Oracle 内存分配与调整

作者介绍

冯春培，毕业于北京信息工程学院。曾做电信计费后台程序开发，从事过开发 DBA 工作，目前公司主要做数据库优化产品开发。热爱 ORACLE，在 www.itpub.net 任数据库管理版块版主(biti_rainy)，个人兴趣主要在 oracle internal、performance tuning。对数据库管理、备份与恢复、数据库应用开发、SQL 优化均有广泛理解。希望大家一起探讨 oracle 及相关技术。

前言

对于 oracle 的内存的管理，截止到 9iR2，都是相当重要的环节，管理不善，将可能给数据库带来严重的性能问题。下面我们将一步一步就内存管理的各个方面进行探讨。

概述

oracle 的内存可以按照共享和私有的角度分为系统全局区和进程全局区，也就是 SGA 和 PGA(process global area or private global area)。对于 SGA 区域内的内存来说，是共享的全局的，在 UNIX 上，必须为 oracle 设置共享内存段（可以是一个或者多个），因为 oracle 在 UNIX 上是多进程；而在 WINDOWS 上 oracle 是单进程（多个线程），所以不用设置共享内存段。PGA 是属于进程（线程）私有的区域。在 oracle 使用共享服务器模式下（MTS）,PGA 中的一部分，也就是 UGA 会被放入共享内存 large_pool_size 中。

对于 SGA 部分，我们通过 sqlplus 中查询可以看到：

```
SQL> select * from v$sga;

NAME                 VALUE
-------------------- ----------
Fixed Size           454032
Variable Size        109051904
Database Buffers     385875968
Redo Buffers         667648
```

Fixed Size

oracle 的不同平台和不同版本下可能不一样，但对于确定环境是一个固定的值，里面存储了 **SGA** 各部分组件的信息，可以看作引导建立 **SGA** 的区域。

Variable Size

包含了 shared_pool_size、java_pool_size、large_pool_size 等内存设置

Database Buffers

指数据缓冲区，在 8i 中包含 db_block_buffer*db_block_size、buffer_pool_keep、buffer_pool_recycle 三部分内存。在 9i 中包含 db_cache_size、db_keep_cache_size、db_recycle_cache_size、db_nk_cache_size。

Redo Buffers

指日志缓冲区，log_buffer。在这里要额外说明一点的是，对于 v$parameter、**v$sgastat**、**v$sga** 查询值可能不一样。**v$parameter** 里面的值，是指用户在初始化参数文件里面设置的值，**v$sgastat** 是 **oracle** 实际分配的日志缓冲区大小（因为缓冲区的分配值实际上是离散的，也不是以 **block** 为最小单位进行分配的），**v$sga** 里面查询的值，是在 **oracle** 分配了日志缓冲区后，为了保护日志缓冲区，设置了一些保护页，通常我们会发现保护页大小大约是 **11k(**不同环境可能不一样**)**。参考如下内容

```
SQL>    select substr(name,1,10) name,substr(value,1,10) value
  2    from v$parameter where name = 'log_buffer';

NAME                          VALUE
--------------------        --------------------
log_buffer                     524288


SQL> select * from v$sgastat ;

POOL    NAME                    BYTES
----------- --------------------
fixed_sga                        454032
buffer_cache                    385875968
log_buffer                       656384

SQL> select * from v$sga;

NAME                             VALUE
--------------------             ----------
Fixed Size                      454032
Variable Size                   109051904
Database Buffers                385875968
Redo Buffers                     667648
```

关于各部分内存的作用，参考 oracle 体系结构，在此不再叙述。

Ⅰ SGA 的大小

那么我们现在来考察内存参数的设置。实际上，对于特定的环境，总是存在着不同的最优设置的，没有任何一种普遍适用的最优方案。但为什么在这里我们还要来谈设置这个话题呢，那仅仅是出于一个目的，避免过度的犯错误。事实上，在任何一个生产系统正式投入使用之前，我们不拥有任何系统运行信息让我们去调整，这样就只有两种可能，一是根据文档推荐设置，另外一种就是根据经验设置。相对来说，根据经验的设置比根据文档的设置要可靠一些。尤其是那些 24*7 的系统，我们更要减少错误的发生。那么我们尝试去了解不同的

