文章编号：1005-8451 (2011) 12-0009-04

一种新的自动化测试的多任务调度模型

张永超，陶宏才
（西南交通大学 信息科学与技术学院，成都 610031）

摘 要：在分析传统多任务调度模型缺陷的基础上，提出了一种新的自动化测试的多任务调度模型。它不仅可以实现传统多任务调度模型的任务间并发，还能够实现单个任务内不同作业执行线程之间的并发。实践表明，该模型在支持大量并发任务场合具有较好的效果，极大地提高了自动化测试执行效率。

关键词：自动化测试；多任务调度；并发执行
中图分类号：TP311 **文献标识码：**A

Novel multi-task scheduling model of automated testing

ZHANG Yong-chao, TAO Hong-cai
(School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: On the basis of analyzing the defects of traditional multi-task scheduling model, it was proposed a novel multi-task scheduling model of automated testing. The new model could not only implement the task concurrent for traditional model, but also realize the concurrent of different executing threads within a single task. The practice showed that this model had better effect than the traditional model in the occasions of running many tasks, and increased the execution efficiency of the automated testing greatly.

Key words: automated testing; multi-task scheduling model; concurrent execution

在自动化测试执行过程中，目前普遍采用多任务调度模型实现多个测试任务的并行执行。传统的多任务调度模型的核心思想是将任务和线程分离，抽象出实际工作的任务，并将它们映射到少量的线程上，这样可以大大减少系统中运行线程的数目，从而降低线程切换开销在整个系统资源消耗中所占比重[1]。不过，传统多任务调度模型存在以下问题：

（1）由于每个测试脚本运行所需要的测试环境可能不一样，因此脚本测试时不可避免地需要申请和释放测试环境，从而带来处于调度队列不同位置而测试环境相同的测试脚本需要频繁地申请和释放测试环境，导致较大的环境切换开销。

（2）传统模型只实现了任务间的并发，不能完成任务内的并发，从而影响测试效率的提高。

为此，本文提出了一种新的多任务调度模型，旨在降低因执行环境不同而带来的线程切换开销以及任务内测试脚本的并发执行。实践表明，新模型在支持大量并发任务场合具有较好的效果，极大地提高了自动化测试执行效率。

收稿日期：2011-02-16
作者简介：张永超，在读硕士研究生；陶宏才，教授。

1 新的多任务调度模型

1.1 传统模型结构

传统多任务调度模型的结构比较简洁如图1。主要涉及3个对象：任务对象、任务线程对象和任务调度线程对象。任务对象是对用户提交运行任务的一种抽象，它包含要进行测试的测试脚本以及该任务运行所需的测试环境；任务线程对象则对应实际的线程，所有的任务由它负责执行，每个任务线程对象都对应一个任务对象；任务调度线程对象负责管理所有的任务线程对象。在传统的多任务调度模型中，任务调度线程对象会根据待执行任务队列是否为空来判断是否需要进行调度。当待执行任务队列不为空时，任务调度线程对象就循环遍历队列中的每个任务线程对象，根据任务对象描述的测试环境进行测试环境的申请。如果申请成功，就运行该任务线程对象；如果申请不成功，就下移指针开始对下一个任务线程对象申请测试环境。当多个任务对象都申请环境成功，系统中就存在多个任务线程对象在同时运行，这样就简单实现了多个测试任务之间的并发执行。但是在传统的多任务调度模型中，每个提交运行的

