SONY

Android Systems Programming Tips and Tricks

Tim Bird

Sony Network Entertainment, Inc

< tim.bird (at) am.sony.com >

SONY

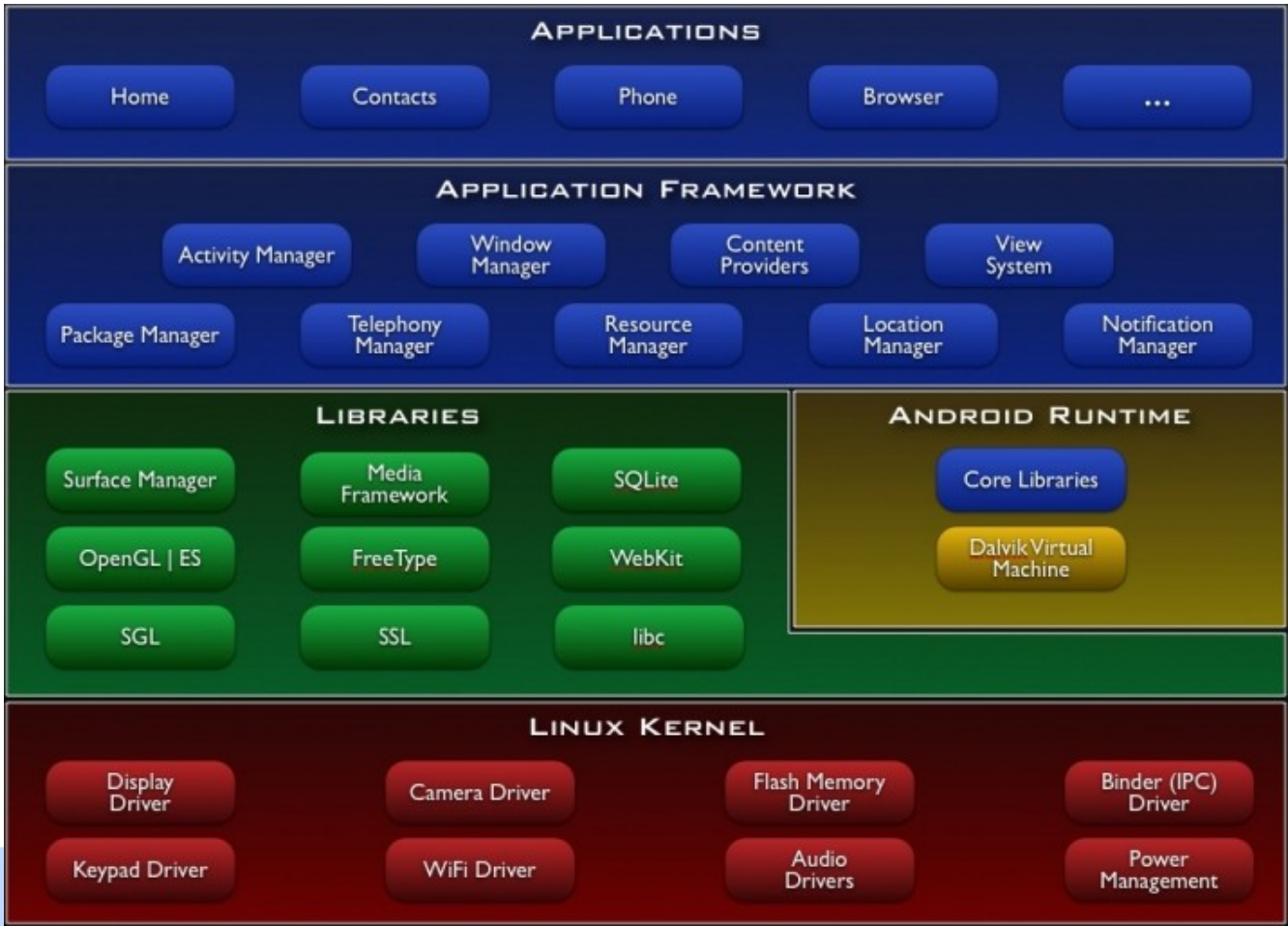
Overview

- Intro to Android
- Working with source
- Interacting with the target
- Trace and debug tools
- Performance tools
- Random thoughts on Android
- Resources

