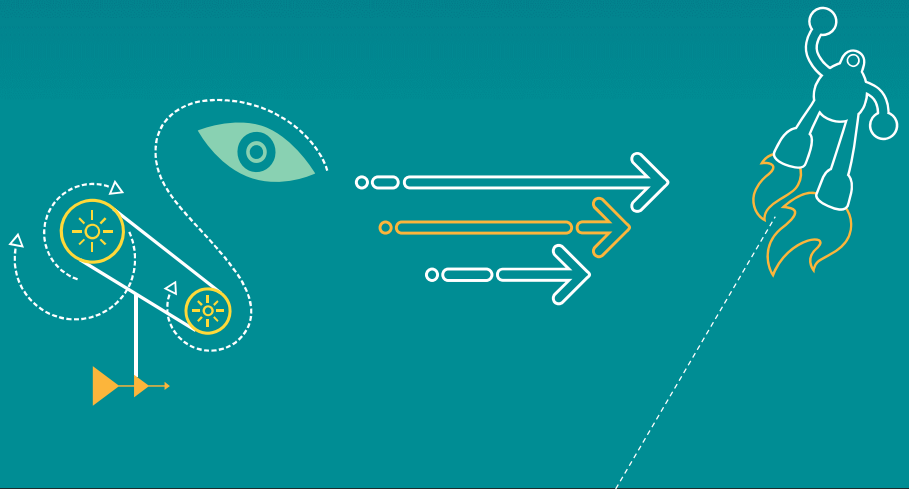
高通多媒体技术期刊 20150304

Qualcomm Technologies, Inc.



Confidential and Proprietary – Qualcomm Technologies, Inc.

Revision History

Revision	Date	Description
A	Mar 2015	Initial release

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

Contents

Profile GPU Rendering

Profile GPU Rendering概述

- 我们在2月6日的多媒体技术期刊中介绍了Google发布的一系列Android性能优化的视频，https://www.youtube.com/playlist?list=PLWz5rJ2EKKc9CBxr3BVjPTPoDPLdPIFCE, 视频中介绍了‘Profile GPU Rendering’ 工具。下面高通将对‘Profile GPU Rendering’工具进行进一步分析说明。
- ‘Profile GPU Rendering’ 工具是在android JellyBean引入的对应用渲染流水线进行性能调试的辅助工具。
  - 尽管工具名称叫Profile GPU Rendering, 其实工具对CPU 时间和GPU 时间都会测量。根据HWUI 流水线分为3个阶段：
    - draw time - 创建draw commands (display list) 的时间，由CPU完成。
    - process time - 处理draw commands (display list) 的时间，由CPU完成。
    - execution time - 等待commands (gpu command) 执行完成的时间，由GPU完成。
  - 需要强调, 上面3个阶段是在多个帧之间是并行进行的
- HWUI的操作方式在JB, KK 和L 之间也发上了一些改变
  - HWUI的变化增加了在各个阶段总体的并行程度
  - 这样可以减少系统的延时性
- 关于此工具的更多信息:
  - http://developer.android.com/about/versions/jelly-bean.html

HWUI在不同Android版本中的演进

- JB 上HWUI是完全串行的
  - Display list 创建和执行是在同一个线程
  - Buffer Queue 和 Dequeue 并没有使用任何fence相关同步机制
  - 因此在JB上有两个串行功能块，所以CPU 渲染和GPU渲染都必须在同一个Vsync 周期里完成。
- KK 上引入了Fence 机制进行Buffer管理
  - 允许CPU 运行的更快，运行在GPU 前面而不需要等待GPU
    - 增加了在系统显示前已经有一帧数据准备好了的概率。这样即使单独一帧数据错过Vsync也不会导致系统显示的卡顿。
- L 上 display list 创建和执行是在分开的两个线程里
  - 这意味这两个CPU的阶段可以单独占满一个VSync周期，而不会造成系统显示卡顿。
- 这样会使人对‘Profile GPU Rendering’工具在屏幕上显示的时间产生误解，因为每个阶段的时间加在了其后的阶段上面，而不是把它们并列地显示出来。
  - ‘dumpsys gfxinfo’ 命令会更加有用，因为它能把每个阶段的时间分开显示出来。

