第35卷 第17期 计 算 机 工 程 2009年9月
Vol.35 No.17 Computer Engineering September 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)17—0274—03 文献标识码：A 中图分类号：TP311.11

共轭梯度法的GPU实现

夏健明[1,2]，魏德敏[1]
(1. 华南理工大学土木工程系，广州 510640；2. 广东水利电力职业技术学院土木工程系，广州 510635)

摘 要：提出基于图形处理单元(GPU)实现矩阵与向量相乘的新算法，只需渲染四边形一次即可实现矩阵与向量乘法。并给出实现向量元素求和的新算法，与缩减算法不同，该算法不要求向量大小为2的幂。基于这2种算法使用OpenGL着色语言(GLSL)编程，用GPU实现求解线性方程组的共轭梯度法。与Krüger算法相比，该方法所用计算时间更少。

关键词：图形处理单元；OpenGL着色语言；共轭梯度法

GPU Implementation of Conjugate Gradient Method

XIA Jian-ming[1,2], WEI De-min[1]
(1. Dept. of Civil Engineering, South China University of Technology, Guangzhou 510640;
2. Dept. of Civil Engineering, Guangdong Technical College of Water Resource and Electric Engineering, Guangzhou 510635)

【Abstract】This paper presents a new algorithm for computing matrix-vector multiplication based on a Graphics Processing Unit(GPU). A matrix is presented by a texture and matrix-vector multiplication can be realized by rendering a quadrilateral one pass. Instead of vector reduction, it presents a new algorithm for summing all entries of a vector, not requiring the size of the vector be the power of two. Based on these algorithms, the conjugate gradient method for solving linear equations is implemented using the GPU with OpenGL Shading Language(GLSL). The computation is compared against that on Krüger's algorithm, proving the efficiency of the proposed algorithms.

【Key words】Graphics Processing Unit(GPU); OpenGL Shading Language(GLSL); conjugate gradient method

1 概述

图形处理单元(Graphics Processing Unit, GPU)是常用于兼容PC机上的廉价而运算能力强的并行处理器。随着图形硬件的发展，图形处理器在一些复杂领域中具有可编程能力，以代替原有的固定功能。除了计算机图形学领域外，GPU正被用于其他更广泛的领域，使通用图形处理器(General Purpose Graphics Processing Unit, GPGPU)成为目前研究的热点[1]。如使用GPU模拟用偏微分方程描述的动力学现象，例如沸腾、对流和化学扩散等；使用GPU模拟树枝状冰晶体的生成；使用GPU计算一系列的流体力学问题；使用GPU计算 N 个物体的相互引力作用，当物体数量大时，基于GPU的计算比基于CPU的计算快50倍；使用GPU实现量子力学的蒙特卡罗(Monte Carlo)计算；使用GPU可视化地模拟浅水波的流动。使用GPU实现快速傅里叶变换、复杂边界的三维实时流体模拟、加速分子动力学的计算进行有限元计算；使用GPU加速分子动力学计算，计算固态氩的热传导性能，比基于CPU的计算快10倍；使用GPU计算液晶自适应光学波前重构，使计算速度提高10倍，充分利用颜色通道的并行计算，可使计算速度增快3倍多。

矩阵与向量相乘、矩阵与矩阵相乘等算法是科学计算的基础，其效率对计算有较大影响。GPU能有效地处理密集的数值运算。文献[2]用二维纹理表示一个向量，提出矩阵与向量相乘的GPU算法，并用汇编语言实现求解线性方程组的共轭梯度法和Gauss-Seidel法。文献[3]也提出基于GPU的求解稀疏线性方程组的共轭梯度法。

本文提出一个实现矩阵与向量相乘的新算法，该算法比Krüger算法更有效。同时提出向量大小不是2的幂的向量元素求和算法。根据所提出的算法，使用OpenGL着色语言(OpenGL Shading Language, GLSL)编程，实现了基于GPU求解线性方程组的共轭梯度法，计算结果与Krüger算法作比较，证实本文方法比Krüger方法高效。

2 OpenGL着色语言

OpenGL着色语言是一种用于OpenGL的高级过程着色语言，是近年图形编程领域中出现的重要新型开发技术[4]。OpenGL着色语言及支持它的OpenGL扩展已被批准为ARB扩展，目前已被OpenGL2.1添加到OpenGL核心中。

