Session 1

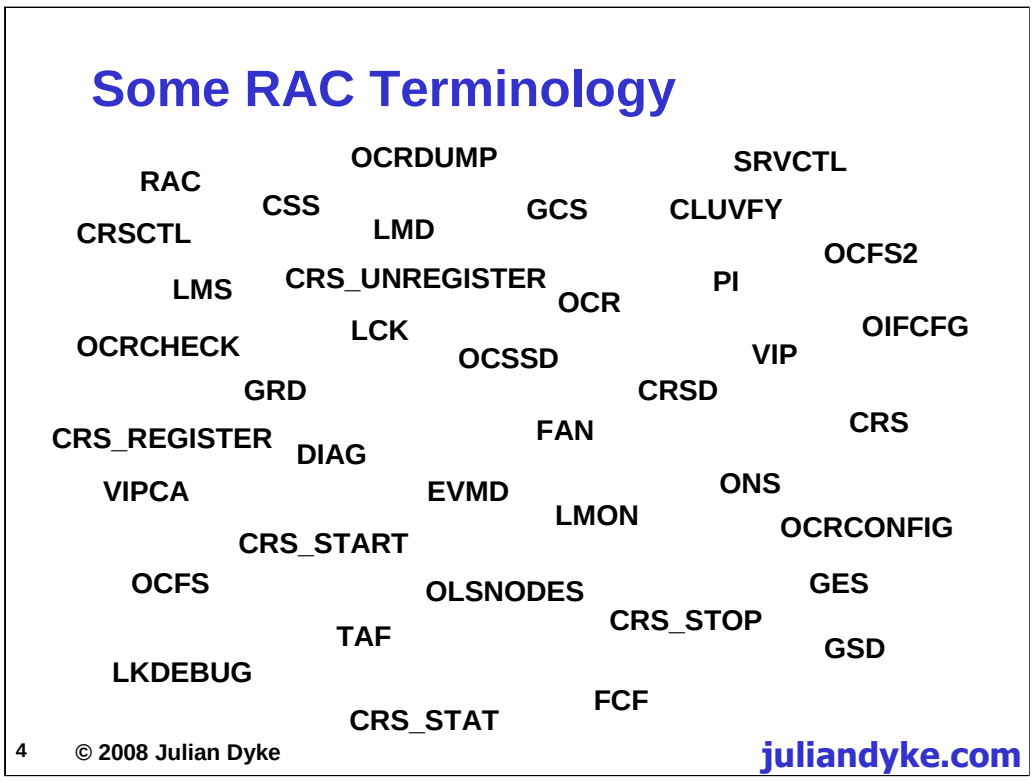
Introduction and Concepts

Advanced RAC
Auckland - May 2008

Agenda

- Introduction
  - Definitions
  - Background
  - OPS versus RAC
  - Reasons for Deployment
  - Alternatives
- Concepts
- Discovery

Introduction



Even if you are an experienced Oracle DBA, there is a lot of new terminology that you might need to understand in order to work with RAC. The above slide shows some of the RAC specific features, services, background processes, tools and utilities that we will encounter during this seminar.

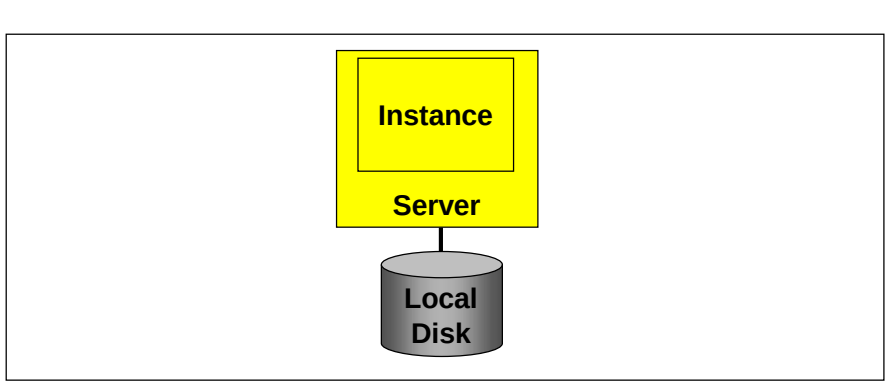
We will also examine many features and concepts commonly found in single-instance databases, that have extended or enhanced functionality in a RAC environment such as

- Oracle Universal Installer (OUI)
- Database Configuration Assistant (DBCA)
- Automatic Storage Management (ASM)
- Enterprise Manager (EM)

and many more...

Definitions
What is Single Instance Oracle?

- Single instance running on individual server
- Single database on dedicated storage
  - DAS - Direct Attached Storage



5 © 2008 Julian Dyke juliandyke.com

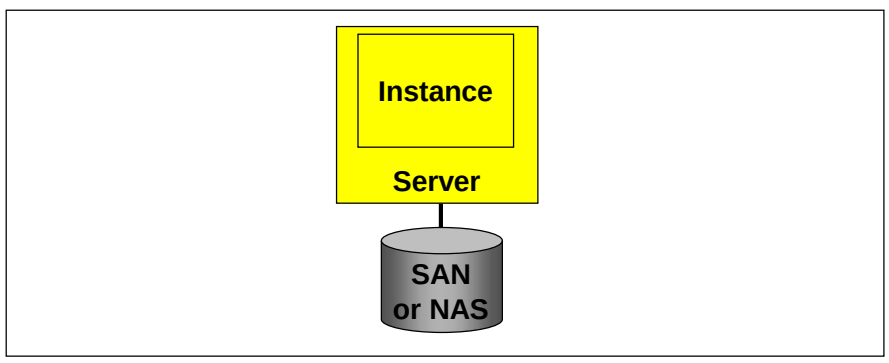
Oracle databases can be single-instance or multi-instance. A multi-instance Oracle database is known as a clustered database and is implemented using the Real Application Clusters (RAC) option which is an additional cost option for Enterprise Edition users.

A single instance database contains a set of datafiles which are stored on disk. Originally these datafiles were stored on disks which were locally attached to the server using a technology such as SCSI. This type of storage is still in frequent use for smaller Oracle database

An instance runs on a specific server. It consists of an area of shared memory and a set of co-operating processes. The processes maintain the state of the database in the memory structures and manipulate the data in the database.

Definitions
What is Single Instance Oracle?

- Single instance running on individual server
- Single database on networked storage
  - SAN - Storage Area Network
  - NAS - Network Attached Storage



6 © 2008 Julian Dyke juliandyke.com

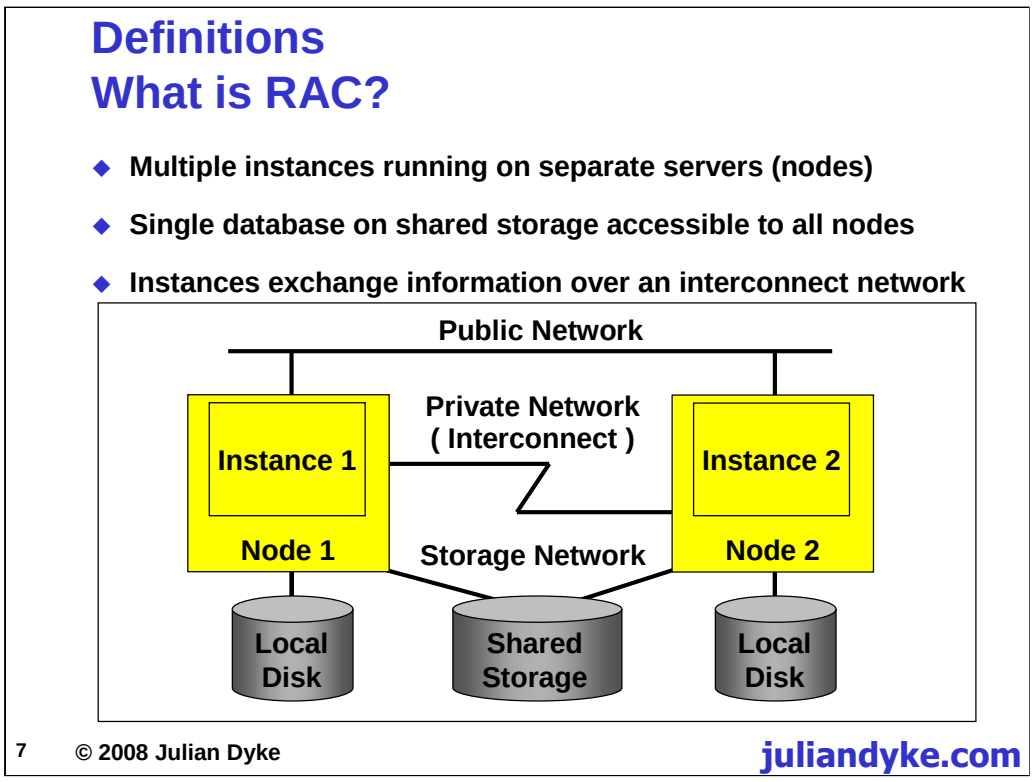
In the 1990s, it was recognized that storage could be pooled between servers. This brought economies of scale and simplified management where there were large numbers of servers. For example, backups could be standardized and disks could be allocated where the need was greatest.

Two types of storage subsystem developed, SAN and NAS.

SAN - Storage Area Network - In a SAN, servers are usually connected to the storage device via fibre cables. At present these are normally 2Gb, but during 2006 4Gb fibre is becoming common. Disks within the SAN are organized into disk groups or LUNs which are allocated to disk controllers. Almost all SANs have at least two disk controllers to ensure redundancy.

NAS - Network Attached Storage - NAS disks are organized similarly to those of a SAN. However, the server communicates with the NAS using a modified version of NFS over a TCP/IP ethernet network.

Both SAN and NAS technology can be used to store datafiles for single-instance databases and RAC dartabases.



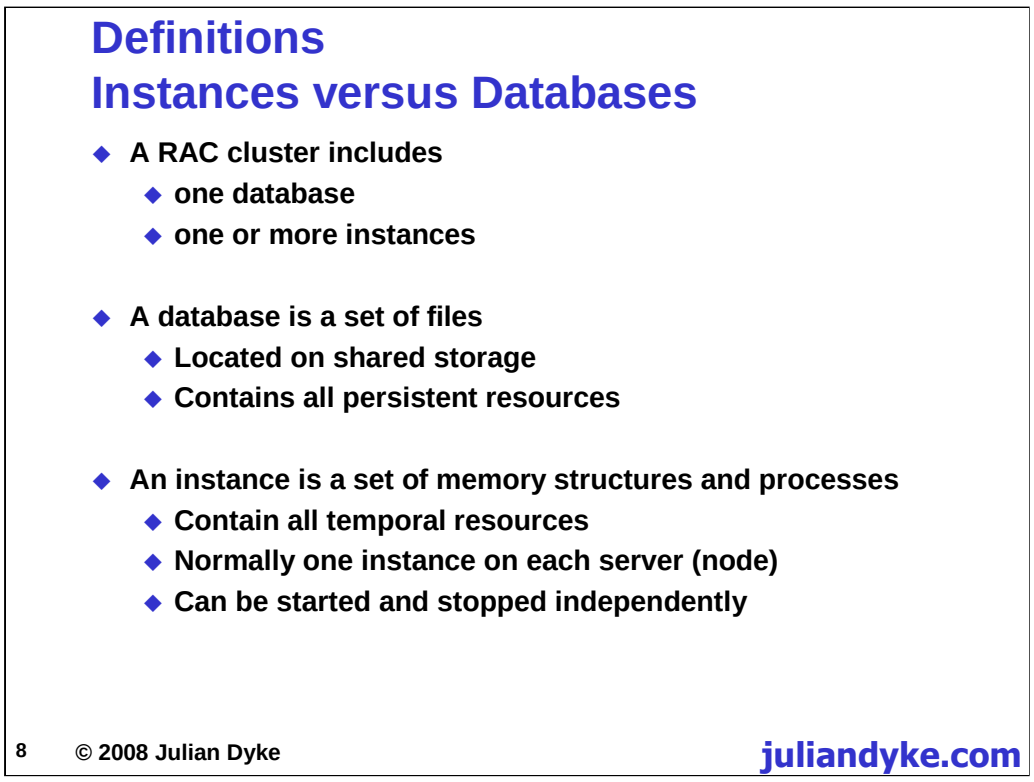
A RAC database consists of multiple instances which run on separate servers (known in RAC terminology as nodes).

There is a single copy of the database which must be located on shared storage that is accessible to all nodes. The shared storage is usually a Storage Area Network (SAN) or less frequently Network Attached Storage (NAS). Each server can optionally include locally attached storage which usually contains the operating system image, swap space and optionally the Oracle binaries.

There are usually a minimum of two instances in a RAC cluster. However, the cluster software must be capable of running on a single instance in order to allow the database to be started up, shutdown and to provide failover capabilities in the event of the failure of the other nodes.

Each instance can update any data in the database. To avoid problems with changes being overwritten, instances maintain global locks which apply to all instances in the cluster.

The instances exchange information over a dedicated private network known in RAC terminology as the interconnect. The interconnect is used to transmit information about locks held on specific resources by the instances. The interconnect is also used to transfer blocks from one instance to another, thereby avoiding the need for one instance to write changed (dirty) blocks back to disk before forwarding that block to another instance.



It is important to understand the difference between a database and an instance. If your experience is on single-instance databases, which still represent more than 90% of Oracle installations, you may not be entirely clear about the distinction between these concepts.

A database is a set of files that is located on shared storage. With a few exceptions all persistent data is stored in the database. Exceptions might include parameters, passwords and network configuration.

On the other hand, an instance is a set of memory structures and processes. There is normally one instance on each server (node). Each instance can be started and stopped independently of its peers.

When all instances are shutdown, the database is closed. If one or more instances is started, then the database can optionally be mounted and opened.

Background
A Brief History of Oracle

Version	Date
2	June 1979
3	March 1983
4	October 1984
5.0	April 1985
6.0	July 1988
7.0	June 1992
7.1	May 1994
7.2	May 1995
7.3	February 1996

Oracle has been around for almost three decades. In that time it has seen significant enhancements. Oracle was originally developed for the US Department of Defence. There was never a recognised version 1. Version 2 saw the introduction of features such as joins and subqueries.

In Version 3 more features were added including transactions, rollback and before/after image journaling.

Oracle 4 saw a great leap forward with the inclusion of read consistent queries.

In Oracle 5, the client/server architecture was introduced. Oracle 5 also saw the first version of Oracle Parallel Server (OPS) for DEC VAX. This product was the forerunner of RAC.

Oracle 6 saw the introduction of the buffer cache, row level locking and anonymous PL/SQL blocks.

Oracle 7.0 was a major release which introduced the library cache, shared SQL cursors, the cost-based optimizer, stored PL/SQL procedures, functions and packages, DML triggers, snapshots and the two-phase commit. The new features matured in Oracle 7.1 with the introduction of advanced replication and parallel query. Some stability was achieved in Oracle 7.2 which included inline views. Finally Oracle 7.3, still in service at some sites today saw the introduction of bitmapped indexes, standby databases and hash joins.

Background
A Brief History of Oracle

- **Continued....**

Version	Date	Release Name
8.0	June 1997	Oracle 8
8.1.5	February 1999	Oracle 8i Release 1
8.1.6	November 1999	Oracle 8i Release 2
8.1.7	August 2000	Oracle 8i Release 3
9.0.1	June 2001	Oracle 9i Release 1
9.2	May 2002	Oracle 9i Release 2
10.1	January 2004	Oracle 10g Release 1
10.2	July 2005	Oracle 10g Release 2
11.1	July 2007	Oracle 11g Release 1

Since Oracle 8, most new features have been aimed at subsets of existing users. The Partitioning option, index organized tables (IOTs) and large objects (BLOBs and CLOBs) were introduced in Oracle 8.0, which also saw the somewhat less successful introduction of the Objects option. This release also introduced reverse key indexes which are still popular tuning tools in contemporary RAC databases. RMAN was introduced in Oracle 8.0 although the initial syntax proved to be highly unpopular with users.

Oracle 8.1.5 introduced numerous new features including materialized views, function based indexes and composite partitioning, This release also saw the first Java engine (JVM) in the database. In OPS, Oracle 8.1.5 included Cache Fusion Phase I (discussed in subsequent slides)

Oracle 8.1.6included the first version of STATSPACK. It also included cursor sharing and analytic statements.

The terminal Oracle 8i release, Oracle 8.1.7 included the first versions of Data Guard.

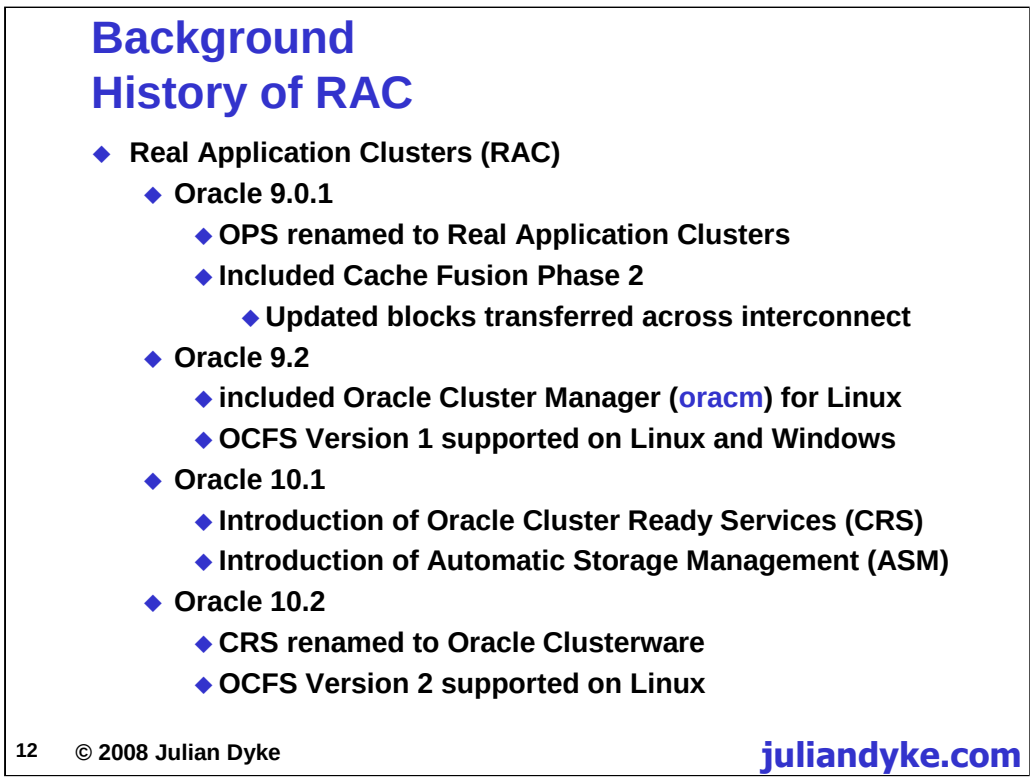
Oracle 9.0.1 saw the introduction of hundreds of new features, but by far the most important was the Real Application Clusters (RAC) option, an enhancement and upgrade of Oracle Parallel Server (OPS).

