Andernfalls Prozess am Hals

Über die Prozesse in der Oracle Datenbank

Eero Mattila
Dell Software GmbH
Köln

Schlüsselworte
Oracle Prozesse, Hintergrundprozesse, Multithreaded Execution

Einleitung

Das Oracle RDBMS kommt mittlerweile mit einer wahren Dschungel von Prozessen daher. Konnte Oracle7 noch mit mindestens vier Hintergrundprozessen betrieben werden, so kommt 12c im Extremfall auf über 400. Die Ausgabe der laufenden Prozesse auf einem Datenbankserver kann einem Datenbankadministrator inzwischen Angst einflößen.

Welche Arten von Prozessen gibt es? Was machen diese ganzen Prozesse eigentlich? Brauche ich sie alle, oder kann man sich vielleicht welche Sparen?

Dieser Vortrag soll (vor allem jüngeren) DBAs vermitteln, wie die diversen Prozesse der Oracle Datenbank erkannt und analysiert werden können – auch mit dem Ziel, die Ehrfurcht davor zu nehmen, aber den Respekt davor zu wahren.

Kurzer Rückblick

Oracle7 kam 1992 auf den Markt und erneuerte das RDBMS recht grundlegend – neben tatsächlich funktionierender referentieller Integrität, Stored Procedures und Functions, Triggern, neuem User- und Rollenkonzept und vielen anderen Neuigkeiten wurden auch einige neue Hintergrundprozesse für die Instanz eingeführt, wie der Checkpoint (CKPT), Distributed Database Recovery (RECO), Lock (LCKn) oder Snapshot Refresh (SNPn). Eine Oracle7-Instanz konnte jedoch mit einem Minimum von vier Prozessen betrieben werden: Database Writer (DBWR), Log Writer (LGWR), Process Monitor (PMON) und System Monitor (SMON). Der Checkpoint-Prozess konnte optional zur Entlastung des LGWR konfiguriert werden, und RECO, SNPn und LCKn kamen nur selten in verteilten Systemen bzw. dem Parallel Server (Vorläufer von RAC) zum Einsatz. Den Archiver (ARCH) gab es natürlich auch, obligatorisch war er technisch gesehen nicht.

Mit Oracle8i und Oracle9i folgten Neuerungen wie Advanced Queuing und Real Application Cluster (RAC), welche die Prozesslandschaft etwas erweiterten, aber noch recht überschaubar. Bei einer typischen Installation blieb die Anzahl der Hintergrundprozesse bei sechs bis zehn, und selbst wenn alle Optionen aktiv waren, konnte die Prozessauflistung bequem auf einer Bildschirmseite eingesehen werden. Oracle10g kannte bereits 35 verschiedene Hintergrundprozesstypen (neu war vor allem ASM), und eine wahre Explosion erlebte das RDBMS ab Oracle11g: Die Dokumentation listete über 50 neue Prozesse auf. Führt man heute ein count(*) auf die virtuelle View V$BGPROCESS einer Oracle12c Datenbankinstanz aus, so erhält man als Ergebnis die stolze Zahl von rund 370 bis über 400, je nach Konfiguration und installierter Optionen. Selbst wenn man beachtet, dass viele davon parallele Instanzen von den verschiedenen Prozesstypen sind, so kann von einer Übersichtlichkeit

keine Rede mehr sein. Ein Trost ist allerdings, dass in einer RDBMS- oder auch ASM-Instanz meist nur eine Untermenge aller möglichen Prozesse tatsächlich vorkommt.

Prozesse: Definitionen

Der Begriff Prozess wird im Folgenden der Einfachheit halber synonym zu Thread verwendet. Bei Windows läuft Oracle als ein Hauptprozess, und die Hintergrundprozesse sind als Threads des Hauptprozesses implementiert, während unter Unix- und Linux-Betriebssystemen alle Prozesse einzeln zu sehen sind.