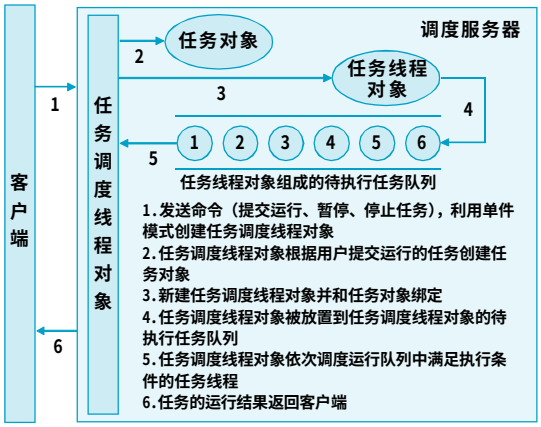

图 1　传统的多任务调度模型结构

任务都被抽象成任务对象，并绑定到一个任务线程对象，每个任务线程对象都只对应一个测试环境，这样就没法实现任务内多个测试脚本之间的并发执行。

1.2　新模型结构

为解决传统模型的结构缺陷，本文提出了一种新的多任务调度模型，如图 2。

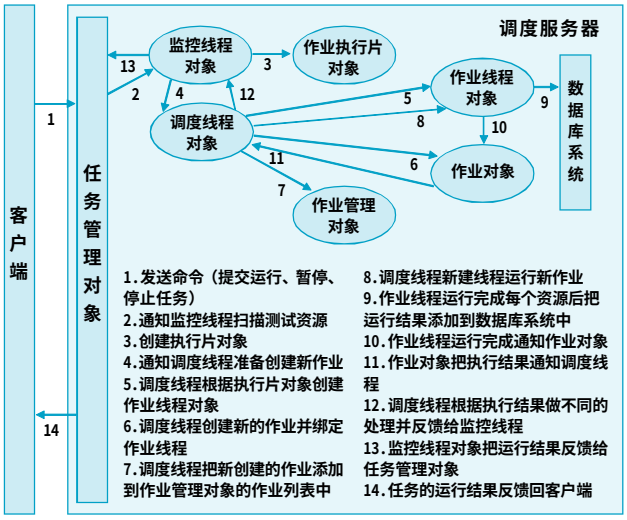

图 2　新的多任务调度模型结构

新的模型在原来结构的基础上，引入了监控线程对象和调度线程对象，并把实际的测试任务进一步抽象，根据测试任务中每个测试脚本所需测试环境的不同进行分组，抽象出作业线程对象，由它来负责运行具有相同测试环境的测试脚本，这样就可以避免在运行测试环境不同的测试脚本时频繁地申请和释放测试环境，并可实现任务内不同测试脚本之间的并发执行。监控线程对象和调度线程对象的处理过程如下：

（1）每个任务都对应一个监控线程对象。当用户提交运行一个测试任务时，监控线程对象就会扫描用户提交的任务所包含的测试脚本，并根据其测试环境进行分组，一个测试环境对应一个或多个测试脚本。然后遍历分过组的测试环境列表，申请对应的测试环境，如果申请成功，就根据该测试环境对应的测试脚本创建作业执行片对象，然后通知调度线程对象准备创建新作业并调度执行。如果没有申请到环境，则移动到下一组测试环境进行环境申请。

（2）每个任务都对应一个调度线程对象。当调度线程对象收到监控线程对象发送的创建新作业通知时，就会根据监控线程对象创建的作业执行片对象创建相应的作业线程对象，并将其与相应的作业对象进行绑定，之后把该作业对象添加到作业管理对象的作业对象列表中。调度线程对象会根据设定的时间间隔定期去查询作业管理对象中是否有待执行作业，如果有就开始调度执行相应的待执行作业线程。

2　新模型的实现

2.1　模型的实现类图

新的多任务调度模型的实现类图如图 3。

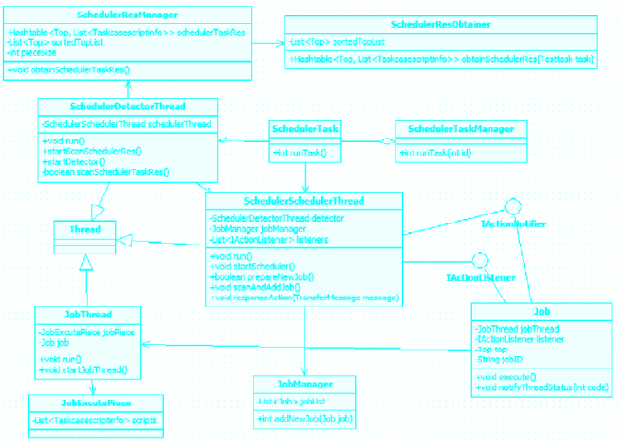

图 3　新模型的实现类图

图 3 中，类 SchedulerDetectorThread、SchedulerSchedulerThread和类JobThread都继承自线程类Thread，分别对应图2中的监控线程、调度线程和作业线程。调度线程在新建作业对象时就把自身作为监听对象和作业对象关联起来。当

