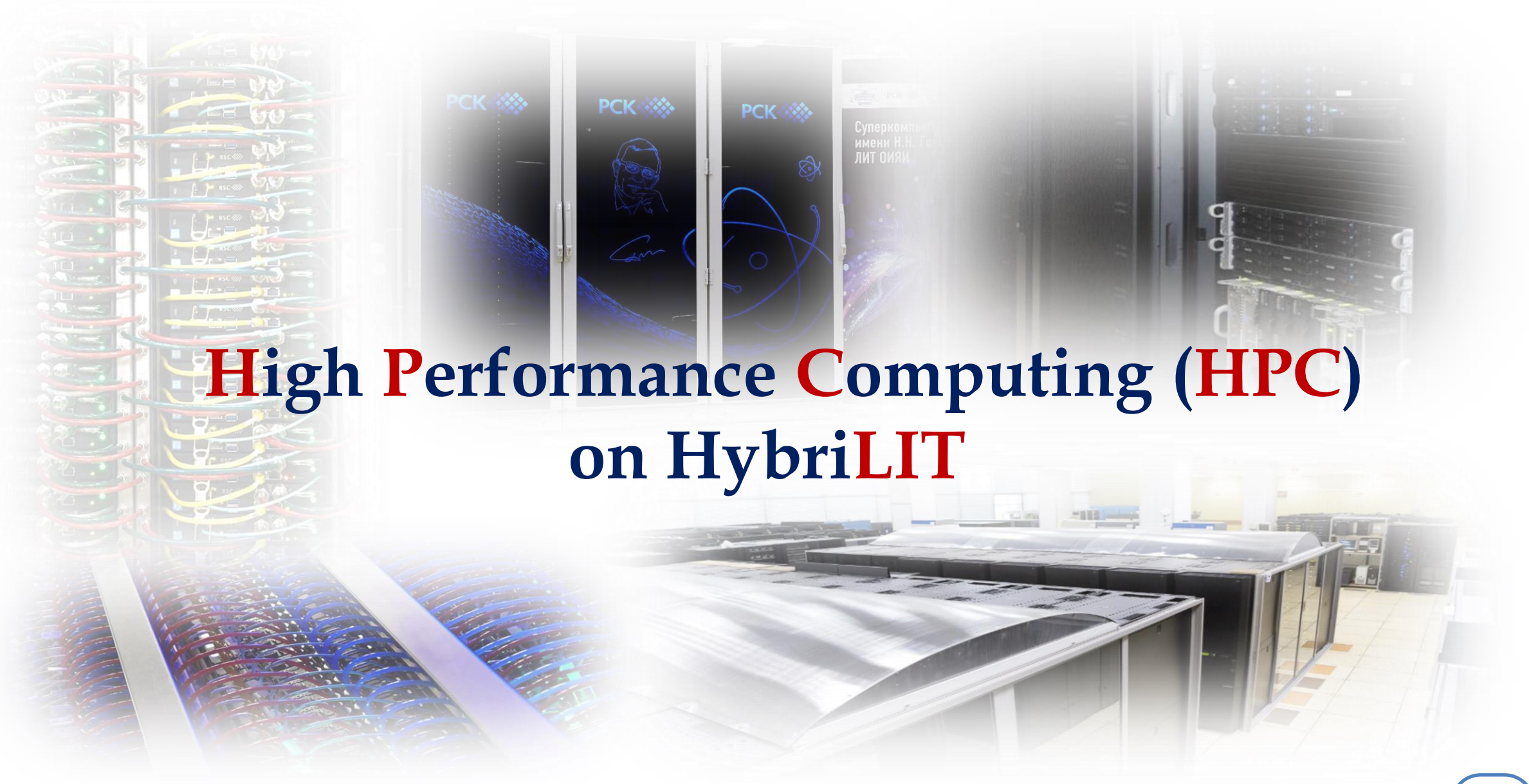
Parallel Algorithms for Investigation of Long Josephson Junctions *

M.V. Bashashin[1,3], A.V. Nechaevskiy[1], D.V. Podgainy[1], I.R. Rahmonov[2], Yu.M. Shukrinov[2,3], O.I. Streltsova[1,3], E.V. Zemlyanaya[1,3], M.I. Zuev[1]

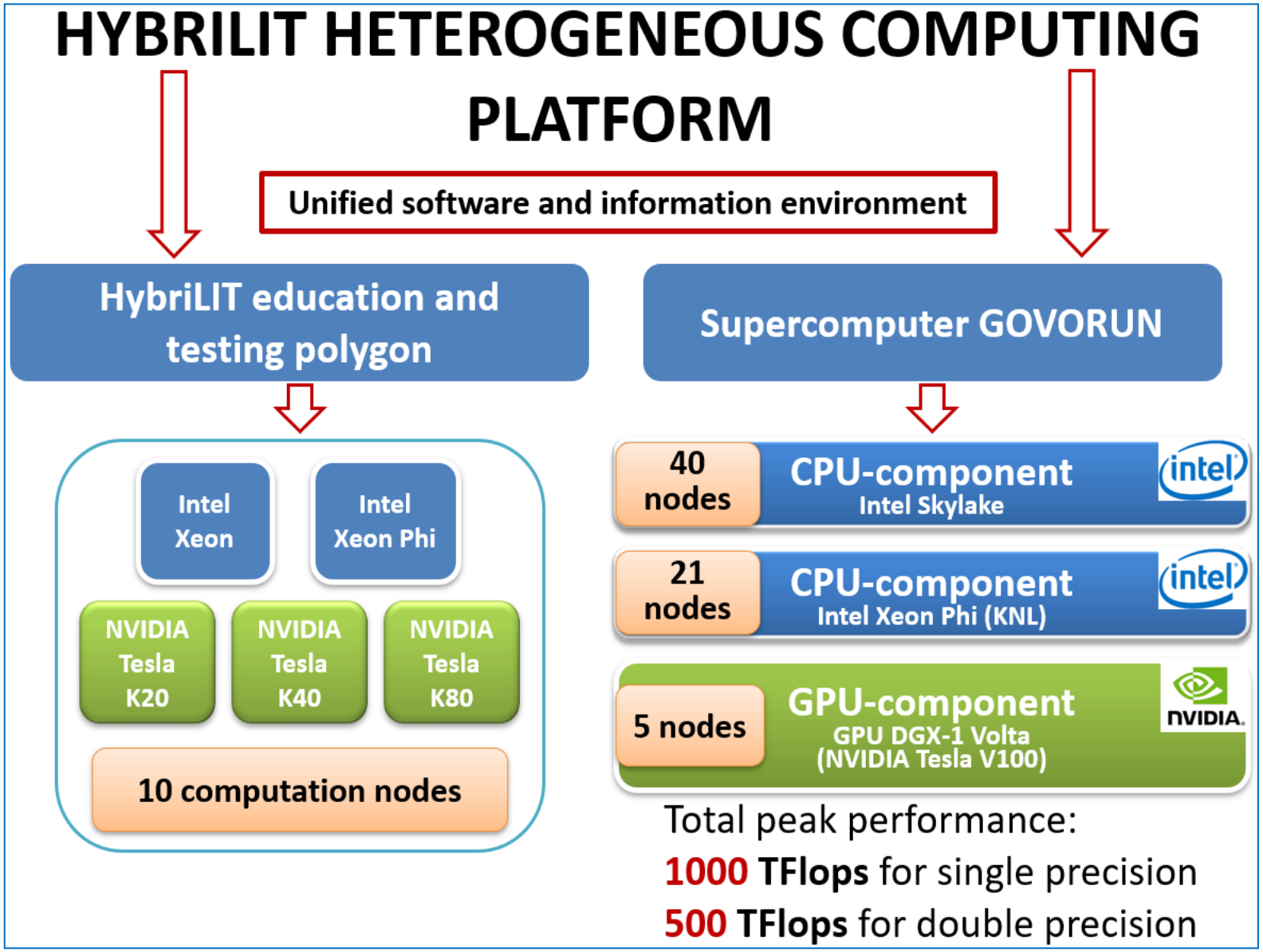
[1]Laboratory of Information Technologies, JINR
[2]Bogoliubov Laboratory of Theoretical Physics, JINR
[3]Dubna State University

