InterWorks 2002
THE HP TECHNICAL TRAINING CONFERENCE

Exploring Disk Size and Oracle Disk I/O Performance

White paper

September 2002

Table of Contents

Introduction

As processing loads grow and storage demands expand, a primary challenge for Database Administrators (DBAs) is to extract maximum performance from applications, especially where performance is dependent on disk I/O. Improvements in drive densities and spindle speeds have been impressive, but they have not kept pace with demands for ever-better performance.

Exploring Disk Size and Oracle Disk I/O Performance

The continuing debate around application performance (often measured in disk I/O) is the size and spindle speed of drives being deployed in an Oracle environment. The common perception is to optimize Oracle by deploying a large number of smaller capacity disk drives in the data base storage array. In most environments this perception is accurate. The move to larger member disk drives has been observed by authors to seriously impact performance. This has to do with a reduction in the number of disk members and therefore a reduction in the ability of the disk environment to service I/O requests. This reduction in member count reduces the ability of the disk subsystem to satisfy I/O by diminishing what might be called the "parallelism" of the drive environment. As storage requirements continue to grow exponentially, the drive capacity issue remains central to application performance.

The issues surrounding disk capacity/performance persist because of the notion that in random read/write environments more disks moving simultaneously is optimal (the "parallelism" mentioned in the previous paragraph). The notion of striping data (RAID 1) to initiate concurrent disk drive read/writes has been an effective means of generating disk performance gains. The notion also persists because of the fear that large capacity drives have longer latency periods as the drive heads locate desired disk positions.

The Great Debate
Those favoring smaller capacity drives argue:

a) Smaller capacity drives (9GB, 18GB) have been upgraded to match the spindle speeds (10,000 rpm) and faster access times of the larger disks.
b) Aerial densities (a product of track density and the number of bits per track) have exploded, so the time needed to access a given gigabyte of data has deteriorated. Densely packed bits can mean fewer disks for a given amount of storage to be located and retrieved.

The large capacity drive camp counter-argues:

a) Larger capacity drives (36GB, 73GB, 180 GB) are individually faster than their smaller brethren
b) Increasing aerial densities are more than offset by improvements in the hardware technology (processors, intelligent caching).

There are a number of issues that should be reviewed together to determine how disk capacity affects application performance.

Disk Technology Changes

Disk capacities have changed dramatically – storage providers have been supporting 36GB and 73GB drives for some time, and even larger capacity drives are expected in the future. The drive manufacturers have created faster spindle speeds (7200, 10K, and 15K rpm) and created shorter latency periods. Access times are down. Transfer speeds are up, and seek times are down. Please see Table 1 for some representative disk drive performance variables.

Disk Storage Footprint Impact	**1 Terabyte Example**	**Average Access Times**	**Internal Transfer Rate**	**Vertical Inches of Rack Space**	**Number of 9-Bay Rackmount Enclosures**
Disk Capacity	112x9GB	9.9ms	7-10MBs	112in (64U)	13
	56x18GB	7.7ms	11-17MBs	56in (32U)	7
	28x36GB	5.4ms	17-29MBs	28in (16U)	4
	14x73GB	5.6ms	26-40MBs	14in (8U)	2
	6x180GB	7.4ms	28-50MBs	7in (4U)	1

