DB2 性能优化快速入门

级别： 初级

李越 (liyyue@cn.ibm.com), 软件工程师, IBM
王飞鹏 (wangfp@cn.ibm.com), 软件工程师, IBM
狄浩 (dihao@cn.ibm.com), 软件工程师, IBM
张蓉蓉 (rrzhang@cn.ibm.com), 软件工程师, IBM

2009 年 6 月 01 日

本文是针对 DB2® Universal Database（DB2 UDB）的初学者在遇到 DB2 的性能问题的时候不知道从何分析，从何入手的这一问题，从 DB2 的体系结构介绍开始，由浅入深的讲述了 DB2 性能优化的一些基础知识，简单地介绍调优的原理和工具。同时针对性能上的一些典型问题，如 DML 性能问题 (查询和插入操作)， DDL 性能问题（建表，建索引），并发性问题等等，介绍一种渐进的分析方法，同时结合一些具体的优化案例进行分析以及成型的 DB2 调优工具 (DB2 Performance Expert) 的介绍，使得初学者能够在很短的时间内掌握 DB2 性能优化的基本技巧，并且能够在数据库早期设计时考虑到一些性能因素而防范于未然。

DB2 性能优化概述

DB2 性能优化是一件较为复杂的综合性的工作 , 需要对问题的根源作全方位的探索和思考。同时也需要较深厚的数据库管理经验与优化知识。这对于初学者来说可能有些勉为其难。但是在很多情况下，随着 DB2 数据库中的数据量的不断增长或者用户数的激增，数据库系统的性能会显著下降，而此时快速定位性能上的瓶颈则至关重要。下面简要地介绍一下 DB2 的调优的一些因素和工具，以及一些原理，使初学者对性能优化能够有一个大致的了解。

DB2 的性能优化可以从三个方面分析：内存，CPU 和 I/O 。

内存因素

在内存方面，主要是考虑缓冲池 (BUFFERPOOL) 的使用。缓冲池是一片用来缓冲从磁盘上读取的数据和索引的内存区域，这些数据和索引信息在缓冲池中进行运算后最终还要写回磁盘。缓冲池的页面大小有四种 (4K,8K,16K,32K)，分别对应四种不同页面大小的表空间。缓冲池的大小决定了能够从磁盘上缓冲数据的容量大小。当然缓冲池也不是越大越好，缓冲池过大可能会导致连接数据库的时间过长，因为在连接数据库时要为数据库的缓冲池分配内存空间。可以通过计算缓冲池的命中率来评估缓冲池的使用效率：缓冲池命中率 =(1-((数据物理读 + 索引物理读)/(数据逻辑读 + 索引逻辑读))) *100%，缓冲池命中率越大说明缓冲池的使用效率高。缓冲池命中率太小说明缓冲池太小应当调大。其中的数据物理读，索引物理读以及数据逻辑读和索引逻辑读都可以从缓冲池的快照中获取。

在内存方面要考虑的另外几个重要因素是排序堆 (SORTHEAP)，锁列表 (LOCKLIST), 日志缓冲区 (LOGBUFSZ) 。排序堆在查询结果带有排序选项而没有相关索引对应时将会被使用，排序堆太小会产生排序溢出 (Overflowed), 那些在排序堆中装不下的排序数据将会溢出到一个临时表中，这会使性能下降。与 SORTHEAP 参数相关的是 SHEAPTHRES_SHR 和 SHEAPTHRES，SHEAPTHRES_SHR 限制了一个数据库中共享排序的最大内存，SHEAPTHRES 限制了私有排序的最大内存。 LOCKLIST 指的是一个数据库中用来存放锁的内存空间，当这个参数设得过小会导致在锁用光这部分资源后导致锁升级（即多个行锁转化为一个表锁来释放出更多的资源）。这会导致系统的并行性下降，很多应用连接出现挂起，使得系统的性能衰退。所以尽可能调大 LOCKLIST 参数，这里需要指出 LOCKLIST 指的并不是锁的个数，而是以数据库页为单位的一片内存区域（在 32 位系统中每个锁需要 96 个字节，锁上加锁的话每个锁则需 48 个字节。在 64 位系统中每个锁需要 128 个字节，锁上加锁的话每个锁则需 64 个字节）。与 LOCKLIST 参数对应的是 MAXLOCKS 参数，MAXLOCKS 定义的是一个百分数，它指定了一个应用程序所能占用的最大的锁空间占 LOCKLIST 的比例。日志缓冲区 (LOGBUFSZ) 指的是日志在写到磁盘以前用于缓冲的一片内存空间，这样可以减少写日志带来的过多的 I/O 。

