

一个基于.NET的多线程信息提取框架

鄢沛^{1,2}, 郭皎^{1,2}, 应宏²

(1. 西北大学 信息科学与技术学院, 陕西 西安 710069;

2. 重庆三峡学院 数学与计算机科学学院, 重庆 404000)

摘要:异构环境的信息提取技术在企业应用集成中有着很大的作用。介绍了.NET环境的多线程程序设计技术,分析了多线程信息提取框架的需求,提出了基于Mediator/Wrapper的多线程信息提取框架的架构模型,详细阐述了该框架的设计与实现,并分析了.NET中的线程中止方法Abort()的缺陷和改进方法。该框架的可以应用在很多领域,它的优点体现在性能、持续反馈和坚持最大响应时间上。

关键词:多线程;中介器/包装器;信息提取;.NET

中图分类号:TP311.5

文献标识码:A

文章编号:1673-629X(2007)12-0096-03

A .NET - Based Multithreaded Information Extraction Framework

YAN Pei^{1,2}, GUO Jiao^{1,2}, YING Hong²

(1. Information Science and Technology College of Northwest University, Xi'an 710069, China;

2. Mathematics and Computer Science College of Chongqing Three Gorges University, Chongqing 404000, China)

Abstract: The technology of information extraction from heterogeneous environment is very important in enterprise application integration. Introduces the technology of multithreaded programming based on .NET platform, analyses the requirement for information extraction framework, presents a model of multithreaded information extraction framework's architecture based on Mediator/Wrapper, expatiates the design and implementation of the framework in detail, and focuses on the Thread.Abort() method's limitation and improved way. The framework is useful for numerous purposes. Strong points of the framework are its performance, continuous feedback, and adherence to maximum response times.

Key words: multithreaded; mediator/wrapper; information extraction; .NET

0 引言

今天企业应用软件大多采用横向架构,这是由企业业务模式组成方式所决定的。同时客户也为了完成业务功能需求,经常把异构的系统结合起来。所以协作能力和企业应用集成是当今构建企业应用所必须考虑的问题。

在企业应用集成中,为了实时的数据报告和数据分析,经常需要把数据从异构的数据源集成到一个单一的系统中。在文中,提出了一个基于.NET的采用Mediator/Wrapper(中介器/包装器)的Internet信息提取框架,描述了一个被称为Mediator/Wrapper的并发的从不同数据源提取数据的.NET的框架。该框架由

Mediator控制,高效且可靠地搜集数据并返回结果。针对不同的资源类型,可定制不同的Wrappers,并向下提供资源兼容^[1]。Mediator(中介器)^[2]为远程处理提供统一的数据格式。基于该框架,可以快速地开发一个特定领域的Wrappers,提取异构逻辑系统的数据,然后集成一个新的应用系统^[3]。通常按串行方式执行系统完成的总时间消耗是每个任务的时间和,为了提高效率,可以考虑采用多线程的方式来并发地执行数据处理过程^[4]。并且为了提供一个有效的数据实时查询功能,可以使多个Wrappers同时工作,而网络访问消耗通常超过远程数据处理的时间,所以网络访问也必须采用并行方式。为了实现这个框架,基于.NET 2.0平台,使用C#语言开发,因为.NET 2.0提供了多线程机制和优秀的网络支持功能等特征。

1 .NET2.0中的多线程及其基本操作

操作系统使用进程将它们正在执行的不同应用程序分开。线程是操作系统分配处理器时间的基本单

收稿日期:2007-02-30

基金项目:重庆市自然科学基金(CSTC, 2005BB2001);重庆三峡学院青年项目(sxxyqn2005004)

作者简介:鄢沛(1976-),男,四川营山人,硕士研究生,实验师,研究方向为基于Internet的企业计算、Web可用性;应宏,教授,研究方向为网格计算。

元,并且进程中可以有多个线程同时执行代码。每个线程都维护异常处理程序、调度优先级和一组系统用于在调度该线程前保存线程上下文的结构。采用多线程可以快速提高应用程序的吞吐量和响应性能。

