

AVOIDING COMMON RAC PROBLEMS

Dan Norris, dannorris@dannorris.com, DanNorris.com

INTRODUCTION AND TERMINOLOGY

Oracle Real Application Clusters (RAC) doesn't work for everyone in every situation; that's a fact. But in many cases RAC can work—and work well—if configured and used properly. Some issues are caused by simple misconfiguration. Others are more complex, like application-related logic issues that only surface in RAC environments. There are some issues that appear with a relatively high frequency, and many of them can be fixed without knowledge of internals. This session will review common issues in the areas of networking, storage, people, applications, and testing, examining the issues and their solutions.

To support the discussion that follows, these definitions may be helpful:

Term	Definition
Database	the control files, data files, and online redo logs
Instance	the shared memory and background processes that operate on a database
Clusterware	software that manages cluster membership and monitors the nodes and networks in a cluster
Storage Area Network (SAN)	a storage environment where multiple servers can utilize a single storage array; a storage network is commonly implemented using fiber channel technology
Local Storage	disk space that is available to exactly one node in a cluster; this storage may be part of a SAN or may be direct-attached to the server
Shared Storage	disk space that is available to more than one node in a cluster at the same time; this storage is commonly part of a SAN
Raw Device	the character (unbuffered) special device presented by the operating system
Cluster Filesystem	a special filesystem that can be accessed by multiple cluster nodes at the same time
Oracle Automatic Storage Management (ASM)	software that will manage disks directly and provide volume management functions like striping and mirroring; this feature was new with Oracle Database 10g
Single-instance Database	the traditional "stand alone" database configuration used by most Oracle environments today
Multi-instance Database	a single database serviced by more than one instance
Oracle Services	an entity defined in RAC databases that allow you to group database workloads in order to route work to the instances best able to complete the work optimally

RAC ARCHITECTURE REVIEW

To make sure that the rest of this paper is clear, we must first review some of the key RAC architecture points that must be understood when deploying and managing RAC environments.

RAC is a multi-instance database. That means that there is exactly ONE database that is serviced by more than one instance. The instances are on separate nodes. Above all else, this key architectural point is the most important one to understand--not just remember, but truly understand. With it in mind, many other corollaries can be derived if you also understand the relationship between an instance and a database. For example, instances generate redo and undo (not databases) which means that you'll have separate redo threads and undo tablespaces for each instance in the cluster.

Oracle Clusterware is required in all RAC environments starting with Oracle 10g Release 1. While an additional, third-party clusterware (i.e. Veritas Cluster Server or IBM HACMP) may be used in addition to Oracle Clusterware, the Oracle Clusterware is still required. Oracle Clusterware serves as the cluster manager and manages cluster membership using a sophisticated set of heartbeat mechanisms. If there is communication failure that may jeopardize the integrity of the database, Oracle Clusterware may reboot one or more nodes to ensure that the database remains safe from corruption. Oracle Clusterware maintains the Oracle Cluster Registry (OCR) to track metadata related to cluster membership and status.

Oracle RAC is based on a "shared everything" architecture. This means that all servers in the cluster have direct access to all storage in the cluster at all times. It also means that activity happening on node #1 may affect the performance of nodes #2-5 in your cluster even if nothing is happening locally on them.

NETWORK ISSUES

I find that many RAC environments struggle with key parts of network configuration and administration. This section will discuss several important aspects of network configuration and management that will help improve your cluster's redundancy, performance, or manageability.

SERVICES, WORKLOAD MANAGEMENT

Oracle Services were introduced with Oracle8i, but are rarely used by most single-instance database environments. A service is defined in the database (DBA_SERVICES, DBMS_SERVICE) and also in the clusterware (using srvctl) to manage workload. Some environments create services to segregate groups of users; others use services to manage multiple applications that share a database. Many services can be defined, and the right number depends on how you want to manage your users and their associated workload. The more services you define, the more granular you can control and measure your workload, with the associated slight increase in management overhead.

When Oracle 10g was released, services became more useful because many counters were added to track statistics by service name. These statistics gave some additional incentive to use services because the per-service statistical information can help identify the source of issues, especially performance issues.

