Advanced Parallel Features für volatile Workloads und große Datenmengen

Dr.-Ing. Holger Friedrich
sumIT AG
Baden, Schweiz

Schlüsselworte
Performance, Data Warehousing, Workload Management, Parallel Server, Parallel Query, Degree of Parallelism, Auto DOP, Statement Queueing, Tuning, Increasing data volumes.

Einleitung
Typisch für Cloud Computing, Data Warehousing und Konsolidierungsszenarien sind stark variierende Workloads, sich ändernde zu verarbeitende Datenmengen und wechselnde Menge verfügbarer Ressourcen. Oracles 'Advanced Parallel Execution' Features, insbesondere In-Memory Parallel Execution, Automatic Degree of Parallelism (Auto DOP) und Parallel Statement Queuing, können genutzt werden, um zuverlässig hohe Performance und optimale Systemauslastung unter ständig wechselnden Randbedingungen zu gewährleisten. Der Einsatz des Oracle Resource Managers zusammen mit den Advanced Parallel Features ergibt weitere Möglichkeiten zur Verwaltung der beschränkten Systemressourcen zur Bewältigung verschiedenster day-to-day Einsatzszenarien.
Die richtige Kombination verschiedener Datenbankfunktionalitäten und des Ressourcenmanagements versetzt so DBAs und Entwickler in die Lage Systemkonfigurationen und Anwendungen geeignet anzupassen, so dass sie ihren Kunden eine bessere Servicequalität und größere Flexibilität bei gleichzeitig optimierter Systemauslastung und reduzierten Kosten bieten können.

Im folgenden werden zunächst kurz das Workload Management und dessen Problemdimensionen diskutiert. Danach werden Werkzeuge der aktuellen Oracle Datenbankversion vorgestellt, mit denen DBAs und Entwickler das Workload Management bezüglich paralleler Abfrageausführung in den Griff bekommen und zu grossen Teilen automatisiert, erfolgreich bewältigen können.

Dimensionen des Workload Managements
Was ist das grundsätzliche Ziel des Workload Managements einer jeden Datenbank? Bestimmte Dienste sollen den zugreifenden Applikationen und Nutzern in zugesicherter, gleichbleibender Qualität und Leistung über einen langen Zeitraum zuverlässig zur Verfügung gestellt werden. In der heutigen Welt verabredeter Service Level Agreements heisst dies, dass eine Menge gleichzeitiger Datenbankabfragen mit garantierten Antwortzeiten bewältigt werden müssen. Dies muss geleistet werden, obwohl im zeitlichen Verlauf Nutzerzahlen, sowie die Anzahl abgefragter Objekte und Datenmenge schwanken.
Das Problem beinhaltet, insbesondere im Data Warehousing, verschiedene Dimensionen. Unter Anderem sind diese:

- Die Art der Abfragen ändert sich bei klassisch zyklisch beladenen Data Warehouse Systemen beispielsweise abhängig von der Tageszeit. So laufen des nachts DML-Queries zum beladen des DWH. Von diesen werden in der Regel nicht viele gleichzeitig gestartet, jeder dieser Abfragen verarbeitet jedoch grosse Datenmengen. Dagegen werden tagsüber eine weit grössere Zahl von Abfragen gleichzeitig auszuführen sein, von denen jede einzelne aber potentiell weniger Daten zu bewegen hat.
- Die Datenmenge, die in einzelnen Quellstrukturen (z.B. Tabellen) gespeichert ist, ändert sich im Laufe der Zeit.
- Die Anzahl gleichzeitig aktiver Benutzer und deren Bedeutung bzw. die Priorität ihrer Abfragen variiert. So werden zum Monatsende oder Quartalsende bestimmte Nutzergruppen, wie beispielsweise Mitarbeiter des Controlings verstärkt Daten abfragen und sollten dabei

hoch priorisiert oder zumindest mit garantierter Leistung bedient werden.