从版本 9 以后 DB2 推出了一个新特性自调节内存管理器 (STMM: Self Tuning Memory Manager), 这个特性使得很多内存参数如前面所述的 SORTHEAP，LOCKLIST，LOGBUFSZ 等进行自动调节，当数据库参数

SELF_TUNING_MEM 设为 ON, 这些参数设为 AUTOMATIC 即可以进行自动调整。这样可以节省很多人工调整的时间。

CPU 因素

关于 CPU 因素首先是考虑 DB2 优化器 (OPTIMIZER) 对访问计划 (ACCESS PLAN) 的分析与优化。一般来说，一条 SQL 在执行时首先会被解析，然后进行语义分析，进而重写 SQL, 优化器会对重写过的 SQL 进行基于成本的分析最终选择最有效的访问计划。最终生成可执行代码（执行计划）来执行这条语句。查询访问计划的工具有很多，既有图形化工具 Visual Explain，也有命令 db2exfmt 来格式化解释表 (Explain tables) 中的数据生成 ACCESS PLAN 。还有命令 db2expln 查询 ACCESS PLAN 。

在 DB2 里的优化级别分为九级，缺省是第五级，级别越高优化器分析得程度越深。这个级别有数据库配置参数 DFT_QUERYOPT 决定。并不是级别设得越高性能越好，因为对于一些较为简单的 SQL 语句，如果优化级别过高那么花在优化 SQL 上的时间就会过长，而执行时间相对来说很短，有些得不偿失。在选择访问计划时，索引扫描的效率往往会比表扫描要高，所以索引的优化也是值得注意的。正确的建立索引会使查询性能大幅度的提高。

在 DB2 中连接 (JOIN) 分为三种：嵌套循环连接 (nest-loop join), 合并连接 (merge-join), 散列表连接 (hash-join) 。一般来说效率最低的是嵌套循环连接，这种连接采用的是笛卡儿集，进行多次循环遍历得到结果。而合并连接和散列表连接只进行一次循环遍历，相对来说效率较高。其中散列表连接可以采用多个等式做为条件而合并连接只能采用单个等式作为条件。但是在有索引扫描的情况下嵌套循环连接效率则更高。当优化级别等于零时，连接只能采用嵌套循环连接， 当优化级别大于等于 1 时，连接可以采用合并连接。当优化级别大于 5 时连接可以采用散列表连接。散列表连接要求 SORTHEAP 比较大，因为要为生成散列表准备空间。

在考虑 CPU 因素时还要考虑 CPUSPEED 这个参数，这个参数标明了 CPU 的运行速度，它会帮助优化器评估最好的访问计划。一般来说这个参数设为 -1，优化器将自动计算 CPU 的速度。另外运用多分区的特性可以把一个数据库分布到多台机器上，这样可以充分利用多台机器的 CPU 的资源对应用程序的事务进行并行处理，从而提高数据库的性能。

I/O 因素

关于 I/O 因素要考虑以下几个方面：首先是磁盘的 I/O, 为了能够最大化磁盘的 I/O 可以把数据，索引以及日志分别放在不同的硬盘上。因为在一个事务中数据和索引可能需要同时访问，而在事务提交时，数据和日志要同时写入磁盘，而且有可能索引也要同步维护，所以将它们放在不同的硬盘上可以使它们的读写并行运行，从而不致使磁盘成为瓶颈。同时选择数据库管理表空间 (DMS) 要比系统管理表空间 (SMS) 性能要好，因为读写 SMS 需要经过操作系统的 cache 再到缓冲池，而可以采用裸设备的 DMS 则不需要。但是 DMS 相对 SMS 来说维护起来较麻烦。

其次要考虑的是日志文件的大小，当数据库在写事务日志时当一个日志文件写满后会转向另外一个日志文件，这种日志文件的切换会造成操作系统上的开销。所以应当尽量将日志文件大小（LOGFILSIZ）设得大一些，这样可以减少日志文件切换的次数。但是日志文件过大难免会造成一些空间的浪费。

同时也要考虑到隔离级别的因素，在 DB2 中隔离级别分成 4 级：可重复的读，读稳定性，游标稳定性和未提交的读。这四种级别逐个降低。越高的隔离级别越能保证数据完整性，但却会降低并发性，所以应当综合权衡后做出决定。隔离级别可以通过如下命令来改变：
CHANGE ISOLATION TO=CS|RR|RS|UR

在连接方面还要考虑到代理和连接的关系，这也会影响到数据库的并发性，具体信息可以参考资源部分。

最后要考虑的还是关于多分区的特性。在多分区数据库中，一个请求首先传到协调分区，然后由协调分区将请求细分成多个部分发送到其他分区，这样数据可以在各个分区进行并行读写，实现 I/O 最大化。

