优化SQL Server性能的十大技巧

作者：Patrick O'Keefe和Richard Douglas

摘要

对SQL Server进行性能优化可能具有挑战性。对于如何解决一般的性能问题，有大量信息可供参考。但是，对于具体问题并没有太多详细的描述，关于如何在自己的环境中运用这些特定知识的信息甚至更少。

本白皮书将以SQL Server 2008和2012为背景，提供有关如何优化SQL Server性能的10大技巧。虽然没有明确列出最重要的技巧，但从我们提供的技巧入手绝不会错。

作为数据库管理员，您无疑有自己的考虑角度，并且有自己喜爱的优化技巧、秘诀和脚本。何不加入SQLServerPedia的讨论呢?

简介

想想您的调整目标。

我们都希望通过SQL Server部署获得最大价值。提高数据库服务器的效率可腾出系统资源用于其他任务，如业务报告或临时查询。要充分利用贵公司的硬件投资，您需要确保数据库服务器上运行的SQL或应用程序工作负载尽可能以最快的速度和最高的效率执行任务。

但是如何进行调整取决于您的目标。您可能会问，“我能否通过SQL Server部署获得最佳效率？”但其他人可能会问，“我的应用程序是否能扩展？”以下是调整系统的几种主要方式：

- 进行调整以达成服务级别协议(SLA)或关键性能指标(KPI)目标
- 进行调整以提高效率，从而腾出资源用于其他用途
- 进行调整以确保可扩展性，从而帮助在未来继续满足SLA或KPI要求

力求最大限度提高所有数据库服务器的可扩展性和效率，即使已满足当前的业务需求也是如此。

请记住，调整是一个持续的过程，而不是一次性的修复。

性能优化是一个持续的过程。例如，如果是要针对SLA目标进行调整，您可能已经“完成”调整。但是，如果您的调整目标是提高效率或确保可扩展性，您的工作就永远不会真正完成；这种类型的调整应该一直持续到性能达到“足够好”的水平。以后，如果应用程序的性能不再那么好，应再次执行调整周期。

“足够好”通常由SLA或系统吞吐量需求等业务需求来定义。除这些需求外，您还应该主动地尽量提高所有数据库服务器的可扩展性和效率，即使已满足当前的业务需求也是如此。

关于本文档

对SQL Server进行性能优化可能具有挑战性。各个数据点上都存在大量的一般性信息，例如性能计数器和动态管理目标(DMO)，但是有关如何处理这些数据以及如何对其进行解释的信息却非常少。本白皮书将介绍在实际操作中非常有用的10种重要技巧，让您能够将部分此类数据转变为可付诸行动的信息。

技巧10. 建立基准的方法可帮助您发现问题。

基准建立流程概述图1显示了基准建立过程中的步骤。以下部分将介绍该过程中的关键步骤。

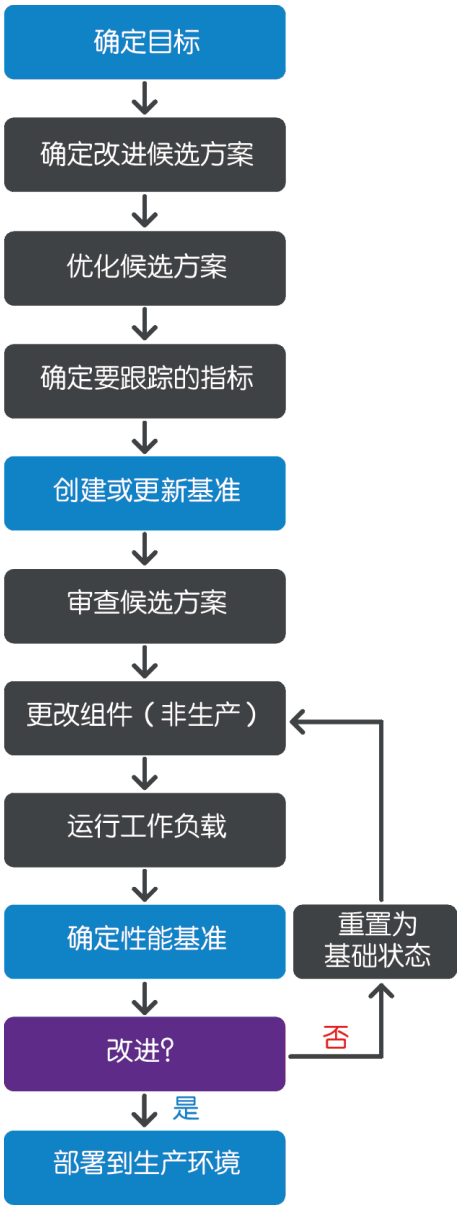

图1. 建立基准的过程

在开始进行更改之前，获取有关当前系统性能的记录。进行调整和优化时，大多数人会倾向于直接开始调整。当您从机修工那里取回汽车时，是否怀疑过它的性能实际上还不如以前了？您可能会抱怨，但是无法确定。您可能还怀疑您现在发现的问题是由机修工造成的，或者是在修理之后出现的。不幸的是，数据库的性能可能遭遇同样的问题。

阅读完本白皮书后，您将会有很多想法，并且会希望立即实施基准策略。您需要做的第一步不是最令人兴奋的，但肯定是最重要的行动之一：确定您的环境还要改善多少才能符合您计划更改的标准。

确定您的目标。
对您的系统进行任何操作之前，请先确定要实现的目标。您的SLA是否存在需要考虑到性能、资源消耗或容量的方面？您是否正在解决当前的生产问题？是否收到过有关资源时间的抱怨？请设定一些明确的目标。

我们大多数人都需要管理很多数据库和实例。为了从工作中获得最大价值，我们需要仔细考虑对某一特定系统的要求，以便该系统具有卓越的性能并满足用户期望。如果您过度地进行分析和调整，就会发现用在低优先级系统上的时间过多，这不利于您的核心生产系统。明确您要通过测量和调整工作实现的确切目标。确定这些目标的优先顺序，最好获得业务发起人的认可和同意。

