Compiling Open Babel to Pure Java Bytecode with NestedVM - An Unsuccessful First Attempt

Richard Apodaca

Blog post published November 19, 2007 in *Depth-First*
https://doi.org/10.59350/baa6m-gzr66

Abstract

Wouldn't it be great to be able to compile code written in languages like FORTRAN, C, and C++ to Java bytecode? NestedVM - almost magically - can do just that. This article documents a failed first attempt to compile the popular cheminformatics toolkit Open Babel, which is written in C and C++, to pure Java bytecode with NestedVM.

Keywords

Physical and Theoretical Chemistry (Subfield)

Recommended citation

Apodaca, R. (2007, November 19). Compiling Open Babel to Pure Java Bytecode with NestedVM - An Unsuccessful First Attempt. *Depth-First*. https://doi.org/10.59350/baa6m-gzr66

Copyright

Copyright © Richard Apodaca 2007. Distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Wouldn't it be great to be able to compile code written in languages like FORTRAN, C, and C++ to Java bytecode? NestedVM - almost magically - can do just that. This article documents a failed first attempt to compile the popular cheminformatics toolkit Open Babel, which is written in C and C++, to pure Java bytecode with NestedVM.

A previous article described the successful compilation of the InChI toolkit, a C library, to a platform-independent executable jarfile.

https://doi.org/10.59350/baa6m-gzr66

The Problem

Open Babel is one of cheminformatics' most widely-used open source packages. It interconverts dozens of molecular languages, performs a host of cheminformatics analyses, and serves as a platform for many programs and Web services.

As useful as Open Babel is, it doesn't run directly on a Java Virtual Machine (JVM). Although an Open Babel JNI interface does exist, using it introduces a platform dependency, which in many cases is not acceptable. JNI is a great solution in some cases, but when maintaining a single version of a program is important, or when applets need to be used, or when code needs to work with unusual system configurations, it's a poor choice.

Our goal is to compile Open Babel's "babel" command-line utility into pure Java bytecode that can be run on any recent JVM without using JNI.

Overview of NestedVM

In a nutshell, NestedVM converts MIPS binaries to Java class files. In theory, this allows software written in any language that can be compiled to a MIPS binary to be run on a JVM.

To do this, NestedVM distributes two categories of tools: (1) a complete MIPS cross-compiler toolchain; and (2) a MIPS binary to Java bytecode compiler and accessories.

Building NestedVM

The preferred method to install NestedVM is to compile it from source found in the project repository. There are a number of prerequisites your system must meet in order to be able to do so. For now, this article assumes your system has all of them. Some of the following steps can be found in these instructions as well.

To obtain the source code from the NestedVM darcs repository:

```
darcs get --repo-name=nestedvm http://nestedvm.ibex.org
```

Then change into the **nestedvm** directory and build the main code:

```
cd nestedvm
make
```

On my machine, this step takes 10-15 minutes.

To make sure your build works, run the tests:

```
make test
...
1.574000e+00
-4.315000e+01l
-43
-4.315000e+01
4.315000e+01
Hello, World
7F
fabs(-2.24) = 2.34
Destructor!
```

NestedVM doesn't build the g++ compiler by default - it's something that needs to be done manually. Fortunately, it's not difficult to do:

```
make cxxtest
...
java -cp build tests.CXXTest
Test's constructor
Name: 0x50b40
Name: PKc
Is pointer: 1
Name: 0x50b3c
Name: i
Is pointer: 0
Hello, World from Test
```

```
Now throwing an exception
sayhi threw: const char *:Hello, Exception Handling!
Test's destructor
```

Finally, with all tools built, we need to set up our environment:

```
make env.sh
source env.sh
```

We're now ready to cross-compile Open Babel.

https://doi.org/10.59350/baa6m-gzr66

Cross-Compiling Open Babel

For this tutorial, we'll use the Open Babel 2.1.1 source distribution. Unpack the tarball and change into the directory.

Next, we'll need to set up our cross-compiler environment. Fortunately, NestedVM has made this easy. If you check your environment variables, you'll find that `CXX` and `CC` have both been set. All that remains is to notify the configure script that we'll be cross-compiling:

```
./configure --host=mips-unknown-elf
```

Then we build the MIPS binaries:

```
make
```

Peeking into the `tools` directory, we can see all of the Open Babel command line tools have been built, including `babel`.

Unless you're running a MIPS machine, though, this binary won't be executable.

So far, it looks like everything worked. Although it didn't work the first time I tried it, the NestedVM team were most helpful.

Building the Java Class File

We're now ready for the final stage in the process, converting the MIPS binary to a Java class file. Again, NestedVM makes this simple:

```
cd tools
java org.ibex.nestedvm.Compiler -outfile Babel.class Babel babel
Exception in thread "main" java.lang.IllegalStateException: unresolved phantom target
        at org.ibex.classgen.MethodGen.resolveTarget(MethodGen.java:555)
        at org.ibex.classgen.MethodGen._generateCode(MethodGen.java:664)
        at org.ibex.classgen.MethodGen.generateCode(MethodGen.java:618)
        at org.ibex.classgen.MethodGen.dump(MethodGen.java:888)
        at org.ibex.classgen.ClassFile._dump(ClassFile.java:193)
        at org.ibex.classgen.ClassFile.dump(ClassFile.java:160)
        at org.ibex.nestedvm.ClassFileCompiler.__go(ClassFileCompiler.java:380)
        at org.ibex.nestedvm.ClassFileCompiler._go(ClassFileCompiler.java:72)
        at org.ibex.nestedvm.Compiler.go(Compiler.java:259)
```

Unfortunately, NestedVM has blown up with an exception. Although our target class file, **Babel.class** is now in our working directory, it is not complete and won't run.

What Went Wrong?

After bringing this problem to the NestedVM mailing list, it appears that this is a NestedVM bug.

However, the way `babel` works is to load its various language modules dynamically. It may be possible to fix the problem by producing a version of `babel` containing all of its modules in a single binary.

Although there is a major issue to be resolved, this tutorial illustrates the full process of compiling C++ code to Java bytecode using NestedVM.