JCTC Journal of Chemical Theory and Computation

Massively Multicore Parallelization of Kohn–Sham Theory

Philip Brown,[†] Christopher Woods,[†] Simon McIntosh-Smith,[‡] and Frederick R. Manby*[,†]

Centre for Computational Chemistry, School of Chemistry, University of Bristol, Bristol, BS8 1TS, United Kingdom, and ClearSpeed Technology plc, 3110 Great Western Court, Hunts Ground Road, Bristol, BS34 8HP, United Kingdom

Received July 1, 2008

Abstract: A multicore parallelization of Kohn–Sham density functional theory is described, using an accelerator technology made by ClearSpeed Technology. Efficiently scaling parallelization over 2304 cores is achieved. To deliver this degree of parallelism, the Coulomb problem is reformulated to use Poisson density fitting with numerical quadrature of the required three-index integrals; extensive testing reveals negligible errors from the additional approximations.

1. Introduction

Recent advances in computing technology and algorithm design have allowed ab initio electronic structure theory methods to be applied to large biological molecules.[1,2] However to model these systems realistically, relevant free energy differences must be computed. To do this, one must perform dynamics calculations requiring many thousands of calculations. Electronic structure calculations are computationally demanding, so treatment of a full biological system is generally impractical. The quantum mechanical/molecular mechanical (QM/MM) method divides a system into a small QM region and a larger residue which is treated with classical techniques, significantly reducing the size of the QM calculations required.[3−5] Despite this, QM/MM dynamics[6,7] have been largely restricted to computationally inexpensive semiempirical QM techniques such as AM1,[8] PM3,[9] or tight binding.[10]

Density functional theory (DFT)[11−13] provides an excellent balance of accuracy and computational cost; however, current implementations are an order of magnitude too slow for QM/MM dynamics in enzymological problems: a 20 000 time-step dynamics calculation on a 50-atom QM region might take around 1 year using a conventional serial implementation. New accelerator technologies can provide significant performance gains compared to commodity central processing units (CPUs).

Several groups have investigated the use of graphics processing units (GPUs) to calculate two-electron repulsion integrals (ERIs),[14,15] one of the bottlenecks in many electronic structure methods. They report speedups of $8-15\times$[14] and $80-130\times$[15] for their ERI kernels. Yasuda has also implemented exchange-correlation quadrature using GPUs[16] and reports a speedup of $5-10\times$ against a commodity CPU. However, both groups had to make significant efforts to minimize the errors caused by the lack of double-precision support on current GPUs. We feel that this would significantly complicate the programming effort for a full DFT implementation.

ClearSpeed Technology plc produces a mature, low-power accelerator, with full double-precision support. Each CSX600 chip has 96 single-instruction, multiple-data (SIMD) processing elements (PEs) with 6 KB cache each, providing roughly 33 billion floating point operations per second (FLOPS) of double-precision performance in a matrix multiplication.[17] The ClearSpeed e620 mounts two CSX600 chips with 1 GB of random access memory (RAM) on a PCI-Express card.[18] An e620 consumes roughly 33 W. The ClearSpeed-accelerated tera-scale system (CATS) allows twelve e620 boards housed in one rack-mounted server unit to be attached to a host, providing aggregate performance of ~1 TFLOPS. We present a heterogeneous approach to accelerate DFT, combining ClearSpeed's low-power 64-bit accelerator technology in parallel with the host CPU.

* To whom correspondence should be addressed. E-mail: fred.manby@bristol.ac.uk.
[†] University of Bristol.
[‡] ClearSpeed Technology plc.

The ClearSpeed architecture requires a very fine degree of parallelization. To use a CATS node efficiently, work must be divided over $12 \times 2 \times 96 = 2304$ processing elements. A significant effort has been made over the past years to parallelize DFT for the relatively coarse architecture of multicore workstations[19] and vector supercomputers.[22−25] However, good scaling has been achieved mainly over tens or hundreds of processors. Efforts have also focused on implementing parallel linear-scaling methods,[20,21] which are less important for the relatively small QM region in a QM/MM dynamics calculation. We therefore propose a different approach, which uses the Poisson density fitting method[26−28] to shift all of the bottlenecks into finely parallelizable numerical quadrature.

