# Java Performance

*Aurora VLSI, Inc.*
*March 5, 2009*

This white paper discusses the measured Java performance of Aurora's DeCaf-M Java bilingual processor, and compares it to measured performance of:

- Java on cell phones and smart phones
- Java running under various JVM implementations running on Linux boxes
- The same application re-coded in C

The benchmark used was the CaffeineMarks benchmark. All performance numbers are from running the benchmark on real hardware that was idle except for the running benchmark (and system overhead).

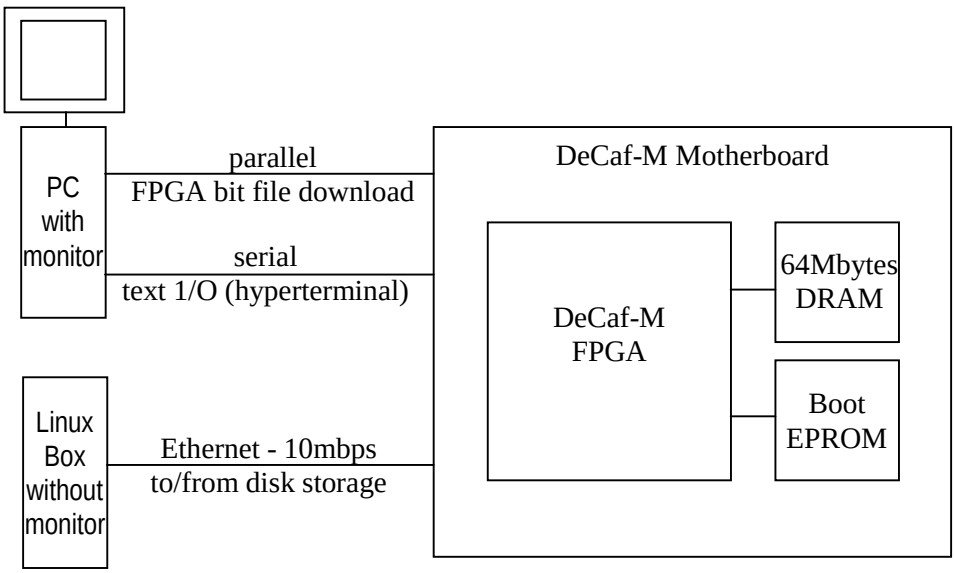
Aurora's DeCaf-M Java bilingual processor Java performance is three to thirty times faster than the Java performance of state-of-the-art cell/smart phones. It is also about 10% faster than the fastest known Java- the J2EE HotSpot JIT JVM, that can only be run on servers and other enterprise level computers with large amounts of memory and high performance CPUs. Additionally, it is about 10% faster than a C version of the same application.

## Aurora FPGA System

A DeCaf-M high performance Java bilingual processor FPGA system was built to demonstrate Aurora's hardware Java acceleration technology. This system provided a way to:

- Finish completely debugging the hardware design
- Develop system software for Aurora's Java bilingual processor
- Port a Java Virtual Machine (JVM) to Aurora's Java bilingual processor
- Measure the Java performance of Aurora's DeCaf-M Java bilingual processor

A block diagram of the FPGA system is shown below.




DeCaf-M Java Bilingual Processor FPGA System

The DeCaf-M Java bilingual processor implemented in an FPGA, is the main processor of a simple diskless and headless computer that was built. In addition to the DeCaf-M processor, this computer's motherboard has:

- 64Mbytes of DRAM
- Boot PROM
- Parallel port
- Serial port
- 10 mbps Ethernet interface

The DeCaf-M Verilog model is compiled into the FPGA bit file on a Windows PC. The bit file is then downloaded into the FPGA over the parallel connection between the Windows PC and DeCaf-M motherboard.

The DeCaf-M computer uses a window on the Windows PC monitor as its screen display. This screen display is implemented by a hyperterminal utility running on the Windows PC, and communicating with the DeCaf-M computer over a serial link.

Linux was ported to the DeCaf-M Java bilingual processor. Because Linux runs on the DeCaf-M computer, the most straightforward way to provide disk storage for the DeCaf-M computer, is to have it use the disk of another Linux box, not the Windows PC. The DeCaf-M computer accesses its remote disk storage over the Ethernet.

