高技术通讯 2022 年 第 32 卷 第 11 期:1126-1133

doi:10.3772/j.issn.1002-0470.2022.11.003

基于图元光栅化触发的高效 GPU 深度数据预取①

田 泽② 张 骏③ 许宏杰

(西安翔腾微电子科技有限公司 西安 710068)

(航空工业西安航空计算技术研究所 西安 710068)

(集成电路与微系统设计航空科技重点实验室 西安 710068)

摘 要 数据预取技术已经广泛应用在各类中央处理器(CPU)设计领域,取得了很好的效果。而图形处理器(GPU)对存储带宽的需求更为巨大,与图形渲染流水线和图形算法直接相关,且数据访问模式与通用 CPU 有显著差异,需要更有针对性的有效数据预取机制。针对 GPU 深度测试关键功能,本文提出一种图元光栅化触发的高效深度数据预取机制——DPRT,通过图元光栅化过程中实时扫描到的片段块地址来触发 Z 缓存(Z Cache)的深度数据预取,同时为了适应不同实现中流水线处理延迟,为 Z Cache 数据块增加访问一次标志(OTT),保证深度数据预取有效性。实验结果表明,DPRT 使深度测试时 Z Cache 访问命中率平均提升9.51%,深度测试延迟平均降低 40.43%。

关键词 图形处理器(GPU);光栅化;扫描;深度测试

0 引 言

3D 图形处理与 2D 图形处理最显著的区别在于引入了深度的概念,使绘制结果产生了有层次的空间立体真实感受。3D 几何图形绘制渲染过程中的一个重要步骤就是深度测试。帧缓冲区的颜色缓冲区存储像素的颜色值,深度缓冲区存储像素的深度值,深度测试将像素的深度值与当前深度缓冲区中对应的值进行比较,如果大于深度缓冲区的值,则丢弃该部分的像素,否则利用这个像素对应的深度值和颜色值分别更新深度缓冲区和颜色缓冲区,这样就实现了判断该像素是否需要绘制的问题。

深度测试能避免对最终不会看到的像素点的绘制,从而提高图形流水线的执行效率,其功能是在图形流水线中的片段处理阶段实现的,在光栅化阶段之后,对光栅化输出的像素的深度数据进行比较测试,是图形处理器(graphic processing unit,GPU)像素填充率性能指标的重要影响因素。

图元光栅化(rasterization)将连续方式描述的几何图元(点、线、离散三角形和扇带)的顶点数据转换为由离散片元(fragment)组成的二维图像。片元是最初级的像素,每个片元包括颜色值(RGBA)、深度值(Z)、二维屏幕坐标(X、Y)等信息,与帧缓冲中的一个像素对应。在 Nvidia、AMD 等主流图形处理器普遍采用的 IMR(immediately mode rendering)架构[1]中,为了最终能够正确显示位置重叠区域图形距离观察者最近的图元像素颜色,每一个像素的深度数据 Z 都需要在片上进行缓存,以便进行实时深度测试来分辨当前正在绘制像素的深度值与已经绘制完成的像素深度值之间的前后遮挡关系。深度测试频繁的深度数据读写操作不但需要的存储器数据带宽大,而且还在很大程度上决定了 GPU 的像素填充性能。为了降低深度测试对片内存储资源和显存

① 核高基重大专项(2016ZX01012101-004)资助。

② 男,1965 年生,博士,研究员;研究方向:SoC 设计方法学,航空专用集成电路设计;E-mail:tarmz@163.com。

③ 通信作者,E-mail:77664999@qq.com。

(收稿日期:2021-11-06)

带宽的需求,文献[2]提出的 Early-Z 技术在进行像素染色前就对像素的可见性以一种粗粒度的方式进行提前测试。据 ATI 的测试结果,采用 Early-Z 技术进行提前深度测试能够提前捕捉到 50% 以上最终不会通过深度测试的像素点。即使这样,目前 GPU 像素填充性能普遍超过 30 G pixel/s,所需要的深度测试能力依然对 GPU 片上 Z 缓存(Z Cache)设计提出了很高的要求。

