17

第17章 Shared Pool原理及性能分析

Shared Pool是Oracle SGA设置中最复杂也是最重要的一部分内容，Oracle通过Shared Pool来实现SQL共享、减少代码硬解析等，从而提高数据库的性能。在某些版本中，如果设置不当，Shared Pool可能会极大地影响数据库的正常运行。

本章就Shared Pool的原理及性能问题进行深入探讨，并通过几个具体案例介绍相关性能问题的诊断及解决方法。

17.1 Shared Pool的基本原理

在Oracle 7之前，Shared Pool并不存在，每个Oracle连接都有一个独立的Server进程与之相关联，Server进程负责解析和优化所有SQL和PL/SQL代码。典型的，在OLTP环境中，很多代码具有相同或类似的结构，反复的独立解析浪费了大量的时间以及资源，Oracle最终认识到这个问题，并且从PL/SQL开始尝试把这部分可共享的内容进行独立存储和管理，于是Shared Pool作为一个独立的SGA组件开始被引入，并且其功能和作用被逐渐完善和发展起来。

在这里注意到，Shared Pool最初被引入的目的，也就是它的本质功能在于：实现共享。如果系统代码是完全异构的（假设你的代码从不绑定变量，从不反复执行），那么会发现，这时候Shared Pool完全就成为了一个负担，它在徒劳无功地进行无谓的努力：保存代码、执行计划等期待重用，并且客户端要不停的获取Latch，试图寻找共享代码，却始终一无所获。如果真是如此，那这是我们最不愿看到的情况，Shared Pool变得有害无益。当然这是极端，可是在性能优化中会发现，大多数性能低下的系统都存在这样的通病：代码极少共享，缺乏或从不实行变量绑定。优化这些系统的根本方法就是优化代码，使代码（在保证性能的前提下）可以充分共享，减少无谓的反复硬/软解析。

实际上，Oracle引入Shared Pool就是为了帮助实现代码的共享和重用。了解了这一点之后，开发人员在应用开发的过程中，就应该有意识地提高自己的代码水平，以期减少数据库的压力。这也应该是对开发人员的最初和最基本的要求。

17.2 Shared Pool的设置说明

Shared Pool的大小通过初始化参数shared_pool_size设置。

对于Shared Pool 的设置,一直以来是最具有争议的一部分内容。一方面很多人建议可以把Shared Pool 设置得稍大，以充分 Cache 代码和避免 ORA-04031 错误的出现；另一方面又有很多人建议不能把 Shared Pool 设置得过大，因为过大可能会带来管理上的额外负担，从而会影响数据库的性能。

至于哪一种说法更为准确，这个管理上的额外负担究竟指什么，这并不是一句话就能予以定论的，下面试图通过一些内部分析，从内部原理来回答这个问题。

17.2.1　基本知识

下面用到了 Shared Pool 的转储，所以首先了解一下与其相关的命令。

可以通过以下命令转储 Shared Pool 共享内存的内容：

```
SQL> alter session set events 'immediate trace name heapdump level 2';
Session altered.

本测试中引用的两个 trace 文件:

9i:

SQL> @gettrcname

TRACE_FILE_NAME
--------------------------------------------------------------------------------
/opt/oracle/admin/hsjf/udump/hsjf_ora_24983.trc

8i:

SQL> @gettrcname

TRACE_FILE_NAME
--------------------------------------------------------------------------------
/usr/oracle8/admin/guess/udump/guess_ora_22038.trc
```

其中 alter session set events ’immediate trace name heapdump level 2’是一条内部命令，指定 Oracle 把 Shared Pool 的内存结构在 Level 2 级转储出来。

转储的内容被记录在一个 trace 文件中，这个 trace 文件可以在 undmp 目录下找到。

为了比较 Oracle 8i 以及 Oracle 9i 的不同，文中引用了两个版本的跟踪文件。

这里，gettrcname.sql 是用来获取 trace 文件名称的一个脚本，代码如下：

```
SELECT   d.VALUE
      || '/'
      || LOWER (RTRIM (i.INSTANCE, CHR (0)))
      || '_ora_'
      || p.spid
      || '.trc' trace_file_name
  FROM (SELECT p.spid
          FROM v$mystat m, v$session s, v$process p
         WHERE m.statistic# = 1 AND s.SID = m.SID AND p.addr = s.paddr) p,
       (SELECT t.INSTANCE
          FROM v$thread t, v$parameter v
         WHERE v.NAME = 'thread'
```

```
          AND (v.VALUE = 0 OR t.thread# = TO_NUMBER (v.VALUE))) i,
      (SELECT VALUE
         FROM v$parameter
        WHERE NAME = 'user_dump_dest') d
/
```

