## 1、性能调优基础知识

性能设计的因素非常多，它是数据库运行过程中遇到的棘手问题之一。性能问题当然要考虑数据库，但是也要考虑平台部署的策略。今天，数据库应用的基础架构存在于多个平台之上，包括数据库服务器和应用服务器。需要网络和连接的带宽，也需要考虑不同用户使用情况的复杂性。

性能的奇怪方面之一就是“好的性能”是在缺乏的时候定义的，而不是出现的时候定义的。可以容易地识别出糟糕的性能，而好的性能通常简单定义为没有出现糟糕的性能。

在开始讨论具体的Oracle性能问题之前。我们有必要明确一下研究性能问题的基本方法。

理解如何解决Oracle数据库中的性能问题需要三个基本步骤：

1. 明确性能和性能问题
2. 检查Oracle服务器软件的性能
3. 检查服务器机器的整体性能。

### 1.1明确性能和性能问题

性能调优的第一步就是确认是否真的存在性能问题。

糟糕的性能不可避免是用户感到失望的结果，系统并没有按照用户期望的那样去运行。因此，你首先必须对用户期待的现实性进行评估。

如果这种期待是现实的。例如，性能问题可能仅发生在某个特定的事务且在指定的时间发生，或者所有的事务和报告都低于用户的预期。

一旦明确了对系统期待的性能，你就可以开始查找系统性能问题的所在。当对特定资源的需求大于可用的资源，应用程序必须等待共享资源，系统就会慢下来；这个时候性能问题就产生了。

### 1.2Oracle服务器性能

通常查找资源瓶颈的第一个位置是Oracle数据库软件，可用使用Oracle企业管理器来识别Oracle内部没有被充分利用的资源。数据库中的瓶颈将会导致会话不必要的等待，性能调优的目的就是要排除这些瓶颈。

借助Oracle动态性能视图，可以查看到数据库中的各个瓶颈。在Oracle引入自动化工作负荷仓库（AWR）、自动化数据库诊断监控器（ADDM）以及企业管理器网格控制之前，为了确定瓶颈所在，数据库管理员执行的第一步往往是查询性能视图。所有这些性能视图它们的名字都以V$开头，从Oracle 9i开始，还提供全局视图（用于实时应用程序集群或者RAC数据库中的所有节点），它们的名字则以GV$开头。这些视图中两个特别有助于识别等待资源的源头，它们为管理员分析性能问题提供了非常有价值的信息：

V$SYSTEM\_EVENT

收集了有关整个数据库实例正在等在的资源的系统信息。

V$SESSION\_EVENT

列出了各个会话中累计发生的等待事件。

V$SESSION\_WAIT

收集了每个会话正在等待或者曾经等待过的资源的详细信息。这些信息都是会话相关的。

V$SESSION

收集了每个正在等待或者曾经等待过资源的会话信息。

借助这些视图，数据库管理员可以查明导致最多等待的资源针对该资源采取适当的措施可带来巨大的性能提高。

你可能会发现在你的环境中引起性能问题的原因很简单，例如，数据库高速缓存命中率低于预期。如果是高速缓存没有工作在最佳状态，可以简单地通过修改初始化参数DB\_BLOCK\_BUFFERS来增加高速缓存的大小。这样做可能会改善缓存的命中率。修改之后，可以在V$METRICNAME视图中查看高速缓存的命中率。

其他的情况可能就没有那么清晰，无法通过简单地查看性能视图中的参数就能解决。例如，你肯发现从数据库磁盘中读取数据行花费的时间较长。这种情况可能是由于数据库服务器磁盘产生了竞争而造成的。

### 1.3AWR、ADDM以及企业管理器

进行进行性能监控和管理的最好方法就是使用企业管理器（在RAC实现中也叫做网格控制）。自动工作负荷仓库（AWR）负责捕获和存储有关Oracle工作负荷资源利用情况的信息。在默认情况下，统计信息每30分钟捕捉一次，存储7天。从视图中可以看到这些统计信息。不过企业管理器提供一个更加简单易用的界面。

AWR可也通过比较工作负荷的运行时间帮助Oracle数据库查找潜在的性能问题。这个特性也是Oracle 10g以来引入的各种易管特性的基础；自动化数据库诊断监控器（ADDM）就是其中之一。

ADDM能够自动识别和报告各种资源瓶颈。例如，CPU竞争、锁相关的问题或者某些SQL语句性能很糟糕等。在Oracle 11g中，ADDM可以在集群中执行分析任务。ADDM能够向企业管理器的面板发出告警，向管理人员指出当前发生的资源竞争情况。企业管理器能够提供Oracle服务器资源利用情况的高级视图和详细视图，从而帮助管理员迅速找到性能问题的缘由。可以在企业管理器上设置一个门限值，当特定的资源达到这个临界的利用级别时，就会在面板上提示管理员。企业管理器包含许多顾问工具，它们可以为管理员提供一些关于如何调优应用程序以及如何优化Oracle数据库性能的建议。

如果需要对应用程序进行调优，那么你可能需要使用SQL顾问工具。这个工具由Oracle 11g引入，它集成了SQL调优顾问，SQL访问顾问以及分区顾问的功能。SQL顾问工具能够充分利用AWR中收集的有关CPU和I/O开销的信息，识别出ADDM只是的高影响的SQL语句，并给出建议。具体而言，它会先检查通信是否过时；接着通过SQL Profiling识别最佳路径，并确定是否需要添加索引、物化视图或者其他的数据库结构；最后指示对高影响的SQL语句进行修改是否能够提高效率。

下面将一一介绍关键的数据库调优顾问工具：

内存顾问：

用于为自动化内存管理设置最佳的MEMORY\_TARGET参数，以及为共享内存管理设置最佳的SGA\_TARGET参数。

分段顾问：

用于存储管理和空间分配。

撤销顾问：

用于管理事务。

其他的顾问工具，如故障平均恢复时间（MTTR）顾问，负责优化Oracle的服务器设置，包括日志文件。

### 1.4机器资源利用率

如果数据库可用的资源不足，也可能会遇到性能问题。Oracle数据库部署不当，增加机器的资源最初可能有助于减少性能瓶颈，但是用这种办法解决问题成本相当高。而且，当新增的资源被消耗之后，问题又会重现。Oracle数据库设计合理配置得当，如果出现这类资源短缺的问题，这个时候添加及其资源才是有意义的。

Oracle数据库的性能取决于它如何使用可用的机器资源。这些机器资源包括：处理能力即CPU、内存、磁盘I/O以及网络带宽。

服务器和客户端之间的网络带宽，今天看来已不是问题。在RAC部署环境中，你还应该留意节点之间互连的带宽，过大的流量会导致系统性能下降。不过这种互连的带宽也在不断增加。

