基于 GPU 的 Hartree-Fock 与密度泛函算法及程序

王岩^a 田英齐^{b,c} 金钟^{*,b} 索兵兵^{*,a}

(^a 西北大学 现代物理研究所 陕西省理论物理前沿重点实验室 西安 710127)

(^b 中国科学院计算机网络信息中心 北京 100190)

(^c 中国科学院大学 北京 100049)

摘要 基于图形处理单元(GPU)的算法和程序为解决量子化学中的计算瓶颈开辟了道路。作者设计了基于 GPU 的量子化学算法和程序,实现了 Hartree-Fock 方法和密度泛函理论中双电子排斥积分计算、Fock 矩阵构造以及交换相关泛函的计算。由于计算内核使用 OpenCL 编程框架,程序可以在多种架构的计算设备上执行。对于不同计算模块和分子自洽场计算的测试表明,基于 OpenCL 的 GPU 程序相比 CPU 上的串行程序实现了最快 148 倍的加速。

关键词 图形处理单元; OpenCL; Hartree-Fock; 密度泛函理论; 直接自洽场计算

Hartree-Fock and Density Functional Calculations on Graphics Processing Unit

Yan Wang^a Yingqi Tian^{b,c} Zhong Jin^{*,b} Bingbing Suo^{*,a}

(^a Institute of Modern Physics, Shaanxi Key Laboratory for Theoretical Physics Frontiers, Northwest University, Xi'an 710127, China)

(^b Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

(^c University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract Graphics processing units (GPUs) have become a promising architecture to tackle many computational bottlenecks in quantum chemistry calculations. Herein, we present our new development on using GPU to accelerate Hartree-Fock (HF) and density functional theory (DFT) calculations in Beijing Density Functional (BDF) Package. Our program utilizes the OpenCL platform and thus can execute on a variety of computing devices from different companies as NVIDIA and AMD. All time-consuming steps in HF/DFT, such as calculation of electron repulsion integrals (ERIs), the formation of the Fock matrix, and the exchange-correlation (XC) functional related works, have been implemented on the GPU. In our algorithm, the coulomb- and exchange-matrix are calculated directly on GPU by contracting the primitive ERIs with the density matrix. The 1T1PI (1 thread 1 primitive integral) algorithm in which each thread evaluates one primitive ERI, is used to schedule the computational tasks on GPU. To achieve this task, the primitive Gaussian basis shell pairs $\mu\nu$ are first pre-screened and sorted according to their values. The Gaussian product theorem (GPT) is applied to each shell pairs and the intermediate values are calculated and copied into the GPU memory for further use. Then, the one-dimensional mapping is used to assign 32 work items (threads) into one workgroup to calculate the J matrix element and the permutation symmetry of the primitive ERIs is fully utilized. To calculate the K matrix, two-dimensional mapping is used and every 64 work items are packed into one workgroup. Permutation symmetry of exchanging the bra pair $\mu\lambda$ and the ket pair $\nu\sigma$ is ignored for reducing the expensive commutation between different workgroups on GPU. After a batch of coulomb- or exchange-matrix elements are computed on the GPU, the results are copied back to the CPU and accumulated to the Fock matrix. The XC terms are calculated through a numerical procedure due to the complex form of the XC functionals. We first pack the numerical grids as batches in which one batch has 128 grids. Then the none zero Gaussian basis shells on each grid batch are sifted out. The grid batches and the basis function sieving indices are duplicated on CPU and GPU memory to avoid unnecessary communication between CPU and GPU. The computational tasks are scheduled dynamically according to the grid batches on GPU. All steps as calculating the numerical grids and their weights, electron density and density gradient, the XC functional and its derivative, and the XC energy and the matrix elements of the XC potential, are optimized step by step on GPU. All calculations are carried out in 64-bit double-precision accuracy to achieve the same numerical precision as on the CPU. Benchmark calculations are carried out on several different GPUs from NVIDIA and AMD for assessing the performance of our code. The benchmark result indicates that the algorithm implemented on the GPU can achieve up to 148-fold speedup over a serial CPU implementation, and the total energy calculated on the GPU is as accurate as the resulting calculated on the CPU.