性能优化相关工具

在 DB2 中有很多和性能优化相关的工具和命令，下面简单地介绍几种：

- **SNAPSHOT** : 这是 DB2 获取数据库信息快照的一种方法。它能够获取在数据库中关于缓冲池，锁，排序以及 SQL 等等信息。 DBA 可以通过获取这些信息来对数据库中的各组件进行评估来分析问题的瓶颈。
- **DB2PD** : 这个命令是用来分析数据库的当前状态，它带有很多参数。可以用来分析应用程序，代理，内存块，缓冲池，日志及锁状态等信息。

- **RUNSTATS** : 这个命令是用来收集数据库中数据的最新统计信息，并更新到系统表中。更新统计信息将会促使优化器选择更加符合实际的高效的访问计划，从而提高工作效率。
- **REORG** : 这个命令用来重新整理数据库中数据和索引的碎片，使其在物理上可以得以按一定规则排列，这样可以加快检索的速度。
- **DB2DART** : 这个命令是一个数据库的分析和报告工具，它用来检查表空间，索引以及数据库结构的正确性，分析在性能问题上的一些原因。
- **DB2SUPPORT** : 这个命令用来收集 DB2 和操作系统的所有相关信息并生成一个压缩文件，可传送给优化人员进行分析。

还有一些 DB2 中其他的文件可以用来分析性能问题，比如说诊断日志，追踪文件等。一些第三方的工具也可供参考，如“ tivoli monitor for db2 ”， QUEST 等等。

其他性能因素

- **XML 的优化：** 在 DB2 V9 以后引入了纯 XML 的数据类型，这是一种层次型数据类型。这和传统的关系型数据类型不一样，在 V9 以前 DB2 存储 XML 数据使用 CLOB 数据类型，应用程序在存取 XML 数据的时候必须先要解析 XML 再使用其数据。而在纯 XML 类型中，可以直接读取其中的元素，这样性能会有较大的提高。另外针对纯 XML 还有 XML 的索引，也会增大存取的性能。
- **操作系统：** 数据库存在于操作系统之上，操作系统的性能将直接影响到数据库的运行效率，因此优化操作系统也是优化数据库的一个重要过程。在操作系统级别上可以对内存进行优化，比如说对系统共享内存，信号量以及虚拟内存的设置等等都可以影响到数据库的性能。同时在磁盘的分布上也会影响到数据库 I/O 效率。
- **网络：** 网络将会影响到数据库的 I/O 性能，当数据通过网络在客户端和服务器端进行传送时，网络上出现瓶颈会导致数据库 I/O 性能显著下降。所以选择优良的网络设备以及配置良好的网络环境对数据库性能相当重要。同时也要考虑到防火墙的因素，有时防火墙会阻挡来自某些 IP 的数据包。

DB2 性能问题分类与分析思路

DML 性能问题

DML(Data Manipulation Language) 包括了查询，增加，删除和更新纪录等操作。首先看一下查询的性能问题，在查询一张表或多张表的联合查询时有时反应时间会比较长，这使得用户难以忍受。针对这种问题，可以通过下述方法来分析：

- 在查询的连接或条件子句中的相关字段是否加了索引。 (关于 SQL 的优化可以参见 SQL 优化相关文章，本文不再赘述) 。
- 察看缓冲池的大小，缓冲池太小会造成很多数据不能读到缓冲池而直接从硬盘上读取，造成很大的瓶颈。另一方面关于缓冲池预取的设置，一般能将预取大小 (PREFETCHSIZE) 设定为区段大小与容器个数的积，这样可以最大利用到预取的并行性。
- 在查询中涉及到 order by 字句时，如果排序的字段没有设置索引那么排序将会用到内存中的排序堆 (sortheap) 。如果排序堆过小会造成排序溢出到硬盘上 (Overflowed) 造成性能衰退。
- 同时还要考虑到 RUNSTATS/REORG 因素。 RUNSTATS 命令可以更新表中的统计信息。当表中的数据经过频繁的增删改后其相应的统计信息会发生变化，而优化器选择执行计划的时候是根据这种统计信息来计算的，所以运行 RUNSTATS 此时显得尤为重要。 REORG 可以整理数据存储的物理结构，也能减少数据扫描的时间，提高查询的性能。
- 从存储方面应当注意的是选取裸设备的 DMS 要比 SMS 性能要好，因为它少了一层文件系统的缓冲而直接访问缓冲池。
- 学会使用 optimize for n rows 子句，它可以提高前面 n 条记录的显示速度。这样可以使用户能够先快速查看这 n 条记录，然后再看其他纪录。减少了用户的等待时间。
- 物化查询表 (MQT) 也是提高查询性能的一种手段，它可以将经常用到的查询结果集存储到一张中间表中，在查询时减少了数据检索的时间。
- 在架构上采用 MPP 或 SMP 也是提高查询或写操作性能的手段。
- 针对复杂查询时可以将数据库配置参数 DFT_QUERYOPT(缺省查询优化类) 的值设得高一些（7 或 9），针对简单查询可以将它设得低一些 (3 或 5)，因为设置越高优化器所作的分析就越深入，耗费在生成计划上的时间就越多。
- 针对 C/S 结构的查询可以将查询语句写在服务器端生成存储过程来减少数据的网络传输以及客户端的压力。而经过编译的存储过程执行得更加高效。
- 还要考虑到隔离级别与锁的因素，隔离级别越高越能保证数据的完整性，但同时会减弱并发性。这一点

