

基于多线程技术的自动测试系统优化设计

赵源, 姜小峰*

(苏州大学 计算机科学与技术学院, 江苏 苏州 215006)

(*通信作者电子邮箱 xjiang@suda.edu.cn)

摘要:传统的测试过程对系统性能考虑较少,但随着并行测试方法的广泛应用,对系统性能和数据吞吐量的要求越来越高,利用多线程技术优化软件设计成为有效提高自动测试系统性能的途径之一。对测试过程流水线现象进行建模,采用异步流水线设计模式,结合面向任务的概念,提出了一种适用于测试系统的编程模型。实验结果表明,该模型在测试任务随机输入的条件下可明显缩短样本的平均测试时间,通过在交流接触器特性参数测试实例中的具体应用,表明该模型不仅能够增加测试项目配置的灵活性,而且可以避免测试系统中多线程编程的复杂性。

关键词:自动测试系统;多线程;生产者/消费者模式;异步流水线;接触器

中图分类号: TP302.1; TP274.2 **文献标志码:** A

Optimized design for automatic test system based on multithreading

ZHAO Yuan, JIANG Xiaofeng*

(School of Computer Science and Technology, Soochow University, Suzhou Jiangsu 215006, China)

Abstract: The traditional testing process does not specifically consider the system performance. With the wide application of parallel testing method, more attention was paid to the system performance and data throughput capacity. Optimizing the software design with multithreading technology becomes an effective way to improve the performance of automatic test system. By modeling testing pipeline process, using asynchronous pipeline design patterns and combining task-oriented concepts, an available test system programming model was proposed. The experiment results prove that the model can significantly shorten the average test time in the ideal case of random input of test tasks. Applying this model to an instance of measuring characteristic parameters of Alternating Current (AC) contactor, the results further indicate that this model can significantly increase the flexibility of test configuration and avoid the complexity of multi-threaded programming.

Key words: automatic test system; multithreading; producer/consumer pattern; asynchronous pipeline; contactor

0 引言

传统的测试过程是测试任务依次产生并按四个阶段串行进行的,这种方式的不足之处在于当测试任务处于某个处理阶段时其他阶段能利用的资源将被闲置,当有多个测试任务且其各个处理阶段占有资源种类和数量有很大不同时资源闲置的情况更为严重。为了有效解决这一问题,可以引入并行测试的方法。在实际应用中,为了减少样品的平均测试时间,某些情况下最可行的办法是增加并行测试的样品数量。现代多核心和多线程处理器的普遍应用,为基于并行测试的优化带来了契机,通过充分利用计算机多核心的功能增加并行处理能力,从而提高经济效益,这在生产过程中具有广泛的实际需求。

应用并行测试技术可大幅度提高测试系统的吞吐量,减少测试成本^[1]。采用并行技术优化测试过程的方法,目前主要集中在硬件与软件两方面^[2-3]。本文关注的重点是测试系统软件体系的优化效果。

目前针对现代多处理器计算机体系、并行机制中的多线程和事件驱动等方面已有了相当深入的研究,提出了一些同步或异步并行处理框架^[4-5]并应用于实际生产中,如将异步

并行框架应用于网络服务^[7],该框架极大地提高了网络服务吞吐量,对如何将并行处理技术应用于测试领域有启发意义。文献[1]介绍了多种并行测试技术,其中提到了流水线^[6]优化思想,但只是一种笼统的提法,没有具体实现,文献[7]虽然应用了流水线设计模型,但设计中只处理了两个流程,且两个流程的阶段划分缺乏通用的结构。

并行测试系统的研究过程中出现了各种任务调度算法,如将随机理论引入并行测试任务的调度算法^[8],将退化模拟结合赋时 Petri 网的调度算法^[9],这些调度算法比较复杂,并且只能用于某些应用。本文对测试资源进行了分类,将自动测试过程分为若干阶段,增强了测试任务的独立性,简化了调度过程。

关于自动测试领域并行建模的研究,目前有将面向对象、框架和模式等软件工程的理念应用于自动测试领域^[10],但缺乏对测试系统并行方面主题的建模。以 NI 公司为代表的虚拟仪器平台将大量的多线程技术应用于自己的产品^[11],但 NI 公司的技术文档缺乏系统性的阐述。

