Chemical Physics Letters xxx (2018) xxx-xxx

Contents lists available at ScienceDirect

Chemical Physics Letters

journal homepage: www.elsevier.com

Research paper

An efficient implementation of semi-numerical computation of the Hartree-Fock exchange on the Intel Phi processor

Fenglai Liu, Jing Kong*

Department of Chemistry and Center for Computational Science, Middle Tennessee State University, TN, 37130, United States

ARTICLE INFO

Article history:
Received 30 January 2018
Accepted 11 May 2018
Available online xxx

ABSTRACT

Unique technical challenges and their solutions for implementing semi-numerical Hartree-Fock exchange on the Phil Processor are discussed, especially concerning the single- instruction-multiple-data type of processing and small cache size. Benchmark calculations on a series of buckyball molecules with various Gaussian basis sets on a Phi processor and a six-core CPU show that the Phi processor provides as much as 12 times of speedup with large basis sets compared with the conventional four-center electron repulsion integration approach performed on the CPU. The accuracy of the semi-numerical scheme is also evaluated and found to be comparable to that of the resolution-of-identity approach.

1. Introduction

The density functional theory (DFT) is the most widely applied quantum mechanical methods for chemical and material studies. One way to make DFT computation more productive is to take the advantage of new computer technologies, such as the General-Purpose Graphics Processing Units (GPGPU) and the Intel Many Integrated Core (MIC) architecture. The GPGPU extends the use of the graphics processing unit to perform general purpose computation such as scientific calculations. Compared with a traditional CPU, a GPGPU card possesses more processing cores and overall has much higher theoretical peak floating point operations per second (FLOPS). On the other hand the Intel MIC technology is a redesign of the previous generation of Celeron and Pentium cores and condenses these cores onto one processor. Although the MIC architecture has far fewer cores than the GPGPU (currently about 60–70 cores per processor), it introduces the Vector Processing Units (VPU) to each core, which can execute 8 double-precision floating point operations per CPU instruction cycle [1,2]. With the VPU working as a multiplier the MIC processor is capable of providing similar computing capacity as that of the GPGPU card.

In comparison to the significant development effort made on GPGPU from quantum chemistry software community [3–6], less development efforts have been reported for the Phi processor in computational quantum chemistry, perhaps because it came about more recently than the GPGPU. Leang et al. [7] studied the efficiency of matrix operations on the first generation of the Phi processor "Knight Corner" (KNC) and showed that KNC can yield up to three times speedup compared with the host CPU. Apra et al. employed the KNC for CCSD(T) calculation [8], and the implementation also achieved 2–3 times speedup. On the other hand, Reid etc. [9] modified CP2K Program for the KNC and made performance comparison with a dual Xeon CPU(16 cores) system for energy calculations, and found that the code ran about 5.43 times slower on the KNC than on the CPU-only system, even though the theoretical FLOPS count for the dual CPUs is about 0.8 teraFLOPS versus 1 teraFLOPs for the KNC processor. The second generation of Phi processor "Knight Landing" (KNL) supports the direct compilation and execution of the software on the processor. Such an attempt on the quantum chemistry software LSDalton showed [10] that it is still several times slower on the KNL than a dual Xeon CPU system. The difficulty was mostly attributed to the inefficient use of VPUs, which are the SIMD (Single instruction, multiple data) processing units in the Phi processor [1,2,11]. For effectively utilizing the Phi processor one usually need to rewrite software significantly so that it can be efficiently executed on an SIMD architecture.

In a practical DFT calculation, the computation of Hartree-Fock (HF) exchange is the most time-consuming component, especially due to the development of various techniques for the computation of Coulomb and the numerical integration of exchange-correlation functionals [12–14]. We have recently published the implementation of a semi-numerical integration scheme for computing the HF exchange energy and matrix with self-consistent field(SCF) for conventional CPUs [15]. The scheme is similar to the COS-X algorithm [16,17] and pseudo-spectral scheme [18,19]. In this work we describe the implementation of the scheme on the MIC architecture. We showed that the semi-numerical scheme is more efficient for large basis sets and large molecules due to quadratic scaling with respect the basis set size. Furthermore, it requires fewer temporary variables therefore fit better the Phi Processor that has much smaller cache size than a standard CPU. The semi-numerical HF exchange is also an essential ingredient for the emerging local hybrid functionals [17,20–24].

* Corresponding author.
Email address: jing.kong@mtsu.edu (J. Kong)

https://doi.org/10.1016/j.cplett.2018.05.026
0009-2614/

2. The semi-numerical algorithm for HF exchange

The semi-numerical algorithm to calculate the HF exchange matrix has been discussed in detail in our previous paper [15]. We briefly summarize it here.

