Generic Ahead-of-Time Compilation for Eclipse OMR

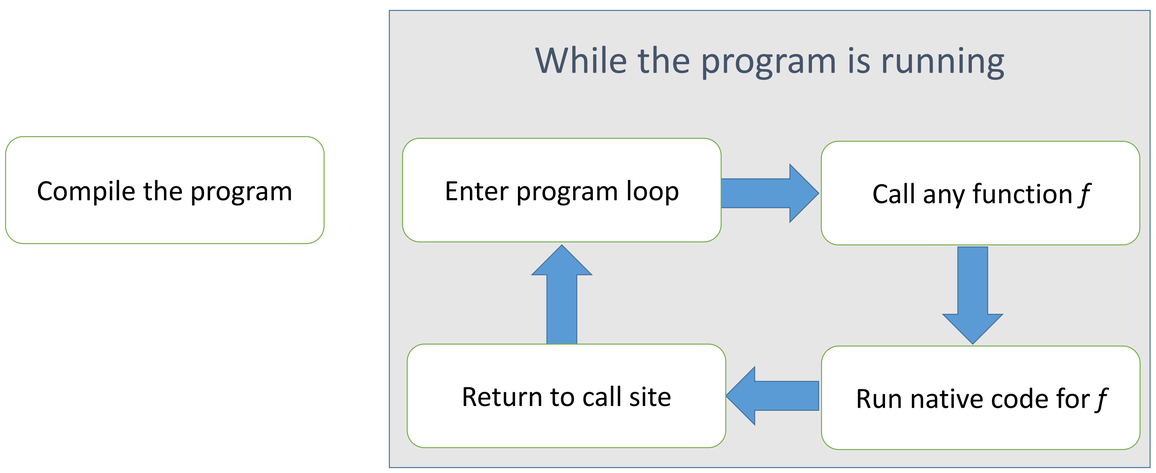
Gerhard Dueck, Kenneth B. Kent, Mark Thom
University of New Brunswick, Faculty of Computer Science
Daryl Maier, Mark Stoodley
IBM Canada
{gdueck, ken, mark.thom}@unb.ca, {maier, mstoodle}@ibm.ca

The Eclipse OMR Project (http://eclipse.org/omr)

- Spawned from IBM's OpenJ9 Java Virtual Machine
- Purpose: provide robust generic components for use in future compilers and language runtimes
- Contains many components at varying stages of completion:
  - Runtime diagnostic tools
  - Garbage collectors
  - Just-in-time compiler
  - Ahead-of-time compiler ???

Ahead-of-time Compilation

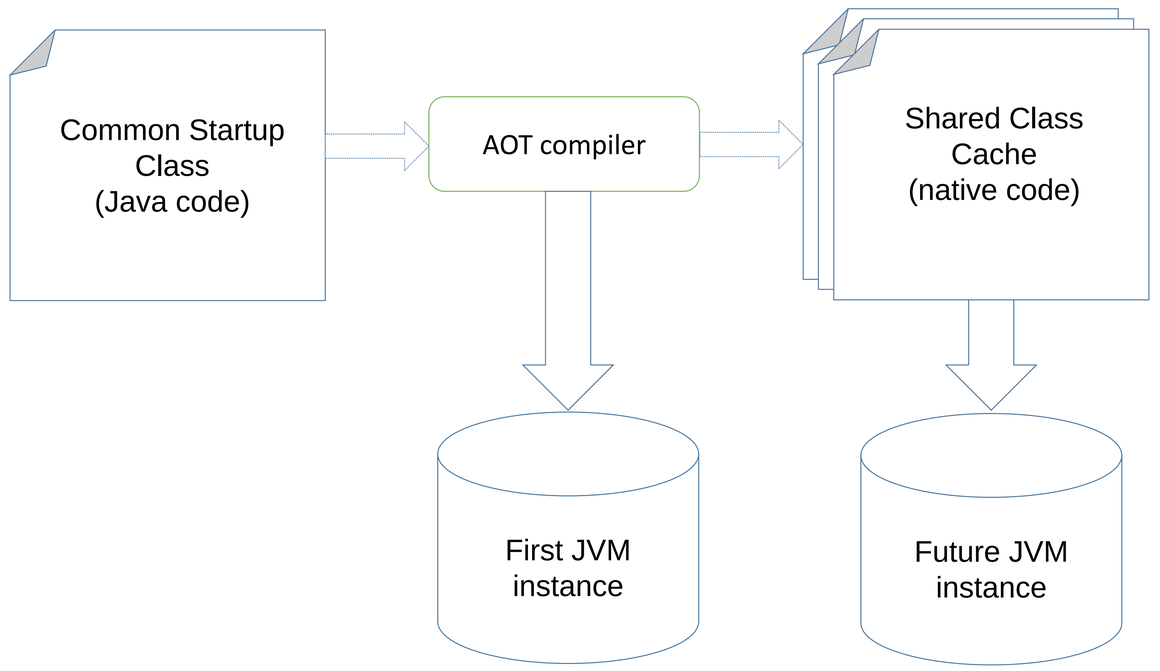
When compiled **ahead-of-time**, native code is generated once, before the program is ever run.



Ahead-of-time compilation (AOT) provides several advantages over other modes of compilation:

- No interpretive overhead at runtime, making it especially attractive for embedded systems and low-latency applications
- Can help to alleviate lengthy startup times in virtual machines
- The cost of performing extensive optimizations on program code is borne only once, at compile-time

The Current Role of AOT Compilation in OpenJ9



In OpenJ9, classes used to bootstrap the Java virtual machine are compiled ahead-of-time when OpenJ9 first starts. The compiled code for these classes is stored in the Shared Class Cache. Future JVM instances are able to pull from the cache when they bootstrap, sparing them the need to compile the same family of classes all over again.

Some loading overhead is required, but it typically takes 1/100th the time needed to compile the classes anew.

Motivation: Extending to a Language Agnostic AOT

OpenJ9's AOT compiler caters specifically to the needs of a Java virtual machine. We aim to generalize beyond this to a language agnostic solution.