每隔大约几个星期，我们就会接到苦恼的 DBA 们的电话，抱怨有关性能的问题。“我们 Web 站点速度慢得像蜗牛一样”，他们叫苦道，“我们正在失去客户，情况严重。你能帮忙吗？”为了回答这些问题，我为我的咨询公司开发了一个分析流程，它能让我们很快找到性能问题的原因，开发出补救措施并提出调整意见。这些打电话的人极少询问费用和成本 － 他们只关心制止损失。当 DB2 或电子商务应用程序的运行不能达到预期的性能时，组织和财务的收益将遭受极大的损失。

为了帮助 DB2 DBA 避免性能灾难并获得高性能，我为我们的客户、用户和 DB2 专家同行总结了一套故障诊断流程。以下详细说明在 Unix、Windows 和 OS/2 环境下使用 DB2 UDB 的电子商务 OLTP 应用程序的 10 条最重要的性能改善技巧 － 并在本文的结束部分作出总结。

**10. 监视开关**

确保已经打开监视开关。如果它们没有打开，您将无法获取您需要的性能信息。要打开该监视开关，请发出以下命令：

db2 "update monitor switches using

lock ON sort ON bufferpool ON uow ON

table ON statement ON"

**9. 代理程序**

确保有足够的 DB2 代理程序来处理工作负载。要找出代理程序的信息，请发出命令：

db2 "get snapshot for database manager"

并查找以下行：

High water mark for agents registered = 7

High water mark for agents waiting for a token = 0

Agents registered= 7

Agents waiting for a token= 0

Idle agents= 5

Agents assigned from pool= 158

Agents created from empty Pool = 7

Agents stolen from another application= 0

High water mark for coordinating agents= 7

Max agents verflow= 0

如果您发现Agents waiting for a token或Agents stolen from another application不为 0，那么请增加对数据库管理器可用的代理程序数（MAXAGENTS 和／或 MAX\_COORDAGENTS取适用者）。

**8. 最大打开的文件数**

DB2 在操作系统资源的约束下尽量做一个“优秀公民”。它的一个“优秀公民”的行动就是给在任何时刻打开文件的最大数设置一个上限。数据库配置参数MAXFILOP约束 DB2 能够同时打开的文件最大数量。当打开的文件数达到此数量时，DB2 将开始不断地关闭和打开它的表空间文件（包括裸设备）。不断地打开和关闭文件减缓了 SQL 响应时间并耗费了 CPU 周期。要查明 DB2 是否正在关闭文件，请发出以下命令：

db2 "get snapshot for database on DBNAME"

并查找以下的行：

Database files closed = 0

如果上述参数的值不为 0，那么增加MAXFILOP的值直到不断打开和关闭文件的状态停止。使用以下命令：

db2 "update db cfg for DBNAME using MAXFILOP N"

**7. 锁**

LOCKTIMEOUT的缺省值是 -1，这意味着将没有锁超时（对 OLTP 应用程序，这种情况可能会是灾难性的）。尽管如此，我还是经常发现许多 DB2 用户用LOCKTIMEOUT= -1。将LOCKTIMEOUT设置为很短的时间值，例如 10 或 15 秒。在锁上等待过长时间会在锁上产生雪崩效应。

首先，用以下命令检查LOCKTIMEOUT的值：

db2 "get db cfg for DBNAME"

并查找包含以下文本的行：

Lock timeout (sec) (LOCKTIMEOUT) = -1

如果值是 -1，考虑使用以下命令将它更改为 15 秒（一定要首先询问应用程序开发者或您的供应商以确保应用程序能够处理锁超时）：

db2 "update db cfg for DBNAME using LOCKTIMEOUT 15"

您同时应该监视锁等待的数量、锁等待时间和正在使用锁列表内存（lock list memory）的量。请发出以下命令：

db2 "get snapshot for database on DBNAME"

查找以下行：

Locks held currently= 0

Lock waits= 0

Time database waited on locks (ms)= 0

Lock list memory in use (Bytes)= 576

Deadlocks detected= 0

Lock escalations= 0

Exclusive lock escalations= 0

Agents currently waiting on locks= 0

Lock Timeouts= 0

如果Lock list memory in use (Bytes)超过所定义LOCKLIST大小的 50%，那么在LOCKLIST数据库配置中增加 4k 页的数量。

