Atitit 并发测试耗时时间工具

[1.1. 关键词 1](#_Toc24922)

[1.2. 主要记录 所有时间，遍历数量，平均时间 1](#_Toc25425)

[1.3. 主要方法伪代码 1](#_Toc24199)

[1.4. 调试方法 2](#_Toc5510)

[1.5. Corecode 2](#_Toc13014)

[1.6. Coede 2](#_Toc8048)

## 关键词

**synchronized**

## 主要记录 所有时间，遍历数量，平均时间

每一条历史就都要博爱路保留，，方便追溯与调试

## 主要方法伪代码

ExecutorService ExecutorService1\_theardpool = Executors.newFixedThreadPool(50);

for (int i = 1; i < 20000; i++) {

ExecutorService1\_theardpool.submit(new Runnable() {

@Override

public void run() {

Statement st = getStt(); // ini envi

long startTime = System.currentTimeMillis();

exec(st);

calcNout\_ex(startTime);

}

}); // end sumbit

sleep4reduceCpu(i);

}

## 调试方法

## Corecode

C:\0wkspc\oploggerPrj\webroot\WEB-INF\classes\com\attilax\oplog\util\MatViewTest2\_dpt1$1.class

public class MatViewTest2\_dpt1 {

public static AtomicLong all\_int = new AtomicLong(0);

public static AtomicLong cnt = new AtomicLong(0);

public static AtomicLong avg = new AtomicLong(0);

synchronized private static void calcNout(long excTime) {

all\_int.addAndGet(excTime);

cnt.incrementAndGet();

avg.set(all\_int.get() / cnt.get());

System.out.println("---------执行时间：" + excTime + "ms,avgtime: " + avg + "ms,cnt:" + cnt + ",alltime:"

+ all\_int);

}

## Coede

package com.attilax.oplog.util;

import java.sql.Connection;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Timer;

import java.util.TimerTask;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.atomic.AtomicLong;

import org.apache.ibatis.session.SqlSession;

import com.attilax.oplog.OperLogUtil;

import cn.freeteam.util.MybatisSessionFactory;

public class MatViewTest2\_dpt1 {

public static AtomicLong all\_int = new AtomicLong(0);

public static AtomicLong cnt = new AtomicLong(0);

public static AtomicLong avg = new AtomicLong(0);

synchronized private static void calcNout(long excTime) {

all\_int.addAndGet(excTime);

cnt.incrementAndGet();

avg.set(all\_int.get() / cnt.get());

System.out.println("---------执行时间：" + excTime + "ms,avgtime: " + avg + "ms,cnt:" + cnt + ",alltime:"

+ all\_int);

}

public static void main(String[] args) {

/\*

\*

\*

\* Timer tmr = new Timer();

\*

\* tmr.schedule(new TimerTask() {

\*

\* @Override public void run() {

\* MybatisSessionFactory.CONFIG\_FILE\_LOCATION="/mybatis\_postgresql.xml";

\* SqlSession session = MybatisSessionFactory.getSession(); Connection

\* conn = session.getConnection(); Statement st; String sql =

\* "refresh materialized view v1;"; try { int i;

\* conn.setAutoCommit(true); st = conn.createStatement(); i =

\* st.executeUpdate(sql); conn.commit(); System.out.println(i); } catch

\* (Exception e) {

\*

\* e.printStackTrace(); }

\*

\* } },50, 3000);

\*/

// if("1"=="1")

// return;

ExecutorService ExecutorService1\_theardpool = Executors.newFixedThreadPool(20);

for (int i = 1; i < 20000; i++) {

ExecutorService1\_theardpool.submit(new Runnable() {

@Override

public void run() {

// new OperLogUtil().log4postgre\_core("ati1", "

// {\"txt\":123} ", "tag");

String sql = "update dpt1 set var1='sssx26647' where id='901982439071686659' ";

MybatisSessionFactory.CONFIG\_FILE\_LOCATION = "/mybatis\_postgresql.xml";

SqlSession session = MybatisSessionFactory.getSession();

Connection conn = session.getConnection();

Statement st;

long startTime = System.currentTimeMillis();

try {

int i;

conn.setAutoCommit(true);

st = conn.createStatement();

i = st.executeUpdate(sql);

conn.commit();

System.out.println(i);

} catch (Exception e) {

long endTime = System.currentTimeMillis();

long excTime = (long) (endTime - startTime);

try {

calcNout(excTime);

e.printStackTrace();

} catch (Exception e2) {

System.out.println(e2);

}

}

}

}); // end sumbit

if (i % 101 == 0)

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

// ExecutorService1\_theardpool.shutdown();

}

}