可配置的拓展2MB事务功能开发文档

修订历史

版本	修订日期	修订描述	作者	备注
Cedar0.3	2017-09-28	可配置的拓展2MB事务功能开发文档	屈兴 朱涛	

1 总体设计

1.1 综述

> Cedar是华东师范大学计算机科学与软件工程学院基于OceanBase研发的可扩展的关系数据库，实现了巨大数据量上的跨行跨表事务。在OceanBase(OB)中，一个事务写操作(IUD)的操作日志数据量的存在限制，不能超过2MB大小。这样的限制导致了OB数据库不支持长事务的执行和大批量更新操作。因此，对于需要写大量数据的事务执行，OB只能通过多个类似的短，小数据量的写事务组合完成。这样具有明显的缺陷：(1)事务的原子性等性质被破坏；(2)执行时间长、资源开销大；(3)程序员维护的业务逻辑复杂。因此原OB数据库只能适用于短、小事务的应用场景。
>
> 为了解决这样的应用场景的局限。在Cedar0.3版本中，实现了基于存储过程的可配置的拓展2MB事务的功能。该功能支持手动配置参数设置事务最大操作日志数据量限制和存储过程调用时使用一个hint，使以存储过程执行的事务可以突破OB原有的2MB大小限制。

1.2 名词解释

- **MS**：MergeServer，Cedar系统中的查询处理服务器，负责接收和解析SQL请求、生成和执行查询计划以及将所有节点的查询结果合并并返回给客户端。
- **UPS**：UpdateServer，Cedar系统中的事务处理服务器，提供事务支持和存储增量数据。

1.3 功能

可配置的拓展2MB事务，提供给用户5个系统配置项，配置写事务最大日志数据量大小限制，重启UPS后生效。为了避免在存在高并发短事务情况下，长事务长期占有分配的内存页，无法释放，导致系统内存资源急速消耗，导致事务执行失败而回滚。允许存储过程调用的时候增加一个hint标识长事务，使长事务与短事务所使用的内存页隔离。

1.4 性能指标

可配置的拓展2MB事务的配置项修改后，不应使Cedar的吞吐率下降和时延增加。

2 模块设计

可配置的拓展2MB事务的实现，可以主要分为三个模块：长事务词法语法解析模块、长事务执行管理模块和日志处理流程缓冲区配置模块。

2.1 长事务词法语法解析子模块设计

通过hint /*+LONG_TRANS*/标识这次事务在UPS上的执行以长事务的方式执行。目标是能够将这个hint的信息被MS解析，然后传递到UPS上，然后走长事务的流程。

简单地讲实现方式是，在处理存储过程的CALL语句时，增加处理hint的流程，发现存在LONG_TRANS时，设置ObResultSet上新定义的long_trans_成员变量为true。在存储过程的执行算子open阶段，通过这个成员变量设置发往UPS的事务头的参数，最后UPS反序列化，能够读取到该事务为长事务的信息。

> 修改内容：
> 1.分别向词法定义文件sql_parser.l和语法定义文件sql_parser.y中添加LONG_TRANS的词法和语法规则，使解析器能够识别该hint。
> 2.在ObResultSet中添加一个成员变量long_trans_用于设置事务执行模式。
> 3.在ObUpsExcutor的open阶段设置事务类型参数为LONG_READ_WRITE_TRANS类型。

2.2 长事务执行管理子模块设计

长事务执行管理子模块的主要目标是进行内存资源的隔离，但是长事务的执行管理模块实际上是非常简单的。

简单说明下，Cedar中事务的在UPS上执行的特点，在UPS上每一个事务开始执行前，会获取事务的session，该session维护有session类型，session描述符，session上下文，session号，锁资源等信息。其中关键的是session类型和session上下文，不同的session类型获得不同的session上下文，而同类型session上下文，共用一个内存分配器。

于是，为了进行资源隔离的目标，新增一种事务上下文类型ST_LONG_READ_WRITE，在session工厂分配产生事务上下文时，对于长事务类型，分配一个长事务上下文。而这个事务上下文实际的数据结构完全和普通的读写事务类型ST_READ_WRITE一样，所以长事务的后续的处理流程完全和普通读写事务完全一致。只是不与普通事务共同使用一个内存分配器。