有兴趣的朋友也可以在我的网站（http://www.eygle.com）找到相关脚本及更多的详细说明。

17.2.2 Shared Pool 的 Free List 管理

Shared Pool 通过 Free Lists 管理 free 内存块（Chunk），free 内存块按不同 size 被划分到不同的部分（Bucket）进行管理。

结合 Dump 文件，可以通过图 17-1 对 Shared Pool 的 Free List 管理进行说明：

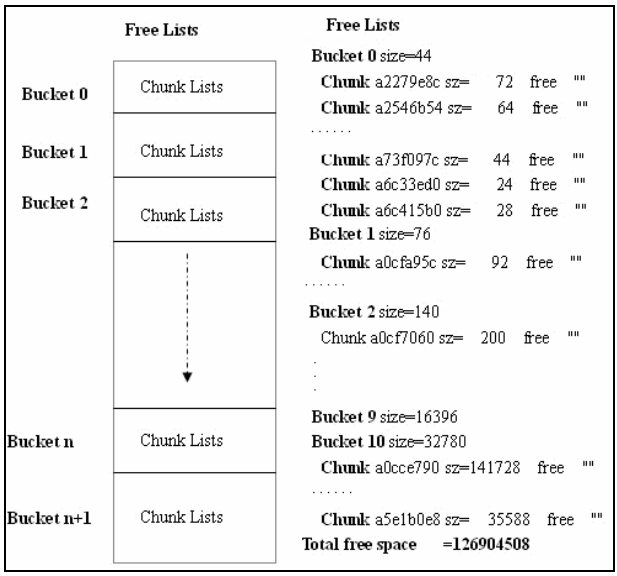

图 17-1 Shared Pool 自由列表

在 Oracle 8i 中，不同 Bucket 管理的内存块的 size 范围如下所示（size 显示的是下边界）：

```
oracle:/usr/oracle8/admin/guess/udump>cat guess_ora_22038.trc|grep Bucket
Bucket 0 size=44
Bucket 1 size=76
Bucket 2 size=140
Bucket 3 size=268
Bucket 4 size=524
Bucket 5 size=1036
Bucket 6 size=2060
Bucket 7 size=4108
Bucket 8 size=8204
Bucket 9 size=16396
Bucket 10 size=32780
```

注 意

本章所引用的所有 trace 文件都可以在本人的个人网站（http://www.eygle.com）上找到。

注意观察这个输出结果，在这里，小于 76 bytes 的块都位于 Bucket 0 上；大于 32780 的块，都在 Bucket 10 上。

初始时，数据库启动以后，Shared Pool 多数是连续内存块。但是当空间分配使用以后，内存块开始被分割，碎片开始出现，Bucket 列表开始变长。

Oracle 请求 Shared Pool 空间时，首先进入相应的 Bucket 进行查找。如果找不到，则转向下一个非空的 Bucket，获取第一个 Chunk。分割这个 Chunk，剩余部分会进入相应的 Bucket，进一步增加碎片。

最终的结果是，由于不停分割，每个 Bucket 上的内存块会越来越多，越来越碎小。通常 Bucket 0 的问题会最为显著，在这个测试的小型数据库上，Bucket 0 上的碎片已经达到 9030 个，而 shared_pool_size 设置仅为 150MB。

通常如果每个 Bucket 上的 Chunk 多于 2000 个，就被认为是 Share Pool 碎片过多。

Shared Pool 的碎片过多，是 Shared Pool 产生性能问题的主要原因。

碎片过多会导致搜索 Free List 的时间过长，而 Free Lists 的管理和搜索都需要获得和持有一个非常重要的 Latch，就是 Shared Pool Latch。Latch 是 Oracle 数据库内部提供的一种低级锁，通过串行机制保护共享内存不被并发更新/修改所损坏。Latch 的持有通常都非常短暂（通常微秒级），但是对于一个繁忙的数据库，这个串行机制往往会成为极大的性能瓶颈。关于 Latch 的机制在这里不过多介绍，那需要太多的篇幅。

