doi:10.3772/j.issn.1002-0470.2022.04.003

一种基于图形处理器压缩结构的预取结构设计①

赵士彭②[*,**,***] 张立志[*,**,***] 章隆兵[*,**,***]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

摘 要 图形处理器(GPU)访存利用率已经成为影响其性能的关键瓶颈之一。在处理器设计中,访存的预取结构设计成为了提高访存利用率的主要方法之一。结合图形处理器的访存密集的特点,在提高预取性能的前提下,减小影响图形流水线正常效率成为热门的研究方向。本文基于一种图形处理器无损压缩的结构,提出了一套图形处理器的预取结构设计。本预取结构设计可在访存密集型的图形流水线中有效提高访存利用率,并不影响当前图形流水线的效率。实验结果表明,在 Godson GPU 图形处理器平台上,与传统预取结构相比,针对访存密集型测试程序,cache 命中率可以提高 15% 以上。针对访存空闲的测试程序,该设计不会对流水线产生负面影响。

关键词 图形处理器(GPU);访存子系统;预取结构;压缩结构

0 引 言

图形处理器(graphic processing unit, GPU)[1]是计算机系统[2]中处理 3D 实时图形程序的专用加速芯片,目前广泛应用于个人电脑、工作站、嵌入式设备与智能手机中。在现代计算机系统中,GPU 已经不仅仅作为图形处理器存在,而是参与了更多的通用计算,其结构及功能也变得越来越复杂[3-5]。中央处理器(central processing unit, CPU)与图形处理器之间的协同工作,不仅减轻中央处理器的负担,而且使计算机系统的整体性能也有了大幅提高[6]。GPU 一般以插卡的方式,通过主板上的图形加速接口(accelerated graphics port, AGP)或高速串行计算机扩展标准(peripheral component interconnect express, PCIe)插槽与 CPU 进行通信。GPU 目前已经逐渐面向通用,通用 GPU 已经在通用计算、机器学习、人工智能等相关领域发挥出不可替代的价值[7-9]。

随着半导体工艺的发展进步,图形处理器的集成度进一步提高,计算单元的数量以及运算速度都已呈现大幅提高。随着图形处理器统一的可编程着色器(shader)的出现,计算的性能及灵活性又出现很大程度的提高。但伴随着计算性能的提高,访存的瓶颈便愈加凸显。计算数据量的不断增大,访存速度的提升远不及计算性能的提升。当前,各大 GPU 厂商以及研究人员都越来越重视 GPU 访存的瓶颈,纷纷开始了大量的研究、设计和生产。

综上所述,鉴于访存子系统对 GPU 性能的严重制约,访存性能也由于技术难度无法和计算速度匹配,图形处理器应当利用有限的内存带宽,提高访存性能。在内存控制器的访存带宽有限的情况下,有效利用访存空闲时间进行预取也成为了提升访存性能的关键。

图形处理器的访存几乎贯穿整条流水线。从顶点着色器(vertex shader)读取顶点信息需要访存,到

① 国家自然科学基金(61521092, 61432016)和中国科学院重点部署项目(ZDRW-XH-2017-1)资助。

② 男,1993 年生,博士生;研究方向:计算机体系结构,处理器设计;联系人,E-mail: zhaoshipeng@ict.ac.cn。

(收稿日期:2021-01-30)

片段着色器(fragment shader)读取纹理信息需要访存,再到最后输出混合单元(output merger unit, OMU)需要读写深度信息及帧缓冲区颜色信息需要访存。由此可见,图形处理器访存的数据量大且密集,访存空闲时间较少,这也使得许多传统的预取机制对于图形处理器这种访存密集型的系统结构很难起到明显的作用,有时会导致白白浪费功耗及面积,甚至会影响性能。