对于，为什么长事务不能和普通事务使用同一个内存分配器，可以见图2.2所示。

> 修改内容：
> 1.新增ST_LONG_READ_WRITE事务上下文类型，以及长事务上下文的获取流程。

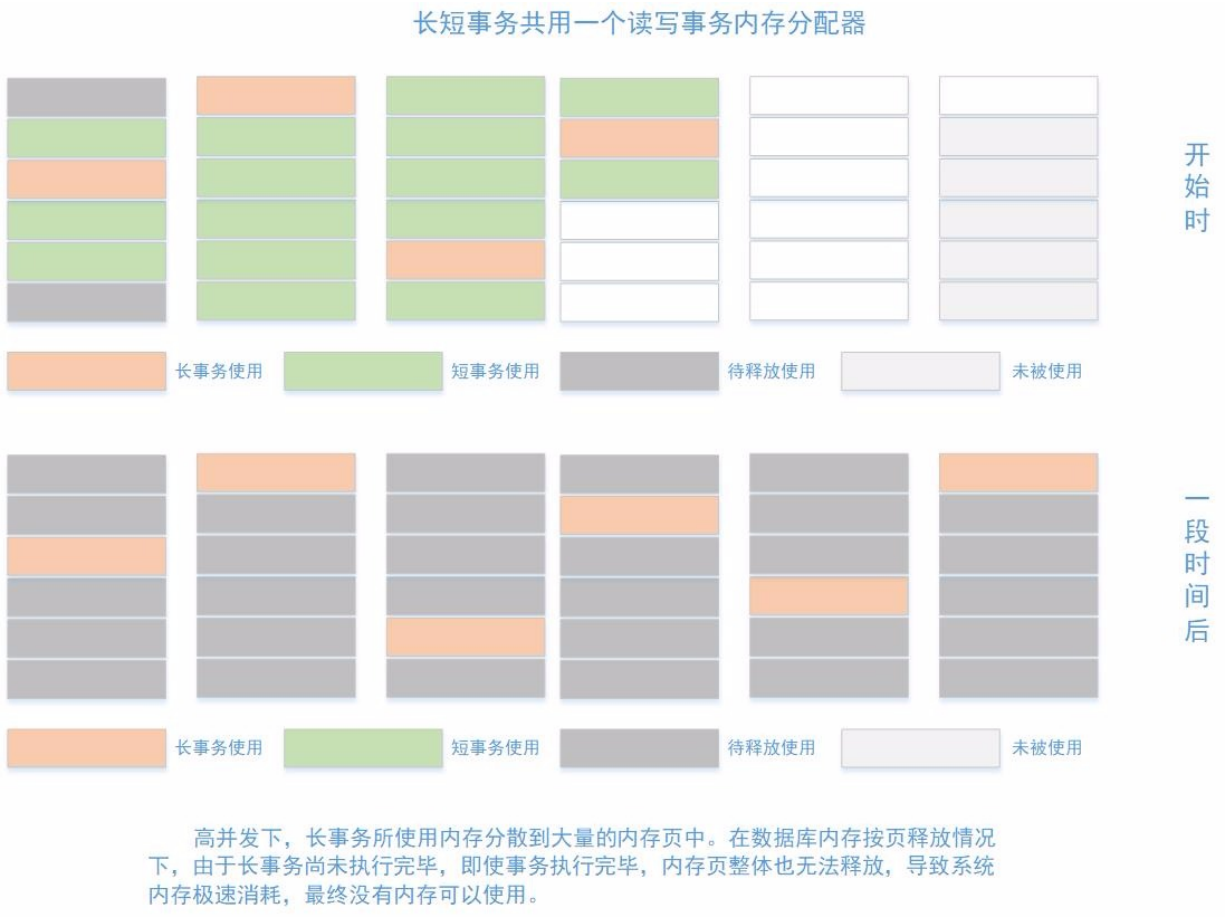

图2.2 长事务与短事务使用同一个内存分配器

2.3 日志处理流程缓冲区配置子模块设计

日志缓冲区配置子模块是比较零散的模块，需要将日志处理流程中的缓冲区大小配置成至少可以容纳一个长事务的大小缓冲区。由于OB设计成面向互联网应用的数据库，认为单个事务的日志数据大小不会超过2MB（实际上为1.875MB），所以日志处理流程的缓冲区大小设置为通常为常量2MB或4MB。

