Casper Popl.Calltrace

2016年6月16日 14:07

Call trace 在程序的分析，bug的自动查找和性能分析中是非常重要的一个组件。但是使用传统的Linux性能分析工具，如perf，gprof, systemtap等，为了获得准确的调用信息，需要在程序中插入大量的回调函数，并且为了回调函数之间不冲突，会加入很多的锁，因此会对性能造成极大的损害。另一种测试工具valgrind通过模拟CPU的运行，可以发现很多的内存错误，但是对性能的损害是10倍以上。类似物理学的Uncertainty Principle，在某些高性能的程序中，这种程度的overhead会极大的影响程序的分析。因此，我们自然希望能够降低程序运行时收集信息造成的开销，和BlackBoxesDebug的文章结合来看，在系统软件中分析程序的性能。

传统的已有的程序跟踪和分析软件的技术可以分为两种类型，第一种专注于用较小的开销收集过程中的控制路径信息，可以通过选择性的找一个程序点的子集，或者为不基于调用点的非循环的路径申请一个固定的编码图。然而因为调用追踪本身就是过程中的，需要通过将函数ID在调用点之前注册，而这个开销很大。另一种的方法主要是解决在线程序的分析，因此专注于对最后一个调用的上下文进行编码和维护一个历史调用上下文的summary。这种方法中，call trace 的收集比剖析要更普遍。

为此，Casper的作者提出一种更加有效和紧凑的方法。作者认为固定的调用现场可以通过调用过程来推断，因此，通过选择性的装置一个小的调用点和返回点的子集，可以只在程序运行时收集一部分的信息。然后在离线时，通过推断和复原得到完整的调用信息。这个几乎不丢失信息的恢复的关键是一个组件的集合。这个原则来源于作者的基于LL语法的跟踪模型。作者证明了要最小化组件集合来满足LL语法是一个NP难问题，但是又提出了一个Hn近似的方法。