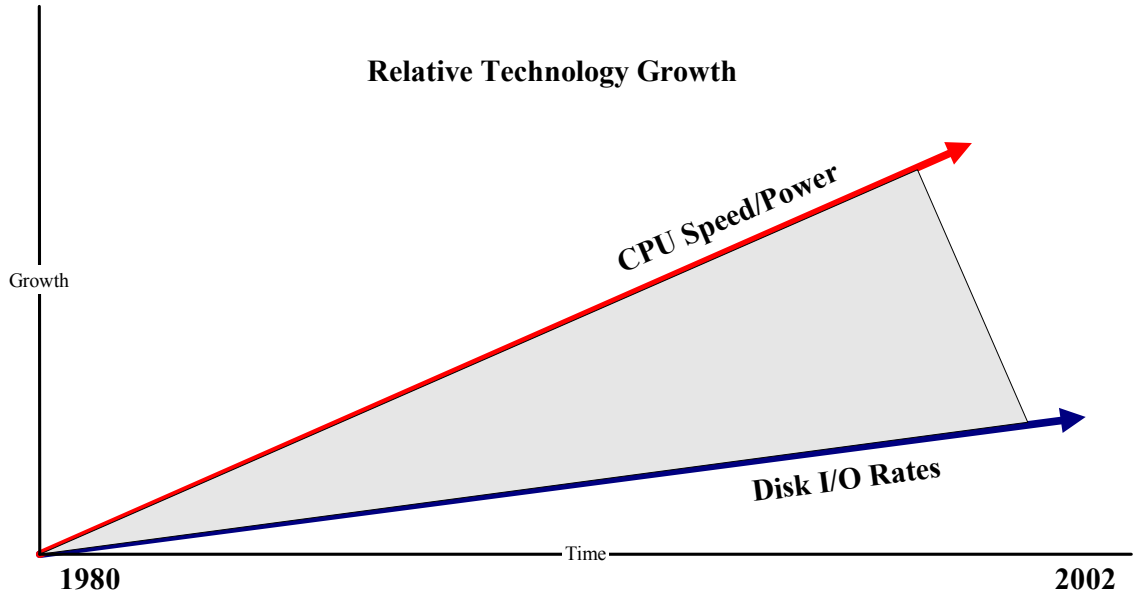
Table 1

The Fibre Channel and SCSI camps continue their respective arguments about which general technology class is best for the user. While fibre drives have some inherent speed advantages (in terms of throughput), fibre drives can also be considerably more expensive.

Completing disk I/O instructions involves several variables including disk access or seek times, read/write times, and transfer rates. While transfer rates have increased dramatically, the greatest percentage of the disk I/O time is still related to the seek times (as in moving the heads to find the data). The seek times can consume 70%-80% of the clock time needed to complete the disk I/O. It is these seek times that storage systems providers are attacking with improved hardware technology (in the server and the storage controller).

Changing Processor Technology

In the Graph 1 (below), the upper slope represents the rate of relative *rate* of change in CPU technology while the lower slope represents the relative *rate* of change in disk drive technology (including spindle speeds, capacity, and access times). The area between the two curves represents time needed to complete the I/O activity. Note that the curves are *diverging* and more time can be spent by faster and more powerful CPUs waiting for the I/Os to complete. As disks have become bigger and faster so has the potential for the server(s) to wait for the disk drives to complete their I/Os – especially in interactive applications (e.g., database queries).



Graph 1

Server manufacturers have achieved huge gains in the performance of the processor, the presence of multiple processors, and the availability of large memory buffers in the server (for staging the I/O instructions). You can consume your entire IT budget stuffing memory into your servers.

Application Content – the Secret to Understanding Storage Performance

Application content is most often the critical issue in resolving disk performance. Application content refers to the read/write mix as well as the random versus repetitive content of the application. All applications are not created equal. Some applications are write intensive while others are read intensive. Users should know their application content mix to assist in selecting their storage array.

The process of determining application activity is fairly simple. Most operating systems contain the necessary utilities, such as SARs (system activity reports), IOSTATs (I/O statistics), or performance monitors that present the application activity at a given point in time. These statistics are more meaningful when derived during known periods of application intensity.

The majority of applications are not random in content (which would benefit from several small capacity drives). While much of the raw data may in fact be randomly stored, there are many repetitive functions present in the database application (e.g., look-up tables, log files, temp files). Some amount of time in the database application is always spent in repetitive disk activity.

Caching Technology

The most common approach for compressing these two technology development curves (to minimize wait states) has been for server and storage providers to load the hardware

with cache. In addition to improved server processor technology, more memory has been introduced – on the disk drives, in the storage controller, and in the server. Most often the memory in the hardware configuration is implemented as a FIFO buffer for staging the I/O instructions.

How the memory scheme is implemented can be more important in dealing with the application content than simply stuffing the hardware with a huge (and perhaps expensive) buffer. More recent advances in memory platforms have included intelligence that can actually reduce the actual frequency of disk reads/writes.

Managing the I/O Cycles

Who manages the disk I/O cycles? The CPU or the external controllers? Storage providers continue to add intelligence (in the form of software programs and search algorithms) to the RAID controllers. The net effect can be the migration of the I/O management from the server to the external controller (thus reducing the number of CPU cycles needed to manage the storage array).

Some storage providers have integrated Solid State Disk (SSD) and RAID technologies to provide the best of both worlds (safety and performance). In recent years, some RAID controllers have emerged with cache that "adapts" to changes in the application read/write activity. One example of such an "adaptive" cache is a retentive scheme that holds frequently used read/write data in memory rather than going to the disk drives for each I/O – and does so *without* user intervention.