需要权衡需求而定。
- 网络因素也不可忽视，将数据库服务器参数 RQRIOBLK 设为 65534 可以相应地提高网络吞吐量。（缺省值 32767）
- 最后需要考虑的是数据库的结构，在某些情况下，在某些表中增加一些冗余字段虽然牺牲了一些空间和维护成本，但是在查询时可以减少很多连接操作，这样可以大大提高查询性能。就是用空间换取时间。

接下来看一下增删改的性能优化方法：

- 首先是索引因素，在做增删改时数据库会对表中的索引做相应的修改。这会消耗一定的资源，所以在保证数据完整性的前提下可以先将索引删除，待到增删改结束后再重建这些索引。这也会节省一些时间。将索引和数据放在不同的硬盘上也可以增加写操作的并行性。
- 其次要考虑日志因素，在数据写操作的同时，数据库系统也在维护着事务日志，所以应尽量减少日志维护的代价。将 auto commit 设为 false，可以减少提交的次数（同时也减少了写日志的次数）。增大 LOGBUFSZ,LOGFILSZ 可以减少刷新日志的次数以及日志文件切换的次数。或者将表的属性改为" ACTIVATE NOT LOGGED INITIALLY ", 这样可以屏蔽表的日志操作，以提高写操作的性能，但是失去事务日志的表的数据很难修复，这一点需要权衡。
- 将日志和数据分别放在不同的硬盘上也可以增加写操作的并行性。
- 在插入记录时采用 APPEND MODE 可以消除 DB2 寻找表中间的空余空间的时间而直接插到表尾，从而提高插入的性能。
- 关于并行性的因素，采用 MPP 模式可以使用并行处理的方式增加写操作的性能。将容器分散在不同的硬盘上也可以增加写操作的性能。
- 还要考虑到约束和触发器的影响，在写操作时应当尽量避免表中有约束和触发器。在保证数据完整性的前提下可在频繁大批量写操作时先将约束或触发器去除，完毕后重建。
- 和查询一样，写操作同样要考虑到隔离级别和锁的因素（参见查询优化部分）。
- 在 insert 语句中包括多行可以减少客户机 - 服务器通信次数，提高插入性能。如：insert into table1 values (1, ' a '),(2, ' b '),(3, ' c ') 。
- 还有一个需要考虑的因素是 DB2 V95 在 UNIX 上的采用线程模型，在操作系统中的开销变小，使得写操作性能要比之前的 DB2 的版本要好。

DB2 实用程序的性能优化

先来看一下如何提高备份操作的性能：

- 提高数据库配置参数 UTIL_HEAP_SZ 的大小，这个内存区域用来为备份和恢复操作提供缓冲。
- 减少整库备份，多采用表空间备份需要的表空间。
- 减少完全备份，多采用增量备份或 DELTA 备份。
- 增加备份命令中的 PARALLELISM 参数来增加备份的并行性（增加线程或进程）。
- 增加备份命令中的 BUFFER 参数值。
- 增加备份的目标目录，最好能将多个目录放在不同的硬盘上，这样可以增加备份的并行程度。

再来看一下如何提高恢复操作的性能：

- 和备份操作一样，需要增大数据库配置参数 UTIL_HEAP_SZ 的大小。
- 增加恢复命令中的 BUFFERS 参数值。
- 增加恢复命令中的 PARALLELISM 参数来增加备份的并行性（增加线程或进程）。
- 容器分布于不同的硬盘上也可以使恢复操作加快（提高并行性）。
- 采用 SMP 模式来激活多代理来增加恢复操作的并行性。

提高导入操作（import）的性能 :

- import 操作类似 insert 操作，因此很多方法可以参见 insert 的调优步骤。
- 添加 compound=x 选项可使导入操作批量进行而减少了网络的通信量。
- 增加 COMMITCOUNT 的值已减少 LOG 的 I/O 次数。
- 启用缓冲区插入，对 db2uimpm 程序包使用 INSERT BUF 选项重新绑定到数据库。在 import 以前执行命令: db2 bind db2uimpm.bnd insert buf

