Unleashing Oracle Services: A Comprehensive Review of "Services" in Oracle Databases

Jeremy Paul Schneider

Abstract—Performance management and high availability have become base requirements for today's enterprise relational database management systems. In Oracle databases users create "services" to meet these challenges and those services touch almost every aspect of database configuration. This paper presents a broad yet thorough exploration and explanation of every aspect of services for both single-instance and cluster (RAC) databases.

CONTENTS

1 INTRODUCTION

1.1 A Mysterious New Feature

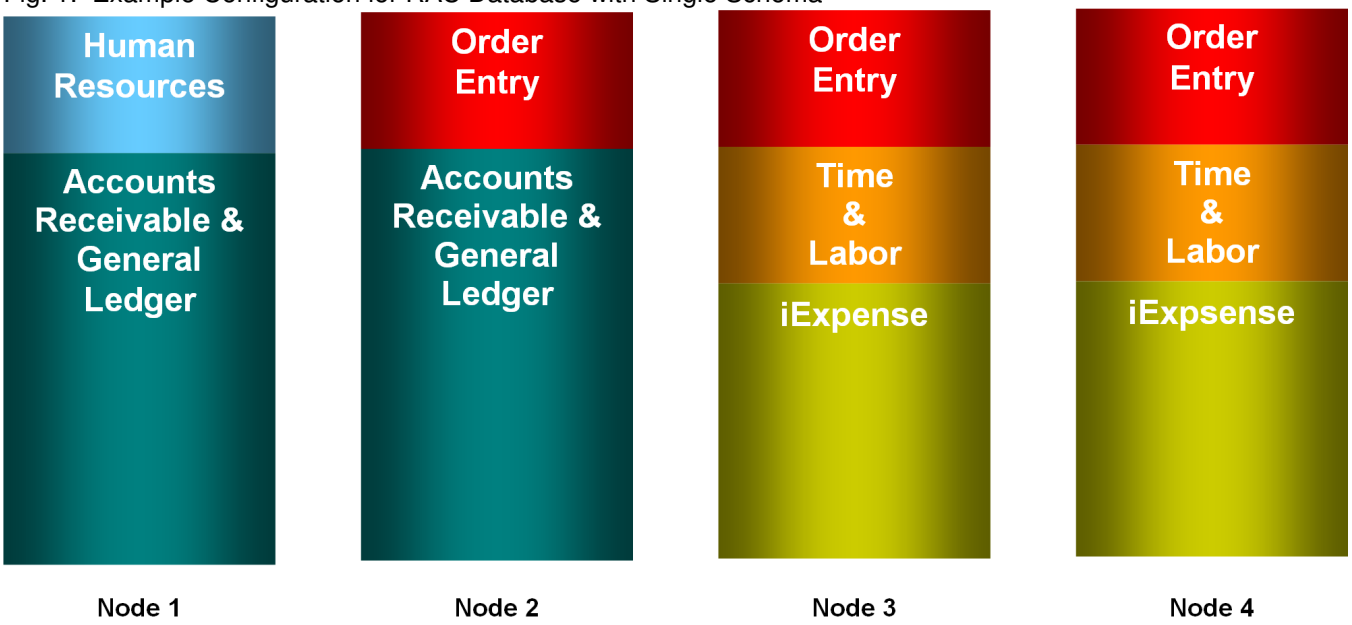
IN 1999, without too much buzz, Oracle 8i introduced a feature into the database networking stack providing a new connection method for clients. The feature was introduced through a simple initialization parameter called SERVICE_NAMES and fifteen sentences in the *New Features* chapter of the manual *Getting to Know Oracle8i* [Ora99]. Through another new feature called Dynamic Registration the PMON process passed the value of this parameter to the listener and clients could then specify only a hostname, port number, and service name to connect to the database. But at the time it seemed a little unclear what the advantages of this approach were; particularly in a single-instance environment. (In Oracle Parallel Server environments it did conveniently simplify Transparent Application Failover configuration.) In a single-instance database there didn't seem to be much difference between specifying the instance name or a service name.

However since its introduction this feature has steadily become more advanced and it now provides the foundation for an extensive and comprehensive workload management framework and high availability solution. The most recently released versions of Oracle have seen the greatest growth in services-related functionality. New features such as server-side failover configuration cannot be used without them and services are referenced in core system views such as V$SESSION and V$SQL.

Today services should be used in all Oracle databases. When planning new system deployments or new applications an understanding of services is a non-negotiable:

Definition 1: A **SERVICE** is a logical abstraction of a *single group of database clients* or a *single application* and is primarily used for performance management and high availability.

Fig. 1. Example Configuration for RAC Database with Single Schema



they should be included from day one alongside datafile layouts and backup strategies.

1.2 So What Exactly Is This Confangled Gadget?

Services have been a feature of Oracle for almost ten years and are recommended for all databases (see chapter 8 of the *Net Services Administrators Guide* [Ora05b]) and yet it's surprising how often they are overlooked and how little they are understood. In addition to their obvious role in Real Application Cluster (RAC) environments they also bring important capabilities to single-instance databases. Their functionality has significantly increased since they were first introduced and yet they still are often sidelined in database-related discussions.

The first question most people have about services is: "what exactly they are they?" In its most common usage, a service is a logical abstraction of a single group of database clients or a single application and is primarily used for performance management and high availability [Smi03][1]. Each service is configured with its own attributes, service level thresholds, and priorities; and database statistics are aggregated per-service. A service may represent one complete application out of several in a particular database, or perhaps one group of application users. For example: the manufacturing floor users, the shipping department, the reporting users, the batch processes, the Customer Relationship Management (CRM) application, the Manufacturing Execution System (MES) application, or the developers' wiki. Services should be used in every Oracle database no matter how small or large the application or user base. (Even if there is only one service defined.)

Most sites define only a handful of services in each database; usually there are fewer than ten distinguishable groups of users and applications per system. Extraneous services should not be created; a service should only be defined for each distinct group of users. Table 1 illustrates a database with four schemas and five groups of users. Three small applications (a customer portal, a custom CRM app, and a wiki) each have a schema in the database therefore they are each assigned their own service. The customer portal has two services defined: one for the portal software itself and a second for a nightly batch load process. Figure 1 illustrates a RAC configuration made up entirely of a single schema. In this example, taken from Dan Norris' white paper *RAC For Beginners: The Basics* [Nor06], there are five different groups of users who all share the same data.

You should always define at least one service at a minimum - but what is the maximum number of services you can or should define? In Oracle 10.2.0.1 you can't (and shouldn't) have more than about 20 active services

1. A less common second usage defines services based on data ranges; Object Relational (O/R) mapping layers and Transaction Processing (TP) monitors can then map work requests based on data keys [Ora06a, page 2-42 "Configuring Services"].

TABLE 1
Example Configuration for Non-RAC Database with Multiple Schemas

Service	Priority
Customer Portal	High
Customer Portal Nightly Bulk Load	Low
Custom CRM App	Medium
Developers' Wiki	Medium
RMAN Repository	Low

TABLE 2
Rules for DNS-compatible Service Names

Case insensitive.
Only contains characters “a” to “z”, “0” to “9”, and “-”.
The character “-” cannot be the first or last character.
Between 3 and 63 characters long.
Fully qualified domain name cannot be more than 255 characters.
Be descriptive.

TABLE 3
Example Service Names

Service	Name
Customer Portal	cust-portal-prod
Customer Portal Nightly Bulk Load	cust-portal-bulk-load
Custom CRM App	crm-prod
Developers’ Wiki	development-wiki
RMAN Repository	rman-repository

in a single database[2].

1.3 What’s In A Name

Like hosts, databases, and instances each service must of course be given a name. Every service should receive a valid fully qualified domain name that represents its function. Without an explicitly assigned domain, services inherit the domain of the database (the DB_DOMAIN initialization parameter). Under no circumstances should the domain name of your database and your services be left as ORACLE.WORLD or left blank; it should always be set to a valid DNS domain that your organization controls. You should have the ability in the future to add the database and service names to the authoritative nameserver if you decide to use the Easy Connect method for client connections.

When creating service names, being descriptive is more important than being short; service names can be up to sixty-three characters in length. Oracle will not throw an error if you create a service name longer than this but the DBMS_SERVICE package will not be able to distinguish between two services with the same initial sixty-three characters. Additionally, sixty-three characters is the maximum allowed length of a DNS entry according to RFC 1035 [Moc87]. Note that you should not use the underscore (_) character; the DNS specification disallows it.

Although abbreviations such as PROD and DEV are useful, avoid coding service names with non-obvious abbreviations. For example if you are naming a Boston-based development environment for an analytical CRM application then CRM-OLAP-BOSTON-DEV.ITCONVERGENCE.COM is a better service name than CRMABD.ITCONVERGENCE.COM. The key is that your service names should plainly describe your services.

The default behavior has changed in version 10g. In 8i and 9i there are no default services although the SERVICE_NAMES parameter does default to a value of DB_NAME + DB_DOMAIN. If you explicitly set the SERVICE_NAMES parameter to nothing in version 8i or 9i then Oracle will not register any services with the listener. There are two important implications of this:

1) You should always include DB_ NAME as the first service when you assign the SERVICE_NAMES parameter in pre-10g databases (DB_DOMAIN will be appended automatically).
2) Database names should always follow the same rules as service names (only use alphanumeric characters and “-” in the name). This rule applies to all versions of the database.

