GPU 技术在 SPH 上的应用

李海州，唐振远，万德成*

(上海交通大学 船舶海洋与建筑工程学院 海洋工程国家重点实验室，
高新船舶与深海开发装备协同创新中心，上海 200240)

*通信作者 Email: dcwan@sjtu.edu.cn

摘要：无网格粒子法能够有效的处理溃坝、晃荡、波浪破碎等具有瞬时大变形的物理问题。然而随着粒子数增加，该方法的计算效率成为限制其在大规模工程应用上的重大瓶颈。通过借助 GPU 技术，可以极大的提高计算效率。本文基于开源软件 DualSPHysics，研究了 SPH 方法在 GPU 上的实现，并重点讨论了邻居例子搜寻方法的具体细节。然后通过对一个带立柱三维溃坝的模拟，验证了 GPU 并行加速在 SPH 方法应用中的有效性。数值结果表明，通过运用 GPU 加速，相比于 CPU 最高可获得两个数量级的加速效果。

关键词：无网格粒子法；SPH；溃坝绕流；GPU；邻居粒子搜索

1 引言

无网格粒子法在处理带有自由面问题时具有很大的优势，其中 SPH（Smoothed Particle Hydrodynamics）和 MPS（Moving Particle Semi-Implicit）是两种常用的粒子法，常被人们用来研究破波[1]、溃坝[2]和液舱晃荡[3]等复杂的流动问题。然而随着粒子数增加,该方法的计算效率成为限制其在大规模工程应用上的重大瓶颈。为了寻求有效的加速手段，本文以 SPH 方法为对象，研究了近些年兴起的 GPU 技术在该方法中的应用。

SPH (smoothed particle hydrodynamics) 是一种基于拉格朗日力学的无网格粒子法，该方法中流体计算域会被离散为一系列带有相关流场信息的粒子，而粒子之间通过核近似来模拟连续场，然后通过对相应的流体方程进行求解，从而实现对流体运动的模拟。其在计算空间导数的时候不需要使用网格离散，而是通过对核函数进行求导加权求和，从而避免了网格方法中网格的扭曲和缠结等最令人头疼的问题。SPH 方法最早在 1997 年由 Lucy 等[4]提出并应用于求解天体物理问题，随着该方法的不断完善，近些年来也被应用于诸多领域，如高速水流[5]、水下爆破[6]等数值模拟。

过去，我们通常通过运用更多的 CPU 线程来使得程序能够进行并行计算，但是目前一块 CPU 芯片所能并发的线程数是很少的，因此我们往往会采用在高性能集群上计算以获得更多的线程数。然而，高性能集群（HPC）的建造维护费用或者租用费用也是很高的，从而限制了 CPU 并行的利用。随着计算机科学的不断发展，一种称为 GPU 的加速技术，以其强大的并行计算能力和相对低廉的成本，满足了诸多领域的计算需求，近年来得到迅猛

的发展。

而目前关于 GPU 在无网格粒子法中应用的研究还不够充分。本文基于对开源软件 DualSPHysics[7]代码的研究，首先讨论了 SPH 方法在 GPU 上的具体实现，并比较了与 CPU 的异同。接着，通过模拟一个带立柱的溃坝绕流问题，对 GPU 的计算可靠性进行了分析。最后，对 GPU 的计算效率进行了比较和探讨，从而为 GPU 加速在诸如 SPH 等无网格方法中的应用提供了一定的参考。

2 SPH 方法

2.1 积分离散

SPH 方法是基于流体描述中的拉格朗日观点的，其系统的状态是用一系列的粒子来描述的，这些粒子包含着各自的材料性质，如密度、压力、内能、速度等。基于这种离散方法，对于任意场函数 f(r),都可以通过核函数近似写成一下积分表达形式：

$$f(r)=\int_{\Omega} f(r')W(r-r',\mathrm{h})\mathrm{d}r' \tag{1}$$

式中：r 为任意点的空间矢量；Ω 为 r 的积分区域；$W(r-r',\mathrm{h})$为核函数，本文中采用 Monagha[8]提出的三次样条函数作为核函数，h 为核函数的光滑长度；对式（1）进行粒子近似，可以得到 i 粒子处场函数的进一步表达式：

$$f(r_i)=\sum_{j=1}^{N}\frac{m_j}{\rho_j}f(\mathrm{r_j})\cdot W_{\mathrm{ij}} \tag{2}$$

