

# 第三章

# Oracle 性能 最大化

## 本章内容：

- 配置和优化有什么不同
- 获得最大的性能
- 配置操作系统
- 配置 Oracle
- 调整和配置数据库对象
- 优化 Oracle

如果你问很多 Oracle DBA “你工作中最大的一部分是什么？”几乎所有的回答都是“数据库的配置和优化。”Oracle 是一种真正复杂和强大的产品，而且它的强大的能力在于它对每个单独的数据库配置都可以以最好的性能运行。本章讲述我们配置和优化 Oracle 数据库的方法，并提供了为站点实现一个高性能数据库的指导方针。

大多数 Oracle DBA 连续的、每天的职业是使 Oracle 数据库获得可能的最好性能。对于“性能”可能有许多定义，但是我们把性能定义为目标和在怀疑有问题的数据库中执行一个典型操作需要的可以测量的时间。是的，这是一个太简单的定义，它忽视了其他的测量尺度，如资源使用。但是让我们正视它：我们期望数据库尽可能地快，因此为了这个目的这是一个合理的定义。

整本书都是以 Oracle 性能为主题写的（参见附录“DBA 使用的资源”以查看我们认为你应该注意的内容，注 1），所以我们不能在一章中就阐述完复杂的 Oracle 性能优化，我们希望提供一种直截了当的性能优化方法并提供能应用到各个不同安装上的实际指南。

从物理和逻辑的实现、处理的事务类型及这些事务的性能需求方面来看，每个 Oracle 安装都是不同的，认识到这点很重要。结果是虽然一些厂商（包括 Oracle）

---

注 1：我们尤其推荐 Mark Gurry 和 Peter Corrigan 的《Oracle Performance Tuning》第二版（O'Reilly&Associates,1997）。

尝试提供，但仍没有一种自动的优化方法，而且也没有单一的一套规则可以提供一种使数据库性能最优的方法。然而，我们可以提供一种方法，在适当应用并结合 DBA 知识和经验的情况下，该方法将使任何给定的数据库有好的性能。

## 配置和优化有什么不同

使一个 Oracle 数据库获得最佳性能需要认真注意数据库的配置和优化两方面。这些术语经常交换使用，但是事实上，它们是两个不同的任务，无可否认的是在它们之间有一少部分内容是重叠的。

配置是设置数据库的物理和逻辑组件的过程，也是配置主机系统的过程，而优化是修改数据库的内部行为的过程，以便操作以特定方式运行。整个过程某种程度上是循环的，因为合适的优化经常包含修改配置，然后再一次查看优化结果。图 3-1 演示了配置和优化过程的基本步骤。

## 可以配置什么

能在一个 Oracle 数据库中配置的一些项目如下：

影响系统进程分配的数据库的组件，例如：

SQL\*Net

MTS ( Multi-Threaded Server , 多线程服务器 )

并行查询 ( Parallel Query )

并行服务器 ( Parallel Server )