*The study was supported by the Russian Science Foundation (the project № 18-71-10095).

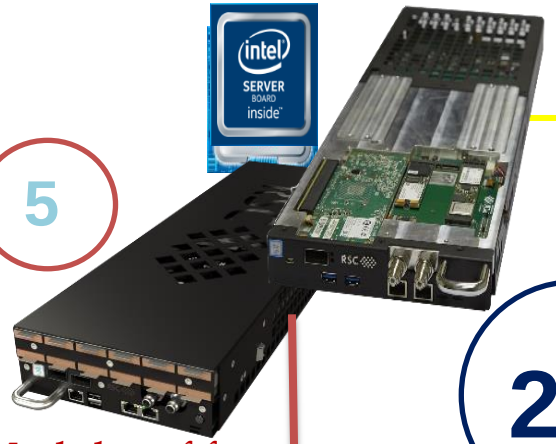
High Performance Computing (HPC) on HybriLIT



Heterogeneous platform HybriLIT



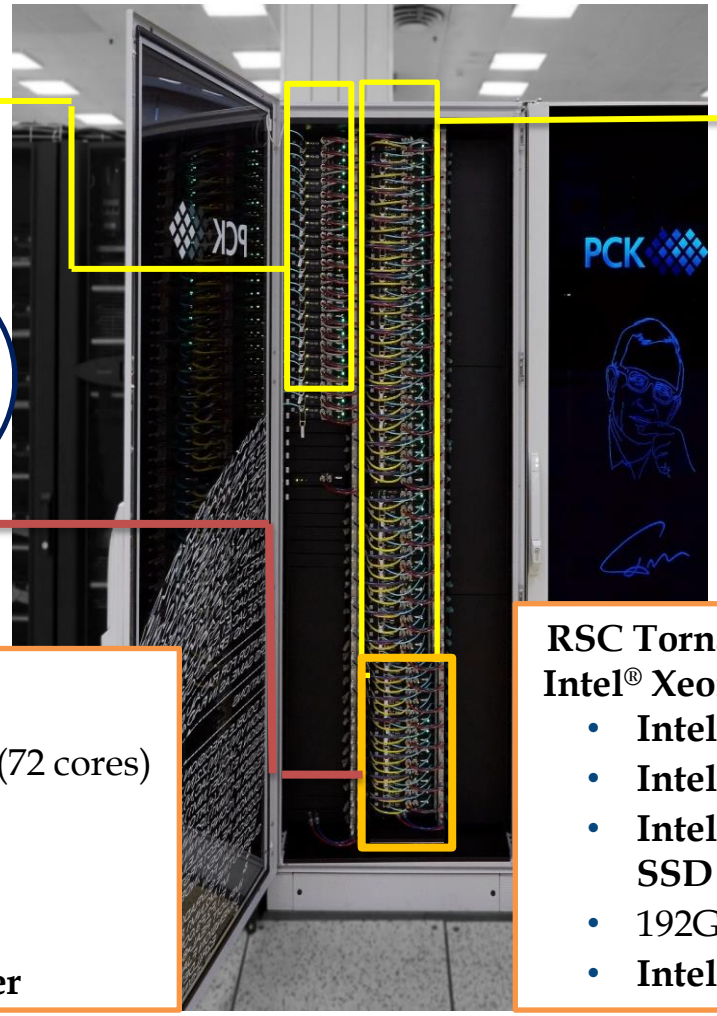
The CPU-component of the supercomputer “Govorun”



5

21

Modules of fast scalable parallel file system (*Lustre, EOS* etc.)



40

RSC Tornado nodes based on Intel® Xeon Phi™:

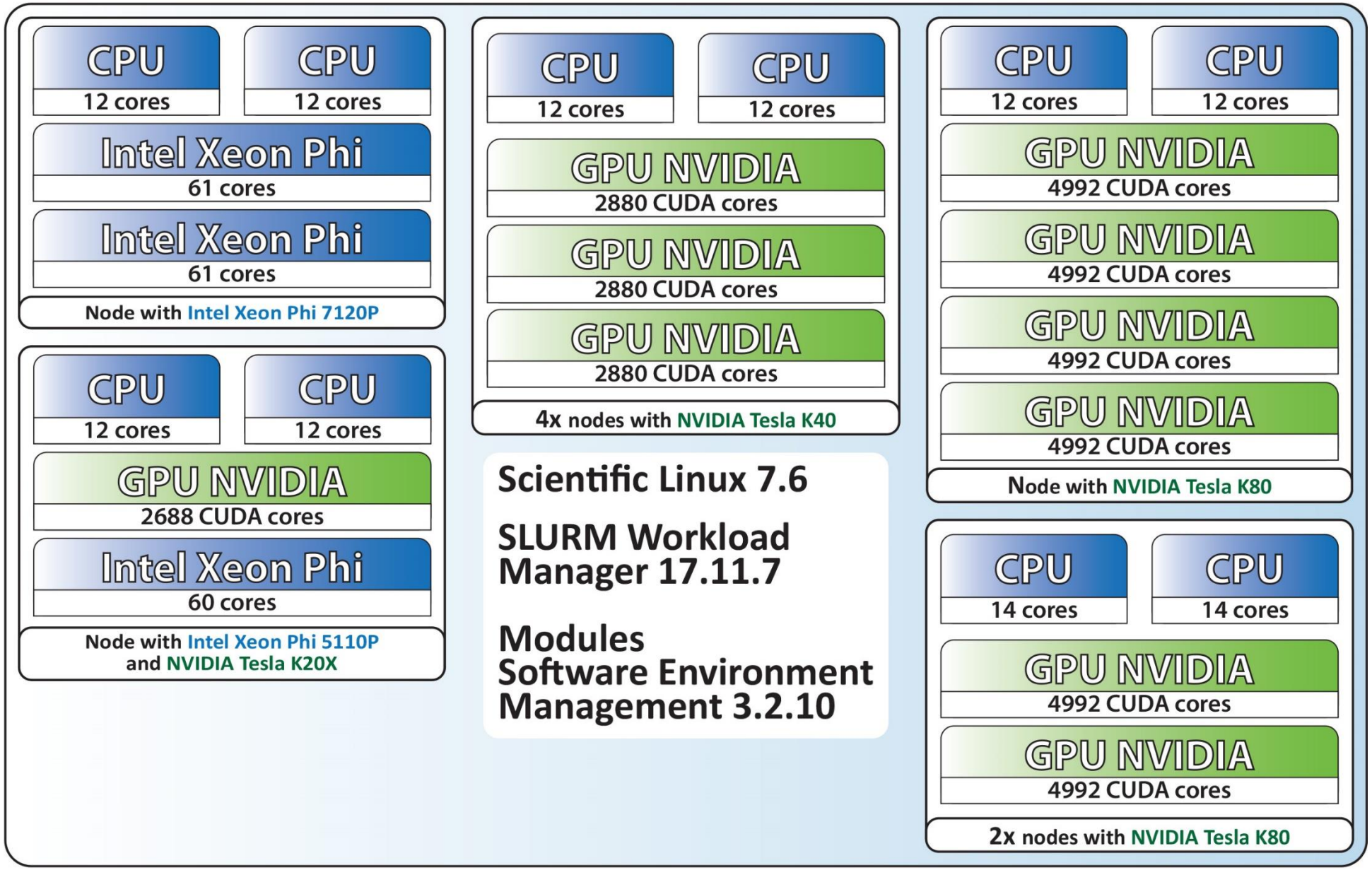
- **Intel® Xeon Phi™ 7190** processors (72 cores)
- **Intel® Server Board S7200AP**
- **Intel®** SSD DC S3520 (SATA, M.2)
- 96GB DDR4 2400 GHz RAM
- **Intel® Omni-Path 100 Gb/s adapter**

RSC Tornado nodes based on Intel® Xeon® Scalable:

- **Intel® Xeon® Gold 6154** processors (18 cores)
- **Intel® Server Board S2600BP**
- **Intel® SSD DC S3520** (SATA, M.2), **2 x Intel® SSD DC P4511 (NVMe, M.2) 1TB**
- 192GB DDR4 2666 GHz
- **Intel® Omni-Path 100 Gb/s adapter**

Hyper-converged system allows to use all Storage nodes as computing ones in parallel with store/retrieve data. This will add 230 TFlops to “Govorun” system, almost doubling the CPU part performance.

Education and testing polygon "HybriLIT"



HybriLIT: GPUs

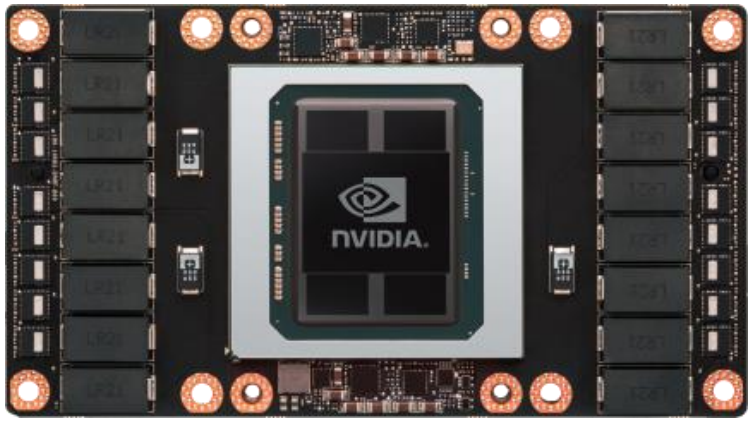
NVIDIA Tesla K40 “Atlas”

2880 CUDA GPU cores
Memory 12 GB GDDR5
Peak performance
- **4.29** TFlops single-precision
- **1.43** TFlops double-precision

NVIDIA Tesla K80

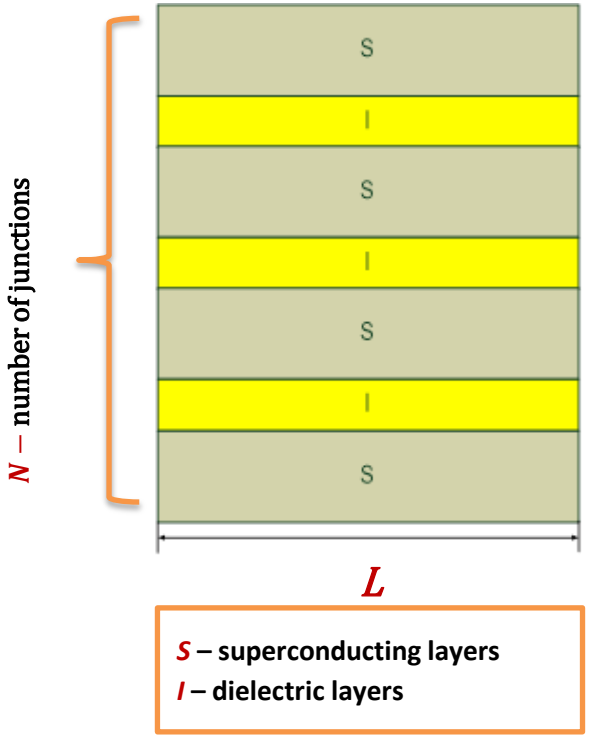
2x Kepler GK210
4992 CUDA GPU cores
Memory 24 GB (**12 GB** per GPU)
Peak performance (with GPU Boost)
- **8.74** TFlops single-precision
- **2.91** TFlops double-precision

