

Accelerate your Mobile Apps and Games for Android™ on ARM

Matthew Du Puy
Software Engineer, ARM

Presenter

Matthew Du Puy

Software Engineer, ARM

Matthew Du Puy is a software engineer at ARM and is currently working to ensuring mobile app performance on the latest ARM technologies. Previously a self employed embedded systems software contractor working primarily on the Linux Kernel and a mountain climber.



Contact Details:

Email: matthew.dupuy@arm.com

Problem: This is not a desktop

- Mobile apps require special design considerations that aren't always clear and tools to solve increasingly complex systems are limited
 - Animations and games drop frames
 - Networking, display, real time audio and video processing eat battery
 - App won't fit in memory constraints



Analysis

- Fortunately Google, ARM and many others are developing analysis tools and solutions to these problems
- Is my app ... ?
 - CPU/GPGPU bound
 - I/O or memory constrained
 - Power efficient
- What can I do to fix it?
(short of buying everyone who runs my app a Quad-core ARM® Cortex™-A15 processor & ARM Mali™-T604 processor or Octo phone)



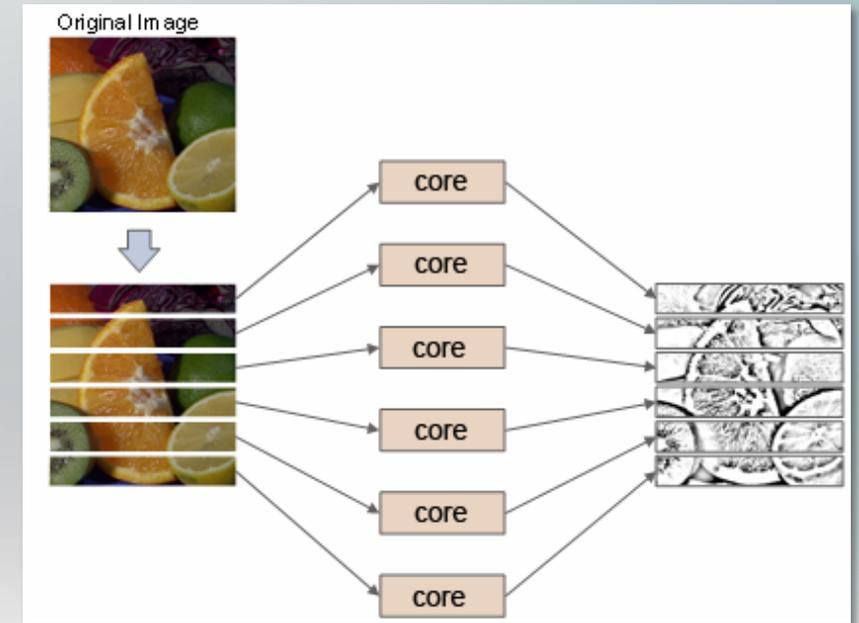
Analysis of Java SDK Android Apps

- Static analysis with SDK Lint tool
- Dynamic analysis with DDMS
 - Allocation/heap
 - Process and thread utilization
 - Traceview (method)
 - Network
- Hierarchy Viewer
- Systrace



But ask yourself these questions

- Is this performance bottleneck parallelizable?
- Is this Java or Native?
Would it be better the other way around?
- Has this been done before? Don't reinvent the wheel.
- Am I being smart with resources?
- What version of Android should I target?



Starting EASY

Static analysis: LINT

Description	Category	Location	
⊗ Obsolete ProGuard file; use -keepclasseswithmembers instead of -keep	Correctness	proguard	Avoid object allocations during draw/layout operations (preallocate and reuse instead)
▷ ⊗ "backup_api_key" is not translated in de, zh-rCN, zh-rTW (21 items)	Correctness:Message	api_key:	
▷ ⊗ Call requires API level 11 (current min is 3): android.app.Activity#ge	Correctness	ActionB	Issue: Looks for memory allocations within drawing code
▽ ⚠️ Avoid object allocations during draw/layout operations (preallocate	Performance	SmallKe	Id: DrawAllocation
⚠️ Avoid object allocations during draw/layout operations (prealloca	Performance	SmallKe	
🐞 This TableLayout should use android:layout_height="wrap_content"	Correctness	chooser.	You should avoid allocating objects during a drawing or layout operation. These are called frequently, so a smooth UI can be interrupted by garbage collection pauses caused by the object allocations.
▷ ⚠️ SharedPreferences.edit() without a corresponding commit() or apply	Correctness	GameCl	
🐞 This tag and its children can be replaced by one <TextView/> and a	Performance	list_item	
▽ ⚠️ Avoid hardcoding the debug mode; leaving it out allows debug and	Security	AndroidI	
⚠️ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidI	The way this is generally handled is to allocate the needed objects up front and to reuse them for each drawing operation.
⚠️ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidI	
⚠️ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidI	
▷ 🐞 Replace "... " with ellipsis character (... , …) ? (32 items)	Usability:Typography	strings.>	
⚠️ The image icon.png varies significantly in its density-independent (Usability:Icons	icon.png	Some methods allocate memory on your behalf (such as Bitmap.create), and these should be handled in the same way.
▷ 🐞 [I18N] Hardcoded string "Push me", should use @string resource (4	Internationalization	main.xr	
⚠️ This Handler class should be static or leaks might occur (name.boyl	Performance	SGTPuzz	
⚠️ Use new SparseIntArray(...) instead for better performance	Performance	SoundM	

Static analysis: LINT

Description	Category	Location
⊗ Obsolete ProGuard file; use -keepclasseswithmembers instead of -keep	Correctness	proguard.cfg:21 (com.glTron.glTron)
⊗ "backup_api_key" is not translated in de, zh-rCN, zh-rTW (21 items)	Correctness:Message	api_keys.xml:9 in values (SGTPuzzles)
⊗ Call requires API level 11 (current min is 3): android.app.Activity#getString	Correctness	ActionBarHoneycomb.java:22 in ActionBarHoneycomb
⚠ Avoid object allocations during draw/layout operations (preallocate)	Performance	SmallKeyboard.java:397 in sgtpuzzles
⚠ Avoid object allocations during draw/layout operations (preallocate)	Performance	SmallKeyboard.java:397 in sgtpuzzles
⚠ This TableLayout should use android:layout_height="wrap_content"	Correctness	chooser.xml:8 in layout (SGTPuzzles)
⚠ SharedPreferences.edit() without a corresponding commit() or apply	Correctness	GameChooser.java:54 in sgtpuzzles
⚠ This tag and its children can be replaced by one <TextView/> and a	Performance	list_item.xml:3 in layout (SGTPuzzles)
⚠ Avoid hardcoding the debug mode; leaving it out allows debug and	Security	AndroidManifest.xml:8 (Hello Whirled)
⚠ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidManifest.xml:6 (Hello Whirled)
⚠ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidManifest.xml:30 (Hello Whirled)
⚠ Avoid hardcoding the debug mode; leaving it out allows debug ar	Security	AndroidManifest.xml:20 (SGTPuzzles)
⚠ Replace "..." with ellipsis character (...), … ? (32 items)	Usability:Typography	strings.xml:22 in values-zh-rCN
⚠ The image icon.png varies significantly in its density-independent (Usability:Icons	icon.png in drawable-ldpi (com.glTron.glTron)
⚠ [I18N] Hardcoded string "Push me", should use @string resource (4	Internationalization	main.xml:12 in layout (Hello Whirled)
⚠ This Handler class should be static or leaks might occur (name.boyl	Performance	SGTPuzzles.java:117 in sgtpuzzles
⚠ Use new SparseIntArray(...) instead for better performance	Performance	SoundManager.java:71 in SoundManager
⚠ Missing the following drawables in drawable-xhdpi: gltron_bitmap.p	Usability:Icons	Hello Whirled
⚠ Use android.util.FloatMath#cos() instead of java.lang.Math#cos to	Performance	Camera.java:232 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#sin() instead of java.lang.Math#sin t	Performance	Camera.java:232 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#sin() instead of java.lang.Math#sin t	Performance	Camera.java:233 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#sin() instead of java.lang.Math#sin t	Performance	Camera.java:233 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#cos() instead of java.lang.Math#cos	Performance	Camera.java:234 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#sin() instead of java.lang.Math#sin t	Performance	Player.java:384 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#cos() instead of java.lang.Math#cos	Performance	Player.java:432 in Game (com.glTron.glTron)
⚠ Use android.util.FloatMath#sqrt() instead of java.lang.Math#sqrt	Performance	TrailMesh.java:104 in Video (com.glTron.glTron)

Use android.util.FloatMath#cos() instead of java.lang.Math#cos to avoid argument float to double conversion

Issue: Suggests replacing java.lang.Math calls with android.util.FloatMath to avoid conversions
Id: FloatMath

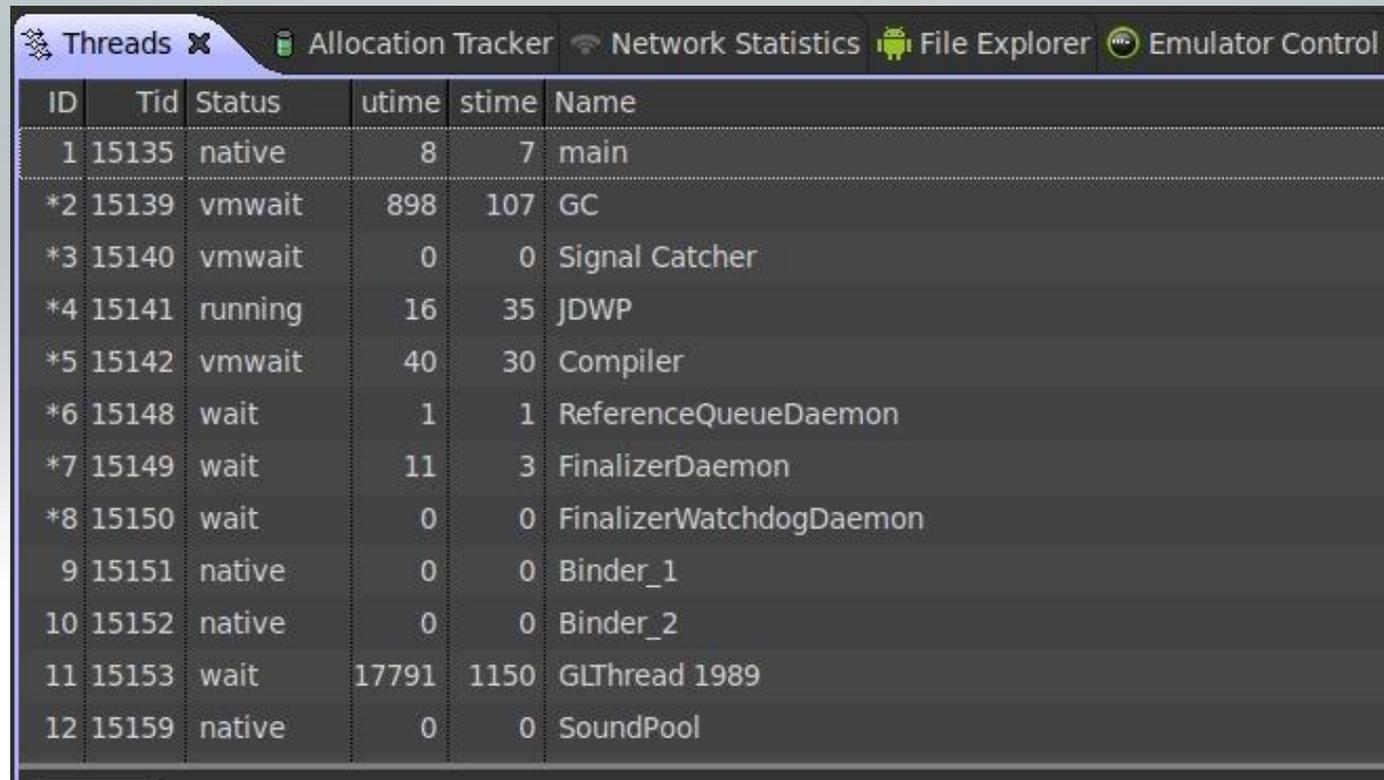
On modern hardware, "double" is just as fast as "float" though of course it takes more memory. However, if you are using floats and you need to compute the sine, cosine or square root, then it is better to use the android.util.FloatMath class instead of java.lang.Math since you can call methods written to operate on floats, so you avoid conversions back and forth to double.

<http://developer.android.com/guide/practices/design/performance.html#avoidfloat>

Beyond static analysis

Dalvik Debug Monitor Server (DDMS)

- DDMS Thread analysis (like “top” but better)

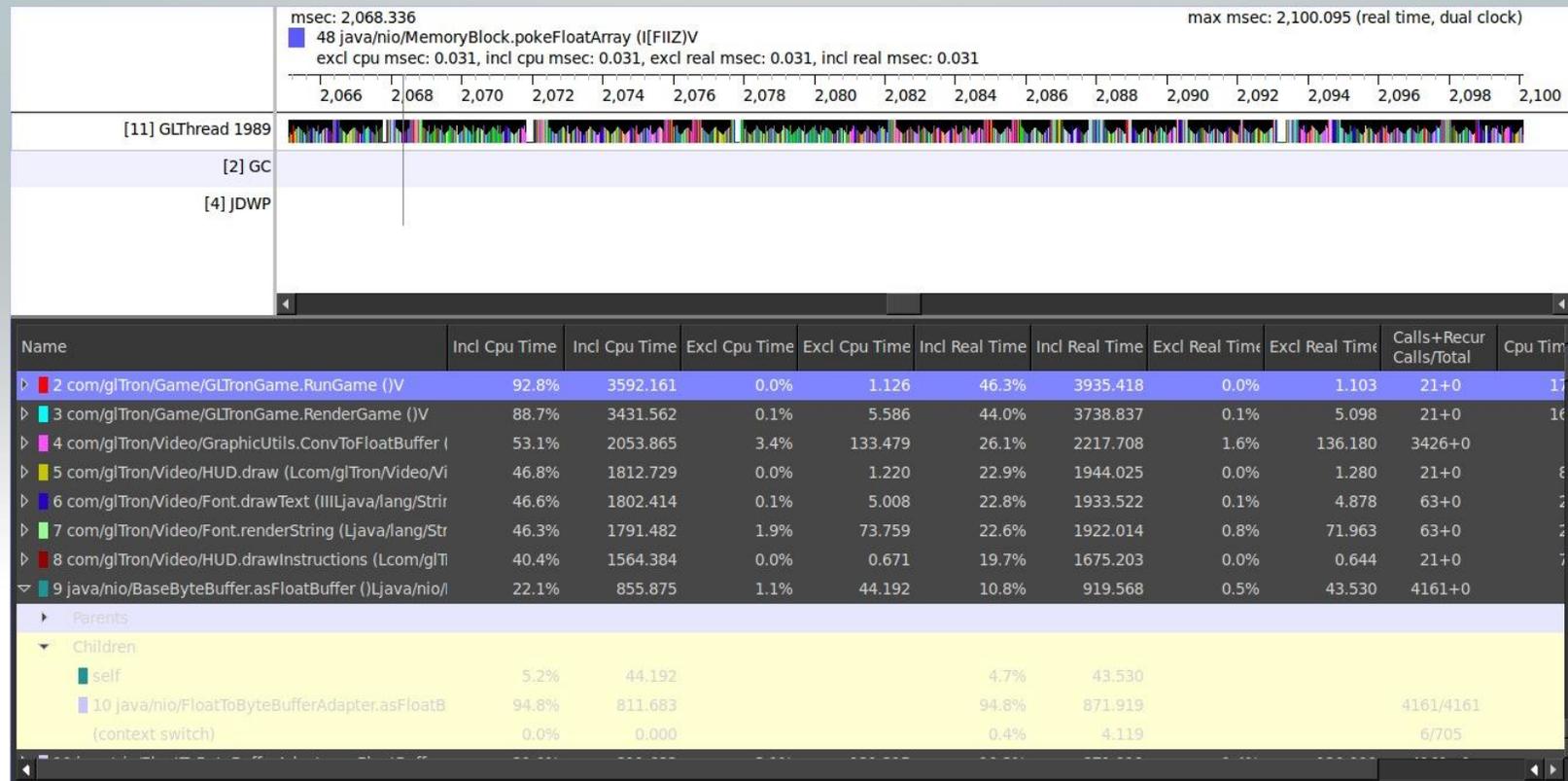


ID	Tid	Status	utime	stime	Name
1	15135	native	8	7	main
*2	15139	vmwait	898	107	GC
*3	15140	vmwait	0	0	Signal Catcher
*4	15141	running	16	35	JDWP
*5	15142	vmwait	40	30	Compiler
*6	15148	wait	1	1	ReferenceQueueDaemon
*7	15149	wait	11	3	FinalizerDaemon
*8	15150	wait	0	0	FinalizerWatchdogDaemon
9	15151	native	0	0	Binder_1
10	15152	native	0	0	Binder_2
11	15153	wait	17791	1150	GLThread 1989
12	15159	native	0	0	SoundPool

