SUNGARD 金仕达

Oracle DBA 手册

组织标准软件过程文档

文档标识

文档名称	Oracle DBA 手册 V1.0
状况	☐ 草案 ☐ 评审过的 ☐ 更新过的 ☑ 定为基线的
模板版本号	<>

文档修订历史

版本	日期	描述	文档所有者
V0.1	2007/6/19	起草	林强、王凌洁
V1.0	2007/6/29	评审	杨疆湖、林强、汤成

此版本文档的正式核准

姓名	签字	日期

分发控制

副本	接受人	机构

目 录

前言

调整数据库的性能必须有一个明确的目标，总的来说可以是以下的 几个目标之一或多个：在调整ORACLE性能之前，首先要有一个性能良好的应用设计及高效的SQL语句，在此基础上调整ORACLE性能的过程有三步：

1) 调整内存分配；
2) 调整I/O；
3) 调整资源争用；

因此，根据上述的原则并根据自己的工作经验，认为对数据库的优化大体上可分为如下几个阶段进行：安装数据库时，对数据库的数据块大小进行确认。此参数在数据库安装之后就不能通过修改初始化参数进行修改或重新创建控制文件进行修改，要改变该值，唯一的方法是重新安装数据库。

在数据库安装完毕之后，对数据库初始化参数进行修改。一个经过调优过的参数，对一个系统而言，可作到事半功倍的功效。调整主机的硬件性能和操作系统的软件性能，使之配合数据库，发挥最大的性能。

进行应用系统的物理设计。

进行应用程序的编写时，对SQL语句的优化。

在试运行时对系统的物理设计以及应用程序的调整。

在系统运行过程中，通过对系统的监控，认识到系统的瓶颈，对系统再进行一次性能调整，此步骤在今后的系统运行中可能要反复多次。

1、设计时优化

1.1 规划磁盘 I/O

I/O瓶颈是最大性能问题。跨所有可用的磁盘均匀地平衡I/O，可以减少磁盘存取的时间。有一个适当的物理布局，表空间（和它们的数据文件）应尽可能地分散在不同的磁盘上。把表空间分散在不同的盘上能够消除或者至少能够减少磁盘争用。磁盘争用在多用户或程序试图同时访问同一张磁盘时发生。例如，你有两个必须经常连结在一起的表，例如 DEPA RTMENTS 和EMPLOYEES表，它们的表空间通常要被分散在两张不同的磁盘上，因为要存取相同磁盘上的表或者索引，所以会导致对同一资源的争用。理想情况就是这四个段放在四张不同磁盘上的四个表空间里。另一种方法是和聚簇那样，把这些段放在相同的磁盘上。你可以研究Oracle版的聚簇技术。

1.2 对数据量估计

1.2.1 对系统中大数据量表要进行分类，如流水帐的表，每天都是增量几十万条数据。这类表要用分区功能处理。提高数据查询速度，对数据库的数据分散管理，对于表的所有操作均适应于分区表的每个分区。

1.2.2 采用定期转存方式，数据到备份库方式，使前台的表保存一定的数据量，优化数据表结构。

1.3 对只读类数据的设计规化

对只读类数据存储优化，如历史报表数据等，表空间或表的参数 pctused 缺省=40%，此参数是每个数据块参能够存储数据空间的大小，表空间参数 pctfree 缺省=10%,此参数是用来 UPDATE 操作时用的。

如果把表空间参数 pctused 设置=90%, 把 pctfree 设置=5%，大大提高数据块存储数据能力，如原来读取 10000 条读录要读 1000 个 IO 块,现在只要读 400 个 IO 块。

做了 dba_object 表做了一个测试，压缩比例压缩为原来的 1/3，当然压缩比根据不同的表会有不同，字段值重复越少压缩比越高，对我们应该还是比较适合。）

注意：此参数调整适用于只读数据

2、部署时调优

1.4 系统参数调整

1.4.1 调整 LGWR

每次 I/O 写的大小依赖于 LOG 缓冲区的大小，该大小由 LOG BUFFER 所设置，缓冲区太大会延迟写操作，太小可能导致频繁的小的 I/O 操作。如果 I/O 操作的平均大小很大，那么 LOG 文件就会成为瓶颈，可以使用 STRIPE REDO LOG 文件避免这个问题。

调整 INIT.ORA 中的参数 LOG_BUFFER 以调节 LOG 缓冲区大小，把 REDO LOG 文件分为几个文件，放在不同的磁盘上。

1.4.2 调整 DBWR 写进程

参数名:db_writer_processes:

说明: 一个例程的数据库写进程的初始数量。如果使用了 DBWR_IO_SLAVES, 则只能使用一个数据库写进程。可以创建多个数据库写进程。

值范围: 1 - 10

默认值: 1

查询参数的 SQL：

```
select * from v$parameter where name like 'db_writer_processes%'
```

Oracle 提供以下方法以防止 DBWR（数据库写进程）活动成为瓶颈：使用异步 I/O 异步 I/O 允许进程继续处理下一个操作，而不必等待在发出写后，最小化了空闲时间，因而改善了系统性能。Solaris 支持原始设备和文件系统数据文件的异步 I/O。使用 I/O 从属 I/O 从属（slave）是专用的进程，其惟一功能是执行 I/O。它们代替 Oracle 7 的多个 DBWR 特性（实际上它们是多个 DRWR 的概括，可以由其它进程分布）。不管异步 I/O 是否可用，它们都可以操作。若设置的话，它们被从 LARGE_POOL_SIZE 分配，否则从共享内存缓冲区分配。初始化参数控制了 I/O 从属的行为，其中 DISK_ASYNCH_IO 和 TAPE_ASYNCH_IO 允许分别为磁盘和磁带设备关闭异步 I/O（因为每个进程类型的 I/O 从属缺省为 0，除非明确设置，否则没有 I/O 从属被发布）。