In einer Oracle Instanz gibt es drei Prozesskategorien:

1) *Serverprozesse* führen diverse Aufgaben im Auftrag von Anwendersessions aus. In einer typischen Umgebung nehmen sie meist den größten Anteil an CPU-Verbrauch in Anspruch.

2) *Hintergrundprozesse* werden mit der Instanz gestartet, und sie erledigen verschiedenste Arbeiten wie Speicherverwaltung, Redo-Log-Management, Wegschreiben von Datenblöcken und so weiter. Einige von ihnen sind unverzichtbar, andere dagegen, sogenannte Utility-Prozesse, sind optional in Abhängigkeit von der Art der Instanz (RDBMS, ASM) und den konfigurierten Features wie RAC, Data Guard, Advanced Queuing und andere.

3) *Slave-* oder *Worker-Prozesse* sind dem Wesen nach ähnlich wie Hintergrundprozesse, aber sie werden entweder von Server- oder Hintergrundprozessen aufgerufen, um zusätzliche Arbeit zu verrichten.

Serverprozesse

Serverprozesse werden gestartet, sobald User sich mit der Datenbankinstanz verbinden, und sie führen sämtliche Requests der Anwendersessions aus. Oracle unterscheidet zwischen *dedizierten* und *shared* Servern. Dedizierte Serverprozesse haben eine eins-zu-eins-Beziehung zu einer Datenbankverbindung oder Session, während bei Shared Servern mehrere Verbindungen oder Sessions sich aus einem Pool von Serverprozessen bedienen.

Dedizierte Server sind die am weitesten verbreitete und auch von Oracle empfohlene Konfiguration. Einzelne Sessions konkurrieren nicht um die Prozessressourcen, selbst bei lang laufenden Transaktionen. Da einige Operationen, wie das Starten und Stoppen der Datenbank, immer über dedizierte Server durchgeführt werden müssen, ist der dedizierte Modus immer vorhanden ohne besondere Konfiguration.

Shared Server, früher *Multithreaded Server* oder *MTS* genannt, wurden eingeführt, um die Anzahl der Prozesse sowie den Speicherbedarf auf einem Datenbankserver zu reduzieren. Richtig eingesetzt, kann die Ersparnis an Prozessressourcen des Betriebssystems erheblich sein: Ein System mit 5.000 angemeldeten Sessions, von denen maximal 50 gleichzeitig aktiv sind, kann mit 50 Shared Servern effizient arbeiten. Mit dedizierten Servern wären 5.000 Serverprozesse erforderlich. Die Ersparnis an Speicherbedarf ist eine direkte Folge der geringeren Anzahl an Prozessen. Jeder Prozess, ob dediziert oder shared, besitzt eine PGA. Statt 5.000 PGAs würden beim obigen Beispiel nur 50 benötigt. Das automatische PGA Memory Management verringert diesen Vorteil jedoch mittlerweile. Andererseits wird im Shared Server Modell mehr Speicher für die SGA benötigt, weil die UGA der Sessions statt PGA in die SGA verlagert wird. Voraussetzung für den reibungslosen Betrieb mit Shared Servern ist, dass die Sessions kurze Transaktionen verarbeiten, um die Serverprozesse nicht zu lange für sich zu

beanspruchen und dadurch das Gesamtsystem abzubremsen oder ganz zum Erliegen zu bringen: Wenn kein Shared Server zur Verfügung steht, kann eine Session ihre Arbeit nicht fortsetzen.