NVIDIA Tesla V100



5120 CUDA cores
Memory 16 GB
Peak performance
- **15.7** TFLOPS single-precision
- **7.8** TFLOPS double-precision

Parallel algorithms for studying the system of Long Josephson Junctions



Formulation of the problem. A generalized model that takes into account the inductive and capacitive coupling between long Josephson junctions (LJJ) is considered. The system of coupled LJJ is supposed to consist of superconducting and intermediate dielectric layers with a length L. The phase dynamics of the system N LJJ taking into account the capacitive and inductive coupling between the contacts is described by the initial-boundary problem for the system of differential equations relative to the difference of phases $\varphi_l(x,t)$ and voltage $V_l(x,t)$ at each l contact ($l = 1,2,\cdots N$). In a dimensionless form, the system of equations has the form:

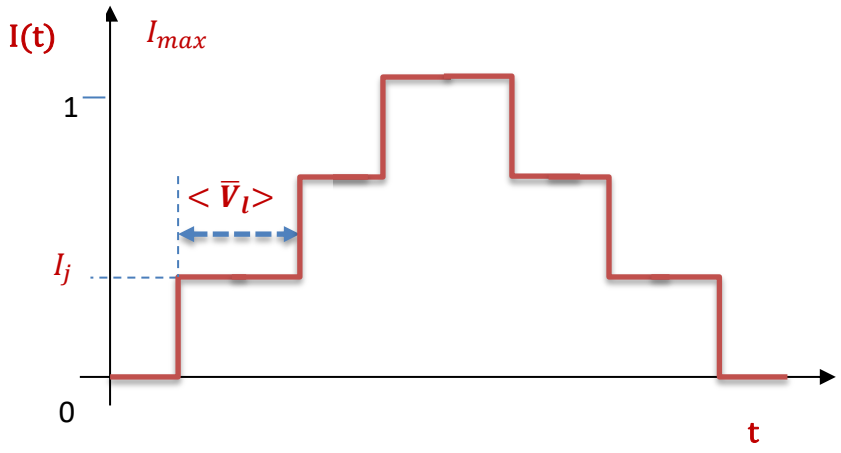
$$\begin{cases} \dfrac{\partial \varphi}{\partial t} = C \cdot V, & 0 < x < L, t > 0 \\ \dfrac{\partial V}{\partial t} = \mathcal{L}^{-1}\dfrac{\partial^2 \varphi}{\partial x^2} - \beta V - \sin(\varphi) + I(t); \end{cases}$$

where $\mathcal{L}$ is the inductive coupling matrix, C is the capacitive coupling matrix:

$$\varphi = \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \cdots \\ \varphi_N \end{pmatrix}, V = \begin{pmatrix} V_1 \\ V_2 \\ \cdots \\ V_N \end{pmatrix}$$

Formulation of the problem

$$L=\begin{pmatrix} 1 & S & 0 & \cdots & & 0 & S \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & S & 1 & S & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ S & 0 & \cdots & 0 & S & 1 & \end{pmatrix},\ C=\begin{pmatrix} D^C & s^C & 0 & \cdots & \cdots & 0 & 0 & s^C \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \cdots & 0 & s^C & D^C & s^C & 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 & s^C & D^C & s^C \\ s^C & 0 & \cdots & 0 & s^C & 0 & s^C & D^C \end{pmatrix},$$



β is the dissipation parameter, the inductive coupling parameter S takes a value in the interval $0<|S|<0.5$; D^C is the effective electric JJ thickness normalized by the thickness of the dielectric layer; s_c is the capacitive coupling parameter, $I(t)$ is the external current.

The system of equations is supplemented with zero initial and boundary conditions:

$$V_l(0,t)=V_l(L,t)=\frac{\partial\varphi_l\,(0,t)}{\partial x}=\frac{\partial\varphi_l(L,t)}{\partial x}=0, l=1,2,\dots,N.$$

The problem when boundary conditions in the direction x were defined by the external magnetic field was also considered.

When calculating the *current-voltage characteristics* (CVC), the dependence of the current on time was selected in a form of steps (a schematic presentation is given in the figure below), i.e. the problem is solved at the constant current ($I=I_j$), while the found functions $\varphi_l, V_l (l=1,2,\dots,N)$ are taken as initial conditions to solve the problem for the current I_{j+1}.

Interaction of a Single Long Josephson Junction with a Ferromagnetic Film

$$\frac{\partial^2\varphi}{\partial^2 t} - \frac{\partial^2\varphi}{\partial^2 x} - \alpha\frac{\partial^3\varphi(x,t)}{\partial^2 x\,\partial t} + \sin(\varphi) + \beta V - I + \frac{dh}{dx} = 0,$$

α - surface dissipation parameter, β - dissipation parameter.

Boundary conditions:

$$\left(\frac{\partial\varphi}{\partial x} + \alpha\frac{\partial^2\varphi}{\partial x\,\partial t}\right)\Big|_{x=0} = h_{ext}\Big|_{x=0}, \qquad \left(\frac{\partial\varphi}{\partial x} + \alpha\frac{\partial^2\varphi}{\partial x\,\partial t}\right)\Big|_{x=L} = h_{ext}\Big|_{x=L},$$

$h_{ext} = h_{F_y}$ – external magnetic field: for the Josephson junction, the source of the magnetic field is the y –-component of the magnetostatic field induced by the inhomogeneous magnetization of a ferromagnet.

The magnetization dynamics of a ferromagnetic film is described by the Landau-Lifshitz-Hilbert equations:

$$\frac{\partial \boldsymbol{m}}{\partial t} = -\frac{\Omega_0}{(1+\alpha_g^2)}\left(\boldsymbol{m}\times\boldsymbol{h}_{eff} + \alpha_g\left[\boldsymbol{m}\times(\boldsymbol{m}\times\boldsymbol{h}_{eff})\right]\right),$$

where $\boldsymbol{m}$ – magnetization vector, Ω_0 – constant characterizing the ratio of the frequency of ferromagnetic resonance to the characteristic frequency of the Josephon junction , α_g – dissipation parameter in a ferromagnet.

