Deploying Elbrus VLIW CPU ecosystem for materials science calculations: performance and problems

Vladimir Stegailov[1,2,3] and Alexey Timofeev[1,2,3]

[1] Joint Institute for High Temperatures of the Russian Academy of Sciences
[2] National Research University Higher School of Economics
[3] Moscow Institute of Physics and Technology (State University)

timofeevalvl@gmail.com

Abstract. Modern Elbrus-4S and Elbrus-8S processors show floating point performance comparable to the popular Intel processors in the field of high-performance computing. Tasks oriented to take advantage of the VLIW architecture show even greater efficiency on Elbrus processors. In this paper the efficiency of the most popular materials science codes in the field of classical molecular dynamics and quantum-mechanical calculations is considered. A comparative analysis of the performance of these codes on Elbrus processor and other modern processors is carried out.

Keywords: Elbrus architecture, VASP, LAMMPS, FFT.

1 Introduction

A large part of HPC resources installed during the last decade is based on Intel CPUs. However, the situation is gradually changing. In March 2017, AMD released the first processors based on the novel x86_64 architecture called Zen. In November 2017, Cavium has presented server grade 64-bit ThunderX2 ARMv8 CPUs that are to be deployed in new Cray supercomputers. The Elbrus microprocessors stand among emerging types of high performance CPU architectures [1, 2].

The diversity of CPU types complicates significantly the choice of the best variant for a particular HPC system. The main criterion is certainly the time-to-solution of a given computational task or a set of different tasks that represents an envisaged workload of a system under development.

Computational materials science provides an essential part of the deployment time for high performance computing (HPC) resources worldwide. The VASP code [3–6] is among the most popular programs for electronic structure calculations that gives the possibility to calculate materials properties using the non-empirical (so called *ab initio*) methods. Ab initio calculation methods based on quantum mechanics are important modern scientific tools (e.g., see [7–11]). According to the recent estimates, VASP alone consumes up to 15-20 percent of the world's supercomputing power [12, 13]. Such unprecedented popularity

justifies the special attention to the optimization of VASP for both existing and novel computer architectures (e.g. see [14]).

Significant part of calculation time of such software packages for computational materials science is the execution time of the Fourier transform. One of the most time consuming components in VASP is 3D-FFT [15]. FFT libraries are tested on the Elbrus processor in order to determine the most optimal tool for performing a fast Fourier transform. The EML (Elbrus Multimedia Library), developed by Elbrus processor manufacturer, and the most popular FFTW library are under consideration.

In this work we present the efficiency analysis of Elbrus CPUs in comparison with Intel Xeon Haswell CPUs using a typical VASP workload example. The results of the FFT libraries testing on Elbrus processors are presented.

2 Related Work

HPC systems are notorious for operating at a small fraction of their peak performance and the deployment of multi-core and multi-socket compute nodes further complicates performance optimization. Many attempts have been made to develop a more or less universal framework for algorithms optimization that takes into account essential properties of the hardware (see e.g. [16, 17]). The recent work of Stanisic et al. [18] emphasizes many pitfalls encountered while trying to characterize both the network and the memory performance of modern machines.

A fast Fourier transform is used in computational modeling programs for calculations related to quantum computations, Coulomb systems, etc. and takes a very substantial part of the program's running time [19], especially for VASP [15]. The detailed optimization of the computation of 3D-FFT in VASP in order to prepare the code for an efficient execution on multi- and many-core CPUs like Intels Xeon Phi is considered in the article [15]. In this article the threading performance of widely used FFTW (Cray LibSci) and Intels MKL on a current Cray-XC40 with Intel Haswell CPUs and a Cray-XC30 Xeon Phi (Knights Corner, KNC) system is evaluated.

At the moment, Elbrus processors are ready for use [1, 2], so we decided to benchmark them using one of the main HPC tools used for material science studies (VASP) and the library that determines the performance of this code (FFT). The architecture of the Elbrus processors [1, 2] allows us to expect that the butterfly cross-linking during the execution of the FFT performs during a smaller number of cycles than for CPUs like Intels Xeon Phi.