Um Shared Server einzusetzen, muss grundsätzlich nur ein Parameter konfiguriert werden (auch wenn für die Optimierung mehrere zur Verfügung stehen):

```
alter system set shared_servers = n scope = both;
```

wobei `n` größer als 0 ist. Zusätzlich muss der Verbindung entsprechend vorgenommen werden – entweder mit dem einem Connect String mit dem Zauberwort SHARED:

```
ora12centos2:1521/DOAG:SHARED
```

oder einem TNSNAMES-Eintrag:

```
DOAG_SHARED=
  (DESCRIPTION=
    (ADDRESS=
      (PROTOCOL=TCP)
      (HOST=ora12centos2)
      (PORT=1521)
    )
    (CONNECT_DATA=
      (SERVER=SHARED)
      (SERVICE_NAME=DOAG)
    )
  )
```

Seit Oracle11g steht noch die Möglichkeit des Database Resident Connection Pooling (DRCP) zur Verfügung, was gewissermaßen eine Mischung aus beiden vorgenannten Konzepten darstellt: Eine Session erhält aus dem Pool einen Serverprozess, der ihr aber für die gesamte Dauer der Session dediziert zur Verfügung steht. DRCP verbindet gewissermaßen die Eigenschaften beider vorgenannten Modelle: Weniger Prozesse durch Connection Pooling, potenzielle Speicherersparnisse, und jede Session hat ihren dedizierten Prozess für die gesamte Dauer ihrer Arbeit. Gute Kandidaten für DRCP sind daher Anwendungen, bei denen sehr viele, meist aber sehr kurze Sessions vorkommen, und die kein eigenes Connection Pooling besitzen, wie z.B. PHP.

Die Konfiguration von DRCP erfolgt über das Package DBMS_CONNECTION_POOL, und der Connect String oder der TNSNAMES-Eintrag muss das Wort `POOLED` enthalten.

Solange das Datenbanksystem nicht unter übermäßiger Last durch unzählige Serverprozesse leidet, sind dedizierte Prozesse in aller Regel zu bevorzugen. Sie erfordern keiner besonderen Konfiguration, und sie erleichtern das Tuning gegenüber Shared Servern: Bei Verbindungen über Shared Server können die Trace-Informationen einer Session über mehrere Trace Files verteilt sein, sodass es wesentlich umständlicher ist, die Aktionen der Session nachzuvollziehen. Falls Shared Server nicht zu vermeiden sind – sei es wegen der hohen Benutzerzahl oder weil die Anwendung oder sonstige Umstände es erfordern –, ist es in jedem Fall ratsam, auch die Entwicklungsarbeit und besonders alle Tests ebenfalls in einer Shared Server Umgebung vorzunehmen, um unangenehme Überraschungen in der Produktion zu vermeiden.

Hintergrund- und Slave-/Worker-Prozesse

Die Unterscheidung zwischen Hintergrund-, Utility- und Slave-Prozessen ist anhand der Oracle Dokumentation und der Datenbankstrukturen nicht ganz eindeutig.

Oracle Database Reference, Appendix F, definiert Hintergrundprozesse als Prozesse, die in der View V$PROCESS vorkommen können und einen non-null Wert in der Spalte PNAME besitzen.

Im Data Dictionary finden wir zwei dynamische Performance Views, die Auskunft über die Hintergrundprozesse geben:

- Die oben erwähnte **V$PROCESS** enthält Daten über alle aktuell laufenden Prozesse in der Instanz. Die Hintergrundprozesse sind darin mit einer 1 in der Spalte BACKGROUND gekennzeichnet.

- **V$BGPROCESS** enthält einen Eintrag für jeden Hintergrundprozess, der in der Instanz gestartet werden kann. Aktuell laufende Prozesse haben in der Spalte PADDR einen Wert größer 00 (RAW Datentyp).

Mit diesem Hintergrund lassen wir uns zunächst nur die Anzahl der aktuell laufenden Hintergrundprozesse in einer frisch gestarteten Oracle 12.1.0.2 Instanz anzeigen, die mit minimaler init.ora-Konfiguration gestartet wurde. Sicherheitshalber prüfen wir die Prozesse mit allen drei genannten Kriterien.