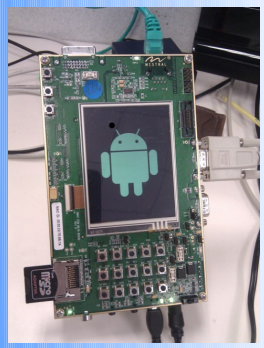
Intro to Android

- Google runtime on top of Linux

Obligatory Architecture diagram:



Android device proliferation



Working with source

SONY

Working with source

- Git
- Repo
- Build system
  - Building fast
  - Adding a program to the build

SONY

Git

- Android open source project uses 'git‘
- You need to learn to use git well,… really
  - Need to know how to do a 'git rebase‘ especially for kernel patches
  - Use ‘git rebase –I’ for interactive rebase
- Lots of online resources
  - Recommended online book: http://progit.org/book/

Repo

- ‘export REPO_TRACE=1’ is handy to see what git commands are called by repo
- Repo tricks
  - Repo forall –c ‘git diff <remote_branch>’
  - Repo forall –c ‘echo $REPO_PATH;git remote –v’
    - Use to see upstream remotes from which to compare and merge with
  - Repo manifest –r –o tag-date.xml
    - Make a repository snapshot manifest

SONY

Build System

- Lots of interesting stuff in build/envsetup.sh
  - help
  - choosecombo/lunch
  - jgrep/cgrep
  - godir
- Interesting ‘make’ targets:
  - showcommands – psuedo-target to show build commands
  - sdk – can build the SDK from scratch

Fast Building

- Parallel make threads
  - ‘make –j6’
    - Use 2 more than your number of CPUs (include hyperthreaded CPUs)
- Compiled output cache
  - ccache is in /prebuilt area
    - ‘export USE_CACCHE=1’
    - Great for rebuilds (21 minutes on my desktop)
- Make only a specific module
  - mm – build only the module(s) in the current directory (and below)
  - I usually combine this with a custom install script, which copies from out/target/product/<board>

Adding a program to the build

- Make a directory under ‘external’
  - E.g. <android>/external/myprogram
- Create your C/cpp files
- Create Android.mk as a clone of external/ping/Android.mk
  - Change the names ‘ping.c’ and ‘ping’ to match your C/cpp files and program name
- Add the directory name in <android>/build/core/main.mk after external/zlib as external/myprogram
- Make from the root of the source tree

SONY

Interacting with the target

Interacting with the target

- Android has some very nice integration engineering
- Tools discussed:
  - Fastboot
  - ADB
- Useful development configurations

SONY

Fastboot

- “fastboot” is both a tool and a bootloader protocol
- Required by Google for certified devices
- Would be really nice to adopt as an industry standard
  - e.g. maybe support fastboot in U-boot
- Fastboot operations
  - Install kernel
  - Install new flash image
  - Boot directly from host
- Very useful for test automation

