业界 移动开发 云计算 软件研发 程序员 极客头条 专题

Android

iOS

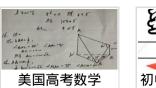
移动游戏

HTML5

智能硬件

Swift

近匠















CSDN首页 > 移动开发

订阅移动开发RSS

Adreno GPU 矩阵乘法——第1讲:OpenCL优化

发表于 2016-10-25 16:35 | 5032次阅读 | 来源 Qualcomm | 0 条评论 | 作者 Vladislav Shimanskiy

GPU OpenCL Qualcomm 骁龙 Adreno

摘要:本篇文章中的概念和下一篇文章中的OpenCL代码清单,表示Adreno 4xx和5xx GPU系列设备端矩阵乘法内核函数和主机端参考代码的优化实现。我们希望本系列文章将帮助和鼓励您使用这些想法和代码示例写出自己的OpenCL代码。

作者简介:

Vlad Shimanskiy是Qualcomm公司GPU计算解决方案团队的高级工程师。他一直致力于 开发和原型设计Snapdragon上OpenCL 2.x新的标准特性,改进Adreno GPU架构,用于 计算和加速重要线性代数算法,包括GPU上的矩阵乘法。

由于近来依赖于卷积的深度学习引起广泛关注,矩阵乘法(MM)运算也在GPU上变得流行起来。我们也收到开发人员的反馈,希望利用配备Adreno™GPU的Qualcomm®Snapdragon™处理器加速深度学习(DL)应用。

本文由我们Adreno工程师Vladislav Shimanskiy撰写,分为两个部分。本篇文章中的概念和下一篇文章中的OpenCL代码清单,表示Adreno 4xx和5xx GPU系列设备端矩阵乘法内核函数和主机端参考代码的优化实现。我们希望本系列文章将帮助和鼓励您使用这些想法和代码示例写出自己的OpenCL代码。

像Adreno GPU这样的并行计算处理器是加速线性代数运算的理想选择。然而,MM算法在密集并行问题中具有其独特性,因为它需要在各个计算工作项之间共享大量的数据。在要相乘的矩阵中,例如 A和B,每个元素对结果矩阵C的不同分量贡献多次。因此,为Adreno优化MM算法需要我们利用GPU内存子系统。

关于GPU 上的矩阵乘法存在哪些困难?

当我们尝试在GPU上加速MM时,上面提到的数据共享问题又可以拆分为几个相关问题:

- MM对相同的值进行重复运算,但是矩阵越大,越有可能必须到内存中读取(缓慢)已有值替换缓存中的值,这样做效率低下。
- 在MM的简单实现中,很自然的将标量矩阵元素映射到单独的工作项。但是,读写标量的效率很低,因为GPU上的存储器子系统和算术逻辑单元(ALU)被优化用于向量运算。
- 同时加载大矩阵A和B的元素有可能导致缓存冲突和存储器总线争用的风险。
- 内存复制很慢,因此我们需要找到一个更好的方法,使数据对CPU和GPU同时可见。

这些问题使MM的主要任务复杂化,即多次读取相同的值并共享数据。

矩阵乘法的OpenCL 优化技术

我们详细说明了一个OpenCL实现,其中包括解决每个问题的技术。

1. 平铺 (Tiling)

第一个众所周知的问题是将从内存(比如高级缓层或DDR)中重复缓慢读取相同矩阵元素的次数降到最低。我们必须尝试对内存访问(读取和写入)进行分组,以使它们在地址空间彼此接近。





CSDN官方微信 扫描二维码,向CSDN吐槽 微信号:CSDNnews



程序员移动端订阅下载

每日资讯快速浏览

微博关注

CSDN

CSDN移动 北京 朝阳区

加关注

《无线传感器网络使用指南》 在物联网飞速发展的今天,随着数据呈指数级增长、安全意识不断提升、企业网络出现越来越多局限,企业升级无线传感器节点的需求不断增大。随着物联网在企业中的普及,IT专业人员部署和建设传感器网络也越来越便捷。 http://t.cn/Ra2fjLT

5月5日 13:36

转发 | 评论

我们改进数据重用的技术是将输入和输出矩阵拆分为称为tile的子矩阵。然后,我们强制执行内存运算指令,使得矩阵乘法得到的点积在整个tile中部分完成,之后我们将读取指针移动到tile边界之外。

我们的算法确认两个层次的平铺:micro-tile和macro-tile。下图表示如何映射矩阵,使矩阵A中的分量乘以矩阵B中的分量,得到矩阵C中的单点积:

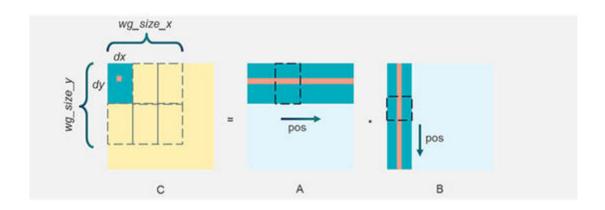


图1:平铺

micro-tile—— $\{dx, dy\}$ 是矩阵内的矩形区域,由内核函数单个工作项处理。每个工作项是SIMD子组中的单线程,反过来又形成OpenCL工作组。通常,micro-tile拥有 $4\times8=32$ 个分量,称之为像素 (pixel)。

macro-tile——{wg_size_x, wg_size_y},通常是由一个或多个micro-tile组成并且对应于工作组的更大矩形区域。在工作组中,我们完全在macro-tile范围内运算。

要计算矩阵C中的4×8micro-tile,我们将重点放在矩阵A和B中分别拥有4×8和4×4大小的区域。我们从 pos = 0开始,计算部分结果或点积,并将其存储在该micro-tile临时缓冲区。同时,相同macro-tile中的其他工作项使用从矩阵A或矩阵B加载的相同数据并行计算部分结果。矩阵A行中所有数据被共享。同样,矩阵B的列中所有数据在同一列的工作项之间共享。

我们计算macro-tile中的所有micro-tile的部分结果,然后在A中水平地增加pos,同时在B中垂直地增加pos。通过进行针对tile的计算并使pos逐渐递增,我们可以最大程度地重复利用缓存中的已有数据。micro-tile继续积累或卷积部分结果,将其增加到点积。

所以,在macro-tile内的所有位置完成所有的部分计算后,我们才移动位置。我们可以完成整个micro-tile,从左到右和从上到下移动pos,然后前进,但是这样做效率不高,因为我们需要的相同数据已经被缓存清除。关键是我们在一个由工作组限制的区域工作,有若干工作项目在同时运行。此方法保证来自并行工作项的所有内存请求均在有边界的地址区域内发出。

平铺(Tiling)通过专注于内存中的特定区域(工作组)来优化运算,这样,我们可以以缓存友好的方式进行工作。与跨越大块内存、必须到DDR中读取不再存于缓存中的值相比,效率得到了极大的提升。

2. 矢量化

由于内存子系统在硬件层面为矢量运算进行过优化,所以最好使用数据向量而不是标量来运算,并且使每个工作项处理一个micro-tile和一个全矢量。因此,我们可以使用每次向量读取操作时获得的所有值。

例如,在32位浮点矩阵的情况下,我们的内核函数使用float4类型的矢量,而不仅仅是一个浮点类型。这样,如果我们想从矩阵中读取一些东西,我们不仅读取矩阵的单个浮点分量,而且读取整个数据块。这一点很重要,因为它同总线设计方式是一致的。因此我们从矩阵中读取4个元素的分量,并使内存带宽饱和。相应地,micro-tile 的大小均为4的倍数。

如果我们在CPU上工作,我们可能一次读取一个2-D数组一个标量元素,但GPU上的OpenCL提供了更好的方法。为使读写更加高效,我们使用数据类型float4或float4或float4的倍数变量进行操作。

3. 纹理管道 (Texture Pipe)

两个矩阵使用独立缓存(L2 direct和Texture Pipe / L1),如下图所示,允许我们避免大多数争用和并行读取操作,以便矩阵A和矩阵B的数据在同一时间得到加载。涉及L1有助于大大减少到L2的读取流量。



相关热门文章

可穿戴机器人时装:触手、纺织品和DragonBoa...
开发者新征程:扩展到新的开发领域
WiFi万能钥匙发公告:停止3.0版本服务
酷玩6发布会再爆看点,王自如将对话刘江峰
易到收获网约车牌照 引价值重估
永洪科技推出SaaS模式,云中部署BI将成主流?
空净峰会"力挺"三星品质,净化器产业迎来里程碑
千面戏骨胡静一直播 从高小琴到向警予无缝切换
锤子科技又卖关子?倒计时海报"泄露"了一些秘密

活动

01-01	英特尔正调查苹果iPhone与PC资料同步 化失败问题
01-01	10个windows8应该改进的地方
01-01	Windows7时代,我们如何攒机?
01-01	英特尔高管称Win7普及快于Vista
01-01	XP升级Windows7 硬盘数据被全部清空
01-01	Windows7 RTM大战Vista SP2! Win7性 能稍强