1. Alle Prozesse in V$PROCESS, die einen non-null Wert in der Spalte PNAME besitzen:

```
select count(*) from v$process where pname is not null;

  COUNT(*)
----------
        55
```

2. Alle Prozesse in V$PROCESS, die den Wert 1 in der Spalte BACKGROUND aufweisen:

```
select count(*) from v$process where background = 1;

  COUNT(*)
----------
        32
```

3. Alle Prozesse in V$BGPROCESS, die einen Wert größer 00 in der Spalte PADDR haben:

```
select count(*) from v$process where paddr <> '00';

  COUNT(*)
----------
        22
```

Die unterschiedlichen Ergebnisse kommen nicht etwa dadurch zustande, dass sich der Lastzustand der Instanz schlagartig geändert hätte. Vielmehr scheint die Definition von Hintergrundprozessen nicht ganz durchgängig zu sein:

- Die Dokumentation zählt sämtliche Slaves zu den Hintergrundprozessen und kennt über 400 Prozesse von gut 110 verschiedene Prozesstypen.
- V$BGPROCESS listet bei Minimalkonfiguration gut 70 Prozesstypen auf. Darunter sind einige, aber nicht alle Utility Prozesse, und keine Slaves oder Worker.
- V$PROCESS zeigt alle laufenden Prozesse an. Ein Urteil darüber, welche Slave- oder Utility-Prozesse als Hintergrundprozesse gelten, behält die View für sich. Offenbar zählen aber zumindest die Parallel Query Slaves und die Job Queue Prozesse nicht dazu, dafür aber Space Management Slaves.

Die Verwirrung soll uns nicht weiter stören – wir halten es mit der Dokumentation und behandeln im Folgenden alle Prozesse, die nicht mit Anwendersessions zusammenhängen, als Hintergrundprozesse.

Was tun die Hintergrundprozesse?

Es ist wenig sinnvoll, die Oracle Dokumentation hier abzuschreiben und jeden Prozess mit Namen und Bedeutung aufzulisten. Viele der Prozesse haben einen recht eindeutigen Namen, der die Aufgabe beschreibt: *Database Block Writer* (DBWn) schreibt geänderte Datenblöcke in die Datendateien, *Log Writer* (LGWR) schreibt die Inhalte des Log Buffers aus der SGA in die Redo Log Dateien. Andere Prozesse wiederum haben einen vermeintlich sprechenden Namen, der aber ihre Aufgabe eher irreführend beschreibt: Der Checkpoint-Prozess (CKPT) ist nicht etwa für die Durchführung Checkpoints (das Schreiben von geänderten Datenblöcken in die Datendateien) zuständig – das macht größtenteils der DBWn, wie eben erwähnt. Der CKPT signalisiert DBWn, dass es Zeit ist, einen Checkpoint durchzuführen und aktualisiert danach die Dateiheader mit den aktuellen Checkpoint-Informationen. Der Distributed Database Recovery Prozess (RECO) hat nichts mit der Wiederherstellung der Datenbank zu tun, sondern mit der „Rettung“ von nicht vollständig durchgeführten verteilten Transaktionen während eines sogenannten *Two-Phase Commit*. Für die ausführliche Beschreibung sämtlicher Prozesse sei auf die bereits erwähnte Dokumentation in *Oracle Database Reference, Appendix F* verwiesen. Wer sich gern ein Bild mit allen Zusammenhängen und Abhängigkeiten anschauen möchte, findet unter www.oracle.com/technetwork/tutorials/posterfiles-1974103.pdf ein beeindruckendes Diagramm.

Viele Hintergrundprozesse können mehrfach vorhanden sein. In den allermeisten Fällen wählt die Datenbank automatisch die für das jeweilige System geeignete Anzahl solcher Prozesse. Ein Beispiel dafür ist der Log Writer, der ab Oracle12c bei Bedarf bis zu 100 zusätzliche Worker-Prozesse (LGnn) starten kann. Der Database Block Writer kann mittels des Parameters DB_WRITER_PROCESSES auf bis zu 100 Prozesse konfiguriert werden. Die Anpassung des automatisch konfigurierten Wertes ist jedoch nur in Ausnahmefällen notwendig, bei extrem schreibintensiven Datenbanken auf Systemen mit sehr vielen CPUs.