SONY

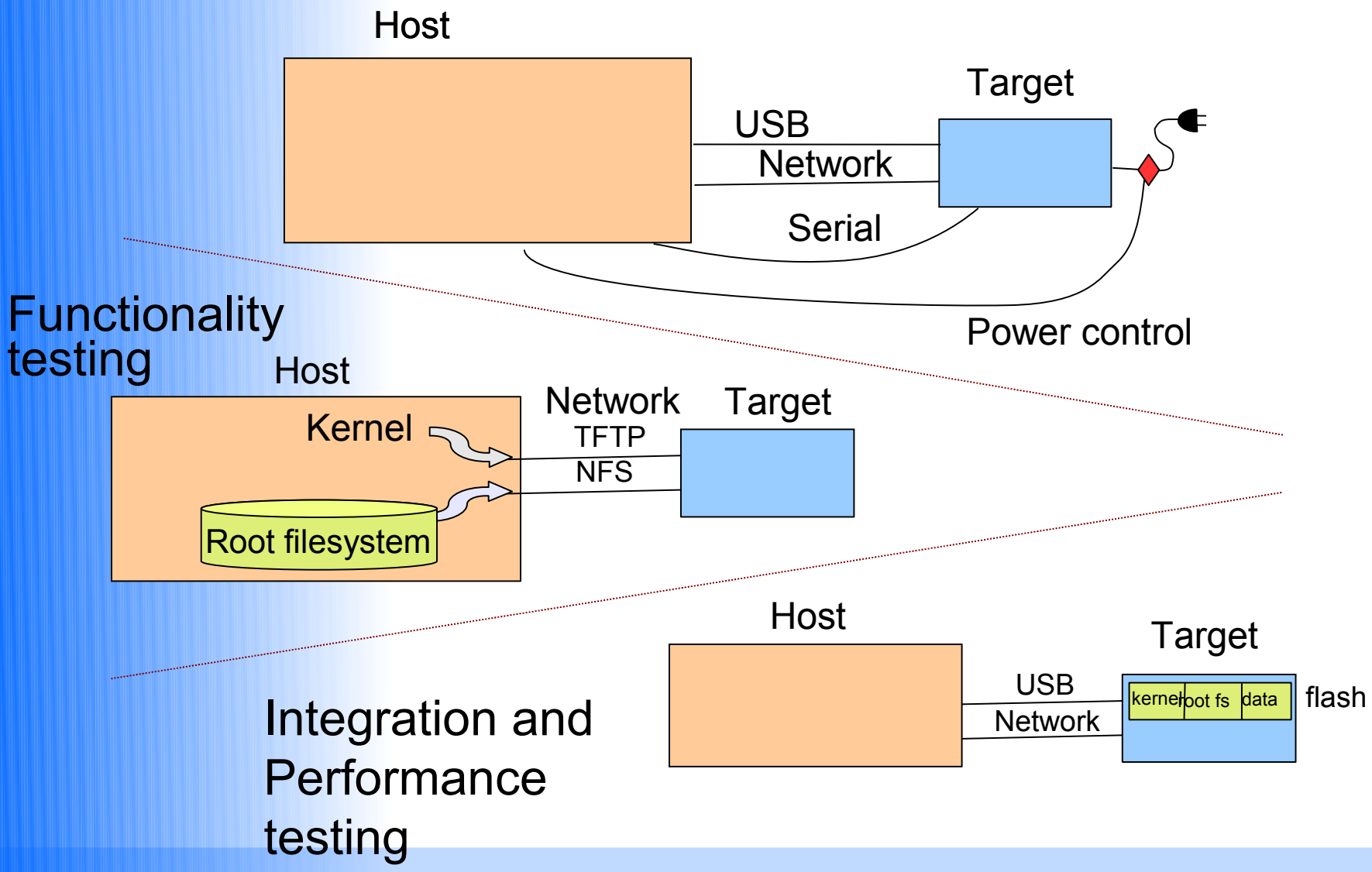
ADB

- Android Debug Bridge
- Tool for all kinds of target interactions
  (install, logging, remote shell, file copy)
  - shell [<command>]
  - push/pull
  - logcat
  - install/uninstall
- Print this and keep it under your pillow…
  - http://developer.android.com/guide/developing/tools/adb.html

ADB (cont.)

- Can work over network, instead of USB
  - Useful if you run build inside virtual machine on host
    - e.g. I build on Ubuntu 8.04 KVM on Fedora 12 (64-bit) host
  - It's simple:
    - export ADBHOST=192.168.2.1
  - For some reason, I have to kill the server after rebooting the target
    - adb kill-server
    - Calling 'adb' will respawn the server automatically

Useful development configurations



SONY

Trace and debug tools

SONY

Trace and debug tools

- Logging
  - Kernel log (dmesg)
  - Logcat
  - Stdio redirection
- Strace
- Bootchart
- Dumpstate/dumpsys
- DDMS
- Gdb