本文在总结和分析以上研究成果的基础上,对整个测试过程建立模型,结合面向对象的编程思想,设计了一种适用于整个测试过程的异步流水线多线程应用模型,通过充分利用

收稿日期: 2013-12-19; 修回日期: 2014-03-06。

作者简介: 赵源(1984-),男,陕西榆林人,硕士研究生,主要研究方向:嵌入式系统; 姜小峰(1972-),男,江苏丹阳人,副教授,博士,主要研究方向:计算机图形图像处理、计算机测控技术。

多线程,合理集中测试资源,利用异步事件驱动技术提高测试系统并行性、增强配置灵活性以及避免多线程编程复杂性^[5]。

1 优化模型分析

1.1 关键概念

流水线 流水线作为一种软件设计模式也被称为管道-过滤器(Pipes-and-Filters)模式,一般的应用方式是将进程作为处理单元,每个处理单元从上游处理单元获得数据并处理,然后传递给下游处理单元继续处理,那么更多的情况是将线程作为处理单元。无论哪种模式,最终的目的是最小化 CPU 空闲时间,这也是本文对测试系统进行优化的一个依据。

生产者/消费者 这是一种高效的数据通信方法,其重要的特点是生产者单元和消费者单元可以以不同的速率处理数据,这一异步特性可以有效协调测试系统各种资源的使用。

多线程 它是现代多核心并行技术的关键概念之一,在流水线模式中,处理单元的角色由线程担任。多线程技术在基于虚拟仪器的测试系统中有着广泛应用,一般基于两个原因^[11]:一是实现比较好的用户体验^[12],一般由一个线程处理数据采集;另一个线程处理用户界面响应;另一原因则是解决处理速度匹配问题,数据采集和数据分析处理速度不同而需要应用多线程进行协调。本文的论述主要基于第二个原因。

1.2 模型提出

在实际应用中,测试系统往往需要实现多个测试目标,每个测试目标的完成涉及不同类型与数量的资源,图1表示了自动测试系统提供的各种资源。

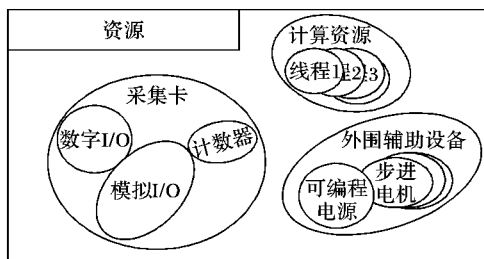


图1 测试系统中的各种资源

在测试系统中资源是稀缺的,并且某些资源具有独占性,因此需要对各种资源进行精确的协调配置。协调过程就是如何有效有序地利用这些资源的过程,处理步骤也被称为算法,这种将资源信息与处理算法集合在一起的思想,抽象出了测试任务的概念。面向任务的设计方式,让测试系统设计者可以将注意力集中在单个测试任务上,从而快速分析出测试系统的测试要求。

在数据源源不断流动的过程中如何发挥系统的最大性能,这是对优化过程的一个直观的认识。要达到这样的目的,测试过程阶段明确划分是建立流水线模型的关键。

基于测试过程的有序性,本文抽象出测试过程的若干阶段,可以分为数据采集、分析数据、显示数据和存储数据阶段,如图2所示。每个阶段占用资源的数量和类型不同,表1给出了各阶段资源占用情况。

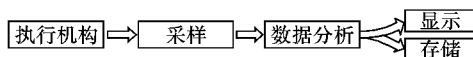


图2 测试过程的流程分析

表1 各阶段资源占有情况

处理阶段	资源
采集阶段	数据采集卡和外围设备
分析阶段	CPU 资源
存储阶段	部分 CPU, I/O, 网络带宽
显示阶段	部分 CPU, 显示界面

如图3所示,如果测试任务的每个阶段都由独立的线程来执行,称这种方式为流水线方式。



图3 流水线操作

设每个处理单元处理时间为 t ,经过 $3t$ 时间后,处理单元将并行运行,资源没有闲置得到了充分利用,任务的平均处理时间大约为 t ,流水线将系统效率提高了接近3倍。但是事实上实际情况会很复杂,如各个处理单元的处理时间不完全相同,占用的资源数量和类型各不相同,不同的任务可能通过不同的处理单元,并且处理单元的划分也不止3个,此时异步的设计方式是必要的。