继续前面的话题，如果 Free Lists 链表过长，搜索这个 Free Lists 的时间就会变长，从而可能导致 Shared Pool Latch 被长时间持有，在一个繁忙的系统中，这会引起严重的 Shared Pool Latch 的竞争。在 Oracle 9i 之前，这个重要的 Shared Pool Latch 只有一个，所以长时间持有将会导致严重的性能问题。

而在大多数情况下，用户请求的都是相对小的内存块（Chunk），这样搜索 Bucket 0 往往消耗了大量的时间及资源，Latch 的争用此时就会成为一个非常严重的问题。

所以，在 Oracle 9i 之前，如果盲目地增大 shared_pool_size 或设置过大的 shared_pool_size，往往会适得其反。这就是大家曾经听说过的：过大的 Shared_Pool 会带来管理上的负担。

在 Oracle 9i 中，Oracle 改写了 Shared Pool 管理的算法，来看一下 Oracle 9i 中的处理方式：

```
 [oracle@jumper oracle]$ sqlplus "/ as sysdba"
SQL*Plus: Release 9.2.0.3.0 - Production on Wed Aug 18 22:13:07 2004

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.2.0.3.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.3.0 - Production

SQL> alter session set events 'immediate trace name heapdump level 2';

Session altered.
```

```
SQL> @gettrcname

TRACE_FILE_NAME
------------------------------------------------------------------------------
/opt/oracle/admin/hsjf/udump/hsjf_ora_24983.trc

SQL>
SQL> !
 [oracle@jumper oracle]$ cd $admin
 [oracle@jumper udump]$ cat hsjf_ora_24983.trc|grep Bucket
Bucket 0 size=16
Bucket 1 size=20
Bucket 2 size=24
Bucket 3 size=28
Bucket 4 size=32
Bucket 5 size=36
Bucket 6 size=40
Bucket 7 size=44
Bucket 8 size=48
Bucket 9 size=52
Bucket 10 size=56
Bucket 11 size=60
……<这里省略了部分内容>……
Bucket 235 size=3116
Bucket 236 size=3180
Bucket 237 size=3244
Bucket 238 size=3308
Bucket 239 size=3372
Bucket 240 size=3436
Bucket 241 size=3500
Bucket 242 size=3564
Bucket 243 size=3628
Bucket 244 size=3692
Bucket 245 size=3756
Bucket 246 size=3820
Bucket 247 size=3884
Bucket 248 size=3948
Bucket 249 size=4012
Bucket 250 size=4108
Bucket 251 size=8204
Bucket 252 size=16396
Bucket 253 size=32780
Bucket 254 size=65548
```

观察以上输出可以看到，在 Oracle 9i 中，Free Lists 被划分为 0～254，共 255 个 Bucket。每个 Bucket 容纳的 size 范围可以进一步细分：

- Bucket 0～199 容纳 size 以 4 bytes 递增。
- Bucket 200～249 容纳 size 以 64 bytes 递增。

从 Bucket 249 开始，Oracle 各 Bucket 步长进一步增加：

- Bucket 249：4012～4107 = 96
- Bucket 250：4108～8203 = 4096

- Bucket 251：8204～16395 = 8192
- Bucket 252：16396～32779 = 16384
- Bucket 253：32780～65547 = 32768
- Bucket 254：>=65548

对比 Oracle 8i，在 Oracle 9i 中 Shared Pool 管理最为显著的变化是，对于数量众多的 Chunk，Oracle 增加了更多的 Bucket 来管理。

0～199 共 200 个 Bucket，size 以 4 为步长递增；200～249 共 50 个 Bucket，size 以 64 递增。这样每个 Bucket 中容纳的 Chunk 数量大大减少，查找的效率得以提高。