Jeder Prozess braucht natürlich Prozessorzeit und andere Systemressourcen, um seine Arbeiten zu erledigen. Um den Ressourcenverbrauch und viele weitere Statistiken der verschiedenen Prozesse anzuzeigen, stellt Oracle u.a. die V$-View *V$SERVICE_STATS* zur Verfügung.

Das Service-Modell von Oracle enthält den eingebauten Service SYS$BACKGROUND, dem alle Hintergrundprozesse zugeordnet sind. Mit Hilfe der View V$SERVICE_STATS können wir die kumulativen Statistiken der Hintergrundprozesse einsehen:

```
SQL> desc v$service_stats
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SERVICE_NAME_HASH                                  NUMBER
```

```
 SERVICE_NAME                                              VARCHAR2(64)
 STAT_ID                                                   NUMBER
 STAT_NAME                                                 VARCHAR2(64)
 VALUE                                                     NUMBER
 CON_ID                                                    NUMBER

SQL> set pages 40
SQL> col service_name for a20
SQL> col stat_name for a35
SQL> break on service_name
SQL> select service_name, stat_name, value
  2  from v$service_stats
  3  where service_name = 'SYS$BACKGROUND'
  4  order by stat_name;

SERVICE_NAME         STAT_NAME                             VALUE
-------------------- ---------------------------------- ----------
SYS$BACKGROUND       DB CPU                                 26994
                     DB time                                 6528
                     application wait time                 123254
                     cluster wait time                          0
                     concurrency wait time                3922469
                     db block changes                       10844
                     execute count                              0
                     ...                                      ...
                     logons cumulative                        150
                     opened cursors cumulative              61178
                     parse count (total)                        0
                     parse time elapsed                         0
                     physical reads                          5539
                     physical writes                          813
                     redo size                            2528680
                     session cursor cache hits              57841
                     session logical reads                 498338
                     sql execute elapsed time                   0
                     user I/O wait time                    215193
                     user calls                                 2
                     user commits                             339
                     user rollbacks                             1
                     workarea executions - multipass            0
                     workarea executions - onepass              1
                     workarea executions - optimal           9526
```

Die Spalte Value zeigt die Werte je nach Metrik in Mikrosekunden bzw. als Zählerstände. Die gleichen Statistiken gibt es für alle selbstdefinierten Services sowie den standardmäßigen Service SYS$USERS. Damit lässt sich der Ressourcenverbrauch der verschiedenen Services vergleichen:

```
SQL> break on stat_name skip 1
SQL> select stat_name, service_name, value
  2  from v$service_stats
  3  order by stat_name, service_name

STAT_NAME                      SERVICE_NAME             VALUE
------------------------------ -------------------- ----------
DB CPU                         SYS$BACKGROUND            26994
                               SYS$USERS               1590731
```

```
DB time                          SYS$BACKGROUND            6528
                                 SYS$USERS               2725734
...                              ...                        ...
```

Oracle12c Multithreaded Execution

Weiter oben wurde vereinbart, die Begriffe Prozess und Thread synonym zu verwenden. Diese Vereinbarung wird für diesen Abschnitt außer Kraft gesetzt.