本文提出了一种基于图形处理器压缩结构的预取结构设计。通过利用图形处理器无损压缩结构,将部分数据压缩为标记位并存储在片上缓存的 meta cache 中。这种无损压缩的设计,使数据的访存被完全压缩掉,无需再向总线发出访存请求。这时,即使访存密集型程序的情况下,访存总线会由无损压缩结构产生访存的空拍,出现访存空闲,预取机制便可利用这一空闲进行预取。本设计基于图形处理器无损压缩结构的特点,解决了图形处理器传统预取机制由于访存数量大、访存密集导致传统预取结构对预取效果提升不明显的问题。同时,由于图形流水线的特点,基于图形处理器无损压缩结构的预取机制,不会产生无效的预取,对整体的性能不会产生负面影响,不影响当前图形流水线的效率。

本文第 1 节介绍了目前常见的图形处理器流水线及图形处理器访存子系统结构中传统的预取机制。第 2 节介绍了本文提出的基于图形处理器压缩结构的预取机制设计。第 3 节介绍了基于本预取机制的实验结果及分析。第 4 节对全文进行了总结。

1 背景介绍

1.1 3D 图形流水线及其存储结构

图 1 是 3D 流水线的操作流程及存储结构,包含片上存储和片外存储两部分。片上存储主要包括用于存储顶点位置和属性数据的 Vertex cache,用于存储纹理数据的 Texture cache,用于存储深度数据的 Depth cache 和用于存储帧缓冲区颜色的 Color cache。Video memory 是图形处理器的片外存储,包含这些数据对应的 buffer[10]。

图形流水线首先通过片上缓存 Vetex cache 将顶点的位置、颜色以及光照及视窗变换信息从片外缓存 Video memory 读出。经过顶点着色器处理后,图元处理引擎(primitive engine,PE) 将顶点信息进行组装及光栅化后,变为片段像素数据送入片段着色器中。由于光栅化后,顶点信息会转换为像素信息,访存请求会出现增加,尤其针对较大图元,图元覆盖像素点很多,片段着色器的访存请求也会大幅增加。片段着色器通过片上缓存 Texture cache 从片外缓存 Video memory 读出像素的纹理信息。之后,这些像素信息会送入输出混合单元进行处理。输出混合单元在处理时需要通过片上缓存 Depth cache 从片外缓存 Video memory 中读出深度数据,经过处理后重新写回。输出混合单元还需要通过片上缓存 Color cache 从片外缓存 Video memory 读出帧缓冲区数据,经过处理后重新写回。

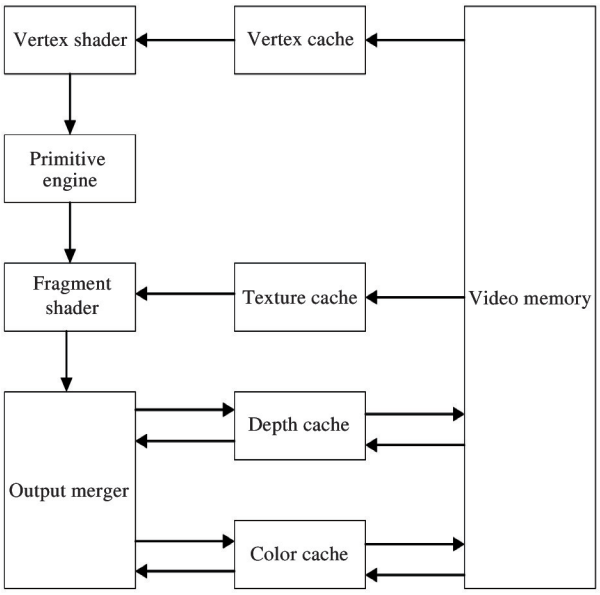

图 1 3D 图形流水线操作流程及存储结构

综上所述,图形处理器的访存请求仅在顶点着色器阶段是以顶点信息进行访存,但从光栅化后访存均以像素点信息进行访存。而图形处理器的访存却恰恰集中在光栅化之后,导致了图形处理器的访存数据量大且密集的特点。

