Speicherverwaltung
–
leicht gemacht

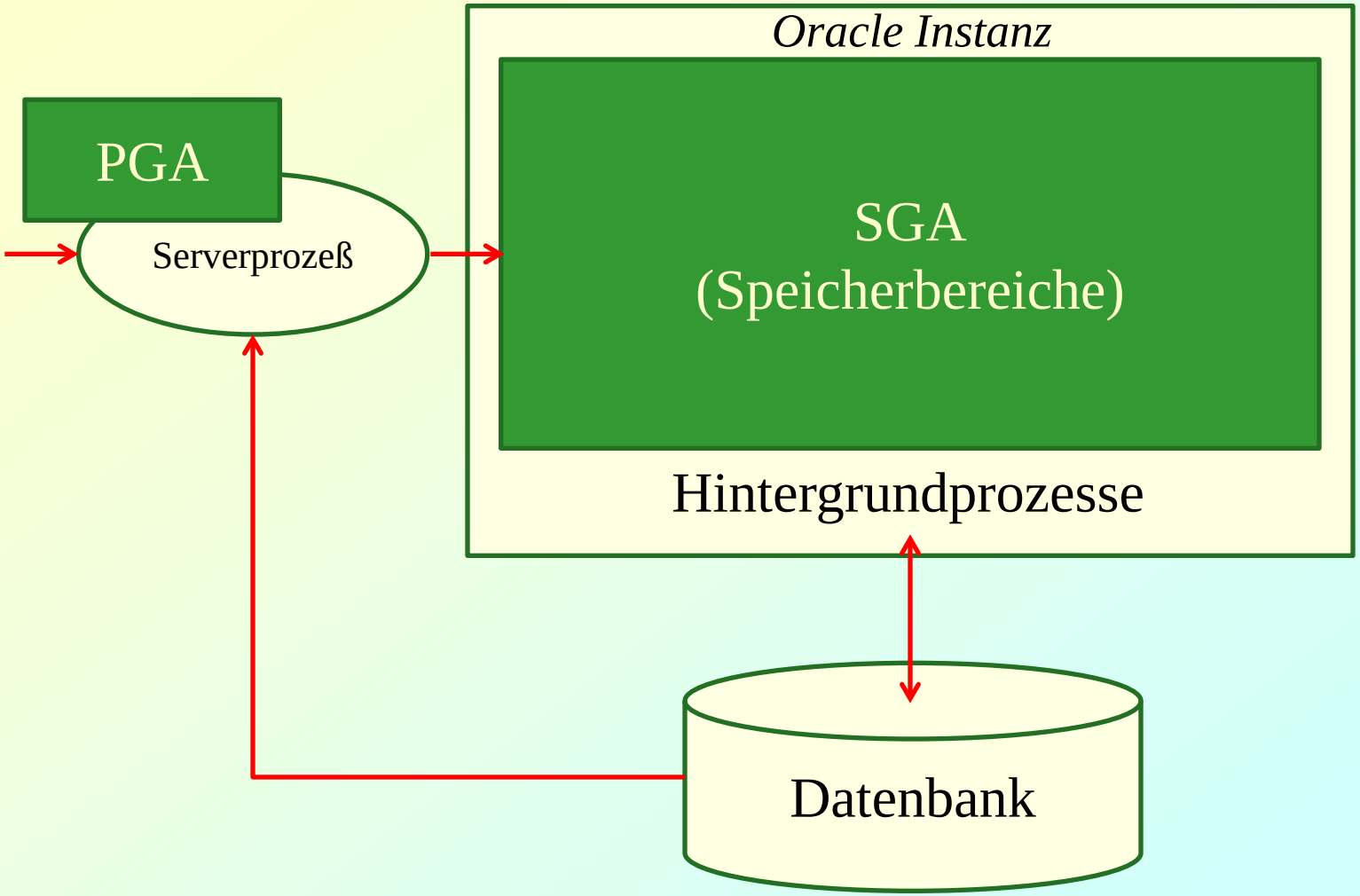
Dr. Frank Haney

DOAG-Konferenz 2011

Inhalt

- Speicherbereiche im Überblick
- Speicherverwaltung mit Initialisierungsparametern
- Automatisches PGA-Management (Oracle 9i)
- Automatisches Shared Memomory Mangement (Oracle 10g)
- Automatisches Memory Mamagement (Oracle 11g)
- Speicherverwaltung aus der Sicht von Linux
- Verwaltung großer SGAs mit Hugepages