System Reliability and Economics

The original impetus for RAID was to add data protection in the event of misbehaving disk drives by writing the data in multiple locations. If a given disk misbehaves, then a hot spare disk can rebuild the data stripes so there is no loss of continuity in operating the system.

The growth in disk capacity has provided the opportunity to re-visit the basic economics and safety (price, footprint, and Mean Time Between Failure (MTBF)) of the array.

As databases (and the related number of disk drives needed to store the data) continue to grow, the older notion that a lot of little drives is questionable from the simple point of reliability. Given newer, larger capacity drives are faster than older, smaller drives, the issue of reducing the number of active devices in a system also provides for improved MTBFs - the long-proven logic of system reliability dictates that a system with fewer components will produce longer MTBFs as there are fewer devices to misbehave.

Cabinet footprint has also become more of a pressing issue as users find their facilities space outgrown (cabinet footprint size, weight, and heat). In addition, co-location companies may charge their clients in terms of floor space or rack space consumed. Please see the above drive capacity table. A terabyte of 18GB drives consumes seven 9-bay racks while a terabyte of 73GB drives consumes two 9-bay racks – a savings of 35 vertical inches (or 20 U's) in the rack, not to mention the savings in heat generation.

Relieving the Management Burden of Solid State Disk (SSD)

Extracting Performance from Interactive Applications
Interactive applications are those characterized by a fair amount of activity between the server and the disk storage array (versus transactional or compute-intensive applications). Application performance issues can exist in several areas of an application, but the common bottleneck is disk access.

SSD is one of the technologies available for addressing the data storage (disk) I/O bottleneck. SSDs are functionally similar to rotating disk drives but are built instead from high-speed memory chips (no moving parts).

Much has been written about the virtues of SSD technology because the age-old problem of disk I/O contention continues as the prevailing issue affecting storage system performance in interactive applications. The root cause of the performance problem has not only continued but also in fact has grown worse. This underlying performance issue that is manifest at the application level is due to the fact that continuing developments in computing technology (e.g., faster processors, multiple processors, tons of local memory, etc.) *still* outpace the relative speed of technology change in accessing data on the hard disk drive (see Graph 1).

Clearly, the application environment is not so simple as to explain the "waiting" times as the simple difference in hardware technology changes. Changes made in the application itself can have a significant impact on performance degradation or performance improvements.

A number of different and interacting variables are at play in the storage system that affect throughput, and some pundits (and plenty of vendors) declare that path to be best addressed by faster servers, multiple processors, more server memory, larger pipes (e.g., fibre channel) or SSD technology.

Many of the recent articles on SSD technology stress the raw horsepower of solid-state electronics versus rotating media but miss the primary point for the user – how to extract the desired performance gains with the minimum grief in implementation and management of the SSD technology.

The Management Burden
The benefits of SSD technology are most clearly realized when the application "hot" files are isolated and migrated into the SSD engine for execution. There is little argument that silicon executes far faster than rotating media, but there are several challenges affecting data migration to the SSD. The most important issues are:

a. Who should do it (isolate the hot files),
b. How to do it (applying hardware or software tools), and
c. How to track the (constant) application content change?

The management burden is primarily one of isolating the "hot" files. The user has to investigate which data are being accessed most often in the application and also determine how often those access frequencies change. In an Oracle environment, how often the user initiates specific functions (e.g., updating the Re-Do logs, changing the look-up tables, or performing an Insert function) directly affects heightened disk I/O (and potential performance degradation).

Many operating systems contain utilities and performance monitors for identifying peak load conditions, but resources (IT staff) are needed to conduct those investigations and then decide how to change the application or re-direct the disk activity to minimize I/O bottlenecks.

That burden is accentuated by the fact that the performance reference points constantly change. In addition, when the user is running multiple applications, the process of isolating hot files is further complicated. The SSD engine can provide much needed performance gains if the users can isolate and migrate the active data into the SSD in the first place.

SSDs in various form factors, implementations, and price points

SSDs have been offered by various vendors in a range of implementations, from stand-alone appliances to form factors using disk drive canisters (for easy insertion into the storage array). Pricing ($$ per gigabyte) varies widely from the vendors, but in general has seen a steady decline in last couple of years. That gradual price erosion has made SSD performance benefits more interesting to an ever-expanding user community hungry for performance.