性能和功耗之间的平衡

- 是不是更低的Bars就更好?
  - 如果只是从性能的角度考虑，那无疑是对的
  - 但是如果把功耗也考虑进来，就不一定
- 当考虑功耗优化:
  - 更长时间运行在更低的电压具有更优的效率
  - 这意味我们想要选择尽可能最低电压产生足够时钟频率，来满足每个阶段都在 Vsync 窗口之内完成
    - 同时保留一些余量，以应对系统和应用的一些可能的变动
- 一个功耗优化的系统实际上要努力去运行更长时间
  - 这样它应该对任务选择最优的运行频率，而不总是选择最高的运行频率。

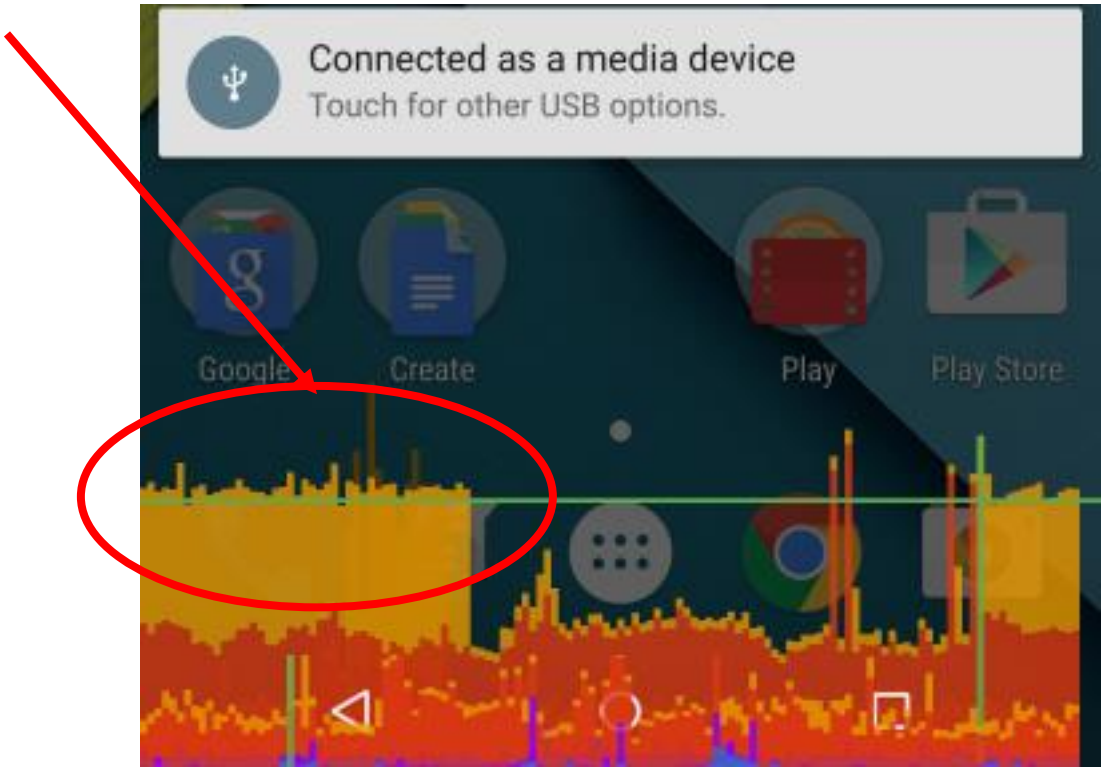
使用Profile GPU Rendering

- 屏幕显示的Bar只是告诉你这里是否可能有问题
  - 由于Android HWUI 各个阶段是并行的，因此总的时间长实际上并不表示屏幕卡顿的发生
    - 这仅仅表示所有阶段的时间加起来超过了一个 Vsync周期。
- ‘adb shell dumpsys gfxinfo’ 能提供更加详尽的信息
  - 如下表它把每个阶段分开
  - 在非孤立帧的情况下，任何一个单独的阶段时间超过一个Vsync必然会造成显示卡顿
- Systrace 工具能得到更多有用信息
  - http://developer.android.com/tools/help/systrace.html

StatusBar/android.view.ViewRootImpl@3a5a0dbe (visibility=0)			
Draw	Prepare	Process	Execute
0.14	0.88	6.75	8.63
0.16	0.5	6.87	9.19
0.16	0.48	6.75	9.08
0.16	0.47	6.76	8.8
0.15	0.66	6.74	9.44
0.16	0.5	6.62	9.08
0.16	0.48	6.2	10.15

