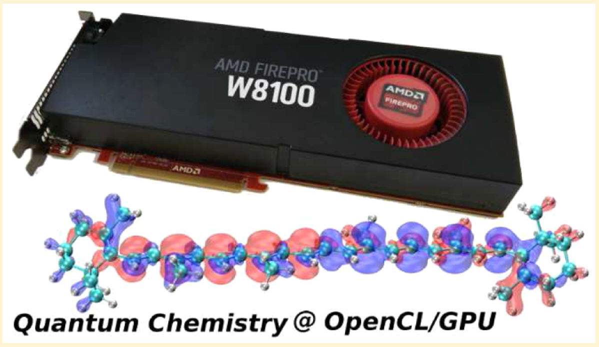
# Employing OpenCL to Accelerate Ab Initio Calculations on Graphics Processing Units

Jörg Kussmann and Christian Ochsenfeld*

Chair of Theoretical Chemistry, Department of Chemistry, University of Munich (LMU), Butenandtstraße 7, D-81377 München, Germany

Center for Integrated Protein Science (CIPSM) at the Department of Chemistry, University of Munich (LMU), Butenandtstraße 5-13, D-81377 München, Germany

**ABSTRACT:** We present an extension of our graphics processing units (GPU)-accelerated quantum chemistry package to employ OpenCL compute kernels, which can be executed on a wide range of computing devices like CPUs, Intel Xeon Phi, and AMD GPUs. Here, we focus on the use of AMD GPUs and discuss differences as compared to CUDA-based calculations on NVIDIA GPUs. First illustrative timings are presented for hybrid density functional theory calculations using serial as well as parallel compute environments. The results show that AMD GPUs are as fast or faster than comparable NVIDIA GPUs and provide a viable alternative for quantum chemical applications.



## I. INTRODUCTION

In recent years, it has been shown that quantum chemical calculations can strongly benefit from employing graphics processing units (GPU).[1−15] To the best of our knowledge, all developments so far are based on CUDA,[17] a proprietary compute platform that is restricted to run only on NVIDIA GPUs. Because NVIDIA provided the first stable, easy-to-use, and well-performing development platform, CUDA and therefore NVIDIA GPUs represent a virtual monopoly in GPU-based high performance computing.

In this work, we present the extension of our quantum chemistry code FermiONs++[10,11] to employ OpenCL compute-kernels, which in general allows the use of a wide variety of compute architectures like CPUs, Intel Xeon Phi, and GPUs. Here, we focus on employing AMD GPUs and also discuss differences to CUDA-based ab initio calculations. A particular focus is set on integral evaluations, which show the largest speed-up as compared to CPU-based calculations and also open a way to accelerate post Hartree−Fock (HF) and complete active space self-consistent field (CASSCF) methods.[12−15] At the example of exchange-type potential evaluation, differences in the optimization strategies for AMD and NVIDIA GPUs, respectively, are discussed. Finally, we present first illustrative calculations employing AMD FirePro 3D W8100 GPUs using a single GPU server as well as a parallel setup of six servers with up to 24 GPUs. The strong-scaling parallelization is demonstrated for the example of real-time time-dependent hybrid density functional theory calculations.

## II. OPENCL-BASED INTEGRAL EVALUATION

In this section, the major aspects of developing integral-kernels using OpenCL are discussed. Because GPUs from both NVIDIA and AMD share the basic technical specifications (at least from a programmers point of view), the general setup, e.g., regarding the data-arrangement,[3] and the loop-structure[5,10] can be directly applied to AMD GPUs. Therefore, the employed OpenCL kernels will have the same basic structure as the CUDA kernels used on NVIDIA GPUs, i.e, we employ for both devices the algorithms for determining Coulomb- and exchange-type potential matrices as described for CUDA in refs 5 and 10. In this work, we will rather focus on differences between the programming interfaces, i.e., OpenCL and CUDA, as well as technical differences between AMD and NVIDIA GPUs that will affect integral evaluation strategies. Note that all GPU-based integral routines in FermiONs++[10,11] are using double-precision accuracy only, so that this aspect should be considered as a basic underlay for the following discussion.

