HP INTERWORKS 2000 PAPER PRESENTATION

Understanding and Optimizing Disk I/O - Strategies, Tools, Hardware, and Applications or Winning over Disk I/O worries

By Jeff Kubler
Kubler Consulting, Inc.

Introduction

As CPU power and speed increase, with the advent of new and faster processing chips, the importance of fast access to data increases. Strategies that help the disk controllers quickly retrieve and pass needed data to processes requesting them are vital in making process throughput faster. Strategies for the reduction of disk I/O is important in reducing the time processes require to complete. A proper understanding of these strategies is vital in promoting fast response time.

This paper will first generally discuss disk I/O, I/O strategies, tools used in enhancing I/O, hardware used to better (by enhancing efficiencies) or make I/O more secure, and applications that can be used to improve I/O. The paper will attempt to leave you with different strategies for "Winning over Disk I/O Worries" to help you have the keys for success in this arena.

Mike Loukides said of disk I/O "This is the single most important aspect of I/O performance." From System Performance Tuning By Mike Loukides O'Rielly & Associates, Inc.

The general issue in helping your disk environment become as efficient as possible is the issue of data locality. The term "Data Locality" describes the location of data on disk (it is sometimes referred to as locality of reference). Data can be located in more or less efficient ways that can help make disk I/O more or less efficient. Data Locality encompasses both the issue of the placement of files on disk or on multiple disks and the issue of records within the files placed on disk. The many different relational databases make any generic conversation difficult. The lack of system available commands capable of analyzing the internal structure make a specific conversation difficult. Thankfully, many of these database structures come with their own analyzer which helps provide guidance when optimizing them for greater efficiency. What general

information we can provide about the whole area of relational database will be presented as part of this paper.

Some general comments about disk efficiency: it is usually good practice to spread the disk workload as evenly as possible and across as many disk drives as possible. Also, provided disk technology is the same, smaller drives and more of them is usually better. Another generally accepted dictum is to get as may controllers as possible, in fact a one for one relationship is great (one controller per disk drive) however, usually too expensive to implement.

What is disk I/O?

The first question to be sure we have a general understanding of is the question, "What is Disk I/O?" Disk I/O is the act of retrieving and/or updating information stored on a disk drive or in a disk environment.

We know that all system activity begins with a process. In the life of a process data is usually one of the key elements needed for completion of that processes purpose. The required data may reside in one of two places, either in memory or on disk. If the data is in memory it had to be moved there from disk at one time and if changed will eventually need to be posted back to disk. In the illustration that follows we show how the disk is involved in serving the needs of the process. Disk will go out and either read or update data depending on the needs of the process. Since disk I/O is the slowest part of any function we always seek to avoid as much disk I/O as possible, but it is still central to most processes. Disk access is usually thought to be at least 7 to 8 times slower (could be 100 to 1000 times slower) then memory access. Therefore, we can conclude that more main memory is always good and helpful in improving performance. How you use that memory and how it is allocated among buffer caches and general memory is a question for another paper.