DDMS:Traceview

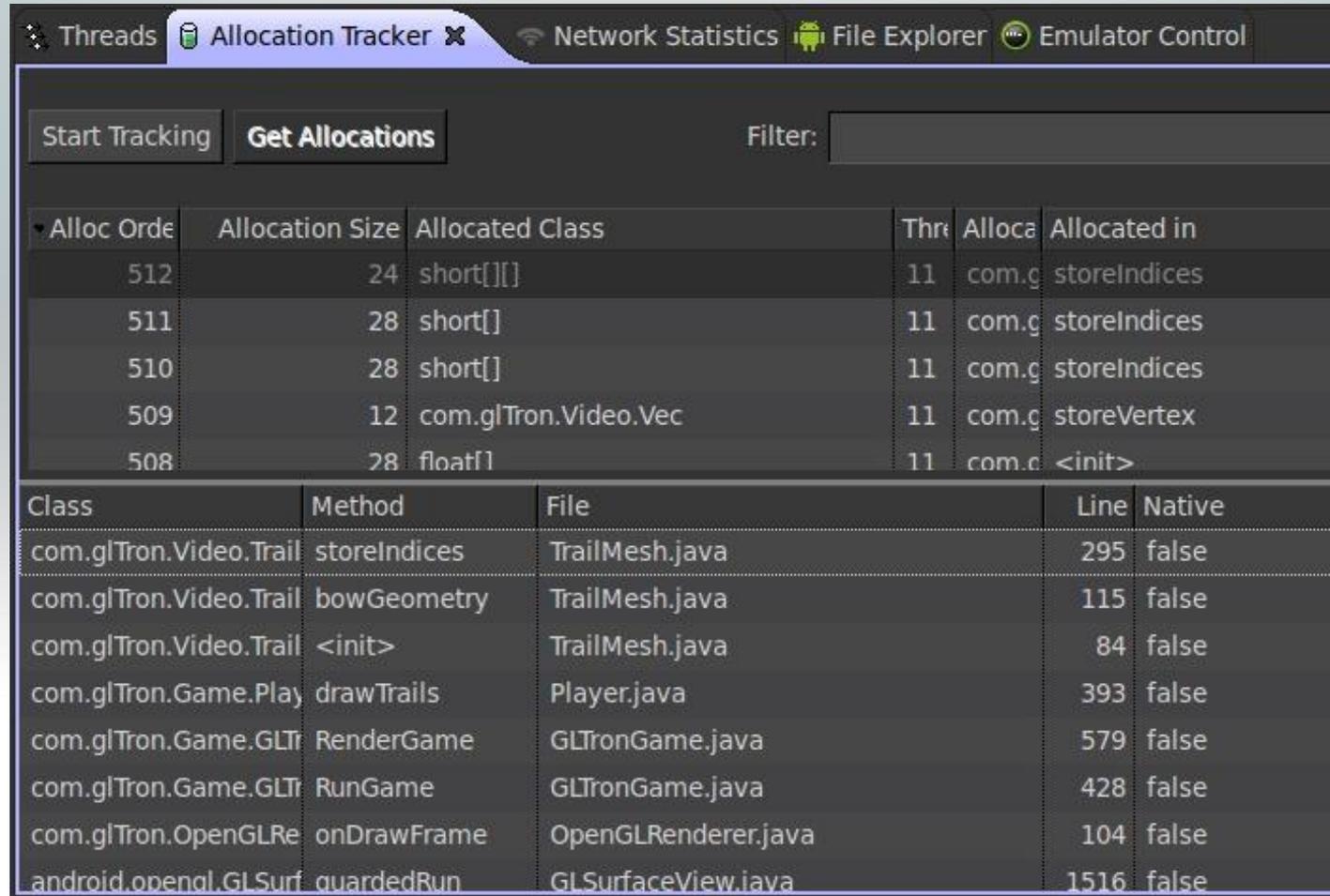
How much CPU time is each method consuming?

- Traceview (start method profiling button)



Allocations and HEAP

are you allocating in a high frequency method?



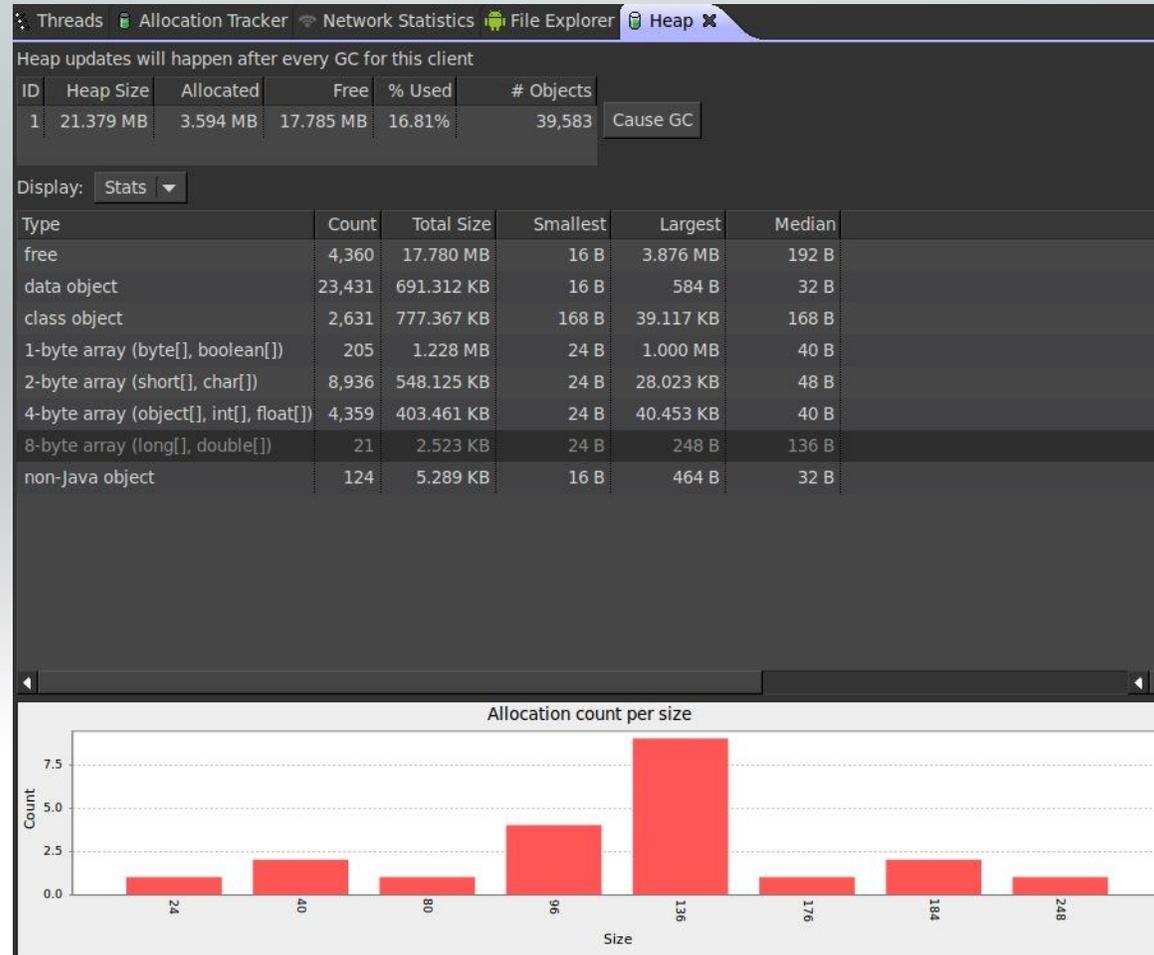
The screenshot shows the Allocation Tracker interface in Android Studio. It includes a toolbar with 'Start Tracking' and 'Get Allocations' buttons, and a 'Filter:' input field. The main area contains two tables. The top table lists individual allocations with columns for 'Alloc Order', 'Allocation Size', 'Allocated Class', 'Thre', 'Alloca', and 'Allocated in'. The bottom table lists the methods responsible for these allocations with columns for 'Class', 'Method', 'File', 'Line', and 'Native'.

Alloc Order	Allocation Size	Allocated Class	Thre	Alloca	Allocated in
512	24	short[][]	11	com.g	storeIndices
511	28	short[]	11	com.g	storeIndices
510	28	short[]	11	com.g	storeIndices
509	12	com.glTron.Video.Vec	11	com.g	storeVertex
508	28	float[]	11	com.c	<init>

Class	Method	File	Line	Native
com.glTron.Video.Trail	storeIndices	TrailMesh.java	295	false
com.glTron.Video.Trail	bowGeometry	TrailMesh.java	115	false
com.glTron.Video.Trail	<init>	TrailMesh.java	84	false
com.glTron.Game.Play	drawTrails	Player.java	393	false
com.glTron.Game.GLTr	RenderGame	GLTronGame.java	579	false
com.glTron.Game.GLTr	RunGame	GLTronGame.java	428	false
com.glTron.OpenGLRe	onDrawFrame	OpenGLRenderer.java	104	false
android.opengl.GLSurf	guardedRun	GLSurfaceView.java	1516	false

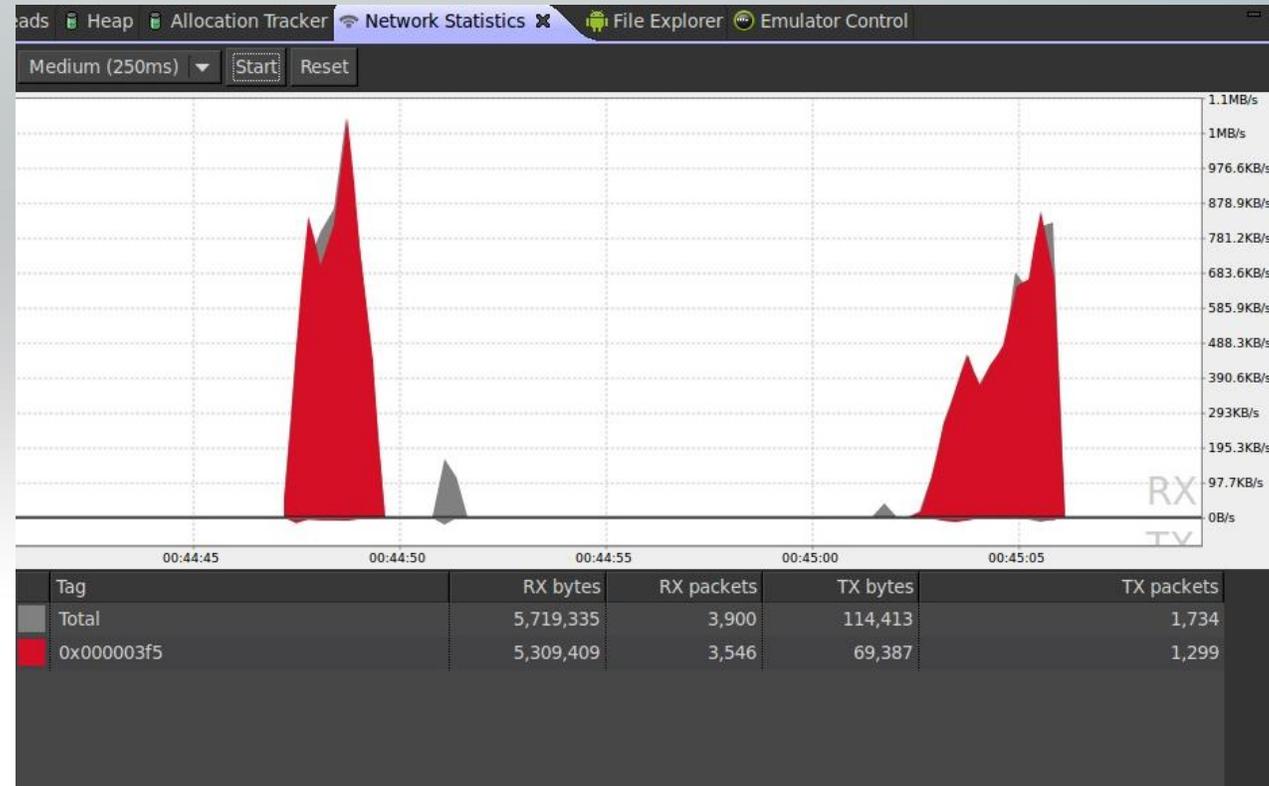
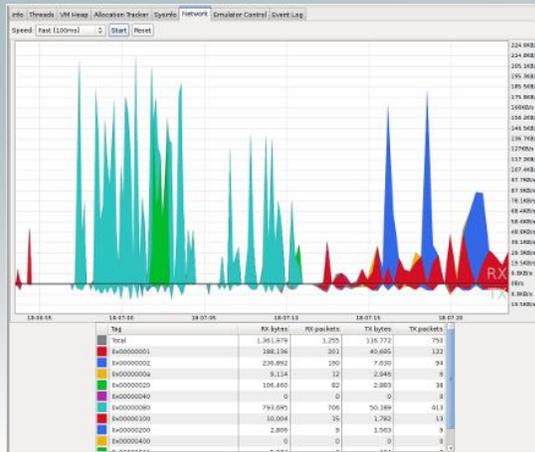
HEAP:

Is your app running out of memory?