Natürlich gehören zum vollständigen Workload Management viele Aspekte der Servicequalität, wie zum Beispiel die Anzahl möglicher Verbindungen für Benutzergruppen, maximal zulässige Idle-Time und zugewiesene CPU-Leistung. Gerade im Data Warehousing liegt der Schlüssel zu bester Query-Performance und maximaler Nutzung der Ressourcen aber in der Parallelisierung der Ausführung von Queries, das heisst im zerlegen und koordinierten Ausführen von Abfragen in Teilabfragen, sowie dem anschliessenden zusammenführen der Ergebnisse.

Workload Management und Parallel Execution
Grundsätzlich gibt es zwei Ebenen auf denen mittels Parallel Execution die beste Ressourcennutzung bei gleichzeitig garantierter Servicequalität angestrebt und erreicht werden kann.

1. Queryebene: Effiziente Ressourcennutzung bei der Abarbeitung einzelner Abfragen.
2. Systemebene: Aufteilung und Zuordnung der Systemressourcen zu Gruppen von Abfragen und Nutzern.

Während sich der erste Punkt auf Query Tuning, Ausführungsstrategien für einzelne Queries und alle damit verbundenen Aspekte bezieht, ist die zweite Ebene auf das Gesamtsystem und mit der Zusicherung und auch Beschränkung von Ressourcen für verschiedene Nutzergruppen und Nutzungsszenarien konzentriert. Ein Beispiel sind die Ausführung nächtlicher Ladeprozesse gegenüber dem analytischen Tagesbetrieb. In der Oracle Datenbank werden Werkzeuge und Funktionen angeboten, die beide Ebenen, aufeinander aufbauend bedienen.

Queryebene
Selbstverständlich sind zur effizienten Ausführung einzelner Datenbankabfragen neben einer effizienten Formulierung in SQL auch korrekte Statistiken, performant entworfene Datenbankstrukturen, z.B. Partitionierung und weitere Dinge notwendig. Wie jedoch bereits gesagt entscheidet zu weiten Teilen der sinnvolle und wohl parametrierte Einsatz paralleler Ausführung über die Performanz und Ressourcennutzung einzelner Queries.

Systemebene
Auf der Systemebene gibt es zwei Aspekte zu beachten. Es sind dies:

- Die prinzipielle Betrachtung und Verwaltung der Ausführung mehrerer gleichzeitiger Queries.
- Die Verteilung und Zusicherung der Ressourcen zur parallelen Ausführung von Abfragen auf verschiedene Nutzergruppen.

Die fortgeschrittene Funktionen zur parallelen Ausführung, sowie der Oracle Resource Manager des aktuellen Oracle Enterprise Edition Datenbank Releases stellen die erforderliche Funktionalität, sowohl auf der Queryebene, als auch auf der Systemebene bereit. In den folgenden Abschnitten werden diese detailliert diskutiert.

Advanced Parallel Execution Features
Parallel Execution ist seit seiner Einführung als 'Parallel Server' mit dem Oracle Release 6.2 ein wichtiger Bestandteil der Datenbank. Die Implementierung paralleler Abfrageausführung wurde im Verlaufe der folgenden Releases der Datenbank immer wieder verbessert und erweitert. Auch wurden erste Versuche zum Selbsttuning der Datenbank in Sachen Parallelität und Systemlast eingeführt, die durch die Datenbank-Initialisierungs-Parameter `PARALLEL_AUTOMATIC_TUNING` und `PARALLEL_ADAPTIVE_MULTI_USER` gesteuert wurden. Diese Methoden waren aber nur in sehr begrenzten Szenarien sinnvoll einsetzbar. Deshalb war bis einschliesslich Release 11.1 de facto manuelle Konfiguration der relevanten Parameter die einzige Möglichkeit, um Parallel Execution, für

Abfragen und Objekte einzustellen.

Manuelle Konfiguration paralleler Ausführung
Die klassische, manuelle Konfiguration der parallelen Ausführung geschieht auf drei Ebenen.

