SUN BLUEPRINTS
[illegible]
APPROVED

Sun / Oracle Best Practices

Bob Sneed - SMI Performance and Availability Engineering (PAE)

Sun BluePrints™ OnLine - January 2001

Sun microsystems
ENTERPRISE ENGINEERING

`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-6198-10
Revision 01, 01/08/01
Edition: January 2001

Sun / Oracle Best Practices

Best Practice Methodology can lay the groundwork for optimal system performance and linearity by consistent application of lessons previously learned. Unfortunately, many Best Practice lessons first become apparent outside of your own organization, far before they become widely known in print.

In this paper, Best Practice concepts are first defined, then specific high-impact technical issues common with Oracle in the Solaris™ Operating Environment (OE) are discussed.

Introduction

Most IT shops have - written or otherwise - certain practices which they regard as "Best". Such "Best Practices" represent the collective opinion of some group or organization. Since not every good idea qualifies as a "Best Practice", there is some challenge in laying out an agreeable definition and scope for use of the term.

Our group, which is part of the Sun Competency^{SM} Center organization, has provided both pre- and post-sales technical support to large ERP customers. We also participate in customer and marketing benchmark activities, with an eye towards the capture and management of useful knowledge. This scope of activities provides us with a unique vantage point on emerging Best Practices in our market segment.

In this paper, the terms "we" and "our" refer to the author's group at Sun Microsystems Inc, and more to the author in particular. Opinions expressed herein should be considered those of the author and do not represent any official position of Sun Microsystems, Inc.

Definitions

We define Best Practices as practices or procedures having these characteristics:

- Consensus of expert opinions, based on actual customer experiences in practice.
- Lessons learned, hopefully applied early in each project's life cycle.
- Proven practices associated with a particular usage profile (OLTP, DSS / DW, Business Critical, etc.).
- Baseline configuration rules - prerequisite to tuning.

In contrast, here are some things that often masquerade as Best Practices, but do not comply with our definitions:

- Tuning guidelines - these require some experimentation during application.
- Quick-Start Guides and Cookbooks - both useful, but not focused on what is "Best".
- "The First Thing that Worked" - lab reports lack the element of extensive comparative analysis.
- Any individual's findings - these lack the element of consensus.
- "The Best Way" - in practice, people compromise due to constraints on time, cost, or talent.

Consensus is the key word which best characterizes Best Practices. Without consensus, one might observe any given practice to be better than some others, but would have poor grounds to claim it as "best".

Best Practices versus Tuning

The distinction between "Best Practices" and "Tuning" deserves some special attention. Tuning varies from Best Practices in many ways:

- Tuning requires some degree of iterated trial-and-error, while Best Practices represent lessons learned which should just be applied consistently.
- Tuning is best done in an environment with a reproducible load (such as on a 'stress test' system) to accurately measure the results. In contrast, Best Practices are put forth as practices which should be routinely followed during system deployment.
- Tuning sometimes reveal unusual interactions between various parameters. Best Practices are put forth as not exhibiting such behaviours.
- In the tuning process, sometimes a negative step is taken. While Best Practices may incorporate some negative potential, documentation of Best Practices should clearly identify these so that no surprises occur.

Tuning techniques are best applied *after* Best Practices have been applied to a well-designed system which is part of an overall architecture that has been properly designed to meet the business requirements and project constraints.

Setting Priorities

Categorization and prioritization of knowledge is important. Regardless of staff and budget constraints, an organization should direct its efforts according to the potential payoff of each activity. In this list of priorities, Best Practices are represented as comprising the second tier of precedence:

1. Supportability Issues - having a supported solution is typically a high priority.
2. Best Practices - given a supported solution, proper configuration is essential. Best Practices may vary between versions and patch levels.
3. Tuning Guidelines - a properly supported and configured solution often requires some tuning.

Sun makes a huge amount of information available online and through a variety of outlets:

- Symptoms and Resolutions DataBase (SRDB)
- Field Information Notices (FINs)
- Patch Descriptions (from SunSolve Online^{SM} and SunSolve CDROM^{SM} services)

While each of theses resources contains a wealth of knowledge, their content is not arranged according to the definition of Best Practices offered here, and many of these documents live far beyond their useful life.

This paper does not intend to replace any of the resources shown above, but may on occasion seem to conflict with certain other data sources.

