

附附附录

A 介绍在Solaris系统上的Oracle

Oracle存在于许多平台上，特别是UNIX（更能表现它的品质）、VMS、NT与NOVELL上。Oracle也移入关系型大型机市场，在关系型大型机市场中通常占优势的是DB2/MVS。然而，在UNIX上Oracle的存在通常是，而且仍然是最强的。因此，这里是特定于UNIX的附录。

A.1 Solaris

当前，Solaris是Oracle端口的第一位UNIX操作平台。Solaris，现代SUN操作系统，是SUNOS的后继者。从Solaris2.0开始，Solaris一直是主流的系统V版本4（SVR4）——基于并遵循许多在90年代初激增的UNIX标准。在写这个版本的同时，Solaris已升级到版本2.6。另一方面，SUNOS，一个学术界和科学界长时期喜爱的产品，是基于伯克利系统分布（BSD）的。除了一些特殊的专用高性能计算（HPC）、大量并行处理（MPP）机制、BSD自己以及一些共享操作系统如Linux外，大多数现代UNIX操作系统极大地基于SVR4。这意味着，不论是否相信，UNIX在这些年更加标准化。这是一个好消息，因为UNIX很长时间沿着主要的系统V与BSD前端分离，进一步分离了生产厂商阵营，并且还进一步被真正的壳（命令行解释程序）分离。排除一些极少数的例外，Oracle DBA今天经常担心的是壳的区别。因为Solaris是目前市场领先的UNIX操作系统，并且是Oracle的第一位端口，这个附录主要使用Solaris示例解释常见的UNIX议题。

A.2 Oracle DBA的UNIX入门

在深入讨论UNIX上更高级的Oracle论题之前，回顾一些Oracle DBA在UNIX平台上工作经常使用的UNIX基础很有用。如果你是一个UNIX老手，跳过本节，否则，在继续下去之前先阅读本节。

A.2.1 壳限制与进程限制

有三个主要的命令解释器，或壳，将运行在SA与DBA的设置中：Bourne、Korn与C。这些壳不只是交互命令解释器，而且还用于书写非交互的称为脚本的程序。B壳是第一个UNIX壳，它的反向能力使它成为可移植的壳选择，然而，它缺少一些重要的功能，如工作控制与命令历史；其次是C壳，使用BSD UNIX开发，它提供了一种简化的类似C语言的命令、工作控制及其他功能。最后是Korn壳，它是B壳的超集。因此，Korn可以运行任何Bourne壳或Korn壳脚本。Korn也合并入许多C壳的功能，如工作控制。C壳仍然是交互壳的用户主流选择。虽然系统管理员更经常地趋向使用B壳。然而，具有POSIX标准的Korn壳赢得了一些普及。

在任何情况下，由于Oracle运行了许多UNIX进程，许多Oracle环境配置是特定于壳的。

一个好的例子是进程限制。在C壳中通过使用limit与unlimit命令限制进程。在Korn壳与Bourne壳中，进程通过使用ulimit命令限制。

注意 记住，当作为Oracle用户而不是其他的用户登录时，使用limit与ulimit命令。

例如，在C壳中，也许想将文件描述器的数量设为无限，使用下述语句：

```
% limit -f
% 1024
% limit -f unlimited
% limit -f
% unlimited
```

这在Korn壳与Bourne壳中是相似的，除了要使用一个不同的命令（ulimit）与不同的选择。参见csh、ksh、sh的帮助页以获取更多信息。

A.2.2 软链接与硬链接

在UNIX中的链接是一个文件名，它只含有一个指向真实文件的指针。即，链接是相同文件的替代名。使用一个硬链接，你只看到文件名，并不能区别两个文件是否真的相同，除非察看信息结点。

```
% ls
% file1
% ln file1 file2
% ls -l file2
% -rwxr-xr-x    user1    group1 10000    file2
% ls -f
% 1234 file1 1234 file2
```

另一方面，使用一个软链接，可以看到所做的链接：

```
% ls
% file1 file2
% ln -s file1 file3
% ls -l file3
% lrwxrwxrwx    user1    group1 10000    file3 -> file1
```

通常，由于这个原因，最好使用软链接，因为除非系统有很好的文档，否则很有可能忘记哪些文件是硬链接的。这并不是说你不能再次指出它，而是由于系统大小增加，这样做的维护在价值上超过几乎任何好处。参见ls帮助页获取更多的信息。