式中：$W_{ij}=W(r_i-r_j,h)$；m_j,ρ_j分别为邻域粒子 j 的质量和密度；N 为邻域中的粒子总数。

2.2 控制方程

拉格朗日形式的流体运动控制方程包括连续性方程和动量方程，分别如下：

$$\frac{D\rho}{Dt}=-\rho\nabla\cdot\mathbf{V} \tag{3}$$

$$\frac{DV}{Dt}=-\frac{1}{\rho}\nabla P+\nu\nabla^2 V+\boldsymbol{g} \tag{4}$$

式中：ρ为流体密度，P为压力，V为速度向量，$\boldsymbol{g}$为重力，ν是运动粘性系数，t为时间。用核近似离散式（3）和式（4），可得到：

$$\frac{d\mathbf{v}_i}{dt}=-\sum_{j=1}^{N}\mathrm{m_j}\left(\frac{P_j}{{\rho_j}^2}+\frac{P_i}{{\rho_i}^2}+\Pi_{ij}\right)\nabla_i W_{ij}+\mathbf{g} \tag{5}$$

$$\frac{d\rho_i}{dt}=\sum_{j=1}^{N}m_j\mathbf{v}_{ij}\cdot\nabla W_{ij} \tag{6}$$

式中，Π_{ij}为人工粘性项[9]，其表达式为：

$$\Pi_{ij}=\begin{cases}\dfrac{-\alpha\overline{c_{ij}}\mu_{ij}}{\overline{\rho_{ij}}} & v_{ij}\cdot r_{ij}<0\\ 0 & v_{ij}\cdot r_{ij}>0\end{cases} \tag{7}$$

式中，$\mu_{ij}=\dfrac{h\mathbf{v}_{ij}\cdot\mathbf{r}_{ij}}{r_{ij}^{2}+\eta^{2}}$，$\overline{\rho_{ij}}=0.5(\rho_i+\rho_j)$为密度的均值，$\overline{c_{ij}}=0.5(c_i+c_j)$是声速的均值，$\eta^2=0.01h^2$,$\alpha=0.01$是一个参数。

2.3 状态方程

通过引入人工压缩性[10]，把一般的不可压缩流体看作可压缩流体，即用准不可压流体状态方程来模拟不可压流体。这样可以不用求解压力泊松方程而直接显示求解压力，计算效率得到了极大的提高。

$$P_i=B\left[\left(\frac{\rho_i}{\rho_0}\right)^{\gamma}-1\right] \tag{8}$$

式中，γ=7，$B=c_0{}^2\rho_0/\gamma$，$\rho_0=1000\mathrm{kg}\cdot\mathrm{m}^{-3}$是参考压力值，$c_0=c(\rho_0)=\sqrt{(\partial \mathrm{P}/\partial\rho)}\Big|_{\rho_0}$则是参考压力下的声速。

3 SPH 在 GPU 上的实现

3.1 GPU 编程模型

GPU（Graphics Processing Unit），即显示芯片，最初主要用于图形处理的加速，随着技术的不断革新，目前 GPU 已发展成为一种高度并行化、多线程、多核的处理器，具有超大的计算吞吐量和很高的存储器带宽。如图 1[11]所示，GPU 相较于 CPU 划分了更多的执行单元（ALU），从而能并发执行相当多数量的线程，而这也正是 GPU 拥有强大并行计算能力的原因所在，也即 GPU 是专为计算密集型、高度并行化的计算而设计。

在处理具体问题时，为了充分利用 GPU 并行加速的优势，首先要确定该问题是否可以表示为数据并行计算的问题——在许多数据元素上并行执行相同的程序，也即基于数据的并行（与之相对的是基于任务的并行）。通常来说，渲染图像的后期处理、视频编码和解码、图像缩放、立体视觉和模式识别等是非常适用于这种数据并行模式的，因而这些领域在GPU上往往能取得很高的计算加速比。在科学计算领域，其实也有不少问题可以分解为这种并行模式，以无网格粒子法为例，其计算是由一系列的流体粒子的相互作用来完成的，在计算中每个粒子所要执行的计算是完全相同的，也即在不同的数据上执行相同的程序，这正好是适合于 GPU 的计算问题。因此，诸如 SPH 等无网格粒子法是可以通过 GPU 来加速计算的。