OpenGL的处理管道如图1所示，其中，顶点处理器①和片元处理器②具有可编程能力，可用着色语言编写代码，实现程序员的特定功能，以代替它们原有的固定功能。在一个OpenGL可编程处理器上执行的OpenGL着色器语言代码称为着色器(shader)，可分为顶点着色器(vertex shader)和片元着色器(fragment shader)，OpenGL提供了编译这些着色器并将它们连接在一起构成可执行代码的机制。

顶点处理器涉及在各顶点发生的操作，尤其是变换和光照。片元处理器是由在各个片元处理器基础上发生的操作，以纹理内存读取和在各片元上应用纹理的操作组成。由于GPU的片元处理器通常比顶点处理器多，而且数据经过片元处理器后，直接输出到帧缓冲区，因此通常利用片元着色器实现GPU的通用计算。

作者简介：夏健明(1967－)，男，副教授、博士研究生，主研方向：计算力学；魏德敏，教授、博士

收稿日期：2008-12-20 **E-mail：**xiajm73@163.com

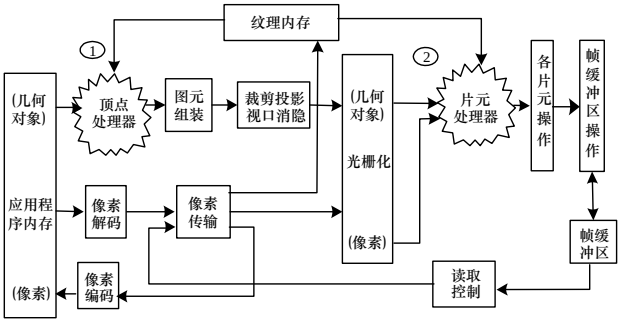

图 1　OpenGL 管道

3　基本运算

本文将矩阵和向量存储于 2 个纹理内存中，利用可编程的片元着色器实现算法运算。

3.1　矩阵与向量乘法

文献[2]用一个二维纹理表示向量，将矩阵的对角元素看作一向量，用一个纹理表示，则 $N\times N$ 矩阵需要 N 个纹理描述。将表示向量的纹理与 N 个表示矩阵的纹理分别相乘，并将乘积相加，实现矩阵与向量的乘法，须渲染相应的四边形 N 遍，而且在每一次渲染中都需要向纹理传输数据。

本文提出一个新的矩阵与纹理相乘的算法。矩阵用一个 4 纹元(r, g, b, a)纹理表示。矩阵 $\boldsymbol{A}$ 可用其列向量 $\boldsymbol{a}_0$，$\boldsymbol{a}_1$，…，$\boldsymbol{a}_{N-1}$ 表示，则一个 $N/2\times N/2$ 二维 4 纹元纹理可表示 $N\times N$ 矩阵。矩阵每个列向量可用 $N/4$ 个纹素表示，每 $N/4$ 个纹素构成的区域称为向量区域(vector zone)，如图 2 所示，该纹理记作 *textureA*。

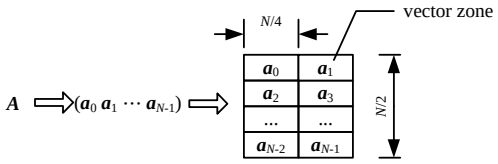

图 2　矩阵的纹理表示

用另一个 $N/2\times N/2$ 纹理表示向量 $\boldsymbol{b}=\{b_0, b_1,\cdots, b_{N-1}\}^{\mathrm{T}}$，记作 *textureB*。在 *textureB* 的第 i 个向量区域全部写入向量 $\boldsymbol{b}$ 的第 i 个元素 b_i，其中，i=0, 1, …, N-1，如图 3 所示。

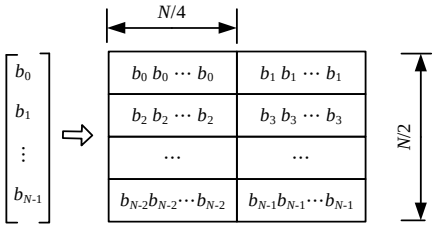

图 3　向量的纹理表示

令纹理 *textureA* 与纹理 *textureB* 相乘，得积纹理 *textureC*，将积纹理 *textureC* 的各向量区域加到第 1 个向量区域，这个操作记作 sumVector，则积纹理的第 1 个向量区域存储了矩阵 $\boldsymbol{A}$ 与向量 $\boldsymbol{b}$ 的积。上述算法只需渲染四边形 1 遍即可实现矩阵与向量乘法，而且不需频繁地向纹理传输数据，因此，与 Krüger 算法相比更为高效。