**6. 临时表空间**

为了改善 DB2 执行并行 I/O 和提高使用TEMPSPACE的排序、散列连接（hash join）和其它数据库操作的性能，临时表空间至少应该在三个不同的磁盘驱动器上拥有三个容器。

要想知道您的临时表空间具有多少容器，请发出以下命令：

db2 "list tablespaces show detail"

查找与以下示例类似的TEMPSPACE表空间定义：

Tablespace ID= 1

Name= TEMPSPACE1

Type= System managed space

Contents= Temporary data

State= 0x0000

Detailed explanation: Normal

Total pages= 1

Useable pages= 1

Used pages= 1

Free pages= Not applicable

High water mark (pages)= Not applicable

Page size (bytes)= 4096

Extent size (pages)= 32

Prefetch size (pages)= 96

Number of containers= 3

注意Number of containers的值是 3，而且Prefetch size是Extent size的三倍。为了得到最佳的并行 I/O 性能，重要的是Prefetch size为Extent size的倍数。这个倍数应该等于容器的个数。

要查找容器的定义，请发出以下命令：

db2 "list tablespace containers for 1 show detail"

1 指的是tablespace ID #1，它是刚才所给出的示例中的TEMPSPACE1。

**5. 内存排序**

OLTP 应用程序不应该执行大的排序。它们在 CPU、I/O 和所用时间方面的成本极高，而且将使任何 OLTP 应用程序慢下来。因此，256 个 4K 页（1MB）的缺省SORTHEAP大小（1MB）应该是足够了。您也应该知道排序溢出的数量和每个事务的排序数。

请发出以下命令：

Db2 "get snapshot for database on DBNAME"

并查找以下行：

Total sort heap allocated= 0

Total sorts = 1

Total sort time (ms)= 8

Sort verflows = 0

Active sorts = 0

Commit statements attempted = 3

Rollback statements attempted = 0

Let transactions = Commit statements attempted + Rollback

statements attempted

Let SortsPerTX= Total sorts / transactions

Let PercentSortOverflows = Sort overflows \* 100 / Total sorts

如果PercentSortOverflows ((Sort overflows \* 100) / Total sorts )大于 3 个百分点，那么在应用程序 SQL 中会出现严重的或意外的排序问题。因为正是溢出的存在表明发生了大的排序，所以理想的情况是发现没有排序溢出或至少其百分比小于一个百分点。

如果出现过多的排序溢出，那么“应急”解决方案是增加SORTHEAP的大小。然而，这样做只是掩盖了真实的性能问题。相反，您应该确定引起排序的 SQL 并更改该 SQL、索引或群集来避免或减少排序开销。

如果SortsPerTX大于 5 （作为一种经验之谈），那么每个事务的排序数可能很大。虽然某些应用程序事务执行许多小的组合排序（它们不会溢出并且执行时间很短），但是它消耗了过多的 CPU。当SortsPerTX很大时，按我的经验，这些机器通常会受到 CPU 的限制。确定引起排序的 SQL 并改进存取方案（通过索引、群集或更改 SQL）对提高事务吞吐率是极为重要的。

**4. 表访问**

对于每个表，确定 DB2 为每个事务读取的行数。您必须发出两个命令：

db2 "get snapshot for database on DBNAME"

db2 "get snapshot for tables on DBNAME"

在发出第一个命令以后，确定发生了多少个事务（通过取Commit statements attempted和Rollback statements attempted之和 － 请参阅 技巧 3）。

在发出第二个命令以后，将读取的行数除以事务数（RowsPerTX）。在每个事务中，OLTP 应用程序通常应该从每个表读取 1 到 20 行。如果您发现对每个事务有成百上千的行正被读取，那么发生了扫描操作，也许需要创建索引。（有时以分布和详细的索引来运行 runstats 也可提供了一个解决的办法。）