Best Practice Benefits

The benefits of Best Practices methodology are not all immediately apparent. While many readers will find the main concepts agreeable at first glance, the extended benefits warrant enumeration:

- Consistent attainment of business results: This is the umbrella goal, which might also be called the "root cause of success".

- Reduce hangs, crashes, and "system-too-slow" complaints: This is increasingly important as Sun™ systems are increasingly deployed in 24-by-365 business environments.
- Improve diagnosability: Keeping systems responsive is key to diagnosing operational issues.
- Promote technical consensus: Both within Sun and amongst Sun partners, this is a huge challenge. By focusing discussions on the most important topics, both the time-to-market and the shelf life of important knowledge can be improved.
- Improve Return On Investment (ROI) from tuning efforts. This impact can be huge! Quite often, good tuning interventions can produce negative results when applied to a system that does not conform with current Best Practices. Such outcomes not only waste expensive consulting efforts, but may also lead to incorrect conclusions and misdirected capital outlays.

Within our group in the Sun Competency Center, we are seriously disinclined to examine any system dynamics (e.g.: sar data, etc), from a system that has not been made compliant with our Best Practices. This simple policy prevents us from endless investment in re-deriving lessons already learned. Since these lessons seem to cover about 80% of the customer problems that have come to our attention, this policy helps us focus our efforts on the 20% of cases where other issues may be in play.

Best Practice Impediments

Various organizations exhibit various sorts of resistance to assimilating knowledge from outside sources. We have seen some recurring themes in this regard that warrant some discussion here.

Downside Potential

As the old adage goes, "*the promise is remembered long after the conditions attached to it are forgotten*".

It is easy with Best Practices to forget that once a practice has been branded as "Best", that it may represent certain tradeoffs and may involve noteworthy downside potential. It is also easy to forget the context for which any given practice was promoted as "Best", and therefore apply it in some inappropriate context.

We strive to document Best Practices that tend to have limited downside implications and to fully disclose any potential downside effects (or illusions of downside effects) known to us. Best Practice methodology should be clear in defining the scope of applicability for any given practice, and attempt to document both real and perceived issues which may be pertinent to the topic.

"Once Bitten, Twice Shy"

It is not uncommon for some wariness of a bug to persist long after it has been fixed. Real problems have a real beginning and a real end. When the bounds of a problem are well documented, one may have more confidence in moving past it. We strive to supply enough information with these Best Practices to promote informed decisions in moving beyond old bugs and issues.

"The" 80/20 Rule

In many shops, "traditional wisdom" holds that 80% of potential gains in applications performance lies in the application code. This can lead to unfortunate circumstances where systems administrators turn away database administrators who come seeking system-level configuration changes. "Go fix your code!", they might say - "we can't give you more than 20% gains - so don't bother us!"

In practice, we have seen up to 500% business throughput gains from system-level configuration changes, without touching the application code! Clearly, this particular 80/20 rule is not universal, and in practice, it can be quite limiting.

"Rules of thumb" and "traditional wisdom" must yield to new knowledge. While it is often true that a small number of factors have the highest impact on performance, preconceptions regarding the domain of these factors are simply not useful. Coordinated efforts by SA and DBA personnel are essential to attaining best results.

With large integrated software systems, exemplified by modern ERP systems such as SAP, Oracle, and PeopleSoft, application tuning and configuration management can become cumbersome and costly - especially through major product upgrades. This can produce strong incentives to squeezing optimal performance from the host platform to avoid unnecessary trouble and expense.

Traditional Configuration Management

A popular and pervasive practice in configuration management (CM) says "change one thing at a time, and if it does not help - change it back". This can be a useful technique for keeping a well-tuned system from being damaged by efforts to make it better. On the other hand, strict adherence to this strategy can prevent one from adjusting a system from its "out-of-the-box" default configuration into conformance with known Best Practices.

A simple graphical analogy is that, in any complex space, navigating to optimal points often requires movement along multiple coordinates. For example, one cannot get to the top of a hill by going straight up, nor can one make progress towards the top by navigating the perimeter.

Univariate sensitivity analysis is often not a viable method for exploring complex spaces. One could say that Best Practices are practices that tend to place a system in the general vicinity of known optimal performance in a complex space.

Prior to Solaris 8 OE, virtual memory management is one area in particular which requires multiple changes to Solaris OE kernel defaults for best effect.

