## 1.案例背景

公司的服务器组件很多都是用SQLite,由于SQLite是一个可以在硬盘中保存成单独文件的,并且不需要像MYSQL或其他数据库一样需要一些系统配置环境,使它变得非常的轻量级, 使用也非常方便,但是对于服务器来说,数据库的存储效率是非常关键的,很大程度的影响到了服务器的性能,关于它的插入读取性能,我整理了一些资料,来深挖一下SQLite的读取性能.

## 2.案例分析

首先需要准备一些环境,我准备在SQLite中插入100000条数据,对于服务器来说,这个数据量并不是特别大,但是也不小了,制作一个大数据插入的性能测试.

数据用Python来编造一些,贴上代码.

filehandle **=** open**(**'test.txt'**,**'w'**)**

**for** i **in** range**(**0**,**1000000**):**

filehandle**.**write**(**"testdata\n"**)**

filehandle**.**close**()**

这样就生成了一个具有100001条数据的文本了,暂定名称为test.txt

在SQLite官方网站上下载最新版本的SQLite源代码,路径为:

<http://www.sqlite.org/2016/sqlite-amalgamation-3150100.zip> 目前的最新版本为3.15.

到这里为止,基本的环境准备的差不多了,该上代码一点C语言代码了.

首先声明下测试笔记本的配置,i5-4300M 2.6GHz双核四线程,主频不高!

在VS2013建立一个工程,使用release发布,并且开启完全优化(/Ox)!

代码如下:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include "sqlite3.h"

#define INPUTDATA "test.txt"

#define DATABASE "test.sqlite"

#define TABLE "CREATE TABLE IF NOT EXISTS TTC (id INTEGER PRIMARY KEY, DATA TEXT, YEAR TEXT, MONTH TEXT, DAY TEXT, HOUR TEXT, SECOND TEXT, Time TEXT)"

#define BUFFER\_SIZE 256

int main**()**

**{**

sqlite3 **\*** db**;**

sqlite3\_stmt **\*** stmt**;**

clock\_t cStartClock**;**

FILE **\*** pFile**;**

int nRetCode**;**

int n **=** 0**;**

char sInputBuf**[**BUFFER\_SIZE**]** **=** "\0"**;**

char sSQL**[**BUFFER\_SIZE**]** **=** "\0"**;**

char **\*** sErrMsg **=** 0**;**

char **\*** tail **=** 0**;**

char **\*** sRT **=** 0**;**

char **\*** sBR **=** 0**;**

char **\*** sVR **=** 0**;**

char **\*** sST **=** 0**;**

char **\*** sVI **=** 0**;**

char **\*** sDT **=** 0**;**

char **\*** sTM **=** 0**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Open the Database and create the Schema \*/

sqlite3\_open**(**DATABASE**,** **&**db**);**

sqlite3\_exec**(**db**,** TABLE**,** **NULL,** **NULL,** **&**sErrMsg**);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Open input file and import into Database\*/

cStartClock **=** clock**();**

pFile **=** fopen**(**INPUTDATA**,** "r"**);**

**while** **(!**feof**(**pFile**))** **{**

fgets**(**sInputBuf**,** BUFFER\_SIZE**,** pFile**);**

sRT **=** strtok**(**sInputBuf**,** "\t"**);**

sBR **=** strtok**(NULL,** "\t"**);**

sVR **=** strtok**(NULL,** "\t"**);**

sST **=** strtok**(NULL,** "\t"**);**

sVI **=** strtok**(NULL,** "\t"**);**

sDT **=** strtok**(NULL,** "\t"**);**

sTM **=** strtok**(NULL,** "\t"**);**

/\* ACTUAL INSERT WILL GO HERE \*/

n**++;**

**}**

fclose**(**pFile**);**

printf**(**"\r\n 插入 %d 记录,共耗时 %4.2f 秒\r\n\n\n"**,** n**,** **(**clock**()** **-** cStartClock**)** **/** **(**double**)**CLOCKS\_PER\_SEC**);**

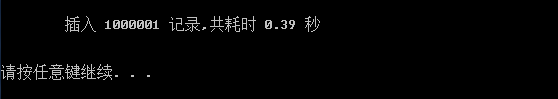
sqlite3\_close**(**db**);**

system**(**"pause"**);**

**return** 0**;**

**}**

先不开始真正的插入数据,先测试下C语言的IO性能,结果如图所示,IO真的是很快,读取100000条数据也就花了0.39秒



接下来插入真正写入数据库的部分代码,根据目前的发现,应该大部分服务器都是这种方式来进行数据插入的,来试一下性能!

