IBM Software Group

DB2通用数据库性能调整的常用方法

DB2 Information Management Software

陈德福
IM, SWG, IBM China
chendefu@cn.ibm.com
010-65391188-3406

e business software

Agenda

- 统计值更新 -- runstats
- 调整Buffer pool
- 调整日志缓冲区大小
- 应用程序堆大小
- 排序堆大小和排序堆值
- 代理程序的数目
- 锁
- 活动应用程序的最大数目
- 页清除程序的数目
- I/O服务器的数目
- 编入组的提交数目

统计信息更新

- 当对 SQL 查询进行优化时，SQL 编译器所做出的决定会受到优化器的数据库内容模型的重大影响。
- 优化器使用该数据模型来估计可以用于解决某个特定查询的其它存取路径的成本。
- 数据模型中的关键元素是一组统计信息，该统计信息收集了有关数据库中所包含的数据和系统目录表中所存储的数据的信息。这包括表、别名（nickname）、索引、列和用户定义的函数（UDF）的统计信息。
- 优化器根据这些统计信息决定最有效方法访问数据的方法，数据统计信息中的变化会引起对存取方案的选择发生变化。
- 适时更新数据统计信息。

统计信息更新

- 适时更新数据统计信息。
  - 当向表装入数据并创建了合适的索引时。
  - 当用 REORG 实用程序重新组织表时。
  - 当存在大量影响表及其索引的更新、删除和插入操作时。（此处的“大量”可能意味着 10% 到 20% 的表和索引数据都受到了影响。）
  - 在绑定性能至关重要的应用程序之前。
  - 当预取数量发生变化时。
- 只有当进行显式的请求时，对象的统计信息才会在系统目录表中被更新。更新部分或全部统计信息方法：
  - 使用 RUNSTATS（运行统计信息，run statistics）实用程序。
  - 使用“reorgchk update statistics”命令。
- 在使用 RUNSTATS 之后需要重新绑定使用静态 SQL 的应用程序，使查询优化器就可以选择新统计信息所给出的最佳存取方案。但是，对于使用动态 SQL 的应用程序而言，没必要进行重新绑定，因为语句的优化是根据统计信息在运行时进行的。

reorgchk update statistics

- 当您不完全知道所有表名或表名实在太多时，进行 RUNSTATS 的最简单方法就是使用“db2 reorgchk update statistics”命令。

正确的脚本如下：

```
db2 -v connect to DB_NAME
db2 -v "select tbname, nleaf, nlevels, stats_time from sysibm.sysindexes“
db2 -v reorgchk update statistics on table all
db2 -v "select tbname, nleaf, nlevels, stats_time from sysibm.sysindexes“
db2 -v terminate
```

Runstats实用程序

- 如果您知道表名并且想避免对大量表运行 RUNSTATS 实用程序（因为这样做可能要花很长时间），那么一次对一张表进行 RUNSTATS 更为可取。
- 命令如下：
  db2 -v runstats on table *TAB_NAME* and indexes all
- 这个命令将收集该表及其所有索引（基本级别）的统计信息。
- 要查看是否对数据库执行了 RUNSTATS，一种快捷方法便是查询一些系统目录表。例如:
  db2 -v "select tbname, nleaf, nlevels, stats_time from sysibm.sysindexes"

调整Bufferpool大小

- 缓冲池是内存中的一块存储区域，用于临时读入和更改数据页（含表行或索引项）。缓冲池的用途是为了提高数据库系统的性能。
- 缺省情况下，应用程序使用缓冲池 IBMDEFAULTBP，它是在创建数据库时创建的。当 SYSCAT.BUFFERPOOLS 目录表中该缓冲池的 NPAGES 值为 -1 时，DB2 数据库配置参数 BUFFPAGE 控制着缓冲池的大小 。
- 可以增大所有数据库的 BUFFPAGE 值。
  db2 -v connect to DB_NAME
  db2 -v select * from syscat.bufferpools
  db2 -v alter bufferpool IBMDEFAULTBP size -1
  db2 -v connect reset
  db2 -v update db cfg for dbname using BUFFPAGE bigger_value
  db2 -v terminate
- 要确定数据库的缓冲池大小是否由 BUFFPAGE 参数所决定，运行：
  db2 -v connect to DB_NAME
  db2 -v SELECT * from SYSCAT.BUFFERPOOLS
  db2 -v connect reset
  db2 -v terminate
  检查结果。如果每个缓冲池都有一个为“-1”的 NPAGES 值，那么缓冲池大小是由数据库配置中的 BUFFPAGE 参数控制的。