Keywords GPU; OpenCL; Hartree-Fock; density functional theory; direct self-consistent-field calculation

* E-mail: zjin@sccas.cn; bsuo@nwnu.edu.cn

Received February 3, 2021; published March 26, 2021.

Supporting information for this article is available free of charge via the Internet at <http://sioc-journal.cn>.

Project supported by the National Key R&D Program of China (No. 2017YFB0203404) and National Natural Science Foundation of China (No. 21873077).

项目受国家重点研发计划(No. 2017YFB0203404)和国家自然科学基金(No. 21873077)资助。

1 引言

近年来,最初设计用于图形显示的“图形处理单元”(graphics processing unit, GPU)由于其强大的并行计算能力和相对较低的功耗,开始广泛应用于量子化学领域^[1]. 基于高斯基函数的 Hartree-Fock (HF)和密度泛函理论(density functional theory, DFT)是分子体系最常用的量子化学方法,其主要瓶颈包括双电子排斥积分(electron repulsion integral, ERI)、Fock 矩阵及交换相关泛函(exchange-correlation functional, XC)矩阵计算三部分. 针对这些计算瓶颈,本工作设计了支持不同厂商硬件平台的通用 GPU 算法,从而加速 HF 和 DFT 计算.

几乎所有使用 GPU 加速的量子化学计算程序都是基于 NVIDIA 公司的 CUDA^[2] (compute unified device architecture)并行计算框架,但 CUDA 是一种专有编程框架,仅支持 NVIDIA 的 GPU,而 OpenCL^[3] (open computing language)是面向异构系统并行编程的标准框架,可以实现所编写的程序对于不同计算架构的通用性,因而我们选用 OpenCL 作为开发平台.

目前,国内还未有基于 OpenCL 利用 GPU 加速量子化学计算的报道,而国外已开始尝试使用 OpenCL 加速量子化学程序^[4]. Kussmann 和 Ochsenfeld^[4a]于 2017 年实现了基于 OpenCL 的库伦及交换矩阵计算. 随后他们^[4b]又报道了 CPU 和 GPU 混合计算双电子积分的算法. 同年 Kalinowski 等^[4c]使用 OpenCL 实现了 CPU-GPU 混合计算任意角动量的双电子排斥积分算法,但他们的工作都是基于 HF,未利用 OpenCL 计算交换相关势矩阵. 我们的目标是设计基于 OpenCL 的双电子积分、库伦及交换矩阵、交换相关势矩阵的计算,以实现完整的基于 OpenCL 加速的 HF/DFT 的直接自洽场计算.

本文所有工作都是在国产量子化学软件 BDF^[5] (Beijing Density Functional)中完成的. BDF 的 HF/DFT 支持精确二分方法(exact two-component, X2C)的能量和梯度计算,能完全利用分子点群对称性,在相对论效应的处理上独树一帜. 而本文的算法首次使 BDF 的 HF/DFT 计算可利用 GPU 加速,未来可进一步推广到所有的依赖于直接积分计算的方法如 TDDFT (time dependent density functional theory)等.

本文先简单介绍了 HF/DFT 方法及基于 OpenCL 的实现,包括双电子排斥积分与 Fock 矩阵构造,密度泛函计算中的 Becke^[6]和 SSF^[7] (Stratmann-Scuseria-Frisch)两种格点权重计算方案,以及交换相关泛函相关的计算;随后分别在 AMD 和 NVIDIA 公司的不同 GPU 硬件设备上进行了测试,结果显示我们的算法在不同计算架构上相对于 CPU 上的串行算法速度都有了很大的提升,且充分展示了 OpenCL 编程框架对不同硬件架构的通用性. 基于 OpenCL 的 BDF 程序加速了分子体系 HF/DFT 计算,使在可接受时间内完成更大的分子体系的量子化学计算成为可能.