若 DISK_ASYNCH_IO 或 TAPE_ASYNCH_IO 无效，则 DBWR_IO_SLAVES 应该设置大于 0，否则 DBWR 将成为一个瓶颈。在这种情况下，Solaris 上 DBWR_IO_SLAVES 的最佳值为 4，而在 LGWR_IO_SLAVES 的情况下，发布的从属不应该超过 9 个。DB_WRITER_PROCESSES 代替 Oracle 7 的参数 DB_WRITERS，指定某实例的数据库写进程的初始数量。若使用 DBWR_IO_SLAVES，则只有一个数据库写进程被使用，而不管 DB_WRITER_PROCESSES 的设置。

如果工作在一个可以支持多进程的操作系统上，应该马上改变这个参数的值，以获得更好的性能。许多 DBA 有个错误的概念，认为此参数受 CPU 数量的限制，实际并非如此。建议为每个数据文件设置两个数据库写入进程。

1.4.3 调整 LGWR I/O

每次 I/O 写的大小依赖于 LOG 缓冲区的大小，该大小由 LOG BUFFER 所设置，缓冲区太大会延迟写操作，太小可能导致频繁的小的 I/O 操作。如果 I/O 操作的平均大小很大，那么 LOG 文件就会成为瓶颈，可以使用 STRIPE REDO LOG 文件避免这个问题。

调整 INIT.ORA 中的参数 LOG_BUFFER 以调节 LOG 缓冲区大小，把 REDO LOG 文件分为几个文件，放在不同的磁盘上。

1.4.4 调整 Dml_locks

用户一次可对表设定锁的最大数目，如果有三个用户修改 6 个表，则需 1 8 个 DML 锁来实现并行操作，如果设定 DML_LOCKS 不够大，操作时执行将中断，你可以通过你的应用程序的操作规模和最大的并行使用的用户数来估算系统所需要的 DML_LOCKS 的值，如果在系统运行中经常发生表锁死的现象，就应该考虑加大该值。

设置 DML_LOCKS =（并行用户的最大数）×（表的数量）。例如，假如你有 U 个用户和 T 个表，那么应设置 DML_LOCKS=（U×T），还可以加上一些余量，例如 10%。

查询参数的 SQL：

```
select * from v$parameter where name like 'dml_locks%'
```

1.4.5 调整 Open_cursors

这个参数的值决定了一个用户同时可以打开的游标数目。可以将游标想象成 Oracle 分配给用户的用于处理 SQL 语句的内存空间。这个参数的初始值太小，当用户无法打开游标时，应用程序将停止运行。提示该参数的值应设得尽可能的大，否则应用程序将由于无法打开游标而停止运行。

建议是设值为：2500-3000

查询参数的 SQL：

```
select * from v$parameter where name like ' open_cursors%'
```

1.4.6 调整 Data_links

参数名:open_links:

说明: 指定在一次会话中同时打开的与远程数据库的连接的最大数量。该值应等于或超过一个引用多个数据库的单个 SQL。

语句中引用的数据库的数量, 这样才能打开所有数据库以便执行该语句。

值范围: 0 - 255 (如果为 0, 不允许分布式事务处理)。 默认值: 4

建议是用按实际 data_link 数来设置

查询参数的 SQL：

```
select * from v$parameter where name like 'open_links%'
```

1.4.7 调整系统进程数 Processes

这个参数的值决定了能同时访问 Oracle 数据库的进程数。这个参数的初始值为 160，但是这个值太小了，因为 Oracle 系统本身所使用的系统进程以及由某个进程所产生的新进程都将占用可使用的进程数。除非想限制使 Oracle 数据库的用户数，或者由于机器的性能不足，否则应将该参数的值尽可能的大。

建议修改后值不能小于：100+(并发用户数*2)

查询参数的 SQL：

```
select * from v$parameter where name like 'processes%'
```

1.4.8 调整会话 Sessions

说明:指定用户会话和系统会话的总量。默认数量大于 PROCESSES, 以允许递归会话。建议修改后值为(Processes*120%)

查询参数的 SQL:

```
select * from v$parameter where name like Sessions%'
```

1.4.9 调整事务 transactions

说明:指定并行事务处理的最大数量。如果将该值设置得较大, 将增加 SGA 的大小, 并可增加例程启动过程中分配的回退段的数量。默认值大于 SESSIONS, 以实现递归事务处理。 值范围: 一个数值。默认值: 派生 (1.1 * SESSIONS)

建议修改后值为(110% * SESSIONS)

查询参数的 SQL:

```
select * from v$parameter where name like 'transactions%'
```

1.4.10 调整 Job 数量

参数名字: job_queue_processes:

它指定每个例程的 SNP 作业队列进程的数量 (SNP0, ... SNP9, SNPA, SNPZ)。要自动更新表快照或执行由 DBMS_JOB 创建的请求。建议是用按实际 JOB 数来设置。

值范围: 0 到 36

查询参数的 SQL:

```
select * from v$parameter where name like ' job_queue_processes %'
```

1.4.11 调整读取数据最大块数

参数名字: db_file_multiblock_read_count:

说明: 在涉及一个完全连续扫描的一次 I/O 操作过程中读取的块的最大数量。默认值:是 8, 最大值 128。建议设:16 或 32

查询参数的 SQL:

```
select * from v$parameter where name like ' db_file_multiblock_read_count%'
```

1.4.12 设置 lock_sga:

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,这样对效率有好处在不支持这种功能的平台上, 该值将被忽略。

值范围: TRUE | FALSE ,默认值: FALSE

```
select * from v$parameter where name like 'lock_sga%'
```

建议设置成 true

1.4.13 设置 timed _ statistics

这个参数告诉 Oracle 记录下系统运行时的一些附加信息，这些信息在调试环境下是非常有用的，但是在实际运行环境下，建议将这个功能关闭，因为这个附加功能将导致系统性能的下降。

没有使用时设置成 FALSE

1.4.14 调整最大回退段数

Max_rollback_segments:

说明: 指定 SGA 中高速缓存的回退段的最大大小。该数值指定一个例程中可同时保持联机状态（即状态为 INUSE）的回退段的最大数量。

值范围: 2 -65535

默认值: 最大值 (30, TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT)

```
SELECT * FROM V$PARAMETER WHERE NAME LIKE 'max_rollback_segments%'
```