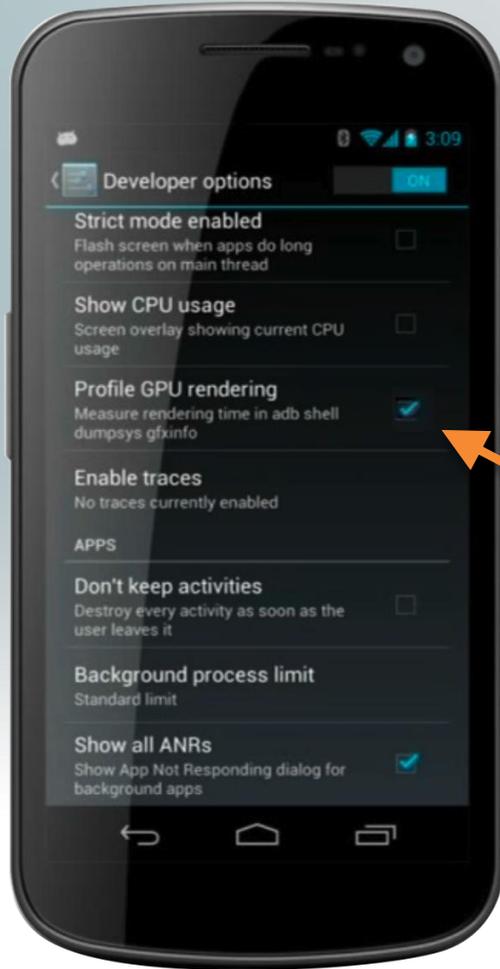


Network Statistics

- Save battery, look for short spikes that can be delayed
- TrafficStats API allows you to tag individual sockets



Adb shell DUMPsys

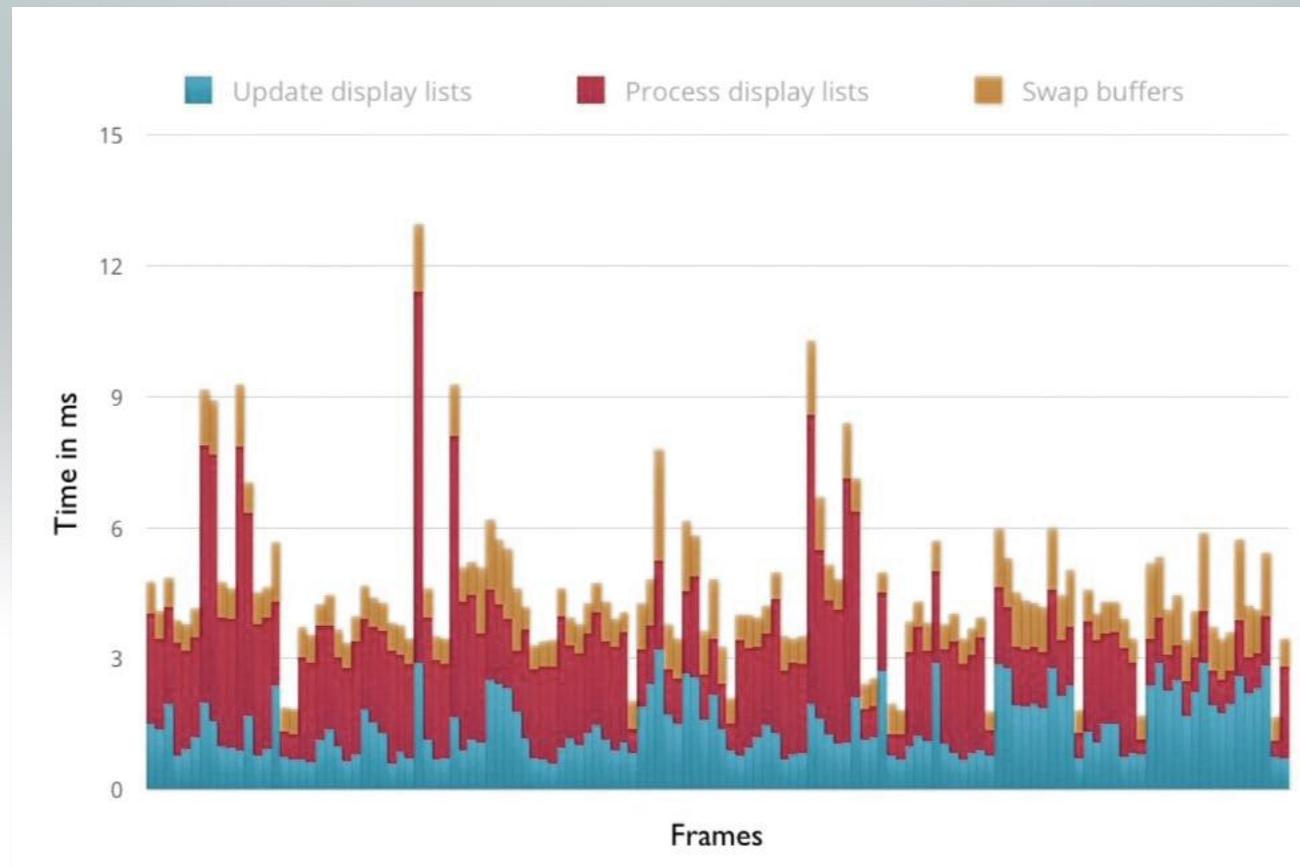


- With dumsys you can check:
 - Event Hub State
 - Input Reader State
 - Input Dispatcher State
 - any number of other systems
e.g. dumsys gfxinfo

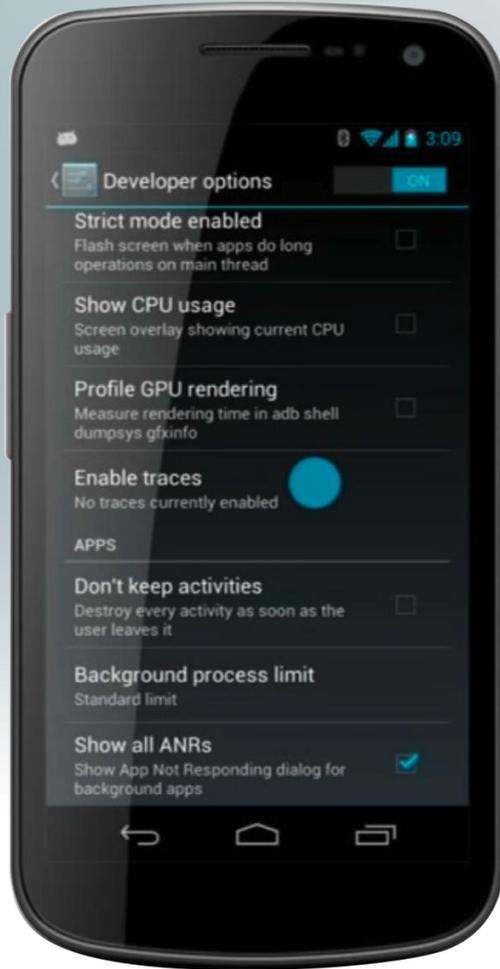
```
Profile data in ms:
com.android.launcher/com.android.launcher2.Launcher/android.view.ViewRootIm
pl@4161f120
Draw Process Execute
0.07 0.60 0.92
0.07 0.68 0.96
0.10 0.96 1.34
0.10 0.94 1.20
0.09 0.94 1.19
0.10 0.83 1.28
0.06 0.83 1.21
0.06 0.61 0.73
0.05 0.55 0.67
```

Dumpsys gfxinfo

- Drop dumpsys data columns in to a spreadsheet and visualize...
e.g. Will my animation drop frames?



Systrace



- I've done all I can to analyze inside my app but still can't find the bottleneck.

Systrace to the rescue!

- Systrace.py will generate a 5 second system level snapshot

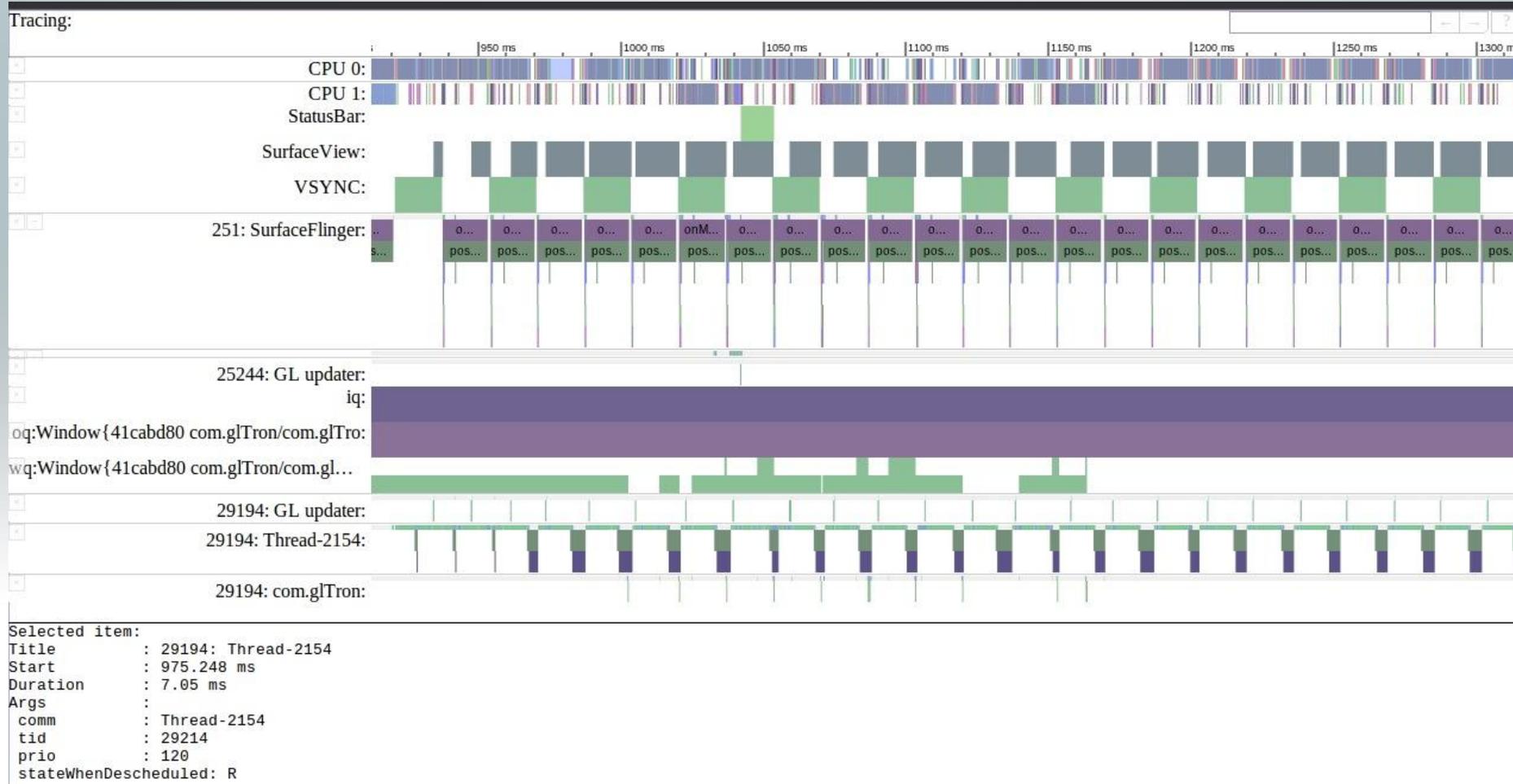


```
android@android ~/android-sdk-linux/tools/systrace $ ./systrace.py
capturing trace... done
downloading trace... done

wrote file:///home/android/android-sdk-linux/tools/systrace/trace.html
android@android ~/android-sdk-linux/tools/systrace $ xdg-open trace.html
```

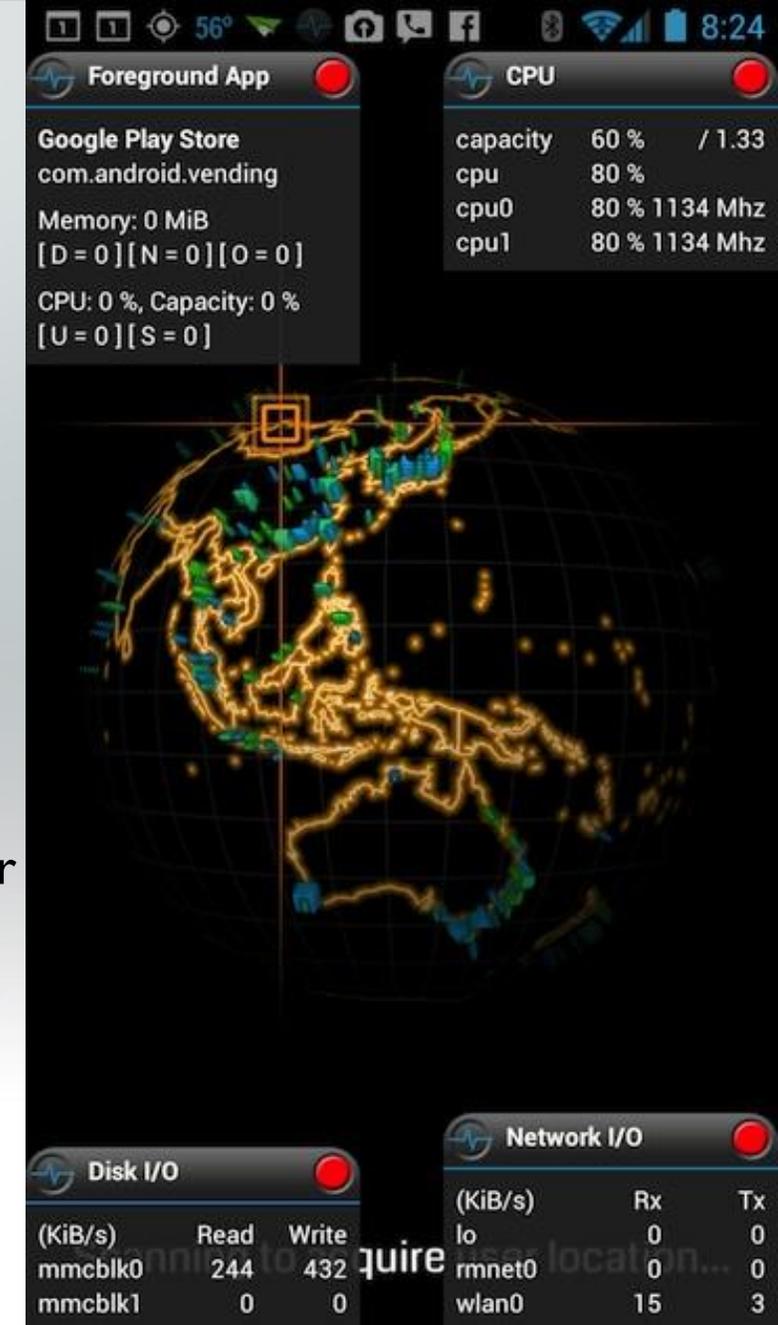
Systrace

html5 page of info: Navigate with 'w'a's'd'