Whereas previous versions strictly follow the SERVICE_NAMES parameter, in every Oracle 10g database there are three default services which are independant of that setting:

1) Oracle always creates a service called DB_UNIQUE_NAME + DB_DOMAIN even if you explicitly set the SERVICE_NAMES parameter to nothing. DB_UNIQUE_NAME is a new parameter in 10g which defaults to the value of DB_NAME; typically it would only be set differently from DB_NAME in a Data Guard configuration. This default service will always be registered with the listener whether or not it is included in SERVICE_NAMES. However you can still specify DB_UNIQUE_NAME as the first service when you assign the SERVICE_NAMES parameter and Oracle will only register the service once.
2) There are two system services called SYS$BACKGROUND and SYS$USERS. Background processes run under the former and users who connect without using a service name (i.e. with a SID) run under the latter. These two system services are not registered with the listener and are only used for workload monitoring and reporting.

Although it may seem trivial, service names should be chosen carefully! Like good coding practices, good service naming practices are an important part of maintainable information systems. Choose service names that follow the rules for DNS compatibility and clearly describe the application or user group they represent.

2 UNLEASHING SERVICES IN ALL DATABASES

2.1 Start Your Engines

Getting started with services is as easy as setting a single instance parameter (unless you’re using RAC or MTS/Shared Servers which will be discussed later). In fact it doesn’t even require you to restart the database! All you have to do is assign a comma-separated list of service names to the initialization parameter SERVICE_NAMES - and Oracle will automatically create each service on-the-fly. After setting it you can check

2. The maximum number of active services is limited by the 255 character limit on the instance parameter SERVICE_NAMES.

Fig. 2. Creating Services

```
SQL> select value from v$parameter where name='db_domain';

VALUE
----------------------
lab.ardentperf.com

SQL> alter system set service_names='jtest2,cust-portal-prod,cust-portal-bulk-load,crm-prod,
development-wiki,rman-repository' scope=both;

System altered.

SQL> host lsnrctl status      -- the update may take a few seconds
...
Service "crm-prod.lab.ardentperf.com" has 1 instance(s).
  Instance "jtest2", status READY, has 1 handler(s) for this service...
Service "cust-portal-bulk-load.lab.ardentperf.com" has 1 instance(s).
  Instance "jtest2", status READY, has 1 handler(s) for this service...
Service "cust-portal-prod.lab.ardentperf.com" has 1 instance(s).
  Instance "jtest2", status READY, has 1 handler(s) for this service...
Service "development-wiki.lab.ardentperf.com" has 1 instance(s).
  Instance "jtest2", status READY, has 1 handler(s) for this service...
Service "rman-repository.lab.ardentperf.com" has 1 instance(s).
  Instance "jtest2", status READY, has 1 handler(s) for this service...
```

what services are active for the instance by querying the V$SERVICES dynamic performance view. It's important that your changes persist, so remember to update your init file or spfile in addition to updating the running instance. In addition to using SQLPlus you can also set the parameter with Enterprise Manager (called Database Control in 10g) or DBCA (at database creation time).

When a service becomes active the PMON process will connect to the listener and inform it of all currently available services. This process - called REGISTRATION - is automatically triggered when the database starts, when you update the SERVICE_NAMES initialization parameter, and every 60 seconds while the database is running (in case the listener restarts) [Ora06b, section 9.8.3.1 or page 9-6]. You can also force PMON to immediately register with the listener by issuing the command ALTER SYSTEM REGISTER. How does PMON know where to find the listener? By default it looks on port 1521. You can tell it to look elsewhere by setting the initialization parameter LOCAL_LISTENER with an alias from your local TNSNAMES.ORA or with an explicit connection descriptor. In fact it is a strong recommendation and best practice to always explicitly set this parameter even if your listener is running on the default port.

But don't let the simplicity of creating services fool you; what's happening behind the scenes is a little more complex. In version 10g, each time you update the SERVICE_NAMES parameter Oracle doesn't just update the listener - it also reads through the list of services and automatically creates a record in the data dictionary for new entries. You can see these entries in the ALL_SERVICES or DBA_SERVICES 10g static data dictionary views. Removing a service from the SERVICE_NAMES parameter will not delete the service but only take it offline[3]. You can manually delete services using the DBMS_SERVICE package. However if you're not using RAC then don't use the DBMS_SERVICE package to manually start or stop services; at least in version 10.2.0.1 it seems a little buggy. Just use the SERVICE_NAMES initialization parameter. Finally, you will notice a number of attributes in the data dictionary view ALL_SERVICES for each service; these apply to RAC and you don't need to worry about them if you're not using a cluster database. (They will be discussed later in this paper.)

2.2 Performance Tuning Version 3.0

No figure of Oracle lore is masked in mystery and shrouded in superstition like the performance tuning expert. Being regarded as a hero does give one a satisfying sense of importance and job security; however in recent years a tremendous amount of energy has been exerted to simplify, automate, and document the performance tuning process and many tasks that required an expert 5 years ago are handled automatically by the database engine today.

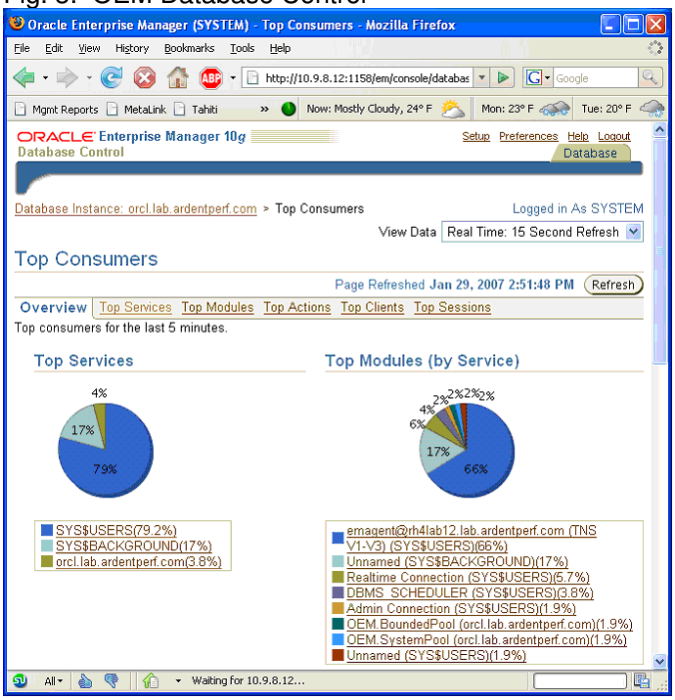
One of the most important performance tuning tasks is capturing the right historical and current performance information. Many tools and scripts have been designed over the years to accomplish this task and many of their most important elements are now incorporated directly into the database kernel. Oracle 8i was the first version to feature STATSPACK, vastly improving on Oracle's prior offering (BSTAT/ESTAT scripts) and moving the

3. It is important to be aware of this limitation. In Oracle 10.2.0.1 it seems that DBA_SERVICES will only keep track of about 100 services. If you create too many new services and don't delete old ones then the data dictionary will fill up and then silently ignore all new services. New services will be registered with the listener but will not be tracked or reported in any database views - even V$ACTIVE_SERVICES and V$SESSION.

logic into the database with a set of PL/SQL packages and jobs. Oracle Database 10g leapt forward again with the introduction of the Automatic Workload Repository (AWR). The AWR features a dedicated background process (MMON) for statistics collection (among other things) and features more statistics out of the box than almost any other commercial database product on the market. As long as the parameter STATISTICS_LEVEL is set to TYPICAL (the default) or ALL then this process is activated.

Services are an integral part of 10g statistics collection. All statistics are automatically aggregated for each service so that it's easy to compare which services are consuming what resources. This makes a powerful starting point for identifying which application or group of users is causing a performance problem. OEM Database Control makes it simple to drill down into statistics by service. For instance the TOP ACTIVITY and TOP CONSUMERS pages both show aggregated service information. Furthermore, the GENERAL tab of each session and the ACTIVITY tab of each SQL statement also display their associated service name.

Fig. 3. OEM Database Control



Even more information is available through dictionary views for scripts and for administrators who prefer the command line. V$SERVICE_STATS and V$SERVICE_EVENT contain a wealth of statistics for each service in the database. V$SERVICE_WAIT_CLASS contains statistics aggregated by class (e.g. APPLICATION and USER I/O). These views can provide a valuable system-level picture of how your resources are being used by each service. Also, DBA_HIST_SERVICE_STAT and DBA_HIST_SERVICE_WAIT_CLASS are AWR views that contain historical records of these statistics.

The V$SESSION and V$SQL dynamic performance views contain associated service names for each active session and each plan in the SQL area The service associated with a session is established at connect-time based the client's connect string and does not change for the duration of the session. Shared SQL areas, on the other hand, are shared between services. The SERVICE_NAME and SERVICE_HASH fields of the V$SQL table only reflect the service of the connected user who parsed the SQL statement. It is important to note that adding services for different groups of users on a single application will not increase the memory requirements for your shared pool.

You can also focus an AWR report on a specific service: the DBMS_WORKLOAD_REPOSITORY ASH_REPORT_TEXT and ASH_REPORT_HTML functions each accept a parameter L_SERVICE_HASH which narrows the report to a specific service and the ASHRPTI.SQL script will prompt for this parameter. (See the documentation in ASHRPT.SQL for how to pass this parameter directly to the ASHRPTI.SQL script.) Oracle requires additional licensing to use any AWR functionality; make sure that you own the appropriate licenses before querying any DBA_HIST_* views or using any DBMS_WORKLOAD_REPOSITORY functions. Also check that the appropriate licenses are enabled or disabled in the SETUP section of OEM Database Control.

2.3 Tuning Is Like The Orchestra (It's All About Instrumentation)

Instrumentation is the key to designing applications for growth and maintainability. Accordingly, understanding the SERVICE > MODULE > ACTION hierarchy is critical to sound application design with Oracle 10g. A service, of course, represents a single application or group of users. Services should then be composed of modules: logical sections of the application such as a single Oracle Form or code segment. Modules are finally composed of actions: individual logical transactions. Registering a module and action allows you to track performance and resource usage at these levels. Furthermore you can map modules and actions to resource classes and you can even enable SQL Trace for a specific module or action - a tremendously useful capability with, for example, Java applications that use connection pools. (These are notoriously difficult to trace since a single process may utilize several different connections.) Registration requires nothing more than a few quick calls to the DBMS_APPLICATION_INFO package or a call to a library such as Hotsos's freely available Instrumentation Library for Oracle (ILO) [Hot07]. And there is no need to be concerned about overhead - there's hardly an impact and the benifits far outweigh it. In fact the OCI client even further optimizes the process; the information is combined with other network calls to eliminate extra round trips and bytes transferred.