物理存储的布局 and 大小

数据库对象的大小，如：

表

索引

回滚段

排序区

临时表空间

重做日志

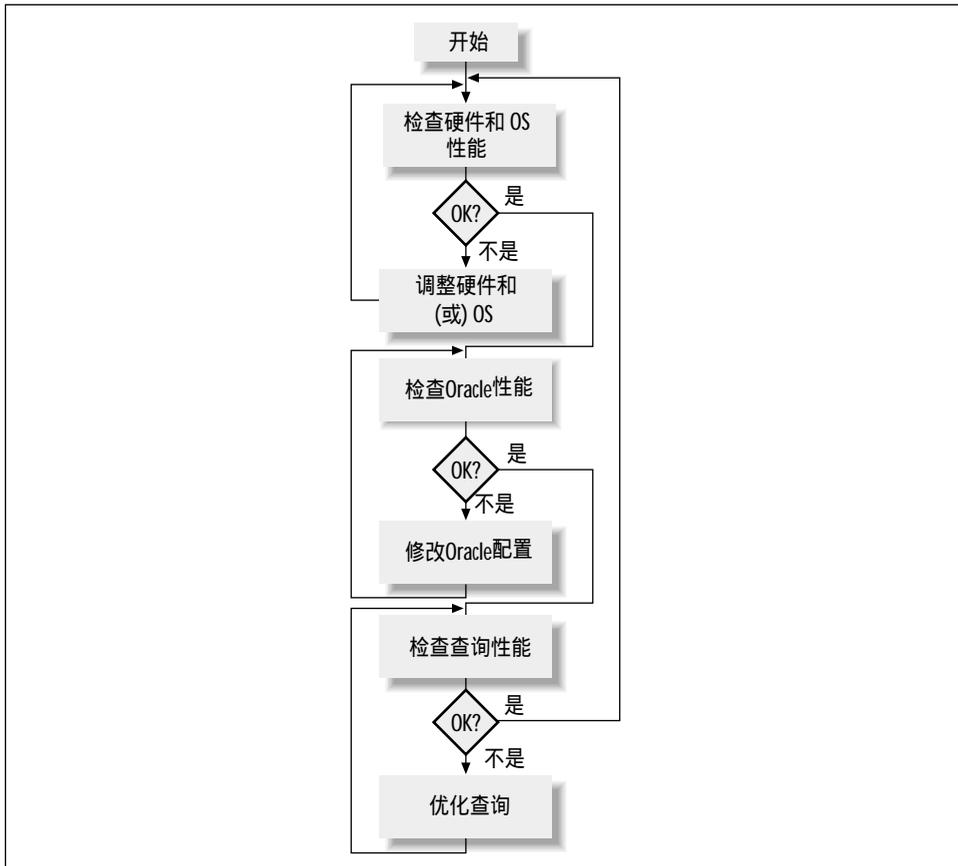


图 3-1：配置和优化过程

分区表

唯一索引 (Index-only) 表

内存的数量和分配，例如：

数据库缓冲区

重做日志缓冲区

共享池

## 可以优化什么

Oracle 数据库可以优化的方面包括下列各项：

内存使用

磁盘使用

SQL 语句执行

## 获得最大的性能

使你的 Oracle 数据库获得最大的性能并不是一下子就可以做到的，这通常是大量辛苦的工作、思考和计划的结果。然而，从付出努力所得到的回报来看，是非常值得的，你的数据库在最高效地运行，你的用户高兴，你也很满意。

我们使性能最大化的方法是按自然层次分类的。需要从 3 个不同方面，并按照顺序来阐述。这 3 个方面是：

操作系统配置

Oracle 资源配置

对象创建和 SQL 语句执行

这些方面不是互不相关的，实际上，对某方面的重要改变可能需要考虑其他方面。它们是顺序依赖的，也就是说，直到你已正确配置和调整了操作系统，你才能使 Oracle 达到很好的性能。同样，查询的快速执行取决于是否合理地配置了 Oracle 环境。

每个 Oracle 数据库的情况都是不同的，因此我们不能精确地告诉你该如何完成你的配置和优化目标，甚至你的目标是什么。我们要做的是为你提供一个我们已经成功的方法。

## 配置操作系统

这通常是容易的，因为那不是你的工作（在大部分情形下）！在大多数安装中，有系统管理员或管理者负责操作系统和硬件事宜。这个系统管理员通常是硬件和操作系统软件方面的专家，而且大多数 DBA 不用再管它。但在服从系统管理员专长的同时，这里有你必须确定的几点：

应该充分利用物理内存，但是交换（swapping）（在交换内存环境中）不应该发生。把内存交换到磁盘的过程非常慢，因此如果系统需要更多的内存，就再买一些内存。尤其是要确定你没有创建对于物理内存来说太大的 SGA，因为 SGA 交换将严重降低 Oracle 性能

CPU 在峰值时应达到 100% 使用，但是进程不应该等待 CPU

磁盘和控制器应该运行在最佳容量（通常是最大值的 60%~90% 或靠近最佳容量），而且没有输入/输出等待。作为一个 DBA，你也有一些对这个区域的控制，我们将会在本章后面描述

网络传输量不应该是一个瓶颈。考虑用主干网络把服务器连接在一起，并且如有可能把客户机/服务器通信和服务器/服务器通信分开

尽量把 Oracle 服务器放在一个单纯的机器上，把用户放到另外的机器上

确保安装了任何可能影响 Oracle 的操作系统组件（包括补丁）

因为 Oracle 是数据库市场中的一个主要提供商，所以大多数硬件提供商中都有 Oracle “专家”的职员，他们能提供可能影响 Oracle 运行的硬件和操作系统方面的建议。要充分利用这些专业意见。

## 配置 Oracle

Oracle 的总性能受所安装的组件以及这些组件如何配置的影响。Oracle 数据库的高性能对于从运行在数据库的事务中获得最大性能是很必要的。这一节为配置 SQL\*Net/Net8、MTS、并行查询（Parallel Query）和并行服务器（Parallel Server）提供了一般配置指南和一些具体建议。

## 配置指南

尽管每次安装都会不同，但总有一些能应用到大多数数据库的普遍适用的配置指南，而不管安装组件的不同和数据库应用的不同。下面章节将描述这些通用的指南。

## 查阅文档

这看起来显而易见，但还是有必要说。即使有经验的 DBA 也会从开始 Oracle 安装之前的快速阅读相关文档中受益。我们推荐你（至少）查阅下列文档：

特定于硬件的 IUG（安装和用户指南）

服务器管理员指南

版本发布说明（通常打包在介质中）

README 文件，通常可以在安装介质中找到，它包含印刷文档中可能没有的最新信息

## 检查资源需求

在开始安装之前，确认是否有足够的系统资源。与使用平台相关的 IUG 资料包含了有关磁盘存储和内存需求的全面信息。记住这些需求是最小需求，而实际上所需资源可能要更大，这与你做的其他配置有关。例如，如果你定义较大的 SGA，就需要更多的内存。

尤其是要确保在你安装 Oracle 软件（一般称为 *ORACLE\_HOME*）的设备上有足够的磁盘空间，以安装所有的软件和辅助文件。

## 检查系统特权

大多数操作系统要求执行 Oracle 安装的账户有特定的权限。一定要查看 IUG 进行确认，而且一定要确认系统管理员已经正确地进行了设置。注意这些特权可能包括在特定设备上创建目录和文件的权利。

## 确定控制文件所在位置

Oracle 要求至少有一个控制文件。你应该设置至少两个（通常更多）控制文件。这点极其重要，因为如果控制文件的所有拷贝丢失，你将不能挂接数据库。因此要把控制文件放在不同的磁盘上，如有可能放置在不同的磁盘控制器上。

## SQL\*Net 配置

要对 SQL\*Net ( Oracle7 ) 和 Net8 ( Oracle8 ) 进行配置，通常使用 Oracle 网络管理器或 Net8 助手。这通常在数据库软件安装后，并且至少有一个 Oracle 实例运行后进行，但是配置应该预先计划好。在开始 SQL\*Net/Net8 配置之前，你应了解如下内容：