Anatomy of a Process

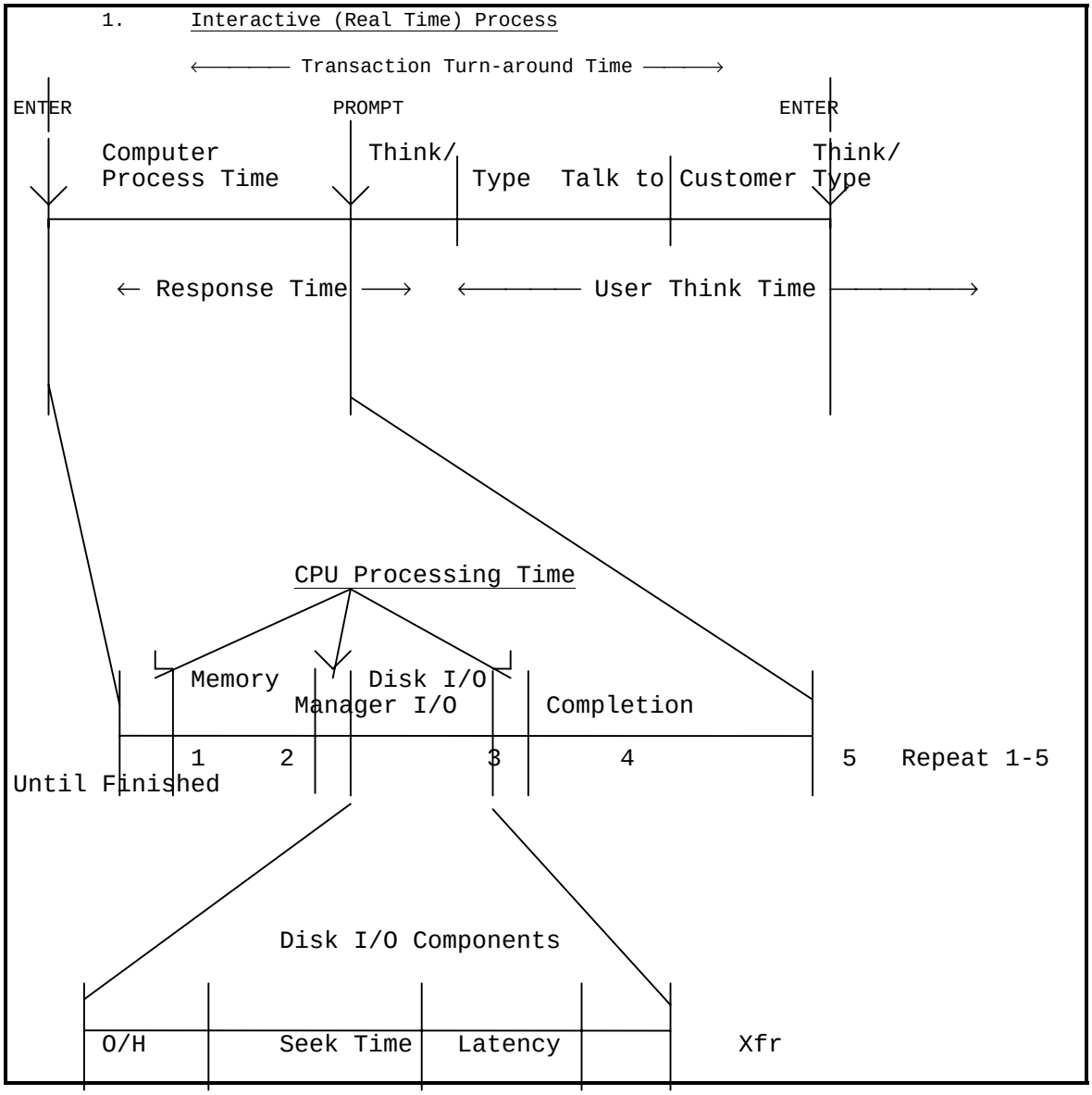


Figure 1 - Anatomy of a Process

In the illustration, “Anatomy of a Process” we see broken out several components of disk I/O (under “Disk I/O Components”). These components make up the time it takes for disk I/O to complete: Overhead - the general action of talking to the disk controller, Seek Time - the time it takes to find the correct track on disk, Latency - the time it may take to arrive at the requested data once the correct track is located (disk drives are spinning so there is a possibility that you will not be over the requested data when the track is located), and Xfr - the transfer of data to the controller and back to memory. The ability to optimize disk activity to shorten the seek times, reduce latency times, and reduce the call for disk I/O is the goal of this paper.

If you miss getting all the data you have to wait until the entire disk revolves around again. This latency can significantly degrade the performance of the disk.

General Measurements of Disk I/O

Now that we have a general understanding of the tasks involved in moving data from disk to memory for use by a process we need to find some measured values that are good indicators of whether we have efficient or inefficient retrieval of data. Here is a list of important indicators you can use to evaluate disk efficiencies.

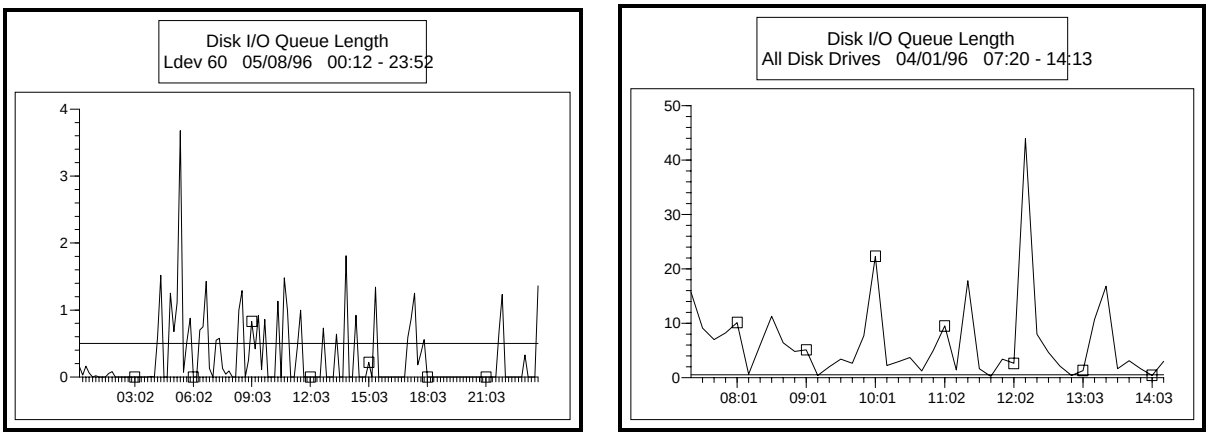


Figure 2 – Disk I/O Queue Length

Disk I/O Queue Length - Disk I/O queue length is the number of disk I/O requests waiting for disk access. When a process needs disk data a disk I/O request is sent to the appropriate controller. If another request is already being serviced the request will queue (line up) behind the previous request.

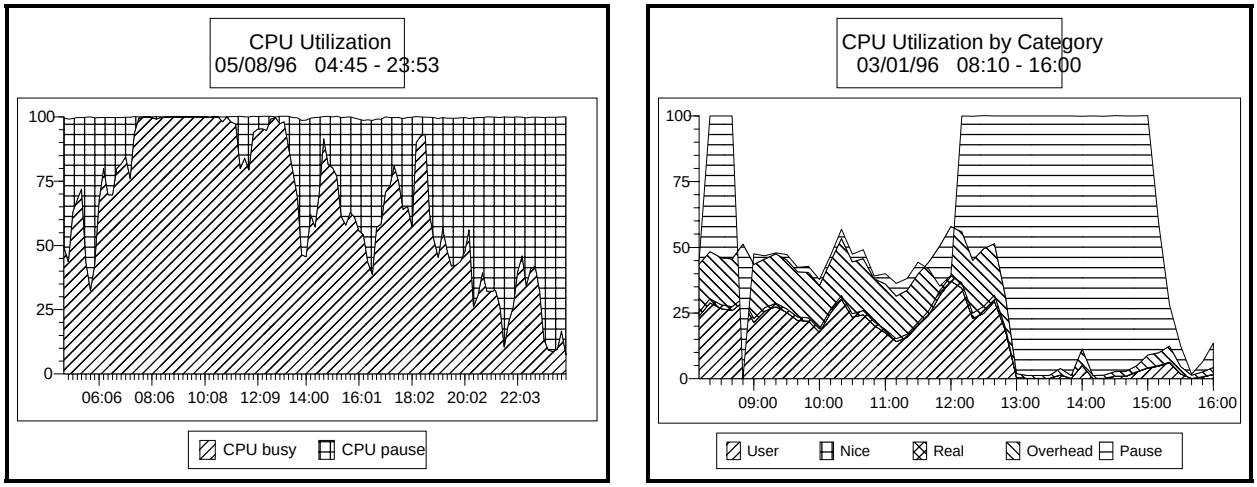


Figure 3 – CPU Utilization

Pause or Wait for disk - This is the percentage of a processes response due to waiting for disk I/O.