建立标准。
确定要达到的目标后，您就需要决定如何衡量成功。哪些操作系统计数器、SQL Server计数器、资源测量标准和其他数据点可为您带来所需的洞察力？

有了上述内容的清单后，您需要建立基准，或者根据您选择的标准确立系统的标准性能。您需要在足够长的时间段内收集足够多的数据，以获得系统典型性能的代表性样本。收集到数据后，您可以通过计算该时间段内的平均值来建立第一个基准。对系统进行修改后，您将获得新的基准，将其与原始基准进行比较，以便客观测量更改的影响。

尽管关注平均值非常重要，但不要只是跟踪平均值，还要跟踪与标准的偏差。至少，计算每个计数器的标准偏差可以为您指示随时间发生的变化。想象一下，攀岩者听说绳子的平均直径是1厘米。这位攀岩者放心地跃上岩石。他摇晃地悬挂在锋利岩石上空的几百英尺处，自鸣得意地笑着。然后，他听说绳子最粗的部分是2厘米，最细的地方只有0.1厘米。糟糕！

如果您不熟悉标准偏差，请参阅关于统计信息的入门者手册。您无需过于深入地了解这些知识，它会帮助您了解基本知识。

这里的关键信息是不仅要跟踪平均值；还要跟踪与标准（平均值）的偏差。确定标准应该是什么（通常可在SLA中找到）。您的任务不是尽可能提高性能，而是尽可能好地实现性能目标，然后尽可能限制与这些目标的偏差。进行任何其他事项都会浪费时间和精力，并且还可能导致对基础架构资源利用不足。

建立基准有助于您发现异常行为，因为您明确指明了什么是异常行为。

您的任务不是尽可能提高性能，而是尽可能好地实现性能目标，然后尽可能限制与这些目标的偏差。

建立一个基准需要多少数据？建立基准所需的数据量取决于负载随时间变化的范围。咨询系统管理员、最终用户和应用程序管理员，他们通常较为了解使用模式的具体情况。您应该收集足够多的数据来涵盖非高峰期、一般时期和高峰期。
测量负载变化量以及变化频率非常重要。对于具有可预测模式的系统，需要的数据较少。变化越多，制定可靠基准所需的测量间隔就越小并且测量时间也越长。我们再进一步看看攀岩的比喻，您检查过的绳子长度越长，发现粗细差异的机会就越大。系统的关键程度及其故障的影响大小也将影响必须进行的检查工作量以及样本必须达到的可靠程度。

存储数据
跟踪的参数越多且频率越低，要收集的数据集就越大。这似乎是显而易见的，但您需要考虑存储测量数据所需的容量。收集到一些数据后，应该可以非常明确地推断出存储库随时间推移将增长到多大。如果您要延长监控时间，可考虑按照间隔汇总历史数据，以免存储库变得过大。

出于性能考虑，您的测量结果存储库不能与您正在监控的数据库位于同一位置。

限制您一次进行的更改数。
尝试限制您在各个基准之间进行的更改数。进行修改以测试一个特定的假设。这使您可以细致地选择或排除各个改进候选项。当您经过反复试验最后确定解决方案时，就会确切地明白为什么会出现行为变化，而这往往会揭示一系列其他潜在的改进方案。

分析数据
对系统进行更改后，您将需要确定它们是否具有预期的效果。为此，需要在有代表性的相似时间范围内重复进行用来获得原始基准的测量过程。然后，您可以对两个基准进行比较：

- **确定您的更改是否具有期望的效果** - 调整配置设置、优化索引或对SQL代码进行一些更改时，通过基准可以了解该更改是否具有期望的效果。如果您收到关于性能降低的抱怨，就可以断定从数据库的角度来看该陈述是否准确。

大多数初级数据库管理员最常犯的错误是过早地得出结论。您经常会看到某个人在完成一次或多次更改后，因发现性能瞬间得到显著增长而欢呼雀跃。他们将更改部署到生产，并迫不及待地发送电子邮件宣称问题已得到解决。但是好景不长，随后不久又出现了同一问题，或者某个未知的副作用导致出现另一个问题。通常，这会导致比原先还要糟糕的状况。您认为发现了问题的解决方法后，要对其进行测试并为结果建立相应的基准。这是确保取得进步的唯一可靠方式。

- **确定更改是否产生了任何意外副作用** - 通过基准您还能客观地看到更改是否影响意料之外的计数器或测量结果。
- **提前预期问题** - 使用基准可以根据典型负载条件建立准确的性能标准。这使您可以根据当前的资源消耗趋势或对未来情景的预计工作负载来预测将来是否会遇到问题以及何时遇到问题。例如，执行容量规划：通过从当前每个连接用户的典型资源消耗情况进行推断，可以预测您的系统是否将遇到用户连接瓶颈。
- **更有效地对问题进行故障排除** - 您是否曾花费多个漫长的日日夜夜对数据库进行紧急调整以解决性能问题，结果却发现实际上与数据库本身没有任何关系？建立基准可用来更加明确地排除数据库实例并找到问题的根源。例如，假设内存消耗突然飙升。您会注意连接数急剧增加，远高于您的基准。迅速呼叫应用程序管理员以确认是否已在电子商店中部署了新的模块。不用多久便证实新的初级开发人员编写的代码无法按预期发布数据库连接。我敢肯定您会想到更多像这样的情况。

通过清理混乱局面排除那些与问题无关的事项，并将注意力集中在导致问题的原因上，可以节省大量时间。有很多示例证明，将系统计数器与SQL Server计数器进行比较可以针对问题快速选择或排除数据库。排除一般的可疑项后，现在可以开始搜索与基准的巨大偏差，将相关指标收集在一起，并开始深入分析根本原因。

根据需要重复执行建立基准的过程。
良好的调整是一个迭代且科学的过程。本文档中介绍的技巧提供了一个很好的起点，但它们也只是一个起点。性能调整是一项高度个性化的工作，受各个系统的设计、结构和使用的控制。

建立基准的方法是进行良好、可靠的性能调整的基石。它为我们提供了找到前进方向和道路所需的地图、参考和路标，确保我们不会在途中迷路。通过结构化的方法可以跨数据库组合构建可靠且一致的性能。

技巧9. 性能计数器为您提供有关当前正在运行的操作的快速有用信息。

监控性能计数器的原因
有关SQL Server性能优化的一个很常见的问题是：“我应该监控哪些计数器？”。在管理SQL Server方面，监控性能计数器有两大原因：

