基于 GPU 的分子动力学模拟 Cell Verlet 算法实现及其并行性能分析

张 帅[1,2] 徐 顺[1,3] 刘 倩[1,3] 金 钟[1,3]

(中国科学院计算机网络信息中心 北京 100190)[1] (中国科学院大学 北京 100049)[2]

(中国科学院计算科学应用研究中心 北京 100190)[3]

摘 要 分子动力学模拟存在空间和时间的复杂性,并行加速分子的模拟过程尤为重要。基于 GPU 硬件数据并行架构的特点,组合分子动力学模拟的原子划分和空间划分的并行策略,优化实现了短程作用力计算 Cell Verlet 算法,并对分子动力学核心基础算法的 GPU 实现做了优化和性能分析。Cell Verlet 算法实现首先采用原子划分的方式,将每个粒子的模拟计算任务映射到每个 GPU 线程,并采用空间划分的方式将模拟区域进行元胞划分,建立元胞索引表,实现粒子在模拟空间的实时定位;而在计算粒子间的作用力时,引入希尔伯特空间填充曲线方法来保持数据的线性存储与数据的三维空间分布的局部相关性,以便通过缓存加速 GPU 的全局内存访问;也利用了访存地址对齐和块内共享等技术来优化设计 GPU 分子动力学模拟过程。实例测试与对比分析显示,当前的算法实现具有强可扩展性和加速比等优势。

关键词 分子动力学,Cell Verlet 算法,GPU 异构计算,互斥同步优化,访存局部性

中图法分类号 TP338.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.10.054

Cell Verlet Algorithm of Molecular Dynamics Simulation Based on GPU and Its Parallel Performance Analysis

ZHANG Shuai[1,2] XU Shun[1,3] LIU Qian[1,3] JIN Zhong[1,3]

(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)[1]

(University of Chinese Academy of Sciences, Beijing 100049, China)[2]

(Center of Scientific Computing Applications & Research, Chinese Academy of Sciences, Beijing 100190, China)[3]

Abstract Molecular dynamics simulation has complexity in spatial and temporal scale, therefore it is critical to optimize the simulation process of molecular dynamics. Based on the characteristics of data parallel on GPU hardware architecture, this paper combined atomic partition and spatial partition of molecular dynamics simulation and optimized the Cell Verlet algorithm for the short-range force calculation, and also designed the core and basic algorithms of molecular dynamics on GPU with optimization and performance analysis. The implementation of Cell Verlet algorithm in this paper is started with atomic partition, each task of particle simulation is mapped to each GPU thread, and then the simulated space is divided into cellular area in the spatial partition way. The cellular index table is established, and the real-time locating simulation particles are realized. Meanwhile, in the force calculation between particles, Hilbert's space-filling curve algorithm was introduced to keep the partial correlation of particle's space layout in line with linear data storage, in order to cache and accelerate the access on GPU global memory. This paper also used the memory address alignment and block sharing memory technology to optimize the design of GPU molecular dynamics simulation process. Testing and comparative analysis of examples show that the current implementation of algorithm has advantages in high parallelism and speedup.

Keywords Molecular dynamics, Cell Verlet algorithm, GPU heterogeneous computing, Mutex and synchronization optimization, Locality of memory access

---

到稿日期:2017-12-16 返修日期:2018-03-05 本文受国家重点研发计划:高能物理高性能计算应用软件系统规模化应用(2017YFB0203203),中国科学院科研信息化工程项目:生物医药计算模拟软件(XXH13506-404),中国科学院青年创新促进会基金(会员号2016156)资助。

张 帅(1990—),男,硕士生,主要研究方向为 GPU 并行优化;徐 顺(1980—),男,博士,CCF 会员,主要研究方向为高性能计算与分子模拟,E-mail:xushun@sccas.cn;刘 倩(1981—),女,博士,副研究员,主要研究方向为高性能计算;金 钟(1974—),男,博士,研究员,CCF 会员,主要研究方向为高性能计算化学,E-mail:zjin@sccas.cn(通信作者)。