调整Bufferpool大小（续）

- 要确定缓冲池大小是否足够大，请在运行应用程序时收集数据库和 / 或缓冲池的快照。下面的脚本为您提供这些所需的信息：
  db2 -v update monitor switches using bufferpool on
  db2 -v get monitor switches
  -- 运行程序 –
  db2 -v get snapshot for all databases > snap.out
  db2 -v get snapshot for dbm >> snap.out
  db2 -v get snapshot for all bufferpools >> snap.out
  db2 -v reset monitor all
  db2 -v terminate
- 请确保您在断开数据库连接之前发出“db2 -v get snapshot”，否则，该数据库停止运行，同时所有快照统计信息将会丢失。

调整Bufferpool大小（续）

- 在数据库快照或缓冲池快照的快照输出中，查找下列“logical reads”和“physical reads”，这样就可以计算出缓冲池命中率，它可以帮助您调优缓冲池：
  -- Related lines from a sample of bufferpool snapshots –
  Buffer pool data logical reads = 702033
  Buffer pool data physical reads = 0
  Buffer pool data writes = 414
  Buffer pool index logical reads = 168255
  Buffer pool index physical reads = 0
- 按如下计算缓冲池命中率：
  (1 - ((buffer pool data physical reads + buffer pool index physical reads) /(buffer pool data logical reads + pool index logical reads))) * 100%
  这个计算考虑了缓冲池高速缓存的所有页（索引和数据）。理想情况下，该比率应当超过 95%，并尽可能接近 100%。
- 要提高缓冲池命中率，请尝试下面这些方法：
  - 增加缓冲池大小。 如果您的机器上有足够大的内存，请将 BUFFPAGE 设置成 40000 个页（160 MB），或者等于机器总内存的 10%。对于大型 OLTP 数据库，在保持系统稳定的同时为缓冲池留出尽可能多的内存。可以先尝试使用 1.6 GB 的内存，然后尝试用更多内存。
  - 考虑分配多个缓冲池，如果可能的话，为每个经常被访问的大表所属的表空间分配一个缓冲池，为一组小表分配一个缓冲池，然后尝试一下使用不同大小的缓冲池以查看哪种组合会提供最佳性能。 表空间的页大小与缓冲池的页大小一样。

调整Bufferpool大小（续）

- create db testdb
  -- using codeset iso8859-1 territory US
  on /data/misserver/dbimages
  collate using identity
  dft_extent_sz 8
  catalog tablespace managed by database
  using (device '/dev/rds01' 131072,
  device '/dev/rds02' 131072,
  device '/dev/rds03' 131072)
  extentsize 32 prefetchsize 96
  ;
  connect to testdb;
  create bufferpool databp size 25000 pagesize 32768;
  create bufferpool bp8k size 16 pagesize 8192;
  disconnect all;

调整Bufferpool大小（续）

- create tablespace datatblsp pagesize 32K
  managed by database
  using (
  device '/dev/rdata0' 30720M ,
  device '/dev/rdata1' 30720M ,
)
  extentsize 8 prefetchsize 48
  bufferpool databp;

- create tablespace idxtblsp pagesize 32K
  managed by database
  using (
  device '/dev/ridxa0' 6144M ,
  device '/dev/ridxa1' 6144M ,
)
  extentsize 8 prefetchsize 48
  bufferpool databp;