The most common error related to services is not using them at all. A significant number of the RAC environments I've encountered do not configure any non-default services and use the default service name configured with the installation. While this paper isn't intended to provide a complete primer on services, it is highly suggested that you research services and make good use of them in your RAC environment for ease of workload management. This is especially true for clusters with more than two nodes since they often have larger user communities and may require more services to enable effective management of the database's workload.

When multiple services are employed, workload management is enabled because services need not be available on all nodes in the cluster. This allows the DBA to designate certain subsets of nodes in the cluster to service a particular group of users or application. Services may be restricted in this way to ensure that other users or applications do not affect them so that they can achieve the performance they require to finish their work in the required time. In other cases, services may be restricted so that they do not affect the rest of the cluster's users (i.e. for batch processes).

For additional information, Jeremy Schneider wrote and published a great whitepaper on Oracle Services on his website: <http://www.ardentperf.com/pub/services-schneider07.pdf>

LACK OF REDUNDANT INTERFACES

Another common issue with RAC clusters is the absence of redundant network interfaces. In most cases, RAC environments have very high availability requirements. To provide the highest availability, the cluster must be designed to eliminate as many single points of failure as possible. On a server level, this requires addressing each individual component of the server separately. In today's standard servers, the CPU and memory are managed by the hardware and software so that if one fails, some servers will disable the failed component during a reboot and the server will come back online. In such cases, there is still an outage, but only as long as it takes to reboot. This is typically much shorter time than required to diagnose and replace a failed CPU or memory component.

Storage redundancy is handled via RAID controllers and configuration. The internal drives in most servers today are connected to an internal RAID controller that presents LUNs to the operating system. Those LUNs are usually mirrors or other redundant configurations that will survive in the event that one of the drives fails. External storage redundancy is typically handled by the storage array(s) providing the external storage. In the case of inexpensive external storage "shelves" that do not have built-in RAID controllers, a software volume manager on the hosts is employed to provide mirroring.

The remaining failure point on the servers is the network interface. On RAC nodes, a minimum of two network interfaces (NICs) are required: one for the public network and one for the private (or interconnect) network. If either of these NICs fails, the node is no longer able to participate effectively in the cluster. Depending on the specific failure, the node may either reboot (i.e. private NIC failure) or continue working without allowing client access (i.e. public NIC failure). To address this single point of failure, it is common for multiple physical NICs to be logically bonded or trunked together to provide redundancy. When a NIC fails in these configurations, the second physical NIC seamlessly takes over the connection and ensures that the node remains in communication with its peers. At a later time, maintenance can be scheduled to service the failed NIC. It is also important to note that multiple, cross-connected network switches should be employed in this configuration so that the two NICs are not connected to the same network switch, creating a single point of failure at the switch.

Bottom line: make sure you avoid making the NIC a single point of failure. The only alternative is to accept that a NIC failure will essentially translate into a node failure. In some (most likely larger) clusters, that may be a reasonable tradeoff. Consider that a NIC is usually relatively inexpensive and losing the processing power of an entire node may not be inexpensive in terms of diminished capacity to process work (business).

CROSSOVER CABLES

Though the number is getting smaller I think, there are still a few places in the wild where one can find two-node RAC environments being constructed with crossover network cables for the private interconnect. This is not an Oracle supported configuration for Oracle RAC and unlike some things are unsupported but still work (like RAC on VMWare), using crossover cables will fail in most cases.

The reason for crossover cables failing has to do with the way some operating systems detect NIC failure. Most OSes detect NIC failure when the NIC loses its link. The link is the indicator that something is attached and powered on at the other end of the crossover cable. Therefore, when one node is shutdown or disconnected from the crossover cable, the other end loses its link and both nodes believe that their NICs have failed. The outcome in this situation is undesirable. When each node is connected to a switch, then the link status for each NIC only relies on the status of the switch to which it connects. Each node may independently drop its link or disconnect from the switch without affecting the link status of the other node.