1.2 图形处理器传统预取结构

在集成电路高速发展的几十年过程中,集成电路的许多方面都经历了指数级的增长,但是这些增长也并非是所有方面。内存速度方面,由于增长幅

度远低于逻辑等方面,导致内存瓶颈越来越突出,内存延迟和内存带宽成为了限制性能的关键因素之一。

解决访存瓶颈的一个最直接的方法就是增加 cache 这一类的片上缓存[11]。虽然 cache 可以缓解缓存的带宽问题,但是不能解决访存延迟的问题。在当前的图形流水线中,每一个访存的地址都可以根据前级流水线提前得知,且通过保证顺序彼此之间数据依赖度比较低,所以预取结构可以有效缓解访存延迟的问题。

传统的预取结构非常简单且容易实现[12]。当处理像素块时,将访存请求发送到访存模块,同时像素块进入先入先出队列(first input first output, FIFO)中排队。待数据从访存总线返回时,将像素块从 FIFO 中取出进行处理。像素块在 FIFO 中的等待时间就是访存系统的延迟时间,如果 FIFO 大小合适,则不会堵塞流水线[13-14]。

图 2 是图形处理器传统预取的结构图。传统预取当获得需要处理的像素块地址后,会进行 cache 中的标签查找。若 cache 命中,则无需发出预取的访存请求。若cache未命中,则需要将访存请求发送

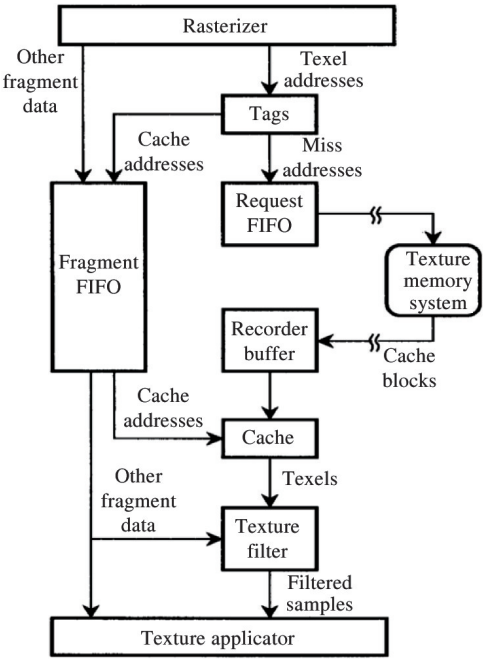

图 2　图形处理器传统预取结构图

到 miss 队列的 FIFO 中。这里与正常访存的 cache miss 请求共用同一队列。在 miss 队列发出预取的访存请求后,等待访存请求返回。内存总线对于请求一般是按序返回,FIFO 可以有效代替重排序缓冲区。但如果访存请求是乱序返回,则需要采用重排序队列对返回请求进行重排序。返回后的数据存入对应的片上缓存 cache 中。待流水级处理到该地址时,便可在片上缓存的 cache 命中,节省了访存的时间,有效利用了内存带宽。但由于图形处理器在处理像素数据时,访存可能是密集的,访存带宽利用率高,访存空闲时间少。在访存密集的情况下,传统的预取机制无法发挥出应有的作用,这样不仅无法有效利用带宽,提高访存性能,反而会给功耗面积等带来更进一步的开销。

2　基于压缩结构的预取结构设计

2.1　图形处理器的压缩结构

图形处理器的访存请求主要集中在流处理器与深度测试和颜色混合模块。流处理器通常拥有两级 cache,L1 cache 中存放解压后的数据,L2 cache 存放压缩数据。深度测试和颜混合模块通常只有一级 cache,存放解压后的数据。

本文基于的图形处理器压缩结构是用与深度和帧缓冲区的无损压缩,对于纹理等有损压缩数据,本结构可以提供旁路操作,在不占用资源的情况下将无需再次压缩的纹理数据直接送至内存控制器中。该压缩结构可直接放置于访存总线上。