In subsequent releases, RAC has stabilized significantly and has also developed rapidly. The current version is Oracle 10.2. At the time of writing (January 2007), Oracle 11g was in beta with an anticipated release during H2 2007.

Background
History of OPS

- Oracle Parallel Server (OPS)
  - Oracle 5
    - OPS developed for DEC/VAX clusters
  - Oracle 6.2
    - First public release of OPS
  - Oracle 7,0, 7.1, 7.2, 7.3 and 8.0
    - OPS continued to be developed
  - Oracle 8.1.5
    - Included Cache Fusion Phase 1
      - Consistent read blocks transferred across interconnect

Development of the first version of Oracle Parallel Server commenced in Oracle 5 with a version for DEC/VAX clusters. OPS continued to be developed in Oracle 6 and culminated in a special release called Oracle 6.2. Development of OPS continued throughout the lifetime of Oracle 7, though there were relatively few customers using it at that time.

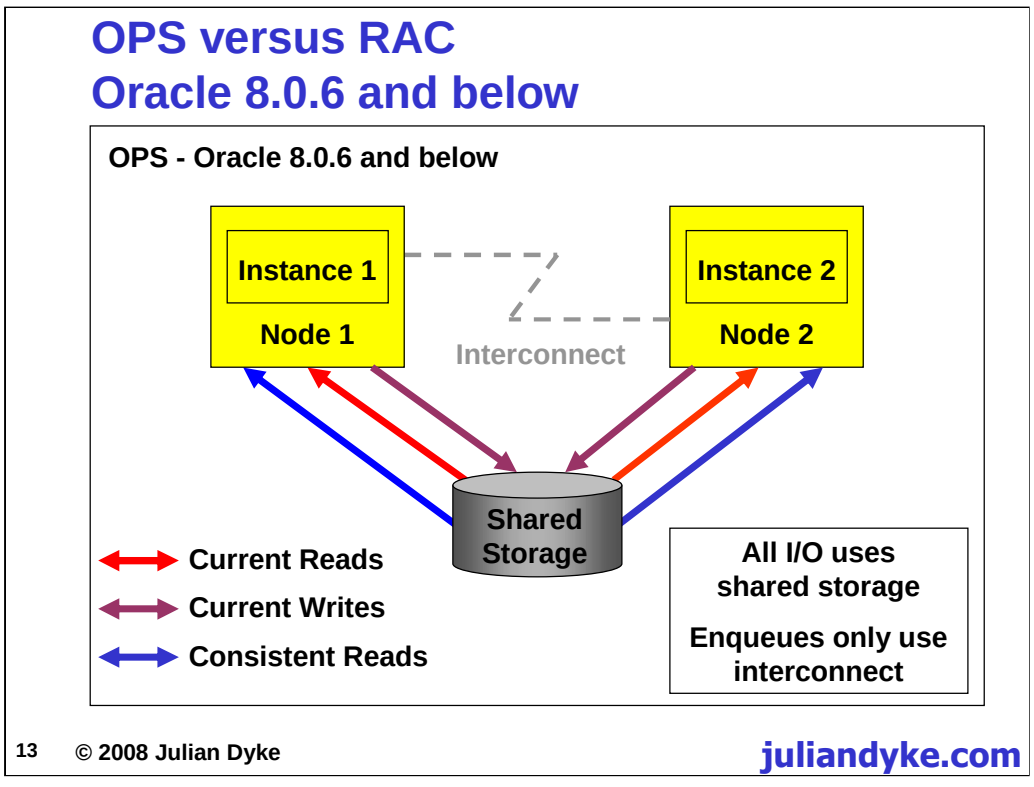


In Oracle 9.0.1, Oracle Parallel Server (OPS) was renamed to Real Application Clusters (RAC). This version included Cache Fusion Phase 2 which allows updated blocks to be transferred between instances across the interconnect.

In Oracle 9.2 and above both Linux and Windows became much more popular as Oracle RAC platforms. As Linux did not have a native cluster manager, Oracle introduced the oracm cluster manager. Neither Linux or Windows had appropriate cluster file systems, so Oracle also developed the Oracle Cluster File System (OCFS Version 1) which is supported on both platforms. On Linux only, OCFS is an Open Source product. On all other platforms, it was necessary to use a proprietary cluster manager.

In Oracle 10.1, Oracle introduced Cluster Ready Services (CRS). This is a generic cluster manager available on all supported platforms. This release also saw the introduction of Automatic Storage Management (ASM) which in a RAC environment provides shared storage for database and recovery files.

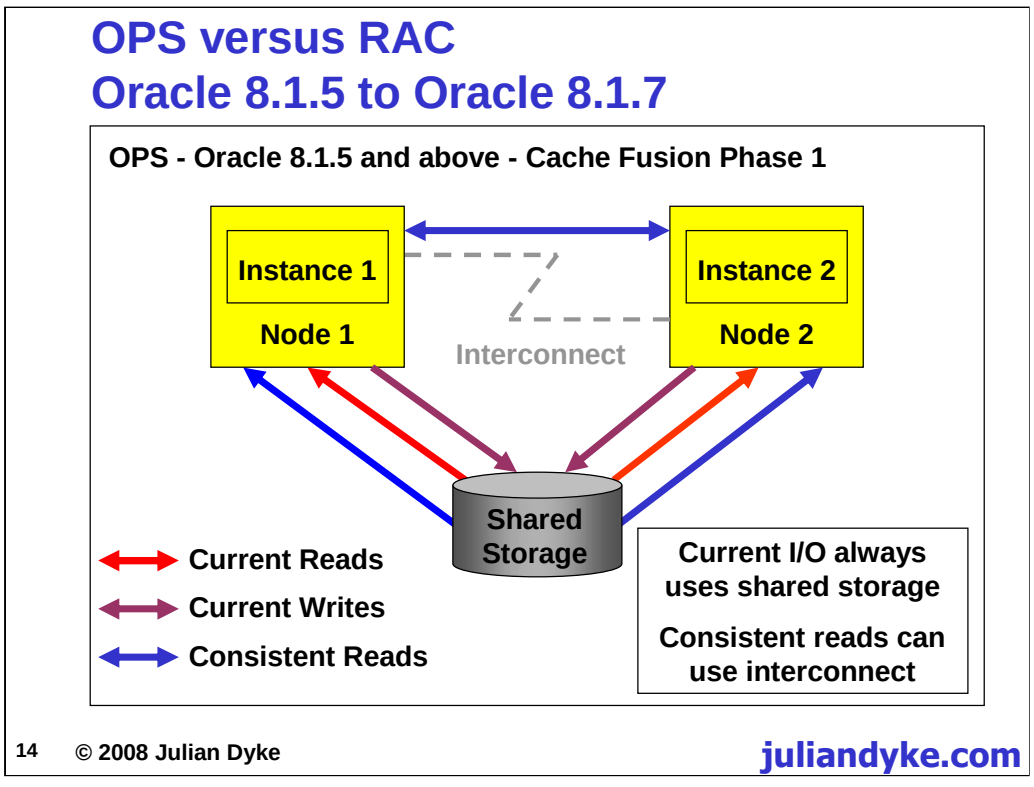
In Oracle 10.2, CRS was renamed to Oracle Clusterware and high-availability support added for third-party applications. During the lifetime of Oracle 10.2, OCFS2 was introduced which includes support for Oracle binaries on shared storage. At the time of writing OCFS2 has seen relatively low uptake possibly due to the relative strengths of ASM.



The main difference between OPS and RAC is the way that modified blocks are passed between instances.

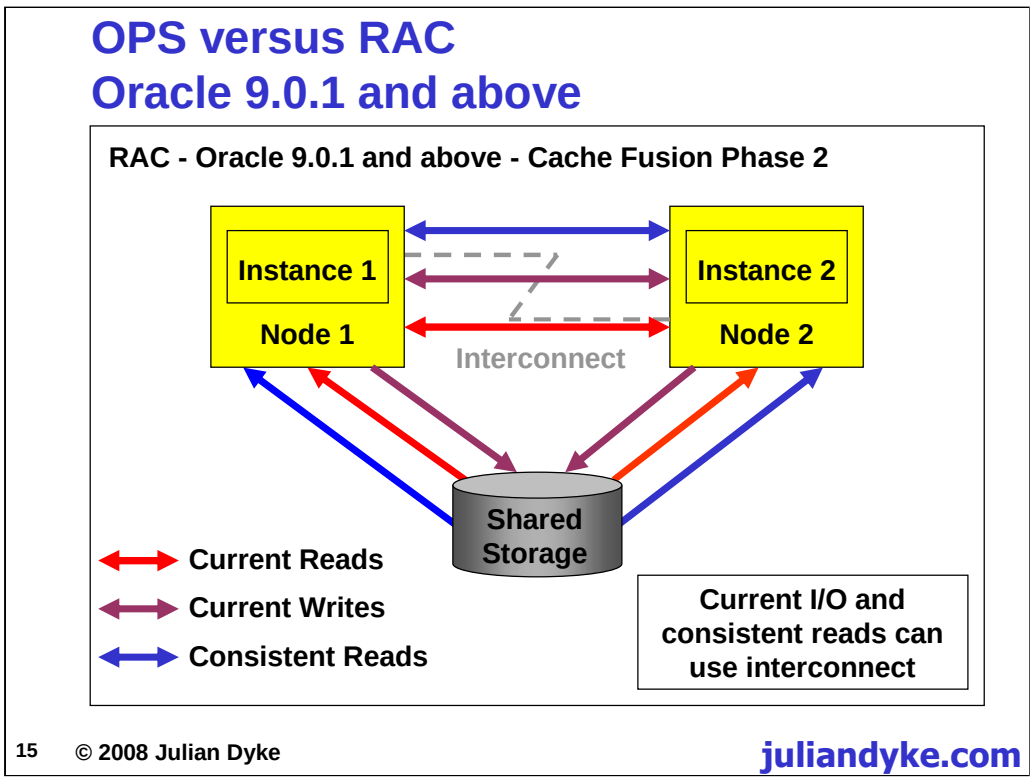
Prior to Oracle 8.0.6, OPS used disk pings to transfer updated blocks from one instance to another. When instance 2 required a block that had been updated by instance 1 but not yet written to disk, it sent a request via the lock manager to instance 1. Instance 1 wrote the updated block back to the shared storage and instance 2 then read the updated block from the shared storage.

In these releases only locks used the interconnect. In update intensive systems, disk pinging will cause a significant amount of physical I/O which will probably limit overall throughput.



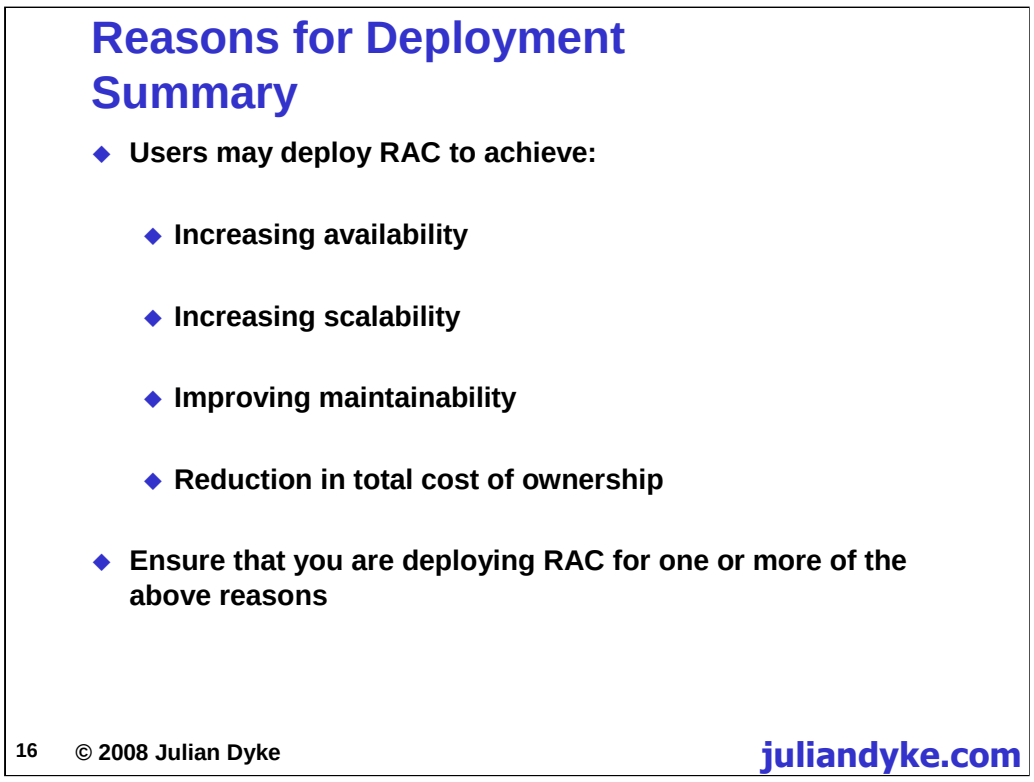
Oracle 8.1.5 saw the introduction of Cache Fusion Phase 1. If instance 2 needs to perform a consistent read on a block that has been updated by instance 1, but not yet written back to disk, instance 1 will build a consistent read version of the block from local undo and then send the block across the interconnect to instance 2.

Cache Fusion Phase 1 brought significant improvements to OPS performance. However it did not address current reads which regularly required for segment headers, index branch blocks etc and for all blocks that will subsequently be updated.



In Oracle 9.0.1, OPS was replaced by RAC partly reflecting the introduction of Cache Fusion Phase 2. In this version of cache fusion, if Instance 2 requires any block that Instance 1 has updated, but not yet written to disk, Instance 1 will send that block across the interconnect. This includes both consistent read blocks and current read blocks.

Cache Fusion Phase 2 is probably the most important concept underlying RAC and was significant enough for RAC to be judged an entirely new product. Cache Fusion Phase 2 was a great technical leap forward and uptake was significantly accelerated with the availability of Windows and Linux ports.



Traditionally the Oracle marketing has described four basic advantages for users purchasing RAC. These are

- Increased availability
- Increased scalability
- Improved maintainability
- Reduction in total cost of ownership

The justification for most users purchasing RAC, especially on Standard Edition is increased availability.

Customers purchasing more than three nodes usually expect increased scalability. Whether or not they achieve this goal is largely dependent on the application, though deployments also fail due to inflexibility within the user environment, usually the result of imposing well-thought through single-instance standards on the new RAC environment.

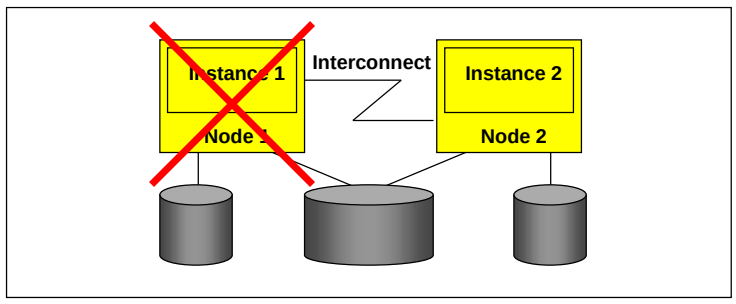
It is rare to see users deploying RAC to achieve improved maintainability though this has been achieved successfully in the past. It is also rare to hear users claim that RAC has brought a reduction in TCO.

However, the other benefits and flexibility brought by RAC usually outweigh any negative impacts.

Make sure before deploying RAC that you are implementing it for one or more of the above reasons and that you have quantified the anticipated benefits.

Reasons for Deployment Availability

- If one node or instance fails:
  - Database can still be accessed by remaining nodes
  - Sessions on existing nodes are unaffected
  - Sessions on failed node must reconnect
  - Cluster is frozen during part of this process
    - Known as brown-out period



In a RAC database failover occurs when a node or instance fails. The Oracle Clusterware running on the remaining node(s) will detect the failure of the node/instance and notify any other nodes. The node detecting the failure will read the redo log of the failed instance from the last checkpoint and apply redo to the datafiles including any undo segments. This process is known as roll-forward. Any remaining uncommitted transactions will then be rolled back. During part of this process, the cluster is entirely frozen and no work can be performed. This is known as the brown-out period.

Reasons for Deployment Availability

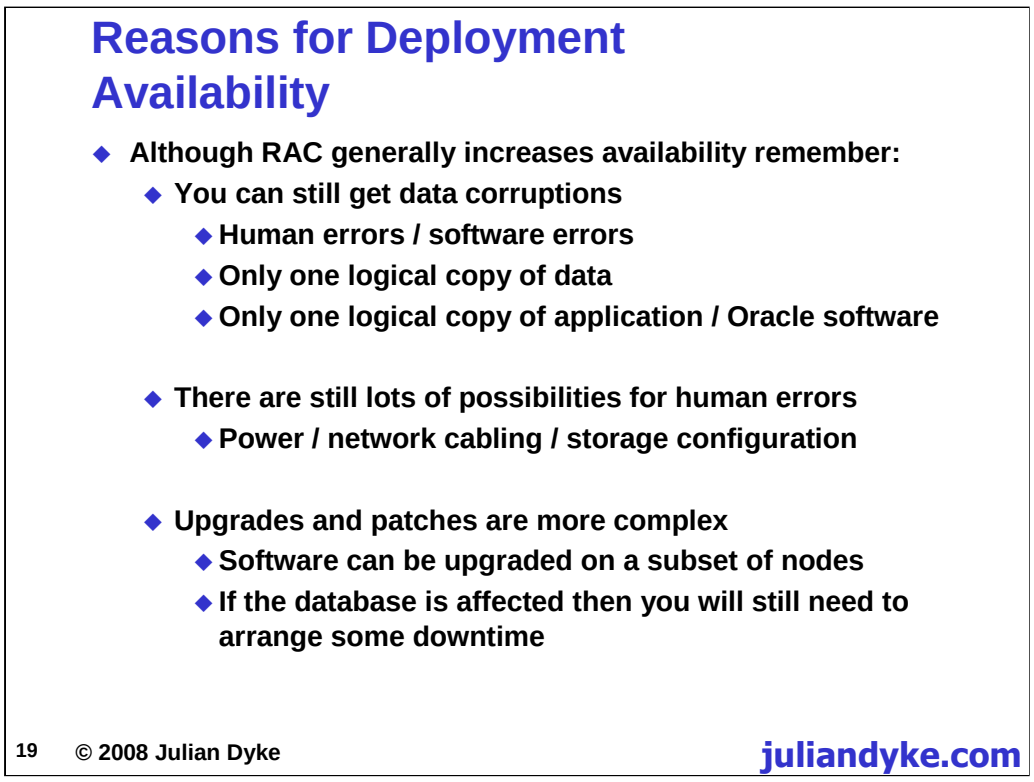
- Deploying RAC:
  - Should achieve less unplanned downtime
  - May require more planned downtime
  - Allows more time to respond to failures
- Instance failover means any node can fail without total loss of service
- There must be overcapacity in the cluster to survive failover
  - May require:
    - additional hardware
    - additional Oracle and RAC licenses
  - Load can be distributed over all running nodes

18 © 2008 Julian Dyke juliandyke.com

For almost RAC users, RAC achieves a reduction in unplanned downtime and therefore increases availability. However, because of the increased complexity of the technology stack, it is possible that the amount of planned downtime will be increased while upgrades and patches are applied.