- 提高运营效率
- 预防瓶颈

尽管它们有一些重叠，但根据这两个原因您可以轻松选择要监控的一些数据点。

监控性能计数器以提高运营效率运营监控可检查常规资源使用情况。它可以帮助回答诸如下面的问题：

- 服务器的CPU、磁盘空间或内存等资源是否即将用尽？
- 数据文件是否能够增长？
- 固定大小的数据文件是否有足够的可用空间存储数据？

建立基准所需的数据量取决于负载随时间变化的范围。

限制在各个基准之间所做的更改数，以便细致地评估每个更改的影响。

您还可以使用数据进行趋势分析。一个很好的例子是，收集所有数据文件的大小，以对它们的增长率进行趋势分析并预测未来的资源要求。

计数器	使您可以
处理器\处理器时间百分比	监控服务器上的CPU使用率
逻辑磁盘\可用MB	监控磁盘上的可用空间
MSSQL$Instance:数据库\数据文件大小(KB)	随时间变化的增长趋势
内存\页/秒	检查分页，分页可以很好地指示内存资源短缺情况
内存\可用MB	查看可供系统使用的物理内存量

监控性能计数器以防止出现瓶颈

瓶颈监控则更多关注性能相关的事项。您收集的信息有助于回答诸如以下问题：

- 是否存在CPU瓶颈?
- 是否存在I/O瓶颈?
- 缓冲区高速缓存和计划高速缓存等主要的SQL Server子系统是否运行良好?
- 数据库中是否存在争用情况?

为了回答这些问题，请看以下计数器：

计数器	使您可以
处理器\处理器时间百分比	监控CPU使用率使您可以检测服务器上的瓶颈（由较高的持续使用率指示）。
高比例的信号等待	信号等待是指工作线程在完成其他等待（如锁定、闭锁或其他等待）之后等待CPU所花费的时间。等待CPU所花费的时间可以指示CPU瓶颈。 在SQL Server 2000上执行DBCC SQLPERF(waitstats)或在SQL Server 2005上查询sys.dm_os_wait_stats可查看信号等待。
物理磁盘\平均磁盘队列长度	检查磁盘瓶颈：如果值超过2，则很可能存在磁盘瓶颈。
MSSQL$Instance:缓冲区管理器\页预期寿命	页预期寿命是页保持在缓冲区高速缓存中的秒数。较低的数字表示页未在高速缓存中保存太多时间即被逐出，这会降低高速缓存的效率。
MSSQL$Instance:计划高速缓存\高速缓存命中率	较低的高速缓存命中率表示计划未重复使用。
MSSQL$Instance:常规统计\已阻止的进程数	较长的阻止时间表示存在资源争用情况。

技巧8. 更改服务器设置可以实现更加稳定的环境。

更改产品中的设置使其更加稳定可能听起来违背直觉，但在这种情况下确实有作用。作为数据库管理员，您的工作是在用户通过其应用程序请求数据时，确保为他们提供一致的性能级别。在不更改下文所列的设置的情况下，您可能会遇到为用户提供的性能毫无征兆地出现降级的情况。这些选项可以在sys.configurations中轻松找到，其中列出了服务器级配置以及其他信息。sys.configurations中的Is_Dynamic属性显示了SQL Server实例在进行配置更改后是否需要重新启动。要进行更改，需要通过相关参数调用sp_configure存储程序。

最小值和最大值内存设置可保证实现特定级别的性能。

假设我们有一个主动/主动群集（或实际上是具有多个实例的一台主机）。我们可以进行某些配置更改，以在故障转移情况下（两个实例将位于同一物理机器上）保证我们满足SLA要求。

在此情况下，我们将更改最小值和最大值内存设置，以确保物理主机具有足够的内存来处理每个实例，不必一直尝试积极精简其他实例的工作集。可以进行类似的配置更改以使用特定的处理器，从而保证实现特定级别的性能。

务必注意，设置最大内存不仅适合群集上的实例，还适合与任何其他应用程序共享资源的实例。如果SQL Server内存使用率过高，操作系统可能会积极精简它可以使用的内存量，以便自己或其他应用程序有运行空间。

- **SQL Server 2008** - 在SQL Server 2008 R2及以前版本中，最小值和最大值内存设置仅限制缓冲池使用的内存量，更具体地说是仅限制单个8 KB页分配。这意味着如果在缓冲池外运行进程（例如扩展的存储程序、CLR或者集成服务、报告服务或分析服务等其他组件），则需要进一步降低该值。
- **SQL Server 2012** - SQL Server 2012更改的内容很少，因为存在中央内存管理器。此内存管理器现在采用多页分配，例如大型数据页和大于8 KB的高速缓存计划。此内存空间现在还包括一些CLR功能。

两个服务器选项可间接帮助提高性能。

没有任何选项可直接帮助提高性能，但有两个选项可间接帮助提高性能。

- **默认压缩备份** - 此选项在默认情况下将备份设置为进行压缩。尽管这可能在压缩过程中产生额外的CPU周期，但是与未压缩的备份相比，通常总体上使用的CPU周期更少，因为向磁盘写入的数据减少了。根据您的I/O架构，设置此选项还可能会减少I/O争用情况。
- 第二个选项可能会也可能不会在后面关于计划高速缓存的技巧中出现。您必须等待，看看它是否在10大技巧列表中。

关于故障排除的明智建议只有五个字：“消除或发现”。

监控性能计数器可以帮助您提高操作效率并避免瓶颈。

技巧7. 在计划高速缓存中查找恶意查询。

发现瓶颈后，您需要找到引起该瓶颈的工作负载。这一工作更容易，因为SQL Server 2005中引入了动态管理对象(DMO)。SQL Server 2000及以前版本的用户将不得不止步于使用Profiler或跟踪（在技巧6中详细介绍）。

诊断CPU瓶颈在SQL Server中，如果您发现了一个CPU瓶颈，首先要做的是获取该服务器上CPU资源的最高占用者。这通过sys.dm_exec_query_stats上的一个简单查询即可完成：

```
SELECT TOP 50
    qs.total_worker_time / execution_count AS avg_worker_time,
    substring (st.text, (qs.statement_start_offset / 2) + 1,
       ( ( CASE qs.statement_end_offset WHEN -1
            THEN datalength (st.text)
            ELSE qs.statement_end_offset END
         - qs.statement_start_offset)/ 2)+ 1)
      AS statement_text,
   *
FROM sys.dm_exec_query_stats AS qs
   CROSS APPLY sys.dm_exec_sql_text (qs.sql_handle) AS st
ORDER BY
avg_worker_time DESC
```