“get snapshot for tables on DBNAME”的样本输出如下：

Snapshot timestamp = 09-25-2000 4:47:09.970811

Database name= DGIDB

Database path= /fs/inst1/inst1/NODE0000/SQL00001/

Input database alias= DGIDB

Number of accessed tables= 8

Table List

Table Schema= INST1

Table Name= DGI\_

SALES\_ LOGS\_TB

Table Type= User

Rows Written= 0

Rows Read= 98857

verflows= 0

Page Reorgs= 0

Overflows 的数量很大就可能意味着您需要重组表。当由于更改了行的宽度从而 DB2 必须在一个不够理想的页上定位一个行时就会发生溢出。

**3. 表空间分析**

表空间快照对理解访问什么数据以及如何访问是极其有价值的。要得到一个表空间快照，请发出以下命令：

db2 "get snapshot for tablespaces on DBNAME"

对每个表空间，回答以下问题：

平均读取时间（ms）是多少？

平均写入时间（ms）是多少？

异步（预取）相对于同步（随机）所占的物理 I/O 的百分比是多少？

每个表空间的缓冲池命中率是多少？

每分钟读取多少物理页面？

对于每个事务要读取多少物理和逻辑页面？

对于所有表空间，回答以下问题：

哪个表空间的读取和写入的时间最慢？为什么？是因为其容器在慢速的磁盘上吗？容器大小是否相等？对比异步访问和同步访问，访问属性是否和期望的一致？随机读取的表应该有随机读取的表空间，这是为了得到高的同步读取百分比、通常较高的缓冲池命中率和更低的物理 I/O 率。

对每个表空间，确保预取大小等于数据块大小乘以容器数。请发出以下命令：

db2 "list tablespaces show detail"

如果需要，可以为一个给定表空间改变预取大小。可以使用以下命令来检查容器定义：

db2 "list tablespace containers for N show detail"

在此，N 是表空间标识号。

**2. 缓冲池优化**

我时常发现一些 DB2 UDB 站点，虽然机器具有 2、4 或 8GB 内存，但是 DB2 数据库却只有一个缓冲池（IBMDEFAULTBP），其大小只有 16MB！

如果在您的站点上也是这种情况，请为 SYSCATSPACE 目录表空间创建一个缓冲池、为TEMPSPACE表空间创建一个缓冲池以及另外创建至少两个缓冲池：BP\_RAND和BP\_SEQ。随机访问的表空间应该分配给用于随机访问的缓冲池（BP\_RAND）。顺序访问（使用异步预取 I/O）的表空间应该分配给用于顺序访问的缓冲池（BP\_SEQ）。根据某些事务的性能目标，您可以创建附加的缓冲池；例如，您可以使一个缓冲池足够大以存储整个“热”（或者说访问非常频繁的）表。当涉及到大的表时，某些 DB2 用户将重要表的索引放入一个索引（BP\_IX）缓冲池取得了很大成功。

太小的缓冲池会产生过多的、不必要的物理 I/O。太大的缓冲池使系统处在操作系统页面调度的风险中并消耗不必要的 CPU 周期来管理过度分配的内存。正好合适的缓冲池大小就在“太小”和“太大”之间的某个平衡点上。适当的大小存在于回报将要开始减少的点上。如果您没有使用工具来自动进行回报减少分析，那么您应该在不断增加缓冲池大小上科学地测试缓冲池性能（命中率、I/O 时间和物理 I/O 读取率），直到达到最佳的缓冲池大小。因为业务一直在变动和增长，所以应该定期重新评估“最佳大小”决策。

**1. SQL 成本分析**

一条糟糕的 SQL 语句会彻底破坏您的一整天。我不止一次地看到一个相对简单的 SQL 语句搞糟了一个调整得很好的数据库和机器。对于很多这些语句，天底下（或在文件中）没有 DB2 UDB 配置参数能够纠正因错误的 SQL 语句导致的高成本的情况。

更糟糕的是，DBA 常常受到种种束缚：不能更改 SQL（可能是因为它是应用程序供应商提供的，例如 SAP、 PeopleSoft或 Siebel）。这给 DBA 只留下三条路可走：

1. 更改或添加索引

2. 更改群集

3. 更改目录统计信息

另外，如今健壮的应用程序由成千上万条不同的 SQL 语句组成。这些语句执行的频率随应用程序的功能和日常的业务需要的不同而不同。SQL 语句的实际成本是它执行一次的成本乘以它执行的次数。

每个 DBA 所面临的重大的任务是，识别具有最高“实际成本”的语句的挑战，并且减少这些语句的成本。

通过本机 DB2 Explain 实用程序、一些第三方供应商提供的工具或 DB2 UDB SQL Event Monitor 数据，您可以计算出执行一次 SQL 语句所用的资源成本。但是语句执行频率只能通过仔细和耗时地分析 DB2 UDB SQL Event Monitor 的数据来了解。

在研究 SQL 语句问题时，DBA 使用的标准流程是：

1. 创建一个 SQL Event Monitor，写入文件：

$> db2 "create event monitor SQLCOST for statements write to ..."