Speicherallokierung in der Oracle-Architektur



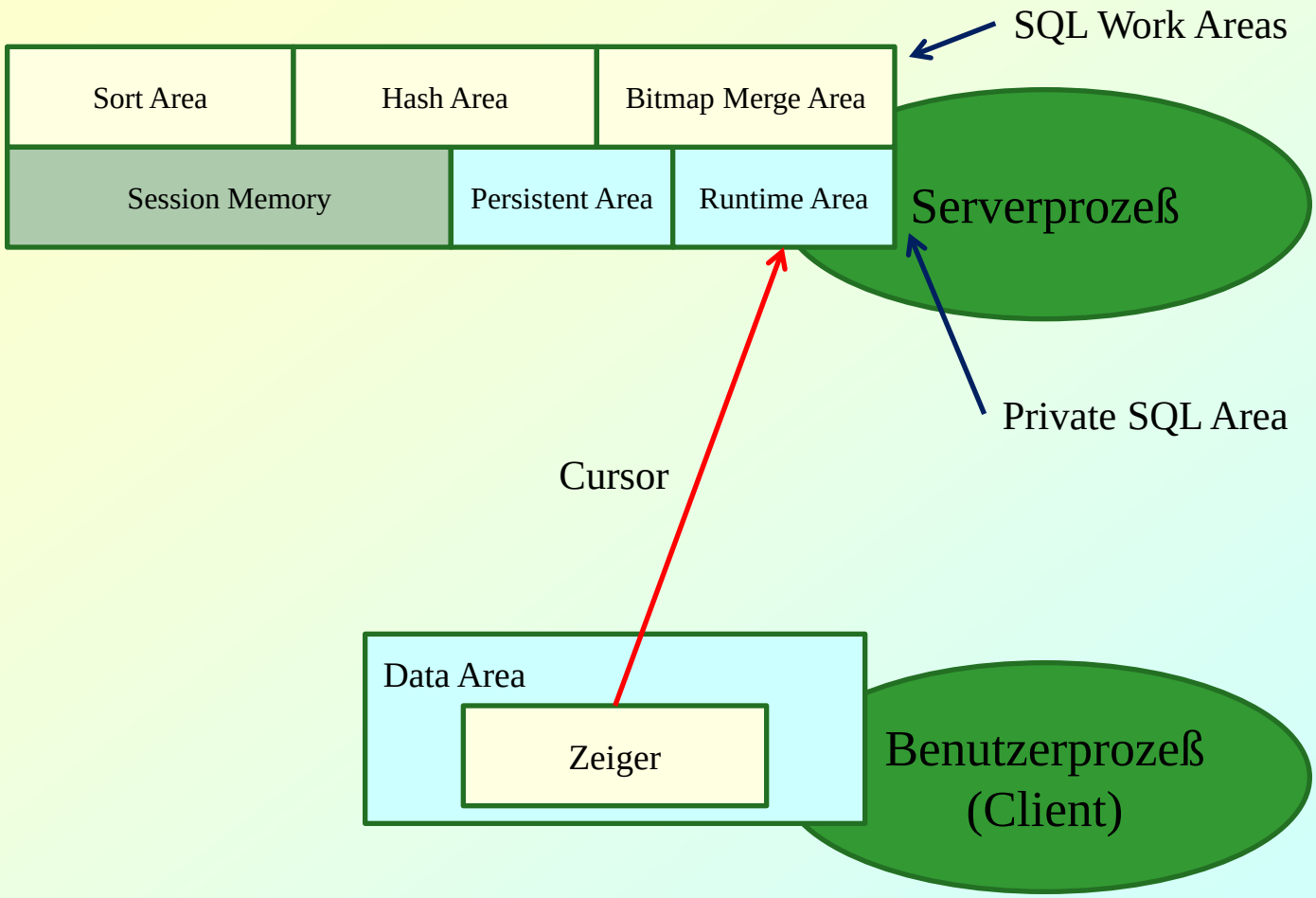
Bestandteile der PGA

Die Program Global Area (PGA) dient für:

- Session-Informationen
- Cursor-Informationen (Private SQL Area)
  - Runtime Area (z.B. Cursor-Status)
  - Persistent Area (z.B. Bindevariablen)
- SQL Work Areas – Arbeitsbereiche für die SQL-Ausführung mit
  - Sortierbereich (Sort Area)
  - Hash-Join-Bereich (Hash Area)
  - Bitmap-Erstellungsbereich (Bitmap Create Area)
  - Bitmap-Zusammenführungsbereich (Bitmap Merge Area)

Bei dediziertem Server wird die PGA für jeden Serverprozeß separat aus dem freien Speicher genommen.
Achtung: Auch die Hintergrundprozesse allokieren PGA.