A.2.3 命名的管道与压缩

UNIX中的管道是一个程序，它的标准输出连接到文件描述器 #1，stdout，它的标准输入连接到文件描述器 #0，stdin，因而，多个命令可以被“管道化”集合到一起以形成一个管道。在下面的例子中，首先运行ls（列出路径内容）命令，然后通过管道输出到lp（打印客户进程）进行打印：

```
% ls
% file1 file2 file3
% ls | lp
% standard input accepted by printerque1
```

注意将程序组成管道，使用垂直分隔线（|）。UNIX中的命名管道是一个用户定义的文件名，目的用于接受输入到stdin并产生输出到stdout。它是具有名字的转移归向。命名的管道驻留在内存中。对Oracle DBA而言，最常用的例子如下：

```
% mknod pipe1 p
% export file=pipe1 &
% dd if=pipe1 | compress | tar cvf /dev/rmt/0
```

这一系列命令创建管道 pipe1。它开始一个到那个文件的后台输出，然后读管道，进行压缩，并将它传送到磁带。当然，可以直接输出到磁带上（或首先到一个文件上），压缩它然后传送到磁带，但这是唯一一种可以不需要首先将压缩文件存储到硬盘上，就可以得到磁带上的压缩文件的方法。就 Oracle 来说，在以下情况特别有用：必须将非常巨大的表导出到磁带，只有极小的富余空间可以用作数据移动区域。

A.2.4 临时路径

在 UNIX 中，至少有三种临时路径：系统易失的、用户定义易失的与系统非易失的。/tmp 路径是系统易失的路径，tmpfs 路径是用户定义易失的路径，/var/tmp 路径是系统非易失的路径。最常使用的是 /tmp。/tmp 路径安装在 UNIX 的称为交换区的虚拟内存区中。它是一种特殊类型的临时文件系统（tmpfs），称为 swapfs。它的内容在系统重新启动时被清空。SA 需要创建其他的 tmpfs 的非交换区路径，这些是基于内存的文件系统。正如所指出的，它们是易失的，也就是说，在系统关闭或切断电源时它们的内容会丢失。然而，这些特殊的文件系统提供了增长的 I/O，因为向这些路径写入与读出的信息比常用 UNIX 文件系统（UFS）需要的每个文件要小。性能会进一步提高，因为你在任何时刻都向主存或虚拟内存读出或写入。显而易见，主存是最快的，但虚拟内存比通常的硬盘更快，因为它已被缓存入内存。UNIX 中的最后一种临时路径类型是 /var/tmp (Solaris)，有时是 /usr/tmp。这个路径典型地比 /tmp 小，但是是非易失的，即，在系统关闭或切断电源时它们的内容不会丢失。/var/tmp 路径用于许多情况，例如，它是 UNIX 排序程序的缺省排序位置，它存储临时高速缓存编辑（vi）文件。主要的事情是它保留了它的内容，除非程序在它之后清空自己。

Oracle 为不同的目的使用这些路径。例如，SQL*Net 监听器文件可以在 /var/tmp/Oracle 下找到，临时安装文件有时存储在 /tmp 路径下。

A.3 UNIX 上的 SA 与 DBA 配置

DBA 确实应该多知道一些 UNIX 系统管理（SA）知识，以对 Oracle RDBMS 与 UNIX 系统上的实例/数据库进行更有效的管理。例如，我是一个 UNIX SA，SYBASE DBA 与 Oracle DBA。其他的企业时常将 SA 与 DBA 的工作分离，虽然这提供了更多的工作，并且确实使技术人员完全专攻一个位置或另一个位置，但它也最终会成为巨大的隔离的工作环境，特别是如果 SA 与 DBA 为不同的老板工作。在这种类型情况下，清楚、简明并且及时的沟通是最重要的。在任何情况下，我鼓励 DBA 学习尽可能多的特定的不同的 UNIX 操作系统，特别是基本的 SA 任务！这附加的努力会回报给你有能力使用带有 Oracle RDBMS 的操作系统。下面的一些题目在《Operating System Specific Installation and Configuration Guide》或《Server Administrator's Reference Guide》中已经描述过。你也可以使用联机文档。

A.3.1 设置 dba 组与 OPS\$ 注册

在 UNIX 中，为了连接 Internal（在 svrmgrl 中），你必须是在 /etc/group 中指定的 UNIX dba 组的成员。/etc/group 中的一个示例条目如下：

```
dba:*:500:oracle, jdoe, jsmith, bfong, ppage
```