The HF exchange matrix is derived through the derivative of the HF exchange energy with respect to the spin-resolved density matrix $P^{\sigma}_{\mu\nu}$:

$$\frac{\partial E^{ex}_{\sigma}}{\partial P^{\sigma}_{\mu\nu}} = \sum_{\lambda\eta} P^{\sigma}_{\lambda\eta} \int dr \int dr' \frac{\varphi_{\mu}(r)\varphi_{\nu}(r')\varphi_{\lambda}(r)\varphi_{\eta}(r')}{|r-r'|} = \sum_{\lambda\eta} P^{\sigma}_{\lambda\eta}(\mu\lambda|\nu\eta), \tag{1}$$

where φ represents a general Gaussian basis function, μ,ν, λ and η are basis function indexes. σ denotes the spin of a Molecular orbital, either α or β. $(\mu\lambda|\nu\eta)$ is the abbreviation of four-center ERI. The conventional way is to evaluate this four-center ERI analytically and then form the HF exchange matrix. But the HF exchange matrix can also be calculated through the following semi-numerical scheme:

$$\frac{\partial E^{ex}_{\sigma}}{\partial P^{\sigma}_{\mu\nu}} = \sum_{r}^{grid} w(r)\varphi_{\mu}(r) \sum_{\lambda\eta} P^{\sigma}_{\lambda\eta}\varphi_{\lambda}(r) \int dr' \frac{\varphi_{\nu}(r')\varphi_{\eta}(r')}{|r-r'|} = \sum_{r}^{grid} w(r)\varphi_{\mu}(r)R^{\sigma}_{\nu}(r), \tag{2}$$

where $\varphi_{\mu}(r)$ is the value of a basis function at a given grid point r. $w(r)$ is the numerical weight of the grid point. $R^{\sigma}_{\nu}(r)$ is defined as the kernel of the semi-numerical HF exchange matrix. In this scheme the numerical integration is performed with the standard DFT atom-centered grid scheme [25–27].

To calculate the kernel $R^{\sigma}_{\nu}(r)$ we need to first combine the spin-resolved density matrix with basis set value $\varphi(r)$ through a BLAS (Basic Linear Algebra Subprograms) level 3 matrix-matrix multiplication:

$$Q^{\sigma}_{\eta}(r) = \sum_{\lambda} \varphi_{\lambda}(r) P^{\sigma}_{\lambda\eta}. \tag{3}$$

The kernel of $R^{\sigma}_{\nu}(r)$ can then be expressed as:

$$R^{\sigma}_{\nu}(r) = \sum_{\eta} Q^{\sigma}_{\eta}(r) v_{\nu\eta}(r), \tag{4}$$

where $v_{\nu\eta}(r)$ is the two-center electrostatic potential (ESP) integral:

$$v_{\nu\eta}(r) = \int \frac{\varphi_{\nu}(r')\varphi_{\eta}(r')}{|r-r'|} dr'. \tag{5}$$

In our implementation the code for the ESP integral was generated with an integral code generator program we published earlier called CPPINTS [28]. It is based on a combination of the Obara-Saika(OS) [29,30] and Head-Gordon-Pople(HGP) [31] algorithms, and applies a greedy search algorithm to produce the minimal number of temporary variables.

The rate-determining step in the above scheme is to calculate the ESP integral at each grid point and involves all the effectively nonzero basis function pairs. In practice a basis function is usually a linear combination of primitive Gaussian functions, and thus the ESP integral calculation also involves a loop over the contraction for each pair of Gaussian basis functions. The total cost of ESP integrals then can be approximately estimated as $O(N_{grids} * N_{bfpairs} * N^2_{contraction})$ ("bf" stands for "basis function"). Therefore our implementation effort concentrates on transferring the ESP integral calculation from the normal CPU platform to the Phi processor.

3. Implementation on the Phi processor

Our first try with the Phi processor is to compile the aforementioned implementation of the semi-numerical scheme with the Intel compiler that has the automated optimization feature for the Intel MIC architecture. The test of the resulting binary on the Phi processor KNL7250 is two times slower than running on one six-core E5-1650 CPU, significantly underperforms with respect to the potential of the Phi processor. Clearly, the program needs to be rewritten for a better usage of the Phi processor. In this section we will discuss the special features of the Phi processor, and our experience on the implementation for it.

For achieving successful implementation on the Phi processor there are two key factors involved. Firstly the computation algorithms need to be effectively vectorized into SIMD operations. Secondly, the algorithms need to be designed and implemented carefully to comply with the small cache structure in the many-core architecture of the Phi processor.

