The Elbrus-4C Based Node as Part of Heterogeneous Cluster for Oil and Gas Processing Researches

Ekaterina Tyutlyaeva[1] ✉, Igor Odintsov[1], Alexander Moskovsky[1], Sergey Konyukhov[1], Alexander Kalyakin[2], and Murad I. Neiman-zade[2]

[1] ZAO RSC Technologies,
{xgl,igor_odintsov,moskov,s.konyuhov}@rsc-tech.ru
[2] MCST/INEUM {kalyakin,muradnz}@mcst.ru

Abstract. This paper briefly examines the advantages and disadvantages of Elbrus architectures as building blocks for Seismic Processing cluster system. The configuration of a heterogeneous clustered system build for research Oil and Gas Company is examined in more detail. In this system, processing nodes with different architecture (x86, GPU and e2k) are integrated in a single computing cluster through a high performance global networking topologies. Heterogeneous cluster with Elbrus node provides a good opportunity for software cross-architectural migration. To demonstrate the potential of Elbrus nodes usage, the multispectral data analysis application has been optimized for e2k architecture. Paper includes performance results and scalability analysis for implemented module using e2k and x86 nodes. It is anticipated that the heterogeneous cluster with Elbrus node will form an integral part of the preparation process of the domestic supercomputer under development, based on the Elbrus processors. The basic software stack in Seismic Processing will be naturally emerged on the use of Elbrus node as part of the heterogeneous cluster.

Keywords: VLIW ·Elbrus in HPC·e2k·Heterogeneous Cluster ·Elbrus-4C Based Node

1 Introduction

HPC facilities have become an important part of scientific researches in all areas of technology. Oil and gas companies use HPC for seismic processing [1], reservoir modeling process, interpretation and visualization. Developing the oil and gas fields can cost hundreds of millions of dollars, especially in geologically, environmentally and operationally challenging areas. Complex innovative techniques for analyzing and visualization of huge seismic and other geophysical datasets using HPC facilities could minimize the costs and risks associated with exploration and drilling in challenging areas.

Most of legacy algorithms used today were designed with x86 and GPUs in mind, so they can take advantage of the compute power that comes from using target hardware. At the same time, however, domestic hardware platform such as e2k is a reliable environment for processing nationally-important data and developing new competitive approaches.

It is well known that cross-architectural software migration is critical issue and requires a lot of time and efforts. Including the Elbrus node in heterogeneous cluster allows to use more specific allocation of work between nodes with different architectures:

– Complex legacy software libraries build for x86 architectures could be used on x86 nodes
– Complex visualization libraries parallelized with CUDA threads could be used on NVidia node
– New modules algorithms and modules could be designed and developed for Elbrus

The noteworthy feature here is decreasing qualified time for making determined efforts to adaptation of existing software for e2k architecture. Engineers could use Elbrus node only for developing new modules.

In addition, users sometimes runs what are called pre-processing and post-processing applications on the master/login node. The pre-processing applications create the input data for jobs that will run on the cluster (a job is a generic term to mean application launched by a job scheduler). The post-processing applications take the output of the cluster job and perform an analysis on the output data. Some of these operations could be also done on the Elbrus node.

Heterogeneous clustered system is the useful solution for the development and testing of advanced libraries for seismic data processing.

The main contribution of this paper is the example of application module ported to Elbrus node and x86 node, and comparative analysis of the optimizations and performance results.

The rest of the paper is structured as follows. Section 2 provides a detailed specification of the used heterogeneous system. Section 3 reviews basic e2k microarchitecture features. After that, Section 4 describes the testing application that processes multispectral image using Fourier transform. Section 5 describes the implementation and architecture-specific optimizations of these algorithms for e2k and x86 nodes. Finally, Section 5.4 provides with measured performance results. Afterwards, Section 6 concludes the paper with discussion of the obtained results.

2 Heterogeneous Clustered System

The heterogeneous system developed for research Oil and Gas Company consists of three types of nodes:

– The `x86 nodes` are 2-socket, based on Intel Xeon E5-2697A v4 processors with DDR4-2400 DRAM Memory, 8 x 16 GB,

– The `e2k node` is 4-socket, based on Elbrus-4C processors with DDR3-1600 DRAM memory, 12 x 4 GB,
– The `GPU node` is based on 2-socket Intel Xeon E5-2697A v4 processors with NVIDIA GRID Tesla M10 co-processor.

