Improved Multithreaded Unit Testing

周昊一 MF1233055

为了更好地利用多核处理器,程序员经常会编写一些并行执行的代码。并行执行的代码之间通常通过共享的内存空间来进行通信，因此会受到并发性bug的困扰，这些bug一般都很难检测，因为他们只有在多线程的情况下，才会发生，并且可能只有在非常罕见的执行顺序下才会出现。

为了验证多线程代码的正确性，开发者通常需要编写多线程的单元测试。在这些测试中，需要确保一些不同线程中语句的执行顺序，但在不控制调度过程的情况下是不可能实现的。为了达到控制调度过程的效果，通常采取的方法是使用延时（例如sleep）。但如果在调度过程中出现了意料之外的情况，测试可能就会失败。并且sleep的时间都是通过估计得到的，很不可靠。

多线程的单元测试主要的问题有以下几个方面：1.可读性；2.模块性；3.可靠性；4.迁移代价；

在这篇文章中，作者介绍了一个新的测试框架IMUnit。IMUnit主要有以下几个贡献：1.调度语言，IMUnit引入了一种可以明确指定多线程调度顺序的语言用来编写单元测试；2.自动化迁移，通过一些工具，可以简单的将基于sleep的测试转化为IMUnit，变为基于事件的测试；3.执行测试与和检查的工具。

通过对200个基于sleep的单元测试的实例的评估，发现IMUnit是非常有效的，其中只有4个例子无法去除sleep。

文章先通过一个单元测试的实例比较了JUnit，MultithreadedTC和IMUnit三种不同的方法，接下来主要是介绍了IMUnit的使用方法：

1. IMUnit中对调度的控制是通过一种特殊语言来表示的。

2. 从原有测试技术的代码迁移到IMUnit框架下可以通过6个步骤的手动操作，也可以通过自动化的迁移工具，迁移工具主要是基于一些自动推断技术（事件和调度的推断）。

3.一个基于JavaMOP的IMUnit Runner，用来运行并监测IMUnit测试。

最后文章对IMUnit的性能进行了一个评估，从自动推断的准确率，与测试执行时间等方面进行了测试。