USING VIPs

A Virtual IP (VIP) address is created during Clusterware installation and managed by the Clusterware. The TNS Listeners in RAC clusters bind their listening endpoint to the VIP address so that users can connect to the listener on each node via that node's VIP address. The reason that the VIP is used instead of the node's public IP address or name is because of TCP timeouts. When a node fails unexpectedly, the client needs to get a response back telling it that the node is no longer online. In this case, if the user connects to the VIP and the node fails, that node's VIP is taken over by a surviving cluster member. The client attempts to make connection to the VIP just as it had before and immediately receives a response that its connection is no longer valid. Oracle clients will then attempt to reconnect to another VIP in the cluster and very likely will establish a connection to a surviving instance where it may have a session already established in the database.

When a failed node's VIP is taken over by a surviving cluster node, it does not accept any connections. It is only there so that clients will receive an affirmative response that their socket was destroyed so that they can quickly enact their mechanism to locate a surviving instance for reconnection.

The error often made by RAC administrators is to allow connection to the public address of the RAC nodes instead of forcing all database connections to be made to the VIP. By default, the Network Configuration Assistant (netca) will configure the listeners to listen on both the VIP and the public address, which allows users to connect to the wrong location (public address). To fix this configuration, the listeners should be manually reconfigured to only allow connection to the VIP address and nothing else. The VIP address is a prerequisite resource for the listener resource, so it is safe and best practice to configure the listeners in this manner. For more information including specific configuration parameters and instructions, see my blog entry on the topic at <http://www.dannorris.com/2008/07/21/tns-listener-configuration-for-oracle-rac/>.

STORAGE

Storage is often one of the more difficult configuration areas to get correct during an installation. However, once configured properly, it is usually very stable and doesn't require much maintenance outside of adding additional storage later.

CONFIRM CERTIFICATION: CFS, ASM, RAW DEVICES

Beginners are sometimes unsure where to start when determining how to manage the shared storage required for RAC environments. As most know, there are several options, each with its own advantages and disadvantages. In current releases, you can use cluster filesystems (OCFS, VxFS, and GPFS are relatively common), ASM, or, less commonly, raw devices. Today, cluster filesystems are still the most common choice, but ASM is gaining significant popularity.

Before making a final decision, check to see if Oracle supports your choice. ASM is supported on all platforms where RAC is supported, so no additional verification is necessary if you're using ASM. If you choose a cluster filesystem, make sure that the certification matrix on the support site (now called "My Oracle Support") contains your OS/RDBMS version/CFS combination. Not all cluster filesystems are supported for RAC. You may also find additional information (such as mount options, special kernel parameters, or additional patches required) in support notes on the support site, so be sure to check there as well. Oftentimes, if you attempt to install in an unsupported configuration, the installer will fail, but not always do a good job explaining the cause of the error.

MULTIPATHING

For configurations that take advantage of multipathing, take special care to ensure that the device used is the correct one. In some configurations, especially those using ASM for storage management, the multipath device isn't the one used to access the storage. In this type of configuration, the storage access will likely work just fine and may even sustain

the failure of one or more paths, as long as they're not the one path being used, but eventually there will likely be a time where an undesirable failure will occur. When that happens, many will question why multipathing didn't work. The answer is usually that it worked fine, but the storage access wasn't configured to use the multipath device and instead used one of the many individual paths.

When configuring storage for ASM, make sure that the `asm_diskstring` parameter is set to a value that ensures that only the multipath pseudo-devices are found and that the direct individual paths to each device do not match the parameter's value. This will ensure that ASM will not be able to access the storage devices except via the multipath device. So, when a storage path failure occurs, ASM should continue processing fine since the multipath subsystem should ensure that the multipath pseudo-device never loses access to the storage it represents and simply reroutes storage traffic via an alternate path. Some multipath drivers are also able to aggregate bandwidth from multiple paths, allowing for increased performance in high-I/O-bandwidth applications.

PEOPLE AND KNOWLEDGE

Many of the issues discussed in this paper ultimately roll up to the larger heading of lack of knowledge. Those that don't understand what they're doing very well can successfully complete creating a RAC cluster thanks to increased ease of installation and improved Oracle Universal Installer checks. However, installing a cluster and making it perform well, keeping it stable, troubleshooting its errors, or making it scale well are topics that require a deeper understanding of the product and the configuration.