The processing unit in each core of a Phi processor is divided into two parts, a redesigned Pentium/Atom core and two VPUs in one core in KNL processor. A VPU is a SIMD processing unit [1], and able to process 8 double precision floating point number simultaneously. As a comparison the floating-point unit (FPU) in a CPU can only handle one double precision floating point number per each instruction cycle. The standard instruction set for the VPU on the KNL processor is the avx-512 instruction set. Modern compilers like Intel compilers contain the so-called automatic vectorization function that can compile the code into this instruction set. However this function does not always work well, because the requirement for a successful vectorization is very strict. For example, an effective vectorization requires the data structure be aligned in unit of the cache line (64 bytes for the Phi processor) and the loop have no backward loop-carried dependencies and be countable [32]. As a consequence, a quantum chemistry program running well on a CPU may still need to be efficiently vectorized to fit the SIMD structure of the Phi processor.

As discussed in the above algorithm section the calculation of ESP integrals is the most time-consuming part. Therefore it is imperative to rewrite the ESP integral code so that to vectorize the code into efficient SIMD operations for the Phi processor. The first modification for the adaptation to the SIMD vectorization is to change the code loop structure. For the normal CPU the calculation for the above algorithm is generally performed in the following three steps (for details regarding the VRR and HRR please refer to the general review [33]):

```
loop over grid points:
loop over Gaussian primitive pairs:
perform vertical recurrence relations(VRR);
end loop
perform horizontal recurrence relations (HRR) to compute
the raw ESP integrals;
digest the raw ESP integrals to produce final result;
end loop
```

For an efficient SIMD vectorization the loop should provide enough and countable cycles for the vectorization. We found that it is

more advantageous to put the loop of grip points inside the loop over primitive pairs as illustrated below:

```
loop over Gaussian primitive pairs:
 loop over grid points:
 perform vertical recurrence relations(VRR);
 end loop
end loop
loop over grid points:
 perform horizontal recurrence relations(HRR) to compute
  the raw ESP integrals;
 digest the raw ESP integrals to produce final result;
end loop
```

This makes the most inner loop is efficiently vectorized.

The second modification for SIMD vectorization is to form aligned struct for integral calculation with VRR. For example, the piece of code below is part of the innermost loop for computing the ESP integral for a (D,D) shell pair on the CPU:

```
double I_ESP_Px_S_M1_vrr=PAX*I_ESP_S_S_M1_vrr-
 PRX*I_ESP_S_S_M2_vrr;
double I_ESP_Py_S_M1_vrr=PAY*I_ESP_S_S_M1_vrr-
 PRY*I_ESP_S_S_M2_vrr;
double I_ESP_Pz_S_M1_vrr=PAZ*I_ESP_S_S_M1_vrr-
 PRZ*I_ESP_S_S_M2_vrr;
double I_ESP_Px_S_vrr=PAX*I_ESP_S_S_vrr-
 PRX*I_ESP_S_S_M1_vrr;
double I_ESP_Py_S_vrr=PAY*I_ESP_SvS_vrr-
 PRY*I_ESP_S_S_M1_vrr;
double I_ESP_Pz_S_vrr=PAZ*I_ESP_S_S_vrr-
 PRZ*I_ESP_S_S_M1_vrr;
```

As one can see, all the operations are scalar operations and fit the FPU well for a CPU. As suggested by Ref. [34], we reorganized the above codes into aligned struct. For the above example piece of code, we can arrange the variables into two C style structs:

```
typedef struct{
 double I_ESP_S_S_M1_vrr;
 double I_ESP_S_S_M2_vrr;
 double I_ESP_S_S_vrr;
 double I_ESP_Px_S_M1_vrr;
 double I_ESP_Py_S_M1_vrr;
 double I_ESP_Pz_S_M1_vrr;
 double I_ESP_Px_S_vrr;
 double I_ESP_Py_S_vrr;
 double I_ESP_Pz_S_vrr;
}__attribute__((aligned(64))) ESPIntsVRR_P_P;
```

and

```
typedef struct{
 double PAX;
 double PAY;
 double PAZ;
 double PRX;
 double PRY;
 double PRZ;
}__attribute__((aligned(64))) ESPIntsVRR_VAR;
```

The C structs are aligned to the 64-bit cache line of the Phi processor to make the data access as efficient as possible. The above sample code then can be trivially rewritten using the structs.

The second important factor for an effective implementation on the Phi processor is the cache size. Compared with the traditional CPU architecture, the Phi processor has much smaller L1/L2 cache and register. The L1 data cache size for the Phi processor is 32 KB per each core, and the L2 cache size per each core is 512 KB. The two caches are shared by 2–4 threads. In comparison, the total cache size of a recent generation of Xeon CPU is about 2 MB per core. Here the semi-numerical scheme has a distinct advantage over the analytical four-center ERI approach. An ESP integral in the semi-numerical scheme is of two-center, and uses far fewer temporary variables for integral calculation than a traditional four-center ERI scheme. Thus it is a better fit for the Phi processor.

