第 1 章 Oracle性能调优：一种系统化方法

Oracle性能调优已经出现好多年了，但多数时候仍然以一种杂乱无章或者低效的方式在进行着。先看看下面这个警示故事。

有个关键的业务系统出现了性能问题。作为一名富有经验的Oracle性能专家，你被请来诊断这个问题。你首先检查了数据库的等待时间，以了解数据库大部分时间都在做什么。稍后我们将看到，通过查询视图`v$system_event`与`v$sys_time_model`，可以轻松得到这些信息。

通过查看这些视图，你发现了两个问题。第一，读取磁盘设备花费了最多的数据库时间。第二，从磁盘读取数据块耗费的平均时间大大超过你对硬盘的预期。

你猜，可能是磁盘阵列没有足够的IO带宽来满足应用需要。也就是说，磁盘阵列的物理磁盘数目不足以支持所需的IO速度。经过快速的计算，你建议将磁盘数量扩容四倍。除了耗费不菲的美金，以及将数据分配到新磁盘所需的宕机时间[①]。尽管如此，事情仍然得做，因此管理层认可了这笔花销以及相对应的宕机时间。扩容完成之后，用户反馈说他们对性能很满意，你也谦虚地接受了所有的赞扬。

升级很成功？你认为是，但……

- 没过几个月，性能问题又出现了，磁盘IO仍然是罪魁祸首。
- 公司请来了另一位Oracle性能专家，他认为修改一下索引就可以解决原来的问题，一分钱不用花也不用系统宕机。
- 新索引创建好之后，IO请求降低到你最初被请来时观察到的1/10。管理层准备在eBay上卖掉富余的磁盘设备，并且给你的咨询记录打上“不再续聘”的标签。
- 你的恋人离开了你，嫁给了一名Oracle销售人员。最后，你剃度做了和尚。

经过几年的苦思冥想，你终于明白，自己已经正确地找出了数据库里最耗费时间的活动，然而却没有分清什么是因什么是果。所以，你错误地去处理这个结果——高的IO请求率，而忽略了真正的原因——索引的缺失。

---

① 若采用某些技术，宕机时间可以避免，但低性能状态将持续一个时期。

本章将介绍一种方法，确保你将精力集中在导致Oracle性能问题的根源上。这样可以避免重复地试错（大量性能调优工作都是在试错），并确保在调优过程中能获得最大的性能提升。

1.1 Oracle 性能调优简史

在20世纪90年代早期，Oracle数据库调优并不像今天这么有章法。事实上，那时的性能调优仅限于几条知名的“经验准则”。

这些准则中最泛滥的当属“应该优化缓冲区高速缓存命中率”（Buffer Cache Hit Ratio）。这个命中率是指在内存中找到SQL语句请求的数据块（相匹配的数据块）的比例。如果请求了10个数据块，其中9个可以从内存中找到，那么命中率就是90%。最常见的建议标准是，提高缓冲区高速缓存的大小，直到命中率达到90%或者95%。其他一些比例，如闩锁（Latch）命中率，也有类似的建议标准值。

“基于命中率”的技术虽然反映了Oracle内部的效率问题，但这些命中率与使用数据库的应用的性能问题关系并不密切。例如，虽然在内存中找到对应的数据块明显更好（更高的命中率），但是SQL语句反复低效地访问同一个数据块通常也会带来更高的命中率。实际上，非常高的命中率常常是SQL调优水平太差的一个征兆。

Oracle版本7.1中出现的等待事件接口提供了另一种性能调优的方法。这些等待信息显示出Oracle会话等待几个不同事件所花费的时间，如等待锁变得可用或者磁盘IO完成。通过集中分析在总等待时间中占比最大的等待事件，可以更有效地完成优化。

系统化Oracle性能调优的先驱们，包括Anjo Kolk——著名的另一种性能剖析方法”（*YAPP，Yet Another Performance Profiling*）[①]方法论的作者，都力挺这一技术。