数据预取是一种提升 cache 命中率的技术,中央处理器(central processing unit, CPU) Cache 数据预取由来已久,但 CPU 数据访问模式随着应用程序行为的变化而变化,表现为线性、跳跃和随机各种模式,具有不确定性,且一次数据访问量较少,只有 4 ~ 16 bytes,导致 CPU 数据预取效果受制于空间局部性和时间局部性因素在一些场景下不够理想。相比之下,GPU 普遍采用高度并行数据处理架构,通常能够一次并行实现 32 ~ 128 个像素的深度测试处理,单次 Cache 数据访问量达到 128 ~ 512 bytes,为预取数据使用的连续性和有效性奠定了基础。并且,在图形渲染过程中,GPU 的颜色和深度数据访问模式在局部范围相对集中和确定,颜色和深度数据在 Cache 中是局部二维排布,与几何图形光栅化后最终在帧缓冲区中的二维颜色和深度数据的排布精确匹配,读入Z Cache中数据使用率通常能够达到 95% 以上,空间局部性捕获效果很好,为数据预取准确性提供了很好的前提。

GPU 深度测试对片外帧缓冲存储器数据访问量较大,在数据的空间局部性强而时间局部性弱的情况下,必定周期性地发生深度 Cache 缺失,导致处理延迟增大。如果能根据几何图形光栅化实时位置及像素 Tile 的屏幕坐标提前访问外部帧缓冲存储器,就可以及时、精确地实施 GPU 深度 Cache 数据预取,有效避免周期性深度 Cache 缺失造成的额外延迟,提升图形渲染效率。像素数据需要 GPU 的光栅化单元、像素着色器以及光栅操作单元对其依次执行不同的操作。如果 GPU 的光栅化单元能将当前被光栅化像素 Tile(由 n 个空间上相邻的片元组成)的屏幕坐标(x,y)提前发送给深度测试单元,则深度测试单元能够依据该信息提前实施精确的预取操作。

本文提出一种图元光栅化触发的图形处理器高效深度数据预取机制(depth data pre-fetching based on primitive rasterizing triggering, DPRT),通过图元光栅化过程中实时扫描到的片段块地址来触发 Z Cache的深度数据预取。同时为了适应不同实现中流水线处理延迟,保证数据预取有效性,为 Z Cache 数据块增加访问一次标志,有效提升了深度测试Z Cache访问命中率,降低了深度测试延迟。

1　相关研究

GPU 研制方面,国外 Nvidia、AMD(ATI)、ARM、Vivante 和 Imagination[3-8],已经形成了完整的技术体系和产品体系,具有完整的配套图形库软件、驱动软件、编译软件和相关专用知识产权等相关设计资源,而且具有强大的技术创新能力。在国内,南京航空航天大学、电子科技大学、山东大学基于 OpenGL ES 设计了嵌入式图形处理器或系统[9-11]。中国科学技术大学设计了一种面向移动设备的 3D 图形处理器[12]。华南理工大学使用 SystemC 设计了图形处理器模型[13]。哈尔滨工业大学、上海交通大学、华东师范大学、北大众志微处理器研究中心、长沙景嘉微电子公司和航空工业计算技术研究所[14-17]也对嵌入式 GPU 进行了深入研究和设计。

预取技术以往主要应用于微处理器的指令和数据,研究人员针对不同类型应用程序的存储器访问特征以及指令和数据在内存中的不同排布模式,提出了多种有效的顺序或非顺序预取策略。如针对指令预取的面向程序连续执行的 Next-Line 指令预取策略、面向程序分支的 Target-Line 指令预取策略[18]、BTA 分支目标地址预取策略[19]和硬件 Markov 预取[20]。针对数据预取,Jouppi[21]提出了流缓冲区的顺序预取机制,Palacharla 等人[22]提出步长检测预取机制,Sherwood 等人[23]提出面向指针密集型的马尔可夫与步长预取结合的数据预取策略,Roth 等人[24]提出了基于跳转指针的数据预取。另外,研究人员还提出了指令和数据的主动推送技术[25-26],与预取技术由 CPU 内核或 Cache 向更底层存储器

发出预取请求,然后逐级向下传递请求,最终由更底层存储器响应请求并返回数据不同,主动推送技术是由主动推送部件对内核将要使用指令和数据的预测发出访存请求,最终将数据向上推送到 Cache 或处理器内核中,时效性更好。

这些预取和主动推送策略已经广泛应用在各类 CPU 设计领域,能有效降低 Cache 缺失率。而 GPU 对于各类图形数据和存储带宽的需求更为巨大,且数据访问与图形绘制过程直接相关,数据访问模式与通用 CPU 应用数据访问模式差异较大,需要更有针对性和有效的数据预取机制。然而,在 GPU 设计领域,目前没有针对 GPU 深度测试和深度数据预取机制的相关研究或报道。本文提出的 GPU 深度数据预取机制基于自主设计的光栅化算法过程中的自定义关键时间节点作为深度 Cache 的触发信号进行数据预取,其预取有效性和准确性好,能够取得较好效果。