2. Theory

DFT has two main bottlenecks when applied to ~50 atom systems: the evaluation of the Coulomb matrix,

$$J_{\alpha\beta} = \sum_{\gamma\delta} \gamma_{\gamma\delta}(\alpha\beta|\gamma\delta) \tag{1}$$

and the numerical quadrature to evaluate the exchange-correlation contribution to the Fock matrix

$$V^{\text{xc}}_{\alpha\beta} = \int \mathrm{d}\vec{r}\; v^{\text{xc}}(\vec{r})\chi_\alpha(\vec{r})\chi_\beta(\vec{r}) \approx \sum_\lambda w_\lambda v^{\text{xc}}_\lambda \chi_{\alpha\lambda}\chi_{\beta\lambda} \tag{2}$$

Here and throughout, we use the notation $(\cdot|\cdot)$ to denote a 2-electron repulsion integral, so for example

$$(\alpha\beta|\gamma\delta) = \int \mathrm{d}\vec{r}_1 \int \mathrm{d}\vec{r}_2 \frac{\chi_\alpha(\vec{r}_1)\chi_\beta(\vec{r}_1)\chi_\gamma(\vec{r}_2)\chi_\delta(\vec{r}_2)}{r_{12}} \tag{3}$$

The numerical quadrature runs over points $\vec{r}_\lambda$ with weights w_λ, and $v^{\text{xc}}_\lambda = v^{\text{xc}}(\vec{r}_\lambda)$ and $\chi_{\alpha\lambda} = \chi_\alpha(\vec{r}_\lambda)$. For much larger systems, quadrature becomes less of a bottleneck, because screening rapidly renders this an $O(N)$ step. Diagonalization becomes a serious bottleneck (scaling as $O(N^3)$) but can be avoided (see, for example, ref 29). The Coulomb problem asymptotically scales as $O(N^2)$ if screening is used, but can be made linear-scaling through the fast multipole method.[30−32]

It is straightforward to parallelize numerical quadrature by distributing batches of integration points between processing elements. The Coulomb term is more problematic. Direct calculation of the Coulomb contribution requires four index electron repulsion integrals (ERIs), $(\alpha\beta|\gamma\delta)$, which for f shells would require a matrix of 10 000 numbers occupying 78 KB of memory. This is difficult to efficiently map to an architecture with only 6KB of local store per PE. Implementations on GPUs,[14,15] which face similar limitations, have used Rys quadrature[33] for higher angular momenta. We propose to avoid the calculation of these ERIs altogether, by a combination of density fitting and use of the Poisson equation.

2.1. Density Fitting. To avoid the need to calculate 4-index ERIs, we use the density fitting method, first proposed by Boys and Shavitt in 1959,[36] and extended to DFT by Baerends et al.[34] and Dunlap et al.[35] The conventional Kohn−Sham density

$$\rho(\vec{r}) = \sum_{\alpha\beta} \gamma_{\alpha\beta}\chi_\alpha(\vec{r})\chi_\beta(\vec{r}) \tag{4}$$

is approximated by an auxiliary basis, Ξ_A:

$$\tilde{\rho}(\vec{r}) = \sum_B d_B \Xi_B(\vec{r}) \tag{5}$$

Rewriting eq 1, the Coulomb contribution becomes

$$J_{\alpha\beta} = (\alpha\beta|\rho) \approx (\alpha\beta|\tilde{\rho}) = \sum_B d_B(\alpha\beta|B) \tag{6}$$

The fitting coefficients, d_B are obtained by minimizing the Coulomb self-energy of the fitting residual

$$\Delta = \frac{1}{2}(\rho - \tilde{\rho}|\rho - \tilde{\rho}) \tag{7}$$

This leads to the linear equations

$$\sum_B J_{AB} d_B = c_A \tag{8}$$

where

$$c_A = \sum_{\gamma\delta} \gamma_{\gamma\delta}(A|\gamma\delta) \tag{9}$$

and $J_{AB} = (A|B)$. Solving the fitting equations, eq 6 can then be used to give the Coulomb contribution to the Fock matrix, $J_{\alpha\beta}$. This method uses only three-index integrals of the form $(A|\gamma\delta)$. However, analytic calculation of these three-index integrals for f shells still requires a matrix of 1000 numbers, occupying 8 KB, posing a problem for a highly parallel implementation. In principle, the Coulomb potential

$$V_A(\vec{r}) = \int \mathrm{d}\vec{r}_1 \frac{\Xi_A(\vec{r})}{r_{12}} \tag{10}$$

of each fitting function could be evaluated on a quadrature grid and a numerical integration performed. However, the Coulomb potential is long-ranged, and we found that grids optimized for exchange-correlation quadrature did not give acceptable accuracy. We therefore use the Poisson method to convert most of our Coulomb integrals to overlap integrals,[26−28] which we can then evaluate using conventional DFT quadrature.