3 Methods and Software Implementation

3.1 Test Model in VASP

VASP 5.4.1 is compiled for Intel systems using Intel Fortran, Intel MPI and linked with Intel MKL for BLAS, LAPACK and FFT calls. For the Elbrus-8S

system, lfortran compatible with gfortran ver.4.8 is used together with MPICH, EML BLAS, Netlib LAPACK and FFTW libraries.

Our test model in VASP represents the liquid Si system consisting of 48 atoms in the supercell. The Perdew-Burke-Ernzerhof model for xc-functional is used. The calculation protocol corresponds to molecular dynamics. We use the time for the first iteration of electron density optimization τ_{iter} as a target parameter of the performance metric.

The τ_{iter} values considered in this work are about 5-50 sec and correspond to performance of a single CPU. At the first glance, these are not very long times to be accelerated. However, *ab initio* molecular dynamics requires usually $10^4 - 10^5$ time steps and larger system sizes. That is why the decrease of τ_{iter} by several orders of magnitude is an actual problem for modern HPC systems targeted at materials science computing.

The choice of a particular test model has a certain influence on the benchmarking results. However, our preliminary tests of other VASP models show that the main conclusions of this study do not depend significantly on a particular model.

3.2 Fast Fourier Transform

FFTW 3.3.6 is compiled using lcc, the analogue of gcc for Elbrus-4S and Elbrus-8S systems. As an input array for the Fourier transforms, a sinusoidal signal, white, pink and brown noise are used. In this article the results are presented for white noise.

The usual pattern when calling FFT (or MKL through its FFTW interface) is as follows:

1. Preparation stage: create plans for FFT computations, e.g., for FFTW via fftw_plan p=fftw_plan_dft(..), for EML via eml_Signal_FFTInit(...).
2. Execution stage: perform FFT computation using that plan, e.g., for FFTW via fftw_execute_dft(p,in,out), for EML via eml_Signal_FFTFwd(...).
3. Clean up.

We consider the work of the first two stages, since they are the most time consuming. Preparation takes the main time when you start the Fourier transform once for a fixed size of the input array. When the Fourier transform is repeatedly started, the running time of the program can determine the execution time of the Fourier transform itself.

So for these two stages we compare libraries FFTW and EML on processors Elbrus-4S and Elbrus 8S. The library EML has fewer useful functions than in the library FFTW for now. In particular, the size of the input array can only be a power of two, so the preparation stage has to be partially implemented by users. The number of functions in the library EML is much smaller than in the library FFTW.

Plan creation with FFTW can happen with differently expensive planner schemes: FFTW_ESTIMATE (cheap), FFTW_MEASURE (expensive),

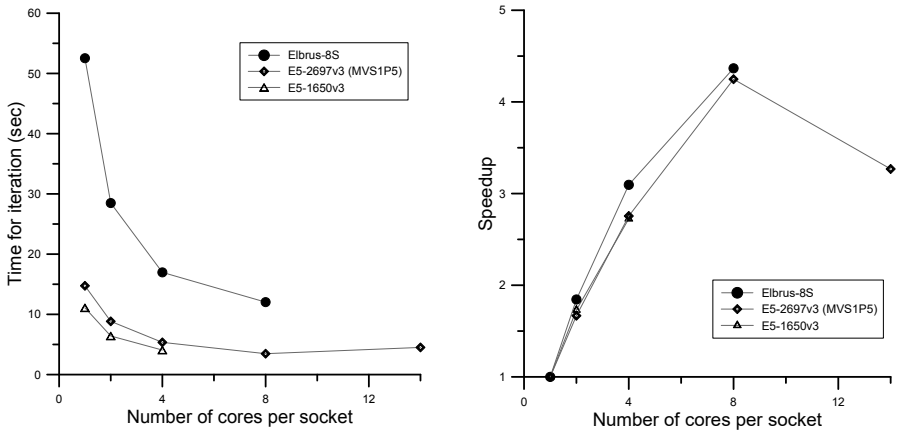
FFTW_PATIENT (more expensive) and FFTW_EXHAUSTIVE (most expensive). Except for FFTW_ESTIMATE plan creation involves testing different FFT algorithms together with runtime measurements to achieve best performance on the target platform. On servers with processors Elbrus-4S and Elbrus-8S due to the lack of libraries authors managed to compile FFTW only for use in the mode FFTW_ESTIMATE, in which the preparation time is short and the execution time is long.