接下来将要专注于Oracle是如何使用CPU、内存和磁盘I/O这三种关键性及其资源的。访问磁盘通常最慢，因而最常见的数据库性能问题都和I/O相关。

数据库服务器可能会遇到由多个资源的竞争而导致的瓶颈。事实上，计算机环境在设计的时候考虑到一种资源匮乏的时候另一种资源可以尝试着补偿，这样做有时候也会导致补偿的资源出现赤字。如果物理内存被用完了，操纵系统会将内存区置换到磁盘中去，这就会导致I/O瓶颈。

可以使用Oracle企业管理器和机器制造商提供的工具或操作系统工具来查看机器资源的利用情况。企业管理器10g开始包含一个自动化性能监控（APM）的性能分析工具。APM使得管理人员可以创建灯塔进程，该进程在客户端上周期性地执行事务并报告响应时间。它处于Oracle环境之外，从一个终端用户的角度来帮助网络管理员掌握性能情况。

### 1.5其他的一切都失灵了

性能调优未达到预期结果也可能导致性能问题的产生。例如，为了解决I/O速度慢的问题，你可能会对磁盘子系统进行调整或者增加其吞吐量。然而，这种情况可能是由于SQL调优没有做好而导致的，重写SQL就能很好地解决此类问题。

这个时候，你应该分析各个模块之间的交互以及应用系统和数据库服务器中的SQL语句。你可能会发现有些SQL语句确实会导致性能问题。不过，更可能的情况是，发现必须重新思考应用系统的设计问题。

企业管理器以及自动化数据库诊断监视器（ADDM）可以自动识别那些占用了最多资源或者没有达到最优的SQL语句。SQL调优顾问组件甚至能够针对发现的性能问题建议解决方案。

### 1.6关于性能基础知识的最后注释

性能潜在影响现实世界的业务。每当你试图解决问题的时候，你必须确保仔细监控你所要改善的领域，尤其要注意在你修改前后系统的变化。AWR收集的重要基线数据包括应用程序、数据库、操作系统、磁盘I/O以及网络统计。

你必须采用一种系统的方法来发现性能问题的源头并实施适当的解决方案。这个方法要求在做任何变动之前必须为资源的利用情况和响应时间建立基线，而且要求在重新考察系统环境改动后的性能之前只能做少量改动。通常管理员只会简单地修正这个问题，而不会采取标准的方法，这种做法往往会给后面的维护工作带来其他问题。

## 2、Oracle与磁盘I/O资源

从机器资源的角度来看，输入/输出操作可以定义为计算机操作系统在数据库服务器底层磁盘子系统中读写数据的过程。I/O操作的数据块可能比较较小，也可能比较大。操作系统不同，数据块大小的下限和上限也不同。Oracle数据库也有自己的块大小，这个值数据库管理员可以自己定义，它又叫做数据库块大小。

Oracle数据库发出的I/O请求有两种基本大小：

单个数据块I/O

多块I/O

Oracle数据库使用多块I/O读取大量数据，因而有时候全表扫描获取数据的速度实际上比基于索引的获取要快（例如，当索引的选择性比较差的时候）。Oracle执行多块操作快于对应的单块操作。

## 3、Oracle数据库I/O规划原则

当你在规划磁盘布局以及各种数据文件安放的时候，你需要考虑Oracle执行I/O的不同原因以及对性能潜在的影响。

Oracle执行I/O操作的主要对象有以下几种：

* redo日志。
* 表中的索引。
* 数据库字典，它在SYSTEM表空间中。
* 排序行为，它在用户执行排序的TEMP表空间中。
* 回滚信息，它分散在包含数据库回滚段的表空间内各个数据文件中。
* 归档的redo日志，它保存在归档日志目标目录（假设数据库运行在ARCHIVELOG模式）

管理这些类型的I/O有一些简单的原则，遵循这些原则可以优化Oracle对数据库服务器磁盘子系统的使用：

利用磁盘分段（disck-striping）技术将I/O操作分散在多个轴（spindle）中进行

利用表空间直接隔离和定位各种类型的I/O

将redo日志及其镜像放置在两个最不忙的设备上

将“系统开销”均匀分摊到可用的磁盘驱动器上

将归档文件和redo日志文件放在不同的设备中

从数据库可用性的角度来看，还有一些其他的文件放置问题需要考虑：

如果将数据库的备份保存在某个磁盘中，那么就不要再将其他的数据库组件放到

该磁盘中

确保用于保存归档日志的设备不要保存任何数据库组件或者数据库备份

即使是使用了具有容错功能的磁盘阵列，也需要规划可靠的备份与恢复策略。使用容错存储只能减小由于单个磁盘驱动故障而导致需要进行数据库恢复的可能性。

## 4、使用RAID磁盘阵列技术

解决由于磁盘I/O而带来的性能瓶颈问题最有效的方法之一就是使用RAID磁盘阵列（Redundant Arrays of Independent Drives）。RAID为廉价（独立）磁盘冗余阵列的首字母简写，处于冗余和性能的考虑，它用于将多个磁盘组织成阵列。

RAID基础知识：

RAID磁盘阵列为磁盘的可靠性和性能提供了硬件的解决方案。RAID硬件分不同级别。下面这些级别和性能关系密切：

RAID-0

这个级别并不关注可用性，磁盘配置成RAID-0，就不具有冗余磁盘阵列。

RAID-1

这个级别提供最简单的磁盘冗余，即进行数据的全拷贝，又叫做镜像。

RAID-0+1

这个级别结合了RAID-1一对一的映射与RAID-0的磁盘列。

RAID-3

这个级别通过在阵列中的单个磁盘上保存奇偶信息来提供冗余特性。当其他磁盘出现故障时，这些奇偶信息可用于恢复其上的数据。相对于RAID-1而言，RAID-3更节省磁盘空间，不过，由于奇偶校验本身可能会成为瓶颈，因而RAID-3并不常使用。

RAID-5

和RAID-3类似，这个级别也使用奇偶校验数据来提供冗余特性；不过，正如实际的数据那样，这些奇偶信息被分散在所有的磁盘上，从而减轻了奇偶校验磁盘上的瓶颈问题。

还有一些其他级别的RAID，比如，RAID-6增加了双重奇偶校验数据，RAID-7和RAID-8在RAID-5特性的基础上加强了性能。

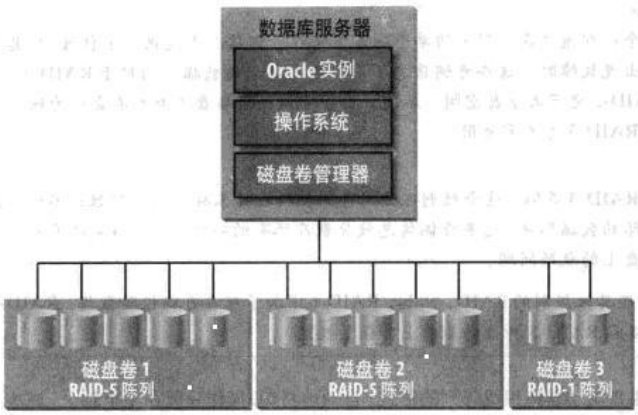
RAID将磁盘驱动器组织成阵列的形式，自动地将I/O操作分散在多个轴上，减少对单个驱动器的竞争。举个例子，假设你把一个包含索引的数据文件放在单个驱动器上。如果有多个进行同时使用这个索引，它们都会向同一个磁盘驱动器发出I/O请求，这个时候就会产生对同一个驱动器的竞争。