此外,在各个执行阶段中任务的明确划分,明确了执行目标,使得各阶段算法修改和调整非常方便。

基于上述讨论,本文设计了测试过程的异步流水线模型。

流水线模型的关键和核心是实现各个阶段的信息传递,按照流水线设计模式,信息通过基于FIFO队列的管道来进行传递,对应于一般的测试过程,可以提取出4个队列,如图4所示。

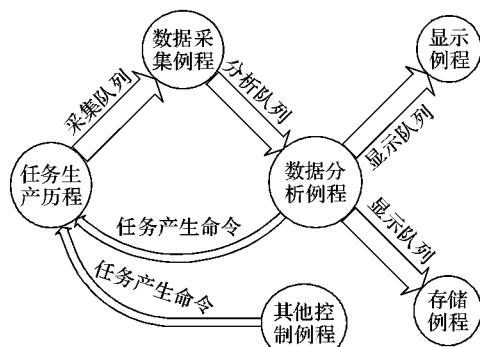


图4 系统中的4个任务队列

队列的具体使用方式是,前一个阶段将处理后的信息加入队列,后一个阶段则从队列中取出信息。第一个是采集队列,由任务产生线程向其添加任务,采集线程提取任务并采集数据;第二个是分析队列,由采集线程向其添加任务,分析线程提取任务并进行分析;第三个是显示队列,由分析线程添加任务,显示线程提取任务更新控制界面;第四个则是存储队列,由分析线程添加任务,由存储线程读取任务,将数据存入文件或数据库,并进行可能需要的网络通信。

测试过程的某一阶段由进入该阶段的队列、主线程和若干空闲的线程组成,主线程负责检查队列,根据队列中任务个数以及空闲线程的个数综合分配执行某任务某阶段处理函数的线程。处理函数执行完后,将结果写入任务数据结构的结

果域,然后调用该阶段出口函数,将任务写入下一阶段的队列,写入过程需要保证线程访问安全。图5是测试过程的基本流水线的模型。

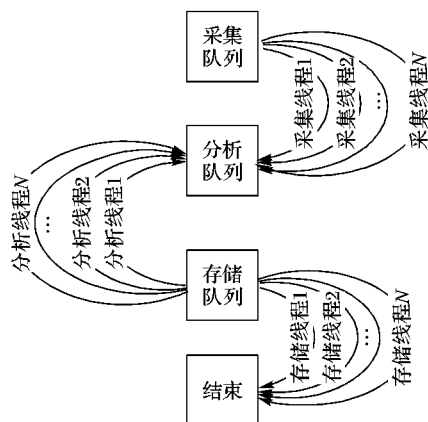


图5 测试过程流水线模型

2 优化模型的实现

上述优化模型的具体实现主要包括三个方面:事件驱动机制与各阶段的信息传递,测试任务的组成结构和运行效率的统计。下面结合作者在工作过程中所使用的 NI LabWindows/CVI 的编程语言,分别介绍这三个方面的具体实现方法。

2.1 信息传递与事件驱动

各测试阶段之间信息的准确传递是系统实现的一个重要环节。在具体的实现过程中,通过队列的数据结构来进行描述。

虽然 LabWindows/CVI 直接提供了安全队列,并提供非常方便的事件驱动编程模式,但在具体实验过程中,我们发现存在两个问题:

第一,LabWindows/CVI 的官方文档中说明配置合适的模式可以实现向安全队列每写入一次数据就触发一次回调函数,但实际情况下,如果向队列写入数据太快,那么一对一的回调函数触发会发生丢失现象。

第二,回调函数运行时,向队列写入数据,如果运行之后没有继续向队列写入数据,回调函数将无法再次触发,那么回调函数运行期间写入队列的数据将得不到及时处理。

而本文提出的模型,实际是测试任务在各个阶段的顺序传递,功能单一且进入队列的数据具有突发性,所以需要设计简单而合理的信息传递和事件驱动机制。

首先构造一个简单的队列,队列中存放测试任务,提供三种操作,入队、出队以及获取队列元素个数的方法,具体采用循环数组来实现,并保障线程访问安全。

其次,通过一个异步时钟定时器^[12]实现事件驱动,将定时器的执行线程作为分派程序,实现一个定时循环,执行算法1,如图6所示。

算法1:

```

If 队列中有数据 AND 有空闲线程 Then
{
    获得队列的锁
    For( Min( 空余线程数, 队列测试任务个数) )
    {
        调度线程运行任务相关阶段的处理函数
    }
}

