解析WiredTiger：事务

zerok

2016-05-30

WiredTiger从被mongoDB收购到成为mongoDB的默认存储引擎的一年半得到了迅猛的发展，也逐步被外部熟知。WiredTiger（以下简称WT）是一个优秀的单机数据库存储引擎，它拥有诸多的特性，既支持BTree索引，也支持LSM Tree索引，支持行存储和列存储，实现ACID级别事务等。其实WT引擎主要是为了解决以下几个方面技术的问题：

设计了充分利用CPU并行计算的内存模型的无锁并行框架，使得WT引擎在多核CPU上的表现优于其他存储引擎。针对磁盘存储特性，WT实现了一套基于BLOCK/Extent的友好的磁盘访问算法，使得WT在数据压缩和磁盘IO访问上优势明显。实现了基于snapshot技术的ACID事务，snapshot技术大大简化了WT的事务模型，摒弃了传统的事务锁隔离又同时能保证事务的ACID。WT根据现代内存容量特性实现了一种基于Hazard pointer page的evict LRU cache模型，充分利用了内存容量的同时又能拥有很高的事务读写并发。

在本文中，我们主要针对它的事务来展开分析来看看它的事务是如何实现的。说到事务，必然要对事务这个概念和它的ACID特性进行简单的介绍。

### 事务与ACID

什么是事务？事务就是通过一系列操作来完成一件事情，在进行这些操作的过程中，要么这些操作完全执行，要么这些操作全不执行，不存在中间状态，事务分为事务执行阶段和事务提交阶段。一般说到事务，就会想到它的特性—ACID，那么什么是ACID呢？

原子性(Atomicity):

组成事务的系列操作是一个整体，要么全执行，要么不执行。

一致性(Consistency):

隔离性(Isolation):

多个事务在并发执行时，事务执行的中间状态是其他事务不可访问的。

持久性(Durability):

事务一旦提交生效，其结果将永久保存，不受任何故障影响。

## WT事务构造

### 事务对象

### WT中的MVCC

### WT事务的snapshot

## WT的事务过程

### 事务执行

### 事务提交

## WT的事务隔离

#### Read-uncommited

#### Read-commited

#### Snapshot

## WT事务日志

### 日志格式

### WAL与日志写并发

### 事务恢复

## 后记