而且在 Oracle 9i 中，为了增加对于大共享池的支持，Shared Pool Latch 从原来的 1 个增加到现在的 7 个。如果系统有 4 个或 4 个以上的 CPU，并且 shared_pool_size 大于 250MB，Oracle 可以把 Shared Pool 分割为多个子缓冲池进行管理，每个 subpool 都拥有独立的结构、LRU 和 Shared Pool Latch。以下查询显示的就是这些 Latch：

```
SQL> select addr, name, gets, misses, spin_gets
  2  from   v$latch_children where name = 'shared pool';

ADDR           NAME              GETS    MISSES  SPIN_GETS
---------------- -------------------- ---------- ---------- ----------
0000000380068F38 shared pool                0        0        0
0000000380068E40 shared pool                0        0        0
0000000380068D48 shared pool                0        0        0
0000000380068C50 shared pool                0        0        0
0000000380068B58 shared pool                0        0        0
0000000380068A60 shared pool                0        0        0
0000000380068968 shared pool          13808572      3089      3087

7 rows selected.
```

子缓冲的数量由一个新引入的隐含参数设置：_KGHDSIDX_COUNT。

可以手工调整该参数（仅限于试验环境研究用），以观察共享池管理的变化：

```
SQL> alter system set "_kghdsidx_count"=2 scope=spfile;

System altered.

SQL> startup force;
ORACLE instance started.

Total System Global Area   80811208 bytes
Fixed Size                 451784 bytes
Variable Size            37748736 bytes
Database Buffers         41943040 bytes
Redo Buffers               667648 bytes
Database mounted.
Database opened.

SQL> col KSPINM for a20
SQL> col KSPPSTVL for a20
SQL> select a.ksppinm, b.ksppstvl
  2  from   x$ksppi a, x$ksppsv b
```

```
  3  where  a.indx = b.indx and   a.ksppinm = '_kghdsidx_count';

KSPPINM                              KSPPSTVL
-------------------------------------------------------------- --------------------
_kghdsidx_count                           2

SQL>

SQL> col name for a20
SQL> select addr, name, gets, misses, spin_gets
  2  from   v$latch_children where name = 'shared pool';

ADDR     NAME                    GETS     MISSES  SPIN_GETS
-------- -------------------- ---------- ---------- ----------
50043078 shared pool                 0          0          0
50042FB0 shared pool                 0          0          0
50042EE8 shared pool                 0          0          0
50042E20 shared pool                 0          0          0
50042D58 shared pool                 0          0          0
50042C90 shared pool              8166          0          0
50042BC8 shared pool               298          0          0

7 rows selected.
```

但是需要注意的是,在Oracle 9i 中,这些新特性同时也带来了一些相应的Bug,跟Shared Pool多缓冲池相关的 Bug 有：3316003，该 Bug 在 9205 中得到了修正，读者可以参考 MetaLink 上的相关链接。

通过这一系列的算法改进,Oracle 9i 中的Shared Pool管理得以增强,较好地解决了大Shared Pool 的性能问题；在 Oracle 8i 中，过大的 Shared Pool 设置可能带来的栓锁争用等性能问题在某种程度上得到了解决。

所以说，如果是在 Oracle 9i 中，设置较大的 Shared Pool 并不一定会给用户带来和 Oracle 8i 中出现的麻烦。

在论坛上经常看到很多人对于 Shared_Pool 的建议一直就是 200MB～300MB，而且一直认为这就是 Shared Pool 性能问题的关键，实际上，这是不确切的。

另外，在这里，想提一下 Buffer Cache 的管理方式，Buffer Cache 的管理也涉及两个重要的 Latch：Cache Buffers LRU Chain 和 Cache Buffers Chains。

其中 Cache Buffers LRU Chain 用于管理 Buffer Cache 中内存块的分配和使用，并按照 LRU 算法进行老化，当新数据需要读到 Buffer Cache 中时，该 Latch 需要被持有以寻找和锁定可用的内存块。而 Cache Buffers Chains 用于 Buffre Cache 中的数据访问（pined），当需要访问数据时，该 Latch 需要被持有。

可以看到，过于频繁的数据访问几乎肯定会引起 Cache Buffers Chains 的竞争，也就是通常所说的热点块的竞争。Oracle 通过两个数据结构来管理这部分内存:Hash Buckets 和 Hash Latchs。

在 Oracle 8i 之前，对于每一个 Hash Bucket，Oracle 使用一个独立的 Hash Latch 来维护，其缺省 Bucket 数量为：

```
next_prime (db_block_buffers/4)
```

由于过于严重的热点块竞争，从 Oracle 8i 开始，Oracle 改变了这个算法，首先 Bucket 数

量开始增加，_db_block_hash_buckets 增加到 2*db_block_buffers，而_db_block_hash_latchs 的数量也发生了变化：