1. Systemebene: Verschiedene Datenbank-Initialisierungs-Parameter konfigurieren die prinzipiell verfügbaren Ressourcen.
   - `PARALLEL_MAX_SERVERS`, maximale Anzahl von Parallel Server Prozessen.
   - `PARALLEL_MIN_SERVERS`, minimale Anzahl fortwährend bereit gehaltener laufender Parallel Server Prozesse.
   - `PARALLEL_DEGREE_LIMIT`, maximaler DOP pro Query.
2. Objektebene: Default Parallelitätsgrad bei Zugriff auf das Objekt.
3. Queryebene: `PARALLEL(Objekt, Degree)`-Hint zur Anforderung eines bestimmten Parallelitätsgrades bei der Ausführung.

All diese Konfigurationen sind statischer Art. Es liegt in der Natur der Sache, dass eine Anpassung an dynamische Veränderung in Workload und Datenmenge auf diese Weise kaum möglich ist. Stattdessen ist für Query-Tuning, geänderte Datenmengen und Durchschnittslasten manuelles anpassen einzelner Objektkonfigurationen und Hints in Queries erforderlich. Dies ist für hochdynamische DWH/BI-Anwendungen mit hunderten und tausenden von Objekten und Abfragen natürlich mehr nicht mit vertretbarem Aufwand möglich. Dies umso mehr, falls Analysten und Dashboards beliebige Ad-Hoc-Abfragen stellen können.

Automatische, dynamische Einstellung des Parallelitätsgrades (Auto DOP)
Oracle nahm sich dieses Problems an. Mit Release 11.2 der Datenbank hielten daher einige neue Funktionen zur dynamischen, automatischen Bestimmung des Parallelitätsgrades in die Datenbank einzug. Diese erlauben es Queries mit unspezifiziertem Parallelitätsgrad zu schreiben und es der Datenbank bei jeder Ausführung zu überlassen mit welchem DOP die Abfrage ausgeführt wird.

Schematisch funktioniert diese Funktionalität wie in der folgenden Graphik dargestellt.

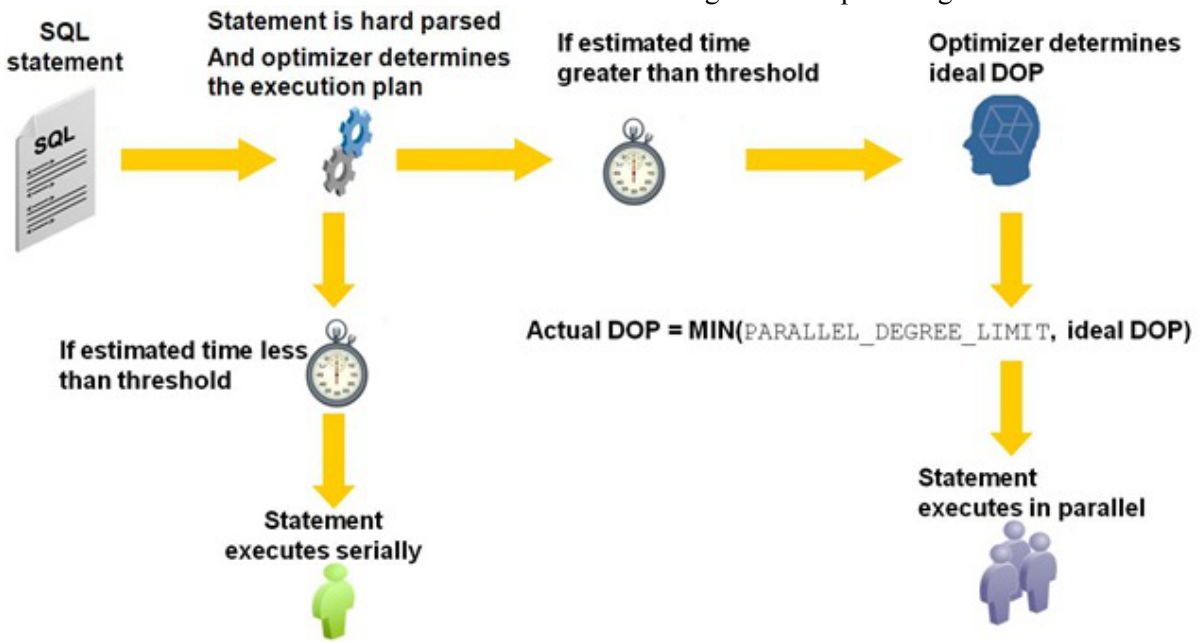

Abb. 1: Algorithmus zur automatischen Bestimmung des DOP

Vorbedingungen
Zwei wichtige Vorbedingungen für deren erfolgreiche Nutzung sind jedoch zu erfüllen.

