Pump up array performance

http://storageMagazine.techtarget.com/magItem/0,291266,sid35_gci1157671,00.html
by: Ray Lucchesi
Issue: Jan 2006

To get the best performance from your array, you need to tune it to match its workload.

AN I/O REQUEST STARTS at a host running an application or a service, travels through layers of its operating system to a host bus adapter (HBA) and then hops through the SAN fabric until it reaches the storage subsystem. When the storage subsystem responds, this course is reversed. This circuitous route offers plenty of performance-killing bottlenecks, as well as opportunities to optimize the performance of your storage subsystem to provide better service for all I/O commands.

I/O performance optimization activities for SAN-attached arrays can occur at the following three distinct times:

- Before the subsystem is purchased when choices about drives, interfaces, cache and data replication features are made.
- At configuration time when choices are made related to RAID level, RAID segment and stripe size; which drives to use to support a RAID group; which LUNs to assign to a RAID group; as well as cache options for the subsystem, RAID group and/or LUN.
- At runtime when the subsystem services its assigned applications, choices can be made or modified for cache options, LUN-to-controller affinity, application-to-LUN assignment and LUN-to-RAID group assignment.

WORKLOAD MONITORING IS AN ONGOING ENDEAVOR THAT SHOULD BE DONE MULTIPLE TIMES BEFORE AND DURING THE LIFE OF A SUBSYSTEM

I/O workloads

Subsystem performance tuning starts with knowing your hosts' I/O workloads. I/O characteristics such as typical I/O transfer length, read-to-write ratio, random-to-sequential I/O ratio, I/O operations/sec, and megabytes read/sec and written/sec need to be monitored and recorded. All data recovery and availability requirements must also be considered. Workload characteristics should measure activity at the I/O subsystem and not just what's driven by the application; a host's buffer can often capture some I/O activity and never present it to the subsystem. Workload characteristics can also change once a subsystem is tuned to support it. Workload monitoring is an ongoing endeavor that should be done multiple times before and during the life of a subsystem.

Workload monitoring tools are available from a number of vendors. For open systems, EMC Corp., Hitachi Data Systems (HDS), IBM Corp., Sun Microsystems Inc. and other vendors offer performance monitors as standalone products or bundled with their storage products. For mainframes, resource measurement facility-based products provide performance data. Some databases, such as Oracle and Microsoft SQL Server, offer special-purpose performance monitors. Operating systems such as AIX, HP-UX, Mac OS X, Red Hat Linux, Windows NT/Server 2003 and others have special-purpose performance/activity monitors that can display this sort of information on a host basis.

Some storage resource management (SRM) applications such as Computer Associates (CA) International Inc.'s BrightStor, Hewlett-Packard (HP) Co.'s AppIQ and others have features or components that provide workload performance statistics. Lacking these tools, open-systems administrators can use iostat and/or sar commands to gather device statistics on a device-by-device basis for a single host. Gathering and merging this information from all hosts generating a workload may produce sufficient results for workload measurement.

From a subsystems perspective, most workloads exhibit a time-dependent mixture of activities; however, primary workloads generally fall into the following three dimensions:

SEQUENTIAL VS. RANDOM:
- Highly sequential workloads require a high degree of read and/or write activities.
- Highly random workloads demand a high number of read and/or write operations/sec with no perceived pattern in the block numbers accessed.

THROUGHPUT VS. TRANSACTION:
- High-throughput workloads access large blocks of data and push multiple megabytes/sec or gigabytes/sec to or from a subsystem.
- High-transaction or low-throughput workloads demand a relatively high number of small blocks of data from a subsystem.

LOW VS. HIGH READ-TO-WRITE RATIO:
- High read-to-write workloads demand a relatively low amount of write activity from a subsystem.