基于等待事件的调优技术从出现到变成主流，经过了非常长的时间：从等待事件接口的最初发布，到这项技术被广泛接受，差不多用了5~10年。虽然现在几乎所有Oracle专业人员都很熟悉基于等待事件的优化方法。

1.2 超越表面分析法

从基于命中率的优化方法转向基于等待事件的优化方法，显著地提升了我们诊断和优化Oracle应用的能力。然而正如前面提到的，一味关注响应时间长的组件，可能导致下面几个不良后果。

---

① 此文档的下载地址为http://www.oraperf.com/download/yapp_anjo_kolk.pdf。——译者注

- 治标不治本。
- 更倾向于选择硬件方案，而非更划算的调整配置或应用。
- 只顾眼前，忽视长远及可扩展能力。

为了规避基于等待事件调优的隐患，需要明确几个阶段。这几个阶段是由应用、数据库、操作系统的交互方式决定。总体来看，数据库进程按照下面几个层次运行。

(1) 应用以SQL语句的形式向数据库发出请求（包含PL/SQL请求）。数据库响应这些请求，并返回对应的返回码与结果集。

(2) 为了处理应用请求，数据库必须解析这条SQL语句。在最终执行它之前，还需要完成多个额外操作（安全、调度及事务管理）。这些操作要用到操作系统资源（CPU和内存），可能会受并发执行的数据库会话之间争用的限制。

(3) 最后，这些数据库请求还需要处理（创建、阅读或变更）数据库中的部分数据。需要处理的确切数据量要视数据库设计（例如索引设计）和应用（例如SQL语句类别）而定。

内存中会有部分被请求数据。这些数据块出现在内存中的概率主要由其中的数据被访问的频率以及可用来缓存数据块的内存总量决定。在内存中访问数据库数据的行为被称为逻辑IO。此外，执行排序和散列操作也需要使用内存。

(4) 如果数据块不在内存中，就必须到磁盘上读取，从而导致物理IO操作。物理IO是到目前为止所有操作中代价最高的，因此数据库会尽量避免不必要的IO操作。然而，有些磁盘读写是无法避免的。在排序和散列操作太大而不能在内存中完成时，也会发生磁盘IO。

以上每一层的活动都会影响对下一层的请求。例如，提交一条SQL语句由于某种原因不能有效利用索引，就需要执行大量的逻辑读操作，而这又将导致争用并最终引发大量的物理IO操作。看到如此大量的IO操作或争用时，你常常会忍不住用优化磁盘布局的方法直接清除这个症状。不过，如果按照层级的顺序来安排调优工作，更有可能抓住问题的根本，并在更低层级上化解性能问题。

