SUN BLUEPRINTS
APPROVED

Tales from the Trenches: The Case of the RAM Starved Cluster

By Richard Elling - Enterprise Engineering

Sun BluePrints™ OnLine - April 2000

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.: 806-5332-10
Revision 01, April 2000

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please
Recycle

Adobe PostScript™

Tales from the Trenches: The Case of the RAM Starved Cluster

"Tales from the Trenches" represent real live stories of how problems in the field are solved by using Sun™ products and technologies.

Many Sun customers use the Veritas File System, VxFS, which provides an extent based, journaling file system for Solaris™ Operating Environment. This paper discusses how VxFS affects memory on a Solaris Operating Environment server. A real world example of the interactions between the Solaris Operating Environment Version 2.5.1, VxFS Version 2.3.1, and user applications is described. The methods used to troubleshoot the problem are described. Finally, general purpose solutions to the problems are discussed.

File Systems and Solaris™ Operating Environment

The UNIX® File System (UFS) in the Solaris Operating Environment uses main memory (RAM) to cache file data. This is generally a good idea. UFS with direct I/O is a feature added to Solaris Operating Environment Version 2.6. With direct I/O, no caching is performed. This is advantageous when performing large sequential I/O since the overhead of managing the RAM cache negatively impacts performance. RAM is often a constrained resource on a system, so caching sequential I/O may cause resource management problems as processes compete for available free RAM with the file system cache. Direct I/O for UFS in Solaris Operating Environment Version 2.6 is a mount option on a per file system basis.

VxFS, has caching and non-caching capabilities similar to UFS. However, VxFS tries to intelligently decide when to use each. By default, for small I/O operations VxFS will cache. For large I/O operations VxFS will use direct I/O and not cache. This is called discovered direct I/O. The `vxtunefs` parameter which dictates this decision is discovered_direct_iosz (in bytes) which is set on a per file system basis. The default value for discovered_direct_iosz is 256kB. Thus, I/O operations that are smaller than 256kB will be cached, while those larger than 256kB will not be cached.

The decision by VxFS of whether to cache or not is based on the size of the I/O operation, not the spatial relationship of consecutive operations. This is a subtle but important distinction. For instance, issuing 1024 2kB contiguous block writes will be cached. Issuing 2 1024kB contiguous block writes will not be cached. Even though both transactions result in 2048kB of contiguous data, the effect on the RAM system due to caching will be quite different. For the 2kB block write case, 2048kB of RAM will be allocated for the cache. If the disk subsystem is slow, a common occurrence, and the amount of free RAM is less than 2048kB, another common occurrence, then the system will see a RAM shortfall and begin paging. With modern processors, the pressure placed on the memory subsystem can be quite substantial. It is not unusual to see page scan rates above 10,000 pages per second sustained for quite some time during such a shortfall on an Ultra Enterprise™ Server. Such a system load is often described by users and system administrators as "hung" or "locked up." The good news is that Solaris Operating Environment will handle the shortfall and eventually the system will return to normal after the I/O is completed.

Example of Such a System

A good example of how this affects a system is a high availability (HA) Oracle Version 7.3.x OLTP cluster. In such a cluster, there is one or more HA daemons which monitor the health of the servers and can initiate fail overs. These HA daemons communicate between servers on multiple networks where they pass heartbeat information to determine if the other server is operational.

Each server in the cluster has 8 UltraSPARC™ CPUs, 2GB of RAM, and shares 100GB of disk located in an external RAID array. The Oracle SGA size is 1GB and uses intimate shared memory (ISM), which locks the shared memory pages into RAM. This leaves 1GB of RAM for the kernel and user processes. The Oracle table spaces are placed in VxFS file systems with discovered_direct_iosz set to the default of 256kB.

Under normal operation, Oracle works well. Preproduction testing of the HA fail over showed no problems and all HA scripts are functional. Users are happy. System administrators are happy.

One day, the databases fill up. The database administrator (DBA) creates a new table space for the growth. The primary machine "locks up." The fail over secondary machine initiates a takeover and recovery. The primary machine panics. Users are unhappy. System administrators are unhappy.

What happened?

Nothing the users or administrators did was extraordinary or operationally improper. No excessive load was induced by the users and the system is properly sized and configured. No hardware failures occurred. No Oracle or Solaris Operating Environment bugs were encountered. One must have a holistic understanding of the entire environment to put the pieces of the puzzle together.

Oracle databases typically use 2kB sized I/O for OLTP queries. These would be cached by VxFS rather nicely. The writes would go to the RAID arrays which would further cache the writes and stage them to disk.