调整日志缓冲区

- LOGBUFSZ 是一个数据库配置参数。它是用于日志缓冲区的参数。它允许您指定数据库共享内存的大小以用作在将日志记录写到磁盘之前这些记录的缓冲区。
- 当下列事件之一发生时会将日志记录写到磁盘：
  - 事务提交。
  - 日志缓冲区已满。
  - 其它某个内部数据库管理器事件发生时。
- 将日志记录存在缓冲区将产生更加有效的日志文件 I/O。如果对专用的日志磁盘有相当多的读操作，或者希望有较高的磁盘利用率，那么可以增加这个缓冲区的大小。
- 当增加这个参数的值时，也要考虑 DBHEAP 参数，日志缓冲区使用的空间由 DBHEAP 参数所控制。

调整日志缓冲区

- LOGBUFSZ 是一个数据库配置参数。它是用于日志缓冲区的参数。它允许您指定数据库共享内存的大小以用作在将日志记录写到磁盘之前这些记录的缓冲区。
- 当下列事件之一发生时会将日志记录写到磁盘：
  - 事务提交。
  - 日志缓冲区已满。
  - 其它某个内部数据库管理器事件发生时。
- 将日志记录存在缓冲区将产生更加有效的日志文件 I/O。如果对专用的日志磁盘有相当多的读操作，或者希望有较高的磁盘利用率，那么可以增加这个缓冲区的大小。
- 当增加这个参数的值时，也要考虑 DBHEAP 参数，日志缓冲区使用的空间由 DBHEAP 参数所控制。

调整日志缓冲区（续）

- LOGBUFSZ的缺省值为 8（4KB 页），这对于 OLTP 数据库而言通常不够大。LOGBUFSZ 的最佳值为 128 个或 256 个 4KB 页。
- 可以使用下面这个命令来更改该参数值：
  db2 -v update database cfg for DB_NAME using LOGBUFSZ 256
  db2 -v terminate
- 使用数据库快照来确定 LOGBUFSZ 参数的值是否为最佳值：
  Log pages read = 0
  Log pages written = 12644
- 一般而言，“log pages read”和“log pages written”之比应当尽可能小。理想情况下，“log pages read”的值应为 0，而“log pages written”的值应很大。当 log pages read 太多时，意味着需要一个较大的 LOGBUFSZ。

调整应用程序堆大小（APPHEAPSZ）

- APPHEAPSZ 是一个数据库配置参数，它定义了代表某个特定代理程序或子代理程序的数据库管理器可以使用的私有内存页数。
- 在为应用程序初始化代理程序或子代理程序时分配堆。
- 分配的堆大小是处理给予代理程序或子代理程序的请求所需的最小值。当代理程序或子代理程序需要更多的堆空间以处理较大的 SQL 语句时，数据库管理器将按照需要分配内存，所分配的内存大小最大可达到该参数所指定的最大值。
- 缺省值是128 个 4KB 页，更改命令是：
  db2 -v update db cfg for DB_NAME using applheapsz 256
  db2 -v terminate
- 当应用程序接收到一个表明应用程序堆中存储空间不够的错误时，应该增加 APPHEAPSZ 的值。

排序堆大小（SORTHEAP）和排序堆阈值（SHEAPTHRES）

- SORTHEAP 是一个数据库配置参数，它定义了私有排序所使用的私有内存页的最大数目，或共享排序所使用的共享内存页的最大数目。如果排序是私有排序，那么该参数影响代理程序私有内存。如果排序是共享排序，那么该参数影响数据库的共享内存。每个排序都有单独的由数据库管理器按需分配的排序堆，在排序堆中对数据进行排序。如果由优化器来指导排序堆大小的分配，那么用优化器提供的信息来分配的排序堆的大小要小于由该参数所指定的排序堆大小。
- SHEAPTHRES 是一个数据库管理器配置参数。私有和共享排序所使用内存的来源不一样。共享排序内存区的大小是在第一次连接到数据库时根据 SHEAPTHRES 值以静态方式预先确定的。私有排序内存区的大小是不受限制的。
- 对于私有排序和共享排序，应用 SHEAPTHRES 参数的方式不同：
  - 对于私有排序，SHEAPTHRES 是对私有排序在任何给定的时间可以消耗的全部内存的实例级“软”限制。当实例的总私有排序内存消耗量达到这一限制时，为其它进入的私有排序请求而分配的内存会大大减少。
  - 对于共享排序，SHEAPTHRES 是对共享排序在任何给定的时间可以消耗的全部内存的数据库级“硬”限制。当达到这一限制时，不允许有其它共享排序内存请求，直到总的共享内存消耗量回落到 SHEAPTHRES 所指定的限制以下。