The first major advantage once an application registers modules and actions is that you can use

Fig. 4. Server Response Time Thresholds

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
  metrics_id => DBMS_SERVER_ALERT.ELAPSED_TIME_PER_CALL,
  warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE,
  warning_value => '500000',
  critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE,
  critical_value => '750000',
  observation_period => 30,
  consecutive_occurrences => 3,
  instance_name => NULL,
  object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE,
  object_name => 'web-oe-prod');
```

OEM Database Control or the DBMS_MONITOR package to collect detailed statistics and traces. In OEM Database Control this is simple; just navigate to the details screen for a service or module (for example by clicking on a module name from the TOP CONSUMERS screen). You can then select modules or actions and use the web interface to enable or disable statistics aggregations and traces. Alternatively, you can call DBMS_MONITOR's SERV_MOD_ACT_STAT_ENABLE function to enable statistics gathering for any combination of services, modules, and actions. You can then query the view V$SERV_MOD_ACT_STATS to see the statistics for each combination. To enable SQL Trace, call DBMS_MONITOR's SERV_MOD_ACT_TRACE_ENABLE function. Note that the SQL Trace functionality is strictly hierarchical: you must specify a service, a service and module, or a service and module and action. Also, the trace information will be spread out over many files so you will need to use the TRCSESS tool to aggregate it.

The second major advantage once an application registers modules and actions is that you can use OEM Database Control or the DBMS_RESOURCE_MANAGER package to map application code segments into Oracle's powerful resource plans at very broad or granular levels. The Oracle database engine has the capability to dynamically change job priorities at run-time as they switch from one action or module to the next. In OEM Database Control you can choose the CONSUMER GROUP MAPPINGS option from the ADMINISTRATION tab to create or change mappings. However it is better to use DBMS_RESOURCE_MANAGER's SET_CONSUMER_GROUP_MAPPING function which presently offers a bit more flexibility than OEM. (OEM in Oracle 10.2 does not yet allow you to specify service/module or service/module/action mappings although it does allow service only, module only, and module/action mappings.) You can show the current mappings by querying the DBA_RSRC_GROUP_MAPPINGS view. You should also remember to check and set the mapping priorities. These priorities can be set from the PRIORITIES tab in OEM or using the DBA_RSRC_MAPPING_PRIORITY view and DBMS_RESOURCE_MANAGER's SET_CONSUMER_GROUP_MAPPING_PRI function.

For any application that runs on the Oracle database stack, it is important that services, modules, and actions are understood and incorporated into the design from day one. A small effort to configure services and to label modules and actions will pay large dividends in understanding and controlling how your application interacts with the database.

2.4 If It Ain't Broke... Create A Notification So You Know When It Is

We have not yet reached the point of predicting every database problem before it happens, but Oracle made significant progress for proactive tuning with the introduction of server-generated alerts in version 10g release 1. This new management feature is built on top of the Oracle Streams Advanced Queuing engine (added in Oracle 8). Essentially, the MMON process is now doing what OEM did in version 9i: monitoring performance metrics and generating alerts when they exceed configured thresholds. The alerts go into a SYS-owned queue called ALERT_QUE which OEM Database Control and OEM Agents subscribe to. It is also possible to configure Oracle to automatically send email notifications of alerts. You can even subscribe to this queue with your own applications using the DBMS_AQ package and methods if you want to further customize alert monitoring and handling. Outstanding alerts are visible in the view DBA_OUTSTANDING_ALERTS and old alerts remain visible in the view DBA_ALERT_HISTORY.

Server generated alerts are controlled by user-configurable THRESHOLDS, including some that are measured by service. This means that every service in the database can have a different WARNING and CRITICAL threshold for each of these metrics. With Oracle's initial implementation of server-generated alerts two metrics are supported: elapsed time per call and cpu time per call. (I hope to see more development in this area for future releases.) For example a public web-facing order-entry service may require very low elapsed time per call, but a reporting service on the same data might expect much higher average elapsed time per call. Oracle already maintains per-service statistics; now it will use those service-level statistics with your custom thresholds to generate appropriate alerts and give you better awareness of your database's responsiveness.

2.5 Getting Connected (Clients)

The grid computing vision is having a wider impact on corporate data centers today than ever before. Since release 10.1 Oracle has preached the message rather persistently that grid computing concepts - presently championed by the academic community - have much to offer the commercial world. In an interesting twist, components like Oracle RAC and OEM's provisioning tool are most trumpeted as keys to a grid-based strategy. But the term GRID COMPUTING, as used for years now by the researchers who coined the term in the 90s, actually describes a set of middleware web services [Sto07], [Fos02], [FKT01][4]. Nonetheless, the concept of computing as a utility can be applied to many levels of infrastructure - databases included. And Oracle database services are an integral part of this blueprint as they provide database clients with a new level of virtualization and database transparency.

As a general rule, Oracle clients should always connect using a SERVICE_NAME and should never connect using a SID. If you are using a connect descriptor (whether in a TNSNAMES file or embedded in the connection string) then the service name belongs in the CONNECT_DATA portion where the SID was set in pre-8i databases. It is very common for upgraded databases and applications not to use service names! Check the systems in your organization and make sure that the Oracle clients are in fact connecting properly. If you don't use the SERVICE_NAME connect parameter then you will be unable to take advantage of Oracle's services framework at all, since every connection will go into the default SYS$USERS service.

The real potential of services for virtualization can be realized by combining them with DNS aliases. By making your applications entirely unaware what database or server they are connecting to you can move data to a different database on a different server without needing application updates. The services can be moved to the new database and the DNS aliases can be updated to point to the new server; the application will connect to the same service without realizing that it has been moved. (You will of course want to make sure that the timeouts are set appropriately low - probably zero - on your DNS server to use this trick. You will also probably want a dedicated subdomain for databases and services to avoid name collisions and to minimize the impact of the low cache settings.) The DNS alias should have the same fully qualified name as the service; this is one reason why it is important to put your databases and services in a domain that you control.

The second advantage to this approach is that on the client side TNSNAMES setup becomes optional. Since version 8.0 Oracle Clients will automatically attempt to locate a DNS entry for each service. With an 8i or 9i client it is called the HOSTNAME Naming Method; for example in SQLPlus you simply connect with USERNAME/PASSWORD@SERVICE. In Oracle 10g you can also specify a non-default port by using USER/PASSWORD@SERVICE:PORT and it is called the EASY CONNECT Naming Method. It is also possible in 10g to specify a hostname that is different from the service name although it is best to make the DNS alias the same as the service name for better understandability. You can still use TNSNAMES file and you must use it to configure any custom settings such as client-side Oracle Net tracing.

TABLE 4
Complete Service Virtualization

Dedicated DNS subdomain for services and databases.
DNS caching disabled for subdomain.
All services and databases registered in subdomain as CNAME pointing to hosting server.
Only use TNSNAMES entry on client if custom settings are needed.

2.6 When The Server Is The Client (Distributed DBs, Scheduler, and PQ)

There are several situations where the Oracle server becomes a client for itself including Database Links, Oracle Streams, the Job Scheduler, and Parallel Execution. For these server-side processes the service name is configured as part of their definition. All of these processes inherit the resource profiles and count toward the statistics of their respective services.

Oracle-based distributed database systems are offer a sophisticated solution for complex data sharing needs. Oracle introduced interdatabase connectivity in version 5 and considerably simplified it with database links in version 6. Furthermore the database offers tight integration with external transaction managers[5]. Today this provides the foundation of a very powerful distributed database engine that's even capable of seamlessly integrating any 3rd party database that has an ODBC or OLE driver. One fundamental component of distributed database configuration is that every Oracle database has a GLOBAL_NAME which is assigned at database creation time using the DB_NAME (not DB_UNIQUE_NAME) and DB_DOMAIN and stored in the data dictionary. For the past several years Oracle has strongly recommended that database link names always match the global names of the databases they reference. Furthermore it is recommended to set the GLOBAL_NAMES initialization parameter to TRUE to enforce this. (In fact some Oracle features require it to be set.)

4. Wolfgang Gentzsch from Sun Microsystems offered an alternative viewpoint defining GRID COMPUTING in much broader terms [Gen02]. However he still emphasized resource sharing among heterogenous, autonomic virtual organizations; not single-platform provisioning or clustering.

5. Since version 7, Oracle Database has offered integration with transaction processing (TP) monitors including BEA Tuxedo, IBM CICS, and NCR TopEnd through an implementation of the *XA Interface API* - part of the X/Open Distributed Transaction Processing (DTP) model [X/O91], [Ora05a, chapter 15].

This has implications for service strategies on distributed systems. First of all, it means that synonyms are required to virtualize data location within distributed applications since database links should always reference remote data using the globally unique database name. If a service in a distributed system is moved from one database to another then you must drop the old database links that referenced it and create new ones with new names. You also will need to drop and recreate the synonyms to point to the new database links. (Synonyms are important so that you don't have to update all of your code!) Secondly, these recommendations require the use of a special syntax for link names if a single database offers multiple services which are being accessed by distributed transactions. The name of the database link should be the global database name followed by the "@" character and an identifier. For example JTEST2.LAB.ARDENTPERF.COM@CRM-PROD. You can put any text after the "@" to distinguish between links such as different services or different fixed-user connections to the same service.

Oracle Streams, Replication, and Advanced Queuing are all built on database links and require GLOBAL_NAMES to be TRUE. If you are using any of these features then you assign the service name in the TNSNAMES entry or connect descriptor when creating the database link and you will need to take the aforementioned points into consideration.

The server also becomes a client when Oracle's job managers are used and services provide a few valuable extensions to the new scheduler's functionality. DBMS_SCHEDULER manages services as an attribute of Job Classes. By default jobs will not take advantage of services because they will run under the class DEFAULT_JOB_CLASS which is configured at database creation without being tied to a service. If a job class is not assigned to a service (by default they are not) then its jobs will run under the internal service SYS$USERS. However there are two major advantages to tying jobs to a service:

1) All statistics for those jobs will automatically be aggregated with the service. Oracle also automatically assigns a module and action name so that you can collect detailed statistics for each job if desired.
2) When you offline a service Oracle will suspend all jobs tied to that service until the service is brought back online. This gives you a new degree of control in job management by logically tying jobs to their services.

Note that although it does not need to be online, a service must exist (visible in the ALL_SERVICES view) before you can tie a job class to it. Services are added to the data dictionary when first brought online; if you want to assign a job class to an offline service you must ensure that it has been created first. (See Figure 5.)

The DBMS_JOB facility, on the other hand, does not have any integration with services. Jobs submitted with this tool will always run under the SYS$USERS class and will not be assigned a module or action. This is one of the many reasons that the DBMS_SCHEDULER should be used whenever possible.

Fig. 5. Offline a Service to Suspend Jobs