Computing and parallel schemes

For the numerical solution of the initial-boundary problem, a uniform grid by the spatial variable (the number of grid nodes NX) was built. In the system of equations (1) the second-order derivative by the coordinate x is approximated using three-point finite-difference formulas on the discrete grid with a uniform step Δx. The obtained system of differential equations relative to the values $\varphi_l, V_l (l = 1,2, \dots, N)$ in the nodes of the discrete grid by x is solved by the fourth-order Runge-Kutta method.

To calculate CVC the averaging $V_l(x,t)$ over the coordinate and time is performed. To do this, at each time step, the integration of voltage over the coordinate using the Simpson method and the averaging are carried out

$$\overline{V_l} = \frac{1}{L}\int_0^L V_l(x,t)dx,$$

then the voltage is averaged over time using the formula

$$< \overline{V_l} >= \frac{1}{T_{max} - T_{min}} \int_{T_{min}}^{T_{max}} \overline{V_l}(t)dt$$

and summed by all JJ. To integrate over time, the rectangle method is used.

Parallel scheme

While numerically solving the initial-boundary problem by the fourth-order Runge-Kutta method over the time variable, at each time layer the Runge-Kutta coefficients (K_i) can be found independently (in parallel) for all NX nodes of the spatial grid and for all N Josephson junctions. Meanwhile the coefficients K_i, $(i = 1,2,3,4)$ are defined one by one (sequentially). Thus, the parallelization is effectively performed on $NX \cdot N$ points. When carrying out the averaging in CVC computing, the calculation of integrals can be performed in parallel as well.

Parallel implementations

To speed up CVC computing, parallel implementations of the computing scheme described above were developed. The results on studying the efficiency of parallel implementations performed at the values of the following parameters: $L = 10; I_{min} = 0;\ I_{max} = 1.1;\ \beta = 0.2;\ N = 1, T_{max} = 200;\ \Delta t = 0.0004$ – are presented below; the number of nodes by the spatial variable is: NX **= 20048.**

OpenMP implementation

The calculations were performed:

- on computing nodes with the processors Intel Xeon Phi 7290 (KNL: 16GB, 1.50 GHz, **72** cores, **4** threads per core supported – total **288** logical cores), the compiler Intel (Intel Cluster Studio 18.0.1 20171018);
- on dual-processor computing nodes with the processors Intel Xeon Gold 6154 (Skylake; 24.75M Cache, 3.00 GHz, **18** cores, **2** threads per core supported – total **72** logical cores per node).

The graphs of the dependence of the calculation **speedup** obtained using the parallel algorithm:

$$S = {}^{T_1}/_{T_n},$$

(where T_1 is the calculation time using one core, T_n is the time of calculations on *n*-logical cores) on the number of threads, the number of which is equal to the number of logical cores, and the graph of the dependence of **efficiency** of using computing cores by the parallel algorithm:

$$E = \frac{T_1}{(n \cdot T_n)} \cdot 100\ \%,$$

characterizing the scalability of the parallel algorithm, are presented below.

OpenMP implementation: Intel Xeon Phi(KNL)

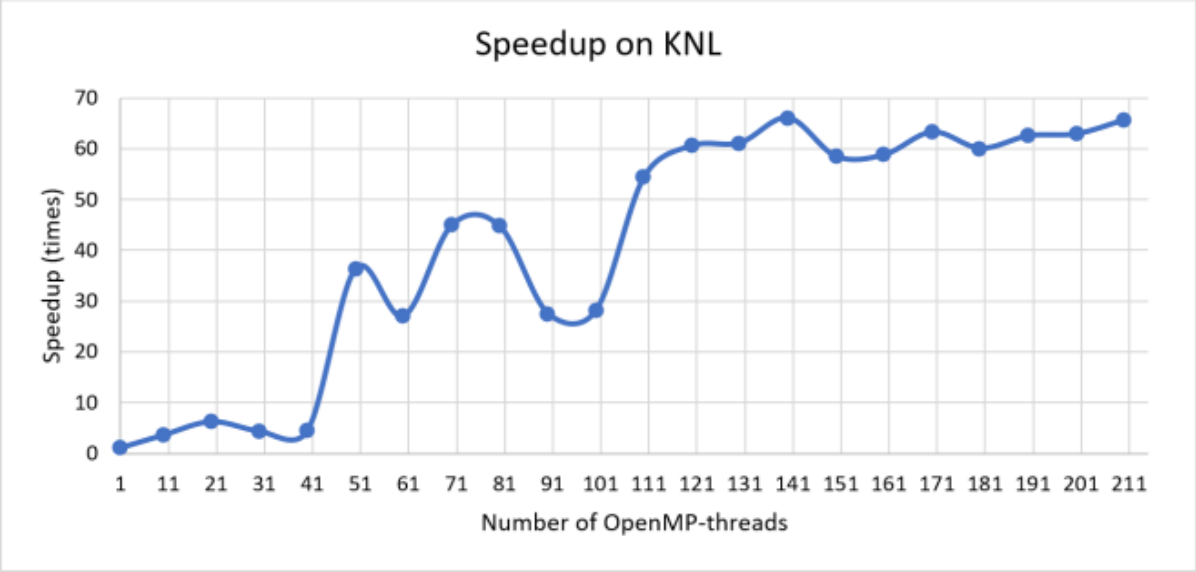


Fig. 1. The dependence of **speedup** of parallel computing on the number of threads.