- Der Datenbank-Server muss Wissen über die ihm zur Verfügung stehende I/O-Leistung besitzen.
- Zu allen an einem Query beteiligten Objekten müssen hinreichend akkurate Statistiken vorliegen.

Die Befriedigung dieser Vorbedingungen ist notwendig, da der genutzte Algorithmus zur dynamischen Bestimmung des Parallelitätsgrades auf der geschätzten zeitlichen Länge erforderlicher Scan-Operationen basiert. Der Algorithmus strebt danach diese zu minimieren, indem mehr parallele Slaves gleichzeitig, die aus dem Storage-Layer zu übertragenden Datenmenge 'ansaugen'.

Um dem System Wissen über die verfügbare I/O-Kapazität zu geben muss das PL/SQL-Paket zur I/O-Kalibrierung ausgeführt werden. Hierfür muss asynchrones I/O für die Datendateien eingestellt sein. Dies kann mit folgendem Query geprüft werden.

```
SELECT NAME,ASYNCH_IO FROM V$DATAFILE F,V$IOSTAT_FILE I
WHERE  F.FILE#=I.FILE_NO
AND    FILETYPE_NAME='DATA FILE';
```

Hernach ist die I/O-Kalibrierungsfunktion, wie folgt auszuführen.

```
SET SERVEROUTPUT ON
DECLARE
  lat  INTEGER;
  iops INTEGER;
  mbps INTEGER;
BEGIN
-- DBMS_RESOURCE_MANAGER.CALIBRATE_IO (<DISKS>, <MAX_LATENCY>, iops, mbps,
lat);
   DBMS_RESOURCE_MANAGER.CALIBRATE_IO (2, 10, iops, mbps, lat);

  DBMS_OUTPUT.PUT_LINE ('max_iops = ' || iops);
  DBMS_OUTPUT.PUT_LINE ('latency  = ' || lat);
  dbms_output.put_line('max_mbps = ' || mbps);
end;
/
```

Während der Kalibrierung sollte keine andere Last auf I/O- und DB-System sein. Die Ergebnisse sind in der Tabelle `DBA_RSRC_IO_CALIBRATE` nachzulesen.

Konfiguration auf System- und Queryebene
Zur Konfiguration auf Systemebene werden zwei Initialisierungsparameter genutzt. Diese können jedoch auch auf Sessionebene dynamisch übersteuert werden. Mit ihnen wird das Auto DOP generell ein- und ausgeschaltet und eine untere Grenze zur Verwendung paralleler Ausführung gesetzt.

- `PARALLEL_DEGREE_POLICY`, stellt den Verarbeitungsmodus ein und steuert damit auch noch das weiter unten beschriebene Workload-Management-Feature des Parallel Statement Queueings.
- `PARALLEL_MIN_TIME_THRESHOLD`, sorgt dafür, dass Queries seriell ausgeführt werden, sollte die geschätzte Ausführungszeit unter dem gesetzten Grenzwert sein.

Zur Konfiguration der parallelen Ausführung auf Queryebene wurde der Parallel-Hint um ein neues Argument erweitert. Mit dem Hint `PARALLEL(AUTO)` bedeutet der Entwickler dem System, dass er

die Wahl des Parallelitätsgrades explizit der Auto-DOP-Funktionalität überlässt. Das System ist dann frei einen DOP mittels oben beschriebener Strategie zwischen 1 (=serieller Ausführung) und `PARALLEL_DEGREE_LIMIT` zu bestimmen und bei der Ausführung zu nutzen.