Current installation includes 6 x `x86 nodes`, 1 x `e2k node` and 1 x `GPU node` connected using high-performance Mellanox Infiniband FDR interconnect and service Ethernet interconnect. Simplified scheme of the installation is shown on the Fig. 1.

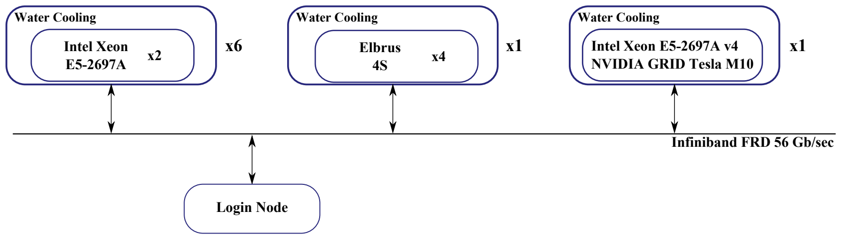

Fig. 1. Heterogeneous HPC System Scheme

Programming for heterogeneous cluster is a difficult task. Among others, performance and fault tolerance are probably the most important and challenging issues of heterogeneous parallel and distributed programming.

By the way, this heterogeneous cluster can be divided on three logical parts. The partition based on `x86 nodes` (Intel Xeon E5-2697A v4) have the property of homogeneity. According to the J. Dongarra and A. Lastovetsky [2], there is only one way for such a system to be homogeneous: all processors in the system have to be identical and interconnected via a homogeneous communication network, that is, a network providing communication links of the same latency and bandwidth between any pair of processors. But this definition is not complete. One more important restriction has to be satisfied: the system has to be dedicated, that is, at any time it can execute only one application providing all its resources to this application. Theoretical peak performance of this partition is about 10 TFlops, that is reasonable number for basic computational seismology tasks.

Visualization tasks could be performed using `GPU node` (NVIDIA GRID Tesla M10 co-processor).

Moreover, unified liquid cooling provides increased density and energy efficiency of the overall heterogeneous system.

3 Elbrus Node Characteristics

Microprocessor Elbrus architecture e2k is classified as VLIW (Very Large Instruction Word) microprocessor architecture. According to the VLIW principles [3], compiler forms a single instruction (Very large Instruction Word) using a multiple independent operations grouped together for parallel execution. Thus, the microprocessor design allows to take advantage of as much Instruction Level Parallelism (ILP) as possible. Optimizing compiler exempts hardware from their parallelization responsibilities and provide performance improvements using ILP techniques.

Along with effective ILP usage, e2k architecture includes the basis for implementing another levels of parallelism, such as vector parallelism, thread parallelism in shared memory, task parallelism in large distributed memory cluster.

Another basic feature of e2k architecture is effective binary compatibility with x86 architecture.

Elbrus software development strategy focused on growth of compatibility. A roadmap for Elbrus-based systems includes large list of planned software adaptation in different areas. For example, the study [4] published in 2018 includes analysis of the wide range of computing, graphical, network and disk-related software in the field of compatibility with Elbrus.

Theoretically, e2k design allows achieving better performance with reduced power consumption rate. As noted above, ompilers for e2k processors are required to package multiple instructions into Very Large Instruction Word, which are later treated by CPU pipeline as single instruction. Conversely, superscalar machines uses hardware for analysis of dependences among instructions at runtime. Using the compiler for the exploiting and scheduling of instructions allows to get rid of energy demanding mechanisms of dynamic instruction scheduling for Elbrus architecture [5].

An important implication of this microarchitecture design is that the way in which the application implemented and further compiler works can have a large effect on code execution performance.

4 Application

The testing application processes multispectral images archive in HDF5 format. All input images are roughly the same size, about 60 Mb of double precision data.

This module repeatedly performs Direct Fourier Transforms and Overdetermined real linear systems solving routines in moving window for each picture. The simplified flow graph of the module is shown on the Fig. 2.

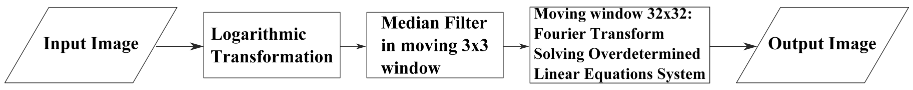
Input Image
Logarithmic Transformation
Median Filter in moving 3x3 window
Moving window 32x32: Fourier Transform Solving Overdetermined Linear Equations System
Output Image

Fig. 2. Simplified Application Flow Graph

Input and output data details are presented on the Table 1.