2 光栅化触发的深度数据预取

GPU 架构的发展经历了 2 个阶段,即分离染色架构和统一染色架构。无论物理架构是分离染色的还是统一染色的,在逻辑上都遵循一套完整的图形处理流程,在实现时映射为图形处理流水线。一般地,GPU 图形渲染流水线需要依次完成顶点着色、几何图元处理、图元光栅化、像素着色,经过片段处理后最终写入帧缓冲区。图元(primitive)是顶点的集合,组成一个 3D 实体。常见的 3D 图元包括点、线、离散三角形和扇带。图元光栅化过程就是把连续的以数学方式描述的几何图元映射为离散的屏幕上像素点的过程,主要涉及扫描转换、像素插值、反走样等算法及其硬件实现。与本文相关的主要是扫描转换过程。图元扫描转换是通过遍历操作,逐个或批量判定出位于图元内部所有有效的像素点。光栅化处理的基本图元包括点、线和三角形图元。无论哪种图元,输入到光栅化阶段的图元顶点坐标都是浮点数表示,是连续的,但是屏幕上的像素是离散的,扫描转换算法的主要任务是如何把连续的图元用最接近的离散的栅格表示出来。对于点的扫描转换,可直接对其实数坐标进行四舍五入,得到最接近该点的离散坐标位置;线图元的 2 种常用扫描转换算法是数字微分分析仪(digital differential analyzer, DDA)算法和 Bresenham 算法[27]。三角形图元的扫描转换过程对 GPU 几何性能影响最大,相对于点和线图元来说更加重要,包括扫描线算法(Scanline)、包围盒算法(Bounding Box)和中心线扫描算法(Centerline)[28],遍历的方式决定了三角形扫描的效率,也很大程度上确定了后续深度测试过程中对深度数据的使用顺序。

对于 GPU 中深度测试功能来说,像素片段的深度数据由图元光栅化阶段产生,并在片段处理阶段根据不同的深度测试函数将每个像素片段的深度与帧缓冲区中对应位置像素的深度数据进行比较,最终决定该像素片段是否应该被写入帧缓冲区,即光栅化处理在前,深度测试在后。在深度测试功能开启的情况下,所有光栅化产生的像素片段必然都要进行深度测试(本文不考虑 Early-Z 等类似提前深度测试技术带来的层次化深度测试效果)。根据图形处理流水线各阶段先后关系以及光栅化和深度测试的这种特性,本文提出光栅化触发的图形处理器高效深度数据预取机制 DPRT,希望能够根据实际光栅化区域所覆盖帧缓冲区像素区域,在片段处理单元进行深度测试前,提前将对应像素深度数据预取到 Z Cache 中,从而达到隐藏存储器访问延迟的目的。

2.1 DPRT 深度数据预取机制

本文采用基于 Tile(4 × 4 像素块)的扫描线算法进行三角形图元扫描转换。在此基础上,DPRT 深度数据预取的工作机制如图 1 所示。

图元经过几何阶段处理后,被送入光栅化单元进行从图元到像素片段的转换,输出的像素片段格式可以根据不同设计进行选择,通常选择以像素片段块(Tile)为单位进行扫描输出,这样不但能够提升光栅化效率,而且更加符合 Z Cache 的数据缓冲特性,访存深度缓冲区效率也较高。光栅化每产生一个 Tile,就根据 Tile 在深度帧缓冲区中的地址形成一个深度数据预取请求发送到 Z Cache,Z Cache 收到预取请求后向外部显示存储器发出深度数据