The issue with SSD form factors and pricing lies not in the simplicity of connecting the SSD to the server or the cost of acquisition but in the ease of implementing the SSD resource in the application and the subsequent management of that performance engine.

Pareto's Law Lives

Users should have some notion about the nature of their applications to know if SSDs can provide a cost-effective performance benefit. All applications are *not* created equal, but vary wide in content. Applications types include:

a. Interactive (e.g., mix of disk reads and writes)
b. Transactional (e.g., processing intensive)
c. Sequential (e.g., read intensive or streaming)

Given the nature of the primary application, SSDs may or may not be an effective performance engine. Examples of interactive applications include database queries, email, and document imaging. Interactive applications are also characterized by a fair amount of repetitive activity.

From the user's perspective, the issue lies in identifying "what's hot" and "what's not" at any point in time as well as knowing how those conditions change. "Hot" data or "hot" files are those most frequently used. But most application content is in fact far from static. What's hot this morning may not be hot this afternoon and depends entirely on what is happening in the application. In addition, many users run several applications on the same servers and storage systems concurrently, which can quickly complicate the task of isolating the most active data.

Pareto's Law (the old 80/20 rule) simply means that in an interactive application, 80% of the time the user is accessing 20% of the same-old data. In a database environment, the more frequently used variables include look-up tables, pointers, indices, etc. that have to be repeatedly utilized to access the raw data. Even those users who suspect their applications are highly random in content are surprised to find out how much of the disk read/write activity repeats.

How To Tell If SSD Technology Can Help

The first order of business is to separate the CPU performance from the disk storage performance in order to evaluate the potential benefit of the SSD. Most operating systems include utilities such as Input/Output Statistics (IOSTAT)s, System Activity Reports (SARs), or Performance Monitors that provide hard metrics about the application activity. The system administrator needs to know the operating system command (to enter at the system prompt) to generate these statistics.

Three of the more critical performance indicators are:

a. CPU latency (often expressed as a percentage)
b. I/O wait times (usually expressed in milliseconds)
c. Individual disk busy percentages

Running the system utilities while the application is active will generate these metrics that identify how busy (or how sleepy) the CPU is, how long the server is waiting for I/O instructions to complete, and which disk drives are log-jammed holding up the process. SSDs can provide a significant performance bump under these conditions.

As a rule-of-thumb, when CPU latency is 60% or more, wait times are 60 milliseconds or more, and disk busy percentages exceed 60%, the user can rest assured the gating issue is disk I/O contention. Likewise, if the CPU is busy 60% of the time, wait times are shorter, and disk busy percentages are well under 50%, the performance grief tends to be CPU-driven (rather than the disk drives).
Likewise, SSD technology is less effective under these circumstances.

User Options

At the hardware system level, the traditional brute force approach to seeking performance gains has been to add faster servers (processors) or more memory in server or controller, or faster disk drives (10,000 rpm and 15,000 rpm spindle speeds). These approaches can sometimes be effective, are often expensive, and do not always provide the means for managing the application content change.

At the application software level, the user has several options available including diagnostic programs (to identify the application bottlenecks), changing the application software, developing test programs, and documenting the changes – all burdensome forms of hand-tuning the application code.

Some of these approaches can be effective, but all require a fair amount of management resource in an effort, which does not guarantee that significant performance gains will result. It is not uncommon for performance investigations and code changes to consume many months of IT staff resource. Thus, it is easy to understand system administrator reluctance to tackle such a challenge at the expense of not attending to other system management responsibilities.

The Ease of Implementation

SSD technology is now available that assists the user in quickly generating significant performance gains with a minimum of user effort. Several vendors employ various forms of write-back caching to tell the server the I/O task is done. Such algorithms are implemented in the storage controller, which is used to manage the disk I/O in background while the server is preparing the next action. Some storage controllers also use pre-fetch (read ahead) algorithms to move expected data from the drives into the controller cache for speedier execution of the I/O instruction prior to the service request from the server. Pre-fetch algorithms can be effective if the expected data (moved into the controller memory prior to the server request) matches the requested data.

An "adaptive" SSD platform that speeds reads and writes and also contains a secondary retentive cache space where a hierarchy of frequently requested and used data patterns reside. Transparent to the server operating system and application software, the caching platform offloads the management of the disk I/O from the server to the controller. The intelligent controller reduces the frequency of accessing the disk drives by operating on the I/O instructions (the frequently used data) in the caching platform.