The steps to run a Java application on the DeCaf-M computer are:

1. Turn on the DeCaf-M computer and download the DeCaf-M bit file into the FPGA.
2. Open the DeCaf-M screen display in a window on the Windows PC by starting the Windows PC hyperterminal utility.
3. Boot the DeCaf-M computer from the boot PROM on the DeCaf-M motherboard.
4. Download Linux into the DRAM on the DeCaf-M motherboard. This causes the login prompt to appear in the Windows PC window used as the DeCaf-M screen display.
5. Login.
6. Go to the directory where the Java application (ie the benchmark) is and run it.
7. Text results, such as the benchmark scores appear on the DeCaf-M screen display- the hyperterminal window on the Windows PC.

### Cell Phones/Smart Phones

Today's cell phones and smart phones with Internet access, are capable of running Java applications that they download over the Internet. All the major cell/smart phone manufacturers have Internet access and Java enabled phones.

For Java performance testing on cell phones and smart phones, the benchmark Java application was put on a popular Java application hosting website- http://www.hostj2me.com. This website is accessible to the various cell phones and smart phones that were benchmarked.

The steps to run a Java application such as the benchmark on a cell/smart phone are:

1. Download the application from http://www.hostj2me.com/d/7373/cm3.jar. For Android, download the application "CaffeineMark" from the Android Marketplace instead.
2. Run it.
3. Text results, such as the benchmark scores appear on the phone's display.

Performance measurements were taken for:

- Blackberry 8300 Curve
- Blackberry 8700g
- Blackberry Storm
- Blackberry Bold
- HTC 8925
- HTC Google G1
- Motorola Q Global
- Nokia N81
- Palm Treo Pro
- Samsung Blackjack 2
- Samsung SGH-A737

### Linux Boxes

The benchmark was also run on Linux boxes. This was done to test the performance of several implementations of the Java Virtual Machine (JVM).

Due to processor speeds, memory system limitations, other hardware factors, characteristics of the various JVMs, etc., all cell/smart phones run the J2ME JVM.

The J2ME JVM is also the JVM that was ported to the DeCaf-M Java bilingual processor. However, unlike the cell/smart phones, there is nothing about the DeCaf-M Java bilingual processor that limits it to the J2ME JVM. For example, the large enterprise/server J2EE JVM can also be ported to the DeCaf-M Java bilingual processor.

Similarly, Linux boxes also do not have limitations that preclude the various JVMs from running optimally on them. The various JVMs tradeoff resource requirements with performance. In other words, benchmark performance will vary when using different JVMs, running on the same hardware that has enough resources to run all the JVMs optimally. Several JVMs, including the highest performance JVM, all run on Linux boxes. Therefore, Linux boxes were used to measure performance of various JVMs of interest, including the fastest known JVM (up until Aurora's hardware accelerated JVM port).

The benchmark was also re-coded in C because C typically runs significantly faster than Java. This is a major reason that more applications are not written in Java.

The J2ME JVM is not the fastest JVM. But its resource requirements are also lower. This is why it is the JVM used in all cell/smart phones. However, to truly understand the DeCaf-M Java bilingual processor Java performance, it is important to compare it to the fastest JVMs and to C, as opposed to only comparing it to devices running the J2ME JVM, even though currently, the DeCaf-M Java bilingual processor only runs the J2ME JVM.

### Benchmark

The benchmark used for performance measurements was the CaffeineMarks 3.0 benchmark. It consists of six separate tests that are run sequentially and timed. These tests are:

- Sieve- integer computations including divide and remainder
- Loop- extensive looping
- Logic- bit variable manipulation and testing
- Float- double precision floating point computations including floating point library routines
- String- character string manipulations
- Method- deeply nested method invocations and returns

A score is computed for each test. Additionally, an overall score for all six tests is also computed.

### JVMs

As mentioned above, several implementations of the JVM exist. Major differences are:

- Just in time (JIT) compilation techniques.
- Resource constraints- ie available memory, processing power
- Pure software JVMs vs hardware accelerated JVMs

There is a large amount of literature available on all these topics. This white paper only reviews the very basics of these topics so that the reader that is unfamiliar with them can better understand the performance issues and results.

When Java first appeared, a major drawback was that it was an interpretted, not compiled, language. And code that is interpretted typically runs significantly slower than code that is compiled. To alleviate this performance drawback, JVMs that compile critical sections of an application's Java code, into the object code of the target processor, were developed. These compiled sections of code run much faster than if they were interpretted. This compilation is done right before the Java application is executed. Thus, these compilers are known as Just In Time compilers, or JITs.