The Fourier transform is fundamental to seismic data analysis [6]. It applied to almost all stages of processing.

Table 1. Data Specifications

Stage	Data Format	Size, MB	Data Type	Dimensions H×W
Input	HDF5	60	Double	3072×4064
Output	ENVI	96	Double	3072×4064

– The analysis of a seismic trace into its sinusoidal components is achieved by the forward Fourier transform.
– The synthesis of a seismic trace from the individual sinusoidal components is achieved by the inverse Fourier transform.
– The two-dimensional (2-D) Fourier transform is a way to decompose a seismic wavefield, such as a common-shot gather, into its plane-wave components, each with a certain frequency propagating at a certain angle from the vertical.

For example, the Fourier transform is used to convert the signal into the frequency domain.

Thus, the application example given covers the most frequently used domain in seismic image processing – analysis in moving window using Fourier transform.

V. Loginov and P. Ishin [7] proposed the optimized FFT algorithm for Elbrus processor including special optimization for 32-bit floating point data type.

5 Implementation Details

The testing application has been implemented for Intel x86 nodes, based on Intel Xeon E5-2697A v4 processors and for e2k node, based on Elbrus-4C processors.

Basic source code has been implemented using C++ programming language to reach the maximal level of compiler-assisted optimization. The basic C++ version of the code implemented in straight-line manner; all repeatedly performed loops have single entry and single, not data-dependent exit. ompliance with these requirements and conditions is effective for both further implementations.

Furthermore, the two architecture-dependent branches has been implemented.

In summary, virtually all multi-core architectures now requires basic optimization in order to obtain a satisfactory result.

The best solution for mathematical calculation for both platforms is architecture dependent mathematical libraries: EML [9] (Elbrus Mathematical Library) for Elbrus and MKL [10] (Math Kernel Library) for Intel.

Each architecture requires specific code optimization.

Table 2. Architecture Dependent Implementations

Feature	Intel Implementation	Elbrus Implementation
MPI Compiler	Intel® MPI Library Version 2018	MPICH-2
C++ Compiler	Intel® Compilers 2018	lcc-1.23
Math Library	Intel® MKL Library (2018 Studio)	EML (Elbrus Mathematical Library) + lapack-3.8.0 (Elbrus optimized)
I/O library	The HDF5-1.8.20 Technology suite	The HDF5-1.8.20 Technology suite
Specific Compilation Keys	-ipo -xCORE-AVX2	-mcpu=elbrus-4c

5.1 Optimization for Intel node

For Intel implementation the auto-vectorization [8] features has been used to to improve the vectorizer performance:

- Avoiding, if it possible, dependencies between loop iterations
- Aligning data to 32 byte boundaries
- Using pragmas to give a hint to the compiler to asserts that data within the following loop is aligned, there aren't any data dependencies etc.

It is worst noting, however, that stronger optimizations, such as using Intel intrinsic instructions (C style functions that provide access to many Intel instruction, including AVX2, available on Intel Xeon E5-2697A v4 processors) haven't been used. So there are still rooms for optimization to fully exploit available processor vector capabilities.

5.2 Optimization for Elbrus node

Optimizations are very important for high performance VLIW, especially for microprocessors with static scheduling. Optimizing compiler [12] have taken up a lot of the work in the program parallelization, including utilization of instruction level parallelism, short vector instruction parallelism, and thread level parallelism. Optimizations for the studied module for Elbrus node included several steps:

- moving some invariant memory allocations/deallocations outside the loops
- replacing frequent invokes of "hypot" function in the innermost loops with vectorized "vhypot" version invoke outside these loops
- minor rearrange of certain local data in order to avoid indirect addressing in hot loops
- collapsing some simple loop nests to reduce overhead of loop software pipelining

– using linearized multidimentional arrays
– building project in -fwhole (-flto) that allowed to inline cross-module invokes of functions; better inline [13] is known to have increased positive effect on VLIW architecture processors
– modifying minor subroutine that contained indirect memory writes in innermost loop; replacing it with loop nest with indirect memory reads, to make loop software pipelining more effective.