Other system profilers to consider:

- Chainfire PerfMon App - Free on XDA-Developers
 - Foreground App
 - CPU
 - Disk I/O
 - Network I/O
- From Qualcomm
 - Trepn Profiler App – overlay mode similar to PerfMon but can monitor Android Intents, log states, allows external control and power monitoring.
 - Adreno SDK and Profiler for profiling the Adreno GPU

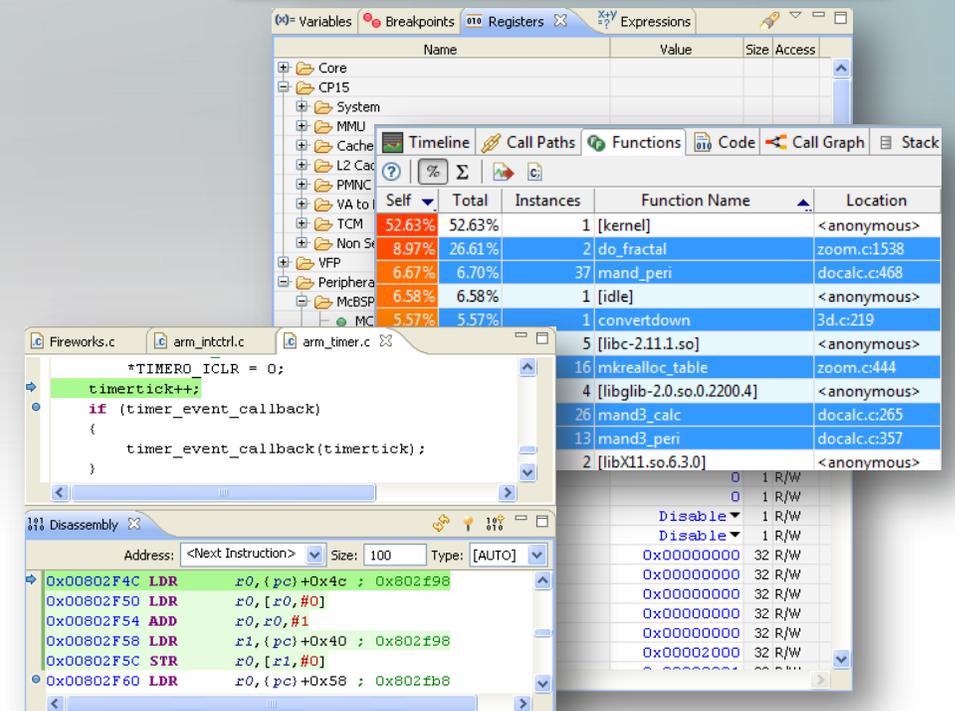
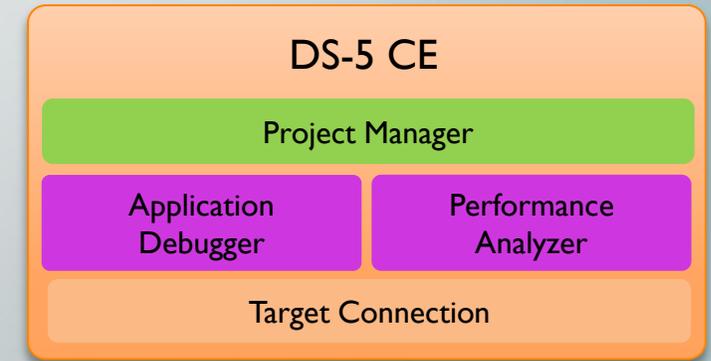


Analyzing native C/C++ (NDK)

- But I didn't use the Java SDK to write my app! How do I analyze my already wicked fast native (or iOS app objective-C port) code?
- What about the Linux kernel part of system analysis?
- Notes of caution
 - Applications that use NDK well will be faster and slicker
 - Ones that don't will be cursed by unhappy users
 - If you build .so libraries for ARM, only ARM devices will be able to run your apps
 - Fortunately not many Android Platforms that aren't ARM
 - Good use of the NDK will narrow the difference between high and low end devices
 - Moving inefficient code to Native doesn't magically make it better code

DS-5 CE for Android App Developers

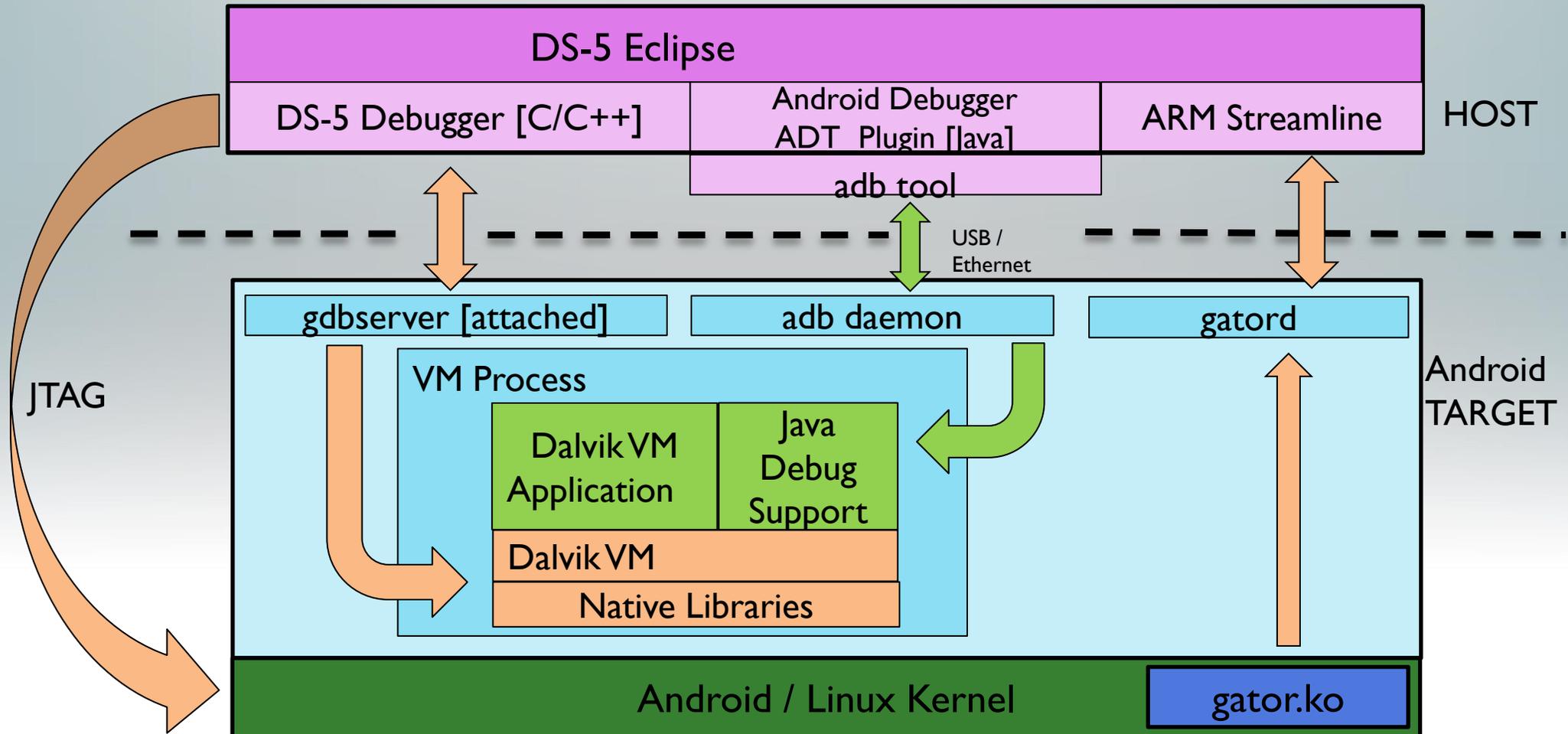
- **Friendly, Reliable App Debugger**
 - Powerful graphical user interface
 - ADB integration for native debug
 - Java* and native debug in the same IDE
- **System-wide Performance Analyzer**
 - In-depth system performance statistics
 - Process to function level profiling (native)
- **Integrated, validated solution**
 - Comprehensive documentation
 - Support via ARM forums
 - Delivered as Eclipse plug-in on arm.com
 - Free of charge



* Java debug for Android requires SDK and ADT

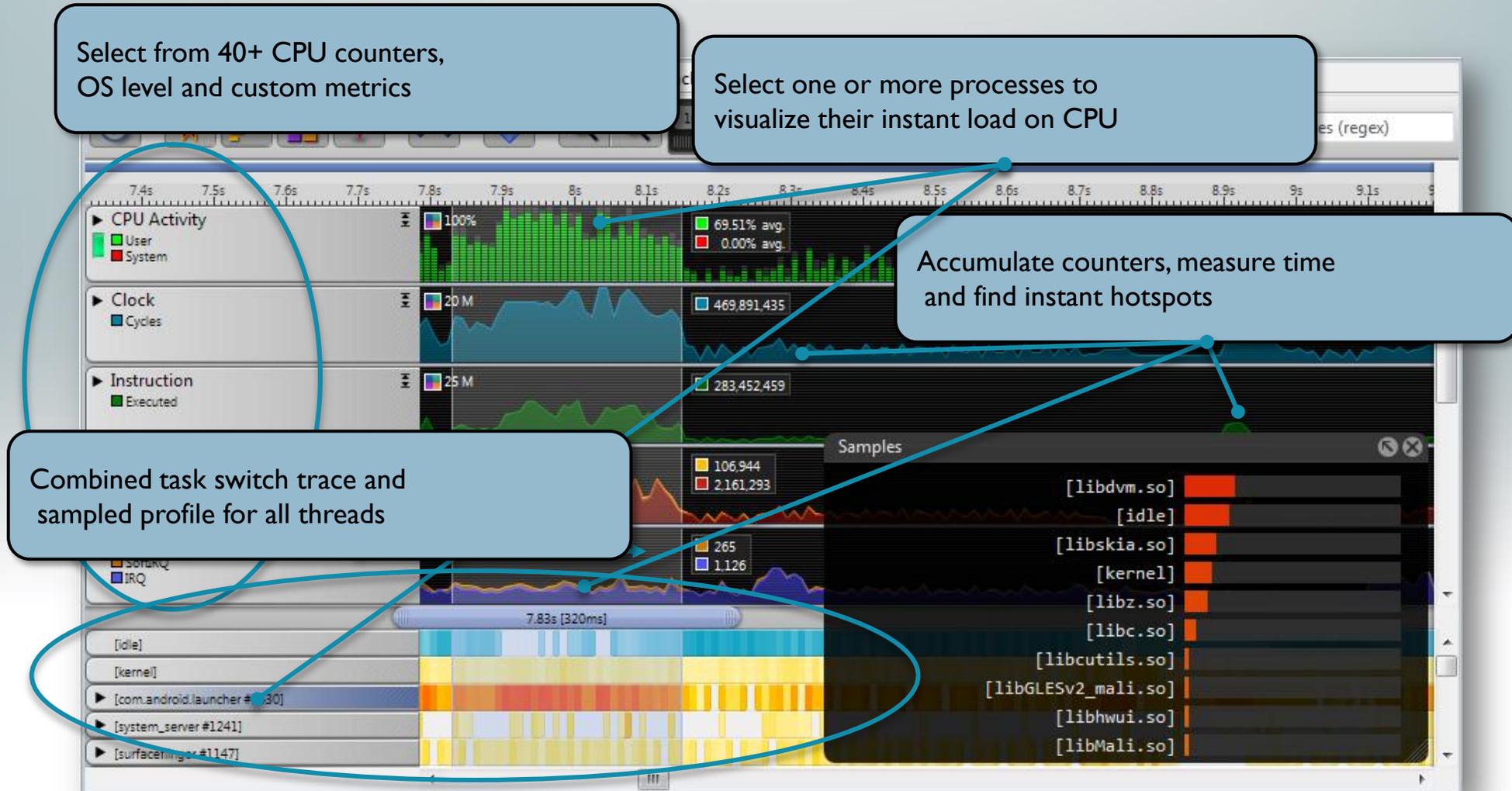
ARM DS-5™ Community Edition

Free Android Native analyzer and debugger

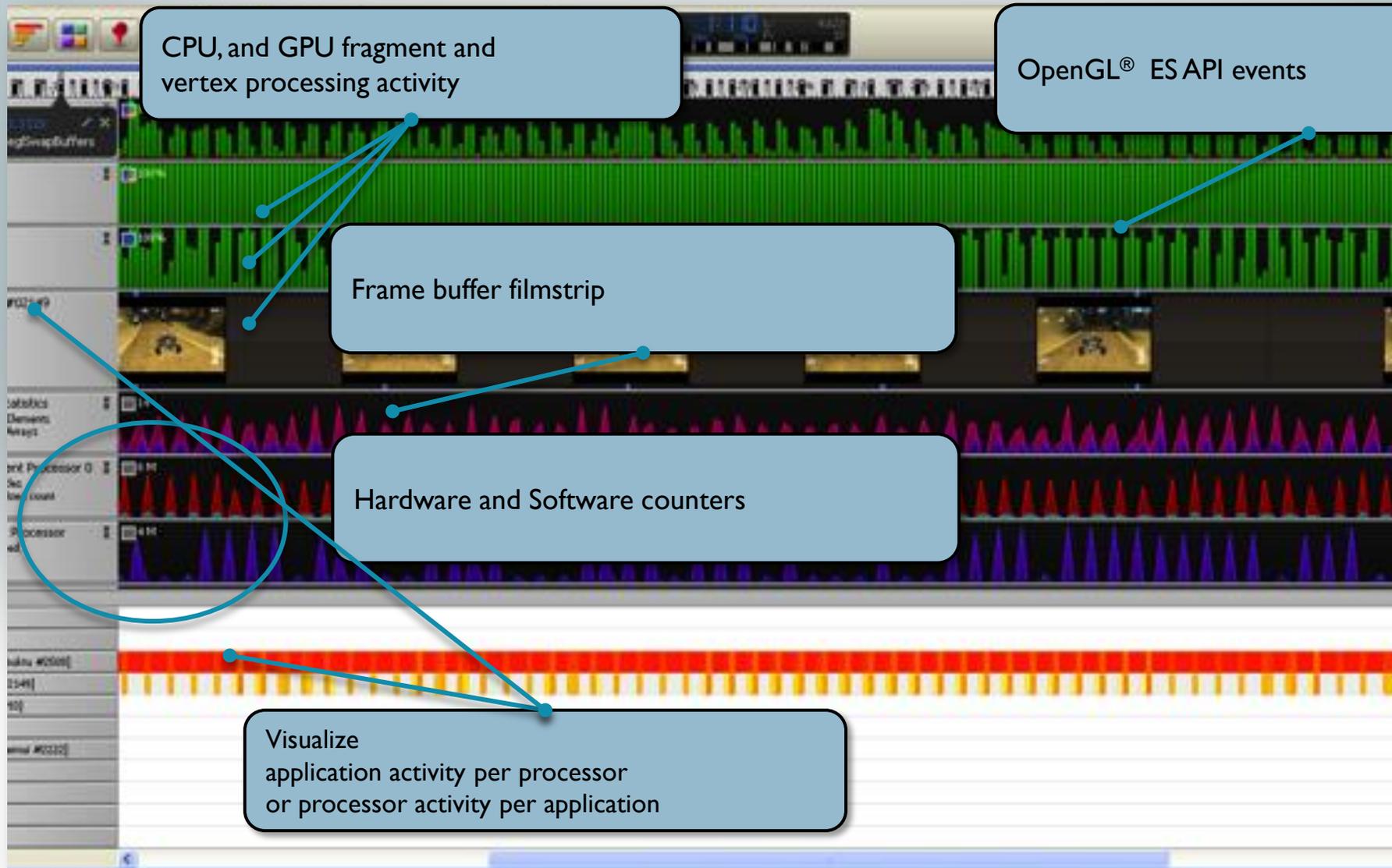


Streamline: The Big Picture

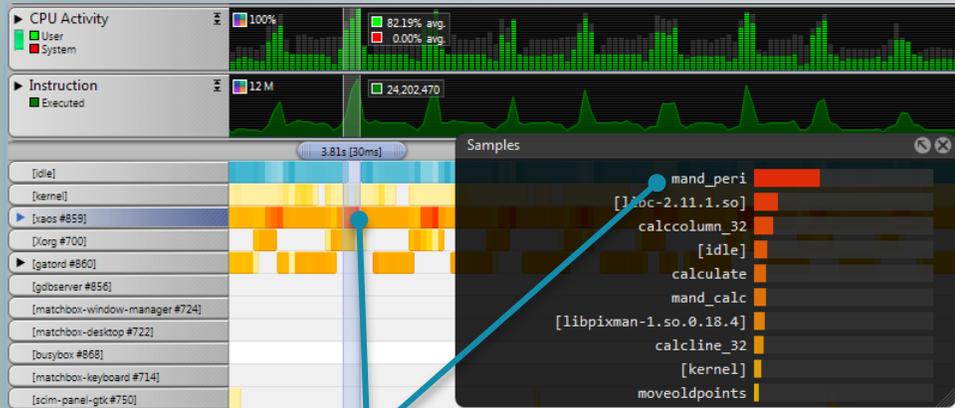
- Find hotspots, system glitches, critical conditions at a glance