When the 2GB table space is created for the additional growth, Oracle issues 128kB sequential I/O operations. VxFS begins to cache these as they are smaller than `discovered_direct_iosz`. The RAID array caches these writes but rather quickly fills its cache. VxFS continues to cache the writes into RAM. Before long, the available free RAM becomes critically short. Remember that there is only 2GB of RAM in the system and Oracle has locked down 1GB for the SGA. The system begins to page then swap. The page scan rate exceeds 10,000 pages per second. The machine becomes very slow. Users begin calling the system administrators complaining that the machine is "hung." The HA software daemons don't get swapped out, but they don't have much memory to work with either and they are constantly fighting the VxFS cache for pages. The heartbeats become irregular and decrease in frequency. The HA software on the secondary server decides that the primary has failed and initiates a forced fail over. The primary server loses control of the disks or shuts down Oracle because of the fail over. Huge quantities of memory are freed as Oracle shuts down. The HA daemons, no longer fighting VxFS for pages, resume their normal heartbeat. But there is a fail over in process and they detect a split brain possibility. The primary server HA daemon panics the kernel to avoid the possible split brain situation.

Troubleshooting

The troubleshooting session began with a briefing on the observed behavior. The DBAs discussed their operational policies and procedures. The correlation between the DBA creating the new Oracle table and the subsequent fail over was confirmed. Log files were analyzed and seemed to confirm the memory shortfall of the primary server.

A test was performed on the primary server while clustered. The production service was moved to an alternate server which allowed full testing of the production cluster. Additional monitoring tools were installed to collect detailed information about the behavior of the primary node during the failure. Of these tools, vmstat and iostat were the most useful and provided sufficient detail of the memory system and I/O activity to diagnose the problem.

Recreate the Original Problem

The original problem was recreated. The system was brought online and stabilized. The DBA began the new table creation. Subsequent RAM starvation, fail over, and forced fail panic were induced. This confirmed that there was a direct causal relationship between the I/O activity of the table space creation and the RAM starvation.

The test went through the fail over to the secondary node in the cluster. The split-brain alert was seen and the forced panic of the primary node occurred as previously described. This eliminated the HA fail over software as a problem because it was operating as designed.

Testing RAM Starvation

The load on the system imposed by the Oracle table space creation was a sequential I/O load. This load can be simulated using dd. The first test was to create a 2GB test file. This test used a 1MB I/O size.

```
# dd if=/dev/zero of=/u01/oradata/testfile bs=1024k count=2048
```

This test allowed the characterization of the I/O subsystem itself. The 1MB I/O size is larger than the `discovered_direct_iosz` and will not be cached. The resulting I/O data shows the effect of the RAID array cache saturation as shown in Figure 1 below.

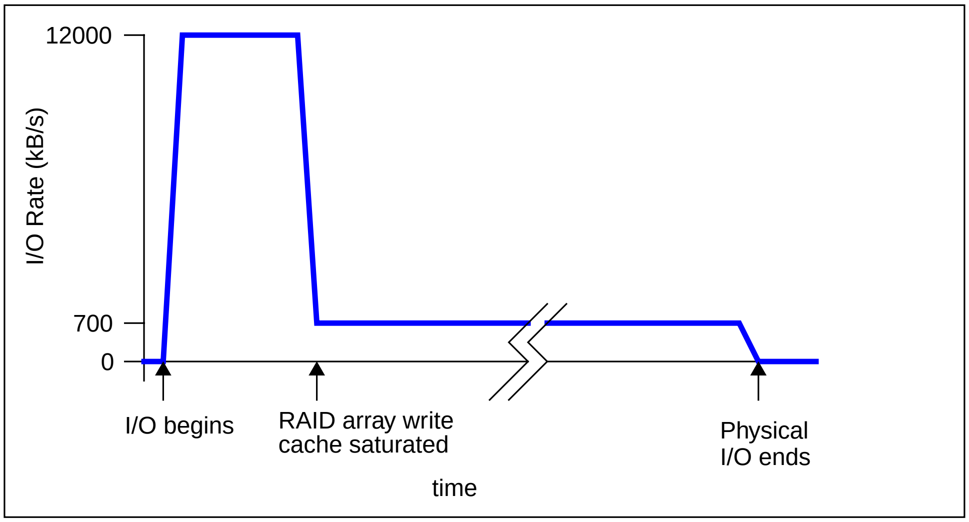

FIGURE 1 1MB Size I/O Throughput Data

The RAID array cache size was deduced to be approximately 128MB. The RAID array back end write throughput was measured at approximately 700kB/s. While this performance is not spectacular, it is also not the root cause of the RAM starvation problem. Significant costs can be incurred by increasing the throughput of the RAID storage system but it would not solve the RAM starvation problem.

The second test changed the block size to match that of the Oracle table space creation. The data from the production problem recreation showed that Oracle was writing a block size of 128kB during the table space creation.

```
# dd if=/dev/zero of=/u01/oradata/testfile bs=128k count=16384
```

Since 128kB is less than the default `discovered_direct_iosz`, the blocks will be cached. As expected, the RAM usage increased rapidly and the system quickly became severely RAM starved. Figure 2 shows the effect of this test on the system.