2.2. Density-Fitted Poisson Method. Manby and Knowles noticed simplifications in density fitting if the density is fitted in so-called Poisson functions: these are obtained by applying the Poisson operator $\hat{P} = -(4\pi)^{-1}\nabla^2$ to Gaussian-type orbitals.[26] The density is expanded in these Poisson functions:

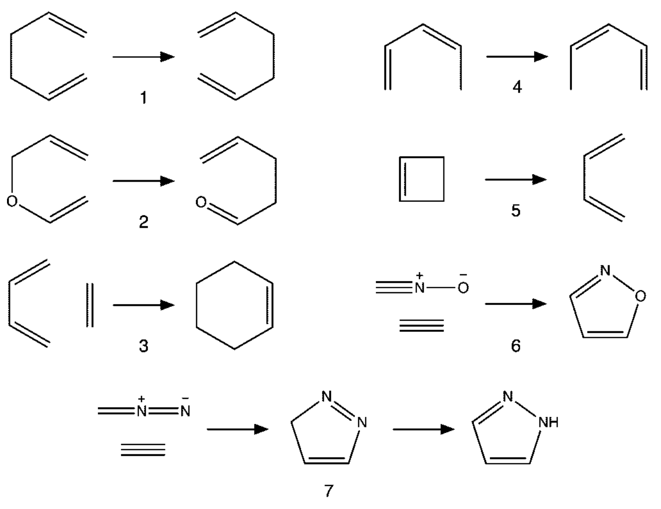
$$\tilde{\rho}(\vec{r}) = \sum_A d_A \hat{P}\Xi_A(\vec{r}) \tag{11}$$

and, using the integral identity

$$\Xi(\vec{r}_1) = \int \mathrm{d}\vec{r}_2 \frac{\hat{P}\Xi(\vec{r}_2)}{r_{12}} \tag{12}$$

the Coulomb matrix elements in the fitting basis simplify to short-ranged three-dimensional integrals, which differ from kinetic energy integrals only by a numerical factor:

$$J_{AB} = (A|B) = \int \mathrm{d}\vec{r}\; \Xi_A(\vec{r})\hat{P}\Xi_B(\vec{r}) \tag{13}$$



8 HCN → CNH 9 $HCOOC_2H_5$ → $HCOOH + C_2H_4$

10 $3C_2H_2$ → C_6H_6 11 HCHO → $CO + H_2$

Figure 1. Selected reactions of molecules for which barriers have been computed.

Similarly, three-index Coulomb integrals, $(A|\gamma\delta)$, can be rewritten as simple overlap integrals:

$$(A|\gamma\delta) = \int d\vec{r}_1 \int d\vec{r}_2 \frac{\hat{P}_1 \Xi_A(\vec{r}_1)\chi_\gamma(\vec{r}_2)\chi_\delta(\vec{r}_2)}{r_{12}} = \int d\vec{r}\ \Xi_A(\vec{r})\chi_\gamma(\vec{r})\chi_\delta(\vec{r}) \quad (14)$$

Further investigation revealed that the fitted density in eq 11 can have no total charge, dipole, or higher multipoles.[27] To alleviate this problem, a small number of ordinary basis functions are introduced, and these describe the charge and higher multipoles. The Poisson functions move the charge around and produce an accurate model density. Setting up a fitting basis with m_c standard and m_p Poisson functions, the fitted Coulomb matrix J_{AB} can be broken down into three types of integrals: standard Coulomb integrals, standard overlaps, and scaled kinetic-energy-type integrals (eq 13). The three-index integrals, $(\alpha\beta|A)$, block into $m_c m(m + 1)/2$ Coulomb integrals, and $m_p m(m + 1)/2$ overlaps, where m is the size of the atomic orbital basis.[27] The small number of standard Coulomb integrals and kinetic energy-like integrals are calculated explicitly, but the overlaps can calculated by quadrature.

This grid-based density-fitted Poisson method (GDFP) for the Coulomb problem has been implemented in serial within Molpro.[37] The energies were calculated on a test set of 21 reactions of small molecules containing first row elements (see Table V of ref 38) and some reactions of larger molecules, Figure 1.[39] Barrier heights for the larger reactions were also calculated. Calculations were performed with the BLYP functional, a cc-pVDZ orbital basis, a cc-pVTZ/jkfit fitting basis[40] for conventional density-fitted Kohn−Sham (DFKS), and the Poisson cc-pVTZ fitting set, described by Polly et al.[28] for density-fitted Poisson (DFP). Probability density plots of the errors of DFKS, DFP, and GDFP relative to standard KS for reaction energies and barriers (Figure 2) show that the grid-based method gives comparable accuracy to the standard DFP method and is in either case not worse than DFKS.