图 3 是本文采用的压缩结构的结构图,图中采用了用于存储压缩后的数据格式和 fast clear 压缩标记的 meta cache, meta cache 是一个在显存上拥有独立存储空间的通用 cache。将一个 4 ×4 或 8 × 8 的 tile 的格式标记和 fast clear 标记存储于一个 offset 上。当数据格式标记显示数据被完全压缩成 0 byte 时,说明 fast clear 标记是有效的。这时无需从显存对应的颜色或深度缓冲区读写数据,仅需要从读取 meta data 区域读写数据即可。由于 meta cache 的标记位很短,而且图形应用的读写具有很好的空间局部性,所以 cache 的命中率很高,可以大量节省访存的带宽。

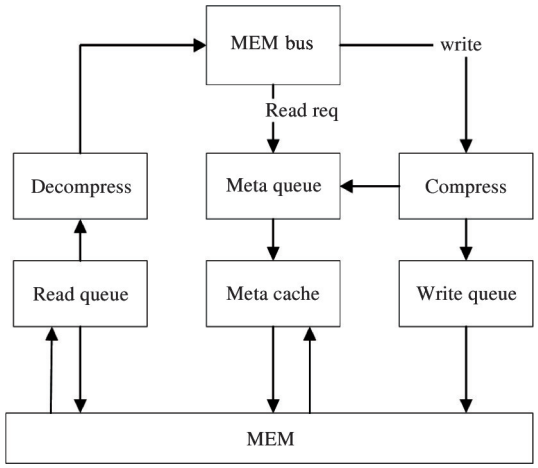

图 3　图形处理器压缩结构图

当总线上发出写请求时,数据会进入压缩模块进行数据的压缩,压缩后将数据的格式写入 meta cache。如果压缩时判定进行 fast clear 操作,则同时将 fast clear 的标志位更新至 meta cache。如果出现 meta cache 缺失的情况,则先存于队列中,待 meta cache 更新后写入。

当总线上发出读请求时,数据会先查询 meta cache。若缺失则进入队列等待 meta cache 更新替换。若命中,则解析数据的格式信息。如果数据解析为 0 byte,则数据为 fast clear 数据,无需再读取显存,直接根据 meta cache 的标记位进行解析即可。否则,需向显存读取压缩数据。读回后,根据数据格式以及压缩算法信息进行对应的解压操作。

这一图形处理器压缩结构可在数据能够以 fast clear 进行压缩时仅写标记存于 meta cache,无需进行总线请求。在这一操作时,即使总线访存密集,也会出现空拍的空闲,给予预取机制进行预取的机会。同时,本图形处理器压缩结构可直接用于总线之上,可作为总线的仲裁,所有模块的访存请求都需通过该压缩结构,也更适用于预取结构。压缩模块的访存队列只需增加状态机的优先级控制即可直接应用于预取结构。

2.2　预取结构

针对图形处理器在处理像素块访存时,由于访存密集导致预取结构的效果不理想的情况,提出了一种基于图形处理器压缩结构的预取机制设计。

图 4 是本预取结构的设计。先从图形处理器的图元处理引擎流水级(PE)或片段着色器流水级(SP)获取预取数据的地址信息。由于在图形流水线中,图元处理模块处于片段着色器之前,而片段着色器又处于输出混合单元之前。图元处理流水级将图元进行光栅化处理,处理后将像素块的位置属性等信息送给片段着色器。所以,光栅化时图元处理流水级明确知道后续流水级需要处理的像素点位置信息,根据位置信息可以计算出后续流水级需要访存的地址信息。

