文章编号:1673-8411 (2018) 01-0118-03

关于异步多线程快速提取 CIMISS 数据入库方法的研究与应用

黎颖智, 史彩霞, 刘世学
(广西气象服务中心, 广西 南宁 530022)

摘 要:为实现异步多线程快速提取 CIMISS 数据入库,使用 c# 5.0 新特性 async 和 await 进行异步操作实现主线程与方法线程的并行执行,解决提取 CIMISS 数据时出现的主界面无响应问题,使用 SqlBulkCopy 则可实现批量数据的快速入库。

关键词:CIMISS;多线程;异步;async/await;SqlBulkCopy
中图分类号:P49 **文献标识码**:A

The research and application of the fast extraction method for the asynchronous multithreading of CIMISS data

Li Yingzhi, Shi Caixia, Liu Shixue
(Guangxi Meteorological Service Center, Nanning Guangxi 530022)

Abstract: To achieve the asynchronous multi-threaded quick extraction CIMISS data warehousing, we used c # 5.0 new characteristic async and await for the asynchronous operation to achieve the concurrent execution of the main thread and method thread and solved the problem of no response to the main interface when extracting CIMISS data. Additionally, using SqlBulkCopy batch data can achieve quick warehousing.

Keywords: CIMISS; multi-thread; asynchronous; async /await; SqlBulkCopy

1 引言

全国综合气象信息共享系统(CIMISS)是中国气象局建设的集气象资料采集、加工处理、存储管理和共享服务等功能于一身的气象信息业务平台。能够为国、省、地、县各级气象科研部门提供规范统一高效的气象数据。虽然气象数据统一访问接口(MUSIC), 为各级应用系统提供了包括实时观测数据、各业务单位产品数据、和整编历史数据在内的基础数据的接入服务[1-4]。但是由于部分科研项目或专业服务所需要数据格式的不同或者所需要的数据需通过进行基础数据再加工,CIMISS 提供的数据有时候并不能直接拿来使用。另外,由于 CIMISS 资料部署在国家和省级数据中心, 地县级部门在使用数据时也可能会到受网络异常因素的影响。因此,如何在不影响用户体验的前提下, 采用异步多线程的方法快速从 CIMISS 中读取数据并加工录入到本地数据库中便是本文需要解决的问题。

2 技术线路

气象数据统一服务接口(MUSIC)提供多种不同的服务方式, 包括客户端调用服务、web service、REST 服务和脚本服务。提供多种语言的客户端开发包,包括 C#、Java、C/C++、Fortran、PHP、Python 等。本文以 REST 服务和 C# 为例用于开发研究 [1-4]。REST 服务用于从 CIMISS 中提取数据到本机内存

收稿日期:2017-06-09
作者简介:黎颖智(1985-),男,本科,工程师,主要从事气象服务。

对象中，而 C# 5.0 新特性 async 和 await 进行异步操作实现主线程与方法线程的并行执行，解决提取 CIMISS 数据时出现的主界面无响应问题，批量数据的快速入库则采用 SqlBulkCopy 来实现。

2.1　多线程

如图 1 所示，线程的生命周期分为新建、就绪、运行、阻塞、死亡五个阶段。单线程程序，是顺序执行的，前一操作是否顺畅会影响到后面的操作，一旦发生阻塞便会使整个程序无法操作。如采用多线程异步执行则可避免阻塞。多线程是使用多个处理句柄同时对多个任务进行控制处理的一种特别的形式，但多线程使用了更小的资源开销。异步是指调用某一操作后，后面的操作可不等待其结果继续执行，如后面无其他操作则当前线程将会睡眠，其他线程在此时则可调用 cpu 资源。在异步操作完成后通过回调函数的方式获取通知与结果[5-6]。在 C# 编程中实现多线程的方法有以下几种：

(1)直接用 Thread 类创建；Thread newThread = new Thread(new ThreadStart(() =>{}))，参数为一个 ThreadStart 类型的委托。这是最简单的方法，但使用该方法创建的线程难于管理，若建立过多的线程反会影响系统性能，而且 Thread 对象也无法解决对于有返回值类型的委托的问题。

