基于BCI的软件运行轨迹捕获技术研究与实现

黄南平，金可音

（湖南工业大学 计算机与通信学院，湖南 株洲 412008）

摘 要：提出并实现了一种基于BCI代理的软件运行轨迹捕获技术。该技术能在不改变源代码，且无需AOP技术所必须的织入器支持的情况下，为目标系统注入监测探针代码，实现监测其运行轨迹的功能，为进一步分析、推断软件运行行为是否异常提供技术支持。

关键词：字节码工具；路径捕获；系统监测

中图分类号：TP311.56 **文献标志码**：A **文章编号**：1673-9833(2011)02-0059-03

Research and Implementation of Trace Capture Technique Based on Byte Code Instrumentation

Huang Nanping, Jin Keyin

（College of Computer & Communication，Hunan University of Technology，Zhuzhou Hunan 412008，China）

Abstract：Proposed and implemented a technology to capture the trace of running software based on BCI agent. Without converting the original code and without the weaver needed in Aspect-Oriented Programming (AOP), it inserted probe code into the target system to implement the function of monitoring the trace capture and provided technical support for further analysis and judgement on the software running behavior.

Keywords：Byte Code Instrumentation（BCI）；trace capture；system monitor

0 引言

软件发展正面临诸如软件运行行为表现出动态性、不可预测性、潜在不安全性等诸多挑战[1]，为此，研究者们需尽可能全面地了解软件系统运行时的内部细节，并据此分析解决所遇到的问题。

捕获软件运行轨迹是系统行为分析、软件故障定位等工作的基础。早期的捕获软件应用中，常采用面向对象的程序设计（object oriented programming，简称OOP）方法实现系统的运行轨迹监测功能，实施过程中往往需要向程序中手动插入用于获取监测信息的代码，即探针代码。随着对监测信息需求的增加，探针代码也更广泛地被分布在系统的诸多角落中[2]。因此，使用OOP技术时，监测需求的实现往往分布在程序的各处，不能自然地适合单个对象或模块[3]。目前的捕获软件应用中，常采用面向方法编程（aspect oriented programming，简称AOP）技术来实现系统的监测功能，通过织入器（weaver）将探针代码织入目标代码中，结果产生一个可监控的目标系统。AOP技术虽能使用松散耦合的、模块化的

收稿日期：2010-12-08
基金项目：湖南省自然科学基金资助项目（05JJ30122），湖南省教育厅科研基金资助项目（04C720）
作者简介：黄南平（1978-），男，湖南株洲人，湖南工业大学硕士生，主要研究方向为可信软件和软件工程，
E-mail：415496179@qq.com

方法，实现监测系统的功能，但由于不同织入器的织入时机和织入方式不同，因而最终目标系统的代码质量受制于第三方的织入器性能。为解决此问题，本文采用字节码工具（byte code instrumentation，简称BCI）技术，即在系统运行时，动态地将探针代码注入目标字节码中，以实现对目标系统的监测功能，且不改变目标系统的文件。

1 BCI介绍

java在JDK1.5引入了java.lang.instrument，可由此实现一个java agent，通过此agent来修改类的字节码（即转换类的定义与实现），且转换在java虚拟机（java virtual machine，简称JVM）定义类之前发生。代理类必须实现公共静态premain方法，其方法头public static void premain(String agentArgs, Instrumentation inst)或public static void premain(String agentArgs)。该方法的原理与main应用程序入口点类似。通过参数inst可以注册多个类文件转换器（java.lang.instrument.ClassFileTransformer）的实例，转换器则将探针代码注入指定的目标字节码文件中，当调用这些类的成员方法时，实际运行的是经过转换后的类的成员方法。在JVM初始化后，每个premain方法将按照指定代理的顺序调用，然后将调用实际的应用程序main方法，这样，main方法就运行在经过转换的类字节码之上，从而达检测目的。

下面将讨论BCI轨迹捕获监测代理的框架结构，分析监测代理各组成部件的作用及它们之间的关系。并在此基础上讨论探针代码的设置，以解决注入什么和在哪里注入的问题，最后讨论如何将捕获到的信息组织成一个时序视图。