系统不同的应用的具体设置情况，从而提供一个参照信息给大家。

为了得出一个参照设置，我们就必须假定一个参照环境。以下所有设置我们基于这样一个假定，那就是硬件服务器上只考虑存在操作系统和数据库，在这个单一的环境中，我们来考虑内存的设置。

在设置参数之前呢，我们首先要问自己几个问题

一：物理内存多大

二：操作系统估计需要使用多少内存

三：数据库是使用文件系统还是裸设备

四：有多少并发连接

五：应用是 OLTP 类型还是 OLAP 类型

根据这几个问题的答案，我们可以粗略地为系统估计一下内存设置。那我们现在来逐个问题地讨论，首先物理内存多大是最容易回答的一个问题，然后操作系统估计使用多少内存呢？从经验上看，不会太多，通常应该在 200M 以内（不包含大量进程 PCB）。

接下来我们要探讨一个重要的问题，那就是关于文件系统和裸设备的问题，这往往容易被我们所忽略。操作系统对于文件系统，使用了大量的 buffer 来缓存操作系统块。这样当数据库获取数据块的时候，虽然 SGA 中没有命中，但却实际上可能是从操作系统的文件缓存中获取的。而假如数据库和操作系统支持异步 IO，则实际上当数据库写进程 DBWR 写磁盘时，操作系统在文件缓存中标记该块为延迟写，等到真正地写入磁盘之后，操作系统才通知 DBWR 写磁盘完成。对于这部分文件缓存，所需要的内存可能比较大，作为保守的估计，我们应该考虑在 0.2——0.3 倍内存大小。但是如果我们使用的是裸设备，则不考虑这部分缓存的问题。这样的情况下 SGA 就有调大的机会。

关于数据库有多少并发连接，这实际上关系到 PGA 的大小（MTS 下还有 large_pool_size）。事实上这个问题应该说还跟 OLTP 类型或者 OLAP 类型相关。对于 OLTP 类型 oracle 倾向于可使用 MTS,对于 OLAP 类型使用独立模式，同时 OLAP 还可能涉及到大量的排序操作的查询，这些都影响到我们内存的使用。那么所有的问题综合起来，实际上主要反映在 UGA 的大小上。UGA 主要包含以下部分内存设置

```
SQL> show parameters area_size
NAME                                 TYPE        VALUE
------------------------------------ ------- ------------
bitmap_merge_area_size               integer     1048576
create_bitmap_area_size              integer     8388608
hash_area_size                       integer     131072
sort_area_size                       integer     65536
SQL>
```

在这部分内存中我们最关注的通常是 sort_area_size，这是当查询需要排序的时候，数据库会话将使用这部分内存进行排序，当内存大小不足的时候，使用临时表空间进行磁盘排序。由于磁盘排序效率和内存排序效率相差好几个数量级，所以这个参数的设置很重要。这四个参数都是针对会话进行设置的，是单个会话使用的内存的大小，而不是整个数据库使用的。偶尔会看见有人误解了这个参数以为是整个数据库使用的大小，这是极其严重的错误。假如设置了 MTS，则 UGA 被分配在 large_pool_size，也就是说放在了共享内存里面，不同进程（线程）之间可以共享这部分内存。在这个基础上，我们假设数据库存在并发执行 server process 为 100 个，根据上面我们 4 个参数在 oracle8.1.7 下的默认值，我们来计算独立模式下 PGA 的大致大小。由于会话并不会经常使用 create_bitmap_area_size 、

bitmap_merge_area_size，所以我们通常不对四个参数求和。在考虑到除这四个参数外会话所保存的变量、堆栈等信息，我们估计为 2M，则 200 个进程最大可能使用 200M 的 PGA。

现在，根据上面这些假定，我们来看 SGA 实际能达到多少内存。在 1G 的内存的服务器上，我们能分配给 SGA 的内存大约为 400—500M。若是 2G 的内存，大约可以分到 1G 的内存给 SGA，8G 的内存可以分到 5G 的内存给 SGA。当然我们这里是以默认的排序部分内存 sort_area_size=64k 进行衡量的，假如我们需要调大该参数和 hash_area_size 等参数，然后我们应该根据并发的进程的数量，来衡量考虑这个问题。

事实上，通常我们更习惯通过直观的公式化来表达这样的问题：

OS 使用内存+SGA+并发执行进程数*(sort_area_size+hash_ara_size+2M) < 0.7*总内存

(公式是死的，系统是活的，实际应用的调整不必框公式，这不过是一个参考建议)