此查询真正有用的部分是您能够使用Cross Apply和sys.dm_ exec_sql_text获得SQL语句，以便对其进行分析。

诊断I/O瓶颈

诊断I/O瓶颈的方法与诊断CPU瓶颈的方法类似：

```
SELECT TOP 50
      (total_logical_reads + total_logical_writes) AS total_logical_io,
      (total_logical_reads / execution_count) AS avg_logical_reads,
(total_logical_writes / execution_count) AS avg_logical_writes,
(total_physical_reads / execution_count) AS avg_phys_reads,
      substring (st.text,
           (qs.statement_start_offset / 2) + 1,
      ((CASE qs.statement_end_offset WHEN -1
                THEN datalength (st.text)
      ELSE qs.statement_end_offset END
                - qs.statement_start_offset)/ 2)+ 1)
               AS statement_text,
   *
FROM sys.dm_exec_query_stats AS qs
    CROSS APPLY sys.dm_exec_sql_text (qs.sql_handle) AS st
ORDER BY total_logical_io DESC
```

技巧6. SQL Profiler是您的好朋友。

了解SQL Server Profiler和扩展事件

SQL Server Profiler是SQL Server附带的本地工具。通过它可以创建跟踪文件来捕获SQL Server中发生的事件。这些跟踪在提供关于您的工作负载和执行效果较差的查询的信息方面具有极大的价值。本白皮书不会详细深入地介绍如何使用Profiler工具。有关如何使用SQL Server Profiler的信息，请观看有关SQLServerPedia的视频教程。

尽管SQL Server Profiler已在SQL Server 2012中标记为过时以支持扩展事件，但应注意的是，这仅针对数据库引擎而不是SQL Server分析服务。Profiler仍可用来实时深入分析许多SQL Server环境中的应用程序工作情况。

使用扩展事件不在本白皮书的介绍范围之内；有关扩展事件的详细说明，请参阅白皮书《如何使用SQL Server的扩展事件和通知来主动解决性能问题》。可以这样说，将扩展事件引入SQL Server 2008并在SQL Server 2012中更新是为了包含更多事件和用户热切期待的UI。

请注意，运行Profiler需要具有ALTER TRACE权限。

如何使用SQL Profiler

下面介绍如何在性能监视器(Perfmon)中建立数据收集过程，并将有关资源使用情况的信息与SQL Server内部所发生事件的相关数据关联起来：

1. 打开Perfmon。
2. 如果您尚未配置数据收集器集，请使用高级选项以及技巧9中的计数器作为指导立即创建一个数据收集器集。暂不启动该数据收集器集。
3. 打开Profiler。
4. 通过指定有关要监控的实例、事件、列以及目标的详细信息，创建新跟踪。
5. 开始跟踪。
6. 切换回Perfmon以启动数据收集器集。
7. 将两个会话保持运行状态，直到捕获所需的数据。
8. 停止Profiler跟踪。保存跟踪，然后将其关闭。
9. 切换到Perfmon并停止数据收集。
10. 在Profiler中打开最近保存的跟踪文件。
11. 单击“文件”，然后选择“导入性能数据”。
12. 导航到数据收集数据文件，然后选择相关的性能计数器。

现在，您将能看到性能计数器与Profiler跟踪文件结合在一起（见图2），这将大大加快解决瓶颈问题的速度。

特别提示：上述步骤使用的是客户端界面；要节省资源，使用服务器端跟踪将获得更高的效率。请参阅联机丛书了解有关如何启动和停止服务器端跟踪的信息。

操作监控检查常规资源使用情况。瓶颈监控则更多关注性能相关的事项。

Sys.Configurations中的Is_Dynamic属性显示了SQL Server实例在进行配置更改后是否需要重新启动。

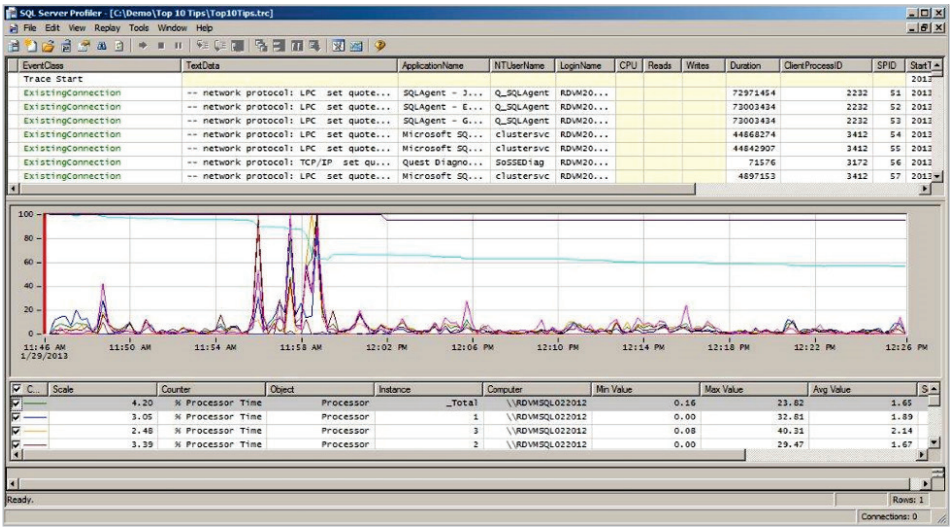

图2. 性能计数器与Profiler跟踪文件相结合的关联视图

技巧5. 配置SAN以提高SQL Server的性能。

存储区域网络(SAN)非常出色。它们能够通过一种简单的方式来配置和管理存储。可以配置SAN以在SQL Server中获得超快性能，但它们通常没有进行这样的配置。企业通常因为存储整合和简化管理等原因来实施SAN，并不是为了提高性能。更糟糕的是，通常您无法直接控制如何对SAN进行配置。因此，您经常会发现已经为必须在其中存储所有数据文件的逻辑卷配置了SAN。