1 引言

分子动力学(Molecular Dynamics,MD)模拟是一种被广泛使用的原子分子尺度的计算建模方法,是基于原子和分子间的作用势,并根据牛顿运动力学原理所发展而来的计算方法。迄今,人们已系统地建立了许多适用于生化分子体系、聚合物、金属与非金属材料的力场,使得计算复杂体系结构与一些热光性质的能力及精准性得到大幅提升[1]。分子动力学模拟中非成键作用力的计算是最基础和最关键的部分。面对千万级原子以上规模的大体系应用场景带来巨大计算量的挑战,设计出高效的并行加速算法显得十分有意义[2]。

由于半导体晶体管电路空间和能耗的限制,发展异构加速的高性能计算技术成为了一种提高"性能-能耗比"的有效方法。可编程的 GP-GPU(通用图像处理单元)就是一种典型的异构体系,它使用了数据线程的并行模式,具有比 CPU 更强大的峰值计算能力和更高的理论存储器带宽;GPU 芯片使得摩尔定律仍存在可观的发展空间。GPU 特殊的硬件架构特别适合高数据并行和高密集计算型的应用场景,而分子动力学中计算作用势具有这种计算密集型的特征。

国际上主流的分子动力学模拟软件 LAMMPS,NAMD,Amber 和 GROMACS 等均以各种方式引入 GPU 进行计算加速,并取得了良好的加速效果[3-4]。其利用 GPU 主要加速非成键作用力的计算,包含短程作用力中的近邻列表构建和长程作用力中的实空间和虚空间计算部分,如范德华势和静电势;而长程作用力的实空间计算是在截断半径之内的计算,类似于短程作用力的计算。可以说,短程作用力计算是分子动力学模拟中最核心的部分,往往决定了体系模拟的整体效率。非成键作用力的计算需要考虑体系中两粒子之间的作用,其计算复杂度为 $O(N^2)$,N 为体系粒子数。而 Verlet List 算法[5]针对短程作用力采用截断半径进行处理,忽略了截断远处的作用,使得力的计算复杂度为 $O(N)$;但该方法在列表更新时,粒子搜索的复杂度仍然为 $O(N^2)$,对大体系模拟的加速效果并不明显。Cell List(或 Cell Linked List)方法[6]将模拟体系空间按截断半径的距离划分 Cell,在周期性条件下每个 Cell 中的粒子在 26 个 Cell 邻居和自己所处的 Cell(共 27 个 Cell)中查找。由于 Cell 的空间划分是已知的,搜索 27 个 Cell 比搜索整个体系的空间明显减小,粒子映射到 Cell 的复杂度为 $O(N)$,更新粒子列表的复杂度为 $O(N)$;但截断半径相同时,相比于 Verlet List 方法,Cell List 方法待计算的粒子列表明显较多。因此通常将 Cell List 算法和 Verlet List 算法进行组合以实现优势互补,本文称这种方法为 Cell Verlet 算法。该方法可有效实现大规模体系的短程作用力的计算。