当 cache buffers 少于 2052 buffers，则

```
_db_block_hash_latches = power(2,trunc(log(2, db_block_buffers - 4) - 1))
```

当 cache buffers 多于 131075 buffers，则

```
_db_block_hash_latches = power(2,trunc(log(2, db_block_buffers - 4) - 6))
```

当 cache buffers 位于 2052 与 131075 buffers 之间，则

```
_db_block_hash_latches = 1024
```

从 Oracle 8i 开始，db_block_hash_buckets 的数量较以前增加了 8 倍，而 db_block_hash_latchs 的数量增加有限，这意味着，每个 Latch 需要管理多个 Bucket，但是由于 Bucket 数量的多倍增加，每个 Bucket 上的 Block 数量得以减少，从而使少量 Latch 管理更多 Bucket 成为可能。

下面通过图 17-2 来简要描述这个变化（图中省略了一些内容）。

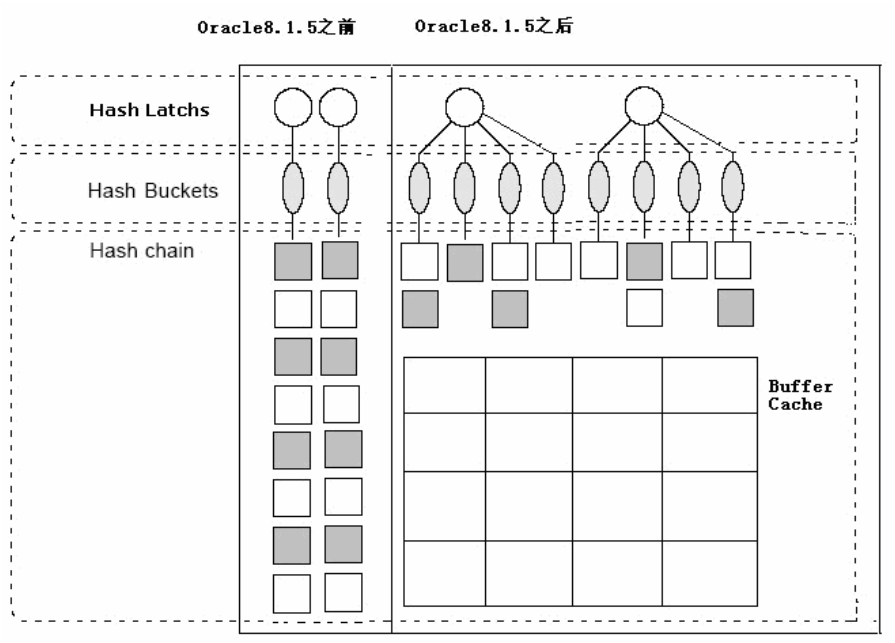

图 17-2　Buffer Cache 管理示意图

Buffer Cache 的这一系列改进和 Shared Pool 的改进极为类似，通过减少每个 Bucket 管理的 Block 的数量，从而提高了管理效率，降低了闩锁的竞争。

关于这部分内容，biti_rainy 网友在他的文章《深度分析数据库的热点块问题》中有详细阐述，在此不再更多涉及。

（**下文略**）

作者简介

盖国强，网名 eygle，ITPUB Oracle 管理版版主，ITPUB 论坛超级版主，曾任 ITPUB MS 版主。CSDN eMag Oracle 电子杂志主编。

曾任职于某国家大型企业，服务于烟草行业，开发过基于 Oracle 数据库的大型 ERP 系统，属国家信息产业部重点工程。同时负责 Oracle 数据库管理及优化，并为多家烟草企业提供 Oracle 数据库管理、优化及技术支持。

目前任职于北京某电信增值业务系统提供商企业，首席 DBA，负责数据库业务。管理全国 30 多个数据库系统。项目经验丰富，曾设计规划及支持中国联通增值业务等大型数据库系统。

实践经验丰富，长于数据库诊断、性能调整与 SQL 优化等。对于 Oracle 内部技术具有深入研究。

高级培训讲师，培训经验丰富，曾主讲 ITPUB DBA 培训及 ITPUB 高级性能调整等主要课程。《Oracle 数据库 DBA 专题技术精粹》一书的主编及主要作者。

可以在 http://www.eygle.com 上找到关于作者的更多信息。