排序堆大小（SORTHEAP）和排序堆阈值（SHEAPTHRES）

- 要更改 SORTHEAP 和 SHEAPTHRES 的值，可以运行如下命令：

  db2 -v update db cfg for DB_NAME using SORTHEAP a_value
  db2 -v update dbm cfg using SHEAPTHRES b_value
  db2 -v terminate

- SORTHEAP是针对单个数据库的，SHEAPTHRES是针对数据库例程的。

排序堆大小（SORTHEAP）和排序堆阈值（SHEAPTHRES）

- OLTP 应用程序不应该执行大型排序，通常，SORTHEAP 大小的缺省值（256 个 4KB 页）就足够了。事实上，对于高并发性 OLTP，您可能希望降低这个缺省值。
- 当需要进一步研究时，可以发出下面这条命令：
  db2 -v update monitor switches using sort on
  然后，让您的应用程序运行一会，然后输入：
  db2 -v get snapshot for database on DBNAME
- 看一下下面这个示例中的输出：
  Total sort heap allocated = 0
  Total sorts = 1
  Total sort time (ms) = 0
  Sort overflows = 0
  Active sorts = 0
  Commit statements attempted = 1
  Rollback statements attempted = 0
  Dynamic statements attempted = 4
  Static statements attempted = 1
  Binds/precompiles attempted = 0

排序堆大小（SORTHEAP）和排序堆阈值（SHEAPTHRES）

- 根据该输出，计算每个事务的排序数目、计算溢出了可用于排序的内存的那部分排序的百分比。
  SortsPerTransaction= (Total Sorts) / (Commit statements attempted + Rollback statements attempted)
  PercentSortOverflow= (Sort overflows * 100) / (Total sorts)
- 如果 SortsPerTransaction 大于 5，它可能表明每个事务的排序太多。如果 PercentSortOverflow 大于 3%，那么可能发生了严重的、未曾预料到的大型排序。发生这种情况时，增加 SORTHEAP 只会隐藏性能问题 — 却无法修正它。这个问题的正确解决方案是通过添加正确的索引改进有问题的 SQL 语句的存取方案。

排序堆大小（SORTHEAP）和排序堆阈值（SHEAPTHRES）

建议：

- 使用合适的索引使排序堆的使用降到最低。
- 当需要频繁进行大型排序时，增加 SORTHEAP 的值。
- 如果增加 SORTHEAP，请确定是否还需要调整数据库管理器配置文件中的 SHEAPTHRES 参数。
- 优化器用排序堆大小来确定存取路径。在更改该参数后请考虑重新绑定应用程序（使用 REBIND PACKAGE 命令）。
- 理想情况下，应当将排序堆阈值（SHEAPTHRES）参数合理地设置为在数据库管理器实例中设置的 SORTHEAP 参数最大值的倍数。该参数至少应当是实例中任何数据库所定义的最大 SORTHEAP 的两倍。

代理程序的数目（MAXAGENTS、NUM_POOLAGENTS 和 NUM_INITAGENTS）

- 这些是数据库管理器配置参数。
- MAXAGENTS 参数表明在任何给定时间接受应用程序请求的数据库管理器代理程序的最大数目。MAXAGENTS 的值应当至少是每个被并发地访问的数据库中的 MAXAPPLS（并发应用程序最大数目）值的总和。如果数据库的数量大于 NUMDB 参数，那么最安全的方案就是使用 NUMDB 和 MAXAPPLS 最大值的乘积。
- NUM_POOLAGENTS 参数是用于评定您希望代理程序池增加到多大的准则。如果所创建的代理程序多于该参数值所指明的数目，那么当代理程序执行完自己当前的请求后将终止运行而不是返回给代理程序池。如果该参数的值为 0，将按照需要创建代理程序，在代理程序执行完自己当前的请求后终止运行。
- 要避免因在并发连接许多应用程序的 OLTP 环境中频繁创建和终止代理程序而产生的成本，请将 NUM_POOLAGENTS 的值增加到接近 MAXAGENTS 值。
- NUM_INITAGENTS 参数决定空闲代理程序的初始数量，这些代理程序是在 DB2START 时在代理程序池中创建的。指定初始代理程序数目要合适。
- 在大多数情况下，将 MAXAGENTS 和 NUM_POOLAGENTS 的值设置成略微大于并发应用程序连接的最大预计数目。NUM_INITAGENTS 保留为缺省值。