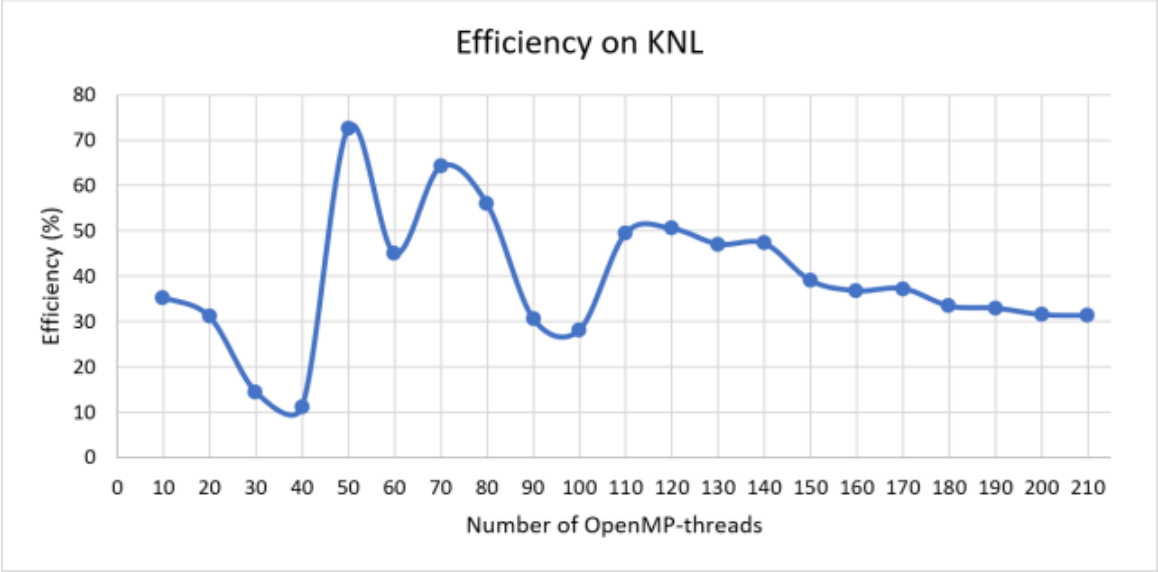


Fig. 2. The dependence of **efficiency** of using computing nodes by the parallel algorithm on the number of threads.

OpenMP implementation: Intel Xeon Phi(KNL)

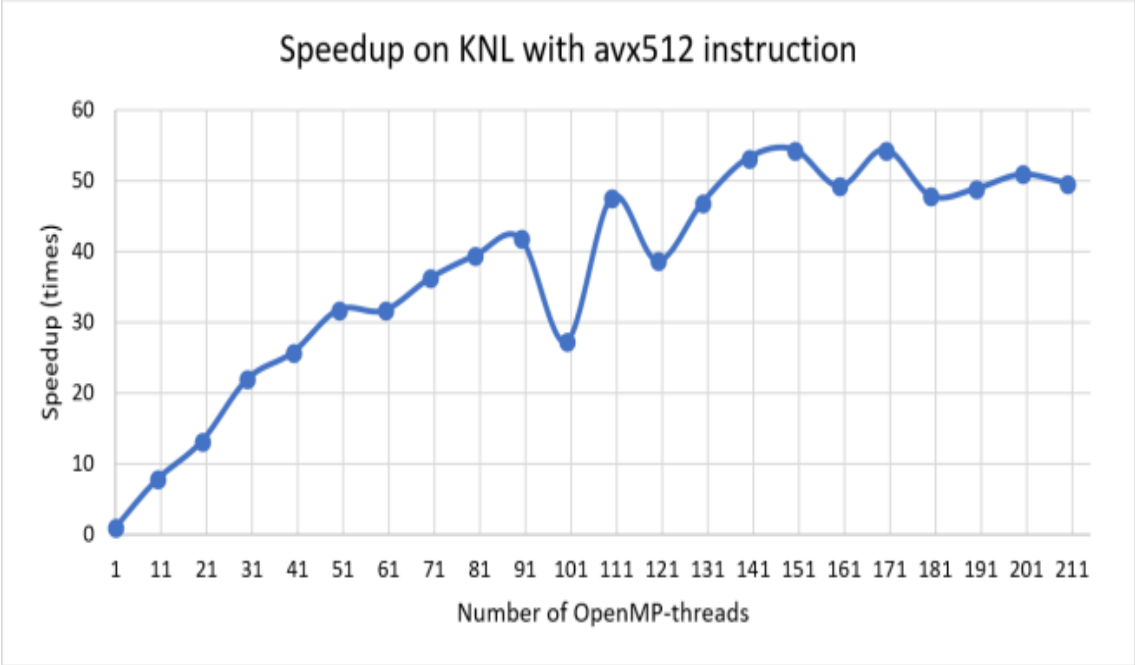


Fig. 3. The dependence of **speedup** of parallel computing on the number of threads. Compilation of the program with the option -xMIC-AVX512.

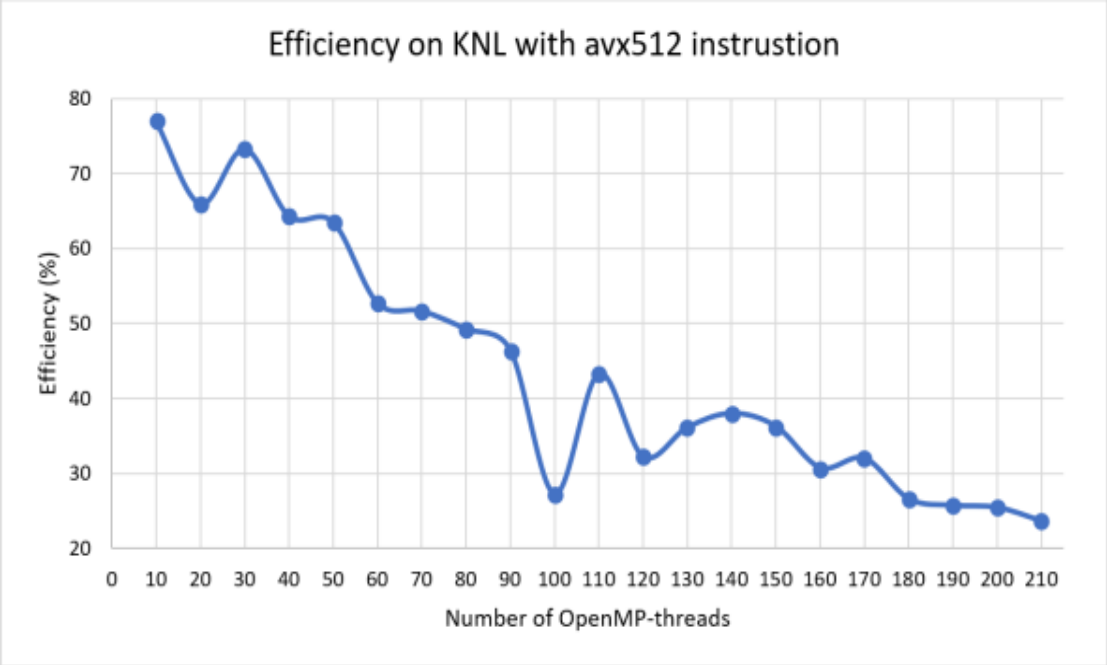


Fig. 4. The dependence of **efficiency** of using computing nodes by the parallel algorithm on the number of threads. Compilation of the program with the option -xMIC-AVX512.