关于配置SAN以提高I/O性能的最佳实践

如果您要获得最佳I/O性能，将所有文件存储在单个卷上通常不是一个好主意。作为最佳实践，您需要：

- **将日志文件放在它们自己的卷上**，与数据文件分开。日志文件几乎完全按顺序写入，而不进行读取（例外情况包括：数据库镜像和AlwaysOn可用性组）。您应始终以提高写入性能为目的进行配置。
- **将tempdb放在其自己的卷上。**Tempdb在SQL Server内部有无数用途，因此将其放在SQL Server的I/O子系统中很有帮助。为了进一步微调性能，首先需要一些统计信息。
- 考虑在VLDB中创建多个数据文件和文件组，**以受益于并行I/O操作。**
- **将备份放在其自己的驱动器上**以实现冗余目的，并且减少在维护期间与其他卷争用I/O的情况。

收集数据

当然，有一些Windows磁盘计数器可用于了解Windows认为正在发生的事情。（请记住根据RAID配置来调整原始数据。）SAN供应商通常会提供自己的性能数据。

SQL Server还提供了文件级的I/O信息：

- **SQL 2005以前的版本** - 使用fn_virtualfilestats函数。
- **更高版本** - 使用动态管理函数sys.dm_io_virtual_ file_stats。

通过在以下代码中使用此函数，您可以：

- 推导出读取和写入的I/O速率
- 获取I/O吞吐量
- 获取每个I/O的平均时间
- 查看I/O等待时间

```
SELECT db_name (a.database_id) AS [DatabaseName],
      b.name AS [FileName], a.File_ID AS [FileID],
      CASE WHEN a.file_id = 2 THEN 'Log' ELSE 'Data' END AS [FileType],
      a.Num_of_Reads AS [NumReads],
      a.num_of_bytes_read AS [NumBytesRead],
      a.io_stall_read_ms AS [IOStallReadsMS],
      a.num_of_writes AS [NumWrites],
      a.num_of_bytes_written AS [NumBytesWritten],
      a.io_stall_write_ms AS [IOStallWritesMS],
      a.io_stall [TotalIOStallMS],
      DATEADD (ms, -a.sample_ms, GETDATE ()) [LastReset],
      ( (a.size_on_disk_bytes / 1024) / 1024.0) AS [SizeOnDiskMB],
      UPPER (LEFT (b.physical_name, 2)) AS [DiskLocation]
  FROM sys.dm_io_virtual_file_stats (NULL, NULL) a
    JOIN sys.master_files b
      ON a.file_id = b.file_id AND a.database_id = b.database_id
ORDER BY a.io_stall DESC;
```

默认情况下，将备份设置为进行压缩可减少I/O争用情况。

分析数据

请特别注意查询中的“LastReset”值，它显示上一次启动SQL Server服务的时间。动态管理对象中的数据不是持续存在的，因此用于调整用途的任何数据均应针对服务已经运行的时间进行验证；否则，可能会做出错误的假设。

使用这些数字可以快速缩小与占用I/O带宽相关的文件的范围，然后提出这样的问题：

- 这是I/O所必需的吗？我是否缺少索引？
- 它是相关文件中的一个表还是索引？我是否可以把该索引或表放在另一个卷的其他文件中？

调整硬件的技巧如果正确放置了数据库文件，而且发现了所有对象热点并将它们分隔在不同的卷上，则可以密切关注一下硬件。

调整硬件是一个专业话题，不在本白皮书介绍的范围之内。不过，我可以与您分享一些最佳实践和技巧，使这项工作更简单一些：

- 创建SQL Server的使用卷时，不要使用默认分配的单元大小。SQL Server使用的单位是64 KB，因此该值应为最小值。
- 检查分区是否正确对齐。Jimmy May就这一主题编写了一篇非常棒的白皮书。未对齐的分区会使性能降低多达30 %。
- 使用诸如SQLIO等工具为系统的I/O建立基准。您可以观看关于此工具的教程。

DatabaseName	FileName	FileID	FileType	NumReads	NumBytesRead	IOStallReadsMS	NumWrites	NumBytesWritten	IOStallWritesMS	TotalIOStall...	LastReset	SizeOnDiskMB	DiskLocation
AWTarget2012	AdventureWorks2012_Data	1	Data	91	5709824	10626	8	73728	2	10628	31/01/201...	205.000000	C:
NewTest	NewTest	1	Data	36	2113536	9963	1	8192	19	9982	31/01/201...	5.000000	C:
AWSource2012	AdventureWorks2012_Data	1	Data	84	5251072	9731	2	16384	0	9731	31/01/201...	205.000000	C:
AW20121	AdventureWorks2012_Data	1	Data	74	4677632	9151	1	8192	0	9151	31/01/201...	205.000000	C:

图3. SQL Server中的文件级I/O信息

SQL Profiler仍可用来实时深入分析许多SQL Server环境中的应用程序工作情况。

技巧4. 光标和其他不良的T-SQL频繁返回会对应用程序造成干扰。

不良代码示例在前面的工作中，我看到了堪称职业生涯中所遇到的最糟糕的代码片段。系统早已被取代，这里概括一下函数的执行过程：

1. 接受剥离参数值。
2. 接受剥离参数表达式。
3. 查找表达式的长度。
4. 将表达式加载到变量中。
5. 循环变量中的每个字符，并检查该字符是否与要剥离的某个值匹配。如果匹配，则更新变量以将其删除。
6. 继续对下一字符进行同样操作，直到对表达式完成彻底检查。

如果您面露不安，则说明您犯了同样的错误。当然，我介绍的是别人在试图编写自己的T-SQL“REPLACE”语句！

最糟糕的是，这个函数用于更新某个邮寄例程的地址，而且每天会调用数万次。

运行服务器端Profiler跟踪可以查看服务器的工作负载，并挑选出频繁执行的代码片段（这正是找到这块“宝石”的方法）。

通过查询计划进行查询调整不良的T-SQL还表现为未使用索引的低效查询，主要是因为索引不正确或丢失。很好地了解如何使用查询计划进行查询调整非常重要。

有关使用查询计划进行查询调整的详细说明不在本白皮书的讨论范围之内。但是，开始该过程的最简单方法是将SCAN操作变为SEEK。SCAN操作会读取表或索引中的每一行，因此对于大型表，SCAN操作会占用大量I/O资源。另一方面，SEEK将使用索引直接访问所需的行，当然，该操作需要一个索引。如果您发现工作负载中使用的是SCAN，则可能表明丢失了索引。

