# Performance of Elbrus Processors for Computational Materials Science Codes and Fast Fourier Transform

Vladimir Stegailov[1,2], Alexey Timofeev[1,2](✉), and Denis Dergunov[1,2]

[1] Joint Institute for High Temperatures of the Russian Academy of Sciences, Izhorskaya st. 13 Bd.2, Moscow 125412, Russia
stegailov@gmail.com, timofeevalvl@gmail.com

[2] National Research University Higher School of Economics, Myasnitskaya st. 20, Moscow 101000, Russia

**Abstract.** Modern Elbrus-4S and Elbrus-8S processors provide a level of floating-point performance close to that of widespread x86_64 CPUs that are predominantly used in high-performance computing (HPC). The uniqueness of the software ecosystem of Elbrus processors requires special attention in the case of their deployment for execution of mainstream computational codes. In this paper, we consider the performance of one widely used code for computational materials science (VASP), as well as FFT libraries. The results for the Elbrus processors are embedded into the context of performance of modern x86_64 CPUs.

**Keywords:** Elbrus architecture · VASP · Fourier transform

## 1 Introduction

A large share 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 presented the server-grade 64-bit ThunderX2 ARMv8 CPUs, which are to be deployed in new Cray supercomputers. The Elbrus microprocessors stand among the emerging types of high-performance CPU architectures [1,2].

The diversity of CPU types significantly complicates 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, which represents the envisaged workload of a system under development.

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

L. Sokolinsky and M. Zymbler (Eds.): PCT 2018, CCIS 910, pp. 92–103, 2018.
https://doi.org/10.1007/978-3-319-99673-8_7

Computational materials science provides an essential part of the deployment time of HPC resources worldwide. The VASP code [3–6] is among the most popular programs for electronic structure calculations. It makes it possible to calculate materials properties using non-empirical (so called *ab initio*) methods. *Ab initio* calculation methods based on quantum mechanics are important modern scientific tools (see, e.g., [7–11]). According to recent estimates, VASP alone consumes from 15 to 20% of the world's supercomputing power [12,13]. Such an unprecedented popularity has led to a special attention directed towards the optimization of VASP for both existing and novel computer architectures (see, e.g., [14]).

The computation of Fourier transforms accounts for a significant part of the calculation time in software packages for computational materials science. One of the most time consuming components in VASP is 3D-FFT [15]. FFT libraries were tested on the Elbrus processor in order to determine the most optimal tool for computing fast Fourier transforms. The EML library, developed by the manufacturer of the Elbrus processor, and the most popular FFTW library are under consideration.

In this work, we present an efficiency analysis of Elbrus CPUs compared with Intel Xeon Haswell CPUs, using a typical VASP workload example. Here we also give the results of the test of FFT libraries on Elbrus processors.

## 2 Related Work

HPC systems are notorious for operating at a small fraction of their peak performance. 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 algorithm optimization that takes into account essential properties of the hardware (see, e.g., [16–18]). The recent work of Stanisic *et al.* [19] 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 significant part of the program's running time [20], especially in the case of VASP [15]. A detailed optimization of the computation of 3D-FFT in VASP to prepare the code for an efficient execution on multi- and many-core CPUs as Intel's Xeon Phi is considered in [15]. In this article, the threading performance of the widely used FFTW library (Cray LibSci) and Intel's MKL on the Cray-XC40 with Intel Haswell CPUs and the modern Cray-XC30 Xeon Phi (Knights Corner, KNC) system is evaluated. Recently, several 64-bit x86_64 and Armv8 CPUs have been compared using a VASP benchmark test with the focus on the memory bandwidth [21,22].

At the moment, Elbrus processors are ready for use [1,2], so we decided to benchmark them using one of the main HPC tools applied in materials 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, during the execution of the FFT, the butterfly computation occurs in a smaller number of cycles than it does on such CPUs as Intel's Xeon Phi.