2 监测代理的框架结构

监测代理主要由类文件转换器、探针部署器、探针器、运行信息收集器、信息管理器和时序视图组成，其主要框架如图1所示。

监测代理运行在Java虚拟机定义各个类之前，并在主程序调用之前完成对各个类的转换工作，把探针代码注入各个目标代码中。类文件转换器主要完成特定的转换任务，每个转换器仅能实现一种转换功能。为了有针对性地对目标系统进行监测，需要指定感兴趣的目标，探针部署器能实现对非感兴趣的目标进行过滤的功能，只将探针注入感兴趣的目标中。目标系统运行过程中将产生大量运行信息，这些信息由收集器进行收集，并保存在信息管理器中。

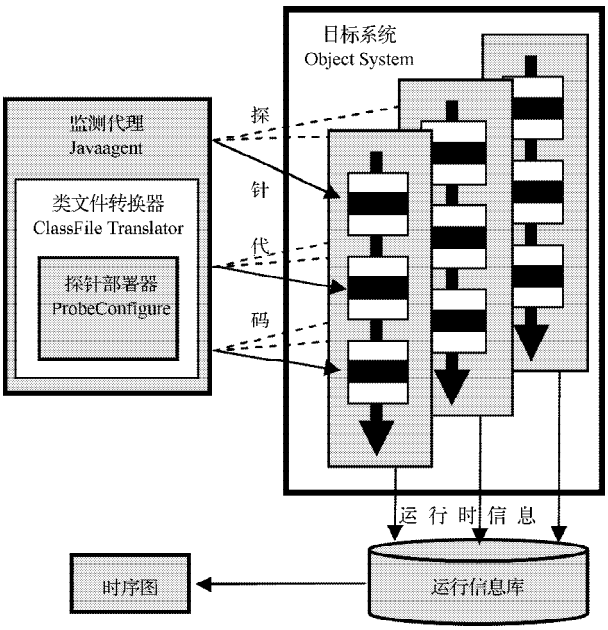

图1 监测代理框架

Fig. 1 Framework of monitor agent

3 监测探针的设置

无论是界面交互类还是应用服务类的软件，且无论是并发运行还是串行运行的软件，其运行过程实质上是通过各种函数调用来实现的，所以软件运行轨迹本质上都是由若干函数构成的[4]。从理论上讲，只要把软件中所有的业务函数定义为观测点，并部署探针代码，就能够描述软件运行轨迹的时序信息。而实际上，线程作为软件运行中资源分配和业务执行的基本单位，能够较为充分地反映软件的运行状态和运行逻辑。线程的运行依托于具体业务逻辑函数的执行。实际操作时以线程为单位观测系统的业务处理轨迹，以了解任务的并发协同情况。函数的执行轨迹直接刻画出线程的迁移，所以对线程的监视在很大程度上就是对线程运行函数的监视[5]。因此，除独立的业务函数外，以线程为单位的业务函数集合也是重要的观测对象。图2为本设计中探针代码的注入方法。

图2所示探针代码注入算法中的伪代码就是在观测点注入探针代码的情况。由Java API无法确定某线程A究竟是在哪个线程中启动的，第1~5行用于捕获父子线程之间的对应关系，第8~10行用于获取当前线程及其父线程的基本信息，第11~13行用于获取业务方法的基本信息，第14,16行分别获取业务方法调用的起始时间和结束时间，第15行表示原业务方法的调用。

```
1) if(isThreadStart(call)) {
2) call.replace("{ 获取父线程名；
3) $_ = $proceed($$);
4) 获取子线程名；
5) 保存父子线程关系；}");
6) }else
7)   { call.replace("{
8)    获取当前线程名；
9)    获取当前线程的状态；
10)   获取当前线程的父线程；
11)   获取函数所在的包名；
12)   获取函数所在的类名；
13)   获取函数签名；
14)   获取函数执行起始时间；
15)   $_ = $proceed($$);// 函数执行
16)   获取函数执行结束时间；}");}
```