为了添加这个组，你必须是根用户。当是根用户时，要用 DAB特权without a password连接，必须作以下两条指令之一。

```
SVRMGRL> CONNECT INTERNAL;
```

```
SVRMGRL> CONNECT / AS SYSDBA;
```

后面一种是首选的，前面一种是过时的，虽然它们都可以工作。任何 Oracle账户的相关机制是OP\$。这使用户可以使用他在UNIX级别作为Oracle帐户拥有的相同的注册。一条建议是设置init.ora参数OS_AUTHENT_PREFIX=""（空串），这样就可以使用如jdoe的名字，而不是OP\$jdope。记住，在UNIX上，所有的事物都是大小写敏感的，所以在任何地方使用相同的大小写。为创建Oracle用户，做如下工作：

```
SQL> CREATE USER jdoe
2> IDENTIFIED EXTERNALLY;
```

当然，可能需要指定缺省表空间、份额，也许是一个环境资源文件。如果用户早已存在，使用ALTER而不是CREATE命令。然而，有一些SQL*Net通道，缺省情况下不支持OP\$注册。为启用它们，设置init.ora参数REMOTE_OS_AUTHENT=TRUE。还有端口监督程序用户必须存在于/etc/passwd中，并且它必须不是一个OP\$注册。

A.3.2 使用oratab文件与dbstart/dbshut脚本

当完成安装的运行时，一个最重要的安装后处理步骤是运行 root.sh脚本（当然作为根用户）。当完成时，你拥有一个 /var/opt/oracle/oratab（或/etc/oracle/oratab）文件，它是一个Oracle SID项目的清单与一些其他的信息。具有两个（实例）条目的安装如下：

```
SID1:/dir1/dir2/dir3/oracle:Y
SID2:/dir1/dir2/dir3/oracle:N
```

Y指明“是”，当dbstart运行时启动这个实例（SID）。dbstart与dbshut程序是Oracle提供的UNIX壳脚本。必须创建你自己的脚本以包括这些脚本，并在启动时作为Oracle运行dbstart，在系统关闭时运行dbshut。创建一个B壳脚本如下所示：

```
#!/bin/sh
# ora.server
ORACLE_HOME=/dir1/dir2/dir3/oracle
# start
case "$1" in
'start')
su - oracle $ORACLE_HOME/bin/dbstart &
# stop
'stop')
su - oracle $ORACLE_HOME/bin/dbshut &
;;
esac
```

将这个文件保存在/etc/init.d路径下。然后如下所示（软）链接这个ora.server脚本：

```
# ln -s /etc/init.d/ora.server /etc/rc0.d/K99ora.server
# ln -s /etc/init.d/ora.server /etc/rc2.d/S99ora.server
```

在系统启动或重新启动（#init 6）时，Solaris调用所有/etc/rc?.d路径下的S*脚本，包括S99ora.server，并传递每个start作为\$1参数。与此类似，在系统关闭（#init 0）时，所有的K*脚本被调用，包括K99ora.server，并传递每个stop作为\$1参数。

A.3.3 使用(c)oraenv脚本与全局注册

Oracle为B壳与K壳提供了一个oraenv壳脚本，为C壳提供了一个coraenv壳脚本。当放入用户的启动文件或交互地读入当前壳时，他们为这个用户设置适当的 Oracle环境，特别是SID值。将 (c) oraenv文件加入用户的启动文件中。用户的启动文件依赖于所用的壳，如果使用B壳或K壳，在用户的.profile文件（或在Korn中的ENV文件）中加入以下行：

```
ORAENV_ASK=NO
. /opt/bin/oraenv
ORANEV_ASK=
```

对于C壳，在用户的.login（或如果喜欢，使用.cshrc文件）中加入下列行：

```
set ORAENV_ASK=NO
source /opt/bin/coraenv
unset ORAENV_ASK
```