```
connect / as sysdba
grant create job to scott;
alter system set
        service_names='myservices,testoffline';
alter system set service_names='myservices';
exec dbms_scheduler.create_job_class('TESTOFF1',
                    service=>'testoffline');
grant execute on testoff1 to scott;

connect scott/tiger
create or replace procedure traceme as
  x number;
begin
  dbms_session.session_trace_enable(true,true);
  select count(*) into x from scott.emp;
  dbms_session.session_trace_disable();
end;
/
exec dbms_scheduler.create_job('TESTJOB1',
        'PLSQL_BLOCK',job_action=>'traceme();',
        job_class=>'TESTOFF1',enabled=>true);
select job_name, schedule_type, job_class,
    enabled, auto_drop, state
    from user_scheduler_jobs;
```

You will see that the job is scheduled to run immediately but it's just sitting in the queue.

```
connect / as sysdba
alter system set
        service_names='myservices,testoffline';
```

You will immediately see a trace file for the job. Once a service comes online its jobs immediately start. You can investigate the trace file to see the service, module, and action under which the job was executed.

Finally, it is worth briefly mentioning parallel execution slaves. Oracle will automatically start a minimum number of parallel servers at database startup (controlled by the PARALLEL_MIN_SERVERS parameter) and can start more servers on demand up to the maximum PARALLEL_MAX_SERVERS. When a parallel query, DML, DDL, or recovery statement is executed its process becomes a query coordinator and hands off the work to several of these parallel servers. Part of the process includes passing along the service name. The parallel servers will inherit this service until they finish processing then revert back to the default service.

2.7 Shared Servers

Although most new installations today are using dedicated servers, many databases still use Multi-Threaded Server (MTS)[6] - especially high-end systems with very large numbers of concurrent users. Architects who plan to use shared servers have additional planning steps when determining their services strategy. As alluded to

6. The terms MTS and SHARED SERVERS are used interchangeably; Oracle first called the feature MTS and later renamed it to SHARED SERVERS.

Fig. 6. Simple MTS Configuration

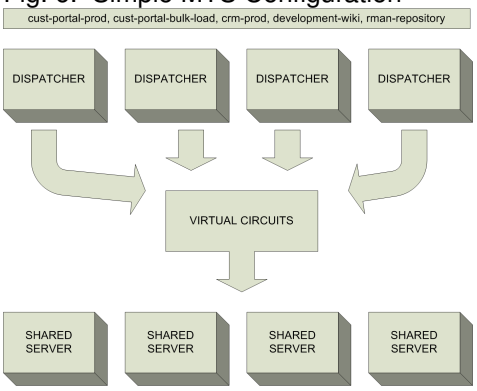
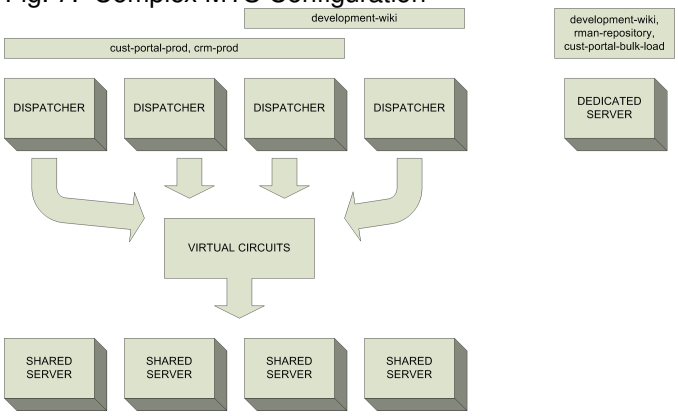


Fig. 7. Complex MTS Configuration



earlier, configuring services with Shared Servers involves more than just the SERVICE_NAMES parameter.

Actually you may not need to set more than just the SERVICE_NAMES parameter. By default Oracle will automatically use a simple MTS configuration where every service uses shared servers by default and supports dedicated server connections when requested. (See Figure 6.) This should be sufficient for most databases and requires no special configuration. However in some situations you may wish to tweak the configuration by defining exactly how many dispatchers are available to each service and whether or not dedicated servers are available. As illustrated in Figure 7, somewhat complex configurations are possible. In particular this may be useful if only a subset of the services on a system use Shared Servers or if you want to increase performance for specific services by reserving dispatchers exclusively for them.

Shared Server configuration is accomplished through the coordinated use of the SERVICE_NAMES and DISPATCHERS parameters. The DISPATCHERS parameter is a multi-valued parameter that allows you to define groups of dispatchers with specific characteristics including what services each group offers. If you do not explicitly define services for any group of dispatchers then by default that group will use the SERVICE_NAMES parameter. (This is why Oracle will default to the simple configuration illustrated above.) But if you explicitly define services for a group then that group will register each listed service with the data dictionary and with the listener. Dedicated connections will only be available for services listed in the SERVICE_NAMES parameter, so you need to list services in both places if you may need both connection types. In Figure 7 you can see that if you request a dedicated server from the CRM-PROD or CUST-PORTAL-PROD service then you will receive an error. Shared servers will always be the default connection type for a service if they are available; you need to explicitly specify dedicated connections when you want to use them. In the example, connections to the DEVELOPMENT-WIKI service will use shared server connections by default even though dedicated servers are available.

These initialization parameter settings will produce the example in Figure 7:

```
SERVICE_NAMES='development-wiki,rman-repository,
  cust-portal-bulk-load'

DISPATCHERS=( '(protocol=tcp)(dispatchers=2)
   (service=cust-portal-prod,crm-prod)',

  '(protocol=tcp)(dispatchers=1)
   (service=cust-portal-prod,crm-prod,
      development-wiki)',