2 理论和 GPU 算法

2.1 HF/DFT 计算方法

基于高斯基函数的 HF/DFT 程序的核心是双电子排斥积分,

$$(\mu\nu|\lambda\sigma) = \iint \chi_\mu(\vec{r}_1)\chi_\nu(\vec{r}_1)\frac{1}{|\vec{r}_1-\vec{r}_2|}\chi_\lambda(\vec{r}_2)\chi_\sigma(\vec{r}_2)d\vec{r}_1d\vec{r}_2 \quad (1)$$

式(1)中, χ 是收缩高斯基函数,又被称为原子轨道,是以原子为中心的原始高斯基函数(primitive gaussian function)的线性组合^[8]. 分子轨道 ϕ 写为原子轨道的线性组合,

$$\phi_p = \sum_{\mu} C_{\mu p} \chi_{\mu} \quad (2)$$

其中 C 为分子轨道系数,通过求解广义本征值方程

$$FC = eSC \quad (3)$$

计算,其中 S 是基函数重叠积分矩阵, F 是 Fock 或 Kohn-Sham 矩阵,我们将一起考虑,写为:

$$F_{\mu\nu} = h_{\mu\nu} + 2 \sum_{\lambda,\sigma=1}^N (\mu\nu|\lambda\sigma)P_{\lambda\sigma} - c_x \sum_{\lambda,\sigma=1}^N (\mu\lambda|\nu\sigma)P_{\lambda\sigma} + V_{\mu\nu}^{xc} \quad (4)$$

式(4)右边的 $h_{\mu\nu}$ 是单电子项,包括电子动能与核吸引两部分. 第二项和第三项分别称为库伦(J)和交换(K)矩阵,通过双电子积分与密度矩阵收缩计算得到,其中密度矩阵 P 为,

$$P_{\mu\nu} = \sum_{i=1}^{nocc} C_{\mu i}^* C_{\nu i} \quad (5)$$

式(4)中 c_x 为精确 HF 交换项的成分, HF 计算中 $c_x=1$. 如果是 DFT 计算, c_x 依赖于选用的交换相关泛函. 最后一项是交换相关势对 Kohn-Sham 矩阵的贡献,称为 V_{xc} 矩阵,对于广义梯度近似(general gradient approximation – GGA)或杂化泛函(hybrid functional),写为,

$$V_{\mu\nu}^{xc} = \int \left[\frac{\partial v_{xc}}{\partial \rho_{\alpha}} \chi_{\mu} \chi_{\nu} + \left(2 \frac{\partial v_{xc}}{\partial \gamma_{\alpha\alpha}} \nabla \rho_{\alpha} + \frac{\partial v_{xc}}{\partial \gamma_{\alpha\beta}} \nabla \rho_{\beta} \right) \cdot \nabla (\chi_{\mu} \chi_{\nu}) \right] d\mathbf{r} \quad (6)$$

其中, ρ 是电子密度, v_{xc} 是交换相关势, γ 为

$$\gamma_{\sigma\sigma} = \nabla \rho_{\sigma}(r) \cdot \nabla \rho_{\sigma}(r) \quad (7)$$

2.2 库伦与交换矩阵的 GPU 计算

为避免存储瓶颈,超过 300 个基函数的 HF/DFT 算法通常基于 Almlöf 提出的积分直接的自洽场计算方案(direct self-consistent-field calculations, Direct SCF)^[9],即每次 SCF 迭代均重新计算双电子积分并直接构造 Fock

矩阵. 我们在 GPU 上使用与 Ufimtsev 等^[1c,1d]提出的“一个线程计算一个原始积分”(1 thread 1 primitive integral, 1T1PI)类似的算法计算双电子积分, 并通过原始高斯基函数积分直接构造库伦和交换矩阵, 相应算法的伪代码在支持信息第 2 节中的图 S3 和 S4 给出.

库伦矩阵在 GPU 上的计算可以充分利用双电子积分的交换对称性. 对于双电子积分($\mu\nu|\lambda\sigma$), 交换指标 $\mu\nu \leftrightarrow \nu\mu$ 或 $\sigma\lambda \leftrightarrow \lambda\sigma$, 积分值不变. 因而, 独立的基函数对 (basis function pair) 为 $M=N(N+1)/2$ 个. 又由于交换左矢指标 $\mu\nu$ 与右矢指标 $\lambda\sigma$ 不改变积分值, 实际独立的双电子积分为 $M(M+1)/2$ 个. 只需考虑对称独立的双电子积分, 计算库伦矩阵的上三角部分. 我们目前的实现限于 s 和 p 类型的高斯基函数, 这里有三种可能的组合: ss、ps 和 pp, 需要编写并优化 [ss|ss]、[ps|ss] 等 9 个 OpenCL 内核程序, 将相同类型的积分组织到一起, 分配到 GPU 的同一个计算单元 (computing unit - CU) 计算.