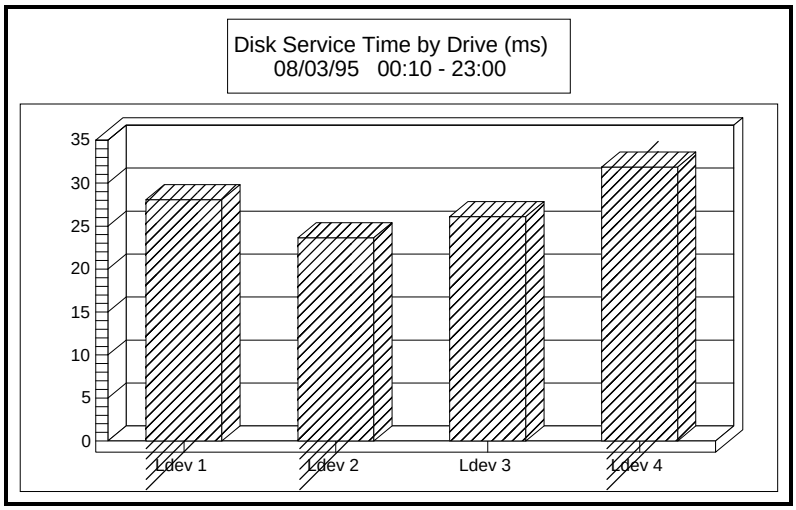


Figure 4 -- Disk Service Time

The disk service time is a measure of how quickly data is typically received for the average request. Different hardware types have a typical expected service time and when the reality is in wide variance from this expectation there are problems.

Other measurements of importance are:The disk utilization – this value tells, over an interval how busy an individual disk device is. Having a disk device operate above 20 or 30 percent means that the device is getting too much activity to be effective; The total disk I/O count -- may indicate that there are simply too many I/O's for the environment to handle; The buffer cache efficiency – these values (there is a read value and a write value) tell how much I/O is being reduced because a good percentage are being found in the buffer, therefore, disk I/O was not required.

Optimal Disk I/O

The best disk I/O is really none at all and some day there may be systems with so much memory you effectively avoid disk I/O. However, until that day, the best situation is to have as much memory as can be afforded or is configurable, the fastest disk I/O environment possible in view of other considerations such as High Availability, and a tuned application and database engine which avoids unneeded disk I/O.

Causes of Disk I/O Inefficiency

Generally, disk I/O inefficiencies are said to occur whenever I/O could have eliminated or reduced had the needed data been more efficiently located on disk. Sometimes performance must be sacrificed in order to ensure other goals are met such as high availability requirements and data integrity needs. However, even when choices are made to sacrifice some performance much can be done

to optimize I/O. A number of disk I/O issues cause the general I/O problems most sites encounter. Fragmentation, disk I/O imbalance, lack of disk space, database engine inefficiencies, poorly planned file systems, poorly configured systems with too little swap space, the use of networked file systems, blah, blah all conspire to make disk I/O inefficient. Before discussing these two quick asides need to be made, one is to discuss the question of data integrity Vs performance and the other is the importance of adequate memory.

Data Integrity VS Performance

An important question when thinking about disk I/O should begin your thought process on disk I/O optimization: “How important is the data I am storing?” Sometimes choices are made which actually make for a less efficient I/O environment simply because of a greater need for data integrity.

Memory Vs Disk

The symbiotic relationship between memory and disk needs to be kept in mind at all times. Several things about this relationship should be kept in mind: too little memory makes the disk I/O environment work harder, improperly utilized memory (too much or too little buffer cache) can also have a detrimental impact on disk I/O.

Fragmentation

Fragmentation is defined as the process of breaking things up into smaller pieces. A fragment is a part detached or broken off of the whole. More specifically fragmentation is defined as “The propensity of the component disk blocks of a file or memory segments of a kernel data structure to become separated from each other.” The greater the fragmentation, the more work has to be performed to retrieve the data.

In disk I/O terms fragmentation exists on two levels: file fragmentation refers to the tendency of files to be split into pieces or extents. This happens in order to make sure that a file can be built. The draw back here is that, for many types of I/O, more time is now needed to find and fetch all of the needed extents.

Disk Fragmentation

Disk fragmentation refers to the phenomena that occurs when the usage of the disk itself becomes very “patchwork” like. As files are built and purged the space they took up may not be readily usable and new files who use the space may need to be cut into many extents. The illustration below helps in understanding what fragmentation looks like from the disk fragmentation view.