As a result, program execution profile shows >85% of cycles spent in library functions, top of them being fftw/eml functions.

5.3 Data Parallelization.

Archive processing structure of the module described in Section 4 enables each picture to be computed independently. This enables the algorithm to be implemented in the data parallel way which is very efficient for high performance computing. Each MPI rank reads, processes and writes results for its own images independently, so there are minimum communications between processes.

For Intel x86 architectures, the OpenMP threads also used to exploit the Intel Hyper-Threading Technology [14] that enables effective multiple threads execution on each core. OpenMP threads used on Sharpness Index computation step for processing independent data in different positions of moving window.

5.4 Performance Testing

The performance testing has been conducted using `x86 node` (based on 2 x Intel Xeon E5-2697A v4 Processors) and `Elbrus node` (based on 4 x Elbrus-4C Processors).

Testing procedure included a series of 10 executions per each combination of input data set and input feature sets.

Appropriate preparatory steps has been done prior to each execution, especially removing the results of previous computations and cleaning up the caches and swap.

The Fig. 3 presents the median execution times for all executions. It's important to outline that number of used cores is equal to the number of processed pictures, so the ideal time result should be straight horizontal line. However, as the number of used cores and processed images is increased, performance characteristics deviate from linearity. It can be concluded that the collective work with memory and I/O slightly increases execution time with increased number of used cores and processed pictures.

According to the obtained results, the one Elbus-4C core requires about 93 sec to process one image; one Intel Xeon E5-2697A v4 core takes about 32 sec to process the same image. However, different levels of optimization should be taken into account.

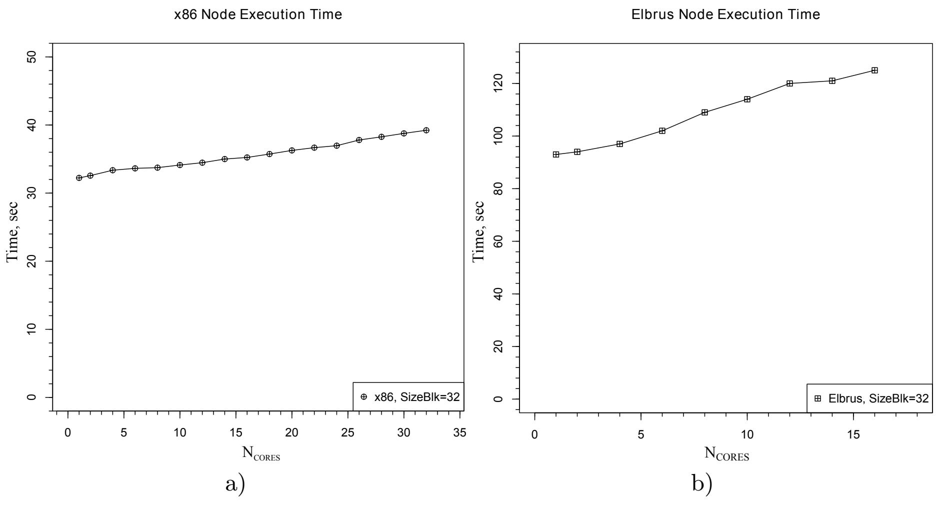

Fig. 3. Median execution times. a) x86 testbed b) Elbrus testbed

6 Conclusion

The important process of import substitution industrialization is actual now, in which domestic industries should be favored over imports. Heterogeneous cluster with Elbrus node provides an good opportunity for natural process of an increase the level of qualified technical expertise for e2k architecture usage and creating software stack in seismic processing area. Implementation Elbrus node as a part of heterogeneous cluster also simplifies software development and cross-architectural migration. Software engineer can easily check the correctness and compare performance results. It is hoped that usage of such architectures will identify possibilities for mutual reinforcement between Elbrus nodes / x86 nodes and Nvidia GPU.

As an example, the data parallel algorithm of multispectral images processing using Fourier transform is successfully implemented for both architectures.

While the optimization complexity is different, the experience gained could be replicated in other optimizations. Some important processing routines could be further reused.

The obtained performance results are in line with expectations, based on technical characteristics of studied nodes. Heterogeneous structure of the cluster allows to easily checking correctness and comparing performance characteristic.