在 .NET 平台下,线程由名字空间 System.Threading 定义^[5],线程的创建可以使用如下语句:

```
Thread thread = new Thread ( new ThreadStart  
(Threadfunc));
```

```
thread.Start();
```

第一条语句创建一个新的 Thread 对象,并指明了一个该线程的方法。当新的线程开始时,该方法 also 被调用执行了。该线程对象通过一个 System.Threading.Threadstart 类的一个实例以类型安全的方法来调用它要调用的线程方法。第二条语句正式开始该新线程,一旦方法 Start() 被调用,该线程就保持在一个 alive 的状态下了,可以通过读取它的 isalive 属性来判断它是否处于 alive 状态。

Thread.Abort 方法可以用来终止线程,但是该方法并不直接导致线程终止,因为调用 Thread.Abort() 方法会引发 ThreadAbortException 异常从而导致目标线程终止,但是目标线程可在 finally 块中执行任意数量的代码。另外调用 Abort 方法时无法直接监控目标线程的执行进度等情况。在本框架的设计中,将采用一个有效的方案来解决 .NET 的这个问题。

2 框架的需求分析

信息提取框架的一个基本需求就是用户指定最大响应时间的可能性^[6]。在指定时段没有传递任何结果的 Wrappers 将会被停止。同时,正如前面所说,在 .NET 中停止一个线程不是绝对可靠,因为 Thread 类的 Abort() 方法仅仅是引发 ThreadAbortException 异常,而不能控制目标线程的具体执行的终止。而使用 Thread.Abort() 方法来停止线程会引起那些已经被锁定的所有 Monitor 解锁。如果任何先前被 Monitor 保护的对象具有不同的状态,这些被损坏的对象针对其他线程变得可见,有些潜在的行为导致无法预见的任意的结果。因此,必须寻找其他的方法来停止线程,使其在某种意义上能够达到先前提及的需求。但是,因为中止线程的 Thread.Abort() 方法是由 .NET 框架所提供,即这是 .NET 本身固有的问题,所以没有适用所有情况的解决方案。

通过 Wrappers 收集的结果可以立即呈现给用户而不会出现延迟。控制所有 Wrapper 线程的 Mediator 在获取信息提取结果时不应该出现“忙等待”现象。在结果处理过程中使用一个特殊的 iterator(迭代器,C#

2.0 中提供的一个新功能),该 iterator 允许立即获取 Wrappers 的结果,而不需要等待信息提取过程完全结束。

3 框架的设计和实现

该框架主要是在不同领域的现有方法重新组合起来的基础上进一步开发的结果,图 1 是该框架的类图,它描述了基于 .NET 的信息提取框架的类的组成和实现以及特定问题领域的用户定义类。这个框架的中心类是 Mediator。Mediator 管理 Wrapper 对象,例如,对于每一个请求,Mediator 都将创建一个会话对象负责启动和控制各自的 Wrappers 或者作业。指定的时间周期一到,还没有交付结果的 Wrapper 作业将被停止。ediator 会话联合 Wrappers 表现为一个 Listeners(监听者),提供 completed(Wrapper.Job)方法。这个方法表现为 Wrapper 的作业或他们各自线程的一个回调方法,或者表示 Wrapper 的数据提取工作已经完成。在这种情形下,Wrapper 的作业将会被添加到已完成作业列表中。使用 start(Request,int)方法,一个新的请求会被启动,同时结果列表也会被 ResultIterator(结果迭代器类)立即遍历。基于这个目的,如果有必要,hasNext()方法将等待下一个提取结果。

抽象类 Request(资源请求类)代表一个直接特定资源(如:数据源)的请求。一个 Request 可以指向任何可能的东西,例如:书籍、数码相机、股票、新闻等等。matches(Item)方法检查 item 是否确切的匹配 Request 的属性。