Figure illustrate the dependence of speedup and efficiency when applying the possibility to use the instruction *AVX-512*. It is noteworthy that the use of this instruction allowed us to reduce the calculation time in **1.8** times.

OpenMP implementation: Intel Xeon Gold 6154

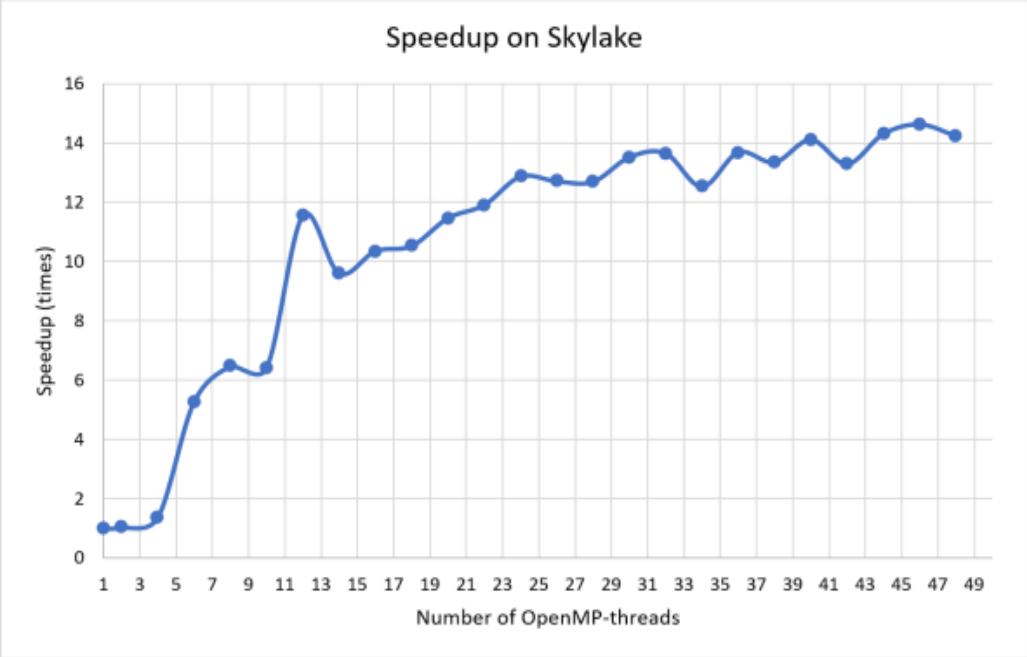


Fig. 5. The dependence of **speedup** of parallel computing on the number of threads.

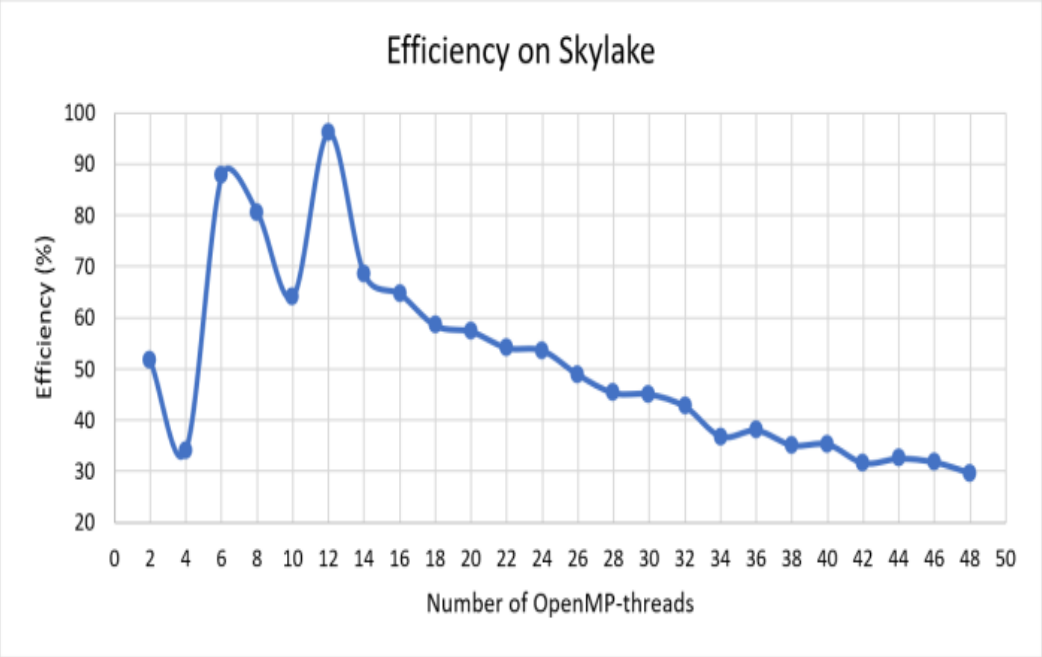
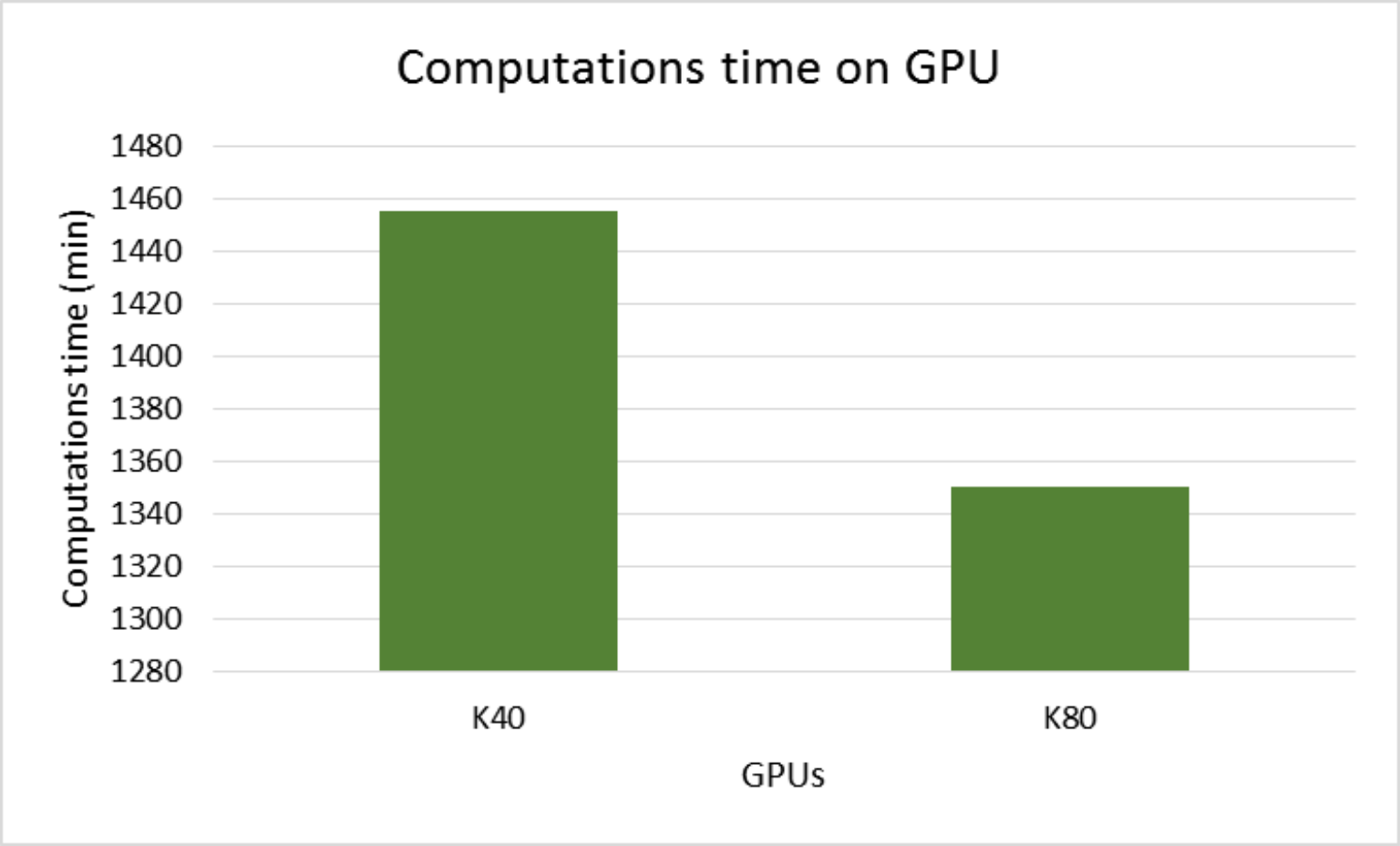