3.2　向量元素求和

在计算向量的范数，求向量的最大、最小元素时，需要把向量所有元素归并到一个元素上。一般的方法是通过纹理的缩减运算实现向量元素的求和[2]，缩减运算要求向量的大小为 2 的幂，但许多情况下向量的大小并不是 2 的幂。本文提出基于 GPU 实现向量元素求和的新算法，不要求向量大小为 2 的幂。渲染表示向量的纹理的第 1 个纹素，编写片元着色器，令该纹理所有纹素加到第 1 个纹素，则第 1 个纹素就是向量元素的和，该操作记作 sumTexel，如图 4 所示。

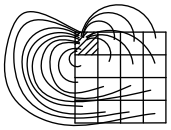

图 4　向量元素求和过程

4　共轭梯度法的 GPU 实现

共轭梯度法(conjugate gradient method)是求解大型稀疏线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的高效迭代方法。在实际工程有广泛应用的有限元法和有限差分法中经常需要求解大型稀疏线性方程组。

利用 GPU 可有效地计算共轭梯度法中的“矩阵-向量”，“向量-向量”乘法，需要 8 副纹理 *textureY, textureX, textureA, textureB, textureP, textureQ, textureR, textureZ*，基于 GPU 的共轭梯度法算法如下所示：

```
textureZ←x
textureY←textureA*textureZ
textureP←sumVector(textureY)
textureR←textureB-textureP
k=0
textureY←textureR*textureR
textureQ←sumTexel(textureY)
rm0=sumTexture(textureQ)
do while (rm0>=eps && k<=N)
    k=k+1
    if (k=1)
        textureP←textureR
        rm1=rm0
    else
        textureY←textureR*textureR
        textureQ←sumTexel(textureY)
        rm1=sumTexture(textureQ)
        beta=rm1/ rm0
        textureQ←textureR+beta*textureP
        textureP←textureQ
    end if
    textureZ←textureP
    textureY←textureA*textureZ
    textureQ←sumVector(textureY)
    textureY←textureP*textureQ
    textureZ←sumTexel(textureY)
    pq=sumTexture(textureZ)
    alpha=rm1/pq
    textureZ←textureX+alpha*textureP
    textureX←textureZ
    textureZ←textureR-alpha*textureQ
    textureR←textureZ
    rm0=rm1
end do
```

5 算例

使用上述算法，用 OpenGL 着色语言编程，用 ATI Radeon X1550 GPU 求解以下线性方程组：

$$\boldsymbol{Ax}=\boldsymbol{b} \tag{1}$$

其中，$\boldsymbol{A}$ 对称，其元素 a_{ij} 满足：

$$a_{ij}=0.1\times i+0.1\times j,\ i=0,1,\cdots,N-1,\ j=i,i+1,\cdots,N-1 \tag{2}$$

$$a_{ii}=1+i$$

向量 $\boldsymbol{b}$ 满足：

$$b_i=1+i \tag{3}$$

$\boldsymbol{x}$ 的初值取 $\boldsymbol{x}_0=\{0,0,\cdots,0\}^{\mathrm{T}}$。用 Krüger 算法解同一方程组，2 种算法的计算时间比较如图 5 所示，当矩阵大小 N=1 024 时，Krüger 算法的计算时间是上述算法的 4.4 倍。

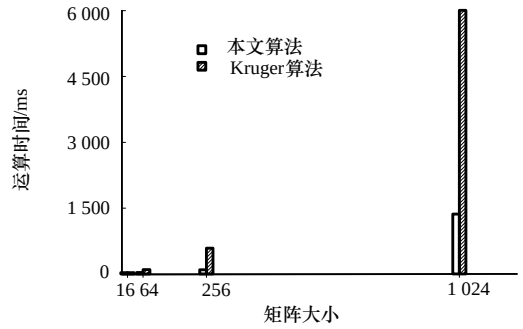

图 5　2 种算法的运算时间比较

6 结束语

本文提出求解线性方程组的共轭梯度法的 GPU 实现算法，算例的计算结果表明，GPU 具有强大的并行数值运算功能，作为计算协处理器，不但能加速线性方程组的求解，也能加速一般科学问题的计算。

下一步研究方向是深入研究 GPU 在科学计算中更复杂的算法，将 GPU 应用于更广泛的科学计算领域和发展 GPU 群，解决大规模的科学计算问题。

参考文献

[1] Qwens J D, Luebke D, Govindaraju N, et al. A Survey of General-purpose Computation on Graphics Hardware[C]//Proc. of Eurographics'05. Grenoble, France: [s. n.], 2005.