If RAC is correctly specified and configured then instance failover means that any node can fail without a complete loss of service. Programmatic intervention will be required in the application if individual client sessions are required to be able to survive the failure of the currently connected node / instance.

In order to survive a node failure, there must be sufficient overcapacity in the cluster during normal operation. Obviously this requires additional hardware resources and also additional Oracle / RAC licences. On the plus side, of course, during normal operation, the workload can be balanced across all nodes in the cluster.



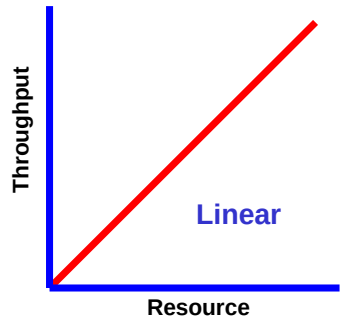
Although RAC increases the availability of the cluster, it cannot prevent data corruptions caused by human errors or software (application) errors. You should always bear in mind that in a RAC database there is only one logical copy of the data and one logical copy of the application / Oracle software.

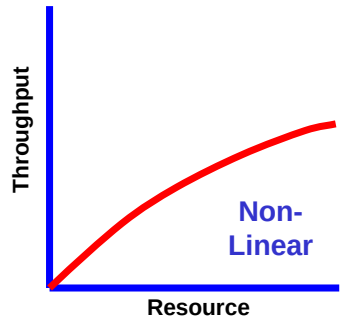
The additional complexity of the hardware in a RAC cluster provides many opportunities for human errors, often caused by ignorance of the requirements of a clustered environment. Common problems I have experienced on site include removal of power supplies, network cabling (especially the interconnect), reconfiguration of the storage, storage fabric and network switches.

In a RAC environment upgrades and patches are more complex. It is possible to bring down and upgrade some of the nodes in the cluster while the rest continue running. However, if the upgrade involves the database, it is often necessary to stop the entire cluster while the upgrades are applied to the database. After this operation is complete the upgraded nodes can be restarted while the remaining nodes are upgraded.

Reasons for Deployment
Scalability

- Scalability is the relationship between increments of resources and throughput
- Can be any resource but with RAC normally refers to adding instances
- Scalability can be
  - linear - optimal but rare
  - non-linear - suboptimal but normal





Scalability is the relationship between increments of resources and throughput. Scalability can refer to any resource, but when considering RAC databases it normally refers to the addition of instances.

The optimal case is that of linear scalability. With linear scalability the addition of a new node will bring about a proportional increase in throughput. For example if a three node cluster can process 300 transactions per second and the addition of a fourth node allows it to process 400 transactions a second then you have linear scalability.

However, there are many overheads associated with a RAC database and therefore linear scalability is rare. Most clusters exhibit non-linear scalability. For example, if a single SMP node can process 120 transactions per second, the same hardware running as a single node RAC cluster might process 100 transactions per second. A two node cluster might process 180 transactions per second and a three-node cluster might process 240 transactions per second. As the marginal number of additional transactions decreases with each additional instance, eventually a point may be reached where the addition of another node results in an overall decrease in throughput. The actual point is dependent on many factors including the application and the version of Oracle RAC.

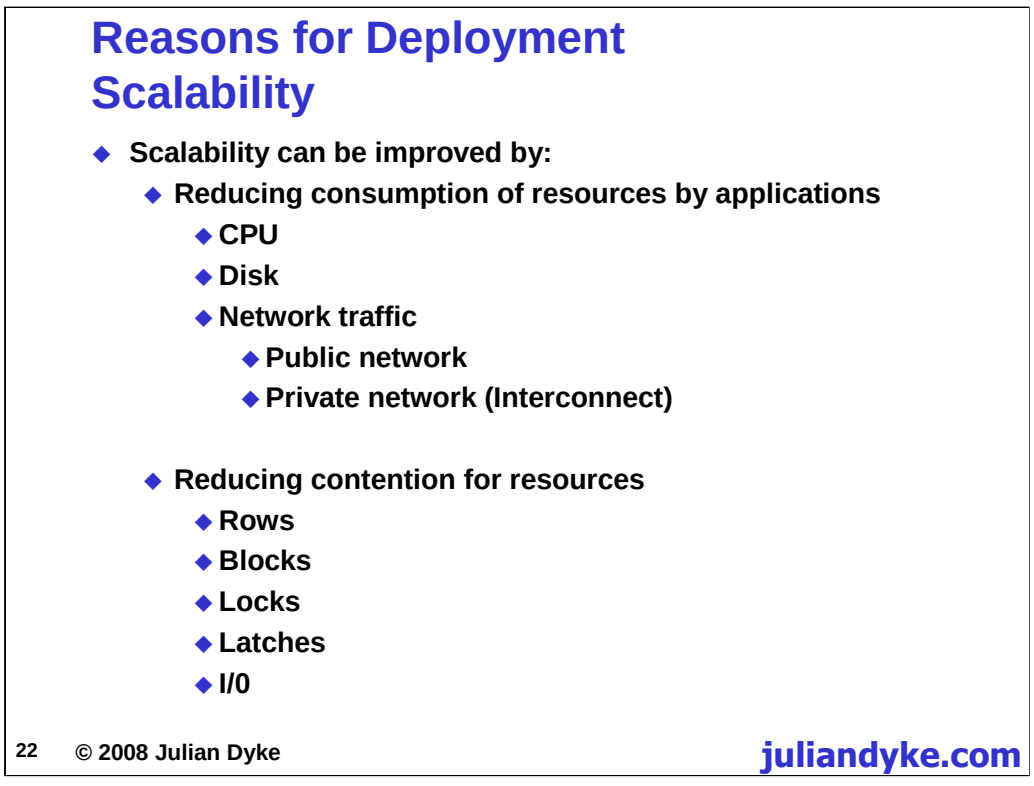
Reasons for Deployment Scalability

- RAC overhead means that linear scalability is difficult to achieve
  - Global Cache Services (blocks)
  - Global Enqueue Services (locks)
- Scaling factor of 1.8 is considered good
- Dependent on application design and implementation
- Scaling factor improves with
  - Node affinity
  - Elimination of contention

RAC imposes an overhead in terms of shared memory, CPU and I/O. Consequently linear scalability is difficult to achieve. This is mainly due to the additional cost of supporting Global Cache Services (GCS) and Global Enqueue Services (GES). A scaling factor of 1.8 for a second instance is considered good by most experienced users.

However, the amount an application scales is almost entirely dependent on the design of the application. Applications which minimizing locking, especially due to parsing and also which achieve high levels of node affinity for database blocks (logical partitioning) are most likely to scale well.

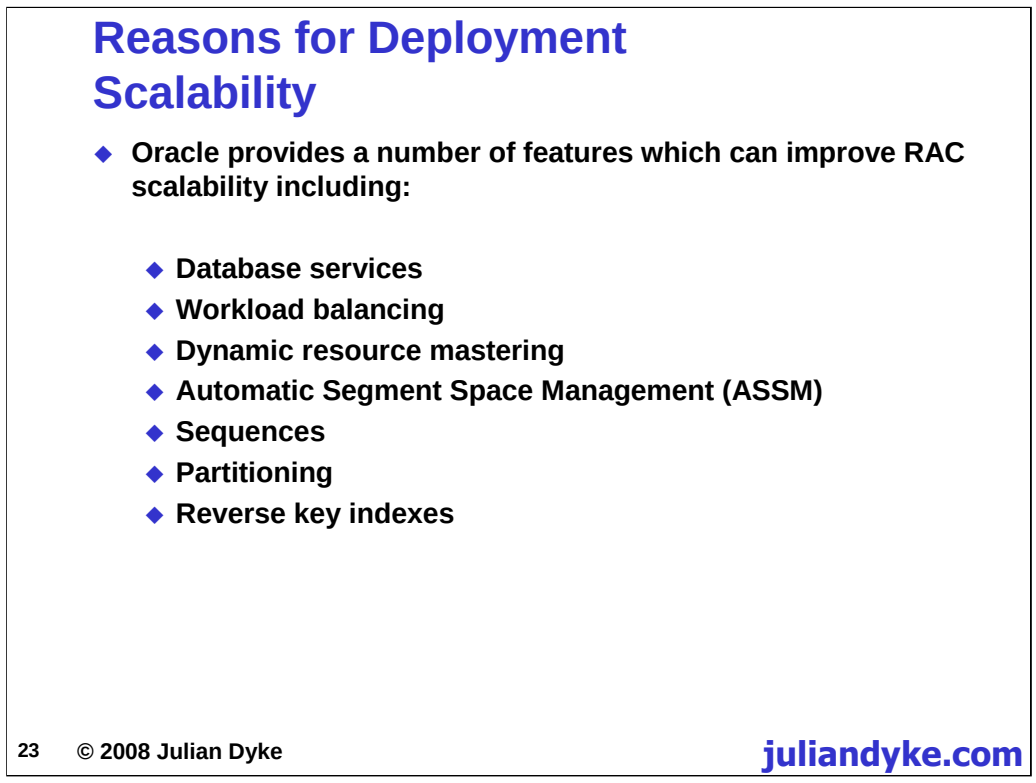
Applications where there is significant parsing or other contention are unlikely to scale.



The simplest way to improve scalability in a RAC database is to reduce the amount of work performed, e.g. not executing unnecessary statements, procedures, reports, batch procedures etc. Further improvements can often be achieved by rescheduling jobs to be performed during quiet periods. When the possibilities for eliminating and rescheduling work have been exhausted, it may be necessary to tune the applications, database objects and database parameters to modify the behaviour of the database. The object is to reduce consumption of CPU, disk and network resources by the application.

The most common form of contention is for physical I/O resources. This is often seen in the form of sequential, scattered and parallel read waits and also as log file sync waits.

Secondary sources of contention in the form of waits for latches and locks. Contention can also occur at cluster level where more than one instance attempts to access the same resource.



Oracle provides a number of features that can improve RAC scalability including:

• Database services

• Workload balancing

• Dynamic resource mastering

• Automatic Segment Space Management (ASSM)

• Sequences

• Partitioning

• Reverse key indexes

Many other features can potentially assist in reducing resource usage including, for example:

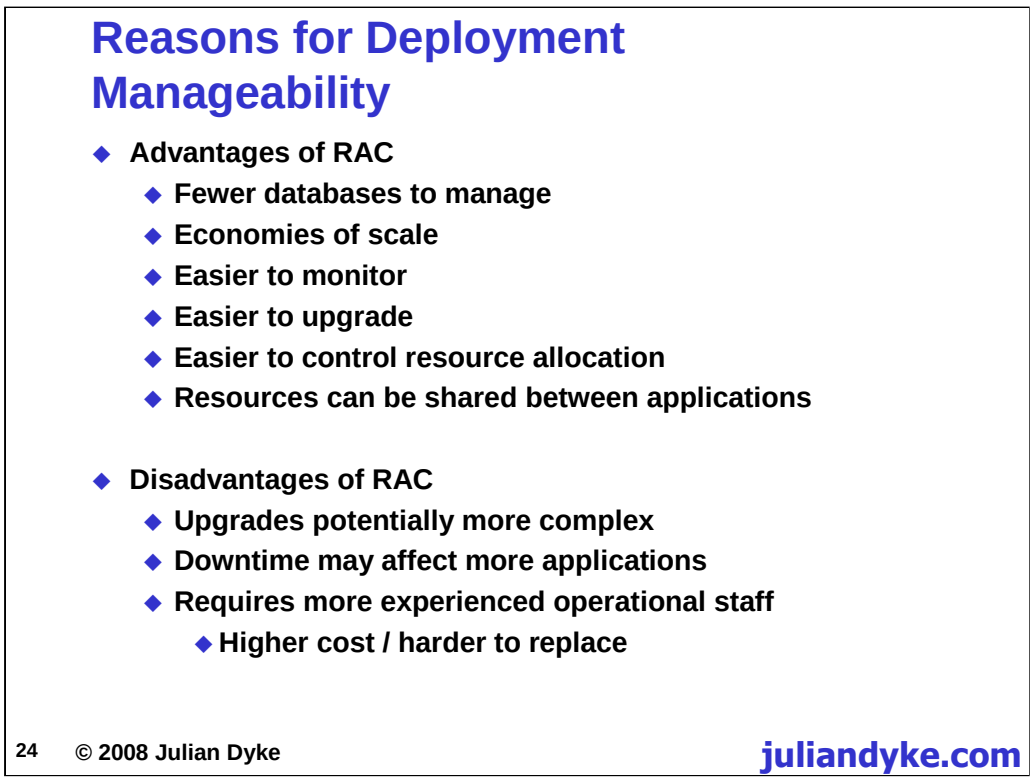
• Index organized tables

• Index compression

• Function-based indexes

• Global temporary tables

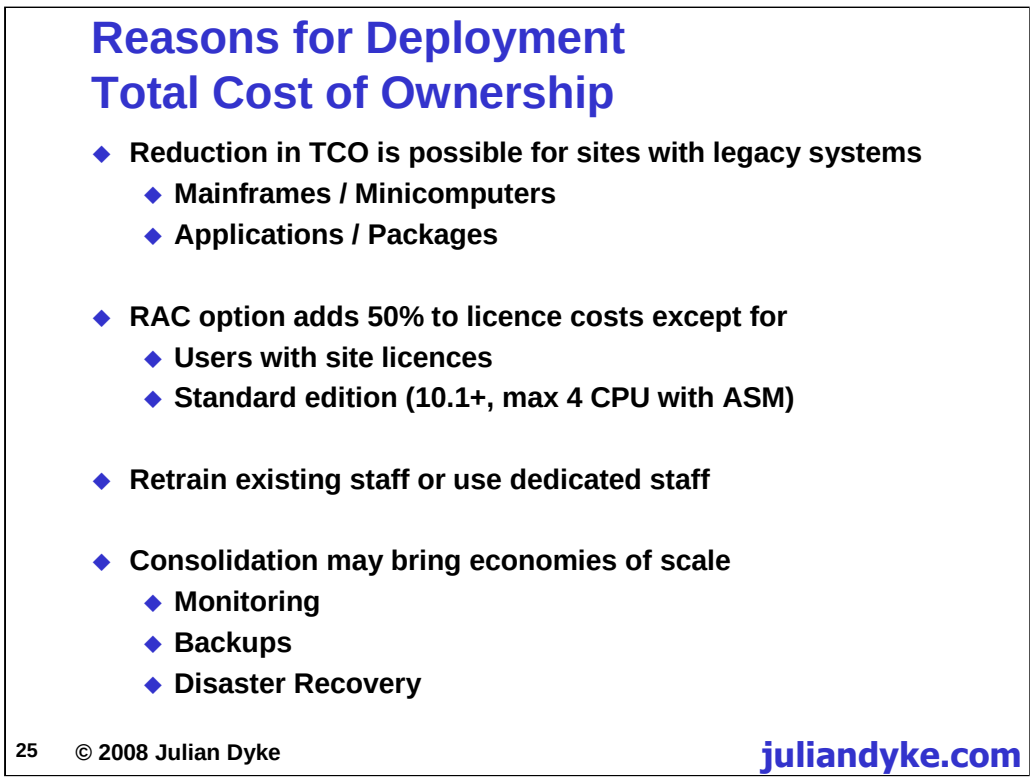
• Materialized views



Improved manageability is often cited as one of the reasons to deploy RAC. For organizations with large numbers of similar databases there may be some justification in this argument. For example many international businesses have successfully consolidated databases running similar applications around the world into a small number of RAC clusters. There are clear economies of scale to be gained here. If there are fewer databases to manage, it is easier to monitor, upgrade and control resource allocation. Resources can be shared between applications.

Grids in which a number of disparate applications are deployed in the same cluster are less popular, but have been successfully deployed by some users. However, the economies of scale to be gained from consolidating many small applications are more difficult to identify.

RAC does also make management more difficult in some respects. Upgrades are potentially more complex with many more dependencies. Any downtime may affect more applications. RAC also requires more experienced operational staff who will come at a higher cost and may be harder to replace.



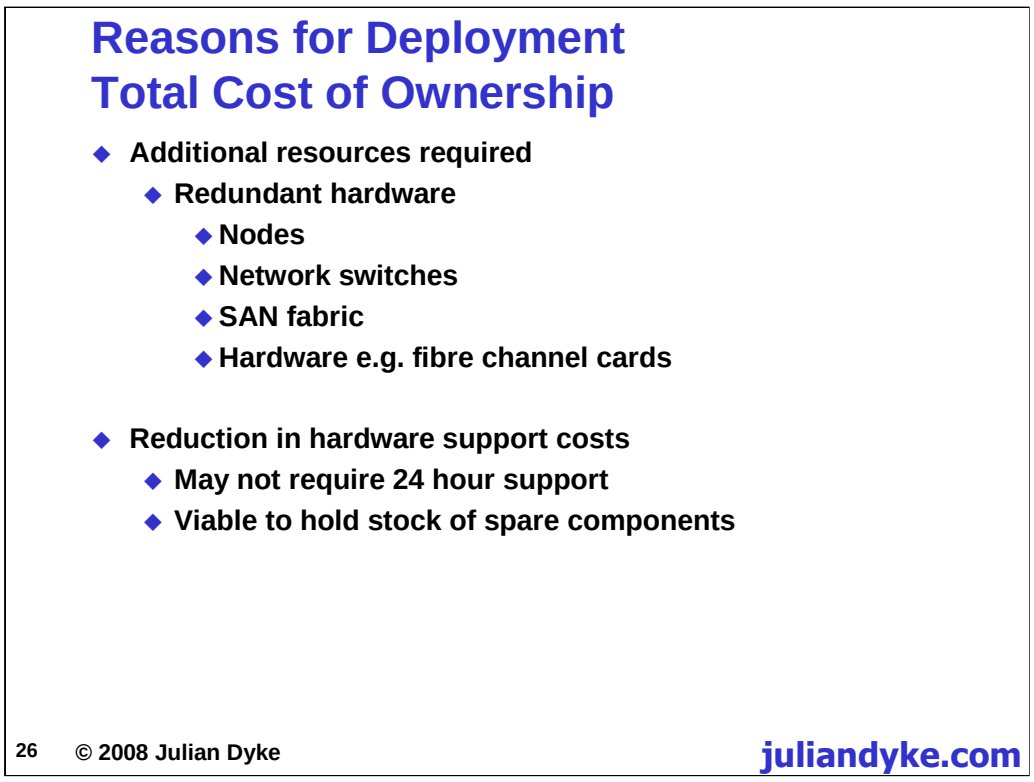
Customers with legacy systems may achieve a reduction in total cost of ownership by replacing old / unsupported mainframes and minicomputers with RAC clusters. Consolidation may result in a reduction in the support costs for legacy applications and packages.