网络协议的类型：用来在你所在的环境中访问 Oracle

命名模式：用来识别 Oracle 网络节点

你所在环境中的所有服务器、网关和多协议交换的名称和网络位置

一旦配置了 SQL\*Net/Net8，就要在每个服务器上设置如下文件：

*listener.ora*

控制 SQL\*Net 监听进程的操作

*tnsnames.ora*

在未使用 Oracle 命名软件时，维护网络中逻辑节点名称（别名）和物理位置之间的关系

*sqlnet.ora*

控制 Oracle 网络操作的登录（不是必需但强烈要求）

如果你使用多线程服务器，也需要在 *INIT.ORA* 文件中进行配置，如下小节所示。

## MTS 的配置

MTS（多线程服务器）在 *INIT.ORA* 文件中进行配置，*INIT.ORA* 样本的参数设置如下所示：

```

mts_dispatchers="ipc,1"
mts_dispatchers="tcp,1"
mts_max_dispatchers=10
mts_servers=1
mts_max_servers=10
mts_service=TEST
mts_listener_address="(ADDRESS=(PROTOCOL=ipc)(KEY=TEST))"
mts_listener_address="(ADDRESS=(PROTOCOL=tcp)(HOST=10.74.72.42)(PORT=1526))"

```

这个例子将配置一个 MTS，它将处理与 TEST 数据库的 TCP/IP 连接。它最多将启动 10 个调度程序，而且将创建多达 10 个服务器进程。

---

注意：记住，每个 MTS 进程都占用在 *INIT.ORA* 参数进程中指定总数中的数量，而且占用在操作系统级为 Oracle 用户开放最大的进程数中的数量。

---

## 并行查询的配置

PQO（并行查询选项）是 Oracle 的一个强大的特性，为了正确地使用它，一定要合理配置数据库。并行查询允许多 CPU 系统把数据库任务（通常是全表扫描）划分为能同时（并行）执行的一些片。为执行该任务要求如下：

通过设置 *INIT.ORA* 中的 *PARALLEL\_MAX\_SERVERS* 参数为一个大于 0 的值来使能多个并行进程

创建表空间必须使用多个数据文件，数据文件要分配到不同设备上。理论上讲，分配给每个表空间的设备数等于系统中 CPU 的数量

利用并行查询的表应该将其并行度值（使用 *CREATE TABLE* 语句中的 *PARALLEL* 子句）设置为包含表空间（表在其中创建）的数据文件的数目

## 并行服务器的配置

为了使用 OPS，并行服务器允许由多个 Oracle 实例共享一个 Oracle 数据库，你可以通过在每个参与实例中使用 *INIT.ORA* 参数来设定并行服务器特性，这些参数包括：

### *PARALLEL\_SERVER*

一定要设为 TRUE，以使能 OPS（只对于 Oracle8）。

### *INSTANCE\_NUMBER*

标识数据库的实例。

### *ROLLBACK\_SEGMENTS*

指定每个实例私用的回滚段。也可以指定公用的回滚段，但是这并非必需的。

*THREAD*

识别与实例相关的重做日志线程。

*GC\_DB\_LOCKS*

实例锁总数（仅在 Oracle7 中）。

*GC\_FILES\_TO\_LOCKS*

数据库文件锁的数目。

*GC\_LCK\_PROCS*

分布锁的总数。

*GC\_ROLLBACK\_LOCKS*

回滚锁的总数。

*GC\_SAVE\_ROLLBACK\_LOCKS*

回滚保存锁的数目（仅在 Oracle7 中）。

*GC\_SEGMENTS*

有影响空间管理行为的段的最大数目，该空间管理行为同时在段上执行（仅在 Oracle 7 中）。

*INSTANCE\_GROUPS*

把实例指定给一个或多个指定组（仅在 Oracle8 中）。

*LM\_LOCKS*

为锁管理器配置的锁数目（仅在 Oracle8 中）。

*LM\_PROCS*

锁管理器的进程数（仅在 Oracle8 中）。

*LM\_RESS*

能被每个锁管理器实例锁定的资源数目（仅在 Oracle8 中）。

*OPS\_ADMIN\_GROUP*

把实例分配给一个组来监视（仅在 Oracle8 中）。

*PARALLEL\_INSTANCE\_GROUP*

标识要用来产生并行查询从属的并行实例组（仅在 Oracle8 中）。

*ROW\_LOCKING*

应该总被设定为 ALWAYS。

*SERIALIZABLE*

应该设定成 FALSE (仅在 Oracle7 中)。

*SINGLE\_PROCESS*

应该设定成 FALSE (仅在 Oracle7 中)。

关于这些参数的详细信息参见第十二章“初始化参数”。因为 OPS 是一个非常复杂的产品，所以你应该在尝试配置并行服务器环境之前查阅《Oracle Parallel Server Concepts》和《Administration Guide》。在做配置时，记住以下几点：

在 Unix 平台上，所有的数据文件一定要创建到裸分区中

当创建一个数据库时，只自动创建重做线程 1，额外的线程需要显式建立，而且你应指明重做日志属于哪个线程

虽然不是必需的，但确保实例数目和线程数目相同将避免混淆

---

注意：术语“并行查询”和“并行服务器”经常被混淆。并行查询指单个 Oracle 实例把操作（比如一个全表扫描）在相同主机上的多 CPU 间分布并且合并结果的能力。另一方面，并行服务器是多个在不同主机上的 Oracle 实例共享一个物理数据库的特性。在这种情况下，工作是通过把用户在多个实例间分布或通过在多实例间产生并行查询进程来在 Oracle 实例间分布的。