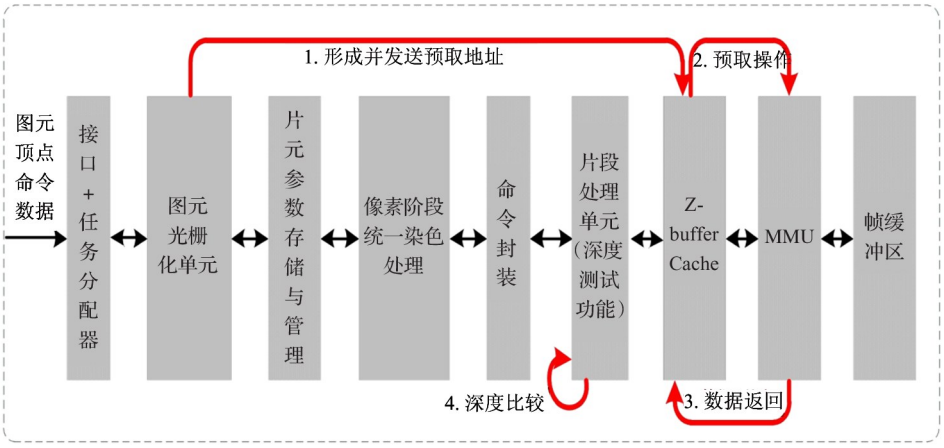

图 1　光栅化触发的深度数据预取机制

访问请求,预取的深度数据通常会在其对应的像素片段 Tile 完成像素染色阶段处理和其他片段处理前进入 Z Cache 中,从而实现隐藏像素片段 Tile 深度数据访问延迟的目的。

在不考虑 Early-Z 技术带来的层次化深度测试效果前提下,所有光栅化单元产生的像素片段 Tile 都必然要在片段处理单元从 Z Cache 中取出深度数据进行深度测试。对 Z Cache TAG 数据结构进行优化,为每一个 Cache 数据块 Tag 域增加一个访问一次标志(once touching tag,OTT),如图 2 所示。

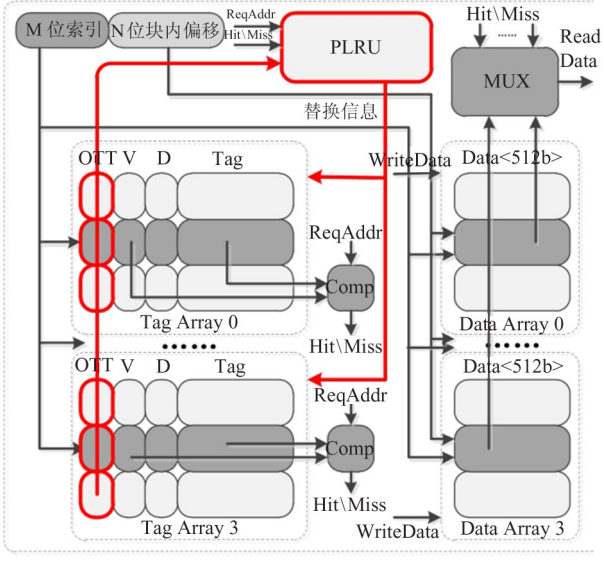

图 2　Z Cache TAG 数据结构优化

预取的深度数据进入 Z Cache 后,未被读取之前置 OTT = 0;当 Z Cache 中的深度数据被读取一次后,置 OTT = 1。每次需要进行 Z Cache 替换时,基于 PLRU 算法优先选择已经被访问的 Cache 数据块(OTT = 1,已经完成深度测试需要的深度数据访问)进行替换,保留 OTT = 0 的 Cache 数据块,从而实现在访存效率较高的情况下,保证预取回 Z Cache 中深度数据不会在使用前就被替换出去,降低了 Z Cache 抖动概率和访问缺失率。

2.2　DPRT 工作流程

图 3 以 4 路组相连 Z Cache 结构说明了 DPRT 机制的工作流程。光栅化模块接收图元光栅化任务后,开始扫描并将当前像素 Tile 的坐标地址发送给 Z Cache 模块,触发深度预取动作。深度地址和请求产生模块将深度数据预取请求发送给仲裁模块。仲裁模块首先要测试本次预取的深度数据是否已经存在于 Z Cache 中,如果已经存在则撤销本次预取;否则就在通过第二级仲裁的情况下,基于 PLRU 替换算法和 OTT 标志位优先选择最近访问次数最少且 OTT = 1 的 Cache 数据块进行替换,并将脏块写回 DDR,将深度预取请求送给存储管理单元(memory management unit,MMU),访问外部显示存储器中的深度缓冲区。由于前期已经为不命中数据准备好了空 Cache 块,当外部显示存储器返回预取的深度数据后,依次查找当前组中的空行,直至将该数据写入 Z Cache DATA ARRAY 中,同时更新 Cache TAG。