SONY

Kernel log

- It's there, use dmesg to access after boot
- Turn on PRINTK_TIMES for timestamps
- Increase buffer size: CONFIG_LOG_BUF_SHIFT
- Can add message to log from user space by writing to /dev/kmsg
  - Very handy to synchronize with kernel messages

SONY

Logcat

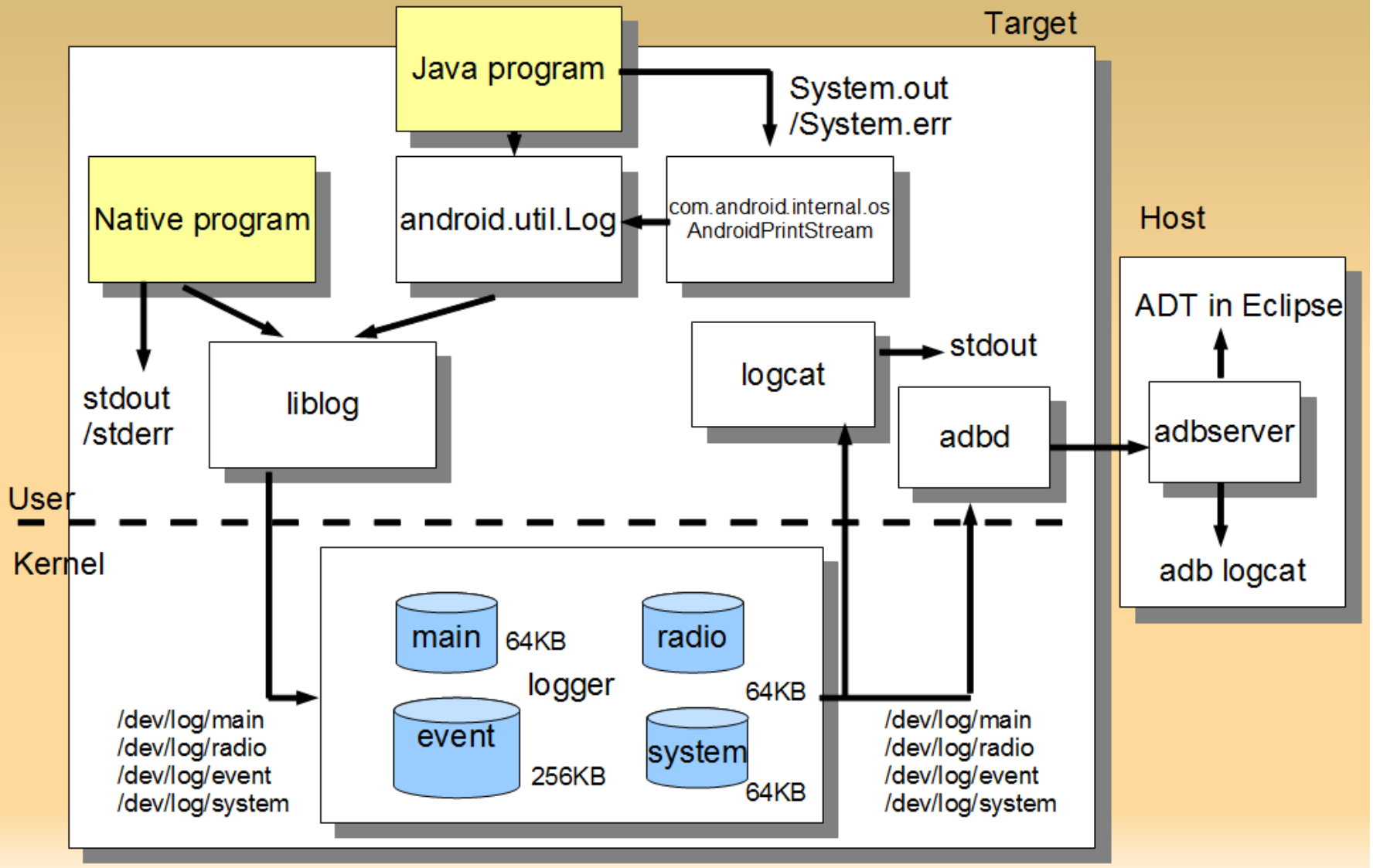
- Logging system in kernel
  - Integrated throughout Android system (C+ and Java access)
- Can Increase logging levels with setprop
  - Flags to control logging level in code
  - (DEBUG emits more??)
- Different logs (main, event, etc.)
  - Event log buffer is funky, is encoded for size
  - See jamboree presentation on log info
    - http://blog.kmckk.com/archives/2936958.html

