

基于 Brook 的排序网络实现

崔雪冰^{1,2}, 张芳芳¹, 李国徽²

(1 河南城建学院 计算机科学与工程系, 河南 平顶山 467001;

2 华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

摘 要: 针对“排序网络”的特点及其广泛运用的意义, 依据现代 GPU 在通用计算方面的功能, 提出了基于 Brook 在 GPU 上执行通用计算的实现方法, 并对“排序网络”在 GPU 的实现进行了测试, 结果表明 GPU 在 Brook 环境下通用计算能力优于 CPU, 且排序网络的并行计算的特点能够在 GPU 上充分展现。

关键词: 排序网络; 数据并行计算; 流计算; GPU 通用计算; Brook

中图分类号: TP311.5

文献标识码: A

文章编号: 1000-7180(2010)05-0053-04

The Realization of Sorting Network Based on the Brook

CUI Xue-bing^{1,2}, ZHANG Fang-fang¹, LI Guo-hui²

(1 Department of Computer Science and Engineering, Henan University of Urban Construction, Pingdingshan 467001, China;

2 School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: The general computing method based on Brook is proposed in the paper according to general computing function of modern GPU, characteristic and extensive use of the “Sorting Network”. The testing result of “sorting network” by GPU’s realization presents the computing ability of GPU has an advantage over CPU and parallel computing characteristic can carried out on GPU under the Brook environment.

Key words: sorting network; data parallel computing; stream computing; GPU general purpose computation; brook

1 引言

“排序网络”^[1]边比较和边合并的一个综合排序算法, 可以对一定范围内的数据进行分类、查找和检索, 体现了并行算法思想, 比较适合此类计算的首选高性能计算机, 而这样做的代价很高, 最近几年开创的基于 GPU 的通用计算^[2] (General-Purpose Computation on GPU, GPGPU) 能够解决性价比的问题, 尤其是 Brook 的环境设计提供给开发者一个 GPU 作为流 (Streams) 处理器的观点, 大大地加强了计算速度^[3], Brook 为 GPU 通用计算提供了流计算模型, 通过流、核心函数 (kernels) 抽象出 GPU 充当流处理器来实现通用计算。事实上, GPU 采用的“SIMD”架构, 拥有大量的浮点数据的精确计算和流计算的特点, 可以把“排序网络”映射到 GPU 上。

2 基于 GPGPU 的 Brook 实现原理

Brook 是通用计算的研究内容^[4], 并适合高性能计算: 通过对流提供支持, 利用流结构代表了一系列能够被并行处理的数据, 即浮点纹理操作^[5]。文献[6]中描述了 GPU 上流的读和写操作、上传和下载纹理数据、聚集来表达纹理操作, 并通过重复和跨越操作来完成纹理取样^[6], 为了支持用户定义的流结构, 在不同的硬件纹理上存储流结构的成员, Brook 依靠 kern61 执行计算, 即在流上操作函数, 流数据存储在纹理上, 利用 GPU 的片段处理器执行。流从纹理初始化, 绘制编译器把代码转化成 GPU 的汇编代码。通过 Brook 可以支持绘制输出的数量和用户定义的数据结构。综上所述, 文中提出基于 Brook 在 GPU 上进行排序网络算法的实现。

收稿日期: 2009-04-17; 修回日期: 2009-06-17

基金项目: 国家自然科学基金项目(60873030)

GPU 程序处理过程图如图 1 所示,首先从 CPU 向 GPU 发送指令,初始化图形流水线的运行状态并控制图形绘制过程.其次,将系统内存中的顶点数据及纹理数据传入显存,经过多级流水线阶段对数据进行处理,直到像素最终被绘制到帧缓冲为止.也就是说,流从纹理初始化,绘制编译器把代码转化成 GPU 的汇编代码.同时,Brook 性能灵活,提供了如下的抽象:依靠内存控制流和“共享”数据对象组成记录;并行执行 kernel 规则的处理^[7].

