文章编号:1001 - 9081(2007)09 - 2304 - 03

基于构件的并行测试系统 TPS 设计与实现

夏 锐,肖明清,付新华,程进军
(空军工程大学 自动测试系统实验室, 西安 710038)
(xiaruiww@163.com)

摘 要:并行测试系统的测试程序集(TPS)目前尚无成熟实用的开发方法。分析了多线程并行测试系统的体系结构,给出了其 TPS 的用例模型。在此基础上,按分层设计的原则,建立了基于构件的层次式框架。最后设计实现了基于 COM 接口的支持并发操作的功能构件。该开发方法为并行测试系统 TPS 的快速开发打下了良好的应用基础。

关键词:并行测试;自动测试系统;测试程序集;多线程;构件
中图分类号: TP274 **文献标志码**:A

Parallel TPS design and application based on components

XIA Rui, XIAO Ming-qing, FU Xin-hua, CHENG Jin-jun
(*Automatic Test System Laboratory, the Air Force Engineering University, Xi' an Shaanxi* 710038, *China*)

Abstract: At present, there are no practical and mature R&D methods for the design and implementation process of the parallel test system's Test Program Set (TPS). Therefore, first, the multithreading architecture of parallel test system was presented, and the user case model of the TPS was set up in this paper. Then, the framework of multi component layers was proposed based on it. Last, the application of parallel processing components based on COM was introduced. The method boosts the design and application of the parallel TPS.

Key words: parallel test; automatic test system; Test Program Set (TPS); multithreading; component

0 引言

并行测试技术(Parallel Test)是自动测试系统(Automatic Test System, ATS)在进一步减少测试时间、降低测试成本的发展趋势下兴起的一项新技术,是下一代 ATS"NxTest"的关键技术之一。它正以不可比拟的优势成为未来 ATS 发展的热点,将是下一代 ATS 的主要特征之一[1-4]。

并行测试主要由多个工作线程同步或异步地并发执行多项测试任务来实现。然而,这些线程往往会竞争共享资源,容易产生死锁、饿死等错误,且问题难以复现,增加了并行测试系统测试程序集(TPS)的复杂性。因此并行测试技术的关键在于其 TPS 的设计。

较大规模的复杂应用系统一般采用基于构件的软件开发方法。基于构件的软件开发不仅具有诸如方便开发和维护、可移植性强、复用性好等优势,而且积累了支持开发实时并发系统的丰富经验和方法模式。因此,本文采用"应用程序框架 + 支持并发构件"的开发方法,设计了并行测试系统 TPS 的构件框架,介绍了基于 COM 接口的支持并行操作构件的实现,为并行测试系统 TPS 的快速开发打下良好的基础。同时,基于构件的 TPS 提高了系统的可靠性,方便了系统的测试、维护与升级,使并行测试系统的应用具有良好的可扩展性。

1 并行测试系统的体系结构

基于多线程结构的并行测试系统是目前并行测试实现的主要形式,图 1 为其体系结构图。

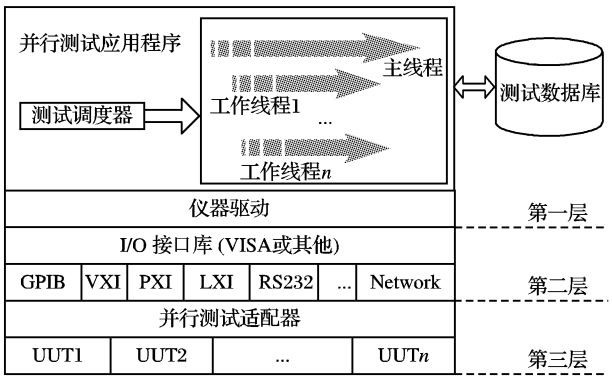

图 1 多线程并行测试系统体系结构

系统结构可分为三层,最高层为并行测试应用程序。调度器按一定的调度策略来分配多个测试工作线程在单个或多个处理器上的运行时间,从而在宏观上表现出多个测试任务同时运行的效果。测试任务的工作流程通过调用测试数据库来完成,测试结果也保存到其中。该层底层是仪器及激励的驱动层,用来管理和控制系统中的仪器设备;第二层 I/O 接口层是系统软件与硬件间执行通信的底层函数库。最底层是硬件资源层,通过并行测试适配器将被测件(UUT)与系统进行电气连接,并可执行多个 UUT 间的切换。

