
ORACLE®

Essentials of Real Application Clusters
with
Some Nice Illustrations
and
Various Discursive Asides

David Austin
Server Technologies RAC/Grid Documentation
Oracle Corporation

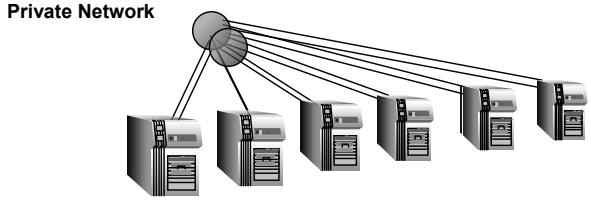
ORACLE

Agenda

- Definitions and Architecture
  - Cluster
  - Network
  - Disk
  - Memory
- Cluster Ready Services
- Real Application Clusters (RAC)
  - Installation
  - Configuration
  - Monitoring
  - Tuning
- Migrating from single instance to RAC

ORACLE

Cluster



ORACLE

A computer cluster is a set of two or more machines that can run in standalone mode, but which are joined through a network and are provided the capability to work together. The network could be a public network but for efficiency and synchronization, it is typically a high-speed private network. In Oracle Real Application Clusters, the only certified configurations involve private networks.

The graphic shows two networks. It is good practice to provide a second, redundant private network to provide continuous service following the failure of one of the networks.

Network Configuration – Private Networks

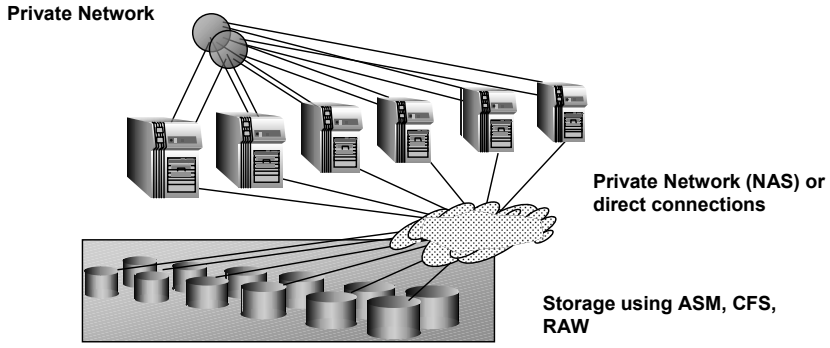
- Carries
  - Node participation and state info used by cluster manager
  - Cache fusion traffic
- Optionally
  - Access to shared storage when using NAS
- Recommendation
  - At least one private network (preferably 1 gigabit or more)
  - Additional private network if NAS used
  - Additional redundant network for high availability

ORACLE

At least one private network is required to carry the cache fusion and other internode traffic for RAC. A second private network is required for network attached storage access, if used.

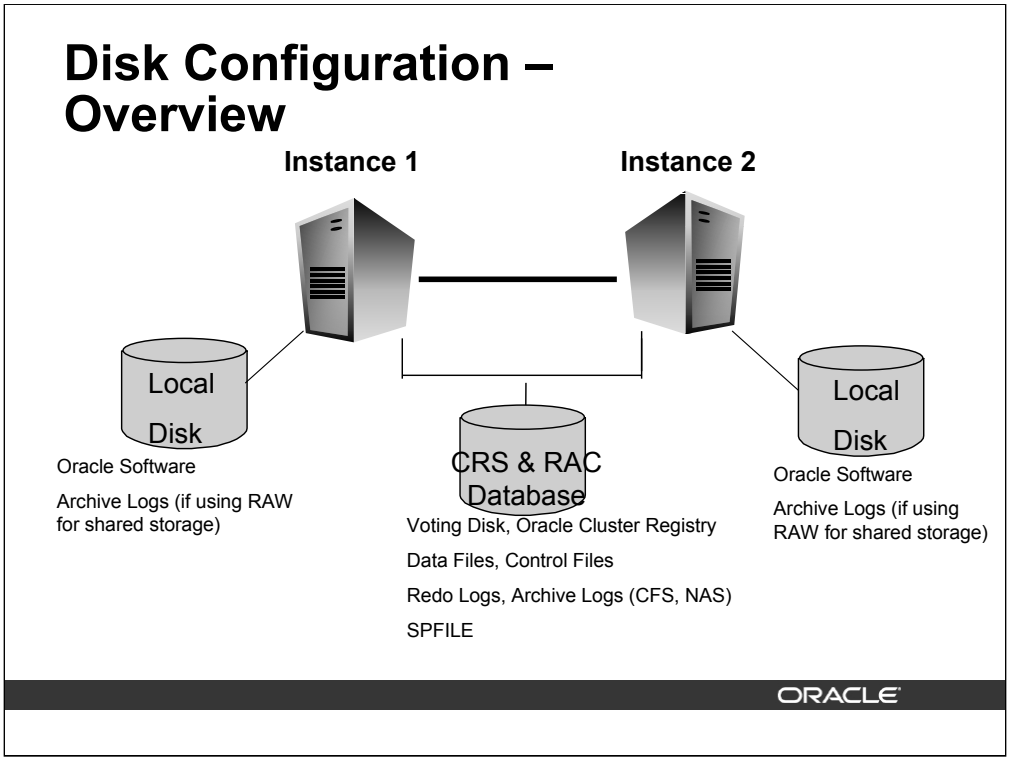
A redundant private network, that can replace the network carrying the cache fusion and other Oracle messages, is recommended to help avoid downtime in the case the primary interconnect fails. Access to this network is typically controlled by the vendor's software.

Cluster with Shared Disks



ORACLE

Most clusters are built to work on the same problem (program, application, database, and so on) concurrently. This typically requires sharing data and, therefore, dictates shared storage. Access to these shared disks can be shared using disk management software provided by the storage vendor, by the cluster hardware vendor, or by the software running on the cluster. In the case of Oracle RAC databases, the database software manages the access to the disks and works with a variety of storage architectures.



Disks can be connected using:

•Direct dual-ported connections to host bus adapter (if using 2 nodes only) but not a recommended method

•Hub/Switch fabric to host bus adapter (generally required if using >2 nodes) - recommended

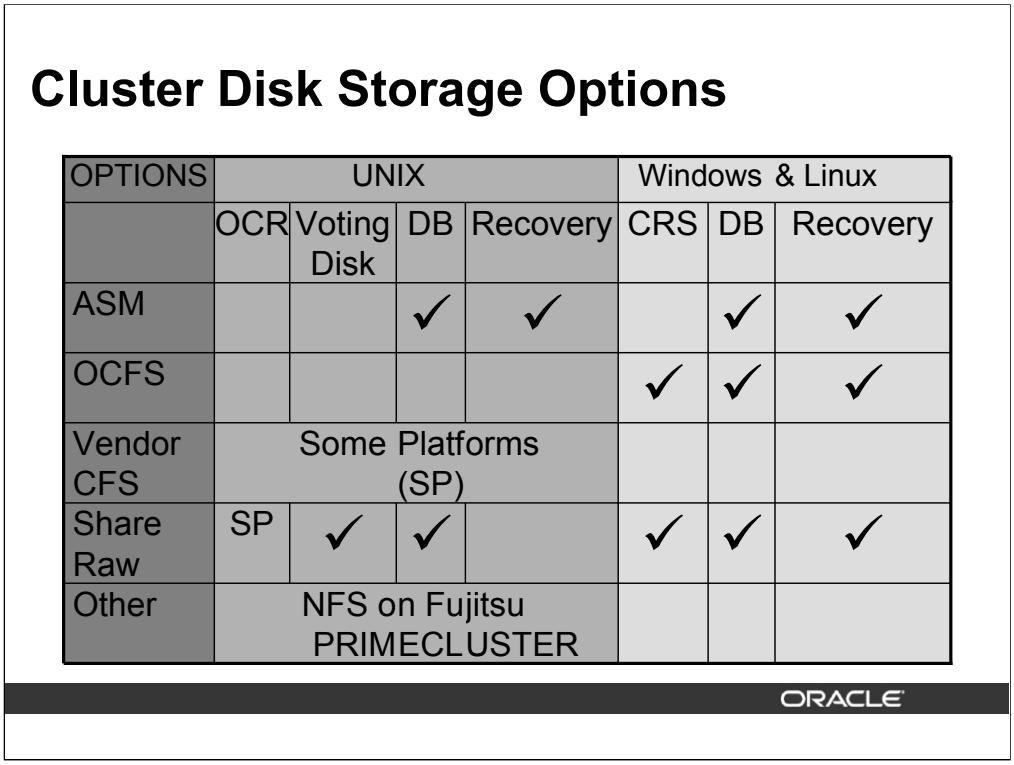
•Network – if using NAS

Datafiles, Control files, Redo Log Files, SPFILE need to be placed on shared disks so that they can be accessible by all instances.