查询每个回滚段可以服务的事务数

```
SELECT * FROM V$PARAMETER WHERE NAME LIKE 'transactions_per_rollback_segment%'
```

建议值:（最大用户并发数 * transactions_per_rollback_segment）*10%

1.4.15 设置 db_cache_advice:

DBA_CACHE_ADVICE 参数说明: 为预计各种容量的高速缓存的行为而启用和禁用统计信息收集。在 V$DB_CACHE_ADVICE 视图中收集信息。值范围: OFF-- 关闭咨询, 并且不为咨询分配内存; ON-- 打开咨询 (即: 将带来 CPU 和内存开销); READY-- 关闭咨询, 但保留

分配给咨询的内存。V$DB_CACHE_ADVICE 是显示了 20 个可能的缓冲 cache 大小的“脱靶”率（范围从当前大小的 10%到 200%）。

警告：当你把 DBA_CACHE_ADVICE 设置为 ON 时，Oracle 会从共享的池中“窃取”RAM 页面，这往往严重影响到了库 cache。例如，如果设置 DB_CACHE_SIZE 为 500m，Oracle 就会从共享池中窃取相当多数量的 RAM 空间。为了避免这个问题，我们应该在 INIT.ORA 文件中把 DBA 设置为 DB_CACHE_ADVICE=READY。这样，Oracle 会在数据库启动时预分配 RAM 内存。

默认值: OFF

建议设置成 READY

1.4.16 调整优化模式 optimizer_mode

这个参数值的设定将影响到 Oracle 处理 S Q L 语句的方式。这个参数有以下 5 种设定值：RBO、CBO、FIRST_ROWS 或 ALL_ROWS 这两个设定值将使 Oracle 使用基于代价的优化方法来执行 SQL 语句。在这种优化方法下，Oracle 在执行 S Q L 语句的时候，将重点考虑表的大小进行优化。例如，数据库知道用户的 customer 表有 10 000 行纪录，而 phone_type 表有 3 行纪录，那么在执行 S Q L 语句的时候，系统将考虑这些因素。• RULE 这个设定值如果使用， Oracle 将使用内部的度量系统来决定 S Q L 语句将如何被执行。比如，如果一张表有索引，则 O r a c l e 在执行 S Q L 语句的时候就将使用索引进行优化。• CHOOSE 这个值是缺省设置。当用户有必需的信息时，则告诉 O r a c l e 使用基于成本优化的方法。反之，则使用基于规则的方法 8. 对已存在的系统使用哪种方法进行优化，建议进行性能统计分析，然后改变系统所使用的优化方法。自 Oracle 第 5 版、第 6 版以来，许多系统用基于规则的优化方法很好地完成了优化。实践证明，如果在没有进行测试的情况下，贸然改变系统所使用的优化方法将对系统造成灾难性的影响。提示如果要使用基于规则的优化方法进行系统优化，可将初始化参数文件中参数 optimizer_mode 的值设为 rule。在决定对系统所使用的优化方法进行改变之前，最好先建立一个测试系统对系统的性能进行评测，然后根据评测结果决定是否改变优化方法。注意：要注意 analyzer 系统时间。

1.4.17 调整优化模式 optimizer_index_caching

说明: 调整基于成本的优化程序的假定值, 即在缓冲区高速缓存中期望用于嵌套循环联接的索引块的百分比。它将影响使用索引的嵌套循环联接的成本。将该参数设置为一个较高的值,可以使嵌套循环联接相对于优化程序来说成本更低。

值范围: 0 - 100 %。 默认值: 0

```
select * from v$parameter where name like 'optimizer_index_caching%'
```

建议设置成 90

1.4.18 调整优化模式 optimizer_index_cost_adj

说明：在考虑太多或太少索引访问路径的情况下，可以用来优化优化程序的性能。该值越低，优化程序越容易选择一个索引。也就是说，如果将该值设置为 50%，索引访问路径的成本就是正常情况下的一半。

值范围：1 -10000 ，默认值：100（一个索引访问路径的常规成本）

```
select * from v$parameter where name like 'ptimizer_index_cost_adj%'
```

将这个参数设小表名索引代价要小于全表扫描，这样就使得使用 CBO 进行成本计算时更倾向于使用索引扫描。建议把这个参数设置成 30 到 50。

1.4.19 调整优化模式 optimizer_max_permutations

对于多表连接查询，如果采用基于成本优化(CBO)，ORACLE 会计算出很多种运行方案，从中选择出最优方案。这个参数就是设置 oracle 究竟从多少种方案来选择最优。如果设置太大，那么计算最优方案过程也是时间比较长的。Oracle805 和 8i 默认是 80000，8 建议改成 2000。对于 9i，已经默认是 2000 了。

1.4.20 并行优化

- parallel_adaptive_multi_user:

  说明:启用或禁用一个自适应算法，旨在提高使用并行执行方式的多用户环境的性能。通过按系统负荷自动降低请求的并行度,在启动查询时实现此功能。当 PARALLEL_AUTOMATIC_TUNING = TRUE 时，其效果最佳。
- Parallel_automatic_tuning:

  说明：如果设置为 TRUE，Oracle 将为控制并行执行的参数确定默认值。除了设置该参数外,你还必须为系统中的表设置并行性。

  值范围：TRUE | FALSE

  默认值：FALSE

  如 PARALLEL_AUTOMATIC_TUNING 是设置 FALSE，Oracle 分配并行执行缓冲在上 SHARED_POOL.