Most workloads can be characterized within these three dimensions. A video production house, for example, might have a high-sequential, high-throughput workload that's characterized by a low read-to-write ratio (e.g., lots of video being written). On the other hand, a video server environment may exhibit similar needs for high sequential/throughput, but a relatively high read-to-write ratio (lots of video being read). Database workloads are more complex. Indices may be considered low throughput and high random, and the read-to-write ratio can be high or low depending on update frequency. Database table access varies between write and read based on the size of rows, the frequency of updates and the amount of ad hoc queries (sequential vs. random).

Pre-purchase consideration

Array performance optimization begins when you're deciding on configuration options and before a purchase is made. Let's start with drives.

Disk drives come in a number of interfaces (Serial ATA [SATA], Serial-attached SCSI

Tune-up tips for arrays

1. Split a "hot" subset of LUNs across controllers and multiple RAID groups.
2. Some applications are cache friendly (highly sequential) and some aren't (highly random), and can adversely affect each other when run on the same subsystem. If available, use partitioning to isolate applications and increase performance.
3. Make sure there are enough Fibre Channel pipes between the host(s) and the subsystem to support the workload.
4. If you have a high-throughput workload, your host bus adapter transfer size should match or exceed the LUN's segment size.
5. Array performance optimization begins when you're deciding which subsystem configuration options to buy.
6. High-random with low-throughput workloads can get by with today's SATA drives because transfer rates aren't much of a concern.
7. Spreading the LUNs across a number of RAID groups or a larger RAID stripe size (and therefore more spindles) yields more balanced activity against all drive spindles.
8. For highly sequential, high-throughput and high-write workloads, disabling write-back caching (enabling write-through) can improve performance. In write-through mode, write data bypasses the cache and goes straight to the subsystem's disk.

[SAS] or Fibre Channel [FC]), capacities (gigabytes) and speeds (rpm). Storage arrays typically include drives with three or more capacity points (73GB, 146GB and 300GB) and at two or more speeds (10,000 rpm or 15,000 rpm).

High random I/O workloads need high rpm drives. These drives typically offer more I/O per drive spindle than lower rpm drives. High-sequential workloads can probably get by with slower rpm drives because sequential I/O doesn't seek as much. For high-throughput workloads, high rpm drives may offer higher MB/sec transfer rates than lower rpm drives, but this often isn't an issue because even slower drives have enough bandwidth to sustain typical workloads. Faster drives typically cost approximately 50% more than slower drives at similar capacities and interface types, so price may be an additional factor to consider.

Today, FC rules for high throughput and any highly sequential workloads where data transfer is a dominant activity, although some SATA vendors may quibble with this statement considering the new SATA 3Gb/sec transfer rates. High-random with low-throughput workloads can get by with today's SATA drives because transfer rates aren't much of a concern. However, highly random workloads need faster drives that are available only with FC. Some drive vendors have recently introduced fast drives supporting SAS and FC, so it won't be long before fast SAS drives show up in storage subsystems.

Capacity is mostly irrelevant to I/O performance, but the number of spindles isn't. Highly random workloads can take advantage of more spindles because much of the I/O time is drive seek activity), so if you have the option, buy more spindles. For highly sequential workloads, the number of spindles isn't a concern because sequential I/O requires minimal seek activity. In any case, you must ensure that the set of spindles can support your minimum throughput requirements.

Most I/O subsystems offer configuration options from two to 100 or more front-end interfaces. This is mostly an availability and connectivity concern for highly random workloads, but may be a real concern for high-throughput workloads. Those types of workloads will have high MB/sec rates and you may need to overconfigure host paths to sustain the workload.

SOME CACHE SETTINGS MAY BE SPECIFIED DURING CONFIGURATION BUT MOST CACHE STEETINGS SHOULD BE LEFT TO DEFAULT VALUES AND THEN TWEAKED AT RUNTIME ONCE ACTUAL I/O WORKLOADS CAN BE OBSERVED

Cache is another important pre-purchase consideration because it can have a major impact on I/O performance. Highly sequential workloads require lots of cache because that's where most of the data will be read. Cache read-ahead algorithms try to predict I/O requests by pre-staging data into cache. However, cache is a zero-sum game; when something goes in, something else gets bumped out. The more cache you have, the more pre-staged data is maintained and performance is improved by not having to fetch data from disk. But cache memory isn't cheap and should be increased only if the application requires a larger cache to run properly.

