PERC 3/QC Release 1.2 can be configured for read-ahead and write-back caching. A read-ahead cache minimally improves performance for random-read workloads because the controller cannot anticipate the next command and the logical address from which to load the necessary data into the cache.

In sequential-read workloads, read-ahead caching can be useful. However, bypassing the read-ahead cache on a heavily loaded system can lead to higher I/O performance, depending on the structure and characteristics of the physical disk array. A write-back cache can significantly improve performance under a random-write workload because the host bus adapter returns control to the operating system as soon as the data is written into the cache.

Tuning the system to reduce I/O delays is particularly important in DBMS environments. Although the benchmark in this article simulates a conventional DBMS application, many database applications incorporate complex data-locking mechanisms. Data-locking mechanisms ensure data consistency by halting related database operations until the database retrieves or updates the necessary data. Therefore, to prevent additional data-lock latency, the I/O subsystem latency should be kept as low as possible.

### Disk scaling

Generally (but not necessarily) as the I/O size and the number of drives in the array increase, the overall disk subsystem performance improves up to a saturation point, and the drawbacks of slower RAID levels are reduced proportionately. The section “How I/O size affects throughput” discusses test results that illustrate the relationship between I/O size and throughput.

A physical read/write operation to a disk, which takes several milliseconds (ms), accounts for the majority of disk service time, whereas data transfer over the SCSI bus, which occurs in

nanoseconds, accounts for a very minor portion. Although the physical disk overhead is unavoidable, administrators can decrease delays from the disk array by increasing the number of disks, which increases the degree of parallel execution.

For example, consider a RAID-0 array with a stripe size of 64 KB where a write operation to disk takes 4.5 ms and incoming I/O is sufficiently large enough to scale. Neglecting additional SCSI protocol overheads, a SCSI channel with a throughput of 160 MB/sec can transfer a maximum of 2.5 stripes per millisecond:

$$160 \text{ MB/sec} \div 64 \text{ KB} \cong 2500 \text{ stripes/sec} = 2.5 \text{ stripes/ms}$$

In 4.5 ms, the channel can send

$$4.5 \text{ ms} \times 2.5 \text{ stripes/ms} = 11.25 \text{ stripes} \sim 12 \text{ stripes}$$

If the logical drive contains 12 disks, the system can write one stripe to each disk. After writing to the twelfth disk, the system can return to the first disk. So in this example, 12 or more disks in a logical drive can prevent queuing on the drives. The controller does not need to wait for the disks to become idle, thus increasing SCSI bus utilization.

### Disk caching

Disk caching can improve performance, especially for random-access workloads. For random writes, write-back caching is advantageous, especially on arrays with a limited number of disks and small I/O size. While the workload randomly accesses the disks, the I/O distribution over the physical disks may not be balanced. A process may repeatedly write to the same disk while other disks in the array remain completely idle. A write cache on the physical disk helps overcome this imbalance by working as a buffer. Also, by increasing the number of disks in a logical drive, the probability of filling up the cache of a particular disk decreases.

For random reads, read caching on the physical disk is not useful unless, in the case of RAID-0,

$$\text{I/O size} \geq (\text{stripe size}) \times (\text{number of disks in the array} + 1)$$

The random-read operation requests the next piece of data from the same disk, making the read-ahead succeed for RAID-0. Using the same methodology, this equation can be re-derived for other RAID levels.

Under a continuous workload in a RAID environment, sequential writes theoretically will not benefit from write caching, regardless of whether a queue exists on the physical disk. If no

Administrators can decrease delays from the disk array by increasing the number of disks, which increases the degree of parallel execution.

queue exists, incoming data does not need to be cached because it can be immediately written to the disk. If a queue exists, the cache will fill up eventually; the cache then behaves like a full buffer, and opening a slot in this buffer depends on the actual disk write time.

However, in practice, applications may not produce the same continuous, heavy, write load. Streaming media applications, for example, access the disk subsystem in bursts, where massive amounts of data are read or written in periods because of constraints in the computing, application, or user environment. Write caching can be partially useful in such conditions. Read caching on the physical disk is useful for sequential reads regardless of I/O size, number of disks in the array, or RAID level.