Oracle Software needs to be placed on local file system.

Note: OracleCFS does not currently support Oracle Executables (because most I/O on OCFS is synchronous, and this is not good for regular file system files such as ORACLE_HOME and Oracle configuration files. This is expected to change in the 10i release timeframe, when caching support will be added. This is not a bug in the current OCFS release)

Archive logs should be on local file system if using RAW, or can be shared if using CFS/NAS



OPTIONS: The possible disk storage options, which are:

•ASM: Automatic Storage Management (required for RAC installed on Standard Edition Oracle Database 10*g*)

•OCFS: Oracle Cluster File System, available only on Windows and LINUX clusters

•Vendor CFS: Vendor-provided cluster file system, available from some UNIX vendors

•Shared Raw: Shared raw storage which includes shared raw disks, shared raw logical volumes, and shared raw partitions, one of more of which are available from most vendors

•Other: NFS file system, which is only supported with Fujitsu PRIMECLUSTER and a certified NAS device on Solaris with SPARC 64-bit-based systems

UNIX: The UNIX platforms for which this table is valid include:

•IBM AIX

•hp HP-UX PA-RISC (64-bit)

•hp Tru64 UNIX

•Solaris Operating System (SPARC 64-bit)

Windows and Linux: The operating systems for which this table is valid include:

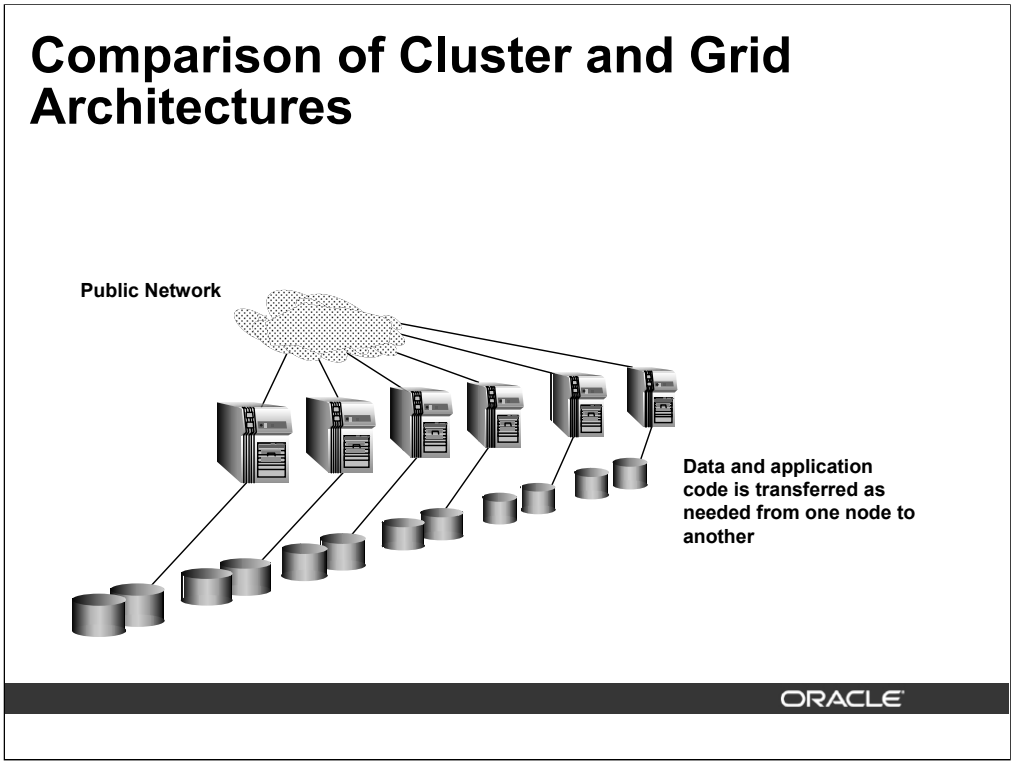
•Windows 2000

•Windows 2003

•Red Hat Enterprise Linux AS/ES 2.1 (Update 3 or higher)

•Red Hat Enterprise Linux AS/ES 3 (Update 2 or higher)

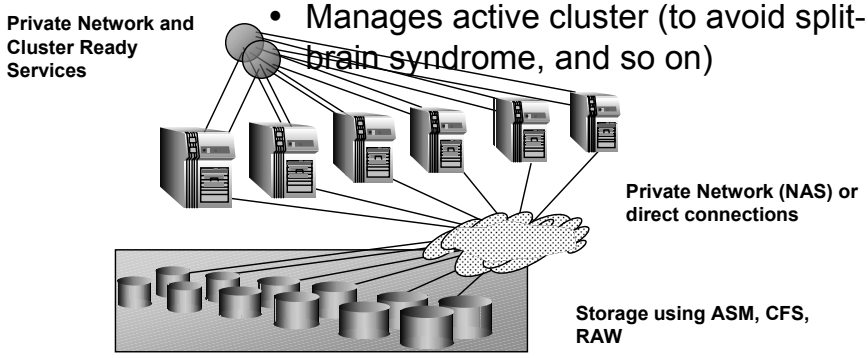
•SuSE Linux Enterprise Server (SLES) 8 (service pack 3 or higher)



In Grid computing, the nodes in the system can be dispersed geographically. The private network architectures are not designed for the distances involved and are replaced with public networks. Storage is also not shareable due to the distances involved and the resulting latency in disk reads and writes.

Cluster Ready Services

- Tracks installed cluster components: nodes, databases, instances
- Registers instance and services at startup
- Manages active cluster (to avoid split-brain syndrome, and so on)



ORACLE

Cluster Ready Services (CRS) is software that manage the cluster and the interaction between the nodes. The main functions of the CRS clusterware include:

•Tracking the installed components for RAC

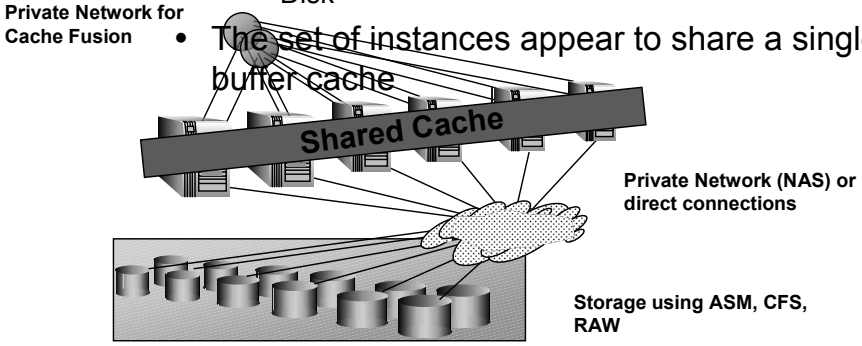
•Identifying the nodes that belong to a cluster and to a specific database

•Registering an instance with the correct database when it starts up and ensuring it starts up the services that are required to support Cache Fusion and related cluster activities

•Ensuring that only one set of nodes is included in the active cluster for a given database (avoids split-brain syndrome, a situation when an interconnect failure causes two sets of nodes, which are no longer in communication with each other, to believe the other set is inactive, thus they work independently of each other, potentially overwriting each other's work, causing data contamination).