(2)线程池(thread pool)：线程池是通过线程共享与回收机制来减少性能开销。当程序要新建线程来执行任务时，线程池才初始化一个线程。在完成任务以后，该线程不会自行销毁，而是以挂起的方式返回到线程池中等待程序再次向线程池发出请求时再度被激活。这样既节省了建立线程形成的性能损耗，也可以让多任务复用同一线程，从而节省大量的开销。

(3)Task 方式：此方式通常用于需要循环执行任务并需要获取执行后的结果。Task 最大的优点就是任务能够控制 task 的执行顺序，使多个任务有序运行。与 Thread 对象相比，Task 对象则可轻松解决对于有返回值类型的委托的问题。

线程池与 Task 的比较：

线程池中每一次 QueueUserWorkItem 的使用都会产生一个工作项进入全局队列进行排队，最后线程池中的的工作线程以 FIFO (First Input First Output)的形式取出，任务委托的线程池不光有全局队列，而且每一个工作线程都有局部队列。当线程不足时，线程池就会创建新的线程来执行任务，直到线程池达到最大线程数(线程池满)，当 FIFO 十分频繁时，会造成很大的线程管理开销。而 Task 在嵌套的场景下，当局部队列中有多个 task，某个 task 的线程执行完任务时，该空闲线程就会从同一队列中以 FIFO 的形式分流和负载其他任务，从而减少了线程管理的开销。这些优势都是线程池所无法比拟的。

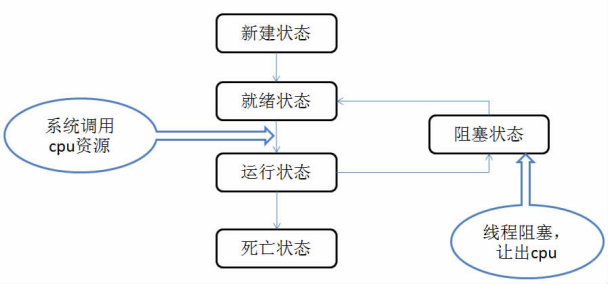

图 1　线程的生命周期

2.2　async/await

异步多线程编程最大的问题是状态、结果跟踪(即数据同步问题)，在 c#5.0 之前，多线程编程相对于单线程会出现一个特有的问题，就是线程安全的问题。所谓的线程安全就是如果代码所在的进程中有多个线程在同时运行，而这些线程可能会同时运行这段代码。如果每次运行结果和单线程运行的结果是一样的，而且其他的变量的值也和预期的是一样的。线程安全问题都是由全局变量及静态变量引起的[7-8]。为了保证多线程情况下，访问静态变量的安全，可以用锁机制来保证，但 Lock 只能锁住一个引用类型的对象。c#5.0 中加入了 async 和 await 方法来保证线程安全，async/await 将多个线程进行串行处理，等到 await 之后的语句执行完成后才执行本线程的其他语句。async/await 使得建立一个同时具备可读性与可维护性的异步解决方案变得很简单。以下是用 async/await 编写的定时自动录入数据线程的主要实现代码：

```
private async void timer1_Tick (object sender, EventArgs e)
    timer.sleep();
    {var dt = await Task <DataTable > .Factory.StartNew(() => { DataTable ta; if (i < 4) { ta = GetDataTable_time(time, i); return ta; } else { ta=null;return ta; } }).ContinueWith<ListViewItem>( r =>
    {using(SqlConnection conn =new SqlConnection ("server=(xxx);database=xxx;uid=xxx;pwd=xxx"))
```

```
{conn.Open();
using(SqlBulkCopy bulkCopy=new SqlBulkCopy(conn))
{
bulkCopy.DestinationTableName =" [dbo].[" + DateTime.Now.ToString("yyyyMM") + "] ";
bulkCopy.WriteToServer(r.Result);
SqlCommand sqlcommand = new SqlCommand(sql1, conn);
sqlcommand.ExecuteNonQuery();
}
conn.Close();}}}
```