To average the resulting values of the operating time and to obtain a spread of results, the calculations were repeated from 30 to 1000 times. The spread of the results was within 1%, and sometimes did not exceed 0.001%.

4 Results and Discussion

4.1 VASP benchmark on Elbrus-8S and Xeon Haswell CPUs

VASP is known to be both a memory-bound and a compute-bound code [14]. Figure 1a shows the results of the liquid Si model test runs. Benchmarks with Intel Haswell CPUs are performed with RAM working at 2133 MHz frequency and C612 chipset.



a) b)

Fig. 1. The dependence of the (a) time and (b) speedup of the first iteration of the liquid Si model test on the number of cores per socket

Figure 1b presents the same data as shown on Figure 1a but in the coordinates of speedup and number of cores. The considered Intel processors and processor Elbrus show a similar dynamics of speedup. The Elbrus processor shows slightly bigger speedup. It is seen that for the E5-2697v3 processor the optimal number of cores is between 4 and 14, approximately 8. The form of the dependence suggests that a greater speedup value can be achieved for the Elbrus processor.

4.2 Fast Fourier Transform on Elbrus CPUs: EML vs FFTW

We divide the process the Fourier transformation into two stages: the preparation of the algorithm (figures 2 - 5), and the execution of the transformation (figures 6 - 8). Preparation takes the main time when you start the Fourier transform once. The algorithm execution time can determine the total running time of Fourier transform for the situations when the Fourier transform is started many times for a fixed size of the input array.

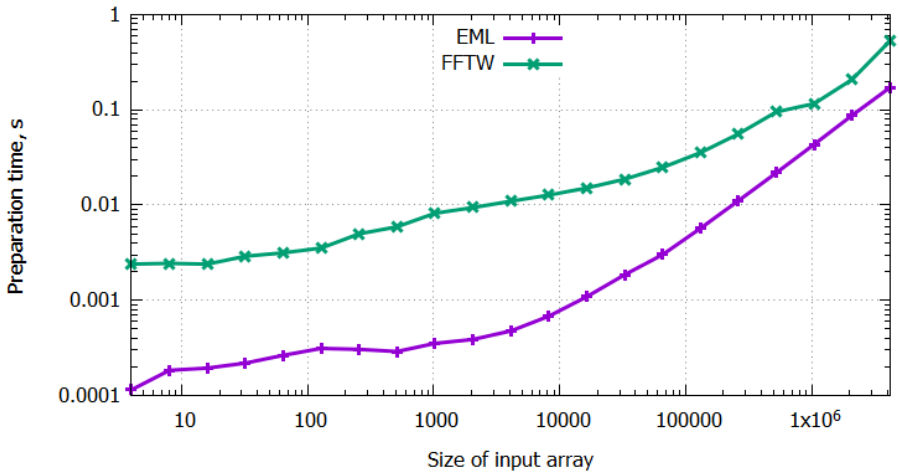


Fig. 2. The dependence of the time for FFT preparation on size of input array for Elbrus-4S

The preparation time of the algorithm FFT for Elbrus-4S appears to be an order of magnitude smaller using EML library in comparison with FFTW for array size smaller than 2^{15} (figures 2, 3). And for large sizes of the array preparation time using EML is only 2-3 smaller than using FFTW. All points have an error less than 1%.

Figures 4 and 5 show that for the Elbrus-8S, the difference in preparation time is even greater. For arrays smaller than 2^{15} the preparation time using EML is 10-20 times less than using FFTW. And for large arrays up to 2^{17} the preparation time using EML is 50-90 times less than the one using FFTW.