分析系统体系结构的目的是在系统的应用与开发之间形成统一、完整的功能视图,作为建模、设计和规划系统的基础。

收稿日期:2007 - 03 - 09;**修回日期**:2007 - 05 - 23。
基金项目:总装十一五重点预研项目(51317030103);电子测试技术国防科技重点实验室基金项目(51487020305JB3201)。
作者简介:夏锐(1982 -),男,安徽含山人,博士研究生,主要研究方向:武器系统检测自动化与智能化; 肖明清(1963 -),男,湖南常德人,教授,博士生导师,主要研究方向:自动测试系统; 付新华(1980 -),男,湖南攸县人,博士研究生,主要研究方向:武器系统检测自动化与智能化; 程进军(1979 -),男,湖北襄樊人,讲师, 博士,主要研究方向:自动测试系统。

2　基于构件的并行测试系统 TPS 架构

从并行测试系统的体系结构分析可以看出,并行测试系统设计的核心在其 TPS 的设计开发。首先分析了并行测试系统的用例模型,接着设计了并行测试系统 TPS 的构件框架,最后通过实例介绍了支持并行操作构件的实现方法。

2.1　并行测试系统 TPS 用例模型

用例模型开发是并行测试系统需求分析阶段的首要任务,图 2 为系统的 Use Case 图。

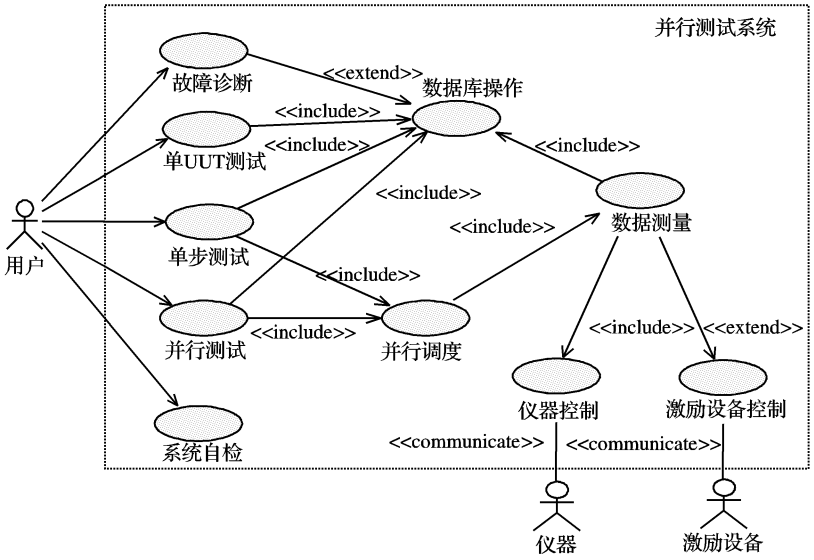

图 2 并行测试系统用例图

从用户的角度来看,并行测试系统应可进行传统的单 UUT 测试、单步测试和并行测试。测试前能够对系统进行自检;测试后可调用测试数据库查看测试结果,分析数据以进行故障诊断。系统按 UUT 一定的测试流程进行数据测量,测试流程通过调用测试数据库实现。这样可以提高系统的通用性,测试不同的 UUT 只需调用不同的数据库,而不必重新修改代码。并行调度是并行测试系统的核心用例,它通过一定的策略来控制多个测试线程、对共享资源进行调度使用,从而实现线程间的同步并避免产生死锁、饿死等问题。