相反，假设你把这个数据文件存放在一个由五个物理磁盘组成的磁盘阵列中。阵列中的每个物理磁盘都可以独立地对索引中的不同数据块执行I/O操作。从而自动增加了Oracle可以执行的I/O数量而又不会导致竞争。

简单地使用磁盘阵列本身并不会将I/O性能提高到最佳状态。正如前面讨论的那样，你还需要从逻辑上将不同类型的Oracle文件放置在不同的驱动器中，即便是这些驱动器被组织成磁盘阵列。在Oracle 10g中，自动化存储管理提供自动条带化和重新平衡数据条带的功能，简化了磁盘数据条带化（striping）的过程。在默认情况下，ASM还提供自动镜像功能。

### 4.1磁盘卷管理器

利用基于主机的条带，逻辑卷管理软件可运行在数据库服务器上。旧的Oracle数据库常使用的这类软件包括惠普公司的逻辑卷管理（LVM）和Veritas软件公司的卷管理器，它们可作为请求I/O的操作系统和底层物理磁盘的接口。卷管理软件将磁盘组织成阵列，而操作系统将磁盘阵列看作是一个“磁盘”。实际的物理磁盘通常是连接到控制器的单个设备，它们构成了磁盘阵列。数据的条带化由卷管理软件来处理，对Oracle来说是完全透明的。如图显示了基于主机的卷管理机制。



Oracle 9i的发布开始为Linux和Windows提供自己的卷管理软件。从10g开始，各个操作系统版本的Oracle数据库都提供了集群文件系统和卷管理器，ASM将会充分利用这两个特性。不过当使用ASM的时候，建议你不要再使用操作系统的卷管理器。

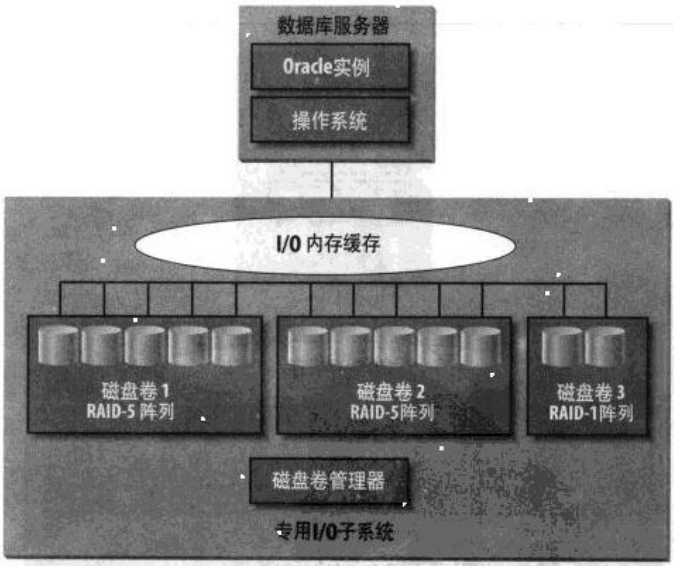
### 4.2专用存储子系统

专用存储子系统通常叫做磁盘场，它包括磁盘、控制器、CPU（通常）用作I/O缓存的内存。专用存储子系统的生成厂家包括EMC、Network Application、惠普、IBM以及Sun等公司。这个子系统将管理磁盘阵列的任务从数据库中提取出来。I/O子系统连接在控制器所在的服务器上。有时候，这些专用的存储设备可以被组织成一个独立的存储区域网络（SAN）。磁盘阵列在专用I/O子系统中定义和管理，而操作系统将磁盘阵列组成的逻辑“磁盘”看成物理磁盘。

这种类型的磁盘卷管理对数据库服务器来说是完全透明的，这样做的优点：

* 数据库服务器无需花费CPU资源管理磁盘阵列。
* I/O子系统使用内存来作为I/O缓存，因而Oracle I/O的性能得到了很大提升（例如，平均I/O事件从10~12毫秒降到3~5毫秒）。
* 只要数据写入子系统的缓存中，写I/O的操作就算完成。
* I/O子系统随后将数据从缓存分阶段写回实际的磁盘。
* 缓存可以满足读I/O，子系统将采用某些算法来理解I/O模式，并提前加载即将发生的读操作的缓存。

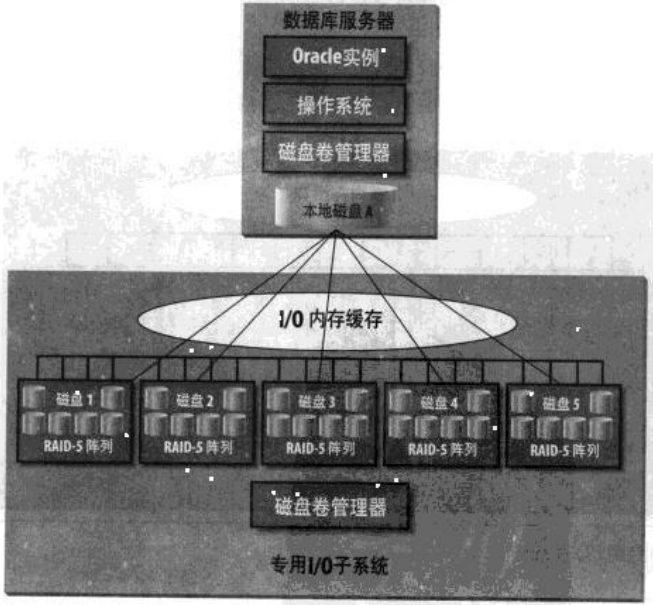
请注意，你必须用某些电池来支持缓存，这样，当电源发生故障时不会导致缓存的数据丢失，因为这些数据还写到物理磁盘中。否则，Oracle可能认为已发送到磁盘的数据丢失了，从而导致数据库崩溃的潜在可能性。如图显示了一个带有专用I/O子系统的数据库服务器。



### 4.3结合基于主机卷管理和I/O子系统卷管理这两种机制

在这种配置中，I/O子系统的磁盘被组织成阵列，并且使用操作系统的卷管理软件将磁盘重组为粗糙的阵列。例如，在EMC系统中，物理磁盘被组织为RAID-1镜像磁盘对或组织成RAID-S条带配置，该配置每个条带集使用四个磁盘。

