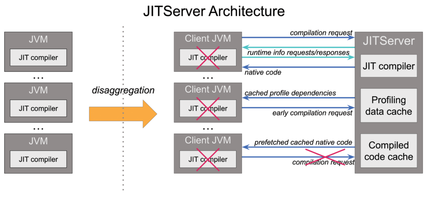
ARIA
APPLIED RESEARCH IN ACTION

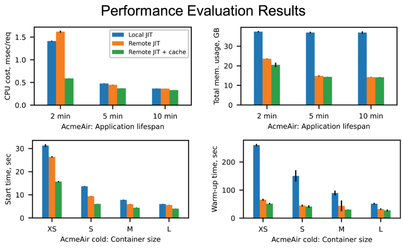
JITServer: Disaggregated JIT Compiler for the JVM

Improving JVM performance and reducing CPU and memory costs in the cloud using remote JIT compilation and caching compiled code and profiling data.

Alexey Khrabrov

Eyal de Lara
ACADEMIC SUPERVISOR





PROJECT SUMMARY

The Java virtual machine (JVM) powers many popular programming languages (e.g., Java, Scala, Kotlin) commonly used in modern cloud computing workloads such as big data analytics, stream processing, and web applications. Managed runtimes such as the JVM use just-in-time (JIT) compilation to improve application performance by converting platform-independent bytecodes into optimized native code dynamically at runtime. Unfortunately, JIT compilation incurs significant CPU (up to 50%) and memory (up tp 100s of MBs) overheads, especially for modern cloud workloads where JVM instances are often short-lived and/or memory-constrained, e.g. FaaS and microservices.

We reduce these overheads by decoupling JIT compilation from the rest of the JVM and offloading it to a separate remote process. This disaggregation reduces the overall memory usage and allows provisioning and scaling compilation resources independently from resources used to run application code. Furthermore, we use the compiler server as a natural point of sharing and reuse of compiler effort across application instances. By caching compiled native code and reusing it in multiple client JVMs, we reduce system-wide resource usage, increase application density, and speed up application start and warm-up. We also reuse profiling data across client JVMs to compile methods earlier, reducing the overhead of slow interpretation.

JITServer is a disaggregated caching JIT compiler that we implemented in the Eclipse OpenJ9 JVM. JITServer is transparent to the application developer and supports all the dynamic features in the JVM specification. In our experiments, JITServer reduced overall CPU cost by up to 77%, overall memory usage by up to 62%, application start time by up to 58% and warm-up time by up to 87%.

SYSTEMS AND NETWORKS LAB

Computer Science
UNIVERSITY OF TORONTO