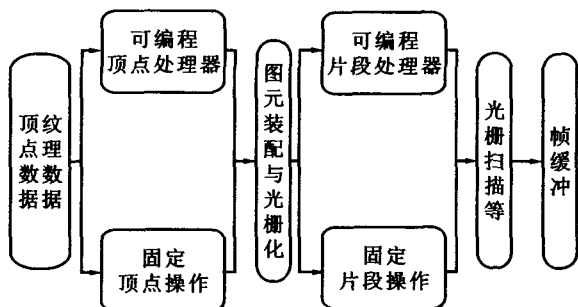


图 1 GPU 图形流水线工作流程

3 基于 Brook 的排序网络具体实现

排序网络是指具有 n 个输入 n 个输出的网络,任意输入的 n 个整数,其输出为 n 个排序之后的整数.在单个 CPU 实现排序网络时,其处理方式是串行的,且无需专门考虑存储方面的问题,而对于在 GPU 上的排序网络运算则可实现为一种数据独立的流式并行计算.基于 GPU 的排序网络算法正是利用了 GPU 的数据并行的硬件特点,主要实现了 $\text{Sort}[n]$ 和 $\text{BitMerge}[n]$ 操作^[1].其中, $\text{Sort}[n]$ 操作将无序序列排序成为双调序列, $\text{BitMerge}[n]$ 操作将双调序列合并为有序序列.

排序网络算法的实现主要分为 CPU 上运行的控制模块和 GPU 上运行的计算模块两个部分. CPU 控制模块完成初始序列和划分段数的初始化工作,同时将数据调度至 GPU 流水线; GPU 计算模块主要包括划分段坐标的计算、比较运算等操作.

首先,在 CPU 控制模块中完成对原始序列的 $\log_2 n$ 次划分,在每一次划分后调用映射函数将该划分的子序列封装为只读二维纹理,并与划分序列段在只读二维纹理中的坐标一起,通过 GPU 图形流水线发送到片元处理器.

对于无序序列构造为双调排序网络的操作 $\text{Sort}[n]$ 的实现如图 2 所示, GPU 将无序序列转换为双调序列的比较网络,可以将双调序列合并算法 $\text{BitMerge}[n]$ 应用到包含 n 个元素的无序序列上,其中

比较网络中的每一级的输出都是双调序列的连接,其长度是输入的两倍.通过连接越来越多的双调序列,最后得到长度为 n 的双调序列.之后,再利用前面所述的双调序列合并算法对长度为 n 的双调序列进行合并.

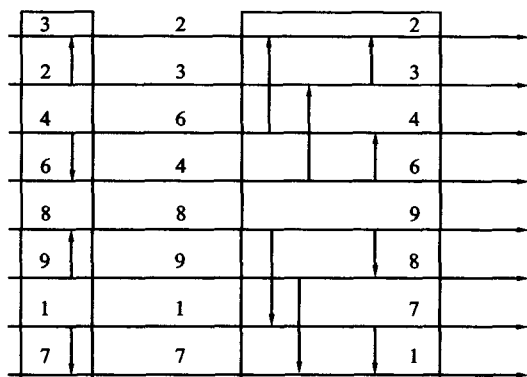


图 2 无序序列构造为双调排序网络的操作 $\text{Sort}[n]$

对于将双调序列合并为一个有序序列的操作 $\text{BitMerge}[n]$,需要将序列在运算上划分为没有依赖关系两部分数据,以便利用片元处理器对数据进行并行处理.因此,在 GPU 上实现算法时,可对已有的双调序列进行 $\log_2 n$ 次划分得到的子双调序列,利用 GPU 对子双调序列各元素实施并行比较运算,通过 $\log_2 n$ 次迭代将两个双调序列合并为一个有序序列,如图 3 所示.

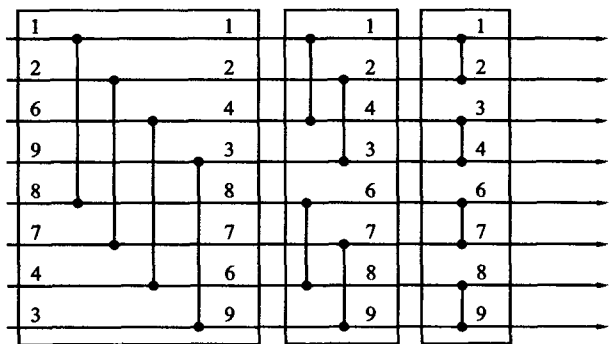


图 3 将双调序列合并为一个有序序列的操作 $\text{Sort}[n]$

应用 GPU 实施上述的计算操作充分利用了 GPU 的多个片处理器的运算能力,实现了数据流的并行处理.为了完成对各子序列的比较,将比较运算表示为一个由片元处理器执行的渲染(SHADER)程序.在计算时通过渲染存储了子序列的纹理块得到比较运算的结果,即由片元处理器通过划分序列段的纹理坐标对只读二维纹理进行采样,完成对两个子序列中对应元素的比较操作,并实施相应的元素值交换操作,之后将输出序列写入只写纹理中对应的元素.由于片元处理器一次只能对子序列中的对应元素进行一次比较,因此实际操作中可将只写纹

理与只读纹理进行交换,以实现运算过程中的数据流即各个子序列中的对应元素的存储.

