ORACLE数据库性能优化的主要方法

张俊红 （湖北电信公司武汉分公司计算中心 430071）

摘要：本文主要以ORACLE数据库性能优化为出发点，围绕数据库参数、I/O分布、应用系统、网络负载等方面讨论了ORACLE数据库性能调整的主要几个方面，由此了解如何分析和改进数据库性能，如何维护一个性能满意的数据库等。

关键词： ORACLE数据库 参数 调整 命中率

Oracle作为一个大型数据库系统，对许多不同的资源如CPU、网络、内存、磁盘等都有复杂的调配和使用，所以需要各组人员（数据库管理员、系统管理员、应用开发人员、网络管理员等）交叉进行性能优化，从而保证数据库各方面以及它的资源尽可能有效的运行。

这里假设服务器硬件、操作系统和网络带宽均没有引起严重的性能问题，在ORACLE环境下，数据库优化调整原则上按以下几步反复进行：

· 有条理的采用各种监测手段收集数据，发现性能问题；

· 做出令人信服的调整（一次只建议改动一两处）；

· 收集统计信息，得到调整后的性能。

重复以上操作直至性能调整满意为止。

我们主要可以从以下几个方面入手进行监测调整：

1 数据库参数的监测调整

在ORACLE数据库的参数文件init.ora中，设置了数据库启动运行后的各种环境参数设置，用Show parameters命令可观察数据库的设置参数，并可运行Utlbstat/Utlestat来收集数据库服务器的各种统计数据。有的参数需要仔细观察，恰当设置，主要有：

1.1 系统全局区SGA

用Show SGA命令可直接查看数据库的SGA设置，总的来说，分配给SGA的内存越大，数据库运行的就越快，因为数据库绝大多数所需的信息都可以直接从内存得到，而不必去访问磁盘。所以要综合考虑服务器上其他进程的开销来设置SGA大小，一般应设为可用内存的50%以上。

(1) 数据库缓冲区。数据库缓冲区用于存放最近从磁盘读取的数据块，它也用来存放尚未存入磁盘的修改。初始化参数文件INIT.ORA中的db_block_buffers参数决定了数据库缓冲区的大小。可以查看V$SYSSTAT来计算数据库缓冲区命中率：

```
SELECT name,value
  FROM v$sysstat
 WHERE name in ('db block gets','consistent gets','physical reads');
```

NAME	VALUE
db block gets	3248568
consistent gets	348408140
physical reads	970380

命中率计算公式

Hit Ratio=1-(physical reads/(db block gets+consistent gets)

上例命中率为99.8%，已非常理想。数据库缓冲区命中率应尽可能高，应在80%以上，其高低依赖于系统环境、优化目标等，如果其命中率较低，就应调高参数文件中db_block_buffers的设置。

(2) 共享缓冲池区。共享缓冲池区用于存放SQL、PL/SQL包和过程，以及锁、数据字典、游标等信息的内存区，初始化文件INIT.ORA中的shared_pool_size参数决定了共享缓冲池区的大小。查看V$LIBRARYCACHE可以了

解SQL、PL/SQL信息的命中率：

```
SELECT sum(pins) 'PINNED',sum(reloads) 'RELOADED'
FROM v$librarycache;
PINNED      RELOADED
..........  ..........
20283212        11755
```

命中率Hit Ratio=reloaded/pinned，上例命中率为99.95%。RELOADED的理想值为0，其高低依赖于系统环境及开发人员的编程水平等，如果其命中率较低，一方面应调高参数文件中share_pool_size的设置，另一方面，优化应用程序。

查看V$ROWCACHE可以检测数据字典的活动情况：

```
SELECT sum(gets) "read requests",
  sum(getmisses) "reads not in memory"
FROM v$rowcache;
read requests          reads not in memory
.....................  ...................................
13579542                        5623
```

命中率Hit Ratio= reads not in memory/ read requests，上例命中率为99.96%，Oracle建议它不能低于85%，如果它持续增长，就要通过增大shared_pool_size来增大数据字典内存。

(3) REDO缓冲区。REDO缓冲区中保存写入到REDO日志文件前的所有REDO信息。初始化文件INIT.ORA中的log_buffer参数决定了REDO缓冲区的大小。可以查看V$SYSSTAT来计算REDO缓冲区的空间请求次数：

```
SELECT name,value
FROM v$sysstat
WHERE name='redo log space requests';
NAME                              VALUE
................................  ...............
redo log space requests           132
```

以上查询结果尽可能为0。通过一段时间监测，若此值持续增长，需增大INIT.ORA中的log_buffer，减少日志缓冲区争用。

1.2 rollback_segments(回滚段)

初始化文件INIT.ORA中的rollback_segments参数将数据库定义的private回滚段置为online。在所有的修改(DML操作)提交以前，回滚段中保存恢复该事务所需的全部信息，如果回滚段大小、个数设置不合理，会影响事务的等待和I/O争用。