**A. OpenCL vs CUDA.** Both OpenCL and CUDA are C-like languages, thus the basic code for the integral evaluation algorithm is similar.[5,10] FermiONs++[10,11] uses code-generators to build the different integral evaluation kernels, so only minor additions to these programs are necessary to target both OpenCL and CUDA architectures.

Regarding the management of device (GPU) memory, OpenCL offers similar interfaces to those of CUDA, although arrays are handled via an abstract built-in data-type (cl_mem) on the host-side (CPU). Similar to CUDA, the execution of the

Received: May 17, 2017
Published: May 31, 2017

compute kernel is managed by partitioning the whole workload into a grid of equal-sized blocks, where the grid and blocks can be up to 3-dimensional. In contrast to CUDA, the OpenCL design concept is rather similar to a script language insofar as the compute kernels are supposed to be compiled on-the-fly with respect to the target architecture. Considering the complexity of integral kernels, in particular for large *l*-quantum numbers of the underlying basis functions, this would seriously decrease the performance of using OpenCL. Therefore, our strategy is to use a library of precompiled kernels for specific architectures, e.g., GPU-types. The kernel binaries are stored on disk and loaded on demand with virtually no overhead as compared to CUDA applications.

**B. AMD vs NVIDIA.** Here, we briefly discuss the differences between AMD and NVIDIA GPUs that mostly affect the two-electron integral evaluation. As examples, we use AMD FirePro 3D W8100 and NVIDIA Tesla K40 GPUs, two professional devices of comparable GPU generations.

In Table I, the technical specifications of both AMD and NVIDIA GPUs are listed. Additionally, we also give the specifications of the NVIDIA GTX Titan Black GPU, which we use for the calculations presented in section III. Note that this GPU features similar specs as compared to the NVIDIA K40 but that it is a nonprofessional consumer GPU, i.e., it does not feature error-correcting code memory (ECC), therefore resulting in it having higher computing performance and being considerably cheaper. The NVIDIA K40 provides significantly more global memory, whereas the AMD FirePro W8100 GPU provides a higher double-precision performance of theoretically 2.1 Tera FLOPS (TF) as compared to 1.7 TF. Since the integral evaluation algorithms can be batched with respect to the memory limits of the GPU and host-device data transfer is not a limiting factor, the difference in global memory does not affect the overall performance.

**Table I. Technical Specifications of AMD FirePro 3D W8100, NVIDIA Tesla K40, and NVIDIA GTX Titan Black GPUs**[a]

| | AMD FirePro 3D W8100 | NVIDIA Tesla K40 | NVIDIA Titan Black |
|---|---|---|---|
| stream processors | 2560 | 2880 | 2880 |
| compute units/SMX | 40 CU, 64 threads/CU | 15 SMX, 192 threads/SMX | 15 SMX, 192 threads/SMX |
| peak single prec. | 5.1 TF | 4.3 TF | 5.1 TF |
| peak double prec. | 2.1 TF | 1.4 TF | 1.7 TF |
| max. mem. bandwidth | 320 GB/s | 288 GB/s | 336 GB/s |
| memory size | 8 GB | 12 GB | 6 GB |
| local memory | 64 kB | 64 kB | 64 kB |
| max. shared memory | 32 kB | 48 kB | 48 kB |

[a]For further explanations, see the text.