注意(c)oraenv文件的位置在/opt/bin路径下，这对Solaris是固定的。对其他的机器，可能是/usr/local/bin。在这个设置中有一些变化，将这些行加到每个用户的本地注册（并且维护它们）可能是单调乏味的，一种代替方法是将它们加到全局注册文件中。对 B壳与K壳，它是/etc/profile，对C壳，它是/etc/.login。当一个用户注册时，首先读全局注册文件，然后是用户的本地注册。因而，本地注册覆盖全局注册信息。通过将这些行加到全局注册文件中，你只需要最多将它们加到两个文件中，而不是加到所有用户的文件中。另外，保证了一个所有用户的通用Oracle环境，除非用户忽略它。不同的用户是否需要不同的环境是一个方针问题。最后一个变化是注册时的SID，简单的方法是在读(c)oraenv之前还要设置ORAENV_ASK变量，然后每个用户将被提问需要使用什么SID，例如：

```
ORACLE_SID = [ SID1 ] ?
```

然后输入另一个SID以覆盖缺省的SID，在本例中是SID1。在注册之后，按下面的做法每个用户可以重新运行 (c) oraenv以改变到另一个实例：

```
. /opt/bin/oraenv for Bourne or Korn shells, or
source /opt/bin/coraenv for C shell.
```

A.4 配置共享内存与信号量

共享的内存理所当然许多进程或线程共享的。一个Oracle实例的SGA驻留在共享内存中。如果全部SGA大小超过了UNIX OS的配置，实例将无法启动。共享内存是分段的，并可以依照三个模型被分配：一个段、连续的多个段或非连续的多个段。Oracle试图按列出的这些模型的次序分配要求的SGA内存。如果所有三个可能的模型都不能分配Oracle SGA要求的足够的共享内存，Oracle产生一个ORA错误，实例启动失败。控制SGA大小的值是DB_BLOCK_SIZE、DB_BLOCK_BUFFERS、SHARED_POOL_SIZE与LOG_BUFFER。排序参数也影响SGA与非SGA内存，请参阅第21章以获取更多的排序配置信息。

你设置的Solaris（大多数SVR4系统）共享内存参数如下：

SHMMAX	一个单独的段的最大尺寸（字节）。
SHMSEG	一个单独进程的段的最大数量。
SHMMNI	系统范围段的最大数量。

这些之中最关键的是SHMMAX与SHMSEG。按以下的例子去做并设置init.ora与UNIX参

数。系统支持一个实例，机器具有 1GB 的主存，没有其他的真正与 Oracle 竞争的应用，特别是没有使用共享内存的应用。换句话说，有一个“专有的”数据库（硬件）服务器。应该以 1/2 或 3/4 主存启动。要以 3/4 主存或 768M 主存启动，你的 init.ora 参数如下所示：

```
DB_BLOCK_SIZE=16384          # 16KB
DB_BLOCK_BUFFERS=45056        # x 16KB = 704MB
SHARED_POOL_SIZE= 16777216    # 16MB
LOG_BUFFER= 8388608           # 8 MB = size of 1 redo log
```

数据库高速缓冲缓存（704MB），加上共享池（16 MB），加上日志高速缓冲（8MB），一共是 728MB，比 768MB 还少 40MB。现在，将共享内存设为 768MB。Solaris 共享内存参数应该设置如下（在 /etc/system 文件中）：

```
set shmsys:shminfo_shmmax= 805306368
set shmsys:shminfo_shmseg=10
```

只要 Oracle 启动时，没有与其他应用的共享内存的竞争，它确切分配一个比 768MB 小一些的共享内存段——并且使用它的大部分。

另外，需要重新定位共享内存段的启动地址，从 www.sun.com 下载白皮书，其中有在 Solaris 上为 Oracle 实现的细节。在一个段中获得所需要的全部共享内存也是必要的。

信号量是全局的（共享）内存锁定机制，它们是“真正的”锁，因为它们是由一个“门”与一个队列组成的，有时它们称为“排队锁”，以与非排队的锁，如闩或自旋锁相对。Oracle 至少为每个 Oracle 进程使用一个信号量，将 UNIX 信号量的最大值设为 init.ora 中的 PROCESSES 参数大一些，用于如果有多个实例并发运行时，所有实例的连接。假设预期会有 100 个以上的并发用户，将 Oracle 后台进程（这些可以适当变化，依赖于其他的 init.ora 参数与运行的 RDBMS 选项）增加到这个数。假设有 15 个 Oracle 后台进程，需要在 init.ora 文件中进行如下设置以具有一定的容错空间：

```
PROCESSES=120
```

在 Solaris（及大多数 SVR4 系统）上，在 /etc/system 文件中进行如下设置，安全性比 120 高一些：

```
set semsys:seminfo_semmsl=150
```