Unter Windows findet man genau einen Prozess für die Oracle Datenbank, oracle.exe. Sämtliche Server- und Hintergrundprozesse sind als Threads des Hauptprozesses implementiert: Sie können zwar innerhalb der Instanz mithilfe der V$-Views wie V$PROCESS gesehen werden, aber auf der Betriebssystemebene findet man nur einen Prozess. Unter Unix und Linux dagegen ist Oracle eine Multiprozessanwendung, und die Ausgabe der auf der Maschine laufenden Oracle Prozesse kann mitunter sehr lang und unübersichtlich werden. Hier die Ausgabe der Prozesse einer einzelnen Instanz mit minimaler Konfiguration:

```
[emattila@ora12centos2 ~]$ ps -eaf | grep DOAG | grep -v grep
oracle   11901     1  0 05:51 ?        00:00:00 ora_pmon_DOAG
oracle   11903     1  0 05:51 ?        00:00:00 ora_psp0_DOAG
oracle   11905     1  1 05:51 ?        00:00:01 ora_vktm_DOAG
oracle   11909     1  0 05:51 ?        00:00:00 ora_gen0_DOAG
oracle   11911     1  0 05:51 ?        00:00:00 ora_mman_DOAG
oracle   11915     1  0 05:51 ?        00:00:00 ora_diag_DOAG
oracle   11917     1  0 05:51 ?        00:00:00 ora_dbrm_DOAG
oracle   11919     1  0 05:51 ?        00:00:00 ora_vkrm_DOAG
oracle   11921     1  0 05:51 ?        00:00:00 ora_dia0_DOAG
oracle   11923     1  0 05:51 ?        00:00:00 ora_dbw0_DOAG
oracle   11925     1  0 05:51 ?        00:00:00 ora_lgwr_DOAG
oracle   11927     1  0 05:51 ?        00:00:00 ora_ckpt_DOAG
oracle   11929     1  0 05:51 ?        00:00:00 ora_lg00_DOAG
oracle   11931     1  0 05:51 ?        00:00:00 ora_smon_DOAG
oracle   11933     1  0 05:51 ?        00:00:00 ora_lg01_DOAG
oracle   11935     1  0 05:51 ?        00:00:00 ora_reco_DOAG
oracle   11937     1  0 05:51 ?        00:00:00 ora_lreg_DOAG
oracle   11939     1  0 05:51 ?        00:00:00 ora_pxmn_DOAG
oracle   11941     1  0 05:51 ?        00:00:01 ora_mmon_DOAG
oracle   11943     1  0 05:51 ?        00:00:00 ora_mmnl_DOAG
oracle   11945     1  0 05:51 ?        00:00:00 ora_d000_DOAG
oracle   11947     1  0 05:51 ?        00:00:00 ora_s000_DOAG
oracle   11963     1  0 05:51 ?        00:00:00 ora_p000_DOAG
oracle   11965     1  0 05:51 ?        00:00:00 ora_p001_DOAG
oracle   11967     1  0 05:51 ?        00:00:00 ora_p002_DOAG
oracle   11969     1  0 05:51 ?        00:00:00 ora_tmon_DOAG
oracle   11971     1  0 05:51 ?        00:00:00 ora_tt00_DOAG
oracle   11973     1  0 05:51 ?        00:00:00 ora_smco_DOAG
oracle   11975     1  0 05:51 ?        00:00:00 ora_w000_DOAG
oracle   11977     1  0 05:51 ?        00:00:00 ora_w001_DOAG
oracle   11979     1  0 05:51 ?        00:00:00 ora_aqpc_DOAG
oracle   11981     1  0 05:51 ?        00:00:00 ora_cjq0_DOAG
oracle   11985     1  0 05:51 ?        00:00:00 ora_p003_DOAG
oracle   11987     1  0 05:51 ?        00:00:00 ora_p004_DOAG
oracle   11989     1  0 05:51 ?        00:00:00 ora_p005_DOAG
```

```
oracle   11991     1  0 05:51 ?        00:00:00 ora_p006_DOAG
oracle   11993     1  0 05:51 ?        00:00:00 ora_p007_DOAG
oracle   11995     1  0 05:51 ?        00:00:00 ora_p008_DOAG
oracle   11997     1  0 05:51 ?        00:00:00 ora_p009_DOAG
oracle   11999     1  0 05:51 ?        00:00:00 ora_p00a_DOAG
oracle   12001     1  0 05:51 ?        00:00:00 ora_p00b_DOAG
oracle   12003     1  0 05:51 ?        00:00:00 ora_p00c_DOAG
oracle   12005     1  0 05:51 ?        00:00:00 ora_p00d_DOAG
oracle   12007     1  0 05:51 ?        00:00:00 ora_p00e_DOAG
oracle   12009     1  0 05:51 ?        00:00:00 ora_p00f_DOAG
oracle   12157     1  0 05:51 ?        00:00:00 ora_qm02_DOAG
oracle   12159     1  0 05:51 ?        00:00:00 ora_qm03_DOAG
oracle   12161     1  0 05:51 ?        00:00:00 ora_q002_DOAG
oracle   12163     1  0 05:51 ?        00:00:00 ora_q003_DOAG
oracle   12165     1  0 05:51 ?        00:00:00 ora_q004_DOAG
```