下面是对上述分层调优方法的一个简要概括。

数据库的某个层级上出现的问题可能是由上一个层级的配置导致的，也可以通过调整上一个层级的配置而得到解决。因此，Oracle调优的合理步骤如下。

(1) 通过调整SQL、PL/SQL以及优化物理设计（分区、索引等）尽可能减少应用的请求。

(2) 通过减少对锁、闩锁、缓存及Oracle代码层级中其他资源的争用来获得最好的并发能力。

(3) 在前两步规范化逻辑IO需求的基础上，通过优化Oracle内存来最小化物理IO的需求。

(4) 到现在为止，物理IO的需求已经比较实际了。接下来，通过提供足够的IO带宽并均匀分布系统负载来配置IO子系统以满足上述物理IO需要。

本书中的优化过程是按照分层调优方法[①]来组织的。在本章的后续内容中，我们将依次考察这些步骤中的每一步，如图1-1所示。

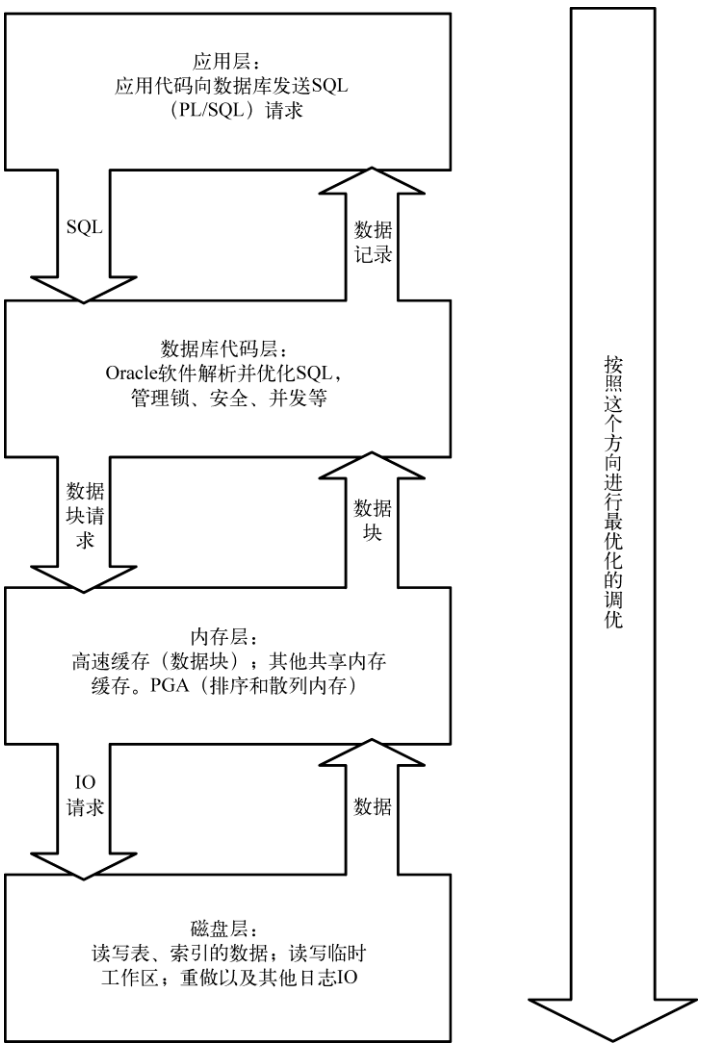

图1-1 Oracle数据库的4个主要层级

① Oracle数据库“按层级进行优化”的基本概念是由Steve Adams最先提出来的（http://www.ixora.com.au/tips/layers.zip）。

1.3 第一阶段：最小化应用负载

我们的第一个目标是：最小化应用对数据库的需求。也就是说，数据库应该以最少的操作来满足应用的数据请求。我们希望Oracle工作得更聪明而不是更勤奋。

宽泛地讲，我们主要使用如下两种技术来降低应用的工作负载。

- 优化应用代码 这可能涉及改变应用代码（C#、Ruby或Java），这样它们就可以向数据发出更少的请求（例如，通过使用客户端缓存）。当然，更多时候还涉及重写应用SQL与（或）PL/SQL代码。
- 修改应用数据库的物理实现 这可能涉及调整索引、反规范化或者分区。

第4章到第14章将详细介绍最小化应用负载的各种技术。特别是如下几种。

- 将应用结构化以避免数据库过载 应用可以避免对数据库发出不必要的请求，也可以被设计得更好以最小化锁和其他争用。
- 在与Orade数据库服务器通信时使用最佳实践 在设计并实现与Oracle进行通信的程序时，保证数据库的网络往返操作和不必要的请求最小化。
- 优化数据库物理设计 这包含通过创建索引、反规范化、创建分区以及其他涉及数据物理结构的方式来减少执行SQL请求相关的操作。
- 优化Oracle的查询优化器 恰当地配置优化器统计信息，在必要的时候覆盖优化器的计划，开始对SQL的性能进行不断监测。
- 优化单条SQL语句的性能 这可能涉及使用提示(hint)、存储概要（stored outline）、SQL剖析（Profile）以及SQL重写（rewrite）来改变SQL语句的执行计划。

