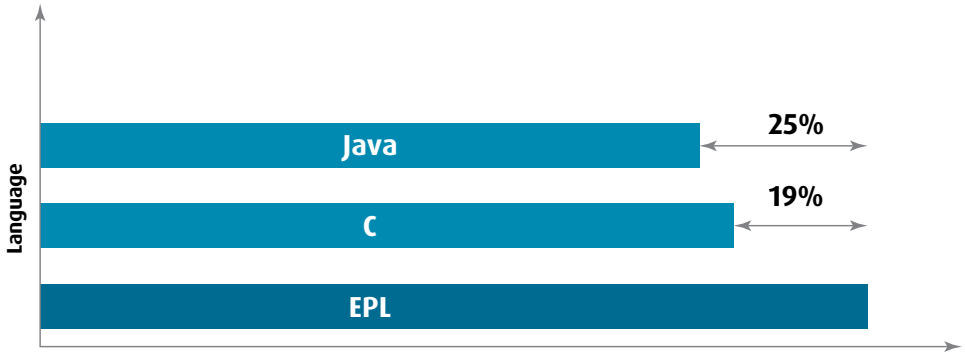
Apama EPL: faster than C?

How can Apama's EPL run faster than both C and Java®?

When a developer compiles a C/C++ or Java application for general distribution, it is typically done using a recent version of the Visual Studio® or gcc compilers for a specific operating system for C/C++ or the Oracle® JDK for Java. Recent microarchitectures for Intel® CPUs include Nehalem or, more recently, Sandy Bridge and Ivy Bridge, each one having a further number of different models. It is these microarchitectures that determine the operating speed of a CPU core. An application is compiled for everyone to use across multiple CPU architectures, which means the resulting application is unable to take full advantage of instruction set optimizations specific to any one CPU.

The Apama Correlator Complex Event Processing (CEP) engine compiles its Event Processing Language (EPL) on every application start-up (as each piece of code is "injected") using a publically available compiler engine known as Low Level Virtual Machine (LLVM) - http://llvm.org. EPL has been encoded into the LLVM engine such that on every start-up, the Apama application is compiled down to native machine code using the instruction set optimizations specific to the local CPU. In this way, the application is always able to take full advantage of the optimizations available on the local CPU no matter where the application is executed. The Apama application has been shown to execute calculations and general imperative code faster than the equivalent implementations in Java and C/C++ which have been compiled for general use across any CPU architecture.

Even the recent C compiler available for LLVM known as Clang will have the same limitation inherent to compiling a C application: that you must compile it for a large audience before distribution. Of course it is possible to compile your C/C++ application to a specific CPU architecture for highly optimized performance on a known platform, but then the local standard libraries that your application uses for complex math calculations (typical of many high speed finance applications) and other standard functions will still likely be compiled for general distribution. A developer would need to take the further step of also compiling special versions of these standard libraries for the target CPU architecture. This is certainly possible, though extremely tedious.



Calculations/second (higher is better)
Based on Black Scholes financial markets modeling algorithm

With Apama, it just happens

In addition to Apama making it easy to write high speed code optimized for the local CPU architecture, you also have the benefit of the development toolkit distributed with the CEP engine, as well as our engineering-maintained connectivity to external sources, such as capital markets trading venues and industry-standard message buses or databases.

So what happens if LLVM doesn't recognize the machine's architecture?

LLVM will still optimize for the next common architecture for the family of CPU used by the local system. LLVM supports a wide array of commonly used architectures, and is continually updated to support new releases. Apama will update its own use of LLVM on a regular basis to reduce the likelihood that a new CPU architecture is not recognized.

If this is so great, why isn't everyone using it?

They are. LLVM is used by (and has contributors from) Apple®, Intel, Google® and many more organizations. See: http://llvm.org/users.html

For more details on Apama, please visit http://www.softwareag.com/apama

Find out how to power up your Digital Enterprise at **www.SoftwareAG.com**

ABOUT SOFTWARE AG

Software AG helps organizations achieve their business objectives faster. The company's big data, integration and business process technologies enable customers to drive operational efficiency, modernize their systems and optimize processes for smarter decisions and better service. Building on over 40 years of customer-centric innovation, the company is ranked as a "leader" in 14 market categories, fueled by core product families Adabas-Natural, Alfabet, Apama, ARIS, Terracotta and webMethods. Learn more at www.SoftwareAG.com.

SAG_Apama_EPL_FS_Jun14

software AG

Get There Faster