Etwa 50 Betriebssystemprozesse – ohne eine einzige Session. Je nach Konfiguration können selbstverständlich noch weitere Prozesse zu sehen sein: Wird die Datenbank im ARCHIVELOG-Modus betrieben (wie hoffentlich jede produktive Datenbank!) enthält die Auflistung den ARCn. Handelt es sich um einen RAC, laufen zusätzlich mehrere Lock Management Prozesse, und so weiter. Nicht selten kommt es außerdem vor, dass auf einem Server mehrere Instanzen installiert sind. Wer mehrere Instanzen auf einen Server konsolidiert hat, und zwar ohne die Oracle12c Multitenant Option, sieht sich schnell mit Aberhunderten von Prozessen konfrontiert. Aus Deutschland sind mir Kunden mit weit über 200 Instanzen auf einer Exadata bekannt.

Oracle12c führt, vermutlich besonders im Hinblick auf Exadata, den Multithreaded Execution Modus ein, um Datenbankserver in konsolidierten Umgebungen zu entlasten. Die meisten Prozesse können damit als Threads von wenigen Containerprozessen betrieben werden. Weniger Prozesse bedeutet weniger Context Switches oder zumindest schnellere Switches, da sie zwischen Threads eines Prozesses meist schneller vonstattengehen als zwischen eigenständigen Prozessen.

Der Multithreaded Modus wird mit dem Parameter THREADED_EXECUTION = TRUE | FALSE gesteuert. Nach dem Umsetzen des Parameters muss die Instanz durchgestartet werden.

```
SQL> alter system set threaded_execution = true scope=spfile;
SQL> shutdown immediate
SQL> startup
```

Sieht man sich die Betriebssystemprozesse nach dem Neustart an, ändert sich das Bild gewaltig:

```
[emattila@ora12centos2 ~]$ ps -eaf | grep DOAG | grep -v grep
oracle   31781     1  0 06:49 ?        00:00:00 ora_pmon_DOAG
oracle   31783     1  0 06:49 ?        00:00:00 ora_psp0_DOAG
oracle   31785     1  1 06:49 ?        00:00:22 ora_vktm_DOAG
oracle   31789     1  0 06:49 ?        00:00:01 ora_u004_DOAG
oracle   31795     1  0 06:49 ?        00:00:13 ora_u005_DOAG
oracle   31802     1  0 06:49 ?        00:00:00 ora_dbw0_DOAG
```

Statt 50 werden nur noch sechs Prozesse aufgelistet. Neben bekannten Prozessen wie PMON, PSP0, VKTM und DBW0 sind zwei neue zusehen, U004 und U005. Dies sind Containerprozesse, die sämtliche Threads, die früher als eigenständige Prozesse gelaufen sind, aufnehmen.