目前,研究者已提出了一些针对 MD 的 GPU 异构计算并行算法。在国外,Michael Brown 等人为 LAMMPS 增加了 GPU 加速计算的模块[3],GROMACS 软件也在最新版本中优化了 GPU 的计算模块[4],同时也发展了专门面向 GPU 环境的 MD 模拟软件 HOOMD-blue[7]。在国内,文献[8]属于较早期的 GPU 优化 MD 的工作,所提方法在计算粒子间的相互作用力时须计算每个粒子与系统内所有其他粒子的作用力。该方法虽然利用了 GPU 的共享内存来加速计算,但是在计算力时没有充分利用截断半径,在算法策略上存在明显不足。另一种基于 GPU 并行的方法[9]采用了元胞列表法组织和检索数据,但是该方法中每个 GPU 线程块对应一个模拟元胞网格,每个线程负责该网格内的一个粒子信息更新,当面对粒子不均匀分布的体系模拟时,元胞网格内的粒子数目不均匀会导致 GPU 线程块内的负载不均匀,其中因分子位置变化,需重新将 GPU 线程块映射到网格,网格映射的逻辑操作较复杂;将属于本网格的分子插入到映射数组中,为了避免内存写入冲突,在移出不属于本网格的分子前还需要备份映射数组,大大增加了 GPU 的内存开销。基于现有研究,我们设计了 Cell Verlet 算法的一种基于 GPU 的有效实现方式,该方式考虑了原子划分[10]与空间划分[11]相结合的策略,在访存方面做了较大的优化,并充分利用了 GPU 数据并行的特点,具有负载均衡和高并行度的特性,我们把这种并行实现方式和一些 MD 关键步骤的实现称作 Cell MD。

Cell MD 模拟的并行策略核心在于粒子划分的并行计算方式。以粒子的角度分析整个物理情景,每个粒子只需要计算它与其周围截断半径之内的粒子间的短程作用力即可,而这些粒子计算各自所受到周围粒子的作用力时是相互独立的,该过程本身是天然的并行操作,于是如何快速地计算每个粒子与其周围截断半径之内的粒子间的短程作用力成为了问题的核心。我们采用空间划分思想,首先采用元胞划分法(Cell List)进行模拟空间的划分,再次实现 Verlet List 方法以建立索引表,从而加快粒子间作用力的计算,同时加入其他 GPU 优化技术以提高整个 MD 的计算性能。

2 算法的原理与实现

2.1 分子动力学模拟

分子动力学模拟是对系统粒子的动力学进行计算模拟,粒子的速度、位移等动力学信息通过计算受力对牛顿运动方程求解而得到。通常采用有限差分法来解决运动方程对粒子速度和位移存在的精度低和离散时间不同步等缺点。本文采用 Velocity-Verlet 积分算法[5,12]进行求解,分析其中第 i 号粒子与第 j 号粒子之间的作用力,定义为 $F_{ij}=-\nabla U(r_{ij})$,其中 U_{ij} 为两个粒子之间的相互作用势,用 a_i 表示第 i 号粒子加速度,m_i 表示粒子 i 的质量,由牛顿力学得 $m_i\times a_i=\sum_j F_{ij}$。其中,第 i 号粒子在 $t+1$ 时刻的速度为 $v_i(t+1)$,其更新公式为:

$$v_i(t+1)=v_i(t)+\frac{a_i(t)+a_i(t+1)}{2}\mathrm{d}t$$

其中,$a_i(t)$表示第 i 号粒子在 t 时刻的加速度,$\mathrm{d}t$ 表示离散时间步。第 i 号粒子在 $t+1$ 时刻的位置坐标 $r_i(t+1)$的更新公式如下:

$$r_i(t+1)=r_i(t)+v_i(t)\mathrm{d}t+\frac{a_i(t)}{2}\mathrm{d}t^2$$

范德华作用势是典型的短程作用力,本文以 Lennard-Jones 势来建模范德华短程作用势,其粒子间作用的表达式为:

$$U(r_{ij})=\begin{cases}4\varepsilon\left[a\left(\frac{\sigma}{r_{ij}}\right)^{12}-b\left(\frac{\sigma}{r_{ij}}\right)^{6}\right], & r_{ij}\leqslant r_c\\ 0 & r_{ij}>r_c\end{cases}$$

其中，ε 表示势能阱的深度，σ 为互相作用的势能正好为零时的两体距离（即平衡距离），一般情况下，不同粒子类型的 ε 和 σ 不同，为了方便，这里取同种类型。r_{ij} 为粒子 i 到粒子 j 的距离，r_c 为作用力的截断距离。系数 a 和 b 分别表示斥力项和引力项的相对大小。

