Build Community Android Distribution and Ensure the Quality

Jim Huang (黃敬群) <jserv@0xlab.org>
Developer & Co-Founder, 0xlab

http://0xlab.org/

Oct 28, 2011 / ELC Europe (Prague)
Rights to copy

© Copyright 2011 0xlab
http://0xlab.org/
contact@0xlab.org

Corrections, suggestions, contributions and translations are welcome!


Attribution – ShareAlike 3.0
You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

**BY:** Attribution. You must give the original author credit.

**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: [http://creativecommons.org/licenses/by-sa/3.0/legalcode](http://creativecommons.org/licenses/by-sa/3.0/legalcode)
0x1ab = $16^2 + 16 \times 10 + 11 = 427$
(founded on April 27, 2009)

0xlab is another Hexspeak.
About Me

(1) Come from Taiwan
(2) Contributor of Android Open Source Project (AOSP)
(3) Developer, Linaro
(4) Contributed to GNU Classpath / Kaffe, Linux internationalization (i18n), Openmoko
Commercial Partners of 0xlab

**ARM / Linaro**
Contribute to Linaro Android since the first line of code

**AzureWave**
Build wireless networking & image processing solutions

**Mediatek**
Android based consumer products

**Open Embedded Software Foundation**
Contribute to the reference implementation

Eventually, partners can benefit from open source efforts and our experience.

http://0xlab.org/partners.html
Agenda

(1) Build Android distribution
(2) Lesson learned from AOSP
(3) Ensure the Quality
(4) Bring enhancements back to Community
Build Android Distribution
based on non typical open source projects
The reason why we built community Android Distribution:

Initially, we just wanted to enable wireless connectivity features on Android for our hardware partners.

But, we never thought that it was difficult to do things efficiently.
0xdroid: enhanced Android distribution

We suffered from performance and usability issues in AOSP. 0xdroid is basically the environment where we can develop and experiment.
What does Oxdroid deliver?

- Hardware enablement: Beagleboard (TI OMAP3), Pandaboard (TI OMAP4), Snowball (ST-Ericsson Ux500; on-going)
- Provide full source code for HAL
- Usability: software cursor, window manager fix, large screen tweaks, network connectivity fix
- Performance: ARM specific optimizations, graphics enhancement
- Features: Bluetooth HID (keyboard/mouse), external modem, 3D effects, customized Launcher
But, Oxlab is not really making yet another Android distribution. We wish to help community.

Oxdroid is just a *testbed* (or reference implementation), and the valuable changes should be merged in upstream or other community projects.
• open source efforts to improve AOSP
• We focus on small-but-important area of Android.
  – toolchain, libc, dynamic linker, skia, software GL, system libraries, HAL, UX

• Develop system utilities for Android
  – benchmark, black-box testing tool, validation infrastructure

• Feature driven development
  – Faster boot/startup time, Bluetooth profile, visual enhancements

• Submit and share changes to...
  – AOSP, CyanogenMod, Android-x86, and Linaro
Working Model by 0xlab
Lesson Learned from AOSP
Let's go Upstream!
Unfortunately, contributing to AOSP is an __art__. You never know how Google thinks of your patches exactly, even through Gerrit (code review system).
Problems We faced

- AOSP looks like “An Open Source Pretender"
- No public roadmap
  - Therefore, we ignore the modifications against Android framework.
- The merged changes usually show up in next 1 or 2 public release.
  - It is really hard to introduce/track the relevant changes.
- Not clear discussions on android-contrib mailing-list. Sometimes, you have to have private communications to Google engineers.
- Version control / Code Review on invisible repositories (internal and far-away GIT tree)
Google provides the great code review tool for AOSP, but...
Flow of AOSP submitted changes

Create Local Branch

work

Fix commit

rebased!

Push to Gerrit

Review

approved/submitted

Upstream repo

Here “upstream” means AOSP master and Google internal tree.
Gap between Google internal GIT/Perforce and AOSP Git

No explicit change history

Android style open source model

projects / toolchain/gcc.git / tree

summary | shortlog | log | commit | commitidiff | tree | review
snapshot

Remove definition of getpagesize() in order to build c++ library
After Gingerbread, it gets much clear for toolchain part.

Patched toolchain to fix a few gcc and binutils bugs.

Sources to build this toolchain are listed on arm-eabi-4.4.3/SOURCES.

And rollback the following 3 patches.

Although we can check git log, we still have no idea why they changed.
• “master” branch in Android is the bridge between AOSP and Google internal tree. There are many contributions merged from companies, organizations, and individuals. But no efficient code review available for non-existing repositories. And, only few Google engineers do review changes.