图 1　GPU 中的更多晶体管用于数据处理

3.2 SPH 的计算流程

DualSPHysics 的 GPU 计算求解流程如图 2 所示。最开始时，先用前处理把计算域离散成粒子，然后程序读入粒子的初始化信息及相应配置并对求解器进行初始化，之后再把初始化的数据复制到 GPU 端开始进行求解。整个求解过程主要分为三大部分：邻居粒子的更新、计算粒子间相互作用和系统信息更新（如位移、速度等）。

SPH 在 GPU 上的实现，从计算流程上来说和 CPU 是基本一致的(具体的实现是很不一样的)，但是多了一个数据复制的过程。因为 GPU 是一个独立的设备，GPU 加速计算的过程是在 GPU 上执行的，而 GPU 和 CPU 之间则是通过 PCI 总线来实现数据交换的。在计算开始的时候，初始数据是位于 CPU 上的，因此需要把数据传输到 GPU 上才能在 GPU 上进行计算。而当计算到一定位置，需要保存数据的时候，则又从 GPU 上把数据复制到 CPU 上，再进行文件的写入等等。

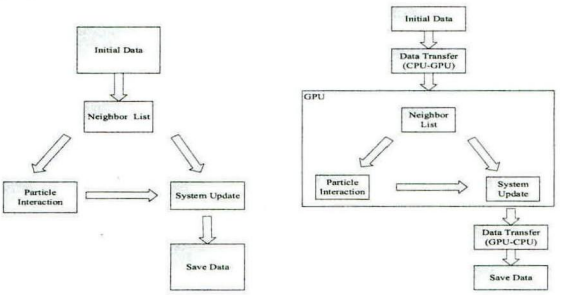

图 2　SPH 计算流程 CPU（左图），GPU（右图）

3.3 邻居粒子搜寻

用 SPH 方法模拟流体的时候，流体系统是通过粒子之间的相互作用来实现的。而每个粒子只有其作用域内的粒子才会计算相互作用。在模拟流体的过程中，粒子间的相对位置是不断变化的，因此必须在每个时间步长都要重新求解每个粒子作用域中所包含的粒子，可以说搜索邻居粒子是相当重要且关键的一步，其在很大程度上决定了程序的整体结构。因此，这一节将具体讨论邻居粒子搜寻在 GPU 上的实现。

通常来说，SPH 较普遍采用的粒子搜寻方法有 CLL（cell-linked list）和 VL（Verlet list）两种[12]，其基本思想都是将粒子搜索域划分成一个个很小的单元，从而缩小搜索范围，从而提高搜索效率。在 DualSPHysics 中所采用的是 CLL 方法，这个方法的 GPU 版本最早可以从 NVIDIA CUDA Toolkit 示例中的 Particles 中找到。

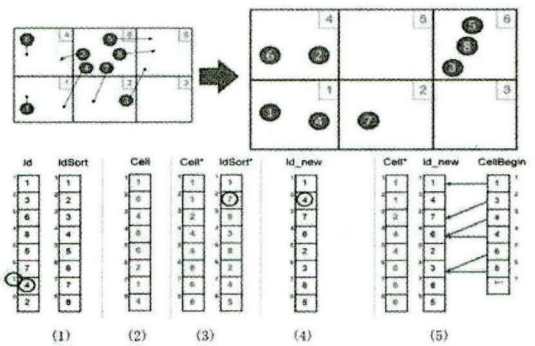

图 3　邻居粒子搜寻方法

如图 3 所示，邻居粒子的搜索主要有以下几个步骤：

(1) 把计算域划分成小的单元，并记录每个粒子在数组中的 ID 与 IdSort，即下标。
(2) 更新每个粒子所处的单元编号。
(3) 根据粒子所处单元编号，对 IdSort 进行排序。
(4) 根据排序后的 IdSort*，对粒子的相关变量（如速度，位置等）进行排序。
(5) 计算每个 Cell 的起始编号，至此，就可以得到每个 Cell 所属的粒子。