Still it is important to utilize the data prefetching pragma to maximize the cache hits on the Phi processor for effectively loading the data from main memory to cache. A simple example below shows how it can be used for assigning values to an array:

```
for (int i=0; i<N; i++){
 #pragma prefetch p:1:16
 #pragma prefetch p:0:6
 for (int j=0; j<2*N; j++){
 p[i*2*N+j] = -1;
 }
}
```

the syntax “p:1:16” means to prefetch the content of array p into the L2 cache for the next 16 cycles, and “p:0:6” means to prefetch the content of p array into L1 cache for the next 6 cycles. Because the cache size in the Phi processor is significantly smaller than the normal CPU, the automatic prefetching data operations by compilers on the Phi processor is designed to be much less aggressive than the CPU [1]. Thus maximizing the use of cache is left to the programmer. Our experience shows that applying the prefetch pragma effectively can boost the performance of the whole implementation by 2–3 times.

We have applied the above techniques to all of the ESP integrals of various shell pair combinations up to (F,F) shell pairs. For the ESP integrals calculation we choose to use the HGP algorithm for implementation, however the above discussions can be applied to the other algorithms such as MD scheme [35] or the RYS scheme [36–38]. Here for this implementation each ESP integral file corresponds to a shell pair combination, and each file has multiple places where the data reorganization is needed for the computation of the fundamental integrals. Furthermore each integral file also has multiple loops that require data prefetching. The efficiency of such a change for the Phi processor is demonstrated in Section 5.

4. Accuracy assessment for the semi-numerical HF exchange algorithm

To assess the accuracy of the semi-numerical implementation on the Phi processor, we performed a benchmark calculation over a series of fullerene molecules including C60, C100, C180 and C240; all in the buckyball form. The basis sets chosen for the test include Pople Basis set 6-31G** and 6-311G(2df), and Dunning Basis set cc-pvdz and cc-pvtz. The implementation of the above semi-numerical scheme is based on the standard atom-centered grid scheme, which

employs the Euler-Maclaurin formula [26,39] for the radial part, and Lebedev grids [27] for the angular part. Three different sets of pruned grid were used. One is the SG-1 grid [39] with each carbon atom having about 3000 points. The other two are the "standard" and "fine" grids from the PQS program [40]. The standard grid employs approximately 5000 points for each Carbon atom, and the fine grid about 9000.

The errors of the semi-numerical scheme are calculated using the standard analytical four-center ERI method as the reference. All the HF SCF calculations employ the core Hamiltonian as the initial guess, which derives the initial density matrix through the diagonalization of the one-electron Hamiltonian matrix. The reference HF exchange matrix was computed with an integral threshold of 1.0×10^{-10}. For the calculations with the semi-numerical scheme, both thresholds for the significant basis function pair and the ESP integral are set to 10^{-12}.

Tables 1–3 list the errors of the semi-numerical scheme for the computation of the HF exchange matrix and the total HF energy in root-mean-square deviation (RMSD), maximum absolute deviation (MAD) and total energy difference per atom. As one can see the RMSD for the HF exchange matrix is at the level of 1.0×10^{-6}, which is two orders of magnitude smaller than that of MAD. The accuracy for the HF exchange matrix improves as the grid becomes finer. But the fineness of the grid does not appear to affect the accuracy of the energy measured as the total energy per atom. The largest errors for the energy is between 150 and 200 $\mu E_h/atom$, comparable to the error of the Resolution-of-Identity method for the HF exchange [41]. It is not clear why the accuracy in energy does not respond to the fineness of the grid while the matrix does. We should note that these pruned grids are fine-tuned for the computation of DFT exchange-correlation energy. In our previous test [15], we observed that the error in energy with an unpruned grid set (128,302) (the radial part is 128 points and the angular part uses 302 points per each atom) is about 5 times smaller than the one with the SG-1 grid. More studies are needed to understand and reduce this discrepancy.

Table 1
Root-mean-square deviations of the matrix elements for the semi-numerical HF exchange with different levels of grid.