在我们的实际应用中，假如采用的是裸设备，我们可适当的增大 SGA(如果需要的话)。由于目前几乎所有的操作系统都使用虚拟缓存，所以实际上如果就算 SGA 设置的比较大也不会导致错误，而是可能出现频繁的内存页的换入与换出(page in/out)。在操作系统一级如果观察到这个现象，那么我们就需要调整内存的设置。

Ⅰ SGA 内参数设置

Log_buffer

对于日志缓冲区的大小设置，通常我觉得没有过多的建议，因为参考 LGWR 写的触发条件之后，我们会发现通常超过 3M 意义不是很大。作为一个正式系统，可能考虑先设置这部分为 log_buffer=1—3M 大小，然后针对具体情况再调整。

Large_pool_size

对于大缓冲池的设置，假如不使用 MTS，建议在 20—30M 足够了。这部分主要用来保存并行查询时候的一些信息，还有就是 RMAN 在备份的时候可能会使用到。如果设置了 MTS，则由于 UGA 部分要移入这里，则需要具体根据 server process 数量和相关会话内存参数的设置来综合考虑这部分大小的设置。

Java_pool_size

假如数据库没有使用 JAVA，我们通常认为保留 10—20M 大小足够。事实上可以更少，甚至最少只需要 32k，但具体跟安装数据库的时候的组件相关(比如 http server)。

shared_pool_size

这是迄今为止最具有争议的一部分内存设置。按照很多文档的描述，这部分内容应该几乎和数据缓冲区差不多大小。但实际上情况却不是这样的。首先我们要考究一个问题，那就是这部分内存的作用，它是为了缓存已经被解析过的 SQL，而使其能被重用，不再解析。这样做的原因是因为，对于一个新的 SQL（shared_pool 里面不存在已经解析的可用的相同的 SQL），数据库将执行硬解析，这是一个很消耗资源的过程。而若已经存在，则进行的仅仅是软分析（在共享池中寻找相同 SQL），这样消耗的资源大大减少。所以我们期望能多共享一些 SQL，并且如果该参数设置不够大，经常会出现 ora-04031 错误，表示为了解析新的 SQL，没有可用的足够大的连续空闲空间，这样自然我们期望该参数能大一些。但是该参数的增大，却也有负面的影响，因为需要维护共享的结构，内存的增大也会使得 SQL 的老化的代价更高，带来大量的管理的开销，所有这些可能会导致 CPU 的严重问题。

在一个充分使用绑定变量的比较大的系统中，shared_pool_size 的开销通常应该维持在300M 以内。除非系统使用了大量的存储过程、函数、包，比如 oracle erp 这样的应用，可能会达到 500M 甚至更高。于是我们假定一个 1G 内存的系统，可能考虑设置该参数为 100M，2G 的系统考虑设置为 150M,8G 的系统可以考虑设置为 200—300M。

对于一个没有充分使用或者没有使用绑定变量系统，这可能给我们带来一个严重的问题。所谓没有使用 bind var 的 SQL，我们称为 Literal SQL。也就是比如这样的两句 SQL 我们认为是不同的 SQL,需要进行 2 次硬解析：

select * from EMP where name = 'TOM';

select * from EMP where name = 'JERRY';

假如把'TOM' 和 'JERRY' 换做变量 V，那就是使用了 bind var，我们可以认为是同样的 SQL 从而能很好地共享。共享 SQL 本来就是 shared_pool_size 这部分内存存在的本意，oracle 的目的也在于此，而我们不使用 bind var 就是违背了 oracle 的初衷，这样将给我们的系统带来严重的问题。当然，如果通过在操作系统监控，没有发现严重的 cpu 问题，我们如果发现该共享池命中率不高可以适当的增加 shred_pool_size。但是通常我们不主张这部分内存超过 800M（特殊情况下可以更大）。

事实上，可能的话我们甚至要想办法避免软分析，这在不同的程序语言中实现方式有差异。我们也可能通过设置 session_cached_cursors 参数来获得帮助（这将增大 PGA）。