With zero user intervention, the caching platform "learns" and retains the frequently used data patterns in the secondary cache space until more frequently used data bumps them out of the hierarchy.

Further, the caching platform is user-segmentable and can be easily changed by the user into one, two, three, or four cache spaces that can be equal or unequal amounts of the physical cache in the controller. If, for example, the user has multiple servers connected to the same controller and knows that one of the servers is more "active" than the others, that server can easily be assigned more of the cache space on the fly.

If the user does nothing, performance gains will be produced because the intelligent caching platform isolates and retains the "hot" files. The user also has the option of easily tailoring the cache sizes to their various application needs as they change over time.

Cost of Implementation Versus the Cost of Management
The most visible costs are those of acquisition. The user realizes the performance gains from the adaptive SSD platform at no additional out-of-pocket cost (over the cost of the storage system). The hidden costs associated with the burden of isolating the "hot" files are also minimized as that function is provided by the caching platform without user intervention.

The Changing Problem

I/O performance can be improved through software and hardware techniques. The traditional software approach has relied on DBAs tuning application software, specifically by indexing disk drive activity, balancing loads and changing application code. This approach fueled a robust consulting market and it can be effective, but the process is long, tedious and always requires thoughtful experimentation. In practice, lack of time and talent often makes this option unfeasible.

The traditional hardware approach to improving I/O performance is to adopt a SSD. A SSD is silicon that emulates a disk drive. Initially for mainframes and later used in server environments, SSDs increase performance by placing data in solid-state storage instead of on a disk. SSDs increase application performance, but at a very high price. SSDs are expensive to buy and require a DBA with accurate knowledge of which specific data should migrate from disk to the SSD. Since SSDs are not "adaptive," meaning their contents must be manually configured; the ever-changing data needs of applications require constant upkeep by an SSD administrator. Without full-time attention, the performance benefits of SSD disappear.

Hands-off Hardware
What if there was a way to increase I/O and improve database application performance that did not require in-depth analysis by DBAs or labor-intense solid-state memory? Is a set and forget storage appliance possible?

"Adaptive Cache" is a memory management technology used to intelligently manage cache contents and deliver exceptional performance for database applications, at a fraction of the cost of SSD. Adaptive Cache, without administrator intervention, adjusts its contents to retain the most frequently used information and maximize ongoing application performance.

Traditional Cache
Virtually every storage system contains cache. Cache is used in disk drives, servers and in RAID controllers- as little as 32 Megabytes to as much as 16 Gigabytes. Cached data can be retrieved 20 times faster than data on disk, but cache capacity is much more expensive than disk capacity. It is therefore important that critical data reside in cache and less important data reside on disk. This means that the size of the cache is less important than how cache contents are utilized.

Traditional cache is also called FIFO (First In, First Out) cache because it employs a Least Recently Used purging algorithm. This algorithm continuously places the newest or Most Recently Used data in cache and purges the oldest or Least Recently Used data. This approach is effective for buffering I/O, but it creates a problem called "Cache Pollution." Cache Pollution occurs when cache is filled with recently but infrequently used data. Cache Pollution displaces data that applications use repeatedly, but not recently. The undesirable result is that important data is remanded to slower disk storage, negating much of the performance benefit.

Adaptive Caching

Adaptive Cache is unique because it divides cache into two discrete segments, *Dynamic Space* and *Protected Space*, and intelligently manages their contents to maximize performance and eliminate Cache Pollution.

Dynamic Space and Protected Space

Similar to the operation of FIFO cache, Dynamic Space uses the *Least Recently Used* algorithm and serves as an I/O buffer. Unlike FIFO cache, Protected Space cache uses a *Least Frequently Used* purging algorithm.

The Least Frequently Used algorithm tracks the usage of each data block and promotes frequently used data from Dynamic Space to Protected Space. Data in Protected Space is accessed at cache speeds and cannot be overwritten by a simple stream of new data requests.

Protected Space is cache for "hot" data. Unlike a SSD, Protected Space is automatically monitored to keep it populated with the "hottest" data. As applications demand different data at different frequencies, Adaptive Cache repopulates Protected Space to keep it up-to-date. Figure 1 shows the hierarchy of Adaptive Cache.