代理程序的数目（MAXAGENTS、NUM_POOLAGENTS 和 NUM_INITAGENTS）

- 更改这些参数的命令：
  db2 -v update dbm cfg using MAXAGENTS a_value
  db2 -v update dbm cfg using NUM_POOLAGENTS b_value
  db2 -v update dbm cfg using NUM_INITAGENTS c_value
  db2 -v terminate

代理程序的数目（MAXAGENTS、NUM_POOLAGENTS 和 NUM_INITAGENTS）

- 在运行期间的任何时候，您都可以使用下面这个命令来获取数据库管理器的快照数据：
  db2 -v get snapshot for dbm
- 看一下下列输出行：
  High water mark for agents registered = 4
  High water mark for agents waiting for a token = 0
  Agents registered = 4Agents waiting for a token = 0
  Idle agents = 0
  Agents assigned from pool = 5
  Agents created from empty pool = 4
  Agents stolen from another application = 0
  High water mark for coordinating agents = 4
  Max agents overflow = 0
- 如果发现“Agents waiting for a token”或“Agents stolen from another application”不等于 0，则可能需要增加 MAXAGENTS 以允许数据库管理器可以使用更多的代理程序。

锁（LOCKLIST、MAXLOCKS 和 LOCKTIMEOUT）

- 这些与锁相关的控制都是数据库配置参数。
- LOCKLIST 表明分配给锁列表的存储容量。每个数据库都有一个锁列表，锁定是数据库管理器用来控制多个应用程序并发访问数据库中数据的机制。行和表都可以被锁定。根据对象是否还持有其它锁，每把锁需要 32 个或 64 个字节的锁列表：
  - 需要 64 个字节来持有某个对象上的锁，在这个对象上，没有持有其它锁。
  - 需要 32 个字节来记录某个对象上的锁，在这个对象上，已经持有一个锁。
- MAXLOCKS 定义了应用程序持有的锁列表的百分比，在数据库管理器执行锁升级之前必须填充该锁列表。当一个应用程序所使用的锁列表百分比达到 MAXLOCKS 时，数据库管理器会升级这些锁，这意味着用表锁代替行锁，从而减少列表中锁的数量。当任何一个应用程序所持有的锁数量达到整个锁列表大小的这个百分比时，对该应用程序所持有的锁进行锁升级。如果用一个表锁替换这些行锁，将不再会超出 MAXLOCKS 值，那么锁升级就会停止。否则，锁升级就会一直进行，直到所持有的锁列表百分比低于 MAXLOCKS。MAXLOCKS 参数乘以 MAXAPPLS 参数不能小于 100。
- 虽然升级过程本身并不用花很多时间，但是锁定整个表（相对于锁定个别行）降低了并发性，而且数据库的整体性能可能会由于对受锁升级影响的表的后续访问而降低。

锁（LOCKLIST、MAXLOCKS 和 LOCKTIMEOUT）

- 使用下列步骤确定锁列表所需的页数：
  - 计算锁列表大小的下限：(512 * 32 * MAXAPPLS) / 4096，其中 512 是每个应用程序平均所含锁数量的估计值，32 是对象（已有一把锁）上每把锁所需的字节数。
  - 计算锁列表大小的上限：(512 * 64 * MAXAPPLS) / 4096，其中 64 是某个对象上第一把锁所需的字节数。
  - 对于您的数据，估计可能具有的并发数，并根据您的预计为锁列表选择一个初始值，该值位于您计算出的上限和下限之间。
- 使用数据库系统监视器调优 MAXLOCKS 值。
- 设置 MAXLOCKS 时，请考虑锁列表的大小（LOCKLIST）：
  - MAXLOCKS = 100 * (512 锁 / 应用程序 * 32 字节 / 锁 * 2) / (LOCKLIST * 4096 字节)
  - 该样本公式允许任何应用程序持有的锁是平均数的两倍。如果只有几个应用程序并发地运行，则可以增大 MAXLOCKS，因为在这些条件下锁列表空间中不会有太多争用。