Selected Technical Topics

While some of the topics here might be broadly applicable to many systems, this particular grouping of topics has evolved specifically from our experiences with high-end ERP systems, and with their backend database systems in particular. These systems are characterized by:

- Oracle as the dominant RDBMS
- SMP (4 or more CPUs)
- Large memory (2/4/8 or more GB RAM)
- Large databases (100/200/300 GB and larger)
- Systems sized to avoid any process paging
- Mission-Critical
- High availability requirements
- Mixed OLTP/DSS workloads
- Veritas VxFS often used (3rd party filesystem)
- Application code (SQL, PL/SQL, ABAP, Forms) not readily accessible, easily tunable, or easily maintained under configuration and change management (CM) procedures

Storage configuration and performance topics are largely avoided in this paper. Storage topics are categorically the most important issues not addressed here, and are often quite difficult to address in a narrative format. Many storage topics are very product-specific, and most could be characterized as ’tuning’.

With all this as context, here are the technical topics that will follow:

- Discipline
- Virtual Memory Management (VMM)
- Intimate Shared Memory (ISM)

- Interprocess Communication (IPC) Parameters
- Asynchronous I/O (AIO)
- VxFS Patch Level
- VxFS Blocksize
- Oracle SQL*Net TCP/IP vs. IPC
- Oracle Trace
- Oracle Online Redo Logs
- Filesystem Buffering Options
- Panic Dumps
- /tmp and tmpfs

Discipline

The topic of discipline is not typical of the Best Practice topics that follow, but one can say in a broad sense that discipline is a required practice for success - and that all the rest may not matter much in an uncontrolled environment.

The Internet explosion has seen many young rapid-growth companies stumble for lack of the kind of discipline long known in the mainframe or "glass house" community. Suffering results from many causes including: inadequate architectures; code that won't scale; inadequate training; inadequate testing; unproven or missing backup and recovery or disaster recovery processes; poor configuration management and change control procedures; and inadequate contracts management.

The downside to discipline is, of course, cost - both in terms of time and money. As they say though, you can "pay now or pay later".

Virtual Memory Management (VMM)

This topic is of such extreme importance, we will discuss it in a bit more detail than most of the other topics.

Solaris OE kernel versions 2.5.1 and 2.6 (at recent patch levels[1]) and version 7 contain a new memory management algorithm known as *priority paging* which is *disabled* by default. In the Solaris 8 OE kernel, priority paging is replaced by an even better third-generation algorithm.

1. 103640-27 or later for 2.5.1, and 105181-10 or later for 2.6

Prior to the Solaris 8 OE, we regard it as a Best Practice to *always* enable priority paging, and also to set related VMM parameters according to a new strategy. In particular, on large memory SMP systems, we routinely use these settings in `/etc/system`:

```
set priority_paging=1
set fastscan=131072
set maxpgio=65536
```

These settings are quite different from those suggested by current books in print, and require some explanation.

Especially in the case of large ERP backend database systems, sizing strategies aim to accommodate all database shared memory and process pages in RAM, with sufficient reserve allowed to accommodate transient memory demands and to provide a suitably sized filesystem buffer cache.

Under the original Solaris 2.x OE kernel's VMM strategy, it was all too common for buffered filesystem activity to provoke undesirable paging activity. Since paging statistics offered by vmstat have not historically segregated 'real paging' from the paging associated with filesystem I/O, it was historically difficult to distinguish 'good paging' (filesystem I/O) from 'bad paging' (process paging)[2]. With settings along the lines shown above, so long as free memory remains comfortably greater than **lotsfree**, one can infer that no 'bad paging' is taking place. If in doubt, one can monitor actual swap I/O rates by using **iostat**.

Preventing paging storms is of such extreme importance that we promote the above settings even in cases where the database is operated against RAW disk! Otherwise, utility operations such as backup and restore or ad-hoc system maintenance can provoke paging which, in turn, can seriously impact service levels. In extreme cases, systems can appear to become *hung* due to paging.

Potential downside effects from priority paging are few in number, but should be carefully considered.