Customers using Enterprise Edition must purchase the RAC option separately. The RAC option costs 50% of the database licence. This applies to licences purchased using both the named user and the processor metrics. The RAC option therefore makes a significant contiribution to the total cost of ownership for many users.

Customers using Standard Edition in Oracle 10.1 and above do not need to licence the RAC option separately although the size of the cluster is restricted to four physical CPUs and ASM must be used for shared storage.

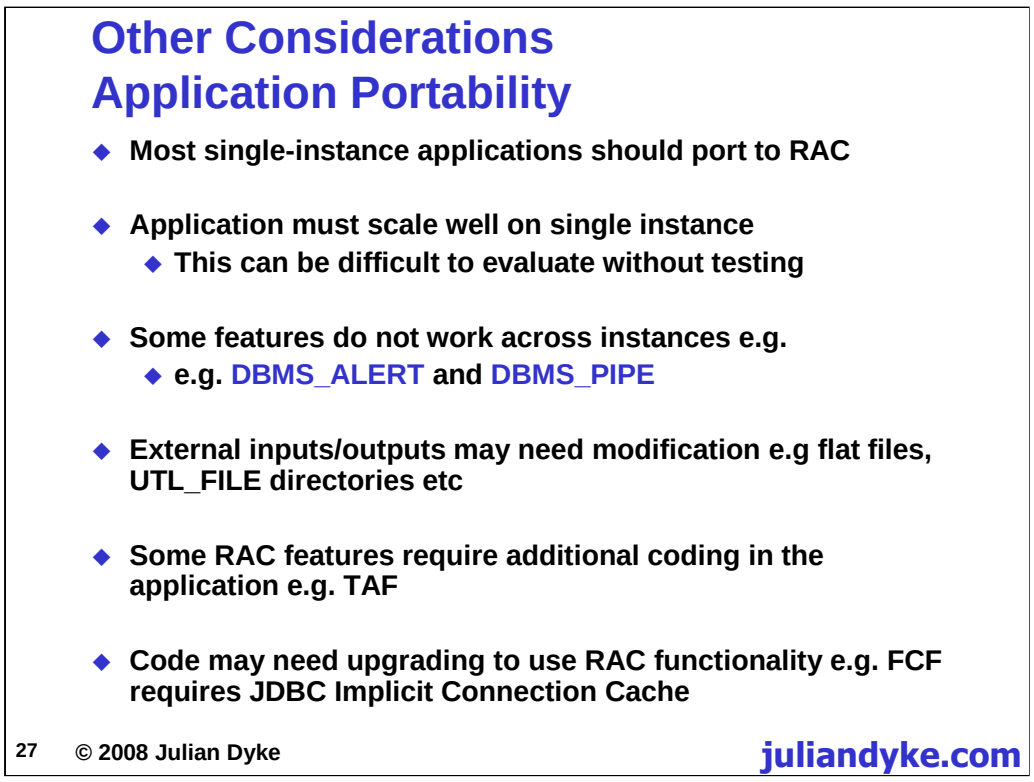
It may be necessary to retrain existing staff or to hire new staff to manage the cluster.

Consolidation may bring economies of scale, in particular for monitoring, backups and disaster recovery. Consolidation may also reduce soft costs such as documentation, developing and testing of operational procedures etc.



Additional resources are required to support a RAC cluster. For example redundant hardware must be purchased including extra nodes, network switches, SAN fabric, fibre channel cards etc.

RAC users may, however, benefit from reductions in hardware support costs. For example it may not be necessary to purchase 24x7 hardware support for servers. As commodity hardware is relatively inexpensive, many sites purchase additional servers which are kept in stock as cold spares. In the event of the failure of one of the nodes in the cluster, the cold spare will be deployed as an emergency replacement while the failed node is repaired. The availability of commodity hardware makes it viable to hold a stock of spare components.



Most single-instance applications should port to RAC without alteration. The application must scale well in a single-instance environment. However, this does not always guarantee that the application will scale well in a RAC environment. It is still quite difficult to evaluate whether an application currently running on a single-instance database will scale well in a RAC environment.

Some features do not work at all in a RAC environment, for example the DBMS_ALERT and DBMS_PIPE packages. If you wish to use these features you must ensure that all sessions accessing them use the same instance. In Oracle 10g and above this can be achieved using database services.

If your application uses external files then it may need modification. Ideally the files will be placed on shared storage and all nodes will have access.

If you wish to use advanced features such as Transparent Application Failover (TAF) or Fast Connection Failover (FCF), you may need to modify your application. Additionally if you wish to use Fast Connection Failover (FCF) you may need to modify your application to use the JDBC Implicit Connection Cache.

Other Considerations
Alternatives to RAC

- Data Guard
  - Physical Standby
    - Introduced in Oracle 7.3
    - Stable, well proven technology
    - Requires redundant hardware
    - Implemented by many sites
    - Can be used with RAC
  - Logical Standby
    - Introduced in Oracle 9.2
    - Still not widely adopted
- Streams
  - Introduced in Oracle 9.2
  - Implemented by increasing number of sites
- Advanced Replication

Before you commit to RAC, make sure that you have considered all of the alternatives, one or more of which might be more appropriate to your business needs.

Physical standby databases were introduced in Oracle 7.3. The redo generated by the primary database is transported and applied to a standby database which is located at another physical site. Standby databases, provide disaster recovery protection. Physical standby technology is well proven and has been implemented by many sites. In Oracle 10.2 and above, it is possible to configure automatic failover so the database can perform an unattended fail over from the primary to the standby site.

In Oracle 9.2 and above, logical standby databases allow a subset of data to be transported and applied to an otherwise open standby database. However, initial problems have resulted in a low uptake for this technology.

Also introduced in Oracle 9.2, Oracle Streams provides a mechanism for transferring data from one database to another.

The functionality of both logical standby databases and Oracle Streams has significantly improved in Oracle 10.2.

Advanced Replication was introduced in Oracle 7.1 and also provides a mechanism for transferring data from one database to another. However, this technology has been largely superseded by Oracle Streams in Oracle 9.2 and above.

Other Considerations
Alternatives to RAC

- Symmetric Multiprocessing (SMP) Systems
  - Single Point of Failure
  - Simplified configuration
  - May eliminate RAC overhead
- Parallel systems
  - For systems with deterministic input. For example
    - Messaging
    - Data Warehouses
- Other Clustering Technologies
  - Active / Passive clusters
  - SAN based solutions e.g. SRDF

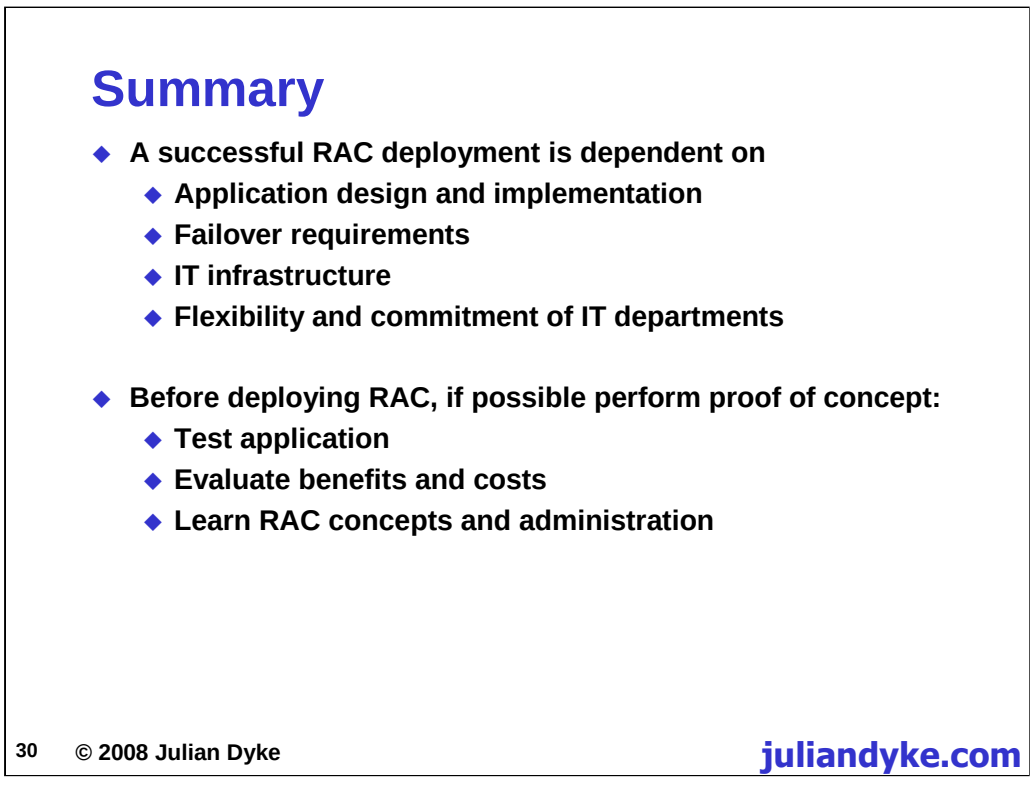
29 juliandyke.com

Other non-Oracle options which can be used instead of RAC include using larger Symmetric Multiprocessing (SMP) systems. If availability is not a key goal, you may achieve better scalability using an SMP box than by deploying the same number of CPUs in a multi-node RAC configuration. This eliminates the RAC overhead.

However, if redo generation is the bottleneck your single-instance database, you may have to implement a RAC solution as this is the only way to configure multiple (parallel) redo streams.

If scalability is not an issue and the feed for the database is completely deterministic e.g. from another system, it may be possible to build two identical single-instance databases and send the same input stream to both. This solution requires slightly less hardware than using RAC as no interconnect is required. For Enterprise Edition customers it is also significantly cheaper as no RAC licence is required. However, availability options are limited. This type of solution has been implemented in messaging systems and is also viable for a data warehouse.

Other clustering technologies may also provide partial solutions such as active / passive clusters, SANs etc.



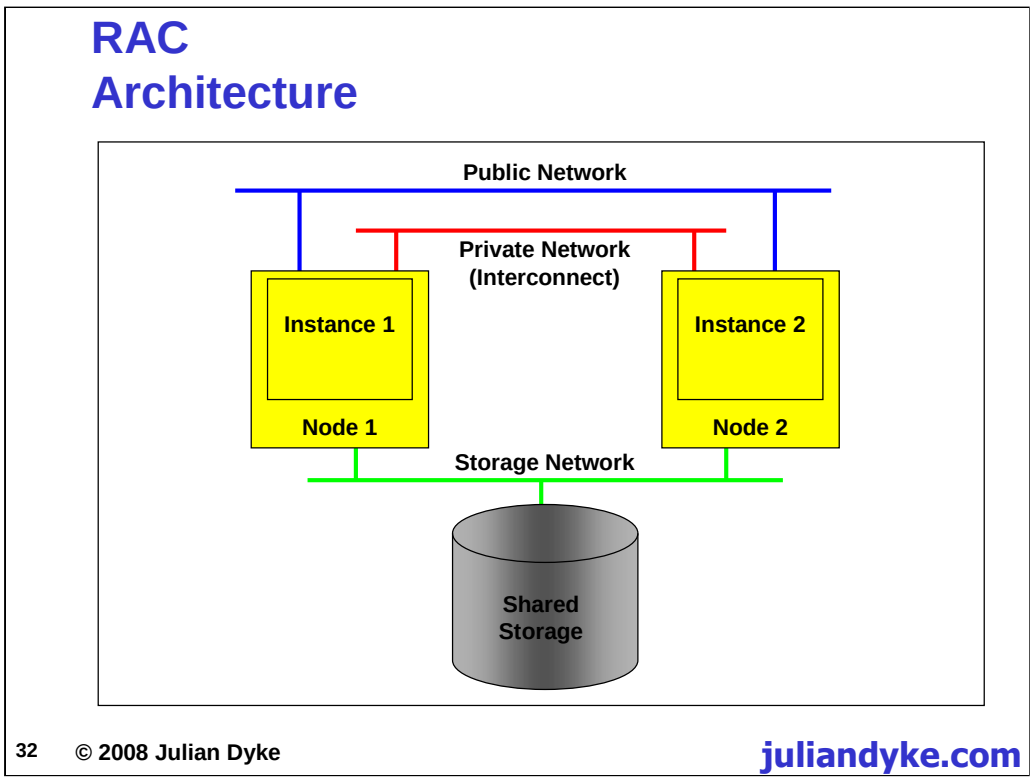
The success of a RAC deployment depends on the design and implementation of the application. The system must meet its failover and performance requirements. For the system to be successful the correct infrastructure must exist. This is largely dependent on the flexibility and commitment of the various teams involved in the project such as application developers, DBAs, networking teams, system administrators and storage administrators.

Ensure you understand the business reasons behind your decision to deploy RAC. Before you deploy RAC, ensure that you have investigated and rejected any alternatives. Prior to deploying RAC you may wish to perform proof of concept testing which involves testing your application and evaluating the benefits and costs of the RAC implementation. Performing proof of concept testing also enables you to learn RAC concepts and administration.

Finally, you might consider buying a good book. I can recommend:

Pro Oracle Database 10g RAC on Linux: Installation, Administration and Performance by Julian Dyke and Steve Shaw, published in August 2006 by Apress ISBN 1-59059-524-6

Concepts



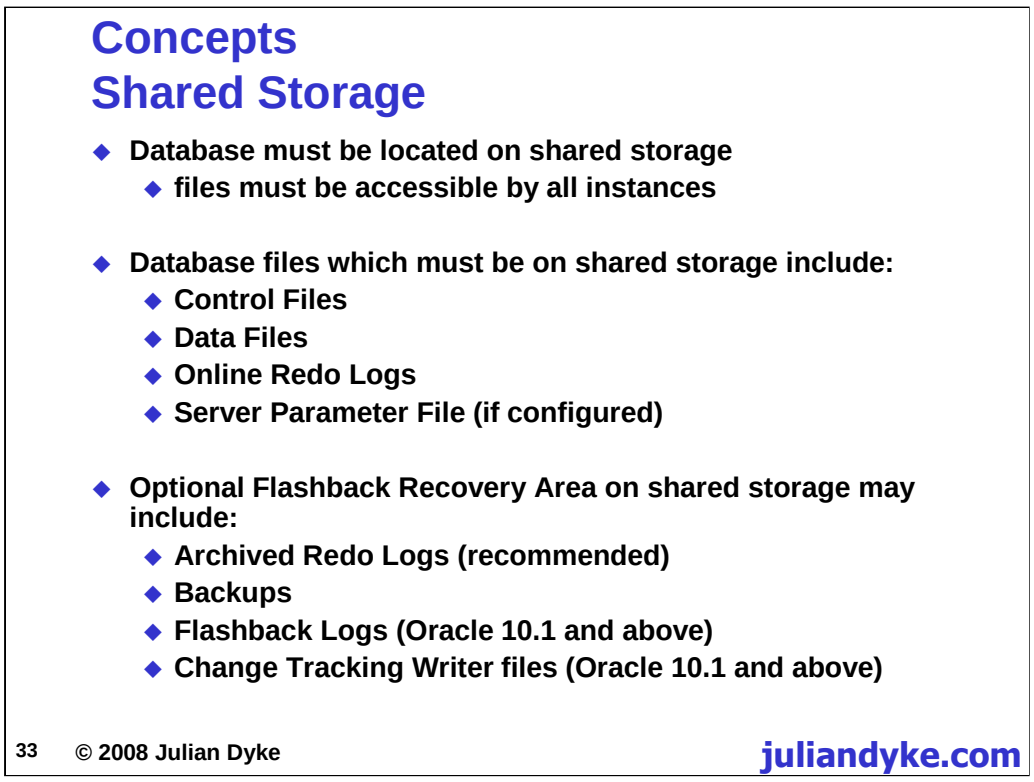
This diagram depicts a two-node cluster. Each node is connected to a minimum of three networks.

The public network is used to communicate with clients outside the cluster. In practise clients rarely connect directly to a RAC cluster. Normally clients communicate with a middle tier application server. All communication with the RAC cluster is routed through the middle tier. This allows connections to be cached and concentrated reducing the demand for resources on the RAC cluster. Consequently the public network is frequently a subnetwork isolated from the rest of the infrastructure.

The private network is used for inter-node communication. It is usually referred to as the interconnect. In addition to the Oracle instances, the interconnect can be used by Oracle Clusterware and OCFS daemons. It is often recommended that the interconnect is bonded (teamed). This means that there are two separate physical connections using redundant hardware. If one connection fails, there is a reasonable chance that the other will survive.

The storage network connects the nodes in the cluster to the shared storage. The storage network may be 1Gb ethernet (NAS or ISCSI) or 2Gb/4Gb fibre (SAN)

Larger sites often implement additional networks for backup/recovery and also for monitoring to avoid impact on the public network. Some sites also implement bonding for the public network.



A RAC database consists of a set of persistent files located on shared storage which is accessible by all instances.

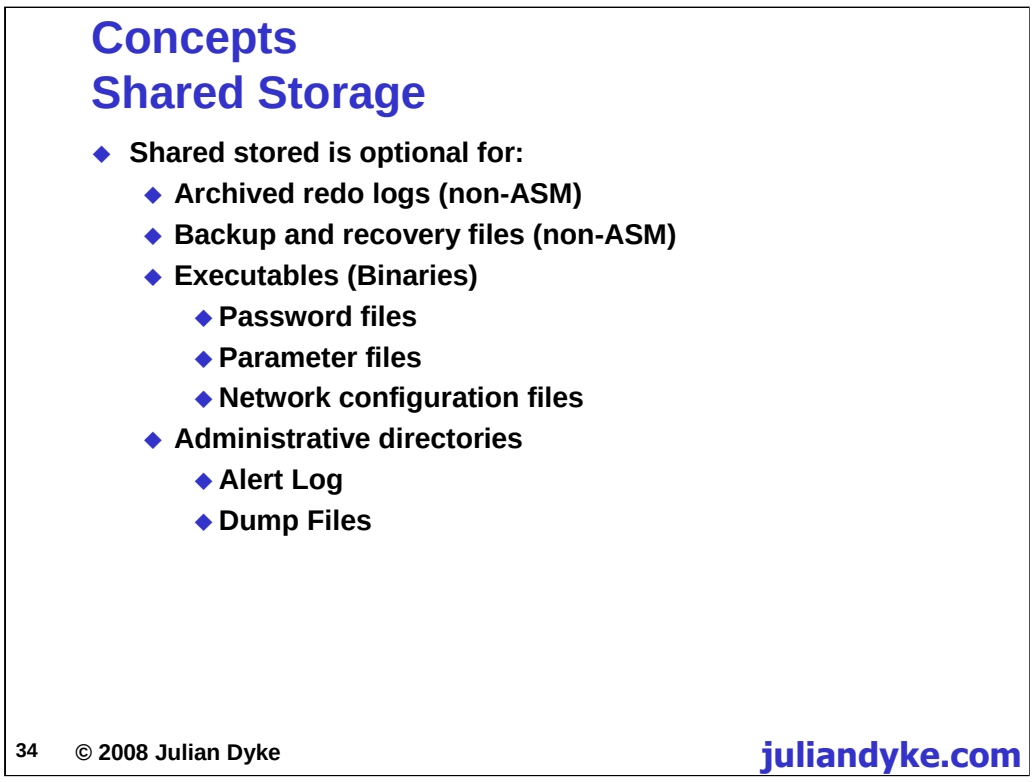
All RAC databases include control files data files and online redo logs.

