Oracle Net Troubleshooting und Tuning

Uwe M. Küchler
OPITZ CONSULTING
Bad Homburg v.d.H.

Schlüsselworte
Oracle, Datenbank, Netzwerk, SQL*Net, TNS.

Agenda

1. Einleitung
2. Kurzeinführung in Oracle Net
3. Warum überhaupt optimieren?
4. Wie optimieren?
5. Troubleshooting: Fallbeispiel „sporadische Ausfälle“
6. Konfigurationsempfehlungen

Einleitung
Meistens fristet Oracle Net (früher: „SQL*Net“) ein Schattendasein: Ist erst einmal ein Listener eingerichtet und die tnsnames.ora verteilt, dann funktioniert eben alles.
Aber was ist, wenn es eines Tages dann doch Probleme bei der Kommunikation mit der Datenbank gibt? Wie kommt man Problemen schnell auf die Fährte? Wie kann man Oracle Net tunen, und muss man das überhaupt? Dieser Vortrag wird anhand von Praxisfällen die Möglichkeiten beim Troubleshooting und beim Optimieren aufzeigen und geht auch auf einige, viel zu wenig bekannte Features ein.

Kurzeinführung in SQL*Net
Oracle Net, in früheren Versionen auch als „SQL*Net“ bekannt ist die wesentliche Grundlage für die Kommunikation mit der Oracle-Datenbank über das Netzwerk. Es ist Teil der „Oracle Net Services“, die die Komponenten

- Oracle Net
- Listener
- Connection Manager
- Oracle Net Configuration Assistant
- Oracle Net Manager

umfassen.

Oft wird Oracle Net missverständlicherweise als Netzwerkprotokoll bezeichnet, was jedoch nicht den Tatsachen entspricht. Präziser formuliert, handelt es sich bei Oracle Net um eine Software, die Verbindungen zwischen einem Client und einem DB-Server herstellt, diese Verbindungen aufrecht erhält und Mitteilungen zwischen Client und Server austauscht. Dies erfolgt über bestehende Netzwerkprotokolle, wobei wir in diesem Vortrag ausschließlich TCP/IP betrachten.
Die allgemeinen Aufgaben übernimmt die Komponente „Oracle Net Foundation Layer“, die Abbildung auf das verwendete Netzwerkprotokoll die Komponente „Oracle Protocol Support“.

Demzufolge sind viele Probleme bei der Performance auch nicht primär bei Oracle Net zu suchen sondern bei der eingesetzten Netzwerk-Infrastruktur.

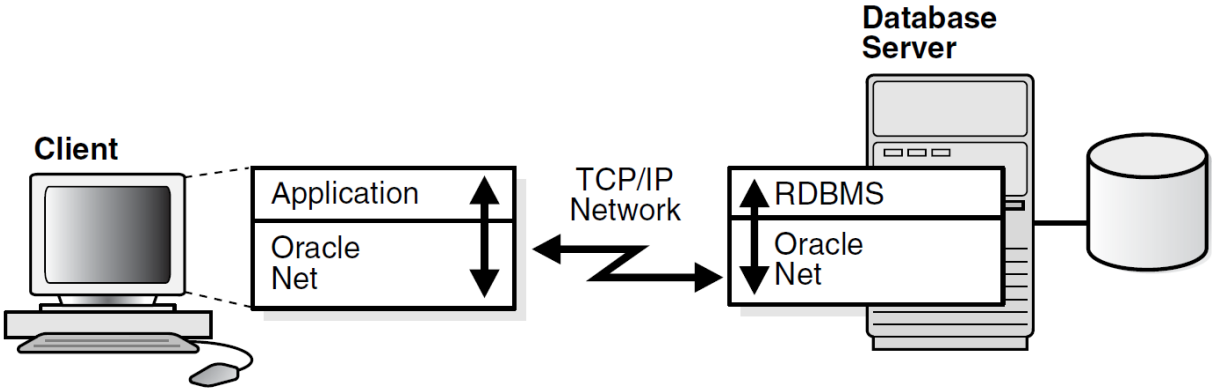

Abb. 1: Verbindung von Client und DB-Server über Oracle Net. Quelle: Oracle

qqq.

Warum überhaupt optimieren?
Der Anlass einer Optimierung kann bei Oracle Net verschieden sein:

1. Performance und Skalierbarkeit
    a. Datendurchsatz
    b. Schneller Verbindungsaufbau
2. Hochverfügbarkeit
    a. Failover im Fehlerfall
    b. Lastverteilung
3. Sicherheit.

Nicht immer können alle diese Ziele gleichzeitig erreicht werden; so kann z.B. die Verschlüsselung als Teilkomponente der Sicherheit den Datendurchsatz und damit die Performance verringern. Betrachten wollen wir daher vor allem den Punkt „Performance und Skalierbarkeit“, werden die anderen Punkte damit aber zwangsläufig streifen.

Wie optimieren?