One *perceived* issue arises with priority paging because the implementation changes the interpretation of the 'scan rate' metric reported by vmstat. This can create the *illusion* of a memory shortage problem. Therefore, scan rate alone cannot be used to diagnose a memory shortage. When free memory remains greater than **lotsfree**, no real paging is occurring, and applications are meeting expectations, there is no memory shortage - regardless of the scan rate.[3]

2. This has been addressed by an enhanced vmstat command, available for the Solaris 7 OE, and included with the Solaris 8 OE

3. This is not to say that more memory might not find good use!

Some real issues have been identified related to priority paging. First, when data files have an **execute** file mode enabled, and are accessed via the **mmap(2)** system call[4], their pages are treated with the same priority as process pages, which is usually not desired. Second, pages associated with *tmpfs* filesystems (eg: `/tmp`) are also treated as having the same priority as process pages, which can have undesirable effects. Third, when implementing priority paging, it is important to remove settings which might have been part of an earlier tuning strategy, such as elevated settings for l**otsfree, desfree**, or **minfree** tunables.

Additional information regarding priority paging is available via SunWorld™ online articles written by Richard McDougall. SAP users can find much of this information summarized in SAP OSS note 200057.

One can think of the new VMM tuning as enhancing the supply of free memory by quickly returning memory to the free list. Some of the greatest demand on the VMM system can come from buffered filesystem operations. Balancing this supply and demand relationship can be key to improving system throughput and linearity under intense loads.

Intimate Shared Memory (ISM)

Intimate Shared Memory is a feature of the Solaris OE kernel[5] which improves the performance of programs that use shared memory, and can also improve overall system performance when many processes are accessing the same shared memory (as with Oracle). With ISM, shared memory is mapped using large pages which are locked in physical memory. The overhead of process context switches is thus greatly reduced. The guarantee that ISM memory will not be paged improves Oracle's performance linearity under load[6]. By default, ISM is enabled both by Oracle and by the Solaris OE kernel.

We regard as Best Practice that ISM should not be turned off without a very specific reason. Under high loads, especially with high process counts, the cascaded negative impact of disabling ISM can be severe. We have seen as much as 40% overall performance degradation cascade from ISM being disabled[7].

We know of three possible downside effects associated with the use of ISM. First, on the Sun Enterprise™ 10000 Server (E10000) under the Solaris 2.6 OE kernel, a data corruption issue was repaired with JKP-15. Second, also on the E10000, there have been a series of issues regarding compatibility between ISM and Dynamic

4. Oracle *does not* use **mmap** on its data files
5. In the shmsys module
6. Oracle will quietly allocate non-ISM memory if sufficient ISM cannot be allocated. Performance can degrade seriously if the Oracle SGA starts paging.
7. In that case, other Best Practices deviations were presentand contributed to that results

Reconfiguration (DR)[8] operations. Finally, if too high a percentage of system memory is allocated as ISM, other demands for RAM may be shortchanged. This, however, is basically a system sizing and tuning issue.

On systems other than E10000 servers, we know of no recent problems with ISM except that old `/etc/system` files from E10000 systems (with ISM disabled) have sometimes been copied to these systems, thus compromising performance for no concrete technical reason.

Interprocess Communication (IPC) Parameters

We regard it as a Best Practice to set many of the IPC tunable settings sufficiently large to put them *out of the way*. Since these settings require a system reboot to take effect, it can be important to set them such that service interruptions are not necessary for this purpose. Avoiding reboots is a quite valid tuning goal.

When the System V IPC facilities were designed and deployed, prevailing system memory sizes were quite small by today's standards, and balancing the impact of static kernel data structures on total memory was an important tuning goal. On today's large memory systems, the cost of very large IPC settings is usually negligible. Indeed, some settings, such as **SHMMAX**, have no cost associated with them *whatsoever*, but merely establish limits.

In the particular case of **SHMMAX**, we regard it as common practice to set this in `/etc/system` as:

```
set shmsys:shminfo_shmmax=0xffffffff
```

- on 32-bit kernels, and possibly higher on 64-bit kernels.

Common misconceptions are that large IPC settings (of **SHMMAX**, in particular) are somehow risky, or that smaller settings are useful for enforcing resource management policies.

For optimal Oracle performance, the SGA must be allocated in a single shared memory segment as ISM. If the SGA cannot be built this way, Oracle will use multiple segments, and will settle for non-ISM allocations. If Oracle still cannot acquire the needed memory, it will fail to start. The key factor for system performance is whether or not an unreasonable proportion of memory is taken by ISM rather than how large any single segment is allowed to be.

8. At this writing (2.6 JKP-22), these issues appear to be largely past. No such issues have been reported under Solaris 7 or 8 OEs.

Asynchronous I/O (AIO)

Asynchronous I/O is an application programming interface (API) through which programs like Oracle can reduce the overhead of attaining I/O concurrency[9]. In general, Oracle uses AIO by default[10]. In the absence of any reason to the contrary, we regard it as Best Practice to exploit the performance gains available from AIO.