2.2　Cell Verlet 算法在 GPU 体系中的实现

Cell Verlet 算法首先将体系划分为 Cell（元胞）形式，每个粒子被映射到了相应的 Cell 中，再基于这些元胞为每个粒子构建 Verlet List，这个 Verlet List 反映了粒子之间近邻关系下的短程相互作用。本文构建的三维元胞的立方体边长为截断半径或者稍微大于截断半径，对于 Cell 中的每个粒子，构建其 Verlet List 是基于 Cell List 的信息的，在三维空间中，搜索其所在的 Cell 和 26 个近邻 Cell，将每个近邻原子的索引号记录在 Verlet List 中以便于加速相互作用力的计算，以上即为 Cell Verlet 的核心思想。该算法的示意图如图 1 所示。粒子的空间结构以二维形式表示，并以 2 号粒子为例，图 1 显示了相应的 Cell List 和 Verlet List 的基本结构。体系的总粒子数为 N，Cell 的个数为 Nc，截断距离为 r_{cut}。

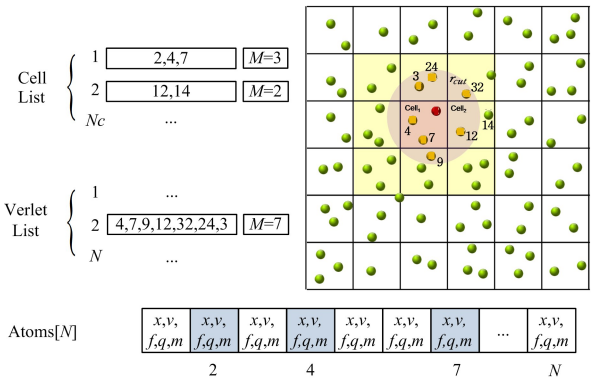

图 1　Cell Verlet 算法示意图

Fig. 1　Schema of Cell Verlet algorithm

针对 GPU 架构，我们设计了一个 CellTable 数据结构来实现 Cell List。该结构是一个关于粒子信息的索引表，每个元胞对应该表的一行，表的行数等于模拟区域划分的元胞个数 Nc，表的每一行实际上是一个数组，其中记录着当前对应元胞里存在的粒子索引号，数组的最后一个元素记录该数组包含多少个粒子。考虑到 GPU 全局内存访问的效率问题，CellTable 采用等长的二维形式，对每个 Cell 一次性预分配等长的数组空间，如果某个 Cell 预分配的空间不够，那么整个 CellTable 需要重新开辟更长的数组长度。

构建元胞索引表，也就是将粒子映射到相应的元胞，该步骤的计算任务放到 GPU 上完成。其中将每个粒子映射到相应元胞，并设计一个 GPU Kernel 函数负责该粒子的映射过程，此过程中需要根据粒子的空间坐标信息计算出其所处的 Cell，可通过以下计算公式实现：

$$r_{\text{cell}}[d]=\left|\frac{r_i[d]}{L_{\text{cell}}[d]}\right|$$

其中，$r_{\text{cell}}[d]$表示第 i 号粒子被映射到的元胞的第 d 维度的坐标，$r_i[d]$表示粒子 i 的第 d 维度坐标，$L_{\text{cell}}[d]$表示元胞的第 d 维长度。在单 GPU 情况下，元胞索引表被加载（memcpy）到 GPU 的全局内存，由相应的 GPU Kernel 函数调用。接下来构建 Verlet List，对每个元胞中的粒子，只需搜索与其同属于 1 个元胞及其周围 26 个邻居元胞中的粒子，使用截断距离来取舍近邻粒子，在截断距离内的粒子将添加到该粒子对应的 Verlet List 中，在单 GPU 情况下，同样将整个 Verlet List 加载到 GPU 全局内存中，并对每个粒子启动一个 GPU Kernel 函数来完成 Verlet List 的构建。构建的 Cell List 与 Verlet List 类似，GPU 的线程并发度都为 N。基于获得的 Verlet List 数据信息，就可为每个粒子设计一个 GPU Kernel 以完成其作用势能的计算，之后就可以进行牛顿第二定律的微分方程求解，进一步更新粒子的坐标和速度，形成动力学模拟。Cell MD 的整个流程如图 2 所示。