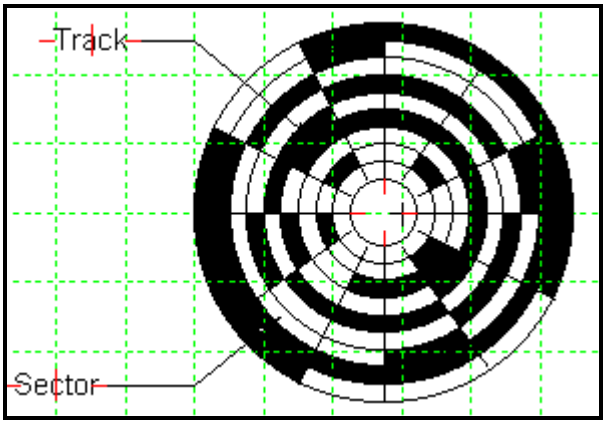


Figure 5 -- Disk fragmentation illustration

File Fragmentation

File fragmentation results from a disk that is quite full or from an already fragmented disk. In order for the file system to find space for a file it must break the file up in to many pieces called extents. Having multiple pieces of files results in a longer time being spent in disk seeks, more time in latency, and more disk transfers.

Frequent creation and deletion of files, and the deletion and extension of logical volumes may cause fragmentation.

Disk I/O Imbalance

Disk I/O imbalance refers to what occurs when one or more disk devices exist in an environment. If requested I/O ends up on one or more of those drives more than others you have an imbalance. This results in more pause for disk and higher disk I/O queue lengths as more requests contend for I/O on that device. This makes processes wait longer for requested information and results in slower response times. The figure below illustrates what a disk I/O imbalance looks like.

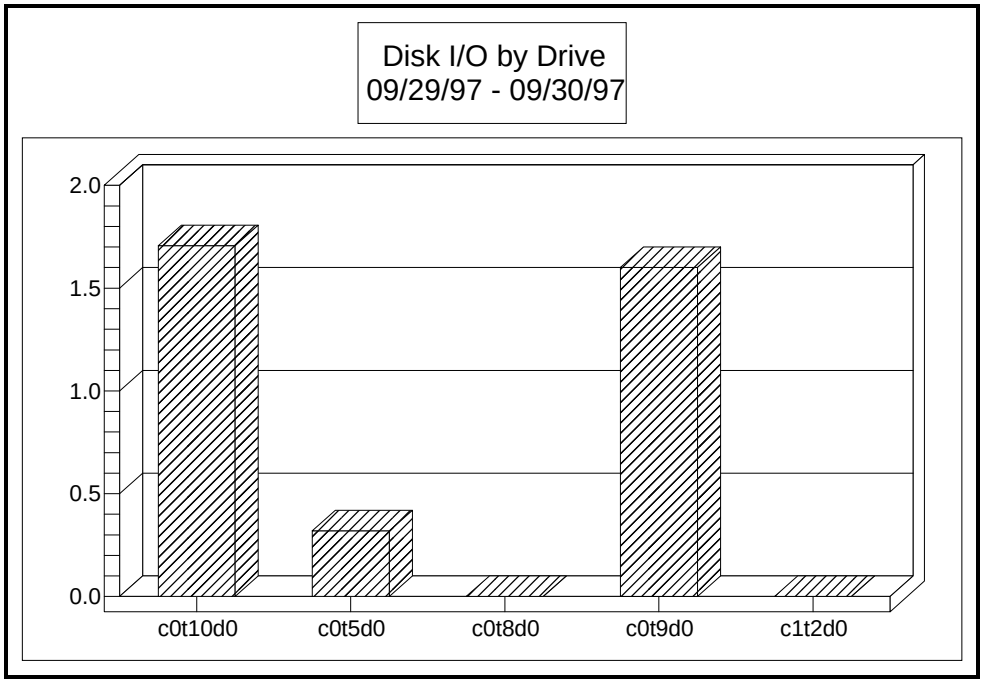


Figure 6 -- Disk I/O by Drive

The vertical axis in Figure 6 is the average number of I/O's per second measured over the interval.

Inadequate Disk Space

Adequate disk space is essential to any computer system. Monitoring disk space is always an important facet of system performance. Running out of disk space (either on a system wide basis or a file system basis) can cause your system performance to stop altogether or a lack of disk space can at least prevent certain programs and applications from executing. The df command will show used and available disk space.

File System Optimization

Several different files systems are currently available. HFS - High Performance File System, VxFS - Veritas File System (JFS or Journaled File System) and NFS - Network File System are usually the names that come to mind. The choice of file systems and the configuration options used in setting them up can be a performance issue.

The High Performance File system - HFS- is probably the least prone to cause performance issues due to configuration issues.

The Journaled File System (JFS) is generally seen as a good way to accelerate the speed of recovery should a system failure occur. This file system performs

an integrity check in seconds that can be favorably compared to the file system check 'fsck' offered with the standard UNIX file system.

NFS Diskless is the technology for sharing file systems and other hardware and software resources among a group of computers in one or more LANs. NFS places important data across the LAN and is thus open to many performance problems as now the performance of the disk drive and the network are important and potential problems.

Once the choice of file systems has been made the individual file systems need to be configured. The newfs command is used to create new filesystems. When creating file systems try to adhere to these goals: distribute the workload evenly,

- keep similar types of files in the same file systems, keep similar projects or groups together, give each filesystem a block size appropriate to the files it will contain, use as few filesystems per disk as possible, avoid the use of file system paging if possible.

Configuration issues

A system configured to too few controllers per drive may run into an I/O bottleneck as too much I/O tries to flood the channel. The environment for this situation would have to occur in a very heavy I/O situation.

The configuration of swap space can lead to another configuration problem. Swap space needs to be adequate (the usual rule of thumb is twice the amount of physical memory - a smaller percentage for larger memory systems) and should be spread over several devices. Inadequate or poorly planned swap space can end up putting a great amount of I/O on one or more disk drives.

Relational Database inefficiencies

In the UNIX world there are many different databases in use. A few examples include: ORACLE, INGRESS, and INFORMIX. The normal make for these relation databases usually includes the following files: TABLES - which hold the data; INDEXES - These provide faster retrieval of data; Rollback logs - containing the transaction information needed to take out a transaction if it is aborted (Also to allow the database to recover after a system or database crash); Before IMAGE LOGS (possibly redo logs - ORACLE or physical logs - INFORMIX) contain information used to reconstruct the database during recovery.