## Test environment

The tests used the Intel IOMeter benchmark tool to stress the disk subsystem. Iometer generated workloads to represent three different environments:

- ▶▶ **DBMS:** 100 percent random access with 67 percent reads and 33 percent writes
- ▶▶ **Streaming read:** 100 percent sequential access with 100 percent reads
- ▶▶ **Streaming write:** 100 percent sequential access with 100 percent writes

The Dell team conducted the tests on a Dell PowerEdge 4400 server using two Intel Pentium® III 733 MHz processors with 256 KB L2 cache, 133 MHz processor bus, and 2 GB PC133 SDRAM. The server ran Microsoft® Windows® 2000 Server with Service Pack 1. PERC 3/QC Release 1.2 was mounted in a 64-bit, 66 MHz PCI slot and equipped with 128 MB of cache memory. Four Dell PowerVault 210S enclosures contained 15,000 rpm, 18 GB disk drives

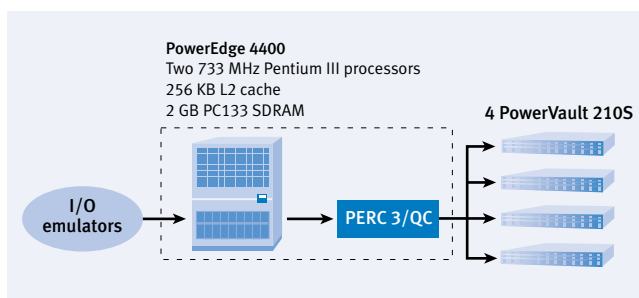


Figure 1. Test configuration

Under a continuous workload in a RAID environment, sequential writes theoretically will not benefit from write caching, regardless of whether a queue exists on the physical disk.

with disk caching enabled. All disks in the arrays were distributed equally across all channels for all tests. Figure 1 illustrates the test configuration.

The Dell team optimized the test system for high throughput by scaling and tuning the disks for high SCSI bus utilization. The disk array used for the RAID-5 tests was fully initialized to achieve higher initial write throughput. In RAID-5, a logical write I/O ordinarily leads to two physical read and two physical write I/Os. With full initialization, the number of physical read I/Os drops to one.

PERC 3/QC Release 1.2 supports stripe sizes of 2, 4, 8, 16, 32, 64, and 128 KB. For RAID levels 0, 5, and 10 with 48, 20, and 48 drives respectively, the tests used a 64 KB stripe size because that size

is commonly used in RAID configurations.

Applications pass data and commands to RAID controllers in blocks, or I/O files, of different sizes. The DBMS tests used a constant 8 KB I/O size. Other tests used 8 KB, 64 KB, 512 KB, 1 MB, and 10 MB I/O sizes to simulate a variety of applications with I/O sizes ranging from small to large. Throughout the experiment, Iometer sustained a constant queue length of 16 I/Os per logical drive, but to further improve throughput, the queue length can be optimized for each RAID configuration.

Figure 2 shows the configuration used for enabling and disabling the controller cache. Several factors affect how the cache contributes to the overall performance (see the section “How the controller cache affects throughput”). Depending on the test conditions, the Dell team enabled or disabled the controller cache to achieve better throughput. For example, under the specified conditions of the streaming write test, the controller cache was enabled only for RAID-5 to maximize the write throughput.

## How I/O size affects throughput

From the 8 KB to the 64 KB I/O size, throughput improves significantly for sequential operations (see Figures 3 and 4). This effect occurs because the more data the controller can put into a stripe, the more data can be transferred within a particular amount of time without the overhead of additional SCSI communications, address conversions, and logical calculations.