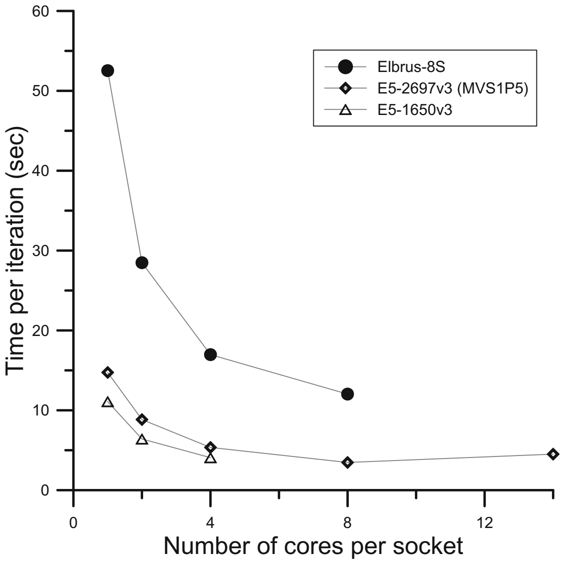

**Fig. 1.** Dependence between the first iteration time in the liquid-Si model test and the number of cores per socket

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

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

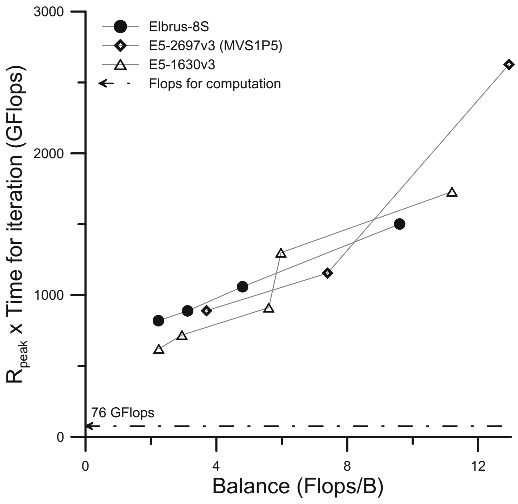

**Fig. 2.** Dependence between the first iteration time in the liquid-Si model test and the number of cores per socket, for reduced parameters $R_{\text{peak}}\tau_{\text{iter}}$ and balance $B$ ($R_{\text{peak}}$ is the total peak performance of all the cores used; the balance $B$ corresponds to the total bandwidth for a single/dual-socket server)

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 systems. As an input array for the Fourier transforms, a sinusoidal signal, white, pink and brown noise are used. In this article, we report the results for white noise.

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

1. Preparation stage: creates plans for FFT computations, e.g., via `fftw_plan p=fftw_plan_dft(..)` for FFTW, and via `eml_Signal_FFTInit(...)` for EML.
2. Execution stage: performs FFT computations using the plan created, e.g., via `fftw_execute_dft(p,in,out)` for FFTW, and via `eml_Signal_FFTFwd(...)` for EML.
3. Clean up.

We consider the work of the first two stages as they are the most time consuming. Preparation takes the main time when one starts 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 the FFTW and EML libraries on the processors Elbrus-4S and Elbrus 8S. For the moment, the EML library has fewer useful functions than the FFTW library. 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 the user. The number of functions in the EML library is much smaller than that in the FFTW library.

Plan creation with FFTW can be done by planner schemes that differ in their costs: `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 the best performance on the target platform. On servers with Elbrus-4S and Elbrus-8S processors, the authors, owing to lack of libraries, managed to compile FFTW only in `FFTW_ESTIMATE` mode, in which the preparation time is short and the execution time is long.

To average the operating time values and obtain the dispersion of the results, calculations were repeated 30 to 1000 times. The dispersion 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 1 shows the results of the liquid-Si model test runs.

Performance comparison of different CPUs usually resembles a comparison of "apples and oranges". To compare CPUs with different frequencies and different peak Flops/cycle values, it is better to use the reduced parameter $R_{\mathrm{peak}}\tau_{\mathrm{iter}}$ [7,23].

Another reduced parameter that characterizes the memory subsystem is the so-called balance $B$, which is the ratio of $R_{\mathrm{peak}}$ to the CPU memory bandwidth (in this work, we measure the latter quantity using the STREAM benchmark).

