All-electron density functional theory on Intel MIC: Elk

W. Scott Thornton, R.J. Harrison

Abstract

We present the results of the porting of the full potential linear augmented plane-wave solver, Elk [1], to the Intel Many-Integrated Core (MIC) architecture. The port was restricted to Elk running in native mode only. We include results of performance of the most-time intensive sections of Elk, the self-consistent loop.

1 Introduction

Moore's law, embodied in the International Technology Roadmap for Semiconductors, describes how the number of transistors per commodity silicon chip doubles over a period of two years, perhaps soon slowing to every three years. How best to employ these transistors is a subject of much research and debate, and is of course colored by metrics for efficiency (speed, energy, programmer productivity), application characteristics, programming models, and architectural history. However, what is clear is that over the next decade computers will massively increase in the amount of fine and medium grain parallelism. Fortunately, this trend for once coincides with the needs of scientific applications. With the scale of massively parallel supercomputers reaching O(1M) nodes, many scientific applications have exhausted all readily available coarse grain parallelism. Moreover, with clock speeds stagnating and the historical trend to increased performance only being coarse grain parallelism some scientific applications (e.g., molecular dynamics simulations of small systems) have seen little performance increase over the past few years. However, most applications have significant amounts of untapped fine and medium grain parallelism and we hence expect simulation in science engineering to benefit greatly over the coming decade, especially as other technologies address other architectural deficiencies including the memory barrier. The above optimism motivated by trends in computer architecture is countered by great pessimism about the complexity of computer software (leading to a high entry barrier and low productivity), hence it is natural and indeed essential to ask not just how efficiently a given architecture executes a specific code or algorithm, but the level of effort required to attain that performance. The Intel Many Integrated Core (MIC) architecture, comprising many simple x86 processors with extended vector units, presents a compelling alternative to mainstream multi-core, and the general purpose graphical processing units (GPGPUs) typified by the products of NVIDIA and AMD. Each represents a different answer to the question of how to use the increasing number of transistors motivated by different application spaces and performance considerations. The GPGPU processors promise huge performance gains compared to mainstream multicore (10-20x for current technologies considering peak floating point speeds; less if considering memory bandwidth) and for well suited applications (massive data concurrency with modest working set size) also promise high productivity programming. However, for the majority of applications with either large data sets or irregular computation there is a substantial amount of programmer effort that is necessary to attain good

performance, if that is even possible. The issues are primarily how to distribute control logic, efficient execution of irregular and small parallel kernels, and compatibility with familiar and mainstream programming models. Although the massive data parallelism model of GPGPUs is expected to be just as effective and efficient on MIC, the most natural and powerful model is actually the parallel scalar/vector model long ago established as highly successful on Cray vector supercomputers.

In this paper, we examine one well-established application in materials science to understand how readily it maps to the MIC architecture and programming model, and start to inquire how well it performs in parallel and in its use of the vector units.

2 Kohn-Sham equations

The Kohn-Sham (KS) scheme is described by the mapping of the minimization of the electron density energy functional,

$$E[\rho] = T[\rho] + U[\rho] + \int V(\mathbf{r})\rho(\mathbf{r})d\mathbf{r}, \qquad (1)$$

of the full interacting system onto an auxillary system, known as the Kohn-Sham system, where the principle consituents are the KS-orbitals, $\phi_i(\mathbf{r})$. The Kohn-Sham orbitals are fictitious, single-particle orbitals which only interact with each other via a local, effective potential, $V_{KS}(\mathbf{r})$. The key point in the construction of the auxillary system is that the electronic density, $\rho(\mathbf{r})$, of the auxillary non-interacting system, i.e.

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} |\phi_i(\mathbf{r})|^2, \qquad (2)$$

is identical to that of the interacting system.

The Kohn-Sham equation describes the relationship of the KS-orbitals to the effective potential, $V_{KS}(\mathbf{r})$, i.e.

$$\left(-\frac{1}{2}\nabla^2 + V_{KS}[\rho](\mathbf{r})\right)\phi_i(\mathbf{r}) = \epsilon_i\phi_i(\mathbf{r}). \qquad (3)$$

The effective Kohn-Sham potential, $V_{KS}[\rho](\mathbf{r})$, is a functional of the electronic density, $\rho(\mathbf{r})$ such that

$$V_{KS}(\mathbf{r}) = V_N(\mathbf{r}) + V_C[\rho](\mathbf{r}) + V_{XC}[\rho](\mathbf{r}). \qquad (4)$$

The quantities on the RHS of eq. (4) are the ion-electron attraction ($V_N(\mathbf{r})$), the classical electrostatic repulsive potential between electrons, ($V_C(\mathbf{r})$), and the exchange-correlation functional ($V_{XC}(\mathbf{r})$) which, in principle, includes all of the static many-body exchange and correlation effects.