The major limiting factors are the usually scarce memory resources on a single streaming multiprocessor (denoted as SMX for NVIDIA GPUs and as compute unit [CU] for AMD GPUs). Both GPUs feature 64 kB of local memory; however, on NVIDIA GPUs, a larger amount (48 kB) can be reserved for shared memory. This is particularly important for higher *l*-quantum number (*l*-qn) basis functions as will be briefly illustrated: As has been determined from tests (see also ref 4), the most efficient setup for integral evaluation is to use $8 \times 8$ thread blocks. In the case of the exchange matrix, for example, each thread evaluates a primitive integral $(\mu\nu|\lambda\sigma)$ and directly multiplies with the density matrix elements $P_{\nu\sigma}$, while the result $K_{\mu\lambda}$ is stored in shared memory. If $\chi_\mu$ and $\chi_\lambda$ represent Cartesian *d*-shells, we have to store 36 double-precision elements per thread, i.e., we need $36 \times 8 \times 8 \times 8 = 18432$ bytes of shared memory, which can be handled by both an SMX/CU on NVIDIA and AMD GPUs. Note that for highler *l*-qn shells, we have to split the evaluation into batches of basis functions because, e.g., if $\chi_\mu$ and $\chi_\lambda$ represent Cartesian *f*-shells, we need 50 kB of shared memory.

Considering this shared memory effort, the hardware configuration of the Tesla K40 shows a less favorable threads/memory distribution. For a single SMX, the 64 kB of local memory are distributed over 192 single instruction, multiple data (SIMD) lanes,[18] whereas the W8100 has only 64 SIMD lanes per compute unit. Note that this local memory bound is a central parameter used in the code-generation process, which can automatically split routines for high *l*-qn shells into processable batches.

**C. AMD vs NVIDIA: Differences in Optimization Strategies.** As mentioned before, the same code generators are used for both CUDA and OpenCL kernels. Their main objectives are to be easy to maintain and to adapt to different architectures while producing efficient compute kernels. Therefore, we ignore possible architecture-specific features for optimization and allow only basic parameters to be modified to either adhere to local memory constraints of the target architecture or general design concepts like the choice of algorithm (McMurchie−Davidson or Rys-quadrature) or, e.g., loop-unrolling. As an example, we compare the exchange-potential evaluation for NVIDIA GTX Titan and AMD FirePro 3D W8100. As described in ref 19, we suggest a mixed use of the McMurchie−Davidson algorithm and Rys-quadrature for basis functions with smaller or larger *l*-quantum numbers, respectively. Note that conventional CPU-based algorithms like the PRISM algorithm by Gill and Pople[16] also use different integral evaluation schemes based on the floating-point operations (FLOP) count. For efficient GPU-based algorithms, however, the FLOP count is not a useful measure because most integral routines are memory bound. Therefore, the choice between McMurchie−Davidson and Rys-quadrature is determined once from test calculations. The evaluation algorithm is statically chosen for the corresponding GPU kernel based on the kernel timings for a DNA fragment containing 16 adenosine-thymine base pairs. As these calculations have shown, the use of McMurchie−Davidson is less often advantageous for the AMD GPUs. Furthermore, Rys-quadrature-based algorithms can benefit from loop-unrolling with respect to the quadrature roots when using CUDA devices, which is also discussed in ref 9. In contrast, first tests have shown that OpenCL tends to significantly increase the number of local scratch variables when loop-unrolling is employed. To illustrate these points, we show the timings for two exchange kernels of a DNA fragment with eight adenosine-thymine base pairs using the def2-SVP basis.[20] Here, NVIDIA/CUDA performed best using the McMurchie−Davidson algorithm (McD), whereas the performance for AMD/OpenCL significantly increases when switching to Rys-quadrature with loop-unrolling (Rys/unroll). The performance is further increased by avoiding loop-unrolling (Rys/looped) as can bee seen from the timings for AMD FirePro 3D W8100:

**Table II. Wall Times (in Seconds) for Evaluation of the Full Coulomb- and Exchange-Matrices for a Series of Adenosine-Thymine DNA Fragments $DNA_x$ (Number of Base-Pairs: $x$ = 1, 2, 4, 8, 16) using PBE0/SVP [99/590][a]**

| | 4× AMD W8100 | | 4× NVIDIA Titan | | | 24× AMD W8100 | | |
|---|---|---|---|---|---|---|---|---|
| | J | K | J | K | XC | J | K | XC |
| $DNA_1$ | 0.2 s | 1.0 s | 0.2 s | 1.5 s | 4.1 s | 0.1 s | 0.3 s | 0.7 s |
| $DNA_2$ | 0.9 s | 6.8 s | 0.9 s | 9.8 s | 6.1 s | 0.3 s | 1.5 s | 1.1 s |
| $DNA_4$ | 4.5 s | 30.0 s | 4.4 s | 40.5 s | 16.1 s | 1.0 s | 6.6 s | 2.7 s |
| $DNA_8$ | 19.4 s | 85.5 s | 18.9 s | 113.5 s | 36.8 s | 4.1 s | 19.5 s | 6.4 s |
| $DNA_{16}$ | 80.1 s | 205.1 s | 74.7 s | 268.3 s | 78.2 s | 16.0 s | 48.6 s | 18.0 s |

[a]An integral threshold of $\vartheta_{\mathrm{int}} = 10^{-10}$ and a pre-selection threshold of $\vartheta_{\mathrm{pre}} = 10^{-3}$ are used. Calculations were executed on a single server with four AMD FirePro 3D W8100 and four NVIDIA GTX Titan Black, respectively, as well as six nodes with a total of 24 W8100 GPUs using MPI. The exchange-correlation potential is evaluated on CPU only (12 cores per node).

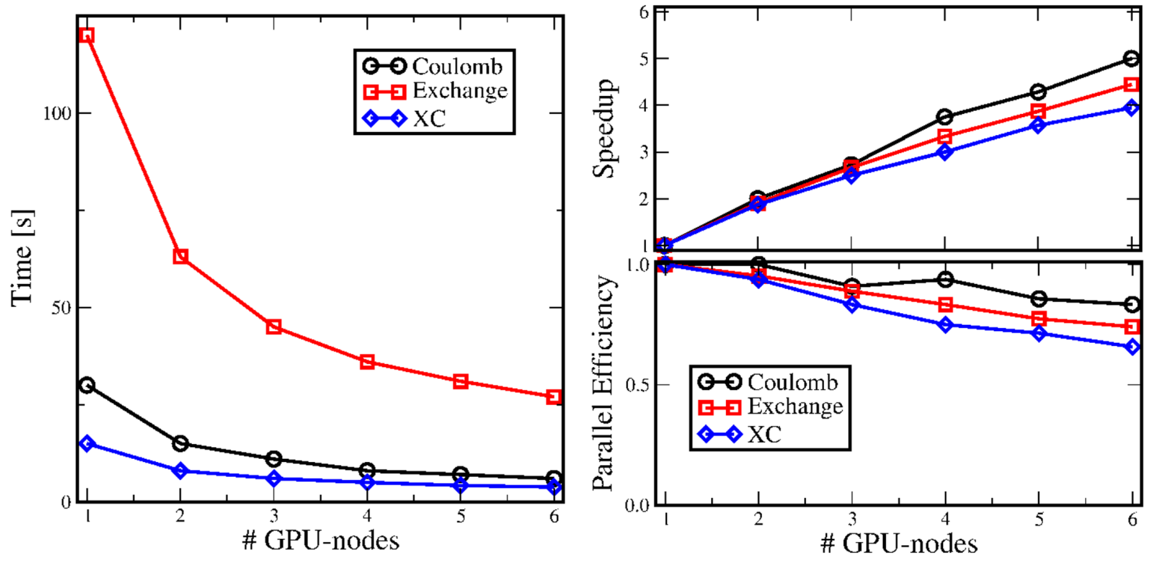


**Figure 1.** Coulomb-, exchange-, and exchange-correlation potential calculations for a DNA fragment of eight adenine-thymine base-pairs using PBE0/def2-SVP using up to six GPU nodes each with four W8100 GPUs. (left) Wall times (in seconds). (right/top) Speed-up; (right/bottom) parallel efficiency.