3. Implementation

The GDFP method and standard exchange-correlation quadrature have also been parallelized for the ClearSpeed accelerator technology. The ClearSpeed 3.0 software development kit (SDK) was used for the implementation. Accelerated code was written in C^n, a language that extends ANSI C[41] through the addition of keywords `mono` and `poly`. These specify scalar and parallel data types respectively, so `poly` data are distributed across all 96 PEs. A full set of standard C libraries as well as optimized mathematics libraries are available. The SDK also provides an implementation of BLAS double-precision matrix multiplication (DGEMM)[42] accessible from C^n, which gives easy access to the full 33 GFLOPS of the CSX600. However, for peak performance the matrices must have dimensions of around 1000 or more and use a blocked data format.

A fifty-atom molecule with a 6-31G* basis will have roughly 600 functions and require a quadrature grid of around 200 000 points. The atomic orbitals (AOs) evaluated on this grid therefore require roughly 900 MB of memory, assuming screening is not used. To avoid excessive communication between the accelerator and the host system, we choose to calculate and use the orbitals on the grid on the accelerator cards. We therefore pass only information about the grid, basis, Fock matrix, and density matrix between the accelerators and the host system. The grid is split equally between each CSX600 chip, and each batch, along with the entire basis, passed to the accelerators.

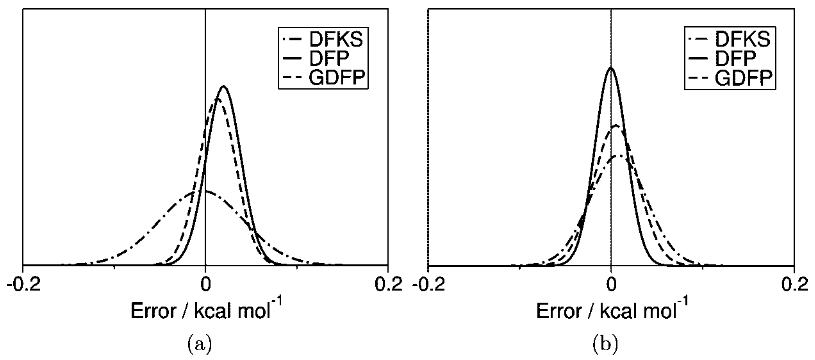


Figure 2. Probability density plots of errors in (a) reaction energy and (b) barrier heights, relative to standard KS method.

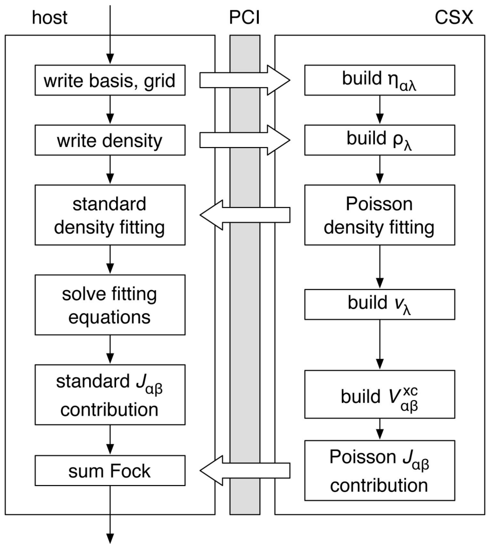


Figure 3. Implementation of GDFP and exchange-correlation quadrature on a hybrid system.

The small local store available on each PE is a challenge for efficient evaluation of the orbitals on the grid,

$$\chi_{\alpha\lambda} = x_\lambda^l y_\lambda^m z_\lambda^n \mathrm{e}^{-\alpha r_\lambda^2} \tag{15}$$

where x_λ, y_λ, and z_λ are the distances from the grid point λ to the center of the orbital on the respective axes, and $r_\lambda^2 = x_\lambda^2 + y_\lambda^2 + z_\lambda^2$. We choose to evaluate the entire basis, one basis group at a time, on eight grid points per PE, transferring the results to board RAM when each group has been completed. We then work through the grid in these tranches of $8 \times 96 = 768$ points. This allows us to make most efficient use of the processing power of the CSX600, while maximizing data bandwidth between PE local store and RAM. If enough RAM is available, we store the AOs on the grid to avoid recalculating them at every iteration. Otherwise we calculate the AOs in the largest block possible.

Further calculations on the card, such as building the density on the grid, are performed with all available AOs on the grid, to maximize the performance of the DGEMM calls. Figure 3 shows the implementation of one iteration the method when enough RAM is available to store the AOs on the grid. The accelerators are initialized, each given a portion of the grid and the complete basis set and the AOs on the grid calculated once, during the first iteration.