  在 ORACLE10 缺省安装就是 400M 建议 SHARED_POOL=200M-400M
- 参数 parallel_max_servers_parameter

  Oracle 一个显著的加强是自动决定 OPQ 并行的程度。由于 Oracle 清楚服务器中 CPU 的数量，它会自动分配合适的子进程的数量来提升并行查询的响应时间。当然，会有其它的外部因素，比如表的划分以及磁盘输入/输出子系统的布局等，但是根据 cpu_count 来设置 parallel_min_servers 参数将给 Oracle 一个合理的依据来选择并行的程度。

  由于 Oracle 的并行操作严重依赖服务器上 CPU 的数量，parallel_max_servers 会被设置成服务器上 CPU 的数量。如果在一台服务器上运行多个实例，则默认值太大了，会导致过度的页面交换和严重的 CPU 负担。并行的程度还依赖于目标表中分区的数量，因此 parallel_max_servers 应该设置成足够大以允许 Oracle 为每个查询选择最佳数量的并行子查询。
- parallel_threads_per_cpu:

  说明：说明一个 CPU 在并行执行过程中可处理的进程或线程的数量,并优化并行自适应算法和负载均衡算法。如果计算机在执行一个典型查询时有超负荷的迹象，应减小该数值。

  值范围：任何非零值。 默认值：根据操作系统而定 （通常为 2）

  CPUS = 4 建议 PARALLEL_THREADS_PER_CPU = 2

PARALLEL_MAX_SERVERS = 64
CPUS = 8 建议 PARALLEL_THREADS_PER_CPU = 4
PARALLEL_MAX_SERVERS = 256

- **parallel_max_servers:**

说明：指定一个例程的并行执行服务器或并行恢复进程的最大数量。如果需要，例程启动时分配的查询服务器的数量将增加到该数量。
值范围：0 -256
默认值：由 CPU_COUNT, PARALLEL_AUTOMATIC_TUNING 和 PARALLEL_ADAPTIVE_MULTI_USER 确定

parallel_min_servers
说明：指定为并行执行启动例程后，Oracle 创建的查询服务器进程的最小数量。
值范围：0 - PARALLEL_MAX_SERVERS。
默认值：0
CPUS = 4 建议 PARALLEL_MAX_SERVERS = 64
CPUS = 8 建议 PARALLEL_MAX_SERVERS = 256

- **改表或视图并行度**

```
ALTER TABLE department1 PARALLEL 10;
ALTER TABLE department2 PARALLEL 5;
CREATE VIEW current_sales AS SELECT /*+ PARALLEL(P, 20) */ * FROM sales P;
```

1.4.21 调整 Checkpoints

一个 checkpoint 是 ORACLE 自动执行的一种操作，当检查点操作时，数据库中的所有缓冲区会写回磁盘，所有数据库的控制文件被更新。Checkpoint 频繁发生会加快数据库的恢复，但是增加了 I/O 次数，会降低系统的性能。

修改 INIT.ORA 文件中的参数 LOG_CHECKPOINT_TIMEOUT 和 LOG_CHECKPOINT_INTERVAL ，增大这两个参数会减少 I/O 次数，提高系统性能。

1.5 内存调整

1.5.1 SGA 调优

调优后的 SGA 区所占用的内存不应超过系统物理内存大小的 60 %。

```
select * from v$parameter where name like ' sga_max_size%'
```

建议设置 SGA 最大值为系统物理内存大小的 50%--60 %。

要思考的问题：

1) 除去 OS 和一些其他开销，能给 ORACLE 使用的内存有多大
2) oracle 是 64bit or 32 bit，32bit 通常 SGA 有 1.7G 的限制(某些 OS 的处理或者 WINDOWS 上有特定设定可以支持到 2G 以上甚至达到 3.7G)
3) sort_area_size 、 hash_area_size 这两个参数在非 MTS 下都是属于 PGA ，不属于 SGA,是为每个 session 单独分配的，在我们的服务器上除了 OS + SGA,一定要考虑这两部分 。

OS 使用内存+ SGA + session*(sort_area_size + hash_area_size + 2M) < 总物理 RAM 为好

这样归结过来，假定 oracle 是 32 bit ,服务器 RAM 大于 2G ，注意你的 PGA 的情况，，则建议

shared_pool_size + data buffer+log_buffer +large_pool_size + java_pool_size < 1.6G

注意满足上面的原则的基础上可以参考如下设置：

1） 如果 512M 内存:

建议 shared_pool_size = 50M, data buffer = 200M

2） 如果 1G 内存:

建议 shared_pool_size = 100M , data buffer = 500M

3） 如果 2G :

shared_pool_size = 150M ,data buffer = 1.2G

物理内存再大已经跟参数没有关系了

假定 64 bit ORACLE

4） 内存 4G :

shared_pool_size = 200M , data buffer = 2.5G

5） 内存 8G :

shared_pool_size = 300M , data buffer = 5G

6） 内存 12G :

shared_pool_size = 300M--800M , data buffer = 8G

以上仅为参考值，不同系统可能差异比较大，需要根据具体情况调整。关于内存的设置，要再进行细致的调整，起的作用不大，但可根据 statspack 信息和 v$system_event,v$sysstat,v$sesstat,v$latch 等 view 信息来考虑微调。

查询系统空闲空间的情况

select * from v$sgastat where name =’ free memory’

得到 shared pool、large pool、java pool 空闲空间情况。
空闲百分比=(空闲内存/共享池字节数)*100%

```
select v$sgastat.POOL,to_number(v$parameter.value)value,v$sgastat.BYTES,
(v$sgastat.BYTES/v$parameter.VALUE )* 100 "percent free"
from v$parameter, v$sgastat
where v$sgastat.NAME='free memory' and v$parameter.NAME='shared_pool_size'
```

1.5.2 调整数据缓冲区

data buffer,在做了前面的设置后，凡可以提供给 oracle 的内存，都应该给 data buffer = (db_block_size * db_block_buffers) 在 9i 中可以是 db_cache_size 。

```
select name ,value from v$sysstat where name in('db block gets','consistent
gets','physical reads')
```

解释：db block gets 表示从内存读取数据; consistent gets 表示读取一致性; physical reads 从磁盘数据文件读取的数据；physical reads 接近于 0,就是说数据全部来自数据缓冲区。
查询数据字典 v$latch_children ,可以检测数据缓冲区是否有空闲，得到的空闲冲突比例应该是接近 0；

```
select child#,sleeps/gets ratio from v$latch_children where name ='cache buffers
lru chain'
```