  '(protocol=tcp)(dispatchers=1)
     (service=development-wiki)' )
```

For the few situations where the resource manager cannot create a satisfactory service performance profile, Oracle offers very granular control over the dispatchers in a shared server configuration. By carefully crafting your strategy for services in an MTS environment you can provide higher service levels and better response times to the users of your database.

2.8 Data Guard

Oracle Data Guard provides the ability to configure a standby database that mirrors production and which is ideal for meeting disaster recovery requirements. Data Guard will not have a significant impact on your services strategy but there are a few minor points to keep in mind when configuring Data Guard.

First of all, the DB_UNIQUE_NAME and DB_NAME parameters are not the same in 10g Data Guard configurations. Remember that the default service for your database will use the DB_UNIQUE_NAME and DB_DOMAIN. When you configure the archive log destinations and FAL parameters you must provide a connect descriptor either directly or as a TNSNAMES alias; you should specify this default service as the SERVICE_NAME.

However, your clients should never use the default service if you have configured a standby database! In fact if you have followed the suggested rules for complete service virtualization (see Table 4 on page 7) and you

are not using TAF then in the event of a failover or switchover you only need to update the DNS entry for your services and clients will immediately begin connecting to the new primary database. Alternatively, Transparent Application Failover can be used to automate failover; in fact it can be combined with Fast Start Failover to provide very small downtime windows in the event of a catastrophe [MSRK07]. Of course using service names is still required.

There is a surprisingly widespread misconception that services only really apply to cluster databases. But after looking more closely at them it is now clear that many of their features are very useful in all databases. Services should always be included in planning and configuration for databases and clients unless there is a very compelling reason not to use them.

3 Unleashing Services In RAC Databases

3.1 The Golden Rule

In Oracle 10g services provide the foundation for an extensive and comprehensive workload management framework and high availability solution. So far, we've discussed numerous workload-related aspects of services: performance views, resource prioritization, notifications, virtualization, background processes, and MTS configuration… however none of these provide high availability.

High availability is a broad topic which is presently receiving a lot of attention. It covers both planned and unplanned downtime and includes everything from weekly off-site tape backups to real-time dynamic reconfiguration of redundant hardware [Kum05]. Furthermore it has become a priority in IT budgets. HP recently found that approximately four out of five managers and executives responsible for business continuity and availability indicated that the area will see an increase in spending this year as compared to 2006 [hps07]. The cornerstone of Oracle's high availability offering is clustering[7]. Oracle clusterware and RAC databases add a layer of software intelligence that creates a fault-tolerant system from several non-fault-tolerant subsystems (i.e. servers). That is, the final single points of failure are eliminated through multiple commodity servers rather than custom-engineered redundant and independent subsystems. The RAC approach has advantages and disadvantages that should be carefully weighed before choosing it as a business solution.

Services are fundamental in RAC-based HA solutions and an integral part of the plumbing needed for database fault-tolerance. In fact they are the only feature whose configuration is spread across both the data dictionary and the clusterware registry. Yet despite the handful of differences between using services on RAC and non-RAC databases, most functionality is exactly the same.

7. For a high-level overview of HA options read the IT Convergence whitepaper *High Availability Options for Oracle Databases* [NCP+06].

In fact there is a golden rule about RAC: DO UNTO RAC AS YOU WOULD DO UNTO NON-RAC. Ok, of course that's not a blanket rule! But a RAC instance and a non-RAC instance both have a library cache, a shared pool, sort areas, redo buffers, temporary tablespaces and rollback segments. There are some important differences but every aspect of services we have discussed so far applies equally on RAC.

So what are the differences between using services on RAC and non-RAC systems? First, services must be added and removed differently in a RAC system. Second, there are a number of additional features that apply only to RAC systems such as clusterware integration, load balancing and failover.

3.2 A Horse of a Different Color? (RAC vs. non-RAC)

Starting your engines on RAC is a bit different from non-RAC databases. Although the SERVICE_NAMES instance parameter should always be set on non-RAC systems, it should never be set in RAC. The Oracle clusterware software (not the RDBMS software) manages services by setting this parameter automatically[8]. If you manually set it to create and enable a service then your service may be overwritten by the clusterware. However although you shouldn't use the SERVICE_NAMES parameter there are three new alternatives for creating services in 10g RAC: DBCA, OEM Database Control, and a new manual method[9].

DBCA in Oracle Database 10g contains numerous enhancements. In addition to creating RAC templates and databases it can now manage RAC instances, ASM instances, and database services. DBCA is much more than just a "Database Creation Assistant"; it is also a

8. For this reason the limit of about 20 concurrently active services remains the same in RAC.

9. A fourth alternative is OEM Grid Control, which offers a similar interface to OEM Database Control for creating services. OEM Grid Control, licensed independantly of the database, provides centralized management of systems running Oracle and non-Oracle technologies.

Fig. 8. Creating and Managing Services with DBCA

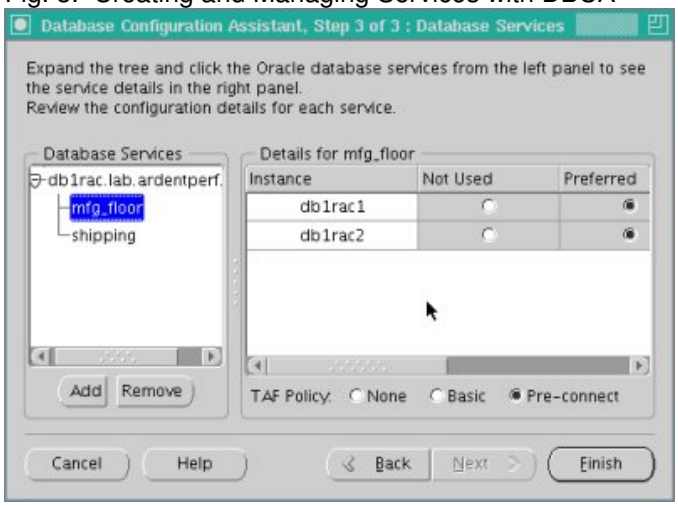
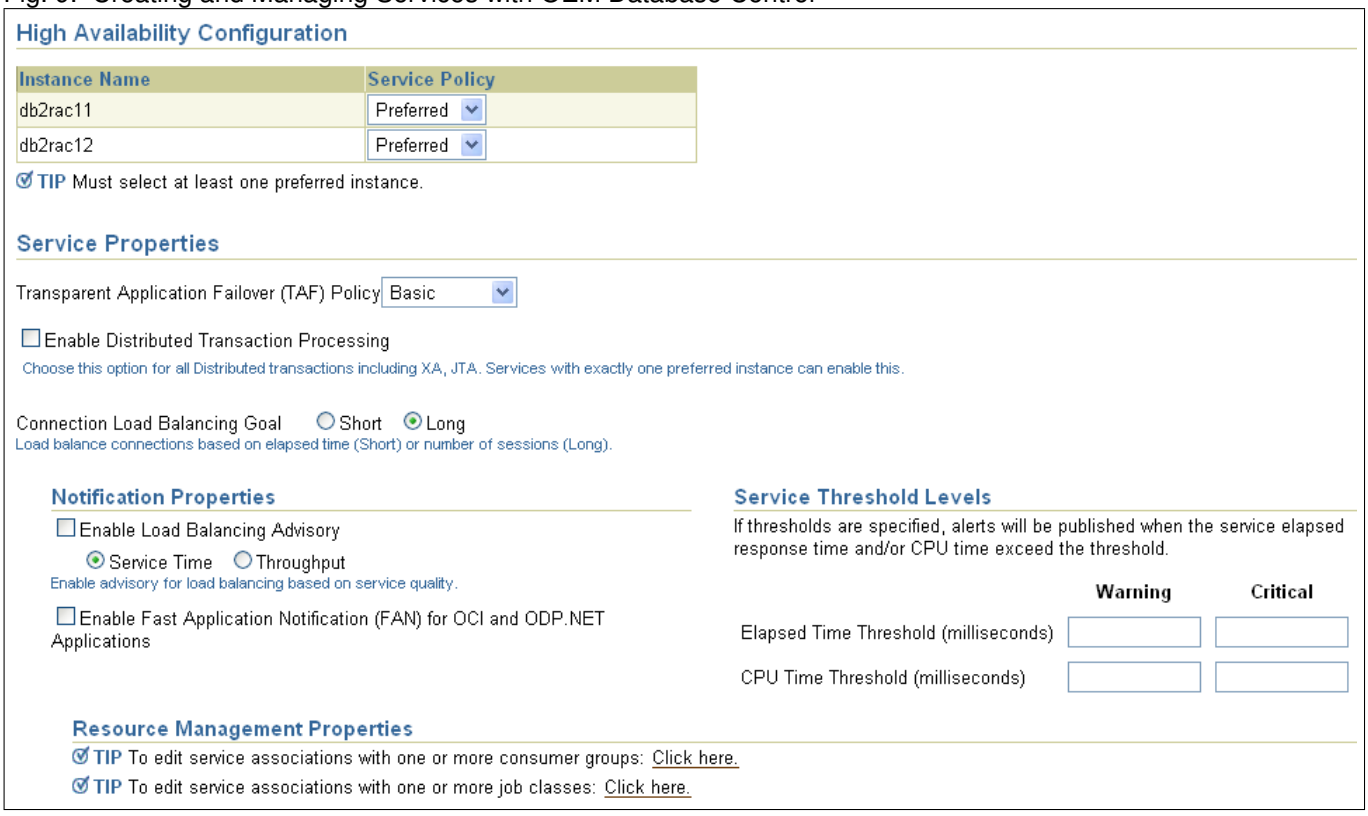


Fig. 9. Creating and Managing Services with OEM Database Control



tool for database reconfigurations such as adding new cluster nodes and modifying service failover settings. (Although sadly the word on the street is that DBCA's service management features will be removed in 11g.) The service management page in DBCA is available in two places.

1) During database creation one of the final steps (number 12 of 15) is specifying the service configuration for the database that you're creating.
2) The services management page is directly accessible from DBCA's main menu to reconfigure databases that already exist and are in use.

Additionally you will notice in Figure 8 that you can specify more than just a name for the service: there is a DETAILS panel with several new settings. (These DBCA options are all related to client-side load balancing and failover.)

OEM Database Control is the second tool capable of configuring services. Like DBCA it offers a simple point-and-click environment for creating and maintaining services; adding a new service is as simple as clicking the CREATE SERVICE button. You can also reconfigure services for databases that already exist and are in use. And you will again notice several new settings here: the SERVICE POLICY and TAF POLICY from DBCA and lots of additional properties that weren't available in DBCA. The service level thresholds, consumer groups, and job classes are GUI panels to set and update these properties of services from one convenient location. The notification properties, load balancing goal, and distributed transaction processing options are specific to RAC. OEM Database Control is a great place to configure most of your service properties; just be careful when configuring TAF since OEM behaves differently from DBCA when setting these properties. (More about this later.)

Fig. 10. Manually Creating RAC Services

```
[oracle@rh4lab15 ~]$ srvctl add service -d db2rac1 \
> -s reports -r db2rac11,db2rac12 -P basic


SQL> begin
  2    dbms_service.create_service(
  3      service_name=>'reports',
  4      network_name=>'reports',
  5      goal=>dbms_service.goal_none,
  6      dtp=>FALSE,
  7      aq_ha_notifications=>FALSE,
  8      clb_goal=>dbms_service.clb_goal_long
  9    );
 10  end;
 11  /


PL/SQL procedure successfully completed.