	Cache policy	Read policy	Write policy
Cache enabled	Direct I/O	Read-ahead	Write-back
Cache disabled	Direct I/O	No read-ahead	Write-through

Figure 2. Configuration parameters for enabling or disabling the controller cache

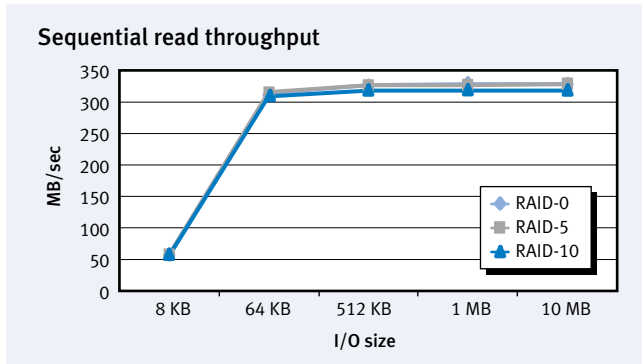


Figure 3. Sequential read throughput with controller cache disabled

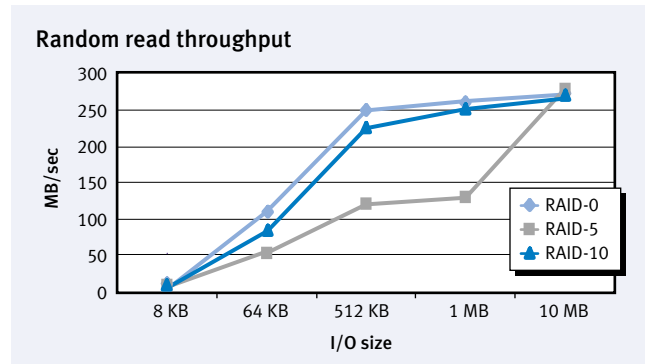


Figure 5. Random read throughput with controller cache disabled

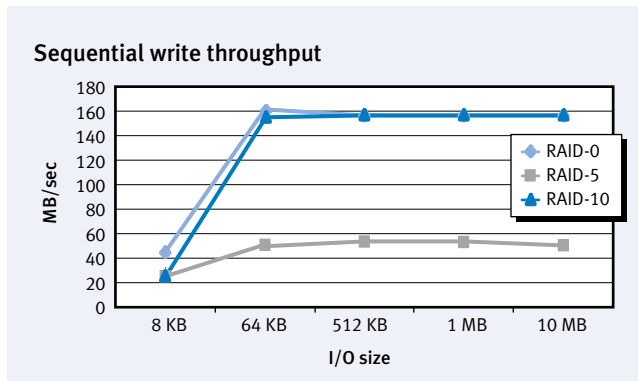


Figure 4. Sequential write throughput with controller cache enabled for RAID-5 only

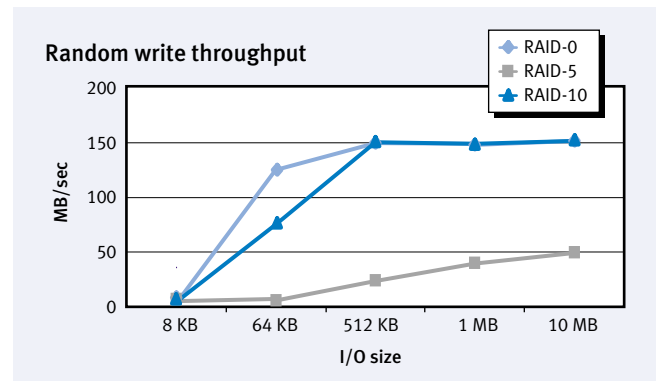


Figure 6. Random write throughput with controller cache enabled for RAID-5

For sequential operations, I/O size is not a critical performance factor as long as it is larger than the stripe size. The read or write operation always leads to track-to-track seeks on the physical drives in this case.

For random operations with large I/O sizes, the situation is different. If the I/O size is larger than the stripe size (64 KB), each I/O is divided into stripes of 64 KB that are transferred one after another as if the request were a sequential read or write. For example, in a random write operation, an I/O of 10 MB will be divided into 64 KB chunks. If the number of disks is less than the number of chunks, then the process must return and write to the previously accessed disk, which leads to a track-to-track seek overhead instead of a full seek overhead on these disks.

This effect can be observed in Figures 5 and 6, where the throughput slightly increases for large I/O sizes. The increase is more significant for RAID-5 because the cache provides space to perform computational operations and it reaches the saturation point only at very large I/O sizes.

For sequential operations, I/O size is not a critical performance factor as long as it is larger than the stripe size.