Mali GPU Graphics Analysis



Drilldown Software Profiling



Filter timeline data to generate focused software profile reports

Quickly identify instant hotspots

Click on the function name to go to source code level profile

Timeline Call Paths Functions Code Call Graph Stack Log Warnings

asm % Σ f(x) The source file is newer than the binary Total: 4 (3.03%)

Self	Process	Total	Stac
0.00%	100.00%	81.06%	
0.00%	27.10%	21.97%	
0.00%	22.43%	18.18%	
0.00%	22.43%	18.18%	1
0.00%	22.43%	18.18%	2
0.93%	13.08%	10.61%	4
0.93%	12.15%	9.85%	5
8.41%	8.41%	6.82%	5
2.80%	2.80%	2.27%	5
0.00%	0.00%	0.00%	5
1.87%	9.35%	7.58%	4
0.93%	7.48%	6.06%	5
4.67%	4.67%	3.79%	5
0.93%	0.93%	0.76%	5
0.93%	0.93%	0.76%	5
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	3
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	1
0.00%	0.00%	0.00%	1
0.00%	0.00%	0.00%	2
0.00%	0.00%	0.00%	2
4.67%	4.67%	3.79%	5
0.00%	0.00%	0.00%	5
0.00%	0.00%	0.00%	5

Find

Source File: C:/Users/guimar01/DS-5/Workspaces/DS-5.9/xaos/xaos-3.5/src/engine/docalc.c

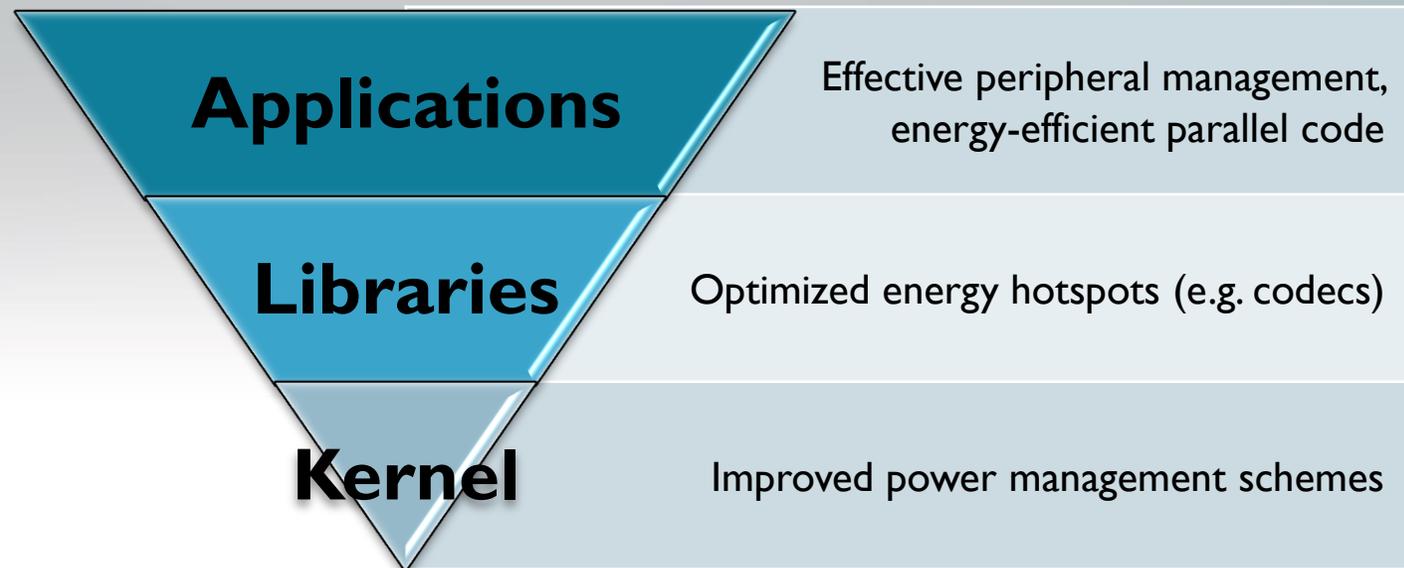
Line	Source
540	if (PCHECK)
541	goto periodicity;
542	FORMULA;
543	if (PCHECK)
544	goto periodicity;
545	FORMULA;
546	if (PCHECK)
547	goto periodicity;
548	FORMULA;
549	if (PCHECK)
550	goto periodicity;
551	FORMULA;
552	if (PCHECK)
553	goto periodicity;

Address	Opcode	Disassembly	File
0x00041040	EE344A01	VADD.F32 s8,s8,s2	docalc.c:548
0x00041044	EE766AA6	VADD.F32 s13,s13,s13	docalc.c:548
0x00041048	EE344A43	VSUB.F32 s8,s8,s6	docalc.c:548
0x0004104C	EEF00A61	VMOV.F32 s1,s3	docalc.c:548
0x00041050	EE460A86	VMLA.F32 s1,s13,s12	docalc.c:548
0x00041054	EE356A44	VSUB.F32 s12,s10,s8	docalc.c:549
0x00041058	EEB03AC6	VABS.F32 s6,s12	docalc.c:549
0x0004105C	EEB43A65	VCMPE.F32 s6,s11	docalc.c:549
0x00041060	EEF06A60	VMOV.F32 s13,s1	docalc.c:548
0x00041064	EEF1FA10	VMRS APSR_nzcv,FPSCR	docalc.c:549
0x00041068	EE203AA0	VMUL.F32 s6,s1,s1	docalc.c:548
0x0004106C	EE246A04	VMUL.F32 s12,s8,s8	docalc.c:548
0x00041070	5A000004	BPL 0x00041088 ; mand_peri + 0x328	docalc.c:549
0x00041074	EE730AE0	VSUB.F32 s1,s7,s1	docalc.c:549

Enabling Energy-Aware Coding

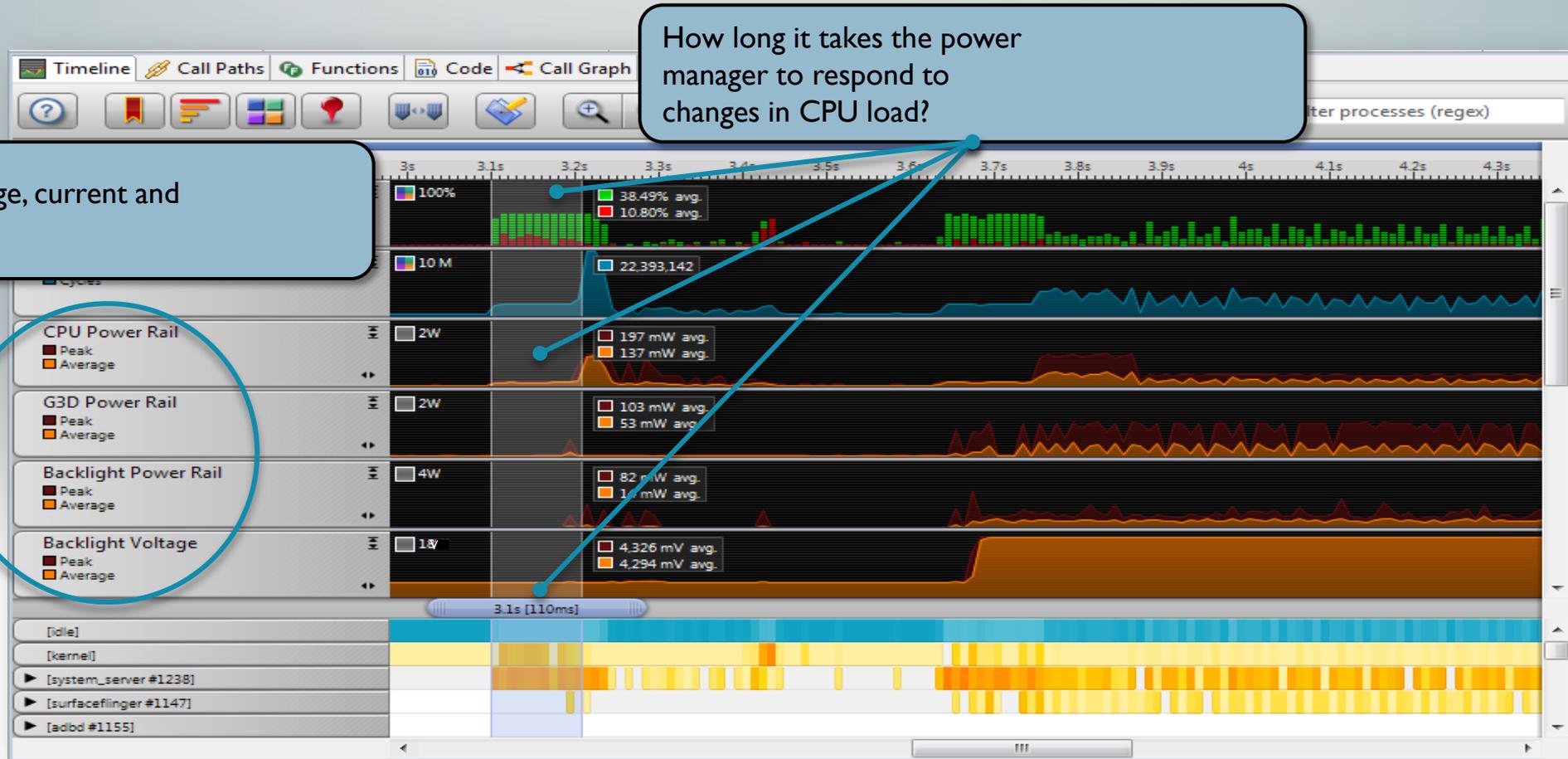
- ARM Energy Probe

- Lightweight power measurement for software developers
- Correlates power consumption with software execution in Streamline
- Monitor up to three voltage rails simultaneously
- Helps developers to make informed decisions at all layers of the software stack



The Power of Having It All in One Place

- How effective are you managing your energy budget?



Application Resource Optimizer (ARO)

- Free / Open Source Network-centric diagnostic tool
 - (yes, it is by AT&T but you don't need an AT&T device)
 - Requires root for pcap/data collection
 - APK on device, java desktop app for captured data analysis



Test Your Application



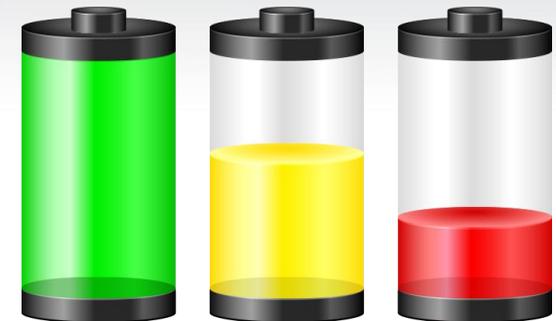
Transfer Trace Files



Process Trace

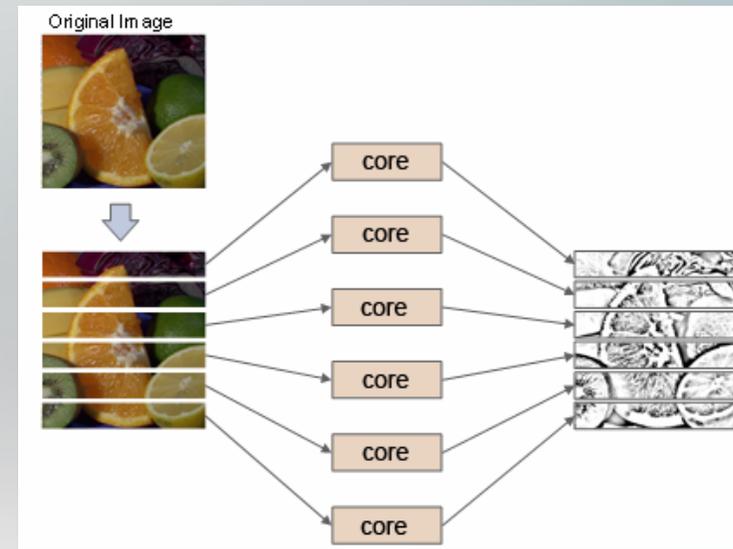
How Can ARO Make Apps Faster?

- The fixes identified by ARO will tune your application to higher performance and speed
 - App-specific Analysis
 - Highlight Key Areas to Improve
 - Increase Network Availability
 - Improve Battery Life
 - Get Faster Response Times
- Simple, common sense development best practices in network environments
 - Reducing connection times
 - Caching files
 - Eliminating errors
- Cross Platform and Network Agnostic



Analysis overload: Fixing the problems...

- **My leading questions:**
- Am I being smart with resources?
- Is this performance bottleneck parallelizable?
- Is this Java or Native?
Would it be better the other way around?
- Has this been done before? Don't reinvent the wheel.
- What version of Android should I target?



