Oracle Net Services:
Performance, Scalability, HA and Security Best Practices

Kant C Patel
Director of Development
Oracle Net

Kuassi Mensah
Group Product Manager
Oracle Net and JDBC

Viswanath Ravishankhar
Technical Product Manager
Intel Corporation
Program

• Overview of Oracle Net
• Why Optimize Oracle Net?
• 11g New Features
• Best Practices
  – Operating System and Network
  – Database Client
  – Net Listener
  – Database Server
• Intel Case Study
• Q/A
Oracle Net Overview

- Primary Communication Foundation for DB
- Formerly known as SQL*Net
- Oracle’s Family of Networking Features:
  - Oracle Net
  - Oracle Net Listener
  - Connection Manager
  - Configuration Tools
    - Net Manager
    - NetCA
Why Optimize Oracle Net?

• System Performance
  – Increase Network bandwidth utilization
  – Lower database CPU utilization
• High Availability
  – Better respond to database/host/network failures
• Network Scalability
  – Scale better with more client connections
  – Load-balance to improve application experience
• Network Manageability
  – Simplify deployment and configuration
• Network Security
  – Protect and recover from Denial of Service attacks
Net Configuration Files

- sqlnet.ora
  - Main Oracle Net configuration file
  - On both Client and Server
- listener.ora
  - Configuration for the Net Listener
  - On Server only
- tnsnames.ora
  - Contains Connect Name to Descriptor mappings
  - Used by the TNSNames Naming adapter
  - On both Client and Server
- ldap.ora
  - Contains LDAP configuration information
  - Used the LDAP Naming adapter
  - On both Client and Server
Oracle Net 11g New Features

• Performance & Scalability
  – Support for large SDU (11.2.0.2)
  – Optimized networking stacks for various data transfer needs
    • Network Fast Path for SQL operations
    • Zero Copy I/O Path for bulk data transfers
  – Support for Database Resident Connection Pools
  – Support for Scalable Operating System Event Models

• High Availability & Manageability
  – IP address list traversal for each hostname during connect (11.2.0.1)
  – Efficient dead-node detection for failover (11.2.0.1)
  – Option to enable connection retries (11.2.0.1)
  – Easy Connect Naming enhancements
  – Integration with Automatic Diagnostic Repository
  – Option for Default Service in Listener
Oracle Net 11g New Features

• Network Security
  – CIDR and wildcard support for valid node checking (11.2.0.1)
  – Authenticated LDAP name lookup - OID and Active Directory
  – Protocol level access control for Listener administration

• IPv6 (11.2.0.1)
  – Support for all features and components in single-instance mode
  – Support for single listener address across all IP(v4/v6) interfaces

<table>
<thead>
<tr>
<th></th>
<th>IPv4-only Server</th>
<th>Dual-stack Server</th>
<th>IPv6-only Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4-only Client</td>
<td>Supported (v4)</td>
<td>Supported (v4)</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Dual-stack Client</td>
<td>Supported (v4)</td>
<td>Supported (v4,v6)</td>
<td>Supported (v6)</td>
</tr>
<tr>
<td>IPv6-only Client</td>
<td>Not Supported</td>
<td>Supported (v6)</td>
<td>Supported (v6)</td>
</tr>
</tbody>
</table>
Operating System Tuning
Why is OS tuning critical?

• Key role in data transmission!
• Some default OS configurations cannot handle modern Ethernet speeds
• Bandwidth x Delay Product (BDP)
  – Amount of data on the “wire” at any given point in time
  – Default OS buffers not large enough to handle this data
  – For example, with 40 Mbits/sec bandwidth, 25 msec delay,
    \[ BDP = \left(40 \times 1000 \div 8 \text{ Kbytes/sec}\right) \times (0.025 \text{ sec}) \sim 128 \text{ Kbytes} \]
• TCP a benevolent algorithm – one size fits all
  – Slow-start
  – Exponential back-off
  – Small Window Sizes
  – TCP performance features may not be enabled by default
TCP Optimization - Linux