2. 激活事件监视器（确保有充足的可用磁盘空间）：

$> db2 "set event monitor SQLCOST state = 1"

3. 让应用程序运行。

4. 取消激活事件监视器：

$> db2 "set event monitor SQLCOST state = 0"

5. 使用 DB2 提供的 db2evmon 工具来格式化 SQL Event Monitor 原始数据（根据 SQL 吞吐率可能需要数百兆字节的可用磁盘空间）：

$> db2evmon -db DBNAME -evm SQLCOST

> sqltrace.txt

6. 浏览整个已格式化的文件，寻找显著大的成本数（一个耗时的过程）：

$> more sqltrace.txt

7. 对已格式化的文件进行更完整的分析，该文件试图标识唯一的语句（独立于文字值）、每个唯一语句的频率（它出现的次数）和其总 CPU、排序以及其它资源成本的总计。如此彻底的分析在 30 分钟的应用程序 SQL 活动样本上可能要花一周或更多的时间。

要减少确定高成本 SQL 语句所花的时间，您可以考虑许多可用的信息来源：

从 技巧 4，务必要计算在每个事务中从每个表中读取的行数。如果产生的数字看上去很大，那么 DBA 可以在 SQL Event Monitor 格式化输出中搜索有关的表名称（这将缩小搜索范围而且节省一些时间），这样也许能够找出有问题的语句。 从 技巧 3，务必计算每个表空间的异步读取百分比和物理 I/O 读取率。如果一个表空间的异步读取百分比很高并远远超过平均的物理 I/O 读取率，那么在此表空间中的一个或更多的表正在被扫描。查询目录并找出哪些表被分配到可疑的表空间（每个表空间分配一个表提供最佳性能检测），然后在 SQL Event Monitor 格式化输出中搜索这些表。这些也可能有助于缩小对高成本 SQL 语句的搜索范围。 尝试观察应用程序执行的每条 SQL 语句的 DB2 Explain 信息。然而，我发现高频率、低成本语句经常争用机器容量和能力来提供期望的性能。 如果分析时间很短而且最大性能是关键的，那么请考虑使用供应商提供的工具（它们能够快速自动化识别资源密集的 SQL 语句的过程）。

继续调节

最佳性能不仅需要排除高成本 SQL 语句，而且需要确保相应的物理基础结构是适当的。当所有的调节旋钮都设置得恰到好处、内存被有效地分配到池和堆而且 I/O 均匀地分配到各个磁盘时，才可得到最佳性能。虽然量度和调整需要时间，但是执行这 10 个建议的 DBA 将非常成功地满足内部和外部的 DB2 客户。因为电子商务的变化和增长，即使是管理得最好的数据库也需要定期的微调。DBA 的工作永远都做不完！

快速回顾最棒的 10 个技巧

对工作负载使用足够的代理程序。

不允许 DB2 不必要地关闭和打开文件。

不允许长期的锁等待。

确保数据库的 TEMPSPACE 表空间的并行 I/O 能力。

保守地管理 DB2 排序内存并不要以大的 SORTHEAP 来掩盖排序问题。

分析表的访问活动并确定具有特别高的每个事务读取行数或溢出数的表。

分析每个表空间的性能特性，并寻求改善读取时间最慢、等待时间最长、物理 I/O 读取率最高、命中率最差的表空间性能以及与所期望的不一致的访问属性。

创建多个缓冲池，有目的地将表空间分配到缓冲池以便于共享访问属性。

检查 DB2 UDB SQL Event Monitor 信息以找到哪个 SQL 语句消耗计算资源最多并采取正确的措施。

一旦排除了高成本 SQL，马上重新评估配置和物理设计设置。