使用数据字典 SYSSTAT 计算数据缓冲区的命中率(HIT RATIO),结果在 90%以上，否则必须增加数据缓冲区大小。

```
Select a.value + b.value "logical_reads", c.value "phys_reads",
round(100 * ((a.value+b.value)-c.value) / (a.value+b.value)) "BUFFER HIT RATIO"
from v$sysstat a, v$sysstat b, v$sysstat c
where a.statistic# = 38 and b.statistic# = 39
and c.statistic# = 40
```

下面记录数据库启动进始的逻辑读和物理读。统计值会增加的。

```
select
sum(decode(name,'consistent gets',value,0)) consistent,
sum(decode(name,'db block  gets',value,0)) dbblockgets,
sum(decode(name,'physical reads ',value,0)) physical,
round(((sum(decode(name,'consistent gets',value,0))+
sum(decode(name,'db block get',value,0))-
sum(decode(name,'physical reads',value,0))/
sum(decode(name,'db block gets',value,0))))*100,2) hitratio from v$sysstat
```

如果数据库含有 LOB 类型数据，在计算数据缓冲区命中率时，可以使用下面的计算方法。

```
Select name ,value from v$sysstat where name in ( 'session logical
```

reads’,’physical reads’,’physical reads direct ’,’physical reads direct(lob)’)

显示此会话全部读取数据量（session logical reads）;数据文件读取数据量（physical reads）;
缓冲区读取数据量（physical reads direct）;LOB 类型在缓冲区读取数据量
Select name,value from v$sysstat where name in (’session logical reads’,’physical reads’,’physical reads direct’,’physical reads direct(lob)’)
命中率公式：
HIT RATIO =1-((physical reads- physical reads direct - physical reads direct(lob))/ session logical reads)

数据库新增了数据字元 v$db_cache_advice，用来计算 DB_CACHE_SIZE 增加时物理读的情况。范围从当前大小的 10%到 200%）。
select size_for_estimate ,buffers_for_estimate, estd_physical_read_factor,estd_physical_reads from v$db_cache_advice

where name =’DEFAULT’ AND block_size=(select value from v$parameter where name like ’db_block_size’) and advice_status=’ON’
查询数据字典 v$buffer_pool 确认数据库缓冲区设置状况。

select name ,sum(buffers) from v$buffer_pool group by name,block_size having sum(buffers)>0

在 ORACLE 9I 确定缓冲区对于哪一个缓冲区设置的。

select name ,block_size,sum(buffers) from v$buffer_pool group by name,block_size having sum(buffers)>0

1.5.3 buffer_pool_keep 缓冲区

buffer_pool_keep 这个数据缓冲池用于存储执行全表扫描的小表，buffer_pool_keep 从 db_block_buffers 中分配，因此也要小于 db_block_buffers。设置好这些参数后，就可以把常用对象永久钉在内存里。适用于 oracle 8i。
建设值暂时再议
如何将小表放入 keep 池中,alter table tablename storage(buffer_pool keep);

1.5.4 buffer_pool_recycle 缓冲区

buffer_pool_recycle 这个 池用来保存进行全表扫描的非常大的表，适用于 oracle 8i。
建设值暂时再议

1.5.5 调整共享缓冲区缓存

不能设置 shared_pool_size 过大，通常应该控制在 200M--300M,如果是 ORACLE ERP 一类的使用了很多存储过程函数、包或者很大的系统，可以考虑增大 shared_pool_size ，但是如果超过 500M 可能是危险的，达到 1G 可能会造成 CPU 的严重负担，系统甚至瘫痪。所以 shared_pool_size 如果超过 300M 还命中率不高，那么应该从应用上找原因而不是一味的增加内存，shared_pool_size 过大主要增加了管理负担和 latch 的开销

共享池基本上包括两个主要的结构： 数据字典缓存、 库缓存、共享的 S Q L 区域。

当一条输入/输出请求发出后，高速缓存查看该请求要求的数据是否已在内存中。如果数据在内存的话，它亲自回答该请求，返回请求的数据。这称为一个命中（ hit）。如果数据不在内存的话，就会有一次到磁盘的访问。这被称为一个遗漏（ miss）。

对于几乎所有的高速缓存装置来说，保证高效性能的准则是 9 0 % +命中率，命中率可以被定义为 100×（1.00- (遗漏的数量/请求的数量)），其中请求的数量=遗漏的数量+命中的数量。

例如，假如你的高速缓存有 4 个遗漏和 4 6 个命中，那么你的命中率是 100×(1.00－(4 / 50)) = 100×(1.00－0.08) = 92 %，这个数字非常好。高速缓存通常由一个最近最少使用的(LRU)算法来管理，该算法保证在任何给定的时刻，最近使用最多的(MRU)数据被保存在高速缓存中并且最近最少使用的数据被拿走。

当 Oracle 对一条 SQL 语句进行语法分析时，通过把一个数学公式应用到 SQL 语句的字母数字文本并使用该结果在高速缓存中存储（供以后查找和利用）它， Oracle 便可以在库缓存（library cache）中为该 SQL 语句分配一个 SQL 区。换句话说，它使用一个哈西函数(hash funtio n)。为了一条语句可以被重复使用，该语句必须是相同的。

说明：以字节为单位，指定共享池的大小。共享池包含如：共享游标，存储的过程，控制结构和并行执行消息缓冲区等对象。较大的值能改善多用户系统的性能。 值范围:300 KB - 根据操作系统而定。默认值：如果是 64 位操作系统，值为 64MB；其他情况下，值为 16MB。

- **监控共享缓存区的命中率**

```
Select sum(pinhits-reloads)/sum(pins) "hit radio",sum(reloads)/sum(pins) "reload
percent"
from v$librarycache;
```

以上 SQL 主要参数：`hit radio <90%`

```
select sum(pins) "Total Pins", sum(reloads) "Total Reloads",
sum(reloads)/sum(pins) *100 libcache
from v$librarycache;
```

以上 SQL 主要参数：`libcache <1%`

- **监控字典缓冲区的命中率**

```
select parameter, gets,Getmisses , getmisses/(gets+getmisses)*100 "miss ratio",
(1-(sum(getmisses)/ (sum(gets)+sum(getmisses))))*100 "Hit ratio"
from v$rowcache
where gets+getmisses <>0
group by parameter, gets, getmisses;
```