Thus, we can make an interim summary, that single launches of the FFT on Elbrus-4S and Elbrus-8S are more efficient using the EML library, because the preparation of the algorithm FFT using EML is 2-20 times for Elbrus-4S and 10-90 times faster for Elbrus-8S than the ones using FFTW.

And here we consider the second stage of FFT implementation, namely, the execution of the algorithm. The stage of execution of the algorithm takes from one to several orders of magnitude less time than the stage of the algorithm

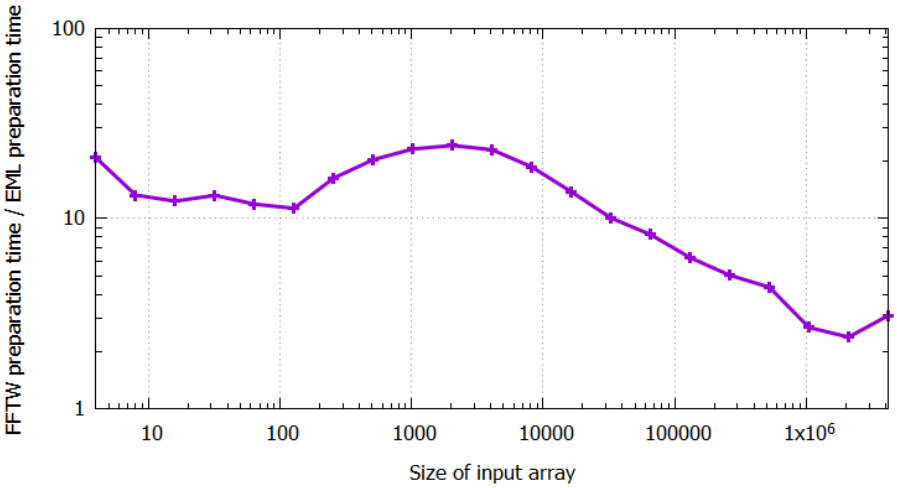


Fig. 3. The dependence of the ratio of FFT preparation time using EML and using FFTW on size of input array for Elbrus-4S

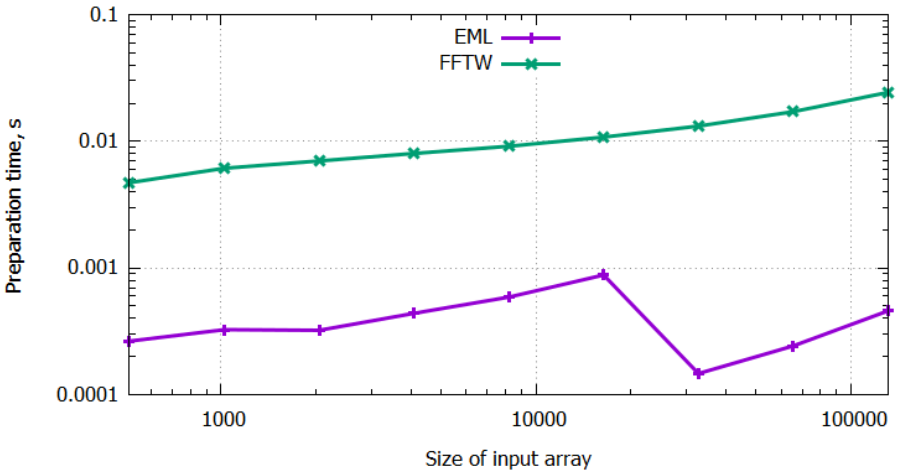


Fig. 4. The dependence of the time for FFT preparation on size of input array for Elbrus-8S