The issues here are that inefficient tables, inappropriately created indexes (or the lack thereof), improperly placed rollback logs or before image logs will hinder performance. What follows is a few quick bits of information to help understand relational databases and a few rules to help.

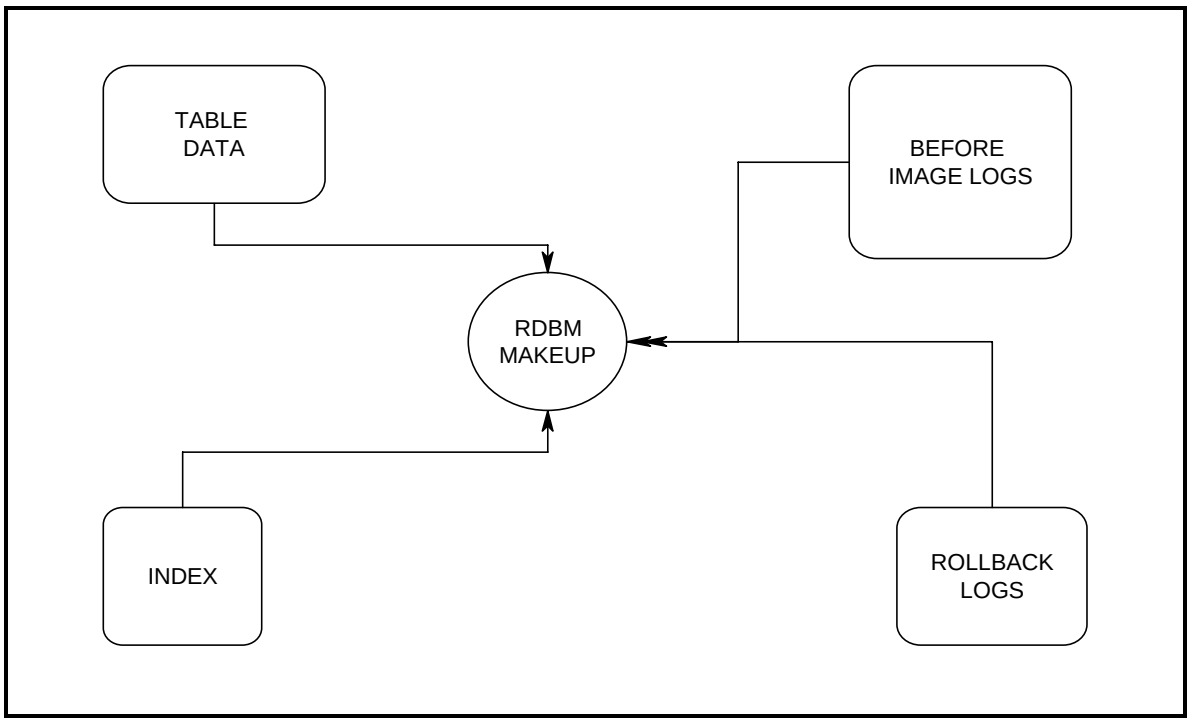


Figure 7 - Typical makeup of Relation Database

Most of the relational databases in use also have their own monitoring programs which allow you to monitor and tune the databases. Check with your database provider to find out how to use the appropriate monitor. Performing the needed housekeeping measures to the tables and indexes will help keep you database in high efficiency. To tune the disk environment for RDB usage try the following:

- Optimize placement of Tables and Indexes - RDBMS' work best with their data tables spread over as many physical disk drives as possible. This lowers the amount of disk head contention. Several small drives should be preferred to a single large drive.

To work toward the best locality use the following rules:

- Place table files, indexes, and logs on separate disk drives.

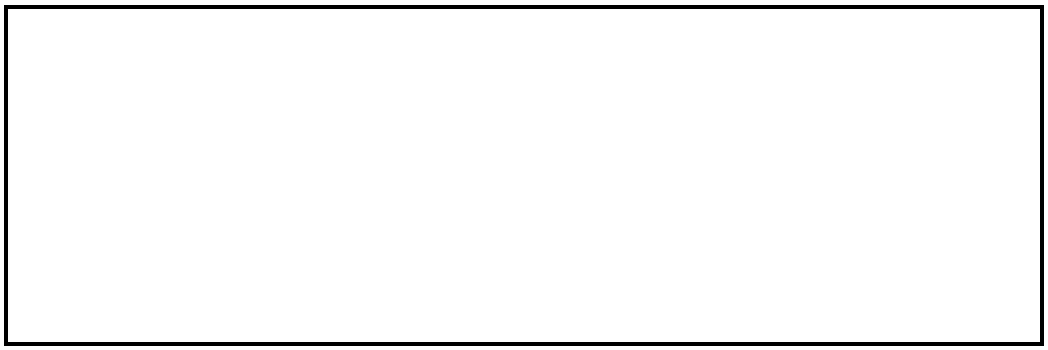


Figure 8 - Typical disk configuration for Oracle

Strategies

Many strategies exist in the area of disk I/O. Some are inherent in the operating system. Some can be configurable choices to either use and/or ignore. Some tend to either support data integrity or help the area of disk I/O. Others deal with specific processes that are helpful in optimizing specific database or file access.

Memory - memory is the scratch pad for all current work. Memory is checked to see if needed data is resident before going to disk, so any data that can be found here will help eliminate I/O. Part of this attempt is a part of memory called the buffer cache.