Cache Fusion

- Allows each instance to obtain a required block image from the most efficient location
  - Another instance's buffer cache
  - A read-consistent image from another instance
  - Disk
- The set of instances appear to share a single buffer cache



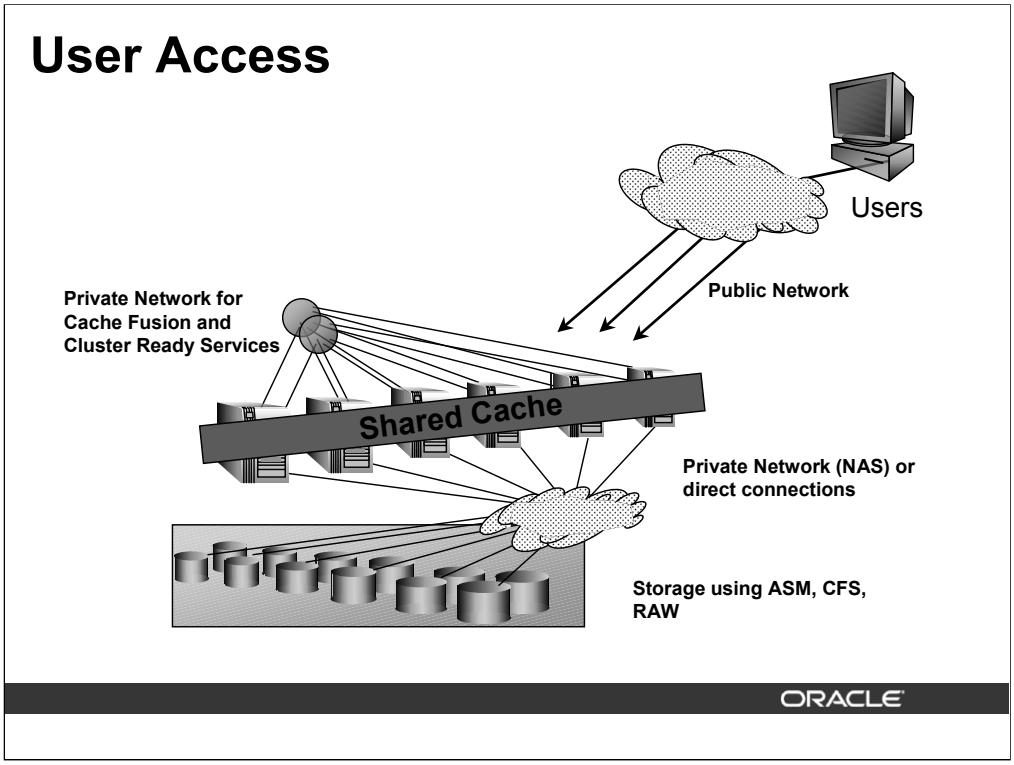
ORACLE

Cache Fusion is the technology that allows multiple instances, each on its own node, to share database blocks as if there were a single database buffer cache. This is accomplished by transmitting messages about block status, and block images themselves if necessary, between instances using the high-speed, private interconnect. The graphic shows this concept as a single shared cache. As with non-clustered databases, some blocks may have many different versions in memory at the same time. In a RAC database, these block versions can be in the same or in different instance buffer caches.

In most cases, an instance can acquire a block image from another instance faster than it could read the block fresh from disk. Therefore, the Cache Fusion process may be used to move current blocks between instances rather than having them re-read from disk. When a consistent block is needed, or a changed block is required on another instance, the Cache Fusion process efficiently transfers the block image directly between the involved instances.

You can use the same options to partition your buffer cache into caches for different uses (keep, recycle, or default) or different block sizes when using a RAC database. The block images are moved into the correct cache on the instance where it is used just as in single instance databases.

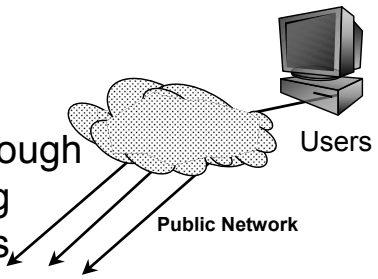
Distributed block status records and local information kept by each instance about the blocks in its buffers ensure that an instance failure will not cause any committed transactions to be lost and, in most cases, will allow ongoing queries to obtain required read-consistent versions of data blocks within the shared cache.



Users can be connected in client-server configuration, through one or more middle tiers, with or without connection pooling. Users may may be DBAs, developers, application users, power users (for example, data miners creating their own searches), and so on.

The public network is typically TCP/IP but can be built on any supported hardware and software combination. It is important not to use the private network for regular user traffic keep user traffic away from the private network otherwise cache fusion and other inter-instance activity will become backlogged, reducing the overall effectiveness of the cluster.

Virtual IP Addresses



- Standard connections through public network if not using Virtual IP (VIP) addresses
- VIP addresses
  - Require an unused IP address in addition to public address
  - Defined during installation
  - Provide improved failover capabilities while maintaining load balancing options

ORACLE

A Virtual IP address is a second public address that your connections use in place of the standard public IP address. To configure VIP addresses, you need to reserve a “spare” IP address, which matches the subnet of the public network, for each node in the cluster. RAC will maintain these IPs as Virtual IP addresses. Each one will be assigned to a node, and client programs use the VIP to access Oracle on that node. If the node fails, the VIP fails over to another node where it won't accept connections, which means that clients trying to connect receive a quick connection-refused error instead of waiting for a relatively slow TCP connect timeout.

VIP TNSNAMES Entries

Configuration when not using TAF

```
ERP= (DESCRIPTION=
  (LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-1vip) (PORT=1521))
  (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-2vip) (PORT=1521))
  (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-3vip) (PORT=1521))
  (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-4vip) (PORT=1521))
  (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

ORACLE

The slide shows the entries for a four-node cluster using VIP aliases to identify each host with Transparent Application Failover (TAF).The only difference between what you see here and what you would see for a non-VIP-oriented cluster is the use of the VIP aliases in the ADDRESS entries.

VIP TNSNAMES Entries

Configuration when using TAF