Basis set	Molecule	SG-1	Standard	Fine
6-31G**	c60	1.04E−05	3.31E−06	1.72E−06
	c100	6.12E−06	1.87E−06	1.26E−06
	c180	4.89E−06	3.75E−06	3.45E−06
	c240	4.11E−06	3.13E−06	2.93E−06
6-311G(2df)	c60	1.23E−05	4.04E−06	2.06E−06
	c100	7.18E−06	1.99E−06	9.82E−07
	c180	5.05E−06	2.98E−06	2.47E−06
	c240	4.40E−06	2.50E−06	2.20E−06
cc-pvdz	c60	1.31E−05	4.29E−06	2.20E−06
	c100	7.37E−06	2.08E−06	1.20E−06
	c180	5.63E−06	3.87E−06	3.39E−06
	c240	4.85E−06	3.31E−06	3.00E−06
cc-pvtz	c60	1.43E−05	4.70E−06	2.50E−06
	c100	8.27E−06	2.29E−06	1.21E−06
	c180	5.77E−06	2.83E−06	2.00E−06
	c240	5.21E−06	2.61E−06	2.07E−06

Table 2
Maximum absolute deviations of the matrix elements for the semi-numerical HF exchange with different levels of grid.

Basis set	Molecule	SG-1	Standard	Fine
6-31G**	c60	2.74E−04	8.75E−05	5.44E−05
	c100	2.74E−04	6.95E−05	3.06E−05
	c180	2.60E−04	1.54E−04	1.02E−04
	c240	2.33E−04	1.26E−04	9.36E−05
6-311G(2df)	c60	3.95E−04	1.40E−04	9.59E−05
	c100	5.11E−04	1.12E−04	3.36E−05
	c180	3.46E−04	1.52E−04	1.02E−04
	c240	3.55E−04	1.34E−04	9.35E−05
cc-pvdz	c60	2.91E−04	9.14E−05	6.20E−05
	c100	2.76E−04	6.30E−05	2.33E−05
	c180	2.53E−04	1.48E−04	9.11E−05
	c240	2.43E−04	1.32E−04	8.88E−05
cc-pvtz	c60	4.60E−04	1.36E−04	1.02E−04
	c100	5.37E−04	1.15E−04	5.46E−05
	c180	5.54E−04	1.50E−04	9.36E−05
	c240	4.37E−04	1.73E−04	9.80E−05

Table 3
Total energy differences per atom for the semi-numerical HF exchange in micro-Hartree with different levels of grid.

Basis set	Molecule	SG-1	Standard	Fine
6-31G**	c60	6.70	16.76	7.02
	c100	48.56	29.04	28.52
	c180	146.02	158.64	154.03
	c240	147.85	160.69	155.85
6-311G(2df)	c60	5.56	24.30	15.45
	c100	45.91	17.63	15.40
	c180	158.11	181.10	174.59
	c240	180.56	203.25	198.20
cc-pvdz	c60	5.45	15.46	6.60
	c100	31.55	13.25	12.46
	c180	100.34	117.23	111.81
	c240	117.62	128.33	122.66
cc-pvtz	c60	1.95	21.27	18.95
	c100	35.81	0.69	1.42
	c180	63.15	140.23	127.88
	c240	153.38	187.83	167.07

The energy error per atom also grows as the system size become larger, as shown in Table 3. This can be understood by the formula of the semi-numerical scheme for the HF exchange matrix:

$$\frac{\partial E_\sigma^{ex}}{\partial P_{\mu\nu}^\sigma} = \sum_r^{grids} w(r)\varphi_\mu(r)\sum_{\lambda\eta} P_{\lambda\eta}^\sigma \varphi_\lambda(r)\int dr' \frac{\varphi_\nu(r')\varphi_\eta(r')}{|r-r'|}. \quad (6)$$

The error in the HF exchange matrix calculation mainly comes from the numerical computation of the basis function pair $\varphi_\mu(r)\varphi_\lambda(r)$. Because the HF exchange is a long-range interaction, the ESP originated from the point r' may still effectively interact with the numerical basis function pair to produce a numerically significant contribution to the HF exchange matrix even though the distance between the r and r' is large. As the molecule grows larger, it can be expected that the number of ESP integrals grows accordingly, resulting in a multiplier effect for the error on the HF exchange matrix calculation. On the other hand, as the distance between r and r' increases the first order reduced density matrix $\rho(r,r')$ actually decreases [42,43]. For the HF exchange energy expressed in first order reduced density matrix form

$$E_\sigma^{ex}(r) = \sum_r^{grids} w(r)\int dr' \frac{\rho_\sigma(r,r')\rho_\sigma(r,r')}{|r-r'|}, \quad (7)$$

One can see that the decrease of the $\rho(r,r')$ makes the contribution from the r' become smaller, which damps the contribution from basis function pairs at large distances. As the system becomes larger these

two factors cancel each other out to an extent, one may expect that the increase of the error on the energy becomes slower. Such phenomenon can be observed in Table 3.

