第 19 讲:日志机制

15-445/645 数据库系统(2022 年秋季)

https://15445.courses.cs.cmu.edu/fall2022/

卡内基梅隆大学安迪·帕夫洛

1 事故恢复

*恢复算法*是一种确保数据库一致性、事务原子性和持久性的技术，尽管发生了故障。当发生崩溃时，内存中所有尚未提交到磁盘的数据都有丢失的风险。恢复算法的作用是防止崩溃后的信息丢失。

每个恢复算法都有两个部分:

- 正常事务处理期间的操作，以确保 DBMS 可以从故障中恢复。
- 将数据库恢复到确保原子性、一致性和持久性的状态失败后的操作。

恢复算法中使用的关键原语是 UNDO 和 REDO。并不是所有的算法都使用这两种原语。

- **UNDO**:删除不完整或终止的事务影响的过程。
- **REDO**:重新应用已提交事务的持久效果的过程。

2 种存储类型

- **易失性存储器**
  - -在电源丢失或程序退出后，数据不持续。
  - -示例:DRAM、SRAM、。
- **非易失性存储**
  - -**在**断电或程序存在后，数据仍然存在。
  - -例如:HDD、SDD。
- **稳定的存储器**
  - **-不存在的非易失性存储形式，可以在所有可能的故障场景中生存。**
  - -使用多个存储设备进行近似。

3 故障分类

由于 DBMS 根据底层存储设备划分为不同的组件，因此 DBMS 需要处理许多不同类型的故障。其中一些故障是可恢复的，而另一些则不能。

类型 1:事务失败

*事务失败*发生在事务出现错误并且必须中止的时候。两种可能导致事务失败的错误类型是逻辑错误和内部状态错误。

- **逻辑错误:**由于某些内部错误条件(例如，完整性，约束违反)，事务无法完成。

- **内部状态错误:**DBMS 必须终止一个活动的事务由于一个错误条件(例如，死锁)

类型#2:系统故障

*系统故障*是承载 DBMS 的底层软件或硬件中的意外故障。这些故障必须在崩溃恢复协议中加以考虑。

- **软件故障:**DBMS 实现出现问题(例如，未被捕获的除零异常)，系统不得不停止。
- **硬件故障:**托管 DBMS 的计算机崩溃(例如，电源插头被拔出)。我们假设非易失性存储内容不会因系统崩溃而损坏。这被称为“故障-停止”假设，并简化了过程恢复。

类型#3:存储介质故障

*存储介质故障*是当物理存储设备损坏时发生的不可修复的故障。当存储介质发生故障时，必须从存档版本恢复 DBMS。DBMS 无法从存储故障中恢复，需要人工干预。

- **不可修复的硬件故障:磁头**崩溃或类似的磁盘故障破坏全部或部分非易失性存储。假定破坏是可检测到的。

4 缓冲池管理策略

DBMS 需要确保以下保证:

- 一旦 DBMS 告诉某人它已提交，任何事务的更改都是持久的。
- 如果事务终止，任何局部更改都是不持久的。

*窃取策略*指示 DBMS 是否允许未提交的事务覆盖非易失性存储中对象的最新提交值(事务可以将属于不同事务的未提交更改写入磁盘吗?)

- **偷窃**:是必需的
- **NO-STEAL**:是不允许的。

*强制策略*规定了 DBMS 是否要求在允许事务提交之前，事务所做的所有更新都反映在非易失性存储上。将提交消息返回给客户端)。

- **FORCE**:是必需的
- **NO-FORCE**:是必需的

强制写使它更容易恢复，因为所有的更改都被保留，但导致运行时性能差。

最容易实现的缓冲池管理策略称为 *NO-STEAL + FORCE*。在此策略中，DBMS 永远不必撤销因未将更改写入磁盘而终止的事务的更改。它也永远不需要重做已提交事务的更改，因为所有更改都保证在提交时写入磁盘。图 1 显示了一个 NO-STEAL + FORCE 的例子。