Struktur der PGA



Verwaltung der PGA

Manuelle PGA-Verwaltung (nicht empfohlen): Parameter setzen absolute Obergrenzen für die Bereiche der SQL Work Areas. Diese Parameter sind dynamisch änderbar.

`SORT_AREA_SIZE`
`SORT_AREA_RETAINED_SIZE`
`HASH_AREA_SIZE`
`BITMAP_MERGE_AREA_SIZE`

Automatische PGA-Verwaltung

`PGA_AGGREGATE_TARGET` (Standard 20% von SGA_TARGET, wenn gesetzt)

`WORK_AREA_POLICY` (Standard `AUTO`)

Achtung: PGA_AGGREGATE_TARGET stellt keine absolute Obergrenze für die Summe der PGAs dar. (Stichwort PL_SQL-Tabellen)

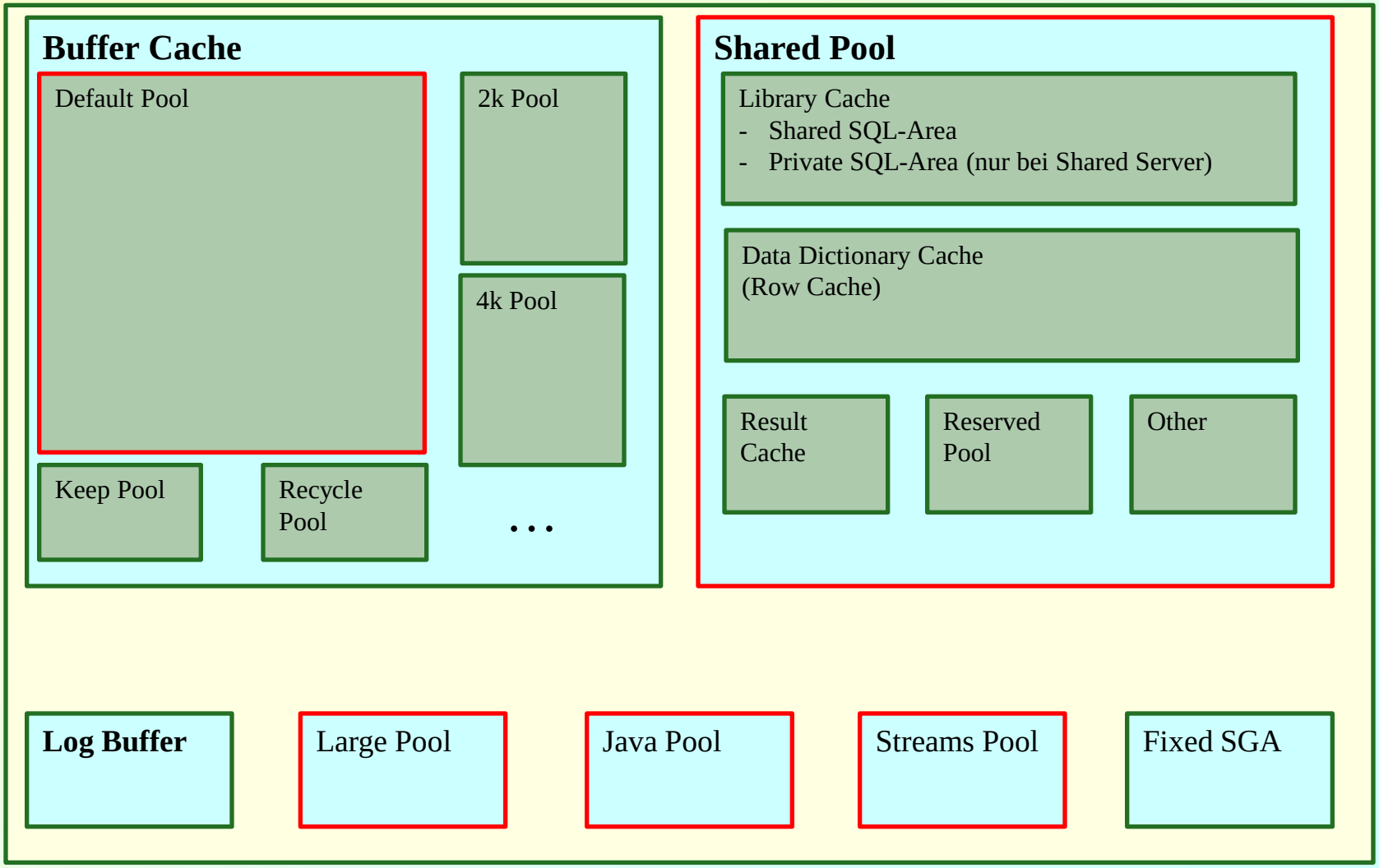
Dimensionierung und Überwachung der PGA

- Einschätzung von Trefferquote und Überallokierung von mit dem PGA-Ratgeber (bei Veränderungen in Bezug auf den aktuellen Wert von PGA_AGGREGATE_TARGET)
  V$PGA_TARGET_ADVICE
- Detailliertere Informationen über Veränderungen der Workarea-Statistik bei Änderung von PGA_AGGREGASTE_TARGET
  V$PGA_TARGET_ADVICE_HISTOGRAM
- Detailliertere Informationen über die von SQL-Cursorn benutzten Arbeitsbereiche
  V$SQL_WORKAREA, V$PROCESS_MEMORY,
- Informationen über momentan vom System allokierte Arbeitsbereiche
  V$SQL_WORKAREA_ACTIVE
- Zusammengefaßte Statistiken über die Verwendung der Arbeitsbereiche Arbeitsbereiche (Optimal, Onepass, Multipass)
  V$SQL_WORKAREA_HISTOGRAM
- PGA-Statistiken
  V$PGASTAT

PGA-Statistiken

NAME	VALUE
aggregate PGA target parameter	268435456
aggregate PGA auto target	216935424
global memory bound	53686272
total PGA inuse	27392000
total PGA allocated	35709952
maximum PGA allocated	104214528
total freeable PGA memory	0
process count	32
max processes count	38
PGA memory freed back to OS	0
total PGA used for auto workareas	0
maximum PGA used for auto workareas	3772416
total PGA used for manual workareas	0