Vorfahrtsregeln
Auch wenn Auto-DOP systemweit oder auf Sessionebene eingeschaltet ist, bleibt immer noch die Möglichkeit die Funktionalität zu umgehen und weiterhin die konventionelle, manuelle Konfiguration zu nutzen. Hierzu muss der Entwickler lediglich seine Queries mit Parallel-Hints mit fest vorgegebenen Parallelitätsgraden versehen. So wird mit Hint `PARALLEL(<TABLE_NAME>)` die Abfrage mit dem `PARALLEL_DEGREE_LIMIT` des referenzierten Objekts ausgeführt werden. Bei explizierter Angabe eines Grades, z.B. per `PARALLEL(<TABLE_NAME>, DEGREE)` wird dieser verwendet, ohne dass Auto DOP eingreift.

Die Vorfahrtsregeln, nach denen der Datenbankserver seit Release 11.2 entscheidet, ob er selbst den Parallelitätsgrad bestimmt oder den expliziten Vorgaben auf System-, Objekt- oder Queryebene folgt, sind komplex. Die folgende Abbildung gibt diese Regeln wieder.

PRECEDENCE

STMT LVL HINT > OBJ LVL HINT >ALTER SESSION FORCE DOP > INTERNAL_DEGREE_LIMIT (CPU x TPC x INST) / PARALLEL_DEGREE_LIMIT / NO PARALLEL CLAUSE (dml/ddl only) > COMPUTED DOP
NOTE1: ALL PARALLEL CLAUSES ARE IGNORED.
NOTE2: Both PARALLEL_DEGREE_LIMIT and INTERNAL_DEGREE_LIMIT are applied to the computed DOP only

insert into t3 as select * from t1, t2 where t1.c1 = t2.c1;

	STMT LVL HINT	OBJ LVL HINT	DDL/DML Alter session <...>	Query Alter session <...>	DML/DDL PARALLEL?	SELECT PARALLEL?	Dop Computation	DOP used for execution
1	Not Set	Not Set	ENABLE	ENABLE	auto	auto	t1, t2, ddl/dml	computed
2	Not Set	Not Set	DISABLE	ENABLE	N	auto	t1, t2	computed
3	Not Set	Not Set	ENABLE	DISABLE	N	N	Not computed	1
4	Not Set	Not Set	ENABLE	FORCE DEFAULT	auto	Y	t1, t2, ddl/dml	computed & > 1
5	Not Set	Not Set	ENABLE	FORCE DEGREE j	auto	Y	ddl/dml	max(j, computed)
6	Not Set	Not Set	FORCE DEFAULT	ANY EXCEPT FORCE DEGREE j	Y	Y	t1, t2, ddl/dml	computed & >1
7	Not Set	Not Set	FORCE DEFAULT	FORCE DEGREE j	Y	Y	ddl/dml	max(j, computed)
8	Not Set	Not Set	FORCE DEGREE i	ANY	Y	Y	Not computed	i
9	PARALLEL(hint_dop)	ANY	ANY	ANY	Y	Y	Not computed	hint_dop
10	PARALLEL(auto)	ANY	ANY	ANY	auto	auto	t1, t2, ddl/dml	computed
11	PARALLEL	ANY	ANY	ANY	Y	Y	t1, t2, ddl/dml	computed & > 1
12	Not Set	PARALLEL (t1, ohint_dop)	ENABLE	ENABLE	auto	Y	t2, ddl/dml	max(ohint_dop, computed)
13	Not Set	PARALLEL (t1, ohint_dop)	ENABLE	FORCE DEGREE j	auto	Y	ddl/dml	max(ohint_dop, j, computed)
14	Not Set	PARALLEL (t1, ohint_dop)	FORCE DEGREE i	ANY	Y	Y	Not computed	max(ohint_dop, i)
15	Not Set	PARALLEL (t1, ohint_dop)	FORCE DEFAULT	ANY EXCEPT FORCE DEGREE j	Y	Y	t2, ddl/dml	max(ohint_dop, computed(>1))
16	Not Set	PARALLEL (t1, ohint_dop)	FORCE DEFAULT	FORCE DEGREE j	Y	Y	ddl/dml	max(ohint_dop, j, computed(>1))