交换矩阵的构造不同于库伦矩阵, 我们没有利用 $\mu\lambda \leftrightarrow \lambda\mu$ 和 $\nu\sigma \leftrightarrow \sigma\nu$ 的交换对称性, 这是由于 $[\nu\sigma]$ 和 $[\sigma\nu]$ 积分对应不同的交换矩阵元, 利用交换对称性会导致昂贵的 GPU 工作组间的通信. 对于 s 和 p 型基函数, 所有的 bra($\mu\lambda$) 和 ket($\nu\sigma$) 被分成四种类型, 即 ss、sp、ps 和 pp, 产生 16 种不同的双电子积分, 考虑到 bra $\leftrightarrow$ ket 的对称性可以利用, 实际计算 K 矩阵的上三角部分, 需要 [ss|ss]、[sp|ss] 等 10 个 OpenCL 内核程序.

基于高斯基函数的双电子排斥积分算法, 如 McMurchie-Davidson (MD)^[10]、Obara-Saika (OS)^[11]、Head-Gordon-Pople (HGP)^[12] 和 Dupuis-Rys-King (DRK)^[13] 等, 通常使用积分的递推关系计算, 一般不适用于 GPU. 因而, 我们基于 McMurchie-Davidson 方法, 将递推公式展开, 并通过计算机自动生成高斯积分的 GPU 代码. 在 GPU 上完成双电子积分计算并与密度矩阵收缩, 得到库伦与交换矩阵元后, 再将其传输回 CPU, 在 CPU 上累加至 Fock 矩阵. 为得到与 CPU 完全一致的精度, 所有的计算均采用双精度浮点数.

2.3 V_{xc} 矩阵的 GPU 算法

由于交换相关泛函形式复杂, V_{xc} 矩阵很难解析计算, 通常使用数值积分方案. 我们先在 CPU 上生成不同原子类型的积分格点, 然后在 GPU 上利用 Becke 或 SSF 分割方法计算所有实空间的积分格点权重, 将积分格点按照每 128 个作为一块对齐, 并对积分格点块所能承载的非零基函数进行预筛选. 积分格点块、格点块所承载的基函数索引信息等数据分别在 CPU 和 GPU 内存保存, 以减少任务分配和调度过程的 CPU/GPU 间的通讯. 在 V_{xc} 计算过程中, 先在 CPU 主机端根据 CPU 核心数与可用 GPU 卡数开启多个 OpenMP^[14] 线程, 并对积分格点块进行动态分配, CPU 与 GPU 同时执行所分配的 V_{xc} 矩阵的计算任务. V_{xc} 的 GPU 计算程序分为基函数值与梯度、电子密度与密度梯度、 V_{xc} 泛函值与泛函导数、

交换相关能及交换相关矩阵等 OpenCL 内核程序. 交换相关项所有代码均使用双精度浮点数执行运算, 支持多 GPU 并行和 CPU/GPU 混合并行模式. 交换相关势计算伪代码详见支持信息第 2 节的图 S5.

3 测试算例

3.1 积分格点权重计算性能

我们首先比较了 Becke 及 SSF 分割两种积分格点权重计算方案在 CPU 与 GPU 上的性能. 表 1 给出了一系列 DNA 片段在 Intel Core i9-9900K CPU 上单核串行程序和 AMD Radeon VII GPU 上两种算法的计算时间 (本文统计的时间均为程序运行的时钟时间), OpenCL 内核使用 ROCm^[15] 平台实现. 表 1 括号中的数值为 GPU 相对于 CPU 计算加速比. 积分格点使用 BDF 程序默认的 medium 类型格点 (H-Ne(75/302), Na-Ar(88/434), K-Kr(112/590), Rb-Og(123/590)).