Data buffer

现在我们来谈数据缓冲区，在确定了 SGA 的大小并分配完了前面部分的内存后，其余的，都分配给这部分内存。通常，在允许的情况下，我们都尝试使得这部分内存更大。这部分内存的作用主要是缓存 DB BLOCK，减少甚至避免从磁盘上获取数据，在 8i 中通常是由 db_block_buffers*db_block_size 来决定大小的。如果我们设置了 buffer_pool_keep 和 buffer_pool_recycle，则应该加上后面这两部分内存的大小。

l 9i 下参数的变化

oracle 的版本的更新，总是伴随着参数的变化，并且越来越趋向于使得参数的设置更简单，因为复杂的参数设置使得 DBA 们经常焦头烂额。关于内存这部分的变化，我们可以考察下面的参数。事实上在 9i 中数据库本身可以给出一组适合当前运行系统的 SGA 相关部分的参数调整值（参考 V$DB_CACHE_ADVICE、V$SHARED_POOL_ADVICE），关于 PGA 也有相关视图 V$PGA_TARGET_ADVICE 等。

Data buffer

9i 中保留了 8i 中的参数，如设置了新的参数，则忽略旧的参数。9i 中用 db_cache_size 来取代 db_block_buffers，用 db_keep_cache_size 取代 buffer_pool_keep, 用 db_recycle_cache_size 取代 buffer_pool_recycle；这里要注意 9i 中设置的是实际的缓存大小而不再是块的数量。另外 9i 新增加了 db_nk_cache_size，这是为了支持在同一个数据库中使用不同的块大小而设置的。对于不同的表空间，可以定义不同的数据块的大小，而缓冲区的定义则依靠该参数的支持。其中 n 可以为 2、4、6、8、16 等不同的值。在这里顺便提及的一个参数就是 db_block_lru_latches，该参数在 9i 中已经成为了保留参数，不推荐手工设置。

PGA

在 9i 里面这部分也有了很大的变化。在独立模式下，9i 已经不再主张使用原来的 UGA

相关的参数设置，而代之以新的参数。假如 workarea_size_policy=AUTO（缺省），则所有的会话的 UGA 共用一大块内存，该内存由 pga_aggregate_target 设置。在我们根据前面介绍的方法评估了所有进程可能使用的最大 PGA 内存之后，我们可以通过在初始化参数中设置这个参数，从而不再关心其他 ”*_area_size” 参数。

SGA_MAX_SIZE

在 9i 中若设置了 SGA_MAX_SIZE，则在总和小于等于这个值内，可以动态的调整数据缓冲区和共享池的大小

SQL> show parameters sga_max_size

NAME TYPE VALUE

---

sga_max_size unknown 193752940

SQL>

SQL> alter system set **db_cache_size** = 30000000;

System altered.

SQL> alter system set **shared_pool_size** = 20480000;

System altered.

l Lock_sga = true 的问题

由于几乎所有的操作系统都支持虚拟内存，所以即使我们使用的内存小于物理内存，也不能避免操作系统将 SGA 换到虚拟内存（SWAP）。所以我们可以尝试使得 SGA 锁定在物理内存中不被换到虚拟内存中，这样减少页面的换入和换出，从而提高性能。但在这里遗憾的是，windows 是无法避免这种情况的。下面我们来参考在不同的几个系统下怎么实现 lock_sga

AIX 5L（AIX 4.3.3 以上）

logon aix as root
cd /usr/samples/kernel
./vmtune (信息如下**) v_pingshm**已经是**1**
./vmtune -S 1
然后 **oracle** 用户修改 **initSID.ora** 中 **lock_sga = true**
重新启动数据库

HP UNIX

Root 身份登陆
Create the file "/etc/privgroup": vi /etc/privgroup

Add line "dba MLOCK" to file
As root, run the command "/etc/setprivgrp -f /etc/privgroup":
$/etc/setprivgrp -f /etc/privgroup
oracle 用户修改 initSID.ora 中 lock_sga=true
重新启动数据库

SOLARIS (solaris2.6 以上)

8i 版本以上数据库默认使用隐藏参数 use_ism = true ，自动锁定 SGA 于内存中,不用设置 lock_sga, 如果设置 lock_sga =true 使用非 root 用户启动数据库将返回错误。

WINDOWS
不能设置 lock_sga=true,可以通过设置 pre_page_sga=true,使得数据库启动的时候就把所有内存页装载，这样可能起到一定的作用。