| | McD | | Rys/unroll | | Rys/looped |
|---|---|---|---|---|---|
| $(pp\|pp)$ | 96.9 s | $\longrightarrow$ | 36.0 s | $\longrightarrow$ | 17.2 s |
| $(sd\|pp)$ | 84.6 s | $\longrightarrow$ | 8.3 s | $\longrightarrow$ | 7.9 s |

## III. ILLUSTRATIVE CALCULATIONS

The OpenCL-based integral evaluation kernels have been implemented in our ab initio code FermiONs++[10,11] using the AMD software development kit "AMD APP SDK";[21] the compute kernels were precompiled for the AMD FirePro 3D W8100 architecture and stored on disk. CUDA-based calculations were also performed with FermiONs++ using CUDA 7.5.[17] Integral evaluations were performed as described in ref 10 using the PreLinK scheme to achieve an asymptotically linear-scaling behavior for the exchange matrix calculations. All calculations employ tight convergence and screening thresholds of $\vartheta_{\mathrm{conv}} = 10^{-7}$, $\vartheta_{\mathrm{int}} = 10^{-10}$, and for preselection, $\vartheta_{\mathrm{pre}} = 10^{-3}$, which is sufficient for sub-$\mu$H accuracy. The DFT grid is constructed from 99 radial points (LOG3[22]) and 590 angular points (Lebedev–Laikov[23]). Note that the evaluation of the exchange-correlation potential is entirely executed on CPUs, and only the generation of the molecular grid using the scheme proposed by Stratmann et al.[24] is performed on GPUs.

As a first example, we present wall-times for the full construction of the Coulomb-, exchange-, and exchange-correlation (XC) matrices for a series of adenine-thymine DNA-fragments $DNA_x$ ($x$: number of base pairs)[25,26] with up to 16 base pairs and 1052 atoms using PBE0/def2-SVP.[20,27,28] The results on the left in Table II were obtained by calculations on a single GPU server featuring four GPUs in total using either AMD FirePro 3D W8100 or NVIDIA GTX Titan Black, respectively. The latter card features less memory than the previously discussed Tesla K40 but shows a higher computing performance also due to the lack of error-correcting code memory (ECC). While the performance of Coulomb evaluations is slightly better using Titan Black GPUs, the exchange is significantly faster on W8100 GPUs. One reason might be the better balance between threads and local memory in AMD GPUs because memory bounds have a far larger impact on exchange potential calculations as compared to the Coulomb potential. Additionally, we also used six fast interconnected GPU servers featuring up to 24 W8100 GPUs for the same calculations. As discussed for CUDA-based calculations in ref 11, we also obtain a good scaling performance with an increasing number of GPU nodes even for rather small systems like $DNA_2$, which again indicates strong scaling behavior.

To analyze the scaling with the number of GPUs, we measured the wall times of calculations for the medium-sized $DNA_8$ fragment containing 524 atoms using up to six AMD