Parameters may be stored locally in text-based initialization parameter files. Alternatively, they may be stored centrally in a server parameter file (SPFILE) which is located on shared storage. Whilst it is possible to use either type of parameter file in a single-instance environment, in a RAC environment it is more efficient to use an SPFILE as this centralizes parameter management and eliminates the possibility of inconsistencies.

It is preferable to locate archived redo logs on shared storage. Both ASM and OCFS can be configured to store archived redo logs. However, it is not possible to store archived redo logs on raw devices. If shared storage is considered too expensive to store archived redo logs, then it is possible to store them in local file systems in which case NFS should be used to allow them to be accessed by other nodes in the cluster.

If possible backups should also be written to shared storage to facilitate restoration by any node in the cluster.

In Oracle 10.1 and above, a number of other file types have been implemented including flashback logs and change tracking writer files. These are effectively additional database files and should be located on shared storage.



Shared storage is mandatory for all datafiles, control files, online redo logs and, if used, the server parameter file (SPFILE).

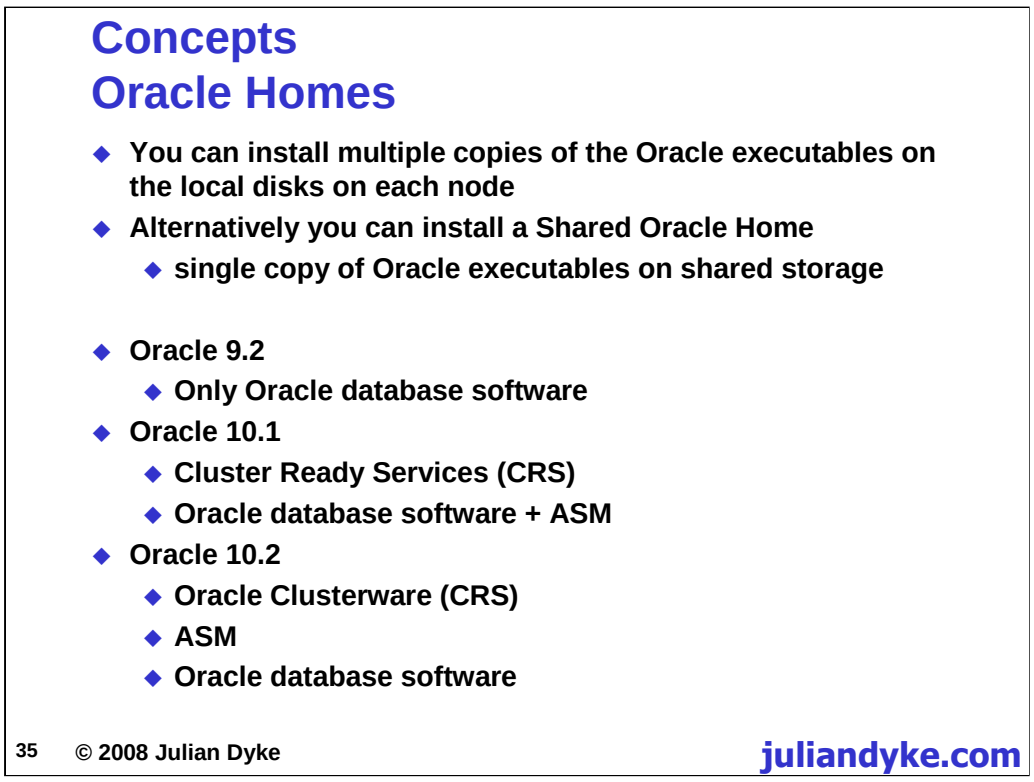
Shared storage is optional for the archived redo logs, executables and administrative directories.

I recommend that you use shared storage for archived redo logs. It is easier to backup and, more importantly, restore the archived redo logs if they are stored in a shared location.

If you have a cluster file system (not ASM) you can also use shared storage for the administrative directories (background and user dump destinations). A single shared location can be defined for each database. This makes finding the alert logs much easier as they will all be in the same directory. All trace files include the instance name in the file name thus preventing instances overwriting trace files from other instances.

The benefits of using shared storage for the executables are less obvious particularly for a two-node cluster. This is because upgrades and patches are more difficult to test and carry a higher level of risk if they are performed on all nodes at the same time. Unless customers have a test system with an identical architecture, I generally recommend that executables are not located in a shared Oracle home.

You may, of course, still choose to use local Oracle home directories, but to locate these on the SAN or NAS. There may be a slight performance benefit as blocks could potentially be stored in the SAN cache. However there are better uses for the SAN cache and most users usually have sufficient unused space available on local storage.



In a RAC environment it is possible to store the Oracle executables in a local Oracle home directory on each server in the cluster. This works well for 2-node clusters, but can complicate management for clusters with more than 2 nodes.

For clusters with a large number of nodes Oracle provides shared Oracle homes. These are located on shared storage. The benefit of a shared Oracle home is that it allows Oracle upgrades and patches to be applied to a number of nodes simultaneously. This is particularly important in environments with more than, say, 4 nodes.

In the past there have been a number of issues with shared Oracle homes which have prevented them being deployed by many users. Whilst most of these issues have been resolved in Oracle 10.2, it is still advisable to use local Oracle homes in a 2-node environment. This allows patches to be applied to one node, whilst the other node continues to service the users. If the patch requires changes to the database then at some point after applying any software changes to the first node, it will be necessary to stop the second node, upgrade the database. The first patched node can then be restarted whilst the changes are applied to the second node.

The amount of risk attached to this operation is largely related to the availability of a QA environment in which any patches can be tested. As almost all patches are architecture-specific it is important that the QA environment has a similar architecture to production.

Concepts
Redo Threads

- In a RAC database there should be one redo log thread per instance
- Redo log threads can be added using the following syntax:

```
ALTER DATABASE ADD LOGFILE THREAD 2
        GROUP 3  SIZE 51200K,
        GROUP 4  SIZE 51200K;

ALTER DATABASE ENABLE PUBLIC THREAD 2;
```

- Each instance writes to a separate redo log thread
- Redo log files must be on shared storage
  - In the event of a node failure all surviving nodes must be able to access log file of failed instance

There are very few differences between a single instance database and a RAC database.

A single instance database can easily be converted into a RAC database of the same Oracle version. It is not necessary to export and import any of the data. In addition it is relatively straightforward to convert a database from 32-bit to 64-bit, in which case it is only necessary to recompile all PL/SQL packages, procedures and functions currently stored in the database. This is a simple task for which Oracle provides a standard script.

A RAC database contains one redo thread per instance. A single instance database will contain one redo thread. It is therefore necessary to create additional redo threads for new instances when the database is migrated to RAC.

Concepts
Undo Tablespaces

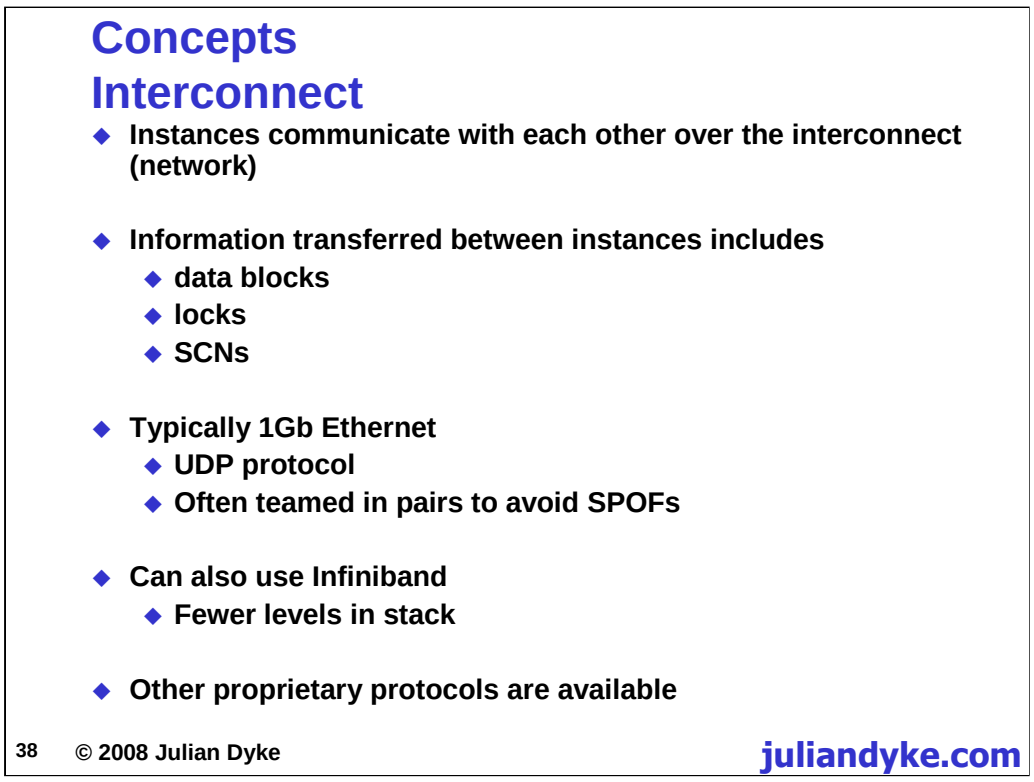
- If you are using Automatic Undo Management each instance requires a separate UNDO tablespace
- Additional undo tablespaces can be added using the following syntax

```
CREATE UNDO TABLESPACE "UNDOTBS2"
DATAFILE  SIZE 25600K  AUTOEXTEND ON
MAXSIZE UNLIMITED  EXTENT MANAGEMENT
LOCAL;
```

37 © 2008 Julian Dyke juliandyke.com

If you are using Automatic Undo Management (which I recommend) then you will also need one undo tablespace per instance.

The only other change required to convert the single-instance to the RAC database is to run $ORACLE_HOME/rdbms/admin/catclust.sql. This script creates a number of additional dynamic performance views (V$, GV$ but not X$) which are only required in a RAC environment. It also creates a handful of objects to ensure backwards compatibility with some legacy views from OPS



The private network is used for inter-node communication. It is usually referred to as the interconnect. In addition to the Oracle instances, the interconnect can be used by Oracle Clusterware and OCFS daemons.

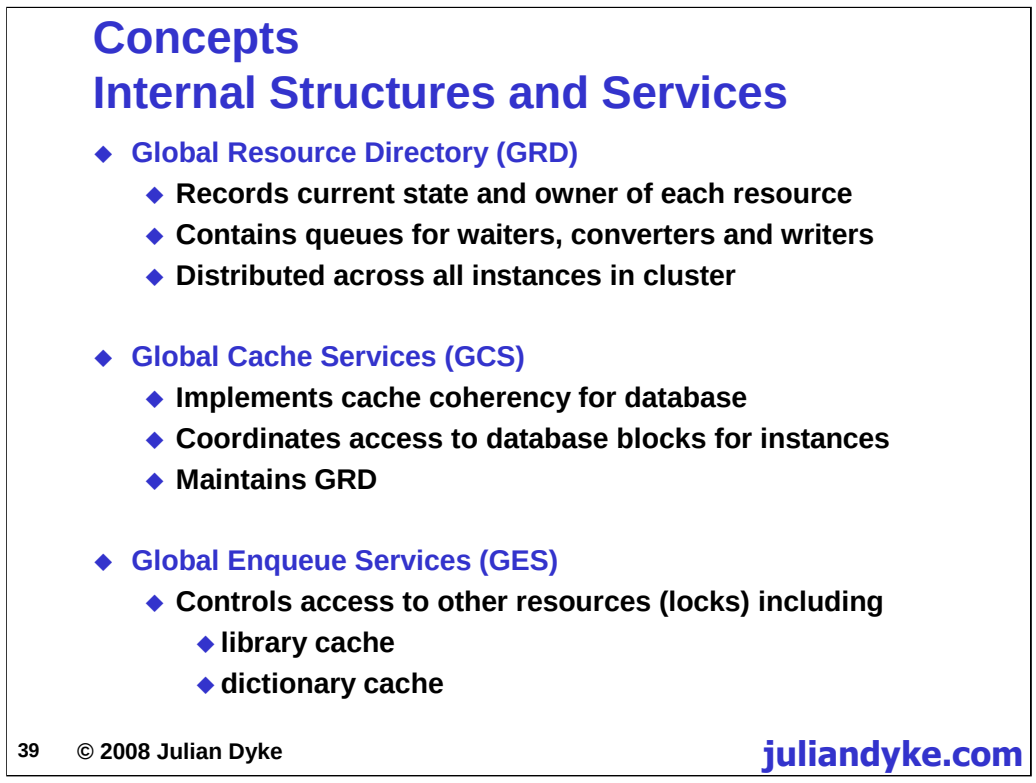
It is often recommended that the interconnect is bonded (teamed). This means that there are two separate physical connections using redundant hardware. If one connection fails, there is a reasonable chance that the other will survive.

Red Hat 4 includes bond drivers. Prior to this release of Red Hat it was often necessary to download drivers from third parties such as Intel.

Infiniband is conceptually very interesting for Oracle users for a number of reasons. It is much faster than existing networks. It also uses a much smaller number of network layers thus reducing the amount of CPU resource required to exchange messages. In the future, Infiniband promises direct memory access which will allow messages to be exchanged between cluster members much more efficiently.

In recent months, Infiniband has been available on Linux. Development of Infiniband has been very slow due to disagreements on standards between the major manufacturers. Infiniband remains relatively expensive and for Oracle RAC clusters, the availability of 10Gb Ethernet threatens to make Infiniband irrelevant for all but a minority of users.

I currently recommend deploying 1Gb Ethernet on clusters and then tuning the application to minimize interconnect traffic. If performance is unsatisfactory then consider upgrading to 10Gb Ethernet first.



The RAC architecture includes three virtual entities which are implemented across all nodes in the cluster. These are:

• Global Resource Directory (GRD) - records the current state and owner of each resource. It contains queues for waiters, converters and writers. The GRD is distributed across all instances in the cluster.

• Global Cache Services (GCS) - implements cache coherency for the database. GCS coordinates access to database blocks for all instances in the cluster, preventing conflicting changes being made to the same block. GCS maintains the Global Resource Directory.

• Global Enqueue Services (GES) - controls access to other resources (locks) including the library cache and the dictionary cache.

Concepts
Background Processes

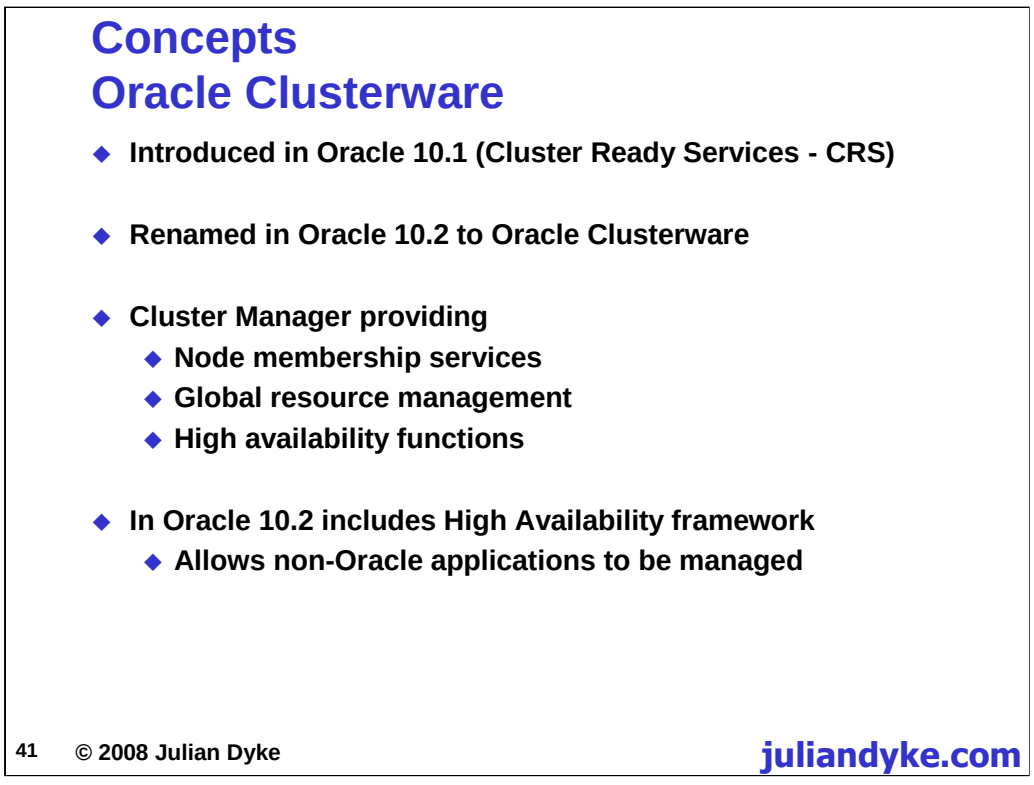
- Each RAC instance has set of standard background processes e.g.
  - PMON
  - SMON
  - LGWR
  - DBWn
  - ARCn
- RAC instances use additional background processes to support GCS and GES including
  - LMON
  - LCK0
  - LMD0
  - LMSn
  - DIAG

40 juliandyke.com

Every RAC instance has a set of standard background processes including PMON, SMON, LGWR, DBWn, ARCn and CKPT.

RAC instances also use additional background processes to support GCS and GES including

- LMON - Lock Monitor
- LCK0 - Instance Enqueue Process
- LMD0 - Global Enqueue Service Daemon
- LMS0 - Global Cache Service Process
- DIAG - Diagnostic Process



Prior to Oracle 10g, Oracle only provided a cluster manager for Linux (oracm) and Windows. Cluster Ready Services (CRS) was introduced in Oracle 10.1 and was the first Oracle cluster manager to be supported on all platforms. CRS was renamed to Oracle Clusterware in Oracle 10.2.

