

GNU Compiler for Java

Thomas Liebetraut

Albert-Ludwigs-Universität Freiburg

December 2010

Introduction

- Part of the GNU Compiler Collection (GCC)
- Native and Bytecode compiler for Java
- GCJ is not Java ;-)

Historical Overview

- Java™ 1.0: 1995
- Kaffe
 - started Jan 1996, working JIT by June
 - highly portable (50+ different architecture/platform combinations!)
 - 2002: merge of GNU Classpath
- GNU Classpath
 - started 1998
 - free implementation of Java runtime library

Historical Overview

- Java™ 1.0: 1995
- Kaffe
 - started Jan 1996, working JIT by June
 - highly portable (50+ different architecture/platform combinations!)
 - 2002: merge of GNU Classpath
- GNU Classpath
 - started 1998
 - free implementation of Java runtime library

Historical Overview

- GCJ

- first release: September 6, 1998
- parallel to the above (legal reasons)
- from 2000 on: incorporated GNU Classpath
- 2007: includes ECJ (Eclipse Compiler for Java) as frontend (JDK1.5 compatible)

- OpenJDK

- release of (most of) Sun's sourcecode under free license (2006-2007)
- IcedTea was RedHat's project to fill in the missing parts with free software
- IcedTea may now use OpenJDK brand
- builds on a completely free system

- Unfortunately, this led to a stall of GCJ development by mid-2007

Historical Overview

- GCJ

- first release: September 6, 1998
- parallel to the above (legal reasons)
- from 2000 on: incorporated GNU Classpath
- 2007: includes ECJ (Eclipse Compiler for Java) as frontend (JDK1.5 compatible)

- OpenJDK

- release of (most of) Sun's sourcecode under free license (2006-2007)
- IcedTea was RedHat's project to fill in the missing parts with free software
- IcedTea may now use OpenJDK brand
- builds on a completely free system

- Unfortunately, this led to a stall of GCJ development by mid-2007

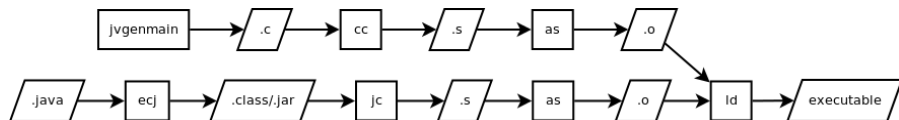
- GCJ
 - first release: September 6, 1998
 - parallel to the above (legal reasons)
 - from 2000 on: incorporated GNU Classpath
 - 2007: includes ECJ (Eclipse Compiler for Java) as frontend (JDK1.5 compatible)
- OpenJDK
 - release of (most of) Sun's sourcecode under free license (2006-2007)
 - IcedTea was RedHat's project to fill in the missing parts with free software
 - IcedTea may now use OpenJDK brand
 - builds on a completely free system
- Unfortunately, this led to a stall of GCJ development by mid-2007

Components of "GNU Java"

- GCJ itself
- GIJ (GNU Interpreter for Java)
- libgcj (runtime library)
- GNU Classpath

- Compile Java source to native (`gcj --main=SomeClass SomeClass.java`)
- Compile Java bytecode to native (`gcj --main=SomeClass SomeClass.class`)
- Compile Java source to bytecode (`gcj -C SomeClass.java`)

GCJ compile process (native)



Some jc internals (as of fall 2005, pre-ecj)

Nothing special here:

- Several parse passes due to forward dependencies
- Generates GENERIC tree language like the other frontends
- From then on: processing and optimization as presented by Jens

- Merge of different libraries
 - libgcj (legacy)
 - GNU Classpath
 - Kaffe
 - ...
- incorporates Java classloader and bytecode interpreter
- Garbage Collector (boehm-gc)

Memory Management

- GCJ makes use of a GC initially written for C++
- Boehm-gc (by Hans Boehm)
 - runs when there is no more memory available
 - mark-sweep algorithm
 - mark all blocks **known** to be referenced
 - build transitive hull by searching for patterns that may be pointers
 - search all blocks for unmarked blocks and free them

Performance: GCJ bytecode vs Java™

- GCJ creates almost the same bytecode, so same performance
- GJ's memory allocation and access slower than in Hotspot
- GJ's garbage collection is less efficient
- GCJ does not favor GJ in any way

Performance: GCJ native vs Java™

- Native binary somewhat faster on number crunching
- Memory operations still very slow
- Garbage collection kicks in earlier
- No switches to tweak memory management
- Native C++ code with all optimizations a lot faster

Conclusion

- GCJ is a robust Java 1.5 to native compiler with decent performance
- Memory-intensive things should be left to Hotspot
- Sun's expertise is good for something
- A lot of work has to be done regarding GJ, JIT and memory management

References

<http://www.cs.wustl.edu/~mdeters/seminar/fall2005/>

<http://lwn.net/Articles/171139/>

<http://lwn.net/Articles/130796/>

http://zigabyte.com/blog/GCJ_vs_Java_JIT_Performance_Comparison.doc

<http://kaffe.org/doc/slides/kaffe-past-present-future-FOSDEM05.pdf>

Thanks for all the GNU