I/O and Storage Tuning An Introduction to I/O and Storage Tuning Tips and Techniques

Randy Kreiser
SGI
12200-G Plum Orchard Road
Silver Spring, MD 20904
Tel: +1-301-572-8926
FAX: +1-301-572-3280
rkreiser@sgi.com

Computational power is getting cheap. Thus, it can be argued that the real cost of computing today lies in reliably storing, and rapidly moving, big data. This article will introduce some principles and methods with which to fine-tune I/O and storage in RAID (Redundant Array of Independent Disks) storage systems.

The existence of an important but unrecognized or under-appreciated RAID capacity/performance relationship should be noted here. It is common to recommend a hardware size for a system predicated on the predicted capacity required by the operation. However, users have performance requirements that must often supersede their capacity requirements. For this reason, in correctly analyzed and sized data storage configurations, one will often find more physical system capacity than is strictly required by the I/O and storage workload. The extra capacity is not unnecessary overhead; it is capacity needed to fulfill both the users' storage requirements and the users' performance requirements.

The statements above are expressed in the RAID hardware as:

- Capacity requirements dictate the number of RAID luns needed
- Performance requirements dictate the number of disks needed per RAID lun

With those important considerations in mind, we move on.

Traditional highly available RAID technology provides redundant disk resources in a number of disk-array configurations that render the storage system more available and improves reliability and performance. Each level of RAID offers a different mix of performance, reliability and cost. Which level of RAID to use is completely dependent on the individual situation. No single RAID level is best for every situation. However, five of the most commonly used configurations are:

Level 0: Striping. The various disks in the array each get portions of a file, which is reconstructed upon retrieval. In this way, RAID 0 is similar to XLV striping in that it stripes the data across all the drives but doesn't offer any parity or redundancy. Thus, in the event of a failed drive, all data across the stripe is lost. Advantage: faster access due to parallel operation of the accesses. Disadvantage: if there is a problem with one of the disks, no data is accessible.

In addition, RAID level 0 striping is done on the RAID hardware. XLV does the striping in software. This distinction is important, as some competitors conduct their RAID parity calculations via software on their CPU's, not in the RAID array.

Level 1: Mirroring. In this simple hardware-mirroring format, all disk contents are mirrored on another disk in a simple primary/secondary relationship. Usually done in conjunction with RAID level 0, this guarantees security and performance.

Advantage: parallel I/O requests can be fulfilled simultaneously.
Disadvantage: data storage costs are doubled.

Level 3: RAID 3 is supported in disk configurations of 4+1 and 8+1 due, primarily, to the architecture of most of the RAID controllers (Clariion and Ciprico) that support RAID 3. Parity is contained on one drive, with the data drive heads accessed in lock-step sequence. This promotes extremely high bandwidth to large, sequentially accessed files such as image, graphical, video and satellite data sets.

The performance-price paid for the extremely high bandwidth produced by the physical configuration (single parity drive and multiple head movements) is that RAID 3 will service 3-4 threads of concurrent I/O's very well, but will chock on 8-9 I/O threads. It's also the case with RAID 3 that random I/O's or smaller, more numerous I/O's will degrade performance.

Level 5: In RAID 5 all the drives operate independently. RAID 5 is good for reads, for small I/O's, for many concurrent I/O's, and for random I/O's. Thus, its characteristics are just the opposite of RAID 3 characteristics.

In RAID 5 the parity blocks are distributed across all disks, together with other data. RAID 5 spreads the parity among all of the drives, but within one single, physical I/O it writes parity to only one drive. The next physical I/O writes the parity to a different drive, thus rotating parity. Simultaneously, the data blocks are being written to the other drives which make up the RAID 5 lun. Thus, it might appear that parity disk bottlenecks would be minimized.

However, any advantage or efficiency gained would be offset by the RAID 5 parity and data distribution calculation on writes. Writes are particularly demanding for RAID 5. To offset this write-performance degradation, memory (cache) can be added to each of the storage processors (SP). The amount of cache is dependent upon the number of disk drives owned by the SP plus the size of the I/O's from the application, the number of concurrent I/O's from the application, and the mix or reads versus writes.

Physical Parity Needs Summary:

RAID 0 requires 9% extra space for parity because it doesn't offer any parity.
RAID 1 requires 100% extra space because it is simply direct hardware mirroring.
RAID 1/0 space requirements are almost identical to RAID 1 because it, too, is basically a mirroring system.
RAID 3 requires 20% extra space in a 4+1 configuration, and 11% extra space in an 8+1 configuration.
RAID 5 can run 3 to 16 drives, so the extra space required is calculated as:
1 / Total # drives in lun. So a 15+1 RAID 5 lun would need 6-1/4% extra space.

RAID levels 1 and 1/0 need 100% more space. RAID level 3 needs 11% or 20% depending on the configuration. RAID 5 needs 6-1/4% to 33% depending on the configuration.

40/30/30 Rule

Fine-tuning the RAID system begins with recognition of the 40/30/30 performance rule.

The 40/30/30 performance rule states that 40% of the performance it's possible to extract from the system is within the hardware set-up; 30% is found in the system software, and another 30% reside in the application software.

This document will concentrate on exploiting the 30% of the fine-tuning opportunity to be found within the application software. After analyzing the application to reveal the characteristics of the application, answers regarding the remaining 70% (40/30) should become more clear as well.

Analyze the Application

Inspection of the application reveals valuable data about the characteristics of the I/O load under which the application is operating. Specifically, the overview will reveal the following broad characteristics:

- Large or small I/O's
- Sequential or random I/O's
- Number of concurrent I/O's
- Percentage mix of Reads and Writes in the I/O
- Direct, buffered, or raw I/O to the filesystem/volume. That is, the method (calls) used within the application code (program) to actually execute the reads/writes/opens, etc.