On Linux Oracle Clusterware is configured in /etc/inittab. It is implemented using three daemons:

• CRS - Cluster Ready Service (crsd)

• CSS - Cluster Synchronization Service (ocssd)

• EVM - Event Manager (evmd)

On Unix platforms an additional daemon process called OPROCD is configured. This process is locked into memory to monitor the cluster and provide I/O fencing. It provides similar functionality to the hangcheck timer on Linux. OPROCD performs its check, stops running, and if the wake up is beyond the expected time, then OPROCD reboots the node. An OPROCD failure results in Oracle Clusterware restarting the node.

In Oracle 10.2 and above, Oracle Clusterware includes a High Availability Framework which allows non-Oracle applications to be managed. The framework actually existed in Oracle 10.1, but the interfaces were not placed in the public domain.

Concepts
Oracle Cluster Registry (OCR)

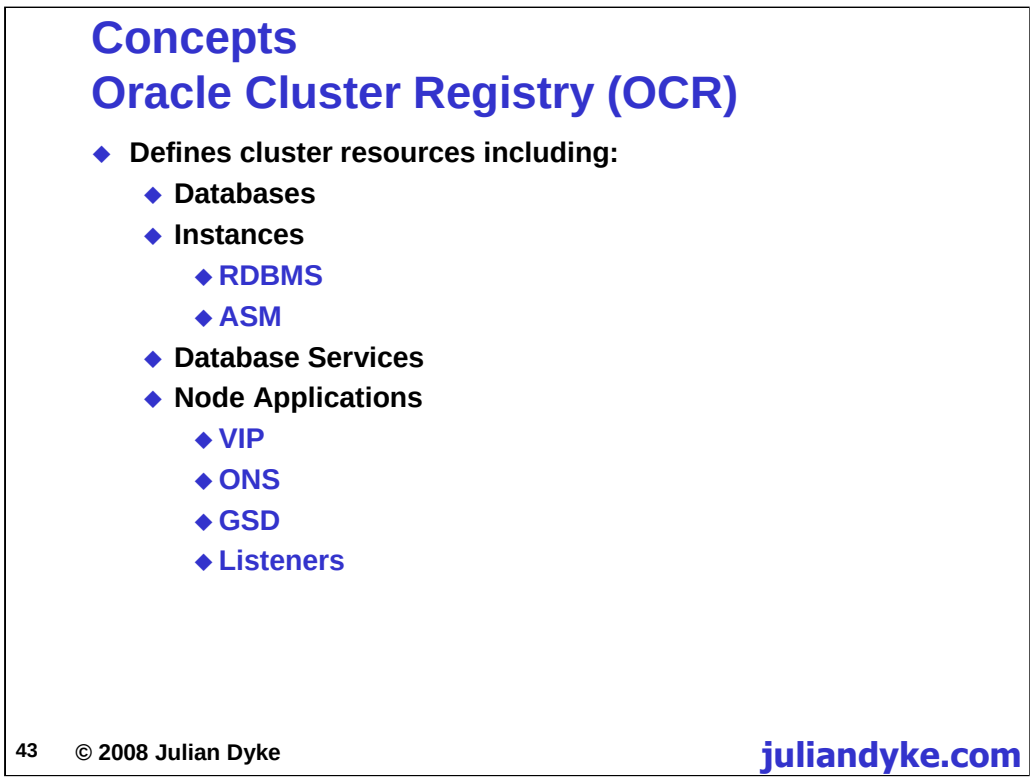
- The Oracle Cluster Registry (OCR) stores the configuration information for Oracle Clusterware / CRS
- Introduced in Oracle 10.1
  - Replaced Server Management (SRVM) disk/file
- Similar to Windows Registry
- Located on shared storage
- In Oracle 10.2 and above the OCR can be mirrored
  - Maximum two copies

The Oracle Cluster Registry (OCR) stores the configuration information for Oracle Clusterware (CRS)

The OCR was introduced in Oracle 10.1. In Oracle 9i it was known as the Server Management (SRVM) disk or file.

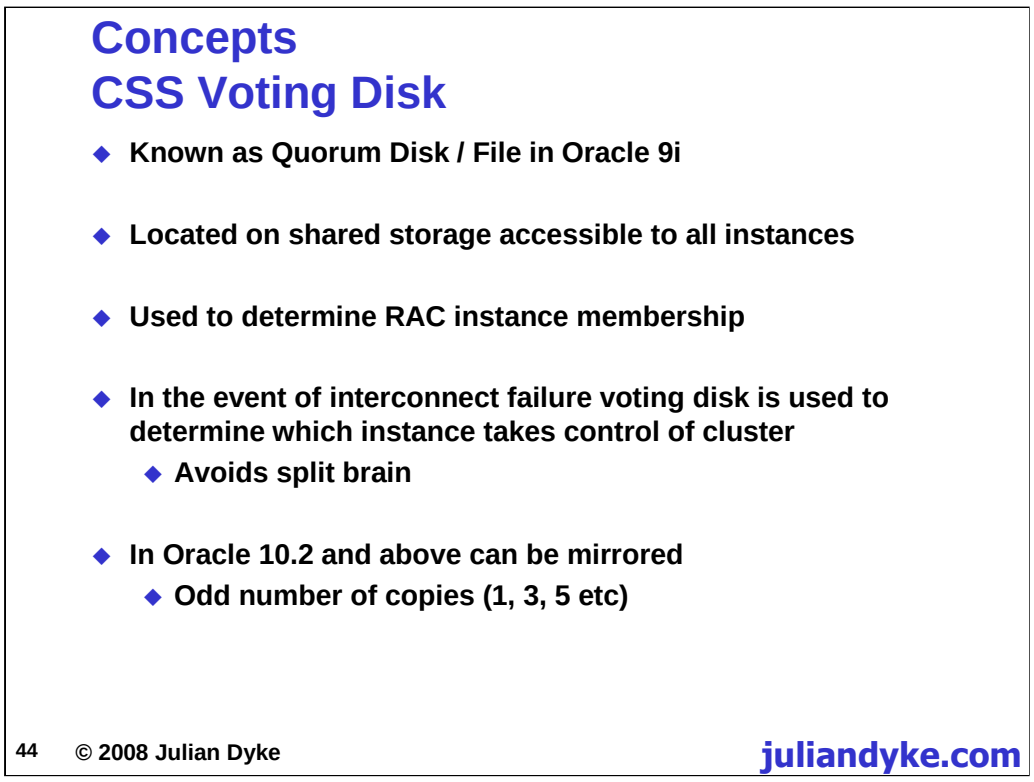
The OCR must be located on shared storage and be visible to all nodes in the cluster. Internally it is similar in structure to the registry used in Microsoft Windows.

In Oracle 10.2 and above, Oracle can maintain a mirrored copy of the OCR, guarding against media failure.



The Oracle Cluster Registry stores the configurations for all cluster resources including:

- Node Applications including
  - VIP
  - ONS
  - GSD
  - Listener
- ASM Instances
- Databases
- RDBMS Instances
- Database Services



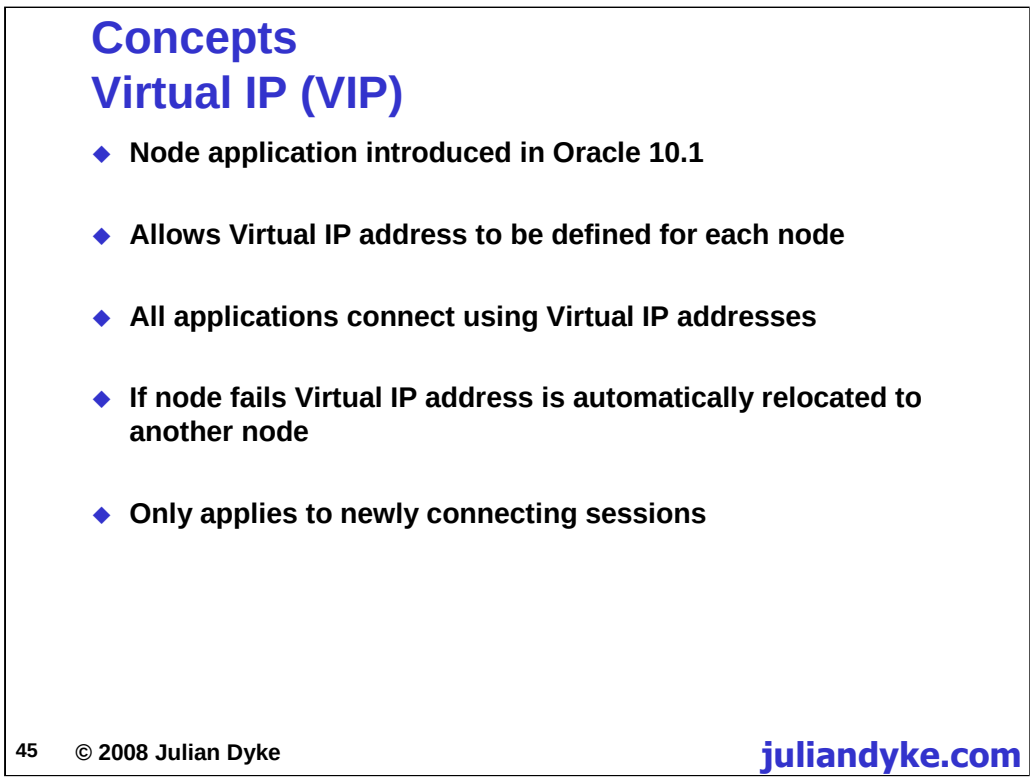
Oracle Clusterware also requires a CSS Voting disk. In Oracle 9i this was known as the Quorum disk.

It is around 20MB and should be located on shared storage accessible to all instances.

In the event of an interconnect failure, the Voting disk is used to determine RAC instance membership. If the interconnect fails all instances attempt to take a lock in the Voting disk. The instances that is successful continues to run; all other instances are shutdown.

In Oracle 10.2 and above, Oracle can maintain mirrored copies of the Voting disk. There should be an odd number of disks e.g. 1, 3, 5 etc. In the event of a cluster failure, the surviving instance must have taken locks on more than 50% of the configured CSS Voting disks.

Voting disks play a particularly important part in extended (stretch) cluster configurations. In a typical two-node extended cluster, three Voting disks will normally be configured; one on each node and a third at a separate, independent location.



Oracle Clusterware runs four default node applications.

• Virtual IP (VIP)

• Oracle Notification Service (ONS)

• Global Services Daemon (GSD)

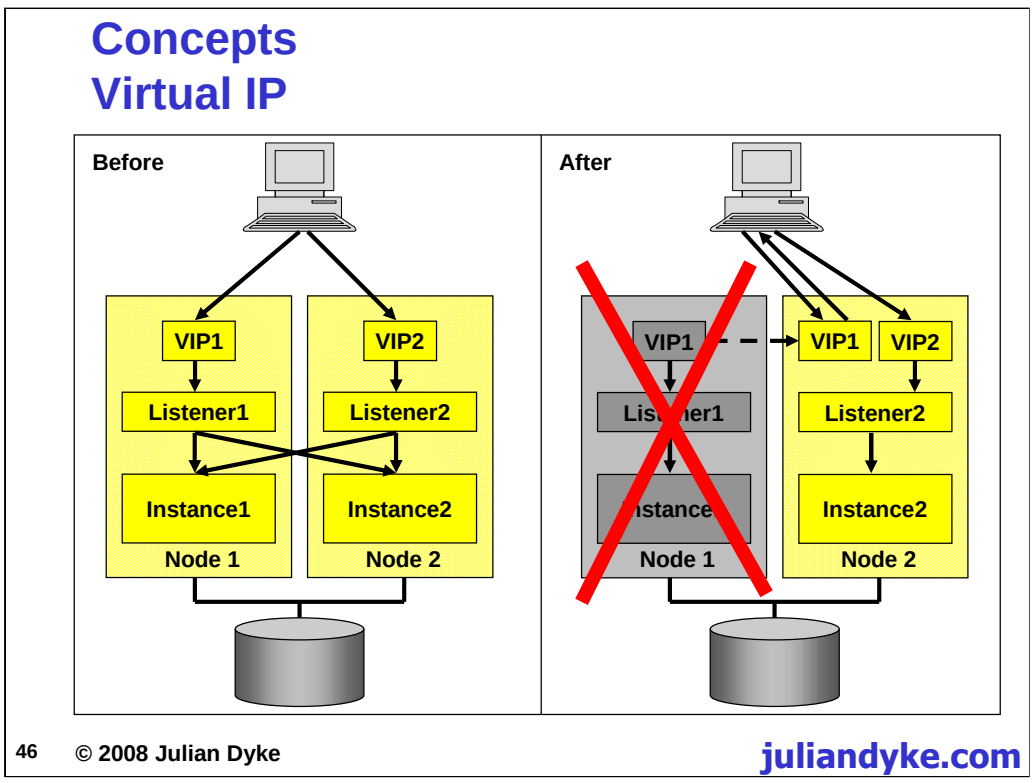
• Listener

Node applications were introduced in Oracle 10.1.

One of the node applications is Virtual IP which allows a virtual IP address to be defined for each node on the same subnet as the public network address. All applications should connect using the Virtual IP address, though you might notice that Oracle does not automatically configure all clients to use Virtual IP addresses by default and it is sometimes necessary to manually amend the configurations.

If a node fails, Oracle Clusterware will fail the Virtual IP address across to another node automatically. A promiscuous ARP message is broadcast across the subnet containing the Virtual IP address and the new MAC address.

The new VIP address only applies to newly connecting sessions.



Oracle VIP addresses were introduced in Oracle 10.1 to solve problems caused by excessively long waits due to TCP/IP timeouts for connecting and reconnecting sessions following a node failure. This slide demonstrates how Oracle Clusterware manages virtual IP addresses.

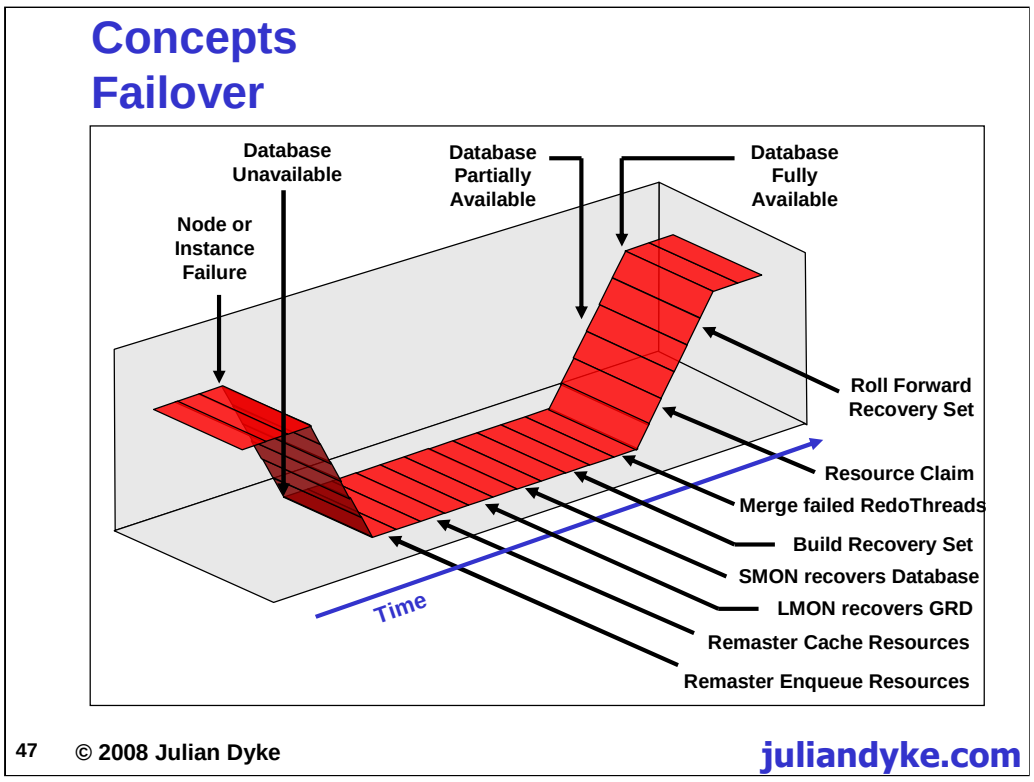
During normal operation, each node will have a unique VIP address. This is different from the public IP address, but must be on the same subnet.

Applications should connect using the VIP address and not the public IP address. In Oracle 10.1 and early versions of 10.2, DBCA incorrectly configured TNSNAMES.ORA to use the public network address rather than the VIP address. This has been corrected in Oracle 10.2.0.3 and above.

The VIP address maps to the MAC address of the adapter. It is possible to use Oracle VIPs with bonded/teamed NICs.

Oracle Clusterware manages VIPs as a node application. When Oracle Clusterware discovers that a node has failed, it will relocate the VIP to one of the remaining nodes. A promiscuous ARP packet is broadcast to inform clients that the MAC address for the VIP has changed.

The relocated VIP does not forward incoming connections to the listener; instead it immediately sends back a failure message to the client. The client can then immediately failover to an alternate address. In the above slide, node 1 has crashed. Oracle Clusterware relocates VIP1 to node2. When the client attempts to connect using VIP1 it receives an error message; when it attempts to connect using VIP2 the connection is forwarded to the listener.



This slide shows the steps performed by the recovery node following a node failure. When a node failure is detected, the database becomes unavailable on all instances in the cluster. Sessions will be unable to continue until all the resources managed by the failed instance have been re-mastered.

The instance detecting the failure manages the recovery process. Recovery consists of a number of steps, some of which can be performed in parallel. These include:

- Remastering enqueue resources (locks)
- Remastering cache resources (blocks)
- Identifying the set of blocks to be recovered
- Reserving the resources required for recovery

At this point the database can be made partially available. Block recovery will continue in the background. Any session attempting to read a block that has not yet been recovered will need to perform that recovery itself as part of the read operation.

The database continues to be partially available until all blocks have been recovered at which point the database is fully available again.

The recovery instance reads the online redo log of the failed instance and applies all changes to the affected data blocks. Any changes made by uncommitted transactions are subsequently rolled back.