---

## 调整和配置数据库对象

要获得最大的数据库性能，数据库对象的合理大小和配置是非常重要的。对象的合理大小是一个不断进行的工作，随着不断创建对象和修改对象，也需要不断地检查对象特性并在需要时将其改变。以下与调整大小相关的问题的数值与性能成反比：

### 表空间碎片 (tablespace fragmentation)

表空间碎片使许多无法使用的小延伸区分散在表空间中。当创建对象时，如果延伸区的 INITIAL 或 NEXT 的值设置不合理就会产生碎片。

### 行链 (row chaining)

这个问题将导致单行的数据驻留在多个 Oracle 块中,典型情况发生在 PCTFREE 设置不足且表不断进行更新时。

### 多个延伸区 (multiple extent)

多个延伸区,可能导致一个特定对象的数据分散在一个或几个数据文件之中,这是由在创建对象时指定了不合适的 INITIAL 或 NEXT 延伸区大小而引起的。这个问题可能会在 MAXEXTENTS 参数被允许为默认值时变得很严重,因为尝试分派一个超过那个数目的延伸区将导致失败。

### 日志等待

当写日志缓冲区记录到一个日志文件或当日志文件切换时,日志等待将引起一个进程等待,这时日志等待能大大增加处理时间。这通常由日志文件数目太少和日志文件太小等原因引起。

### 扩展一个回滚段失败

这样的失败(能引起一个事务回滚)是由于没有分配充足的回滚段数目,或者是由于分配的回滚段不够大。

下列各节讲述了可以避免这些性能问题的一些指南和建议。

## 表

表是 Oracle 数据库中数据存储的基本单位,因此表的配置和由此产生的性能将对数据库总体性能产生很大的影响。下面是表配置的一些指导方针:

试着估计一个表将多大,并且分配一个足够大的初始延伸区来存放整个表。然而,如果你正在使用并行查询,则应跨越不同的数据文件来分配总的空间,使分配的延伸区数目和表的并行度相等

考虑使用多个表空间,每个表空间对应于不同大小或类型的表。例如,你可能有 3 个表空间: LARGE\_DATA、MEDIUM\_DATA 和 SMALL\_DATA,每个会用来存储大小不同的表。如果你使用多个表空间,要确保把每个表分配到恰当的表空间中

确保分配一个 DEFAULT TABLESPACE 给每个用户。如果没有分配,Oracle 将使用 SYSTEM 表空间作为默认值

如果可能，保证 INITIAL 和 NEXT 延伸区大小总是相同大小的单元的整数倍，例如，是 512K 的整数倍。这样，延伸区将是统一的大小，而且会比较容易分配额外的延伸区，而不引起表空间碎片。如果可能，在一个表空间中可以使用大小相同的延伸区

设定 PCTINCREASE 参数为 0，为了防止延伸区分配失控和保持统一的延伸区大小。

设定 MAXEXTENTS 参数为 UNLIMITED。这将防止延伸区用完，因为多个延伸区对它们产生很小的性能影响（虽然广泛分布的延伸区对性能有负面影响）。这样做可以防止错误，但是不要把它作为 INITIAL 大小的替代

如果表没有更新，则设定 PCTFREE 为 0。如果表有更新，则估计行的列的增长程度，并分配一个 PCTFREE 来防止块链接，而在块中没有过多的未使用空间

如果有很多事务同时访问表，将 INITRANS 设定为一个大于 1（默认的）的数将 MAXTRANS 设定为在预期表上同时访问的最大数目。一个较小值将会导致一个或多个事务等待前一个事务完成

## 索引

正确使用索引可以使性能大大提高，这是任何 Oracle 单一特性所不能做到的。虽然许多性能提高获益于优化 SQL 语句（见第八章“查询优化”），但我们也提供一些配置指南：

为索引创建一个单独的表空间，并且保证这个索引表空间的数据文件与包含索引表的表空间的数据文件不在同一个磁盘上

试着去估计索引的大小而且分配一个足够的 INITIAL 延伸区来存储整个索引，除非你正在使用并行查询，否则在这种情况下，你应该在与索引的并行度相同的数据文件之间分配总的空间

如果可能，保证 INITIAL 和 NEXT 延伸区大小总是相同大小的单元的整数倍，例如，是 512K 的整数倍。这样，延伸区将会是统一的大小，而且会比较容易分配额外的延伸区而不引起表空间碎片

设定 PCTINCREASE 参数为 0 以防止延伸区分配失控并保持统一的延伸区大小

设定 MAXEXTENTS 参数为 UNLIMITED。这将防止延伸区用完，而多个延伸区可能产生的性能影响很小（虽然广泛分布的延伸区对性能有负面影响）。这样做可以防止错误，但是不要把它作为 INITIAL 大小的替代