Example: Nexus 6 Status Bar

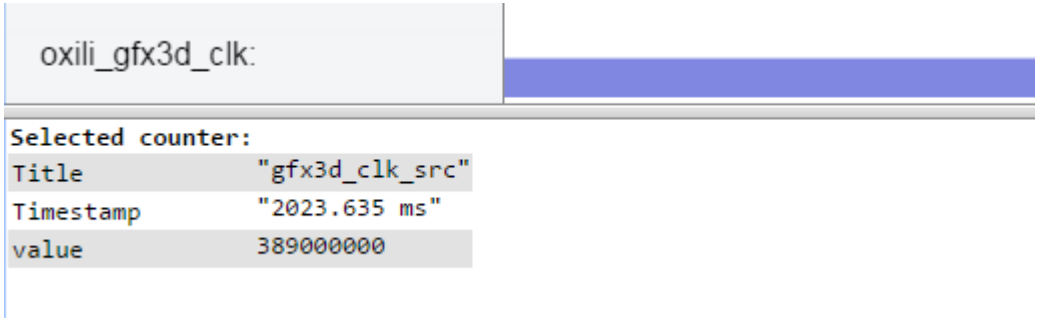
- 这个应用场景，简单地把Status Bar上下拖动
  - 在动画过程中，我们可以看到持续的 GPU 时间过长
  - 几乎所有的帧三个阶段时间总和都超过绿线位置(Vsync周期 16ms)
- 这可能是一个问题
- 需要其它工具进一步分析



Example: Nexus 6 Status Bar

- 我们可以用Google Systrace 工具得到很多有用信息
- SurfaceFlinger 进程显示一个应用queue了多少帧到SurfaceFlinger
  - 在这个例子里, 应用一直是很平滑的60帧/秒，总是有一帧数据准备好了

- 我们之所以看到很长的GPU 时间仅仅是因为GPU 没有运行在最高频率上
  - 让GPU 在较长时间运行在较低的频率上比让GPU在较短时间运行更高频率上对功耗更有效率
  - Systrace还可以显示GPU clock的变化:
    - 在这个例子中GPU 运行在 389 MHz
    - GPU在性能上面还有很多余量