4 三维溃坝模拟

本节以一个带立柱的三维溃坝问题[13]为研究对象，以验证 GPU 并行的可靠性和研究 GPU 并行的计算效率。

4.1 溃坝模拟

如图 4 所示，长方形水槽大小为 1.6 m(长)×0.6 m(宽)×0.6 m(高)，初始时刻在水槽左侧 0.4 m×0.6 m×0.6 m 的区域内会被水体占据，方柱结构物尺寸为 0.12 m(长)×0.12 m(宽)×0.6 m(高)，放置在水体下游 0.5 m 处，方柱侧面距离水槽侧面距离 0.24 m。Wu[14]在实验过程中记录了方柱的受力情况，同时在方柱前方 0.146m、高 0.026m 出放置了流场速度探测装置，用于记录该点处的流体速度变化情况。本计算中采用动时间步长，CFL 设置为 0.2，声速系数为 20，在压力项中计入了人工黏性修正，离散的粒子总数为 1345851。

溃坝试验中，初始时水体前方会放置一块挡板，当挡板移开后，水体在重力作用下形成溃坝波向下游流动。当溃坝波遇到方柱时，可以观察到溃坝被沿方柱表面爬高、波浪破碎和翻卷等现象。计算过程中实时记录了方柱的受力情况，同时对和实验中速度探测点相同位置 G（0.756, 0.3, 0.026）的速度情况进行了记录。

图 5 为探测点 G 处速度随时间的变化情况，其中 SPH 为本文计算结果，naoe-FOAM-SJTU 为曹[13]的数值结果，lab 为实验结果。通过图中对比可以看到：本文结果和实验结果[14]吻合的较好，弧度变化的趋势和大小基本一致；与曹[13]的数值结果相比，刚开始时本文数值结果比曹[14]的要稍大，然后稍小，但最后两者趋于一致。

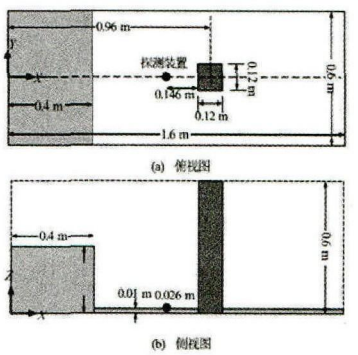

图 4 三维带立柱溃坝计算模型

图 6 为方柱受到拍击力的时间历程。可以看到，本文的 SPH 计算结果和曹[13]的数值结

果在前半程都同实验值[14]吻合得较好，但是在约 1.5s 处本文计算结果和实验值都可以看到明显的尖峰，而曹[13]的数值结果中此时虽然也是达到了最小值，但是之后并不是很快拉升，而是有一个较平的过渡。通过观察流动情况可以发现，此时溃坝波已经被右侧墙壁反射回来再次冲击方柱，在绕过方柱后，又和溃坝波首次被方柱反射的水体以及前方来流相互作用形成了非常复杂的自由面流动。因此，上述数值结果不同的原因可以推测为，基于粒子法的 SPH 在此时能够较好的处理大变形的复杂自由面流动，而网格方法则由于网格的剧烈扭曲而没能很好捕捉到。