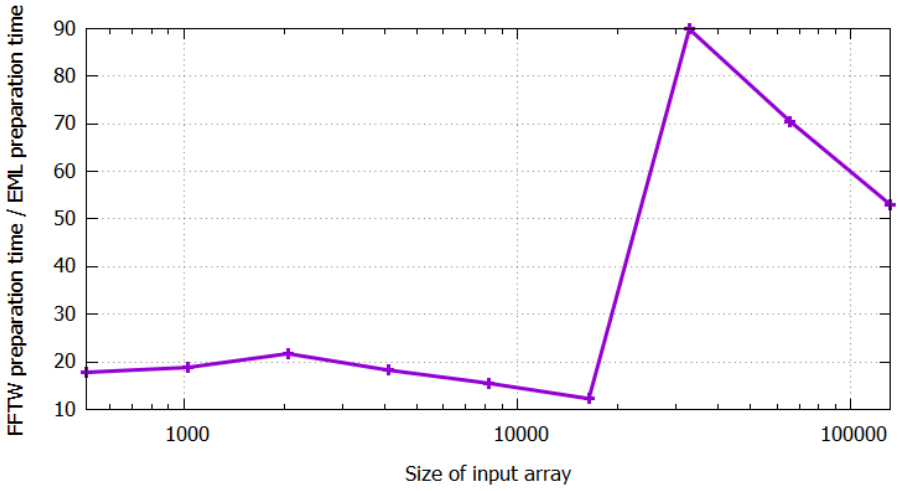


Fig. 5. The dependence of the ratio of FFT preparation time using EML and using FFTW on size of input array for Elbrus-8S

preparation, so it will have a significant effect only if the algorithm is run multiple times after a single preparation. This often happens when we need to perform an FFT on a set of arrays of the same size.

The execution time of the FFT algorithm using EML turns out to be from 1 to 10 times greater than the execution time of the algorithm using FTTW for array sizes less than 2^{11} (figure 8). And for large array sizes the situation reverses and the ratio of execution time using FFTW to the one using EML increases from 1 to 6 for the array of sizes $2^{14} - 2^{22}$.

Figure 8 shows that for the Elbrus-8S, the difference in preparation time is smaller than for the Elbrus-4S. For arrays smaller than 2^{12} the execution time using EML is close to the one using FFTW. And for large arrays up to 2^{18} the ratio of execution time using FFTW to the one using EML is in the range from 1.4 to 1.9.

On Elbrus-4S multiple starts (more than 1000) of FFT using FFTW more efficient for small arrays (less than 2^{11}) than using EML. Execution time using FFTW is 1 to 10 times faster than using a library EML for Elbrus-4S. FFT of array of almost all sizes on Elbrus-8S is more efficient to run using the EML library, but the ration of execution time for FFTW and EML is less than 2.

4.3 Fast Fourier Transform using FFTW on Elbrus CPUs vs Intel Xeon

Let's compare the work of FFTW on the Elbrus-8S and Intel Xeon E5-2660 processors. One can see that initialization is performed on the intel processor three times faster than on the Elbrus processor for a wide range of sizes of the input array (figure 9a). However, this acceleration does not compensate for the

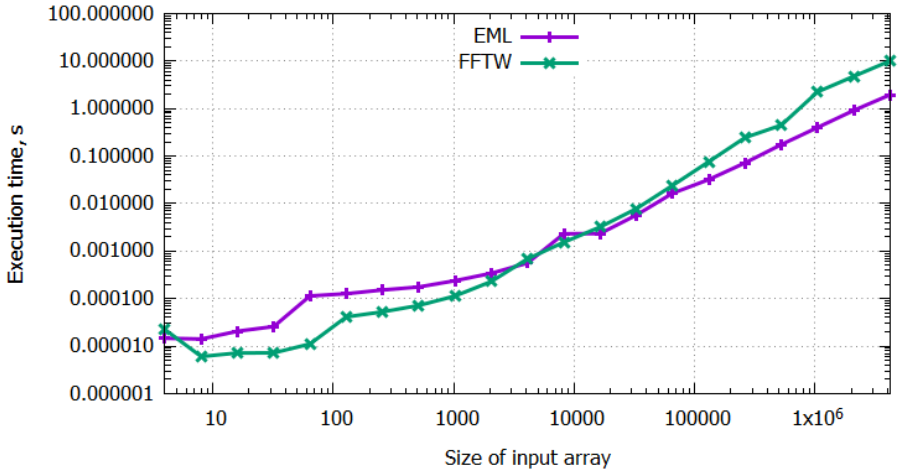


Fig. 6. The dependence of the FFT execution time on size of input array for Elbrus-4S