锁（LOCKLIST、MAXLOCKS 和 LOCKTIMEOUT）

- LOCKTIMEOUT 指定了应用程序为获取锁所等待的秒数。这有助于应用程序避免全局死锁。
- 如果将该参数设置成 0，那么应用程序将不等待获取锁。在这种情形中，如果请求时没有可用的锁，那么应用程序立刻会接收到 -911。
- 如果将该参数设置成 -1，那么将关闭锁超时检测。在这种情形中，应用程序将等待获取锁（如果请求时没有可用的锁），一直到被授予了锁或出现死锁为止。
- 在联机事务处理（OLTP）环境中，这个值从 30 秒开始。在只进行查询的环境中可以从一个更大的值开始。

锁（LOCKLIST、MAXLOCKS 和 LOCKTIMEOUT）

- 更改锁参数的命令：
  db2 -v update db cfg for DB_NAME using LOCKLIST a_number
  db2 -v update db cfg for DB_NAME using MAXLOCKS b_number
  db2 -v update db cfg for DB_NAME using LOCKTIMEOUT c_number
  db2 -v terminate

锁（LOCKLIST、MAXLOCKS 和 LOCKTIMEOUT）

- 使用数据库系统监视器来确定是否发生锁升级，跟踪应用程序（连接）遭遇锁超时的次数，或者数据库检测到的所有已连接应用程序的超时情形。
- 首先，运行下面这个命令以打开针对锁的 DB2 监视器：
  db2 -v update monitor switches using lock ondb2 -v terminate
- 然后收集数据库快照：
  db2 -v get snapshot for database on DB_NAME
- 在快照输出中，检查下列各项：
  Locks held currently = 0
  Lock waits = 0
  Time database waited on locks (ms) = 0
  Lock list memory in use (Bytes) = 504
  Deadlocks detected = 0
  Lock escalations = 0
  Exclusive lock escalations = 0
  Agents currently waiting on locks = 0
  Lock Timeouts = 0
  Internal rollbacks due to deadlock = 0
- 如果“Lock list memory in use (Bytes)”超过定义的 LOCKLIST 大小的 50%，那么就增加 LOCKLIST 的数量。锁升级、锁超时和死锁将表明系统或应用程序中存在某些潜在问题。
- 锁定问题通常表明应用程序中存在一些相当严重的并发性问题，在增大锁列表参数的值之前应当解决这些问题。

活动应用程序的最大数目（MAXAPPLS）

- MAXAPPLS 是一个数据库配置参数。它指定了可以连接到数据库的并发应用程序（本地和远程）的最大数量。
- 该参数值必须大于等于已连接应用程序的数量，加上这些相同的应用程序中完成两阶段提交或回滚过程中可能并发存在的数量的总和。
- 在 OLTP 应用中，请确保将 MAXAPPLS 的值设置正确，以容纳最多的并发用户 / 连接。
- 对于那些使用连接池的应用程序，可以将 MAXAPPLS 的值设置成比连接池的大小大 1 或 2（这样做只是为了以防需要调用命令行连接来同时做一些事情）。

活动应用程序的最大数目（MAXAPPLS）

- 更改 MAXAPPLS 值的命令：
  db2 -v update db cfg for DB_NAME using MAXAPPLS a_number
  db2 -v terminate
- 当应用程序尝试连接数据库，但是连接到数据库的应用程序数已经达到了 MAXAPPLS 的值时，会向应用程序返回下面这个错误，表明连接到该数据库的应用程序数已达到了最大值。
  SQL1040N The maximum number of applications is already connected to thedatabase. SQLSTATE=57030

