Toba: Java for Applications

A Way-Ahead-of-Time Compiler

Todd Proebsting, Gregg Townsend,
Patrick Bridges, John Hartman,
Tim Newsham, Scott Watterson

The University of Arizona

Overview

The Java language is gaining prominence
Java is designed for interpretation
Toba precompiles Java programs
to cut execution time by 2/3
Toba trades flexibility for speed

Toba: An Independent Project

Compatible with Java language spec
Different implementation internals
Different C interface

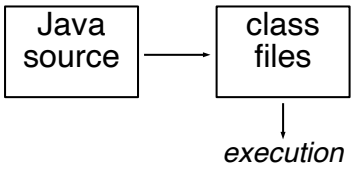
Full Language Support

Java specifies (among other things):
- array bounds checking
- thread support
- garbage collection
- exception handling

These feature affect performance
- some compilers omit them

Original Java Model

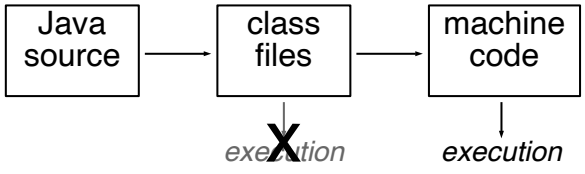
Compile for abstract Java Virtual Machine



Interpretation
Just-in-time (JIT) compilation

Toba: WAT Compilation

Compile JVM code to machine code



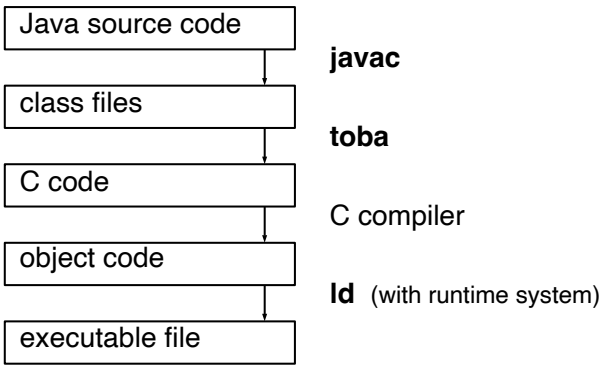
Quick startup – no JIT compilation
Fast execution – better optimization

Applicability of Toba

Not for applets
- not architecture-independent
- machine code can't be trusted

For stand-alone applications
- Java compiler
- "server side" programs
- many other possibilities

Using Toba



The Translator

A 5000-line Java program
Reads class files
Writes (mostly) ANSI C code
- data structure definitions
- static initialization
- executable method code

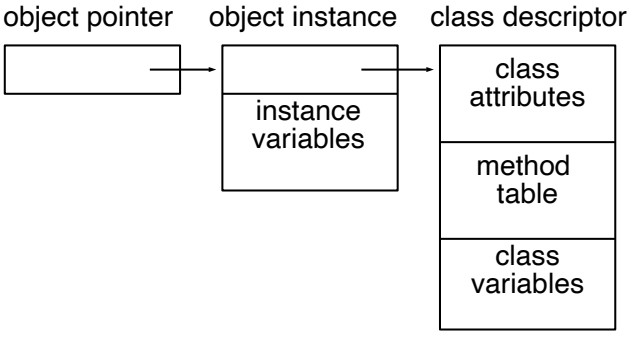
Uses **long long** for 64-bit ints

Runtime Datatypes

Primitive Java types map to C types
- e.g.: **char** –> **unsigned short**

Reference types map to **void** *
- the pointer addresses an allocated object
- Toba does not use handles

Java Objects



Runtime Class Descriptor

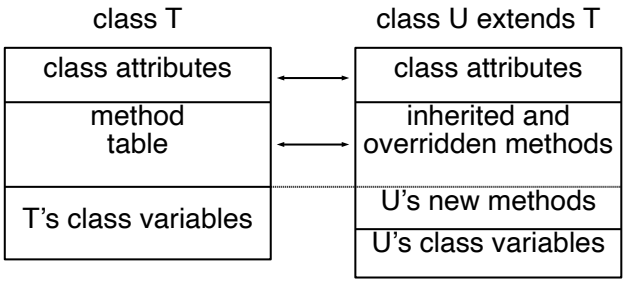
Common attributes
- name, flags, superclass, etc.