maximum PGA used for manual workareas	0
over allocation count	0
bytes processed	324299776
extra bytes read/written	0
cache hit percentage	100
recompute count (total)	9745

Struktur der SGA



„Pflicht“-Bestandteile der SGA im Überblick

SGA-Bestandteil	Funktion
Database Buffer Cache	Speicherbereich für Blöcke, die vom Server-Prozeß aus den Datendateien gelesen und vom Database Writer asynchron auf Platte geschrieben werden (LRU-Mechanismus)
Redo Log Buffer	Speichert Before-Image- und After-Image-Kopien von geänderten Daten, der Log Writer schreibt die Änderungen in die Redo Log Dateien, wird zyklisch wiederverwendet
Shared Pool	Speichert geparste Versionen von SQL-Anweisungen, PL/SQL-Prozeduren und Data Dictionary-Informationen (LRU-Mechanismus)
Reserved Shared Pool	Teil des Shared Pool, der für große zusammenhängende Speicheranforderungen dient
Java Pool	Repräsentation der Java-Methoden, -Klassen und -Objekte im Speicher
Fixed SGA	Interne Metadaten (X$-Tabellen als Basis der V$-Views)
Zusätzliche Metadaten	

Optionale SGA-Bestandteile

SGA-Bestandteil	Funktion
Large Pool	Optionaler Speicherbereich für I/O des RMAN, parallelen I/O etc. (kein LRU-Mechanismus)
Streams Pool	Wird von AQ bzw. Oracle Streams verwendet
Recycle Buffer Cache	Für Blöcke, die schnell wieder aus dem Cache verschwinden können (LRU-Mechanismus)
Keep Buffer Cache	Für Blöcke, die möglichst im Cache bleiben sollen (LRU-Mechanismus)
nk Buffer Cache	Buffer Cache für Tablespaces abweichender Blockgröße
Result Cache	Speicherung der Resultate von SQL-Anweisungen (Bereich im Shared Pool) – neu in 11*g*
Shared IO Pool	Speicherung von Securefiles (Bereich im Buffer Cache) – neu in 11*g*
Smart Flash Cache	Erweiterung des Buffer Cache auf einer Flash Disk – neu in 11*g* Release 2