- **监控字典缓冲区**

```
SELECT (SUM(PINS - RELOADS)) / SUM(PINS) "LIB CACHE" FROM V$LIBRARYCACHE;
SELECT (SUM(GETS - GETMISSES - USAGE - FIXED)) / SUM(GETS) "ROW CACHE" FROM
V$ROWCACHE;
SELECT SUM(PINS) "EXECUTIONS", SUM(RELOADS) "CACHE MISSES WHILE EXECUTING" FROM
V$LIBRARYCACHE;
```

后者除以前者,此比率小于 1%,接近 0%为好。

```
SELECT SUM(GETS) "DICTIONARY GETS",SUM(GETMISSES) "DICTIONARY CACHE GET MISSES"
FROM V$ROWCACHE
```

- **监控 SQL 缓冲区**

```
select osuser,username,sql_text from v$session a,v$sqltext b where
a.SQL_ADDRESS=b.ADDRESS order by address,piece
```

找出最慢的 SQL

```
SELECT EXECUTIONS , DISK_READS,
BUFFER_GETS,ROUND((BUFFER_GETS-DISK_READS)/BUFFER_GETS,2) Hit_radio,
ROUND(DISK_READS/EXECUTIONS,2) Reads_per_run,SQL_TEXT FROM V$SQLAREA WHERE
EXECUTIONS>0AND BUFFER_GETS > 0 AND (BUFFER_GETS-DISK_READS)/BUFFER_GETS < 0.8
ORDER BY 4 DESC;
```

确定缓存的 SQL 语句数量，和它们使用了多少内存和每个 SQL 语句平均消耗的内存。

```
select bytes,sql_count ,bytes/sql_count from (select count(*) sql_count from
v$sqlarea),v$sgastat where name='sql area'
```

1.5.6 库缓冲区

```
select sum(pins)"Total pins",sum(reloads) "Totla reloads",sum(reloads)/sum(pins)*
100 libcache
from v$librarycache
```

参数 Total pins 表示驻留内存次数，Total reloads 表示重新加载到内存的次数，libcache 表示失败率。库失败必需小于 1%,如大于 1%则需要增加参数 SHARED_POOL_SIZE 的值。

```
select (sum(pins-reloads))/sum(pins)"lib cache" from v$librarycache
```

计算库的成功率，必须>99%

1.5.7 调整日志缓冲区

监控 SGA 中重做日志缓存区的命中率，应该小于 1%

```
SELECT name, gets, misses, immediate_gets, immediate_misses,
Decode(gets,0,0,misses/gets*100) ratio1,
Decode(immediate_gets+immediate_misses,0,0,
immediate_misses/(immediate_gets+immediate_misses)*100) ratio2
FROM v$latch WHERE name IN ('redo allocation', 'redo copy');
```

参数名：log_buffer:
说明：以字节为单位，指定在 LGWR 将重做日志条目写入重做日志文件之前，用于缓存这些条目的内存量。重做条目保留对数据库块所作更改的一份记录。如果该值大于 65536，就能减少重做日志文件 I/O，特别是在有长时间事务处理或大量事务处理的系统上。
值范围：根据操作系统而定。
默认值：最大值为 500K 或 128K * CPU_COUNT，两者之中取较大者
log_buffer : 128K - 1M 之间
select * from v$parameter where name like 'log_buffer%'
参数 log_buffer 定义了供即刻写入 redo 日志信息的保留 RAM 的数量，这个参数受 cpu_count 的影响。Oracle 推荐 log_buffer 最大为 cpu_count 乘以 500KB 或 128KB。CPU 的数量对于 log_buffer 来说非常重要，因为 Oracle 会生成多日志写入（LGWR）进程来异步释放 redo 信息。

log_buffer 是 Oracle 中最易误解的的 RAM 参数之一，通常存在下面几个配置错误：log_buffer 被设置得太高（例如，大于 1MB），这回引起性能问题，因为大容量的结果会使得写入同步进行（例如，日志同步等待事件非常高）。
log_buffer 不是 db_block_size 的倍数。在的 Oracle9i 中，log_buffer 应该是 2048 字节的倍数。

1.5.8 调整排序区

查看排序情况

```
select name,value from v$sysstat where name like 'sort%'
```

Sort_area_size: 外部磁盘排序操作（包括临时段的分配）可以使用的最大内存数。默认的用来排序的 SORT_AREA_SIZE 大小是 512K，通常显得有点小，。增大该值可以提高大型排序的效率。如果超过了该内存量，将使用临时磁盘段。这个参数不能设置过大，因为每个连接都要分配同样的排序内存。
（100 并发用户 x 1M）=100M
建议值：平均事务记录数 x 4K
或设置成 2M
值范围：相当于 6 个数据库块的值 (最小值) 到操作系统确定的值 (最大值)。

SORT_AREA_RETAINED_SIZE：一个内存中的排序可以使用的最大内存数。假如一个排序操作需要比 SORT_AREA_RETAINED_SIZE 还多的内存用于内存中的排序，那么它就试图在 SORT_AREA _SIZE 中执行一个外部磁盘排序，在进程中分配一个临时段。假如排序操作需要更多的内存，那么它把该排序任务分为多个排序操作（ sort run）并为此分配多个临时段。服务器进程每次对一个段排序，返回已排序段的合并结果。这些内存分配不存储在 SGA 共享池中，但当使用 MTS 时例外。相反，它们是 UGA 的一部分。如果你正在使用 MTS，它们是 SGA 共享池的一部分，因为 UGA 被重新定位在那里。
SORT_AREA_RETAINED_SIZE。这些设置仅适用于专用服务器。对于 PQO 来说，每一个并行查询服务器需要 SORT_AREA_SIZE。然而，两组并行服务器可以同时工作。所以，对于 PQO 来说，设置如下值：
SORT_AREA_SIZE× 2 × (并行度)
SORT_AREA_RETAINED_SIZE × (并行度) × (排序数> 2)
对于 PQO 来说，最佳值是 1MB。更高的值也不会产生较好的性能。一般说来，除了 MTS 以外，设置 SORT_AREA_SIZE= SORT_AREA _RETAINED_SIZE，对于 MTS 需要一些特殊的考虑。提示对于 MTS 来说，把 SORT_AREA _RETAINED_SIZE 设置为小于 SORT_AREA _SIZE 的数。作为一条准则，你可以如下设置：当一个排序不能完全在内存中进行时，必须创建临时（排序）段。也就是说，当排序操作对内存的需要超过了 SORT_AREA _RETAINED_SIZE 的设置值时，Oracle 就需要分配一个临时段并设法在 SORT_AREA_SIZE 中工作。