下列代码描述了双调序列合并算法和双调排序网络构造算法在 CPU 控制模块和 GPU 计算模块中的实现过程的伪代码:

```

CPU 控制模块:
...
streamRead(sorted1Strm, array);
for (stage=1; stage<=lg-arraySize; stage++) {
    segWidth = (float)pow(2.0f, stage);
    for (step=1; step<=stage; step++) {
        ..... //以下是 GPU 计算模块
        bitonic(sorted1Strm, sorted2Strm, segWidth, offset, 2 *
offset,maxvalue);
    }
    else //以下是 GPU 计算模块
        bitonic(sorted2Strm, sorted1Strm, segWidth, offset, 2 *
offset,maxvalue);
    flip = (flip) ? 0 : 1;
}
}

GPU 计算模块:
kernel void bitonic(float input[ ][ ], out float output<>,
float stageWidth, float offset, float twoOffset,float2 maxvalue)
{
    .....
    getIndexAt (indexof (output), sign * offset, maxvalue,
idx2);
    min = (input[idx1] < input[idx2]) ? input[idx1]: in-
put[idx2];
    max = (input[idx1] > input[idx2]) ? input[idx1]: in-
put[idx2];
    output = (sign == dir) ? min : max;
}

```

由于 Brook 的流模型的作用,屏蔽了诸如数据输入等数据操作的细节.例如数据的输入 streamRead(sorted1Strm, array),为了能处理数据量足够大的数据使用 2D 纹理来存储数据(利用 GPU 的纹理存储数据),利用数组的索引对应纹理的坐标.该函数主要完成数列的输入,其底层的图形操作伪代码如下:

```

void streamRead () {
    glGenTextures (2, yTex2D);
    setupTexture (yTex2D[readTex]);
    glReadPixels (0, 0, texSize, texSize, textureParameters.
texFormat,GL_FLOAT,data);//
    ...
    transferToTexture(dataY,yTex2D[readTex]);
}

```

```

setupTexture (yTex2D[writeTex]);
transferToTexture(dataY,yTex2D[writeTex]);
...
}

```

上段代码中 dataY 指数数据能够传入能读能写纹理的数据由 [readTex] 和 [writeTex] 纹理的属性决定.dataY 的纹理能够在计算过程中进行读写操作,同时,transferToTexture 函数指出了该一维数组 dataY 已经被传到 yTex2D 这个二维纹理.

4 实验结果及分析

从上面的实现,排序算法通过数据直接映射到 GPU 的纹理内存中,在底层仍然实现的是图形的绘制操作.为了能够顺利进行 GPU 通用计算,需要配置表 1 所示的环境.

表 1 NVIDIA GeForce 7600 显卡的相关硬件参数

CPU	DualCore Intel Pentium D 820, 2 800 MHz
主存	2 048 MB
GPU	NVIDIA GeForce 7600 GT
显存	256 MB
操作系统	XP2.0
编程环境	visual stadio 2005
图形接口	OpenGL2.0 和 Direct10
其他	扩展库 GIUT 和 GLEW

实验数据由其五组随机产生的序列,分别为 $64^2, 128^2, 256^2, 512^2, 1024^2$,以控制数组的最大的个数,来比较 CPU 和 GPU 的性能,下面分别给出 CPU 的环境和测试数据结果.对于实验结果主要采取以下几个测试指标:(1)在问题规模的不断扩大的情况下,GPU 计算和 CPU 计算的差别;(2)随着问题规模的扩大,GPU 和 CPU 各自的性能;(3)随着问题规模的扩大,对于在 GPU 和 CPU 数据运算量的比较.图 4 给出了与 CPU 相同规模排序,计算数据传输情形下 GPU 实现排序网络所消耗的时间.