抽象类 Item 是一个具体实际对象的超类,根据特定请求通过 Wrappers 从 Resource(资源类)获取该类的实体可以是一些如书籍、数码相机、股票、新闻等。为了能够对 Items 结果列表排序,子类必须实现 Comparable 接口。Item 类的 getResource()方法返回 Item 对象的资源。抽象类 Resource 描述资源,担任一个通过 Wrappers 的 Items 请求的提供者。潜在的子类包括 web sites,database,news tickers 等。可以通过调用 getName()方法来确定资源的名字。

抽象类 Wrapper 的子类的对象用于控制资源请求和返回其后的结果。调用新创的 Job(作业)对象的 startJob()方法可以启动 Wrappers。使用 stopRunning()方法,Mediator 会话可以停止早期的工作线程。将在下面给出各自机制的实现。

一个 Wrapper 作业对象访问它的 Wrapper 资源对象,存放将被处理的请求的引用,以及结果列表,这个结果列表中插入了从资源返回的元素。一个拥有自己控制流的 Wrapper 作业使用工作线程(内部类 Work-

erThread)来激活,工作线程触发在 action(Wrapper.Job)中指定的动作。在这个方法中,新的元素通过调用 add(Item)方法加入到结果列表中。

线程^[5],在.NET框架中没有一个强制的方法可以停止正在活动进行的线程。所以在该框架中,采用的方法是设置一个线程可以检查到的标志变量。当线程觉

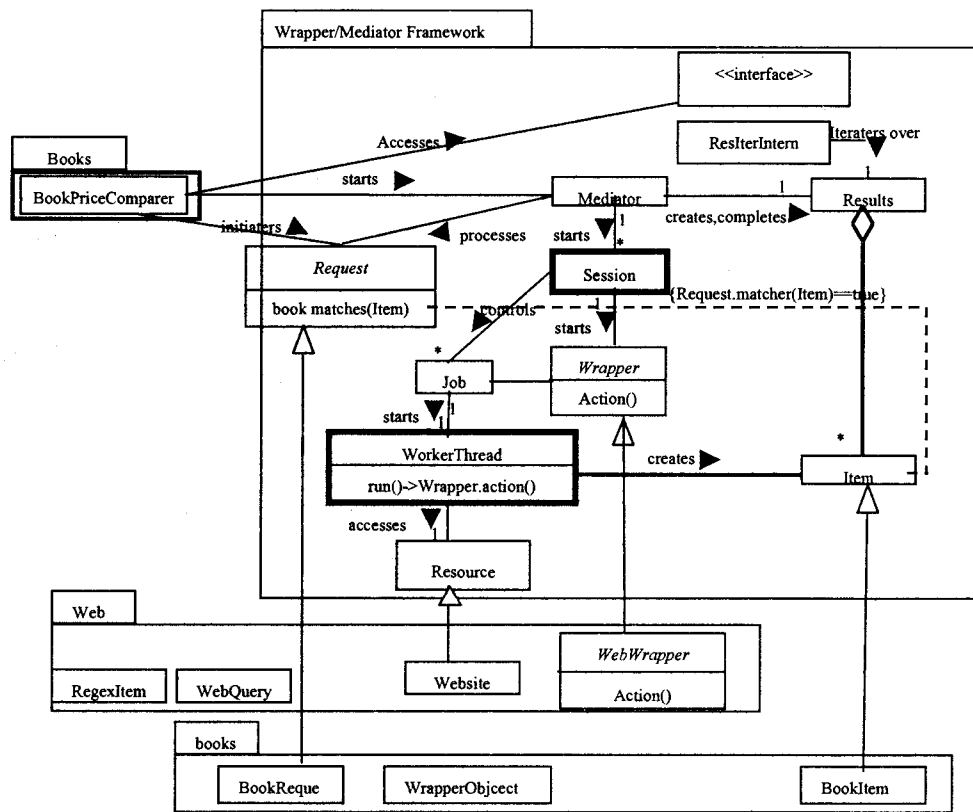


图1 信息提取框架类图

这个元素结果列表被 Results 类的对象管理,主要使用.NET 中的 ArrayList 委托管理^[5]。大多数方法是同步并支持并行访问。只要结果列表不是完备的,新的对象可以通过调用 add(Item)方法插入。由 Result 类的 isComplete()方法来确定列表是否已经完整。如同其名字,方法 waitUntilComplete()是一直等待直到结果列表完整。当这个方法被调用后,最终结果列表随后才能够被存储。有两个排序的方法可以重排结果列表,分别基于比较法或者基于特定的 comparator (比较器)对象。同时,结果列表可以使用通常的迭代程序通过 iterator()方法来遍历。结果列表包含对象的数量由调用 size()方法来决定。为了立即导航通过结果列表的一个调用 resultIterator()返回一个特定的 ResultIterator 对象。这个迭代器实现了与 IEnumerable 相似的 GetEnumerator()接口,可以被用于迭代 Mediator 会话的结果列表元素,甚至当数据提取过程还没有完成时,比如在 Wrapper 作业一直在插入结果的情况下。

如果 Mediator 会话想停止 Wrapper 作业,通常是调用 Thread 类的 Abort()方法。但因为这个方法只是引发 ThreadAbortException 异常并不能立即中止目标

察到这个标志变量被设置了,它就从 run()方法中返回。这个方案在 Wrapper.Job 类的 cancel()方法(参见下面代码)中实现,调用该方法可以停止那些必须停止的线程。