异步页清除程序的数量（NUM_IOCLEANERS）

- NUM_IOCLEANERS 是一个数据库配置参数，指定数据库的异步页清除程序的数目。在数据库代理程序需要缓冲池中的空间之前，这些页清除程序将缓冲池中已更改的页写到磁盘，这允许代理程序不必等待已更改页被写到磁盘就可以读取新页，提高应用系统性能。
- 如果将该参数设置成 0，则不启动页清除程序，结果，数据库代理程序将缓冲池中的所有页写到磁盘。该参数会对存储在多个物理存储设备上的单个数据库的性能产生显著影响，因为在这种情况下其中某个设备极有可能处于空闲状态。如果没有配置页清除程序，则应用程序可能会遇到不时发生的“日志已满”情况。
- 如果连接到数据库的应用程序主要执行更新数据的事务，那么增加清除程序的数目会提高性能。
- 增加页清除程序的数量还会减少“软”故障（比如断电）的恢复时间，因为磁盘上数据库的内容在任何给定时候都是比较新的。

异步页清除程序的数量（NUM_IOCLEANERS）

- 设置该参数时要考虑的因素：
  - 如果有多个事务针对数据库运行，则将该参数的值设置在 1 到该数据库所使用的物理存储器的数量之间。
  - 至少将该参数的值设置成您系统上 CPU 的数量。
  - 在具有高更新事务率的环境下，可能需要配置较多的页清除程序。
  - 在具有大缓冲池的环境下，也可能需要配置较多的页清除程序。

异步页清除程序的数量（NUM_IOCLEANERS）

- 更改该参数的命令：
  db2 -v update db cfg for DB_NAME using NUM_IOCLEANERS a_number
  db2 -v terminate
- 使用数据库系统监视器，利用有关从缓冲池进行写操作的快照数据（或事件监视器）信息来帮助您调优该配置参数。
- 当使用快照和收集缓冲池的快照数据时，监控下列计数器：
  Buffer pool data writes = 0
  Asynchronous pool data page writes = 0
  Buffer pool index writes = 0
  Asynchronous pool index page writes = 0
  LSN Gap cleaner triggers = 0
  Dirty page steal cleaner triggers = 0
  Dirty page threshold cleaner triggers = 0

异步页清除程序的数量（NUM_IOCLEANERS）

- 如果下面这两个条件成立，*减少* NUM_IOCLEANERS：
  “Buffer pool data writes”约等于“Asynchronous pool data page writes”。
  “Buffer pool index writes”约等于“Asynchronous pool index page writes”。
- 只要下面这两个条件有一个成立，增加 NUM_IOCLEANERS：
  “Buffer pool data writes”远远大于“Asynchronous pool data page writes”。
  “Buffer pool index writes”远远大于“Asynchronous pool index page writes”。
- Dirty page steal cleaner triggers 指出调用页清除程序的次数，为了有更好的响应时间，该数值应当尽可能低。利用上面所示的计数器，可以使用下面的公式计算用该元素表示的所有清除程序调用的百分比：
  Dirty page steal cleaner triggers / (Dirty page steal cleaner triggers +
  Dirty page threshold cleaner triggers +
  LSN Gap cleaner triggers)
  如果该比率很高，则它可能表明您所定义的页清除程序太少了。

I/O 服务器的数目（NUM_IOSERVERS）

- 该参数是一个数据库配置参数，用于指定数据库的 I/O 服务器的数目。
- 诸如备份和恢复之类的实用程序使用 I/O 服务器代表数据库代理程序执行预取 I/O 和异步 I/O。
- 超过这个数量的预取和实用程序 I/O 在任何时候都不能在数据库中运行。在启动 I/O 操作时，I/O 服务器处于等待状态。
- 由于从数据库代理程序直接调度非预取 I/O，因此非预取 I/O 不受 NUM_IOSERVERS 约束。
- 在 OLTP 环境中，使用缺省值就可以。
- 更改 NUM_IOSERVERS参数的命令：
  db2 -v update db cfg for DB_NAME using NUM_IOSERVERS a_number
  db2 -v terminate