A more detailed analysis of the application will reveal:

- Transaction I/O size and type (Fixed and large are easier)
- An indicated RAID level to use (5, 3, 1_0, or 1)(An instance of the 40% rule)
- Number of disks to use in each single RAID lun (lun = logical unit number)(4+1 versus 8+1, mirror, etc.)

- Write caching or write buffering (Caching is battery backed-up and protected, write buffering is not protected)
- Cache page size (transaction I/O)
- Percentage mix of reads and writes
- Number of concurrent I/O's
- Whether the I/O is network-based or local
- Whether the data to the filesystem/volume is raw, buffered, or direct. That is, the method (calls) used within the application code (program) to actually execute the reads/writes/opens, etc.

Of these, it has been common to find that the two most important considerations are:

- Sequential versus random I/O
- Raw, direct, or buffered type of I/O to the filesystem/volume

The aggregate weight of the above parameters should define the size of the RAID luns and the write caching parameters to use.

Application Analysis Applied

1. The number of concurrent I/O's bears a direct relationship to the RAID level chosen. Because there can't be one RAID lun for one application and another RAID lun for some other application, the choice must be made between optimizing the environment for fewer, larger I/O's or for more numerous, smaller I/O's.

   It follows from this that transaction size is an important factor in selecting the RAID level to use: small I/O's (<32K) are classically linked to RAID 5, RAID 1, or RAID 1_0 luns. Large, sequential I/O's (>256K) are classic to RAID 3. I/O's between 32K and 256K fall into a gray area and decisions are application-dependent.

2. RAID 3 is good for sequential I/O's, but probably not for more than 3-4 concurrent I/O's. RAID 3 heads are locked together and step out across the drives together to write sequential I/O very well. However, RAID 3 heads don't perform random I/O well and are particularly slow for random writes.

3. Large Sequential I/O is best suited for direct I/O to the filesystem. Near raw performance and the benefits of having a XFS filesystem can result.

4. RAID 5, RAID 1 and RAID 1_0 are generally the best choices for random I/O and for more than 70% of read-based applications.

Observations about RAID

1. Creation of an optiondrive (creating one giant partition encompassing the entire disk drive) is recommended.

2. Disk thrashing will probably be produced by:

   A. Creating an external xfslog on the same RAID lun when “fx’ing” RAID. Doing this will cause the heads to ‘Ping-Pong’ from the extreme inner portion of the drives (when updating the data in the xfslog) to the extreme outside of the drives (to continue writing or reading). (“fx” is an interactive, menu-driven disk utility that creates partitions sizes, disk drive parameters, and writes the volume label of the device.)

      (The author wrote a script that eases the “fx’ing” part. It allows one to “fx” in one execution; all RAID attached to the system as an optiondrive. The script will also perform the command tag queuing (CTQ) setup at the same time. Once “run_fx” has executed, it documents exactly what was just done and, if executed in ‘query mode’, provides configuration documentation regarding the partition setup on all luns. Part of the ‘rktools’ tolls set, the script works on Fibre RAID and JBOD and SCSI-2 RAID and JBOD.

   B. The creation of more than 3 concurrently active partitions on the same lun is not recommended. It doesn’t matter how many partitions are created, the criticality lies in how many partitions are accessed simultaneously. Having more than 3 partitions active simultaneously can cause disk thrashing and thus, poor I/O performance.

More Observations about RAID

3. Always enable CTQ when “fx’ing” RAID. While setting the appropriate CTQ depth is very important when using threaded or buffered filesystem I/O, CTQ’ing is not applicable when the I/O is single-threaded.
   CTQ depth = 256 / Total # luns owned by the DPE (Disk Processor Enclosure).

4. It is monumentally important to select the appropriate stripe unit size (whenever possible, select an even stripe width I/O) and lun interleaving when creating XLV striped volumes. For instance, if there are two busses with four luns per bus, interleave on the volume element line. (XLV devices provide access to disk storage as logical volumes. A logical volume is an object that behaves like a disk partition, but its storage can span several physical disk devices. XLV can concatenate disks together to create larger logical volumes, stripe data across disk to create logical volumes with greater throughput, and plex (mirror) disk for reliability). It is also the case that with an XLV striped volume having multiple luns; each lun could have its own XLV thread of I/O. For instance if you have an XLV striped volume made up of 4 luns, then the stripe group = 4. If you create even stripe width I/O’s an application I/O will fit evenly across all four luns in one physical I/O. This will create up to 4 threads of I/O one to each device. If you had two application threads doing this, you would have two application I/O threads feeding multiple XLV I/O threads.

5. Calculation for even stripe width I/O. Application I/O size = (# of luns * stripe unit). Thus, if the XLV stripe lun group is 4, and the stripe unit is 2048 blocks, the applications' I/O size = 4MB.

Two Ways to Scale I/O

1. Use even stripe widths from the application program.

2. Use more threads of I/O from the application program. Threads could be from the use of posix threads (a set of IEEE standards designed to provide application portability between Unix variants) in the application program, or they could be from running more application program processes.

Underlying Ur-Truth

Until the saturation point is reached, creating more sustained I/O will produce the best overall I/O results.

Summary

Despite, or because of, the massive proliferation of data and our ability to stockpile it in terabyte quantities, many data-intensive operations have trouble quickly and efficiently accessing the information they need. Such operations need to maximize data retrieval, optimize throughput performance, and enhance the performance of both I/O and storage systems. Fortunately, correctly tuned RAID I/O and storage systems can significantly enhance the availability, reliability and performance of the data storage system without significantly increasing the overall system cost.