2.2　并行测试系统 TPS 构件框架

针对并行测试系统的功能特点,按分层设计的原则[5],提出了如图 3 所示的基于构件的并行测试系统 TPS 的层次式框架,如图 3 所示。

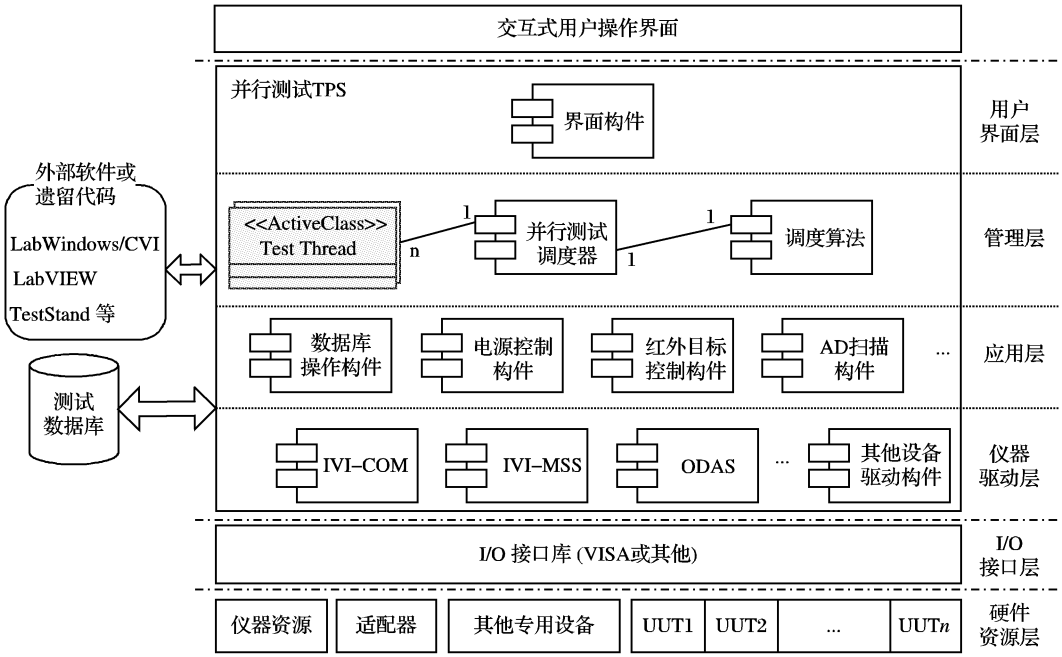

图 3　基于构件的并行测试系统 TPS 层次式框架

整个 TPS 分为四个层次。从上至下依次为:

1)用户界面层。主要由各种界面构件组成,全面实现系统与用户间的交互。通过它用户能够进行测试信息的输入、测试任务的选择以及测试信息的保存、打印和历史记录查询等功能。该层有成熟的构件库供应。

2)测试管理层。该层是交互式界面的后台执行者,它能够响应用户的操作,由并行测试调度器管理调度多个工作线程通过一定的调度算法,调用系统中的可用资源,完成用户指定的操作。各工作线程调用下层的功能构件完成相应功能任务。

3)测试应用层。该层以构件的形式封装了并行测试系统中各种可用资源,如数据库操作构件、电源控制构件、红外目标控制构件等各种对测试仪器、设备的操作构件,提供给上层应用程序框架调用。该层可从以往建立的测试功能构件库中进行裁取或改造。可最大限度地实现代码重用,加速了系统的开发。

4)仪器驱动层。该层由仪器生产厂家提供的仪器驱动程序以及自研设备的驱动程序组成。通过 TPS 外部下层的 I/O 接口层,该层实现 TPS 与硬件资源的通讯并能驱动硬件资源执行相应的功能。该层已有的 IVI 驱动库、ODAS 等都是基于 COM 构件接口的,可方便地实现与上层的无缝连接。

这四个层次之间是层层调用的关系:测试应用层中的各个功能构件通过调用仪器驱动层的服务构件来完成对硬件资源的操作。构成系统应用界面的界面构件通过调用测试管理层中的构件完成用户定义的各项测试任务。测试管理层中的并行测试调度器构件在一定的调度算法的控制下,对多个测试工作线程进行调度,调用各个功能构件,完成各项测试任务。

2.3 支持并行操作的构件设计

由并行测试的需求分析确定了构件的并行性要求。支持并行操作的构件要求采取一定的机制,避免由于多个工作线程的同时调用而对共享资源的使用产生冲突[6-8]。本文通过使用锁的方法来实现支持并行操作构件的设计。以 AD 仪器模块的使用构件的开发为例进行说明,其构件结构图如图 4 所示。

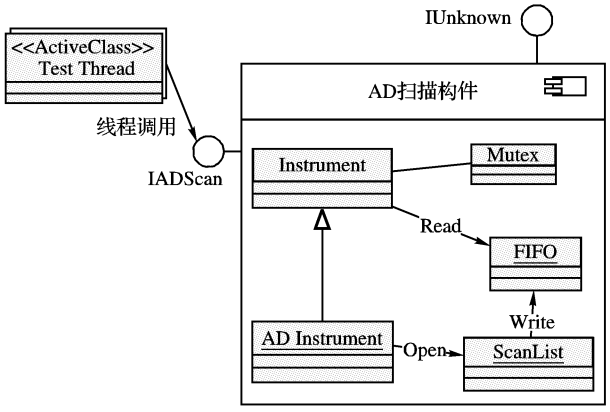

图 4 AD 功能构件结构

由于 AD 仪器模块执行 AD 转换的特殊性,它不能同时被两个线程调用,因此必须设置锁机制来同步两个以上线程对 AD 功能构件的调用。锁机制通过互斥类 Mutex 来实现。