编入组的提交数目（MINCOMMIT）

- MINCOMMIT 是数据库配置参数，它让您把将日志记录写到磁盘的工作一直延迟到执行了最小数量的提交为止。
- 该延迟有助于减少与写日志记录相关的数据库管理器开销。这意味着当您针对数据库运行多个应用程序并且在非常短的时间范围内应用程序请求大量提交时可以提高性能。
- 只有当该参数值大于 1 并且当连接到数据库的应用程序数量大于或等于该参数值时，才会发生这个提交分组。
- 当执行提交分组时，应用程序提交请求会被挂起，直到时间过去 1 秒或提交请求的数量等于该参数值。
- 使用下面的命令更改 MINCOMMIT 值：
  db2 -v update db cfg for *DB_NAME* using MINCOMMIT *a_number*
  db2 -v terminate

编入组的提交数目（MINCOMMIT）

- MINCOMMIT 的缺省值为 1。
- 如果多个读 / 写应用程序通常请求并发数据库提交，则从缺省值开始递增该参数值，这将产生更有效率的日志文件 I/O，因为使用日志文件 I/O 的次数比较少，而每次使用日志文件 I/O 时所写的日志记录比较多。
- 如果您认为缺省值不够大，可以从 3 开始进行调整，在 3 的附近尝试以查看性能对工作负载的影响。
- 可以对每秒钟的事务量进行采样，并调整该参数以适应每秒钟的峰值事务量（或者采用它的某个较大的百分比）。适应峰值活动使得在重负载期间写日志记录的开销减到了最低。
- 如果增大 MINCOMMIT，可能还需要增大 LOGBUFSZ 参数以避免在这些重负载期间强制将已满的日志缓冲区写入磁盘。在这种情况下，LOGBUFSZ 应该等于：
  MINCOMMIT * (每个交易平均日志空间的使用量)

编入组的提交数目（MINCOMMIT）

- 计算每秒钟的峰值事务数：
  通过采用典型一天中的监视器样本，可以确定重负载时期。
  1. 在测量开始时，发出下面这个命令：
     - db2 -v reset monitor for database db_name （这不会使高水位的计数器复位。）
  2. 在测量完毕后，发出下面这个命令：
     - db2 -v get snapshot for database on db_name
  3. 使用以下输出来计算事务的峰值：

     Last reset timestamp = 06-12-2001 14:51:43.786876
     Snapshot timestamp = 06-12-2001 14:56:27.787088
     Commit statements attempted = 1011
     Rollback statements attempted = 10
     Log space used by the database (Bytes) = 3990

  4. 让 totalTransactions 等于“commit statements attempted”和“rollback statements attempted”的总和。
  5. 让 totalElapsedTime（单位为秒）等于“Last reset timestamp”和“Snapshot timestamp”的差。
  6. 如下计算每秒事务数：
     - NumOfTransPerSecond = totalTransactions / totalElapsedTime

编入组的提交数目（MINCOMMIT）

- 计算每个事务所使用的日志空间：
- 通过在一段时间内对一些事务使用抽样技术，可以通过下面这个监视器元素：**log_space_used**（所使用的工作日志空间单元）计算出使用的日志空间的平均值。
  1. 在测量开始时使用下面这个命令将感兴趣的数据库的监视器复位：
  db2 -v reset monitor for database db_name.
  2. 在测量完毕后使用下面这个命令获取快照：
  db2 -v get snapshot for database on db2_name.
  3. 使用下面这个公式计算出每个事务所使用的日志空间：
  LogSpaceUsedPerTrans = log_space_used / totalTransactions

DB2 on the Web

- **IBM**
  - http://www.ibm.com
- **IBM Software**
  - http://www.software.ibm.com
- **IBM Data Management: white papers, products**
  - http://www.software.ibm.com/data
- **DB2 Family**
  - http://www.software.ibm.com/data/db2/performance
- **DB2 Family Performance**
  - http://www.software.ibm.com/data/db2/performance
- **DB2 Solutions Directory Applications + Tools**
  - http://www.mfi.com/softwareguide/DB2-Solutions/
- **"Red Books"**
  - http://www.redbooks.ibm.com/redbooks

谢谢 ！

陈德福
010-65391188-3406
chendefu@cn.ibm.com