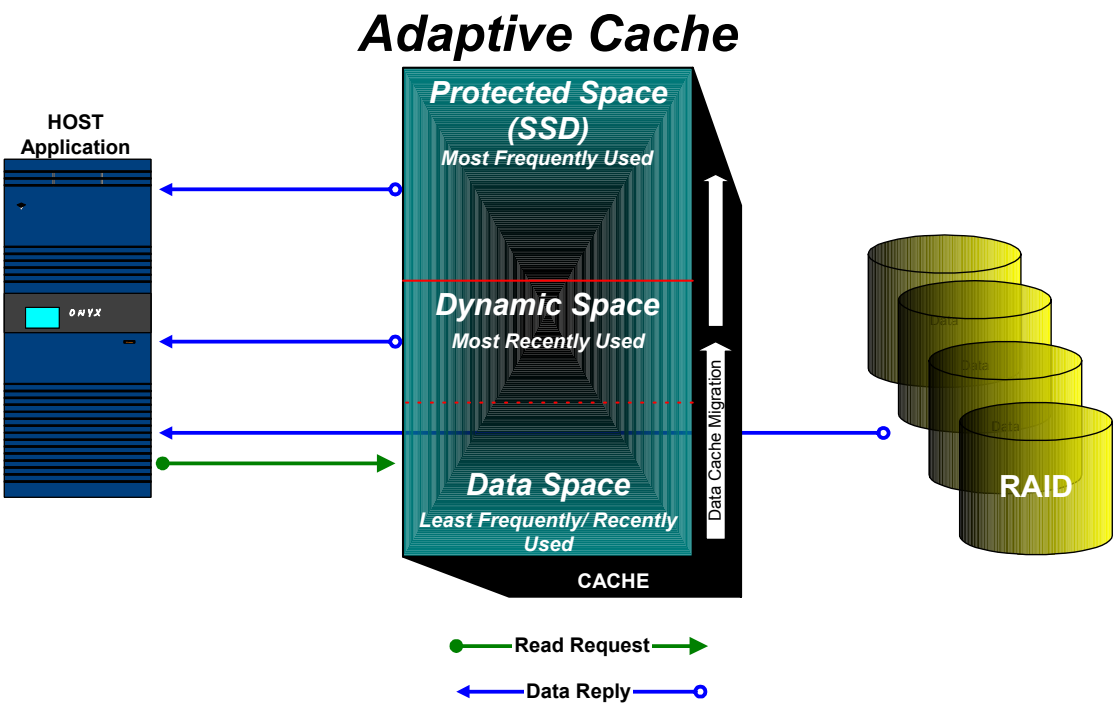


Figure 1.

The database application uses Protected Space cache just as it would RAID disk storage. In other words, Adaptive Cache requires no changes to code or databases design. By constantly monitoring activity and keeping critical data in Protected Space.

Other Acceleration Techniques

Moving critical data from disk to cache is the most important step to faster database application performance, but not the only step. Algorithms called *Pre-fetch* are used for disk reads. When an application requests data that is not in cache, Prefetch accesses the disk, reads the requested data, and then reads the data that immediately follows. The additional data is Pre-fetched because it is likely that the application will subsequently request that data from disk. By performing one large disk read instead of two small ones, Pre-fetch saves resources and pre-positions relevant data in cache for fast access.

If the application requests data that is not sequentially addressed, the Pre-fetch algorithm uses the addresses to ascertain the address of follow-on data. The amount of Pre-fetched data may be adjusted for optimal performance. OLTP applications use a Pre-fetch multiple of 2: for every block of data requested, an additional 2 Prefetched blocks are returned. Sequential operations use Pre-fetch multiple of 4 or 5. The pre-fetched data is placed in dynamic cache for the next read as represented in Figure 2.