图 2　探针代码的注入算法

Fig. 2　Algorithm of inserting probe code

4　时序视图

每次监测获得的信息都是离散、割裂的，而软件的运行是连续、完整的，只有对捕获的信息进行合理有效地组织才能正确呈现软件的运行轨迹。在信息管理中，是以业务方法执行所属的线程为主关键字，以业务方法执行的起止时间为次关键字的顺序，将这些离散、割裂的信息组织成连续、相关的软件的运行轨迹，最后以 XML 文件进行保存。例如，某软件运行时依次启动线程 thread_1 和 thread_3，其运行轨迹如图 3 所示。

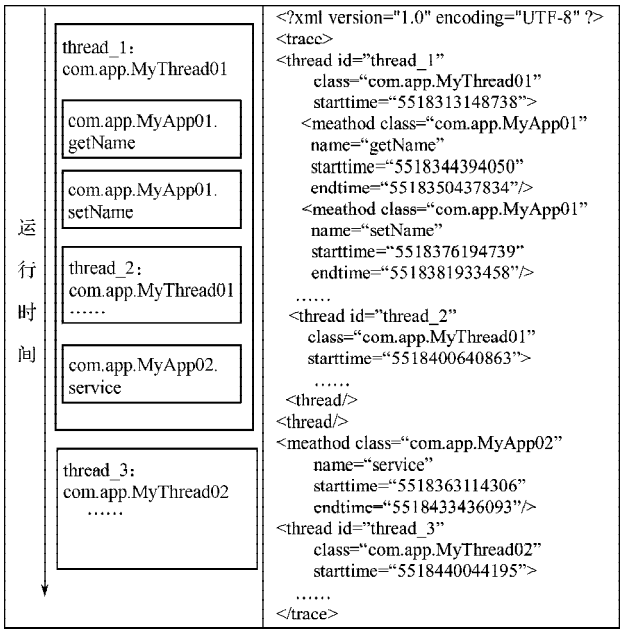

图 3　运行轨迹实例

Fig. 3　Example of running trace

图 3 中，thread_1 依次进入：com.app.MyApp01.getName( )，com.app.MyApp 01.set Name( )，……，thread_2，com.app.MyApp02.service( )，图右侧是该运行轨迹的 xml 存储形式。

5　结语

BCI 是一种非常强有用的字节码工具，它提供了许多底层的 API，为实现软件监测提供了另一种途径。本文通过 BCI 在目标系统的字节码文件转换中注入探针代码，从而达到收集软件运行信息的目的。与 AOP 方法相比，本文提出的方法不会受限于依赖某种织入器，实现了轻松捕获软件运行轨迹的目标，在实践中取得了良好的效果。

参考文献：

[1] 杨芙清，梅　宏，吕　建，等. 浅论软件技术发展[J]. 电子学报，2002，12（增刊）：1901-1906.
Yang Fuqing，Mei Hong，Lv Jian，et al. Some Discussion on the Development of Software Technology[J]. Acta Electronica Sinica，2002，12 (S1)：1901-1906.

[2] Mahrenhol Z D，Spinczy K O，Schrodr-Preikschat W. Program Instrumentation for Debugging and Monitoring with Aspect C++[C]//Proc of the 5th IEEE Int'l Syrup on Object-Oriented Real-Time Distributed Computing. Washington DC：IEEE Computer Society，2000：249-256.

[3] Kicgales G. Aspect-Oriented Programming[EB/OL]. [2010-09-20]. http://en.wikipedia.org/wiki/Aspect-oriented_ programming.

[4] 张瞩喜，王怀民. 基于AOP的软件运行轨迹捕获技术研究与实现[J]. 计算机应用，2008，5(5)：1322-1324.
Zhang Zhuxi，Wang Huaimin. Research and Implementation of Trace Capture Technique Based on Aspect-Oriented Programming[J]. Journal of Computer Applications，2008，5(5)：1322-1324.

[5] 王　亮，周晓聪. AOP在JAVA多线程监控中的应用[D]. 广州：中山大学，2009.
Wang Liang，Zhou Xiaocong. Application of JAVA Monitoring Multithread Based on AOP[D]. Guangzhou：Sun Yat-Sen University，2009.

（责任编辑：廖友媛）