The damping effect of the density matrix can be more pronounced for a stretched system. To test this effect, we performed the HF SCF calculation on a series of alpha helix alanine peptides with a linear tertiary structure. The alanine series include peptides with 35, 40 and 45 alanines, respectively, and the calculations were performed with the SG-1 grid and Pople basis 6-311G(2df,2pd) [44]. All of the other settings are the same as for the fullerene series. The total energy difference per atom is shown in Table 4. It is evident from the table that as the system is extended sufficiently the total energy difference per atom becomes stable. The reason is that the large end-to-end distance results in many effectively zero elements in the distance-decaying reduced first order density matrix $\rho(r,r')$.

5. Efficiency assessment for semi-numerical scheme

The computational performance of the semi-numerical scheme for HF exchange implemented and optimized for the Phi processor is also benchmarked with the fullerene series. The Phi processor for the test is a KNL7250 with 68 cores and each core is at 1.40 GHz. The computing time on the Phi processor is compared with the computing time on a six-core E5-1650 CPU at 3.2 GHz. The code run on the CPU was from the earlier implementation of the semi-numerical scheme optimized for CPUs. The reference calculations with the analytical four-center ERIs were performed on the CPU only. Table 5 lists the processor timing data for the formation of HF exchange matrix for one SCF iteration. The numerical grid for the semi-numerical scheme is SG-1. The number listed in the parenthesis next to a CPU time is the ratio of the CPU time to the processing time on the Phi processor. All the calculations on the CPU were with multi-threads.

Judging the performance of an implementation on the Phi Processor needs a reference. Intel has provided a series of showcases for the KNL7250 processor.[1] Top performers are able to achieve 1.0–1.5 times of speedup on KNL7250 processor compared with the 36-core dual E5-2697(each core is at 2.3 GHz) CPU system. We estimate this CPU is about 4.3 times faster than the E5-1650 CPU used in our test $\left(\frac{2.3\times36}{3.2\times6}\approx 4.3\right)$. Therefore we consider a 4–6 time speedup to be good in our test.

As shown in Table 5 our implementation of the semi-numerical HF exchange for the Phi processor is able to achieve approximately 3–7 times of speedup relative to the implementation for the CPU. This indicates the success of our implementation. Additionally it obtains 7–12 times of speedup in comparison with the traditional analytical integral calculation with large basis sets such as 6-311G(2df) and cc-pvtz. For moderate double-zeta basis sets 6-31G** and cc-pvdz, the speedup ratio is between 1.5 and 4 times. The difference on the speedup between the different basis sets can be explained by the semi-numerical algorithm itself, as it scales quadratic with respect to the basis set size, in contrast to the quadruple scaling with the four-center ERI method [15]. For a given molecule the numerical grid is unchanged for different basis sets, therefore if the "density" of basis functions per molecular size increases, the semi-numerical scheme becomes more efficient per grid point. As a result the semi-numerical scheme is favored for larger basis sets such as 6-311G(2df) and cc-pvtz.

[1] Please refer to the web page https://software.intel.com/en-us/xeon-phi/x200-processor showcase section for more details.

Table 4
Total energy error per atom for the semi-numerical HF exchange calculation on alanine peptides in micro-Hartree.

	ala35	ala40	ala45
Number of atoms	353	403	453
$\mu E_h/atom$	16.61	16.66	13.58

Table 5
Timing of HF exchange matrix calculation with the SG-1 grid on different platforms in seconds.

Basis set	Molecule	Basis set number	Semi-numerical		4-center ERI
			Phi	CPU	CPU
6-31G**	c60	900	134.03	135.56 (1.01)	60.05 (0.45)
	c100	1500	227.31	419.78 (1.85)	176.76 (0.78)
	c180	2700	519.99	1375.07 (2.64)	552.59 (1.06)
	c240	3600	611.35	2455.31 (4.02)	983.19 (1.61)
6-311G(2df)	c60	1800	150.94	396.78 (2.63)	1080.45 (7.16)
	c100	3000	340.36	1241.43 (3.65)	3182.73 (9.35)
	c180	5400	1175.69	4141.91 (3.52)	9994.56 (8.50)
	c240	7200	2120.03	7333.96 (3.46)	18044.39(8.51)
cc-pvdz	c60	840	136.64	323.32 (2.37)	193.60 (1.42)
	c100	1400	242.91	1015.24 (4.18)	585.88 (2.41)
	c180	2520	527.54	3353.91 (6.36)	1897.24 (3.60)
	c240	3360	862.89	5946.02 (6.89)	3403.45 (3.94)
cc-pvtz	c60	1800	182.06	770.52 (4.23)	1874.04 (10.29)
	c100	3000	476.20	2505.30 (5.26)	5794.53 (12.17)
	c180	5400	1659.66	8516.49 (5.13)	18600.08(11.21)
	c240	7200	2970.48	15175.24(5.11)	33220.47(11.18)

