链滴

程序动态分析

作者：Hanseltu
原文链接：https://ld246.com/article/1557322130735
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<blockquote>
原文链接 程序动态分析
</blockquote>
<h3 id="什么是动态分析-">什么是动态分析？</h3>

有时我们想知道执行一个程序运行时的一些信息，通常存在的问题有：

我写的程序运行的时候可能会怎么样？
为什么这种情况会发生？
程序中哪部分是耗时最多的？
哪些代码可以并行化处理？
程序中存在错误怎么处理？

<h3 id="如何进行动态分析-">如何进行动态分析？</h3>
通常我们通过记录程序执行时的一些信息已对程序进行分析。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src="https://b3logfile.com/file/2019/05/how1-a1ec50b1.png?imageView2/2/interlace/1/format/jg">
具体的来说，通常有三种方式可以记录程序运行时的信息：

追踪程序变量等信息并记录
使用查插桩技术，额外添加代码辅助记录
以上两种方法结合使用

<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src="https://b3logfile.com/file/2019/05/how2-e749ecc7.png?imageView2/2/interlace/1/format/jp">
<h3 id="简单想法-基础块剖析">简单想法：基础块剖析</h3>
通常我们不用对程序的每个部分进行分析，有点类似之前文章程序切片技术提到的程序切片技术，在剖基本块的时候我们最关心的问题是效率怎么样，如果一个工具需要的开销很大，在实际应用的时候就以被接受。这时我们通常需要考虑如何选择有效的策略对程序进行剖析。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src="https://b3logfile.com/file/2019/05/simple1-92843f45.png?imageView2/2/interlace/1/format/pg">
<h3 id="有效的剖析策略">有效的剖析策略</h3>
剖析的基本策略如下，在使用剖析技术时需要考虑是否满足：

尽可能简单
确认然后避免冗余的信息
使用采样的方式分析
对程序压缩和编码
由剖析结果指导插桩操作
尽量使用局部分析和推理

<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src="https://b3logfile.com/file/2019/05/simple2-100340a8.png?imageView2/2/interlace/1/format/pg">

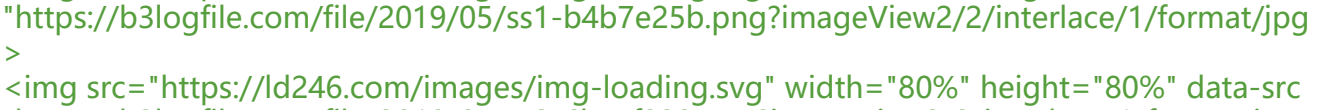
<h3 id="路径剖析">路径剖析</h3>
路径剖析问题关注的是程序运行时执行的路径，对每条可能执行的路径进行编码
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/p1-bead13a3.png?imageView2/2/interlace/1/format/jpg"

采用的步骤如下：
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/step1-6a0548d3.png?imageView2/2/interlace/1/format/j
g">
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/step2-929ff60e.png?imageView2/2/interlace/1/format/jp
">
在对路径进行编码时，通常有两种方法，一般的方法是使用字典来记录，开销比较大，不推荐使
。另一种是使用 CFG 图来进行适配。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/step3-024194d9.png?imageView2/2/interlace/1/format/j
g">
对一些常见的标准测试集进行路径剖析，可得出他们的基本路径信息
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/r1-6a5bfe33.png?imageView2/2/interlace/1/format/jpg">
以上的分析仅包含简单的情况，即没有包含存在循环路径的程序，那么对于存在循环的路径怎么
路径进行编码也是我们需要考虑的问题。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/step4-f759f664.png?imageView2/2/interlace/1/format/jp
">
实际上我们不用考虑的这么全面，对于有循环路径的程序，我们只需要记录什么地方存在循环，
常我们只关注这次运行时通常经过的路径有哪些？或者说是对于这一批输入，程序通常经过的路径有
些？
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/step5-ed3765a9.png?imageView2/2/interlace/1/format/j
g">
但是如果我们不知道程序的输入是什么该如何分析这段程序？
那就需要使用代码建模技术。
<h3 id="近似建模技术">近似建模技术</h3>
当我们不知道程序的输入是什么的时候，我们通常使用程序建模技术，对某个程序的行为进行建

<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/ap1-454a5ce3.png?imageView2/2/interlace/1/format/jpg
">
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/ap2-89a3b7c0.png?imageView2/2/interlace/1/format/jpg
">
<h3 id="如何-什么时候插桩">如何/什么时候插桩</h3>
对程序基本执行路径进行剖析完成后，根据剖析的信息，我们通常需要对程序进行插桩处理进一
分析程序的特性。在不同的阶段或者不同的代码级别上插桩的难度和效率都是不一样的。通常有基于
码或者中间代码的插桩，基于静态二进制的插桩和动态二进制的插桩。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/in1-fc3ae06d.png?imageView2/2/interlace/1/format/jpg"

<h3 id="动态分析步骤">动态分析步骤</h3>
动态分析一般的步骤分为三步：插桩，执行，分析。
<img src="https://ld246.com/images/img-loading.svg" width="80%" height="80%" data-src
"https://b3logfile.com/file/2019/05/pp1-c9bc5e9c.png?imageView2/2/interlace/1/format/jpg
">

其中静态和动态二进制插桩的方式又有所不同。

后面还介绍了 AddressSanitizer 工具的使用，此部分见之前发布的文章 AddressSanitizer VS Valgrind
<h3 id="总结">总结</h3>
总结一下，动态分析关注的是程序运行时的真实行为，通过插桩的方式修改程序的行为并收集相信息以对程序进行充分分析。
如果我们不想深入了解程序的信息，而且不想有程序运行的过程，这样可以分析出来程序的一些性吗？答案当然是可以的，这就需要我们接下来的静态分析技术。
<blockquote>
回到课主目录
</blockquote>