将基于主机的条带机制与EMC I/O子系统结合起来：



## 5、灵活性、可管理性以及磁盘阵列

目前，许多系统都使用了有关技术把多个分离的磁盘驱动器组合成一个磁盘阵列，每个磁盘阵列就如同编排好I/O的单个逻辑磁盘。通过条带机制可以在多个磁盘上扩展I/O，而不会因为处理许多独立的磁盘驱动器而带来规划和管理的开销。

在一个阵列中到底应该使用多少个磁盘经常会引起争议。就一个极端情况而言，使用多个磁盘而不讲它们组织成磁盘阵列，可以提供最多的控制和灵活性，因为每个磁盘都是可见的，可以将某些文件放在各个磁盘上。然而，这种方法要求事先做更多的规划，而且需要更多的管理工作，因为你必须处理每个独立的磁盘驱动器。随着数据库规模的不断庞大，这种方法将会变得无法管理。

就一个极端而言，可以将所有的磁盘组织成一个阵列，让操作系统和Oracle看起来好像只有一个“磁盘”。这种方法只需要简单的规划和管理工作。由于被看作是一个磁盘，你不需要思考该如何存放各种数据库文件。然而，这种方法以牺牲灵活性为代价，并导致必须对I/O瓶颈给出最强有力的解决方案。如果流经阵列的I/O性能不稳定，那么在这个解决方案中就要加入更多的控制器和磁盘。整套磁盘就犹如一个黑匣子，要么像一个单元部件那样工作，要么就根本不工作。显然，这种方法对于大多数系统来说是不可行的。

大多数实用的配置结构都是在灵活性和可管理性之间做一个平衡。例如，有一个包含1000个磁盘的系统。由这1000个磁盘组成一个阵列是不可能的，而让这1000个磁盘都相互独立的工作也是不可能的。可以让每个20个磁盘组成一个阵列，这样得到的50个阵列将会在不增加负担的前提下提供所需的I/O高性能指标。如果系统不需要那么高的灵活性，那么可以让每50个磁盘组成一个阵列，这样的系统结构或许更可取。另一方面，在一个仅有5个磁盘的系统中，把所有的磁盘组织成一个阵列可能是一种罪简单的管理方法。所有说，在这个问题上，没有唯一的答案，你必须评估自己的需求去找一个合适的平衡策略。

Oracle 10g能够自动完成条带化以及条带集重新平衡的流程，从而简化了相应的管理工作。ASM将文件分成1MB大小数据区间，并将这些数据区间均匀地分散在每个磁盘组上。分别用一个指针来跟踪每个数据库范围的位置（而不是使用诸如散列函数这样的数学函数来对数据进行条带化）。因而当磁盘组配置发生变化，各个数据区间可以被移动。与传统的基于算法的条带技术而言，目前的技术不需要重新运行算法，不需要为所有的数据重新分配空间。磁盘配置发生改变之后重新平衡负荷的时候，打开新的数据库文件的时候，或者增大表空间以扩展数据库文件的时候，都需要对数据库范围映射进行修改。在默认情况下，每个1MB大小的数据区间也会被镜像，因而冗余性的管理也得到简化。镜像可以扩展到三重镜像，也可以被关闭。虽然管理人员需要理解如何将多个磁盘组织起来，但是这些磁盘组的条带化和冗余性的实施都由ASM自动完成。

在Oracle 11g最初的发布之后，增加了新的存储管理特性。这些特性在为Oracle的Information Appliances配置存储的时候特别有用。

## 6、Oracle I/O和条带阵列如何相互作用

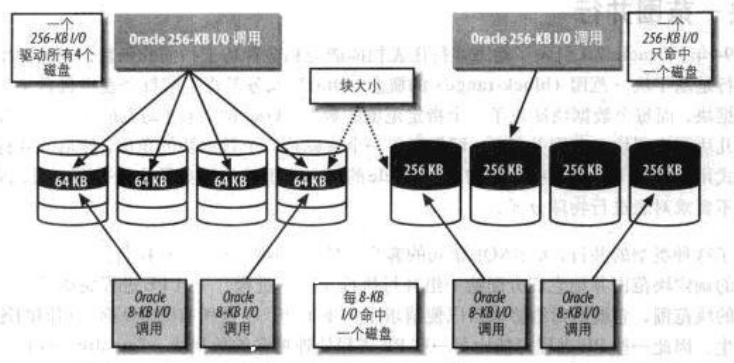
在几乎所有的大型数据库中，对磁盘进行条带化可以增加磁盘I/O的速率，而不会因为要管理分布在多个单独磁盘上的数据文件而增加负担。正如前面讨论的那样，可以利用应用数据库服务器上的卷管理器，专用I/O子系统或者是两者的结合将磁盘组织成磁盘阵列。

如果使用的Oracle发布没有提供ASM，那么你在建立调点磁盘阵列的时候，可以设置用于给磁盘加条带的块大小。块大小就是数据迁移到阵列中的另一个磁盘之前写入当前磁盘的数据量。理解不同条带的块大小与两种大小的Oracle I/O之间的相互作用，对于将I/O性能最大化非常关键。

假设一个Oracle数据库大小为8KB，并且DB\_FILE\_MULTIBLOCK\_READ\_COUNT初始化参数设为32。那么Oracle将支持两种大小的I/O：一个单独的8KB数据块和一个256KB的多块读（8KB的32倍）。然后假设为Oracle配置了一个包括4个磁盘的阵列，而块大小为64KB，那么256KB的数据库将会分散到4个磁盘中，则每个磁盘64KB数据。

每个8KB的I/O将命中一个轴，而这8KN的数据将会置于一个64KB的块中。由于并发的最大化，建立条带可以提高小型I/O的性能，因为每个磁盘可以服务于不同的I/O。256KB的多块I/O将命中所有4个磁盘。如果把块大小从64KB换成256KB，那么平均每个256KB的I/O请求只命中一个磁盘。在这种情况下，多块I/O就只需要更少的I/O调用，因为磁盘上的块比较大。无论是哪一种情况，磁盘都将满足单数据库I/O调用。通过把一个I/O调用分散到多个磁盘上建立条带，可以提高大规模读操作的I/O速率，在上面的64KB块大小和256KB多块I/O的例子已经说明了这一点。