[oracle@rh4lab15 ~]$ srvctl start service -d db2rac1 \
> -s reports


[oracle@rh4lab15 ~]$ lsnrctl services
Service "reports.lab.ardentperf.com" has 2 instance(s)
  Instance "db2rac11", status READY, has 2 handler(s)
  Instance "db2rac12", status READY, has 1 handler(s)
```

Finally, Oracle still supports manual configuration of services in RAC databases but it is different from

manual configuration in non-RAC databases. Manual configuration in RAC involves two steps: creating the service in the clusterware and creating the service in the database. (And neither of these steps involves the SERVICE_NAMES initialization parameter.) The SRVCTL command-line utility is used to register the service in the clusterware and the DBMS_SERVICE package is used to configure it in the database. Internally, both DBCA and OEM Database Control use these utilities to create services and it is generally recommended to use the graphical utilities because they always get the syntax right. However it is fully supported to use these tools manually too. Furthermore if DBMS_SERVICE is not called then the service will be created automatically with default properties the first time it is started. However in RAC systems it is best to explicitly call CREATE_SERVICE so you can control the additional RAC-related properties.

All of the new RAC-specific service properties are configurable through these two utilities. The SERVICE POLICY (preferred and available instances) and TAF POLICY (for client-side TAF) are both available through options to the SRVCTL clusterware utility. The notification properties, load balancing goal, and distributed transaction processing options are available through the DBMS_SERVICE package. There is also an additional new RAC feature configurable manually which is not yet available through DBCA or OEM Database Control: server-side TAF configuration. This feature can only be enabled and configured through the DBMS_SERVICE package.

There are a number of additional initialization parameters that must be set properly in RAC systems and two of these in particular are worth mentioning: REMOTE_LISTENERS and LOCAL_LISTENER. As was already discussed, PMON registers each service with the locally running listener by reading the LOCAL_LISTENER parameter or looking on the default port 1521. However in RAC there is an additional step to this registration process: PMON will also register with every listener in the entire cluster as a REMOTE SERVICE - using the parameter REMOTE_LISTENERS to find all of the cluster listeners. This parameter is mandatory; it enables listeners to redirect client connections to other nodes in the cluster. But where do the clients get connect information for each node in the cluster since this information may not be in the TNSNAMES file? When the PMON process registers with remote listeners it will read its own LOCAL_LISTENER parameter and use this to construct a connect string. That connect string is then passed to clients with the redirect message. The connect string stored by the listener is visible from the command LSNRCTL SERVICES for troubleshooting.

3.3 The Ignition (Starting and Stopping)

You will recall that in non-RAC databases the SERVICE_NAMES initialization parameter is used not only to create services, but also to start and stop them. Services are started when added to this parameter and stopped when removed. However in RAC systems the clusterware manages this initialization parameter automatically and you should not modify it. Instead of using SERVICE_NAMES you should use either OEM Database Control or the clusterware command-line utility SRVCTL to manage your services[10]. (DBCA can create, modify and remove services but it cannot start or stop them.)

OEM Database Control is the easiest method and is recommended for those who prefer a GUI over the command line. It provides an intuitive, easy-to-use interface for starting and stopping services. SRVCTL is what OEM Database Control uses under the covers to start and stop services and also can be used directly. SRVCTL also offers a convenient option to start all available services for a database (you can't do this in a single step from 10g OEM Database Control).

Services do not always start automatically and this has caused confusion for some RAC administrators. By default in Oracle 10g the clusterware will always try to return each service to its previously known state. If a database server experiences an unexpected crash then the clusterware will automatically attempt to restart the database and services when the machine becomes available again. However if you shutdown the service or if you shutdown the database using SRVCTL then the service will not automatically restart when you restart the database[11]. This is not immediately obvious - so take note!

3.4 Under the Hood (RAC Service Configuration)

Configuration information for services in Oracle 10g is stored in no less than four places: the spfile, the data dictionary, the cluster registry (OCR), and the filesystem.

We have already reviewed Oracle's service-related initialization parameters; the most important were LOCAL_LISTENER and REMOTE_LISTENER which instruct PMON how to find and register with listeners. DB_DOMAIN will be appended to unqualified service names in both RAC and non-RAC environments. The DISPATCHERS parameter still needs to be set if you use MTS on a RAC system. And STATISTICS_LEVEL should be set to TYPICAL or ALL to enable the gathering of services-related statistics. Finally, it never hurts to emphasize that SERVICE_NAMES is not to be touched in RAC environments.

10. The DBMS_SERVICE package does provide the START_SERVICE and STOP_SERVICE methods to start and stop services but these should not be used in RAC. As of version 10.2.0.1 it seems that the Oracle manual is incorrect [Ora05c, page 95-14]; these methods do not call out to the clusterware but only set the SERVICE_NAMES parameter - resulting in inconsistencies between the database and clusterware.

11. This is controlled by a clusterware service property called TARGET which is usually "ONLINE". When you shutdown the database for maintainance using SRVCTL the TARGET becomes "OFFLINE" and the clusterware will no longer attempt to automatically start the service. Starting the service manually with SRVCTL sets the TARGET back to "ONLINE".

The data dictionary contains the bulk of services-related configuration. Most importantly the data dictionary contains all currently registered services and their attributes. This master list and most RAC-related attributes are exposed through the view DBA_SERVICES; including metadata (such as the name, various hashes and the service creation date), load balancing configuration, notification options, server-side failover, and the DTP flag. In addition to this view are all the views we have already mentioned: DBA_RSRC_GROUP_MAPPINGS for consumer group mappings, DBA_THRESHOLDS for service-related alert thresholds, and DBA_SCHEDULER_JOB_CLASSES for service mappings to job classes, to name a few.

The OCR only exists on RAC systems but is critical since on clusters services are managed by the clusterware. There are two levels of configuration data that are stored in the OCR: resources and stringpairs. Resources are low-level generic configuration data; they tells the clusterware things like which executable starts and stops them and what internal dependencies exist between them. If you're curious, it can be dumped with the command CRS_STAT -F. This configuration is managed transparently and completely by SRVCTL and you should not need to change it by hand. The stringpairs are one layer above resources and are less structured; like the windows registry Oracle uses them to store a variety of data types in a hierarchical structure [DS06, page 402]. Oracle's non-core-clusterware utilities store their data here (e.g. SRVCTL, RACGONS, OIFCFG, and VIPCA). The spuriously-named utility OCRDUMP in fact only dumps the stringpairs, not the entire OCR. For each service there is configuration at both of these levels. A clusterware resource is automatically created by the SRVCTL utility and the appropriate low-level properties are assigned. Then all of the service-specific properties are stored in stringpairs. These properties include each instance along with preferred/available status and enabled/disabled flag, the client-side TAF policy, some binary environment data, and a global enabled/disabled flag.

Finally, there is an important bit of configuration stored in the filesystem: the TNSNAMES.ORA file. This file contains the connection information for LOCAL_LISTENER and REMOTE_LISTENERS as well as the correct connect descriptors for connecting to the database with load balancing and client-side failover. Getting these connect descriptors right can be surprisingly elusive!

Oracle's utilities will automatically handle most service configuration for you. The SRVCTL utility will automatically manage the OCR. The DBMS_SERVICE package will manage all cluster-related settings in the data dictionary and additional packages and SQL statements exist to manage other service-related settings. The spfile is of course maintained by ALTER SYSTEM commands. The SRVCTL utility has a flag to generate a generic TNSNAMES.ORA entry for you although it's formatted terribly and missing the hostnames (not very useful).

DBCA and OEM Database Control both help automate these steps but there are a few differences between these two tools. DBCA will automatically maintain TNSNAMES.ORA for you and will call SRVCTL for you to create and start the service but it will not call DBMS_SERVICE. So at startup the service will be automatically created in the database with default properties for notification, load balancing goal, and distributed transaction processing and you will have to manually set these later using the DBMS_SERVICE package or OEM Database Control. On the other hand OEM Database Control will take care of SRVCTL and DBMS_SERVICE calls for you and will allow you to control all aspects of the service except server-side TAF. It also can manage a number of additional settings such as service level thresholds, consumer groups, and job classes. However OEM Database Control will not maintain the TNSNAMES.ORA file.

TABLE 5
DBCA vs. OEM Database Control

DBCA	OEM Database Control
Maintains TNSNAMES.ORA	Doesn't maintain TNSNAMES.ORA
Only supports OCR settings; doesn't call DBMS_SERVICE	Supports many settings; calls DBMS_SERVICE and other packages

The best solution in Oracle 10g is a combination of all three methods: use DBCA to create and delete services and then use OEM Database Control to configure additional settings as necessary. If you are setting up server-side TAF then you must manually call the DBMS_SERVICE package for that step. In Oracle 11g, OEM Database Control will likely become the central place to create and edit services.

3.5 Deep Platform Integration (Clusterware)

In Oracle10g the biggest RAC-related new feature was the inclusion of a complete clusterware product. In 9i RAC, Oracle provided cluster manager software only for Linux and Windows but on other platforms required third party clusterware on other platforms (such as IBM HACMP or Sun Cluster). That clusterware provided basic functionality such as maintaining the master list of node membership, tracking node up/down status, implementing I/O fencing and providing methods for high-speed inter-node communication. Oracle's Clusterware product that shipped with 10g is an independent, complete product providing these functions and more through a public API. This provided a tremendous advantage to Oracle's engineers. Knowing that the clusterware API would be the same across all platforms it became possible to integrate the database and clusterware more tightly than ever before. Oracle 10g utilizes more advanced, less standardized clusterware functionality like automatic resource monitoring and failover and it

allows the clusterware to manage everything from IP addresses to database startup and shutdown.

The advanced resource management capability of the clusterware is exactly what Oracle services leverage in a cluster environment. For example, the clusterware automatically monitors each service and has a configurable policy controlling the allowed number of failures and restarts. It can also start services at boot-time and is aware of dependencies such as which instances should be started to run a particular service.

When configuring services on RAC you are always presented with the selection of PREFERRED and AVAILABLE instances. This critical setting should be carefully planned because it will become the foundation of your strategy for availability and workload management. You absolutely should have a solid understanding of the terminology and implications of these settings before proceeding with RAC service configuration. There are some small differences depending on the TAF settings in the OCR but generally these terms have a consistent meaning: the service runs on all your PREFERRED instances during normal operation and AVAILABLE instances can be used when a preferred instance becomes unavailable for any reason (planned or unplanned). The single most important component of your services strategy will be determining the preferred instances for each service. Spreading a single service (or multiple services that share data) across multiple nodes will cause more inter-node communication and increase the load on the interconnect. Partitioning services across node lines will decrease processor and I/O utilization since spare resources on one node cannot be utilized by overloaded services on other nodes. Furthermore you must also factor in consumer group service mappings as you are planning your workload. While Oracle 10g RAC has become more automated and intelligent than ever before, it also offers more tools and options for tuning the workload than ever before. It still requires a good working knowledge of your application to choose the best RAC configuration and service strategy. An example preferred service layout was illustrated in Figure 1 on page 2. In addition to planning your strategy for PREFERRED instances you also need to carefully decide which instances will be AVAILABLE to each service. If you are using client-side preconnect TAF then make sure that you understand the implications of this for your available instances. Although client connections are not usually routed to these instances during normal operation it is important to carefully decide where the connections will go when a node fails. If there is a node failure then the last thing you want to do is complicate the problem with an overloaded active node!

It's worth mentioning one final note about clusterware integration with services: in 10g there is no automatic failback of services. If the failure of a preferred node causes a service to relocate to an available node then you have to be aware that after you bring the failed node back online the service will not automatically relocate back to it. If you need the functionality then it would not be difficult to automate this by hooking into Oracle clusterware's notification service but it would require custom scripting. It is trivial to manually failback the service. For example, the command in Figure 11 will relocate the service REPORTS from AVAILABLE node2 back to PREFERRED node1.

Fig. 11. Relocating a Service