• Master branch = the latest AOSP + Partial changes by Google (bug-fixes from internal tree)
  – Not fully verified codebase.

• The best hints are the opinions written by Google engineers inside Gerrit.
  – Send patches if possible
My interpretation of Android:

Hardware is Revolution;
Software is basically Evolution;
Android is Hardware-driven Software Revolution
The overall design is consistent, but the current model prevents from diverse community contributions.
AOSP statistics for Gingerbread (Dec 2010)

- 4204 google.com
- 1354 android.com
- 98 sonyericsson.com
- 71 gmail.com
- 39 codeaurora.org
- 39 samsung.com
- 38 intel.com
- 32 nokia.com
- 32 holtmann.org
- 29 0xlab.org
- 25 trusted-logic.com
- 17 openbossa.org
- 11 nxp.com
- 11 linux.org.tw
- 10 ti.com
- 10 acer.com.tw
- 8 themaw.net
- 8 garmin.com
- 7 snpe.rs
- 7 motorola.com
- 7 mc.pp.se
- 7 googlemail.com

The number are commits since Froyo release. However, the valuable changes from community such as CyanogenMod are usually absent due to long-time review process.
Ensure the Quality
text here...
Mission in our development:

Improve UX in SoC

UX = User Experience
SoC = Integrated Computing Anywhere
Quality in custom Android Distribution

• 0xlab delivers the advantages of open source software and development.
  – Quality relies on two factors: continuous development + strong user feedback

• Several utilities are developed to ensure the quality and released as open source software.
  – 0xbench (Android benchmarking tool)
  – ASTER (Android System Testing Environment and Runtime)
  – LAVA (Linaro Automated Validation Architecture)

• In the meanwhile, performance is improved by several patches against essential components.
Android benchmark running on LAVA. Automated Validation flow includes from deploy, then reboot, testing, benchmark running, and result submit.

Android support on LAVA
https://wiki.linaro.org/Platform/Validation/LAVA

Android related commands in LAVA:
* deploy_linaro_android_image
* boot_linaro_android_image
* test_android_basic
* test_android_monkey
* test_android_0xbench
* submit_results_on_host
0xbench: comprehensive open source benchmark suite for Android

- A set of system utilities for Android to perform comprehensive system benchmarking
  - Dalvik VM performance
  - OpenGL|ES performance
  - Android Graphics framework performance
  - I/O performance
  - JavaScript engine performance
  - Connectivity performance
  - Micro-benchmark: standard C library, system call, latency, Java invocation, ...
Collect and Analyze results on server-side

The Linpack Benchmark is a numerically intensive test that has been used for years to measure the floating point performance of computers.
Android Functional Testing
(1) stress test
(2) Automated test
Stress Test

• According to CDD (Compatibility Definition Document), Device implementations MUST include the Monkey framework, and make it available for applications to use.

• `monkey` is a command that can directly talks to Android framework and emulate random user input.  
  `adb shell monkey -p your.package.name -v 500`

• Decide the percentage of touch events, keyboard events, etc., then run automatically.
ASTER: Automated Test

- Blackbox-test vs. Whitebox-test
- An easy to use automated testing tool.
Functional Test

Desktop: Sikuli

1. click (Trash)
2. click (Empty)
3. click (Empty Trash)
Aster

Designed for non-programmer

Easy to use IDE

Batch executing of test scripts

Multiple chain of recall command
Bring Enhancements back to Community
What do we deliver to community?

• Patches merged in AOSP, CyanogenMod, and Android-x86
• Implement 100% open source OpenGL|ES adaptation based on Mesa/3D into Android – The world-first, important to Android-x86
• Performance: ARM specific optimizations, graphics enhancement
• Features: Bluetooth HID (keyboard/mouse), external modem, 3D effects, customized Launcher
### Arithmetic on Nexus S

#### Tuned Dalvik VM performance (armv7)

<table>
<thead>
<tr>
<th>benchmark</th>
<th>mflops</th>
<th>logarithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linpack</td>
<td>14.83</td>
<td></td>
</tr>
<tr>
<td>Scimark2:COMPOSITE</td>
<td>20.64</td>
<td></td>
</tr>
<tr>
<td>Scimark2:FTT</td>
<td>13.43</td>
<td></td>
</tr>
<tr>
<td>Scimark2:SOR</td>
<td>36.67</td>
<td></td>
</tr>
<tr>
<td>Scimark2:MONTECARLO</td>
<td>5.72</td>
<td></td>
</tr>
<tr>
<td>Scimark2:SPARSEMATMULT</td>
<td>18.37</td>
<td></td>
</tr>
<tr>
<td>Scimark2:LU</td>
<td>28.99</td>
<td></td>
</tr>
</tbody>
</table>