Performance

Oracle Net selbst bietet nur wenige Parameter, die sich für eine Optimierung anbieten. Viele Optimierungen müssen, wie eingangs schon erwähnt, im Bereich der Netzwerk-Konfiguration oder seitens der Applikation vorgenommen werden.

Im Wesentlichen reduzieren sich die Optimierungsansätze auf zwei:

1. Zur Verbesserung des Datendurchsatzes:
   Packe so viel Information wie möglich in so wenige Pakete wie nötig.
2. Zur Verbesserung des Verbindungsaufbaus:
   Brich den Verbindungsaufbau so früh wie möglich ab und versuche ggf. eine alternative Verbindung (bei RAC oder Data Guard)

Zur Verbesserung des Datendurchsatzes bzw. der optimalen Ausnutzung der TCP-Pakete bietet Oracle Net den Parameter **SDU_SIZE** an.

„SDU“ steht für „Session Data Unit“. Hierbei handelt es sich um einen Puffer in der Network Session (NS)-Schicht, über den Oracle Net-Pakete an die Transport-Schicht (NT) weitergereicht werden. Ab Oracle 11.2 ist die SDU bereits auf 8192 Bytes voreingestellt (früher 2kB), bei Shared Server sogar 65535 Bytes (dies ist das Maximum in 11.2, in 12.1 liegt das Maximum bei 2097152 Bytes). Die SDU einer Verbindung wird zwischen Client und Server ausgehandelt und dabei auf den kleineren, konfigurierten Wert gesetzt.

Mittels der SDU_SIZE lässt sich regulieren, wie viele Informationen – bei einem SELECT z.B. die Ergebnisse – in ein Oracle Net-Paket geschrieben werden, bevor sie an die NT gehen. Damit lässt sich der Aufwand für die Paket-Header, die für jedes Oracle-Net-Paket mit übertragen werden müssen, reduzieren.

Um die SDU z.B. auf 2 MB zu erhöhen, muss die Einstellung serverseitig im Listener und Client-seitig in der sqlnet.ora oder tnsnames.ora getroffen werden:

```
-- listener.ora
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (SDU = 2097152)
      (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = meinhost)
        (PORT = 1521)
      )
    )
  )

-- sqlnet.ora (Client)
DEFAULT_SDU_SIZE=2097152

-- tnsnames.ora (Client)
DB12 =
  (DESCRIPTION =
    (SDU = 2097152)
    (ADDRESS =
      (PROTOCOL = TCP)
      (HOST = meinhost)
      (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = DB12)
    )
  )
```

Wenn nun eine möglichst große SDU den Aufwand reduziert, warum steht sie dann nicht von vornherein auf dem Maximum? Das liegt vor allem daran, dass die Puffer pro Session angelegt werden und somit RAM verbrauchen. Die Voreinstellung auf 8kB stellt daher einen guten Kompromiss zwischen Speicherverbrauch und Aufwand für den Transport dar.

Oracle empfiehlt eine Änderung der SDU in folgenden Fällen:

- Daten vom Server kommen in fragmentierten Paketen an.

- Es wird über ein WAN kommuniziert, auf dem lange Laufzeiten beobachtet werden.
- Große Datenmengen werden übertragen.
- Die Paketgrößen sind durchgehend gleich.

Ein typischer Anwendungsfall wäre damit die Übertragung von größeren Datenmengen über ein WAN in ein Data Warehouse.

Wie so oft bei der Performance-Optimierung gilt aber: „Etwas nicht zu tun ist schneller, als es überhaupt zu tun". Viel effizienter als die Einstellung der SDU_SIZE ist daher die **Vermeidung überflüssiger Kommunikation**!

Troubleshooting: Fallbeispiel

Manche Probleme — besonders die, die nur sporadisch auftreten — sind nicht so leicht zu diagnostizieren und erfordern einen tieferen Einstieg in die Materie. In diesem Praxisbeispiel bedeutet das: SQL*Net Tracing und Verständnis für die Funktionsweise des Betriebssystems, speziell die Generierung von Zufallszahlen.

Der vorliegende Fall eignet sich daher sehr gut, um eine Vorgehensweise beim Troubleshooting von Verbindungen zur Datenbank exemplarisch darzulegen.

Szenario

Ein in Java geschriebener Batch-Job führt regelmäßig einen Datenabgleich mit einer entfernten Datenbank durch. Dazu werden von der Anwendung bis zu fünf Verbindungen kurz hintereinander aufgebaut.

Dabei kommt es gelegentlich zu Timeouts beim Verbindungsaufbau, so daß nur ein Teil der Verbindungsversuche erfolgreich ist. Ein genauerer Blick in die Logfiles der Anwendung zeigt, daß viele Verbindungen in weniger als einer Sekunde aufgebaut werden, wohingegen andere über 20s benötigen.