为解决缓冲区过小，无法完整接收一个事务日志数据量，Cedar0.3中的基本思想是将设置缓冲区的大小常量改为变量，这些变量在构造函数中或设置缓冲区大小的接口中读取UPS的系统配置项，然后将为缓冲区分配内存。

> 修改内容：
> 1.在ob_update_server_config.h中新增配置项max_log_buffer_size、log_buffer_max_size、block_bits。
> 2.新定义三个公共变量OB_MAX_LOG_BUFFER_SIZE、OB_LOG_BUFFER_MAX_SIZE和OB_DEFAULT_BLOCK_BITS，用于设置日志相关缓冲区大小
> 3.修改BatchPacketQueueThread、AppendBuffer、ThreadSpecificBuffer等的构造函数，使用缓冲区变量初始化缓冲区大小。
> 4.修改ObLogGenerator对象的init函数调用处使用OB_MAX_LOG_BUFFER_SIZE设置允许生成的最大日志大小。
> 5.修改ob_ring_buffer.h/cpp文件将DEF_BLOCK_BITS和DEF_BLOCK_SIZE成员设置为变量，将UPS通信数据的缓冲区大小由OB_DEFAULT_BLOCK_BITS设置。
> 6.修改ob_single_log_reader.cpp文件，将日志读取器的缓冲区大小由OB_LOG_BUFFER_MAX_SIZE设置。
> 7.修改ob_log_replay_worker.cpp文件，将ObLogReplayWorker日志回放线程的缓冲区大小由OB_LOG_BUFFER_MAX_SIZE设置。

3 模块接口

3.1使用方法

1. 超时时间

日志超过2MB大小的事务，执行时间通常很长，需要重新设置数据库超时时间（单位us）。超时时间有两项，一个是整个查询的超时时间，另外一个是事务执行的超时时间。设置超时时间方法：

会话级设置

- set @@session.ob_query_timeout=9000000000;
- set @@session.ob_tx_timeout =9000000000;

全局级设置

- set @@global.ob_query_timeout=9000000000;
- set @@global.ob_tx_timeout =9000000000;

2. UPS系统配置项

在执行可超过2MB事务前，需要通过修改UPS系统配置项，设置日志相关的缓冲区大小，各个配置项和作用如下：

- max_log_buffer_size：UPS生成的最大日志大小，需要为5个配置项中最小的一个。默认值为1.875MB。
- log_buffer_max_size: 日志处理过程的相关缓冲区大小，应不小于max_log_buffer_size。默认值为2MB。
- block_bits：数据库网络通信数据块的大小，应不小于log_buffer_max_size 。默认值为22，即4MB大小。
- log_cache_block_size：备集群同步主集群日志缓冲区大小，应不小于log_buffer_max_size。默认值为32MB。
- commit_log_size：日志提交缓冲区大小，应不小于log_buffer_max_size。默认值为64MB。

修改系统配置项的语句：

```
ALTER SYSTEM SET argument_name = value SERVER_TYPE = UPDATESERVER;
```

例如，欲使事务最大可写数据量为32MB，可以进行如下设置（33554432 =32 MB，67108864=64MB，commit_log_size为默认值）：

```
ALTER SYSTEM SET max_log_buffer_size = 33554432 SERVER_TYPE = UPDATESERVER;
ALTER SYSTEM SET log_buffer_max_size = 67108864 SERVER_TYPE = UPDATESERVER;
ALTER SYSTEM SET block_bits = 26 SERVER_TYPE = UPDATESERVER;
ALTER SYSTEM SET log_cache_block_size = '64MB' SERVER_TYPE = UPDATESERVER;
```

在设置完配置项后，可通过如下两种方式查看系统配置项：

- select * from __all_sys_config ;
- select * from __all_sys_config_stat;