Concepts
Transparent Application Failover (TAF)

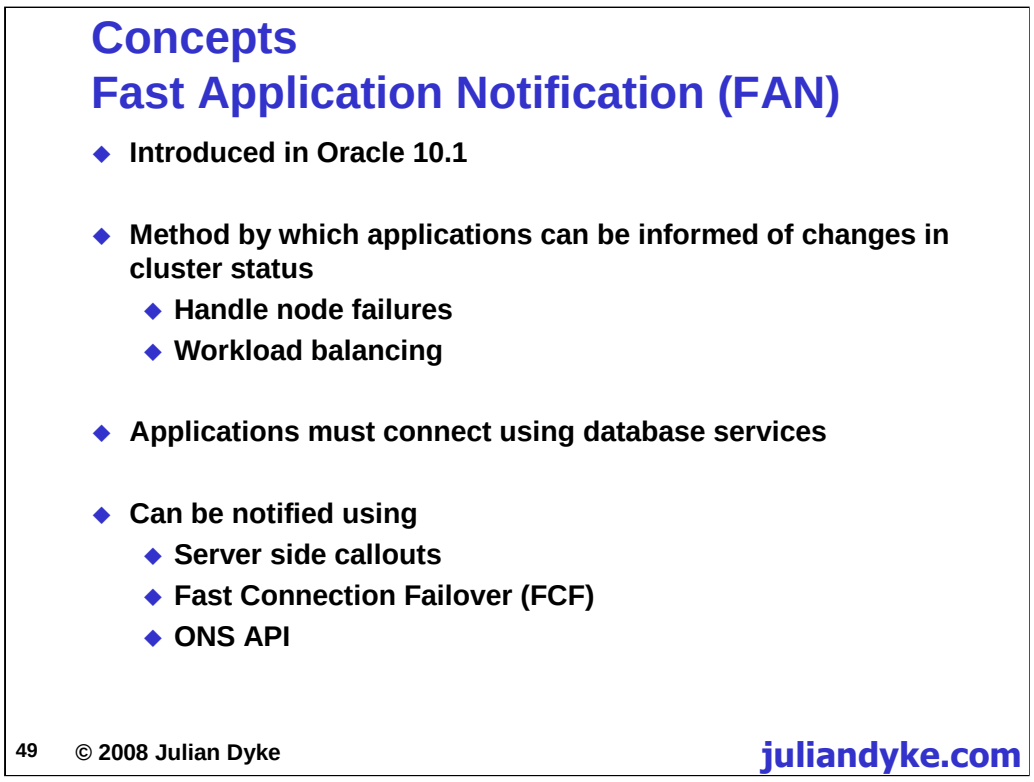
- TAF is Transparent Application Failover
- Sessions connected to a failed instance will be terminated
  - Uncommitted transactions will be rolled back
- Sessions can be reconnected to another instance automatically if using TAF
  - Can optionally re-execute in-progress SELECT statements
    - Statement re-executed with same SCN
    - Fetches resume at point of failure
  - Session state is lost including
    - Session parameters
    - Package variables
    - Class and ADT instantiations

48 juliandyke.com

TAF requires additional coding in client

TAF also requires configuration in TNSNAMES.ORA. For example:

```
RAC_FAILOVER =
 (DESCRIPTION =
   (ADDRESS_LIST =
     (FAILOVER = ON)
     (ADDRESS = (PROTOCOL = TCP)(HOST = node1)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = node2)(PORT = 1521))
   )
   (CONNECT_DATA =
     (SERVICE_NAME = RAC)
     (SERVER = DEDICATED)
     (FAILOVER_MODE=
       (TYPE=SELECT)
       (METHOD=BASIC)
       (RETRIES=30)
       (DELAY=5)
     )
   )
 )
```



Fast Application Notification (FAN) was introduced in Oracle 10.1. It is a method by which applications can be informed of changes in the status of the cluster allowing applications to handle node failures and also to perform workload balancing.

In order to use FAN, applications must connect using database services. Applications can be notified using server-side callouts, Fast Connection Failover (FCF) or programs developed using the ONS API.

The Oracle Notification Service (ONS) was introduced in Oracle 10.1 an runs as a node application under Oracle Clusterware on each node.

ONS allows out-of-band messages to be sent to nodes in the cluster, middle-tier application servers and clients connecting to the database.

ONS is the underlying mechanism supporting Fast Application Notification (FAN).

Concepts
Database Services

- Database Services are logical groups of sessions
- In Oracle 10.1 and above, each service can have
  - One or more preferred (primary) instances
    - used by default
  - One or more available (secondary) instances
    - used if preferred instances have failed
- Database Services can be used to support:
  - Workload balancing
  - Transparent Application Failover (TAF)
  - Node affinity
  - Statistics collection
  - Trace collection

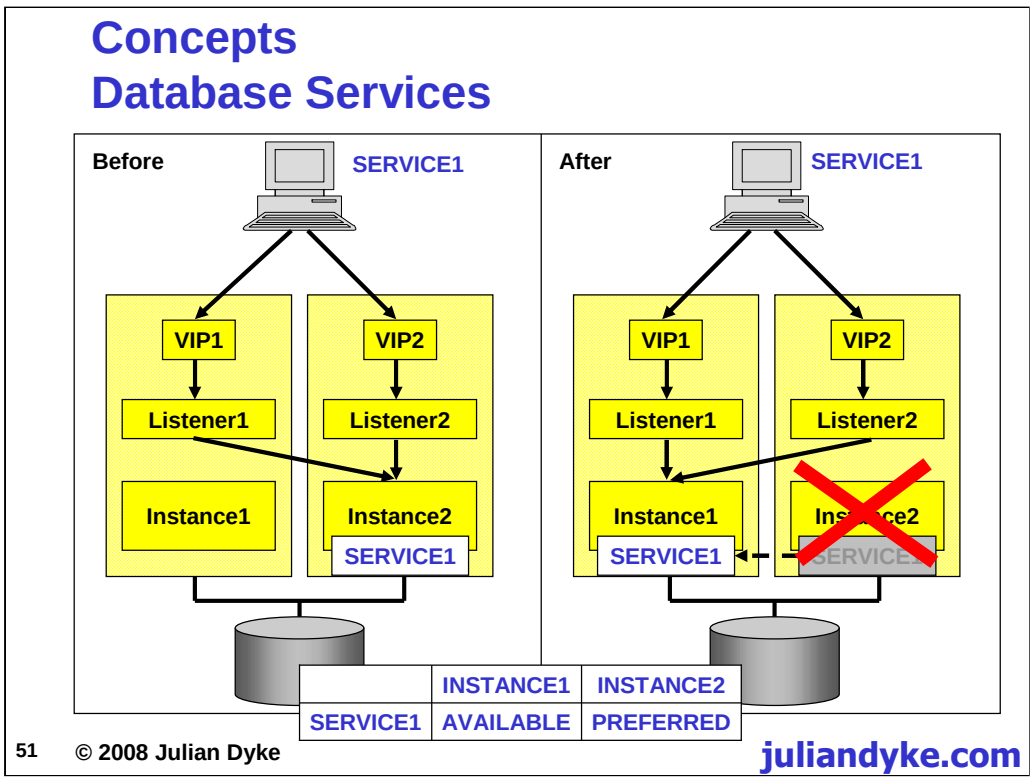
If you have an asymmetric workload, then database services are probably the most powerful tool available to manage that workload. Database services allow you to divide your workload into logical components (or groups of sessions) and then to control which instance(s) run these services. Each service can have one or more preferred instances. By default connections to these services will be directed by the listener process to a preferred instance. If the preferred instances are not available then new connections will be directed one of the available instances.

If is also possible to specify that a service should never be used by an specific instance.

Database services can be used with Resource Manager to control resource usage e.g. CPU usage and Parallel Execution.

Database services can be used for monitoring using the V$SERVICE_STATS dynamic performance view which is enabled by default.

More granular diagnostics can be obtained for individual database services and also for the modules and actions performed within them using the DBMS_MONITOR package. This package can also enable statistic collection at both module and action level.

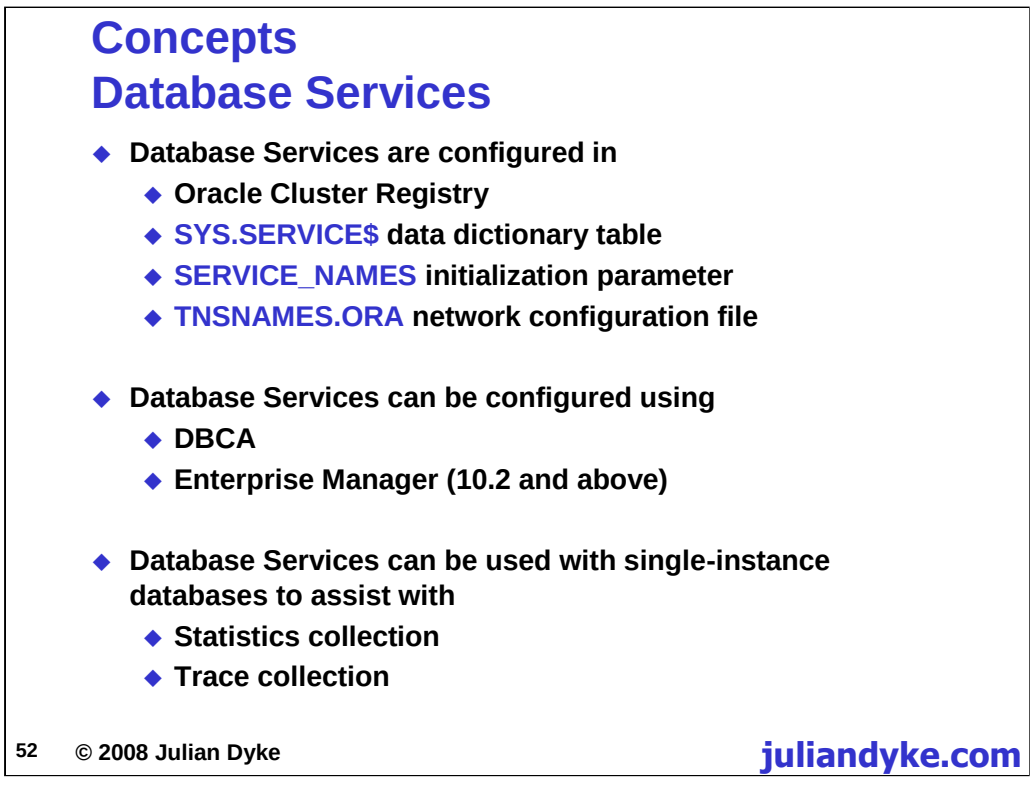


This slide shows how services can be used to specify preferred and available instances. In the above example SERVICE1 has one preferred instance (INSTANCE2) and one available instance (INSTANCE1).

During normal operation, all connections to the service will be sent to INSTANCE2. In this example, the client is performing client-side load balancing so connections can be sent to the listener process on either node. The listener is aware that SERVICE1 is currently running on INSTANCE2 and therefore forwards all incoming SERVICE1 connections to that instance.

If INSTANCE2 is shutdown or fails then Oracle Clusterware will automatically relocate SERVICE1 to the available instance, in this case INSTANCE1. The listener processes on both nodes will now send all incoming SERVICE1 connections to INSTANCE1.

Note that if INSTANCE2 is subsequently restarted, SERVICE1 will not be automatically relocated back to INSTANCE1. SERVICE1 will continue to run on INSTANCE1 until it is manually relocated back to INSTANCE2 using Enterprise Manager or SRVCTL.



In Oracle 10.1 DBCA can be used to configure database services.

In Oracle 10.2 and above, both DBCA and Enterprise Manager can be used to configure services.

In theory database services can also be configured using:

• SRVCTL (Oracle Cluster Registry only)

• SQL*Plus (Data Dictionary and Parameter)

• Text editor (Network Configuration)

However, manual configuration not recommended.

Concepts
Connection Balancing

- Balancing of workload across available instances
- Can have
  - Client-side connection balancing
  - Server-side connection balancing

- Client-side connection balancing
  - Workload distributed randomly across nodes

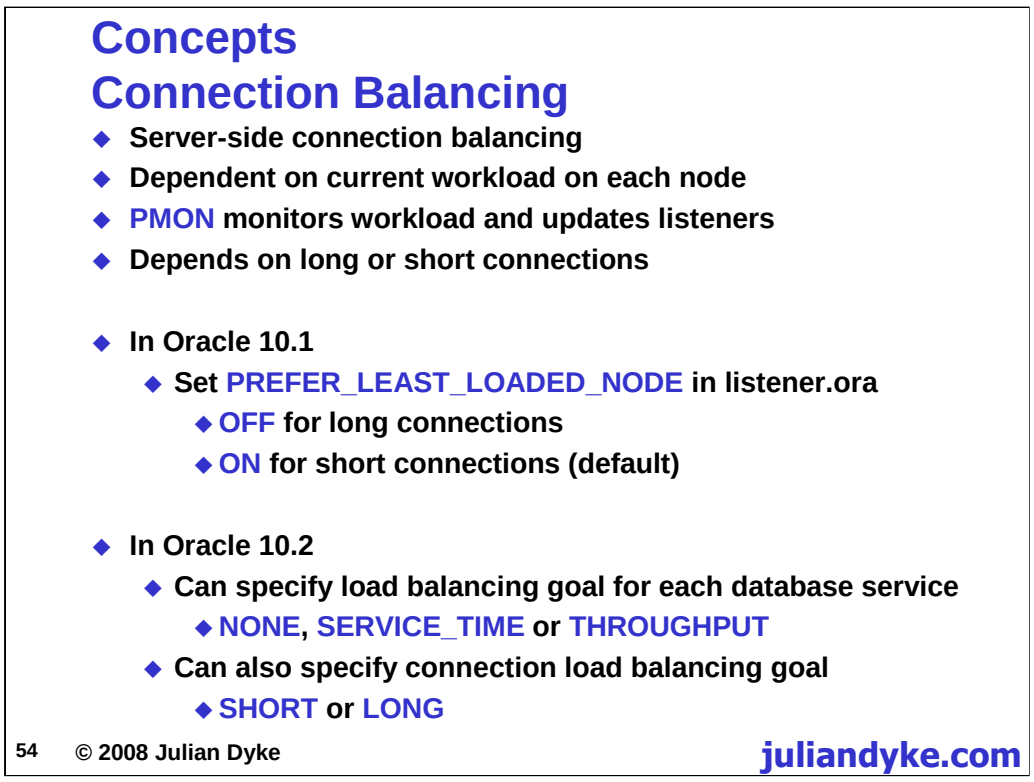
```
RAC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = london1-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = london2-vip)(PORT = 1521))
    (LOAD_BALANCE = ON)
    (FAILOVER = ON)
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (FAILOVER_MODE = (TYPE = SELECT)(METHOD = BASIC))
    )
  )
```

53 © 2008 Julian Dyke juliandyke.com

In order to distribute the workload across all nodes in the cluster you can use workload balancing. This involves distributing sessions across the instances in the cluster. In RAC this is achieved at connection time. In other words, once a session is connected to a specific instance, it will not migrate to another instance during its lifetime.

RAC provides two forms of connection balancing: client-side connection balancing and server-side connection balancing. You can use either or both.

Client-side connection balancing allows the client to select a listener process on one of the cluster nodes. The listener is selected at random from the list of nodes. Client-side connection balancing should really be called listener load balancing because it provides a method for balancing the workload across all available listener processes.



Server-side connection balancing is performed by the listener processes and reflects the current workload on each node. The PMON background process monitors the current workload and if there is a significant change, sends a message to all listener processes in the cluster notifying them of the current workload for the node.

Connections can be short or long. Short connections might be from middle tier applications that connect, execute a SQL statement and then disconnect immediately. Long connections include batch processes and those using connection caches.

The action taken by the listener processes depends on configuration. Prior to Oracle 10.2, there was no formal configuration. However it was possible to set the PREFER_LEAST_LOADED_NODE parameter to OFF for long connections or ON (the default) for short connections.

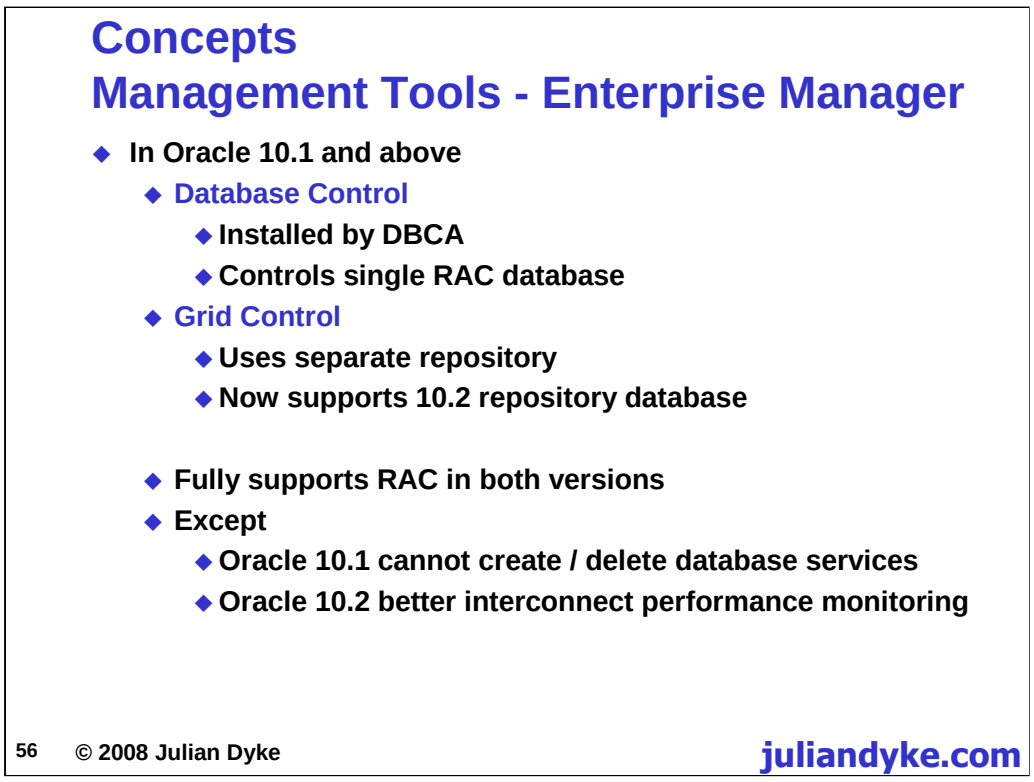
In Oracle 10.2 and above, the configuration has been formalised. For each database service you can specify a load balancing goal which can be NONE, SERVICE_TIME or THROUGHPUT. For each database service you can also specify a connection load balancing goal which can be SHORT or LONG.

Concepts
Management Tools

- RAC-aware management tools include
  - Enterprise Manager
    - Database Control
    - Grid Control
  - SRVCTL
  - CRSCTL
  - SQL*Plus
  - DBCA
  - VIPCA
  - NETCA
  - CLUVFY

RAC aware management tools include:

- Enterprise Manager
  - Database Control
  - Grid Control
- SRVCTL
- CRSCTL
- SQL*Plus
- DBCA
- VIPCA
- NETCA
- CLUVFY



Enterprise Manager has been completely rewritten to run as a three tier browser-based application in Oracle 10.1 and above.

It is available in two versions:

• Database Control: This can be installed by the Database Configuration Assistant when a database is created. Database Control can only be used to manage a single RAC database. The repository is located in the target database. A management agent is installed on each node which collects information about the database and transmits this to the database control.