Abb. 2: Vorfahrtsregeln zur Bestimmung des Parallelitätsgrades ab Oracle Release 11.2

Parallel Statement Queueing (PSQ)
Zwar bringt die Auto-DOP-Funktionalität die lang ersehnte Dynamik und damit Reduktion des Wartungsaufwandes in die Ausführung einzelner Queries, das Problem wachsenden Workloads und von Belastungsspitzen bleibt jedoch davon unberührt. Dieses Problem wird durch ein weiteres Feature adressiert, welches die Datenbank seit Release 11.2 an Bord hat. Es ist dies das sogenannte Parallel Statement Queueing.

Ist der oben bereits erwähnte Parameter `PARALLEL_DEGREE_POLICY` auf `AUTO` eingestellt, ist PSQ aktiv und wird verwendet. Zur Konfiguration des Verhaltens des PSQ wird hauptsächlich ein Initialisierungsparameter verwendet. Es ist dies der Parameter `PARALLEL_SERVER_TARGET`. Er konstituiert den Grenzwert für die maximale Anzahl Parallel Slave Prozesse, von deren errichen an PSQ beginnt aktiv einzugreifen. `PARALLEL_SERVER_TARGET` ist dabei kleiner oder gleich `PARALLEL_MAX_SERVERS` zu wählen.

Die Grundidee ist es alle Queries mit dem für sie optimalen Parallelitätsgrad auszuführen, ohne dabei das System mit zu vielen Parallel Server Prozessen zu überlasten. Hierzu werden Abfragen, die mehr Parallel Slaves benötigen, als momentan bis zum erreichen der festgelegten Maximalanzahl (siehe

PARALLEL_SERVER_TARGET) verfügbar sind in eine Queue eingestellt. Die Queue wirddem nach dem FIFO-Prinzip abgearbeitet. Das dahinterliegende Erfahrungswissen besagt, dass es für DWH/BI-Abfragen allemal besser ist einige Zeit auf die Freigabe ausreichender Parallel Slaves zu warten, als, wie bisher, serialisiert gestartet zu werden und dann in Summe um ein vielfaches länger zur Ausführung zu benötigen.

Weiterhin erlaubt der Parameter PARALLEL_QUEUE_TIMEOUT die Angabe eines Zeitraums nach dem in der Queue auf Ressourcen wartende Prozesse mit einem Timeout-Fehler (ORA-07454) abgebrochen werden.

Durch das Verhältnis der beiden Parameter PARALLEL_DEGREE_LIMIT und PARALLEL_SERVER_TARGET können DBAs bzw. Systemverantwortliche festlegen wie viele Abfragen gleichzeitig mit maximaler Parallelität ausgeführt werden können, bevor es zum Statement Queueing kommt. Hiermit kann also bezüglich vorhandener Systemressourcen eine optimale Einstellung von einzelner Abfrageperformance im Verhältnis zum Workload getroffen werden.

Folgende Graphik verdeutlicht den Ablauf des PSQ.