Figure 2 shows the same data as Fig. 1 but in reduced coordinates. This allows to eliminate the differences in floating-point performance and memory bandwidth between dissimilar CPU cores. In these reduced coordinates, the scatter of data points is much smaller, and there is an evident common trend.

The test model considered fixes the total number of arithmetic operations (Flops) required for its solution. An increase in $R_{\mathrm{peak}}\tau_{\mathrm{iter}}$ (that is proportional to the number of CPU cycles) leads to an increase in overhead due to the limited memory bandwidth. More CPU cycles are required for the CPU cores involved in computations to get data from DRAM.

We calculated the number of floating-point operations that corresponds to $\tau_{\mathrm{iter}}$. We used a system with Intel Core i7 640UM CPU. This CPU does not support AVX instructions and the performance counters work unambiguously. The resulting value of $N_{FP} = 76$ GFlops is shown in Fig. 2 as a dashed-dotted horizontal line. The ratio $R_{\text{peak}}\tau_{\text{iter}}/N_{FP}$ indicates the overhead of CPU cycles that are not deployed for computations because the required data from DRAM are not available. We should notice that the overall trend in Fig. 2 corresponds quite well to the limiting case $R_{\text{peak}}\tau_{\text{iter}} \to N_{FP}$ when $B \to 0$.

## 4.2 Fast Fourier Transform on Elbrus CPUs: EML vs. FFTW

We split the Fourier transformation process into two stages: the preparation of the algorithm (Figs. 3, 4, 5 and 6), and the execution of the transformation (Figs. 7, 8, 9 and 10). The preparation takes the main amount of time when one starts the Fourier transform once. The algorithm execution time can determine the total running time of the Fourier transform in situations when the Fourier transform is started many times for a fixed size of the input array.

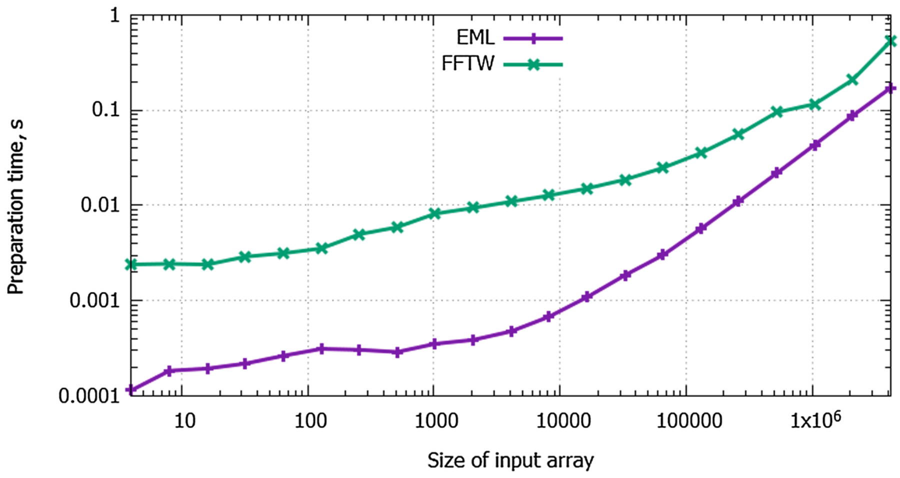

**Fig. 3.** Dependence between the FFT preparation time and the size of the input array, for Elbrus-4S

The preparation time of the FFT algorithm for Elbrus-4S appears to be an order of magnitude smaller when using the EML library than it is when using FFTW, for array sizes smaller than $2^{15}$ (Figs. 3 and 4). For larger array sizes, the preparation time is only 2 to 3 times smaller with EML than it is with FFTW. All points have an error less than 1%. As Figs. 5 and 6 show, the difference in preparation time is even greater for the Elbrus-8S. For array sizes smaller than $2^{15}$, the preparation time when using EML is 10 to 20 times less than it is when using FFTW. For larger array sizes (up to $2^{17}$), the preparation time when using EML is 50 to 90 times less than it is in the case of FFTW.