[2] Krüger J, Westermann R. Linear Algebra Operators for GPU Implementation of Numerical Algorithms[J]. ACM Transactions on Graphics, 2003, 22(3): 908-916.

[3] Bolz J, Farmar I, Grinspun E, et al. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid[J]. ACM Transactions on Graphics, 2003, 22(3): 917-924.

[4] Rost R J. OpenGL 着色语言[M]. 天宏工作室, 译. 北京: 人民邮电出版社, 2006.

编辑　金胡考

（上接第 273 页）

企业获取隐性知识和适应外部环境的能力。目前集成系统已在相应 4 个企业进行应用试点，运行良好。

集成平台的 CBR 流程定义界面如图 5 所示。

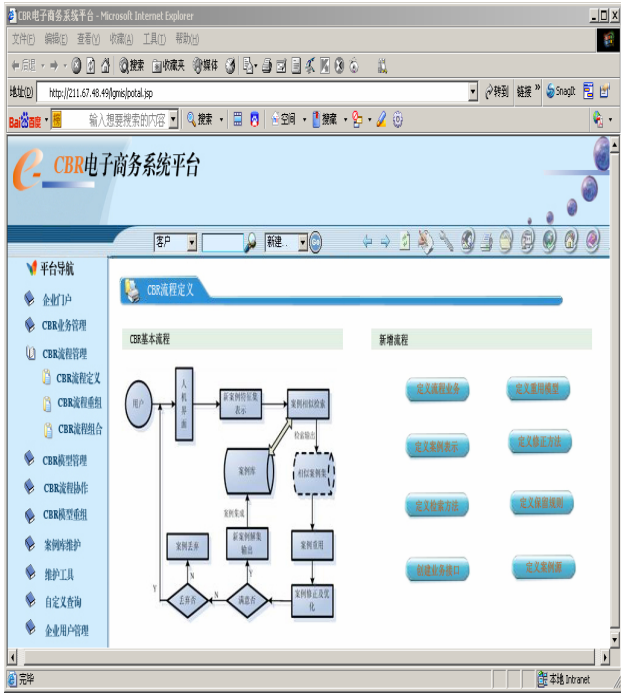

图 5　集成平台的 CBR 流程定义界面

7 结束语

随着 CBR 系统应用的发展，多 CBR 系统应用集成将成为隐性知识集成的重要手段，建立多 CBR 系统应用集成平台，进一步提高知识集成的智能化，同时 SOA 为多 CBR 系统的应用集成提供集成框架，利用现有服务和构件，企业能快速、有效地重组或更改 CBR 推理流程和推理模型，获取不同时期的隐性知识。本文开发的多 CBR 的电子商务集成系统平台，为多 CBR 系统的应用集成提供较为深入的实现和应用模式。

参考文献

[1] Shiu S C K, Paul S K. Case-based Reasoning: Concepts, Features and Soft Computing[J]. Applied Intelligence, 2004, 21(3): 233-238.

[2] 李玉刚，纪卓尚，林　焰. 基于 SOA 和 Web 服务的造船虚拟企业应有集成[J]. 计算机工程, 2008, 34(7): 263-265.

[3] 魏武华. 基于 CBR 和 XML 的知识管理系统架构研究[J]. 计算机工程与设计, 2006, 27(14): 2608-2610.

[4] 孙德建，陶　旭，李　鹏. 基于 SOA 的军事信息系统应用集成研究[J]. 情报杂志, 2008, 27(1): 57-60.

[5] 绕　云. 面向服务体系结构的企业资源计划系统应用模型与集成策略[J]. 计算机集成制造系统, 2006, 12(20): 1570-1576.

编辑　陆燕菲