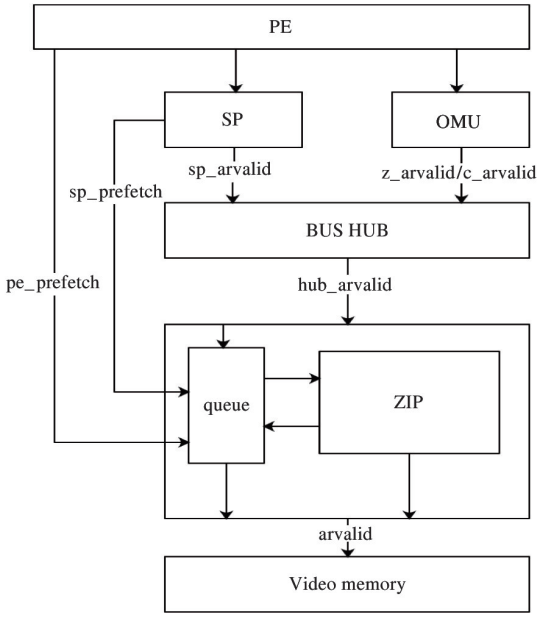

图 4　基于压缩结构的预取结构图

获取到预取数据的地址信息后,将预取请求进行片上缓存的 cache 查找。如果片上缓存 cache 中已经存在该预取地址的数据信息,则无需再进行预取操作,数据已经存在于片上缓存的 cache 中。如果片上缓存 cache 缺失,则说明片上缓存 cache 中没有该预取地址的数据,这时就需要对该地址进行预取操作。

将片上缓存 cache 缺失的预取请求送至位于总线的压缩模块访存队列中。压缩模块的访存队列是一个由状态机控制的队列。队列会对访存请求进行合并,当流水线访存请求发送到队列中时,若队列已经存在预取信息,则进行合并并优先发送请求,以保证不堵塞流水线。若队列中的预取信息已经发出,则在队列中等待预取请求返回即可。

由于图形处理器在光栅化之后的访存可能是密

集型的,而预取机制则需要利用访存空闲时间进行访存,不能影响正常的访存请求。基于图形处理器的压缩结构,可将数据进行压缩得到访存的空闲。压缩结构当数据可以被压缩进 meta cache 时,在读请求时是无需发出访存请求的,只需直接从 meta cache 中读取出对应的标记位。这时,访存总线出现请求的空闲。预取机制可以利用这一压缩结构制造出的访存空闲时间进行预取。压缩结构即使在访存密集型程序下,也有可能将数据压缩进 meta cache 中,所以预取机制依然会起到一定的作用。

基于图形流水线的特点,预取地址是明确且必定需要访存的。所以在压缩结构的访存队列进行合并可以有效提高访存性能。在图形流水线执行至正式访存时,如果地址对应的预取访存已经返回至片上 cache 中,则在 cache 中命中,并直接返回图形处理流水线进行数据处理。如果预取访存还未返回至片上 cache,则按照正常的访存流水线进入压缩结构的访存队列中。这时,如果访存队列中地址对应的预取信息已经发出访存请求,但数据还未返回,则进行合并,等待访存请求的返回。这样,预取机制可有效减少访存延迟。如果访存队列中对应的预取信息还未发出访存请求,则进行合并后将队列状态机由原预取状态更改为访存状态,优先级进行了提升,立刻发出访存请求。这样的处理,也使得预取机制没有起到作用时也并未浪费性能。

3 实验结果及分析

为了评估本预取结构的性能及效果,本文采用 Godson GPU(GSGPU)高性能图形处理器[15],GSGPU 图形处理器实现了本设计提及的压缩结构。结合了 GSGPU 高性能图形处理器基于 mesa 架构的驱动设计[16],采用了 Linux 基准图形测试集 GL _ MARK[17]。对照组选择了采用传统预取机制的 GSGPU 高性能图形处理器,根据 GSGPU 图形处理器片上缓存 cache 的命中率对预取结构的效果进行评估。

3.1 图形测试集

本文实验测试采用的基准测试集是 GL _ MARK。GL _ mark 是由 Linaro 发行的一款图形基准测试集,使用 Open GL-ES 进行开发,提供了一系列丰富的图形测试。涉及图形单元性能的各个方面,涵盖光照、阴影、超多图元、简单 2D 等多种类型的测试,是目前 Linux 操作系统上较为全面的测试集之一。