有许多关于这一主题的好书，其中包括：

- “Professional SQL Server Execution Plan Tuning”（《专业的SQL Server执行计划调整》），作者：Grant Fritchey
- “Professional SQL Server 2012 Internals & Troubleshooting”（《专业的SQL Server 2012内部机制和故障排除》），作者：Christian Bolton、Rob Farley、Glenn Berry、Justin Langford、Gavin Payne、Amit Banerjee、Michael Anderson、James Boother和Steven Wort
- “T-SQL Fundamentals for Microsoft SQL Server 2012 and SQL Azure”（《Microsoft SQL Server 2012和SQL Azure的T-SQL基础》），作者：Itzik Ben-Gan

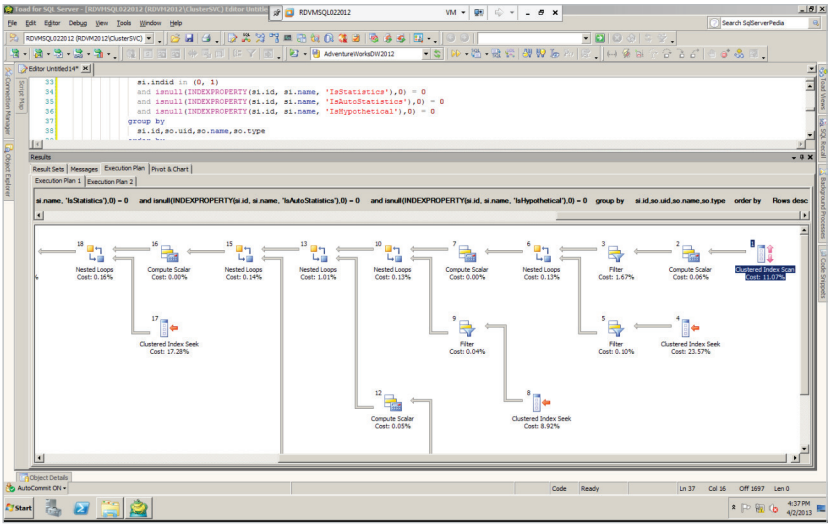

图4. 大型查询计划示例

技巧3. 最大限度地计划重用以进行更好的SQL Server高速缓存。

为什么重用查询计划非常重要

在执行SQL语句之前，SQL Server将首先创建一个查询计划。该计划定义了SQL Server使用何种方式获取查询的逻辑指令，并将它实施为针对数据的物理操作。

创建一个查询计划可能需要大量CPU资源。因此，如果SQL Server可以重用查询计划，而不是每次执行SQL语句时创建一个新的查询计划，则可以大大提高运行效率。

评估您的计划重用效果是否良好

SQL Statistics性能对象中提供了一些性能计数器，会告诉您计划重用的效果是否良好。这个公式可告诉您提交进行编译的批处理比率：

您希望该数字尽可能小。1:1的比率意味着提交的每个批处理都将进行编译，并且没有任何计划重用。

(Batch Requests/sec – SQL Compilations/sec) / Batch Requests/sec

可以配置SAN以在SQL Server中获得超快性能，但它们通常没有进行这样的配置。

定位效果较差的计划重用

准确地找到导致计划重用效果较差的工作负载并不容易，因为问题通常存在于提交查询的客户端应用程序代码中。因此，您可能需要检查提交查询的客户端应用程序代码。

要查找嵌入客户端应用程序中的代码，您将需要使用扩展事件或Profiler。通过将SQL:StmtRecompile事件添加到跟踪中，您将能够看到何时发生重新编译事件。（也有一个名为SP:Recompile的事件，提供该事件是为了实现向后兼容性，因为在SQL Server 2005中重新编译的发生级别从程序级别更改为语句级别）。

一个常见的问题是，代码不使用已准备好的参数化语句。使用参数化查询不仅可改进计划重用和编译开销，还可降低通过字符串串联传递参数所涉及的SQL注入攻击风险。图5显示了两个代码示例。尽管这些示例是刻意的，但它们说明了通过字符串串联构建语句与将准备的语句和参数一起使用之间的差异。

SQL Server中的文件级I/O信息可帮助您准确找到哪些文件在占用I/O带宽。

```
public void ExecuteSomeSQL(int aParam) {
    //cmd is a SqlCommand created somewhere else

    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select foo1, foo2, foo3 from bar where foo1=" + aParam.ToString();

    SqlDataReader dr = cmd.ExecuteReader();
    try {
        while (dr.Read()) {

        }
    } finally {
        dr.Close();
    }

}
```

不良

```
public void ExecuteSomeSQL(int aParam) {
    //cmd is a SqlCommand created somewhere else

    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "select foo1, foo2, foo3 from bar where foo1 = @foo1";

    cmd.Parameters["@foo1"].Value = aParam;

    SqlDataReader dr = cmd.ExecuteReader();
    try {
        while (dr.Read()) {

        }
    } finally {
        dr.Close();
    }

}
```

良好

图5. 通过字符串串联构建语句的代码与使用准备的语句和参数的代码比较

SQL Server无法重用图5中的“不良”计划。如果某个参数是字符串类型，则该函数可能会载入SQL注入攻击。“良好”示例不容易受到SQL注入攻击，因为使用了参数，并且SQL Server能够重用计划。

技巧8中缺少的配置设置

如果您记忆力很好，会记得在技巧8（我们讨论了配置更改）中有一个待讲解的建议。SQL Server 2008引入了名为“优化临时工作负载”的配置设置。该设置将指示SQL Server在计划高速缓存中存储存根计划，而不是存储整个计划。这对于使用动态生成的T-SQL代码或使用可能导致代码无法重用的Linq的环境尤其有用。

技巧2. 了解如何读取SQL Server缓冲区高速缓存，并尽量减少高速缓存抖动。

为什么缓冲区高速缓存非常重要正如前文所暗示，缓冲区高速缓存是SQL Server使用的大型内存区域，可降低执行物理I/O的需求。所有SQL Server查询执行都不会直接从磁盘读取数据；数据库页从缓冲区高速缓存中读取。如果常用页不在缓冲区高速缓存中，则物理I/O请求将进行排队。然后，查询需要等待，并且从磁盘中获取页。