1.5.9 调整大池缓冲区

说明 : 指定大型池的分配堆的大小, 它可被共享服务器用作会话内存, 用作并行执行的消息缓冲区以及用作 RMAN 备份和恢复的磁盘 I/O 缓冲区。
值范围: 600K (最小值); >= 20000M (最大值是根据操作系统而定的)。
默认值 : 0, 除非配置了并行执行或 DBWR_IO_SLAVES

large_pool_size :如果不设置 MTS，通常在 RMAN 、OPQ 会使用到，但是在 10M - 50M 应该差不多了。假如设置 MTS,则由于 UGA 放到 large_pool_size 的缘故，这个时候依据 session 最大数量和 sort_ares_size 等参数设置，必须增大 large_pool_size 的设置，。这里要提醒一点，不是必须使用 MTS，我们都不主张使用 MTS，尤其同时在线用户数小于 500 的情况下。
建议值 session * (sort_area_size + 2M)

```
select * from v$parameter where name like 'large_pool_size%'
```

1.5.10 调整 JAVA 池缓冲区

说明: 以字节为单位, 指定 Java 存储池的大小, 它用于存储 Java 的方法和类定义在共享内存中的表示法,
java_pool_size : 若不使用 java，给 30M 通常就够了

查询参数的 SQL：

```
select * from v$parameter where name like 'java_pool_size%'
```

1.6 调整表空间

1.6.1 避免动态空间管理

Oracle 数据库增长空间是就以区的单位扩展的，区由块组成，区的增长方式有两种，一种是 allocation_type 是 UNIFORM，每次分配区的大小是一致的，另一种 Allocation_type 是 SYSTEM 自动分配。区的大是 initial_extent,next_extent,pct_increase 决定的。由增长管理的效率越高，性能也就越好。表空间选择自动增长方式，设置增长率，适用于大数据量的数据插入。

预分配表不仅在空间增长方面上，而且在与这个表空间相关的性能方面都会提供好的粒状控制。如果有一个表在你只分配 10 MB 的时候只是偶尔扩展，那么为什么要分配 100MB 呢？通常，预分配的存储单元越出色，性能就越好。

如果数据库操作引起数据增加并超出了分配的表空间，ORACLE 会自动扩展表空间，动态扩展会降低系统性能。

确定一段比较长的时期内数据的最大大小；选择存储参数值，使 ORACLE 分配足够大的分区，避免频繁自动扩展；

1.6.2 查看回退段

```
select rn.name,rs.gets,rs.waits,(rs.waits/rs.gets)*100 ratio from v$rollstat rs, v$rollname rn where rs.usn=rn.usn
```

1.6.3 查看数据库中回退段信息

```
SELECT rn.name,rs.GETS,rs.WAITS,(rs.WAITS/rs.GETS)*100 ratio from v$rollstat rs,v$rollname rn
where rs.USN =rn.usn

select name,value
from
v$pgastat
order by
value desc

sort_area_retained_size
```

说明：以字节为单位，指定在一个排序运行完毕后保留的用户全局区 (UGA) 内存量的最大值。最后一行从排序空间中被提取后，

该内存将被释放回 UGA，而不是释放给操作系统。

值范围：从相当于两个数据库块的值到 SORT_AREA_SIZE 的值。

默认值：SORT_AREA_SIZE 的值

pga_aggregate_target:

说明：指定连接到例程的所有服务器进程的目标 PGA 总内存。请在启用自动设置工作区之前将此参数设置为一个正数。这部分内存不驻留在

SGA 中。数据库将此参数值用作它所使用的目标 PGA 内存量。设置此参数时，要将 SGA 从可用于 Oracle

例程的系统内存总量中减去。然后可将剩余内存量分配给 pga_aggregate_target。

值范围：整数加字母 K, M 或 G，以将此限值指定为千字节，兆字节或千兆字节。最小值为 10M，最大值为 4000G

默认值："未指定"，表示完全禁用对工作区的自动优化。

1) PGA_AGGREGATE_TARGET-此参数用来指定所有 session 总计可以使用最大 PGA 内存。这个参数可以被动态的更改，取值范围从 10M —— （4096G-1 ）bytes.
2) WORKAREA_SIZE_POLICY-此参数用于开关 PGA 内存自动管理功能，该参数有两个选项：AUTO 和 MANUAL，当设置为 AUTO 时，数据库使用 Oracle9i 提供的自动 PGA 管理功能，当设置为 MANUAL 时，则仍然使用 Oracle9i 前手工管理的方式。

缺省的，Oracle9i 中 WORKAREA_SIZE_POLICY 被设置为 AUTO.