作业执行完成或执行异常时，作业对象就可以通过监听对象通知调度线程来做相应的处理。

2.2 任务间和任务内的并行实现

新模型继承了传统模型多个测试任务之间并行执行的优点，只要提交的测试任务对应的测试脚本有可以运行的测试环境，就可以运行任务。同时，新模型支持单个测试任务内多个作业线程的并行执行，如图4，因此能最大程度地利用测试环境，提高测试执行效率，降低频繁申请/释放环境所带来的系统开销。

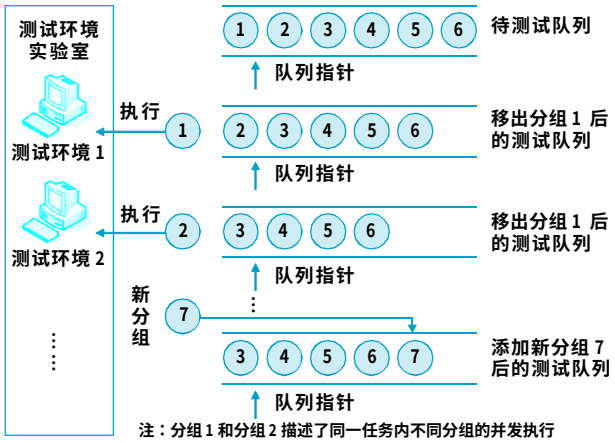

图 4　任务内的并行执行

2.3 作业线程执行结果的处理

新模型通过设置状态码的形式来告知调度线程对作业线程的执行结果进行处理。作业线程执行结果包括：JOB_COMPLETE_CODE（作业线程成功执行完毕）、JOB_ERROR_CODE（作业线程执行出错，网络、测试环境异常或作业线程重新执行次数超过最大值）、JOB_RETRY_CODE（作业线程执行完毕，但其中存在测试失败的用例，需要重新执行失败的用例）。调度线程对象会根据作业线程返回的结果状态做如下处理：

（1）JOB_COMPLETE_CODE：通知监控线程释放相应的测试环境。

（2）JOB_ERROR_CODE：通知监控线程释放测试环境，并把测试环境标识为不可用。

（3）JOB_RETRY_CODE：通知监控线程把刚才执行失败的测试脚本新建一个分组添加到待测试队列中，等待调度执行，如图 4 的分组 7。

当然，实际应用中，可以根据具体情况在模型中添加其它类型的状态码，来标识作业线程的执行结果，并通知调度线程做相应的处理。

3 新模型的应用

下面以一个自动化测试工厂为例来说明该模型的应用。采用本模型的自动化测试工厂的主要工作是自动调度运行测试用户提交的测试任务，根据测试脚本运行环境的不同申请对应的测试环境，把测试脚本发送给相应的测试环境执行。测试脚本执行完成后，工厂自动保存测试脚本的执行结果，并将任务的运行结果通过邮件的方法反馈给测试用户。

假设测试用户通过自动化工厂客户端创建提交了一个包含100个测试脚本的测试任务，每个测试脚本都对应一个测试环境文件，并在测试实验室中存在相应的测试环境。那么自动化工厂引入该模型后对该任务的处理流程如下：

（1）系统通过单件模式生成一个任务管理对象（SchedulerTaskManager），负责统一管理系统中所有的监控线程和调度线程。

（2）任务管理对象根据任务的唯一标识创建一个测试任务对象（SchedulerTask）。同时，该测试任务对象会创建调度线程对象（SchedulerSchedulerThread）和监控线程对象（SchedulerDetectorThread）。

（3）任务管理对象启动监控线程和调度线程，监控线程启动后开始扫描测试任务相关的测试脚本并按照其测试环境进行分组，创建待测试分组队列，扫描队列中的每个分组，为每个分组申请测试环境。如果申请成功，则创建JobExcutePiece类的一个作业执行片对象，并通知调度线程准备创建新作业；如果申请失败，则下移队列指针，对下一个分组申请测试环境。调度线程启动后执行步骤（5）。

（4）调度线程在收到监控线程发送的准备创建新作业消息后，首先创建作业管理对象，该对象用于保存所有运行的作业对象。然后，根据申请到环境的作业执行片创建作业线程对象（JobThread），并绑定到作业对象（Job）。最后，调度线程把作业对象添加到作业管理对象中。