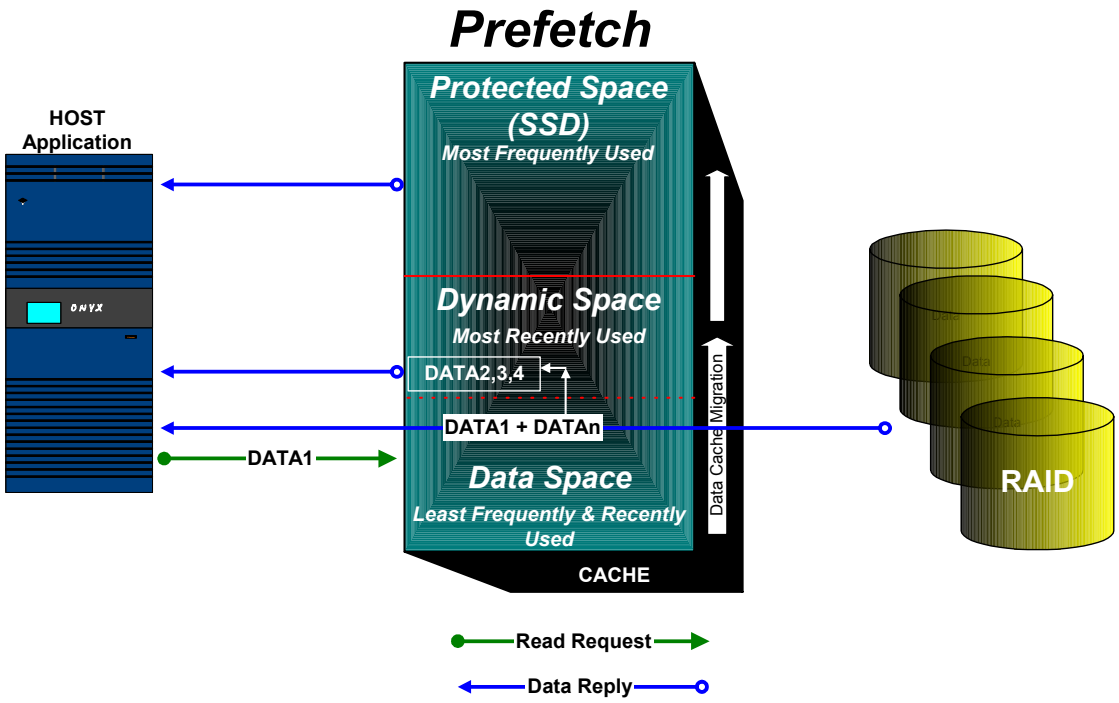


Figure 2.

For writes to disk, using Write Back cache improves performance. Write Back cache holds written data in a buffer, but signals the CPU that the disk write operation is complete. This frees the CPU to perform other tasks instead of idling while data is deposited on disk. To further accelerate disk writes using a Concatenation process also increases write performance. When an application writes data to disk a Concatenation algorithm sequences that data in cache based on where it resides on the physical disk. It then writes data to disk in fewer, larger, write operations. Every avoided disk write operation saves precious milliseconds of disk seek time, improving performance for end-users.

RAID Performance Trade-offs

Conventional RAID demands a performance versus efficiency trade-off. This trade-off is rooted in RAID Level 5, the most frequently implemented and commercially successful variant of RAID. RAID 5 makes efficient use of disk space but often uses four distinct write operations to put data onto disk. This is referred to as the "RAID 5 Write Penalty." This penalty consumes resources and impedes overall application performance.

A technique called the Block Pooling is used to avoid the RAID 5 Write Penalty. The Block Pool is reserved space on disks that functions as "virtual cache" for disk write processes. When data is to be written to RAID 5 disk, the RAID 5 Write Penalty is avoided by first writing that data to the Block Pool using swifter RAID 0+1. Then in a background process, relocates the data from the Block Pool to disk using the more space-

efficient RAID 5. The Block Pool eliminates performance-robbing "backpressure" on the application from sluggish RAID 5 disk write operations. Figure 3 shows the interaction of Concatenated Writes, Write Back Cache and the Block Pool.