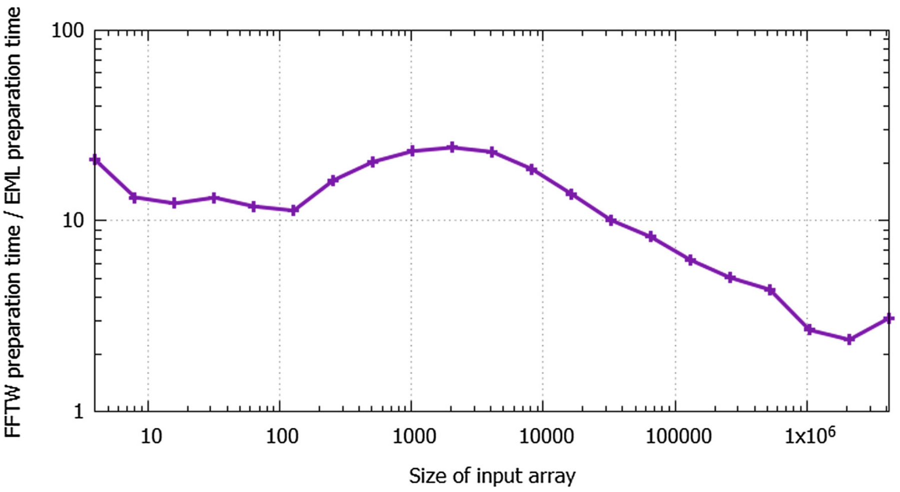

**Fig. 4.** Dependence between the ratio of FFT preparation time with FFTW to that with EML and the size of the input array, for Elbrus-4S

We can thus make an interim summary: single launches of the FFT on Elbrus-4S and Elbrus-8S are more efficient when using the EML library because the preparation of the FFT algorithm when using EML is faster (2 to 20 times for Elbrus-4S, and 10 to 90 times for Elbrus-8S) than it is when using FFTW.

And now we consider the second stage of the FFT implementation, namely the execution of the algorithm. The execution stage of the algorithm takes from one to several orders of magnitude less time than its preparation stage, so it has a significant effect only if the algorithm is run multiple times after a single preparation. This often happens when we need to execute an FFT on a set of arrays of the same size.

For array sizes less than $2^{11}$, the execution time of the FFT algorithm using EML turns out to be from 1 to 10 times greater than it is when using FTTW (Figs. 9 and 10). For larger array sizes, the situation reverses, and the ratio of the execution time with FFTW to that with EML increases from 1 to 6 for array sizes between $2^{14}$ and $2^{22}$. Figures 9 and 10 show that the difference in preparation time is smaller for the Elbrus-8S than for the Elbrus-4S. For arrays smaller than $2^{12}$, the execution time when using EML is close to that when using FFTW. For larger arrays (up to $2^{18}$), the ratio of execution time with FFTW to that with EML ranges from 1.4 to 1.9.

On Elbrus-4S, multiple starts (more than 1000) of FFT for small arrays (less than $2^{11}$) are more efficient when using FFTW than they are when using EML. On Elbrus-4S, the execution time when using FFTW is 1 to 10 times faster than it is when using the EML library. On Elbrus-8S, FFT for arrays of almost all sizes is more efficient when using the EML library, but the ratio of the execution time for FFTW to that for EML is less than 2.