SEMMSL 设置每个集合的信号量的最大数量。由于共享内存按段分配，信号量按集合分配。也可以设置另两个参数，但它们不像前面那些参数那样关键，如下所示：

SEMMNI 系统范围的集合的最大数量。

SEMMNS 系统范围的信号量的最大数量。

还有一些其他的 Solaris（SVR4）内核参数，也可以在 /etc/system 内核文件中设置，影响共享内存、信号量与其他基于内存的结构。请参阅 Solaris 系统配置指示手册以获取更进一步的信息。

A.5 理解OFA

由 Oracle 版本 7 提供的最佳弹性体系结构（OFA）中，提供了一套安装建议与指南。OFA 是按 UNIX 开发的，OFA 白皮书是免费的，并可以从 www.Oracle.com 下载。记住，这些并不是必需的。另外，建议的命名规则也只是建议，而不是要求。OFA 在这方面也是有灵活性的。

OFA给予的灵活性是拷贝巨大安装、许多实例与许多版本的能力。而且，当具有通过许多时期，或阶段，如开发、集成、测试与产品化的实例时，OFA会有所帮助。

维护分离的环境会是一个挑战，但使用与OFA相似的设置会帮助减轻实现过程中的系统管理任务。OFA提供的不只是路径与文件组织和命名建议，它还处理了表空间命名与分离。在任何情况下，双版本、双实例（产品与开发）配置的一个例子将具有如下的环境变量设置与OFA结构：

```
$ORACLE_BASE=/u01/oracle
```

```
$ORACLE_HOME=$ORACLE_BASE/product/7.3.3 for the production SID
```

```
$ORACLE_HOME=$ORACLE_BASE/Product/8.0.3 for the development SID
```

```
$TNS_ADMIN=$ORACLE_HOME/network/admin; 注意像$ORACLE_HOME一样，这一次只能属于一个实例。
```

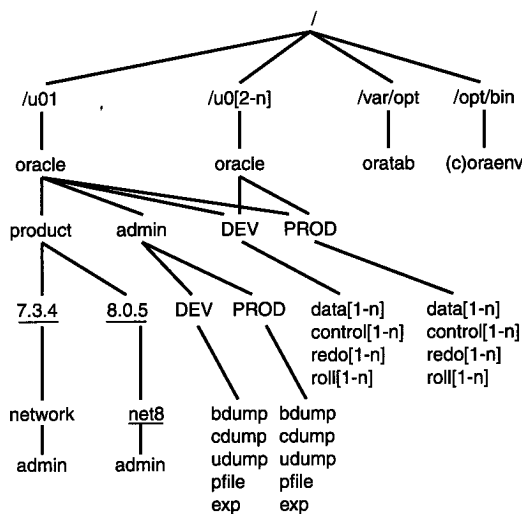
```
oratab位于/var/opt 或/etc下，
```

```
(c) oraenv文件位于/opt/bin or/usr/local/bin下，
```

```
数据文件，控制文件，重做登录文件和回滚数据文件位于/u0
```

```
[1-n] /oracle/<sid>子目录下，这里 n代数表根 /u0目录号，软链接需要时也可使用。管理文件存储在 $ORACLE_BASE/ADMIN/<SID>，例如bdump、cdump、udump、bfile和exp子目录下。
```

请参阅OFA白皮书以获取有关这方面指导的细节，虽然OFA是由UNIX上的Oracle而来，但它可以更新用于NT、NETWARE、VMS与其他平台。图A-1显示了OFA结构的一个示例图表，使用DEV作为开发SID与产品PROD的名字。



图A-1 一个双版本、双实例配置的OFA路径结构

A.6 裸盘与UFS比较

裸盘与UFS的比较是一个传统的在Oracle与其他RDBMS DBA之间的争论，至今还没有结论。换句话说，没有任何所有应用都适用的声明。UNIX上的裸盘分区是一个没有创建UNIX文件系统（UFS）的硬盘分区，因此，在该部分上没有文件系统（或常用的UNIX I/O）高速缓存。事实上，抛开它在UNIX级别上的硬盘文件名，UNIX事实上对裸盘不做任何事情。为什么使用裸盘分区呢？如果接近自由的I/O是你的应用类型的典型使用模型（例如，OLTP），裸盘分区可以帮助绕过UNIX高速缓存。如果应用类型包括巨大的连续的I/O（例如DSS与DW），高速缓存帮助最小化机电的缺点，换句话说，如果当I/O是瓶颈时，UNIX（读然后向前）高