Verwaltung der SGA – Überblick

SGA-Bestandteil	Parameter	Standard
Database Buffer Cache	DB_CACHE_SIZE	48 MB oder 4 MB*Anzahl der CPUs
Redo Log Buffer	LOG_BUFFER (statisch!)	5-32 MB in Abhängigkeit von SGA, Anzahl der CPUs und OS
Shared Pool	SHARED_POOL_SIZE	64 MB bei 32-bit 128 MB bei 64-bit
Reserved Shared Pool	SHARED_POOL_RESERVED_SIZE	5% von SHARED_POOL_SIZE Maximal 50%
Java Pool	JAVA_POOL_SIZE	24 MB
Large Pool	LARGE_POOL_SIZE	0
Streams Pool	STREAMS_POOL_SIZE	0
Recycle Buffer Cache	DB_RECYCLE_CACHE_SIZE	0
Keep Buffer Cache	DB_KEEP_CACHE_SIZE	0
nk Buffer Cache	DB_*n*K_CACHE_SIZE	0
Result Cache	RESULT_CACHE_MAX_SIZE	Abhängig von anderen Parametern
Smart Flash Cache	DB_FLASH_CACHE_SIZE	0

Neue SGA-Bestandteile in Oracle 11*g*

Result Cache

- Speichert Ergebnisse häufig ausgeführter SELECT-Anweisungen
- Gut bei gro0eßen Datenmengen und kleinem Result Set
  - `RESULT_CACHE_MAX_SIZE` – Größe, max. 75% des Shared Pool
  - `RESULT_CACHE_MAX_RESULT` – Größe des einzelnen Resultats (Default 5%)
  - `RESULT_CACHE_MODE` – `AUTO`, `MANUAL` oder `FORCE`
- Hint [`NO_RESULT_CACHE`] überschreibt Einstellungen auf Anweisungsebene.

Smart Cache Flash

- Wird gesteuert mit den Parametern
  - `DB_FLASH_CACHE_FILE` – Dateiname oder Diskgroup
  - `DB_FLASH_CACHE_SIZE` – Größe des Bereichs (Keine automatische Verwaltung)
- Übernimmt Blöcke, die aus dem Buffer Cache ausaltern (nur Standard-Blockgröße)
- Buffer Cache enthält LRU-Listen für den Smart Flash Cache (Keep- und Default-Pool)
- DBWR schreibt in den, Vordergrundprozeß liest aus dem Flash Cache
- Verwendung wird mit der Storage-Klausel FLASH_CACHE KEEP|NONE|DEFAULT angegeben.

Shared IO Pool

- Speichert Securefiles (LOBs), wenn diese mit der NOCACHE-Klausel angelegt werden.
- Subpool des Buffer Cache – wird gesteuert mit dem verborgenen Parametern
  - `_SHARED_IO_POOL_SIZE` – Größe des Bereichs (Automatische Verwaltung)

Überwachung der SGA

- Überblick
  `V$SGA`
- Größe der einzelnen Bereiche einschließlich Granule Size und freiem Speicher
  `V$SGAINFO`
- Detailliertere Statistiken über die SGA-Verwendung
  `V$SGASTAT`
- Effekt einer Vergößerung oder Verkleinerung der SGA (SGA-Advisor)
  `V$SGA_TARGET_ADVICE`
- Ratgeber für einzelne Pools
  `V$DB_CACHE_ADVICE, V$SHARED_POOL_ADVICE,`
  `V$JAVA_POOL_ADVICE, V$STREAMS_POOL_ADVICE`
- Statistiken für die verschiedenen Buffer Pools
  `V$BUFFER_POOL`
- Statistiken für den Shared Pool
  `V$LIBRARYCACHE, V$ROWCACHE`

Automatisches Shared Memory Management (ASMM)

Seit Oracle 10*g* kann der Oracle Server innerhalb gesetzter Obergrenzen zwischen bestimmten Bestandteilen der SGA umallokieren. Verantwortlich dafür sind die Parameter SGA_MAX_SIZE (statisch) und SGA_TARGET (dynamisch mit SGA_MAX_SIZE als Obergrenze). Die dynamisch verwaltbaren Bereiche sind:

- Buffer Cache
- Shared Pool
- Large Pool
- Java Pool
- Streams Pool

- Setzen von SGA_TARGET schaltet das ASMM ein. Die automatisch verwalteten Bereiche stehen dann standardmäßig auf 0.
- Setzen der einzelnen Parameter setzt für diese Minimalwerte, die von ASSM nicht unterschritten werden dürfen.
- Der automatisch verwaltbare Teil der SGA wir durch die nicht dem ASSM unterliegenden eingeschränkt.
- Aktuelle Werte werden in den SPFile geschrieben (doppelter Unterstrich vor Parameternamen)