Networking Resources

- Close Connections
 - >80% of applications do NOT close connections when they are finished
 - 38% more power on LTE (18% more power on 3G)
- Cache Your Data
 - 17% of all mobile traffic is duplicate download of the same unaltered HTTP content (1)
 - “It’s just a 6 KB logo” -- $6 \text{ KB} * 3 \text{ DL/session} * 10,000 \text{ users/day} = 3.4 \text{ GB/month}$
 - Reading from local cache is 75-99% faster than downloading from the web
 - Even if caching IS supported – it is OFF by default
- Manage Every Connection
 - Group your connections
 - Save battery, speed up applications

(1) “Web Caching on Smartphones: Ideal vs. Reality”, http://www.research.att.com/~sen/pub/Caching_mobisys12.pdf

Closing Connections: CODE

- MultiRes Sample app from Android SDK

```
URLConnection getimagecloseconn = (URLConnection) url.openConnection();
```

```
getimagecloseconn.setRequestProperty("connection", "close");
```

```
getimagecloseconn.connect();
```

```
String cachecontrol = getimagecloseconn.getHeaderField("Cache-Control");
```

```
InputStream isclose = getimagecloseconn.getInputStream();
```

```
    bitmap = BitmapFactory.decodeStream(isclose);
```

```
    getimagecloseconn.disconnect();
```

Caching Methods (How do I do it?)

ETags

```
HTTP/1.1 304 Not Modified
Date: Mon, 29 Oct 2012 20:53:03 GMT
Server: Apache
Connection: close
ETag: "2d4f-4cd3729619900"
Expires: Tue, 29 Oct 2013 20:53:03 GMT
Cache-Control: max-age=0
```

- Each file has a Unique Tag
- Revalidated on server for each request
 - High Performance Web Sites:
Rule 1 – Make Fewer HTTP Requests ⁽¹⁾
 - Adding a connection drains battery,
adds 500-3,000 ms latency

Cache Control Headers

```
HTTP/1.1 200 OK
Date: Mon, 29 Oct 2012 20:51:38 GMT
Server: Apache
Last-Modified: Mon, 28 Jun 2004 00:03:33 GMT
Accept-Ranges: bytes
Content-Length: 27007
Cache-Control: max-age=1437313
Connection: close
Content-Type: image/jpeg
```

- Important to carefully assign Max-Age times
- App will not check file on server until Max-Age is reached
 - Retrieval is strictly file processing time

(1) http://developer.yahoo.com/blogs/ydn/posts/2007/04/rule_1_make_few/

Caching: Worth the Effort?

Android 4.0:

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);
```

Add this!

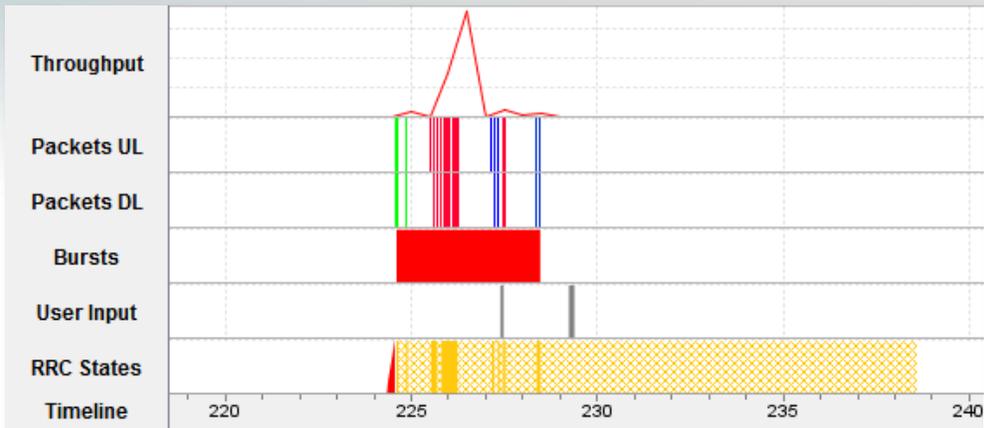
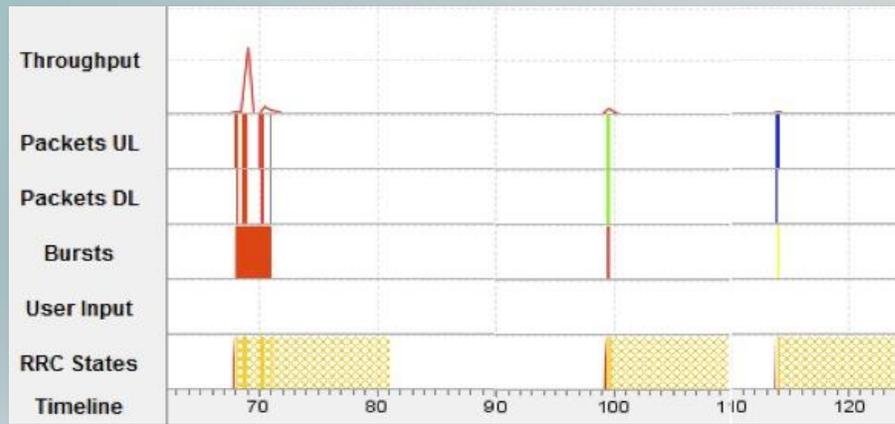
```
    //establish a cache  
    try {  
        File httpCacheDir = new File(getCacheDir(), "http");  
        long httpCacheSize = 10 * 1024 * 1024; // 10 MiB  
        HttpResponseCache.install(httpCacheDir, httpCacheSize);  
        //  
    }  
    catch (IOException e) {  
        Log.i(TAG, "HTTP response cache installation failed:" + e);  
    }
```

Don't leave older devices in the cold: Consider adding reflection for older versions of Android

Grouping Connections

1. Download an image every 60s
2. Download an Ad every 60s
3. Send Analytics to a Server every 60s

Ungrouped: 38J of energy used!!

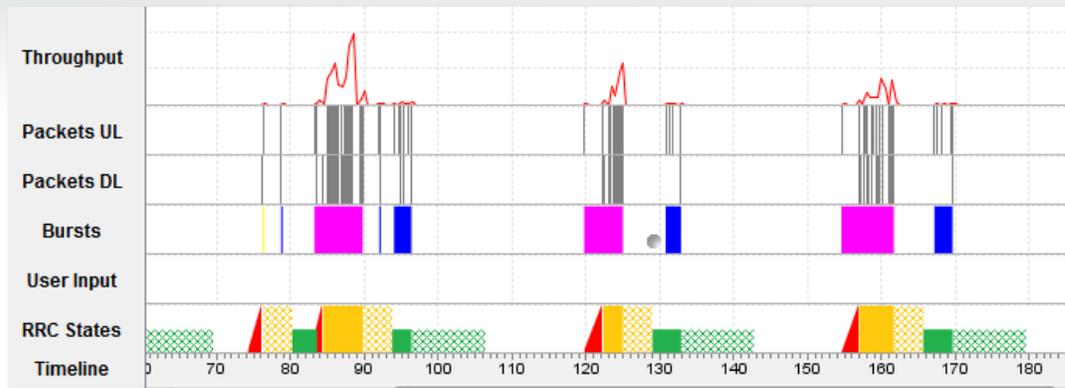


Grouped: 16J of energy used!!
58% savings!

Other best network practices

- Remove redirects to files, they add ~2-3 seconds per request
- Pre-fetching files that are used often
- Thread file downloads instead of serial download
- No 4xx 5xx http response error codes should occur
- Decouple user feedback from network activity.
- Be careful with periodic connections
 - Regular 3 minute polls for updates could remain connected for 1.2 hours of the day consuming around 20% of your battery.

Ad download every 30s



Going Native (NDK)

<http://developer.android.com/sdk/ndk/index.html>



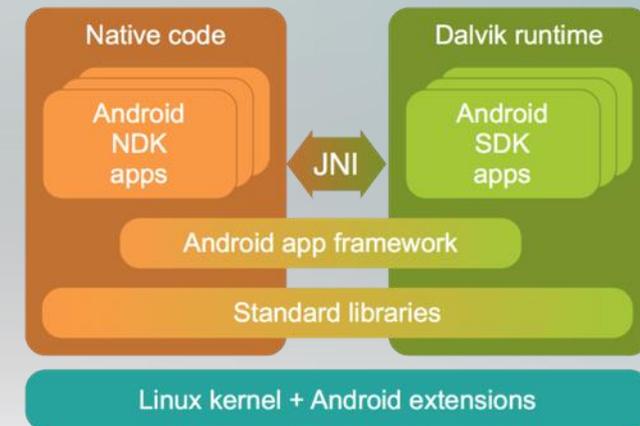
- Native Development Kit is used for writing native C/C++ code and calling your it from within an Android App through Java Native Interface (JNI).

Native Development Kit (NDK) for ARM

- NDK is a comprehensive tool kit to enable application developers to write directly for the ARM processor



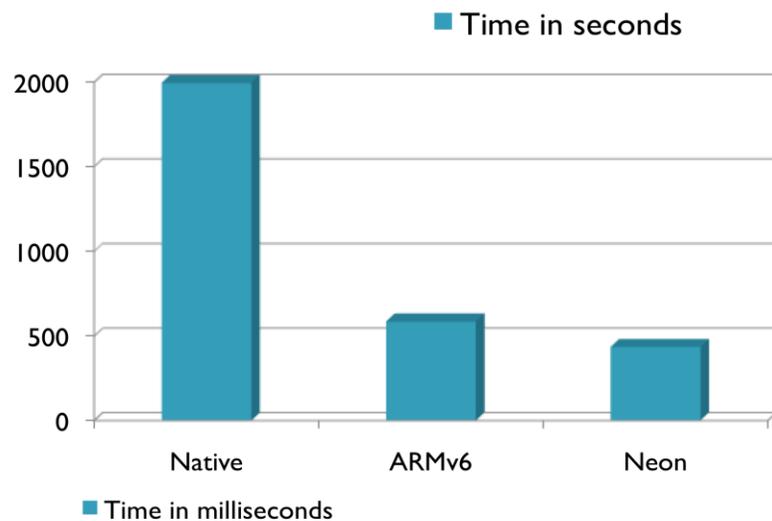
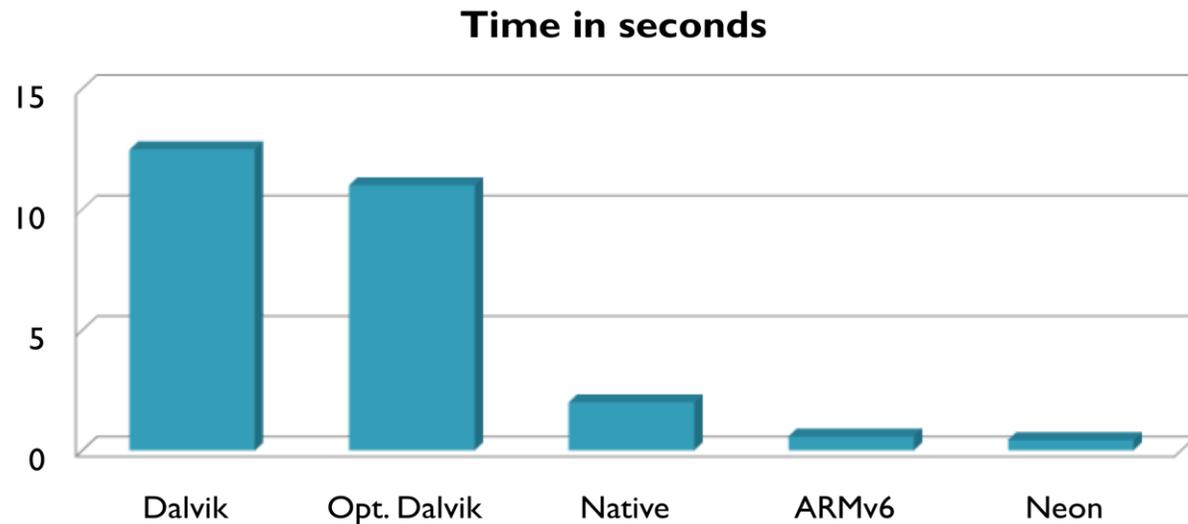
ARM Improvements	<ul style="list-style-type: none">■ Improved <i>performance</i> and <i>code density</i> with GCC 4.4.3<ul style="list-style-type: none">■ Optimizations for Cortex-A9■ Support for VFPv3
General highlights	<ul style="list-style-type: none">■ New <i>NativeActivity</i> feature eliminates need to write Java■ Addition of default C++ STL■ New API's<ul style="list-style-type: none">■ Input subsystem, sensor data■ Windows, surface subsystem■ OpenSL ES Audio API■ Access to APK graphics assets■ EGL library to create and manage OpenGL ES textures and services



Android™ applications can be written in Java, native ARM code, or a combination of the two

NEON supported since the r5 release

Benchmark Results - specific media intensive test case



Just Native

For more info on NDK, see my webinar at:

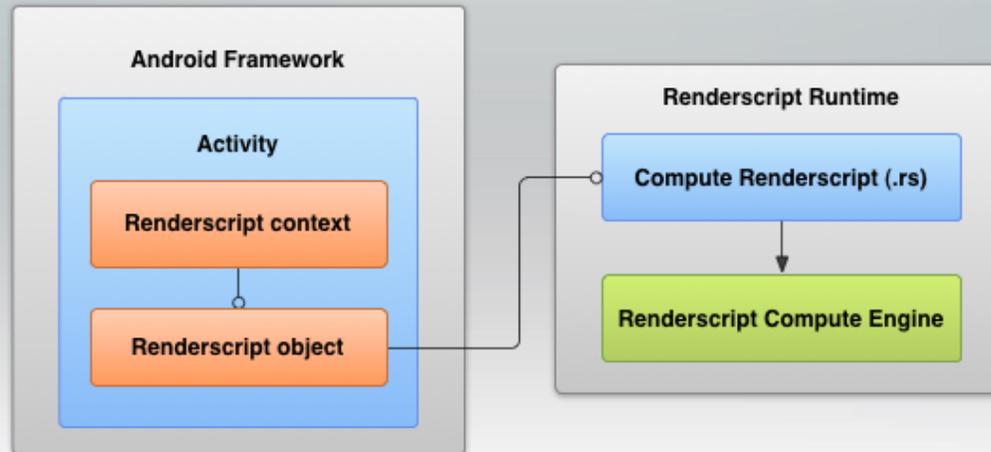
<http://goo.gl/GTwPH>

SMP and parallelization

- Nearly every Android and mobile device on the market today is multicore and the trend will continue – Design multi-threaded apps
- Davlik Java threads and IPC
 - AsyncTask is often the simplest way to quickly push a task onto a background worker thread with little IPC complexity
- Bionic C library implements a version of the Pthreads API
 - most of the `pthread_*` and `sem_*` functions are implemented but no SysV IPC
 - if it is declared in `pthread.h` or `semaphore.h`, it will mostly work as expected

SMP and parallelization

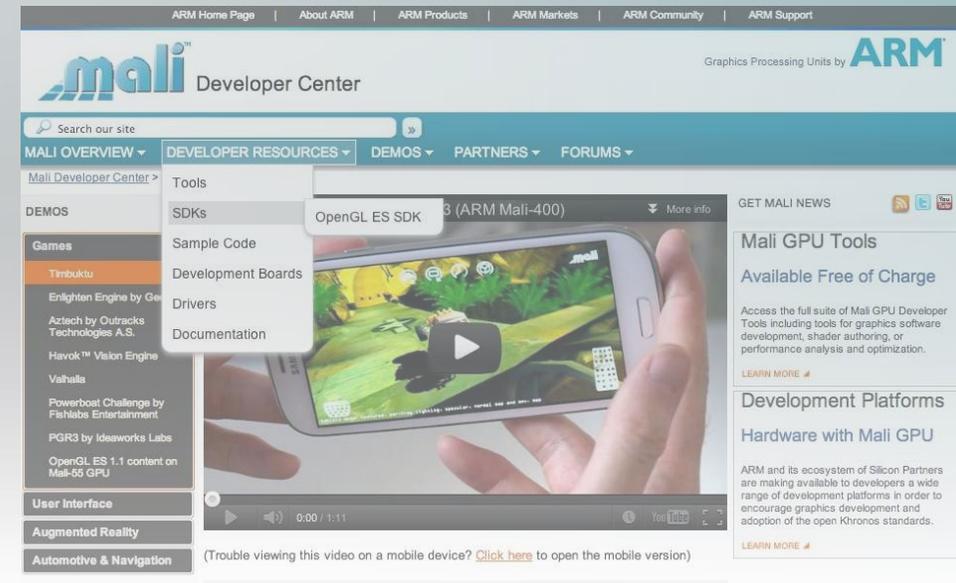
- GPU Compute: Renderscript Compute offers a high performance computation API at the native level
 - Write in C (C99 standard)
 - Run operations with automatic parallelization across all available processor cores
 - Platform independent



- Simpler to use than you might expect.
 - Your Renderscript code resides in .rs and .rsh files in the <project_root>/src/ directory
 - Call `forEachRoot()` with your renderscript function, input and output allocations.
 - See: developer.android.com/guide/topics/renderscript

SMP and parallelization

- OpenGL[®] ES 2.0 enables full programmable 3D graphics for programmable embedded GPUs
 - Royalty-free, cross-platform API
 - 2D and 3D graphics
 - Supported in both Android's framework API and the NDK
- malideveloper.arm.com
 - OpenGL ES SDK and Sample Code
 - Shader libraries and compiler
 - Texture compression and ASTC Codec
 - Asset compiler and conditioning tools
 - OpenGL ES 2.0 and 3.0 emulators
- Full profile OpenCL[™] is an option with some GPUs and possible to use in Linux but not supported by Google in Android. Developer beware.



Write java for mobile/embedded/battery

■ new

- Don't call this. Ever.
- At least not in CPU bound/frequent activities
- Try to use static variables or only allocate upfront or at natural pauses in activity
- Avoid triggering Garbage Collection (use DDMS)
watch Google IO 2009: <http://goo.gl/7xCMg>
- In JellyBean use features for graphics like
 - `android.view.Choreographer` for v-sync pulses
 - `myView.postInvalidateOnAnimation()`
 - don't draw stuff that won't be displayed `c.quickReject(items...)`, `Canvas.EdgeType.BW`

Android Dev Pro Tips – New API tips from IO 2013

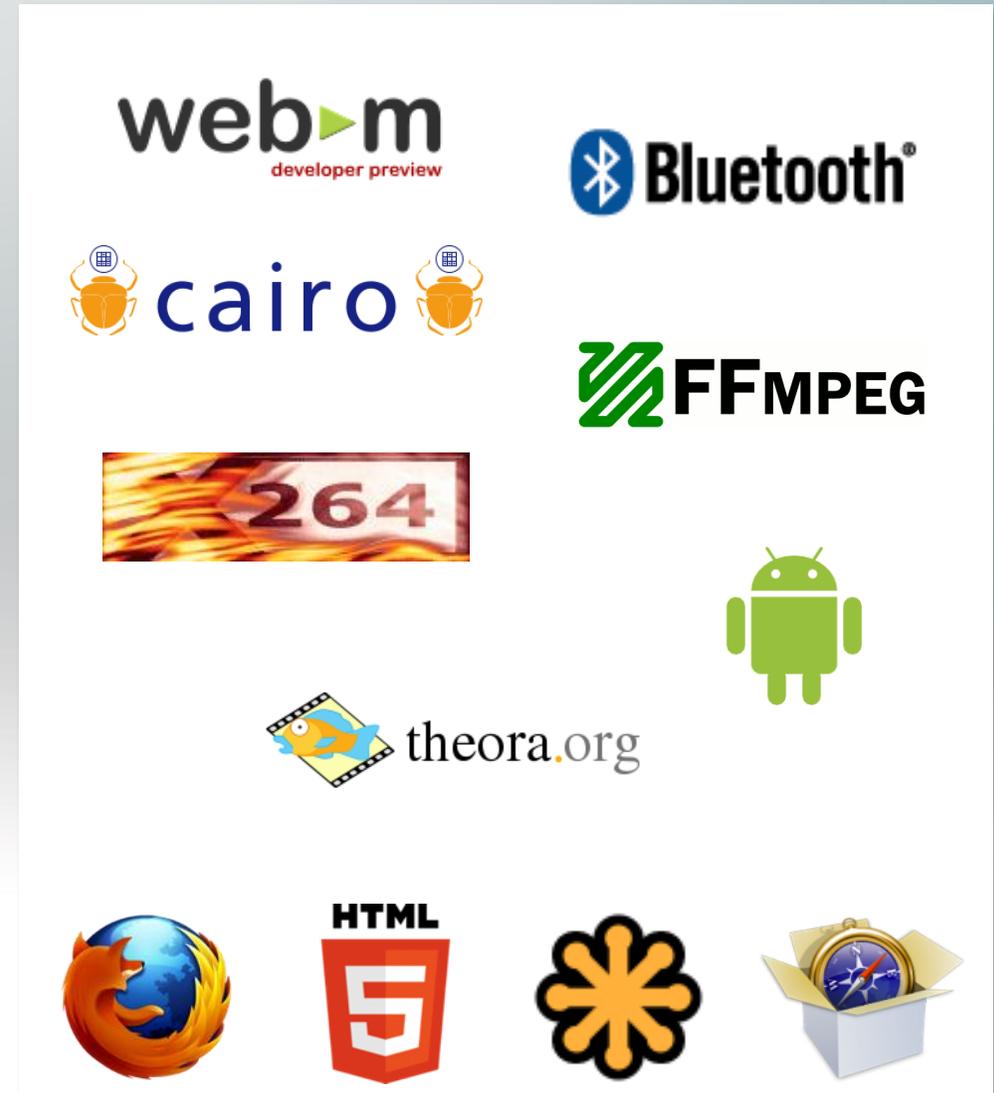
- Use Google Cloud Messaging to get notified of new info to synchronize rather than time based polling. `public class MySyncAdapter extends AbstractThreadedSyncAdapter {}`
 - GCM allows for persistent XMPP connections.
 - You can use to upstream data now.
- Fused Location Provider Uses Wifi and accelerometer for indoor position & GPS outside
 - Don't worry about monitoring Wifi, GPS and accelerometer yourself.
 - HIGH_ACCURACY – updates every 5 sec, 7.25%/hr bat drain
 - BALANCED_POWER – updates on 20 sec interval, 0.6%/hr
 - NO_POWER mode – accurate to 1 mile
- GeoFencing: Rather than poll a users location, just setup a fence.
 - `addGeoFence` saves 2/3rds power over `addProximityAlert()`...

SIMD: NEON

- General purpose SIMD processing useful for many applications
- Supports widest range multimedia codecs used for internet applications
 - Many soft codec standards: MPEG-4, H.264, On2 VP6/7/8, Real, AVS, ...
 - Supports all internet and digital home standards in software
- Fewer cycles needed
 - NEON will give 1.6x-2.5x performance on complex video codecs
 - Individual simple DSP algorithms can show larger performance boost (4x-8x)
 - Processor can sleep sooner => overall dynamic **power saving**
- Straightforward to program
 - Clean orthogonal vector architecture
 - Applicable to a wide range of data intensive computation.
 - Not just for codecs – applicable to 2D/3D graphics and other processing
 - 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
 - Off-the-shelf Tools, OS, commercial & open source ecosystem support

Don't Reinvent the wheel! NEON in Open Source Today

- **Google WebM** – 11,000 lines NEON assembler!
- **Bluez** – official Linux Bluetooth protocol stack
- **Pixman** (part of cairo 2D graphics library)
- **ffmpeg (libav) – libavcodec**
 - LGPL media player used in many Linux distros and products
 - Extensive NEON optimizations
- **x264** – Google Summer Of Code 2009
 - GPL H.264 encoder – e.g. for video conferencing
- **Android** – NEON optimizations
 - **Skia** library, S32A_D565_Opaque **5x** faster using NEON
 - Available in Google Skia tree from 03-Aug-2009
- **LLVM** – code generation backend used by Android RenderScript
- **Eigen2** – C++ vector math / linear algebra template library
- **TheorARM** – libtheora NEON version (optimized by Google)
- **libjpeg / libjpeg-turbo** – optimized JPEG decode
- **libpng** – optimized PNG decode
- **FFTW** – NEON enabled FFT library
- **Liboil / liborc** – runtime compiler for SIMD processing
- **webkit** – used by Chrome Browser



How to use NEON



- Opensource libraries, e.g. OpenMAX, libav, libjpeg, Android Skia, etc.
 - **Freely available Open Source optimizations**
- Vectorizing Compilers
 - Exploits NEON SIMD automatically with existing source code
 - **Status:** Released (in DS-5 armcc, CodeSourcery, Linaro gcc and now LLVM)
- C Intrinsics
 - C function call interface to NEON operations
 - Supports all data types and operations supported by NEON
 - **Status:** Released (in DS-5 and gcc)
- Assembler
 - For those who really want to optimize at the lowest level
 - **Status:** Released (in DS-5 and gcc/gas)
- Commercial vendors
 - Optimized and supported off-the-shelf packages



What is Project Ne10?



- NE10 is designed to provide a set of common, useful functions which
 - have been optimised for ARMv7 and NEON
 - provide consistent well tested behaviour
 - and that can be easily incorporated into applications
 - Is targeted at Android and Linux to maximize app performance
- Features
 - Usable from C/C++ and Java/JNI
 - The library is modular; functionality that is *not* required within an App can be discarded
 - Functions similar to the Accelerate Framework provided by iOS

Why use Project Ne10?

- It is Free
 - No commercial complications- 'build and ship' BSD License
 - No liability offered from ARM, no money paid to ARM
 - well-tested behavior with example code
- Use of the Ne10 library should be a joy, not a chore
 - Out-of-box and user experience is critical to success
 - Build and go, accessible documentation, clear code
 - Code promotes the **best** of the ARM Architecture- build on it
 - Lets **you** get the most out of ARMv7/NEON without arduous coding
 - Supported by ARM, community contributions welcome

Ne10Droid – The App in action

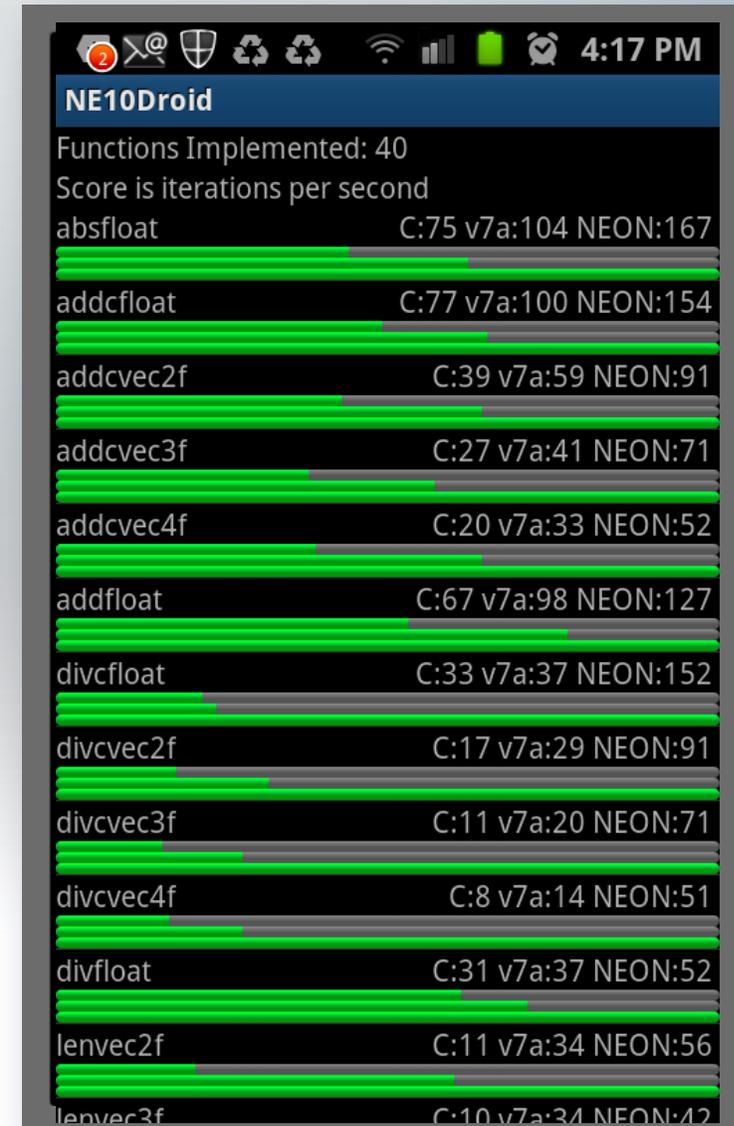
- NE10Droid is a benchmarking Android App that uses NE10.
- Routines are written using VFP in C, VFP in Assembly and NEON.

■ Example routines:

```
arm_result_t normalize_vec2f(arm_vec2f_t *  
dst, arm_vec2f_t * src, unsigned int  
count);
```

```
arm_result_t normalize_vec3f(arm_vec3f_t *  
dst, arm_vec3f_t * src, unsigned int  
count);
```