NO STEAL +FORCE 的一个限制是，所有的数据(即:交易需要修改的写入集必须放入内存中。否则，该事务将无法执行，因为 DBMS 是不允许的

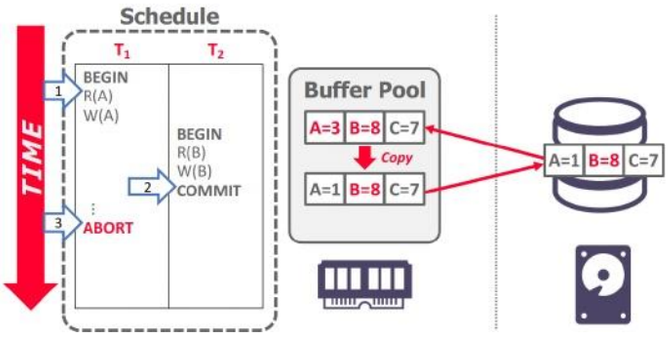

图 1:使用 NO-STEAL + FORCE 的 DBMS 示例—事务中的所有更改仅在事务提交时才写入磁盘。一旦调度从步骤#1 开始，T1 和 T2 的更改就被写入缓冲池。由于 FORCE 策略，当 T2 在步骤#2 提交时，它的所有更改都必须写入磁盘。为此，DBMS 在磁盘中复制内存，仅应用来自 T2 的更改，并将其写回磁盘。这是因为 NO-STEAL 防止将 T1 中未提交的更改写入磁盘。在步骤#3 中，DBMS 回滚 T1 很简单，因为磁盘上没有来自 T1 的脏更改。

在事务提交之前将脏页写入磁盘。

5 影子分页

影子分页是对先前方案的改进，DBMS 在写时复制页面以维护数据库的两个独立版本:

master:只包含来自已提交 txns 的更改。

- *影子*:由未提交事务产生的更改的临时数据库。

更新只在影子副本中进行。当事务提交时，影子副本被原子切换为新的 master。旧主子最终会被垃圾回收。这是一个 NO-STEAL+FORCE 系统的例子。图 2 展示了一个影子分页的高级例子。

实现

DBMS 以树结构组织数据库页面，其中根是单个磁盘页面。树有两个副本，*主*副本和*影子*副本。根总是指向当前的主副本。当事务执行时，它只对影子副本进行更改。

当事务想要提交时，DBMS 必须安装其更新。要做到这一点，它只需要覆盖根目录使其指向数据库的影子副本，从而交换主目录和影子目录。在覆盖根目录之前，所有的事务更新都不属于常驻磁盘的数据库。覆盖根后，所有事务更新都是磁盘驻留数据库的一部分。这种对根的覆盖可以原子性地完成。

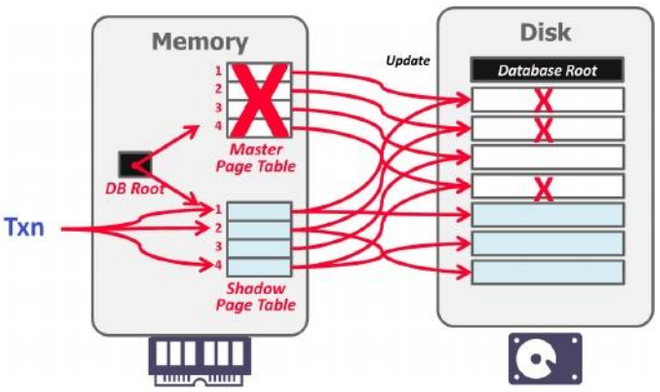

图 2:影子分页——数据库根指向一个母版页表，而母版页表又指向磁盘上的页(所有这些页都包含提交的数据)。当一个更新事务发生时，一个影子页表被创建，它指向与主数据库相同的页面。对磁盘上的临时空间进行修改，并更新影子表。为了完成提交，数据库根指针被重定向到影子表，影子表成为新的主表。

复苏

撤销:移除阴影页。保留主数据库和数据库根指针。

重做:根本不需要。

缺点

影子分页的一个缺点是复制整个页表的开销很大。在现实中，只需要复制树中导致更新叶节点的路径，而不需要复制整棵树。此外，影子分页的提交开销也很高。提交需要页表，以及根，除了每一个更新的页面都要被刷新。这会导致数据碎片化，也需要垃圾回收。另一个问题是，这一次只支持一个 writer transaction 或者一个 batch transaction。

6.日志文件

当事务修改页面时，DBMS 在覆盖主版本之前将原始页面复制到单独的日志文件中。重新启动后，如果存在日志文件，那么 DBMS 将恢复它以撤销未提交事务的更改。

7 预写日志

使用*预写日志*记录，DBMS 在对磁盘页面进行更改之前，将对数据库所做的所有更改记录在日志文件中(在稳定的存储上)。日志包含足够的信息来执行必要的撤销和重做操作，以在崩溃后恢复数据库。DBMS 必须先将数据库对象的更改对应的日志文件记录写入磁盘，然后才能将该对象刷新到磁盘。图 3 显示了 WAL 的一个示例。WAL 是一个偷窃+无强制系统的例子。