Überwachung von ASSM

- Informationen über aktuellen, minimalen und maximalen Wert der automatisch verwalteten Komponenten sowie Anzahl und Typ der Operationen
  `V$SGA_DYNAMIC_COMPONENTS`
- Informationen über Speicher, der für Umallokierungen zur Verfügung steht
  `V$SGA_DYNAMIC_FREE_MEMORY`
- Informationen über stattgefundene und laufende Operationen zur Umallokierung
  `V$SGA_RESIZE_OPS, V$SGA_CURRENT_RESIZE_OPS`

Für die automatische Überwachung und Umallokierung ist der Hintergrundprozeß MMAN (Memory Manager) zuständig. Die Tätigkeit des MMAN läßt sich mit verborgenen Parametern steuern. (z.B. `_MEMORY_BROKER_STAT_INTERVAL`)

Fehlgeschlagene Operationen liefern den Fehler `ORA-4031`. Gründe können sein:

- Speicher reicht nicht aus, um die nötige Anzahl von Granules zu allokieren.
- Speicher ist zu fragmnetiert zum Umallokieren

Automatisches Memory Management (AMM)

Seit Oracle 11*g* kann der Oracle Server innerhalb gesetzter Obergrenzen auch zwischen SGA und PGA umallokieren. Verantwortlich dafür sind die Parameter `MEMORY_MAX_TARGET` (statisch) und `MEMORY_TARGET` (dynamisch mit `MEMORY_MAX_TARGET` als Obergrenze).

- Setzen von `MEMORY_TARGET` schaltet das AMM ein. `SGA_TARGET` und PGA_AGGREGATE_TARGET stehen dann standardmäßig auf 0.
- Setzen der einzelnen Parameter setzt für diese Minimalwerte, die von AMM nicht unterschritten werden dürfen.
- Auch hier werden die momentanen Werte in den SPFile geschrieben (doppelter Unterstrich vor Parameternamen)
- Die aktuelle Allokierung kann größer als `MEMORY_TARGET` sein. Der Grund ist der gleiche, aus dem `PGA_AGGRGATE_TARGET` keine absolute Obergrenze für die Summe aller PGAs ist, sondern nur ein Richtwert. (Stichwort PL/SQL-Tabellen)