In der View V$PROCESS sind alle Prozesse nach wie vor zu sehen. Da aber nun die Oracle Prozesse (sowohl Server- als auch Hintergrundprozesse) als Threads von wenigen Containerprozessen laufen, kann die Spalte SPID nicht mehr zur eindeutigen Identifikation der Prozesse bzw. jetzt Threads verwendet werden. Die View wurde daher in 12c um die Spalten SOSID, STID und EXECUTION_TYPE erweitert: SOSID ist die Kombination der IDs des Containerprozesses und des Threads, STID gibt die Thread-ID an, und EXECUTION_TYPE verrät, ob ein Oracle Prozess auf der Betriebssystemebene als Thread oder eigenständiger Prozess läuft.

```
SQL> select pname, sosid, spid, stid, execution_type
  2  from v$process order by 1

PNAME SOSID           SPID       STID       EXECUTION_
----- --------------- ---------- ---------- ----------
...
CKPT  31789_31804     31789      31804      THREAD
...
DBW0  31802           31802      31802      PROCESS
...
LGWR  31789_31803     31789      31803      THREAD
...
```

Damit auch Anwenderprozesse in Threads ausgeführt werden, muss noch die LISTENER.ORA-Datei angepasst werden, indem die folgende Zeile an das Ende hinzugefügt wird:

```
DEDICATED_THROUGH_BROKER_LISTENER = ON
```

```
SQL> select s.sid, s.serial#, s.username, p.spid, p.stid, p.execution_type
  2  from v$session s, v$process p
  3  where s.PADDR=p.ADDR
  4  and s.username='EMATTILA';


 SID  SERIAL# USERNAME   SPID  STID   EXECUTION_
---- -------- ---------- ----- ------ ----------
  13     8133 EMATTILA   31795 13969  THREAD
```

Möchte man eine Usersession, die in einem Thread ausgeführt wird, gewaltsam beenden, so funktioniert es datenbankseitig genauso wie bisher:

```
SQL> alter system kill session '13,8133';
System altered.
```

Dagegen darf man die altbewährte, betriebssystemseitige Brechstangenmethode mit `kill -9 <Prozess-ID)` nicht mehr verwenden, weil dadurch im Zweifel mehr als nur die gewünschte Session beendet wird.

Ein weiterer Punkt zu beachten bei Multithreaded Execution ist, dass die Authentifizierung durch das Betriebssystem nicht mehr möglich ist. Dies gilt sowohl für die sogenannten OPS$-User als auch für die bequeme Art, sich mit `connect / as sysdba` als SYS in der Datenbank anzumelden. Im normalen Alltagsbetrieb ist das zwar lästig, aber vertretbar – schließlich muss man sich daran gewöhnen, wenn man mit Pluggable Databases arbeitet. Ärgerlicher wird es, wenn sich etablierte

Skripte auf die OS-Authentifizierung verlassen – darunter die mitgelieferten dbstart- und dbshut-Skripte, oder oft auch RMAN-Skripte für die Datensicherung.

Ähnlich wie bei den Shared Server Prozessen wird im Multithreaded Execution die UGA vom Betriebssystem in die SGA verlagert. Dies, sowie die Tatsache, dass für wesentlich weniger Prozesse eine PGA allokiert werden muss, kann zu erheblichen Ersparnissen im Memoryverbrauch führen, andererseits muss für die SGA wiederum mehr Speicher reserviert werden. Kandidaten für dieses Feature sind vor allem Systeme, wo viele Non-Container Instanzen auf einen großen Server konsolidiert werden sollen und dadurch das Potenzial für Ersparnisse entsprechend hoch ist.

Quellen:

Oracle Dokumentation (Oracle7, Oracle8i, Oracle9i, Oracle10g, Oracle11g, Oracle12c)
Thomas Kyte, Darl Kuhn: Expert Oracle Database Architecture, 2014
Joel Goodman: The Blog from the DBA Classroom (dbatrain.wordpress.com)
asktom.oracle.com, Thomas Kyte, Oracle Corporation

Kontaktadresse:
Eero Mattila
Dell Software GmbH
Im Mediapark 4e
DE-50670 Köln

Telefon: +49 (0) 221-5777 4169
Fax: +49 (0) 221-5777 450
E-Mail eero.mattila@software.dell.com
Internet: http://software.dell.com