    (Presentation by Tetsuyuki Kobayashi)

Logcat

- Use from host to redirect to a file
- To get main log info, use:
  - e.g. adb logcat –v time –d *:V >test.log
- To get info from 'events' log, use -b:
  - e.g. adb logcat –b events –v time –d | grep boot
- Filter using <tag>:<loglevel>
  - Can use ANDROID_LOG_TAGS environment variable.
- I wrote my own logdelta tool, to see time between events
  - See http://elinux.org/Improving_Android_Boot_Time#logdelta

Overview of Android Logging System



*Shameless ripoff of Tesuyuki Kobayashi

Logcat output (events)

```
I/boot_progress_start(  754): 12559
I/boot_progress_preload_start(  754): 17879
I/boot_progress_preload_end(  754): 28546
I/boot_progress_system_run(  768): 29230
I/boot_progress_pms_start(  768): 29697
I/boot_progress_pms_system_scan_start(  768): 30117
I/boot_progress_pms_data_scan_start(  768): 44171
I/boot_progress_pms_scan_end(  768): 50006
I/boot_progress_pms_ready(  768): 50505
I/boot_progress_ams_ready(  768): 53166
I/boot_progress_enable_screen(  768): 56793
```

Stdio redirection

- You can send stdout and stderr to the log:
- Redirecting Dalvik output:

```
# stop
# setprop log.redirect-stdio true
# start
```

- Redirecting C/cpp output:
  - myprogram | xargs log
    - Assumes you have busybox xargs installed

Strace

- Shows system calls for a process (or set of processes)
- Is part of AOSP since eclair
- Can add to init.rc to trace initialization.
  - For example, to trace zygote startup, in /init.rc change:

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
```

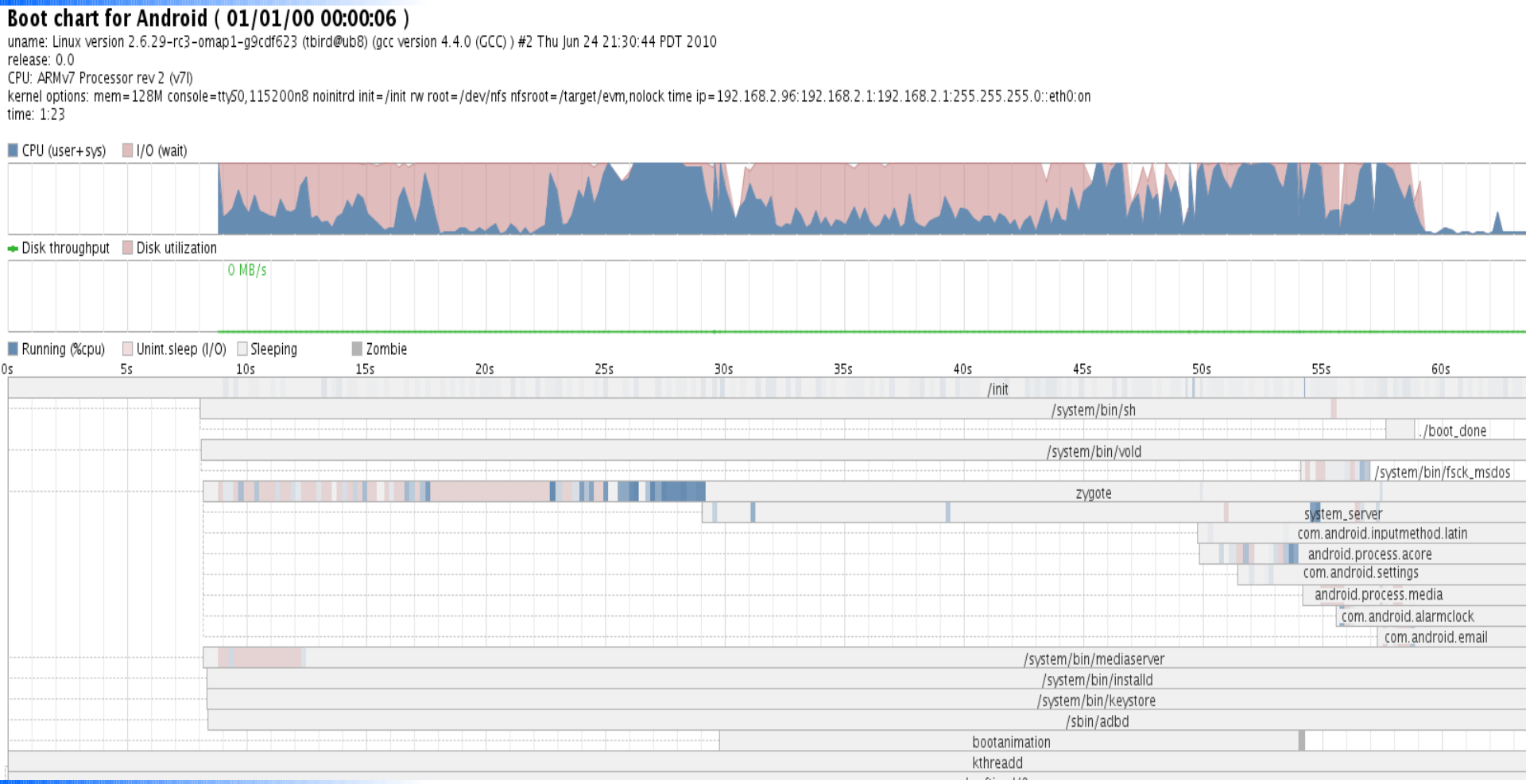
to