限于篇幅,本文采用基于 COM 接口的构件实现主体框架简略如下:

```
//AD 仪器类、ScanList 类、FIFO 类的定义
class ADInstrument : public instrument
{
  public:
    Initialize ();
    Rest ();
    ReadFIFO ();
    …
  private:
    …
};
class ScanList                                    //具体定义略
class FIFO                                        //具体定义略
//AD 构件接口定义
Interface IADScan : IUnknown
{
  Virtual void__stdcall FADScan();
}
//AD 功能构件定义
Class CADScan : public IADScan
{
  public:
    //IUnknown 接口定义略
    //IADScan 接口实现
    Virtual void__stdcall FADScan ()
    {
      AD 参数定义及动作函数实现
    }
  private:
    CMutex Mutex (FALSE, NULL, NULL);
    ScanList ADscanlist;
    FIFO ADfifo;
    …
};
```

其协作工作模式如图 5 所示。

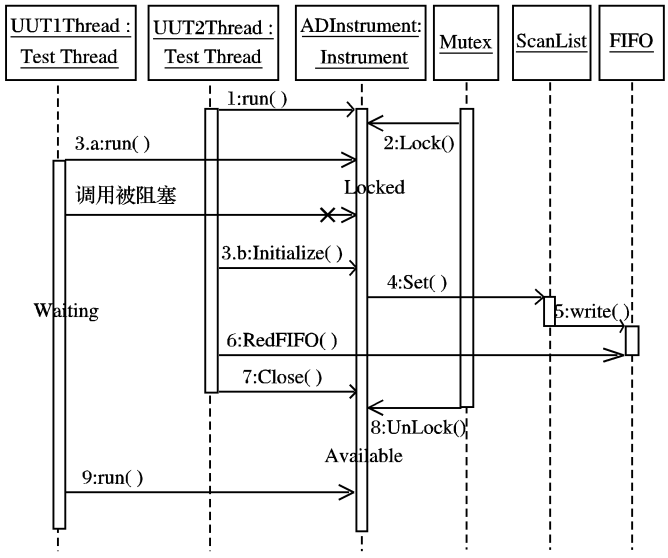

图 5 AD 功能构件顺序图

3 系统实现

专用的测试工程语言如 LabWindows/CVI、LabVIEW 及 TestStand 等目前还不支持面向对象的开发,因此还需要一定的工作量。VC++语言功能强大,可靠性高,可移植性好,且提供了丰富的支持多线程的并行机制,用它编写多线程程序简单、方便和安全。由分析设计阶段得到的 UML 可视化模型还可以在VC++中自动生成代码框架,实现了系统的快速开发。

4 结语

传统 ATS 的 TPS 体系结构的面向对象、基于构件及模式等设计方法已越来越走向规范和成熟。并行测试系统作为新兴系统,其 TPS 的设计与开发目前国内外尚无统一的模式和标准。因此,持续深入地开展并行测试系统 TPS 开发设计的方法及模型研究,对推动并行测试技术的发展是十分必要和重要的。

(下转第 2309 页)

如下:

```
1)   A+ := Θn
2)   C := Θn+1
3)   k := B(1,1)
4)   C(k) := 1
5)   for i from 1 to e do
6)     A+ ( B(i,2)) := A+ ( B(i,2)) + 1
7)     if B(i,1) not equal k then
8)       k := B(i,1)
9)       C(k) := i
10)    end if
11)  end for
12)  C(n + 1) := e + 1
13)  for i from 1 to n do
14)    if A+ (i) equal 0 then
15)      Snew := Snew ∪ {i}
16)    end if
17)  end for
18)  repeat
19)    Sold := Snew
20)    Snew := ϕ
21)    for all i such that i ∈ Sold do
22)      if C(i) not equal 0 then
23)        m := C(i)
24)        l := i + 1
25)        repeat
26)          t := C(l)
27)          l := l + 1
28)        until t not equal 0
29)        for j from m to ( t -1) do
30)          A+ ( B(j,2)) := A+ ( B(j,2)) - 1
31)          if A+ ( B(j,2)) equal 0 and B(j,2) ∉ Snew then
32)            Snew := Snew ∪ { B(j,2)}
33)          end if
34)        end for
35)      end if
36)    end for
37)  until Snew equal ϕ
```

在此处理过程中,第 5 ~ 11 步的循环是实现式(2)的计算,并且提取数组 B 的结构信息存储在 C 中;第 13 ~ 17 步的循环是搜索原图中的所有无前驱顶点,初始化 S_{new};第 18 ~ 37 步是拓扑排序的主要迭代过程,其中,第 30 步是进行弧删除操作,第 31 ~ 33 步是搜索新产生的无前驱顶点。