Buffer Cache - "The buffer cache is a pool of buffers that provides intermediate storage for data moving to or from the system's disk drives." System Performance Tuning, by Mike Loukides. Buffer cache size is of key importance. It is controlled by a min and max percent and too high a value can cause occasional problems, as demand for user memory can not be met. Too low will cause additional I/O.
Some strategies are made by choice. Here are some of the available:

JBOD - This stands for "Just a bunch of disks" and is the simplest and possibly the most straightforward way of understanding and working with disk I/O performance.

Striping- disk striping is another technique some suggest to increase per-process disk performance. With disk stripping the file system is placed across more than one drive. The files in the file system are also across more than one drive. This should allow disks to work in parallel. Disk stripping is effective for programs that stress disk I/O to large files.

File System vs Raw I/O - is favored by database applications as it bypasses the file system management routines. Reads and writes are made directly from memory to the surface of the disk.

"This is a topic that has been generating much discussion and confusion within the Oracle community. The general assumption that raw devices will always give you a 10-15% performance boost is not always true. Disks and OS file system technologies are changing rapidly and there are cases where OS level buffering could provide better performance than raw devices. Also, another factor to consider when deciding between raw devices and file systems is the administrative overhead of raw devices."

"The difficulty in deciding between raw devices and file systems is that it is not possible to predict in advance whether your database will benefit from raw partitions. Also, it is not easy to convert database files that are stored on raw devices to a file system and vice versa. Therefore, you should tune other parts of your subsystem first and only move to raw devices when I/O is still a

performance bottleneck.“ Oracle 7 Tuning Reference Guide, For OLTP Applications, Version 1.0, Design And Migration Services, Oracle Worldwide Alliances, March, 1997.

Tools

Database reloads and loads. All RDMS engines contain the ability to unload the current data, delete the table and re-add the information. Re-indexing follows and what you should end up with is a more efficient storage of data and a more efficient database. How often you perform this depends on the environment and the demand for faster access to the data.

System reloads - system reloads are an effective means to reduce or eliminate disk and file fragmentation. However, they can be costly in time and do rely upon good data preserved on tape.

Online JFS on-line defrag tool - HP OnLineJFS allows administration of a Journaled File System without taking it offline, eliminating the need for planned downtime. JFS expansion, backup, and disk defragmentation can all be achieved on-line using HP OnLineJFS.

Tools in the disk I/O optimization area consist of a number of commands: iostat - Reports I/O statistics for terminals, disks, and the system; Bdf - reports the number of free disk blocks; glance and sos - two commonly used and helpful monitoring tools; link - symbolic links allow for the movement of files and the creation of a directory entry pointing to the file on the new location; df command - reports the number of free file system blocks.

Hardware

The appropriate hardware is vital in the gaining the proper performance from the I/O environment. Regardless of what configuration/software/hardware is in use ,it is important to keep up with technology. Here are several different options in the hardware area:

Mirroring - This allows data to be protected by placing a copy of the data in two places. The big disadvantage here is that you have to have twice to hardware. This can be expensive as you must pay for more disk drives, controllers etc. The issue with mirroring is that there are now:

- 2 places to write to
- 2 places that can be read from.

If writing to a mirrored logical volume, LVM sends the request to all the physical extents in the logical extent in a parallel way. This duplicity causes some additional overhead. The system must process each request so there is a possible performance degradation. However, reads have two potential places to read from.

RAID Configurations - RAID (or Redundant Array of inexpensive Disks) is a group of disk connected to the same controller. There are a number of different configurations of RAID drives available. One issue with arrays is that they tend to be of large size. More I/O comes to these devices. Because of this they have a tendency to cause a bottleneck on I/O.

AUTORAID - AutoRAID is a high performance fault tolerant RAID subsystem that is said to require little or no knowledge of RAID to install and configure. AUTORAID disk array technology offers the fault tolerance of RAID but is said to be easier to use. AutoRAID Array Technology continually tunes the array for optimal performance and intelligently adapts to the workload.

Solid State Disk - are extremely fast storage devices that are designed to appear as a standard disk drive to your operating system. Instead of rotating mechanical heads and platters, they utilize solid-state memory (RAM).

SSA - SSA offers alow-cost, full-duplex, frame multiplexed communication interface for the interconnection of storage devices, storage subsystems, servers, workstations and PCs. SSA technology is said to provide superior performance due to its greater bandwidth for multiple, concurrent, full duplex I/O operations.

Large Cached Systems - A number of vendors combine a large disk farm with a large amount of memory to act as a cache of disk data. EMC is one of the most notable (others include SEEK Systems, and Imperial Technology – sorry if I have neglected others) describing their systems as “Integrated Cached Disk Arrays”. These devices are often seen as both potential disk I/O problem solutions and high availability solutions. For EMC, in particular, I have heard the value of 100GB’s of storage as a good decision point for selecting EMC as the high availability disk subsystem of choice. The others I have no “rule of thumb” to recommend their use but they can be great for highly active OLTP environments.

The following general information should be kept in mind in reference to disk hardware. The five most common disk interfaces are: single ended SCSI - this has a transfer rate of 5 Mb per second. These average about 15 to 20 milliseconds for disk service time; Differential SCSI - can transfer up to 10 Mb per second and the cable connection can total 25 meters in length; FAST & Wide SCSI - it gives a transfer rate of 20 Mb per second. These average 10 to 15 milliseconds per I/O; HP Fibre Link (HP-FL) - These average 30 to 35 milliseconds per I/O; HP Instrument BUS (HP-IB) - the transfer rate is no more than 1 Mb per second. These average 30 to 35 milliseconds per I/O.

