Gestiunea memoriei în Oracle

DBA-ul de servici: Optimizarea alocării memoriei în Oracle

– Monica Ibănescu

Oracle utilizează zone de memorie partajată și procese de background pentru a gestiona structurile sale de memorie și de fișiere.

Dacă ar fi să citești un capitol dintr-o carte, care ar fi cel mai rapid mod de transmitere a informațiilor altcuiva? Ar putea ca cealaltă persoană să citească și ea capitolul, dar ar fi mai rapid dacă ai putea ține în memorie toate informațiile și apoi să-i transmiți acele informații din memorie celei de-a doua persoane.

System Global Area (SGA), într-o bază de date Oracle, are același scop-facilitează transferul informației între utilizatori. SGA stochează, de asemenea, cele mai des cerute informații structurale despre baza de date. Aceasta este împărțită în mai multe secțiuni: *Data Block Buffers, Redo Log Buffers* și *Shared SQL Pool.*

Data Block Buffer Cache (sau *Buffer Cache*) este o zonă din SGA în care sunt stocate blocurile de date care sunt citite din segmentele de date ale bazei de date, cum ar fi tabele, indecși, clustere. Mărimea *Data Block Buffer Cache* este dată de parametrul `DB_BLOCK_BUFFERS` (exprimat în număr de blocuri ale bazei de date) din fișierul *init.ora*. Gestionarea mărimii Data Block Buffer Cache este o parte impotantă a gestiunii și optimizării bazei de date.

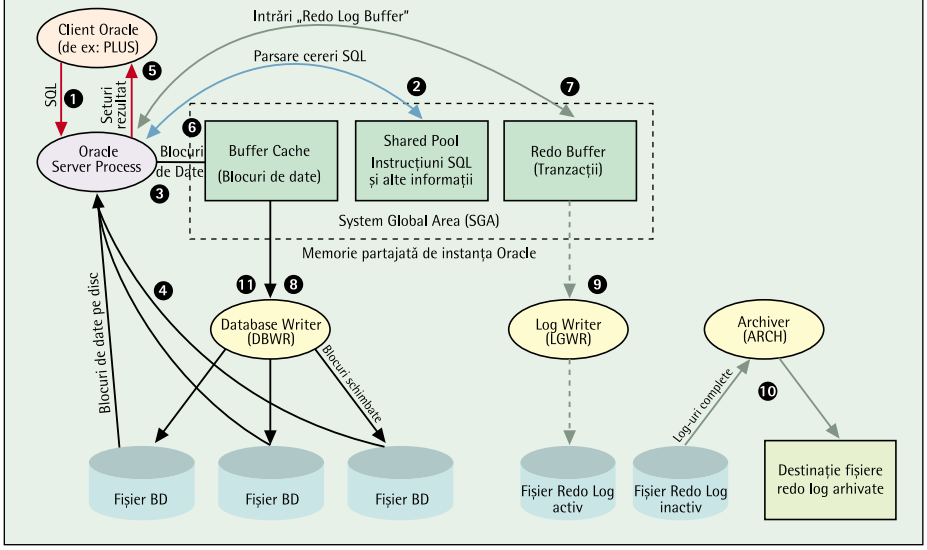
Cum *Data Block Buffer Cache* are o dimensiune fixată care este de obicei mai mică decât spațiul ocupat de segmentele de date, nu poate conține la un moment dat toate segmentele bazei de date în memorie. De obicei, *Data Block Buffer Cache* reprezintă 1-2% din dimensiunea bazei de date. Oracle gestionează spațiul disponibil în cache prin algoritmul LRU(least recently used). Când este nevoie de spațiu liber în cache, cele mai puțin utilizate blocuri vor fi scrise înapoi pe disc și noi blocuri de date le vor lua locul în memorie. Astfel, cele mai frecvent utilizate blocuri de date vor fi păstrate în memorie. Dacă SGA nu este destul de largă pentru a cuprinde cele mai frecvent utilizate blocuri de date, obiecte diferite se vor „bate" pentru spațiu în *Data Block Buffer Cache.* Drept urmare, cererile de date din *Data Block Buffer Cache* vor avea ca rezultat o proporție scăzută de reușite. Aceasta conduce la operații I/O pentru citirea datelor și implicit la degradarea peformanței.

Redo Log Buffer este acea zonă din SGA în care sunt înregistrate tranzacțiile înainte de a fi scrise în fișierele redo log online. Mărimea (în bytes) a *Redo Log Buffer* este setată prin intermediul parametrului LOG_BUFFER din init.ora.

