Improving Trace-Based JIT Optimisation Using Whole-Program Information

Vandercammen, Maarten; De Roover, Coen

Publication date: 2016

Citation for published version (APA):
Improving Trace-Based JIT Optimisation using Whole-Program Information

Maarten Vandercammen
Software Languages Lab, Vrije Universiteit Brussel, Belgium

Coen De Roover

Trace-based JIT Compilation

Program

function f(a) {
  var b = a > 0 ? 1 : 2;
  return a + b;
}

Execution Trace

loop:
  LOAD n
  CMP_GREATER
  LOAD_CONST 1
  LOAD_CONST 0
  CMP_GREATER
  LOAD a
  LOAD_CONST 1
  ADD_INT
  LOAD result
  ADD_INT
  LOAD n
  DEC
  JUMP loop

AOT Compilation

C++

Imprecise

static information

Run-time analysis

My Approach

Hybrid:

Refine static information with observed information

Time

Run-time + Compile-time analysis

Scope

Local + whole-program analysis

Constant Variables Found

Initial + Run-Time analysis 1 & 2

Initial Analysis

No Analysis

Initial Analysis

Refine initial analysis with run-time information

Refine static information with observed information

Whole-program optimisation

Extended trace optimisations

Future Work

1) Analysis Launch Point?
2) Extent of Scope?

Example

(let (a 1)
(b (user-input));
(c (user-input));
(declare (loop1 n))
(if (< n 0) ; RT Analysis 1
  done
(loop1 (- n a b c))))
(loop1 1000)
(if (< b 0) (set-b!))
(set! c (user-input))
(declare (loop2 n))
(if (< n 0) ; RT Analysis 2
  done
(loop2 (- n a b c))))
(loop2 1000)
(if (< c 0) (set-c!!))