High write activity, for example, may require more cache because with some controllers write activity may be mirrored between clustered controllers and cached before being written back to disk. These options will be discussed more later. Before purchasing an array, you must determine if you'll use write-back caching and/or write mirroring, and then size your cache accordingly. In addition, on some high-end subsystems, asynchronous remote mirroring of I/O activity consumes additional cache. On midrange systems, remote replication takes less cache, as data and meta data are typically held on disk only while being replicated. Sometimes onboard replication of data (point-intime copies) uses cache to retain updates while the copy is active.

Configuring LUNs and RAID groups

Most I/O subsystems map LUNs to specific RAID groups, and the RAID level you use to support those LUNs affects performance. Also, in most midrange systems LUNs are typically assigned to a preferred controller in a cluster of controllers, although this can change during runtime. If there are LUNs reserved for specific controllers, dividing LUN I/O activity across multiple controllers can balance the controller workload.

Most subsystems support several RAID levels. RAID 5 and RAID 1 are typical, and RAID 5+0 (or 50) and 1+0 (10) may also be available. The choice of RAID level is usually based on the workload's read-to-write ratio and randomness. Although the write penalty for RAID 5 keeps getting smaller, it still exists. For most highly random, write-heavy workloads, it's probably wise to use RAID 1. For sequential workloads with low or high write activity, RAID 5 will suffice because sequential writes have less of a write penalty than random I/O. Throughput isn't a big factor when choosing a RAID level as long as the stripe can sustain the workload. However, cost is a factor to consider: RAID 1 consumes twice the physical drive capacity to support a LUN. RAID 5 can require anywhere from 8% to 33% more physical drive capacity to support a LUN, depending on the stripe size.

Some subsystems let you select the RAID segment or chunk size. For high-throughput workloads, if you have a predictable transfer size you can set the RAID segment size equal to or greater than your transfer size and gain some optimization of I/O performance. RAID stripe size is another parameter that some subsystems allow you to configure using RAID x+0 (or x0) levels. The +0 indicates that a LUN can span more than one RAID group. For example, you have two RAID 5 groups with a three data and one parity drive configuration using RAID 5+0 with two RAID groups; this requires six data drives to support this LUN. However, one RAID 5 group with six data drives and one parity drive can support it equally as well. On some subsystems, a RAID group is spread across all spindles in a subsystem so it is, by definition, an x+0 RAID level. Larger stripe size helps when dealing with "hot" LUNs. Hot LUNs exhibit more I/O activity than the typical LUN in your subsystem. By spreading the LUN's workload across a number of RAID groups or a larger RAID stripe size (and therefore more spindles), you get more balanced activity against all drive spindles.

Runtime performance tuning

Some cache settings may be specified during configuration, but most cache settings should be left to default values and then tweaked at runtime once actual I/O workloads can be observed. Some subsystems have cache settings that can be assigned to LUNs, some have cache settings only for the subsystem level and others are somewhere in between. The cache options discussed later in this article may not apply to a specific LUN, but may need to be specified for higher LUN groupings instead.

Write back vs. write through

There are a number of ways to tune the subsystem cache to optimize I/O performance. For high-write workloads, write-back caching can be an effective technique to optimize performance. For highly sequential, high-throughput and high-write workloads, however, disabling write-back caching (enabling write-through) can improve performance. In writethrough mode, write data bypasses the cache and goes straight to the subsystem's disk.

Another cache option is write mirroring, which keeps controllers in a cluster pair in synch with respect to the disk images they maintain. Write mirroring can impact performance for high-write workloads because all write data has to be transferred to the other controller's cache. There are also data availability implications to disabling write mirroring that need to be considered.

For a highly sequential workload, read-ahead caching is a must. For these workloads, it's important to specify a read-ahead cache amount that's roughly equal to the data transferred during the time it takes to perform one disk-read operation (for 15K rpm FC drives at 2Gb/sec, this could be 1.5MB to 3MB). Most highend subsystems optimize this value in real-time, taking into consideration current I/O workload characteristics and cache size.