The capabilities of each type vary in terms of their ability to retrieve data. The fastest are the Fast and Wide SCSI.
Several disk drives can be placed on one controller. The ideal here would be to have one controller per drive but cost usually makes this ideal impossible. As practical limit, you should limit your systems to 7 single ended devices per controller,15 fast and wide devices per controller, 6 HP-IB devices per controller

and 8 HP-FL devices per controller (of course having devices or as old a technology as the last two violates other rules - get ride of um!). Different vendors may have their own limitations gained from experience. You should ask them for their limitations!

Applications

Applications to aid disk I/O are found either from Hewlett-Packard as part of the available software, add on purchases or from third parties.

Logical Volume Manager - LVM is a tool for managing disk space with more flexibility than was available. Prior to LVM disk were carved out into contiguous partitions which made up file systems. Increasing the size of a partition meant that space on the disk had be found. LVM allows a file system to be spread across disks of different types. LVM - LVM is an additional layer of software between the file system and the device drivers. At file system creation, newfs parameters are read from /etc/disktab to obtain the best parameters for the new file system, based on the disk type. LVM allows a file system to be spread across disks of different types. The additional level needs to be processed so there is a performance penalty to pay for using LVM. However the increased flexibility in space usage and spreading out of I/O probably makes it well worth it.

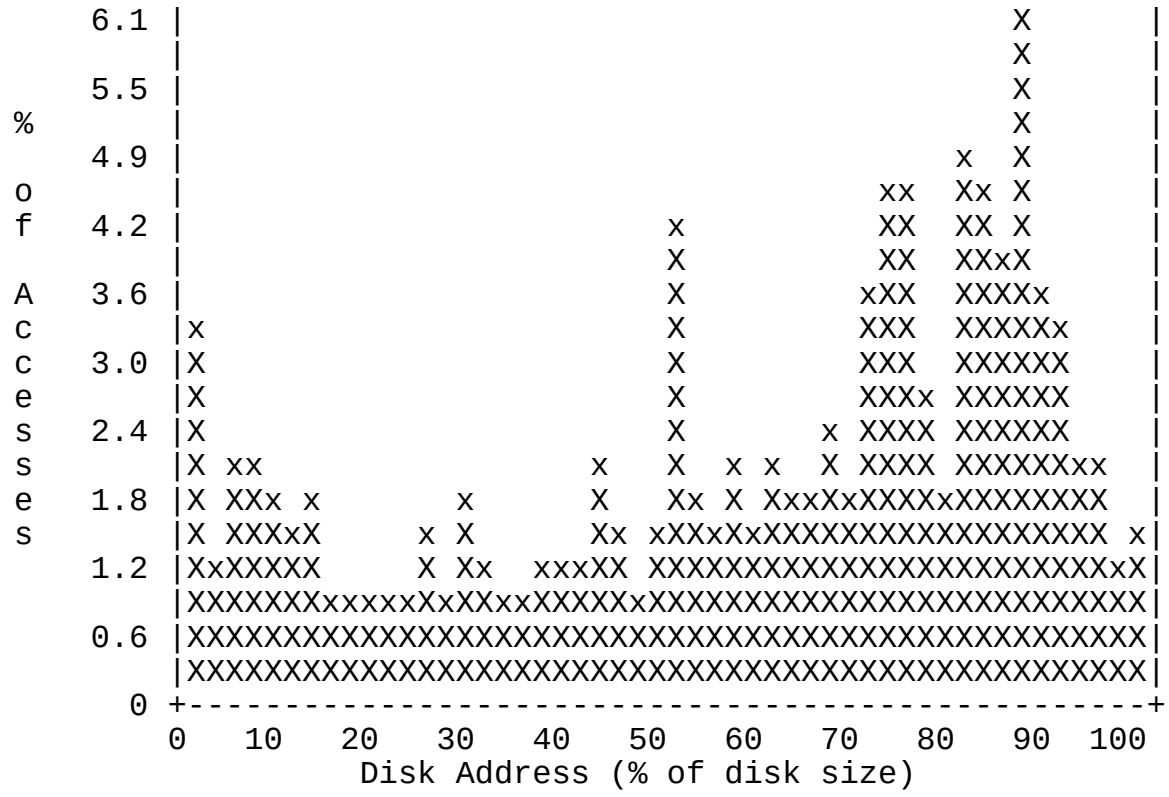
These two products are from EAGLE Software, Inc.
Diskpak - DISK_PAK and DISK_PAK OnLine! are file/file system defragmenters. They make each individual file contiguous, cluster directories together, and make free space contiguous. They should be run periodically (weekly/monthly) when the system is fairly idle.

With DISK_PAK, the file system had to be un-mounted during optimization. DISK_PAK OnLine! replaced DISK_PAK, and runs with the file system mounted. Both work ONLY on HFS file systems on HP-UX—not on JFS, and obviously NOT on raw partitions.

Seekrite - SEEKRite works with any file system (HFS or JFS), AND with raw partitions. It ISN'T a file defragmenter--it doesn't know anything about "files". It includes a device driver that is put in the data path (similar to LVM) to monitor how often each block of data is accessed. The disk can then be "optimized" by clustering the most frequently accessed blocks together to create an "organ pipe" distribution of accesses, to reduce average seek distance.