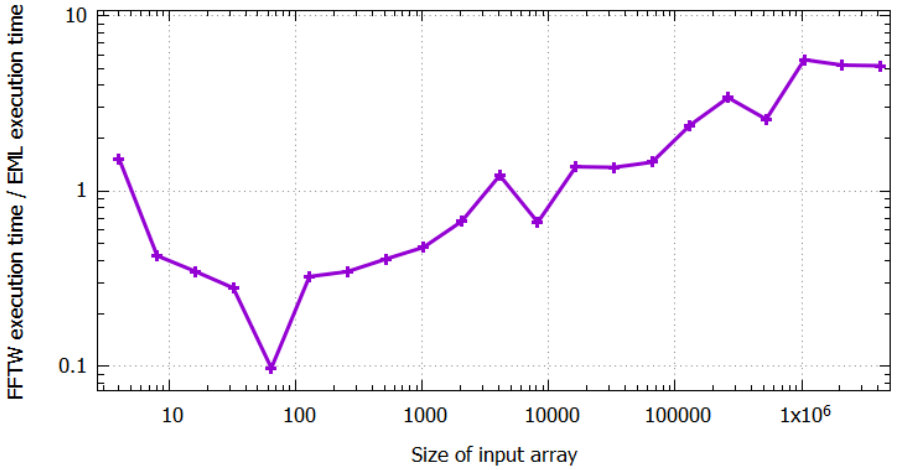


Fig. 7. The dependence of the ratio of FFT execution time using EML and using FFTW on size of input array for Elbrus-4S

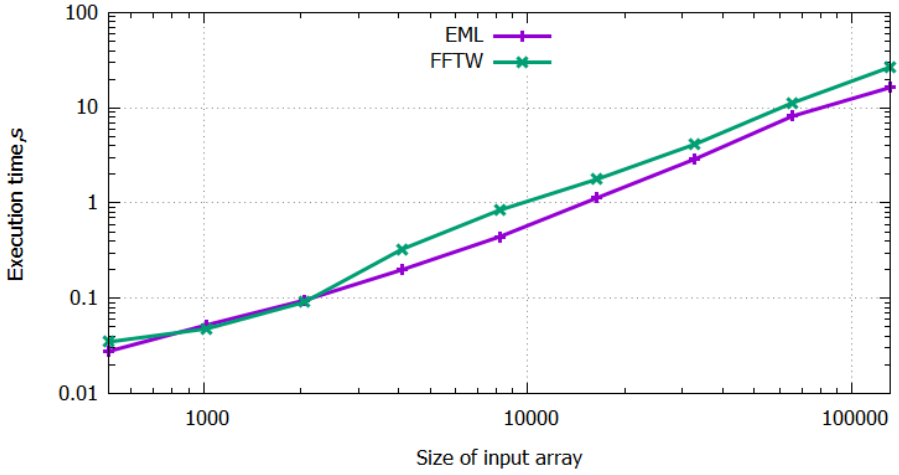
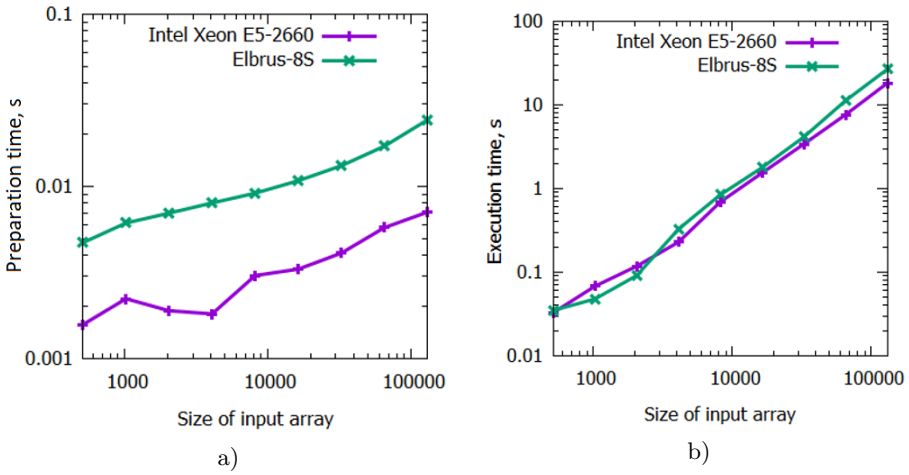