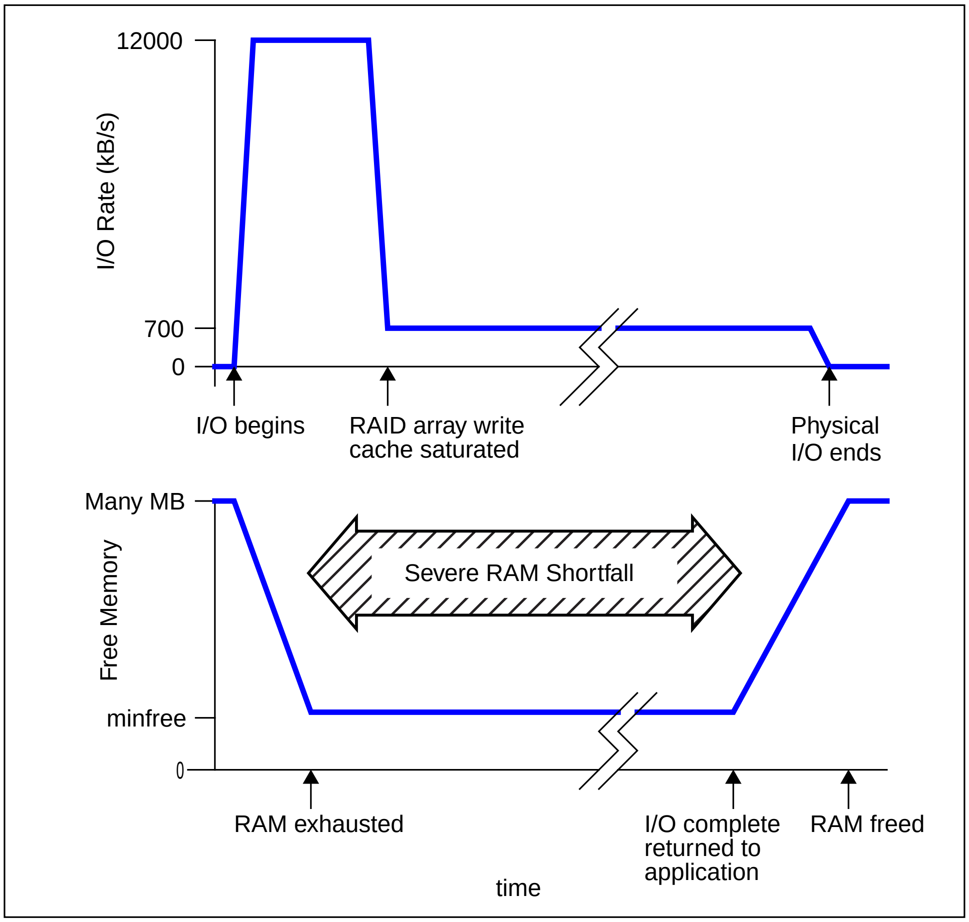

FIGURE 2 RAM Consumption by VxFS Write Cache.

The Solution

The work around for this problem is to use `vxtunefs` to set the VxFS `discovered_direct_iosz` to something less than 128kB. For Oracle OLTP, 64kB seems reasonable. This would cause the table space create to avoid the RAM cache, thus avoiding the severe RAM shortfall and preventing the HA fail over.

Verify current settings

```
# vxtunefs -p /dev/vx/datadg/db01
Filesystem i/o parameters for /u01/oradata
...
discovered_direct_iosz = 262144
...
```

Change discovered_direct_iosz

```
# vxtunefs -o discovered_direct_iosz=64k /dev/vx/datadg/db01
vxfs vxtunefs: Parameters successfully set for /u01/oradata
```

Verify changes

```
# vxtunefs -p /dev/vx/datadg/db01
Filesystem i/o parameters for /u01/oradata
...
discovered_direct_iosz = 65536
...
```

The vxtunefs changes are not permanent and will revert to the default when the system reboots. Permanent changes can be made by adding them to the `/etc/vx/tunefstab` file.

```
# Set discovered_direct_iosz for Oracle database table spaces
/dev/vx/datadg/db01  discovered_direct_iosz=64k
```

The original problem was found on a system running Solaris Operating Environment Version 2.5.1 and VxFS version 2.3.1. Subsequent testing shows that this solution is appropriate for the Solaris Operating Environment Version 8 and VxFS version 3.3.3.

Conclusion

Holistic understanding of complex systems is often the key to predicting their behavior. In this case a database cluster was failing during routine database maintenance. The problem was analyzed and a solution found. This solution included tuning of the VxFS `discovered_direct_iosz` variable to avoid RAM shortfalls when creating table spaces.

References

Information on how to administer VxFS file systems can be found in the *VVeritas File System*^{TM} *System Administrator's Guide.*

The manual pages on `vxtunefs(1m)` and `tunefstab(4)`discuss VxFS tunable parameters and how to set them.

Author's Bio: Richard Elling

Richard is a Senior Engineer in Enterprise Engineering for the Computer Systems at Sun Microsystems in San Diego, California. Richard had been a field systems engineer at Sun for five years. He was the Sun Worldwide Field Systems Engineer of the Year in 1996. Prior to Sun, he was the Manager of Network Support for the College of Engineering at Auburn University, a design engineer for a startup microelectronics company, and worked for NASA doing electronic design and experiments integration for Space Shuttle missions.