```
service zygote /system/xbin/strace -tt -o/data/boot.strace /system/bin/app_process -Xzygote
    /system/bin --zygote --start-system-server
```

Bootchart

- 'init' gathers data on startup
  - Must re-compile 'init' with support for bootchart data collection
- A tool on the host produces a nice graphic
- See http://elinux.org/Bootchart and http://elinux.org/Using_Bootchart_on_Android

Bootchart output



SONY

Dumpstate/dumpsys

- Dumps huge amounts of information about the system, including status, counts and statistics
- Dumpstate reproduces lots of stuff from /proc
  - Does a dumpsys as well
- Dumpsys show status information from Android services
  - e.g. dumpsys alarm
- First part of dump has list of services you can dump

DDMS

- Dalvik Debug Monitor Service
  - http://developer.android.com/guide/developing/tools/ddms.html
- Lots of features, controllable via eclipse
- To watch allocations in C/c++ code, try:
  - Set “native=true” in ~/.android/ddms.cfg
  - Use standalong ddms program
  - On target do:

```
# setprop libc.debug.malloc 1
# stop
# start
```

Gdb

- How to invoke:

1. adb forward tcp:5039 tcp:5039
2. adb shell gdbserver :5039 <exename> <arguments if any>
3. In another shell, gdbclient <exename>

Or, manually: $ arm-eabi-gdb

…

file ./out/target/product/generic/symbols/system/bin/app_process

set solib-search-path *./out/target/product/generic/symbols/system/*lib

target remote localhost:5039

- Note that gdbclient is a function in build/envsetup.sh
- Files are stripped in output dir
  - Unstripped files are at:

./out/target/product/generic/obj/EXECUTABLES/<name of module>_intermediates/LINKED/<name of the executable>

More debug tips

- See http://omappedia.org/wiki/Android_Debugging
- Tons of tips, including:
  - How to debug a native program segfault
  - How to use kernel profiler and oprofile
  - How to use gdb and DDD
- Info is for Zoom2 board, but some things should work on your board also

SONY

Performance tools

SONY

Performance Tools

- Smem
- Traceview
- 0xbench
- Perf??

Smem

- Tools for analyzing system-wide memory usage
  - Can slice, dice, and visualize memory info snapshot
- Run smemcap on target, grab data with adb, then analyze on host
- See http://elinux.org/Using_smem_on_Android

SONY

Traceview

- Shows trace of Java methods
- Also shows profile information
- User can start and stop tracing either using DDMS
- App can start and stop tracing programmatically
- Google: “android traceview”

0xbench

- Has several built-in benchmarks, such as Linpack, Scimark2, and LibMicro
- Project page at: http://code.google.com/p/0xbench
- Is available in Android Market
- Some tests require root privileges

Perf

- Standard kernel tool for performance analysis
- Now that Android is up to 2.6.35 kernel, should be a breeze to use
  - Have to admit I haven't done it yet – I'm stuck on 2.6.29
    - Anyone here done it?

SONY

Miscellaneous tools

- procrank
- setprop/getprop
- sqlite (command line)
- start/stop
  - Can stop/start whole system

Procrank