一般每个回滚段大小设置成至少足够处理一个完整事务，并建立大的回滚段供大的事务专门指定使用。回滚段的数量建议以并发事务个数来估算，1~16个并发事务至少需要4个回滚段；17~32个并发事务至少需要8个回滚段；33以上个并发事务至少需要10个以上回滚段，但不应超过50个。

1.3 SORT_*

初始化文件INIT.ORA中的SORT_*系列参数指定用于排序的临时内存区域的大小,排序区域用于创建索引、处理分组功能-比如order by,group by等，合理设置可加快排序、统计速度，一般情况下，sort_*至少设置为需排序的最大表的二分之一大。

1.4 DML_LOCKS

初始化文件INIT.ORA中的DML_LOCKS参数决定数据库表能同时被修改的锁的入口数，例如：三个用户正同时修改两个表中的数据，则入口数至少为6。所以并发事务越多，事务同时修改的表数目越多，DML_LOCKS参数设的值也应越大。

当然还有许多其他参数，我们应根据情况分析和设置。

2 I/O监测与调整

优化磁盘I/O是一个长期的任务，我们可以通过操作系统命令发现磁盘存取的统计信息，例如，在AIX环境下主要以数据库应用为主的服务器上，用IOSTAT命令，可查看到：

```
tty:tin   tout   avg-cpu:  % user   % sys    % idle % iowait
   0.0   95.7              16.4      3.5      40.1     40.0
Disks:% tm_act  Kbps      tps    Kb_read Kb_wrtn
hdisk1  51.9    911.1     89.7   4444    516
hdisk0   4.0     29.2      4.8   0       146
cd0      0.0      0.0      0.0   0       0
```

此例中，系统比较平稳，空闲率idle达40.1%，但I/O等待占CPU的40%，略微偏高，其两个磁盘hdisk0、hdisk1的读Kb_read、写Kb_wrtn差别也较大，如果这种情况一直存在，说明两个磁盘hdisk0、hdisk1上的数据库物理文件分布不合理，有必要进行调整。

我们还可以通过数据库命令监测数据库物理文件的I/O，视图V$FILESTAT包含数据库最后一次启动以来物理读、写的统计信息，命令如下：

```
SELECT  name,phyrds,phywrts
    FROM  v $ datafile,v$filestat
    WHERE v $ datafile.file # =v $ filestat.file#
NAME                PHYRDS        PHYWRTS
.....               .....         .....
/dev/rrtemp112_1      1             623
/dev/rrwh112_idx    700012          84911
/dev/rrwh112_1       5454           12130
/dev/rrwh112_2        20              0
/dev/rrwh112_3      127497          49509
```

PHYRDS和PHYWRTS之和即该数据文件总的I/O，从而了解每个数据库物理文件被访问的频次。

建立和优化数据库物理文件分布应遵循以下原则：

· 为表和索引分别建不同的表空间，置于不同的磁盘上；

· 确定最常访问的表、索引，以及它们的表空间，并将它们分开放于单独的磁盘上；

· 确定最常被同时访问的表空间，利用分布在不同磁盘上的物理文件组建此表空间；

· 可能的话，将回滚段分放在多个单独的表空间中，以及多个独立的磁盘中；

· 将redo日志文件及转储日志文件尽量置于不同的磁盘上；其和回滚段也尽量存于不同磁盘上；

· 如果可能，在多个磁盘控制器之间分离物理磁盘，以获得更好的I/O容量。

3 应用系统开发

开发一个优秀的应用系统对提高Oracle数据库系统效率有着及其关键的作用，我们可以从许多方面着手来使程序执行效率更高、更好，这需要开发人员有深厚的编程经验，并对ORACLE数据库有较好的理解等，一般在以下方面需要重视：

3.1 优化SQL语句

较好的SQL语句可以提高SGA区的命中率，减少I/O请求数目，减少对网络带宽的占用等。优化SQL语句主要从两方面着手，1、应尽量使用索引；2、用规范的格式和访问数据库对象的一致顺序书写SQL语句，使得相同访问的SQL代码完全相同，以提高共享缓冲池的命中率。

需要的话，还可使用ORACLE提供的工具EXPLAIN PLAN和TKPROF来帮助开发员优化SQL语句。EXPLAIN PLAN用于检查ORACLE如何执行SQL语句的（它并不真正执行），TKPROF更能进一步提供执行SQL的磁盘I/O、CPU时间、调用分析等（它真正执行此SQL语句）。其实开发人员真正了解数据及其用途，有些情况下，开发人员比优化器更能选择一个有效的执行方法。

3.2 索引使用的一些原则

在大型应用开发或表较大的情况下，使用索引可以极大减少数据库读写次数，从而提高数据库访问速度，所以，索引在应用程序开发中经常被使用，但如何正确使用索引以发挥它的优势需遵循以下原则：

· 在主键（primary key）的索引方面，不应有超过25%的列成为主键，而只有很少的普通列，这会浪费索引空间；

· 在索引的使用效率方面，当选择数据少于全表的20%，并且表的大小超过ORACLE的5个数据块时，使用索引才会有效，否则用于索引的I/O加上用于数据的I/O就会大于做一次全表扫描的I/O；

