TEST PDF

Pdf.js 테스트

뉴진스

Dynamic languages such as JavaScript are more difficult to com-pile than statically typed ones. Since no concrete type informationis available, traditional compilers need to emit generic code that canhandle all possible type combinations at runtime. We present an al-ternative compilation technique for dynamically-typed languagesthat identifies frequently executed loop traces at run-time and thengenerates machine code on the fly that is specialized for the ac-tual dynamic types occurring on each path through the loop. Ourmethod provides cheap inter-procedural type specialization, and anelegant and efficient way of incrementally compiling lazily discov-ered alternative paths through nested loops. We have implementeda dynamic compiler for JavaScript based on our technique and wehave measured speedups of 10x and more for certain benchmarkprograms

2. Overview: Example Tracing RunThis

section provides an overview of our system by describinghow TraceMonkey executes an example program. The exampleprogram, shown in Figure 1, computes the first 100 prime numberswith nested loops. The narrative should be read along with Figure 2,which describes the activities TraceMonkey performs and when ittransitions between the loops.TraceMonkey always begins executing a program in the byte-code interpreter. Every loop back edge is a potential trace point.When the interpreter crosses a loop edge, TraceMonkey invokesthe trace monitor, which may decide to record or execute a nativetrace. At the start of execution, there are no compiled traces yet, sothe trace monitor counts the number of times each loop back edge isexecuted until a loop becomes hot, currently after 2 crossings. Notethat the way our loops are compiled, the loop edge is crossed beforeentering the loop, so the second crossing occurs immediately afterthe first iteration


SUP