表 1 Becke 及 SSF 分割算法在 CPU 与 GPU 上计算性能比较

Table 1 Comparing the performance of the Becke and SSF partition algorithms on CPU and GPU

Molecule	$N_{\text{atoms}}/N_{\text{BF}}$	Wall time (Second) and speedup		
		1CPU ^a	1GPU ^{b,c}	2GPU ^{b,c}
Becke partition				
(DNA) ₁	62/625	16	0.5(32.0)	0.3(53.3)
(DNA) ₂	128/1332	138	4(34.5)	2(69.0)
(DNA) ₄	260/2746	1166	33(35.3)	17(68.6)
(DNA) ₈	524/5574	9582	269(35.6)	136(70.5)
SSF partition				
(DNA) ₁	62/625	6	0.6(10.0)	0.4(15.0)
(DNA) ₂	128/1332	55	4(13.8)	2(27.5)
(DNA) ₄	260/2746	517	37(14.0)	18(28.7)
(DNA) ₈	524/5574	4401	287(15.3)	145(30.4)

^a Reference calculations were performed on a single core of Intel Core i9-9900K processor. ^b Calculations were executed on AMD Radeon VII GPU.

^c Numbers in parentheses represent the speedup.

对于 Becke 分割算法, 在一块和两块 GPU 卡上最快分别达到 35 和 70 倍的加速, 而 SSF 分割在一块和两块 GPU 卡上最快分别达到 15 和 30 倍的加速. 随着测试分子体系的增大, 两种积分格点权重算法在 GPU 上都获得了更高的加速比. 虽然 CPU 上的 SSF 算法计算比 Becke 方法快很多, 但其在 GPU 上的加速比不到 Becke 方法的一半, 这是由于 SSF 算法存在众多分支判断, 造成 GPU 上大量的系统开销, 从而严重影响其在 GPU 上的计算性能, 这与 GPU 的硬件特点是相吻合的.

3.2 Fock 矩阵计算性能测试

积分直接的 SCF 方法, 双电子积分与 J 和 K 矩阵计算是绑定在一起的, 单独测试积分的计算性能没有实际意义. 因而, 我们通过 (DNA)₂ 分子测试 J 和 K 矩阵构造在不同架构计算设备上的性能. 由于我们在 GPU 上以实现 s 和 p 类型的高斯基函数积分为目标, 这里的测试使用对 DNA 体系仅含 s 和 p 型函数的 6-31G 基组^[16]. 此

外,为了更好地反映高斯积分及 Fock 构造方法在 GPU 上的计算性能,我们关闭了直接 SCF 方法的积分预筛选功能. 开启积分预筛选会同时降低 CPU 和 GPU 上的 Fock 矩阵计算时间. 图 1 给出了 J 和 K 矩阵在 AMD Radeon VII、NVIDIA Geforce RTX 2080Ti、Tesla V100 和 A100 等处理器上相对于 Intel Core i9-9900K CPU 上单核串行程序构造时间的加速比,其中 GPU 上的计算时间统计包含 CPU 和 GPU 间的数据传输等系统开销时间. 为了评估 Fock 矩阵计算在 CPU 上的并行加速,8CPU 核计算 J、K 矩阵时间及相对于单 CPU 核的加速比在支持信息的第 7 节表 S4 中给出.

图 1 库伦和交换矩阵计算在不同架构 GPU 上的加速比

Figure 1 Comparing the speedup of the J- and K-matrix calculation on different GPU architectures

在所有测试的硬件中, NVIDIA 最新的专业级计算卡 A100 表现出所有计算设备中最好的性能. J 矩阵的构造在 A100 上达到了 148 倍的加速, NVIDIA 的上一代专业计算卡 V100 对 J 矩阵的计算也表现出 71 倍的加速. 支持信息中第 3 节表 S1 给出了本文所用的 GPU 硬件的理论双精度性能. A100 理论双精度性能为 9.7TFLOPS, 约是 V100 卡(7TFLOPS)的 1.3 倍. J 矩阵构造算法在 A100 卡上展现出相对于 V100 卡约 2 倍实际加速比, 优于理论加速比. K 矩阵、交换相关势矩阵的计算都有类似的行为, 这是由于 GPU 架构的技术进步造成的. A100 的 Ampere 架构相对于 V100 的 Volta 架构重新设计了流处理器(streaming multiprocessor, SM)和访存模式, 具有更大的 L2 缓存、更高的内存带宽, 这些进步虽不能直接体现在理论浮点计算能力, 但对软件的实际执行效率有极大的影响^[17].