- 使用并行SQL　允许使用多个进程来执行SQL语句。
- 优化并使用PL/SQL程序　在特定环境下可以使用PL/SQL来提高应用性能，PL/SQL程序为调优带来了独特的挑战与机遇。

这些技术不仅是调优工作的起点，也能带来最显著的性能提升。通过对SQL调优而带来百倍甚至千倍的性能提升是件很平常的事情，但是在优化争用、优化内存或者调整物理磁盘布局时很少能见到这么大的提升。

1.4 第二阶段：降低争用和瓶颈

在将应用的需求降低到合理的范围后，接下来要考虑处理Oracle服务器内部的争用。有两个或多个会话同时访问一个资源，就会发生争用，如锁争用或内存缓冲区争用。

在应用的需求涉及数据库时，争用——这个众所周知的“瓶颈”会限制系统可以完成的工作量。从应用的角度看，数据库处理得非常慢，甚至完全没有反应。从更低层面看，例如从磁盘子系统来看，响应请求的速度比它的实际处理速度更慢。争用瓶颈导致IO请求无法通过数据库代码层达到IO子系统，图1-2展示了这种现象。

在Oracle数据应用中，表内记录的争用（通常表现为锁等待）和共享内存区域的争用（表现为闩锁等待、内存缓冲区等待以及诸如此类的等待）是两种最常见的争用行为。

应用设计是导致锁争用的主要原因：在Oracle的锁机制中，需要读操作永远不等待锁，写操

作也永远不用等待读操作，并且只在行级别上应用锁[①]，这样Oracle才可以很好地支持高并发的应用。锁争用发生时，一般会涉及大量进程同时更新同一条记录，或者特定的锁持有的时间过长（可能是由于应用使用了悲观锁模型）。在不调整业务逻辑（我们认为应该在第一阶段的优化中就解决此问题）的情况下，这种争用几乎无法消除。此外，有些情况下，数据库配置或模式（Schema）中的问题，或者数据库的内部机制，都可能导致严重的锁争用。

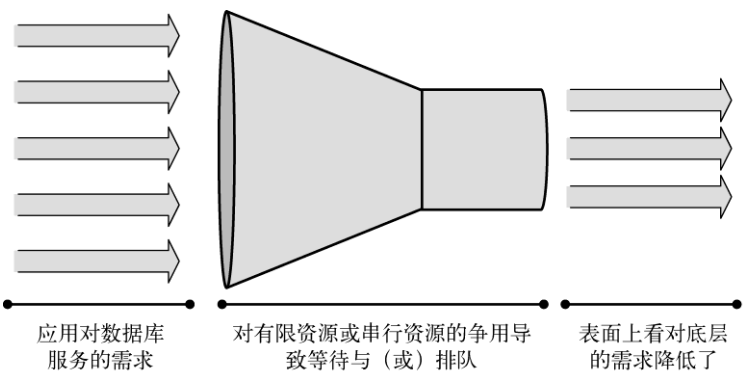

图1-2　争用是众所周知的瓶颈

当多个会话想要同时读写SGA中的共享内存时，会发生共享内存(shared memory)的争用。所有的共享内存都由闩锁（latch）来保护。闩锁与锁类似，不过它阻止的是对共享内存的并发访问，而锁则防止对表中数据的并发修改。如果会话希望修改内存中的部分数据，它需要取得相关的闩锁或者互斥（mutex），如果另一个会话也希望访问或修改同样的数据，也同样需要取得这个闩

① 实际上，有些时候，Oracle应用的锁也不只是在行级别上。Oracle的块锁主要表现为ITL等待（lock mode = 4），因为Oracle的每个块只支持一定数量的事务槽（ITL，Interested Transaction List），当一个块中的事务槽达到上限的时候，要修改这个块上的数据就必须等待其他事务释放这个块上的某个事务槽才可以。事务槽的上限由两个因素决定：(1)创建表、索引时是否指定了max itl count；(2)表、索引的块内是否有空间创建新的事务槽。相关内容可以参考我的博客文章www.dbthink.com/?p=575。——译者注