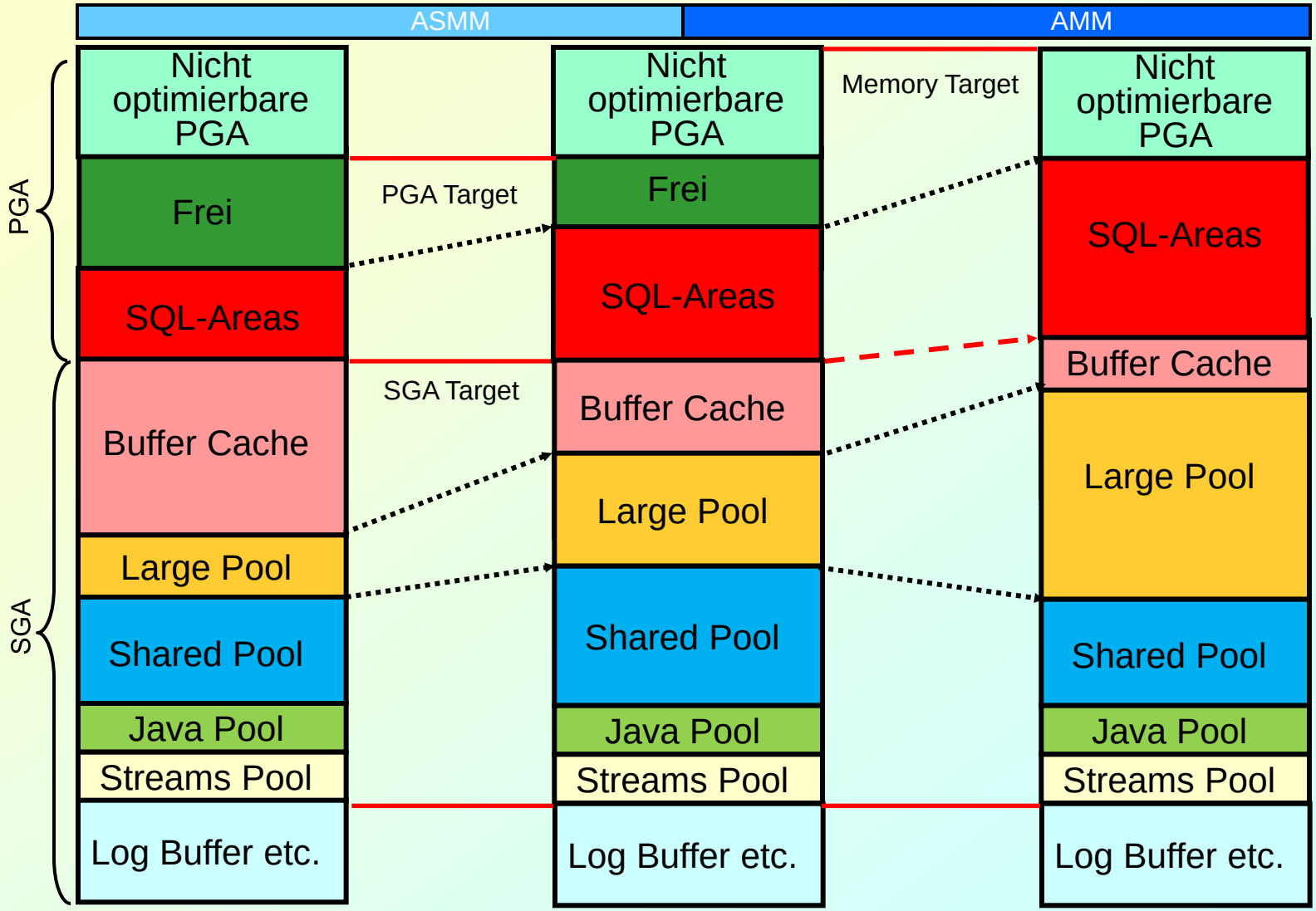
Überwachung von AMM

- Informationen über aktuellen, minimalen und maximalen Wert der automatisch verwalteten Komponenten sowie Anzahl und Typ der Operationen
  `V$MEMORY_DYNAMIC_COMPONENTS`
- Informationen über stattgefundene und laufende Umallokierungen
  `V$MEMORY_RESIZE_OPS`, `V$MEMORY_CURRENT_RESIZE_OPS`
- Effekt einer Vergößerung oder Verkleinerung des Oracle zur Verfügung stehenden Speichers
  `V$MEMORY_TARGET_ADVISE`

AMM-Parameter im SPFILE

```
ora11r2.__java_pool_size=8388608
ora11r2.__large_pool_size=4194304
ora11r2.__pga_aggregate_target=301989888
ora11r2.__sga_target=536870912
ora11r2.__shared_io_pool_size=75497472
ora11r2.__shared_pool_size=314572800
ora11r2.__streams_pool_size=4194304
*.memory_target=838860800
```

AMM – Zusammenfassung



19

SGA-Management aus Linux-Sicht (Kernel-Parameter)

- **SHMMAX** definiert die maximale Größe eines einzelnen Shared-Memory-Segments in Byte. Dieser Parameter sollte mindestens so groß wie die größte SGA im System sein. Empfohlen werden von Oracle für 32bit-Linux 4 GB -1 oder die Hälfte des zur Verfügung stehenden Arbeitsspeichers, je nachdem, was unter den gegebenen Umständen kleiner ist. Für 64bit-Linux empfiehlt Oracle generell die Hälfte des zur Verfügung stehenden Arbeitsspeichers.
- **SHMMNI** bezeichnet die maximale Anzahl von Shared-Memory-Segmenten systemweit. Der Standard von 4096 ist hier ausreichend.
- **SHMALL** gibt die Gesamtmenge an Shared Memory in Speicherseiten an und sollte wenigstens SHMMAX/PAGE_SIZE, aufgerundet auf eine ganze Zahl, betragen.

SGA-Allokierung

Die **SGA-Allokierung** kann in drei Modi erfolgen, die vom System in der gegebenen Reihenfolge versucht werden:

1. Ein einziges Shared Memory Segment
2. Mehrere zusammenhängende Segmente
3. Mehrere im Speicher verteilte (unzusammenhängende) Segmente

Tools für die Diagnose: `$ORACLE_HOME/bin/sysresv` oder `SQL>oradebug ipc`
Achtung: Bei zu kleinem SHMMAX wird die SGA in mehreren Segmenten allokier.
Beispiel:

SGA-Größe 603 MB kernel.shmmax = 209715200

`$ORACLE_HOME/bin/sysresv` zeigt nach dem Neustart der Instanz 4 Shared-Memory-Segmente:

```
IPC Resources for ORACLE_SID "ora11r2" :
Shared Memory:
ID         KEY
720910     0x00000000
753680     0x00000000
786449     0x00000000
819218     0x052a0228
```

Dagegen wird bei `kernel.shmmax = 209715200` nur ein Segment allokiert.