• Grid Control: This consists of three components:

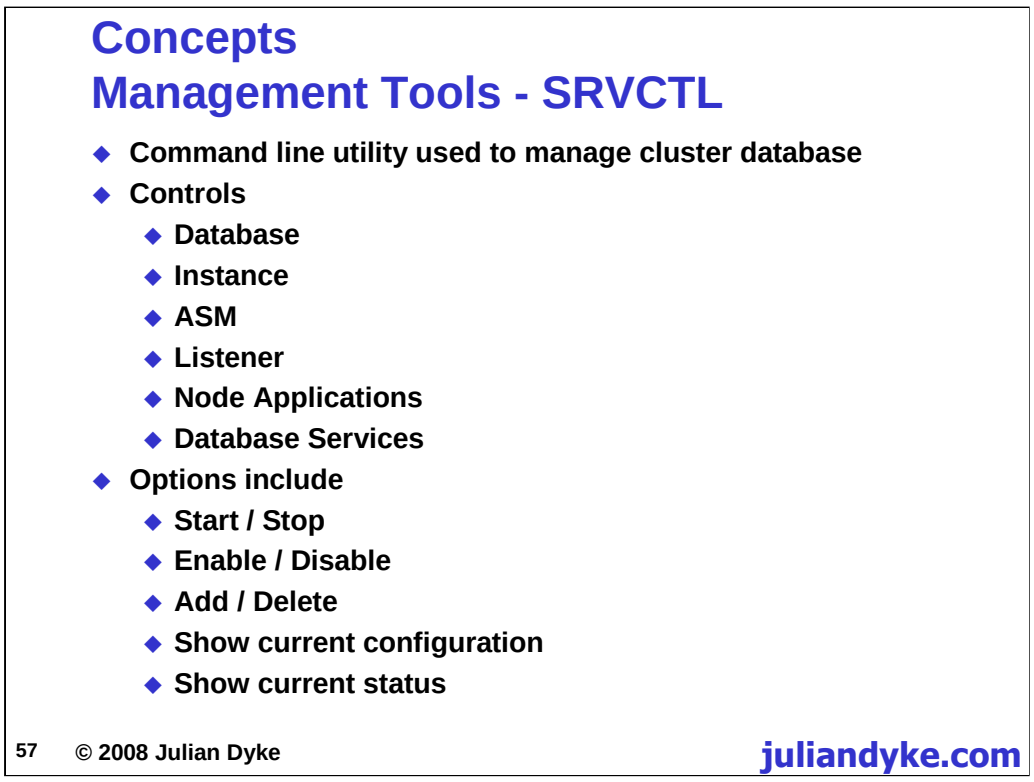
  • Management Agent - this is a native process running on each node which can monitor resources such as the database, application servers etc.

  • Oracle Management Service (OMS) - this is a J2EE application that processes the information collected by the agent, renders it in the Enterprise Manager Grid Control GUI and stores it in the repository.

  • Management Repository - this is an Oracle database that stores Enterprise Manager data. The repository can be a RAC database.

In Oracle 10.2, Enterprise Manager fully supports RAC in both Database Control and Grid Control. In Oracle 10.1, you cannot use Enterprise Manager to create / delete database services. However, you can enable / disable services and change preferred / available instance assignments.

In Oracle 10.2, the interconnect performance monitoring has been significantly improved in the Enterprise Manager pages.



SRVCTL is a command line utility which is used to manage the cluster database. SRVCTL can control most of the resources configured in the Oracle Cluster Registry (OCR) including:

• Databases

• Instances

• ASM Instances

• Listeners

• Node Applications

• Database Services

SRVCTL updates the configuration stored in the Oracle Cluster Registry (OCR). Options vary for each type of object, but include:

• Start / Stop

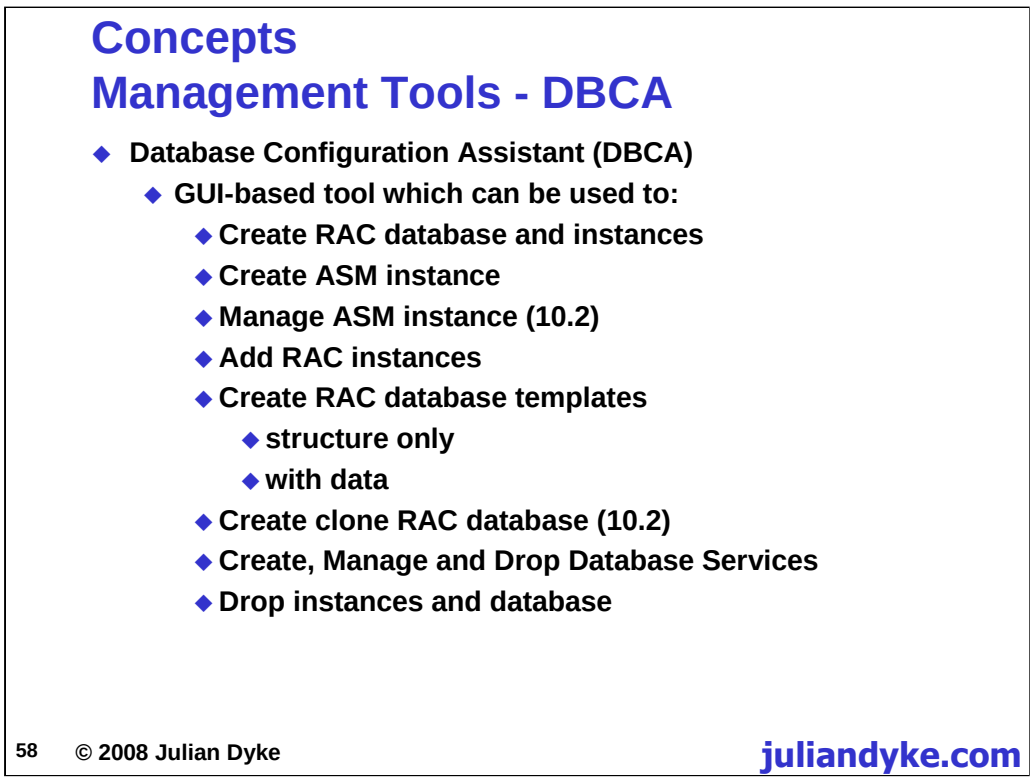
• Enable / Disable (not node applicaitons)

• Add / Delete

• Show current configuration

• Show current status

• Relocate (database services only)



The Database Configuration Assistant (DBCA) is a GUI-based tool which can be used to

• Create RAC databases and instances

• Create ASM instances

• Manage ASM instances (in Oracle 10.2 and above)

• Add RAC instances

• Create RAC database templates - structure only or including data

• Clone RAC databases (10.2)

• Create, manage and drop Database Services

• Drop instances and databases

DBCA is fully RAC aware and can update all nodes simultaneously. For example during database creation, the DBCA will create the administrative directories, parameter files and password files on each node in the cluster if required.

Concepts
Management Tools - CLUVFY

- Introduced in Oracle 10.2
- Supplied with Oracle Clusterware
  - Can be downloaded from OTN (Linux and Windows)
- Written in Java - requires JRE (supplied)
- Also works with 10.1 (specify -10gR1 option)
- Checks cluster configuration
  - stages - verifies all steps for specified stage have been completed
  - components - verifies specified component has been correctly installed

For example, to check the configuration before installing Oracle Clusterware on nodes london1 and london2 use:

```
sh runcluvfy.sh stage -pre crsinst -n node1,node2
```

This command checks

• node reachability (ping)

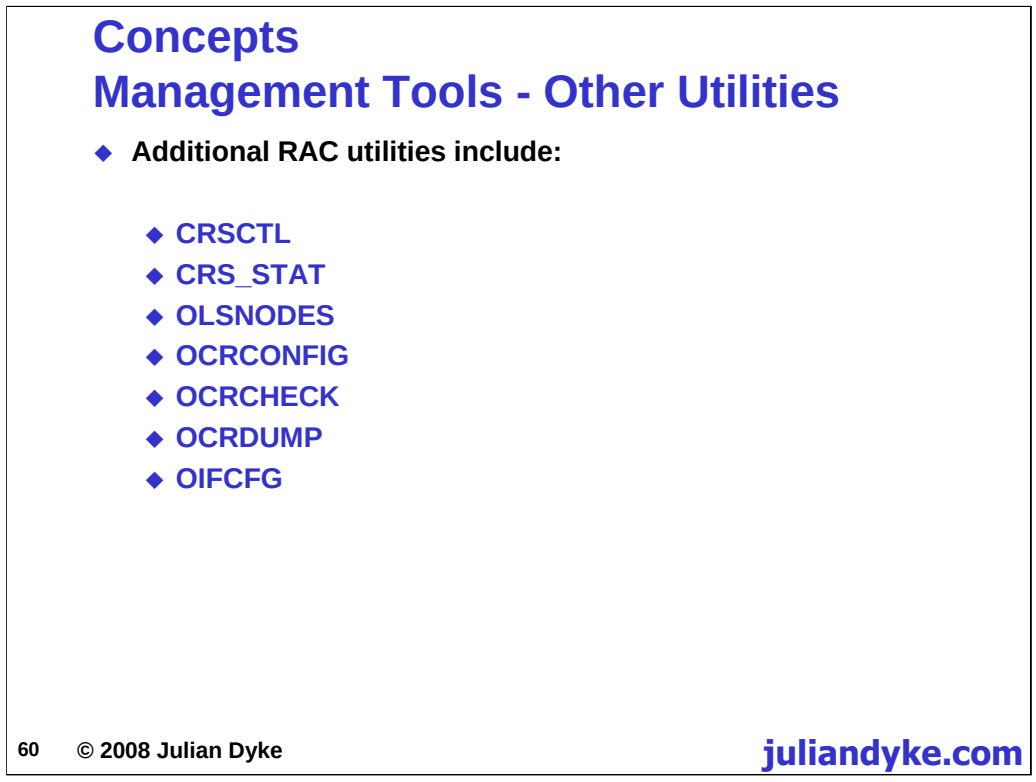
• user equivalence

• administrative privileges

• node connectivity (ssh)

• shared storage accessibility

If any of the checks fail then append the -verbose option to the command to display more information.



A number of other utilities supports RAC including (Oracle 10.2):

- CRSCTL
- CRS_STAT
- CRS_START
- CRS_STOP
- CRS_REGISTER
- CRS_UNREGISTER
- OCRCONFIG
- OCRCHECK
- OCRDUMP
- OLSNODES

Additional RAC diagnostics can be obtained using:

- Numeric Events
- Symbolic Events
- the ORADEBUG utility
    - DUMP options
    - LKDEBUG option

Discovery
Cluster Name

- To check the name of the cluster use **OCRDUMP**
- **For example:**

```
[root@server3 tmp]# ocrdump -stdout -keyname SYSTEM.css.clustername
03/30/2008 15:24:23
/u01/app/11.1.0/crs/bin/ocrdump.bin -stdout -keyname SYSTEM.css.clustername

[SYSTEM.css.clustername]
ORATEXT : cluster1
SECURITY : {USER_PERMISSION : PROCR_ALL_ACCESS, GROUP_PERMISSION : PROCR_READ,
OTHER_PERMISSION : PROCR_READ, USER_NAME : root, GROUP_NAME : root}
```

- **In this example the cluster name is cluster1**

Discovery
Oracle Cluster Registry

- On Linux the location of the OCR is specified in /etc/oracle/ocr.loc
- For example:

```
ocrconfig_loc=/dev/ocr1
local_only=false
```

- In Oracle 10.2 and above, the OCR can be mirrored
- For example

```
ocrconfig_loc=/dev/ocr1
ocrmirrorconfig_loc=/dev/ocr2
local_only=false
```

- The OCRCHECK utility also reports the locations of the OCR and, if configured, the OCR mirror

Discovery
Voting Disks

- To discover the current location of the voting disks use the CRSCTL utility:

```
[root@server3]# crsctl query css votedisk
 0.     0    /dev/raw/raw1
located 1 votedisk(s).
```

- If multiple voting disks have been configured, these will be included in the output:

```
[root@server3]# crsctl query css votedisk
 0.     0    /dev/raw/raw1
 1.     0    /dev/raw/raw2
 2.     0    /app/oracle/votedisk/votedisk_3
located 3 votedisk(s).
```

Discovery
Oracle Inventory

- On Linux the central inventory location is specified in
  - /etc/oraInst.loc
- For example:

```
inventory_loc=/u01/app/oraInventory
inst_group=oinstall
```

64 © 2008 Julian Dyke juliandyke.com

On Solaris the central inventory location is specified in:

• /var/opt/oracle/oraInst.loc

Discovery
Oracle Home Names

- To check the names of Oracle homes on Unix / Linux look in the central inventory e.g.

  /u01/app/oraInventory/ContentsXML/inventory.xml

```
<HOME_LIST>
<HOME NAME="OraCrs11g_home" LOC="/u01/app/11.1.0/crs" TYPE="O" IDX="1" CRS="true">
  <NODE_LIST>
    <NODE NAME="server3"/>
    <NODE NAME="server4"/>
  </NODE_LIST>
</HOME>
<HOME NAME="OraDb11g_home1" LOC="/u01/app/oracle/product/11.1.0/db_1" TYPE="O" IDX="2">
  <NODE_LIST>
    <NODE NAME="server3"/>
    <NODE NAME="server4"/>
  </NODE_LIST>
</HOME>
</HOME_LIST>
```

Discovery
Databases

- **Databases configured to run on a node can be discovered using:**

```
[oracle@server3 ~]$ srvctl config database
PROD
TEST
```

- **Instances for a specific database can be displayed using the -a option**

```
[oracle@server3 ~]$ srvctl config database -d TEST
server3 TEST1 /u01/app/oracle/product/11.1.0/db_1
server4 TEST2 /u01/app/oracle/product/11.1.0/db_1
server11 TEST3 /u01/app/oracle/product/11.1.0/db_1
server12 TEST4 /u01/app/oracle/product/11.1.0/db_1
```


Additional attributes can be displayed using the -a option for srvctl config database e.g.:

```
[oracle@server3 ~]$ srvctl config database -d TEST -a
server3 TEST1 /u01/app/oracle/product/11.1.0/db_1
server4 TEST2 /u01/app/oracle/product/11.1.0/db_1
server11 TEST3 /u01/app/oracle/product/11.1.0/db_1
server12 TEST4 /u01/app/oracle/product/11.1.0/db_1
DB_UNIQUE_NAME: TEST
DB_NAME: TEST
ORACLE_HOME: /u01/app/oracle/product/11.1.0/db_1
SPFILE: +DATA/TEST/spfileTEST.ora
DOMAIN: null
DB_ROLE: null
START_OPTIONS: null
POLICY:  AUTOMATIC
ENABLE FLAG: DB ENABLED
```

Some of the above information can be obtained from the /etc/oratab file (/var/opt/oracle/oratab on Solaris). However, this file must be manually updated if the database and its contents are not always very reliable.

Discovery
Interconnect Configuration

- Use ORADEBUG to dump the IPC information

```
[oracle@server3 ~]$ sqlplus / as sysdba
SQL> ORADEBUG SETMYPID
SQL> ORADEBUG IPC
```

- Determine the name of the current trace file:

```
SQL> ORADEBUG TRACEFILE_NAME
/u01/app/oracle/diag/rdbms/test/TEST1/trace/TEST1_ora_6806.trc
```

- Interconnect details are printed at the end of the IPC report

```
SSKGXPT 0xb7fd6aec flags SSKGXPT_READPENDING    socket no 5     IP
192.168.3.103       UDP 28801
```

- In the above example:
  - Interconnect IP address is 192.168.3.103
  - Protocol is UDP

In Oracle 11.1 and above, you can also obtain the name of the current trace file from the V$DIAG_INFO dynamic performance view. For example:

SQL> SELECT value FROM v$diag_info
WHERE name = 'Default Trace File';

VALUE

/u01/app/oracle/diag/rdbms/test/TEST1/trace/TEST1_ora_6806.trc

Discovery
Cluster Interconnects

- In Oracle 10.2+ [G]V$CLUSTER_INTERCONNECTS reports details of all private interconnects including:
  - IP Address
  - IP Name
  - Source
- Source can be
  - OS dependent software
  - Oracle Cluster Registry
  - CLUSTER_INTERCONNECTS parameter
- For example:

```
SELECT * FROM GV$CLUSTER_INTERCONNECTS;
```

INST_ID	NAME	IP_ADDRESS	IS_PUBLIC	SOURCE
1	eth1	192.168.2.1	NO	Oracle Cluster Repository
2	eth1	192.168.2.2	NO	Oracle Cluster Repository

The V$CLUSTER_INTERCONNECTS dynamic performance view was introduced in Oracle 10.2 and returns a list of cluster interconnects which can have been configured in the operating system, Oracle Cluster Registry or using the CLUSTER_INTERCONNECTS parameter.

V$CLUSTER_INTERCONNECTS only returns interfaces configured for the private network.

V$CLUSTER_INTERCONNECTS is based on the X$SKGXPIA fixed view. X$SKGXPIA was introduced in Oracle 10.1

Selecting from the global version GV$CLUSTER_INTERCONNECTS you can display all network interfaces currently configured in the private network (interconnects).

On most clusters there will only be one interface per node. The slide shows a typical configuration in which there the private network (interconnect) has been configured to use the eth1 interface on each node. The private network (interconnect) has been configured in the Oracle Cluster Registry which is normal for Oracle Clusterware.

Note that the Oracle Cluster Registry (OCR) is incorrectly described as the Oracle Cluster Repository in this view.

Discovery
Public Interconnects

- In Oracle 10.2+ [G]V$CONFIGURED_INTERCONNECTS reports details of all public and private networks including:
  - IP Address
  - IP Name
  - Public/Private flag
  - Source
- For example:

```
SELECT * FROM GV$CONFIGURED_INTERCONNECTS;
```

INST_ID	NAME	IP_ADDRESS	IS_PUBLIC	SOURCE
1	eth1	192.168.2.1	NO	Oracle Cluster Repository
1	eth0	147.43.1.101	YES	Oracle Cluster Repository
2	eth1	192.168.2.2	NO	Oracle Cluster Repository
2	eth0	147.43.1.102	YES	Oracle Cluster Repository

The V$CONFIGURED_INTERCONNECTS dynamic performance view was introduced in Oracle 10.2 and returns a list of all interconnects which can have been configured in the operating system, Oracle Cluster Registry or using the CLUSTER_INTERCONNECTS parameter.

V$CONFIGURED_INTERCONNECTS returns interfaces configured on both the public and private networks.

V$CONFIGURED_INTERCONNECTS is based on the X$KSXPIA fixed view. X$KSXPIA was introduced in Oracle 10.1

Selecting from the global version GV$CONFIGURED_INTERCONNECTS you can display all network interfaces currently configured in all public and private networks

On most clusters there will be two interfaces per node. The slide shows a typical configuration in which there the public network has been configured to use the eth0 interface on each node and the private network (interconnect) has been configured to use the eth1 interface on each node. Both networks have been configured in the Oracle Cluster Registry which is normal for Oracle Clusterware.

Note that the Oracle Cluster Registry (OCR) is incorrectly described as the Oracle Cluster Repository in this view