· 使用索引尤其应当注意的是，在表连接操作时的驱动表/被驱动表的关系。ORACLE核心使用至底向上、从右至左的规则，如：FROM子句中的最后一个表才是ORACLE用做为驱动表的表；WHERE子句的最后一个条件中所含的列，它所属的表才是最先被引用的表。总之，在FROM子句中，将表名按被驱动表-驱动表排序，在WHERE子句中，将条件语句按最少约束-最多约束排序。

· 当指向被删除行的索引所占空间超过总索引空间的20%时，就应删除并重建索引，以节省空间，提高性能。

3.3 网络负载

在完成一定的任务量情况下，应尽量减少客户机与服务器之间传递的数据量，合理地分配任务处理。

· 采用ORACLE自身的完整性约束机制，而非让应用程序来检查完整性，可大大减少客户机/服务器间的网络访问。

· 使用触发器、过程和包。开发人员利用触发器、过程和包把应用逻辑移到数据库服务器执行以便减少网络I/O，提高性能。

· 使用显式游标。当SELECT语句使用隐式游标时将增加网络调用，尤其多次执行相同的SQL语句时，显式游标不需传输这些额外的包重新打开游标，从而避免了不必要的网络传输。

4 其他优化手段

还有其他一些手段对提高性能也大有益处，但需要

ORACLE的特殊选件，或需要系统平台的特有支持。

4.1 并行查询技术

ORACLE的并行查询选件可以实现查询的并行查询，充分利用系统的硬件资源（如多个CPU、多个I/O设备），动态提高系统性能，特别是能改善大型系统中的复杂查询效率，对于任务要求时效高的应用（如DSS和OLTP系统）将是一个可行的解决方法。它要求硬件系统具有多个CPU、多个硬盘、高带宽的数据I/O、大容量的内存等。

4.2 分区（patition）技术

分区使数据库管理员能把较大的数据库对象分解成更易于管理的较小段，对于特大容量的数据对象使用分区技术对提高数据库可用性、优化数据存储和访问将更有意义，ORACLE8提供的分区选件使分区技术达到一个更高的水平。

例如采用范围分区创建如下客户表：

```
CREATE table customes (
  Customes_name    varchar2(16),
  Customes_ID      varchar2(18),
  Customes_birth   varchar2(8),
  .
  .
  )
PATITION by range(customes_birth)
  (patition customes_1960 values less then ('19600000')
   tablespace ts_1960,
   (patition customes_1980 values less then ('19800000')
   tablespace ts_1980,
   (patition customes_now values less then ('20000000')
   tablespace ts_now);
```

即将此客户表分段（按出生年代）放在不同的表空间上了。

其后，可通过监测统计其访问频率等合理变动分区，安排其磁盘分布。

5 调整优化举例

我单位特服1000号系统于2000年5月完成系统平台的集成，它采用的是IBM小型机和ORACLE数据库，2000年8月上线运行，到10月份期间系统性能表现出一些缓慢现象，从数据库方面我们做了如下工作。

当时，我们利用Oracle数据库的Utlbstat/Utlestat工具，在数据库运行高峰时间段（2小时）采集了若干数据库参数的统计数据，其中一部分如下：

LIBRARY	GETS	GETHITRATI	PINS	PINHITRATI	RELOADS	INVALIDATI
.....	.....	.....	.....	.....	.....	.....
BODY	17	1	17	1	0	0
CLUSTER	0	1	0	1	0	0
INDEX0	0	1	0	1	0	0
OBJECT	0	1	0	1	0	0
PIPE	0	1	0	1	0	0
SQL AREA	46492	98	214809	988	765	0
TABLE/PROCED	2680	996	9774	997	16	0
TRIGGER	2	1	2	1	0	0

8 rows selected.

通过查看其GETHITRATI（表示共享缓冲池区中SQL、PL/SQL信息的命中率）统计数据，基本都在98%以上，可初步说明该数据库此时的共享缓冲池区内存配置基本合理，但查看V$SYSSTAT计算REDO缓冲区的空间请求次数：

```
SELECT name,value
FROM v$sysstat
WHERE name='redo log space requests';
NAME                                    VALUE
.....                                   .....
redo log space requests                 185
```

发现为185次，按建议为0次考虑，将数据库参数文件中的log_buffer值由163840(bytes)调为327680(bytes)，通过一段时间监测，此值不再持续增长，可见减少了日志缓冲区争用。

另一方面，在与应用开发者交流中发现，特服系统运行高峰期，其并发事务可达到36个左右，据此，将原来的6个回滚段扩到12个，以减少对回滚段的争用。

通过以上两方面的优化，数据库性能有所改善，但随着应用功能模块的增加和更新，还需要我们不断的观察调整数据库的设置。■

参考文献

1 *ORACLE8优化技术，Michael J.Corel著，刘晓霞、王联华等译，机械工业出版社。*

2 *ORACLE数据库开发指南，singh,leigh,zafian,et al著，史森、夏丽丽译，清华大学出版社。*

3 *ORACLE8 SERVER REFERENCE。*

4 *ORACLE8 SQL REFERENCE。*