Shared SQL Pool conține *Data Dictionary Cache* și *Library Cache.* Informațiile privind obiectele bazei de date sunt stocate în tabelele dicționarului de date. Când aceste informații sunt necesare bazei de date (de exemplu, pentru a verifica autorizația unui utilizator de a efectua o cerere asupra unei tabele), tabelele dicționarului de date sunt citite și datele returnate sunt aduse în SGA, în *Dictionary Cache*(sau *Row Cache*). Datele din *Dictionary Cache* sunt gestionate tot cu algoritmul LRU. Mărimea acestui buffer este gestiontă intern de baza de date. Dacă *Dictionary Cache* este prea mic, baza de date va interoga frecvent tabelele dicționarului de date, interogări cunoscute și sub numele de „recursive calls"- apeluri repetate, mai greu de rezolvat decât cererile de date din *Dictionary Cache.*

Dacă *Dictionary Cache* face posibilă partajarea informațiilor structurale de către utilizatorii bazei de date, *Library Cache* permite partajarea celor mai uzuale interogări SQL. Această zonă de memorie conține planul de execuție și arborele de parcurgere al cererilor SQL rulate asupra bazei de date. La a doua rulare, de către orice utilizator, o cerere SQL identică este capabilă să țină cont de informațiile de parcurgere disponibile în *Shared SQL Pool.* Gestionate tot prin intermediul algoritmului LRU, atunci când *Shared SQL Pool* se umple, planurile de execuție ale celor mai puțin utilizate fraze SQL vor fi șterse din *Library Cache* pentru a face loc unor noi intrări. Dacă *Shared SQL Pool* este prea mică, frazele SQL vor fi continuu încărcate în *Library Cache*, ducând la diminuarea performanței.

Instanța Oracle: fluxul de execuție al proceselor și utilizarea SGA



Pentru a înțelege mai bine modul în care lucrează Oracle, în figura de mai jos este prezentat fluxul de execuție al proceselor în cadrul unei instanțe Oracle.

Iată și activitățile corespunzătore:

(1) Programul client (`SQL*PLUS`, de exemplu) trimite fraza `SELECT` spre execuție procesului server.

(2) Procesul server caută în *Shared Pool* o fraza `SELECT` identică. Dacă nu o găsește, procesul server va realiza planul de execuție al comenzii (cere timp CPU) și va insera fraza `SELECT` în *Shared Pool* (necesită o blocare internă Oracle pentru a preveni modificarea concurentă a aceleiași zone din SGA de mai multe procese).

(3) Procesul server caută în *Buffer Cache* blocurile de date necesare. Dacă le găsește, aceste blocuri trebuiesc mutate la sfârșitul listei LRU(Least Recently Used). Aceasta necesită încă o blocare.

(4) Dacă blocurile de date nu sunt regăsite în *Buffer Cache* procesul server trebuie să le citească din fișierele de date(necesită o operație fizică de I/O) și să le introducă în *Buffer Cache*. O nouă blocare este necesară înaintea aducerii noilor blocuri în *Buffer Cache*.
(5) Procesul server returnează rândurile selectate procesului client (întârziere cauzată de rețea sau comunicare).
(6) La o comandă `UPDATE`, procesul server va modifica rândurile selectate prin modificarea blocurilor de date din memoria partajată și va genera noi intrări în buffer-ul segmentului de roolback.
(7) Comanda `UPDATE` va genera, de asemenea, o intrare în *Redo Log Buffer* care înregistrează detaliile tranzacției.
(8) Procesul de background *DBWR* va copia blocurile modificate din *Buffer Cache* pe disc, în fișierele de date.
(9) La o comandă `COMMIT`, procesul *LGWR* copiază conținutul din *Redo Log Buffer* în fișierul redo log, pe disc. Comanda `COMMIT` nu va returna controlul sesiunii Oracle pâna când *LGWR* nu termină de scris.
(10) Dacă baza de date rulează în modul *ARCHIVELOG*, procesul *ARCH* va arhiva fișierele redo log.
(11) La intervale regulate de timp sau la apariția unei comenzi `SWITCH LOG FILE`, Oracle realizează un *punct de verificare* (*checkpoint*). La apariția acestuia, toate blocurile de date modificate din *Buffer Cache* vor fi scrise pe disc.

Având în vedere cele prezentate mai sus, putem înțelege de ce mărimea și configurarea *SGA* pot avea un efect substanțial asupra performanței bazei de date. Este dificil de determinat în avans cât de mari ar trebui să fie componentele *SGA*. În general, prin supradimensionarea zonelor *SGA* performanța nu va avea de suferit, atâta timp cât *SGA* încape în memorie. Așa că, dacă memoria nu este o problemă, mărirea dimensiunii *SGA* nu este, de obicei, o idee rea.