```
ERP= (DESCRIPTION=
   (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-1vip) (PORT=1521))
   (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-2vip) (PORT=1521))
   (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-3vip) (PORT=1521))
   (ADDRESS=(PROTOCOL=TCP) (HOST=clusnode-4vip) (PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=ERP))
   (FAILOVER_MODE=(BACKUP=ERP) (TYPE=SELECT)
     (METHOD=BASIC) (RETRIES=180) (DELAY=5)))
```

ORACLE

Here, the same information as seen for the non-TAF case but with the additional FAILOVER_MODE entry which provides failover capability. The connection to a failed node will be attempted on one of the surviving nodes when the VIP is moved. This avoids the long wait for a TCP timeout to occur when attempting to connect to a normal IP address.

Services in Oracle Database 10*g*

- In previous releases, the **SERVICE_NAMES** parameter and TNS entries identified a set of nodes in a cluster for failover and load balancing, as seen in previous example
- In Oracle Database 10*g*, services can be
  - Associated with service levels set in Resource Manager
  - Identified in callouts
  - Enabled and disabled on different nodes of the cluster
  - Allocated to preferred and available instances

ORACLE

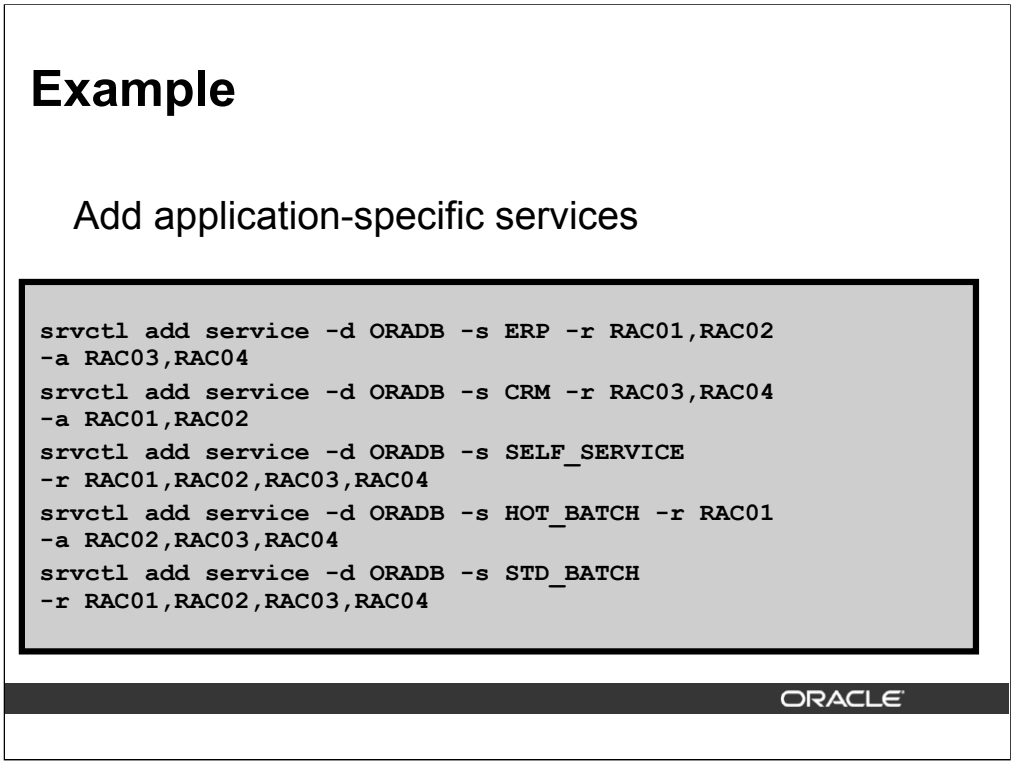
Oracle 10*g* provides a more sophisticated type of service than was available in previous releases. A service is an entity that can be managed with service-level commands that apply to the whole service regardless of the number of nodes with which the service is associated. Some of the features of services in Oracle Database 10*g* include:

•Enabling service level agreements by application. This is accomplished by setting resource limits in Resource Manager for an application-related service.

•Providing application APIs that allow a program to react to a changed condition in a service.

•Allowing you to associate a service with one or more nodes in a cluster and to change one or many of these associations without disabling the service on the unaffected nodes.

•Allowing you to define an instance for a service as *preferred* or *available*. A preferred instance is one where the service will attempt to run when started up; an available instance is one to which the service will migrate if it stops running on a preferred instance due to some type of failure.



The example shows five services being added to the ORADB database as follows:

ERP: Service for the Enterprise Resource Planning application which runs on instances RAC01 and RAC02 by default and on instances RAC03 or RAC04 in failover situations.

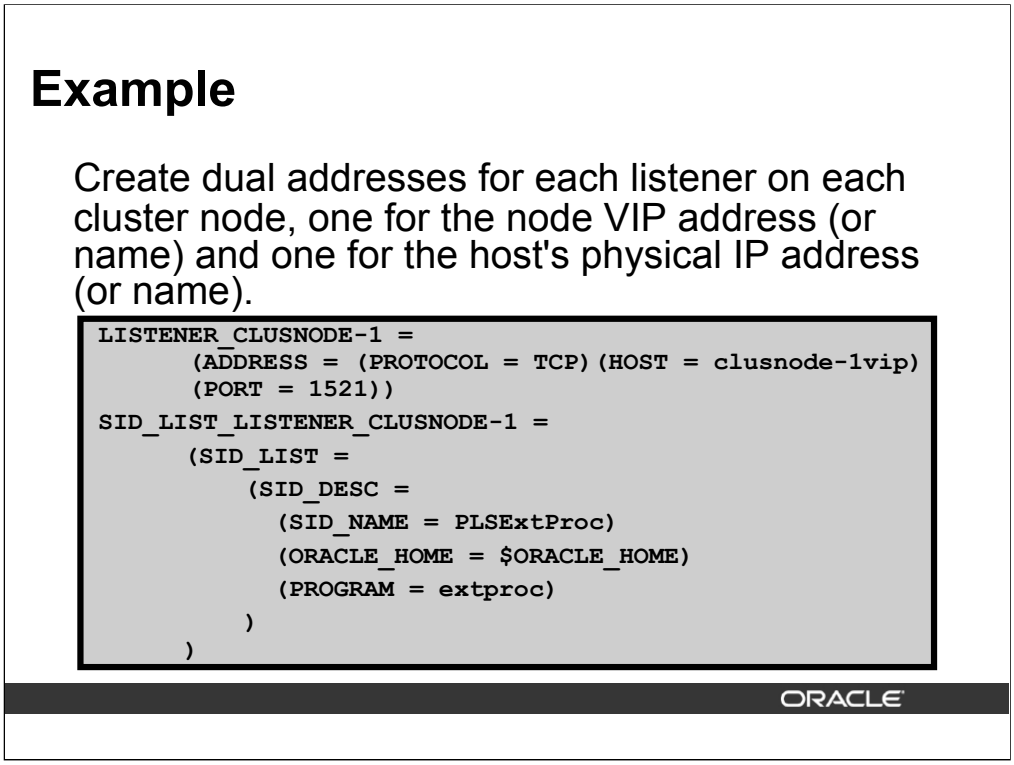
> **Note:** ERP is the service for which example TNS connection descriptions were shown earlier.

CRM: Service for the Customer Relationship Management application which runs on instances RAC03 and RAC04 by default and on instances RAC01 or RAC02 in failover situations.

SELF_SERVICE: Service for interactive user supports which runs on all four instances, RAC01, RAC02, RAC03, and RAC04, by default.

HOT_BATCH: Service for critical batch processing that has a limited period in which to complete. This service runs on RAC01 by default and can failover to any of the other three instances if necessary, RAC02, RAC03, and RAC04.

STD_BATCH: Service for non-critical batch processing which runs on all four instances by default and doesn't attempt to failover.



You should cross-register your listeners using your REMOTE_LISTENERS initialization parameter so that all of your listeners know about all of your services and the instances on which they run. The listeners should use server side load balancing, optionally based on session count for connection. The listeners must be listening on the VIPs and on the cluster aliases, when available. The listeners must not listen on the host name - listening on the host name will result in disconnected sessions when VIPs are relocated automatically back to their owning nodes.

This list shows the interconnect data for the example.

Public physical node names: clusnode-1, clusnode-2, clusnode-3, clusnode-4

Public IP addresses: 139.184.101.201, 139.184.101.202, 139.184.101.203, 139.184.101.204

Physical interface name(s): hme0 [, hme1] (the same for all four nodes)

Public virtual IP names: clusnode-1vip, clusnode-2vip, clusnode-3vip, clusnode-4vip

Virtual IP addresses: 139.184.201.1, 139.184.201.2, 139.184.201.3, 139.184.201.4

Logical interface names: hme0:1 [, hme1:1] (the same for all four nodes)

Private interconnect IP address: 172.16.0.1, 172.16.0.2, 172.16.0.3, 172.16.0.4

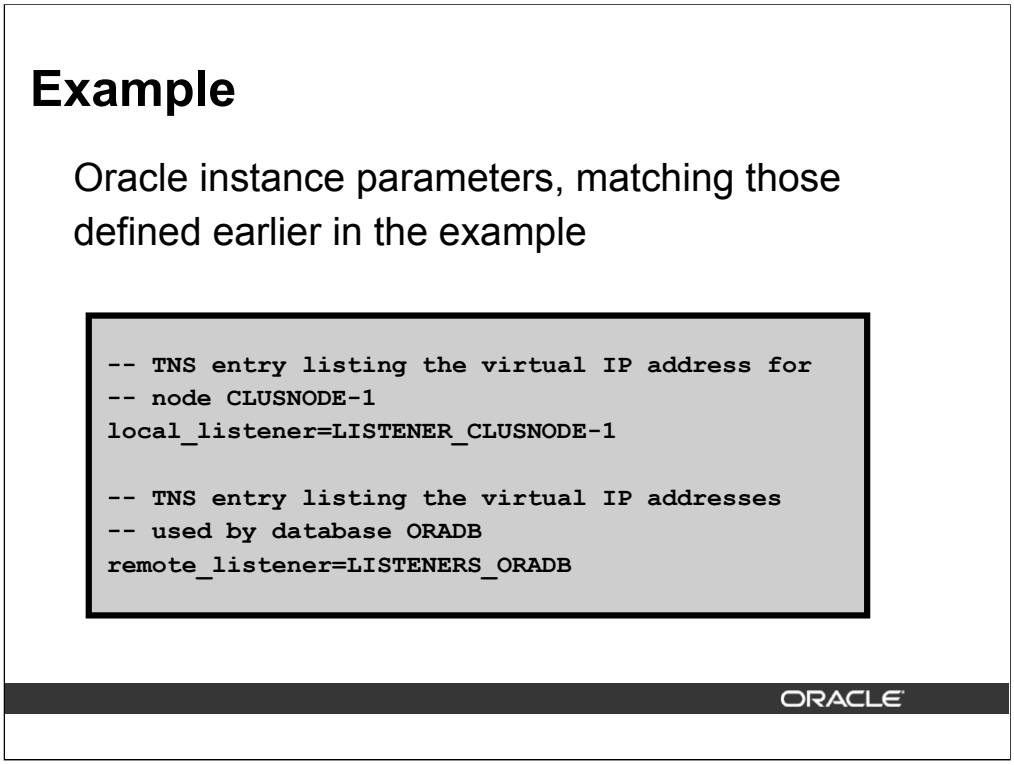
Physical interface name: qfe0 (the same for all four nodes)

Example

Sample remote **`listener.ora`** entries