提高导出操作 (export) 的性能：

- Export 操作类似 select 操作，因此很多方法可以参见 select 的调优步骤。
- 将 export 操作导出的文件放在与数据和日志不同的硬盘上以减少 I/O 的竞争。

提高载入操作 (load) 的性能：load 操作中日志的写操作比 import 要少，所以 load 的性能比 import 要好很多，下面还是看看如何更好地提高 load 的性能。

- 在多分区环境下，db2 load 会进行并行装载，性能会大幅度提高。
- 添加 buffer 参数可以增加装载过程中的缓存空间，提高性能。

并发连接时的性能考虑

一般来说在连接数较少情况下，db2 的性能会比较稳定。因为这时连接的应用所产生的请求比 db2 代理池中所能产生的协调代理少，这时基本上能够满足每一个请求都能够被及时的协调代理所响应处理。 在连接集中器激活（MAX_CONNECTIONS > MAX_COORDAGENTS）的情况下，如果连接数超过了协调代理，这时连接所过来的请求就会进入队列等候协调代理服务，并发的连接数提高了，但是某些连接的性能就会显著下降。此时应当考虑激活分区间并行 (SMP) 或多分区（MPP）特性来增加 I/O 的并行性以及多个 CPU 的并行运算。

案例分析

查询优化案例

接下来这里从一个试验来看一下 DML 操作过程中优化的详细步骤和具体数据。首先看一个查询优化的例子，下面是试验中的建表语句：

```
CREATE TABLE MCLAIM.T1_DMS (
C11 VARCHAR (10)  NOT NULL ,
 C12 VARCHAR (15)  NOT NULL  ,
 C13 VARCHAR (20)  NOT NULL  ,
 CONSTRAINT C11_PK PRIMARY KEY ( C11)  ) IN DMS_Space;

CREATE TABLE MCLAIM.T2_DMS (
C21 VARCHAR (15)  NOT NULL ,
C22 VARCHAR (25)  NOT NULL  ,
C23 VARCHAR (30)  NOT NULL  ,
CONSTRAINT C21_PK PRIMARY KEY ( C21)  ) IN DMS_Space;

CREATE TABLE MCLAIM.T3_DMS (
C31 VARCHAR (10)  NOT NULL ,
C32 VARCHAR (25)  NOT NULL  ,
C33 VARCHAR (35)  NOT NULL  ,
CONSTRAINT C31_PK PRIMARY KEY ( C31)  ) IN DMS_Space;
```

最初的环境没有优化，表空间类型 SMS 表空间，查询的表中没有索引，sortheap 过小等等。在这种情况下执行下列查询语句：

```
select C12 from TESTOPT.T1_SMS,%SCHEMA%.T2_SMS,%SCHEMA%.T3_SMS
where substr(C12,1,10)=substr(C21,1,10) and C22=C32
order by C12 asc
```

在没有优化的情况下得到的总的执行时间是 653 秒，而经过优化后得到总的执行时间是大概是 15 秒左右。在优化中采用了如下优化步骤：

1. 选择 DMS 表空间。
2. 添加索引：

```
CREATE UNIQUE INDEX INDEX_C12 on T1_DMS (C12 ASC);
CREATE UNIQUE INDEX INDEX_C22 on T2_DMS (C22 ASC);
```

```
CREATE UNIQUE INDEX INDEX_C32 on T2 _DMS (C32 ASC);
```

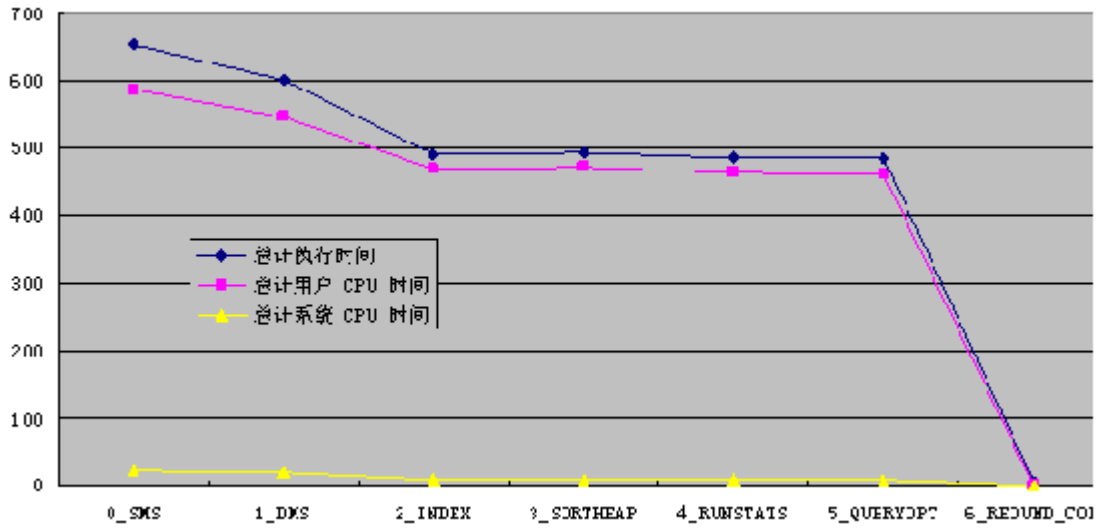
3. 增大 sortheap 的大小
4. 执行 runstats
5. 选择适当的优化级别
6. 改进表结构，增加冗余字段。以空间换时间：

```
ALTER TABLE T1 ADD C12_Red VARCHAR(10);
ALTER TABLE T2 ADD C21_Red VARCHAR(10);
UPDATE T1 SET C12_Red=SUBSTR(C12,1,10);
UPDATE T2 SET C21_Red=SUBSTR(C21,1,10);
```