## 回滚段

Oracle 用回滚段来维护数据的一致性，允许事务的取消或回滚。回滚段使用很多的输入/输出，下面是配置回滚段的一些指导方针：

为回滚段创建一个单独的表空间，如果可能，把这个表空间的数据文件放在一个与其他数据文件不同的磁盘上

永远不要在 SYSTEM 表空间中创建回滚段（除了在数据库创建期间需要的临时回滚段以外，见第二章“安装”）

确保为回滚表空间分配了足够大的空间，以允许回滚段为了适应大的更新事务来按照需要增长空间。记住批事务容易产生很大的回滚段

总是保持回滚段的 INITIAL 和 NEXT 延伸区使用相同的数值（在 CREATE TABLESPACE 语句中的 DEFAULT STORAGE 子句中定义）。为回滚段分配大小相等的块可以防止空间碎片

记得每个回滚段必须至少有两个延伸区，因此段的初始大小实际上是 INITIAL + NEXT 的总和

定义一个 OPTIMAL 值，以便使为了延伸适应一个大事务而增长的回滚段可以回缩到一个合理的大小。然而，不要让这个值太小，否则会浪费时间来为回滚段分配额外的延伸区

## 排序区

Oracle 使用 *INIT.ORA* 参数 SORT\_AREA\_SIZE 来为数据排序分配内存。当一个排序不能在内存中完成时，Oracle 使用数据库中的临时段，但这非常慢。应当小心平衡 SORT\_AREA\_SIZE，因为大的排序区可以通过减少输入/输出来显著增加性能，但是这将用光内存并引起分页。

---

注意：记住这个参数应用到每个用户进程。每个执行排序的用户进程都将分配 SORT\_AREA\_SIZE 内存。因此，如果 SORT\_AREA\_SIZE 被设定为 /MB，而有 100 个用户进程正在执行排序，那么将分配总数为 100MB 的内存。

---

## 临时表空间

如果没有为执行排序的用户进程分配足够的内存，那么 Oracle 将通过为用户在 TEMPORARY TABLESPACE 参数指定表空间中创建临时段来在磁盘上执行排序。除此之外，临时段用来执行复杂查询，如连接、UNION、MINUS 和索引创建。临时区的指南如下：

为临时段创建一个单独的表空间（通常叫做 TEMP），如果可能，把这个表空间对应的数据文件放在一个单独的磁盘上

在 CREATE TABLESPACE 命令的 DEFAULT STORAGE 子句中指定 INITIAL 和 NEXT 参数。把两者的值设为相等，以消除空间碎片，在 TEMP 表空间中极易产生碎片，因为在那里不断地创建并删除对象

要确保为每个用户指定一个 TEMPORARY TABLESPACE。如果没有指定，Oracle 将把 SYSTEM 作为默认的表空间，而这样对性能有负面的影响

## 重做日志

重做日志，也称联机重做日志文件，对 Oracle 的失效恢复能力至关重要。重做日志的适当配置不但对数据库的总体性能很关键，而且对恢复数据库的能力也很重要（见第四章“防止数据丢失”）。相关指南如下：

使用 Oracle 内嵌的镜像特性，把重做日志文件的多组放在不同的磁盘上

分配足够的重做日志文件以便 Oracle 无须为了重复使用一个文件而等待它。Oracle 至少要求有两个重做日志文件，但是 4 个或更多个是必要的

分配的重做日志文件要足够大以防止太多的日志文件切换，但又要适当的小以保证当前联机日志文件失效时很好地恢复。如果文件较小，可能恢复已经归档的所有事务，而大的日志文件使数据库有可能丢失更多的事务

设置 *INIT.ORA* 参数 *LOG\_CHECKPOINT\_INTERVAL* 值大于重做日志文件的大小，这样将避免检查点（checkpoint）进程，直到日志文件满为止（引起一个检查点进程）。这个参数以数据库块来表达

---

警告：记住一个日志切换将导致从 SGA 将脏缓冲区（例如有更新）写入到磁盘。

---

如果你正在运行 Oracle7，考虑设定 *INIT.ORA* 参数 *CHECKPOINT\_PROCESS* 为 TRUE。这么做将创建一个执行检查点进程的单独进程，而并非由 LGWR（日志写入进程）处理。见第十章“Oracle 实例”，可以了解更多信息

## 归档日志目的地

在配置方面一个经常需要注意的问题是要保证归档日志目的地有足够的空间。如果数据库正在归档日志模式中运行，那么当一个联机重做日志文件填满时，Oracle 的 ARCH 进程将复制这个文件的内容到 *INIT.ORA* 参数 *ARCHIVE\_LOG\_DEST* 指定的目录。如果目的地太小，ARCH 就不能复制日志文件，而一旦所有的联机日志文件满，整个数据库就会停止，直到这个问题解决。有经验的 DBA 已经意识到这种情况，这种情况大多数在半夜发生，就像 REM 休眠一样。

## 优化 Oracle

或许 DBA 的工作没有哪一方面能像优化这样消耗时间。成功的 Oracle 优化既要求知识又要求经验，挑战和挫败也同时存在。整卷都在写 Oracle 优化（参见附录“DBA 使用的资源”），但我们不能在一节中包括优化的所有方面。相反，正如我们前面提到的，我们将列出可以应用到各种情况的优化方法的大纲。

## 结构化优化方案

Oracle 数据库的成功优化需要仔细的、有规则的方案。像整体系统配置一样，优化一定要包括下列各项：

硬件和操作系统性能

## Oracle 实例性能

### 单独的事务 (SQL) 性能

这些方面应该按顺序进行,因为没有硬件和操作系统的良好优化,Oracle的性能优化是不可能的。没有数据库的有效运行,一个单独的SQL语句不可能很好地被优化。优化这些方面中的任何一方面时,都包括3个步骤:

