Trace-Driven, Just-In-Time Compilation with a New Application of Static Single Assignment Form

Tech ID: 18785 / UC Case 2006-460-0

BACKGROUND

A decade after Java arrived, there have been improvements in the runtime performance of platform-independent virtual-machine based software. However, using such machine-independent software on resource-constrained devices such as mobile phones and PDAs remains a challenge, as both interpretation and just-in-time compilation of the intermediate VM language run into technological limitations. Running VM based code strictly in interpreted mode has severe performance overheads, and as a result requires the device's processor to run at a higher clock speed than if native code were run instead. This leads to an increased power consumption, reduced battery autonomy, and may require the overall use of more expensive processors vs. a pure native-code solution.

Just-in-time compilation produces more efficient native code, but the process of getting to that native code may be very costly for our current resource-constrained embedded devices.

Consequently, distinct embedded just-in-time compilers have emerged, in which trade-offs are made between resource consumption of the just-in-time compiler and the ultimate execution performance of the code being run on top of the VM. Embedded just-in-time compilers achieve their results using significantly fewer resources than their larger counterparts by using simpler algorithms. One example is the use of linear-scan register allocation instead of a graph-coloring approach, which not only reduces the run-time of the algorithm, but also greatly diminishes the memory footprint. Embedded just-in-time compilers also tend to use less ambitious data structures than "unconstrained" compilers-for example, while the use of Static Single Assignment (SSA) form is fairly standard in large just-in-time compilers running on server-class machines, the time and memory needed to convert the 10% most frequently executed methods to SSA using traditional techniques exceeds the resources of most embedded computers.

TECHNOLOGY DESCRIPTION

University researchers have developed a just-in-time compiler that pursues a new dynamic compilation approach. The compiler is an add-on to the JamVM virtual machine for embedded devices. Unlike other just-in-time compilers that are "intertwined" with the virtual machine hosting them, ours requires changing no more than 20 lines of JamVM's source code. The first prototype of the compiler was designed as add-on for Sun's KVM virtual machine. Porting the compiler to JamVM only required minimal changes to both the University's new JIT compiler as well as the JamVM source base.

The new JIT runs in a total footprint of 150kB (including code and data) while for regular code still achieving speedups similar to those of heavyweight JIT compilers. Key to the success of the University approach is trace-based compilation using SSA. Similar to other systems before, the new "HotpathVM" JIT dynamically identifies execution traces that are executed frequently-we build dynamic traces from bytecode (which would have been interpreted anyway) rather than from native code, so that the relative overhead of trace recording is much less critical. The real novelty of the University system only comes to bear after a hot trace has been identified: it is then dynamically compiled into native code via a nontraditional application of SSA form, which we call Trace SSA (TSSA).

In the classical use of SSA, a control flow graph is translated into SSA in its entirety and ! nodes are placed in control flow join nodes. In this new approach, we differentiate between the values in a trace being compiled, which are in SSA, and values in the rest of the VM, which are not. The VM explicitly moves data from the stack and local variables into dedicated SSA variables before any generated native code is called, and

CONTACT

Doug Crawford
doug.crawford@uci.edu
tel: 949-824-2405.

OTHER INFORMATION

CATEGORIZED AS

» **Computer**
» Software

RELATED CASES

2006-460-0

explicitly moves non-dead SSA results back onto the stack and local variables on every exit from such an optimized trace (including side exits). This approach enables the just-in-time compiler to perform aggressive optimizations on the trace, including moving operations on SSA values across side exit points. Because instruction traces are essentially linear (they may contain only internal back edges) liveness analysis and placement of ! nodes are straightforward. This new system also supports fairly sophisticated merging of multiple traces that have a common ancestor.

APPLICATIONS

This invention can be used for efficient execution of bytecode such as Java or Microsoft.NET, especially on embedded systems with restricted resources.

PATENT STATUS

Country	Type	Number	Dated	Case
United States Of America	Issued Patent	8,769,511	07/01/2014	2006-460

UCI Beall
Applied Innovation

5270 California Avenue / Irvine,CA
92697-7700 / Tel: 949.824.2683