Equations (2), (3), and (4) constitute a set of nonlinear partial differential equations (PDE), and thus, must be iterated until a condition of self-consistency has been reached.

3 Basis sets for crystalline systems

In most DFT solvers, equations (2), (3), and (4) are iterated numerically using a finite set of basis functions such that the set of PDE's are transform into a set of matrix equations. Indeed, by expanding the the orbitals, $\phi_i(\mathbf{r})$, into a finite set of basis functions, i.e.

$$\phi_i(\mathbf{r}) = \sum_{\alpha} c_{i\alpha}\chi_\alpha(\mathbf{r}), \qquad (5)$$

eq. (3) can be transformed into the matrix equation

$$\sum_{\beta} H_{\alpha\beta}c_{i\beta} = \epsilon_i \sum_{\beta} S_{\alpha\beta}c_{i\beta}, \qquad (6)$$

where $H_{\alpha\beta}$ and $S_{\alpha\beta}$ are known as the Hamiltonian and the overlap matrices, respectively.

In crystalline systems, this set of basis functions needs to be constructed in such a way as to efficiently handle the boundary conditions of a crystalline material. The boundary conditions are summarized by (1) the Bloch condition and (2) the Born-von Karman boundary conditions.

The crystalline system can be defined as a group of atoms which inhabit a localized region of space in such a way where this region of space is replicated at regular intervals via an integral number of lattice translations. This localized region of space is defined in terms of three *lattice* vectors: $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$ and is known as the *unit cell*. The crystal is defined such that an integral number of lattice translation via $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$ will result in an identical environment. This defines the innermost portion of the crystal known as the *bulk*. The majority of solid-state calculations are performed within the bulk portion of the crystal.

The boundary of the crystal is treated by imposing periodic boundary conditions then allowing the number of cells in the crystal, N, to go to infinity. This application of the Born-von Karman boundary condition has the effect of imposing a lattice periodicity in quantities such as the electronic density and the KS effective potential, i.e. $\rho(\mathbf{r}+\mathbf{R}) = \rho(\mathbf{r})$ and $V_{KS}(\mathbf{r}+\mathbf{R}) = V_{KS}(\mathbf{r})$ (where $\mathbf{R} = n_1\mathbf{a_1}+n_2\mathbf{a_2}+n_3\mathbf{a_3}$ is a lattice vector). The other effect is the emergence of the Bloch condition. The Bloch condition is characterized by the fact that the KS-orbitals aquire a phase when translated by a lattice vector, i.e. $\phi_{n,\mathbf{k}}(\mathbf{r}+\mathbf{R}) = e^{i\mathbf{k}\cdot\mathbf{R}}\phi_{n,\mathbf{k}}(\mathbf{r})$. As a result the KS-orbitals aquire the analytic form:

$$\phi_{n,\mathbf{k}}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}u_{n,\mathbf{k}}(\mathbf{r}), \tag{7}$$

where $u_{n,\mathbf{k}}(\mathbf{r})$ is periodic with respect to a lattice translation, i.e. $u_{n,\mathbf{k}}(\mathbf{r}+\mathbf{R}) = u_{n,\mathbf{k}}(\mathbf{r})$.

The Bloch orbital given in eq. (7), is associated with two indicies: the band index, n, and wavevector, $\mathbf{k}$. With the application of the Born-von Karman boundary conditions on the crystal, $\mathbf{k}$ is essentially a continuous quantity. However, in almost all cases, $\mathbf{k}$ varies so slowly that it proves computationally efficient to sample $\mathbf{k}$ over a finite grid.

The primary source of parallelism in the electronic structure of crystalline structures, is that the Hamiltonian matrix in eq. (6) has a block diagonal structure where each block is associated with a wavevector $\mathbf{k}$. This means that the construction and the solving of eq. (6) can be done in parallel where each process (or thread) is responsible for a given $\mathbf{k}$ sector.

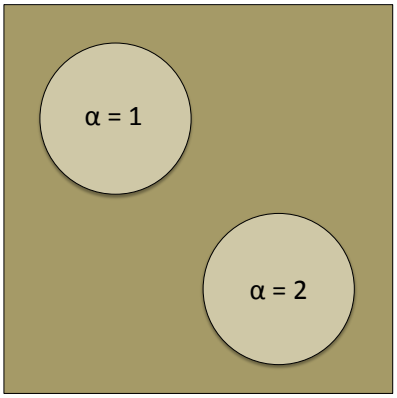

Figure 1: Partioning scheme in LAPW methods.