That being said, there have been bugs, both real and perceived, pertaining to AIO in the Solaris OE which have led many away from this practice.

Many circumstances can result in an Oracle write operation actually failing to complete in ten minutes, which results in an ORA-27062 error. Most often, this is the result of a hardware failure, but other causes include peculiar use of the OS scheduler causing Oracle preemption, serious storage configuration errors, and serious deviation from the Best Practices prescribed in this paper. When this happens to most data files, media recovery is required. If it happens to the online logs, database recovery can be quite painful!

Regardless of the cause of an Oracle failure resulting in ORA-27062, AIO tends to get the blame, because this error is only reported when AIO is used. At this writing , there are no bugs known to us in libaio.so which are pertinent to Oracle operations, whether or not *Kernel* AIO (KAIO) is actually being used[11]. An issue in libaio.so which only effected Oracle using RAW and Veritas Quick I/O (QIO) was repaired by 2.6 JKP-17. Also, an issue with Veritas Volume Manager (VxVM) 3.0.4 was implicated in some ORA-27062 incidents, and recently repaired by patch 109686-01.

An AIO problem has often been *perceived* when **truss** is used to observe Oracle I/O operations. With AIO on filesystems, a **kaio(2)** call can be observed to routinely return ENOTSUP results, and the mistaken conclusion is drawn that AIO is 'broken'. Actually, the **kaio(2**) system call is not an *exposed interface*. That is, only the library functions **aioread(3)** and kin are published as supported by Sun, and **kaio(2)** is one of the secrets to the libaio.so implementation underlying the AIO API. In other words, this is not a bug - it's a feature!

AIO and KAIO are discussed in detail in one of Jim Mauro's SunWorld online articles.

There is a possible downside effect from AIO to files when the SGA is grown too close to the DBWR process stack. Increasing Oracle's I/O concurrency to very high levels can provoke this problem.[12] The correct solution to this potential problem is to properly size the SGA and relink the Oracle binaries according to Oracle's prescribed method as required to properly accommodate larger SGA sizes.

9. It may cause some confusion that synchronous and asynchronous I/O are not mutually exclusive. AIO refers to managing I/O concurrency, while synchronous writes refer to I/O completion criteria.

10.Certain Oracle 8.1 releases suppress AIO use on ordinary filesystem files

11.At this writing, KAIO actually occurs only on RAW and Veritas Quick I/O files

12.VxFS installation increases the system-wide default lightweight process (LWP or user thread) stack size, which can contribute to this problem occuring

Finally, when AIO is used in favor of, or in conjunction with, multiple DB writer processes - related Oracle tuning will vary relative to when AIO is disabled.

VxFS Patch Level

We now consider VxFS 3.3.2, with its patch #2 as a minimum recommended VxFS patch level. Especially in the Solaris 2.6 OE, all prior versions exhibited some undesirable behaviours for which various patches were offered at various times.

VxFS 3.3.2 is not among the VxFS releases which have been specifically qualified for use with Sun™ Cluster software. This should be well considered in contemplating potential upgrade strategies.

Note – VxFS 3.3.3 is the minimum level required for use with the Solaris 8 OE.

VxFS Blocksize

By default, VxFS filesystem creation uses a blocksize that varies with the filesystem size. For filesystems that will be dedicated to storage of Oracle datafiles, this is generally not optimal. Since it can be painful to change this after lots of data has been loaded, we regard it as a Best Practice to assure that VxFS filesystems are built with a purpose-set blocksize.

Since Oracle databases most frequently use an 8K or larger blocksize these days, and since Sun's UltraSPARC™ processors use an 8 KB page size, it just works best for VxFS to use an 8 KB blocksize for Oracle data files as well.

The Veritas **vxva** visual administrator offers no means to set this parameter when building filesystems, so it is best done from the command line[13]. For example:

```
mkfs -F vxfs -o \
bsize=8192,largefiles
/dev/vx/rdsk/diskgroup/volume
```

The `fstyp -v` command can be used to verify the blocksize of existing filesystems.

13.The new **vmsa** GUI *does* allow setting filesystem create and mount options

Oracle SQL*Net TCP/IP vs. IPC

It is relatively inefficient for Oracle client processes to use TCP/IP to communicate with their shadow processes when both client and shadow are on the same system. This has appeared as a common configuration issue at many sites. At high levels of usage, this causes high %sys CPU utilization coupled with high traffic on the local (**lo0**) interface, and poor user response.