TRAIN DBAs

This is not an advertisement for Oracle University training. I've met many DBAs in the world and they all learn in slightly different ways. Some get the most benefit from traditional instructor-led training sessions, some do better with mentoring from a consultant, others learn the most by building a test system and relying on internet resources like the Oracle-L mailing list or user groups like IOUG. My point is that, regardless of the training style employed, DBAs taking on RAC need to first take on the knowledge required to ensure its success. You wouldn't get behind the wheel of your car without first seeking lessons or training because it'd be reckless and dangerous to drive without them. While it is not necessarily life-threatening to attempt RAC management without training, it is dangerous to the organization's goals to entrust its most significant asset (data) to a DBA that lacks both experience and knowledge. Also keep in mind that RAC is often employed because of its high-availability characteristics and organizations count on that availability when they purchase and deploy RAC. However, without proper knowledge, there is likely to be more downtime with the more complex RAC environment than with a single-instance environment if proper training isn't part of the deployment plan.

TRAIN STORAGE ADMINS

Similarly, but possibly to a lesser extent, unusual requests will be made of the storage administrators since it is relatively unusual for two or more servers to share the same storage at the same time. This is often a challenge to explain to storage administrators when starting a RAC configuration since they believe that such a configuration will certainly corrupt the data. Additionally, for those sites choosing to use ASM for storage management, a determination must be made regarding who will manage the ASM environment. While it is possible, and possibly advantageous, to have the storage or system administration team manage the ASM environment, it is still not commonly the case. Furthermore, having DBAs managing storage via ASM often makes for challenging management situations as internal groups raise the question of which group *should* manage the storage. The Oracle Database 11g server documentation organization includes a separate book for storage administration that is intended to allow ASM management to be addressed separate from the traditional DBA team. It is for some of these non-technical reasons (like who should manage ASM) that ASM adoption has been slower than many of us would like to see.

Regardless of how deep the training is, storage administrators should be offered the chance to understand what happens to the storage at a higher level (like database or server-tier) to ensure that they properly configure and monitor what is allocated. During troubleshooting scenarios is not the best time to wish that the storage administrators had been given some additional training.

One last note on training for storage administrators: today, there is no ASM training course available from Oracle University. In fact, I don't know of any other courses available from any vendor that focus exclusively on ASM. While there has been a good Oracle Press book published on ASM (see references for title and information), its details are focused on the DBA audience and are likely more in-depth than what the typical storage administrator needs or wants to know. Today, I believe one of the best options for ASM training is to engage a consultant that can provide some hands-on mentoring with an outline of topics to cover including storage configuration, host configuration (ASMLib, multipathing, etc.), and diskgroup creation and maintenance. Such training would likely be 2-3 days long for most storage administration teams. If it goes much longer than that, information overload sets in and may be counterproductive.

TRAIN DEVELOPERS

The final frontier: Developers. While they're often the last to be included in thoughts of training for RAC, that's a huge oversight as developers have the greatest ability to help (or harm) the database environment. While we've all heard that RAC doesn't require code changes in order for the application to benefit, and that might be true in some cases, for most cases, there will be significant advantages if the application code is reviewed and probably modified in some places. While I've never seen an application that completely failed in a RAC environment, I have seen more than one that performed worse under RAC than on a single-instance database. Those same applications, with some key modifications, ran as well or better in a RAC environment.

The key knowledge that will help make developers successful when creating applications for RAC are the impact of connection pooling, what happens during session failover/failure, disadvantages of doing small, sub-atomic transactions, and the advantages of using services. These are obviously in addition to all other design principles of Good Applications (like inserting a whole row instead of inserting and then immediately updating it, for example). Sources for training for developers are some of the great books that have been written on Oracle database design, performance tuning, and some of the great presentations and articles written by those that have "been there, done that" and shared it with the community. You'll find many such papers and presentations in the conference proceedings from Collaborate (IOUG track especially) and Oracle OpenWorld (mainly customer presentations). In addition, Tom Kyte's "Effective Oracle by Design" and Jonathan Lewis' "Cost-Based Oracle Fundamentals" books are excellent resources for developers and DBAs alike. While most of what is contained in these two books are not specific to RAC, they provide a strong foundation for good application programming that will ultimately benefit RAC-backed applications as much or more than single-instance applications.

