Vom Client zum Server
der Verbindungsaufbau im Detail

Martin Berger

Wien

Schlüsselworte

Datenbank, sqlplus, listener, Verbindung, bequeath, prelim, shared servers, SCAN

Einleitung

Die Verbindung eines Clients zu einer Oracle Datenbank ist wohl eine der selbstverständlichsten Aktionen in heutigen IT Organisationen. Kaum ein Systemarchitekt würde sich jemals darüber Gedanken machen. Erfahrene DBAs sind meist schon froh, wenn sie gefragt werden, ob direkte Verbindungen oder Shared Server (früher Multi-Threaded Server) verwendet werden sollen.
Und doch haben diese so einfach scheinenden Aktionen einige Tücken im Detail.
Im Folgenden werden einige verschiedene Arten des Verbindungsaufbaus genauer beschrieben und auf die daraus entstehenden Tücken eingegangen. Einige Beispiele aus dem technischen Alltag des Autors lockern die einzelnen Absätze etwas auf.
Der Vortrag bezieht sich nur auf Oracle 11.2 auf Linux. Weder werden neue Features von 12c noch andere Architekturen (z.b. Windows) besprochen.

bequeath connection

Die auf den ersten Blick einfachste Verbindung zu einer Datenbank ist die `bequeath connection`. Die wohl bekannteste Art ist `sqlplus „/ as sysdba“.` Dabei fällt jede Verbindung, bei der auf Betriebssystem Ebene nur `ORACLE_HOME` und `ORACLE_SID` gesetzt werden und bei der Verbindung kein Alias oder Connection String angegeben wird unter diese Kategorie. (Eine Besonderheit ist die Zusätzliche Verwendung der Umgebungsvariable `TWO_TASK` – die trotzdem eine Verbindung über einen Listener initiiert).
Jede Oracle Instanz beginnt mit einer bequeath connection. Der sqlplus Prozess startet einige weitere Prozesse. Aus den Werten von ORACLE_HOME und ORACLE_SID werden Hash-Werte gebildet, die für die Semaphore und Shared Memory der Instanz verwendet werden. Einer der von sqlplus gestarteten Prozesse verbindet sich zu diesen konkreten Semaphoren und Shared Memory.

```
Connected to an idle instance.

SQL> startup upgrade
ORACLE instance started.
```

Abb. 1: idle instance

Mit dem `startup` werden weitere Prozesse gestartet, die alle die Eigenschaften Ihrer „Eltern“ erben.

Mit dem Befehl strace kann dieses Starten weiterer Prozesse sehr gut beobachtet werden.

```
18230 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2b4943c40b80) = 18231
18231 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18283
18283 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18284
18284 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2ab005560ec0) = 18292
18292 clone( <unfinished ...>
18292 <... clone resumed> child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2ab005560ec0) = 18293
18231 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18296
18231 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18302
18302 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18303
18231 clone( <unfinished ...>
18231 <... clone resumed> child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18304
18304 clone( <unfinished ...>
18304 <... clone resumed> child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2afb7e669ec0) = 18305
```

Abb. 2: Startup Prozess Sequenz

Dieses Muster wiederholt sich, bis alle für die Instanz notwendigen Prozesse gestartet sind.

Im konkreten Beispiel sind die ersten Prozesse mit Ihren Abhängigkeiten so dargestellt:

sqlplus	18230	18231	18283	18284	18292	18292	18293	
			18296					
			18302					
PMON				18303				
			18304					
PSP0				18305	18306			
VKTM						18307	18308	18309
					18310			
GEN0						18311		

Abb. 3: Startup Prozesse und Abhängigkeiten

Ab dem Erscheinen von PSP0 ist die Liste wenig spannend. Oracle bezeichnet diesen Prozess zu Recht so: „*PSP0 (process spawner) spawns Oracle processes*.“

Diese Art, wie eine Oracle Instanz mit all ihren Prozessen gestartet wird hat zur Konsequenz, dass alle diese Prozesse die Eigenschaften des ursprünglichen `sqlplus` erben. Diese Eigenschaften sind zum Beispiel Umgebungsvariablen, Systemlimits oder user/groups, um nur einige zu nennen.
Diese Eigenschaften sind zentral am einfachsten im `/proc` filesystem zu erfahren. Zum Beispiel `/proc/<pid>/limits` oder `/proc/<pid>/status`. Leider sind einige dieser files nur mit root Rechten lesbar.

Ein "`sqlplus /`" bei einer laufenden Instanz hat viel Weniger Arbeit zu leisten. Es verbindet sich zu den existierenden shared memory und semaphore Strukturen. Danach befüllt es die entsprechenden session und process Strukturen (sichtbar z.B. in `v$session` und `v$process`) und ist verfügbar.