---

### Tuned Dalvik VM performance (armv7)

<table>
<thead>
<tr>
<th>benchmark</th>
<th>mflops</th>
<th>logarithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linpack</td>
<td>15.56</td>
<td></td>
</tr>
<tr>
<td>Scimark2:COMPOSITE</td>
<td>21.84</td>
<td></td>
</tr>
<tr>
<td>Scimark2:FTT</td>
<td>14.01</td>
<td></td>
</tr>
<tr>
<td>Scimark2:SOR</td>
<td>38.53</td>
<td></td>
</tr>
<tr>
<td>Scimark2:MONTECARLO</td>
<td>5.92</td>
<td></td>
</tr>
<tr>
<td>Scimark2:SPARSEMATMULT</td>
<td>19.41</td>
<td></td>
</tr>
<tr>
<td>Scimark2:LU</td>
<td>31.30</td>
<td></td>
</tr>
</tbody>
</table>
### 2D on Nexus S

Apply extra performance tweaks against optimized build (NEON)

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td>55.56</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td>29.15</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td>51.23</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td>32.81</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td>47.12</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td>53.36</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td>55.29</td>
<td></td>
</tr>
</tbody>
</table>

### run_Nexus_S:GRJ90_2011/08/03-18:11:16UTC

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td>56.06</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td>33.19</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td>49.87</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td>42.42</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td>54.64</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td>55.85</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td>55.44</td>
<td></td>
</tr>
</tbody>
</table>
## Benchmark: 2D (arm11-custom)

### Options

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td>49.93</td>
<td></td>
<td>48.38</td>
<td></td>
<td>14.65</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td>23.29</td>
<td></td>
<td>22.68</td>
<td></td>
<td>10.32</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td>18.84</td>
<td></td>
<td>18.80</td>
<td></td>
<td>9.77</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td>7.64</td>
<td></td>
<td>8.80</td>
<td></td>
<td>5.76</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td>14.92</td>
<td></td>
<td>14.32</td>
<td></td>
<td>8.40</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td>5.59</td>
<td></td>
<td>5.50</td>
<td></td>
<td>3.10</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td>19.56</td>
<td></td>
<td>19.44</td>
<td></td>
<td>9.00</td>
<td></td>
</tr>
</tbody>
</table>

### M3 + Linaro Toolchain

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td>58.35</td>
<td></td>
<td>58.57</td>
<td></td>
<td>38.64</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td>38.91</td>
<td></td>
<td>37.53</td>
<td></td>
<td>22.32</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td>18.67</td>
<td></td>
<td>17.92</td>
<td></td>
<td>19.64</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td>19.71</td>
<td></td>
<td>19.26</td>
<td></td>
<td>16.23</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td>26.84</td>
<td></td>
<td>24.68</td>
<td></td>
<td>24.66</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td>6.73</td>
<td></td>
<td>6.69</td>
<td></td>
<td>6.22</td>
<td></td>
</tr>
<tr>
<td>DrawImage2</td>
<td>19.16</td>
<td></td>
<td>19.06</td>
<td></td>
<td>15.69</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td>29.22</td>
<td></td>
<td>29.28</td>
<td></td>
<td>25.66</td>
<td></td>
</tr>
</tbody>
</table>

### M3

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38.64</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.32</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.64</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16.23</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.66</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.22</td>
<td></td>
</tr>
<tr>
<td>DrawImage2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.69</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.66</td>
<td></td>
</tr>
</tbody>
</table>

### 2.6.35 (2.6.32 pmem)

<table>
<thead>
<tr>
<th>benchmark</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
<th>2d-fps</th>
<th>linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrawCanvas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38.64</td>
<td></td>
</tr>
<tr>
<td>DrawCircle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.32</td>
<td></td>
</tr>
<tr>
<td>DrawCircle2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.64</td>
<td></td>
</tr>
<tr>
<td>DrawRect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16.23</td>
<td></td>
</tr>
<tr>
<td>DrawArc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.66</td>
<td></td>
</tr>
<tr>
<td>DrawImage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.22</td>
<td></td>
</tr>
<tr>
<td>DrawImage2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.69</td>
<td></td>
</tr>
<tr>
<td>DrawText</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.66</td>
<td></td>
</tr>
</tbody>
</table>
This explains that we have several system tools and development flow to help customers/community to verify the performance and improve.
• Android C/C++ library
• 0xlab’s Optimizations (merged in Android upstream)
  – Memory operations: Use ARMv6 unaligned access to optimize usual cases
  – Endian/Data Type conversion: Use ARMv6 fast endian primitives. Useful for TCP/IP (big endian / little endian converting)
  – Various ARM optimized string operations
    • memcpy, strcmp, strcpy, memset
Dynamic Linker Optimizations
Why and How?