TRAIN MANAGEMENT

This category of IT professional isn't really one that needs training in the traditional sense. More accurately, managers may need to be "untrained" by instructing them what realistic expectations really are. While it is true that many applications can work well on RAC, a relative few make the transition without bumps along the way. The RAC migration path is often laden with performance diagnosis and code modification. So, it is important to allow sufficient testing, debugging, and break-fix time into the implementation cycle. This is especially important because sometimes they get the impression that migration will be "seamless" and "easy" for their applications. In reality, it is impossible to know whether migration will be either of those things until you test it thoroughly and completely.

Once you complete the implementation, a whole new set of expectations take over. The operational expectations, especially those related to maintenance, can often be set too high. At present, most clusters are running Oracle 10g R2.

With Oracle 10g R2, there are several maintenance events that require a complete cluster outage to complete. Most patches, some database maintenance, and some clusterware patches or upgrades may require cluster-wide outages.

Besides operational and uptime expectations, the application behavior and scalability are also frequently expected to be higher than is possible. Sometimes the application can be modified to achieve the desired goals, but in other cases, modifications aren't possible or it isn't easy to determine what needs to be changed. When this happens, expectations may need to be lowered or additional effort required to identify and make the changes required to meet the goals. Regardless, there is usually a significant gap between "it works" and "it works, performs, and scales well."

APPLICATION AND/OR FAILURE TESTING

Application testing is probably the most common area where I see RAC-related issues. While I consider myself well informed about RAC and educated by both training and experience, I don't believe anyone can tell whether an application will behave well under stress in a RAC environment without actually testing it. For complex applications, like Oracle E-Business Suite or Siebel, even the same application may behave differently at different customer sites because of different configurations, hardware, data, or other factors.

STRESS TESTS

Of all the testing conducted, it is stress testing that is the most important and perhaps the most challenging to conduct. A majority of organizations have no capability to put their applications under realistic stress with respect to the database load they generate. Stress testing is the process of making the application go through its normal activity at an increased or peak rate. Typically, this will also create the peak database load. The hard part of this test is to create a *realistic* load on the database. Some organizations can create a load on their application, but it doesn't often include the proper variety or proportion of actions which means that it doesn't accurately represent the production load.

Starting with Oracle Database 11g, the Database Replay feature allows you to capture actual **database** workload from your production database (from 9i R2, 10g R2 or 11g) and replay it on the 11g database. The source and destination databases can be any combination of RAC or single-instance that makes this feature an excellent tool to use for database-tier stress testing. Unfortunately, this feature is an extra licensed option, so will not be available to all customers. For other customers, there are other application-tier load generator tools (Oracle Enterprise Manager now offers one too, but I believe that's also an extra-cost option) that can be programmed to simulate many users accessing the application. These solutions are approximations and can also require expensive tools and significant effort to configure properly.

No matter what method you choose, stress testing your application using a realistic or the closest approximation possible is very important. There have been too many applications that go into production without proper stress testing and then spend many weeks debugging issues and performance problems after the go-live. Even if stress testing takes several weeks to conduct, consider that not properly stress testing the application can cause outages or "brown-outs" that can last more than several weeks. For customer-facing applications, such outages or degradation can result in lost customer revenue. It may not be easy, but it is very important and will yield significant benefits for those that do it seriously and competently.

FAILURE TESTS

RAC is almost always expected to provide the highest degree of availability for the application(s) it serves. Therefore, testing failures should be part of every new deployment. Specifically, failure testing of the actual application (not just a sample database) is absolutely mandatory to ensure that the expected behaviors are tested, observed, and verified.