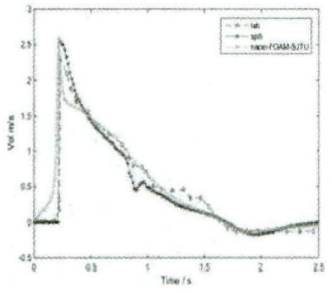

图 5　探测点 G 处速度随时间变化曲线

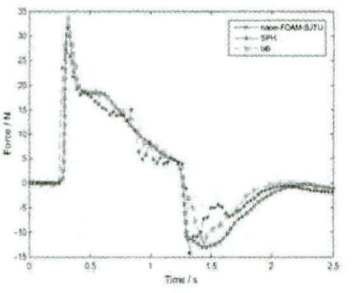

图 6　方柱受到的拍击力随时间变化曲线

4.2　GPU 计算效率

文章通过将 GPU 计算所耗费时间与 CPU 所耗费时间进行对比，研究 GPU 的加速效率问题。本文所采用的 GPU 为 GTX970，该显卡拥有 1664cores，4GB 显存空间，主频为 1.18GHz，显卡驱动为 7.0；所采用 CPU 为 i7-4970k（四核八线程），主频为 4.00GHz，可自动睿频至 4.5GHz，内存为 16GB，测试平台为 win8.1、64 位系统。本节分别取粒子间距 0.06，0.05，0.04，0.035，对应粒子数 440118，727161，1345851，2010536 来测试不同粒子数下 CPU 和 GPU 的计算效率。

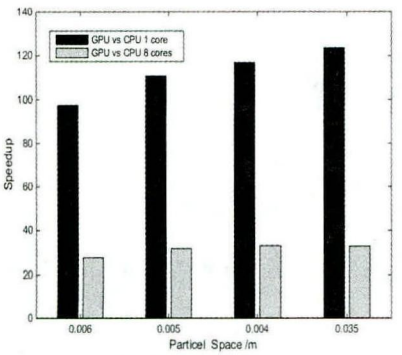

图 7　不同粒子间距下 GPU VS CPU 获得的加速比

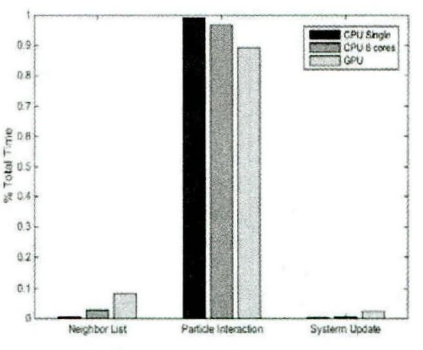

图 8　粒子间距为 0.04 时各部分所占时间比重

图 7 显示了不同粒子间距下，GPU 相对较于 CPU 单线程和 8 线程并行所获得的加速比。可以看到，对于 GPU 相较于单线程可以获得最高 123 倍的加速比，而相对于 8 线程并行则可以获得最高 33 倍的加速比，并且加速比随着粒子数的增多有增大的趋势，这是因为当粒子数足够多的时候，GPU 的多线程能被充分利用，从而可以有效隐藏计算延迟，因而能最大发挥 GPU 多线程并行的优势。

可以看到，运用 GPU 可以获得相当可观的加速比。为了更好地理解这其中的加速机制，

本小节以粒子间距为0.04时为例，统计了SPH算法中三个主要部分：邻居粒子搜索（NL）、粒子间相互作用（PI）以及系统更新（SU）在运行时间中所占的比例，如图8所示。在使用单线程CPU时，PI在运行时间中所占的比例高达99.1%，而在使用8线程并行时，PI所占比例则降至96.8%。使用GPU并行时，PI所占比例则为89.3%，同时NL和SU所占的比例则增大了。从以上分布及变化情况并综合加速比的变化可以推测出:NL以及SU相对于PI来说是有着很高计算效率的；同时，随着并行程度的增大，PI在总时间中的比例逐渐减少，而整体的效率则在不断增加；加速的有效性也说明了SPH算法是可以有效分解为数据流并行问题的。

5 结论

本文基于DualSPHysics讨论了GPU技术在SPH方法中的实现，分析了GPU编程与CPU编程的异同，重点研究了GPU编程中邻居粒子搜索的实现细节。然后通过模拟一个带立柱的三维溃坝，并与实验结果进行对比，验证了GPU代码的有效性。至于计算效率，则是在粒子数分别为440118，727161，1345851，2010536的情况下，比较了单线程CPU、8线程CPU并行与GPU并行的计算时间。结果表明，使用GPU最高可比单线程CPU提速两个量级左右，比之8线程并行也有三十倍左右。为了进一步了解GPU的加速情况，本文统计了粒子数为1345851时各主要计算部分所占时间比例的情况分布。可以发现，随着并行程度的增大，最耗时的粒子相互作用计算部分在整体中的占比不断下降。综上所述，使用GPU技术可以在无网格粒子法SPH中获得相当可观的加速比。