A few subsystems let you specify how much cache to devote to write activity vs. read activity. These settings aren't necessarily intuitive, but start with your overall subsystem's I/O workload read-to-write ratio and use the read percent vs. the write percent as a starting point. A better approach for most workloads may be to let the subsystem handle this dynamically, as write activity can be highly time dependent (normal work vs. backup activity).

As discussed previously, LUNs may be mapped to controllers and as you monitor your I/O workload, there may be some controller bottlenecks. Once you discover which LUNs are causing the problems, move them to the alternate controller to balance the workload. Some subsystems do this automatically.

In addition to write mirroring, there are other data availability, recovery and subsystem feature issues that impact performance. Many subsystems offer some form of copy-on-write technology that copies a block when it's written to retain replication. This may cause every write to invoke multiple I/Os, so using this feature will impact write performance.

SOME APPLICATIONS ARE CACHE FRIENDLY (HIGHLY SEQUENTIAL) AND SOME AREN'T (HIGHLY RANDOM). WHEN RUN TOGETHER ON THE SAME SUBSYSTEM, THESE APPLICATIONS MAY HAVE AN ADVERSE EFFECT ON ONE ANOTHER AND SLOW PERFORMANCE.

Asynchronous remote replication may consume cache on high-end subsystems and back-end bandwidth on midrange subsystems. With high-end subsystems, write data is typically retained in the cache until it's copied to the remote location. For midrange systems, data is quickly flushed from the cache and then read back only as it's being copied to remote subsystems. So for midrange systems, remote replication doesn't consume lots of cache, but it does produce additional back-end I/O activity. In either case, anything that consumes cache or bandwidth may impact overall subsystem performance.

Using synchronous vs. asynchronous remote replication can cause additional performance hits. For synchronous mirroring, the original write I/O operation is held up until the data is copied to the remote location. Depending on distance and other network factors, this can take a considerable amount of time for each write operation. For asynchronous mirrors, the original I/O isn't held up.

As discussed, some subsystems mitigate the problem of "hot" LUNs automatically or via x+0 RAID group levels. If your subsystem doesn't do this or you haven't used x+0 striping, you may wish to monitor LUN activity to see if there's a particular subset driving a majority of your I/O. If that's the case, splitting the "hot" subset of LUNs across controllers and multiple RAID groups often yields better performance.

Some applications are cache friendly (highly sequential) and some aren't (highly random). When run together on the same subsystem, these applications may have an adverse effect on one another and slow performance. Some subsystems can be partitioned to isolate counterproductive workload combinations. The partitioning splits one physical subsystem into two or more logical subsystems that can be dedicated to support a specific workload. Some subsystems do this by dedicating a portion of cache to a set of LUNs. Others partition by splitting up the entire subsystem--cache, processors and data paths. Yet another method is to split the workload across multiple subsystems.

In some cases, the fabric in front of your subsystem may be a performance bottleneck.

For heavy throughput workloads, make sure there are enough FC pipes between the host(s) and the subsystem to support the workload. And remember that HBA parameters need to be in sync with your workload. If you have a high-throughput workload, its HBA transfer size should match or exceed the LUN's segment size. If the HBA transfer size is below the LUN's segment size, the workload won't perform as well.

There are many ways to improve the performance of storage subsystems. Of course, some subsystems may offer more alternatives than presented here, and some array controllers automatically optimize some aspects of subsystem performance such as how cache is configured. Sophisticated caching algorithms have existed in the mainframe arena for many years and for at least a decade for open systems. The bottom line for performance tuning is knowing your application requirements.

***Ray Lucchesi** is president of Silverton Consulting and since 2004 has helped startups to Fortune 500 companies improve storage product development and marketing. Ray has also helped end-users better select and use storage subsystems.*
mailto:info@SilvertonConsulting.com
http://www.SilvertonConsulting.com