RenderScript

Writing high performance applications with renderscript

Ghilotti Giorgio
Matr. 765836, (giorgio.ghilotti@mail.polimi.com)

Report for the master course of Embedded Systems
Reviser: PhD. Patrick Bellasi (bellasi@elet.polimi.it)

Received: April, 01 2011

Abstract

This is a normal text in 10pt type size and 12pt line spacing. Place here a summary of your report. Focus on what is the problem and how you propose to tackle it. Briefly resume achieved results.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras dapibus. Donec fermentum mauris non lectus. Nulla porttitor pede id ante. Donec in erat pellentesque erat ultrices rutrum. Pellentesque id tellus. Donec vel mi non dui adipiscing tempor. Nunc laoreet pede ut lectus. Aenean aliquam quam a sem. Duis at elit? Ut dolor massa, dictum vel; convallis quis, ullamcorper id, urna.

1 Introduction

In the last seven years GPUs have become useful for a lot more applications than just traditional graphics due to their greater FLOPS and memory bandwidth versus CPUs, so they are really good at data parallel tasks, high performance computing, supercompiuting, and so on. And now these programmable GPUs are arriving on tablets and phones. In respect to desktop/server architectures, in mobile devices we have that CPU and GPU share the same pool of physical memory (see **Figure 1**), so they can tranfer data each other without pass them across a PCI express bus that, compared to the speed of memory bandwidth, is very slow. Another important difference between mobile and desktop is the number of architectural solutions. In mobile you have a lot of CPU, and even more GPU, vendors in respect to desktop. Particularly within the GPU space, the architecture are very different from one another. You also have issues like the GPU may be busy rendering a lot of pixels. For example on Nexus 10, we have a 2560x1600 display, which is equivalent to a 30 inch desktop monitor, and we're trying to drive that on the GPU with 80 gigaflops instead of over a teraflops. The goal of RenderScript is to develop high performance applications for these wide variety of SoCs without sacrificing performance portability. RenderScript framework provides a platform-indipendent computation engine that operates at native level so the programmer not have to take care about the architecture running underneath.

The first thing you will notice about the RenderScript API is that is focused on the system. You don't get a list of devices, or a big collection of device properties so you don't have to try to figure out at run time which device you should use. The runtime handles that for you. You simply have a computation that you want to run quickly and runtime will try to put that on the best processor it can. What this means is that this gives developers a consistent target that will run well across any SoC. Google works with SoC vendors and get drivers for their GPUs, DSPs, ISPs, and makes those available on tablets and phones. This way, the runtime knows about whatever processors and capabilities it can use.
Performance critical kernels are written in a C99 based language. This kernels are distributed with your application as architecture indipendent bit code. They are then JIT compiled at run time to one or more processor targets. Java classes are also reflected for easy integration with existing applications. This means essentially that every time you have a renderscript file, a set of java classes are generated, so that you can control execution of that directly from your java without relying on string based APIs, or having to use JNI. We also have a collection of script intrinsics. Essentially these are built in, very fast operations that we can tune to specific architectures very well. They're things like YUV conversion, or convolution.Common operations you may use very often, where you can get a really significant speedup by tuning very, very closely to a particular piece of hw.

1.1 The first subsection of the first Section

This is a citation [1] and here is another citation [2]. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
And this is the reference to a single column figure (see **Figure ??**). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus dapibus convallis odio. Nunc sollicitudin laoreet ante! Vivamus dictum euismod orci.

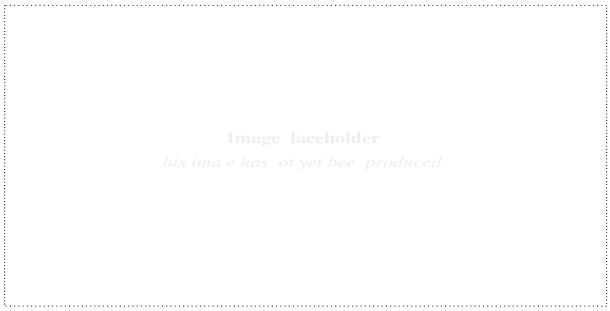

Figure 1: Some single-column figure caption.

1.2 The second subsection of the first Section

And this is the reference to a single column figure (see **Figure 2**). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam consectetur neque at orci. Curabitur non metus. Praesent congue porta nisl. Suspendisse ultricies, sem ac ultrices aliquam, erat nisi fermentum est; a rhoncus mauris arcu eget nibh. Aliquam sollicitudin velit non erat. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nulla diam, facilisis vel, accumsan sed; molestie egestas, massa. Fusce malesuada, ipsum et pulvinar aliquet, est dolor laoreet enim, quis porttitor erat mauris eget sapien. Integer vitae urna. Duis lectus.

2 The Second Section

The first thing you will notice about the API is that is focused on the system. You don't get a list of devices, or a big collection of device properties so you don't have to try to figure out at run time which device you should use. The runtime handles that for you. You simply have a computation that you want to run quickly and runtime will try to put that on the best processor it can. What this means is that this gives developers a consistent target that will run well across any SoC. Google works with SoC vendors and get drivers for their GPUs, DSPs, ISPs and all other processors, and have those available on tablets and phones. This way, the runtime knows about whatever processors and capabilities it can use.

2.1 The first subsection of the second Section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam consectetur ante at eros. Vestibulum mi nisi, venenatis sollicitudin, tempus sed, auctor id, tortor. Fusce orci. Duis tellus arcu, euismod sed, consequat sit amet, elementum vel, mauris. Curabitur leo diam; dapibus quis, condimentum vitae, dignissim ut, diam. Nulla et nulla eget elit volutpat sagittis.

2.2 The second subsection of the second Section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris eget mauris. Nulla facilisi. Ut condimentum tempor eros? Integer metus mauris, consectetur sit amet, tempor a, facilisis eu, nisl. Vestibulum at turpis. Ut vitae tortor pretium nisl vestibulum blandit. Nulla nibh urna, semper et, elementum at, mattis ut, nisi! Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Morbi vel ligula eget lacus convallis venenatis. Aliquam lacinia tincidunt felis. Ut dui.

References

[1] Ramsey, N.: Learn technical writing in two hours per week. Technical report, Harvard University (2006)

[2] Hughes, S.P.J.J.L.J.: How to give a good research talk. SIGPLAN Notices (1993)

Figure 2: Some wide-figure caption.