1. 测量目前的性能。
2. 做适当的变化。
3. 评估结果。

---

警告:对Oracle实例的一些改变可能引起对操作系统环境的改变。比如,分配附加的数据库缓冲区可能导致操作系统开始分页,而这可能要求额外的操作系统优化来消除。

---

优化进程几乎总是反复的。也就是说,在完成3个步骤之后,DBA必须回到第一步骤并且重复这个过程。这将一直持续到不会再有性能改善为止。

## Oracle 实例优化

Oracle实例层的大多数性能的提高将通过两个方面达到:内存使用和磁盘输入/输出。

### 内存使用

基于内存的操作比磁盘操作快得多(有时成千上万倍),这一点是不值得惊讶的。结果将导致用内存访问数据来代替磁盘输入/输出,以使性能得到巨大的提高。相关的3个主要方法描述如下:

#### 分配额外的 *DB\_BLOCK\_BUFFERS*

这或许是改善总体性能的单一的最有效的方法,特别是在查询上。更多的数据库缓冲区允许更多的数据块数据保持在内存中,因此可以按内存速度访问包含在这些块中的数据,而不需要磁盘输入/输出。缓冲区是由 *INIT.ORA* 参数

DB\_BLOCK\_BUFFERS 来分配的，它的数值是要分配的数据块缓冲（block buffer）数目。因此，如果数据库块大小是 8192，每个 DB\_BLOCK\_BUFFER 将是 8192 字节。注意改变 DB\_BLOCK\_BUFFERS 值后直到下次数据库重启才生效。

---

注意：注意不要分配太多的 DB\_BLOCK\_BUFFERS 而导致操作系统开始分页，分页将消除你获得的性能，而且将对整体性能产生负面影响。

---

### 分配额外的共享池

共享池大小由 *INIT.ORA* 的参数 SHARED\_POOL\_SIZE 控制，它指定了以字节为单位的共享池大小。共享池的主要内容是字典缓存区（dictionary cache）和共享 SQL 区（shared SQL area）。由于字典缓存区的各部分组件由 Oracle 自动分配，所以共享池的任何增加都会使字典缓存区和共享 SQL 区增加。

共享 SQL 区包含最近执行的 SQL 语句的拷贝，连同相关信息，如它们的执行计划。共享池越大，特定 SQL 语句被解析并且驻留在共享 SQL 区就越有可能，因此节省了需要再次处理这个语句的时间。在相同的 SQL 语句被多次执行且对速度有要求的事务处理系统中，这是个特别重要的值。

### 分配更多的日志缓冲区空间

日志缓冲区是用来存储将要写到联机重做日志文件上的数据的。日志缓冲区的大小由 *INIT.ORA* 参数 LOG\_BUFFER 控制，其数值以字节表示。为日志缓冲区分配更多的内存，将减少磁盘输入/输出，尤其是在事务特别长或数量很多时。

### 磁盘输入/输出

磁盘访问是任何计算机系统上的最慢的操作。作为一个数据库系统，Oracle 的存储和访问数据非常依赖磁盘访问。考虑一个典型的更新一个表的一行的 SQL 语句，将发生下列各项操作：

1. 读数据字典获得关于表和正在被操作的行的信息。
2. 读适当的索引来定位要更新的行。
3. 读包含行的数据块。

4. 写回滚信息到一个回滚段。
5. 写更新信息到联机日志文件。
6. 重写数据块。
7. 重写索引块。

虽然一些操作可以通过有效使用内存消除，正如我们前面提到的，但所有这些操作都潜在地要求磁盘输入/输出。通过尽可能使磁盘输入/输出有效，可增强总性能。使磁盘输入/输出最大值的基本指南如下：

只要可能，分离输入/输出操作到单独的磁盘上。这样，在执行另外一个时，不需要等待一个磁盘操作完成。例如，如果回滚段和日志文件在相同的磁盘上，需要写回滚记录，然后磁盘磁头需要移动到日志文件记录要写的那部分磁盘。这是非常耗时的

把高输入/输出的磁盘放在不同的控制器上。现代的控制器的处理有限数目的并发操作，但是尽可能使用更多的控制器将消除任何控制器等候并加速性能。把最忙的文件和表空间（例如，日志文件、回滚段、一些索引）放在最快的可用磁盘上

### 关于 RAID 的注释

磁盘技术的最新发展使 RAID（廉价磁盘冗余阵列）成为许多系统上很常用的一个选项。通常，当使用术语 RAID 时，硬件管理员会立刻想到 RAID 5（或 RAID-5），它允许多个磁盘结合成一个大的设备。通过分配一个磁盘设备来存储冗余数据，一个 RAID-5 磁盘阵列可以承受阵列中的任何单一磁盘失效，并且经常是热交换的，也就是当一个磁盘失效时，其他的磁盘继续工作的同时更换这块磁盘，而不必关闭系统。

事实上 RAID-5 是非常强大且廉价的。当配置 Oracle 数据库时，在大多数情况下要避免使用这种技术。这可能听起来很刺耳，但是事实上虽然 RAID-5 成本较低，而且提供了很好的数据保护级别，但是它的磁盘输入/输出花费很高。尤其是在 RAID-5 阵列上的写操作要比在单一磁盘上的写操作慢得多。

RAID-5 阵列的一个好的替代品是 RAID1，就是大家都知道的磁盘镜像。虽然比 RAID-5(一半磁盘用来存储冗余数据)更贵，但是 RAID-1 提供了完全的数据保护，而在输入 / 输出效率上没用降低。