图 1 显示, K 矩阵计算在 A100 卡上相对于单 CPU 核的计算加速比为 67.8, 约为 J 矩阵的一半, 在其他计算平台也展现了类似的性能. 这是由于 K 矩阵在 GPU 上的计算牺牲了双电子积分左矢指标($\mu\lambda$)和右矢指标($\nu\sigma$)的交换对称性导致的. 理论上, 在 GPU 上要计算的双电子积分要比 CPU 上多算一倍, 这与其他基于 GPU 的 HF 计算程序表现也是相吻合的^[14].

3.3 交换相关项性能

图 2 展示了交换相关矩阵在不同架构 GPU 上的性能,

测试分子及硬件平台与 Fock 矩阵计算时相同, 但加入了多 GPU 的结果. 计算使用 PBE^[18]泛函和 cc-pVDZ^[19]基组, 积分格点为 BDF 中的 medium 型格点. 如图 2 所示, 与 Fock 矩阵的表现相同, 交换相关矩阵计算在 NVIDIA A100 上表现出最好的性能, 单卡最高加速比 17.7, 在两块 A100 卡上达到 34.6 倍的加速, 多卡对单卡接近理论上的理想加速比. 与 Fock 矩阵计算不同, 我们没有对交换相关泛函的计算热点如密度矩阵, 交换相关泛函矩阵等展开计算公式, 而是利用了与 CPU 上类似的代码, 代码中的循环、分支判断较多, 因而 V_{xc} 计算在 GPU 上的加速比最低.

图 2 交换相关泛函矩阵计算在不同架构 GPU 上的加速比

Figure 2 Comparing the speedup of the exchange correlation functional matrix calculation on different GPU architectures

3.4 SCF 性能测试

最后, 我们利用 cocaine、(DNA)₁、taxol、(DNA)₂、valinomycin 和 (DNA)₄ 等六个分子进行了在 CPU 和 GPU 上的 SCF 计算对照测试. CPU 统计的是 Intel Core i9-9900K CPU 上串程序的计算时间, GPU 为 AMD Radeon VII 和 NVIDIA A100 单块 GPU 卡上的计算时间. 计算使用 6-31G 基组与 B3LYP^[20]泛函. 在 CPU 和 GPU 上计算 J 和 K 矩阵时都关闭了双电子积分的预筛选, 交换相关矩阵计算使用 BDF 的 medium 型格点. 表中分别给出了一次 SCF 迭代、J 与 K 矩阵及交换相关矩阵的计算时间, 并给出了总的 SCF 计算时间, 其中括号中的数字为 GPU 相对于 CPU 的计算加速比.

从表 2 可以看出, SCF 计算的 GPU 加速效果随着体系的增大有了显著提高, 其中 (DNA)₄ 的算例在 Radeon VII 和 A100 上分别达到了最快 19 倍和 86 倍的加速, 几乎是 cocaine 分子加速比的近 2 倍, 说明只有包含足够多的计算任务才能充分发挥 GPU 的计算能力. 从一次 SCF 迭代计算时间来看, 对于 A100 卡, J 矩阵的加速比普遍在 100 倍以上, K 矩阵加速比约是 J 矩阵的一半, V_{xc} 矩阵计算加速比最低. 相比于 J 矩阵的计算, K 矩阵与交换相关矩阵计算是目前程序的瓶颈, 是未来进一步优化的方向. 需要特别指出的是, 为了保证计算精度, 我们在 GPU 上的所有代码使用双精度浮点数运算, 并且关闭了 OpenCL 编译器对数学运算的优化, 因而 CPU 和

表2 SCF 计算在 CPU 与 GPU 上的性能对比^aTable 2 Comparing performance of the SCF calculation on CPU and GPU^a