分配给计划高速缓存的内存驻留在缓冲池中。因此，臃肿的计划高速缓存会减少可在缓冲区高速缓存中存储的数据页量，这样就需要通过更多往返从I/O子系统获取数据，从而产生非常高昂的成本。

通过DELETE或UPDATE操作对页中数据所做的更改也会应用到缓冲区高速缓存中的页。稍后，这些更改将从磁盘中清除。这整个机制使SQL Server可以通过以下几种方式优化物理I/O：

- 可以通过一个I/O操作读取和写入多个页。
- 可以实施预读。SQL Server可能会注意到，对于某些类型的操作，读取连续的页可能非常有用（假设在读取请求的页后，您立即要读取相邻的页）。

注意：索引碎片会抑制SQL Server执行预读优化的能力。

评估缓冲区高速缓存运行状况

缓冲区高速缓存的运行状况有两个指标：

- MSSQL$Instance:**缓冲区管理器\缓冲区高速缓存命中率** - 此为在高速缓存中找到的页数与未在高速缓存中找到的页数（需要从磁盘读取的页数）的比率。理想情况下，您希望该数字尽可能高。可能存在这样的情况：命中率很高，但仍然出现高速缓存抖动。
- MSSQL$Instance:**缓冲区管理器\页预期寿命** - 这是SQL Server逐出页之前将其保持在缓冲区高速缓存中的时间。Microsoft指出页预期寿命大于五分钟属于正常情况。如果预期寿命低于该值，则可能表示存在内存压力（内存不足）或高速缓存抖动。

我想用一个比喻来结束本节的内容；许多人认为防抱死刹车系统和其他辅助技术的创新意味着制动距离应减少，同时这一新技术还会使速度上限得到提高。300秒（5分钟）页预期寿命的警告值导致在SQL Server社区中产生了类似的争论：一些人认为这是一个硬性的规定，而其他人则认为当今大多数服务器的内容容量增加意味着该数字应为数千秒而不是数百秒。在我看来，这种观点上的差异突显了基准的重要性，以及为什么详细了解您环境中每个性能计数器应具有的警告级别是如此重要。

高速缓存抖动

在一次大型的表或索引扫描过程中，扫描的每一页都必须经过缓冲区高速缓存，这意味着可能逐出有用的页以便为那些不太可能多次读取的页腾出空间。这会导致I/O较高，因为必须从磁盘重新读取消除的页。此高速缓存抖动通常表示正在扫描大量表或索引。

要找出哪些表和索引占用缓冲区高速缓存中的大部分空间，您可以检查sys.dm_os_buffer_descriptors DMV（在SQL Server 2005中提供）。下面的查询示例说明了如何访问

如果正确放置了数据库文件，而且发现了所有对象热点并将它们分隔在不同的卷上，则可以密切关注一下硬件。

```
SELECT o.name, i.name, bd.*
  FROM sys.dm_os_buffer_descriptors bd
    INNER JOIN sys.allocation_units a
      ON bd.allocation_unit_id = a.allocation_unit_id
    INNER JOIN
    sys.partitions p
      ON (a.container_id = p.hobt_id AND a.type IN (1, 3))
        OR (a.container_id = p.partition_id AND a.type = 2)
    INNER JOIN sys.objects o ON p.object_id = o.object_id
    INNER JOIN sys.indexes i
      ON p.object_id = i.object_id AND p.index_id = i.index_id
```

运行服务器端Profiler跟踪使您可以查看服务器的工作负载，并选出频繁执行的代码片段。

SQL Server中那些占用缓冲区高速缓存空间的表或索引列表：

您也可以使用索引DMV来找出哪些表或索引有大量物理I/O。

技巧1. 了解如何使用索引和查找不良索引。

SQL Server 2012提供了一些有关索引的有用数据，您可以使用SQL Server 2005版本中实施的DMO获取这些数据。

使用sys.dm_db_index_ operational_stats DMO

sys.dm_db_index_operational_stats包含有关每个索引的当前低级别I/O、锁定、闭锁和访问方法活动的信息。使用此DMF来回答下列问题：

- 我是否有“热”索引？我是否有存在争用情况的索引？“以毫秒为单位的行锁定等待时间/以毫秒为单位的页锁定等待时间”列指明了是否曾等待过此索引。
- 我是否有使用效率低下的索引？当前哪些索引导致了I/O瓶颈？page_io_latch_wait_in_ms列告诉我们将索引页引入缓冲区高速缓存期间是否已存在I/O等待，这可以很好地指示是否存在扫描访问模式。
- 正在使用何种访问模式？range_scan_count和singleton_lookup_count列可以告诉我们对特定索引使用了何种访问模式。

图6显示了按PAGE_IO_LATCH总等待时间列出索引的一个查询输出。这在尝试确定I/O瓶颈中涉及哪些索引时非常有用。

使用sys.dm_db_index_usage_ stats DMO

sys.dm_db_index_usage_stats包含不同类型索引操作的计数以及各类型操作的上次执行时间。使用此DMV回答下列问题：

- 用户如何使用索引？user_seeks、user_scans和user_lookups列可以告诉您对其进行操作的索引的类型和重要性。
- 索引的成本是多少？user_updates列可以告诉您索引适用什么级别的维护。
- 上次使用索引是什么时候？last_*列可以告诉您上次对索引执行操作的时间。