To put a cluster into production without testing all single points of failure is irresponsible. For example, if you expect that multipathing should handle a path failure for a storage device, then run your application, put it under stress (since many

failure are more likely to happen under stress), and pull a cable out of one or more of the servers. In fact, if your multipathing is active/passive, ensure you pull the active cable. Observe any application or database errors and measure response time performance from a user perspective during the test. After all, we all know that Murphy's Law is alive and well. Some of the single points of failure that should be tested are:

- Storage path
- Network interface (if you configure redundancy for NICs)
- Disk drive
- Oracle instance
- RAC cluster node
- Session death (kill an active session)

For each of these failures, you should collect information about application behavior, performance, and time to recover. What you'll likely find is that some transactions or session state may be lost in some cases. This may cause application errors or unexpected errors to be displayed to the user. During failure testing, it is common to add application code to handle the additional special exceptions that Oracle may throw as a result of the failure.

During testing, it is always good to attempt to predict the outcome of any given failure and then verify it. You may also find during this exercise that the business prediction (or desire) and the technical predictions are different. This usually opens discussions about setting proper expectations, which are necessary and important discussions to have prior to the production release.

CONNECTIVITY

Connectivity issues usually involve failure testing, but also include connectivity configuration. First, failure testing is an essential part of understanding RAC connectivity. Understanding the difference between `BASIC` and `PRECONNECT` as well as `SESSION` and `SELECT` failovers are keys to predicting how your application will behave when a failure occurs. Those configurations are well-documented and relatively easy to test in any cluster using `sqlplus` or other similar tools.

More importantly, the configuration of load balancing, both client and server-side, as well as the configuration of listeners and the proper instance-level parameters to register with the proper listeners is very important and often overlooked. There have been several articles written on this topic in addition to the standard server documentation, some listed in the references of this paper. In short, both client-side and server-side load balancing should be configured and enabled. All connectivity with the listeners should be via the VIP addresses on the cluster and the listeners should only be configured to listen on these VIPs.

The other connectivity topic that deserves mention here is Oracle Services. Services were introduced in Oracle8i, but most sites didn't know much about them or make good use of them until at least Oracle 10g. Oracle RAC has helped increase awareness of this great built-in feature due to the integration between services and Oracle Clusterware. Services are the best way to manage workload in a cluster. Services are created with and managed by Oracle Clusterware to provide a way to group users or applications together. Within the database, statistics are tracked on a per-service basis and resource management can be applied to services, providing excellent ways to measure and control groups of users or applications that share the RAC database environment. For more information about services, see the references section of this whitepaper.

ATOMIC TRANSACTIONS + CONNECTION POOLING

More than once, I've helped troubleshoot a RAC database performance issue where the application it supported ran better (i.e. faster) on a single-instance database than it did on RAC. In a few of those cases, a similar root cause was identified.

The basic scenario is that the application performed several small updates, each one its own transaction, and all of the updates operated on the same set (sometimes the same row) of data. These updates were performed from a connection pool and it seemed that somewhere in the development process, a pattern emerged where only a single change was made each time the connection was obtained from the pool. Once the small change was made (and committed), the connection was returned to the pool. Since many updates happened on this same subset of data within a short period, there were many requests to the connection pool. The connection pools reside in the application tier, so there is no direct impact on the database when the connection pool is overworked.

However, in a RAC environment where the connection pool may have connections to more than one instance in the cluster, a potential problem emerged. The problem is that when a series of transactions all deal with the same subset of data (especially when they're updating that data), it is likely that each request for a connection from the connection pool may obtain connections to different instances in the cluster. This requires RAC to send the data upon which the application is operating over the interconnect to other instances in rapid succession and essentially "thrashes" the Cache Fusion mechanism in RAC. As the data is shifted from instance to instance in order to support the requests from the application, waits are introduced by waiting for CPU and/or network transfers to occur.

While RAC was blamed initially, the real issue was a poor application design choice that was exacerbated by RAC. The fix was relatively easy: do less work! Once the proper code was identified, it was relatively easy to remove the "extra" calls to return the connection to the pool and obtain a new connection from the pool for each of these small updates. The transaction logic was also modified to reduce the number of commits that the transaction performed as this also improved performance. Once the changes were made, the application was able to perform as well or better on RAC than it did on the single-instance environment. In fact, it even performed better than it had before on the single-instance environment, proving that RAC was really just the debugging tool in this case, not the root cause of the issue.

The moral of the story: Plan your transactions. What seems to have no difference or impact may later prove to be a serious problem when the environment changes.