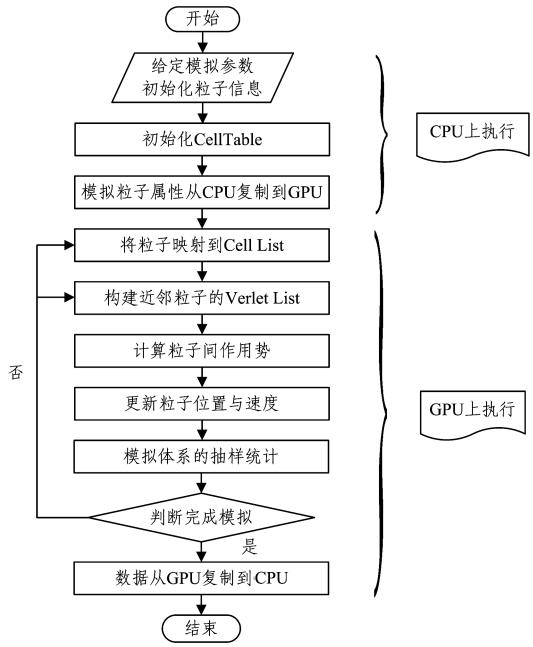

图 2　Cell MD 算法的流程图

Fig. 2　Flowchart of Cell MD algorithm

本文除了要实现 Cell MD 的主要流程外，还需要添加多种 GPU 优化以加速 MD 的模拟过程，主要包括并行计算时的互斥同步和 CPU-GPU 访存优化。在同步互斥方面，构建 Cell List 时，在 GPU Kernel 计算的 Block 空间内，例如 128 个线程组，要对同一个 Cell List 添加粒子索引操作，必须实现 List 表的临界区保护；构建 Verlet List 时每个粒子被对应映射到一个线程，每个线程只负责单个原子的 List 的构建，不会产生互斥问题。另外一个需要进行互斥操作的是在计算每个粒子受其他粒子作用力的累加时，需要施加原子操作。GPU 的同步也需要用到互斥，如构建 Cell List 和构建 Verlet List 的两个 GPU Kernel 时需按先后顺序执行，Kernel 上的 Block 线程块访问共享内存时也需要同步。在计算体系动能时，为了减少对全局共享变量的互斥累加，开辟 Block 内的 shared 内存数组 block_var，维数 Dim_Block 等于 Block 内的线程数，定义形式为：

```
_shared_real block_var[Dim_Block]
```

每个 Block 内的线程将结果保持到块内数组索引为线程 id 的元素空间中，这个过程不产生互斥，最后由 $id=0$ 的线程将这个块内数组值累加，从而避免了更多的全局变量锁操作。

在访存优化方面，主要在 GPU Kernel 线程访问粒子的

信息数据和查找元胞索引表两方面做了优化。其中,我们对粒子信息的结构体数据做了地址对齐操作,加速了 Kernel 线程的合并读取,对于 GPU Kernel 连续线程,访问 GPU 的全局内存并多次使用该信息,将 GPU 线程块的共享内存当作临时变量,将要多次使用的粒子信息本地化到共享内存,加速了粒子的访问。此外,在计算粒子间的相互作用力时,考虑了内核线程访问元胞索引表的空间局部性,采用希尔伯特填充曲线[13-14]方式来组织元胞索引表数组,使得空间相邻的元胞索引表数据在线性数组上也相邻,加速了对元胞索引表的访问。

