MicroJIT: A Lightweight Just-in-Time Compiler to Improve Startup Times

Outline

Objective: minimize the startup time

The startup phase of an application represents a tiny fraction of the total runtime, but it is considered nevertheless, a critical phase for both client and server environments.

We are investigating whether using two different Just-in-Time (JIT) compilers in the same JVM can improve startup time. In our approach, the lightweight JIT system (MicroJIT) performs an initial, low-optimized, but fast compilation while, at a later time, the standard JIT recompiles Java bytecodes with better, but more expensive, optimizations.

Motivation

- The time spent in the startup phase must be minimized to allow the application to reach the throughput phase as quickly as possible.
- JIT produces highly optimized native code but with a cost in time and computation.
- Investigate if using a lightweight JIT system can improve startup time.

Other techniques

- Ahead-of-Time compilation (AOT) uses code compiled before the start of the JVM, usually saved from a previous execution.
- Offline profiling saves profiling information between JVM runs. The information is used by JIT to optimize the compilation of Java methods.
- Multilevel JIT systems allow for compilation tuning for better (slower) or worse (faster) optimization.

Design

We are porting MicroJIT, a lightweight JIT system, originally developed for JAVA ME, to IBM J9 (Java 8).

MicroJIT differs from the standard JIT:

- Single pass synchronous compilation (no intermediate language).
- Small footprint.
- Template-based code generation.
- Works on the same stack structure (Java stack) used by the Interpreter (i.e., no stack transitions).
- Only a small subset of optimizations.
- Uses the same interface with the VM as the standard JIT.

JIT/VM interface

The JIT system must cooperate with the VM to perform operations that change the state of the JVM: object allocation, read/write barriers, synchronization, method invocation, Java stack "management".

MicroJIT compiles the method depending on different criteria: Counting: if the method is executed n times (default: hundreds). Profilling: if profiling information is used.

VM

MicroJIT

Sampling

JIT

MicroJIT

Compilation criteria

MicroJIT compilation is triggered by the VM when a method is interpreted n times (default: hundreds).

Standard JIT recompiles the method depending on different criteria:

- Counting: if the method is executed m times (default: thousands).
- Profiling: if profiling information is used.
- Only a small subset of optimizations.

Compilation criteria

VM

MicroJIT

Sampling

JIT

Federico Sogaro, Kenneth B. Kent, Eric Aubanel, Peter Shipton
University of New Brunswick, IBM Canada
Faculty of Computer Science
{fsogaro|ken|aubanel}@unb.ca, Peter_Shipton@ca.ibm.com

Federico Sogaro, Kenneth B. Kent, Eric Aubanel, Peter Shipton
University of New Brunswick, IBM Canada
Faculty of Computer Science
{fsogaro|ken|aubanel}@unb.ca, Peter_Shipton@ca.ibm.com