EXHIBIT 2-13

Dalvik JIT

Posted by Tim Bray on 25 May 2010 at 2:57 PM

[This post is by Dan Bornstein, virtual-machine wrangler. — Tim Bray]

As the tech lead for the Dalvik team within the Android project, I spend my time working on the virtual machine (VM) and core class libraries that sit beneath the Android application framework. This layer is mostly invisible to end users, but done right, it helps make Android devices run smoothly and improves developer productivity.

The 2.2 release is particularly pleasing to me, as it is the first release since before 1.0 in which we have been able to deliver significantly new VM technology. And unlike much of what my team and I do, it is something that can be experienced directly by end users.

“Dalvik” isn’t exactly a household word (at least in my country), and most people wouldn’t know a virtual machine if it hit them in the face, but when you tell them you were able to make their existing device work better — run faster, use less battery — they will actually take notice!

What Makes This Possible?

We added a Just In Time (JIT) compiler to the Dalvik VM. The JIT is a software component which takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run. If you want to learn more about the design of the Dalvik JIT, please watch the excellent talk from Google I/O 2010 given by my colleagues Bill Buzbee and Ben Cheng, which should be posted to YouTube very soon.

To be clear, the differences aren’t always dramatic, nor do they apply uniformly to all applications. Code that is written to run the CPU all-out can now do more in the same amount of time (running faster), and code that is written to be rate-limited can get its work done using less time and less of the CPU (using less battery). On the

Tags

performance front in particular, we have seen realistic improvements of 2x to 5x for CPU-bound code, compared to the previous version of the Dalvik VM. This is equivalent to about 4x to 10x faster than a more traditional interpreter implementation.

The team is proud of our new JIT in general, but we are especially proud of two aspects of it:

Many previous JIT implementations react slowly, delivering performance improvements only after a long warm up period. In the extreme, it can be minutes or even hours before the code is fully up to speed. On the other hand, the Dalvik JIT reacts quickly, so that mere moments after you hit the “Start” button on your favorite game, you are already benefiting from the work of the JIT.

We are also very pleased with how little memory the JIT uses. The code for the JIT itself is well under 100k, and each process that the JIT runs in will typically only use another 100k or so of RAM. On the current generation of Android phones, device users won’t even notice this additional memory usage; on my own phone, I can still have literally dozens of applications warmed up in memory and ready to go.

The Dalvik team isn’t resting on its laurels, either. We are hoping to see the Dalvik JIT deployed on many devices in the coming months. Looking forward, the team has an endless list of ideas for making the VM and library code better, which we are diligently working on.

TRACKBACKS

- Froyo + G1 = EPIC FAIL
- Android 2.2 □□□□□□
- Android 2.2 “Froyo”
- Going Deeper With Android 2.2's JIT Compiler
- Dalvik and JIT on Android

Create a Link

Newer Post Home Older Post