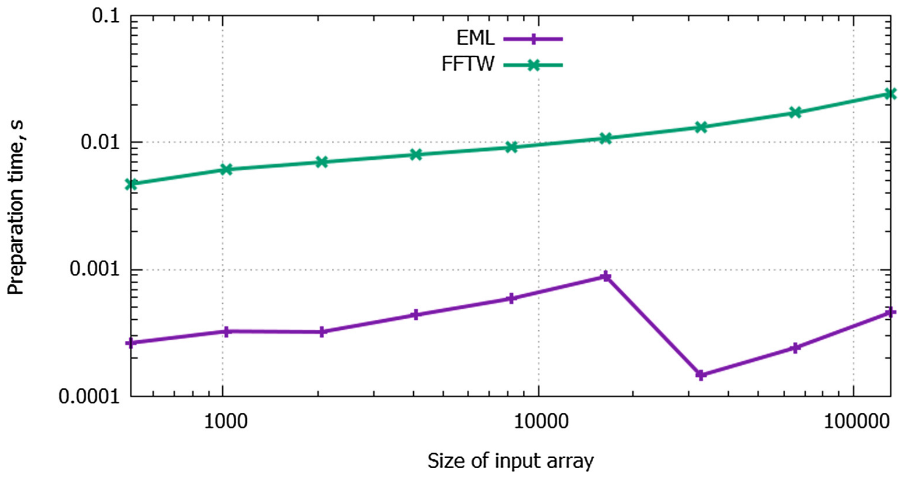

**Fig. 5.** Dependence between the FFT preparation time and the size of the input array, for Elbrus-8S

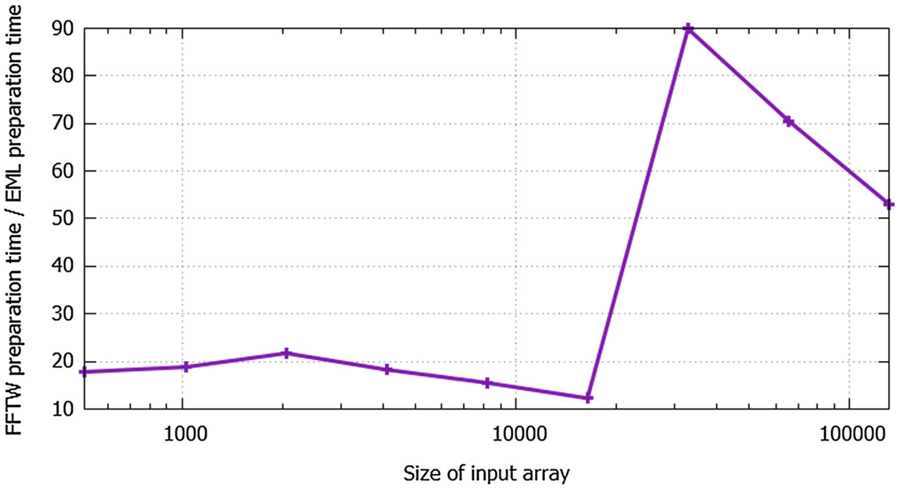

**Fig. 6.** Dependence between the ratio of FFT preparation time with FFTW to that with EML and the size of the input array, for Elbrus-8S