We treat the host and card environments as parallel pipelines. During every iteration of the calculation, the density matrix, $\gamma_{\alpha\beta}$, is passed to each card and a density on the grid calculated,

$$\rho_\lambda = \sum_{\alpha\beta} \gamma_{\alpha\beta}\chi_{\alpha\lambda}\chi_{\beta\lambda} \tag{16}$$

The vector

$$c_A = \sum_\lambda w_\lambda \rho_\lambda \Xi_{A\lambda}, \quad A \in \text{Poisson} \tag{17}$$

is calculated and passed back to the host. The host has calculated the conventional integrals,

$$c_A = \sum_{\alpha\beta} (\alpha\beta|A), \quad A \in \text{standard} \tag{18}$$

and can then solve

$$d_B = \sum_A [\mathbf{J}^{-1}]_{AB} c_A \tag{19}$$

for the fitting coefficients. The coefficients for the Poisson section of the fitting basis are transferred to the accelerators, and the contribution, to the Coulomb matrix,

$$J_{\alpha\beta} \leftarrow \sum_{B\in \text{Poisson}} d_B \sum_\lambda w_\lambda \chi_{\alpha\lambda}\chi_{\beta\lambda}\Xi_{B\lambda} \tag{20}$$

is built. The accelerators also calculate the exchange-correlation potential

$$v_\lambda^{\text{xc}} = f(\rho_\lambda) \tag{21}$$

analogues for density gradients, and all relevant contributions to the exchange-correlation matrix,

$$V_{\alpha\beta}^{\text{xc}} = \sum_\lambda w_\lambda v_\lambda^{\text{xc}} \chi_{\alpha\lambda}\chi_{\beta\lambda} \tag{22}$$

Meanwhile, the host calculates the conventional Gaussian contribution to the Coulomb matrix,

$$J_{\alpha\beta} \leftarrow \sum_{B\in \text{standard}} d_B(\alpha\beta|B) \tag{23}$$

All contributions to the Coulomb and exchange-correlation matrices are returned to the host and summed into the Fock matrix,

$$F_{\alpha\beta} \leftarrow \frac{1}{2}J_{\alpha\beta} + V_{\alpha\beta}^{\text{xc}} \tag{24}$$

The calculation of the density on the grid and contributions to the Fock matrix are key steps as they scale $O(N^3)$ with molecular size. Fortunately they can be decomposed to a cubic-scaling matrix-multiplication, along with some matrix−vector operations with lower scaling. The calculation of the density on the grid (eq 16) breaks down the matrix multiplication,

$$\tilde{\chi}_{\alpha\lambda} = \sum_\beta \gamma_{\alpha\beta}\chi_{\beta\lambda} \tag{25}$$

and a series of dot products, one for each grid point,

$$\rho_\lambda = \sum_\alpha \chi_{\alpha\lambda}\tilde{\chi}_{\alpha\lambda} \tag{26}$$

Similar decompositions can be applied to eqs 20 and 22. Using the C^n DGEMM implementation allows us to harness the full power of the ClearSpeed accelerators, and we routinely see 26 GFLOPS (80% of peak) per CSX600 in our DGEMMs and thus 624 GFLOPS aggregate on a single CATS node. All other steps, such as the calculation of the Coulomb fitting vector (eq 17) or exchange-correlation potentials (eq 21) are computationally trivial for our target molecules. Additionally, the calculation of the density on the grid can be shared between the Coulomb and exchange-correlation, enhancing our efficiency. Finally we are able to overlap the calculation on the accelerators with the computation of the conventional Coulomb integrals by the host.

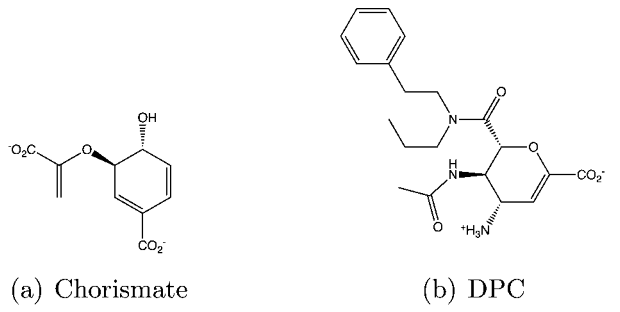


Figure 4. Structures of test molecules.