- Use TCP auto-tuning in kernel (2.4.27, 2.6.7)
  - /proc/sys/net/ipv4/tcp_moderate_rcvbuf (1=on)
- Tune TCP Max Memory
  - /proc/sys/net/ipv4/tcp_rmem and tcp_wmem
    - 4096  87380  174760  ➔ Tune this to 2xBDP
- Tune the socket buffer sizes
  - /proc/sys/net/core/rmem_max and wmem_max
    - Set this to 2xBDP
- Ensure that TCP Performance features are enabled
  - /proc/sys/net/ipv4/tcp_sack
  - /proc/sys/net/ipv4/tcp_window_scaling
  - /proc/sys/net/ipv4/tcp_timestamps
TCP Optimization - Windows

• Vista / Server 2008 supports TCP auto-tuning
• For other versions, tuning necessary under RegKey
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
  – Turn on Window Scaling and Timestamps
    Tcp1323Opts = 3
  – Set TCP Window Size to 2xBDP
    GlobalMaxTcpWindowSize = <2xBDP>
  – If desired, tune Window Size at the Interface Level
    Tcpip\Parameters\Interfaces\<interfaceGUID>\TcpWindowSize
Don’t forget the Hardware

- Use Jumbo Frames for GigE networks
- Use NICs with TCP off-loading capabilities
- Monitor switches and OS for packet loss
  - Causes numerous issues
Tuning the Socket Buffers

• Set send and receive socket buffer sizes in tnsnames.ora or sqlnet.ora using:
  – SEND_BUF_SIZE – OS send buffer size
  – RECV_BUF_SIZE – OS receive buffer size
• Set this size to accommodate the BDP (2x)
• Also set on the server
• Large buffer sizes help
  – Application queue more data to the OS
  – Have more data on the wire
  – Better utilize available bandwidth
  – In WAN deployments
Tuning the Session Data Unit

- Controls SQL*Net packet size
  - 11g default 8k, Pre-11g default 2k
  - Max is 64k

- Set in
  - sqlnet.ora: DEFAULT_SDU_SIZE
  - tnsnames.ora: SDU in address

- Larger SDU gives
  - Better Network throughput
  - Fewer system calls to send and receive data
  - Less CPU usage – system and user

- Side-effect of larger SDU: Network buffers take up more memory
SDU Recommendations

- Optimal SDU varies with application
- Increase SDU on both client and server
  - SDU for a connection negotiated down to the lower of the two peers
- Increase SDU to 8k
  - Good default value for most users
- For bulk data transfer scenarios, increase to 64k
  - LOB transfers
  - XML DB
- Do not set to MTU value
  - SDU and MTU are independent
Database Client
Manageability
Introduction to Net Naming

• Descriptors can be mapped from a Connect Name
  sales = ⇨ Connect Name
  (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=sales))) ⇨ Connect Descriptor

• Naming Adapters map Name to Descriptor:
  – Local file: tnsnames.ora
  – Hostname based: Easy Connect
Easy Connect

- Simple, easy to use connect syntax for TCP/IP
  
  ```plaintext
  [//]host[:port][/service_name][:server][/instance_name]
  ```

  Example: `sqlplus scott/tiger@sales-server/sales`

- Useful when no connect descriptor customization is necessary
- No need for any client side configuration files

  `sales-server/sales`

  is equivalent to

  ```sql
  (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=sales)))
  ```
Oracle Net 11g and Easy Connect

- Support for IPv6 hostnames and addresses
- Use URL syntax to specify IPv6 addresses

[2001:fe80:12]:1522/sales.us.example.com:dedicated/inst1

is equivalent to

(DESCRIPTION=
 (ADDRESS=(PROTOCOL=tcp)(HOST=2001:fe80:12)(PORT=1522))
 (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)
 (INSTANCE_NAME=inst1)
 (SERVER=dedicated)))
Naming Recommendations

• Use Easy Connect whenever possible

• If Descriptors do not change often, use tnsnames.ora
  – Best for small deployments
  – TNS_ADMIN can be on a shared file system

• If Descriptors change often or for large deployments, use a directory
  – Oracle Internet Directory
  – Active Directory on Windows
  – Enable authenticated binds if needed
Database Client
High-Availability
Connection Establishment Timeouts

• Detect dead hosts faster

• Timeout for TCP connection establishment
  – TCP.CONNECT_TIMEOUT – 11g feature
  – Enabled by default (60 seconds) in 11gR2

• Timeout for connection to a DB server process
  – SQLNET.OUTBOUND_CONNECT_TIMEOUT – 10gR2 feature
  – Set if session establishment takes a long time

• Configurable at connect string level

• Can be used individually or at the same time
  – Outbound Connect Timeout must be greater than TCP Timeout

• Option to enable connection retries
Address and Description Lists

• Use client side load-balancing when using RAC
  
  (DESCRIPTION= (ADDRESS_LIST=
      (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-1)(PORT=1521))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-2)(PORT=1521)))
  
  – Address to use picked at random

• Use Failover when using Dataguard
  
  (DESCRIPTION_LIST =
      (LOAD_BALANCE=off)(FAILOVER=on)
      (DESCRIPTION = ...)
      (DESCRIPTION = ...))