Motivation für die Verwendung von Hugepages

Problem: Die betriebssystemseitige Verwaltung großer SGAs kann zu einem großen Overhead führen. Das soll die folgende Beispielrechnung zeigen:

1. Jeder Prozeß (Vordergrund- und Hintergrundprozesse) pflegt seine eigene Page Table.
2. Pro Speicherseite von 4kB (Standard unter Linux) werden dafür 12 bit benötigt.
3. Bei 100 Prozessen und 24 GB SGA summieren sich die Page Tables auf 900 MB.

Das Problem wird mit wachsender Zahl der Sessions und steigender SGA-Größe immer gravierender, ist also vor allem auf 64-bit-Implementierungen relevant.

Lösungsmöglichkeiten:

- Begrenzung der Sessionzahl durch Shared Server oder ein Profil – In der Praxis ist das in den wenigsten Fällen sinnvoll.
- Einsatz von Hugepages (Größe bei Linux x86-64 2 MB)

Vorteile von Hugepages

- Der für die Page Tables total benötigte Speicher reduziert sich drastisch. Im vorstehenden Beispiel mit 24 GB SGA und 100 Sessions kämen bei 2-MB-Seiten nur noch rund 1,76 MB zusammen.
- Ein Nebeneffekt ist, daß der Paging-Mechanismus von Linux nur mit den üblichen 4-kB-Seiten arbeitet. Eine Hugepage kann also nicht ausgelagert werden. Damit läßt sich wirksam verhindern, daß SGA ausgelagert wird.

Konfiguration von Hugepages

Konfiguration:

1. Ermittlung der benötigten Anzahl von Pages für die Summe aller SGAs (Dafür gibt es in Support Note 401749.1 ein Skript.)
2. Einstellung des kernel-Parameters vm.nr_hugepages auf den ermittelten Wert (Bei SLES in die Datei /etc/sysconfig/oracle übernehmen)
3. Parameter `USE_LARGE_PAGES=ONLY` setzen (11.2.0.2) – Allokierungsprobleme werden in die alert.log geschrieben.
4. Neustart des Servers

Überprüfung:

- Abfragen von /proc/meminfo
- Resultat

```
HugePages_Total:   302          -Anzahl der Seiten im Pool
HugePages_Free:    159          -Anzahl der Momentan nicht benutzten Seiten
HugePages_Rsvd:    56           -Anzahl der für Applikationen reservierten Seiten
Hugepagesize:      2048 kB
```

`HugePages_Free – HugePages_Reserved =` Anzahl der überallokierten Seiten

Probleme bei der Verwendung von Hugepages

1. Wenn nur eine Seite fehlt, werden keine Hugepages verwendet. Die SGA wird komplett aus dem restlichen Speicher allokiert, sofern verfügbar.
2. Verwendung ist von anderen Einstellungen abhängig (z.B. memlock)

```
****************** Huge Pages Information *****************
Huge Pages memory pool detected (total: 302 free: 302)
Memlock limit too small: 67108864 to accommodate segment size: 417333248
Huge Pages allocation failed (free: 302 required: 199)
Startup will fail as use_large_pages is set to "ONLY"
```

3. Hugepages und AMM sind inkompatibel:

```
Huge Pages are not compatible with specified SGA memory parameters
use_large_pages = "ONLY" cannot be used with memory_target,
memory_max_target, or use_indirect_data_buffers parameters
Huge Pages are compatible with sga_target and shared_pool_size
```

   Bei AMM und USE_LARGE_PAGES=TRUE werden Hugepages ignoriert.
4. Das Setzen des Parameters USE_INDIRECT_DATA_BUFFERS=TRUE, wie es für die Allokierung großer SGAs (VLM) unter 32-bit-Linux notwendig ist, verhindert die Verwendung von Hugepages für den Buffer Cache. (siehe Support Note 829850.1) Die anderen SGA-Bestandteile können Hugepages nutzen.

Q&A

Literatur:

Oracle® Concepts
Oracle® Reference
Oracle® Administrator's Guide
Oracle® Performance Tuning Guide

Dr. Frank Haney
info@haney.it
Tel.: 03641-210224