3 算法实现特性的分析

在整个 MD 的步骤中,Cell MD 算法实现了元胞索引表的构建、粒子间力的计算,每个粒子的这些计算任务都是并行完成的,负责粒子的这些计算任务的 GPU Kernel 线程之间几乎不需要数据通信,计算量主要在于粒子间相互作用力的计算上,每个粒子的周围元胞中的邻居粒子个数是一个与粒子规模无关的有限常量,因此所提算法的理论复杂度为 $O(N)$。在实际模拟中构建 Cell Verlet 列表时,可以对 r_{cut} 添加一个增量 r_{skin},使得构建出来的近邻列表可以间隔几步更新以便节省时间。计算力时,各个粒子的计算任务是完全独立的,充分利用 GPU 线程的块内并行与块间并行来完成并行加速计算。Cell 的数量 Nc 取决于 r_{cut} 的大小,与体系粒子数 N 无直接关联,Nc 的增长率远小于 N。

Cell MD 在 GPU 上的实现方式也可以推广到 OpenMP 等多线程并行编程环境。当前该算法是针对单 GPU 设计的,对于 GPU 全局内存大的环境,其具有非常好的适用性。在多 GPU 的环境下,Cell MD 可以在 GPU 的统一内存(unified memory)模式下得到直接推广。

4 实验结果与分析

本文采用经典 Lennard-Jones 势的体系作为测试用例,并使用了约化单位和 Velocity-Verlet 积分算法,模拟氩气在短程作用力下的演化过程。实验的两个 GPU 测试平台分别为 Intel 平台和 IBM 平台,它们都搭配了多块 GPU,具体配置如表 1 所列。

表 1 测试平台的参数

Table 1 Parameters of testing platforms

参数	Intel 平台	IBM 平台
GPU	Tesla K20m,5.12GB (通过 PCIe 与 CPU 相连)	Tesla 100-SXM2-16GB (通过 Nvlink 与 CPU 相连)
CPU	Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz	IBM POWER8 NVL 4.0GHz revision:1.0
内存	DDR3 128GB	DDR4 256GB

Cell MD 程序采用单精度 float 编译,编译选项为:-arch sm_35-O3-Xcompiler "-g-Wall-O3-fopenmp"。

4.1 模拟体系的可扩展性分析

本节测试文中所设计实现的 Cell Verlet 算法在两个 GPU 平台下的并行扩展性,每个平台都选用单块 GPU 卡测试。实验结果如图 3 所示,相同模拟步数下消耗的时间随粒子规模的变化基本呈线性关系,符合该算法的计算复杂度 $O(N)$;同时可见,间隔几步更新近邻列表能提高计算速度,GPU 的 P100 与 K20m 产品的 Cell MD 运算速度比约为4∶1。

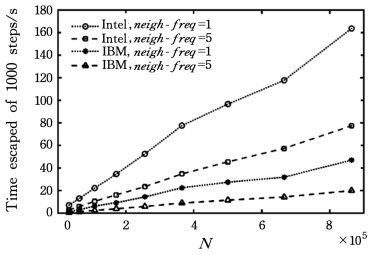

图 3 Cell Verlet 算法在两种平台上模拟 1000 步定长的耗时

Fig. 3 Time consuming of 1000 steps simulated by Cell Verlet algorithm on two platforms

4.2 与 LAMMPS GPU 模块的相应算法的对比

对比 Cell MD 的实现与 LAMMPS 的 GPU 模块,选用同一个 Lennard-Jones 势,测试不同体系大小下 1000 步的耗时,都选用同一个 GPU 卡(Intel 平台),结果如图 4 所示。由于 LAMMPS GPU 模块是基于空间划分方法的,与 Cell MD 的实现有显著差异,在每步更新近邻列表时($neigh\text{-}freq=1$),Cell MD 实现的速度更快,当增加近邻列表构建的步数间隔时,两者的速度相近。由于两者的实现方式存在较大差异,内部细节很多,该结果只提供对该实例整体计算效率的评估参考。