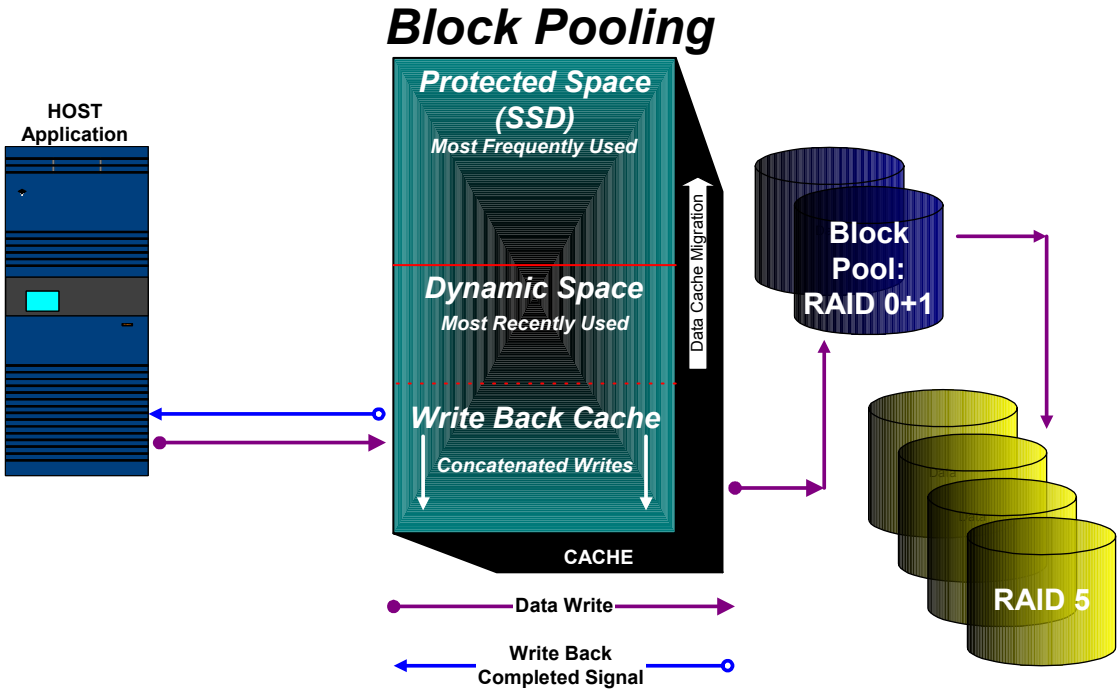


Figure 3.

The Prime Beneficiary: Database Applications

Interactive applications (e.g., database queries) enjoy the most impressive performance gains from Adaptive Cache, Pre-fetch, Concatenated Writes, Write Back Cache and the Block Pool. Critical database files are moved into Protected Space cache to increase I/O. Commonly cached files include Temporary TableSpace (or Workspace), transaction Log Files, and heavily used Indexes and TableSpaces. Temporary TableSpace is where intermediary processing occurs. Complex queries that repeatedly write and update data create intense write activity to TempSpaces. Table Sorts, Updates, Joins and similar commands are also taxing on TempSpace.

Transaction Log files can be very I/O intense with a high percentage of disk writes. Log Files are where the database records its transactions so it can perform a Rollback to previous states, should that be necessary. How these files are used varies by database engine, but queries using commands such as Rollback, Begin, Save, and Commit are heavy users of the Transaction Log files.

These techniques increase database and application server CPU utilization by offloading I/O tasks to the RAID controller and decreasing aggregate disk I/O processes.

Better Performance Is Your Bottom Line

Improvements in drive densities and spindle speeds have not kept pace with demands for higher performance. Until now DBAs had to invest in analysis and administration (people) or expensive capital equipment that requires more administration (more people).

Caching technologies gives DBAs a storage solution that increases database application performance and obviates the need for SSD administration or frequent in-depth analysis.

A combination of caching, cache management and Block Pool technologies make it an easy way to improve database application performance.

The Bottom Line
While both camps, in the disk size debate, continue their arguments and individual data points, what matters most in determining performance is the application content.

Given repetitive and intensive disk activity in the application (confirmed in the IOSTATs), an optimal approach lies in a combination of intelligent controller cache and large capacity drives. Not all application content is created equal (in its randomness). New generation large capacity drives are faster than their smaller (older generation) brethren. The presence of cache in the storage system sub-assemblies can provide substantial performance gains but *how* the memory platform is implemented (FIFO buffer versus intelligent or "adaptive" cache) can have greater impact on application performance.

Users should use the available system utilities to know their application content. Vendors should know which mix of storage technology best fits the users' application. Users should first use their operating system utilities (e.g., I/O statistics, performance monitors) to confirm their performance bottlenecks are more disk related than compute intensive. SSDs are more effective performance engines for intensive disk I/O environments.

Users with interactive applications that are considering SSD technologies as a potential performance engine should also investigate *how* the SSD is implemented. Users should also evaluate what "hot" file isolation and management resource, if any, is needed by the user to realize the performance benefits of the technology.

Conclusion
The move to larger and larger disk members can be problematic due to the diminished impact of parallelism in accomplishing I/O. However, a number of methods exist that help alleviate the potential performance issues involved and the use of larger member disk drives. With in-depth knowledge of the application and the time and ability to use available utilities the application can be tuned to better performance. Additional memory certainly helps performance while placing a higher demand on the management of the memory provided by the systems processes. The SSD solution, with the Adaptive Cache is potentially a very productive solution that should be considered when optimizing Oracle Applications for ideal performance.