2.3　DPRT 有效性分析

数据预取有效性包括 2 个关键要素:时效性和准确性。时效性是指数据既要在访问请求到达前到

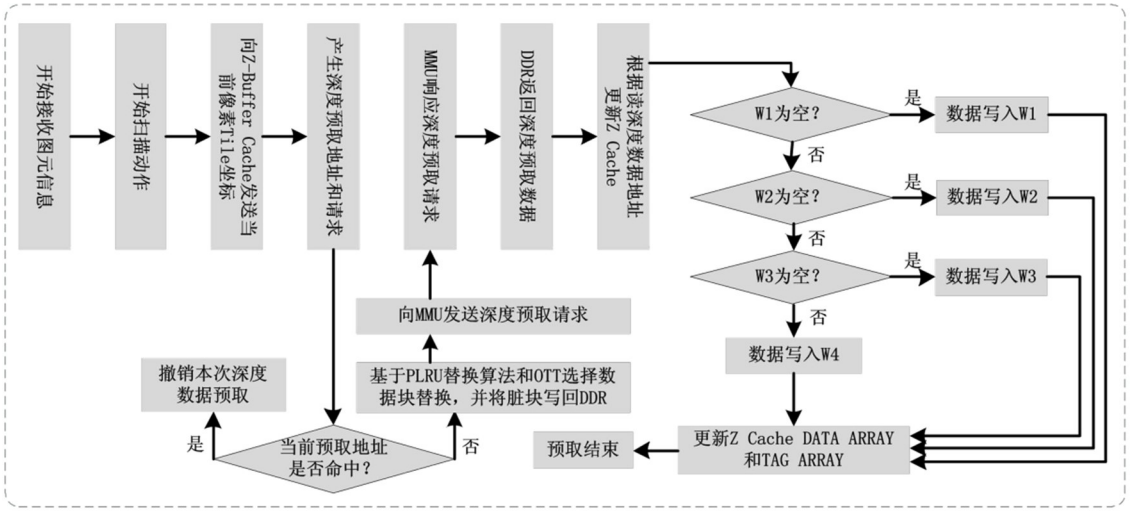

图 3　DPRT 工作流程

达 Cache 中,同时还不能被过早替换出去。

首先,由于进行深度测试前像素片段 Tile 还需要在统一染色阵列上完成像素染色阶段的处理和其他片段处理功能(混合、逻辑操作等),延迟较大,可以认为形成预取地址到预取完成间的延迟与形成预取地址到使用深度数据间的延迟匹配。从光栅化、像素染色和外部显示存储器访问延迟来看,深度数据能够在进行深度测试前被预取进 Z Cache 中。

其次,还需要考察预取的深度数据是否能够不被过早地替换出去。DPRT 预取方案中,预取动作的触发条件是开始光栅化扫描或者扫描步进,而光栅化扫描持续步进的条件是以前光栅化输出的像素片段 Tile 都完成像素染色阶段处理,并发送给片段处理模块。而片段处理模块完成像素片段 Tile 处理(包括深度测试功能)前会阻塞光栅化扫描步进过程。也就是说,预取操作存入 Z Cache 中的深度数据在使用完成前,不会过早发生扫描线步进动作,那么下一次深度预取就不会开始,即 Z Cache 中的深度数据不会被过早替换出去。另外,为了防止访存延迟的变化和不同类型图元导致的预取地址特征的变化,专门在 Cache Tag 域中加入了 OTT 标志位,从而能够优先选择最近访问次数最少且已经被访问至少一次的 Cache 数据块进行替换,尽量保证未被使用过的深度数据不会被过早替换出去。

对于深度数据预取来说,准确性是指在尽量多的覆盖确定需要的数据的前提下,要尽量少地预取非图元覆盖区域的像素深度数据。一次预取包括少量非图元覆盖区域的像素深度数据是难免的,也是被允许的,这些数据可以作为其他图元的深度预取数据来使用,但其所占比例不能太大,否则不但浪费外部显示存储器带宽,而且容易造成 Z Cache 的频繁抖动。

预取的特性决定了会取回一些当前用不到的数据,但只要比例不大并且能够覆盖到本次需要使用到的数据即可。其他多余的数据可能会在绘制其他图形的时候被使用到,从而提升了外部显示存储器的带宽利用率,并且不会造成 Z Cache 的抖动。

3　测试与评估

采用基于现场可编程门阵列(field programmable gate array, FPGA)构成的原型系统进行验证,包括基于 FPGA 和一系列外围的接口子卡,如 PCIe 2.0 x16 子卡、VGA 显示子卡和外部显示存储器存储子卡等,如图 4所示。