表格 1 比较了三种使用邻接矩阵的拓扑排序算法的存储和计算复杂度,这三种算法分别为:

1)算法 1 是我们提出的算法;

2)算法2[3]是使用方阵形式的邻接矩阵,将某一行全部置零进行弧删除操作,当检测到某列的元素之和为0,则说明顶点的入度为0,则相应的顶点即可输出;

3)算法 3 和算法 1 使用相同的存储结构,若某一顶点的编号不出现在矩阵 B 的第2列中,则说明顶点的入度为0,即可输出。要删除从顶点 i 所发出的弧,则可处理为将矩阵 B 第1列中值为 i 的元素所在行的第2列也令为 i(同时也标记了顶点 i 已经输出)[7]。

表 1　三种算法存储和计算复杂度对照表

比较项	算法 1	算法 2	算法 3
存储空间	$O(e)$	$O(n^2)$	$O(e)$
计算量	$O(e)$	$O(n^2)$	$O(ne)$

从表格 1 可以看出,与直接使用方阵形式邻接矩阵的算法 2 相比,算法 1 的存储和计算复杂度都大大减少;而与使用相同存储结构的算法 3 相比,算法 1 的存储复杂度与之相当,实际上是稍有增加,主要是使用了 A^+ 和 C 的缘故,但在计算量上却有大大的改进,因为本算法利用了数组 B 的结构信息以及根据 DAG 的性质 1,限定了选取无前驱顶点和进行弧删除操作的搜索范围,将选取无前驱顶点和进行弧删除结合在一起进行,相对于原来的需要对数组 B 扫描 $2n$ 遍[3],本算法只需要扫描 2 遍,并且只需要进行 e 次加法和 e 次减法。

5　结语

相对于单顶点算法框架,集合算法框架可以运用集合论等数学工具,更容易反映拓扑排序处理过程中无前驱顶点选取、弧删除等操作的数学实质,更有利于从宏观上减少循环次数来降低拓扑排序算法的计算量,不仅适用于这里的一般拓扑排序问题,而且更加适用于各种具有特殊需求而存在限制条件的拓扑排序问题。

参考文献:

[1] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 1992.

[2] 王晓瑛, 魏正军. 关于拓扑排序算法的讨论[J]. 西北大学学报: 自然科学版, 2002, 32(4):344 - 346.

[3] 徐绪松. 一个新的拓扑排序算法[C]//第三届全国数据结构研讨会论文集. 上海: 上海科技出版社, 1993:165 - 169.

[4] CORMEN T H, LEISERSON C E, RIVEST R L, *et al.* Introduction to algorithms[M]. 2nd ed. Cambridge: MIT Press, 2001.

[5] 屈长青. 邻接矩阵的应用[J]. 郴州师范高等专科学校学报, 2000, 21(6):19 - 23.

[6] MAYEDA W. 图论[M]. 葛真, 钟宁晖, 邓祖善, 等译. 贵州: 贵州人民出版社, 1985.

[7] 傅清祥, 王晓东. 算法与数据结构[M]. 北京: 电子工业出版社, 1998.

(上接第 2306 页)

参考文献:

[1] 夏锐. 并行测试系统设计与应用[D]. 西安: 空军工程大学工程学院, 2005.

[2] 夏锐, 肖明清, 朱小平, 等. 并行测试技术在自动测试系统中的应用[J]. 计算机测量与控制, 2005, 13(1):7 - 10.

[3] 朱小平, 肖明清, 夏锐. 基于 RUP 过程的并行测试建模分析与设计[J]. 空军工程大学学报: 自然科学版, 2005, 6(6):63 - 65.

[4] 夏锐, 肖明清, 赖根. 并行测试设计与开发[J]. 计算机测量与控制, 2006, 14(7):841 - 843.

[5] 黄柳青, 王满红. 构件中国[M]. 北京: 清华大学出版社, 2006.

[6] GOMAA H. 用 UML 设计并发、分布式、实时应用[M]. 吕庆中, 译. 北京: 北京航空航天大学出版社, 2004.

[7] DOUGLASS B P. 实时设计模式[M]. 麦中凡, 陶伟, 译. 北京: 北京航空航天大学出版社, 2004.

[8] HUGHES C, HUGHES T. C++ 并行与分布式编程[M]. 肖和平, 译. 北京: 中国电力出版社, 2004.