Table 1. Timings for the Accelerated GDFP Method versus Standard DFP and Density-Fitted KS Methods[a]

	AO basis	fit basis	Wall times (s)	
	N_{AO}	N_{Fit}	Fock build	total
		DPC		
	6-31G*	cc-pVDZ/jkfit		
DFKS	446	2581	2050	2383
DFP	446	3382	1380	1413
grid DFP	446	3382	80	112
	cc-pVTZ	cc-pVTZ/jkfit		
DFKS	1218	3022	6730	7096
DFP	1218	3382	4043	4409
grid DFP	1218	3382	278	644
		Chorismate		
	6-31G*	cc-pVDZ/jkfit		
DFKS	240	1304	398	404
DFP	240	1680	283	289
grid DFP	240	1680	18	24
	cc-pVTZ	cc-pVTZ/jkfit		
DFKS	592	1504	1281	1323
DFP	592	1680	1154	1194
grid DFP	592	1680	41	82
		Ala_{12} helix		
	6-31G*	cc-pVDZ/jkfit		
DFKS	978	5696	9747	10489
DFP	978	7476	6577	7327
grid DFP	978	7476	446	867

[a] Calculations were performed on one core of a 2× Dual Core Opteron 2218 2.6 GHz server with 8 GB RAM attached to one ClearSpeed CATS (12 xe620 cards). All DFP calculations use the Poisson cc-pVTZ fitting set described by Polly et al.[28] and the BLYP exchange-correlation functional.

4. Results

A timing analysis of the method for chorismate ($C_{10}H_8O_6$, reactant substrate of chorismate mutase, Figure 4a[43]), a typical neuraminidase ligand, DPC ($C_{20}H_{27}O_5N_3$, Figure 4b[44]), and an alanine helix (Ala_{12}, $C_{37}H_{61}O_{13}N_{12}$), for various basis sets is shown in Table 1. Overall application speedup varies from 7× to 15× versus DFP and 11× to 19× compared to DFKS. If we consider only the time spent constructing the Fock matrix, we see between 15× and 30× acceleration compared to host-only DFP.

At the moment, we have not implemented any screening within the GDFP method, so the accelerators are performing considerably more work than the equivalent host implementation. We anticipate a further factor of at least 1.5× from the implementation of screening. Additionally, since the Coulomb energy has a significantly higher contribution to the total energy than the exchange-correlation, to maintain numerical stability we were forced to use the fine grid for

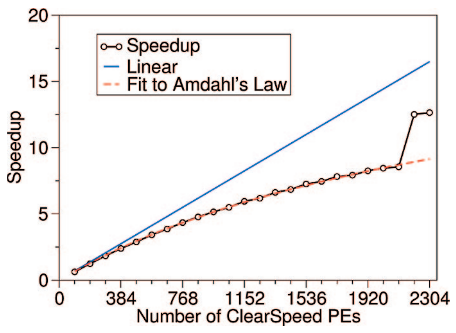


Figure 5. Scaling of GDFP with a number of ClearSpeed processing elements for DPC with a 6-31G* AO basis, Poisson cc-pVTZ fitting set, and the BLYP exchange-correlation functional. The dashed line is a fit of the first 22 data points to Amdahl's law, corresponding to a 95% parallelization of the code. Calculations were performed on one core of a 2× Dual Core Opteron 2218 2.6 GHz with 8 GB RAM attached to one ClearSpeed CATS (12 × e620 cards).

Table 2. Breakdown of the Calculation on DPC with a 6-31g* AO Basis, Poisson cc-pVTZ Fitting Set, and the BLYP Exchange-Correlation Functional[a]

component	time %	time in DGEMM %
exchange-correlation	58	17
Coulomb	27	9
common	15	11

[a] Calculation performed on one core of a 2× Dual Core Opteron 2218 2.6 GHz server with 8 GB RAM attached to one ClearSpeed CATS (12 × e620 cards). The "common" times are for building the orbitals and density on the grid, which are needed for evaluation of both exchange-correlation and Coulomb contributions.

all of the iterations of the GDFP calculation. Figure 5 shows the scaling of the method relative to number of ClearSpeed processing elements for DPC with a 6-31G* basis and the BLYP exchange-correlation functional. The method scales well over the 2304 processing elements of a CATS node.

Fitting to Amdahl's law reveals that we have parallelized ~95% of the calculation. We can clearly see that the remaining work on the host has become a bottleneck; the diagonalization of the Fock equations now takes ~30% of the total runtime. We can also observe a significant performance gain for the last two points, where enough memory becomes available to store the AOs on the grid, removing a significant portion of the work performed by the accelerators. It is also important to note that if we consider only time spent on the accelerator cards, we see perfect linear speedup, with a significant jump above linearity when storing the AOs becomes possible.

A breakdown of the time spent on the accelerator cards is given in Table 2 for DPC. Building the orbitals and density on the grid is common to both parts. It is worth noting that while DGEMMs account for 98% of the floating point operations, they only take ~37% of the time. This suggests significant opportunity for improving the efficiency of the implementation of other sections of code.