The basis functions that serve as the foundation of Elk is that of the (linearized) augmented plane waves (APW/LAPW) [2]. The APW basis revolves around the notion, first discovered by Slater [3], that in the vicinity of the atoms, the orbitals and the potential should have almost perfect spherical symmetry; also Slater noted that in the region of space in between the atoms, the orbitals and potential are very smooth. As a consequence, function values in the vicinity of a given atom could be expanded in terms of solutions to the spherically symmetric Schrödinger equation. In the space in between atoms, the function, because of its smoothness, could be adequately represented with plane waves. As shown in figure (1), the APW basis set partitions space into two regions: the first region is the atomic spheres centered at each atom, and the second region is the interstitial region inbetween the atomic spheres. In the atomic sphere region, also called muffin-tin region, a function is expanded in the following basis:

$$\phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r}) = \sum_{\alpha=1}^{N_\alpha}\sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}\sum_{j=1}^{M_l^\alpha} A_{jlm}^\alpha(\mathbf{G}+\mathbf{k})u_{jl}^\alpha(r)Y_{lm}(\hat{\mathbf{r}}), \tag{8}$$

where $u_{jl}^\alpha(r)$ is the j^{th} energy derivative of the solution to the radial Schrödinger equation of the α-th species.

In the interstitial region, the basis is given as a plane wave,

$$\phi_{\mathbf{G}+\mathbf{k}}(\mathbf{r}) = e^{i(\mathbf{G}+\mathbf{k})\cdot\mathbf{r}}. \quad (9)$$

4 Computational scheme

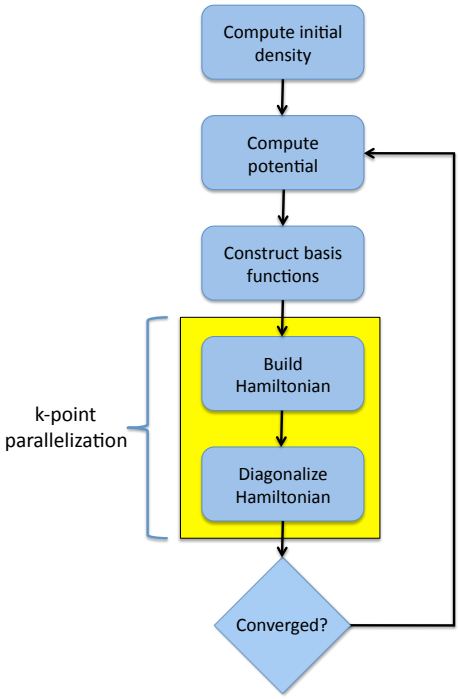

Figure 2: Flowchart of the Kohn-Sham scheme.

Computing the electronic structure with LAPW scheme begins by constructing an intitial guess for the electronic density from the combination of the the atomic densities. From the density, we use eq. (4) to compute the effective Kohn-Sham potential, V_{KS}. This task is the starting point of the self-consistent loop. From the spherically symmetric portion of V_{KS} inside each muffin-tin, we can integrate the radial Schrödinger equation equation at fixed energies to obtain the radial functions, $u^{\alpha}_{jl}(r)$. From there, we can apply the appropriate boundary conditions on the surface of the muffin-tins, and then construct the LAPW basis functions in (8) and (9).

At this point, since the construction and diagonalization of eq. (6) are independent for each $\mathbf{k}$ wavevector, we begin the primary parallel section of the algorithm. For each $\mathbf{k}$-point, we then construct the Hamiltonian and overlap matrices using the basis functions (8) and (9). To obtain the orbital solutions to eq. (3), we diagonalize using the standard routines from LAPACK. This parallelism in the algorithm is expresses with OpenMP compiler directives where the eigenvalues, ϵ_i, and the eigenvectors, $c_{i\beta}$ have been kept thread private. In addition to the parallelism achieved though $\mathbf{k}$ vector independence, there is additional parallelism in the construction of $H^{\mathbf{k}}_{\alpha\beta}$ via different atoms and atomic species. These loops have also been parallelized using OpenMP directives.