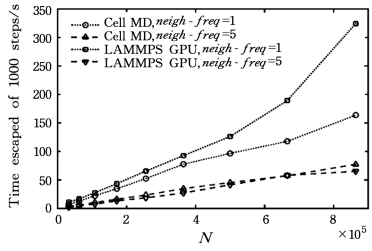

图 4 Cell MD 与 LAMMPS 的 GPU 模块比较

Fig. 4 Comparison of GPU module of Cell MD and LAMMPS

4.3 与 OpenMP 多线程设计的加速性能的对比

为了对比 GPU 与 OpenMP 的效率问题,同时考虑测试的准确性,我们对比了 Cell List 算法在多核 CPU 的实现与 GPU 的实现,在 Intel 平台上,测试并对比各种算法模拟等长步数的时间消耗。如图 5 所示,随着 CPU 多线程的开启,模拟 1000 步的时长变短,但并不是线程越多速度越快,而是存在一个上限,大约超过 12 个 CPU 多线程时,Cell List 算法的速度便不再显著提升,而单个 GPU 上的运行耗时非常低,OpenMP 多线程的实现方式存在一个性能加速上限,该上限也不及 GPU 的加速性能。

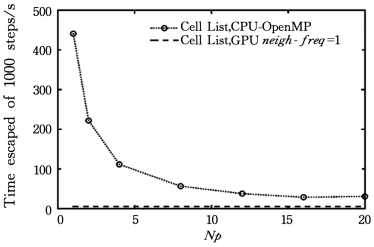

图 5 Cell List 算法与 GPU 和 OpenMP 实现方式的耗时对比

Fig. 5 Time-consuming of Cell List algorithm compared with GPU and OpenMP implementations

(下转第 299 页)

[3] MEYER D. The Software-Defined-Networking Research Group [J]. IEEE Internet Computing,2013,17(6):84-87.

[4] BU C,WANG X,HUANG M,et al. SDNFV-based Dynamic Network Function Deployment: Model and Mechanism [J]. IEEE Communications Letters,2018,22(1):93-96.

[5] BU C,WANG X,CHENG H,et al. Enabling Adaptive Routing Service Customization via the Integration of SDN and NFV[J]. Journal of Network & Computer Applications,2017,93:123-136.

[6] LV J,WANG X,HUANG M,et al. RISC:ICN routing mechanism incorporating SDN and community division [J]. Computer Networks,2017,123:88-103.

[7] HELLER B,SHERWOOD R,MCKEOWN N. The controller placement problem [J]. Acm Sigcomm Computer Communication Review,2013,42(4):7-12.

[8] KOPONEN T,CASADO M,GUDE N,et al. Onix:a distributed control platform for large-scale production networks [C]// Usenix Conference on Operating Systems Design and Implementation. USENIX Association,2010:351-364.

[9] HASSAS YEGANEH S,GANJALI Y. Kandoo:a framework for efficient and scalable offloading of control applications [C]// The Workshop on Hot Topics in Software Defined Networks. ACM,2012:19-24.

[10] CURTIS A R,MOGUL J C,TOURRILHES J,et al. DevoFlow: Scaling flow management for high-performance networks [J]. Acm Sigcomm Computer Communication Review,2015,41(4):254-265.

[11] CURTIS A R,MOGUL J C,TOURRILHES J,et al. DevoFlow: Scaling flow management for high-performance networks [J]. Acm Sigcomm Computer Communication Review,2015,41(4):254-265.

[12] KANG N X,LIU Z M,JENNIFER R,et al. Optimizing the one big switch abstraction in software-defined networks[C]// ACM Conference on Emerging NETWORKING Experiments and Technologies. 2013:13-24.

[13] SYRIVELIS D, PARISIS G, TROSSEN D, et al. Pursuing a Software Defined Information-centric Network [C]// European Workshop on Software Defined Networking. 2012:103-108.

[14] BARATH R,MARTÍN C,TEEMU K,et al. Software-defined internet architecture: decoupling architecture from infrastructure [C]// ACM Workshop on Hot Topics in Networks. 2012:43-48.