速缓存胜过裸盘分区存取。然而，还有其他的因素，例如思考时间与计算时间。思考时间是用户在一个交互系统如 OLTP 系统中交互之间需要的时间，计算时间是思考时间的批量类推，它是程序在显著的函数任务中用于计算的时间。在任何一种情况下，如果思考时间或计算时间在巨大 I/O 下很高，UNIX 高速缓存的好处会被忽略，因此，裸盘分区会产生更好的结果。下面是一些使用裸盘分区的主要优点与缺点：

优点：

降低 I/O 开销（没有 UNIX 高速缓存/文件系统缓存）。

提高随机 I/O 或大量的思考/计算时间。

写保证（Oracle->硬盘代替 Oracle->UNIX->硬盘）。

缺点：

降低巨大序列 I/O 的性能。

如果使用不当，降低性能（比文件系统低）。

难于管理。

简要看一下最后一条缺点，因为使用裸盘还有一些内在的管理问题，主要包括以下三条：

创建或改变一个分区需要一个完全的硬盘卸载/重新装载。例如，假设已设置了裸盘分区与数据库，然后，发现需要增加一个裸盘分区的大小。如果没有其他的可用方法，必须增加一个分区，你将不得不卸载整个硬盘内容、重新进行分区，然后重新装载。

只有很少的 I/O 工具（只有 dd）是专门用于备份的工具。dd 工具一次只能备份一个分区，并且不能跨越磁带（它不是一个多卷工具）。而且，dd 不能感觉到磁带已到尾，所以必须确保裸盘分区大小正好。你不希望在恢复过程中发现裸盘不够大。

数据文件的分配具有较小的灵活性，可能会潜在地浪费空间，特别是随时间进行累积。假设由于性能的原因需要移动数据文件（即，平衡 I/O）。这在使用文件系统的文件时并不复杂，然而，移动两个裸盘分区，除非它们大小相等，否则会需要数据库文件所在的硬盘的完全卸载与重新装载。

最后要注意，如果选择使用裸盘，在这样做之前，在所有可能使用的硬盘上进行完全硬盘检测（坏扇区检测）。确保有足够的硬盘空间——还要另加一些。使所有裸盘分区大小相等，或选择三个或四个固定的分区大小，如 256MB、512MB、768MB 与 1024MB（1GB），这将增加裸盘管理的灵活性。后一种使用一些大小类的解决方案对大多数安装更适合。例如，它增加了具有适当大小分区，并以一些内部碎片（浪费空间）为代价的可能性。即使这样，这在使用裸盘时也几乎是不可避免的。

警告 当格式化裸盘时，总是跳过柱面 0 以为硬盘标签留下足够的空间。否则，Oracle DBMS 将有可能在最坏的可能时间重写硬盘标签。通过比较，高级 UFS I/O 存取例程内嵌地会跳过它。

A.7 补充 UNIX 性能调整技巧

A.7.1 使用异步输入/输出

通常，一个 DBWR/LGWR 进程必须写、等待、写、等待、如此往复。虽然含有在高速缓

存交换，这也不是实时的。这就是同步（串序）I/O。异步（并行）I/O（AIO）减少了在高速缓存读/写之间的等待时间。AIO在UFS或裸盘上同时作用。如果支持，使用KAIO，它是AIO驻留在内核中，而不是在内核之上的附加提高。因而，它也更快一些。确信已设置以下init.ora参数，缺省情况下是使能的：

```
ASYNC_WRITE=TRUE
ASYNC_READ=TRUE
or ASYNC_IO=TRUE in place of the previous two for some platforms.
```

A.7.2 使用多个数据库写入（DBWR）进程

如果没有使用AIO，设置以下init.ora参数：

DB_WRITERS=<含有数据文件的独立硬盘的数量>

在必要时将这个值增加为含有数据文件的独立硬盘数量的两倍，这种并行I/O不需要使用异步I/O。因为缺省情况下每个DBWR的I/O是同步的。首先使用KAIO，第二使用AIO，最后使用多个DBWR，但不要一起使用。

A.7.3 使用readv()系统调用

对大量的串行I/O，readv()减少了高速缓存——高速缓存的传递时间。readv()系统调用缺省情况下是不使用的，通过设置以下参数启用它：

```
USE_READV=TRUE
```