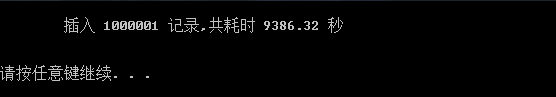
代码如下:

/\* ACTUAL INSERT WILL GO HERE \*/

sprintf**(**sSQL**,** "INSERT INTO TTC VALUES (NULL, '%s', '%d', '%d', '%d', '%s', '%s', '%s')"**,** sRT**,** sBR**,** sVR**,** sST**,** sVI**,** sDT**,** sTM**);**

sqlite3\_exec**(**db**,** sSQL**,** **NULL,** **NULL,** **&**sErrMsg**);**

结果如下:



大概用了2.6小时才完成这些操作! 这就是我们经常使用的操作的工作效率,是应该改进一下了.

## 3.解决过程

事实是这样的,在上面的操作中SQLite对于INSERT都是一个独一无二的事务,所以每次进行插入操作的时候,SQLite都会进入Insert事务,结束操作的时候退出该事务,并将数据保存在SQLite文件中,我们不去控制它的事务,虽然说比较方便,但是却浪费了很多的性能,所以在执行大量操作的时候,我们应该让SQLite提前进入事务,使用” BEGIN TRANSACTION”,在完成所有插入操作后结束事务,使用” END TRANSACTION”,一次性将数据写入到文件中,这样应该能提高性能.

代码如下:

sqlite3\_exec**(**db**,** "BEGIN TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

pFile **=** fopen**(**INPUTDATA**,** "r"**);**

**while** **(!**feof**(**pFile**))** **{**

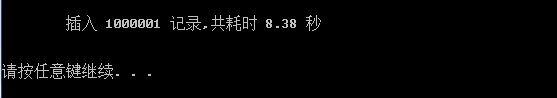
**...**

**}**

fclose**(**pFile**);**

sqlite3\_exec**(**db**,** "END TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

结果如下:



看上去是不是好了很多,但是还可以更快!

使用Begin Transaction提高了巨大的性能,但是每次去更新SQL语句会影响一些性能,假如不断的重复一条SQL语句,应该更快一点.

使用sqlite3\_prepare\_v2去构造SQL语句,并且使用sqlite3\_bind\_text去更新元素,这样应该会更好一些.

代码如下:

sprintf**(**sSQL**,** "INSERT INTO TTC VALUES (NULL, @RT, @BR, @VR, @ST, @VI, @DT, @TM)"**);**

sqlite3\_prepare\_v2**(**db**,** sSQL**,** BUFFER\_SIZE**,** **&**stmt**,** **NULL);**

sqlite3\_exec**(**db**,** "BEGIN TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

pFile **=** fopen**(**INPUTDATA**,** "r"**);**

**while** **(!**feof**(**pFile**))** **{**

fgets**(**sInputBuf**,** BUFFER\_SIZE**,** pFile**);**

sRT **=** strtok**(**sInputBuf**,** "\t"**);**

sBR **=** strtok**(NULL,** "\t"**);**

sVR **=** strtok**(NULL,** "\t"**);**

sST **=** strtok**(NULL,** "\t"**);**

sVI **=** strtok**(NULL,** "\t"**);**

sDT **=** strtok**(NULL,** "\t"**);**

sTM **=** strtok**(NULL,** "\t"**);**

/\* ACTUAL INSERT WILL GO HERE \*/

sqlite3\_bind\_text**(**stmt**,** 1**,** sRT**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 2**,** sBR**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 3**,** sVR**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 4**,** sST**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 5**,** sVI**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 6**,** sDT**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 7**,** sTM**,** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_step**(**stmt**);//执行Insert操作**

sqlite3\_clear\_bindings**(**stmt**);//记得要清除**

sqlite3\_reset**(**stmt**);//重置**

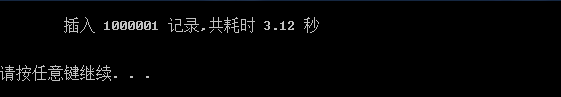
n**++;**

**}**

fclose**(**pFile**);**

sqlite3\_exec**(**db**,** "END TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

结果如下:



看上去好了一点了,又提高了一倍性能,但是还可以更快!

SQLite中有一个默认行为,它会去调用系统的函数去等待数据写入到磁盘中,如果将这个行为关闭应该会更快一些,虽然说这样会有风险,如果系统出现异常(如突然停电),那就悲剧了,会导致数据文件损坏!不过就测试性能来说,还是非常有意义的!

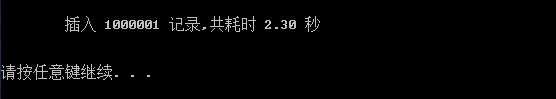
代码如下:

sqlite3\_open(DATABASE, &db);

sqlite3\_exec(db, TABLE, NULL, NULL, &sErrMsg);

sqlite3\_exec(db, "PRAGMA synchronous = OFF", NULL, NULL, &sErrMsg);

结果如下:



这样一来性能又提高了一些了,但是还能更快!