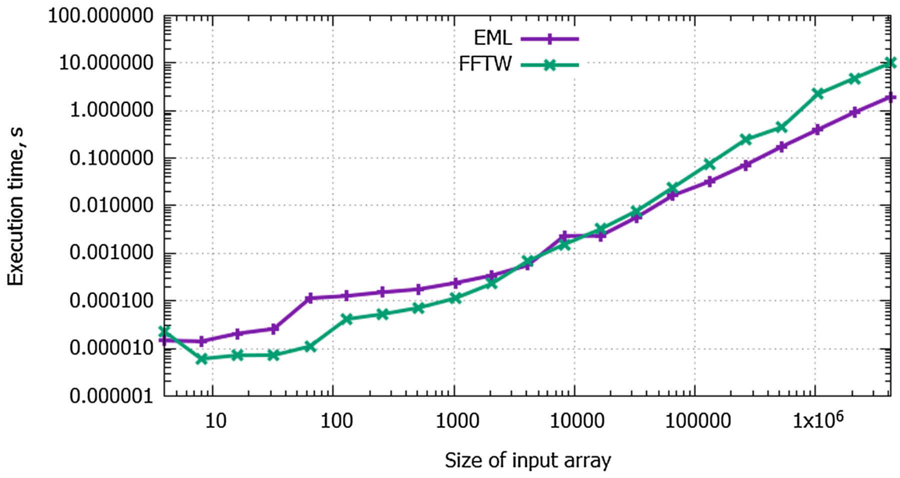

**Fig. 7.** Dependence between the FFT execution time and the size of the input array, for Elbrus-4S

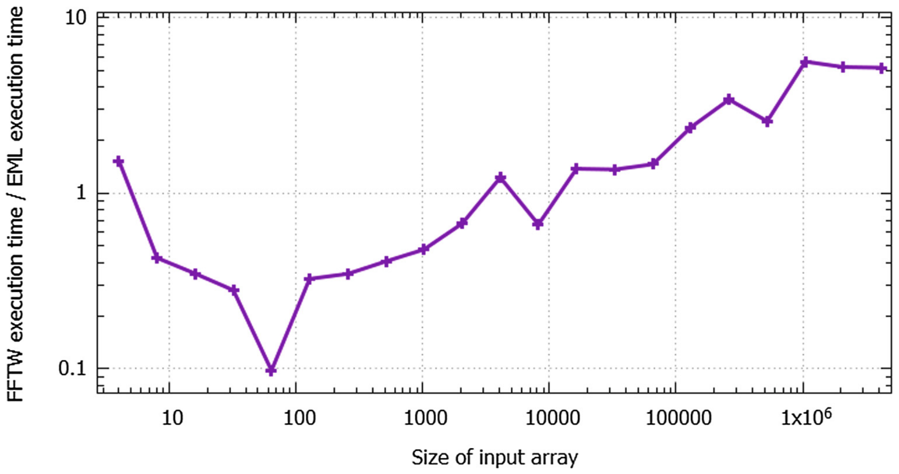

**Fig. 8.** Dependence between the ratio of FFT execution time with FFTW to that with EML and the size of the input array, for Elbrus-4S

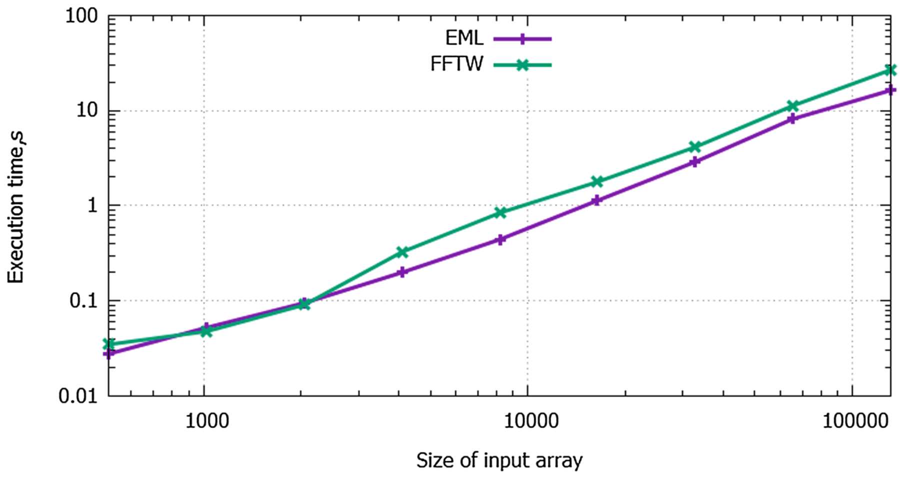

**Fig. 9.** Dependence between the FFT execution time and the size of the input array, for Elbrus-8S

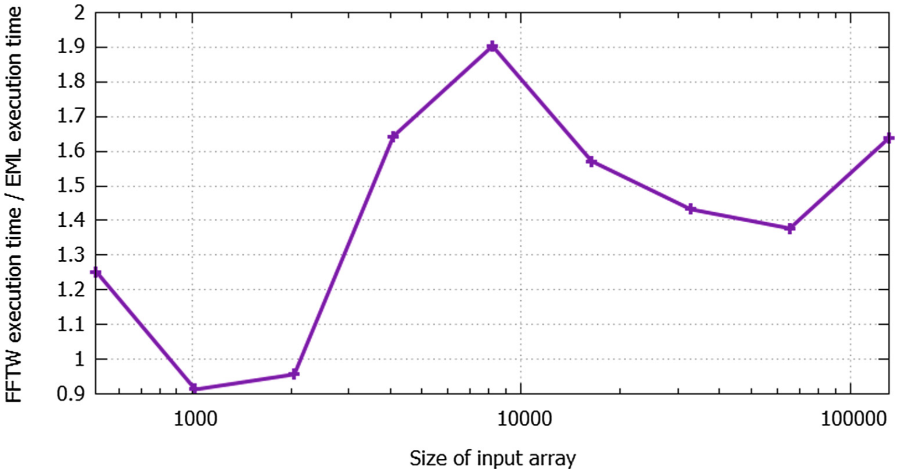

**Fig. 10.** Dependence between the ratio of FFT execution time with FFTW to that with EML and the size of the input array, for Elbrus-8S

## 5 Conclusions

We 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, but there is not a large gap between the Elbrus-8S performance and that of Xeon Haswell CPUs. The major target for optimization, which could significantly speed up VASP on Elbrus-8S, is the FFT library.