Method table
- function pointers and hash values

Class (static) variables

Subclass Descriptor

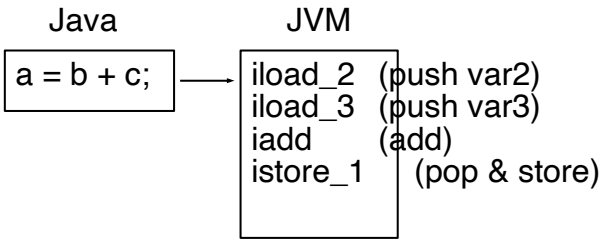
Duplicates and extends superclass layout



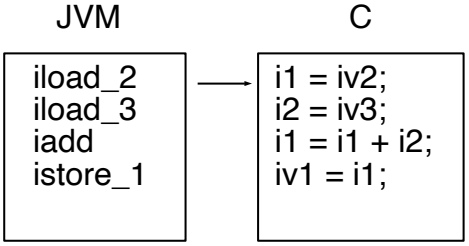
Code Generation

One **.c** / **.h** pair for each class
One function per method
Each method is translated independently
- a key difference from Harissa

JVM is Stack-Based



C Code Models Stack



i1, **i2** are first two integer stack positions
iv1, **iv2**, **iv3** are Java local variables

Naive But Effective

The generated code is simpleminded
C compilers produce efficient code

Control Flow

Use C **goto** statements when possible
- conditional and unconditional jumps

Use **switch** when target is not fixed
- JVM **ret** instruction is an indirect jump

Three Kinds of Method Calls

- Static method
  - direct call to C function
- Instance method
  - indirect call through method table
- Interface method
  - search for matching name and signature
  - use hashcode for quicker searching

Java Exceptions

- Java programs can throw and catch exceptions across method boundaries
- Exceptions are thrown by
  - explicit code
  - execution errors
- Toba uses **setjmp** / **longjmp**

Exception Setup

- A method that catches exceptions
  - calls **setjmp** on entry
  - maintains a PC variable
- No overhead if no **catch**

Exception Dispatching

- **throw** executes a **longjmp** to the innermost **setjmp** point
- Dispatching is based on exception class and PC
- Unselected exceptions are rethrown

The Runtime System

- Java library (API) from Sun
- Toba runtime system (C code)
  - 3000 lines of API support
  - 3000 lines of language support
- Boehm-Demers-Weiser conservative garbage collector
- BISS AWT (window toolkit)
- Thread package

Garbage Collection

- Use Boehm-Demers-Weiser collector
- A free conservative collector
- Originally needed minor changes for Java finalization
- Now used off-the-shelf

Threads and Synchronization

- Thread layer interfaces to system
- Solaris threads supported now
- Others in works
- Thread support impacts performance even when not used

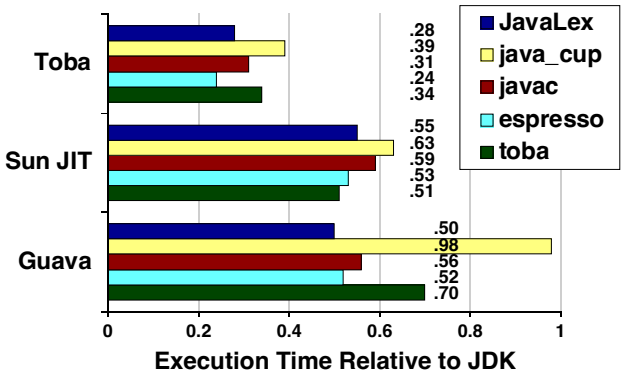
Performance Measurements

- Benchmarks use software tools found on the net
- Toba is compared with three other full implementations:
  - **JDK**: the original Sun interpreter
  - **Sun JIT**: a Sun just-in-time compiler
  - **Guava**: an independent JIT compiler

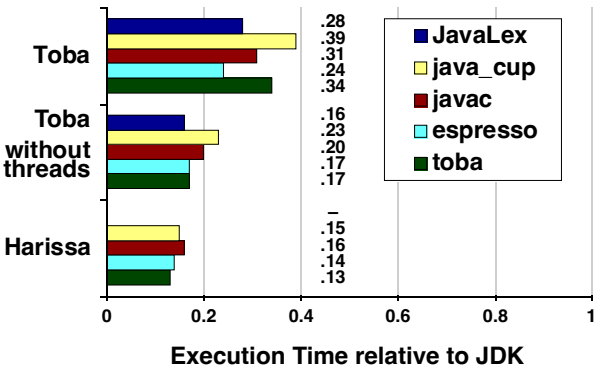
Micro-Benchmarks

- UCSD microbenchmarks isolate specific language features
- Toba is generally faster than others
  - Guava interface calls 16% faster
  - A few others <10% faster
  - Toba wins most comparisons

Application Performance



The Effect of Thread Support



Toba and Harissa

- Toba has thread support
- Harissa does class hierarchy analysis, inlining, and optimization
- Different exception handling approach
- Different null-pointer checking

Current Status

31

Full implementation for Solaris
SGI has ported to Irix with pthreads
Partial implementation for Linux, NT
(no threads, no AWT)
Source code is on the Web

Future Work

32

JIT compiler
JDK version 1.1
More full ports, using POSIX threads

Software Distribution

33

Source code and documentation:

http: // www.cs.arizona.edu /
sumatra / toba /

* Toba is a large lake on Sumatra,
the island just west of Java.