如图显示了不同大小Oracle I/O与使用不同块大小进行条带化的阵列之间的相互作用：



## 7、Oracle与并行性

并行化操作能力是巨型数据库（Very Large Database，VLDB）最重要的特性之一。带有多个CPU的数据库服务器，也叫做SMP，目前是大多数数据库服务器的标准配置。当性能需求以及数据量不断增长，更加需要使用多处理器和多磁盘来减少完成给定任务所需的时间。通过运行实时应用集群，Oracle既可支持但SMP服务器中的并行，还可以支持多个节点间的并行。以并行方式执行一个SQL语句将消耗更多的机器资源——CPU、内存和磁盘I/O——但是可以缩短整个任务的完成时间。

并行操作以一种比较线性的方式影响着执行一个给定任务所需要的内存数量和CPU资源。每个并行执行进程都有一个程序全局区（Program Global Area，PGA），它需要占用一定的内存来完成工作。每个并行执行进程会占用自己的CPU时间片，但是很多并行进程都可以减少磁盘I/O的时间，磁盘I/O往往是瓶颈最容易产生的地方。

Oracle数据库中存在两种可能的并行类型：

块——范围并行：

由数据库块的范围驱动。

基于分区的并行：

由操作中设计的一定数量的分区或子分区驱动。

### 7.1块——范围并行

1994年，Oracle 7.1引入了动态并行化表扫描能力和各种基于扫描的函数。这种类型的并行是基于块——范围（block-range）的概念，Oracle服务器将了解每个表中包含的一组数据块，而每个数据块涵盖了一个指定范围的数据。Oracle 7通过动态地把一个表分解成几块来实现块——范围并行，分解得到每一个块就是一个数据块的范围，然后以并行的方式用多进程来处理这些分解的块。Oracle的块——范围并行的实现是独一无二的，因为它不要求对表进行物理分区。

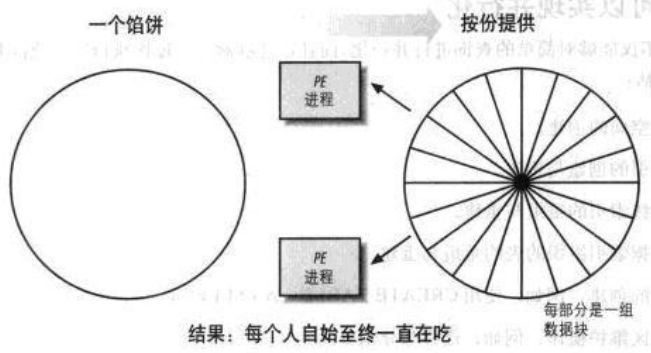
有了这种类型的并行，发出SQL语句的客户会话将透明地成为并行执行协调器，它将动态的确定范围并把它们分配给一组并行执行（PE）进程。一旦PE进程完成了一个分配的块范围，它就返回给协调器以便请求下一步工作。不是所有的I/O都以相同的速率发生，因此一些PE进程可能比另一些PE进程处理更多的数据块。“stealing work”的概念允许所有的进程充分参与各项任务，从而最大限度地利用主机资源。

如果有足够的硬件资源，那么块——范围并行的规模与PE进程的数量成线性关系。实现并行的伸缩性关键在于硬件基础。每个PE进程均运行在一个CPU上，并且只对一个设备请求I/O。如果有足够的CPU去读取足够的磁盘，并行将成比例扩展。如果系统在这些资源中的某一点遇到一个瓶颈，伸缩性则会收到影响。例如，有4个CPU，但是只读取2个磁盘，那么还会带来并行能力的下降。同样的，2个CPU读取20个磁盘也不会成比例地将性能提高20倍。系统硬件必须在并行的同时依照比例关系进行平衡。

大多数的大型系统中，磁盘的数量要远远大于CPU的数量。在这些系统中，并行导致了I/O和I/O子系统的随机化。这对于不同用户的PE进程在不同时间读取不同磁盘的并行数据方式是很有用的，因为它使得I/O分布在多个可用的磁盘上。

打个比方，动态的并行性犹如吃馅饼。这个馅饼就像是该操作要读取的一组数据块，并且目的是尽可能快地让一定数量的人吃掉这个馅饼。Oracle按份来提供这个馅饼，当其中一个人吃完他的第一份时，他可以再回去要更多份。并不是每个人都以相同的速度吃馅饼，因此某些人将会比其他一些人吃得多。

如图显示了将一组数据块分隔成几个数据区间：



### 7.2表与表的分区之间的并行

Oracle 8中引入了分区表，一个操作可能会涉及一个分区表中的一个、多个或所有的分区。对于一个常规的表而言，块——范围并行机制将动态地分割一组数据以便读取，这一点与分区表并没有本质的区别。一旦优化器确定了操作该访问哪一个分区，所有涉及到的分区的数据块则被看作是一个分隔成多个数据区间的池。

优化器的这种假设导致了使用并行和分区表的一个关键问题。并行度（即用于整个表的并行执行进程的数目）需要应用到某一项操作所用的一组分区上。如果分区中没有包含操作要用的数据，那么优化器就会排除这一分区。例如，如果一个表的某个分区包含的ID号在1000以下，并且要查询请求ID号在1100到5000之间，那么优化器就会知道这项查询不可能访问到这个分区。

Oracle 9i开始，你也可以基于一组特定的值来对表进行分区，不过这种类型的分区通常适合于维护操作的分区表。

如果你期望你的查询使用分区排除或修剪而且计划采用并行性，那么应该在足够多的驱动器上条带化每个分区从而使之能够有效扩展。这种做法不论访问分区的数目大小都可以确保可扩展性。条带化的工作可以通过在多个磁盘上保存多个数据文件的方法手工完成。这些磁盘能够以条带化的形式组成阵列，也可以结合这两种方法组成阵列。

### 7.3什么可以实现并行化

Oracle不仅能够对简单的查询进行并行化，而且还能够对许多操作执行块——范围并行化，具体包括：

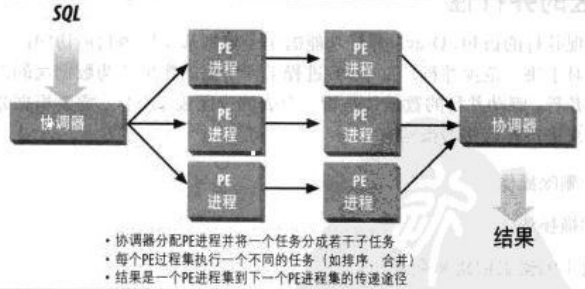
* 表空间的创建。
* 索引的创建与重建。
* 在线索引的重组与重建。
* 根据索引组织的表的重组与重建。
* 表的创建，例如，使用CREATE TABLE…AS SELECT。
* 分区维护操作，例如，迁移与分割分区。
* 数据加载。
* 完整性约束的实施。
* 统计的收集（Oracle 10g开始可自动收集）。
* 备份与恢复（Oracle 11g中包含大型文件）。
* DML操作（INSERT、UPDATE、DELETE）。
* 查询处理操作。
* OLAP集合（就Oracle 10g而言）。