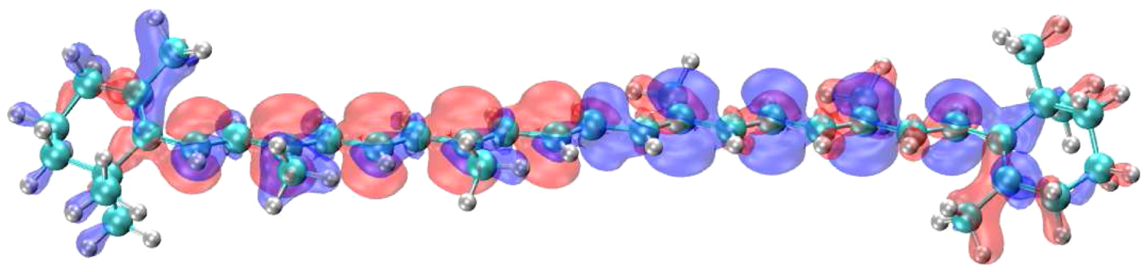


**Figure 2.** *β*-Carotene: transition density of the first excited state (TD-PBE0/def2-SVP).

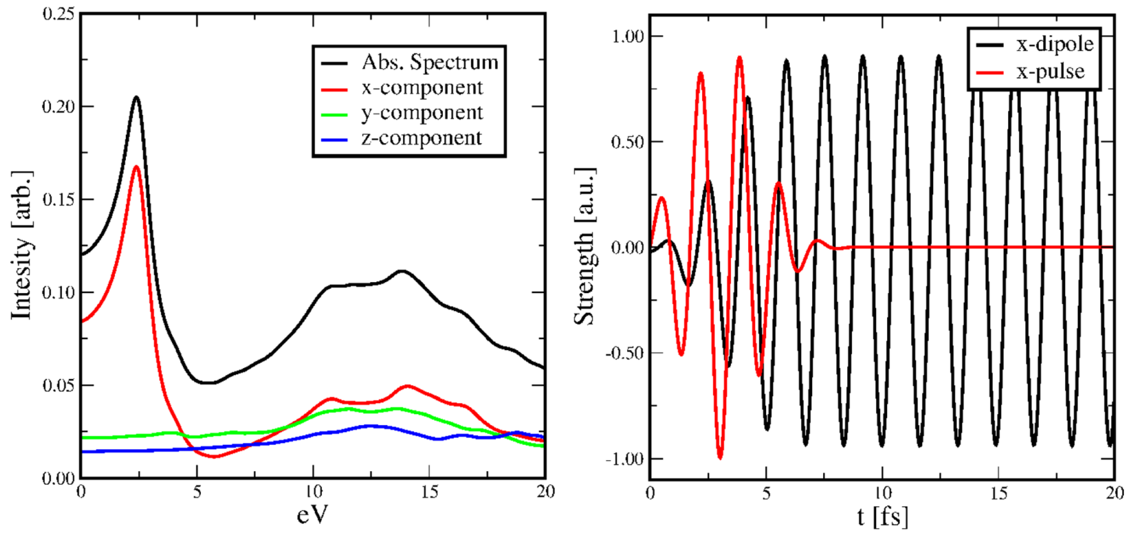


**Figure 3.** Absorption spectrum and resonance excitation of *β*-carotene with PBE0/def2-SVP. (left) Absorption spectrum (black line) from RT-TDDFT calculations as well as single Cartesian dipole components. (right) Dipole fluctuation and external pulse (*x*-component) from RT-TDDFT calculations (pulse strength scaled by $10^4$).

GPU nodes as shown in Figure 1. It should be stressed that the same framework to handle multiple GPUs in a single compute node is employed for both AMD and NVIDIA devices,[10,11] i.e., a single thread is dedicated to a specific GPU. Note that this concept is also retained within a parallel setup containing multiple GPU nodes using MPI, i.e., local resources are parallellized via POSIX threads, and MPI is only used for internode communication. As can be seen from the results shown in Figure 1, we can obtain good scaling with the number of employed GPU nodes even for smaller systems similar to MPI/ CUDA-based calculations,[11] indicating a strong-scaling behavior in an MPI/OpenCL parallel environment. Considering a parallel efficiency threshold of 0.5, the results indicate that these calculations would still benefit if more GPU nodes were employed.