Fig. 8. The dependence of the FFT execution time on size of input array for Elbrus-8S



a) b)

Fig. 9. The dependence of the FFTW preparation time (a) and execution time (b) on the Elbrus-8S and Intel Xeon E5-2660 processors

difference in time between performing FFT using EML and FFTW libraries on the Elbrus processor. Thus, we can conclude that the phase of preparing the Fourier transform performs on the Elbrus processor using the EML library several times faster than the same phase is performed using the FFTW library on the Intel Xeon E5-2660 processor. The phases of execution of the algorithm on two processors take almost the same time (figure 9b). This is quite surprising given the fact that the frequency of Intel processor 2.2GHz, and the frequency of the Elbrus processor is 1.3GHz. It is also worth noting that FFTW is significantly

optimized by itself and it is especially strongly optimized for Intel processors and it is absolutely not optimized for Elbrus processors.

5 Conclusions

We have performed test calculations for the VASP model on Intel Xeon Haswell and Elbrus-8S CPUs with the best choice of mathematical libraries available. Elbrus-8S shows larger time-to-solution values, however there is no large gap between performance of Elbrus-8S and Xeon Haswell CPUs. The major target for optimization that could significantly speed-up VASP on Elbrus-8S is the FFT library.

We have performed test of native EML library and unoptimised FFTW library on processors Elbrus-4S and Elbrus-8S. Single launches of the FFT on Elbrus-4S and Elbrus-8S are more efficient using the EML library. Multiple starts (more than 10000) of FFT using FFTW is more efficient for small arrays (less than 4000) than using EML. FFT of any array using Elbrus-8S is more efficient to run using the EML library. FFTW performance on Elbrus-8S is competitive with Intel Xeon Broadwell CPUs. EML on Elbrus-8S (1.3GHz) appears to be close or even more effective than FFTW on Intel Xeon E5-2660v4 (2.2 GHz).

Acknowledgments. The authors acknowledge Joint Supercomputer Centre of Russian Academy of Sciences (http://www.jscc.ru) for the access to the supercomputer MVS1P5. The authors acknowledge JSC MCST (http://www.msct.ru) for the access to the servers with Elbrus CPUs. The authors are grateful to Vyacheslav Vecher for the help with calculations based on hardware counters.

The work was supported by the grant No. 14-50-00124 of the Russian Science Foundation.

References

1. Tyutlyaeva, E., Konyukhov, S., Odintsov, I., Moskovsky, A.: The Elbrus Platform Feasibility Assessment for High-Performance Computations, pp. 333–344. Springer International Publishing, Cham (2016). doi: 10.1007/978-3-319-55669-7_26
2. Kozhin, A.S., Polyakov, N.Y., Alfonso, D.M., Demenko, R.V., Klishin, P.A., Kozhin, E.S., Slesarev, M.V., Smirnova, E.V., Smirnov, D.A., Smolyanov, P.A., Kostenko, V.O., Gruzdov, F.A., Tikhorskiy, V.V., Sakhin, Y.K.: The 5th generation 28nm 8-Core VLIW Elbrus-8C processor architecture. Proceedings - 2016 International Conference on Engineering and Telecommunication, EnT 2016 pp. 86–90 (2017). doi: 10.1109/EnT.2016.25
3. Kresse, G., Hafner, J.: Ab initio molecular dynamics for liquid metals. Phys. Rev. B **47**, 558–561 (1993). doi: 10.1103/PhysRevB.47.558
4. Kresse, G., Hafner, J.: Ab initio molecular-dynamics simulation of the liquid-metal-amorphous-semiconductor transition in germanium. Phys. Rev. B **49**, 14,251–14,269 (1994). doi: 10.1103/PhysRevB.49.14251