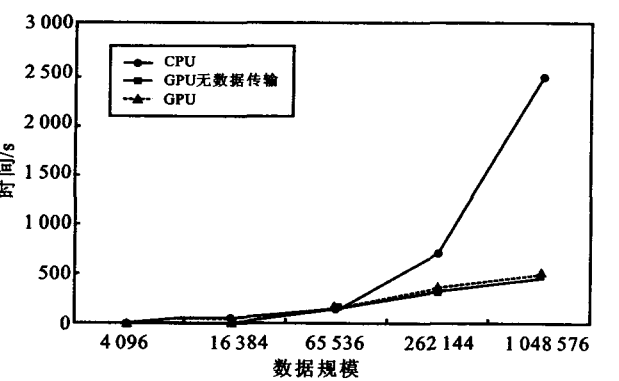


图 4 CPU 和 GPU 网络排序时间比较图

通过上面的实验结果得出以下结论:随着问题规模的扩大,三者的处理速度都在降低,CPU 的处理所耗时间更长,即处理同样规模的问题时,CPU 在规模放大时执行计算时间成几何级数上涨,而 GPU 没有明显变化,说明 GPU 在某些情形下具有较高的并行能力;在没有考虑 CPU 向 GPU 的传输数据的时间内,GPU 依然取得了较高的处理速度,说明 GPU 得到了高效的传输带宽的支持;在数据量小的时候,CPU 和 GPU 无明显变化,但是一旦数据规模较大时将会变化很大,如问题规模在 (512×512) 以后,计算速度分歧加大。

5 结束语

文中讨论了 GPU 和 CPU 的并行计算性能,并得出 GPU 的并行性能在问题较大时能够优于 CPU。通过实验发现,在少量的数据的排序算法中 GPU 的高性能不太能够体现,但是进行大规模的数据排序是很有效的。同时,排序网络算法在 Brook 上的实现,也说明了基于图形接口的可扩展和图形操作的可抽象能够为程序设计者提供方便有效的思路,具有较好的发展前景。

参考文献:

- [1] Donald E Knuth. 计算机程序设计艺术[M]. 2 版. 苏运龄,译. 北京:国防工业出版社,2002:490-560.
- [2] Michael Macedonia. The GPU enters computing's mainstream[J]. IEEE Computer Society, 2003, 36(10): 106-108.
- [3] 吴恩华,柳有权. 基于图形处理器的通用计算[J]. 计算机辅助设计与图形学学报,2004,16(3): 601-613.
- [4] Bolz J, Farmer I, Grinspun E, et al. Sparse matrix solvers on the GPU: conjugate gradients and multigrid[J]. ACM Trans. Graph,2003,22(3):917-924.
- [5] Mark W R, Glanville R S, Akeley K, et al. A system for programming graphics hardware in a C-like language[J]. ACM Trans. Graph, 2003,22(3):896-907.
- [6] Harris M J, Baxter W V, Scheuermann T, et al. Simulation of cloud dynamics on graphics hardware[C]//Proceedings of Graphics hardware. Eurographics Association, 2003:92-101.
- [7] 池凤彬,潘日华,陈扉,等. 基于 GPU 的 VLSI 的 DRC 加速系统[J]. 微电子学与计算机,2007,24(4):51-52.

作者简介:

崔雪冰 女,(1972-),硕士,讲师.研究方向为信息处理。
李国徽 男,(1973-),博士,教授.研究方向为数据库系统理论及集成技术。

(上接第 52 页)

法来确定 LDPC 译码器迭代译码的收敛情况,并提出了一种低复杂度的提前结束迭代准则。这种准则实现简单,只需要少量的硬件消耗,能在很小的性能损失的情况下很大程度地降低平均迭代次数,并可以根据实际系统需要实现性能和平均迭代次数的折衷。

参考文献:

- [1] 徐华. LDPC 码相关技术研究[J]. 微电子学与计算机, 2005,22(12):178.
- [2] Lin C C, Lin K L, Chung C C, et al. A 3.33 Gb/s (1200, 720) low-density parity check code decoder[C]// Proc. ESSCIRC. European, 2005: 211-214.
- [3] European Telecommunications Standards Institute (ETSI). Digital video broadcasting (DVB) second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; TM 2860r1 DVBS2

-74r8[S]. European Telecom Munieations Standard Institute.

- [4] Brack T, Alles M, Kienle F, et al. A synthesizable IP core for WIMAX 802.16E LDPC code decoding[C]//Proc. IEEE 17th Int. Symp. Personal, Indoor and Mobile Radio Communications. Helsinki, 2006:1-5.
- [5] Hagenauer J, Offer E, Papke L. Iterative decoding of binary block and convolutional codes[J]. IEEE Trans. Inform. Theory, 1996(42):429-449.
- [6] Shao R Y, Lin S, Fossorier M P C. Two simple stopping criteria for turbo decoding[J]. IEEE Trans. Commun., 1999(47):1117-1120.

作者简介:

李春阳 男,(1983-),硕士研究生.研究方向为信道编解码、低功耗数字基带芯片。
黑勇 男,(1974-),研究员.研究方向为可重构数字信号处理器、低功耗数字基带芯片。
郭琨 女,(1982-),博士研究生.研究方向为信道编解码、低功耗数字基带芯片。