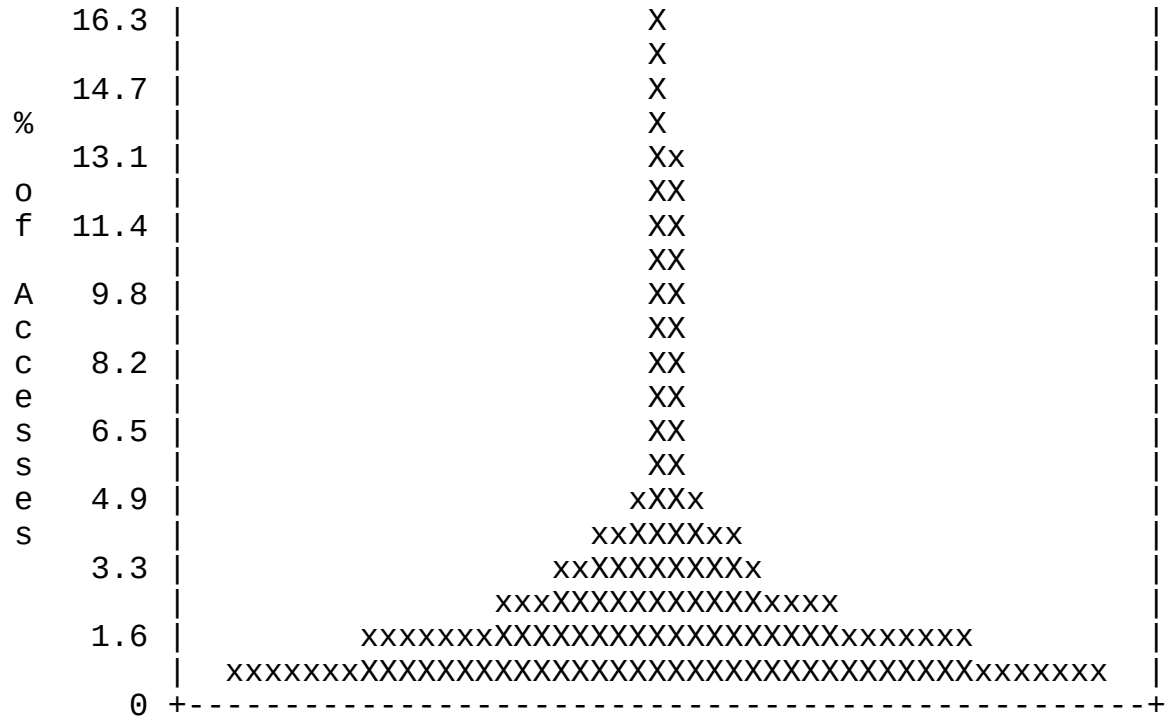
Here are actual histograms. First, before SEEKRite. It shows that accesses are widely spread across the disk, resulting in a relatively high average "expected average seek distance"--if the accesses occurred randomly, the heads would have to seek, on average, 28.7% of the way across the disk.

Access distribution without SEEKRite.
Expected avg seek: 28.7%



After optimization with SEEKRite, the most heavily accessed blocks have been clustered together, and the expected average seek distance has been reduced dramatically.

Access distribution with SEEKRite (optimized).
Expected avg seek: 16.5% Improvement: 42.4%



```
0   10   20   30   40   50   60   70   80   90  100
          Disk Address (% of disk size)
```

Number of chunks to move: 419255/420922 (99.6%)

The right application can save you a great deal of money in hardware purchases.

Solutions

Do you have existing I/O bottlenecks? Most will find that they do, and that they have been living with them for a long time. However, they tend to catch up with you so my recommendation is to be proactive. Manage these now to avoid the performance problems that will come your way as the disk I/O environment degrades and users demands grow. Here are a number of ideas to use in the process.

Optimize your databases and database access - Optimizing your database access usually involves making sure that, first of all, there are indexes for the most highly accessed inquiries. Optimizing databases means unloading the table data, deleting the table and reloading. Once the table is reloaded, re-indexing must also be performed. For each particular RDBMS in the environment you will need to master the optimization tool that comes with the particular database you utilize.

Spread out the I/O - For Unix this revolves around the creation of the file system. You may need to backup the filesystem, rebuild it and tar the files back to disk. What follows is a description of a method for spreading the I/O out on a system by moving subdirectories to new disc devices. It is involves moving the subdirectories to another disk and then using a symbolic link to point things to the new location.

Upgrade to the latest technology disk drives - The Fast and Wide SCSI drives will out perform the HPIB/HPFL disk drives for most types of I/O. While the HPIB/HPFL drives have a disk service time of 30 to 35 milliseconds the Fast and wide average 15 milliseconds doubly the speed of your I/O environment.

Avoid configuration problems - Don’t configure too many drives on one controller. Give careful consideration to the type of I/O a file system will handle when setting up the file system. If you know this at set up time I/O can be optimized. When setting up file systems in UNIX set them up with the files they will hold in mind and separate the related files.

Deal with fragmentation - Over time files and disk become fragmented. This fragmentation results in longer disk seeks times, more overhead, more I/O, longer disk queue lengths, and a slower response time. Use a reload of the system to treat disk and file fragmentation. Disk Pak and Seekrite are alternatives to a reload.

Avoid disk space problems - Pay attention to the files the system uses. Purge or Delete old or unused files, and archive little used files. Get enough disk space to handle the workload.

Conclusion

In general you should follow strategies that lead to more balanced and faster I/O. Databases need to optimized using the available means for each platform. Placement of files which make up databases is an important step in good I/O. Files that are related in usage should be kept on separate drives whenever possible. Disk drives should be kept in the cleanest state possible. Unused files should be purged in order to provide adequate disk space. Disk and file fragmentation should be treated with either a reinstall of the files or the usage of a product to defragment the files and the disk drives. Careful planning when disk drives are configured, file systems are built, and volume sets are created should help kept as many drives in service as possible. When many disks are involved in the execution of I/O disk queue lengths and pause for disk (or wait for disk) is kept to a minimum and per process disk performance is enhanced.

Finally, disk I/O should be minimized, optimize and scrutinized. Applications which perform a large amount of I/O should be looked at for ways to decrease the I/O. The size of memory should be maximized in order to ensure that response times to users are and minimized. Disk I/O is the weakest and slowest link in the resource chain so it pays to work on it.