Ein besonderer Prozess ist `sqlplus /prelim „/ as sysdba"` [1]. Dieser Prozess startet gleich wie soeben beschrieben, verzichtet aber auf den letzten Schritt, hat also keine session und prozess Strukturen. Dieser Zustand kann wünschenswert sein, wenn Probleme in der SGA ein reguläres einloggen verhindern. Um dennoch ein oradebug hanganalyse durchführen zu können, kann ein solcher prelim Prozess einen anderen dazu bringen, den Dump zu erzeugen:

```
ORADEBUG SETOSPID OS_PID
ORADEBUG DUMP HANGANALYZE 3
```

Listener

Die meisten Verbindungen zu einer Oracle Datanbank werden über einen Listener initiiert. Oracles offizielle Beschreibung dazu ist funktional korrekt. Allerdings wird die genaue Implementierung nicht besprochen und dadurch kein technisches Verständnis vermittelt.

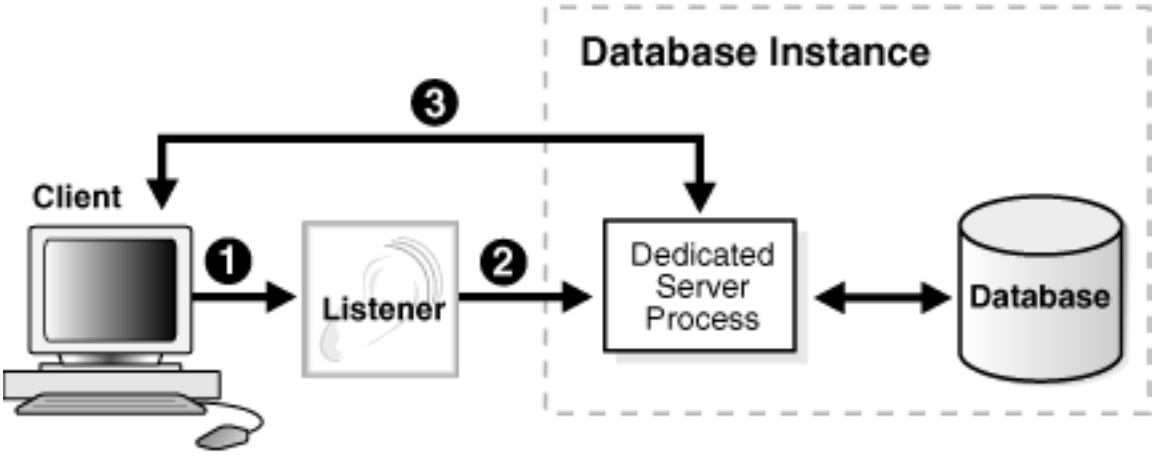


Abb. 4: Listener und dedicated Server

Wieder einmal hilft `strace`, die genauen Abläufe zu identifizieren.

Nachdem der Listener den Netzwerk-Request entgegen genommen hat, muss erst die Instanz, zu der die Verbindung aufgebaut werden soll identifiziert werden. Danach startet der Listener 2 neue Prozesse.

```
[pid  2979] clone(Process 27028 attached
child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x2aedd9914b80) = 27028
[pid  2979] wait4(27028, Process 2979 suspended
 <unfinished ...>
[pid 27028] clone(Process 27029 attached (waiting for parent)
Process 27029 resumed (parent 27028 ready)
child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x2aedd9914b80) = 27029
[pid 27028] exit_group(0)               = ?
Process 2979 resumed
Process 27028 detached
[pid  2979] <... wait4 resumed> [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0,
NULL) = 27028
[pid 27029] close(15 <unfinished ...>
[pid  2979] --- SIGCHLD (Child exited) @ 0 (0) ---
[pid 27029] <... close resumed> )       = 0
[pid  2979] close(14 <unfinished ...>
[pid 27029] close(16 <unfinished ...>
[pid  2979] <... close resumed> )       = 0
[pid 27029] <... close resumed> )       = 0
[pid  2979] close(17)                   = 0
```

Der erste Prozess beendet sich sofort wieder. Er scheint nur notwendig zu sein, um die Parent ID des Prozesses auf 0 zu setzen.