3.2 图形处理器架构

本文使用的 GSGPU 高性能图形处理器平台主要由命令处理器(command processor,CP)、全局任务调度器(global task scheduler,GTS)、图形处理集群(graphics processing cluster,GPC)、二级静态缓存(L2 Scache)和内存控制器(memory controller,MC)等 5 部分组成。其中图形处理集群又由计算处理引擎(compute engine,CE)、几何处理引擎(geometry engine,GE)、图元处理引擎(PE)、局部任务调度器(local task scheduler,LTS)、流处理器集群(stream processor cluster,SPC)和输出合并单元(OMU)等 6 部分组成。整体结构如图 5 所示。

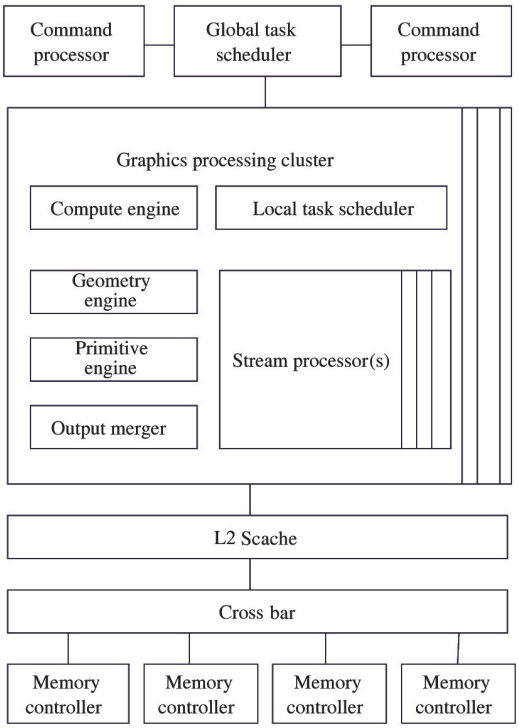

图 5 GSGPU 图形处理器顶层结构

3.3 预取结构的性能分析

搭载本设计的 GSGPU 高性能图形处理器光栅化后的访存基本单位是 4 ×4 的像素块,所以在设计 cache 行时选择了 4 ×4 的像素块作为 cache 行宽

度。经过性能与面积功耗权衡后，选取四路组相连 cache，容量大小选择 32 kB。

本设计跑通 GL _ MARK 中的所有基准测试用例，测试用例中包含了阴影测试、折射测试、反射测试、2D 少图元测试，超多图元测试、复杂 shader 测试等多种测试环境。测试结果如表 1 所示。

传统预取在访存密集情况下，预取性能会出现急剧下降，预取效果不明显。采用本设计的预取结构，结合压缩结构特性，即使在访存密集的情况下，依然可以利用压缩产生访存空闲，使预取结构产生良好的效果。表 1 是本设计与传统预取结构的 cache 命中率提升比较。从表 1 中可以看出，在 GL _ MARK 基准测试集的测试环境下，本设计的 cache 命中率对比传统预取结构均有所提升，说明本设计的预取结构在 GL _ MARK 测试集下均有不错的表现，可以起到预取的效果，且本设计的预取结构不会对传统预取结构产生不利影响。

表 1　GL _ MARK 测试集下本设计与传统预取结构的 cache 命中率提升

测试集	访存请求/个	Cache 命中率
Horse _ shadow	120 402	+4.28%
Cat _ notex	49 648	+8.16%
Buffer	74 076	+14.04%
Bump	24 368	+5.70%
Conditionals	107 372	+17.46%
Effect _ 2D	8070	+0.66%
Function	107 078	+17.32%
Loop	107 414	+17.40%
Jellyfish	316 088	+17.42%