5 Porting and testing of Elk on Intel MIC

The goal of this project was to demonstrate the ease of porting an existing scientific application to the Intel MIC platform. More specifically, the initial goal was to show that existing source code could be ported to the MIC to be executed natively without any code changes. This is exactly what happened. The only changes that were needed were to the Makefile. Using the Intel MIC version of *ifort*, we merely added the *-mmic* option to *ifort*.

The performance of Elk was tested by running a calculation of crystalling silicon with differing numbers of threads. Initially, a $3 \times 3 \times 3$ $\mathbf{k}$-point grid was employed; we upgraded to a $4 \times 4 \times 4$ grid later. We show both the actual and ideal speedup for the self-consistent iteration portion of the code using a $3 \times 3 \times 3$ $\mathbf{k}$-point grid and the $4 \times 4 \times 4$ $\mathbf{k}$-point grid (figures (3) and (4), respectively). For the case with

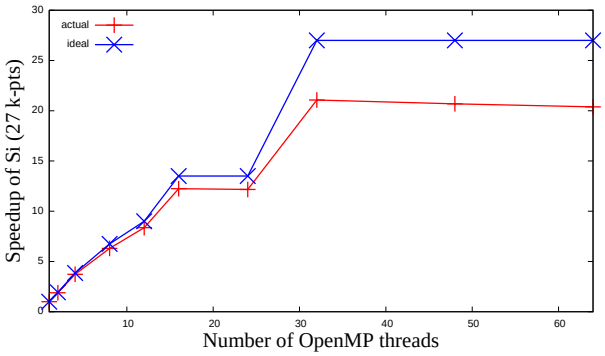

Figure 3: Speedup of silicon with 27 **k** points.

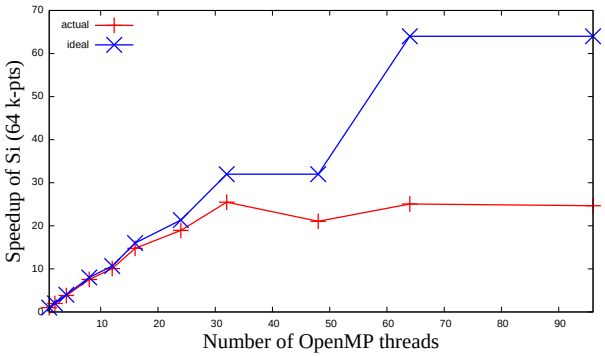

Figure 4: Speedup of silicon with 64 **k** points.

ideal speedup, we used the following model:

$$S = \frac{T_s + N_w T_p}{T_s + \text{ceil}(N_w/N_t)T_p}, \tag{10}$$

where T_s is the total sequential time. T_p is the time needed to execute one parallelizable unit of work by a single thread. N_w is the parallelizable integral number of units of work which in our case is the total number of **k**-points. For the case where we use a $3\times3\times3$ **k**-point grid, N_w is 27; for the $4\times4\times4$ grid, N_w is 64. N_t is the number of OpenMP threads. For the ideal case, we set the sequential time to be zero, i.e. $T_s = 0$.

In (3), we show good agreement between the ideal and actual speedup until we get to 24 threads. At the 32 thread point, we note that there is a sharp rise in both the actual and ideal speedup but with a more significant discrepancy. We attribute this discrepancy to the fact that we have set the sequential time, T_s, to zero in our model. In the case of the $4\times4\times4$ **k** grid in figure (4), we note that beyond 32 threads, the model speedup and the actual speedup diverge significantly. We speculate that the cause of this divergence can be attributed to the different threads competing for processor resources, particularly the floating point unit. However, for a more conclusive diagnosis, we plan to run more tests in the future.

6 Conclusions

We have ported the all electron density functional theory solver, Elk, to the Intel Many Integrated Core architecture to be executed in the native mode. We have execute the Elk solver using crystalline silicon as a benchmark with two different **k** grids: a $3\times3\times3$ **k** grid and a $4\times4\times4$ **k** grid. For the $3\times3\times3$ mesh, we show good agreement with the speedup model given in eq. (10); any discrepancy can be attributed to setting the sequential time to zero. For the $4\times4\times4$ **k** grid, we show good agreement with the model, but beyond 32 threads the actual speedup flattens. It is speculated that the divergence between the ac-

tual and the model speedup is caused by competition amongst the threads for the floating-point unit.

References

[1] http://elk.sourceforge.net/

[2] David J. Singh and Lars Nordstrom, *Planewaves, Pseudopotentials, and the LAPW method* Springer Science-Business Media, Inc., New York, 2nd Edition, 2006.

[3] J.C. Slater, Phys. Rev 51, 846 (1937).