聚合数据亮相GMIC大会 API经济获热捧

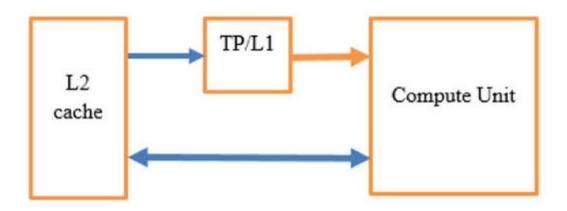


图2:纹理管道(Texture Pipe)

Adreno和许多其他GPU一样,每个计算单元具有到纹理管道(TP)单元的独立连接。TP具有其自己的L1缓存,并独立连接到L2缓存。

我们增加带宽的技巧是通过TP加载一个矩阵,通过直接加载/存储管道加载另一个矩阵。因为我们在矩阵乘法中重用了这么多的分量,所以我们还获得了L1缓存的优势。最终,从TP/L1到计算单元的流量远高于从L2到L1的流量。该区块显著降低了流量。如果不利用TP,只是连接到L2,就不会有太大帮助,因为在两个总线之间有很多争用和仲裁。

结果导致直接连接上产生大量流量,而从TP/L1到L2流量却很少。这有助于我们增加总内存带宽,平衡ALU运算,实现更高的性能。我们等待数据从缓存返回的时间几乎和ALU运算相同,我们可以对其采用管道化方式,使它们不致成为瓶颈。

4. 内存复制预防

我们的OpenCL实现有两个部分:运行在GPU上的内核函数和运行在CPU上的主机代码,并由主机代码控制内核函数的执行。如果我们实现一个GPU加速库(如BLAS)来做矩阵乘法,那么输入矩阵将在CPU虚拟内存空间,并且乘法结果也必须在CPU内存中可用。为了加速GPU上的矩阵乘法,矩阵必须首先被传输到GPU内存。

传统方法是将矩阵复制到GPU地址空间,让GPU执行其计算,然后再将结果复制回CPU。但是,复制大矩阵所需的时间可能抵得上在GPU上总的计算时间,因此,我们希望避免使用低效率的CPU内存复制。Adreno GPU具有共享Snapdragon处理器内存硬件的优势,我们可以加以利用,而不是显式复制内存。

那么,为什么不简单地分配在CPU和GPU之间自动共享的内存?可惜,这样并不可行,因为我们需要解决诸如对齐等等限制。只有使用OpenCL驱动程序例程正确完成分配,才能使用共享内存。

结果

下图显示了Adreno各版本单精度一般矩阵乘法(SGEMM)的性能提升:



图3: Adreno GPU 4xx和530的性能数据

该图基于常用浮点运算数据。使用不同数据类型(8位、16位、固定点等)的其他MM内核函数可以根据我们在SGEMM采用的相同原理进行有效实现。

3 of 5 2017年05月12日 11:25

一般来说,我们对Adreno GPU优化的MM实现比简单实现至少快两个数量级。

接下来?

在下一篇文章中, 我将给出这些概念背后的OpenCL代码清单。

矩阵乘法是卷积神经网络中一个重要的基本线性代数运算。尤其是DL算法性能与MM相关,因为DL卷积的所有变化均可以简化为乘法矩阵。

上面描述的概念和您在下一篇文章中看到的代码并不是计算卷积的唯一方法。但事实上,很多流行的 DL框架,比如Caffe,Theano和谷歌的TensorFlow往往将卷积运算分解为MM,因此沿着这个方向思 考不失为一个好办法。敬请关注第2部分中的代码示例。

相关阅读:

Qualcomm Adreno GPU 如何获得更好的OpenCL性能——内存优化篇

经验分享: Silk Labs 如何以极低的成本,获得软硬件开发资源

如何开始使用Adreno SDK for Vulkan

Vulkan开发系列视频教程

更多Qualcomm开发内容请详见: Qualcomm开发者社区。



顶 0 踩 0









推荐阅读相关主题:

相关文章

最新报道

已有0条评论

还可以再输入500个字



有什么感想,你也来说说吧!

haijunz 欢迎您!

发表评论

最新评论

最热评论

还没有评论,赶快来抢沙发吧。

请您注意

- ·自觉遵守:爱国、守法、自律、真实、文明的原则
- ·尊重网上道德,遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- ·严禁发表危害国家安全,破坏民族团结、国家宗教政策和社会稳定,含侮辱、诽谤、教唆、淫秽等内容的作品

2017年05月12日 11:25

- ·承担一切因您的行为而直接或间接导致的民事或刑事法律责任
- ·您在CSDN新闻评论发表的作品,CSDN有权在网站内保留、转载、引用或者删除
- ·参与本评论即表明您已经阅读并接受上述条款



公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

2017年05月12日 11:25