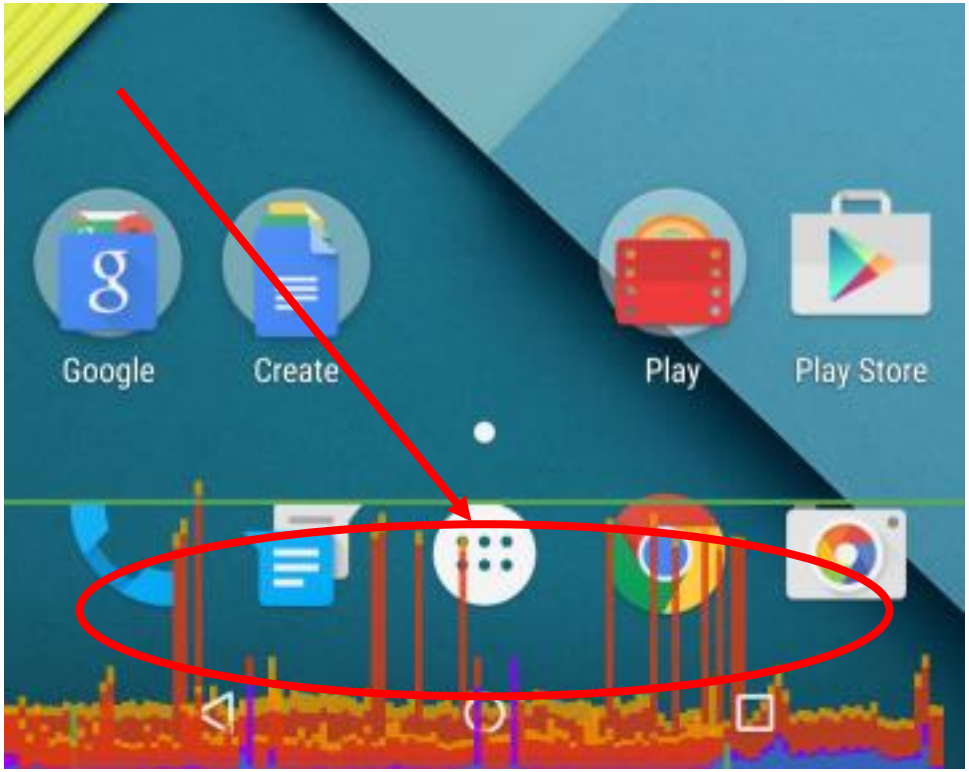
Example: Nexus 6 Status Bar

- 通过Systrace 的信息我们可以断定这不是一个真正的issue
  - 应用运行在60帧/秒，没有任何卡顿
  - GPU运行在较低频率，而不是较高频率以节省功耗
- 附加Profile GPU Rendering工具，查看‘dumpsys gfxinfo’ 同样可以很清楚地看到:
  - 每一个串行阶段都在Vsync周期之内
  - 系统显示看不到任何卡顿

StatusBar/android.view.ViewRootImpl@3a5a0dbe (visibility=0)			
Draw	Prepare	Process	Execute
0.14	0.88	6.75	8.63
0.16	0.5	6.87	9.19
0.16	0.48	6.75	9.08
0.16	0.47	6.76	8.8
0.15	0.66	6.74	9.44
0.16	0.5	6.62	9.08
0.16	0.48	6.2	10.15

Example: Nexus 6 Homescreen Hold

- 这个应用场景, 简单地把Homescreen从一边拖到另一边并保持
  - 开始显示很短的GPU 时间
  - 在保持了一会儿之后，看到GPU 时间(GPU command 执行的时间) 变长了
- 这可能是一个问题
- 需要其它工具进一步分析



Example: Nexus 6 Homescreen Hold

- 我们再一次使用Google Systrace 工具
- GPU Profiling 的红色部分等价于下面这部分

- 从上面我们看到多余的时间花费在了一个不可间断的睡眠操作上（uninterruptable sleep）。
  - GPU 进入很低的功耗模式，正在被唤醒。
- 我们可以说没有任何一个Vsync 被错过，因为整个操作是在一个Vsync周期完成的。
- 结论
  - 节省功耗引起的必然行为，没有引起任何显示卡顿。

Display

最新Display文档概述

- 高通发布了以下最新Display文档帮助客户定位和解决显示相关的问题
- 80-NR121-1_Android Display Debug Overview概括描述了Display问题调试的一般方法， 更详细的调试内容请参考80-NP925-1_Android_Display_Debug_Guide或者中文视频
- 80-NT784-1_Common_Debug_Mechanisms_Android_Display_Overview详细介绍了各种用于显示kernel和user space分析的脚本和命令，希望大家认真阅读
- 80-NV475-1_DSI_Display_Fading_Clock_Lane_Issue详细描述了我们在8939和8994上某些芯片可能出现的的DSI Clock Lane Stuck Issue， 文档对问题和解决方法都有详细的描述
- 80-NV489-1_DSI_Display_Fading_Data_Lane_Issue详细描述了我们在8994（不适用于8939）上某些芯片可能出现的的DSI Data Lane Stuck Issue， 文档对问题和解决方法都有详细的描述

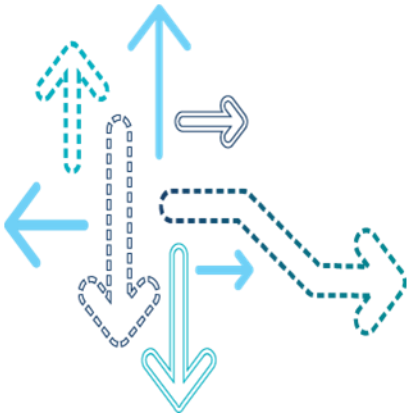
NOTE: 客户只能下载签约平台相关的文档

Power

最新Power文档概述

- 高通发布了以下最新Power文档帮助客户定位和解决功耗相关的问题
- 80-NT614-1_Android Power Basics and Power Feature Overview概括介绍了Android功耗相关的基本知识和Features,同时这篇文档是总目录树，文档中列举了大量参考文档供客户继续深入学习
- 80-NT615-1_Android Multimedia Power Debug Guidelines详细描述了多媒体功耗相关的调试方法和常见场景，建议客户重点阅读
- 80-NT616-1_Multimedia Power Debugging Case Studies Overview 更进一步对特定几个多媒体场景进行详尽的调试说明
- 80-NL239-48_Power_Consumption_Optimization_Debug_MSM8916介绍针对8916平台的功耗优化方法

NOTE: 客户只能下载签约平台相关的文档



Questions?

https://support.cdmatech.com