查询语句变成：

```
select C12 from TESTOPT.T1_DMS, TESTOPT.T2_DMS, TESTOPT.T3_DMS
where C12_Red=C21_Red and C22=C32 order by C12 asc
```

图 1. 查询操作优化示意图



从图中可以看出选择好的表空间类型 (数据库管理表空间) 和添加索引会对性能有很大的改善作用。而添加冗余字段对性能的改进作用最大。当然这会涉及表结构的变化，是需要在数据库设计阶段考虑的因素。同时代价是增加磁盘的占用空间。

写入操作优化

接下来是一个写操作的例子（插入）。下面是试验的脚本：

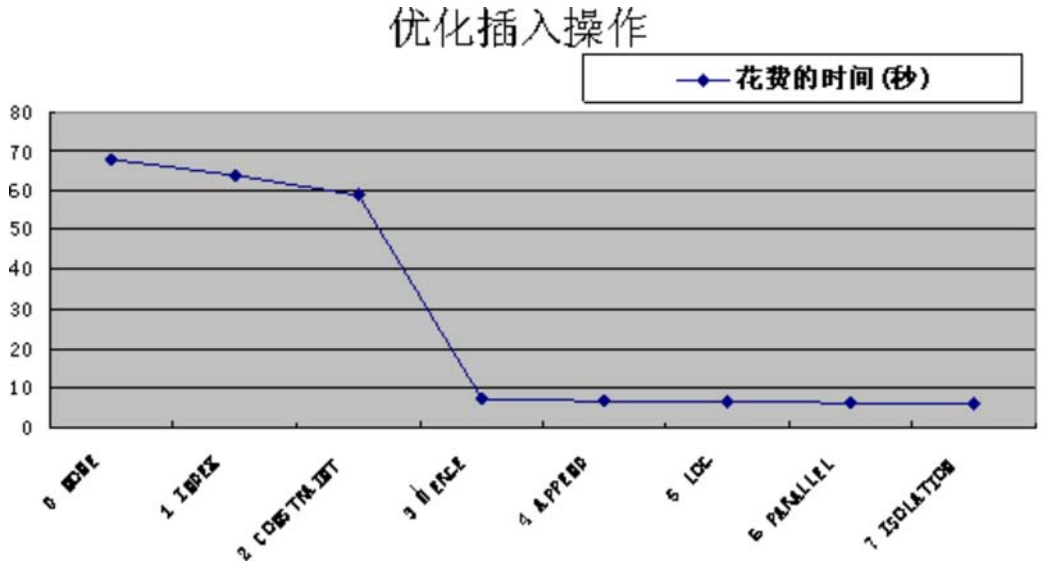
```
 CONNECT TO FFTEST;
 CREATE SCHEMA TESTOPT;
 DROP TABLE TESTOPT.T3;
CREATE TABLE TESTOPT.T3 (
C31 VARCHAR (10)  NOT NULL ,
C32 VARCHAR (15)  NOT NULL  ,
 CONSTRAINT C31_A CHECK ( C31 LIKE 'A%' or C31 LIKE 'a%'));

 CREATE INDEX TESTOPT.INDEX_C31 on TESTOPT.T3 (C31 ASC);
 ALTER TABLE TESTOPT.T3 ADD CONSTRAINT C31_A CHECK (substr(C31,1,1)= ' a '
   or substr(C31,1,1)= ' A ' )
```

```
ALTER TABLE TESTOPT.T3 APPEND OFF;
CONNECT RESET;
```

最初的表没有优化，含有索引，约束等因素，插入 4 万条记录大约花了 68 秒钟，而最终优化后插入 4 万条记录只需 6 秒钟。如下是优化步骤：

1. 去除索引。
2. 去除约束。
3. 在 insert 语句中包括多行。
4. 采用 Append 模式
5. 屏蔽表的日志操作。
6. 采用并行写操作。
7. 采用严格的隔离级别。