PERFORMANCE TUNING, INTERCONNECT

A common mistake for new RAC database administrators is to forget all they know about Oracle Database tuning and instead start looking at the most detailed settings in the environment before reviewing the basics. Tuning a RAC database is not very different from tuning a non-RAC database. The first step (diagnosis) is the same in either case, RAC or single-instance and the same tools should be used (OS monitoring tools, AWR, ADDM, statspack, SQL tracing, common sense, et cetera).

However, there does come a time when those initial diagnostic findings will indicate a symptom that you can only find in a RAC environment. These "special" symptoms are the issues directly related to the interconnect and/or Cache Fusion in RAC.

The typical wait events you may observe when the interconnect is involved in the problem are those including the words "global", "GCS", or "GES". For example, "global cache busy" or "GCS lock open X" are two events that may occur related to RAC.

In these cases, the wait events may be related to an issue with the interconnect such as a hardware issue, duplex negotiation mismatch, or misconfiguration. However, it is much more common that these symptoms are caused by the same things that cause many other wait events: application behavior or code. The scenario described earlier in the section that discusses connection pooling and atomic transactions is one example where the symptom is a lot of interconnect traffic and waits on the global cache service (GCS) to transfer the necessary information from instance to instance. However, the interconnect had no hardware issues, wasn't performing poorly, and showed no signs of stress (plenty of capacity remaining). Yet we observed that the wait events were almost entirely related to global cache events.

The bottom line is that you should monitor the interconnect traffic and check the interconnect network for errors. However, when all symptoms indicate a potential interconnect problem, don't automatically think that you should go out and buy additional network cards or upgrade to Infiniband. Sometimes, the issue is one that just didn't appear as an issue until RAC came along.

SEQUENCES

Sequences are often forgotten since they are “small” and references to them are often nested deep within application logic. In a RAC environment, sequences can pose problems that aren't observed in non-RAC environments. For example, to guarantee sequential numbers without gaps, some customers set `NOCACHE ORDER` for their sequences. With RAC environments, such a configuration provides the worst performance since each sequence request may potentially have to wait on a remote instance to send the current sequence information (the block from seq\$) over the interconnect. However, this configuration has the highest likelihood to provide a consecutive, no-gap set of numbers for all processes in the cluster. It also is likely to cause the highest amount of waiting.

The recommended configuration for sequences in RAC is `CACHE 1000` (or higher if you have very high transaction rates) and `NOORDER`. The default `CACHE 20 NOORDER` is usually not sufficient for most applications and performance will likely improve if you use larger cache amounts. The larger the cache size, the larger the potential for gaps and overlaps in the sequence numbers. All sequence numbers will be unique, but the sequence number 2503 may possibly be used before the sequence number 1906 if one instance performs a large amount of transactions (using up more of its cache) than another instance. This may impact your application and business rules must be considered in order to ensure that processes don't always assume that the sequence numbers (which may be invoice numbers, for example) are sequential.

When diagnosing issues with sequences, you are likely to observe heavy Global Enqueue Services traffic due to Global TX and TM enqueues, contention for the SQ enqueue, waits on “DFS Lock Handle”, and/or high wait times for “row cache lock”. Besides adjusting the `CACHE` value for sequences to be higher, you should also consider using `DBMS_SHARED_POOL.KEEP` to pin the sequence in the shared pool rowcache and reviewing your indexes performance for possible block split contention.

REFERENCES

- Jeremy Schneider, “Unleashing Oracle Services: A Comprehensive Review of “Services” in Oracle Databases”, <http://www.ardentperf.com/pub/services-schneider07.pdf>
- DanNorris.com blog article “TNS Listener Configuration for Oracle RAC”, <http://www.dannorris.com/2008/07/21/tns-listener-configuration-for-oracle-rac/>
- Joel Goodman, “Managing Sequences in a RAC Environment” presentation and webcast (8-May-2008), Oracle RAC SIG, www.oracleracsig.org

FROM THE LAWYERS

The information contained herein should be deemed reliable but not guaranteed. The authors have made every attempt to provide current and accurate information. If you have any comments or suggestions, please contact the author at [dannorris\(at\)dannorris.com](mailto:dannorris@dannorris.com).