```
srvctl relocate service -d db2rac1 -s reports
  -i db2rac12 -t db2rac11
```

3.6 Load Balancing

The practice of evenly distributing a load among multiple computers is a fundamental concept in cluster computing. In Oracle 10g load balancing is built on services and client work is always distributed among AVAILABLE instances of the service being connected to. (This is partly why the selection of preferred and available instances is so important!)

There are three kinds of load balancing in Oracle: client-side connection load balancing, server-side connection load balancing, and run-time load balancing. Client-side connection load balancing is the simplest and most straightforward. It is configured in the TNSNAMES file by simply including more than one ADDRESS, ADDRESS_LIST, or DESCRIPTION. Client-side CLB is controlled through the LOAD_BALANCE property which can be nested inside the DESCRIPTION_LIST, DESCRIPTION, or ADDRESS_LIST and defaults to TRUE. When load balancing is enabled the client will randomly choose one of the entries for each new connection. The connections will be distributed roughly the same across the addresses but they will not be exactly even. James Morle recently demonstrated how unevenly client-side CLB will distribute connections in his white paper about RAC Connection Management [Mor06]. In the vast majority of RAC configurations this is used hand-in-hand with server-side connection load balancing to spread initial connections across listeners on different nodes.

The second type of load balancing is server-side connection load balancing. It is enabled by default and is more sophisticated than client-side load balancing. When both are being used the server-side process is who determines where client connections ultimately go. Server-side load balancing is built into the listener; the listener knows the current load on each database instance and can reroute client connections to the most responsive instance or the instance with the smallest number of sessions. The configuration is stored in the data dictionary and is maintained with the DBMS_SERVICE package or OEM Database Control. The two most relevant properties are exposed in DBA_SERVICES: GOAL and CLB_GOAL. GOAL controls a new 10g process called the Load Balancing Advisory which pushes load information based on factors such as response time and total

session count to its subscribers. The listener subscribes to this service and receives notifications through FAN events sent by the clusterware. GOAL can be set to NONE (which disables it), THROUGHPUT (heavily factors in the rate at which work is completed), or SERVICE_TIME (heavily factors in the elapsed time for work). On the other hand CLB_GOAL is specifically for the listener and should reflect whether the incoming connections to this service will be SHORT (seconds/minutes) or LONG (like oracle forms sessions or connection pools). For long connections the listener will attempt to balance the total number of sessions on each instance and for short connections the CPU load or service response time is considered more heavily. Server-side connection load balancing is configured independently for each service on the system and should definitely be considered when planning your services strategy.

Third, Oracle supports run-time load balancing within connection pools. The connection pool can be configured with extra connections to all instances and when a thread requests a connection from the pool Oracle can automatically return a connection on the least loaded cluster instance. Like server-side load balancing it utilizes the Load Balancing Advisory; so the GOAL property must be set appropriately on the service. Furthermore only JDBC connection pools are able to subscribe directly to FAN events from the clusterware; ODP.NET connection pools use AQ notifications to receive load information. This is not enabled by default. To use run-time load balancing with ODP.NET connection pools you must set the property AQ_HA_NOTIFICATIONS on the service to YES. In 11g OCI connection pools will also likely be capable of run-time load balancing through AQ. Run-time load balancing offers the ability to dynamically rebalance work across the cluster at runtime for applications that are built with connection pools. If your application uses connection pools then this feature should definitely be considered while planning your services strategy.

Oracle's new dynamic load balancing capabilities are based on response-time statistics. It's worth pointing out in passing that these are gathered per-service and not per-instance. The advantage of this approach is that if factors like resource manager mappings cause some services to be more responsive on a node than others then the load balancer will still route work appropriately for that service's goal.

In general it is a best practice to use all three kinds of load balancing at the same time. (If you aren't using connection pools then use the client and server-side load balancing.) This will offer the best workload distribution across the cluster.

3.7 Failover

One of the defining characteristics of enterprise database systems is how they handle failures. Since a primary goal of Oracle Services is high availability we would expect them to respond robustly when something goes wrong. And indeed they do albeit with some important limitations.

There are three levels on which failover can occur in RAC: at the service level, at the connection level, and at runtime. The first two are fairly straightforward but the third is somewhat more complex. We have already discussed service failover: during normal operation a service runs only on its PREFERRED nodes but if the clusterware detects the failure of an instance then it will automatically relocate that service to an AVAILABLE node (if there are any). Clusterware will not change the status of that node to PREFERRED however the service will remain there until it is restarted or until you manually relocate it to the original node. (It would be possible to automate this with a script.) This service relocation is independent of any existing connections to the database; relocating a service will not disconnect the old node's existing connections (unless you explicitly pass the -F flag to SRVCTL when you relocate the service).

The second failover level is connection failover. Like client-side load balancing it is easily configured in the TNSNAMES file by simply including more than one ADDRESS, ADDRESS_LIST, or DESCRIPTION. Connection failover is controlled through the FAILOVER property which - like the LOAD_BALANCE property - can be nested inside the DESCRIPTION_LIST, DESCRIPTION, or ADDRESS_LIST and defaults to TRUE. Connection failover causes the client to automatically retry the connection with a different address when an address fails. The client will only fail to connect if all addresses fail. This is used almost universally in RAC deployments so that the connection will still succeed as long as one listener is available.

Lastly Oracle provides run-time failover: under certain conditions the client can detect a node failure and automatically move existing connections to a surviving node with no interruption to the application. However there are significant limitations; most importantly there is absolutely no support for transactions. Furthermore Oracle will not preserve the PL/SQL environment or session state; it simply establishes a completely new session and transparently gives the client that new session. It is possible to work around these limits, but not without custom coded application handlers for connection failures. If your application is read-only or if it can tolerate a few errors or if you're willing to write the custom code to handle failovers then run-time failover could be a very powerful feature for you.

There are three different ways to configure run-time failover: client-side TAF, server-side TAF and Fast Connection Failover (FCF)[12]. Client-side TAF is the oldest of the three, having been around since the Oracle 8 timeframe. In this configuration the TAF failover settings are stored in each client's TNSNAMES file using a FAILOVER_MODE block nested in the CON-

12. For a more detailed discussion of TAF configuration strategies check out the short article *Centralized TAF Configuration in 10g* [Sch07].

NECT_DATA section. Server-side TAF is a new feature in 10g and allows you to specify the TAF failover settings as properties of each service with the DBMS_SERVICE package. If any server-side properties are specified for a service then these settings will override anything in the client's TNSNAMES files.

There are two kinds of transparent application failover: BASIC and PRECONNECT. The PRECONNECT method is only available with client-side TAF. BASIC failover creates a new connection at failure-time whereas PRECONNECT establishes a backup session at connect-time for faster failover when a failure is detected. The backup session is made with a different TNSNAMES entry, specified by the BACKUP property in the FAILOVER_MODE block. In general it is a best practice to use BASIC failover because PRECONNECT does not significantly improve failover time. The additional complexity and performance is not usually worth the additional resources it consumes. In addition to the database, the clusterware also keeps track of each service's TAF configuration. If the clusterware knows that a service is using PRECONNECT failover then it will always start up an auxiliary service on all AVAILABLE instances called [SERVICENAME]_PRECONNNECT which can be used for these backup sessions. DBCA will automatically maintain both the OCR and all TNSNAMES entries appropriately, which is largely why it's recommended to use DBCA for service configuration in Oracle 10g.

Failover configuration can also specify SESSION or SELECT failover to determine how Oracle responds if a SQL statement is in the middle of execution during a node failure. With SESSION failover the statement will always return an error when the node fails. However, SELECT failover tells Oracle to prevent errors for non-transactional select statements by transparently reexecuting them on the new connection after failure. This does require a small amount of client memory to store some state information for select statements such as the SCN, SQL plan hash, and fetch count but usually is worth the trade-off even though it won't benefit code that modifies data (which admittedly is most code).

Finally, it is possible to specify RETRIES and DELAY properties to tweak the timing of BASIC failover after a failure is detected. All of these properties can be specified client-side in the TNSNAMES file or server-side with the DBMS_SERVICE package. The server-side settings are visible for each service in the DBA_SERVICES view and you can also always see the settings for your currently connection in the V$SESSION view.

Starting in Oracle 10g, there is also a third run-time failover configuration called Fast Connection Failover. Like run-time load balancing, FCF is a feature for JDBC connection pools that works by subscribing to FAN events. When a node goes down or comes up, the connection pool will automatically rebalance its connections among all available nodes. In practice there are some benefits but they're not earth-shattering. James Morle gives a good concise summary in his recent paper on connection management:

> FCF gives a small improvement in failover time compared to a VIP, and some handling for node UP events. It still requires logic in the body of the application (outside of the connection pool) to handle connection failover. [Mor06]

3.8 Notifications, Distributed DB and Parallel Execution

We have alluded to something called FAN EVENTS several times without explaining what they are. Fast Application Notification (FAN) is a new feature of Oracle 10g RAC that notifies other local and remote processes about cluster-related events such as service load information and service UP/DOWN events. The events are pushed to subscribers by a server process called Oracle Notification Service (ONS) which is automatically started and managed by the clusterware in a default RAC installation. Oracle 10g listeners, connection managers and JDBC clients are already integrated with FAN. ODP.NET clients can also receive some events through Oracle Streams Advanced Queuing although they do not currently receive UP events and thus cannot redistribute connections when nodes come back online.

FAN was designed from day one to be extensible. We've already mentioned that the Load Balancing Advisory utilizes FAN events to publish service load information. It is also possible to hook into this infrastructure with your own applications and receive cluster-related events. There are three ways to tap into FAN events:

1) At the lowest level, you can receive events on the server directly from the clusterware by placing a custom script in $ORA_CRS_HOME/RACG/USRCO which will then be called directly as events are published. Oracle has made a number of excellent sample scripts available on their website[13].
2) Your program can register locally or remotely using the ONS API.
3) You can subscribe to an Advanced Queue for cluster events. If you use this method then you need to make sure that AQ_HA_NOTIFICATIONS is set to YES for the service you're monitoring.

This is arguably one of the most powerful and least known availability-related features of RAC Services: a complete framework for extending the system with custom handlers for configuration and load changes.

Oracle's distributed database engine is one of the most versatile and powerful on the market, offering both in-database SQL-based distributed transactions and integration with external TPMs. However there are a few important limitations when using distributed transaction processing capabilities with RAC database services.

13. Available at {http://www.oracle.com/technology/sample_code/products/rac/index.html}

Most importantly are limitations around multithreaded applications using DTP/XA. In these applications there can be multiple threads operating on the same global transaction. These threads can be tightly coupled or loosely coupled [X/O91]. For tightly coupled transactions Oracle acquires DX locks on all data, allows threads to see changes from other threads in the same global transaction and takes advantage of a one-phase commit optimization; whereas for loosely coupled transactions Oracle achieves greater concurrency by not acquiring DX locks but always requires the full two-phase commit. Oracle 10g allows loosely coupled threads of a single global transaction to be processed on different RAC cluster nodes but tightly coupled threads always must be processed on the same node [Ora05a, chapter 15]. (In Oracle 11g this restriction may be lifted.) In order to guarantee this you should configure services for distributed transactions. This is done for each service through either OEM Grid Control or the DBMS_SERVICES package and is no more complicated than setting the DTP flag to YES. Oracle also advises using DTP-enabled services for SQL-based distributed transactions. Distributed transactions are not a trivial technology to implement, but RAC offers major advantages for organizations that make the investment to understand the architecture and limitations.

Finally it is worth briefly mentioning parallel execution with RAC services. Oracle RAC is capable of automatically distributing parallel workers across multiple nodes of the cluster but in Oracle 10g this is not tied to the service definition. In other words, you may have configured a service to only run on two of four nodes by making those nodes PREFERRED - but as soon as you kick off a parallel operation the slaves can still run across all four nodes. There is no built-in connection between service definitions and parallel execution; parallel execution is still controlled by the initialization parameters INSTANCE_GROUPS and PARALLEL_INSTANCE_GROUPS. That being said, it is in fact possible to limit parallel execution for each service to its PREFERRED instances. Define an instance group for each service then set the PARALLEL_INSTANCE_GROUPS parameter at the session level to the group corresponding to its service either through a login trigger or in the application code. In Oracle 11g it appears that this may be fixed and parallel slaves may automatically map to services.

4 Conclusion

4.1 Services in Oracle Databases

After reviewing the many aspects of services it is apparent that this feature has developed significantly over the past ten years. It's easy to see why services are so important to all modern Oracle databases - and especially to cluster database systems. They add a completely new level of workload and performance management and they provide the foundation for RAC's most important availability features. Naturally there are still a few weaknesses, bugs, and features that just aren't very useful in real-world deployments but the extensive benefits of services far outweigh the small learning curve. The bottom line is that understanding services has become a basic, core responsibility of anyone who plans or maintains Oracle database systems today.

References

[DS06] J. Dyke and S. Shaw, *Pro Oracle Database 10g RAC on Linux: Installation, Administration, and Performance*. Berkeley, California: Apress, August 2006. [Online]. Available: {http://amazon.com/o/ASIN/1590595246/}

[FKT01] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, p. 200, 2001. [Online]. Available: {http://www.globus.org/alliance/publications/papers/anatomy.pdf}

[Fos02] I. Foster, "What is the grid? a three point checklist," *Grid Today*, vol. 1, no. 6, 2002. [Online]. Available: {http://www.gridtoday.com/02/0722/100136.html}

[Gen02] W. Gentzsch, "Response to ian foster's "what is the grid?"," *Grid Today*, vol. 1, no. 8, 2002. [Online]. Available: {http://www.gridtoday.com/02/0805/100191.html}

[Hot07] "Instrumentation library for oracle," Hotsos, 2007. [Online]. Available: {https://portal.hotsos.com/products/hotsos_ilo}

[hps07] "Hp survey: Business continuity and availability solutions a high priority for corporate spending in 2007," *Enterprise Networks and Servers*, April 2007. [Online]. Available: {http://www.enterprisenetworksandservers.com/monthly/art.php?3061}

[Kum05] S. Kumar, "Oracle database 10g release 2 high availability," Oracle Corporation, Redwood Shores, California, White Paper, May 2005. [Online]. Available: {http://www.oracle.com/technology/deploy/availability/pdf/TWP_HA_10gR2_HA_Overview.pdf}

[Moc87] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, IETF, November 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[Mor06] J. Morle, "Connection management in an oracle rac configuration," Scale Abilities Ltd, Cirencester, Glos, White Paper, October 2006. [Online]. Available: {http://www.scaleabilities.co.uk/}

[MSRK07] J. Meeks, M. T. Smith, A. Ray, and S. Kyathappala, "Oracle data guard 10g release 2 fast-start failover best practices," Oracle Corporation, Redwood Shores, California, White Paper, January 2007. [Online]. Available: {http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_FastStartFailoverBestPractices.pdf}

[NCP+06] D. Norris, D. Cutrone, A. G. Paez, D. D. Morales, and P. O'Sullivan, "High availability options for oracle database," IT Convergence, White Paper, 2006. [Online]. Available: {http://www.itconvergence.com}

[Nor06] D. Norris, "Rac for beginners: The basics," IT Convergence, White Paper, 2006. [Online]. Available: {http://www.itconvergence.com}

[Ora99] *Getting to Know Oracle8i, Release 2 (8.1.6)*, Oracle Corporation, Redwood Shores, California, December 1999. [Online]. Available: {http://download-west.oracle.com/docs/cd/A87860_01/doc/server.817/a76962/toc.htm}

[Ora05a] *Oracle Database Application Developer's Guide - Fundamentals, 10g Release 2 (10.2)*, Oracle Corporation, Redwood Shores, California, November 2005. [Online]. Available: {http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14251/toc.htm}

[Ora05b] *Oracle Database Net Services Administrator's Guide, 10g Release 2 (10.2)*, Oracle Corporation, Redwood Shores, California, October 2005. [Online]. Available: {http://download-west.oracle.com/docs/cd/B19306_01/network.102/b14212/toc.htm}

[Ora05c] *Oracle Database PL/SQL Packages and Types Reference, 10g Release 2 (10.2)*, Oracle Corporation, Redwood Shores, California, June 2005. [Online]. Available: {http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14261/toc.htm}

[Ora06a] *Oracle Database Administrator's Guide, 10g Release 2 (10.2)*, Oracle Corporation, Redwood Shores, California, May 2006. [Online]. Available: {http://download-west.oracle.com/docs/cd/B19306_01/server.102/b14231/toc.htm}

[Ora06b] *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide, 10g Release 2 (10.2) for Linux*, Oracle Corporation, Redwood Shores, California, September 2006. [Online]. Available: {http://download-east.oracle.com/docs/cd/B19306_01/install.102/b14203/toc.htm}

[Sch07] J. Schneider, "Centralized taf configuration in 10g," *Ardent Performance Computing*, February 2007. [Online]. Available: {http://www.ardentperf.com/2007/02/22/centralized-taf-configuration-in-10g/}

[Smi03] G. Smith, "Oracle database 10g services," Oracle Corporation, Redwood Shores, California, White Paper, November 2003. [Online]. Available: {http://www.oracle.com/technology/products/database/clustering/pdf/twp_rac_services_10gr1_112503.pdf}

[Sto07] H. Stockinger, "Defining the grid: A snapshot on the current view," *The Journal of Supercomputing*, March 2007. [Online]. Available: {http://hst.home.cern.ch/hst/publications/DefiningTheGrid-JSC2007.pdf}

[X/O91] *Distributed Transaction Processing: The XA Specification*, X/Open Company Ltd., Reading, Berkshire, December 1991. [Online]. Available: {http://www.opengroup.org/onlinepubs/009680699/toc.pdf}

Jeremy Paul Schneider is a Senior Consultant at IT Convergence, a global enterprise application services provider. His interests include applying grid and cluster technology to computationally intensive problems and he spends a lot of time working with Oracle RAC databases. Jeremy has been involved with the computer science field in academic, professional, and volunteer capacities and has worked with small and large organizations in the healthcare, manufacturing, energy, government, education, and nonprofit sectors. He is an enthusiastic advocate of and contributer to open source projects. As a passionate technologist and lifelong learner he also maintains the Oracle technology blog *Ardent Performance Computing*.