图 2. 插入操作优化示意图



从图中可以看出减少索引和约束可以大幅度提高插入性能，而将多条插入语句合并成一行产生的效果更加明显。

性能调优注意事项

- 为了得到高性能将缓冲池调得过大，导致数据库连不上。这对没有经验的用户来说可能是个灾难，这意味着数据库可能要重建。最初我们曾经犯过这样的错误。现在可以通过调节 DB2 注册参数 DB2_OVERRIDE_BPF 来设置缓冲池的大小，从而能够再次连接数据库。当然最好将 STMM 激活，使内存能够自动调整。
- 往往忽视 runstats 和 reorg 的作用，我们发现不止一个的性能问题，都是由于优化器选择了错误的 access plan 导致系统整体性能下降。而对外显示的则不光是 SQL 执行慢，同时也能会表现出 I/O 瓶颈或系统响应时间长。这往往会误导我们去分析其他地方。但究其根源，很多时间是由于优化器的错误。这些问题往往在重新执行 runstats 和 reorg 之后就解决了。所以这两个命令也要特别注意。
- 在进行数据加载的时候往往忽略了索引因素，导致性能加载性能下降。我们遇到过这样的一个例子，一张表导入 1000 条记录花了 5 分钟，检查了很多配置找不到原因，最后发现这张表上有 1 个主键，还有 4 个外键。将他们删除后重新导入只花了几秒钟。所以在进行 load 或者是 insert 的时候尽量将主外键或相关索引删除，加载完成后重建相关索引。主外键尽量通过加载程序来保证它的数据完整性。这一点往往会被忽略，所以在加载数据前先检查一下所有表的索引状态及引用关系。
- 在修改 db2 参数的时候，一次最好修改一个参数，然后看看效果，在调节其他参数。否则一次多个参数，调好了也没弄清楚是哪个参数起的作用。下次还得全部来一遍。还要注意，并非所有参数都是越大