6. Conclusion

Efficient implementation of quantum chemistry methods to take the full advantage of new computer technologies is important for the large scale applications. In this work we extended our previous implementation of a semi-numerical method for HF exchange to the Intel Phi processor. The utilization of VPUs on the Phi processor, which operates in a SIMD fashion, imposes a major challenge for implementation since algorithms for the ERI naturally fits the MIMD-type. The other major technical obstacle is the relatively small cache size. These difficulties are overcome by organizing the data for computing various types of fundamental integrals to be aligned with the VPU register, and prefetching necessary data for effective use of the cache. The accuracy and efficiency of the implementation are evaluated with the computation of a series of buckyball molecules with basis sets of different sizes. The results showed that the accuracy of the semi-numerical implementation is similar to that of the RI method. The computational efficiency of the implementation matches the achievements of other types of software for the Phi processor. The advantage of the combination of the semi-numerical algorithm with the Phi processor can be seen through the 7–12 times of speedup relative to the computation with the traditional four-center ERI method on a six-core E5-1650 CPU with 6-311G(2df) and cc-pvtz basis sets.

Acknowledgement

This project is supported by Petroleum Research Funds - USA (PRF 55229-DNI6) and National Science Foundation - USA (CHE-1665344). F. Liu is grateful to the Colfax International and Intel Remote Access Program for free access to the Intel Phi processors. The authors thank Dr. Peter Pulay and Dr. Jon Baker for letting us use their pruned DFT grids.