Due to the nature of the architecture, we pad the matrix dimensions to a multiple of 96. This leads to significant inefficiency for small molecules, especially with a small basis. Timings for ethane with one accelerator card are given

Table 3. Timings for Ethane for the Accelerated GDFP Method versus Standard DFP and Density-Fitted KS Methods[a]

	AO basis	fit basis	
	N_{AO}	N_{Fit}	wall time(s)
	6-31G*	cc-pVDZ/jkfit	
DFKS	40	278	8
DFP	40	362	8
grid DFP	40	362	5
	cc-pVTZ	cc-pVTZ/jkfit	
DFKS	144	338	35
DFP	144	362	30
grid DFP	144	362	8

[a] Calculations were performed on one core of a 2× Dual Core Opteron 265 1.8 GHz workstation with 4 GB RAM and one ClearSpeed xe620 card. All DFP calculations use the Poisson cc-pVTZ fitting set described by Polly et al.[28] and the BLYP exchange-correlation functional.

in Table 3 for two basis sets. With the large basis set, we observe a reasonable speedup of 4×.

5. Conclusions

We have implemented the GDFP method on ClearSpeed accelerators and demonstrated that an order of magnitude speedup is possible, with good scaling over thousands of PEs. The accelerator code shows perfect scaling over 2304 processing elements, while we see the expected behavior for the full application. There are however still many areas to improve. The introduction of screening should improve the efficiency of the method and ensure that it scales effectively to larger problem sizes. We anticipate improving the host/card load balancing at the same time, allowing the host to process batches of grid points. We also aim to implement gradients with respect to the nuclear positions, to allow the method to be used for dynamics calculations. The algorithm we have presented would map well onto GPUs, addressing some of the concerns expressed by Yasuda about the difficulty of fine-grained parallelization of the Coulomb problem.[16] Additionally, the current generation of GPUs have double-precision support,[45] greatly simplifying the implementation. Our algorithm is also suitable for implementation on standard shared memory parallel architectures, such as multicore x86, on which we expect that the algorithm would scale well.

Acknowledgment. The authors are indebted to several members of ClearSpeed Technology plc for input on algorithm design. They particularly thank Daniel Kidger and Thomas Bradley. The authors are grateful for funding from the School of Chemistry, University of Bristol, ClearSpeed Technology plc, and the Royal Society. Computer time from the Advanced Computing Research Centre at the University of Bristol is gratefully acknowledged.

References

(1) Scuseria, G. *J. Phys. Chem. A.* **1999**, *103*, 4782.

(2) Gogenea, V.; Suárez, D.; van der Vaart, A.; Merz, K. W., Jr. *Curr. Opin. Struct. Biol.* **2001**, *11*, 217.

(3) Friesner, R. A.; Gullar, V. *Annu. Rev. Phys. Chem.* **2005**, *56*, 389.

(4) Warshel, A.; Levitt, M. *J. Mol. Biol.* **1976**, *103*, 227.

(5) Field, M. J.; Bash, P. A.; Karplus, M. *J. Comput. Chem.* **1990**, *11*, 700.

(6) Ridder, L.; Rietjens, I. M. C. M.; Vervoort, J.; Mulholland, A. J. *J. Am. Chem. Soc.* **2002**, *124*, 9926.

(7) Gao, J.; Truhlar, D. G. *Annu. Rev. Phys. Chem.* **2002**, *53*, 467.

(8) Dewar, M. J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. J. P. *J. Am. Chem. Soc.* **1985**, *107*, 3902.

(9) Stewart, J. J. P. *J. Comput.-Aided. Mol. Des.* **1990**, *4*, 1.

(10) Frauenheim, T.; Seifert, G.; Elstner, M.; Hajnal, Z.; Jungnickel, G.; Porezag, D.; Suhai, S.; Scholz, R. *Phys. Stat. Sol. (B)* **2000**, *217*, 41.

(11) Kohn, W.; Sham, L. J. *Phys. Rev. A.* **1965**, *140*, 1133.

(12) Parr, R. G.; Yang, W. *Density-Functional Theory of Atoms and Molecules*; Oxford University Press: New York, 1989.

(13) Kohn, W.; Becke, A. D.; Parr, R. G. *J. Phys. Chem.* **1996**, *100*, 12974.

(14) Yasuda, K. *J. Chem. Theory Comput.* **2008**, *4*, 1230.

(15) Ufimtsev, I. S.; Martínez, T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222.

(16) Yasuda, K. *J. Chem. Theory Comput.*, in press.

(17) Clearspeed CSXL 3.0 User Guide, Section 5. http://support.clearspeed.com/documentation/software/release3 (accessed March 11, 2008).

(18) Advance e620 Product Brief. http://www.clearspeed.com (accessed March 11, 2008).