在访存总线有空闲的情况下，传统预取结构也可以取得良好的性能提高，从而提高 cache 的命中率，也具有良好的效果。但传统预取结构在访存密集型程序下，访存总线始终被占用，没有访存的空拍情况下，传统预取结构的效果明显不足。这时，本设计的基于压缩结构的预取机制可以利用压缩结构的特点制造出访存的空拍，使预取结构可以利用该空拍进行预取，得到了 cache 命中率的明显提升，达到很好的预取效果。

但有些访存密集型程序，例如 Horse _ shadow，由于压缩结构无法取得很好的无损压缩效果，所以无法制造出更多访存的空拍，产生更多的访存空闲，所以导致本设计的预取效率的提升不明显。可针对特定的测试用例，增加无损压缩结构的压缩标记，提高压缩概率，更进一步增加访存空拍，使本设计的预取结构有所提高。

从表 1 中可以看出，本设计在访存密集型程序下，cache 命中率大部分均可提升达到 15% 以上。个别测试用例在本设计中提升不高，是受限于颜色及深度的场景对于无损压缩不敏感。由于图形处理器的预取地址是明确的，后续流水线对于预取地址必然需要访存操作，所以不存在预取无效数据浪费访存的性能及功耗的情况，不会对本身图形流水线产生负面影响。本设计在压缩结构的访存队列中采用了合并操作，所以本设计即使在访存没有空闲无法预取的情况下，也不会多占用访存总线的带宽。

图 6 是在 GL _ MARK 测试集下访存数量与 cache 命中率的提升比较。从图 6 中可以看出，本设计随着访存数量的增加，对比传统预取结构的提升基本成正比。访存数量越多，访存越密集，本设计的预取结构对比传统预取结构优势越明显。

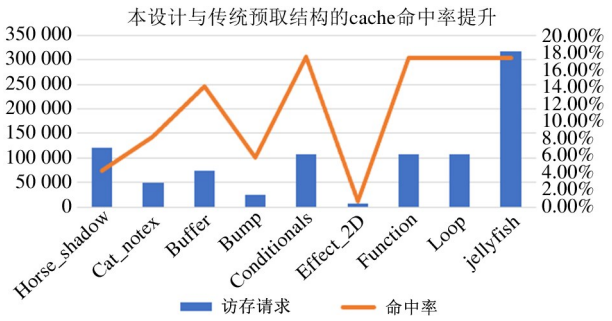

图 6　GL _ MARK 测试集下访存请求数量与 cache 命中率提升比较

4　结　论

本文分析了图形处理器传统预取结构在图形流水线中的特性，结合图形应用程序的特性，提出了一种基于图形处理器压缩结构的预取结构设计。针对访存密集型程序访存空闲时间少、导致传统预取结构预取效果不明显的情况，本设计利用图形处理器压缩结构的特点，重新设计了基于压缩结构的预取结构。本设计与传统的预取结构相比，在访存密集

型程序下,不仅可以有效提高访存的利用率,同时也使图形流水线的处理时间得到有效的缩短。即使在预取失效的情况下,对图形流水线也不会产生负面影响。图形处理器先进的关键技术长期掌握在国外厂商手中,国产自主可控处理器的研发变得越来越重要,本设计不仅为设计先进国产图形处理器访存子系统结构提供了方向,同时也为提高图形处理器访存子系统性能提供了方法与借鉴。

参考文献

[1] McCLANAHAN C. History and Evolution of GPU Architecture[R]. Atlanta: College of Computing, Georgia Institute of Technology, 2010

[2] AKENINE-MOLLER T, HAINES E, HOFFMAN N. Real-Time Rendering[M]. Natick/Boca Raton: A K Peters/CRC Press, 2019

[3] AMD Corporation. RDNA architecture presentation[EB/OL]. https://gpuopen.com/wpcontect/uploads/2019/08/RDNA_Architecture_public.pdf: AMD, (2019-05-23),[2020-12-01]