Molecule	$N_{\text{atoms}}/N_{\text{BF}}$	Intel Core i9-9900K ^b				AMD Radeon VII				NVIDIA A100			
		One SCF iteration			SCF time	One SCF iteration			SCF time(speedup)	One SCF iteration			SCF time(speedup)
		J	K	V_{xc}		J	K	V_{xc}		J	K	V_{xc}	
cocaine	43/240	47	48	12	1569	3(15.7) ^c	5(9.6)	2(6.0)	136(11.5)	0.4(117.5)	1(48.0)	0.9(13.3)	34(46.1)
(DNA) ₁	62/369	82	84	19	2347	4(20.5)	7(12.0)	5(3.8)	191(12.3)	0.6(136.7)	2(42.0)	1(19.0)	45(52.2)
taxol	110/647	718	731	69	22589	33(21.8)	57(12.8)	14(4.9)	1524(14.8)	5(143.6)	12(60.9)	4(17.3)	323(69.9)
(DNA) ₂	128/796	1333	1355	113	41678	57(23.4)	92(14.7)	16(7.1)	2439(17.1)	9(148.1)	20(67.8)	7(16.1)	542(76.9)
valinomycin	168/882	1993	2030	146	61849	85(23.4)	153(13.3)	21(7.0)	3855(16.0)	14(142.4)	30(67.7)	9(16.2)	803(77.0)
(DNA) ₄	260/1650	9347	9492	472	287831	367(25.5)	572(16.6)	68(6.9)	14815(19.4)	67(139.5)	122(77.8)	28(16.9)	3329(86.5)

^a Wall times (in seconds). ^b Reference calculations are performed on a single core of Intel Core i9-9900K processor. ^c Numbers in parentheses are speedup.

GPU 上 SCF 计算的能量在要求的收敛精度 1×10^{-8} Hartree 下完全一致(详见支持信息第 6 节表 S3)。

3.5 GPU 相对 CPU 并行计算的性能分析

BDF 的 HF/DFT 支持 MPI^[21] 及 OpenMP^[14] 混合并行。支持信息的第 7 节表 S4 给出了表 2 中的所有算例 SCF 计算在 Intel Core i9-9900K 8 核 CPU 上的计算时间及加速比。可以看出, 相对于单 CPU 核, 使用 8 个 CPU 核心 BDF 的 SCF 计算获得了约 6.5 倍的加速比。对于本文计算测试的最大体系 (DNA)₄ 分子, A100 相对于 Intel i9-9900K 的 SCF 计算加速了约 13 倍。A100 卡的功耗约为 250 W, 约为 Intel i9-9900K 功耗(95 W) 的 2.6 倍, 从功耗角度讲, 利用 GPU 计算效率更高。

4 结论

基于 OpenCL 编程框架, 我们在 BDF 中开发了可用于不同厂商 GPU 设备的 HF/DFT 计算程序, 优化了双电子排斥积分、Fock 矩阵及交换相关矩阵等计算热点, 并通过一系列算例测试了程序的性能。结果显示, 我们的程序适用于多种架构的计算设备, 并展现了良好的性能。在 GPU 上的库伦矩阵计算相比 CPU 上的串行算法最高可达 148 倍的加速, 完整的 DFT 能量计算实现了最高 86 倍的加速。与本文的工作结合, 我们将发展 CPU/GPU 混合计算的 HF/DFT 算法, 将低角动量高斯函数如 s、p 相关的积分在 GPU 上处理, 其他的双电子积分在 CPU 上处理, 并发展 HF/DFT 梯度算法, 以期能充分利用商用的 CPU/GPU 异构架构计算工作站的计算能力, 实现 BDF 中 HF/DFT 能量及梯度计算、几何结构优化的关键量子化学计算模块加速。

致谢

感谢中国科学院计算机网络信息中心马英晋博士与靳润锋在程序代码和测试设备方面的帮助。

References

- [1] (a) Yasuda, K. *J. Comput. Chem.* **2008**, *29*, 334. (b) Yasuda, K. *J. Chem. Theory Comput.* **2008**, *4*, 1230. (c) Ufimtsev, I. S.; Martínez,