[15] MCKEOWN N,ANDERSON T,BALAKRISHNAN H,et al. OpenFlow:enabling innovation in campus networks [J]. Acm Sigcomm Computer Communication Review,2008,38(2):69-74.

[16] Aaron. PIPELINING[EB/OL]. http://blog.163.com/sunshine_linting/blog/static/4489332320117250104945 4.

[17] OpenFlow Switch Specification, Version 1.1.0[EB/OL]. http://archive.openflow.org.

(上接第 294 页)

结束语　本文在 GPU 上实现了分子动力学模拟核心算法——Cell Verlet 算法,并设计了相应的模拟程序 Cell MD。通过实验分析,Cell MD 的算法实现具有可扩展性,相比于多核 CPU,其具有明显优势,能够达到当前主流 LAMMPS 软件同等的并行效率。后期,我们考虑将 Cell MD 实现扩展到多 GPU 环境,以进一步提升加速比。

参 考 文 献

[1] SCHLICK T,COLLEPARDOGUEVARA R,HALVORSEN L A,et al. Biomolecular modeling and simulation:a field coming of age[J]. Quarterly Reviews of Biophysics,2011,44(2):191.

[2] YUKO O. Molecular simulations in generalised ensemble[J]. Molecular Simulation,2012,38(14-15):1282-1296.

[3] BROWN W M,WANG P,PLIMPTON S J,et al. Implementing molecular dynamics on hybrid high performance computers-short range forces [J]. Computer Physics Communications,2011,182(4):898-911.

[4] ABRAHAM M J,MURTOLA T,SCHULZ R,et al. GROMACS:High performance molecular simulations through multi-level parallelism from laptops to supercomputers[J]. Softwarex,2015,1-2(C):19-25.

[5] VERLET L. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules[J]. Health Physics,1967,22(1):79-85.

[6] MATTSON W,RICE B M. Near-neighbor calculations using a modified cell-linked list method[J]. Computer Physics Communications,1999,119(2-3):135-148.

[7] GLASER J,NGUYEN T D,ANDERSON J A,et al. Strong scaling of general-purpose molecular dynamics simulations on GPUs[J]. Computer Physics Communications,2015,192:97-107.

[8] FEI H,ZHANG Y Q,WANG K,et al. Parallel Algorithm and Implementation for Molecular Dynamics Simulation Based on GPU [J]. Computer Science,2011,38(9):276-278. (in Chinese)
费辉,张云泉,王可,等. 基于 GPU 的分子动力学模拟并行化及实现[J]. 计算机科学,2011,38(9):276-278.

[9] CHEN F G,GE W,LI J H. Implementation of Complex Multiphase Flow Molecular Dynamics Simulation on GPU [J]. Chinese Science (Series B:Chemistry),2008,38(12):1120-1128. (in Chinese)
陈飞国,葛蔚,李静海. 复杂多相流动分子动力学模拟在 GPU 上的实现[J]. 中国科学:B 辑,2008,38(12):1120-1128.

[10] PLIMPTON S. Fast parallel algorithms for short-range molecular dynamics[J]. Journal of Computational Physics,1995,117(1):1-19.

[11] PLIMPTON S,HENDRICKSON B. A new parallel method for molecular dynamics simulation of macromolecular systems[J]. Journal of Computational Chemistry,1994,17(3):326-337.

[12] GREST G S,DÜNWEG B,KREMER K. Vectorized link cell Fortran code for molecular dynamics simulations for a large number of particles[J]. Computer Physics Communications,1989,55(3):269-285.

[13] HILBERT D. Über die stetige Abbildungeiner Linie auf ein Flächenstück[M]// Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes. Berlin: Springer,1935,38(3):459-460.

[14] BUTZ A R. Alternative Algorithm for Hilbert's Space-Filling Curve[J]. IEEE Computer Society,1971,20(4):424-426.