---

警告：RAID-1 要求有足够的硬件资源。尤其是由于每次写操作实际都要对磁盘进行两个写操作，所以控制器上的负荷是非 RAID 的两倍。

---

现在性能最好的 RAID 是 RAID-0+1，有时叫做 RAID-10。RAID 的这个级别在多个驱动器上结合了带有数据条带 (RAID-0) 的镜像磁盘 (同 RAID-1)，这可以消除等待磁盘头定位的延迟。而这在其他的 RAID 控制器中都没有，RAID-0+1 是值得考虑的。

### 操作系统条带化

许多操作系统提供了在多个磁盘设备中磁盘扇区的自动条带化。条带化允许磁盘输入 / 输出连续，而没有头定位的延迟。虽然这个技术提供了比在一个单一磁盘上更好的性能，但也有一些缺点：结合多个磁盘为一个单一的条带单元意味着 DBA 不能够控制单个文件在单独磁盘设备的位置。如果你的系统上只有少数的大磁盘，你应该考虑操作系统条带化，但是多磁盘设备或多 RAID-0+1 阵列通常会从 Oracle 产生更好的性能。

### Oracle 条带化

作为 DBA，通过小心地把数据文件分配到单个磁盘设备或 RAID-0+1 阵列，你可以达到与操作系统条带化相似的结果。例如，建立跨越 4 个磁盘的 Oracle 条带化需做下列各项工作：

创建一个有 4 个数据文件的表空间，每个位于一个不同的磁盘设备上

在表空间中创建对象，指定 MINEXTENTS 4。Oracle 将把 4 个延伸区分配到 4 个数据文件上，这样就实现了条带化。这个行动不是自动的，还需使用 ALTER TABLE ... ALLOCATE EXTENT 命令

Oracle 条带化技术非常强大，尤其和并行查询结合的时候，将通过多 CPU 处理查询过程。

## SQL 优化

假如主机服务器和操作系统正在你的站点顺利运行，而且你配置并优化了 Oracle，使其在最好状态运行，但是你的重要应用程序仍然运行很差。不幸的是这种情况经常发生。解决方法是通过检查和优化正在被执行的 SQL 语句来优化应用程序。

SQL 优化这个题目值得写一本书。实际上，在市场上有很不错的书阐述得比这里更为详细。我们建议你检查附录中列出的 DBA 资源。在这小节中，我们将为你提供优化 SQL 语句的一些概要建议和指南。

### 查询处理

第八章“查询最优化”描述了 Oracle 如何为一个特定的 SQL 语句创建一个计划。Oracle 现在用两个方法中的一个来决定该如何执行一个 SQL 语句：

#### *基于规则的方法*

应用一个标准的、固定的（但是经常有效的）规则集到语句中。

#### *基于费用的方法*

考虑由一个 SQL 语句（连同可得的索引一起）引用的对象的可用统计信息，并基于这些统计建立一个计划。

优化一个 SQL 语句的关键是理解 Oracle 查询优化器如何工作和知道怎样改变 Oracle 的行为，以便它能更有效地处理语句。

当然，在优化一个 SQL 语句之前，必须知道它在做什么和如何做。今天市场上的许多工具将有助于完成这个工作，而且最有用的工具之一是 SQL\*Plus 中的 EXPLAIN PLAN 命令。通过创建一个计划表（通常为 PLAN\_TABLE）并且检查 EXPLAIN PLAN 语句的结果，你会容易地看到 Oracle 如何执行一个特定的语句。例如，SQL 语句：

```
SELECT ename,loc,sal,hiredate
FROM    scott.emp, scott.dept
WHERE   emp.deptno=dept.deptno;
```

可用下面的命令解释：

```

EXPLAIN PLAN SET STATEMENT_ID='DEMO' FOR
SELECT ename,loc,sal,hiredate
FROM    scott.emp, scott.dept
WHERE   emp.deptno=dept.deptno;

```

存储在 PLAN\_TABLE 中的结果可以通过下面的简单查询看到:

```

SELECT LPAD(' ',2*level) || operation || ' ' || options || ' ' ||
       object_name EXPLAIN_PLAN
FROM   plan_table
CONNECT BY PRIOR id = parent_id
START WITH id=1

```

看起来像这样:

```

EXPLAIN_PLAN
-----
NESTED LOOPS
  TABLE ACCESSFULL DEPT
    TABLE ACCESSFULL EMP

```

这个计划表明将使用全表扫描来对 DEPT 和 EMP 表访问。这对象 EMP 和 DEPT 一样的小表很好,事实上,我们想要它们全表扫描,因为表将缓存到内存中,并且不需要磁盘输入/输出(至少在第一次运行之后)。然而,如果表很大,这个查询将进行很长时间,所以我们想改变查询执行的方式。

有 3 个基本方法可以改变 Oracle 查询优化器的行为:

在执行查询时提供一个或多个索引来用

重写 SQL 来使用一个更为有效的方法

以提示 (hint) 的形式提供查询优化器指导

如果我们试第一选项而且在 EMP (deptno) 上增加一个索引,计划将改变为:

```

EXPLAIN_PLAN
-----
NESTED LOOPS
  TABLE ACCESSFULL DEPT
    TABLE ACCESSBY ROWID EMP
      INDEXRANGE SCAN EMPDEPT_IX