```
boolean cancel() { // of
class Wrapper.Job
if (! workerThread. isAlive
()) return true;
stopRunning(); // set Job.
running to false
workerThread. interrupt();
// try to interrupt and wait
a bit
try { workerThread. join
(100); } catch (Interrupt-
edException ignore) {}
if (! workerThread. isAlive
()) return true;
workerThread. setPriority
```

```
(Thread. MIN_ PRIORITY); // minimize damage
return false;
}
```

4 结 论

文中展现了一个基于.NET 的在不同环境中并发提取数据的框架。这个框架适合从不同的网站收集特定的数据,这也可以通过调用适当可用的 Web Services 来实现(例如亚马逊网站就提供了相应的 Web Services)。不过这个框架还是可以应用在无数的应用领域中,比如:股票查询、新闻爬虫、不同的数据库查询、基于其它技术(CORBA, RMI 等)的查询等。

通常该 Wrapper/Mediator 的信息提取框架可以应用到必须并发查询不同的系统的这些情况,由于其可以并发地持续返回结果,所以在性能上优越于其他的实现方案。

通过使用相应的应对措施来避免.NET 多线程程序设计的通用问题。并用 UML 类图表现了框架的基础结构,下一步工作是努力完善该框架的新的功能,比如日志、缓存等。

(下转第 102 页)

对图 5 中各个标识 M_0, M_1, M_2, M_3, M_4 (列向量) 中第 2 个和第 3 个元素求“逻辑或”运算, 则分别得 $M_0 = [0100]^T, M_1 = [1000]^T, M_2 = [0100]^T, M_3 = [0001]^T, M_4 = [0010]^T$ (求得新的 M_i 的四个元素分别对应原系统的 S_1, S_{23}, S_4, S_5 库所)。其中 $M_0 = M_2$, 则图 5 中节点 V_0, V_2 可以合并为一个节点 V_{02} , 二者之间的变迁如 t_5, t_6 省略, 二者向同一外部构件的变迁如 t_1, t_2 可以合并为 t_{12} , 则可以快速得到新系统的可达标识图, 如图 6 所示。其结果与先合并构件得到系统 SA 模型再求其基于 Petri 网的 RMG(PSA) 一致, 时间复杂度为 $O(3 * n)$ 。

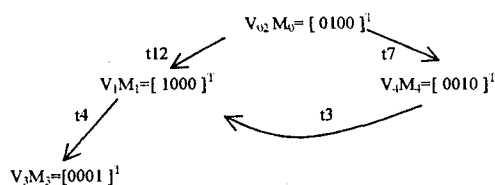


图 6 合并后的 RMG

(5) 拆分 SA 中构件 C_j 成若干个构件 $C_{j1}, C_{j2}, \dots, C_{jn}$ 。其过程与合并互为逆过程, 也能快速求取新系统的可达标识图, 从而分析新系统的各项特性。这里不再重述。