References

[1] R. Rahman, Intel Xeon Phi Coprocessor Architecture and Tools: The Guide for Application Developers, Apress, 2013.
[2] J. Jeffers, J. Reinders, A. Sodani, Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition, Morgan Kaufmann, 2016.
[3] K. Yasuda, Two-electron integral evaluation on the graphics processor unit, J. Comput. Chem. 29 (3) (2008) 334–342.
[4] I.S. Ufimtsev, T.J. Martinez, Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation, J. Chem. Theory Comput. 4 (2) (2008) 222–231.
[5] I.S. Ufimtsev, T.J. Martinez, Quantum chemistry on graphical processing units. 2. Direct self-consistent-field implementation, J. Chem. Theory Comput. 5 (4) (2009) 1004–1015.
[6] N. Luehr, I.S. Ufimtsev, T.J. Martinez, Dynamic precision for electron repulsion integral evaluation on graphical processing units (gpus), J. Chem. Theory Comput. 7 (4) (2011) 949–954.
[7] S.S. Leang, A.P. Rendell, M.S. Gordon, Quantum chemical calculations using accelerators: Migrating matrix operations to the nvidia kepler gpu and the intel xeon phi, J. Chem. Theory Comput. 10 (3) (2014) 908–912.
[8] E. Apra, M. Klemm, K. Kowalski, Efficient implementation of many-body quantum chemical methods on the intel xeon phi coprocessor, In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press, 2014, pp. 674–684.
[9] F. Reida, I. Bethunea, Optimising cp2k for the intel xeon phi, Partnership for Advanced Computing in Europe (PRACE) 140.
[10] M. Barth, K. Sweden, M. Byckling, C. Finland, N. Ilieva, N. Bulgaria, S. Saarinen, M. Schliephake, V. Weinberg, L. Germany, Best practice guide intel xeon phi v1, LRZ Germany March 31.
[11] X. Tian, et al., Effective simd vectorization for intel xeon phi coprocessors, Sci. Programm. 2015 (2015) 1.
[12] L. Fusti-Molnar, J. Kong, Fast and accurate coulomb calculation with gaussian functions, J. Chem. Phys. 122 (7) (2005) 074108.
[13] C.-M. Chang, Y. Shao, J. Kong, Ewald mesh method for quantum mechanical calculations, J. Chem. Phys. 136 (11) (2012) 114112.
[14] C.-M. Chang, N.J. Russ, J. Kong, Efficient and accurate numerical integration of exchange-correlation density functionals, Phys. Rev. A 84 (2) (2011) 022504.
[15] F. Liu, J. Kong, Efficient computation of exchange energy density with gaussian basis functions, J. Chem. Theory Comput. 13 (6) (2017) 2571–2580.
[16] F. Neese, F. Wennmohs, A. Hansen, U. Becker, Efficient, approximate and parallel hartree-fock and hybrid dft calculations. a 'chain-of-spheres' algorithm for the Hartree-Fock exchange, Chem. Phys. 356 (1-3) (2009) 98–109.
[17] H. Bahmann, M. Kaupp, Efficient self-consistent implementation of local hybrid functionals, J. Chem. Theory Comput. 11 (4) (2015) 1540–1548.
[18] R.A. Friesner, Solution of self-consistent field electronic structure equations by a pseudospectral method, Chem. Phys. Lett. 116 (1) (1985) 39–43.
[19] R.A. Friesner, An automatic grid generation scheme for pseudospectral self-consistent field calculations on polyatomic molecules, J. Phys. Chem. 92 (11) (1988) 3091–3096.
[20] A.D. Becke, Real-space post-Hartree-Fock correlation models, J. Chem. Phys. 122 (6) (2005) 064101.
[21] J.P. Perdew, V.N. Staroverov, J. Tao, G.E. Scuseria, Density functional with full exact exchange, balanced nonlocality of correlation, and constraint satisfaction, Phys. Rev. A 78 (5) (2008) 052513.
[22] F. Liu, E. Proynov, J.-G. Yu, T.R. Furlani, J. Kong, Comparison of the performance of exact-exchange-based density functional methods, J. Chem. Phys. 137 (11) (2012) 114104.
[23] A.D. Becke, Density functionals for static, dynamical, and strong correlation, J. Chem. Phys. 138 (7) (2013) 074109.
[24] J. Kong, E. Proynov, Density functional model for nondynamic and strong correlation, J. Chem. Theory Comput. 12 (1) (2016) 133–143.
[25] A.D. Becke, A multicenter numerical integration scheme for polyatomic molecules, J. Chem. Phys. 88 (4) (1988) 2547–2553.
[26] C.W. Murray, N.C. Handy, G.J. Laming, Quadrature schemes for integrals of density functional theory, Mol. Phys. 78 (997–1014) (1993) 997.
[27] V. Lebedev, Quadratures on a sphere, USSR Comput. Math. Math. Phys. 16 (2) (1976) 10–24.
[28] F. Liu, T. Furlani, J. Kong, Optimal path search for recurrence relation in cartesian gaussian integrals, J. Phys. Chem. A 120 (51) (2016) 10264–10272.
[29] S. Obara, A. Saika, Efficient recursive computation of molecular integrals over cartesian gaussian functions, J. Chem. Phys. 84 (1986) 3963.
[30] S. Obara, A. Saika, General recurrence formulas for molecular integrals over cartesian gaussian functions, J. Chem. Phys. 89 (1988) 1540.
[31] M. Head-Gordon, J. Pople, A method for two-electron gaussian integral and integral derivativeevaluation using recurrence relations, J. Chem. Phys. 89 (1988) 5777.
[32] M. Deilmann, et al., A guide to vectorization with Intel C++ compilers, Intel Corporation.
[33] P. Gill, Molecular integrals over gaussian basis functions, Adv. Quant. Chem. 25 (1994) 141–205.
[34] J. Jeffers, J. Reinders, Intel Xeon Phi coprocessor high performance programming, Newnes, 2013.
[35] L. McMurchie, E. Davidson, One-and two-electron integrals over cartesian gaussian functions, J. Comput. Phys. 26 (2) (1978) 218–231.
[36] H. King, M. Dupuis, Numerical integration using rys polynomials, J. Comput. Phys. 21 (2) (1976) 144–165.
[37] M. Dupuis, J. Rys, H. King, Evaluation of molecular integrals over gaussian basis functions, J. Chem. Phys. 65 (1976) 111.
[38] J. Rys, M. Dupuis, H. King, Computation of electron repulsion integrals using the rys quadrature method, J. Comput. Chem. 4 (2) (1983) 154–157.
[39] P.M.W. Gill, B.G. Johnson, J.A. Pople, A standard grid for density functional calculations, Chem. Phys. Lett. 209 (5-6) (1993) 506–512.
[40] J. Baker, T. Janowski, K. Wolinski, P. Pulay, Recent developments in the pqs program, WIREs Comput. Mol. Sci 2 (1) (2012) 63–72.
[41] F. Weigend, A fully direct ri-hf algorithm: implementation, optimised auxiliary basis sets, demonstration of accuracy and efficiency, PCCP 4 (18) (2002) 4285–4291.
[42] W. Kohn, Density functional and density matrix method scaling linearly with the number of atoms, Phys. Rev. Lett. 76 (1996) 3168–3171.
[43] R. Baer, M. Head-Gordon, Sparsity of the density matrix in kohn-sham density functional theory and an assessment of linear system-size scaling methods, Phys. Rev. Lett. 79 (1997) 3962–3965.
[44] A.D. McLean, G.S. Chandler, Contracted gaussian basis sets for molecular calculations. i. Second row atoms, z=11-18, J. Chem. Phys. 72 (10) (1980) 5639–5648.