(19) Baker, J.; Füsti-Molnár, L.; Pulay, P. *J. Phys. Chem. A.* **2004**, *180*, 3040.

(20) Gan, C. K.; Challacombe, M. *J. Chem. Phys.* **2004**, *121*, 6608.

(21) Gan, C. K.; Challacombe, M. *J. Chem. Phys.* **2003**, *118*, 9128.

(22) Von Arnim, M.; Ahlrichs, R. *J. Comput. Chem.* **1998**, *19*, 1746.

(23) Sosa, C. P.; Ochterski, J.; Carpenter, J.; Frisch, M. J. *J. Comput. Chem.* **1998**, *19*, 1053.

(24) Furlani, T. F.; Kong, J.; Gill, P. M. W. *Comput. Phys. Commun.* **2000**, *128*, 170.

(25) Sosa, C. P.; Scalmani, G.; Gomperts, R.; Frisch, M. J. *Parallel Comput.* **2000**, *26*, 843.

(26) Manby, F. R.; Knowles, P. J. *Phys. Rev. Lett.* **2001**, *87*, 163001.

(27) Manby, F. R.; Knowles, P. J.; Lloyd, A. W. *J. Chem. Phys.* **2001**, *115*, 9144.

(28) Polly, R.; Werner, H.-J.; Manby, F. R.; Knowles, P. J. *Mol. Phys.* **2004**, *102*, 2311.

(29) Helgaker, T.; Larsen, H.; Olsen, J.; Jørgensen, P. *Chem. Phys. Lett.* **2000**, *327*, 397.

(30) Rokhlin, V. *J. Comput. Phys.* **1985**, *60*, 187.

(31) White, C. A.; Johnson, B. G.; Gill, P. M. W.; Head-Gordon, M. *Chem. Phys. Lett.* **1994**, *230*, 8.

(32) Strain, M. C.; Scuseria, G. E.; Frisch, M. J. *Science* **1996**, *271*, 51.

(33) Dupuis, M.; Rys, J.; King, H. F. *J. Chem. Phys.* **1976**, *65*, 111.

(34) Baerends, E. J.; Ellis, D. E.; Ros, P. *Chem. Phys.* **1973**, *2*, 41.

(35) Dunlap, B. I.; Connoly, J. W. D.; Sabin, J. R. *J. Chem. Phys.* **1979**, *71*, 3396.

(36) Boys, S. F.; Shavitt, I. *A Fundamental Calculation of the Energy Surface for the System of Three Hydrogen Atoms; Rep WIS-AF-13*; University of Wisconsin Naval Research Laboratory: Madison, WI, 1959.

(37) Werner, H.-J.; Knowles, P. J.; Lindh, R.; Manby, F. R.; Schütz, M.; Celani, P.; Korona, T.; Rauhut, G.; Amos, R. D.; Bernhardsson, A.; Berning, A.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Hampel, C.; Hetzer, G.; Lloyd, A. W.; McNicholas, S. J.; Meyer, W.; Mura, M. E.; Nickla, R.; Schumann, U.; Stoll, H.; Stone, A. J.; Tarroni, R.; Thorsteinsson, T. *Molpro*, Version 2006.4; University College Cardiff Consultants Limited: Cardiff, U.K., 2006; http://www.molpro.net.

(38) Werner, H.-J.; Manby, F. R. *J. Chem. Phys.* **2006**, *124*, 054114.

(39) Nunn, J.; Harvey, J. N.; Manby, F. R. manuscript in preparation.

(40) Eichkorn, K.; Weigend, F.; Treutler, O.; Ahlrichs, R. *Theor. Chim. Acta.* **1997**, *97*, 119.

(41) ISO/IEC 9899 - Programming languages - C. http://www.open-std.org/jtc1/sc22/wg14/www/standards.html#9899 (accessed May 22, 2008).

(42) Basic Linear Algebra Subroutines. http://www.netlib.org/blas/ (accessed May 22, 2008).

(43) Claeyssens, F.; Harvey, J. N.; Manby, F. R.; Mata, R. A.; Mulholland, A. J.; Ranaghan, K. E.; Schutz, M.; Thiel, S.; Thiel, W.; Werner, H. J. *Angew. Chem., Int. Ed.* **2006**, *45*, 6856.

(44) Birch, L.; Murray, C. W.; Hartshorn, M. J.; Tickle, I. J.; Verdonk, M. L. *J. Comput. Aid. Mol. Des.* **2002**, *16*, 855.

(45) nVidia Telsa C1060 Datasheet. http://www.nvidia.com/object/teslaroductiterature.html (accessed July 31, 2008).

CT800261J