Optimizarea zonei Shared Pool

Algoritmul utilizat de Oracle pentru gestionarea datelor în *Shared Pool* tinde să păstreze *Dictionary Cache* în memorie mai mult decât *Library Cache*. De aceea, o optimizare a dimensiunii *Library Cache* presupune de cele mai multe ori o dimensionare corectă și a *Dictionary Cache*. Oricum, lipsa unor date în *Library Cache* sau *Dictionary Cache* este mai grava decât lipsa de date din *Buffer Cache*. Din acest motiv, se va aloca suficientă memorie zonei *Shared Pool* înaintea alocării memoriei pentru *Buffer Cache*.

Există câțiva indicatori care pot determina dacă este necesară o supradimensionare a zonei *Shared Pool* (mărimea implicită a acesteia este de 3,5Mb). Este bine ca aceste statistici să fie colectate după o semnificativă activitate a bazei de date și nu imediat după startarea ei. Indicatorii „*Library Cache Hit Ratio*" și „*Row Cache Hit Ratio*" pot fi determinați prin interogarea vederilor `V$LIBRARYCACHE` și `V$ROWCACHE`. Aceste vederi conțin statistici privind activitatea zonelor *Library Cache și Dictionary Cache* de la ultima pornire a instanței. Implicit, ele sunt disponibile utilizatorului *SYS* sau oricărui utilizator cu drept de *SELECT ANY TABLE*(cum ar fi utilizatorul *SYSTEM*).

```
SELECT SUM(pins) "Executions",
   SUM(reloads) "Cache misses while executing",
   SUM(pins-reloads)/SUM(pins)*100 "Library Cache Hit Ratio"
FROM V$LIBRARYCACHE;

SELECT SUM(gets) "Data Dictionry Gets",
       SUM(getmisses) "Dat Dictionary Get Misses",
   SUM(gets-getmisses)/SUM(gets)*100 "Row Cache Hit Ratio"
FROM V$ROWCACHE;
```

Totalul de memorie liberă din *Shared Pool* este raportat în vederea `V$SGASTAT`:

```
SELECT * FROM V$SGASTAT
WHERE UPPER(NAME)='FREE MEMORY';
```

Dacă valorile indicatorilor „*Library Cache Hit Ratio*" și „*Row Cache Hit Ratio*"sunt mai mici decât 95% iar memoria libera este aproape de zero ar trebui mărită zona *Shared Pool* pâna când valorile celor doi indicatori încetează să mai crească.

Reducerea numărului de reîncărcări în *Library Cache* poate fi făcută fie prin alocarea de memorie adițională zonei *Shared Pool* fie prin scrierea de fraze SQL identice, ori de câte ori acest lucru este posibil. Două fraze SQL sau două blocuri PL/SQL pot utiliza aceeași zonă Sql partajată dacă sunt identice în conformitate cu următoarele criterii:

- Textul frazelor SQL sau al blocurilor PL/SQL trebuie să fie identic, caracter cu caracter, inclusiv spații libere sau literă mare. De exemplu, următoarele fraze nu vor utiliza aceeași zonă SQL partajată:
```
SELECT * FROM EMP;
SELECT * FROM emp;
```
- Obiectele referite de frazele SQL trebuie să aparțină aceleiași scheme. De exemplu, dacă schemele a doi utilizatori Bob și Ed conțin ambele tabela `EMP` și ambii realizează următoarea interogare, frazele SQL executate nu vor partaja aceeași zonă SQL.
```
SELECT * FROM EMP;
```

Dacă ambele fraze interoghează aceeași tabelă și califică tabela cu schema de care aparține, ca în exemplul următor, utilizatorii vor folosi aceeași zonă SQL partajată.

```
SELECT * FROM BOB.EMP;
```

- Variabilele de substituție din frazele SQL trebuie să fie identice ca nume și tip. În următorul exemplu nu se va folosi aceeași zonă partajată:
```
SELECT * FROM EMP WHERE DEPTNO = :DEPARTMENT_NO;
SELECT * FROM EMP WHERE DEPTNO = :DEP_NO;
```
- Frazele SQL trebuiesc optimizate utilizând același tip de optimizare (fie bazat pe reguli fie bazat pe cost).