将本文提出的 DPRT 深度数据预取机制进行 FPGA 原型系统测试与评估。光栅化单元以 4 × 4 的像素片段 Tile 为单位进行扫描输出,Z Cache 容量为 32 kB,4 路组相连,能够容纳 8192 个像素点的深度数据。进行 FPGA 综合和布线,工作频率可达 100 MHz。

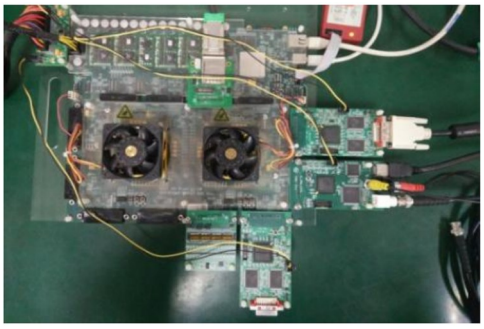

图 4　FPGA 原型系统

由于 OpenGL 协议的官网只提供针对性能的测试程序和面向 API 功能的符合性测试程序,为了全面精准地评测深度测试的功能和性能,本文基于 FPGA 平台,在深度测试功能打开的情况下,选择 4 个典型 3D 游戏场景作为测试程序[29],测试场景见图 5。这些测试场景所绘制的图元类型不同,图元个数不同,场景中的图元深度值、深度比较函数的设置也不同,深度测试阶段的数据访问模式也不同,能较为全面地覆盖深度测试功能的各种测试情形。

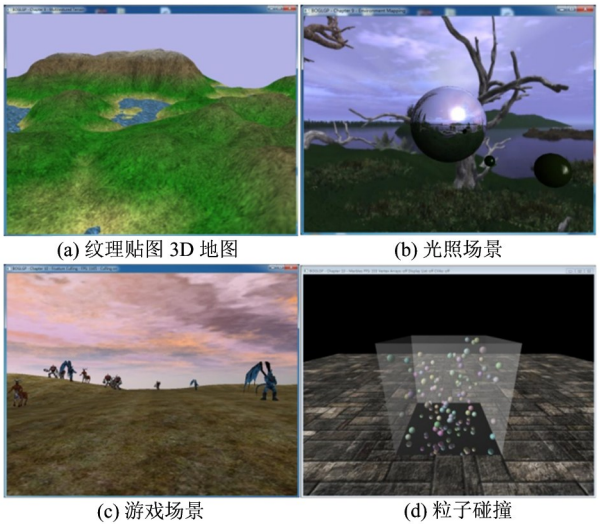

图 5　深度测试场景

本文从 Z Cache 平均命中率性能和深度测试平均延迟性能 2 个方面测试和评估使用 DPRT 机制的设计和没有使用 DPRT 机制的设计(GPU BASE)。其中,GPU BASE 是包括了完整 3D 图形绘制流水线,但没有添加 DPRT 机制的自主研制 GPU 结构,具体包括:顶点处理阶段、几何处理阶段、光栅化阶段和像素处理阶段。DPRT 机制的关键是在光栅化模块与像素处理阶段的深度 Cache 之间增加预取触发控制通路,并为深度 Cache 增加数据预取功能。

3.1　Z Cache 命中率性能评估

相对于 CPU 来说,GPU 图形算法硬件实现虽然复杂,但有针对性的算法硬件优化却能够取得较好的效果。相对于 GPU BASE 结构来说,使用 DPRT 机制后,4 个测试场景的 Z Cache 命中率均有显著提升,如图 6 所示。场景 2 的 Z Cache 命中率提升最高,达到11.74%,场景 4 的 Z Cache 命中率提升最低,达到 7.82%,平均提升 9.51%。

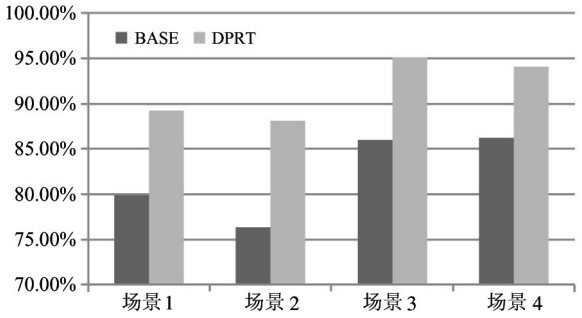

图 6　DPRT 机制对 Z Cache 命中率的提升情况

3.2　深度测试延迟性能评估

在开启深度测试的情况下,对 4 个场景的深度测试延迟情况进行了统计。由于 Z Cache 命中率的提升,相对于 GPU BASE 结构来说,使用 DPRT 机制后,4 个场景的深度测试延迟均有显著降低。如图 7 所示,场景 3 的深度测试延迟降低最多,达到 39.87%,场景 1 的深度测试延迟降低最少,达到 30.83%,平均延迟降低 40.43%。