```

```

}
释放队列锁
}
间隔 50 ms, 下一次循环

```

除了分派线程,各个阶段还需要若干空闲线程作为运行资源,线程数可固定或动态配置,本文采用固定设置的方法,通过线程池实现。

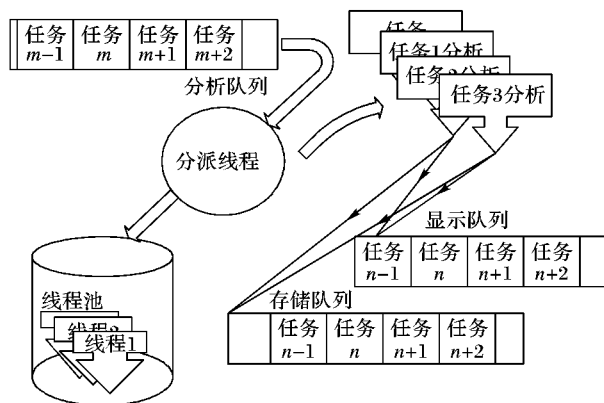


图6 某个阶段的处理过程

测试过程中,完成某项测试要求的数据采集时间一般在秒级,而 Windows 系统异步时钟线程间隔在 20 ms,这样的事件驱动响应符合一般的测试过程。

需要注意的是针对不同的任务,某阶段处理的运行时间会有不同,此时需要一个公共的写入函数以实现同步,保证多个线程同时向一个队列写数据的情况不会发生,该写入函数被称为该阶段的出口。

另一个问题是向队列添加数据的形式。模型中的队列被看作是一种具有 FIFO 缓冲效果的信号传递通道,队列中存储的是指向任务的指针或引用,通过指针或引用找到关于测试任务的所有信息,如图7所示。这种只传递指针或引用的方式,极大地降低了数据复制的开销。

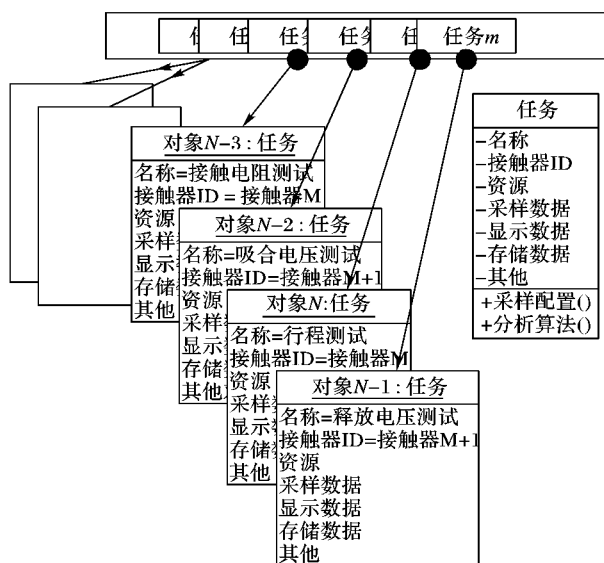


图7 队列中的数据

2.2 面向任务的实现

整个测试系统的运行围绕任务展开,从任务的建立到任务各个阶段处理函数的执行,最后到任务的结束。

一个测试目标的具体实现需要集中于任务相关的所有信

息。一个任务包含的信息有测试任务的标识、任务完成的资源(包括采集卡、CPU和屏幕显示资源等)以及任务各个阶段的处理线程,使得对任务某些方面的修改变得集中和方便。这种将数据与方法封装在一起方式,体现了面向对象的思想。

任务(Task)数据结构定义如下:

```
struct task
{
    char taskName[50];           // 任务的名称
    double * ChannelData[16];    //按通道分开后的采集数据数组
    int ChannelDataReferenceCount[16];
                                //标记采集数据,实现共享机制
    DAQ_INFO * dataInfo;         //采集卡信息
    TEST_RESULT testResult;      //分析结果
    double timeStatistic[6];
    //记录各个阶段运行时间,用于性能分析
    int (* acquisition)(struct Task * task);
    int (* analyse)(struct Task * task);
    int (* display)(struct Task * task);
    int (* store)(struct Task * task);
    //各个阶段的处理函数
}
```

各个阶段的处理线程与具体不同任务的特性直接相关,不同的测试任务有不同的测试配置、处理算法以及显示存储要求。

需要强调的是,各测试任务是完全独立的,没有先后顺序之分,只有这样才能大大简化并行处理工作。

2.3 运行效率的统计

每个测试任务需要经过以下三种类型的运行时间:

- 1)事件驱动过程中响应延迟时间。即在某个阶段将任务加入队列、到下一个阶段从队列中读出任务所花费的时间。
- 2)各个阶段分配例程调度空闲线程的调度时间。即为从开始调度线程到线程真正运行任务的处理函数所需要的时间。
- 3)任务处理函数执行的时间。

每个任务的总运行时间是多个以上三种类型的叠加和。为了精确计算运行时间,可以采用 Windows 提供的两个系统函数:

```
BOOL WINAPI QueryPerformanceFrequency
(_Out_ LARGE_INTEGER * lpFrequency);
BOOL WINAPI QueryPerformanceCounter
(_Out_ LARGE_INTEGER * lpPerformanceCount);
```

此外还需要记录从第一个任务进入系统,到最后一个任务完成离开系统的总时间,用于统计该框架实际运行效率。

3 模拟结果

为了验证本文所提出的优化模型的正确性和有效性,作者在研究过程中进行了模拟。

首先通过 GenerateTask 线程不断地顺序产生任务,产生任务的时间间隔随机,并记录为采样时间,然后写入分析队列,进入分析阶段由主线程和若干空闲的线程完成分析阶段的处理,主线程根据算法1执行任务的分析函数,分析函数执行完后,将结果写入测试任务结构的结果域,然后调用该阶段出口函数,将任务写入显示存储队列,显示存储阶段执行类似分析阶段的处理过程,不同于其他阶段,存储阶段作为最后一个处理阶段要进行与运行效率相关的数据记录与统计工作。

任务分析和存储阶段执行函数通过 Windows 系统函数 void Delay (double numberOfSeconds) 来进行模拟,假设执行 2 s 或者随机时间,任务生成时间随机在 0 到 2 s 之间,分析与存储阶段的处理时间分别固定为 3 s 和 2 s,其他实验条件由表 2 给出。

表2 实验条件

名称	描述
操作系统	Windows XP
处理器	Intel Pentium Dual E2200
开发平台	Labwindows 9.1
分析阶段线程数	10
存储阶段线程数	5
异步时钟间隔	0.1 s

实验过程中未出现任务在队列中阻塞等待的现象,运行时 CPU 占用率接近 60%。图 8 给出了采用顺序方式和采用本文所述流水线方式对不同数量任务具体完成时间对比的实验结果。

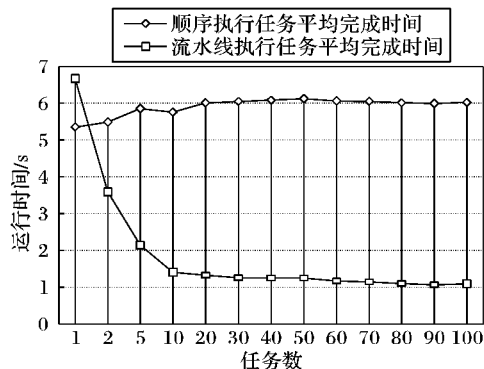


图8 顺序与流水线任务平均完成时间对比

图8中,横坐标是输入系统的任务数,纵坐标为任务平均完成时间。顺序方式下的任务平均完成时间定义为所有任务采集、分析和存储阶段运行时间的叠加总和除以任务数。而流水线方式下的任务平均完成时间定义为从第一个任务输入系统到最后一个任务离开系统的总时间除以任务数。从上述实验得出,异步流水线的方式将效率提高了5到6倍。

这表明本文所述设计模型通过利用多线程的优势能显著提高系统吞吐量,在高速数据采集硬件以及复杂多任务的应用场合下,系统性能将有极大的提高。

4 交流接触器的测试实例

为了进一步验证本文所述方法的正确性,作者将本文所设计的流水线模型应用于交流接触器测试系统。

交流接触器需要测试的参数包括吸合时间、释放时间、跳跃时间、吸合电压、释放电压、行程和超程等^[13]。根据具体要求,本系统的测量参数为吸合电压、释放电压、行程、超程和保持电流这五个指标。

相对于前面所述的理想模拟情况,在具体的交流接触器测试系统实现过程中,除了将上述任务的分析、存储和显示用实际的处理函数代替外,最为关键的是具体测试任务的生成。

首先根据测试时间分成3个阶段、8个周期,如图9所示。