Iată câteva strategii pentru scrierea unor fraze SQL identice:

(1) Utilizarea variabilelor substituibile în locul constantelor declarate explicit. De exemplu, următoarele fraze SQL nu folosesc aceeași zonă partajată din *Library Cache* pentru că nu sunt identice caracter cu caracter:
```
SELECT * FROM EMP WHERE DEPTNO = 10;
SELECT * FROM EMP WHERE DEPTNO = 20;
```
În locul frazelor de mai sus, se pote utiliza următoarea frază în care se va introduce 10 sau 20 în funcție de interogarea dorită:
```
SELECT * FROM EMP WHERE DEPTNO = :DEP_NO;
```
(2) Folosirea aceluiași tip de optimizare pentru cererile SQL;
(3) Stabilirea unor convenții privind numele variabilelor de substituție sau spațierea frazelor SQL, convenții ce vor fi respectate de toți utilizatorii;
(4) Utilizarea procedurilor stocate ori de câte ori este posibil. Pentru încărcarea în *Shared Pool* a blocurilor PL/SQL des utilizate poate fi folosit pachetul `DBMS_SHARED_POOL`. Acesta trebuie creat în prealabil executând scriptul dbmspool.SQL din subdirectorul /rdbms/admin al directorului în care a fost instalat Oracle. Pentru încărcarea în *Shared Pool* a pachetului „Students" al utilizatorului Scott se va executa:
```
EXECUTE DBMS_SHARED_POOL.KEEP('Scott.Students','P');
```
Iar pentru scoaterea lui din zona de memorie partajată:
```
EXECUTE DBMS_SHARED_POOL.UNKEEP('Scott.Students','P');
```

Estimarea dimensiunii zonei *Shared Pool* poate fi făcută în modul următor: setarea parametrului `SHARED_POOL_SIZE` din *init.ora* cu o valoare foarte mare (de exemplu, de patru ori memoria sistemului), oprirea bazei de date, repornirea instanței. După o perioadă reprezentativă de timp, în care s-a înregistrat o activitate semnificativă a bazei de date, pot fi colectate următoarele statistici:

```
SELECT SUM(sharable_mem) "Packages/Views"
FROM V$DB_OBJECT_CACHE;

SELECT SUM(sharable_meme) "SQL Statements"
FROM V$SQLAREA
```

```
WHERE executions>5;

SELECT SUM(250*users_opening) "SQL Users"
FROM V$SQLAREA;
```

Prin adunarea celor trei valori și multiplicarea rezultatului cu 2,5 se va obține o valoare estimativă pentru mărimea zonei *Shared_Pool* după cum urmează:

```
SELECT SUM(a.spspv) "Packages/Views",
   SUM(a.spssql) "SQL Statements",
   SUM(a.spsusr) "SQL Users",
   ROUND((SUM(a.spspv)+SUM(a.spssql)+SUM(a.spsusr))*2.5,-6)
   "Estimated Shared_Pool_Size"
FROM (SELECT SUM(sharable_mem) spspv, 0 spssql, 0 spsusr
        FROM V$DB_OBJECT_CACHE
        UNION ALL
        SELECT 0, SUM(sharable_mem), 0
        FROM V$SQLAREA
        WHERE executions>5
        UNION ALL
        SELECT 0, 0, SUM(250*users_opening)
        FROM V$SQLAREA) a;
```

Baza de date poate avea dificultăți în găsirea unui bloc contiguu de memorie pentru a satisface cererile mari de memorie. Pentru a rezolva aceste cereri(compilarea unor blocuri PL/SQL sau triggeri) administratorul bazei de date poate rezerva memorie, în cadrul zonei *Shared Pool*. Obiectele mici nu vor fragmenta acest spațiu asigurând astfel existența unor zone largi de memorie în lista rezervată. Dimensiunea ei este stabilită de parametrul `SHARED_POOL_RESERVED_SIZE` din fișierul *init.ora*. Pentru a monitoriza lista rezervată poate fi interogată vederea `V$SHARED_POOL_REZERVED`, coloana acesteia, `REQUEST_FAILURES` dând numărul de accesări nereușite ale listei rezervate. Dacă `REQUEST_FAILURES` este mai mare ca 0 și în creștere, lista rezervată trebuie supradimensionată, dacă este 0, lista rezervată este dimensionată corect sau poate fi prea mare.