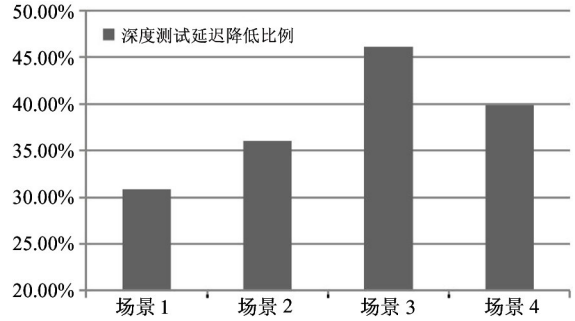

图 7　DPRT 机制对深度测试延迟的降低情况

4　结　论

深度测试是 GPU 的关键功能,不但对存储带宽

需求很大,而且决定了 GPU 的像素填充率性能。针对 GPU 图形渲染流水线的结构特性和深度测试功能的访存行为特征,本文提出一种图元光栅化触发的高效深度数据预取机制 DPRT,使用图元光栅化过程中实时扫描到的像素片段块地址来触发 Z Cache 的深度数据预取,并为 Z Cache 数据块 Tag 域增加访问一次标志,保证了深度数据预取有效性。DPRT 使深度测试时 Z Cache 访问命中率平均提升 9.51%,深度测试延迟平均降低 40.43%,取得了较好的效果。

考虑到不同的 GPU 硬件实现中像素染色处理和片段处理单元的处理延迟以及显示存储器的访问延迟可能存在差异,未来可以进一步研究深度数据预取时机与上述几方面处理延迟间的动态自适应机制,使得预取数据进入 Z Cache 的时机与对应像素片段 Tile 的处理延迟更加匹配,从而进一步降低 Z Cache 抖动概率和深度测试延迟。

参考文献

[1] WONG T H. A comparison of graphics performance of tile based and traditional rendering architectures[D]. Melbourne: Royal Melbourne Institute of Technology Melbourne,2010:10-24

[2] KIM H Y, YU C H KIM L S. A memory-efficient unified early Z-test[J]. *IEEE Transactions on Visualization and Computer Graphics*, 2011, 17(9):1286-94

[3] WITTENBRINK C M, KILGARIFF E, PRABHU A. Fermi GF100 GPU architecture[J]. *IEEE Micro*, 2011, 31(2): 50-59

[4] LINDHOLM, ERIK, NICKOLLS, et al. Nvidia Tesla: a unified graphics and computing architecture.[J]. *IEEE Micro*, 2008,28(2):39-55

[5] AMD Corparation. ATI Radeon™ HD 5450 user guide [EB/OL]. http://cdn-docs.av-iq.com/other/ATI_Radeon_HD_5450_Guide.pdf: AMD,(2010-11-01),[2021-09-06]

[6] OLSON T. Mali-400 MP: a scalable GPU for moblie device [EB/OL]. https://www.highperformancegraphics.org/previous/www_2010/media/Hot3D/HPG2010_Hot3D_ARM.pdf: ARM,(2012-01-02),[2021-09-06]

[7] VERISILCON LTD. Vivante graphics cores[EB/OL]. https://www.techylib.com/en/view/sixmileugliest/vivante_graphics_cores:TechLib,(2012-01-01),[2021-09-06]

[8] Imagination. PowerVR graphics processors imagination [EB/OL]. https://www.imaginationtech.com/graphics-processors/:Imagination, (2021-01-10),[2021-09-06]