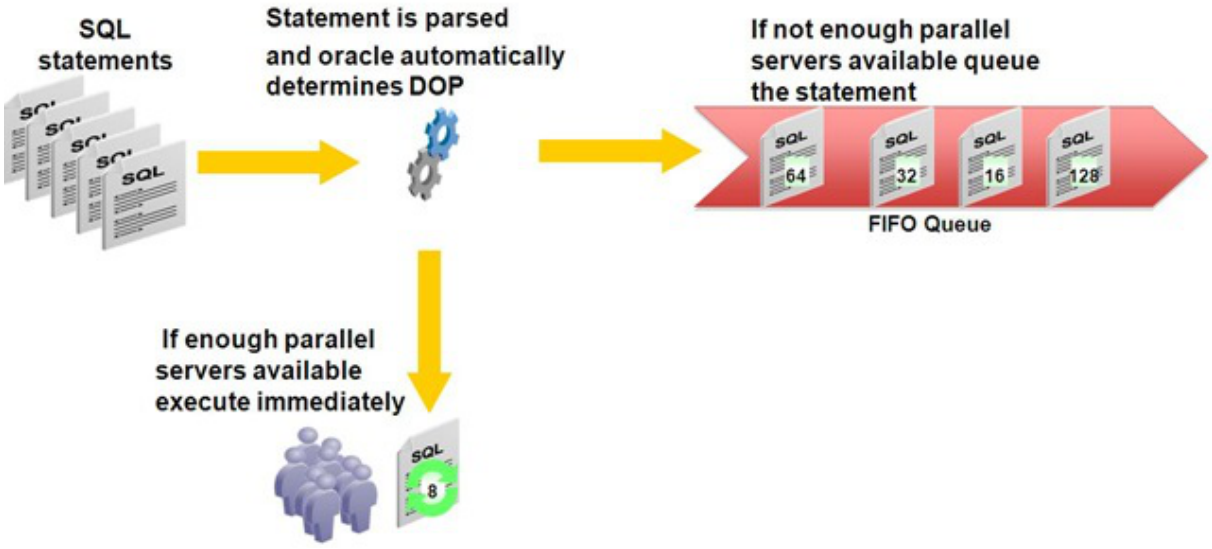

Abb. 3: Schematischer Ablauf des Parallel Statement Queueing ab Oracle Release 11.2

Auto-DOP und PSQ im Oracle Resource Manager (ORM)
Es ist immer ratsam in Umgebungen, die Benutzergruppen mit unterschiedlichen Bedürfnissen und Service Level Agreements dienen den Oracle Resource Manager zum Mangement der Ressourcen einzusetzen. Damit auch verschiedene Workloads unterschiedlicher Nutzergruppen differenziert gemangt werden können, hat Oracle ORM derart erweitert, dass er auch die Konfiguration von Auto DOP und PSQ für einzelne Benutzergruppen ermöglicht.

Hierzu können einige der obigen Parameter ressourcegruppenspezifisch eingestellt werden. Zudem ist insbesondere ein neuer Parameter hinzugekommen. Die auch ausserhalb des ORM verfügbaren Parameter sind die Folgenden.

- Für Auto DOP ist PARALLEL_DEGREE_LIMIT pro Ressourcengruppe einzustellen.
- Für PSQ kann der PARALLEL_QUEUE_TIMEOUT spezifisch je Gruppe eingestellt werden.

Zusätzlich ist für PSQ der neue Parameter `PARALLEL_TARGET_PERCENTAGE` im ORM verfügbar. Er dient zur Berechnung der maximal verfügbaren Parallel Slaves pro Ressourcengruppe, im Falle dass das `PARALLEL_DEGREE_LIMIT` systemweit ausgeschöpft werden sollte. Komplettiert wird das Paket durch eigene FIFO Queues pro Ressourcengruppe. So kann das Queueing gruppenspezifisch stattfinden. Abfragen einer höher priorisierten Nutzergruppe werden dann nicht länger als notwendig von Statements einer niedriger priorisierten Gruppe blockieren.

Folgerung
Durch die Kombination aus Auto DOP, Parallel Statement Queueing und deren Berücksichtigung im Oracle Resource Manager ab dem Datenbanrelease 11.2 ist Oracle ein weiterer grosser Schritt in die Richtung einer 'self managed' und 'self tuning' Datenbank gelungen.

Natürlich müssen DBAs und Systemverantwortliche weiterhin die Grundkonfigurationen durchführen und die Ausnutzung und Performance des Systems periodisch überwachen. Die Kombination der vorgestellten Datenbank Feature erlaubt jedoch zweifelsohne, und nebenbei gesagt vollkommen geräuschlos und ausgesprochen bugarm das effiziente Management volatiler Workloads in Systemen mit grossen und stetig wachsenden Datenmengen.

Kontaktadresse:
Dr.-Ing. Holger Friedrich
sumIT AG
Täfernstrasse 28
CH-5405 Baden

Telefon:	+41 (0) 56-470 2500
Fax:	+41 (0) 56-470 2505
E-Mail	Holger.Friedrich@sumit.ch
Internet:	www.sumit.ch