锁或互斥，这样就会发生等待。缓冲区高速缓存（buffer cache）中的数据块也会由于其他一些原因而发生争用，比如当一个内存块因为多个会话的处理请求相互冲突而不可访问时，就会出现大量缓冲区（buffer）等待。

第15章至第17章将介绍消除Oracle争用的相关技术，主要内容如下。

- 检测并处理锁争用，包含Oracle内部锁。
- 优化保护Oracle共享内存的闩锁机制。
- 定位并解决共享内存本身的争用。

1.5 第三阶段：降低物理 IO

到目前为止，应用需求已经最小化，可能掩盖此需求的争用也已经消除，第三阶段将集中注意力降低IO等待的时间开销。也就是说，在试图降低每个IO的时间耗费（IO延时）之前，先尝试降低IO的数量。事实证明，降低IO数量几乎总可以降低单个IO的时间延迟，因此，先解决IO规模的问题将取得事半功倍的效果。由于已经通过SQL优化和其他方法减少了应用请求，现在，我们将通过配置内存来缓存（cache）与缓冲[①]（buffer）IO请求，从而进一步降低IO。

Oracle数据库中发生的大部分物理IO，要么是应用会话为了查询或变更数据而请求数据，要么是会话必须排序或散列数据，或者必须创建临时段以支持大型连接、ORDER BY或类似操作。

Oracle的共享内存（SGA）存储数据块的副本，如果请求的数据块已经在这块内存中，就可

以免除相应的磁盘IO请求。

过去，分配SGA内存是一件很随机的事情。而现在Oracle服务器可以自动调整内存配置，或者通过检查咨询（advisory，可以准确预报将内存池调整到不同大小时的效果）来衡量调整内存池大小后的效果。

在Oracle中，可以配置几个独立的内存区域来缓存不同大小的数据块，也可以指定特定的区域来缓存需要存放在内存中的数据块。Oracle并不会自动调整所有这些内存区域的大小，也不会一开始就自动分配这些区域，或者指定特定的数据段存放到这些区域，这些任务都是交由DBA来处理的。

除了访问不在共享内存中的数据需要磁盘读操作外，在排序、分组、连接操作的过程中，对数据进行排序或者散列操作时，Oracle也可能执行大量的IO操作。Oracle会尽可能在内存中进行排序和散列操作，这块内存是专门为程序使用而分配的程序全局区（Program Global Area，简称PGA）。然而，如果这块内存的容量不足，Oracle将向位于磁盘上的临时段中写入（读出）数据以完成排序和散列操作。

在最近的版本中，Oracle已经提高了自动管理这些内存区域的能力。Oracle 10*g*可以自动在PGA与SGA内部调整内存配置，但并不会在这两个区域之间移动内存。Oracle 11*g*可以根据需要在PGA与SGA之间移动内存，或者至少在Oracle计算后认为必要时移动内存。

虽然数据库能自动管理内存，但是在确保最优性能方面仍然有大量工作需要数据库管理员

① 关于缓存（cache）与缓冲（buffer）之间的差别，请参考我的博客文章。——译者注

（DBA）来处理。

- 确保Oracle能否从操作系统得到足够的内存。
- 确保内存在PGA与SGA之间的合理划分。在Oracle 11*g*中，启用自动内存管理（Automatic Memory Management）。
- 在特定的内存区域中对段的分配做出微调。
- 对控制排序和连接的参数做出微调。
- 监控Oracle的内存分配，并且在必要时进行人工干预。