```

现在你能看见 Oracle 使用索引通过 ROWID 来从 EMP 取回行,ROWID 是从新创建的索引中获得的,而不再需要全表扫描。

通常使用 SQL 会有不止一种实现一个特定功能的方法，在确定使用正确的 SQL 语句之前尝试几种不同的方法（有适当的基准）是一种很好的实践方式。第八章“查询最优化”提供了关于 SQL 优化的更详细信息。

## 其他有用的优化特性

Oracle 通过增加提高性能的新特性来不断改良数据库产品。检查即使只有很小更新的 Oracle 版本注释是很重要的，因为这其中可能就包含了新的性能特性。你可能觉得有用的一些特性和工具列表如下：

### 分区表（partitioned table）

从 Oracle8 开始分区表就允许在多个子表上创建表，每个子表包含了表数据的一个特定子集。例如，一个表可以按年分区，所有 1998 年的数据在一个分区中，所有 1999 年的数据在另一个分区中等等。分区对大的表特别有用，因为对包含在一个可识别的子集中的数据进行查询可以在对应的分区中操作，而不用访问其他的分区。例如，更新 1999 年的记录只要求 Oracle 在 1999 年的分区上执行输入 / 输出操作。可在 CREATE TABLE 语句中指定分区。为了使用这个特性，你必须：

标识将要定义分区的数据字段（例如 sales\_year）

在 CREATE TABLE ... PARTITION BY RANGE 子句中指定值的范围

为表的每个分区指定一个另外的表空间（为了得到最好的性能，把表的每个分区放置在不同的磁盘上）。注意单独的表空间不是必需的，但是这种做法可以允许表的一个分区脱机，而维持访问表的剩余部分

分区表通常应该伴随着相应的分区索引，如下：

使用 CREATE INDEX 命令的 LOCAL 关键字来告诉 Oracle 为索引表的每个分区创建一个单独的索引

使用 CREATE INDEX 命令的 GLOBAL 关键字，以便告诉 Oracle 使用不会对应到索引表分区的值来创建一个单一索引。全局（GLOBAL）索引也可以分区

## 惟一索引表 ( index-only table )

在某些情况下,正常时存储在一个表中的所有数据可以存放在一个索引中,这样表就没必要了。从 Oracle8 开始,一个惟一索引表就使数据按照主键列排序。对这种类型的对象有一些限制:

由于数据没有存储在一个表中,所以没有 ROWID 可用

必须为表定义一个主键

不能创建其他的索引,只有主键可以被创建为索引

惟一索引表通过使用 CREATE TABLE 命令的 ORGANIZATION INDEX 子句创建。

## 位图索引 ( bitmap index )

当被索引的数据有低基数 ( cardinality ) 时 ( 也就是说索引列有相对较少的确定值时 ), 创建位图索引可以大大改善性能。位图索引的一个好的例子就是性别 ( GENDER ), 它只有两个值 “ M ” 或 “ F ”。对于销售总额 ( SALES\_AMOUNT ), 将不适合建立位图索引,因为它对于每行都可能有一个不同的值。

创建一个位图索引类似于创建一个标准的索引,你可以在 CREATE INDEX 语句中包括关键字 BITMAP。例如,在 EMPLOYEE\_MASTER 表的 GENDER 列创建一个位图索引,指定下列语句:

```
CREATE BITMAP INDEX empmast_ix ON employee_master(gender);
```

## 临时表空间

Oracle7 引进了临时表空间的概念,这专门为 Oracle 的排序段使用。通过消除连续不断地分配和释放排序空间的空间管理操作,当排序很大以致内存不能容纳时,所有使用排序的操作将得到性能的提高。这在运行 OPS 的时候尤为重要。

---

注意: 一个临时表空间只能用于排序段,不要在临时表空间中创建永久对象。

---

要创建一个临时表空间,需要在 CREATE TABLESPACE 语句中使用关键字 TEMPORARY。例如,下面语句将创建一个叫做 TEMP 的临时表空间:

```
CREATE TABLESPACE TEMP
DATAFILE '/disk99/oracle/oradata/TEST/temp01.dbf' SIZE 50M
DEFAULT STORAGE(INITIAL 64K NEXT 64K MAXEXTENTS UNLIMITED)
TEMPORARY;
```

一个已存在的非临时表空间可以转变为临时表空间，如果它不包含永久对象，可使用下面 SQL 语句进行转换：

```
ALTER TABLESPACE tablespace TEMPORARY;
```

## 不能恢复的操作

由 Oracle 7.2 开始，在创建表或索引时就可能不写重做日志记录。这个选项提供了较好的性能，因为需要的输入 / 输出少了很多。要利用这个特性在对象创建语句中指定 UNRECOVERABLE( Oracle7 语法 ) 或 NOLOGGING( Oracle8 语法 )。例如，你想使用数据库链接从另外一个数据库移动数据可使用下面这个语句：

```
INSERT INTO newtable
SELECT * from oldtable@oldlink;
```

这个方法当然会奏效，但是将为每次插入创建重做日志记录，这将浪费资源。可以用下面语句完成相同的任务：

```
CREATE TABLE newtable AS
SELECT * from oldtable@oldlink
NOLOGGING;
```

当重建索引的时候，NOLOGGING 选项将特别有用。NOLOGGING 关键字可以大大减少索引创建的时间。SQL 语句与下面语句相似：

```
CREATE INDEX indexname ON table(column)
NOLOGGING;
```

注意，如果在执行一个不能恢复的语句之后，在某点发生了系统失效，你将不能使用回滚机制来恢复这个事务，你必须意识到一个系统失效已经发生并且要重新运行语句。