```
# TNS alias entry maps to REMOTE_LISTENER
# initialization parameter:
LISTENERS_ORADB=
(ADDRESS_LIST =
(ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521)))
```

ORACLE



Ensure that the LOCAL_LISTENER, REMOTE_LISTENER, and ACTIVE_INSTANCE_COUNT initialization parameter values are valid to use the VIPs for your services. You must ensure that the ACTIVE_INSTANCE_COUNT parameter is left at its default value - this parameter must not be set.

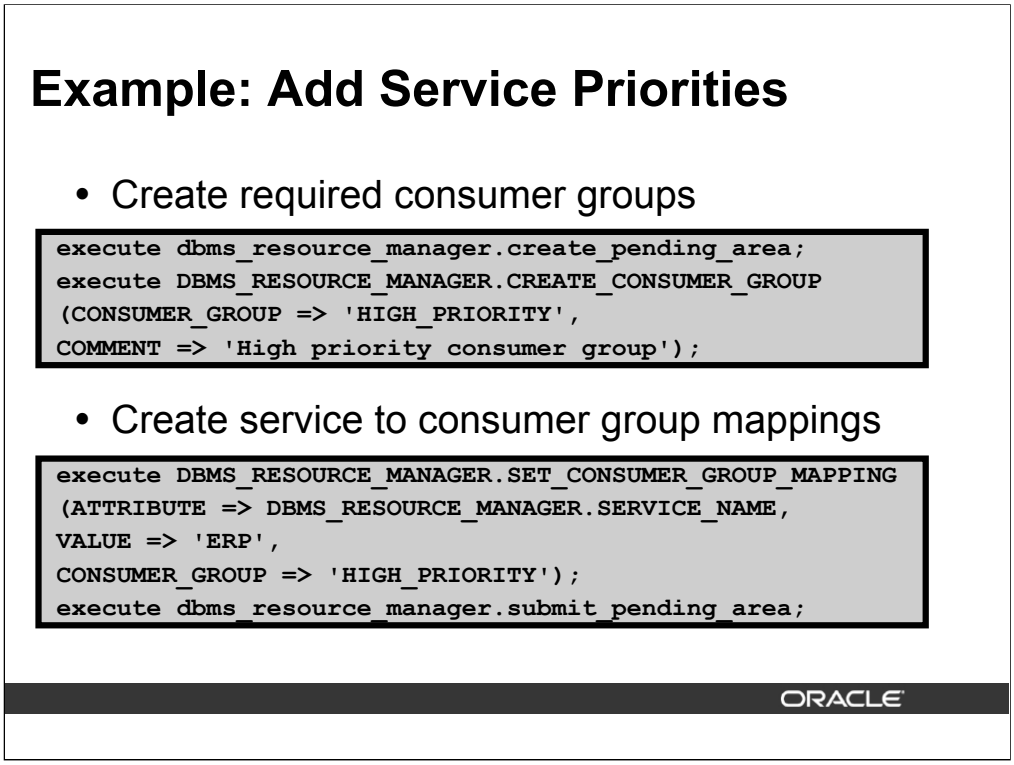
Manual Configuration for Workload Management

There are 4 steps to configure services for workload management

1. Add service priorities
2. Add job classes
3. Add service performance thresholds
4. Enable service, module, and action monitoring

ORACLE

The steps enumerated in the slide are discussed, with examples, on the next few pages.



For each level of service that you wish to use, create a related consumer group. For example, the five applications shown earlier, ERP, CRM, SELF_SERVICE, HOT_BATCH, and STD_BATCH, you decide on three levels of service, high priority, standard priority, and low priority. You then must associate the applications to the desired level of service by mapping them to the corresponding consumer group. In the examples above, the high priority consumer group is created and then the ERP application is mapped to this group.

Example: Add Job Classes

- The batch processes need to be associated with an appropriate batch job queue.
- Assume the database employs two batch queues managed by the Job Scheduler, called **HOT_BATCH** and **STD_BATCH**, corresponding to job classes with services of the same name. Create the required job classes with PL/SQL code as in this example:

```
execute DBMS_SCHEDULER.CREATE_JOB_CLASS( JOB_CLASS_NAME =>
'HOT_BATCH', RESOURCE_CONSUMER_GROUP => NULL,
SERVICE => 'HOT_BATCH', LOGGING_LEVEL =>
DBMS_SCHEDULER.LOGGING_RUNS, LOG_HISTORY => 30,
COMMENTS => 'P1 batch');
```

ORACLE

You can query the database for information about your job classes and service associations with a SQL*Plus query such as the following:

```
col service format a30 trunc
select JOB_CLASS_NAME, SERVICE from
DBA_SCHEDULER_JOB_CLASSES;
```

The query output would be similar to this:

```
JOB_CLASS_NAME                 SERVICE
------------------------------ ------------------
DEFAULT_JOB_CLASS
AUTO_TASKS_JOB_CLASS
HOT_BATCH                      HOT_BATCH
STD_BATCH                      STD_BATCH
```

The jobs executing in these job classes execute on instances offering the service.

Example: Add Service Performance Thresholds

- Add call elapsed time thresholds for the **ERP** and **HOT_BATCH** services on RAC01 as follows

```
REM ERP service, warn at 0.5 secs, critical at 0.75 secs:
execute DBMS_SERVER_ALERT.SET_THRESHOLD(
dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '500000',
dbms_server_alert.operator_ge,'750000', 1, 5, 'RAC01',
dbms_server_alert.object_type_service, 'ERP');
REM HOT_BATCH service, warning at 1.0s, critical at 1.5s:
execute DBMS_SERVER_ALERT.SET_THRESHOLD(
dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '1000000',
dbms_server_alert.operator_ge, '1500000', 1, 5, 'RAC01',
dbms_server_alert.object_type_service, 'HOT_BATCH');
```

ORACLE

You can add performance thresholds for all of your services and for all parameters that can be set in Resource Manager. In the above example the `DBMS_SERVER_ALERT.SET_THRESHOLD` package is called twice to set elapsed time per call thresholds on RAC0, once for **ERP** and once for **HOT_BATCH**. To set the thresholds for all nodes where these services run, you need to issue an additional call for each service on each node.

The thresholds being set are for a warning level and for a critical level. In the case of **ERP**, the warning level is set to half a second, converted to 500,000 microseconds (millionths of a second), and the critical level to three quarters of a second, converted to 750,000 microseconds. Similarly, the warning level for the **HOT_BATCH** service is 1 second (1,000,000 microseconds) and the critical level is one and half seconds (1,500,000 microseconds).

In both cases, the values of 1 and 5 are for the `observation_period`, designated in minutes, and the `consecutive_occurrences` count respectively. The former determines how often the metrics are computed for the threshold in question and the latter determines many times the computed results can exceed the specific threshold before the alert is issued.

Example: Enable Service, Module, and Action Monitoring

- Enable performance data and tracing for important modules and actions within each service.
- Query **V$SERV_MOD_ACT_STATS** for performance statistics
- This example enables monitoring for the exceptions pay action in the module, payroll, under the **ERP** service