（5）调度线程开始检查作业管理对象中是否有待执行作业。如果没有，则循环等待直到作业管理容器中有待执行作业对象；如果作业管理对象中有待执行作业，则调度线程开始启动作业线程

执行测试。

（6）作业线程执行完或执行过程中出现异常或错误，作业对象通过监听接口通知调度线程。

（7）监控线程在收到调度线程发送的作业执行情况消息后，根据作业的执行情况做出相应的处理。

（8）循环以上步骤，直至所有作业线程都执行完毕。

（9）所有作业执行完毕后，自动化工厂通过邮件方式把任务的执行结果发送给测试人员。

4　结束语

本文提出的自动化测试的多任务调度模型，本质上就是一个面向应用的用户级的线程库，因此继承了用户级线程库低开销的优点。调度模型以减少系统申请释放测试环境额外开销和实现任务内测试脚本并行执行为出发点，对传统的多任务调度模型进行了改进，使其适用于自动化测试环境，是一种比较实用的多任务调度模型。作为一种抽象的软件结构模型，它在需要同时开启大量任务的自动化测试场合具有很强的实用性。

参考文献：

[1] 胡　宁，张德运，史宏锋．一种低开销的多任务调度模型[J]．微电子学与计算机，2005．

[2] 崔启亮，胡一鸣．国际化软件测试[M]．北京：电子工业出版社，2006．

[3] K. Mustafa, R. A. Khan. 软件测试：概念与实践[M]．北京：科学出版社，2009．

[4] 赵　斌．软件测试技术经典教程[M]．北京：科学出版社，2007．

[5] Srinivasan Desikan, Gopalaswamy Ramesh. 软件测试原理与实践[M]．北京：机械工业出版社，2009．

[6] 陈汶滨，朱小梅，任冬梅．软件测试技术基础[M]．北京：清华大学出版社，2008．

责任编辑　方　圆

（上接P8）

配系统自2010年8月11日上线以来，运行平稳，客流预测较为准确，站车反应良好，对系统上线前后京津城际列车快通卡票额、刷卡人数进行了统计，得到表3和图4。

表3　京津城际列车快通卡票额分配前后利用情况分析表（日均）

日期段	定员	有席售出	有席上座率	快通卡票额	快通卡人数	快通卡票额利用率
0711-0731	67 975	48 665	71.6%	7 787	1 713	22.0%
0811-0831	67 975	51 972	76.5%	2 216	1 776	81.2%

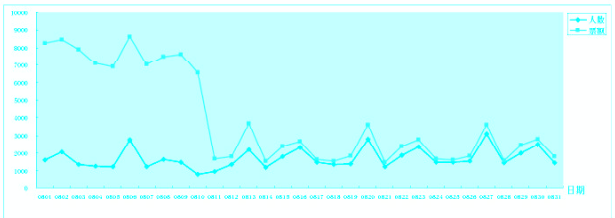

图4　京津城际列车快通卡票额与刷卡人数对比图

由表3和图4分析可得出

（1）系统应用后，8月11日～31日京津城际列车日均有席人数、有席上座率均较7月份有了明显提高，有席上座率提高了近5%。

（2）8月11日～31日京津城际列车快通卡日均刷卡人数较7月份增加63人次，快通卡日均票额由分配前的7787张降低到2216张，快通卡票额利用率较7月份增加了近60%，增幅十分显著。

4　结束语

京津城际列车快通卡票额智能分配系统已投入稳定运行，达到了在基本确保快通卡旅客“快通”的前提下，尽量减少快通卡票额浪费的预期目标，从而提高了列车运营和社会效益。该系统在遇到突发客流时，仍可以通过人工调整手段进行票额调整，但由于调整时间紧，调整数量未知等问题，使得人工调整票额不够科学。下一步需研究实时智能分配票额，实现对偏常态客流的动态监控与动态票额调整。

参考文献：

[1] 李　业．预测学[M]．武汉：华南理工大学，1988．

[2] 郭孜孜．铁路客流预测方法研究[D]．成都：西南交通大学硕士研究生学位论文，2005．

责任编辑　方　圆