We tested the native EML library and an unoptimized FFTW library on the Elbrus-4S and Elbrus-8S processors. Single launches of the FFT on both Elbrus-4S and Elbrus-8S are more efficient when using the EML library. Nevertheless, for small arrays (less than 4000), multiple starts (more than 1 000) of FFT are more efficient with FFTW than they are with EML. On Elbrus-8S, FFT for arrays of any sizes is more efficient when running with the EML library.

## References

1. Kozhin, A.S., et al.: The 5th generation 28nm 8-core VLIW Elbrus-8C processor architecture. In: Proceedings - 2016 International Conference on Engineering and Telecommunication, EnT 2016, pp. 86–90 (2017). https://doi.org/10.1109/EnT.2016.25
2. Tyutlyaeva, E., Konyukhov, S., Odintsov, I., Moskovsky, A.: The Elbrus platform feasibility assessment for high-performance computations. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2016. CCIS, vol. 687, pp. 333–344. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-55669-7_26
3. Kresse, G., Hafner, J.: Ab initio molecular dynamics for liquid metals. Phys. Rev. B **47**, 558–561 (1993). https://doi.org/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**, 14251–14269 (1994). https://doi.org/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. Comput. Mater. Sci. **6**(1), 15–50 (1996). https://doi.org/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**, 11169–11186 (1996). https://doi.org/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: Malyshkin, V. (ed.) Parallel Computing Technologies. LNCS, vol. 9251, pp. 469–473. Springer, Heidelberg (2015). https://doi.org/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. Russ. J. Phys. Chem. A **90**(3), 700–703 (2016). https://doi.org/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. J. Mol. Struct. **1132**, 139–148 (2017). https://doi.org/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**, 224102 (2015). https://doi.org/10.1103/PhysRevB.92.224102
11. Minakov, D., Levashov, P.: Thermodynamic properties of LiD under compression with different pseudopotentials for lithium. Comput. Mater. Sci. **114**, 128–134 (2016). https://doi.org/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: 2016 Proceedings of the Cray User Group (2016)
15. Wende, F., Marsman, M., Steinke, T.: On enhancing 3D-FFT performance in VASP. In: 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 2010, pp. 1–11. IEEE Computer Society, Washington (2010). https://doi.org/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). https://doi.org/10.1145/2588788
18. Mantovani, F., Calore, E.: Performance and power analysis of HPC workloads on heterogeneous multi-node clusters. J. Low Power Electron. Appl. **8**(2), 13 (2018)
19. 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 and Distributed Processing Symposium (RepPar Workshop), Orlando, United States, pp. 1588–1597 (2017)
20. Baker, M.: A study of improving the parallel performance of VASP. Ph.D. thesis, East Tennessee State University (2010)
21. Stegailov, V., Vecher, V.: Efficiency analysis of Intel and AMD x86_64 architectures for ab initio calculations: a case study of VASP. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2017. CCIS, vol. 793, pp. 430–441. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-71255-0_35
22. Stegailov, V., Vecher, V.: Efficiency analysis of Intel, AMD and Nvidia 64-bit hardware for memory-bound problems: a case study of ab initio calculations with VASP. In: Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K. (eds.) PPAM 2017. LNCS, vol. 10778, pp. 81–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78054-2_8
23. Nikolskiy, V.P., Stegailov, V.V., Vecher, V.S.: Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics. In: 2016 International Conference on High Performance Computing Simulation (HPCS), pp. 682–689 (2016). https://doi.org/10.1109/HPCSim.2016.7568401