Hash_area_size:

说明：与并行执行操作和 DML 或 DDL 语句相关。它以字节为单位，指定要用于散列联接的最大内存量。有关详细信息，

请参阅手册 Oracle8i Concepts。

值范围：0 到根据操作系统而定的值。

默认值：派生:2 * SORT_AREA_SIZE 参数值

1.6.4 在缓冲区驻留对象（BUFFER_POOL_KEEP）

参数类型：字符串
语法：BUFFER_POOL_KEEP = {integer |
(BUFFERS: integer [, LRU_LATCHES: integer]) }
这里 *integer*是缓冲区数，和LRU锁存器的数
参数类：静态
默认值:无
BUFFER_POOL_KEEP 可以使你在DB_BLOCK_BUFFERS 下作为保留缓冲池来驻留对象。你也可以用分配一个LRU的一部分（用DB_BLOCK_LRU_LATCHES ）
可以指定5种格式，比如简单的：
BUFFER_POOL_KEEP = 5

或指定缓冲区的组合项和LRU锁存器，如：

```
BUFFER_POOL_KEEP = (BUFFERS: 400 [, LRU_LATCHES:3] )
select
sum(decode(name,' consistent gets' ,value,0)) consistent,
sum(decode(name,' db block gets' ,value,0)) dbblockgets,
sum(decode(name,' physical read' ,value,0)) physical,
round(((sum(decode(name,' consistent gets' ,value,0)) +
       sum(decode(name,' db block gets' ,value,0)) -
       sum(decode(name,' physical reads' ,value,0))/
    sum(decode(name,' consistent gets' ,value,0)) +
    sum(decode(name,' db block gets' ,value,0 ))))
    *100,2 ) hitratio
from v$sysstat;
```

批过程的命中率：

```
    col hitratio for 999.99
    select  username,  consistent_gets , block_gets, physicaL_reads,
    100*(consistent_gets + block_gets  -  physical_reads) /
       ( consistent_gets + block_gets) hiratio
    from  v$session, V$sess_io
    where  v$session.sid = v$sess_io.sid
    and ( consistent_gets + block_gets ) > 0
    and username is not null ;
```

.HASH_MULTIBLOCK_IO_COUNT

- 此参数指定一个散列连接一次可以读取和写入的块数；
- 增加此参数可以减少散列表单元的数量；
- 默认值为 4;

5.SORT_MULTIBLOCK_READ_COUNT

- 推荐使用默认值；
- 此参数指定一个排序每次读取临时段时可以读取的数据库块数目；
- 当数据在内存不能进行排序时就用临时段；
- 执行过多的 I/O，内存较空闲，则加大此参数的值。

6. DISK_ASYNCH_IO / TAPE_ASYNCH_IO

- 此参数启用或禁止操作系统的异步 I/O 功能；
- 允许服务器进程在执行表扫描时可进行重叠 I/O 请求；
- 推荐值为 TURE.

```
select name,value from v$sysstat where name in ('table scans (short tables)','table
```

scans (long tables)');

1.7 碎片调优

调整的一个主要目标是避免不必要的碎片。
碎片对系统的影响
（1）导致系统性能减弱
如上所述，当要满足一个空间要求时，数据库将首先查找当前最大的自由范围，而"最大"自由范围逐渐变小，要找到一个足够大的自由范围已变得越来越困难，从而导致表空间中的速度障碍，使数据库的空间分配愈发远离理想状态；
（2）浪费大量的表空间
尽管有一部分自由范围（如表空间的pctincrease为非0）将会被SMON（系统监控）后台进程周期性地合并，但始终有一部分自由范围无法得以自动合并，浪费了大量的表空间。

1.7.1 自由范围的碎片计算

由于自由空间碎片是由几部分组成，如范围数量、最大范围尺寸等，我们可用FSFI--Free Space Fragmentation Index（自由空间碎片索引）值来直观体现：
FSFI=100*SQRT(max(extent)/sum(extents))*1/SQRT(SQRT(count(extents)))
可以看出，FSFI的最大可能值为100（一个理想的单文件表空间）。随着范围的增加，FSFI值缓慢下降，而随着最大范围尺寸的减少，FSFI值会迅速下降。
计算FSFI值：

```
select tablespace_name,sqrt(max(blocks)/sum(blocks))*

       (100/sqrt(sqrt(count(blocks)))) FSFI

from dba_free_space

group by tablespace_name order by 1;
```

统计出了数据库的FSFI值，就可以把它作为一个可比参数。在一个有着足够有效自由空间，且FSFI值超过30的表空间中，很少会遇见有效自由空间的问题。当一个空间将要接近可比参数时，就需要做碎片整理了。

1.7.2 段的碎片整理

如果段的碎片过多，将其数据压缩到一个范围的最简单方法便是用正确的存储参数将这个段重建，然后将旧表中的数据插入到新表，同时删除旧表。这个过程可以用Import/Export（输入/输出）工具来完成。
Export（）命令有一个（压缩）标志，这个标志在读表时会引发Export确定该表所分配的物理空间量，它会向输出转储文件写入一个新的初始化存储参数--等于全部所分配空间。若这个表关闭， 则使用Import（）工具重新生成。这样，它的数据会放入一个新的、较大的初始段中。例如：

```
exp user/password file=exp.dmp compress=Y grants=Y indexes=Y
        tables=(table1,table2);
```

若输出成功，则从库中删除已输出的表，然后从输出转储文件中输入表：

```
imp user/password file=exp.dmp commit=Y buffer=64000 full=Y
```

这种方法可用于整个数据库。

1.8 查找使用 CPU 多的用户

```
select                                  a.sid,spid,status,substr(a.program,1,40)
prog,a.terminal,osuser,value/60/100 value
from v$session a,v$process b,v$sesstat c
where c.statistic#=12 and c.sid=a.sid and a.paddr=b.addr order by value desc;
```

1.9 查看用户的内存使用情况

```
select a.sid,b.name,a.value from v$sesstat a,v$statname b
where (b.name like '%uga%' or b.name like '%pga%') and a.statistic# = b.statistic#
order by value desc
```

1.10 监控表空间的 I/O 比例

```
select df.tablespace_name name,df.file_name "file",f.phyrds pyr,
f.phyblkrd pbr,f.phywrts pyw, f.phyblkwrt pbw
from v$filestat f, dba_data_files df
where f.file# = df.file_id
order by df.tablespace_name;
```

1.11 监控文件系统的 I/O 比例

```
select substr(a.file#,1,2) "#", substr(a.name,1,30) "Name",
a.status, a.bytes, b.phyrds, b.phywrts
from v$datafile a, v$filestat b
where a.file# = b.file#;
```

1.11.1 检测 Redo 日志缓冲区锁存：

```
select name,value from v$sysstat
where name = 'rado log space requests' ;
```

value 值应接近 0 若较大则应加大 INIT.ORA 中的 LOG_BUFFER 项的值。