• Usage not limited to RAC and Dataguard
RAC + Data Guard Example

sales-scan=(sales-1,sales-2)  backup-scan=(backup-1,backup-2)
The Optimal Connect Descriptor would be

```sql
(DESCRIPTION_LIST =
    (LOAD_BALANCE=off) (FAILOVER=on)

(DESCRIPTION =
    (LOAD_BALANCE=on)
    (ADDRESS=(PROTOCOL=tcp) (HOST=sales-scan) (PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=sales.example.com)))

(DESCRIPTION =
    (LOAD_BALANCE=on)
    (ADDRESS=(PROTOCOL=tcp) (HOST=backup-scan) (PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=sales.example.com))))
```
The Connect Descriptor internally expands to

\[
\text{(DESCRIPTION\_LIST} = \\
(\text{LOAD\_BALANCE}=\text{off}) (\text{FAILOVER}=\text{on})
\]

\[
\text{(DESCRIPTION} = \\
(\text{ADDRESS\_LIST}= \\
(\text{LOAD\_BALANCE}=\text{on}) \\
(\text{ADDRESS}=(\text{PROTOCOL}=\text{tcp})(\text{HOST}=\text{sales-1})(\text{PORT}=1521)) \\
(\text{ADDRESS}=(\text{PROTOCOL}=\text{tcp})(\text{HOST}=\text{sales-2})(\text{PORT}=1521)) \\
(\text{CONNECT\_DATA}=(\text{SERVICE\_NAME}=\text{sales.example.com}))
\]

\[
\text{(DESCRIPTION} = \\
(\text{ADDRESS\_LIST}= \\
(\text{LOAD\_BALANCE}=\text{on}) \\
(\text{ADDRESS}=(\text{PROTOCOL}=\text{tcp})(\text{HOST}=\text{backup-1})(\text{PORT}=1521)) \\
(\text{ADDRESS}=(\text{PROTOCOL}=\text{tcp})(\text{HOST}=\text{backup-2})(\text{PORT}=1521)) \\
(\text{CONNECT\_DATA}=(\text{SERVICE\_NAME}=\text{sales.example.com})))
\]
Fail-over for Connected Sessions

- Established client connections could hang when
  - Database host crashes
  - Remote Networks fail

- Detection of such failures could take a while
  - TCP behavior - timeouts in minutes
  - Depends on what the client does

- To catch such failures
  - Set a Receive Timeout
    - If your application is active and does not use long-running queries
  - Use Fast Application Notification (FAN)
Thin-JDBC Tuning

- SDU passed through the connect string
  "jdbc:oracle:thin:@(DESCRIPTION=(SDU=...))"

- Connect Timeout set through property
  `oracle.net.CONNECT_TIMEOUT`

- Read Timeout set through
  `oracle.net.READ_TIMEOUT`
  - Note: Do not use as a query-timeout.

- For Query Timeout, use
  `Statement.cancel` or
  `Statement.setQueryTimeout`
Net Listener
Scalability. HA. Security.
What is the Net Listener?

- First process that clients talk to
- Brokers client requests, handing them off to service handlers
  - Dispatchers
  - Dedicated servers
  - Connection Broker – DRCP (11g)
- Receives load updates from the database
- Does server side load-balancing across instances in RAC
- Does server side failover across nodes in RAC
- Can listen on multiple end-points or protocol addresses
- Also supports other presentations – HTTP, FTP
Database Registration with Listener

• Use Dynamic Registration
  – PMON updates the listener about
    • Offered services and available service handlers
    • Load statistics – frequently updated
  – To configure, set in init.ora
    • LOCAL_LISTENER: Address of listeners on local host
    • REMOTE_LISTENER: Address of listeners on remote hosts
  – By default
    • PMON connects to listener on port 1521
    • Automatically setup with RAC