```
arm_result_t normalize_vec4f(arm_vec4f_t *  
dst, arm_vec4f_t * src, unsigned int  
count);
```



Conclusions

- For the Simple, Quick-to-Market option, stick with Dalvik but consider JellyBean's tools and NDK options
 - Can always optimize in version 1.1
- Be smart about your network resources
- Why write code you don't have to?
 - Look for highly optimized code with a compatible license
 - Can be beneficial without being a perfect fit
- Ideal candidates for Threads, NEON and GPU Compute: audio, image, video and game code
- There is a lot of extra performance there if you really need it
 - Particularly if you can use ARMv6, v7 extensions (NEON)
 - Learning new stuff is fun, so experiment

Solution Center for Android

- The SCA offers developers the widest range of Android resources for ARM architecture. Over 200 ARM Connected Community members come together to share their Android development expertise, solutions and services, including:
 - Development tools
 - Resources for building devices
 - Porting Guides
 - White papers
 - Android training
- community.arm.com/groups/android-community
- arm.com/solution-center-android
- androidtools.org
- projectNe10.org
- malideveloper.arm.com



Extra slides:

- Extra slides:
- [Link to this presentation](#)

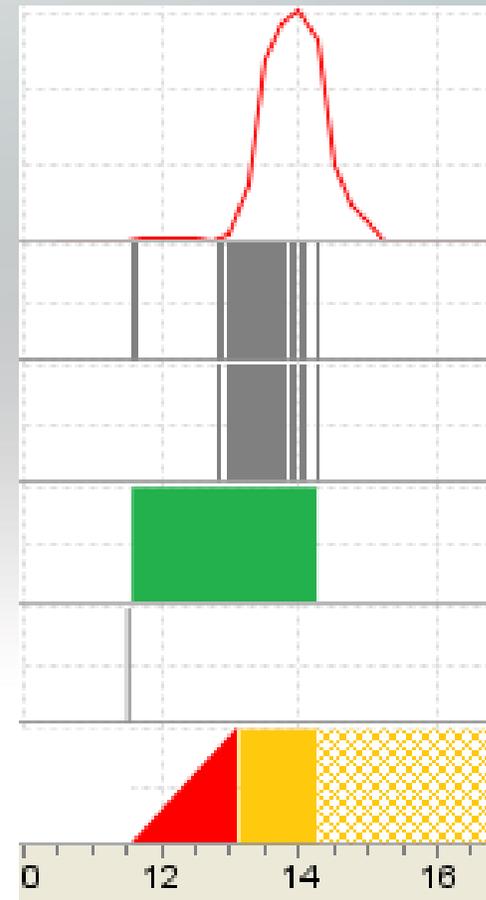
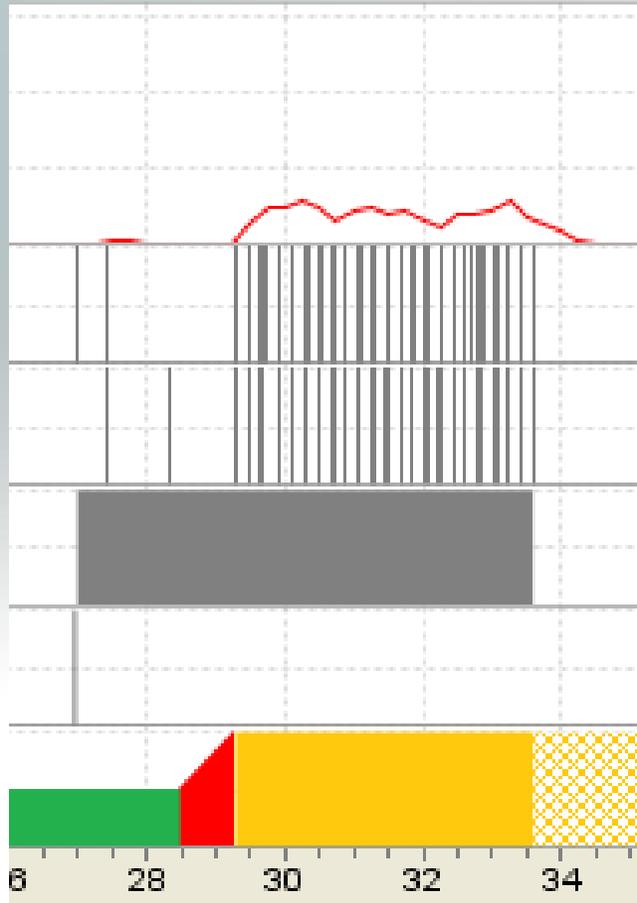


How Do I Group Connections?

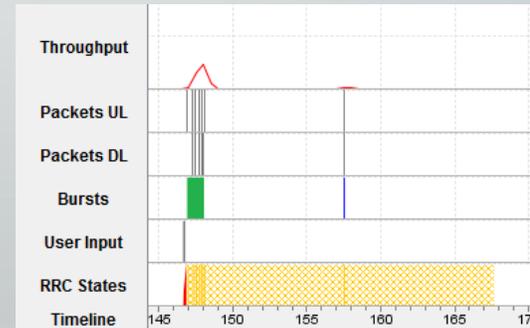
```
if (Tel.getDataActivity() >0){  
    if (Tel.getDataActivity() <4){  
        //ok, we are passed the minimum time to check  
        //and we found network activity-  
        //download the image here using image getter  
        imagegetter(counter, numberofimages);  
        //and show the ad  
        AdRequest adRequest = new AdRequest();  
        adRequest.addTestDevice(AdRequest.TEST_EMULATOR);  
        adView.loadAd(adRequest);  
        // Initiate a generic request to load it with an ad  
        adView.loadAd(new AdRequest());
```

Grouping Connections for Speed

- Threading file downloads vs. serial download



Closing Connections: Example



38% more power on LTE!
(18% more power on 3G)



Time	Direction	Type	Payload Length
190.284	DOWNLINK	DATA	1257
190.284	UPLINK	ACK	0
190.288	UPLINK	CLOSE_CONN	0

Time	Direction	Type	Payload
147.873	UPLINK	ACK	0
147.873	UPLINK	ACK	0
148.018	DOWNLINK	DATA	1296
148.037	UPLINK	ACK	0
157.484	DOWNLINK	CLOSE_CONN	0
157.556	UPLINK	ACK	0

What's in Ne10 today?



- Absolute value, multiply and accumulate and other arithmetic operations of floating point arrays with scalar and constant values along with utility functions.
- DSP FIR/IIR, CFFT/CIFFT and RFFT/RIFFT functions
- SIMD Component-wise Arithmetic on Two Vectors
- Normalize up to 4 dimensional vectors of the input array and store them in the corresponding elements of the output array.

$$\hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{a_1}{\|\mathbf{a}\|}\mathbf{e}_1 + \frac{a_2}{\|\mathbf{a}\|}\mathbf{e}_2 + \frac{a_3}{\|\mathbf{a}\|}\mathbf{e}_3$$

*(*normalize_vec3f)(arm_vec3f_t * dst, arm_vec3f_t * src, unsigned int count)*

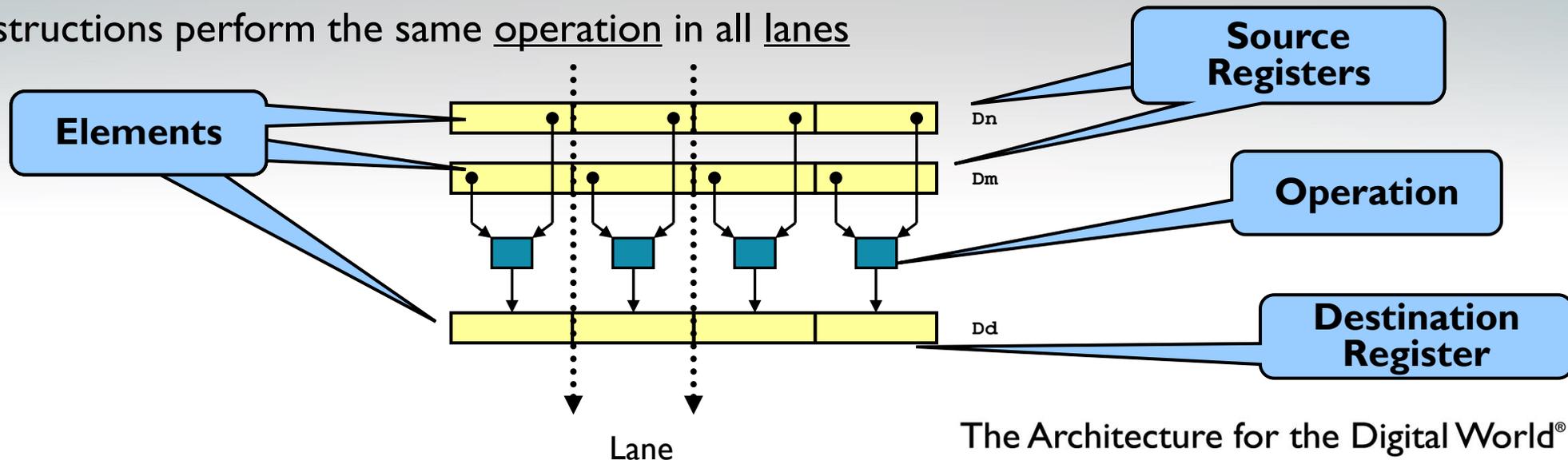
- Matrix-Constant Arithmetic e.g. add, sub, mult, div, invert, transform, identity, Matrix-Vector, Vector-Vector and Matrix-Matrix Algebra up to 4 dimensions provided

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & K \end{bmatrix}^T = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & D & G \\ B & E & H \\ C & F & K \end{bmatrix}$$

*(*invert_mat3x3f)(arm_mat3x3f_t * dst, arm_mat3x3f_t * src, unsigned int count)*

What is NEON?

- NEON is a wide SIMD data processing architecture
 - Extension of the ARM® instruction set
 - 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
- NEON Instructions perform “Packed SIMD” processing
 - Registers are considered as vectors of elements of the same data type
 - Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, single prec. float
 - Instructions perform the same operation in all lanes



Using NDK with a Web UI

- Prefer Web-style UI to the normal Android Widgets?
- Or just familiar with HTML and don't want to change?
 - Good news, you can still take advantage of the NDK
- Put a little extra zip in your Javascript
 - The pages don't have to be part of the assets
- Embed your web pages into an Android App Shell
 - You can extend the Javascript libraries with calls to Java
 - And those calls to Java can call NDK JNI Functions

```
WebView web = (WebView) findViewById(R.id.webview);
ProxyBridge jsriptBridge = new ProxyBridge();
web.addJavascriptInterface(jsriptBridge, "pBridge");

WebSettings settings = web.getSettings();
settings.setJavaScriptEnabled(true);

web.loadUrl("file:///android_asset/index.html");
```

Runtime Choice of Binaries

- Ideally want to ship one binary for all Android Devices
 - This can be done with a little thought
- Build a shared library for each HW variant you want
 - Remember to keep a Java implementation as fallback
- Use a Interface or Abstract Class pattern select the right code at runtime
- Use either Board (which gives device information) or a manual scan of `/proc/cpuinfo` to select which code to use
- Will bloat your App
- Use Thumb or Thumb2 instead ARM where you can
 - Native Thumb is still much faster than Dalvik
 - Thumb-2 is similar to ARM, supports NEON and V6 SIMD
- Weigh each choice of Native method carefully

Example Code

```
// Use Build, if you know exactly what you're looking for
if (Build.BOARD.compareTo("DualCoreA9DevBoard") {
    System.loadLibrary("SMPOptimisedA9Lib");
}

or

// Create a class to parse the OS's '/proc/cpuinfo' data
CPUInfo cpuinfo = CPUInfo.getCPUInfo();
if (cpuinfo != null) && (cpuinfo.isARM()) { //
    System.loadLibrary("medianFilterC");
}
if ((cpuinfo != null) &&
    (cpuinfo.getCPUArchitecture().isCompatible(CPUArch.ARMv5TEJ))) {
    System.loadLibrary("medianFilterv5");
}
if ((cpuinfo != null) &&
    (cpuinfo.getCPUArchitecture().isCompatible(CPUArch.ARMv6TEJ))) {
    System.loadLibrary("medianFilterv6");
}
if ((cpuinfo != null) && (cpuinfo.hasFeature(CPUFeature.ARM_Neon))) {
    System.loadLibrary("medianFilterNeon");
}
}
```

Use a Factory Pattern to hand your main code an appropriate implementation

```
MedianFilter filter = MedianFilterFactory.getMedianFilter(cpuinfo);

filter.doFilter(...parameters...);
```

NEON Visualizer

<http://szeged.github.com/nevada/>



▶
▶
↶
✎
↷
?
Random

NEVADA

Code

View Mode

- vadd.u8 q0, q1, q2
- vsub.8 q15, q2, q3
- vsub.s16 q13, q2, q3
- vsub.i16 d15, d2, d3
- vadd.i32 q0, q4, q5
- vand q0, q1, q2
- vand.u32 q0, q1, q2
- vand q15, q14
- vorr q0, q14, q15
- vbic d5, d6, d14
- vand d29, d28, d30
- vmov r1, r10, d22
- vmov d24, r1, r10
- vmov d25, d24
- vmov q13, q12

NEON Registers

Decimal uint8								Decimal uint8										
165	180	238	245	239	157	101	204	D1	41	233	207	78	220	49	179	63	D0	Q0
182	158	179	216	170	85	134	115	D3	165	188	195	132	192	225	38	209	D2	Q1
48	12	128	2	88	129	42	48	D5	46	181	210	169	84	164	159	236	D4	Q2
171	102	31	155	105	197	116	150	D7	51	220	162	2	92	129	42	181	D6	Q3
14	66	12	249	228	140	59	226	D9	142	20	27	123	97	221	56	94	D8	Q4
161	39	108	215	197	214	29	138	D11	18	106	195	54	168	182	37	215	D10	Q5
197	114	157	215	207	202	181	43	D13	216	210	248	248	41	120	116	60	D12	Q6
239	30	15	172	22	140	160	94	D15	7	211	107	196	5	62	148	143	D14	Q7
26	89	223	175	77	68	90	103	D17	242	88	163	197	58	132	58	167	D16	Q8
191	204	103	121	145	188	124	86	D19	131	250	198	112	222	199	107	103	D18	Q9
93	43	204	21	200	161	20	205	D21	199	152	248	233	83	111	186	49	D20	Q10
68	80	199	43	218	212	204	57	D23	230	94	4	166	20	231	105	157	D22	Q11
222	12	206	35	111	126	98	164	D25	88	88	34	82	25	89	99	101	D24	Q12
98	50	104	231	77	76	103	85	D27	250	217	48	167	248	35	117	55	D26	Q13
165	180	238	245	239	157	101	204	D29	41	233	207	78	220	49	179	63	D28	Q14
32	48	104	229	78	12	101	68	D31	41	201	0	6	216	33	49	55	D30	Q15

Working Memory

Decimal uint8	Decimal uint8								
7..0	0	0	0	0	0	0	0	0	0
15..8	0	0	0	0	0	0	0	0	0
23..16	0	0	0	0	0	0	0	0	0
31..24	0	0	0	0	0	0	0	0	0
39..32	0	0	0	0	0	0	0	0	0
47..40	0	0	0	0	0	0	0	0	0
55..48	0	0	0	0	0	0	0	0	0
63..56	0	0	0	0	0	0	0	0	0
71..64	0	0	0	0	0	0	0	0	0
79..72	0	0	0	0	0	0	0	0	0
87..80	0	0	0	0	0	0	0	0	0
95..88	0	0	0	0	0	0	0	0	0
103..96	0	0	0	0	0	0	0	0	0
111..104	0	0	0	0	0	0	0	0	0
119..112	0	0	0	0	0	0	0	0	0
127..120	0	0	0	0	0	0	0	0	0

ARM Registers

Decimal uint32	Register
0	R0
0	R1
0	R2
0	R3
0	R4
0	R5
0	R6
0	R7
0	R8
0	R9
0	R10
0	R11
0	R12
0	SP
0	LR
40	PC
0	PSR
0	FPSCR

Messages

