Electronic Design

WILLIAM WONG | Technology Editor

Zing Exploits LLVM for Java JIT Compiler

Azul's Falcon compiler targets high-performance Java applications using LLVM underpinnings and continuous garbage collection.

Azul supplies a number of different Java runtimes, including Zing and Zulu (there is an embedded version of the latter). Azul's latest offering is an updated version of Zing, that features Falcon, a new JIT compiler, that is based on LLVM. Prior versions of Zing were based on the C2 JIT compiler found in both Zing and Oracle HotSpot.

LLVM is an open-source compiler backend project that is used by a number of compilers, including LLVM's own C/C++ compiler, Clang. LLVM boasts a number of advantages, including contributors associated with hardware platforms, which allow LLVM to support the latest features found on this hardware (e.g., Intel's vectorization support). Likewise, LLVM enjoys wide hardware support for the same reason.

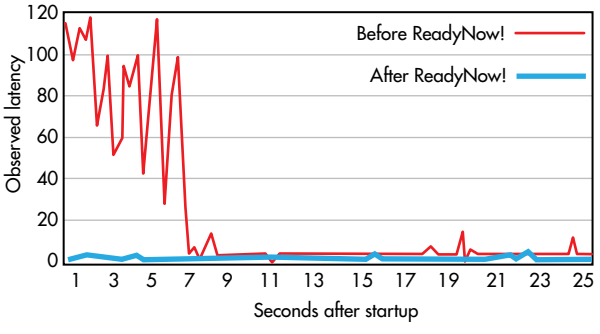
LLVM has traditionally been used with conventional compilers that have a source code front end. Just-in-time (JIT) compilers tend to have different requirements. Getting LLVM to handle these needs was part of the contributions that Azul delivered to the LLVM project. Falcon takes advantage of these additions to implement this Java JIT compiler.

Part of the challenge with JIT compilers it they must understand the underpinnings of the target platform, which has features like continuous garbage collection and handles incremental compilation. More conventional compilers address the entire program or module. A few years ago, LLVM would not be a suitable candidate for a JIT compiler. It is now, and Azul is taking advantage of it.

The use of LLVM is significant for a number of reasons. First, it provides a performance boost where much of that advantage comes from LLVM's efficiencies. Contributors associated with hardware platforms typically provide optimizations for LLVM, and developers using it gain those advantages. This allows Falcon to perform better that the conventional C2 JIT compiler that had been used in prior versions of Zing. Second, it offloads back-end development, allowing Azul designers to concentrate on enhancements to LLVM as well as improving Falcon's front end.

Two things that developers are often concerned about with Java deployment is the garbage collector and the start-up speed of the JIT compiler. Zing is using Azul's C4 concurrent garbage collector. An application does not stop to handle garbage collection. Instead, the garbage collector operates as one or more concurrent tasks. The compacting garbage collector operation is transparent and does not usually require any configuration, although some low-latency applications may increase the number of garbage collector threads.

JIT compilers typically have some start-up performance issues because an optimizing compiler needs to track the execution to see what to optimize. Initially the cache is empty, so optimization does not improve until at least a few thousand instructions have been executed. This can cause a little delay when an application starts.



Azul's ReadyNow! technology improves the startup process by using a stored copy of the JIT cache from a prior run.

Falcon includes Azul's ReadyNow! Technology, which starts with a cache filled with information from a prior run of an application that is stored with the application. This results in a significant speed-up when an application starts *(see figure)*. An application only needs to be run once to gain this benefit, and the information can be used by any number of instances of the application.

The Zing platform targets enterprise Java installations, but

it is equally applicable to x86 platforms with a few gigabytes of RAM. This would include a wide range of embedded applications, including IoT gateways, although pricing tends to be oriented toward large servers.

Azul Zulu and Zulu Embedded may represent a better alternative for embedded applications. Zulu Embedded target smaller platforms. Zulu Embedded is Azul's version of OpenJDK. Zulu does have a low pause time garbage collector. It also lacks the ReadyNow! support.