• Remove static SID_LIST configuration in listener.ora
  – Keep only if you want to remotely start the database
Server-side Load Balancing

- Change behavior by setting Connection Load Balancing Goal
  - Long – for applications with long-lived connections (default)
  - Short – for applications with short-lived connections
Listener Logon Storm Handler

• Logon storm
  – Sudden spike in incoming connection rate
    • Normal – middle-tier reboot
    • Abnormal – DoS attack
  – Storms cause CPU starvation for existing sessions

• Enable the Connection Rate Limiter feature
  – Configure in LISTENER.ORA
  – Provides end-point level control of throttling
    LISTENER=(ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp) (HOST=sales) (PORT=1521) (RATE_LIMIT=3))
      (ADDRESS=(PROTOCOL=tcp) (HOST=lmgmt) (PORT=1522) (RATE_LIMIT=no)))
  – Set the Rate Limit to a value that matches your machine capabilities
Logon Storm Comparison

- 150 concurrent connections

RATE_LIMIT = no  
RATE_LIMIT = 3/sec
Other Best Practices

• Increase the maximum concurrent requests per end-point
  – QUEUESIZE parameter in listener.ora
  – Set to your expected Connection Request rate
  – Definitely set on Windows

• Do not set a listener password
  – Listener administration secure by default – OS User Authentication

• Optimize Environment variables for the oracle account
  – Longer the PATH, longer it takes to fork off the Oracle process
    • Ensure that PATH is small
    • Does not include any network shares
  – Cut down the number of environment variables
Database Server

Scalability
Oracle Server Architecture Overview

• Choosing the right server architecture is critical to meeting scalability requirements

• Oracle Database Server supports three architectures
  – Dedicated Server (default)
  – Shared Server aka MTS
  – Database Resident Connection Pool (11g)
Dedicated Servers

• Each client connection has its own process (thread on Windows)
• Dedicated process ensures lower latencies
• Have to start a new process on connect
• Have to tear down a process on disconnect
• Scalability limits
  – Memory
  – Number of Processes
Shared Servers (aka MTS)

• Each server handles multiple clients
• Dispatchers relay requests and responses between clients and servers
• Idle connections will not consume much memory
• Good for large number of connections with many idle
• Latency increase due to man-in-the-middle
Database Resident Connection Pool (11g)

- Pooled dedicated servers shared across client systems and processes
- Low connect/disconnect costs
  - Server “locked” on connect
  - Server “released” on disconnect
- Low-latency performance of dedicated servers
- Extreme scalability with a DRCP-capable client driver
Dedicated vs. Shared vs. DRCP

• Use dedicated for:
  – High-performance connections
  – Active, long-running, data transfer intensive operations

• Use shared for:
  – Sessions that may be idle for some time
  – Clients that frequently connect and disconnect

• Use DRCP (11g):
  – When you have thousands of clients which need access to a database server session for a short period of time
  – Applications mostly use same database credentials, and have identical session settings
  – PHP (OCI8 extension), Python (cx_Oracle), Perl (DBI)
Using Shared Servers

- Enable shared servers with init.ora parameters
  - Becomes new default
- To force server type, specify server type during connect
  - Dedicated:
    sales-server/sales.us.example.com:dedicated
  - Shared:
    sales-server/sales.us.example.com:shared
- Rough guidelines:
  - 20 or 30 Shared Servers per 500 sessions, then tune from there
  - 1 dispatcher for every 50-100 sessions
- Significant performance improvements in 11g
Using DRCP

- Pooling is enabled by the DBA using
  
  ```sql
  EXECUTE DBMS_CONNECTION_POOL.START_POOL
  ('SYS_DEFAULT_CONNECTION_POOL');
  ```

- Change connect string on client in tnsnames.ora:
  
  ```sql
  (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=sales)(SERVER=pooled)))
  ```

- Can use Easy Connect syntax too
  
  ```sql
  sqlplus joeuser@sales-server:1521/sales:POOLED
  ```

- In test environment, we were able to support more than 20,000 connections to a 2 GB Database Server