3. 执行

执行一个写日志数据量超过2MB的存储过程前，在存储过程函数末尾添加一个hint '/*+LONG_TRANS*/'：

```
CALL procedure_name(procedure_argument_list) /*+ LONG_TRANS*/
例：CALL ptest() /*+LONG_TRANS*/
```

4 使用限制条件和注意事项

1. 为什么max_log_buffer_size 要小于log_buff_max_size ，而不能等于或大于？

大于的情况很好解释，日志的缓冲区存放不下超过其大小的日志，会导致错误，事务回滚。不能等于的原因是生成的日志在各个集群处理和通信过程中会包含一些包头等结构，这样日志大小就有可能超过缓冲区大小，所以需要略小于后者（比如小于0.5MB或1MB等）。

2. 为什么配置大小，并不是手动设置的大小？

为了预防用户的错误输入，程序在实际使用的配置项时，会检查是否在阈值范围内，不满足则进行一定的修正，例如设置max_log_buffer_size 大小至少log_buff_max_size 要小0.5M，最小值为默认值(1.875M)。

3. 事务实际写入的数据，能否达到配置的限制大小？

不能达到。配置的大小实际是事务产生的日志数据的大小，写入的数据存储成日志格式数据会有一定的扩展，例如添加日志号、校验位等数据，导致实际事务能够写入的数据量略小于设置限制的大小，但是相差不大。

4. 长事务执行失败或回滚，客户端返回错误信息：ERROR 119 (25S03): transaction is rolled back？ERROR 14 (HY001): OB-14: Memory overflow？

对于119 错误，并且日志出现类似ERROR check_log_size (ob_log_generator2.cpp:454) [140122427606784] log_size[2049777] + reserved[1024] + header[52] > log_buf_len[1966592] 信息，说明缓冲区过小，需要重新设置日志缓冲区相关的配置项。

对于14 错误，并且日志出现ERROR： alloc_new_page (page_arena.h:163) cannot allocate memory.sz=28339766…，说明RWSESSION的内存分配器无法分配内存，系统内存资源不够。

5. 存储过程for…loop/loop语句执行客户端返回未知错误？

这是loop 语法的问题。存储过程for loop、loop循环体中的sql语句默认是成组执行，一旦循环次数过多则导致发送到ups上执行的物理计划本身不全，或者产生的数据量超过单条sql数据量的限制，将导致事务执行失败。

一种解决方式时调用这个存储过程时，添加hint，关闭成组执行。调用存储过程时加上**/*+ NO_GROUP */**设置成不成组执行（如，call ptest() /*+ NO_GROUP */ ; ）。注意，使用hint语法，mysql 连接ms时需要添加**-c** 参数。

另外一种是，使用while 语法，while语法，在存储过程中并未做循环体的成组执行。

6. 当新的集群加入时，应该如何做？

新集群启动完毕后，需要重新启动新集群的UPS，以此使现有集群上的配置在新的集群上同步并生效。

7. 三集群中某个备集群出现数据不一致如何处理？

备集群，由于各种原因不可挽回的出现了数据不一致。最后的手段是，将备集群中的数据完全清掉，然后重建目录，从最开的日志号开始回放。最终恢复到一致。

8.在调整缓冲区大小后，重启UPS出现UPS很快自己挂掉？

如果在ups的退出前的日志中没有发现严重错误，可能的原因是由于此时服务器内存不够用了，导致操作系统杀掉了UPS。检测猜测可以用sudo cat /etc/log/messages 或 cat /var/log/messages 中查看操作系统日志。

9.没有完全同步2MB设置到所有集群，就执行事务？

如果在备集群的MS，CS不存在的情况下，重新调整2MB包的配置项，会导致备集群没有能成功设置新的配置。此时，如果重新恢复备集群的状态（四种server都存在）后，稍等一下（避免配置未持久化到配置文件，可检查），然后重启UPS，可以使备集群同步到最新的2MB配置。但是，如果在备集群没有同步到配置的情况下，执行超过2MB的事务，可能导致这些事务在集群上丢失（若原未同步的备集群被设为主？）。

因此，我们建议在调整缓冲区的时候，尽量检查三集群是正常状态。