```
execute DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(
SERVICE_NAME  => 'ERP', MODULE_NAME=>'PAYROLL',
ACTION_NAME => 'EXCEPTIONS PAY');
```

- Enable monitoring for the all actions in the module, posting, under the **HOT_BATCH** service

```
execute DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(
SERVICE_NAME => 'HOT_BATCH',
MODULE_NAME =>'POSTING', ACTION_NAME => null);
```

ORACLE

You can enable performance data and tracing for important modules and actions within each service. The performance statistics are available in the V$SERV_MOD_ACT_STATS view. The following commands, executed in a SQL*Plus session, perform these actions:

1. Enable monitoring for the exceptions pay action in the module, payroll, under the ERP service
2. Enable monitoring for the all actions in the module, payroll, under the ERP service
3. Enable monitoring for the all actions in the module, posting, under the HOT_BATCH service
4. Confirm the configuration by querying DBA_ENABLED_AGGREGATIONS

You can validate your monitoring setup by executing the following commands in a SQL*Plus session:

```
col AGGREGATION_TYPE format a20 trunc heading
   'AGGREGATION'
col PRIMARY_ID format a9 trunc heading 'SERVICE'
col QUALIFIER_ID1 format a7 trunc heading 'MODULE'
col QUALIFIER_ID2 format a14 trunc heading 'ACTION'
select * from DBA_ENABLED_AGGREGATIONS ;
```

The query output would look like:

```
AGGREGATION          SERVICE   MODULE  ACTION
-------------------- -------   ------- --------------
SERVICE_MODULE_ACTIO ERP       PAYROLL EXCEPTIONS PAY
SERVICE_MODULE_ACTIO ERP       PAYROLL
SERVICE_MODULE_ACTIO HOT_BATCH POSTING
```

Example: Using Services with Job Scheduler

- Use the **DBMS_SCHEDULER.CREATE_JOB** procedure to define jobs to execute under the job classes.
- This statement sets up the **MY_NAME.MY_PROC** procedure to run in the **HOT_BATCH** service as a result of the job class assignment defined earlier.

```
execute DBMS_SCHEDULER.CREATE_JOB(JOB_NAME =>
'my_report_job',
JOB_TYPE => 'stored_procedure', JOB_ACTION =>
'my_name.my_proc();',
NUMBER_OF_ARGUMENTS => 4,START_DATE => SYSDATE+1,
REPEAT_INTERVAL => 5,
END_DATE => SYSDATE+30, JOB_CLASS => 'HOT_BATCH',
ENABLED => TRUE, AUTO_DROP => false,
COMMENTS => 'my report on daily status');
```

ORACLE

Use the `DBMS_SCHEDULER.CREATE_JOB` procedure to define jobs to execute under the job classes. In this example, the `MY_NAME.MY_PROC` procedure will run in the HOT_BATCH service because of the job class assignment defined earlier.

Using Callouts for Fast Application Notification

- Custom-written application callouts are programs or shell script wrappers used to start and stop
  - on- or off-cluster applications
  - connection pools managed by middleware
- Callouts are immediately executed by RAC when a service or any part of the service starts, stops or fails to automatically restart.
- Actions that can be encoded as callouts (besides restarting applications) include:
  - restarting applications
  - logging fault tickets
  - e-mailing or paging administrators
  - invoking third-party event systems or clusterware components
- Callouts are not a requirement to deploy RAC-HA on CRS, but Oracle strongly recommends that you build notification mechanisms using callouts.

ORACLE

Custom-written application callouts are programs or shell script wrappers that can be used to start and stop on- or off-cluster applications, or connection pools managed by middleware. They are immediately executed by RAC when a service or any part of the service starts, stops or fails to automatically restart. Other actions that can be encoded as callouts (besides restarting applications) include: logging fault tickets, e-mailing or paging administrators, and invoking third-party event systems or clusterware components.

Callouts are not a requirement to deploy RAC-HA on CRS, but Oracle strongly recommends that you build notification mechanisms using callouts. Unless your CRS home directory is shared across the network, you must deploy each new callout under `/private/oracle/crs/racg/usrco` directory on each RAC node.

RAC Installation Steps

- Set kernel parameters
- Configure shared storage
- Install Cluster Ready Services on shared disk
- Install database software from one node
  - If not using shared disks, the installer will identify other cluster nodes and copy executables there
- Create Database
  - Use DBCA
- Configure Oracle Net
- For Enterprise Manager support
  - Can use local or grid management
  - Required processes automatically started by DBCA

ORACLE

The extra steps required for RAC that are not needed for a single instance are:

Configure shared storage

Install Cluster Ready Services

Other than that, RAC is as simple as single instance and there is no longer any need to be a RAC-expert (so all those DBAs can remove that specialty from their resume!).

Server Control Utility (srvctl)

- Command line tool for managing
  - Real Application Cluster databases
  - RAC instances
  - Services
  - Node applications (e.g. listeners)
- Fully documented in the “Oracle Real Application Clusters Administrator’s Guide”

ORACLE

SRVCTL provides a command line interface for managing the Oracle software on your cluster, including the database, its instances, node applications (listeners, and so on). It can add information to the cluster registry, start and stop individual or database-wide instances, associate services with instances, and similar activities.

Examples of SVRCTL commands include:

```
srvctl add database –d mgrt –o $ORACLE_HOME
srvctl add instance –d mgrt –i mgrt1 –n node1
srvctl start/stop –p mgrt –s mgrt1
```

RAC Configuration – Some Interesting RAC Parameters

Control File Parameters:

- **MAXINSTANCES = 16**
  - Specifies maximum number of instances that can access a database concurrently

Initialization Parameters:

- **CLUSTER_DATABASE = true**
  - If true, Oracle starts in shared mode
- **CLUSTER_DATABASE_INSTANCES = 2**
  - Used for default memory calculations
- **THREAD = 2**
  - Specifies the log thread for instance at startup
  - If not set, can change based on which instance starts first

**** Note DBCA configures all these parameters automatically ****

ORACLE

The parameters listed above are not all the parameters that impact RAC instances but they are only useful in a RAC instance. They are set by default or as the result of dialogs in the DBCA when you create your database with that tool.

RAC Configuration – Instance Specific parameters

- Use SPFILE for RAC
  - Not necessary to copy between nodes – file is raw device or cluster file system file
  - Maintains dynamically changed parameters
- For RAC, one parameter file can contain both common and instance specific parameters
  - SID.parameter_name = value
- Can combine cluster wide parameters with instance specific parameters:
  - *.db_cache_size=200M
  - SID1.db_cache_size=400M
  - Instance with SID1 will have a cache size of 400M
  - All other instances will have a cache size of 200M

ORACLE

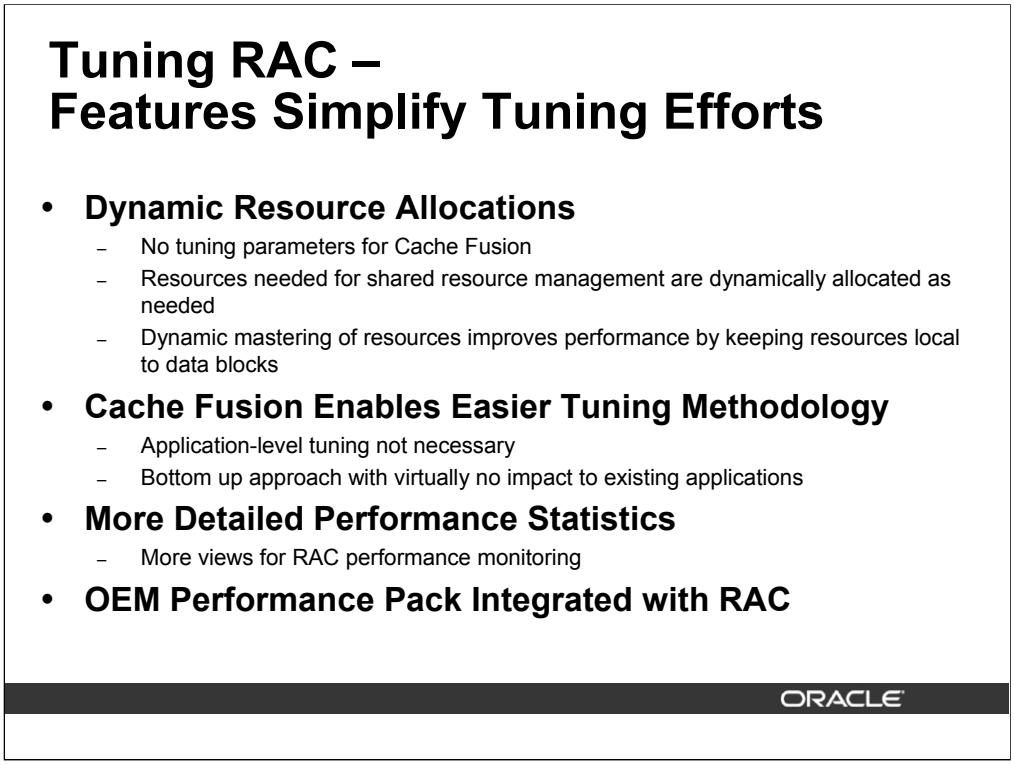
Note that the SPFILE is a binary file and cannot be edited directly. The examples with the SID prefixes are what you will see when you create a editable text version of the SPILE.

Monitoring RAC

- Use Oracle Enterprise Manager
  - Performance Manager has cluster-wide charts
  - Console has cluster-wide startup / shutdown interfaces
- Statspack
  - RAC aware
- **GV$** views
  - Global Views based on **V$** views

ORACLE

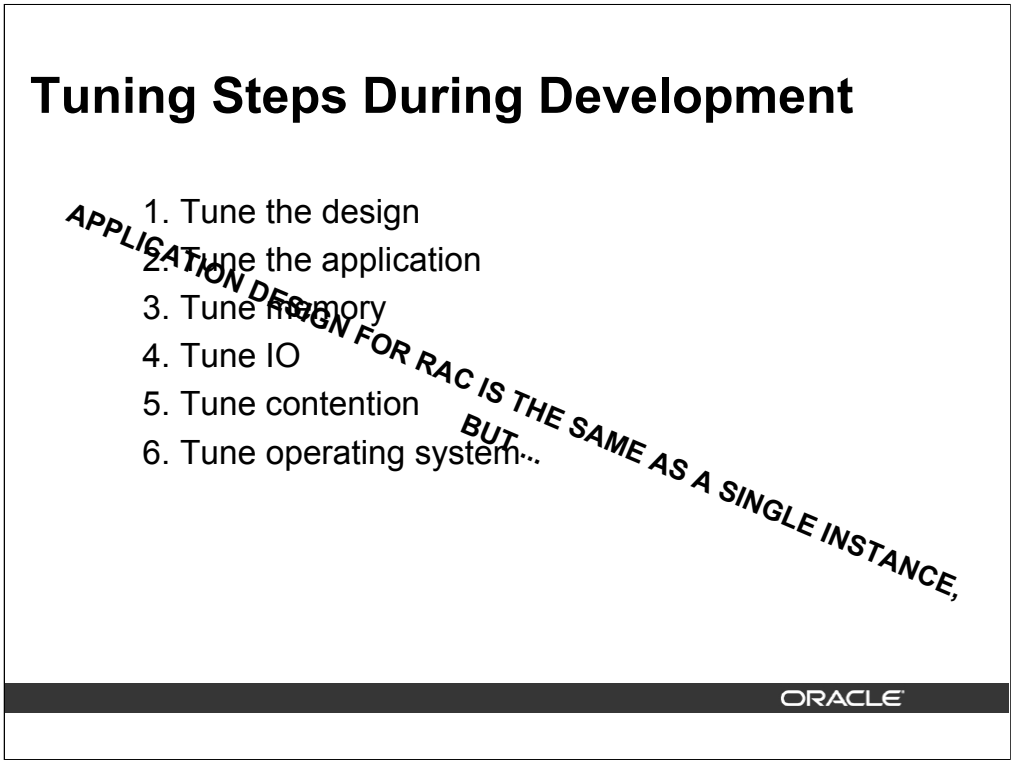
The `catclustdb.sql` script creates the `GV$` views – you should run this script following database creation in earlier releases where it is not automatically for you when using DBCA to create your database.



The point is you do not need to do special tuning for RAC – it scales out of the box!

However, the following slides talk about generic database tuning which may be of interest when deploying RAC – especially if you are looking for scalability across a greater number of CPUs.

Beware of applications that do not scale on an SMP platform – moving to RAC will not necessarily help and may even cause performance degradation.



Tuning Steps During Development

No special application design or coding required for RAC. All applications that run well in a single instance environment should run well on RAC

But….contention problems in single instance environment can be worse in a RAC environment.

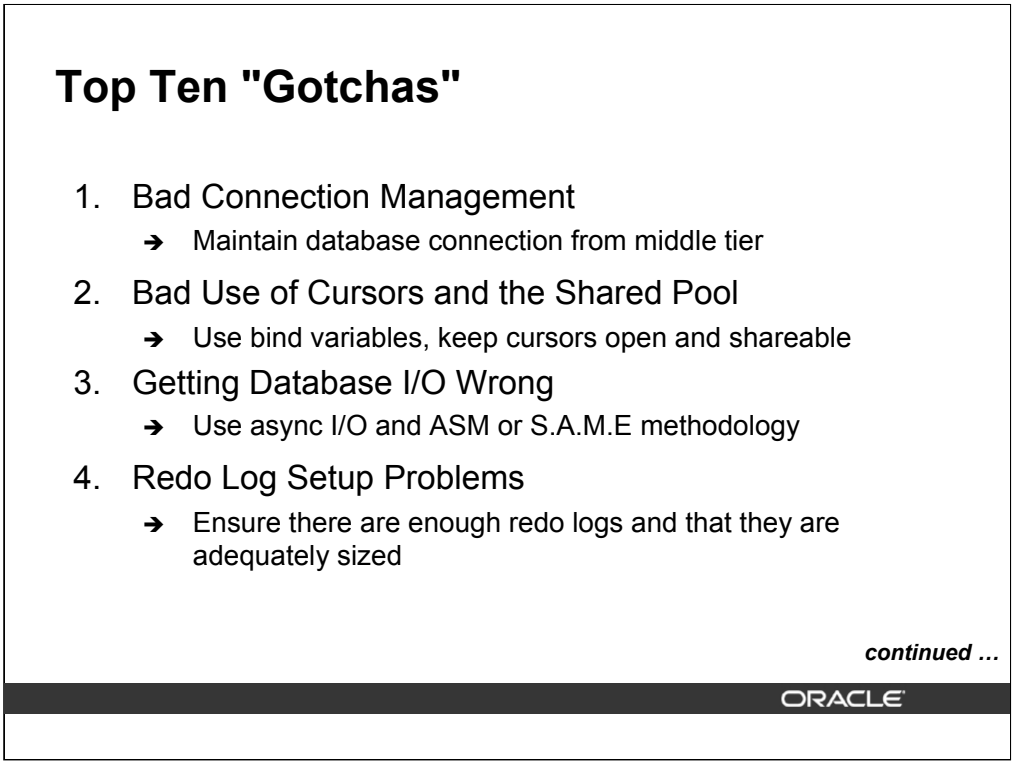
During the development of a new system, the recommended order in which to implement tuning is as follows:

1. The design
2. The application
3. The memory
4. Input/output (I/O)
5. Contention
6. Operating system

Repeat the process if your goals have not yet been achieved.

The rationale for this structure is that improvements early in the sequence may save you from having to deal with problems later. For example, if your applications are using a lot of full table scans, this may show up as excessive I/O. However, there is no point in resizing the buffer cache or redistributing disk files, if you can rewrite the queries so that they access only four blocks instead of four thousand.

The first two steps are typically the responsibility of the system architects and application developers; however, the DBA may also be involved in application tuning.



1.Bad Connection Management

The application connects and disconnects for each database interaction. This problem is common with stateless middleware in application servers. It has over two orders of magnitude impact on performance, and it is totally unscalable.