I 关于内存参数的调整

关于参数调整，是 oracle 的复杂性的一个具体体现。通常来讲，我们更倾向于让客户做 statspack 报告，然后告诉我们 os 监控的状况，在这些的信息的基础上，再向客户索取具体的详细信息以诊断问题的所在。系统的调整，现在我们通常采用从等待事件入手的方法。因为一个系统感觉到慢，必然是在某个环节上出现等待，那么我们从等待最多的事件入手逐步诊断并解决问题。

对于内存的调整，相对来说简单一些，我们首先可以针对数据缓冲区的大小来看。首先观察命中率

数据缓冲区命中率

```
SQL> select value from  v$sysstat where name ='physical reads';
     VALUE
----------
     14764
SQL>  select value from  v$sysstat where name ='physical reads direct';
     VALUE
----------
        50
SQL> select value from  v$sysstat where name ='physical reads direct (lob)';
     VALUE
----------
         0
SQL> select value from  v$sysstat where name ='consistent gets';
     VALUE
----------
    167763
```

```
SQL> select value from v$sysstat where name = 'db block gets';

     VALUE
----------
     14305
```

这里命中率的计算应该是

令 **x = physical reads direct + physical reads direct (lob)**

命中率 **=100 - (physical reads - x) / (consistent gets + db block gets - x)*100**

通常如果发现命中率低于 90%,则应该调整应用可可以考虑是否增大数据缓冲区

共享池的命中率

```
SQL> select sum(pinhits-reloads)/sum(pins)*100 "hit radio" from v$librarycache;

 hit radio
----------
 99.809291
```

假如共享池的命中率低于 95%,就要考虑调整应用（通常是没使用 bind var ）或者增加内存

关于排序部分

```
SQL>    select name,value from v$sysstat where name like '%sort%';
NAME                                                                  VALUE
---------------------------------------------------------------- ----------
sorts (memory)                                                         67935
sorts (disk)                                                               1
sorts (rows)                                                            7070

SQL>
```

假如我们发现 sorts (disk)/ (sorts (memory)+ sorts (disk))的比例过高，则通常意味着 sort_area_size 部分内存较小，可考虑调整相应的参数。

关于 log_buffer

```
SQL>  select name,value from v$sysstat
  2    where name in('redo entries','redo buffer allocation retries');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
redo entries                                                         2325719
redo buffer allocation retries                                            10
```

假如 redo buffer allocation retries/ redo entries 的比例超过 1%我们就可以考虑增大 log_buffer

通常来说，内存的调整的焦点就集中在这几个方面，更多更详细的内容，建议从 statspack 入手来一步一步调整。最后关于内存的调整，再强调这一点，**一定要结合操作系统来衡量，任何理论都必须要实践来检验。**在操作系统中观察 page in/out 状况，发现问题严重，应该考虑调小 SGA。

Ⅰ 32bit 和 64bit 的问题

对于 oracle 来说，存在着 32bit 与 64bit 的问题。这个问题影响到的主要是 SGA 的大小。在 32bit 的数据库下，通常 oracle 只能使用不超过 1.7G 的内存，即使我们拥有 12G 的内存，但是我们却只能使用 1.7G，这是一个莫大的遗憾。假如我们安装 64bit 的数据库,我们就可以使用很大的内存，我们几乎不可能达到上限。但是 64bit 的数据库必须安装在 64bit 的操作系统上，可惜目前 windows 上只能安装 32bit 的数据库，我们通过下面的方式可以查看数据库是 32bit 还是 64bit：

```
SQL> select * from v$version;
BANNER
----------------------------------------------------------------
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
PL/SQL Release 8.1.7.0.0 - Production
CORE    8.1.7.0.0       Production
TNS for 32-bit Windows: Version 8.1.7.0.0 - Production
NLSRTL Version 3.4.1.0.0 - Production
```

但是在特定的操作系统下，可能提供了一定的手段，使得我们可以使用超过 1.7G 的内存，达到 2G 以上甚至更多。在这里我们针对不同的平台下的具体实现方式做一个总结。

附：本文将继续完善，同时补上不同平台下增加 31bit oracle 使用的超过 1.7G 内存的方法。同时也将陆续写出其他文章，一起发表于 itpub 预计 9 月底推出的杂志上。如发现本文观点有误，或者您有其他补充见解，欢迎您发 mail 给我。

Mail: biti_rainy@itpub.net