[4] AMD Corporation. RDNA 1.0 instruction set architecture [EB/OL]. https://gpuopen.com/wp-content/uploads/2019/08/RDNA_Shader_ISA_5August2019.pdf: AMD, (2019-07-07),[2020-12-01]

[5] AMD Corporation. RDNA architecture whitepaper[EB/OL]. http://www.amd.com/system/files/documents/rdna_whitepaper.pdf: AMD, (2019-05-23),[2020-12-01]

[6] LINDHOLM E. NVIDIA Tesla: a unified graphics and computing architecture[J]. *IEEE Micro*, 2008, 28(2): 39-55

[7] HESTNESS J, KECKLER S W, WOOD D A. A comparative analysis of microarchitecture effects on CPU and GPU memory system behavior[C]// IEEE International Symposium on Workload Characterization, Raleigh, USA, 2014: 150-160

[8] KIM G S, LEE M, JEONG J, et al. Multi-GPU system design with memory networks[C]// IEEE/ACM International Symposium on Microarchitecture, Cambridge, UK, 2014: 484-495

[9] 卢俊,颜哲,田泽,等. 一种高效 GPU 存储系统体系架构设计[J]. 计算机技术与发展, 2015, 25(4):6-9

[10] 韩立敏, 田泽, 张骏, 等. 图形处理器流水线数据压缩技术研究综述[J]. 计算机应用研究, 2018,35(3): 648-653

[11] HAKURA Z S, GUPTA A. The design and analysis of a cache architecture for texture mapping[C]// Proceedings of the 24th International Symposium on Computer Architecture, Denver, USA, 1997: 108-120

[12] LGEHY H, ELDRIDGE M, PROUDFOOT K. Prefetching in a texture cache architecture[C]// Workshop on Graphics Hardware, Lisbon, Portugal, 1998: 133-142

[13] ANDERSON B, MACAULAY R, STEWART A, et al. Accommodating memory latency in a low-cost rasterizer [C]// Proceedings of the 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware, Los Angeles, USA, 1997: 97-102

[14] KILGARD M J. Realizing OpenGL: two implementations of one architecture[C]// Proceedings of the 1997 SIGGRAPH/ Eurographics Workshop on Graphics Hardware, Los Angeles, USA, 1997: 45-56

[15] 张立志, 赵士彭, 赵皓宇, 等. 高性能 GPU 模拟器的实现[J]. 高技术通讯, 2020,30(6): 553-560

[16] 赵士彭, 张立志, 赵皓宇, 等. 高性能 GPU 模拟器驱动设计研究[J]. 高技术通讯, 2020,30(5):435-442

[17] GitHub Inc. glmark2[EB/OL]. https://github.com/glmark2/glmark2: GitHub, (2017),[2020-12-01]

A prefetch architecture design based on graphics processor compression architecture

ZHAO Shipeng[*,**,***], ZHANG Lizhi[*,**,***], ZHANG Longbing[*,**,***]

([*] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Graphics processing unit (GPU) memory access utilization has become one of the key bottlenecks affecting performance. In processor design, memory access prefetch architecture design has become one of the main methods to improve memory access utilization. Combined with graphics processor memory access, due to the dense features, under the premise of improving the prefetch performance, reducing the influence on the normal efficiency of the graphics pipeline has become a popular research direction. Based on a graphics processor lossless compression architecture, this paper proposes a set of graphics processor prefetch architecture design. The design of the prefetch architecture can effectively improve the memory access utilization in the memory-intensive graphics pipeline, and does not affect the efficiency of the current graphics pipeline. The experimental results show that on the Godson graphic processing unit (GSGPU) graphics processor platform, compared with the traditional prefetch architecture, the cache hit rate can be increased by more than 15% for the memory-intensive test program. For the test program with idle memory, it will not have a negative impact on the pipeline.

Key words: graphic processing unit (GPU), memory access subsystem, prefetch architecture, compressed architecture