第18章至第20章介绍了这些内存优化技术。

1.6 第四阶段：优化磁盘 IO

到这一步，我们已经把应用的工作负载，特别逻辑IO数量控制在了合理范围内。此外，也已经消除了可能阻塞（从而掩盖）这些逻辑IO请求的争用。而且已经配置了可用的内存来尽可能减少这些逻辑IO引发的物理IO请求。现在（也只有到现在），才轮到考虑磁盘IO子系统是否可以应对这个挑战的问题。

优化磁盘IO子系统是一项复杂且专业化的工作，但基本原理则是浅显易懂的。

- 确保IO子系统有足够的带宽来处理物理IO请求。这主要由已经配置好的各种磁盘设备的数量来决定。不同的磁盘性能差异很大，但每秒基本上都能处理100次随机IO请求。通常磁盘的使用率低于100%（比如说50%至75%）是保证响应时间的关键。对于大部分数据

库，满足IO请求意味着多加磁盘，而不是更大的存储空间。只有足够多的磁盘才能保证满意的响应时间与IO速度，空间再大也只能用于存储数据。

- 在配置的所有磁盘上均匀地分布负载。RAID 0（条带化）是最佳方式；对于大部分数据库来讲，RAID 5是最差的方式，因为它在写IO上有严重的缺陷。

IO子系统压力过载的一个明显特征是IO请求的响应时间过长。不同磁盘的预期延时（也称为服务时间，service time）各不相同，但是，即使是最慢的磁盘也不应该超过10 ms。带有大容量内存高速缓存的磁盘阵列以及SSD（Solid State Disk，固态盘）延时较短。NAS（Network Attached Storage，网络附加存储）设备的服务时间里还可能包含相当一部分网络延时。

在磁盘间分散IO负载的工作，最好交由硬件或软件的条带（striping）完成。Oracle的ASM[①]技术提供了一种简单通用的方法来为普通的磁盘设备做条带。将数据文件分布在磁盘上效果通常要差一点，但仍然比完全不使用条带好。大部分的高端数据库系统都使用硬件磁盘阵列的条带化功能。

对大部分数据库来讲，为读操作优化数据文件效果最显著，因为Oracle会话一般不会在数据文件的写操作上产生等待；数据库写进程（DBWR）是以异步方式将数据写到磁盘的。然而，如果DBWR进程跟不上数据库的写进度，那么会话需要等待DBWR赶上才能处理。同样，我们也需要确保闪回与重做日志写进程（LGWR）可以赶上。另外，用户会话也需要等待这

① ASM是Automatic Storage Management的缩写，Oracle为了解决低端IO设备的存取、IO条带与IO镜像而开发的管理工具。——译者注

些进程。

第21章和第22章介绍与优化磁盘IO相关的技术。

- 理解Oracle的IO机制——缓存IO与直接IO，重做与归档日志IO，闪回IO及其他机制。
- 测量IO性能以及计算最优磁盘配置。
- 使用包含RAID级别在内的数据条带机制。
- 使用特定的IO相关技术，如ASM与SSD。

1.7 本章小结

当面对明显IO受限的数据库时，很容易想到立即去解决症状最明显的系统（IO子系统）。解决了外在症状而没有解决内在病因，并且代价高昂，最终还会失效。因为数据库的某个层面出现的问题可能是由上一层的问题导致的，可以通过解决上一层的问题来解决，所以优化数据库最快最好的方法就是从根本的问题开始，一层层地剥茧抽丝。

(1) 通过优化SQL语句、优化物理设计（区分、索引）以及优化PL/SQL来最大程度地降低应用需求。

(2) 通过最小化对锁、闩锁、缓冲区以及Oracle代码级别的其他资源的争用来最大化并发能力。

(3) 在依靠前序步骤将逻辑IO需求控制在合理范围内之后，通过优化Oracle内存来最小化相应的物理IO。

(4) 到这一步，物理IO需求都是实实在在的了。最后，通过提供足够的IO带宽以及在磁盘上

均衡负载来配置IO子系统以满足应用需要。