Shadow processes with '(LOCAL=YES)' or '(PROTOCOL=beq)' use the more efficient Interprocess Communication (IPC) facilities.

SQL*Net's AUTOMATIC_IPC feature was designed to automatically use the most efficient communication method possible. The AUTOMATIC_IPC implementation presented its own performance issues at high connection rates, and has since been discontinued with Net8 and Oracle 8i.

We regard it as a Best Practice to routinely screen for this potential performance issue, and take appropriate corrective action based on the Oracle version and the usage pattern.

Oracle Trace

Some 7.X versions of Oracle shipped with Oracle trace enabled by default. This causes each new shadow process to add data to a log file in the `$ORACLE_HOME/` otrace directory when starting. As the log file grows, system response degrades due to the exclusive file access required by each new login, along with the scanning of the file done by each login process.

We regard it as a Best Practice to routinely confirm that this issue is not present. It is a primary suspect when performance is reported as degrading over time.

Oracle Online Redo Logs

Oracle's online redo logs are of critical importance to update/modify throughput and play a key role in database recovery operations. Failure of an online log group can lead to the most painful of all Oracle database recovery scenarios. For these reasons, online logs deserve very special attention.

We regard as Best Practice that Oracle's log multiplexing be used in favour of, or in addition to, RAID features. For optimal availability, members of multiplexed log groups should be located such that they share no common points of failure at the disk, channel, or board level. Finally, where update/modify performance is important, logs should be located on dedicated (or otherwise quiet) disk spindles.

These practices have two known downside effects. First, one must typically configure an apparent excess of disk capacity to meet all these constraints. With per-disk capacities soaring, this is increasingly difficult to justify. Second, since Oracle must do each log write twice to keep both log group members synchronized, peak throughput is slightly reduced. Still, this proves a small price to pay for providing optimal protection to these most critical files.

Filesystem Buffering Options

Many will argue passionately over the intrinsic superiority of either RAW or filesystems for Oracle database storage. In environments where filesystems have been mandated, debates rage over what filesystems and usage options are "Best".

We regard as Best Practice that the comparative benefits of various buffering options be considered in the physical layout of Oracle databases. In effect, buffering is a *tuning dimension* that often goes overlooked or unexploited. We make no claims that any particular usage is a Best Practice, but that thoughtful consideration of the facts in this regard *is* a Best Practice. We can offer guidelines pertinent to exploiting buffering, but best results will depend on many factors including workload, memory sizing, and storage architecture. While "your mileage may vary", we have concluded that optimal application performance often depends upon system-level buffering factors.

With such a Best Practice now laid out, the task of explaining all of the various relevant factors can be somewhat daunting. Space does not allow a complete treatment here, but we will try to touch on the major factors.

Each and every file in an Oracle database can be configured with varying options at the system level. There is no requirement that they all be RAW versus real files, or that all the files use the same mount options.

Even when comparing RAW to files, the highest impact distinction amongst the options is whether or not buffering is used at the OS level. Leaving behind RAW for now, let's look at filesystem factors in particular, since most sites adopt files as a design constraint.

Use of the buffer cache with Oracle files should be guided by the fact that all Oracle data files are opened in data synchronous mode[14]. Oracle's reliability depends on the assumption that all writes are flushed from the system (versus being deferred) before they are considered done.

14.This is the result of the hard-coded `O_DSYNC` flag being passed to **open(2)**

Two historical design motivations for the OS buffer cache were write deferral and write coalescing. With synchronous writes, neither of these features is realized for Oracle writes. In short, use of buffer cache represents a detour for Oracle writes, and involves extra logic and an extra copy operation which is of no benefit to the writes *per se*.

Two further features of the buffer cache pertain to reads. Read hits in the buffer cache avoid physical I/O, and pre-fetching at the filesystem level can speed sequential file processing. These features offer great potential benefit to Oracle throughput. However, these features are not equally applicable to all categories of Oracle I/O. As it turns out, rollback and index I/O are the most likely beneficiaries for read hits, and pre-fetching can be useful for any operation that provokes sustained sequential reading.

Both UFS (since the Solaris 2.6 OE) and VxFS offer unbuffered modes of operation called *direct* I/O. With UFS, buffering can be disabled with the **forcedirectio** mount option. With VxFS, the **mincache** and **convosync** mount options provide separate control over the buffering of reads and synchronous writes, respectively. VxFS further offers an option known as *Quick* I/O (QIO) which offers further performance and manageability features.