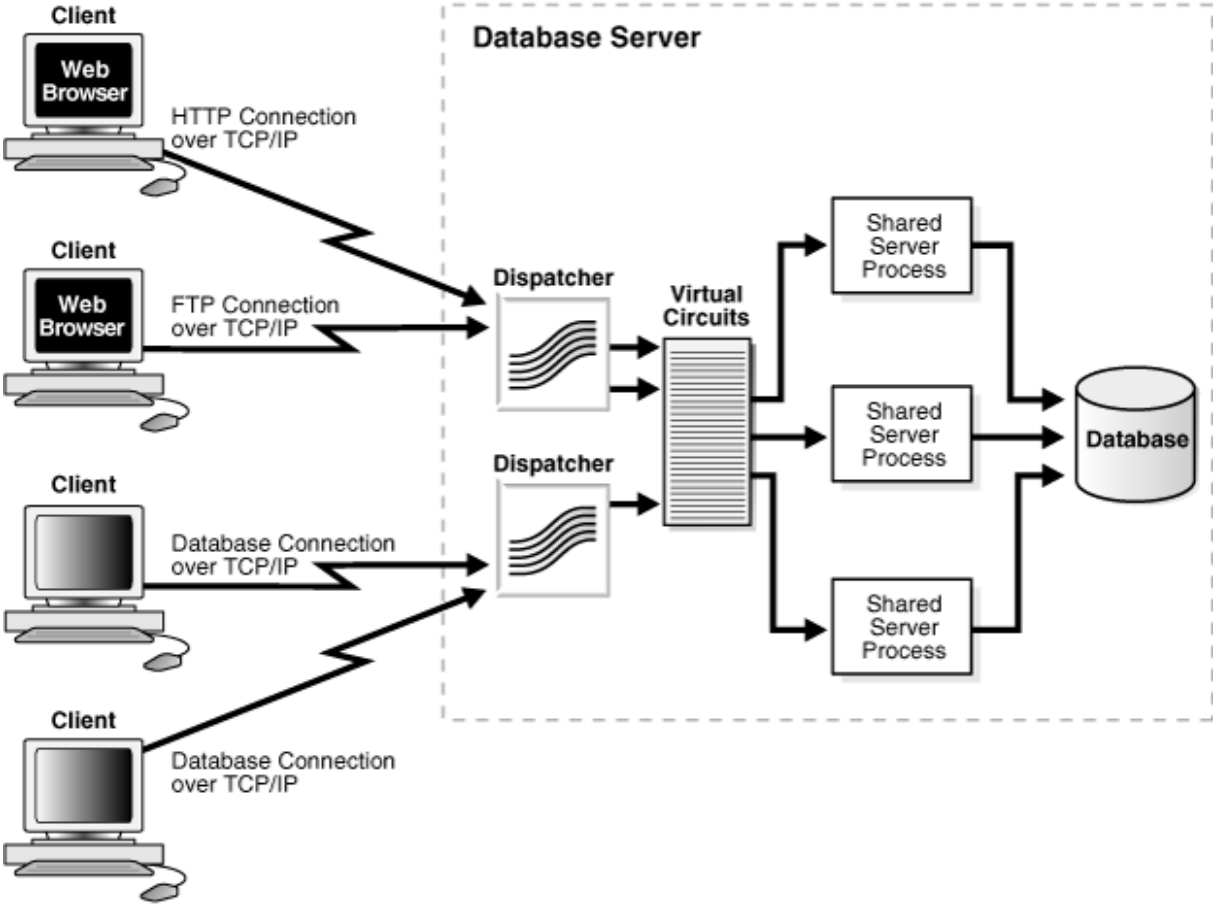
Danach verwandelt sich dieser Prozess in einen echten Server Prozess:

```
[pid 27029] execve("/appl/oracle/product/rdbms_112022_a/bin/oracle",
["oracleTTT051", "(LOCAL=NO)"], [/* 109 vars */]) = 0
```

execve erzeugt im kontext des aktuellen Prozesses einen neuen. Die PID und die file descriptoren bleiben erhalten. Allerdings kann sich die effective userID ändern. Dieser Prozess kann sich nun zum shared memory und den Semaphoren der Instanz attachen. Die Informationen dazu (`ORACLE_HOME` und `ORACLE_SID`) bekommt der Prozess als setzte Informationen über zwei Sockets vom Listener. Danach schließt der Listener selbst alle filehandles – besonders auch seine Netzwerk Verbindung des Clients. So ist nun der neue Prozess als decicated process der einzig verantwortliche für den Client. Die korrekten Werte für `ORACLE_HOME` und `ORACLE_SID` hat der Listener entweder aus der `SID_LIST` oder der `PMON` Prozess hat sie im Rahmen der service registration übergeben.