2.Bad Use of Cursors and the Shared Pool

Not using cursors results in repeated parses. If bind variables are not used, then there is hard parsing of all SQL statements. This has an order of magnitude impact in performance, and it is totally unscalable. Use cursors with bind variables that open the cursor and re-execute it many times. Be suspicious of applications generating dynamic SQL.

3.Getting Database I/O Wrong

Many sites lay out their databases poorly over the available disks. Other sites specify the number of disks incorrectly, because they configure disks by disk space and not I/O bandwidth.

4.Redo Log Setup Problems

Many sites run with too few redo logs that are too small. Small redo logs cause system checkpoints to continuously put a high load on the buffer cache and I/O system. If there are too few redo logs, then the archive cannot keep up, and the database will wait for the archive process to catch up.

Top Ten "Gotchas"

5. Serialization of data blocks in the buffer cache due to lack of free lists, free list groups, transaction slots (**INITRANS**), or shortage of rollback segments
   - ➔ Implement automatic segment free space management and system managed UNDO
6. Long Full Table Scans
   - ➔ Verify indexes, consider materialized views
7. Disk Sorting
   - ➔ Review execution plans and use automated user memory management (PGA_AGGREGATE_TARGET)

continued …

ORACLE

5.Serialization of data blocks in the buffer cache due to lack of free lists, free list groups, transaction slots (INITRANS), or shortage of rollback segments.

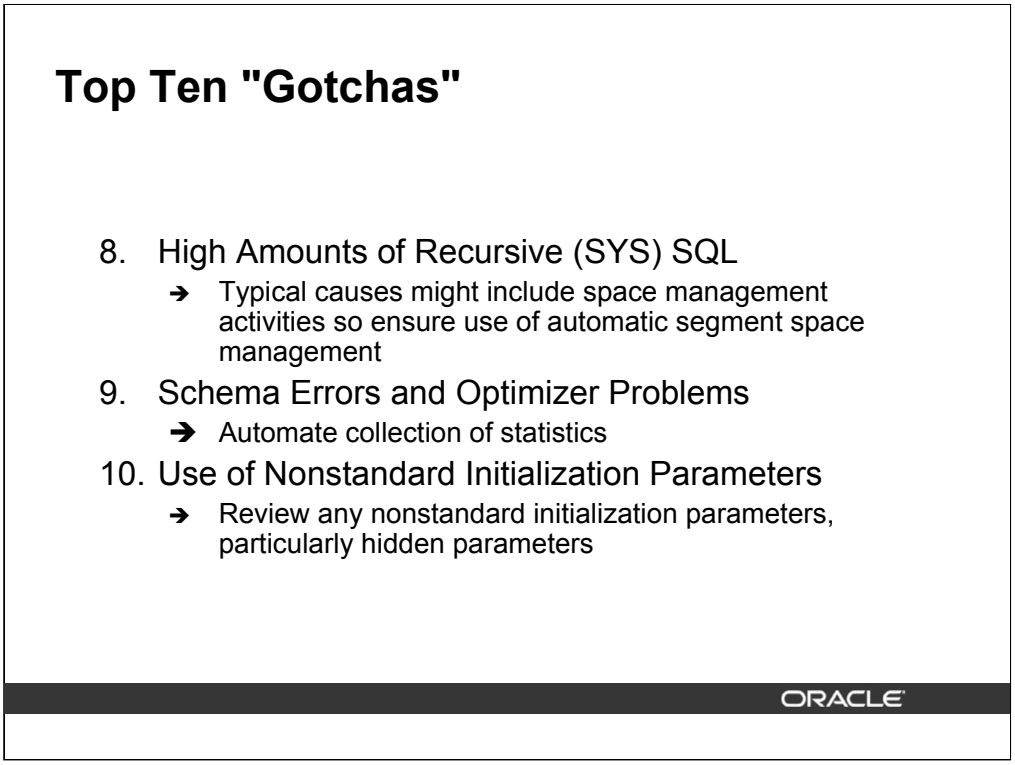
This is particularly common on INSERT-heavy applications, in applications that have raised the block size to 8K or 16K, or in applications with large numbers of active users and few rollback segments.

6.Long Full Table Scans

Long full table scans for high-volume or interactive online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Long table scans, by nature, are I/O intensive and unscalable.

7.In Disk Sorting

In disk sorts for online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Disk sorts, by nature, are I/O-intensive and unscalable.



8.High Amounts of Recursive (SYS) SQL

Large amounts of recursive SQL executed by SYS could indicate space management activities, such as extent allocations, taking place. This is unscalable and impacts user response time. Recursive SQL executed under another user ID is probably SQL and PL/SQL, and this is not a problem.

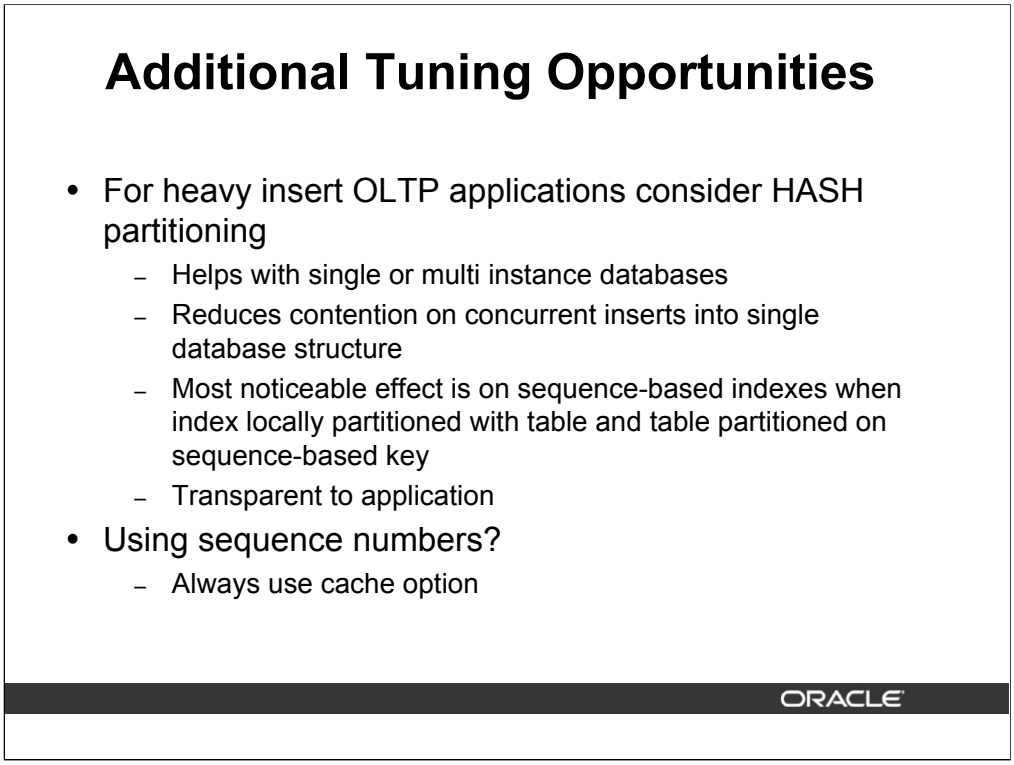
9.Schema Errors and Optimizer Problems

In many cases, an application uses too many resources because the schema owning the tables has not been successfully migrated from the development environment or from an older implementation. Examples of this are missing indexes or incorrect statistics. These errors can lead to sub-optimal execution plans and poor interactive user performance. When migrating applications of known performance, export the schema statistics to maintain plan stability using the DBMS_STATS package.

Likewise, optimizer parameters set in the initialization parameter file can override proven optimal execution plans. For these reasons, schemas, schema statistics, and optimizer settings should be managed together as a group to ensure consistency of performance.

10.Use of Nonstandard Initialization Parameters

These might have been implemented based on poor advice or incorrect assumptions. In particular, parameters associated with SPIN_COUNT on latches and undocumented optimizer features can cause a great deal of problems that can require considerable investigation.



HASH partitioning tables/indexes for OLTP can give great benefit in RAC or single instance.

Note that Index range scans can not be used on an index with hash partitioning.

Sequence numbers – with cache option be aware that:

- Sequence numbers can be lost
- No guarantee of order (even with ORDER option)


Q&A
ORACLE


ORACLE®