- Shows a quick summary of processes, sorted by VSS, RSS, PSS or USS
  - See http://elinux.org/Android_Memory_Usage
- Output:

```
# procrank
  PID      Vss      Rss      Pss      Uss  cmdline
 1217   36848K   35648K   17983K   13956K  system_server
 1276   32200K   32200K   14048K   10116K  android.process.acore
 1189   26920K   26920K    9293K    5500K  zygote
 1321   20328K   20328K    4743K    2344K  android.process.media
 1356   20360K   20360K    4621K    2148K  com.android.email
 1303   20184K   20184K    4381K    1724K  com.android.settings
 1271   19888K   19888K    4297K    1764K  com.android.inputmethod.latin
 1332   19560K   19560K    3993K    1620K  com.android.alarmclock
 1187    5068K    5068K    2119K    1476K  /system/bin/mediaserver
 1384     436K     436K     248K     236K  procrank
    1     212K     212K     200K     200K  /init
  753     572K     572K     171K     136K  /system/bin/rild
  748     340K     340K     163K     152K  /system/bin/sh
  751     388K     388K     156K     140K  /system/bin/vold
 1215     148K     148K     136K     136K  /sbin/adbd
  757     352K     352K     117K      92K  /system/bin/dbus-daemon
  760     404K     404K     104K      80K  /system/bin/keystore
  759     312K     312K     102K      88K  /system/bin/installd
  749     288K     288K      96K      84K  /system/bin/servicemanager
  752     244K     244K      71K      60K  /system/bin/debuggerd
```

setprop/getprop

- Many services have debug elements controlled by properties
- Many properties are set in /init.rc
- You can also query and set properties on the command line
  - Use 'getprop' (with no args) to see list of properties
- Have to examine source for properties with special meanings (or see something on a mailing list)
  - Example: setting the DNS server address manually:
    - setprop net.nds1 xx.yy.zz.aa

Sqlite

- You can inspect and modify sqlite data directly from the command line
  - Here's an example of setting the http_proxy for a development board

```
# cd /data/data/com.android.providers.settings/databases
# sqlite3 settings.db
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> insert into system values(99,'http_proxy','192.168.1.1:80');
sqlite>.exit
#
```

- Most databases are under a directory called 'databases', and end in '.db'

SONY

Wrapup

SONY

Random thoughts on Android

- Throws POSIX out the window
  - Hurray!... Darn...
- Lots of talk about Android fragmentation
  - Fragmentation doesn't matter for custom programming work
    - If Android works for you, then use it
  - Soon, vendors will have to ensure compatibility, rather than app makers
- Seems destined to be a major embedded Linux platform
  - Only drawback(?) is non-native apps
    - But even this has pros and cons

Resources

- eLinux wiki Android portal:
  - http://elinux.org/Android_Portal
- Use android-porting, android-platform, and android-kernel mailing lists, depending on where your issue is
  - See http://elinux.org/Android_Web_Resources#Mailing_Lists
- My e-mail: tim.bird (at) am.sony.com

SONY

Thanks for your time

Questions and Answers