Listener mit shared server

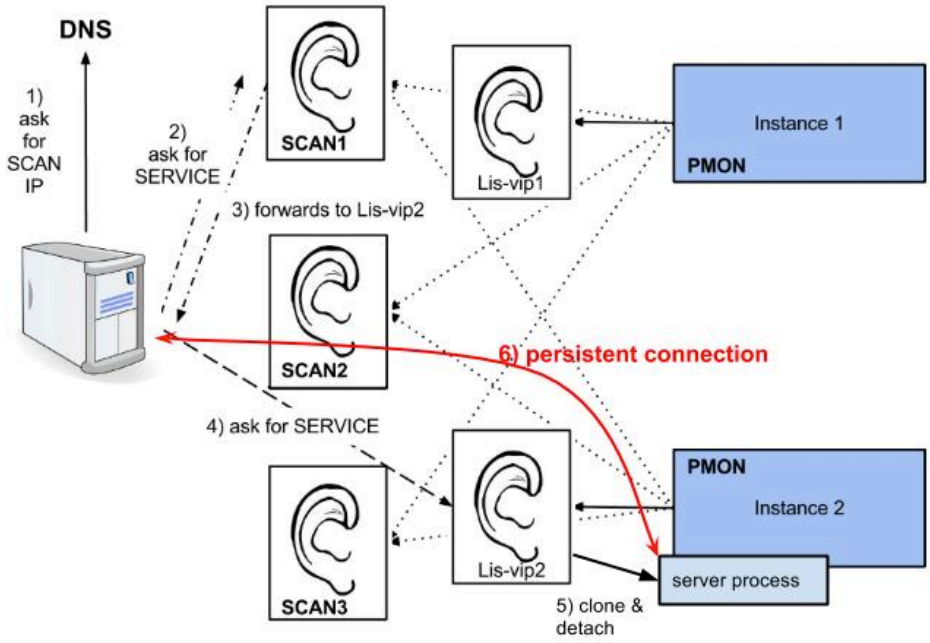
Shared Server (früher Multi Threaded Server genannt) sind Prozesse, die von der Instanz vorab gestartet werden und die Anfragen der Clients beantworten. Dabei gibt es Dispatcher Prozesse, wobei jeder Client immer einen konkreten Dispatcher Prozess hat. Die Dispatcher werden von der Instanz beim Listener Registriert und können ab diesem Zeitpunkt Verbindungen der Clients behandeln.



In dieser Konfiguration kümmert sich der Listener nicht um das Erzeugen der Prozesse. Die Verbindung wird lediglich weitergereicht.

SCAN Listener mit shared server

In der Version 11.2 hat Oracle „Single Client Access Name“ (SCAN) eingeführt.
Prinzipiell werden die schon länger verfügbaren Technologien rund um die Parameter `local_listener` und `remote_listener` zu einem neuen Feature erweitert, um die Konfiguration des Clients zu vereinfachen. Lediglich der Client wurde um ein kleines Feature erweitert, um bei der Nichtverfübgarkeit eines SCAN-Listeners den nächsten zu versuchen. Deshalb empfielt es sich, bei älteren Clients auch die herkömliche Variante mit `ADDRESS_LIST` zu verwenden.
Für SCAN registriert jeder PMON alle Services beim lokalen Listener, der in `local_listener` angegeben ist. Zusätzlich registriert PMON bei allen `remote_listener` alle Services, zusammen mit der derzeitigen lokalen Systemlast und dem `local_listener` – genau so wie im parameter `local_listener` angegeben.



Beim Verbindungsaufbau eines Clients fragt dieser zuerst beim DNS nach den IPs, die zu den SCAN Eintrag gehören (1). Zu einer dieser IPs wird eine Verbindung hergestellt und nach dem SERVICE gefragt (2). Der SCAN Listener liefert genau den `ADDRESS` Eintrag des `local_listener` mit der geringsten Systemlast zurück (3). Darauf verbindet sich der Client zu diesem local_listener (4) – nochmals für dasselbe SERVICE. Ab diesem Schritt folgen die Schritte wie schon im Abschnitt **Listener** beschrieben.

Quellen

[1] Tanel Poder - Oradebug hanganalyze with a prelim connection and “ERROR: Can not perform hang analysis dump without a process state object and a session state object.” - http://blog.tanelpoder.com/2012/05/08/oradebug-hanganalyze-with-a-prelim-connection-and-error-can-not-perform-hang-analysis-dump-without-a-process-state-object-and-a-session-state-object/

Kontaktadresse:

Martin Berger
Lederergasse 27/2/14
A – 1080 Wien

Telefon: +43 660 660 83306
E-Mail martin.a.berger@gmail.com
Internet: berxblog.blogspot.com
Twitter: @martinberx