如上代码所示，在 c#5.0 中定义异步方法和定义同步方法一样简单，在 timer1_Tick 使用关键字 await 可让<DataTable>和<ListViewItem>任务线程在后台运行而不会堵塞 UI 线程，避免出现 UI 主界面卡死的现象出现。

2.3 SqlBulkCopy

SqlBulkCopy 功能非常强大，具有快速且高效能够批量插入数据性能，较之传统的使用 SQL 语句至少快数十倍。SqlBulkCopy 类仅用于向 SQL Server 表中写入数据。但是数据的来源却不局限 SQL Server，只要能载入 DataTable 实例读取的任何数据来源均可。Microsoft SQL Server 数据库的 bcp 的常用命令行应用工具提供了用于快速批量复制录入大文件数据到库表或视图以及格式文件导出等功能。SqlBulkCopy 类可以编写提供与其相似的功能的托管代码解决方案。

在 SqlBulkCopy 类出现前，C# 在 Sqlserver 中批量插入数据一般采用的是使用 sql 语句 insert 循环逐条插入的方法，经过实验发现插入一百万条数据，大概需要 52 分钟左右，每插入一条数据耗时约 3 毫秒。而采用 SqlBulkCopy 插入同样数量级的数据仅耗时 8 秒。由此可见与常用 insert 语句相较，在需要插入数十万百万数据的时候，利用 insert 插入的速度十分慢的。其原因除了 SqlBulkCopy 原理是采用了 SQL Server 的 bcp 协议进行数据的批量复制之外，还在于 insert 语句在 for 循环中直接进行数据库操作，数据库的每一次连接、打开及关闭都是相当耗时的，虽然在 C# 中存在数据库连接池，也就是当使用 using 或者 conn.Close()进行释放连接时，其实并非真正关闭数据库连接，连接以类似于休眠的方式存在，当需要再次操作的时候，会从连接池将其唤醒。而 SqlBulkCopy 无需使用循环，便不存在这类的损耗。

3 小结

随着 CIMISS 在气象领域的应用不断加深，各级气象部门中无论是普通业务还是科研项目研究都将不可避免的涉及到 CIMISS 的使用，创建一个保证数据安全并提供快速稳定的数据接入环境是十分必要的，本文通过探讨如何采用多线程异步编程快速录入数据，为各气象业务与科研项目在解决类似问题时提供了一种可以参考的可行技术方案。

参考文献：

[1] 曹威，张冰松，李鑫.基于 CIMISS 的省市县三级气象信息传输监控系统的设计与实现[J].信息与电脑(理论版)，2017，(21)：73-76..

[2] 李志鹏，胡佳军，杨立苑，李显风，邓卫华.基于 CIMISS 的气象数据处理时效监视系统设计与实现[J].气象与减灾研究，2016，39(04)：309-313.

[3] 王宏记，杨代才.基于 CIMISS 的长江流域气象水文信息共享系统设计与实现研究 [J]. 安徽农业科学，2014，42(32)：11565-11570.

[4]]荣裕良，张霞，马忠芬，薛正平.松江智慧气象为农服务系统开发研究[J].气象研究与应用，2017，38(1)：102-106.

[5] C# 程序设计经典教程 [M]. 清华大学出版社，罗福强，2011.

[6] 陈翠娥，王学伶.C# 属性、特性和反射的应用研究[J].电脑与电信，2015，(9)：51-53.

[7] 彭庆喜，陈军威，周威.基于 C# 多线程的 Web 实体抽取设计与实现[J].软件导刊，2013，12(01)：84-86.

[8] 秦江林，符合，杨秀好，杨忠武，罗同基，雷秀峰.林业病虫害气象服务系统的创新设计与应用 [J]. 气象研究与应用，2017，38(2)：57-60.

[9] 石涛，刘军，张丽，陈金华.基于 GIS 和意愿调查法的气象为农服务效益评估 [J]. 气象研究与应用，2016，37(4)：86-89+131.