### How the controller cache affects throughput

Depending on the workload type and the physical characteristics and configuration of the disk subsystem, the controller can achieve higher throughput without using the controller cache. This effect occurs because direct response from the disk drives in a well-optimized disk subsystem can sometimes be faster than the cache response. When a cache is used, the controller must write into and read out of it, using twice as much bandwidth as direct disk access.

A DBMS workload comprises random, small, logical reads and writes of 8 KB. For such small I/O sizes, the size of the cache and the workload distribution algorithm also affect the overall throughput. The DBMS workload leads to lower throughputs, as shown in Figure 7. As the throughput gets smaller, the throughput difference between cache-enabled and cache-disabled configurations decreases.

### Various factors affect disk subsystem performance

The tests show that, in streaming media environments, performance improves when the I/O size

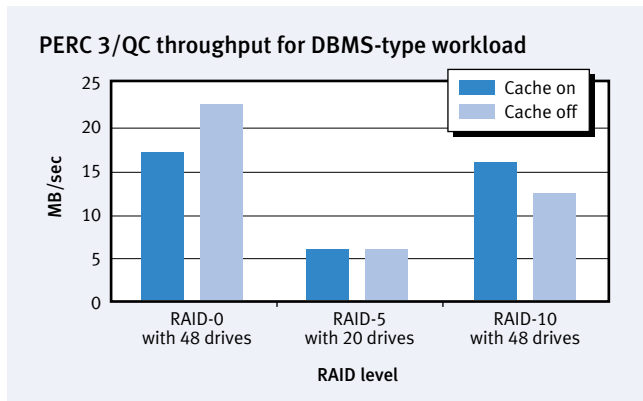



Figure 7. DBMS throughputs for several different configurations

increases at least to the size of the stripe. For relational database management systems, the relational characteristics of the application programs can be a limiting factor. Increasing the I/O size may cause performance pitfalls in some application environments; for example, 8 KB is the optimal I/O size for performance in Microsoft SQL Server.

PERC 3/QC Release 1.2 offers a maximum stripe size of 128 KB. However, the usefulness of a large stripe size again depends on the characteristics of the application environment; the application may not benefit from reading or writing large stripes of data.

Enabling the controller cache does not necessarily increase performance. To see a performance difference between cached and non-cached systems, an IT specialist needs to sufficiently stress the disk subsystem, which can be accomplished in a benchmarking environment.

Many factors affect the throughput of the disk subsystem, such as type, pattern, relational characteristics, intensity, and I/O size of the workload; RAID level and stripe size; technology of physical devices; cache type and size; number, type, configuration, and distribution of disk drives across the channels; PCI bandwidth; and driver and firmware optimization. The IT specialist should consider these factors, as well as fault tolerance requirements and configuration of available hardware resources, when designing a storage subsystem with minimum disk latency. 

**Mike Molloy, Ph.D.** ([mike\\_molloy@dell.com](mailto:mike_molloy@dell.com)) is a senior manager on the System Performance Team at Dell and has held professorships at the University of Texas and Carnegie Mellon University. He has been active in computer performance for more than 20 years and has served as chairman of the Association for Computing Machinery (ACM) special interest group on computer performance. Mike has published many papers and most recently a book, *Fundamentals of Computer Performance Modeling*. Mike received a Ph.D. in Computer Science from UCLA.

**Serdar Acir** ([serdar\\_acir@dell.com](mailto:serdar_acir@dell.com)) is a performance development engineer and advisor on the System Performance Team at Dell. His area of specialization is I/O subsystem performance for servers and storage systems. He worked on several industry performance benchmarks such as TPC-C<sup>®</sup> (Transaction Processing Performance Council Benchmark C), TPC-W<sup>™</sup>, and SPC (Storage Performance Council). Serdar holds an M.S. in Computer Engineering with a specialization in Distributed Computing from Duke University.

#### FOR MORE INFORMATION

**Storage performance analysis and benchmarking standards:**

<http://www.storageperformance.org>

**I/O interface standards:**

<http://www.t10.org>

**Drive interface standards:**

<http://www.t13.org>

**IOmeter benchmark tool:**

<http://developer.intel.com/design/servers/devtools/iometer/index.htm>

**PERC 3/QC Release 1.2:**

<http://www.dell.com>