- T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222. (d) Ufimtsev, I. S.; Martínez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 1004. (e) Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. *J. Chem. Theory Comput.* **2011**, *7*, 949. (f) Miao, Y.; Merz, K. M. *J. Chem. Theory Comput.* **2013**, *9*, 965. (g) Manathunga, M.; Miao, Y.; Mu, D.; Götz, A. W.; Merz, K. M. *J. Chem. Theory Comput.* **2020**, *16*, 4315. (h) Gordon, M. S.; Barca, G.; Leang, S. S.; Poole, D.; Rendell, A. P.; Galvez Vallejo, J. L.; Westheimer, B. J. *Phys. Chem. A* **2020**, *124*, 4557. (i) Walker, R. C.; Gtz, A. W. *Electronic Structure Calculations on Graphics Processing Units: From Quantum Chemistry to Condensed Matter Physics*, Wiley, New York, **2016**.
- [2] NVIDIA. CUDA. <https://docs.nvidia.com/cuda/index.html> (accessed Jan. 20, 2021).
- [3] OpenCL. <https://www.khronos.org/opencl/> (accessed Jan. 20, 2021).
- [4] (a) Kussmann, J.; Ochsenfeld, C. *J. Chem. Theory Comput.* **2017**, *13*, 2712. (b) Kussmann, J.; Ochsenfeld, C. *J. Chem. Theory Comput.* **2017**, *13*, 3153. (c) Kalinowski, J.; Wennmohs, F.; Neese, F. *J. Chem. Theory Comput.* **2017**, *13*, 3160.
- [5] (a) Liu, W.-J.; Hong, G.-Y.; Dai, D. D.; Li, L.-M.; Dolg, M. *Theor. Chem. Acc.* **1997**, *96*, 75. (b) Liu, W.-J.; Wang, F.; Li, L.-M. *J. Theor. Comput. Chem.* **2003**, *02*, 257. (c) Liu, W.-J.; Wang, F.; Li, L.-M. *Recent Advances in Relativistic Molecular Theory*, Vol. 5, World Scientific, **2004**, p. 257. (d) Zhang, Y.; Suo, B.-B.; Wang, Z.-K.; Zhang, N.; Li, Z.-D.; Lei, Y.-B.; Zou, W.-L.; Gao, J.; Peng, D.-L.; Pu, Z.-C.; Xiao, Y.-L.; Sun, Q.-M.; Wang, F.; Ma, Y.-T.; Wang, X.-P.; Guo, Y.; Liu, W.-J. *J. Chem. Phys.* **2020**, *152*, 064113.
- [6] Becke, A. D. *J. Chem. Phys.* **1988**, *88*, 2547.
- [7] Stratmann, R. E.; Scuseria, G. E.; Frisch, M. J. *Chem. Phys. Lett.* **1996**, *257*, 213.
- [8] Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular Electronic-Structure Theory*, Wiley, New York, **2000**, p. 336.
- [9] Almlöf, J.; Faegri Jr, K.; Korsell, K. *J. Comput. Chem.* **1982**, *3*, 385.
- [10] McMurchie, L. E.; Davidson, E. R. *J. Comput. Phys.* **1978**, *26*, 218.
- [11] Obara, S.; Saika, A. *J. Chem. Phys.* **1988**, *89*, 1540.
- [12] Head-Gordon, M.; Pople, J. A. *J. Chem. Phys.* **1988**, *89*, 5777.
- [13] Dupuis, M.; Rys, J.; King, H. F. *J. Chem. Phys.* **1976**, *65*, 111.
- [14] OpenMP. <https://www.openmp.org/> (accessed Jan. 20, 2021).
- [15] AMD. ROCm. <https://rocm.docs.amd.com/en/latest/> (accessed Jan. 20, 2021).
- [16] (a) Hehre, W. J.; Stewart, R. F.; Pople, J. A. *J. Chem. Phys.* **1969**, *51*, 2657. (b) Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56*, 2257. (c) Dill, J. D.; Pople, J. A. *J. Chem. Phys.* **1975**, *62*, 2921. (d) Binkley, J. S.; Pople, J. A.; Hehre, W. J. *J. Am. Chem. Soc.* **1980**, *102*, 939.
- [17] (a) NVIDIA. NVIDIA Tesla V100 GPU Architecture. (b) NVIDIA. NVIDIA A100 Tensor Core GPU Architecture.
- [18] Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.* **1996**, *77*, 3865.
- [19] (a) Dunning, T. H. *J. Chem. Phys.* **1989**, *90*, 1007. (b) Woon, D. E.; Dunning, T. H. *J. Chem. Phys.* **1993**, *98*, 1358.
- [20] (a) Becke, A. D. *J. Chem. Phys.* **1993**, *98*, 5648. (b) Kameo, H.; Ishii, S.; Nakazawa, H. *Dalton Trans.* **2012**, *41*, 8290.
- [21] MPI. <https://www.mpi-forum.org/> (accessed Jan. 20, 2021).

(Cheng, B.)