- DB-Server: Oracle Linux 5, Oracle EE 11.2.0.3
- JDBC: Oracle ojdbc6.jar, Version 11.2.0.3

Vorbemerkungen

Das Eröffnen einer Session in Oracle läuft wie folgt ab:

1. Client schickt einen Request an den Oracle-Listener
2. Listener prüft, ob er eine DB-Instanz bzw. einen Service für den Request kennt
3. Listener geht an den pmon-Hintergrundprozeß der Datenbankinstanz
4. pmon startet einen Serverprozess auf dem Datenbankserver (sichtbarer Prozeß auf OS-Ebene)
5. Der Serverprozess allokiert einen freien IP-Port auf dem Datenbankserver im Bereich von üblicherweise 9000-64000 (s.o., dort war es Port 33600)
6. pmon schickt die Information über den gestarteten Serverprozeß mit OS-Port an den Listener
7. Listener schickt die Information an den Client
8. Ab jetzt hat der Listener mit der Datenbankverbindung nichts mehr zu tun!
9. Client startet eine neue IP-Verbindung DIREKT zum Serverprozeß mit dem vom Listener ausgehandelten Port

10. Der Serverprozess fordert nun den Client zur Authentifizierung auf.

Erfolgt die Authentifizierung nicht innerhalb des vorgegebenen Timeouts, dann beendet der pmon den Serverprozess und schließt die IP-Verbindung zum Client. Diese Aktion wird im alert.log von Oracle protokolliert und ist oben zu beobachten.

Es ist also schon einmal klar, daß die Verbindung zwischen Client und Server grundsätzlich immer zustande kommt, es dann aber gelegentlich nicht zu einer erfolgreichen Authentifizierung innerhalb des vorgegebenen Zeitfensters kommt.

Trace als Beweismittel

Um noch klarer zu belegen, daß das Problem nicht auf Seiten des DB-Servers liegt, bietet sich ein Tracing des SQL*Net-Protokolls an, bei dem detailliert jeder Schritt des Verbindungsaufbaus dargelegt wird.

Das Tracing kann an zwei Stellen erfolgen: Am Listener und am Serverprozess. Aus den ersten Erkenntnissen wissen wir bereits, daß der Listener seine Arbeit erfolgreich durchgeführt hat. Daher muss für den Serverprozess eine weitere Einstellung in der Datei “sqlnet.ora” gemacht werden:

```
TRACE_LEVEL_SERVER=16
```

Die Einstellung greift sofort ab dem Abspeichern für alle neu gestarteten Serverprozesse, kann also in kurzer Zeit sehr viele und große Dateien erzeugen. Sie sollte daher schnellstmöglich wieder abgeschaltet werden!

Um die Tracefiles in ein lesbareres Format zu bringen, liefert Oracle das Tool “trcasst” mit. Die passende Kommandozeile für unseren Anwendungsfall ist “trcasst -o” und liefert folgende Ausgabe (auch hier nur auszugsweise):

```
…

<--- Received 158 bytes - Data packet timestamp=04-NOV-2013 10:50:46:024

Start of user function (TTIFUN)

Get the session key (OSESSKEY)


---> Send 210 bytes - Data packet timestamp=04-NOV-2013 10:50:46:031

Return opi parameter (TTIRPA)


<--- Received 992 bytes - Data packet timestamp=04-NOV-2013 10:51:13:023

Start of user function (TTIFUN)

Generic authentication call (OAUTH)

…
```

“Send” steht hier für den Server, “Received” für empfangene Nachrichten vom Client.

Man sieht hier, daß der Server zunächst eine Anfrage erhalten hatte, diese nach 7 ms bereits beantwortete und der Client daraufhin nach etwa 27 s erst wieder antwortete. Die weitere Authentifizierung lief dann schnell und ist hier nicht mehr dargestellt.

Eine Überprüfung der Netzwerkaktivität mit Wireshark belegte desweiteren, daß die gesendeten 210 Bytes auch nicht irgendwo im Netzwerk "hängen geblieben" waren sondern mit den üblichen Latenzen ausgeliefert wurden. Der Client war nun eindeutig als Ort der Zeitverzögerung identifiziert!

Die weitere Lösung des Problems wird im Laufe des Vortrags dargestellt.

Literatur

- Oracle 12c Net Services Reference
- Oracle 12c Net Services Administrator's Guide
- "Oracle Net Performance Tuning" (Doc ID 67983.1)
- "Undocumented or Lesser Known SQL*Net/Net8/Net8i Features & Parameters" (Doc ID 39357.1)
- Stefan Koehler, SAP on Oracle Blog: „[Oracle] SQL*Net researching - Setting Session Data Unit (SDU) size and how it can go wrong"
- Setting Parameters for Scan and Node Listeners on RAC, Queuesize, SDU, Ports, etc (Doc ID 1292915.1)

Kontaktadresse:

OPITZ CONSULTING Deutschland GmbH
Uwe Küchler
Norsk-Data-Str. 3
Bad Homburg v.d.H.

Telefon: +49 (0) 2261-6001 0
E-Mail uwe.kuechler@opitz-consulting.de
Internet: www.opitz-consulting.de