[9] 范彩霞. 基于 ARM + FPGA 的嵌入式图形处理系统设计[D]. 南京:南京航空航天大学信息科学与技术学院,2009:29-51

[10] 熊霖峰. 嵌入式图形加速器的几何处理引擎设计与实现[D]. 成都:电子科技大学计算机科学与工程学院,2009:34-85

[11] 杨国东. 嵌入式图形处理器的研究与实现[D]. 济南:山东大学微电子学院,2010:41-90

[12] 杨毅,郭立,史鸿声,等. 面向移动设备的 3D 图形处理器设计[J]. 小型微型计算机系统,2009,30(8):1668-1674

[13] 黄伟钿. 面向移动平台的 3D 图形处理器的设计[D]. 广州: 华南理工大学电子与信息学院,2011:24-46

[14] 王旭. 图形处理器的仿真验证[D]. 哈尔滨: 哈尔滨工业大学微电子中心,2007:18-40

[15] 韩伟. 图形处理芯片的研究与设计验证[D]. 上海: 上海交通大学微电子学院,2008:21-77

[16] 姜宁. 嵌入式硬件图形加速器的研究与设计[D]. 上海:华东师范大学信息学院,2006:20-60

[17] 田泽,刘天江,张骏,许宏杰、黎小玉. 基于扫描线填充的三角形图元双向光栅化技术[J]. 小型微型计算机系统,2015,36(6):1398-1402

[18] SMITH J E, HSU W C. Prefetching in supercomputer instruction caches. [C]//Proceedings of the 1992 ACM/IEEE conference on Supercomputing, Minneapolis, USA, 1992:588-597

[19] GADE P R, PAILY R, HA Y. A branch target instruction prefetching technique for improved performance. [C]//The 15th International Conference on Advanced Computing and Communications, Guwahati, India, 2007:345-351

[20] JOSEPH D, GRUNWALD D. Prefetching using Markov predictors[J]. *IEEE Transactions on Computers*, 1999, 48(2):121-133

[21] JOUPPI N P. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffer[C]//The 17th Annual International Symposia

on Computer Architecture, Seattle, USA, 1990:364-373

[22] PALACHARLA S, KESSLER R E. Evaluating stream buffers as a secondary cache replacement. [C]// The 21st Annual International Symposium on Computer Architecture, Chicago, USA, 1994: 24-33

[23] SHERWOOD T, SAIR S, CALDER B. Predictor-directed stream buffers. [C]// The 33rd International Symposium on Microarchitecture, Monterey, USA, 2000:42-53

[24] ROTH A, SOHI G S. Effective jump-pointer prefetching for linked data structure. [C]// The 26th Annual International Symposium on Computer Architecture, Atlanda, USA, 1999:111-121

[25] YANG C L, LEBECK A R, TSENG H W, et al. Tolerating memory latency through push prefetching for pointer-intensive applications[J]. *ACM Transactions on Architecture and Code Optimization*, 2004, 1(4):445-475

[26] SUN X H, BYNA S, CHEN Y. Improving data access performance with server push architecture[C]// The 21st International Parallel and Distributed Processing Symposium, Long Beach, USA, 2007: 1-6

[27] 张升. 直线的扫描转换和直线反走样的研究[D]. 太原:中北大学电子与计算机科学技术学院, 2010:7-13

[28] 张俊杰. 嵌入式 GPU 中光栅化及深度预测试单元的研究与设计[D]. 天津:天津大学信息工程学院,2013:9-12

[29] DAVE A, KEVIN H. Beginning OpenGL Game Programming Source Code[M]. Stamford: Course Technology, 2004

An efficient GPU depth data pre-fetching strategy triggered by primitive rasterizing

TIAN Ze, ZHANG Jun, XU Hongjie

(Xi'an XiangTeng Micro-Electronic Technology Co. Ltd, Xi 'an 710068)

(AVIC Computing Technique Research Institute, Xi 'an 710068)

(Key Laboratory of Aviation Science and Technology on Integrated Circuit and Micro-System Design, Xi 'an 710068)

Abstract

Data pre-fetching technique has already been widely applied to many kinds of central processing unit (CPU) design, and obtained very good results. While, graphic processing unit (GPU) has even larger requirement for memory bandwidth, directly interrelates with graphic rendering pipeline structure and graphic algorithm, and has extraordinarily differences in memory accessing pattern in contrast to general CPU, which needs more well-directed and effective data pre-fetching strategy. Aiming at the GPU depth test function, this paper proposes a depth data pre-fetching based on primitive rasterizing triggering, called DPRT, which triggers Z Cache depth data pre-fetching by using current scanned pixel fragment tile address. To adapt graphic rendering pipeline latency in different implementation, a once touching tag (OTT) is added in Z Cache tag array to assure the pre-fetching validation. Experiment result indicates that, comparing to base graphic rendering pipeline structure, DPRT increases Z Cache hit rate by average 9.51% and reduces depth test latency by average 40.43%.

Key words: graphic processing unit (GPU), rasterizing, scan, depth test