For the eight-month period from 1-Jan-2007 to 31-Aug-2007...

I spent most of my time working on the Dalvik VM (the Java-ish virtual machine used on the Android platform). The VM was first made generally available within Android in January, replaced our previous VM (JamVM) as the default in March, and was sufficiently fast and trouble-free that we dropped support for JamVM in July. Highlights:

- Reworked various pieces to maximize data-sharing between processes. For example, the Dalvik classfiles are now uncompressed to cache files after being byte-swapped and aligned, allowing direct access from C to shared read-only memory-mapped pages. The VM's "write once" data, such as method names and signatures, are stored in a separate allocation area to minimize the "write" in "copy-on-write" after a fork().

- Improved VM performance significantly. I developed a "DEX optimizer" that rewrites bytecodes as the classfiles are uncompressed, and created a low-overhead "inline" native call mechanism. I removed debugger overhead from the interpreter while retaining support for the debugger. I learned ARM assembly and rewrote the JNI call bridge in it. I wrote several Java performance tests to exercise the VM, and ran them frequently to measure results.

- Implemented bytecode verification. This involves verifying that classes are constructed properly, and "executing" every bit of code in every class to ensure that "bad stuff" can't happen, e.g. passing an integer when an object reference is expected.

- Continued development of core features. Most of the major items were in place, but we still needed a lot of little things (uncaught exception handling, Java language asserts, etc.) that are expected in a commercial VM.

- Improved support for Java debuggers. IntelliJ is now fully supported, and all common JDWP command-line options are handled.

- For the first half of the year we supported both Dalvik VM and JamVM simultaneously (switchable at runtime). I maintained JamVM until we dropped it.

Another major item was the development of the Dalvik Debug Monitor system. Debugging a Java app on the device required a complicated series of steps. The Dalvik Debug Monitor Server (DDMS) monitors all Java processes running on a device, allows you to examine thread and heap information, and provides pass-through ports for easy debugger connections. I borrowed a book on the SWT toolkit to write the GUI interface, and piggybacked the DDM messages on top of JDWP (the Java Debug Wire Protocol). (The stand-alone DDMS app has since been transformed by xav into a much nifter Eclipse plug-in.)

I used what I learned while working on the Dalvik VM to significantly rewrite and expand our "Efficient Java" document, which describes best practices for Java development on small embedded systems.

I frequently worked with others to measure and resolve performance problems, providing analysis or adding instrumentation to the VM.

- I write good code. It does what it's supposed to, and is well commented. (The Dalvik VM sources have extensive descriptions of "why" as well as "what".) I fix bugs as they come up, even if they're not related to the feature I'm currently working on.

- I find and fill needs before their absence becomes critical. Debugging Java apps on the device was difficult, so I made it easy with DDMS. The tools were developed and stable before the team shifted from the simulator to the device as the primary development platform.

- I pick up new technologies quickly. I learned how to write GUI apps with the SWT toolkit so I could develop DDMS, and I learned how to program in ARM assembly language to make a piece of the VM run correctly (and quickly!).

Last review, horowitz wrote:

"I would like to see Andy reach out more and offer his advice and guidance on system level architecture, even in areas he is not responsible for."

I've stayed pretty focused on the VM.