Fig. 6. The dependence of **efficiency** of using computing nodes by the parallel algorithm on the number of threads.

CUDA implementation

A CUDA implementation of the parallel algorithm was developed for computing on NVIDIA graphics accelerators. The parallel reduction algorithm using shared memory was applied for calculating integrals. The calculation time on the graphics accelerators Nvidia Tesla K40 and Nvidia Tesla K80 is presented in the diagram below.



Comparative analysis of parallel implementations

Best (minimum) calculation time obtained on various computing architectures (in minutes) and corresponding speedup of calculations.

KNL		Speedup
1 OpenMP thread	120 OpenMP threads	
148376,2	2450,243	60,55572
KNL with option -xMIC-AVX512		
1 OpenMP thread	150 OpenMP threads	
114647,8	2108,911	54,36352
Skylake		
1 OpenMP thread	44 OpenMP threads	
18464,54	**1290,667**	14,30621

Calculation time (in minutes) on 1 OpenMP thread of the Skylake processor and on the graphics accelerators Nvidia Tesla K40 and Nvidia Tesla K80.

GPU		CPU
Nvidia Tesla K40	Nvidia Tesla K80	Intel Xeon Gold 6154
1455	**1350**	18464,54

Hot results !!!

Intel- CPU implementation

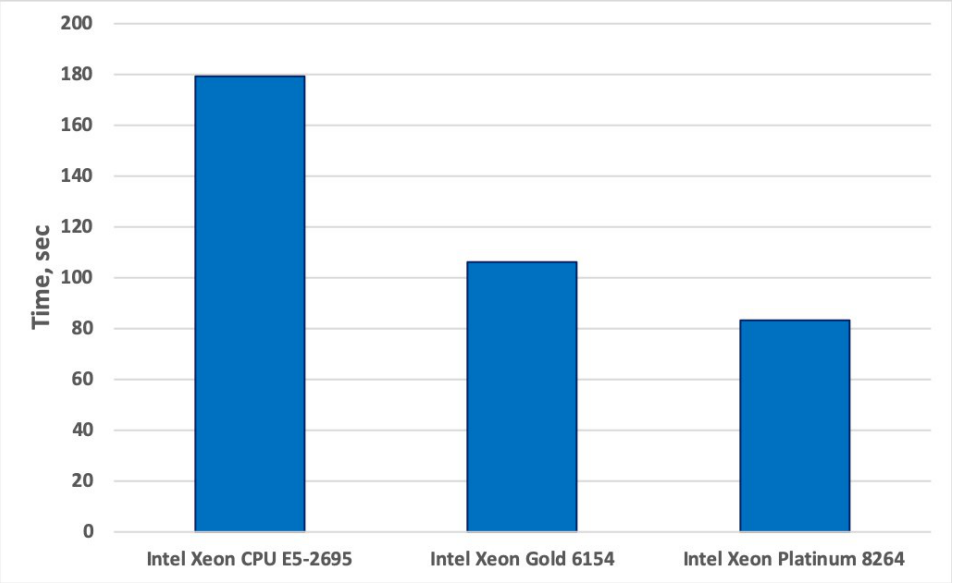
“RSC Tornado” upgrade at JINR (2019)

Intel® Xeon® Scalable gen 2 nodes:

- Peak peformance – **463**ТФЛОПС
- Intel® Xeon® Platinum 8268 processors (24 cores)
- Intel® Server Board S2600BP
- Intel® SSD DC S4510 (SATA, M.2),
  2 x Intel® SSD DC P4511 (NVMe, M.2) 2 Tbytes
- RAM – 192 GB DDR4 2933 ГГц
- Intel® Omni-Path 100 Gbit/s
- 48-port Intel® Omni-Path Edge Switch 100 Series co 100% liquid cooling

Computation time (in sec)

Intel(R) Xeon(R) CPU E5-2695 (18 cores)	Intel® Xeon® Gold 6154 processors (18 cores)	Intel® Xeon® Platinum 8268 processors (24 cores)
179.16	106.23	**83.23**



Thanks for your attention !