Oracle还能为各个查询处理步骤提供并行化的优势。可能实现并行化的查询处理特性包括：

* 表扫描。
* 嵌套循环。
* 排序合并关联。
* 散列关联。
* 位图星型关联。
* 索引扫描。
* 面向分区的关联。
* 反关联（NOT IN）。
* SELECT DISTINCT。
* UNION和UNION ALL。
* ORDER BY。
* GROUP BY。
* 聚集。
* 导入。
* 用户自定义函数。

1）并行度

Oracle实例有一个对于数据库用户可用的PE进程池。这些进程会消耗CPU、内存和I/O资源。控制活跃PE进程的最大数量是很重要的，因为太多的PE进程将会给主机带来过重的负担，导致出现资源瓶颈和性能的退化。高度的并行性还会导致全表扫描，这种情况可能好，也可能坏。如图显示了PE进程组内和组间的透明并行性。



在有多个用户并且工作负荷不断变化的情况下，确定最佳的并发度具有一定的挑战性。例如，当查询并发度为8时，就能满足1~2个用户查询操作对性能的需求，但是如果有20个用户查询同一个表，又会怎样？这个时候可能需要160个PE进程（8个PE进程服务1个用户），这么多的PE进程可能会导致及其超载。

将并行度设置为最小公分母（例如2），在用户较多的时候，能够提供有效的并行性，但是当用户较少的时候，并不能充分利用资源。

2）自调整适应性并行

当系统负荷增加时，这个特性将自动地按比例减小并行度；当系统负荷减少时，则会自动地成比例增大并行度。当某个操作要求具有一定的并行度时，Oracle会检查系统的负载并降低实际操作并行度以避免系统负载过重。要求并行操作的用户越多，他们接受到的并行度就会越来越低，直到操作连续地执行完毕。如果操作减少，随后的操作将赋予不断增大的并行度。这种适应性可以把DBA从本来不可能做到的任务中解放出来，这个不可能的任务就是在面对并发性和工作负载不断变化的条件下确定最佳的并行度。

在确定赋予一项操作的并行度的过程中，自适应并行要考虑两个因素：

* 系统负荷。
* 在数据库资源管理器活跃的条件下，用户客户群体的并行资源限制。这一点非常重要，因为这意味着如果存在适应性并行，就还需要考虑资源的规划。

### 7.4基于分区的并行性

对于需要实现并行的语句，Oracle并行功能的小子集则基于该语句所访问的分区或子分区的数据。对于块——范围并行，每个PE进程工作所需的数据片为数据块的范围；对于基于分区的并行，驱动并行的数据片是指一个表的分区或子分区。需要根据分区或者子分区的数目来进行并行化的操作有：

* 更新和删除操作。
* 索引扫描操作。
* 分区表上的索引创建和重建。

1）表分区为和子分区的并行

Oracle 8引入了对并行DML（数据操作语言）的支持，即支持并行的执行INSERT，UPDATE和DELETE语句。这种类型的并行改进大量数据操作（例如更新一个大型表中所有行的操作）的性能。

在Oracle 8中更新和删除操作的并行度与涉及到的分区数量是密切相关的；而对于Oracle 8i及以后的版本，更新和删除操作的并行度则与涉及到的分区或子分区的数量有关。一个由12个分区的表（例如，每个分区对应一年中的每个月）对于并行更新和删除操作可以用于最大为12的并行度。仅对一个月的数据进行更新操作不会并行，因为它只涉及到一个分区。如果通过使用Oracle的复合分区（例如，每个月分区中根据PRODUCT\_ID再分为4个散列子分区）创建一个表，对于整个表最大的并行度将是48，即有12个分区，每个分区又有4个子分区。因此对于一个月数据的更新操作并行度可以为4，因为每个月包含4个散列子分区。如果这个表没有被分区，Oracle就不能以并行的方式执行更新操作。

在Oracle 8及之后，可以使用类似于并行DML的语义，即每个索引分区或子分区对应一个PE进程，同时以并行的方式对分区的索引执行创建、重建和扫描操作。

2）对未分区的表进行快速完全索引扫描

人们常常会认为只有对目标索引进行分区之后Oracle数据库才能并行处理索引扫描。Oracle 7.3引入了在某些场合下对未分区的表进行快速完全索引扫描的功能。如果索引扫描操作是“不受约束的”，即对这个索引的访问是为了满足查询的需求，那么Oracle7.3及更高的版本将使用块——范围并行机制，从而并行化对整个索引的访问。因此，当Oracle可以对未分区的表进行快速完全索引扫描时，这个特性适用于范围有限的查询。基于分区的索引扫描则适用于更大范围的查询操作。

3）未分区表和分区表的并行插入操作

Oracle可以通过INSERT INTO tableX SELECT…FROM tableY这样的语句以并行的方式对未分区表和分区表执行一条插入语句。Oracle用一组PE进程对插入语句中SELECT部分执行块——范围并行化。这些PE进程把多行数据传递到第二组PE进程，第二组OE进程再把这些数据插入到目标表中。目标表可以是未分区表。也可以是分区表。因此被插入操作的并行不是严格意义下的块——范围并行或基于分区的并行。

## 8、Oracle与内存资源

访问内存中的信息比访问磁盘中的信息速度要快得多。Oracle实例可以利用数据库服务器的内存资源对信息进行高速缓存从而提供性能。Oracle使用了一个叫做SGA（系统全局区）的共享内存区，而每个服务器进程使用一个叫做PGA（程序全局区）的私有内存区。

在Oracle 9i之前，只能初始化文件中指定SGA或其任意组件数组库高速缓存共享池或大型池的大小，而且在实例一直运行的情况下，这些内存大小的分配是不能改变的。在Oracle 9i中，可以基于叫做“粒度”的最小内存分配来动态调整这些池的大小。Oracle 10g及之后都能够自动管理共享内存。并且Oracle 11g增加了对SGA和PGA的自动内存管理功能。

数据库中所有的内存都耗尽将会导致性能下降。如果你正在使用旧版本的Oracle，那么你应该观测一下Oracle使用的各种内存区的大小，或许你需要增大机器的内存以防因内存不足而导致问题。在设置各个内存区大小的时候，需要综合考虑你的应用程序的运行方式、所使用的数据以及你对性能的需求。

## 9、Oracle如何使用系统全局区——SGA

Oracle使用SGA完成以下操作：

* 对包含表和索引数据的数据块进行缓存，并放在数据库高速缓存中。
* 对被解析和优化的SQL语句、存储过程以及数据词典信息进行缓存，并放在共享内存池中。
* 日志信息写入磁盘之前，把它们先缓存在日志缓冲器中。