5. Kresse, G., Furthmuller, J.: Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. Computational Materials Science **6**(1), 15 – 50 (1996). doi: 10.1016/0927-0256(96)00008-0
6. Kresse, G., Furthmüller, J.: Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. Phys. Rev. B **54**, 11,169–11,186 (1996). doi: 10.1103/PhysRevB.54.11169
7. Stegailov, V.V., Orekhov, N.D., Smirnov, G.S.: HPC hardware efficiency for quantum and classical molecular dynamics. In: V. Malyshkin (ed.) Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings, pp. 469–473. Springer International Publishing, Cham (2015). doi: 10.1007/978-3-319-21909-7_45
8. Aristova, N.M., Belov, G.V.: Refining the thermodynamic functions of scandium triflouride scf3 in the condensed state. Russian Journal of Physical Chemistry A **90**(3), 700–703 (2016). doi: 10.1134/S0036024416030031
9. Kochikov, I.V., Kovtun, D.M., Tarasov, Y.I.: Electron diffraction analysis for the molecules with degenerate large amplitude motions: Intramolecular dynamics in arsenic pentafluoride. Journal of Molecular Structure **1132**, 139 – 148 (2017). doi: 10.1016/j.molstruc.2016.09.064
10. Minakov, D.V., Levashov, P.R.: Melting curves of metals with excited electrons in the quasiharmonic approximation. Phys. Rev. B **92**, 224,102 (2015). doi: 10.1103/PhysRevB.92.224102
11. Minakov, D., Levashov, P.: Thermodynamic properties of lid under compression with different pseudopotentials for lithium. Computational Materials Science **114**, 128 – 134 (2016). doi: 10.1016/j.commatsci.2015.12.008
12. Bethune, I.: Ab initio molecular dynamics. Introduction to Molecular Dynamics on ARCHER (2015)
13. Hutchinson, M.: Vasp on gpus. when and how. GPU technology theater, SC15 (2015)
14. Zhao, Z., Marsman, M.: Estimating the performance impact of the MCDRAM on KNL using dual-socket Ivy Bridge nodes on Cray XC30. In: Proceedings of the Cray User Group — 2016 (2016)
15. Wende, F., Marsman, M., Steinke, T.: On Enhancing 3D-FFT Performance in VASP. CUG Proceedings p. 9 (2016)
16. Burtscher, M., Kim, B.D., Diamond, J., McCalpin, J., Koesterke, L., Browne, J.: Perfexpert: An easy-to-use performance diagnosis tool for HPC applications. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC '10, pp. 1–11. IEEE Computer Society, Washington, DC, USA (2010). doi: 10.1109/SC.2010.41
17. Rane, A., Browne, J.: Enhancing performance optimization of multicore/multichip nodes with data structure metrics. ACM Trans. Parallel Comput. **1**(1), 3:1–3:20 (2014). doi: 10.1145/2588788
18. Stanisic, L., Mello Schnorr, L.C., Degomme, A., Heinrich, F.C., Legrand, A., Videau, B.: Characterizing the Performance of Modern Architectures Through Opaque Benchmarks: Pitfalls Learned the Hard Way. In: IPDPS 2017 - 31st IEEE International Parallel & Distributed Processing Symposium (RepPar workshop), pp. 1588–1597. Orlando, United States (2017)
19. Baker, M.: A study of improving the parallel performance of vasp. Ph.D. thesis, East Tennessee State University (2010)