Considering these many factors during physical database design adds complexity, but can have significant rewards. While tuning via mount options can be somewhat cumbersome, the benefits are worthy of investigation.

What is best for Oracle applications throughput might not be best for other types of I/O. For example, in cases where direct I/O is deployed, utility programs such as backup and restore or ftp can be adversely affected, since these programs are typically designed to assume buffered I/O operations. These tradeoffs must be evaluated on a case-by-case basis.

Panic Dumps

Nobody running mission critical applications wants to try reproducing circumstances of a system panic on a production server!

In the event of a panic, it is critical to get a correct diagnosis if possible, so that measures might be taken to assess the cause and eliminate future exposure to business disruptions from the same cause. With the **dumpadm** command introduced in the Solaris 7 OE, the goal of reliably capturing a panic dump is greatly advanced.

We regard it as a Best Practice to become completely familiar with **savecore** and **dumpadm** operations, and work with Sun Enterprise Services to configure for reliable capture of panic dumps. Rehearsal of related procedures is highly advisable.

The potential for panic dumps tends to decrease with each OS release as refinements are made to deal more gracefully with exception conditions. Often, panics are obviously due to operational errors and will not require analysis. Most systems will never see a panic that lacks an obvious cause. Since panics are rare events, it is important to capture the evidence with a fully automated process and be able to make a correct diagnosis of every such incident.

The obvious downside to these measures is cost, both in terms of time and in terms of dedicating sufficient disk space to the cause. As with any insurance, the payoff comes under circumstances which we hope will never occur.

`/tmp` and tmpfs

By default, when the operating system is installed, `/tmp` is established as a *tmpfs* filesystem with the default capability of consuming all of system virtual memory. If this ever occurs, the system will first start to page fiercely, then ultimately hang. Without preventative measures being exercised, this can be the basis of a denial-of-service attack by any login user. Even in the absence of malice, any runaway program writing `/tmp` can hang a system.

No one can ever know how many system hangs have been due to runaway `/tmp` consumption. When systems hang, they are rebooted. After rebooting, no evidence from any *tmpfs* filesystems will remain.

Use of `/tmp` can be restricted as a matter of policy, but lacking any means to enforce such a policy, we regard it as a Best Practice to either limit the potential impact of `/tmp`[15] or convert `/tmp` to real disk.

There are no downside effects to limiting tmpfs usage per se, but some applications might be negatively impacted if `/tmp` is converted to real disk, or if `/tmp` becomes full. As mentioned earlier, since *tmpfs* pages are handled at the same priority as process pages by the new priority paging algorithm, budgeting *tmpfs* RAM consumption can be crucial to system performance and linearity.

15.This is possible by using the **size=** mount option in `/etc/vfstab`. See: **mount_tmpfs(8)**. NOTE: A longstanding bug may require that the argument to the **size=** mount option be less that 2048m under certain OS releases

Closing Remarks

The final technical topic discussed above (about limiting *tmpfs* VM usage) is actually applicable to any and all Solaris OE systems, especially if they are mission critical. With this in mind, one might implement a process to systematically avoid ever experiencing excessive *tmpfs* VM consumption. Whatever that process might be, it should also address all of the other topics discussed here, along with any others that prove important to your business success. That takes us back to the first technical topic discussed above - discipline.

References

Timely information on Sun topics is often first published online at `http://www.sun.com/blueprints` and `http://sunworld.com`.

Acknowledgments

My sincere thanks to Richard McDougall, Jim Mauro and Richard Elling for encouraging me to write all this down. Their own many and varied publications have proven invaluable to me and thousands of others. Thanks also to Phil Morris and Rey Perez for the opportunity to write this, and for their repeated peer review.

Author’s Bio: Bob Sneed

Mr. Sneed is a currently a Staff Engineer on the Systems Engineering team within Sun’s Performance and Availability Engineering (PAE) group. They are charged with addressing various product integration challenges within Sun’s Integrated Products Group (IPG). He stays obsessed with system performance and usability topics, often involving Oracle and data storage technologies. Prior to joining Sun in 1997, he worked variously as a Sun ISV, a Sun reseller, and a Sun customer. His early career experiences in realtime industrial controls and software portability engineering helped shape his unique perspectives on database and system performance issues.