在Oracle 9i之前，SGA中每个内存区预分配内存的大小可以在启动过程中使用初始化参数来设置，运行过程中时无法修改并动态生效的。

调优工作主要集中数据库告诉缓存和共享池。

### 9.1SGA的大小自动调整

Oracle 10g提供了SGA内存大小自动调整功能，不需要管理员手工调整SGA内存。有了自动化的内存共享管理特性之后，数据库可自动分配下面的SGA内存池：数据库高速缓存、共享内存池、大型内存池、Java内存池以及流内存池。管理员仅需通过设置SGA\_TARGET初始化参数来指定所需要内存的总数。

由于10g开始，Oracle数据库可以动态监控每个内存池的内存需求，并在适当的时候，动态增加内存。有了自动化的SGA内存大小调整特性之后，数据库管理员仍可以使用下面的初始化参数来指定SGA中各个池的最小内存数量：DB\_CACHE\_SIZE、SHARED\_POOL\_SIZE、LARGE\_POOL\_SIZE、JAVA\_POOL\_SIZE以及STREAMS\_POOL\_SIZE。SGA中还有几个池需要使用LOG\_BUFFER、DB\_KEEP\_CACHE\_SIZE以及DB\_RECYCLE\_SIZE等参数手工设置大小。

### 9.2数据库高速缓存

如果你决定将SGA\_TARGET设置为0，那么需要手工设置各个内存池的初始化参数（除非你期望使用以前的大小）。对于数据库高速缓存而言，你应该评估一下用户从缓存中请求到数据块与磁盘中请求到数据块的百分比。它又叫做命中率。如果响应时间太长而且命中率低于90%（作为经验法则），增加初始化参数DB\_CACHE\_SIZE的值可以提高性能。

通常认为持续增加数据库高速缓存的大小可以不断提高数据库的性能。然而，这种想法仅当高速缓存中的块真正被重复利用才是对的。大多数OLTP系统仅有被大量使用的核心表较少的一部分（例如，在表中查找有效编码）。I/O的其他方面往往比较随机，事务的过程中可能访问各种数据库一行或者两行数据。因而，在数据块没有被大量重新使用的场合，更大的高速缓存并不能提高性能。

另外，并不是所有的操作都是对数据库高速缓存进行读操作。例如，大型的全表扫描往往被限制只能使用较少的缓存，以免由于占用过多的缓存而影响其他用户。如果应用程序需要执行大量的表扫描工作，增加高速缓存也许并不能提高性能，因为这个时候高速缓存往往不包含所需的数据块。并行表扫描会完全绕过高速缓存而直接将数据行交给用户进行。由于性能问题错综复杂，因而深入理解应用程序实际使用数据的方法将有助于调整数据库的高速缓存。

### 9.3共享内存池

在Oracle数据库每个操作执行的过程中，会多次用到共享内存池。例如，利用共享内存池来存储发送给数据库的SQL语句以及执行SQL语句所需的数据字典信息。由于共享内存池处于数据库操作的核心位置，因而，相对于数据库高速缓存来说，共享内存池的容量大小对性能的影响更大。如果被请求的数据块不在数据库高速缓存中，那么Oracle将会通过执行一次I/O操作来获取数据，这就会导致一次性能命中。

由于各种原因，共享内存池太小会导致性能变糟，从而影响所有的用户。这些原因包括：

* 无法存储足够多的数据字典信息，频繁地查询和修改数据字典将导致许多额外的磁盘操作。
* 无法存储足够多的的SQL语句，会导致内存频繁的置换，即将有用的语句导出以便给出新的语句腾出空间。一个设计良好的应用程序会重复发出相同的SQL语句。如果没有足够的控件来存储应用程序所用的全部SQL语句，那么将会对相同的语句进行分析、存储，不断的换出换进，导致CPU资源被浪费，也增加了每个事务的开销。
* 无法存储足够多的存储过程，这将会导致在数据库中存储和执行的程序逻辑产生同样的内存置换以及性能问题。

如果你正在手动管理共享内存池，而且已经诊断出问题所在，解决方法则比较简单；使用初始化参数SHARED\_POOL\_SIZE来增加共享内存池的大小。在大型活跃的数据库中，共享内存池的大小通常被设置为150~250MB。可以通过检查共享内存的活动情况来查找问题。

### 9.4redo日志缓存

尽管与数据库高速缓存以及内存池相比，redo日志缓存在SGA中只占很小的一部分，但是它对于性能却非常重要。在事务执行过程中，如果对数据进行了修改，那么就会在日志缓存中写日志信息。在一个事务提交之后，或者当日志缓存被写满三分之一的时候，日志缓存中的内容将被写入磁盘日志文件中。而当日志缓存的内容被写到磁盘的时候，Oracle会确保正在写磁盘的缓存内容不能被改变，直到日志信息已经安全地被存放在磁盘上为止。事务可以继续在剩余的日志缓存中写日志信息。在一个繁忙的数据库中，写磁盘操作没有结束之前，事务可能会产生足够的日志，将剩余的日志缓存写满。如果发生了这种情况，事务必须等待写磁盘的I/O完成，因为此时redo日志缓存已经没有可用的空间了。这就会对性能产生影响。可以用个“redo缓存分配尝试”这项统计来了解情况。通过动态性能视图V$SYSSTAT中的统计信息可以了解等待分配redo日志缓存的频率。可以使用下面这个例子来查询V$SYSSTAT的统计信息：

SELECT name,value FROM V$SYSSTAT

WHERE name=’redo buffer allocation retries’;

可以持续观察这些统计信息一段时间来了解当前的趋势。在某个时刻的值反映了自从实例启动以来的累计值，而作为单个时间点来考虑是没有意义的。请注意，对于所有用于性能调优的统计来说，这一点都是正确的。在理想情况下，redo缓存分配尝试这项统计应该接近于0。如果你监控到这条统计在增加，那么就应该增加redo日志缓存的大小。通过LOG\_BUFFER初始化参数来设置。

### 9.5查询结果缓存

Oracle 11g中最重要的性能特点之一就是用来帮助提高重复查询的性能。Oracle通过缓存数据库块和索引块避免了资源密集型的磁盘读操作；通过缓存SQL语句省去了对查询进行重复解析和优化的过程。但是在Oracle 11g之前，被缓存的SQL语句仍然需要被重复执行以返回结果集。

这个特点允许Oracle 11g将完整的结果集缓存在共享内存池中。有了这个特点，对于重复执行的查询可以直接返回之前缓存的结果集。由于结果集对于重复的查询来说总是相同的，因而缓存这样的查询结果集对于诸如网页服务等应用环境的性能影响非常大，因为在网页服务中相同的网页会被重复请求。这个特点还可以用于对PL/SQL函数的结果集进行缓存。

Oracle 11g还提供了在客户端上缓存查询结果集的功能，并且能够自动保证被缓存的结果集能够随着变化而得到及时地更新。这个特点不仅能够通过缓存结果集获得性能的提高，而且省去了不必要的网络流量。