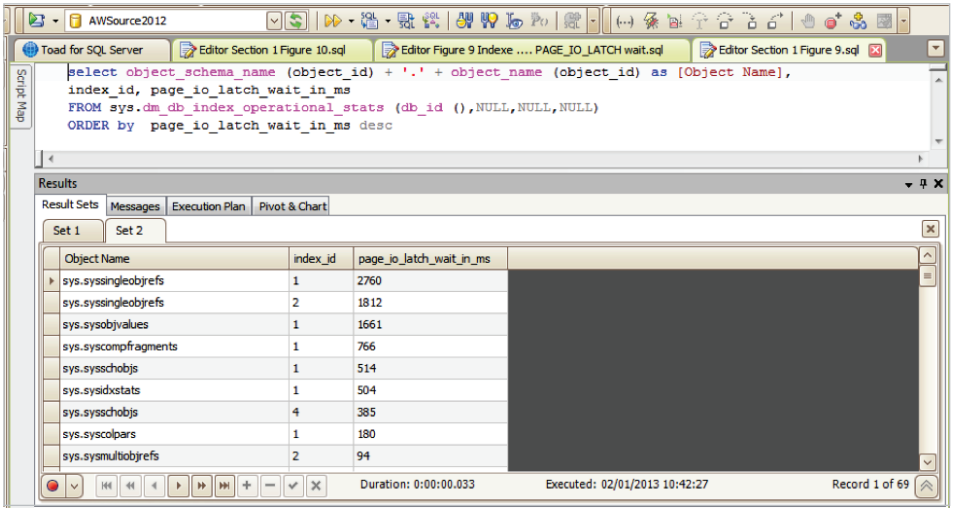

图6. 按PAGE_IO_LATCH等待总计列出的索引

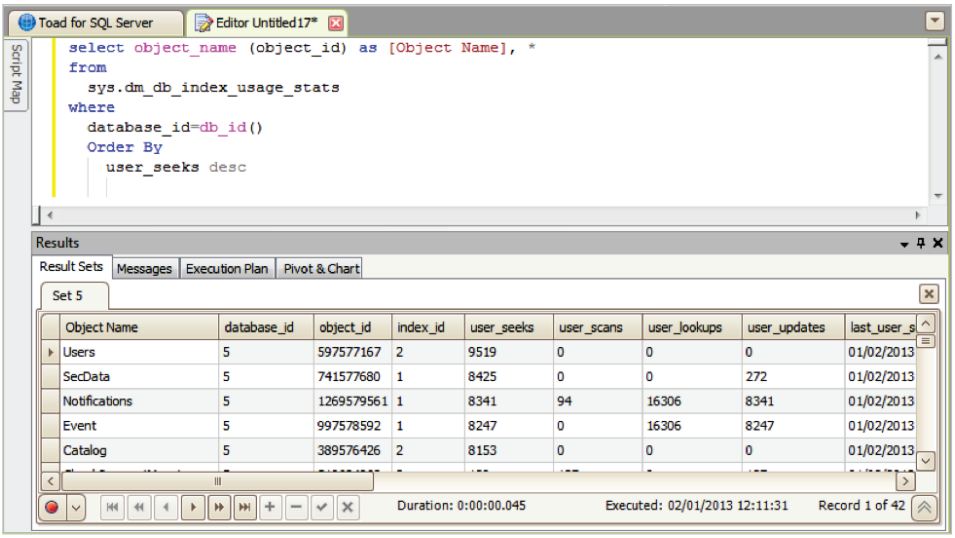

图7.按user_seeks总数列出的索引

图7显示了按user_seeks总数列出索引的一个查询的输出。如果您不希望确定具有高比例扫描的索引，则可以按user_scans列进行排序。现在，您有一个索引名称，如果能找出使用该索引的SQL语句岂不是很好？在SQL Server 2005及更高版本中，可以实现该目标。

当然，还有许多其他方面可编制索引，如设计策略、整合与维护。如果要更深入了解SQL Server性能调整的这一重要方面，请访问SQLServerPedia或者查看有关该主题的戴尔网络广播或白皮书。

总结

当然，对于SQL Server性能，需要了解的事情远不止10件。但是，本白皮书提供了一个很好的起点和有关性能优化的一些实用技巧，可运用到您的SQL Server环境中。

总括来说，优化SQL Server性能时，请记住以下10件事：

10. 建立基准有利于对工作负载行为进行比较，通过好的正常行为标准作参照发现异常行为。
9. 性能计数器可及时为您提供有关当前正在运行的操作的有用信息。
8. 更改服务器设置可以实现更加稳定的环境。
7. DMO可帮助您快速识别性能瓶颈。
6. 了解如何使用SQL Profiler、跟踪和扩展事件。
5. SAN不仅仅是执行I/O的黑盒子。
4. 光标和其他不良的T-SQL频繁返回会对应用程序造成困扰。
3. 最大限度地计划重用以实现更佳的SQL Server高速缓存。
2. 了解如何读取SQL Server缓冲区高速缓存，以及如何最大限度地减少高速缓存抖动。

优化SQL Server性能的首要技巧：

1. 通过学习如何使用索引以及如何查找不良索引掌握建立索引的知识。

> 页预期寿命小于五分钟时表明存在内存压力（内存不足）或高速缓存抖动。

使用参数化查询不仅可改进计划重用和编译开销，还可降低通过字符串串联传递参数所涉及的SQL注入攻击风险。

付诸行动

我相信您很想实践在本白皮书中学习到的知识。下表列出了开始进一步优化SQL Server环境需要执行的操作。

操作	子任务	目标日期
获得批准以启动项目	与您的产品线经理讨论并展示已将该项目准备就绪的案例，您可以主动出击而不是被动回应。	
确定性能目标	与公司利益相关方讨论以确定可接受的性能级别。	
为系统性能建立基准	收集相关数据并将其存储在自定义构建的存储库或第三方存储库中。	
确定最佳性能计数器并配置跟踪和/或扩展事件	下载戴尔的Perfmon计数器海报。	
查看服务器设置	特别要注意内存和“优化临时工作负载”设置。	
查看I/O子系统	如果合适，与您的SAN管理员进行讨论并考虑使用SQLIO等工具执行I/O负载测试，或者只是确定可进行密集读取和写入操作的速率，例如在执行备份时。	
确定执行效果较差的查询	分析跟踪、扩展事件会话和计划高速缓存返回的数据。	
重构执行效果较差的代码	通过SQLServerPedia的供稿博客服务了解最新的最佳实践。	
索引维护	确保索引尽可能保持最佳状态。	

了解详情

关于戴尔

Dell Inc.（纳斯达克代码：DELL）倾听客户的声音，并为他们提供值得信赖、备受青睐的全球性创新技术、业务解决方案和服务。有关详情，请访问www.dell.com。

如果您对本材料的可能使用存在任何疑问，请联系：

Dell Software
5 Polaris Way
Aliso Viejo, CA 92656
www.dell.com
请访问我们的网站，了解有关地区和国际办事处的相关信息。