The strong scaling behavior is highly important in the context of ab initio simulations over significant time lengths. Thus, as a final example, we applied our MPI/OpenCL-parallelized code to real-time time-dependent DFT (RT-TDDFT) calculations at the PBE0/def2-SVP level to simulate the resonance excitaton of the first excited state in *β*-carotene (Figure 2, 96 atoms). Here, we employ the approximations to the propagator as discussed in ref 29 and use a time-step of $\Delta t = 24$ as for a total of 20 fs. The absorption spectrum on the left in Figure 3 is obtained by three RT-TDDFT calculations, where at $t = 0$ a dipole pertubation $(x,y,z)$ with a strength of $10^{-4}$ au is applied. The first excitation at approximately 2.5 eV is mostly in the *x*-direction. To simulate a resonance excitation, we performed further RT-TDDFT calculations where we apply a dipole-pulse in the *x*-direction of frequency 2.5 eV. The pulse shape is defined by a Gaussian window-function with a width of 3.1 fs. As can be seen on the right in Figure 3, the pulse incites fluctuations in the *x*-component of the molecular dipole, which prevail after the pulse vanishes, i.e., a free oscillation because we hit an eigenfrequency of the system. For all RT-TDDFT calculations, we employed six GPU nodes with four AMD FirePro W8100 GPUs each, so that a simulation for 20 fs and approximately 850 time-steps can be performed in ~1.3 h. A single time-step takes around 5.5 s on average, where it should be stressed that each step contains at least two complete Fock-builds using a complex-valued density matrix, i.e., an additional exchange matrix build is necessary. Also note that so far all linear algebra operations, i.e., the formation of the propagator by matrix exponentiation, are executed on the master node only.

## IV. CONCLUSIONS

We have presented a first implementation of efficient integral routines for Hartree−Fock and Kohn−Sham DFT-based calculations using OpenCL to utilize AMD GPUs in ab initio quantum chemistry. It has been shown that the performance gain is comparable to NVIDIA GPUs or even better. The strong-scaling behavior was also employed to show that real-time TDDFT simulations can be performed very efficiently when multiple GPU nodes are employed.

If a CUDA-based implementation is already available, the extension to OpenCL and thus AMD GPUs is straightforward, and because we found that recent GPU drivers have shown a stable performance, we think that OpenCL and therefore AMD GPUs are a viable alternative to CUDA and NVIDIA GPUs in quantum chemical calculations. Furthermore, because NVIDIA currently has a virtual monopoly on GPU-accelerated high-performance computing, AMD GPUs are currently significantly lower priced while providing similar computational efficiency. Combined with the possibility of easily porting algorithms to different architectures also including Intel Xeon Phi or conventional CPUs, using OpenCL in ab initio program packages can be very attractive to many researchers.

## ■ AUTHOR INFORMATION

### Corresponding Author

*E-mail: christian.ochsenfeld@uni-muenchen.de.

### ORCID

Christian Ochsenfeld: 0000-0002-4189-6558

### Funding

C.O. acknowledges financial support by the SFB 749 "Dynamik und Intermediate molekularer Transformationen" (DFG) and the DFG cluster of excellence (EXC 114) "Center for Integrative Protein Science Munich" (CIPSM).

### Notes

The authors declare no competing financial interest.

## ■ REFERENCES

(1) Yasuda, K. Two-electron integral evaluation on the graphics processor unit. *J. Comput. Chem.* **2008**, *29*, 334−342.

(2) Yasuda, K. Accelerating Density Functional Calculations with Graphics Processing Unit. *J. Chem. Theory Comput.* **2008**, *4*, 1230−1236.

(3) Ufimtsev, I. S.; Martínez, T. J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation. *J. Chem. Theory Comput.* **2008**, *4*, 222−231.

(4) Ufimtsev, I. S.; Martínez, T. J. Graphical Processing Units for Quantum Chemistry. *Comput. Sci. Eng.* **2008**, *10*, 26−34.

(5) Ufimtsev, I. S.; Martínez, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *J. Chem. Theory Comput.* **2009**, *5*, 1004−1015.

(6) Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units (GPUs). *J. Chem. Theory Comput.* **2011**, *7*, 949−954.

(7) Isborn, C. M.; Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. Excited-State Electronic Structure with Configuration Interaction Singles and Tamm-Dancoff Time-Dependent Density Functional Theory on Graphical Processing Units. *J. Chem. Theory Comput.* **2011**, *7*, 1814−1823.

(8) Wu, X.; Koslowski, A.; Thiel, W. Semiempirical Quantum Chemical Calculations Accelerated on a Hybrid Multicore CPUGPU Computing Platform. *J. Chem. Theory Comput.* **2012**, *8*, 2272−2281.