致谢

本文工作得到国家自然科学基金项目（Grant Nos 51379125，51490675，11432009，51411130131），长江学者奖励计划(Grant No. 2014099)，上海高校特聘教授（东方学者）岗位跟踪计划(Grant No. 2013022)，国家重点基础研究发展计划（973计划）项目（Grant No. 2013CB036103），工信部高技术船舶科研项目的资助。在此一并表示衷心感谢。

参 考 文 献

1 MONAGHAN J J. Energy distribution in a particle alpha model [J]. Journal of Turbulence，2004，5: 22.
2 Zhang, Y.X. ,Tang, Z.Y., D.C.Wan. (2013). A Parallel MPS Method for 3D Dam Break Flows. Proc. 8th Int. Workshop on Ship Hydro., IWSH-2013, Septemper 23, 2013 – Septemper 25, 2013, Seoul, Korea, 135-139.
3 Yuxing Zhang, Zhenyuan Tang, Yaqiang Yang, Decheng Wan (2014),Parallel MPS Method for Three-Dimensional Liquid Sloshing, The Twenty-fourth International Ocean and Polar Engineering Conference, Busan, Korea, 257-265
4 Lucy L B. A approach to the testing of the fission hypothesis [J]. The Astsion，1977，8(12) : 1013-1024.
5 LIBERSKY L D, RANDLES P W, CARNEY T C, et al.Recent improvements in SPH modeling of hypervelocity impact[J]. International Journal of Impact Engineering,1997, 20(6-10): 525-532. 6
6 LIU M B, LIU G R, LAM K Y, et al. Smoothed particle hydrodynamics for numerical simulation of underwater explosion[J]. Computational Mechanics, 2003, 30(2): 106-118.
7 A.J.C. Crespo, J.M. Domínguez, et al, DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH), Computer Physics Communications, Volume 187, February 2015, Pages 204-216, ISSN 0010-46559
8 FLUCK D A，QUINN D W. An analysis of 1-D smoothed particle hydrodynamics Kernels [J]. Journal of Computational Physics，1996，126: 699-709.
9 Monaghan, J.J.,1992. Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics

30,543–574.
10 Monaghan,J.J, et al ,1999.Gravity currents descending a ramp in a stratified tank. Journal of Fluid Mechanics 379,39–70.13
11 CUDA_C_Programming_Guide. NVIDIA, CUDA Toolkit Doc , 2015.
12 Dominguez J M, Crespo A J C, Gómez - Gesteira M, et al. Neighbour lists in smoothed particle hydrodynamics[J]. International Journal for Numerical Methods in Fluids, 2011, 67(12): 2026-2042.
13 曹洪建, 万德成, 杨驰. 三维溃坝波绕方柱剧烈流动的数值模拟[J].水动力学研究与进展 A 辑, 2013, 4: 008.
14 WU T. A numerical study of three-dimensional breaking waves and turbulence effects[D]. Cornell University,Ithaca, USA, 2004.

A study of GPU-acceleration for SPH method

LI Hai-zhou, TANG Zhen-yuan, WAN De-cheng *

(State Key Laboratory of Ocean Engineering, School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, Collaborative Innovation Center for Advanced Ship and Deep-Sea Exploration, Shanghai 200240, China)
*Corresponding author, Email: dcwan@sjtu.edu.cn

Abstract: Mesh-free methods can effectively deal with problems with instantaneous very large defomation. However, following the increase of the particle number, the calculation efficiency becomes a bottleneck for applying the method to the engineering practice. With the using of GPU technology, it can greatly improve the computing efficiency. Based on open source software DualSPHysics, this paper studies the implementation of SPH method on GPU, and discusses the specific details of the neighbor search method. Then, the simulation of a three dimensional dam break with square cylinder is used to verify the effectiveness of GPU in the application of SPH method. Numerical results show that by using the GPU acceleration, up to two orders of magnitude speedup can be obtained in comparison with the CPU.

Key words: Messless method; SPH; Dam-breaking wave; GPU; Neighbor list search