Google Android Experiences in porting, tips and tricks

Mark Gross

April 12, 2010

Introduction and apology

- I don't have as many tricks to share as I planned.
- This is meant to be an talk, not a lecture.
- I work directly for the product group that is creating the mobile chip sets and platforms
- I've suddenly gotten really busy lately.
  - These are ugly slides, sorry.

Approximate Outline

- What Linux could learn from Google Android
- Google Android PM - the good and the ugly
- Out of tree kernel code == /me getting good at git rebase.
- Google Android Graphics – simple but still difficult (for Intel)
- Performance.
- Fastboot
- Fastboot implemented on top of kboot
- Repo, tips and tricks
- SCM approaches for android
- Garret, first impressions.
- Hacking repo (adding a format-patch command)
- Things Intel is focused on WRT Android..AFAIK.
- Closing observations and babbling

What embedded Linux could learn from Google Android

- Frameworks are important for creating developer communities.
  - User mode PM and application life cycle standards are important.
  - ISV's need to be able to develop applications without being system integrators and kernel hackers is important
- Integration enabling is important.
  - Logcat, ADB, fastboot

Android PM the good the, meh, and the ugly

- Good
  - Its a complete solution out of the box.
  - The wake lock concept in user mode is pretty cool, at least its a standard all Android applications can follow.
- Meh (I'm not ready to say “bad” yet...)
  - ABI assumed by stack (early suspend notification, wake locks)
    - Early suspend is actually the one I like the least.
  - The suspend notification goes all the way up into the surface flinger and helps control graphics rendering at screen on/off time.
    - Worker threads doing blocking reads on wait_for_fb_sleep and wait_for_fp_wake in /sys/power.
  - brakes PM for a typical Linux stack running on top of an android enabled kernel
- Ugly
  - Grabbing and releasing wake locks in kernel is bad.
  - You can't just have a few, you grab and release one you'll quickly end up with wake_lock-itis throughout your kernel

Out-of tree enableing == lots of git rebase-ing

- Our current kernel is a patched 2.6.31.6 kernel initially developed for moblin.
  - We'll move to 2.6.3x sometime this summer after it gets stable.
- I rebased the patches to the android.git.kernel.org/kernel/common.git android-2.6.32 branch to it.
  - It worked ok, only a few fix ups needed.
  - Not going to scale well in the future.
  - Scared that wake_lock-itis patch sets making this impossible to deal with over time.
- Gripe: why did they need to add the 2.6.32.9 patches to the android-2.6.32 branch to common.git?
  - Thanks for making he harder for me.

Google Android graphics

- Its dumb, yet still a PIA
- Lots of 2D rendering all done in SW on CPU
- 3D is really only used for texture blitting of the 2D buffers by the surface flinger.
- Games and NDK applications drive more complete utilization of the HW on most of the platforms
- Rumors of it getting overhauled on the net.

Performance

- First order hot spots for SW graphics are:
  - memcpy
  - skia
  - Memset16
- After enabling HW graphics:
  - Skia
  - assorted
- Oprofile
  - Needed x86 enabling to work. (done.)
- Vtune.
  - Works if you put my LFS /tools hack in the root FS.

Fastboot is cool.

- Fast boot is a USB gadget based application for automated target update and booting.
- It with ADB you can automate, zero touch, validation builds prior to change set acceptance
- We implemented Fastboot as an application on top of Kboot.

Kboot implementation of fastboot

- Starting with android.git.kernel.org/kernel/lk.git I implemented an application and gadget driver hack that implements fast boot within kboot.
  - Note: there is also a fast boot implementation in bootable/bootloader/legacy
  - Fastboot host application is in system/core/fastboot
- It works pretty well.
- After using it for a short time it becomes easy to see why google insists on it for anything they run in their lab.

Repo tricks

- Repo forall is useful
- Environment variables REPO_PROJECT, REPO_PATH, REPO_REMOTE
- Read some python see .repo/repo/commands/forall.py
- Export REPO_TRACE=1 is handy to see what git commands are happening
- Repo manifest -r -o tag.xml
- Its just python code.
- Use ipython and python debug tricks to explore what's goping on in it

Repo tricks

- Repo forall -c 'git diff remote/branch'
- Repo forall -c 'echo $REPO_PATH;git remote -v'
  - In AOSP to build a script to set AOSP upstream remotes from which to compare and merge with.
- Repo manifest -r -o my-tag-file.xml

Hacking repo

- I have a patch to repo I'm trying to get cleaned up an accepted. It does a format patch off a “tag” manifest.xml file.
- Repo is just some python code.
- Its pretty fun to hack on.
- Repo is eventually moving to git subproject manifests and away from the xml files.
- I would like to see better project branching support exposed through repo and garret.

SCM options:

- Just use repo forall to aggregate the git projects, and lay down test and release branches.
  - Set up repo mirror + manifest
  - Repo forall -c'git branch test;
  - Repo forall -c'git status'
  - …
- Use garret
  - Its becoming “the android way of doing things”

SCM garret fist impressions:

- Takes getting used too
- Its focus is on integration.
- Really locks down the integration process
  - A good thing
  - Integration is hard and garret helps.
- Its not clear yet how to use it to do topic branches, or how to deal with multiple product / customer branches.
- Hacking the manifest file is painful
- I like it enough, but its not for everything.

When repo / garret fall over:

- Merge conflicts are a pain
- It will stall out or freeze all together if not given enough resources, when 2 dozen folks are hitting it with repo syncs and uploads.
  - Give the server a lot of headroom
- Doesn't enable experimental collaboration

What Intel is focused on WRT Android

- Runs best on IA.
  - Re-use existing optimizations from Chrome
- Full featured Android BSP's for Intel UMG hardware.
- Get x86 fully represented and supported in upstream AOSP.
- Update generic_x86 AOSP toolchain
- Collaborate as possible with google.

Other observations

- Linux mindsets sometimes take a while to adjust to android.
- BSP's reaching up into the user mode stack are hard to port to android.
- Google pretty much owns it, and you need a business relationship with them if you care about branding or their marketplace access.
- Android PM is not just about wake-locks.
- Lots of new tools, and ways of doing things

Bringing up a new device, list of things to do:

- Bring up fastboot
- Harden ADB
- Bring up the home screen.
- Set up a garret or git-pool mirror, going
  - Use repo manifest -r -o tag-date.xml as a way to specify a snapshot.
- Automate your acceptance testing!
  - Its too easy to break a build or regress the system with multiple teams “helping”
  - Last week was difficult in that way.

Odds and ends

- Really look at all the stuff up on android.git.kernel.org There are some interesting things there
  - Kernels, toolchains, tools, and of course Android.
- Make file phony targets of note
  - Showcommands, sdk
- I'm still not good at the Android.mk hacking.
- build/envsetup.sh is nice.
  - Mm is sometimes really handy.