(9) Asadchev, A.; Gordon, M. S. New Multithreaded Hybrid CPU/GPU Approach to Hartree-Fock. *J. Chem. Theory Comput.* **2012**, *8*, 4166−4176.

(10) Kussmann, J.; Ochsenfeld, C. Pre-selective screening for matrix elements in linear-scaling exact exchange calculations. *J. Chem. Phys.* **2013**, *138*, 134114.

(11) Kussmann, J.; Ochsenfeld, C. Pre-Selective Screening for Linear-Scaling Exact Exchange Gradient Calculations for Graphics Processing Units and General Strong-Scaling Massively-Parallel Calculations. *J. Chem. Theory Comput.* **2015**, *11*, 918−922.

(12) Maurer, S.; Kussmann, J.; Ochsenfeld, C. A Reduced Scaling J-Engine Based Reformulation of SOS-MP2 using Graphics Processing Units. *J. Chem. Phys.* **2014**, *141*, 051106.

(13) Hohenstein, E. G.; Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. An atomic orbital-based formulation of the complete active space self-consistent field method on graphical processing units. *J. Chem. Phys.* **2015**, *142*, 224103.

(14) Snyder, J. W.; Hohenstein, E. G.; Luehr, N.; Martínez, T. J. An atomic orbital-based formulation of analytical gradients and non-adiabatic coupling vector elements for the state-averaged complete active space self- consistent field method on graphical processing units. *J. Chem. Phys.* **2015**, *143*, 154107.

(15) Fales, B. S.; Levine, B. G. Nanoscale Multireference Quantum Chemistry: Full Configuration Interaction on Graphical Processing Units. *J. Chem. Theory Comput.* **2015**, *11*, 4708−4716.

(16) Gill, P. M. W.; Pople, J. A. The prism algorithm for two-electron integrals. *Int. J. Quantum Chem.* **1991**, *40*, 753−772.

(17) Nvidia cuda toolkit ver. 7.5 (September 2015). https://developer.nvidia.com/cuda-toolkit (2015).

(18) Note that the term "core" as used by, e.g., NVIDIA, basically represents a SIMD lane that can only execute in lock-step, i.e., it cannot execute independently as compared to a CPU core.

(19) Kussmann, J.; Beer, M.; Ochsenfeld, C. Linear-Scaling Self-Consistent Field Methods for Large Molecules. *WIREs Comput. Mol. Sci.* **2013**, *3*, 614−636.

(20) Weigend, F.; Ahlrichs, R. Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297−3305.

(21) AMD APP SDK, version 3.0. http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/.

(22) Mura, M. E.; Knowles, P. J. Improved radial grids for quadrature in molecular density-functional calculations. *J. Chem. Phys.* **1996**, *104*, 9848−9858.

(23) Lebedev, V. I.; Laikov, D. N. A quadrature formula for the sphere of the 131st algebraic order of accuracy. *Doklady Mathematics* **1999**, *59*, 477−481.

(24) Stratmann, R. E.; Scuseria, G.; Frisch, M. J. Achieving linear scaling in exchange-correlation density functional quadratures. *Chem. Phys. Lett.* **1996**, *257*, 213−223.

(25) Kussmann, J.; Ochsenfeld, C. Linear-Scaling Method for Calculating Nuclear Magnetic Resonance Chemical Shifts Using Gauge-Including Atomic Orbitals within Hartree-Fock and Density-Functional Theory. *J. Chem. Phys.* **2007**, *127*, 054103.

(26) Kussmann, J.; Ochsenfeld, C. A density matrix-based method for the linear-scaling calculation of dynamic second- and third-order properties at the Hartree-Fock and Kohn-Sham density functional theory levels. *J. Chem. Phys.* **2007**, *127*, 204103.

(27) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865−3868.

(28) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple [Erratum]. *Phys. Rev. Lett.* **1997**, *78*, 1396.

(29) Cheng, C.-L.; Evans, J. S.; Van Voorhis, T. Simulating molecular conductance using real-time density functional theory. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2006**, *74*, 155112.