个人资料

搜狐网友**26465131**

博客年龄：8年7个月
访问：2386次
文章：10篇

个人描述
姓名：***
职业：**
年龄：**
位置：中国，**
个性介绍：
****

博主最新文章

- 志愿服务连接你我，携手共建和谐校园
- 祝贺计算机系获得研究生师生长跑综合二等奖！
- 祝贺计研2在本届系运会中取得男子团体总分第一、精神文明奖以及团体总分第三的好成绩！
- 【原创】Pin 2.0 User Guide 翻译之一——概述
- 祝贺计算机系研究生再次蝉联一二九合唱比赛一等奖！

日志正文

【原创】Pin 2.0 User Guide 翻译之一——概述

标签： Intel Pin 插装 使用说明

2007-01-07 03:52 阅读(788) 评论(0)

————————————————————

概述

————————————————————

--- translated by chenlei. **DO NOT COPY WITHOUT PERMISSION !**

Pin

源代码，而是可执行文件。Pin截获可执行文件的执行的第一条
码序列。然后它将控制权交给新生成的这个代码序列。新生成的
以保证在遇到分支指令时夺回控制权。得到控制权之后Pin会为
Pin通过把所有生成的代码保存在内存中，便于代码的重用和直

参考。Pin通过生成新代码给用户提供插入自己代码（插装）的机

概念上，插装包含两个部分：

· 决定哪里或者哪些代码需要插装

· 插装点需要执行的代码

Atom称这两个部分为插装代码和分析代码。实际上Atom把这两部分分离在不同的文件里。在Pin里不再需要分离这两部分，他们可以在同一个可执行文件里，即Pintool。Pintool可以认为是能够修改Pin的代码生成过程的插件。

Pintool在Pin里注册回调线程，这样在需要生成新代码时就能回调。这个线程对应了插装部分。它监视需要被生成的代码，分析静态属性，决定是否以及哪里需要插入调用分析代码，可以调用Pintool里的任意函数。Pin确保寄存器状态会在必要的时候保存和回复，也允许函数参数传递。

观察

由于Pintool像插件一样工作，所以它必须与Pin和需要插装的可执行代码运行在同一地址空间。这样Pintool才能访问所有可执行数据，也能与可执行代码共享文件描述符和其他进程信息。

Pin和Pintool控制程序从第一条指令开始执行。对于与共享库一起编译的可执行代码，执行动态装载和共享库都会对Pintool可见。

"服务主战场，成就大事业"——"启航，到祖国需要的地方去"主题团日活动计算机系活动报道

以史为鉴，面向未来 ——记计研2联合党支部活动 - 观看《东京审判》

师生同乐 共庆日月 ——祝贺高性能所2006年中秋晚会圆满成功

新生知识竞赛 06年9月22日

序言

更多文章>>

在写工具的时候，调试分析代码比调试插装代码更重要。因为插装代码只执行一次，但是分析代码会被执行很多次。

插装粒度

如上所述，Pin的插装是实时的。插装发生在代码序列第一次执行之前。我们称这种操作模式为跟踪插装(trace instrumentation)。

trace插装让Pintool在可执行代码每一次执行时都能进行监视和插装。trace通常开始于选中的分支目标并结束于一个条件分支，包括调用(call)和返回(return)。Pin能够保证trace只在最上层有一个入口，但是可以有很多出口。如果在一个trace中发生分支，Pin从分支目标开始构造一个新的trace。Pin根据基本块(BBL)分割trace。一个基本块是一个有唯一入口和出口的指令序列。基本块中的分支会开始一个新的trace也即一个新的基本块。通常为每个基本块而不是每条指令插入一个分析调用。减少分析调用的次数可以提高插装的效率。trace插装利用了TRACE_AddInstrumentFunction API call。

为了方便编写Pintool，Pin还提供乐指令插装模式（instruction instrumentation），让工具可以监视和插装每一条指令。本质上来说这两种模式是一样的，编写Pintool时不需要在为trace的每条指令反复处理。就像在trace插装模式下一样，特定的基本块和指令可能会被生成很多次。指令插装用到了 INS_AddInstrument Function API call。

Pin的指令插装模式以及trace插装模式与Atom的插装模式完全不同。Atom在每一次执行(pass)前从头开始插装整个可执行代码。Pin把执行和插装交替进行，那些没有被执行的代码则不会被插装。

然而，有时候也需要采用除了trace之外的其他的粒度。为此Pin提供了另外两种模式：镜像(image)和线程(routine)插装，类似Atom采用的前期处理模式。这些模式通过缓存插装请求来实现，因此需要额外的空间。

镜像插装让Pintool在IMG第一次导入的时候对整个image进行监视和插装。Pintool的处理范围可以是镜像中的每个块(section，SEC)，块中的每个线程(routine, RTN)，线程中的每个指令（instruction, INS）。插装可以在一个线程或者一条指令开始执行之前或者结束执行之后执行。镜像插装用到了 IMG_AddInstrumentFunction API call。镜像插装依靠符号信息判断线程的边界，因此需要在PIN_Init之前调用PIN_InitSymbols。

线程插装让Pintool在线程第一次调用之前监视和插装整个线程。Pintool的处理范围可以是线程里的每条指令。这里没有足够的信息把指令归并成基本块。插装可以在一个线程或者一条指令开始执行之前或者结束执行之后执行。当镜像中只有少量线程被执行时，线程插装能在时间上和空间上比镜像插装更高效。线程插装用到了RTN_AddInstrumentFunction API call。插装在线程结束后不一定能可靠地工作，因为当最后出现调用时无法判断何时返回，例如for xscale。

上一篇： 祝贺计研2在本届系运会中取得男子团体总分第一、精神文明奖以及团体总分第三的好成绩！

下一篇：祝贺计算机系研究生再次蝉联一二九合唱比赛一等奖！