在执行过程中发现有一个\*.sqlite-journal的临时文件,该文件是SQLite控制日志,假如我们将SQLite的文件控制日志保存在内存中,应该也有一些性能提升.

代码如下:

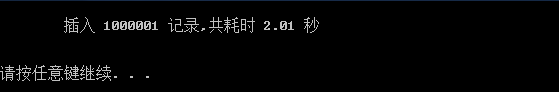
sqlite3\_open**(**DATABASE**,** **&**db**);**

sqlite3\_exec**(**db**,** TABLE**,** **NULL,** **NULL,** **&**sErrMsg**);**

sqlite3\_exec**(**db**,** "PRAGMA synchronous = OFF"**,** **NULL,** **NULL,** **&**sErrMsg**);**

sqlite3\_exec**(**db**,** "PRAGMA journal\_mode = MEMORY"**,** **NULL,** **NULL,** **&**sErrMsg**);**

结果如下:



又有一点提升了,看来是比较艰难了.

把SQLite文件保存在RAM中,这样应该也能提升一点性能.

代码如下:

#define DATABASE ":memory:"

结果



也有改进,不过这个不太常用,不建议使用这种方式.

下面把C代码重构一遍,char \*转换这块省略,直接写入.

代码如下:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include "sqlite3.h"

#define INPUTDATA "test.txt"

#define DATABASE "test.db"

#define TABLE "CREATE TABLE IF NOT EXISTS TTC (id INTEGER PRIMARY KEY, DATA TEXT, YEAR TEXT, MONTH TEXT, DAY TEXT, HOUR TEXT, SECOND TEXT, Time TEXT)"

#define BUFFER\_SIZE 256

int main**()**

**{**

sqlite3 **\*** db**;**

sqlite3\_stmt **\*** stmt**;**

clock\_t cStartClock**;**

FILE **\*** pFile**;**

int nRetCode**;**

int n **=** 0**;**

char sInputBuf**[**BUFFER\_SIZE**]** **=** "\0"**;**

char sSQL**[**BUFFER\_SIZE**]** **=** "\0"**;**

char **\*** sErrMsg **=** 0**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Open the Database and create the Schema \*/

sqlite3\_open**(**DATABASE**,** **&**db**);**

sqlite3\_exec**(**db**,** TABLE**,** **NULL,** **NULL,** **&**sErrMsg**);**

sqlite3\_exec**(**db**,** "PRAGMA synchronous = OFF"**,** **NULL,** **NULL,** **&**sErrMsg**);**

sqlite3\_exec**(**db**,** "PRAGMA journal\_mode = MEMORY"**,** **NULL,** **NULL,** **&**sErrMsg**);**

sprintf**(**sSQL**,** "INSERT INTO TTC VALUES (NULL, @RT, @BR, @VR, @ST, @VI, @DT, @TM)"**);**

sqlite3\_prepare\_v2**(**db**,** sSQL**,** BUFFER\_SIZE**,** **&**stmt**,** **NULL);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Open input file and import into Database\*/

cStartClock **=** clock**();**

sqlite3\_exec**(**db**,** "BEGIN TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

pFile **=** fopen**(**INPUTDATA**,** "r"**);**

**while** **(!**feof**(**pFile**))** **{**

fgets**(**sInputBuf**,** BUFFER\_SIZE**,** pFile**);**

/\* ACTUAL INSERT WILL GO HERE \*/

sqlite3\_bind\_text**(**stmt**,** 1**,** strtok**(**sInputBuf**,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 2**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 3**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 4**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 5**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 6**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_bind\_text**(**stmt**,** 7**,** strtok**(NULL,** "\t"**),** **-**1**,** SQLITE\_TRANSIENT**);**

sqlite3\_step**(**stmt**);**

sqlite3\_clear\_bindings**(**stmt**);**

sqlite3\_reset**(**stmt**);**

n**++;**

**}**

fclose**(**pFile**);**

sqlite3\_exec**(**db**,** "END TRANSACTION"**,** **NULL,** **NULL,** **&**sErrMsg**);**

printf**(**"\r\n 插入 %d 记录,共耗时 %4.2f 秒\r\n\n\n"**,** n**,** **(**clock**()** **-** cStartClock**)** **/** **(**double**)**CLOCKS\_PER\_SEC**);**

sqlite3\_close**(**db**);**

system**(**"pause"**);**

**return** 0**;**

**}**

结果如下:



这个结果出乎了我的意料,从2.6小时(9386.32s)到1.91s,效率提升4914.3倍!

## 4.案例总结

要多去关注一些对技术有帮助的知识,可能真正创造他的人早已经想到了它真正存在的意义和价值,不能把对技术的认知只是针对它的表面,要去学习它更深层次的东西,也许不是你创造的,但是可以比别人用的更好.

全文完