A.7.4 首先使用外部硬盘柱面

因为SUN使用零位记录（ZBR）技术，外部的柱面趋向优于内部的柱面。外部的柱面为0、1、3、与4、5、6、7相对。

警告 记住，不要使用柱面2，重叠的柱面，因为它保留有硬盘的总量大小。当这个柱面改变时，将阻碍一些程序。

A.7.5 使用Oracle postwait驱动器

如果你的平台可用，Oracle postwait驱动器提供了一种信号量的替代机制，比信号量更快，并为Oracle提供了与信号量相同的功能。

A.7.6 使用ipcs与tstshm监控共享内存与信号量

为监控共享内存，使用UNIX内部进程通信状态（ipcs）命令的-mb选项，或使用Oracle共享内存工具（tstshm）。要监控信号量，使用ipcs -sb。这些工具帮助决定使用了多少共享内存与多少信号量。它们也可以帮助决定共享内存的碎片。参见ipcs的帮助页以获取更详细的细节，在此是一个每种工具的示例运行。

```
# ipcs -mb
IPC status from <running system> as of Tue Nov 18 11:59:34 1997
T  ID   KEY      MODE    OWNER    GROUP    SEGSZShared Memory:
m   500   0x0898072d  --rw-r-----  oracle   dba      41385984
m   301   0x0e19813f  --rw-r-----  oracle   dba      38871040
m   200   0x0e3f81dc  --rw-r-----  oracle   dba      45301760
```

```

hostname:oracle> tstshm
Number of segments gotten by shmget() = 50
Number of segments attached by shmat() = 10
Segments attach at lower addresses
Maximum size segments are not attached contiguously!
Segment separation = 4292345856 bytes
Default shared memory address = 0xfeed80000
Lowest shared memory address = 0xffff00000
Highest shared memory address = 0xfeed80000
Total shared memory range = 4010278912
Total shared memory attached = 20971520
Largest single segment size = 2097152
Segment boundaries (SHMLBA) = 8192 (0x2000)

# ipcs -sb
IPC status from <running system> as of Tue Nov 18 11:59:39 1997
  T   ID   KEY             MODE             OWNER      GROUP      NSEMS
Semaphores:
s    327680 00000000  --ra-r-----  oracle    dba        25
s    327681 00000000  --ra-r-----  oracle    dba        25
s    196610 00000000  --ra-r-----  oracle    dba        20
s    400000 00000000  --ra-r-----  oracle    dba        25
s    500000 00000000  --ra-r-----  oracle    dba        25

```

A.7.7 使用直接输入/输出

一些UNIX系统支持使用UFS的直接输入/输出，有效地越过了UNIX高速缓存。它忽略了使用裸盘作为一个选择的需要。如果平台支持它，启用它，使用它代替裸盘分区。

A.7.8 不要使用进程捆绑或改变计划表类

因为SMP机器占有大多数Oracle数据库服务器的市场份额，所以做这个建议。书写在SMP机器上运行的程序是最有效的，它们的进程是与处理器无关的，S在SMP中代表均衡性。存在可用的工具（tcpctl、pbind、nice/renice与prioctl），可以重新形成进程的相似性与进程的优先级。通常，在SMP机器上，将这些参数与Oracle系统（后台）进程混合是一个坏主意。

A.7.9 不要修改内存高速缓存

不要修改常用的UNIX内存高速缓存或文件系统高速缓存，除非你非常清楚你正在做什么。通常，修改这些UNIX内核与UFS参数对Oracle性能的影响很小。例如，内核参数bufhwm是常用UNIX I/O高速缓存可以使用的最大KB数。缺省情况下，UNIX高速缓存可增长到可用内存的2%。程序tunefs控制UFS逻辑块（或高速缓存），缺省是8KB。在改变这些参数时要小心，这么做一定要有一个好的原因，例如，如果你有大量的串行 I/O并正使用UFS，可以增长/etc/system中的bufhwm。

A.7.10 保证UFS文件系统充满程度不超过90%

除非进行修改，否则每个UFS的minfree不得低于10%。一个UFS至少要低于90%的容量，才是速度最优的。大于或等于90%，UFS优化机制会切换到空间优化，因此，尝试保持UFS存储Oracle数据文件与其他Oracle组件为89%或更低。