Optimizarea Buffer Cache

După cum spuneam mai sus, *Buffer Cache* este o zonă din *SGA* care stochează copii ale blocurilor de date în memorie pentru a reduce operațiile fizice de I/O. Oracle stochează statistici privind accesul la date în vederea *V$SYSSTAT*. Interogarea acesteia poate ajuta la optimizarea *Buffer Cache*, prin calcularea indicatorului „*Buffer Cache Hit Ratio*”:

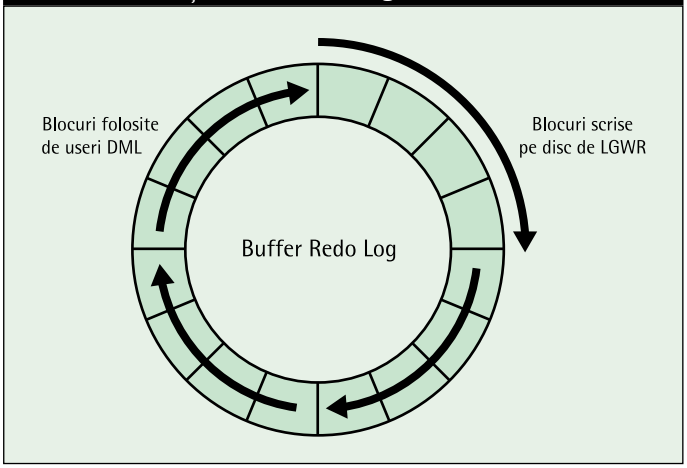
```
SELECT name, value
FROM V$SYSSTAT
WHERE name IN ('DB BLOCK GETS', 'CONSISTENT GETS','PHYSICAL
   READS')

SELECT 1-(pr.value/(dbg.value+cg.value)) "Buffer Cache Hit
   Ratio"
FROM V$SYSSTAT pr, V$SYSSTAT dbg, V$SYSSTAT cg
WHERE pr.name='physical reads'
AND dbg.name='db block gets'
AND cg.name='consistent gets';
```

Suma valorilor `DB BLOCK GETS` și `CONSISTENT GETS` reprezintă numărul total de cereri de date în timp ce `PHYSICAL READS` reprezintă numărul de cereri de date care au rezultat în accesarea fișierelor de date de pe disc. *Buffer Cache* este dimensionat optim dacă valorea indicatorului „*Buffer Cache Hit Ratio*” este peste 0.9, în caz contrar acesta trebuie mărit printr-o valoare mai mare a parametrului `DB_BLOCK_BUFFER` din init.ora. O greșeală frecventă este continuarea creșterii valorii `DB_BLOCK_BUFFER` fără nici un efect atunci când se realizează citiri integrale de tabele (full table scans) sau alte operații care nu folosesc buffer-ul.

Optimizarea Redo Log Buffer

Parametrul `LOG_BUFFER` din init.ora rezervă spațiu pentru *Redo Log Buffer*. Tranzacțiile efectuate sunt înregistrate în *Redo Log Buffer* timp în care procesul *LGWR* mută porțiuni din *Redo Log Buffer* pe disc, în fișierele redo log. *LGWR* începe să scrie întotdeauna când *Redo Log Buffer* începe să se umple. Din acest motiv, un buffer mai mare reduce posibilitatea de „ciocnire” a noilor intrări cu porțiuni încă nescrise pe disc. Modul de funcționare a *Redo Log Buffer* este prezentat în figura „Modul de funcționare a Redo Log Buffer”.

Modul de funcționare a Redo Log Buffer



Redo Log Buffer este, în mod normal, mic în comparație cu mărimea totală a SGA și o modestă creștere a dimensiunii lui poate îmbunătăți semnifictiv performanța. Un indicator care ne poate ajuta la dimensionarea *Redo Log Buffer* este „Space Request Ratio”:

```
SELECT rlsr.value/re.value "Space Request Ratio"
FROM V$SYSSTAT rlsr, V$SYSSTAT re
WHERE rlsr.name='redo log space requests'
AND re.name='redo entries';
```

Dacă proporția este mai mare de 1:5000, trebuie mărită dimensiunea *Redo Log Buffer* până când valoarea indicatorului încetează să mai scadă.

Concluzii

O dimensionare corectă a componentelor SGA poate îmbunătăți semnificativ performanța bazei de date. Nu este surprinzător faptul că Sysem Global Area există în primul rând pentru a reduce numărul de operații fizice de I/O prin păstrarea în memorie a informației de pe disc și de a micșora numărul de reparcurgeri al frazelor SQL. Totuși, memoria cerută de Oracle depinde de aplicație, așa că, înainte de a optimiza alocarea memoriei este bine să optimizăm aplicația și cererile SQL.

Monica Ibănescu este ORACLE Programmer la Romanian Data Soft. Poate fi contactată pe email la: s_ibanes@yahoo.com.

Bibliografie

✓ Oracle8I DBA Handbook: Kevin Loney,Marlene Theriault
✓ Oracle8 Secrete:
Edward Honour, Paul Dalberth, Ari Kaplan, Atul Mehta
✓ Oracle7 Administration:
David Austin, Laie Caindec, Julie Frazier, Sue Jang