JITs solved the Java performance problem, but they introduced new disadvantages. Two major disadvantages of JITs are that they need a lot of memory and a lot of processing power. Servers and other enterprise computers tend to have the memory and processing power needed by JITs. However, the cost sensitive and battery powered consumer devices do not have the memory and compute power required by the highest performance JITs, such as Sun's J2EE HotSpot JIT. Lower performance JITs that can run in smaller memory footprints and need less compute power, have also been developed, such as Sun's J2ME CLDC HotSpot Implementation (CLDC HI).

Another solution to improve Java performance on resource constrained devices is to add a hardware Java accelerator to them. The most popular hardware Java accelerator is ARM's Jazelle. Its performance is comparable to the pure software JIT based JVM for resource constrained devices- J2ME CLDC HotSpot Implementation (CLDC HI). The DeCaf-M Java bilingual processor also provides a hardware Java accelerator for resource constrained devices, and it is significantly faster than ARM's Jazelle (see the "Measured Performance" section below).

## Measured Performance

The CaffeineMarks Overall Score for the DeCaf-M Java bilingual processor is 19.9 CaffeineMarks/MHz (CM/MHz). This is three to thirty times faster than cell/smart phone Java performance. Additionally, it compares favorably with the fastest known Java performance- the server/enterprise J2EE HotSpot JIT based JVM that has a CaffeineMarks Overall Score of 18.5 CM/MHz. The benchmark re-coded in C has an Overall Score of 17.4 CM/MHz. Thus, the DeCaf-M Java bilingual processor is as fast as the C version of the same application.

The table below shows the CaffeineMarks Overall Score normalized to the processor clock rate (CM/MHz) for many cell/smart phones, including some of the newest, state-of-the-art smart phones. All cell/smart phones run the J2ME JVM that is for resource constrained devices.

| Phone | Applications Processor | CM/MHz | Speedup Using DeCaf-M CM/MHz |
|---|---|---|---|
| Blackberry 8300 Curve | Intel PXA901 @ 312 MHz | 0.7 | x 27.7 |
| Blackberry 8700g | Intel PXA901 @ 312 MHz | 0.7 | x 29.0 |
| Blackberry Storm | Qualcomm MSM 7600 @ 524 MHz | 0.8 | x 25.8 |
| Blackberry Bold | Marvell Tavor PXA930 @ 624 MHz | 0.9 | x 22.3 |
| HTC 8925 | Qualcomm MSM7200 @ 400? MHz | 7.4 | x 2.7 |
| HTC Google | Qualcomm MSM7201A @ 528 MHz | 0.6 | x 32.9 |
| Motorola Q Global | TI OMAP 2420 @ 325 MHz | 2.8 | x 7.2 |
| Nokia N81 | Freescale iMX31L @ 369 MHz | 4.9 | x 4.0 |
| Palm Treo Pro | Qualcomm MSM7201 @ 400 MHz | 1.0 | x 19.7 |
| Samsung Blackjack 2 | TI OMAP 1710 @ 260 MHz | 6.2 | x 3.2 |
| Samsung SGH-A737 | Qualcomm @ 225 MHz | 4.4 | x 4.5 |

Cell/Smart Phone Java Performance

The Java performance of DeCaf-M that runs the slower J2ME JVM is also compared to the Java performance of other JVMs. The table below shows the CaffeineMarks Overall Score for various JVMs all running on a Linux box.

| Other JVM | Device Class | CM/MHz | Speedup Using DeCaf-M CM/MHz |
|---|---|---|---|
| J2EE HotSpot JIT | Enterprise, servers (not resource constrained) | 18.5 | x 1.1 |
| J2ME HotSpot Implementation (JIT) | Consumer (somewhat resource constrained, no commercial examples found) | 4.7 to 6.6 | x 3.0 to x 4.3 |
| J2ME without JIT | Consumer, cell/smart phones (resource constrained) | 0.7 | x 28.7 |

Java Performance of Various JVMs on Linux Boxes

## Summary

Aurora's DeCaf-M Java bilingual processor Java performance is three to thirty times faster than the Java performance of state-of-the-art cell/smart phones. It is also about 10% faster than the fastest known Java- the J2EE HotSpot JIT JVM, that can only be run on servers and other enterprise level computers with large amounts of memory and high performance CPUs. Additionally, it is about 10% faster than a C version of the same application.