## 10、Oracle如何使用程序全局区——PGA

每个服务器都有一个程序全局区（Program Global Area，PGA），它是一块包含正在支持的服务器进程工作信息的私有内存。每个服务器进程都有一个PGA。所有PGA的内存总量是所有活动的服务器进程数量的一个函数。这些进程是Oracle实例的一部分。用户的数量越多，服务器进程数量就越多，PGA占用的内存空间也就越大。使用多线程服务器可以减少服务器进程的数量，从而减少PGA占用的内存大小。

PGA包括用于存储服务器进程临时变量等内容的工作内存区，存储服务器进程正在执行的SQL信息的内存，以及作为SQL执行一部分的排序记录所占用的内存。用于保存变量的PGA工作区最初的大小叫做栈空间，由于它是基于数据库服务器正在使用的操作系统预先确定的，因此不能直接控制。而PGA中的其他部分则都能控制。

### 10.1用于SQL语句的内存

当服务器进程为某个用户执行SQL语句时，它可以跟踪相关SQL语句特定会话的信息以及某些过程的执行情况，这些信息存储在PGA中的私有SQL区中，这些过程又叫做游标（cursor）。请不要将它与共享内存池中的共享SQL区混淆。共享SQL区包含SQL语句相关的可共享的详细信息，例如，优化规划等。

私有SQL区包含会话中SQL语句执行的特定信息，例如，到目前为止获取的行数等信息。一旦SQL语句被处理，它的私有SQL区可以被其它SQL语句重用。如果应用程序对那些私有SQL区已经被重用的SQL语句重新发起执行请求，那么它的私有SQL区必须重新初始化。

每次收到新的SQL语句，必须在共享内存池对它的共享SQL区进行定位（或者，如果定位失败，必须重新加载）。类似地，必须在PGA中定位SQL语句的私有SQL区，如果定位失败，服务器进程必须对之进行初始化。重新初始化的过程是比较昂贵的，会消耗不少的CPU资源。

具有PGA服务器进程可以包含较多的私有SQL区，这样对于新进入的SQL语句来说，将会少花一些时间进行私有SQL区的初始化过程。如果服务器进程不必对新的语句使用现有的私有SQL区，那么原来语句的私有SQL区应该会被完整地保存。虽然与更大的共享内存池的功能相似，但是一个更大的PGA可以避免私有SQL区内存置换。减少私有SQL区的重复被利用，相应地就会减少相关的CPU开销，因而可以提高性能。当然，在分配PAG内存和整体性能之间还需要一个折中的考虑。

OLTP系统通常会有一个用户提交的SQL语句工作集。例如，某个用户在一个应用中重复使用相同的表单预定出租汽车。如果用户服务器进程有足够的空间来存储哪些表单发出的SQL语句，那么性能将会得到很大改善。应用开发人员写SQL语句时需注意，应该将SQL语句写得易于重用。这可以通过在SQL语句中指定绑定变量来实现，而不需要硬编码。

### 10.2PGA中用于排序的内存

每个服务器进程在把排序记录返回给用户之前，都会使用PGA的内存来存放它们。如果分配给用于记录排序的内存不够，那么服务器进程将进行多轮（称作run）排序。中间的轮次写到用户的临时表空间中，由于涉及到磁盘I/O，因而会降低排序的性能。

在Oracle 10g之前，确定PGA的排序区大小是调优的一个关键点。排序区太小无法满足普通的排序需求，会导致临时的表空间磁盘I/O，从而影响性能。而排序区过大，则会导致内存浪费。

Oracle 10g提供了自动化的PGA大小调整功能。在默认情况下，这个内存管理功能是开启的，PGA调整的工作区是以SGA内存大小的20%为基础。有了这个功能，数据库管理员就不再需要调整PGA各个部分的内存大小。

Oracle 11g在SGA和PGA中引入了自动内存管理。通过简单的设置一个MEMORY\_TARGET初始化参数，PGA和SGA将会自动设置适当的初始值。

## 11、TimesTen

2005年，Oracle收购了业界领先的内存数据库TimesTen。内存数据库通过减少数据库获取操作的时延而提供最佳性能。TimesTen数据库中的优化器掌握了数据在内存中的位置信息，并能够创建执行计划来充分利用这个位置信息。另外，由于不需要再编码处理数据在磁盘中的各种情况，内存数据库中机器指令的实际数量在很大程度上被减少。高负荷的OLTP环境要求有极高的吞吐量和实时响应，TimesTen非常适合于这种环境。

TimesTen实例可以用作Oracle数据库的一个缓存。可以将部分Oracle表导入TimesTen数据库实例中，它们之间的数据同步可以有Cache Connect to Oracle这个特性负责。

你还可以打开在不同机器上多个TimesTen实例之间的复制功能，以满足负载均衡和高可用性的需求。

## 12、Oracle与CPU资源

Oracle数据库与运行在同一台服务器上的其他所有软件共享CPU资源。如果CPU处理能力不足，减少Oracle或者非Oracle程序的CPU消耗可以改善服务器上所有进程的性能。

如果机器上所有CPU都繁忙，这些进程将排队等待使用CPU。由于进程需要等待CPU资源，因而这些进程所形成的队列又叫做运行队列。CPU越忙，进程在运行队列中等待的时间就越长。处在队列中的进程并不做任何工作，运行队列变长，响应时间就会下降。

对CPU的利用率进行调优实质上就是对每个单独的任务进行调优：它将减少完成所需要的命令数量，并/或者减少要执行的认为数量。通过负载均衡、SQL调优以及改进应用程序的设计等方式来具体完成CPU调优工作。从事这样的调优工作需要深入理解这些任务的目的以及它们被执行的方式。

一些常见的任务经常会导致CPU被过度使用。如果在数据库中遇到CPU资源缺乏的情况，可能需要考虑以下因素：

糟糕的SQL语句：

过多的解析：

数据库的工作负荷：

非数据库工作负荷：

## 13、数据库资源管理器

前面讲述了由于CPU资源短缺而导致性能降低的各种情况。数据库资源管理（DRM）可以帮助管理人员自动避免这样的一些问题。

在使用DRM的时候，你需要指定消费者组，并对该消费者组可以使用的计算机资源进行限制。这样就可以确保每个组或者组成员不会使用过多的系统资源，同时也可以确保为不同的用户组提供有保证的服务。可以创建DRM层次结构来为组内的小组指定资源数量。

下面的DRM特性可以结合起来使用，以解决低性能问题：

预测资源使用情况：

切换消费者组：

限制连接的数量：

在Oracle 11g中，数据库安全了默认的DRM计划。该计划将会限制优化器统计收集、自动化分段顾问以及自动SQL调优顾问等自动化维护任务占用的资源数量。