The potential for Elbrus microprocessors for domestic HPC is tremendous; our current implementation has used previous generation of processors. However, the Elbrus processors must be further studied to fully exploit it potential. Elbrus-8S, new generation of Elbrus microprocessors is now available. The Elbrus-8S [15] includes eight cores, and is manufactured using a 28 nanometer process. Elbrus-8S equipped with L3 cache, shared across cores; It has 1300 MHz frequency versus

750 MHz for Elbrus-4C and supports up to 4 DDR3-1600 channels. According to the Elbrus developers plans, published this year [5], technical features of ”Elbrus-16SV” is expected to match Intel Xeon E74850 v4 (Broadwell) and Xeon Platinum 8153 (Skylake).

The experience of heterogeneous cluster development could be replicated in other cases, such as in the systems of ultra-high performance on domestic hardware and software Elbrus platform development, for example. Kim A.K., Perekatov V.I. and Feldman V.M [16] proposed an Elbrus-based data centers solution with water cooling and 4D-torus interconnect to provide highest performance and high density for the large domestic system.

Acknowledgments. This research is funded from the Ministry of Education and Science of the Russian Federation (Programme “SKIF-NEDRA”, the government contract 14.964.11.0001 dated 17 June 2015)

References

1. Liao, T.: HPC Challenges in Oil & Gas Upstream Scientific Applications. PPME Workshop, Portland, OR, USA, pp. 1-30 (2013)
2. Dongarra, J., Lastovetsky, A.: An Overview of Heterogeneous High Performance and Grid Computing. Engineering the Grid: Status and Perspective, American Scientific Publishers, February (2006)
3. Kim, A., Perekatov, V., Ermakov S.: Microprocessors and Computing Systems of the Elbrus Family. Saint-Petersburg.: Piter, 272 p.(2013) (In Russian) ISBN 978-5-459-01697-0
4. Molchanov I. A., Bychkov I. N.: Study of Elbrus computing platform in the field of compatibility including software adaptation. Voprosy radioelektroniki, no. 2, pp. 1422 (2018) (In Russian)
5. Kim, A.K., Perekatov, V.I., Feldman, V.M.: On the way to russian exasistemes: plans of the Elbrus hardware-software platform develpers on creation of an exaflops performance supercomputer. Voprosy radioelektroniki, 2018, no. 2, pp. 613. (In Russian)
6. Yilmaz, O.: Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data. Society of Exploration Geophysics, vol. 1 (2001)
7. Loginov, V.E., Ishin, P.A.: 32-bit Floating-Point Fast Fourier Transform Optimization for Elbrus Processor. Voprosy radioelektroniki, IVT Series, vol. 3 (2012) (In Russian)
8. Intel Corporation: A Guide to Vectorization with Intel C++ Compilers (2012)
9. EML Mathematical Library Home Page. url: http://www.mcst.ru/vysokoproizvoditelnye_biblioteki
10. Intel Math Kernel Library Documentation. url: https://software.intel.com/en-us/mkl/documentation
11. Ishin, P.A., Loginov, V.E., Vasilyev, P.P.: Computation Acceleration using High-Performance Mathematical and Multimedia Libraries for Elbrus Architecture. Aerospace Defence Herald, No 4 (8), pp.64-68 (2015) (In Russian)
12. Volkonskiy, V., Breger, A., Buchnev, A., Grabezhnoy, A., Ermolitsky, ., Mukhanov, L., Neimanzade, M., Stepanov, P., Chetverina, O.: Program parallelization methods implemented in optimizing compiler. Voprosy radioelektroniki, vol 4, issue 3, pp. 63–88 (2012)

13. Ermolitckii, A., Neiman-Zade, M., Chetverina, O., Markin, A., Volkonskii, V.: Aggressive Inlining for VLIW. Proceedings of the Institute for System Programming, vol. 27, issue 6, pp. 189-198 (2015)
14. Marr, D., Binns, F., L Hill, D., Hinton, G., A Koufty, D., J Miller, A., Upton, M.: Hyper-Threading Technology Architecture and Microarchitecture. Intel Technology Journal (2002)
15. Microprocessor Elbrus-8S URL: http://mcst.ru/elbrus-8c (In Russian)
16. Kim, A.K., Perekatov, V.I., Feldman, V.M.: Data centers based on Elbrus servers. Voprosy radioelektroniki, 2017, no. 3, pp. 612. (In Russian)