Scalable Event Models

- Oracle uses the poll system call on most platforms
  - Poll does not scale well for more than 1000 connections
- Newer, more efficient polling methods now supported on some platforms
  - epoll on Linux – Kernel 2.6
  - /dev/poll on Solaris and HP-UX (11.2.0.1)
  - pollset on AIX (11.2.0.2)
  - other platforms (in the works)
- Excellent scalability for Shared servers and DRCP
- Enabled by default for DRCP
- To enable, set in server sqlnet.ora
  - USE_ENHANCED_POLL = on
Inbound Connect Timeouts

- Limits the time taken for a client to connect and authenticate

- `SQLNET.INBOUND_CONNECT_TIMEOUT`
  - Controls timeout for Database server processes

- `INBOUND_CONNECT_TIMEOUT_listener_name`
  - Controls timeout for the listener

- Available from 10gR1 onwards
- Default value of 60 seconds in 10gR2 and above
- Independent of client-side timeouts
TCP Valid Node Checks

• Use TCP Invited Nodes
  – List of IPs or hostnames that are permitted to connect

• Use TCP Excluded Nodes
  – List of IPs or hostnames that are NOT permitted to connect

• Use CIDR notation and wildcard format for ease of configuration whenever possible

• Invited nodes takes precedence over excluded

• To enable, set in sqlnet.ora

  VALIDNODE_CHECKING = YES
  TCP.INVITED_NODES = (hostname1, hostname2)
  TCP.EXCLUDED_NODES = (hostname3, hostname4)
Intel Case Study
Advanced Query Performance Tuning – A Case Study from Intel Corporation

Viswanath Ravishankhar
(viswanath.b.ravishankhar@intel.com)
**Intel Mfg Decision Support Challenges**

### Data Size (MB) | Data Rate Seen (mbps) | Time Taken (min)
--- | --- | ---
250 | ~1 | 33
1000 | ~1 | 135

*Very Slow*

**Challenge:** Low throughputs on Intercontinental WAN due to:
- Long distances (5K, 10K, or more)
- Countries with weak infrastructure
- Bandwidth is expensive
- Used for or other IT functions also
- Opportunity: Make software more effective and efficient by:
  - Inline compression to reduce data size
  - Tune Oracle + TCP Stack for higher throughput – real value proposition

*How do we tune the Oracle + TCP Stack?*
Tuning for WAN Environments

- Determine appropriate send and receive socket buffer sizes (in sqlnet.ora)
- Determine appropriate array size (in SQL*Plus)
- Increase SDU Size (in sqlnet.ora)
- Increase TCP Window Size beyond 64KB (OS function)
- Use Bandwidth Delay Product (BDP)
- Use IPERF to get a benchmark for TCP performance between WAN endpoints

<table>
<thead>
<tr>
<th>Tuning Parameter</th>
<th>DOE Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket Buffer</td>
<td>8KB-1MB</td>
</tr>
<tr>
<td>Array Size</td>
<td>3000 for SQL*Plus</td>
</tr>
<tr>
<td></td>
<td>50000 for OCI Client</td>
</tr>
<tr>
<td>SDU Size</td>
<td>8KB, 16KB, 32KB, 64KB*</td>
</tr>
<tr>
<td>TCP Window Size</td>
<td>64KB-2MB</td>
</tr>
<tr>
<td>Bandwidth Delay Product</td>
<td>312KB</td>
</tr>
<tr>
<td>IPERF Max TCP WAN Throughput</td>
<td>4 mbps</td>
</tr>
</tbody>
</table>

*Special Patch from Oracle*

Oracle + TCP Tuning is an exercise in Multi-Variate Analysis
What we discovered and learned

<table>
<thead>
<tr>
<th>Client</th>
<th>50MB Query Time (Sec)</th>
<th></th>
<th></th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32K SDU</td>
<td>64K SDU</td>
<td>Delta</td>
<td></td>
</tr>
<tr>
<td>SQL*Plus</td>
<td>43</td>
<td>26</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>OCI Client</td>
<td>39</td>
<td>24</td>
<td>39%</td>
<td></td>
</tr>
</tbody>
</table>

Larger SDU gave big performance boost after tuning other parameters to optimal values

Array size level analysis shows that 64KB SDU yields:
- Improvement in overall performance
- Improvement in consistency of performance

Summary:
- Overall performance tuning gave the best performance boost for large data extracts
- More opportunities exist to improve query performance at network level
The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.