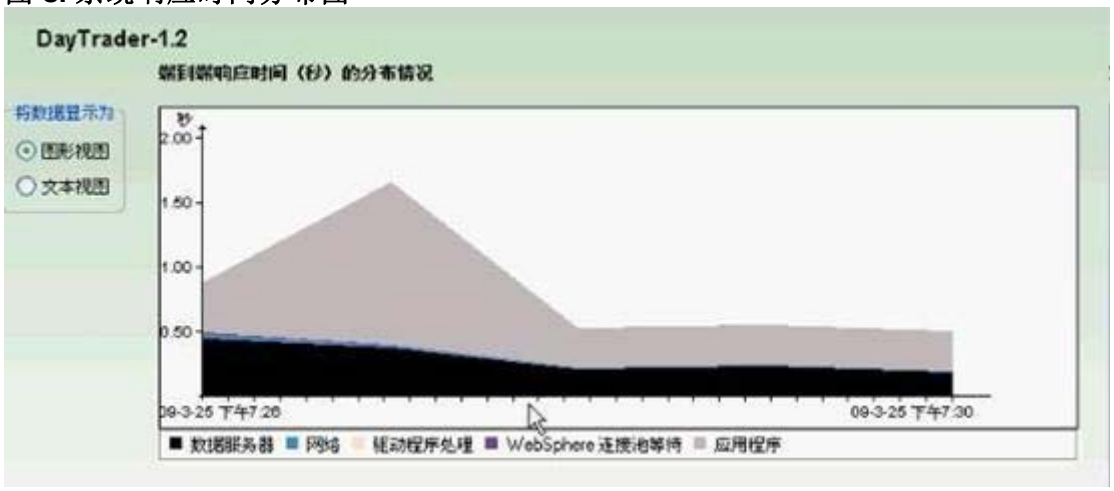
越好，有时可能会适得其反。

- 注意索引的试用，优化好的索引对查询语句性能的提高往往会产生数十倍的性能改进。所以，调优前可以先察看一下相关语句的索引利用情况。这可以通过察看 SQL 语句和执行计划，看一下已有索引是否被利用起来了或是否需要建立新的索引。这往往比 DB2 系统调优更重要。但切记考虑插入操作，索引也会降低插入的性能。这一点要综合考虑。
- 由于 XML 数据可以跨页存储，在设计 XML 数据库时要尽可能的使用较大的数据页，这样可以避免 XML 数据跨页查询，以提高查询性能。
- 采用表分区：有这样一个例子：客户有一张表的数据量非常大，每天都会产生大约 30 万条记录，同时每天都会删除五天前的记录，所以此表大概有 150 万条记录，现在客户在每天的第一次查询时要重新对表进行索引（因为晚上会产生很多数据，所以新增加的数据都没有建索引），导致响应非常慢！对于这种问题，后来采用了表分区，用 6 个分区表来分别装载原来 6 天的数据。所以查询和插入都只涉及一张表，所以响应速度得到大幅度提高。
- 了解 CHNGPGS_THRESH 参数，是缓冲池写日志的阀值。有一个例子，在创建索引时比较慢，经过检查发现 CHNGPGS_THRESH 参数过大，造成每次写日志的时候数据量过大，造成 I/O 瓶颈，适当减小这个参数值，可以增加写日志的次数，但数减少每次写日志的数据量，这对于大缓冲池里的大表上创建索引时很有效的。
- 在导入数据时尽量采用 load, 少用 import, 我们做过统计，用 import 花费 10 分钟的数据，用 load 大概只需要 1 分钟，这大大提高了工作效率。
- 注意 db2diag.log 的大小，当这个文件很大的时候，数据库的所有操作，包括停启 db2 都会特别的慢，有时甚至挂起。所以要经常看看这个文件的大小，过大时最好删掉，重启 db2 。当然 DIAGLEVEL 不要设得太高，除非为了诊断某个问题获得更多信息，一般默认的 3 足够了。

复杂的应用环境中的性能优化 (DB2 Performance Expert)

现在的生产环境都是非常复杂的，性能问题涉及到了应用程序，应用服务器，数据库，网络等各种因素。要从复杂的环境中迅速定位性能的瓶颈非常困难。下面介绍一个非常有用的工具可以帮助用户解决这个难题。这个工具就是 IBM 的 DB2 Performance Expert 。运用 DB2 Performance Expert V3.2 可以很快的找到系统的性能瓶颈。如下图所示：

图 3. 系统响应时间分布图



从这个截图可以看出目前应用程序（灰色部分）和数据库（黑色部分）占用了很大的比例，是系统瓶颈所在。而下图则详细描述了数据库的一些状态信息。

图 4. 数据库重要指标信息图

从上图可以看出，排序溢出很大 (到了 100%)，说明 sortheap 需要调整，缓冲池 IBMDEFAULTBP 的命中率很低（只有 69.5%），说明缓冲池太小，需要调大。所以在分析系统性能问题是使用 DB2 Performance Expert 是一个不错的选择。

结束语

通过本文对 DB2 的各项性能要素的分析以及性能工具的介绍，希望能使读者在分析性能问题时有一个清晰的脉络，从而由潜入深地去解决相关的性能问题。也希望初学者能够以此作为入门的台阶，开始深入系统地学习 DB2 相关的领域。

免责声明

本文仅代表作者的个人观点，不代表 IBM 公司的立场。

参考资料

学习

- 通过访问 DB2 9 技术资源中心 ，查看 DB2 9 相关技术文章和教程的资源。
- 通过访问 DB2 Express-C 产品专题，了解 DB2 Express-C 相关的产品和文档、教程等技术资源。
- 通过访问 DB2 V9.5 信息中心 ，了解 DB2 V9.5 的新特性概述和调优基本方法。
- 通过访问 DB2 V9.1 信息中心 ， 了解 DB2 V9.1 的新特性概述和调优基本方法。
- 通过访问 DB2 工具信息中心 ， 了解 DB2 相关工具的新特性概述和使用方法。
- 通过访问 理解与优化 DB2 的代理和连接 ，了解 DB2 Universal Database（DB2 UDB）代理的工作原理以及连接集中器的特性。

获得产品和技术

- 现在可以免费使用 DB2 。下载 DB2 Express-C，这是为社区提供的 DB2 Express Edition 的免费版本，它提供了与 DB2 Express Edition 相同的核心数据特性，为构建和部署应用程序奠定了坚实的基础。
- 下载 DB2 Enterprise V9.5 试用版，试用本文中描述的特性。
- 下载 信息管理软件试用版，体验它们强大的功能。

讨论

- 参与 developerWorks blog 并加入 developerWorks 社区。

作者简介

李越 IBM 中国软件开发中心 WebSphere Federation Server 测试部门软件工程师。曾在 developerWorks 中国发表过有关优化 DB2 的代理和连接的文章。

王飞鹏，IBM 中国软件开发中心的软件工程师，主要从事 DB2 Everyplace 相关的研发和客户支持工作；曾在 developerWorks 中国发布了多篇与 DB2 Everyplace 产品相关的技术文章 。

狄浩，IBM 中国软件开发中心软件工程师，主要从事 IBM CM 内容管理产品的相关工作，最近对 DB2 的性能调优比较感兴趣。

张蓉蓉，IBM 中国软件开发中心软件工程师。