• The major reason to optimize dynamic linker is to speed up application startup time.

• Approaches:
  • Implement GNU style hash support for bionic linker
  • Prelinker improvements: incremental global prelinking
    – reduce the number of ELF symbol lookup aggressively

• Changed parts
  – apriori, soslim, linker, elfcopy, elfutils
(normalized) Dynamic Link time

- bootanimation
- mediaserver
- app_process
- keyrstore
- dbus-daemon
- debugger
- servicemanager
- rild
- installd
(normalized) Symbol Lookup number
DT_GNU_HASH: visible dynamic linking improvement =
    Better hash function (few collisions)
    + Drop unnecessary entry from hash
    + Bloom filter

```c
void foo (){
    printf("fooooo");
    bar();
}
```
<table>
<thead>
<tr>
<th>Symbols in ELF</th>
<th>lookup#</th>
<th>fail#</th>
<th>gnu hash</th>
<th>filtered by bloom</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnu gp</td>
<td>3758</td>
<td>23702</td>
<td>19950</td>
<td>23310</td>
</tr>
<tr>
<td>gnu gp.re</td>
<td>3758</td>
<td>20544</td>
<td>16792</td>
<td>19604</td>
</tr>
<tr>
<td>gnu lp</td>
<td>61750</td>
<td>460996</td>
<td>399252</td>
<td>450074</td>
</tr>
<tr>
<td>gnu lp.re</td>
<td>61750</td>
<td>481626</td>
<td>419882</td>
<td>448492</td>
</tr>
</tbody>
</table>

Bit array

\[ H = \{x, y, z\} = \text{hash functions} \]

Hash function may collision → Bloom filter may got false positives
Bluetooth HID (Keyboard/Mouse)
UI customizations

• Provide several UI/Launcher combination for small and large screen devices.
  • Sizing from HVGA, VGA, SVGA (Phone), to 720p/1080p (TV)

• Either modified Android Launcher or new replacement

• Licensed under Apache Software License
Some UI Changes

- Hardware enablement: Beagleboard (TI OMAP3),
- BottomBar
  Source code: [http://gitorious.org/0xdroid/packages_apps_launcher](http://gitorious.org/0xdroid/packages_apps_launcher)
- PositionBar
  Visible Hint
- ThemeSelector
  [http://code.google.com/p/0xdroid/wiki/LauncherTheme](http://code.google.com/p/0xdroid/wiki/LauncherTheme)
Products with Advanced 3D UI

- 3D effects and the ARM optimizations are enabled in Qualcomm platforms: MSM7x27 (with GPU) and MSM7x25 (software only)
Android Boot Time Optimizations
Qi Boot-loader

- **Only one** stage boot-loader
- **Small footprint** ~30K
- Currently support
  - iMX31
  - Samsung 24xx
  - Beagleboard
- **KISS concept**
  - Boot device and load kernel

<table>
<thead>
<tr>
<th></th>
<th>Qi Boot-loader</th>
<th>U-Boot + XLoader</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>~30K</td>
<td>~270K+20K</td>
</tr>
<tr>
<td><strong>Time to Kernel</strong></td>
<td>&lt; 1 s</td>
<td>&gt; 5s</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>Product</td>
<td>Engineering</td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td>Simple</td>
<td>Complicated</td>
</tr>
</tbody>
</table>
Optimized ARM Hibernation

• Based on existing technologies thus requires little modification to userspace
  – TuxOnIce
• Release clean-pages before suspend
• Swap out dirty-pages before save image
• Image size reduced leads to faster resume time.
Further Boot Time Optimization

- Save the heap image (like core dump) of Zygote after preloading classes
- Modify Dalvik to make hibernation image after system init and before Launcher startup
- Parallize Android init
- Cache & Share JITed code fragment
Resources

- 0xdroid Roadmap: http://code.google.com/p/0xdroid/wiki/Roadmap
- Source repository: http://gitorious.org/+0xlab
- Wiki: http://code.google.com/p/0xdroid/w/list
- Demo videos: http://www.youtube.com/0xlab
- Mailing-list:
  - General discussion: http://groups.google.com/group/0xlab-discuss
  - Technical / Development: http://groups.google.com/group/0xlab-devel
- IRC channel (FreeNode): #0xlab
Thanks for Attending

Special thanks to AzureWave, who sponsors me for a long time.

Any Questions?