在影子分页中，DBMS 需要对磁盘上随机的不连续页面执行写操作。预写日志允许 DBMS 将随机写转换为顺序写，以优化性能。因此，几乎每个 DBMS 都使用预写日志记录(WAL)，因为它具有最快的运行时性能。但是 DBMS 使用 WAL 的恢复时间比影子分页要慢，因为它必须重放日志。

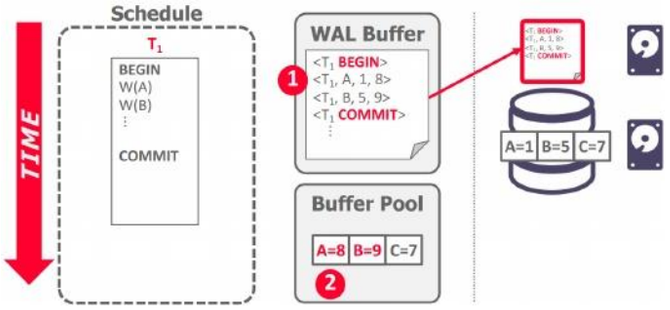

图 3:提前写日志记录-当事务开始时，所有更改都记录在内存中的 WAL 缓冲区中，然后再向缓冲池进行更改。在提交时，WAL 缓冲区被刷新到磁盘。一旦 WAL 缓冲区安全地放在磁盘上，事务结果就可以写入磁盘。

实现

DBMS 首先将所有事务的日志记录存储在易失性存储器中。在允许在非易失存储中覆盖该页之前，与已更新页相关的所有日志记录都被写入非易失存储。直到事务的所有日志记录都被写入稳定存储，事务才被认为已提交。

当事务开始时，为每个事务写入一个<BEGIN>记录到日志中，以标记其起始点。

当事务完成时，将<COMMIT>记录写入日志，并确保在它向应用程序返回确认之前，所有日志记录都被刷新。

每个日志条目都包含了倒带或重播对单个对象的更改所必需的信息:

- 交易 ID。
- 对象 ID。

值之前(用于撤销)。

值后(用于重做)

DBMS 必须将事务的所有日志条目刷新到磁盘，然后才能告诉外界事务已成功提交。系统可以使用“组提交”优化来批量处理多个日志刷新，以分摊开销。刷新要么发生在日志缓冲区满的时候，要么发生在连续刷新之间经过了足够的时间的时候。DBMS 可以随时将脏页写入磁盘

要的，只要是冲完之后的相应日志记录。

日志结构系统

在日志结构的 DBMS 中，事务的日志记录被写入称为 MemTable 的内存缓冲区。当这个缓冲区满了，它就被刷写到磁盘上。这种方法仍然需要一个独特的预写日志。这是因为 WAL 的刷新通常比 MemTable 的刷新更频繁，并且 WAL 可能包含未提交的事务。WAL 用于在从崩溃中恢复时重新创建内存中的 MemTable。

8 种日志记录方案

日志记录的内容可能因实现而异。

物理日志记录：

记录对数据库中特定位置所做的字节级更改。

示例:git diff

逻辑日志记录：

•记录事务执行的高级操作。

不一定局限于一个页面。

与物理日志记录相比，每个日志记录需要更少的数据写入，因为每个记录可以在多个页面上更新多个元组。然而，当不确定的并发控制方案中存在并发事务时，用逻辑日志实现恢复是困难的。此外，恢复需要更长的时间，因为你必须重新执行每个事务。

•示例:由事务调用的更新、删除和插入查询。

生理记录：

混合方法，其中日志记录针对单个页面，但不指定页面的数据组织。也就是说，根据页面中的槽号识别元组，而不指定更改位于页面中的确切位置。因此，DBMS 可以在日志记录被写入磁盘后重新组织页面。

数据库管理系统中最常用的方法。

9 检查点

基于 wal 的 DBMS 的主要问题是日志文件将永远增长。在崩溃之后，DBMS 必须重播整个日志，如果日志文件很大，这可能需要很长时间。因此，DBMS 可以定期采取*检查点*，将所有缓冲区刷新到磁盘。

DBMS 应该多久执行一次检查点取决于应用程序的性能和停机时间需求。过于频繁地使用检查点会导致 DBMS 的运行时性能下降。但是在检查点之间等待很长时间可能同样糟糕，因为重启后系统的恢复时间增加了。

阻塞检查点实现：

•DBMS 停止接受新的事务，并等待所有活动事务完成。

将当前驻留在主内存中的所有日志记录和脏块刷新到稳定存储中。

•写入一个<CHECKPOINT>条目到日志中并刷写到稳定存储中。