2) 系统活性分析。

建立基于 Petri 网的 SA 模型除了可以定量分析构件贡献大小即演化时的波及效应外, 还可以来分析所建立的系统的活性 (如系统是否存在死锁)。若一个 Petri 网中不存在死锁, 则 Petri 网具有活性, 一个 Petri 网死锁时, 它的标识是死标识^[8]。因此, 可达标识图中若有终止结点, 则 Petri 网可能死锁。例如, 图 5 中 $[00001]^T$ 是终止结点, 所以图 4 中 Petri 网可能死锁。

除此之外, 建立基于 Petri 网的 SA 模型还可以进行系统故障诊断与检测, 还可以利用 Petri 网技术设计自校正能力的 SA 模型, 相关理论还有待进一步探讨。

总之, 利用文中提出的 PSA 模型建立的可达标识

图 RMG, 可以用来完成软件架构的分析, 与系统可达矩阵相比, 所需的时间复杂度低, 并且直观明了, 易于计算。

2 结束语

运用基于 PSA 模型的可达标识图 RMG 来分析软件架构性能, 一方面能给系统架构师提供系统信息参考, 如构件贡献大小可以帮助合适地改变系统架构, 同时也可适当地更改原有系统可达标识图快速分析变更后新系统的特性; 另一方面, 首次用 Petri 技术分析系统架构性能为软件架构研究提供了全新的一个思路。进一步的工作是利用 Petri 网辅助完成软件架构其他方面的设计, 如辅助完成系统运行时的故障诊断与自校正设计。

参考文献:

- [1] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline[M]. Upper Saddle River: Prentice Hall, 1996.
- [2] 张世琨, 王立福, 杨美清. 基于层次消息总线的软件体系结构风格[J]. 中国科学(E 辑), 2002, 32(3): 393-400.
- [3] Petri C A. Kommunikation mit automaten[D]. Bonn, West Germany: University of Bonn, 1962.
- [4] 王映辉, 张世琨, 刘 瑜, 等. 基于可达矩阵的软件体系结构演化波及效应分析[J]. 软件学报, 2004, 15(8): 1107-1115.
- [5] Garlan D, Shaw M. An introduction to software architecture [R]. [s. l.]: Carnegie Mellon University, 1994.
- [6] 冯 冲, 江 贺, 冯静芳. 软件体系结构理论与实践[M]. 北京: 人民邮电出版社, 2004.
- [7] Murata T. Petri nets: properties, analysis and applications[J]. Proceeding of IEEE, 1989, 77(4): 541-582.
- [8] 袁崇义. Petri 网原理与应用[M]. 北京: 电子工业出版社, 2005.

(上接第 98 页)

参考文献:

- [1] Barillot C, Benali H. Federating Distributed and Heterogeneous Information Sources in Neuroimaging: The NeuroBase Project [EB/OL]. 2005. <http://www.irisa.fr/visages/demo/Neurobase/Download/IRISAResearchReport1712.pdf>.
- [2] Josifovski V, Risch T. Query Decomposition for a Distributed Object-Oriented Mediator System[J]. Distributed and Parallel Databases, 2002, 11(3): 307-336.
- [3] 袁晓洁. 基于 Mediation 的异构数据集成系统 HDIS 设计与实现[J]. 计算机工程与应用, 2006(10): 162-165.
- [4] Schneider R. Work Queue based multi-threading[EB/OL]. 2005. <http://www.codeproject.com/csharp/workqueuethreading.asp>.
- [5] Microsoft. MSDN Library for Visual studio. Net 2005[DB/CD]. Microsoft corporation, 2005.
- [6] 房 俊. 一个基于中介机制的动态数据集成框架[J]. 计算机科学, 2003(增刊): 124-126.