Heterogeneous platform “HybriLIT”

The International IT-School

“Machine Learning, Parallel and Hybrid Computations & Big Data Analytics”

in frames of the International Conference

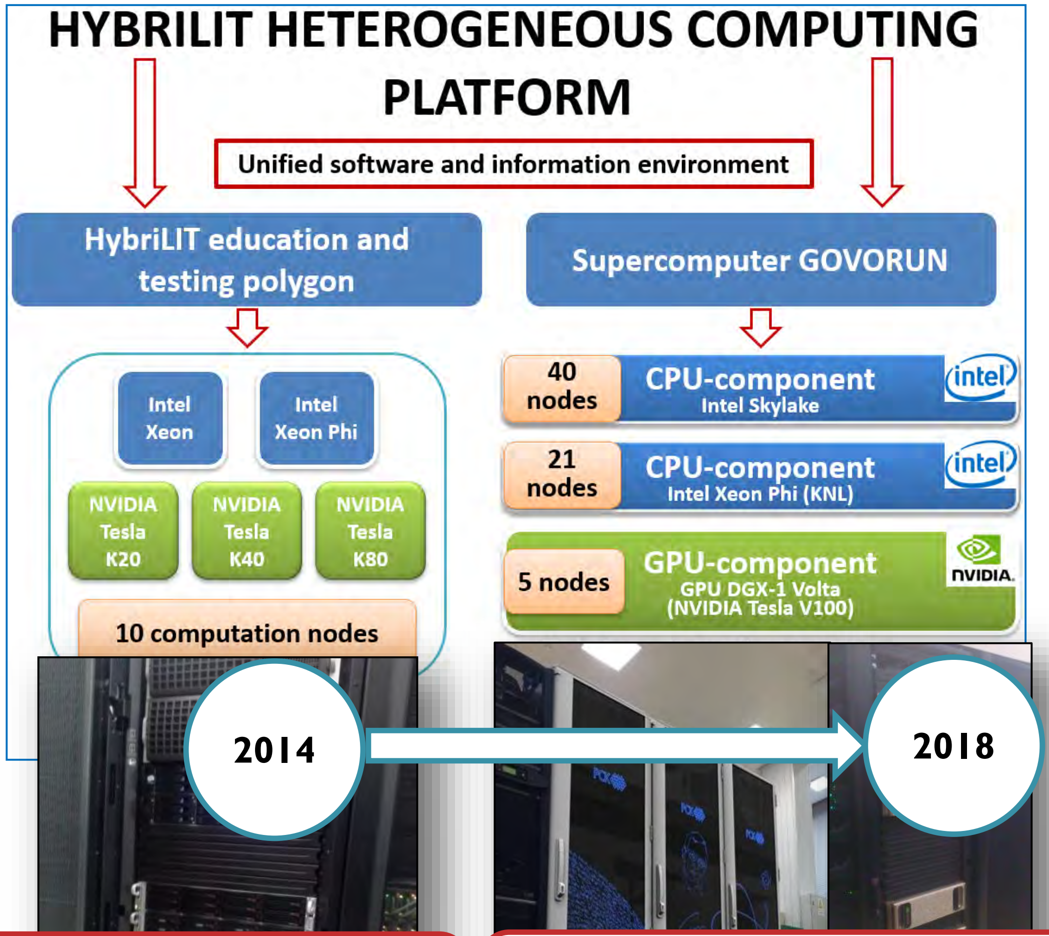
“Mathematical Modeling and Computational Physics, 2019”

Stará Lesná, High Tatra Mountains, Slovakia

01–04 July 2019

Heterogeneous platform “HybriLIT”

Heterogeneous platform HybriLIT



Total peak performance:
140 TFlops for single precision;
50 TFlops for double precision

Total peak performance:
1000 TFlops for single precision;
500 TFlops for double precision

The commissioned supercomputer "Govorun" named after *Nikolai Nikolayevich Govorun*, with whom the development of information technologies at JINR has been connected since 1966, became a natural elaboration of the heterogeneous cluster HybriLIT being the MICC component. The supercomputer "Govorun" is a joint project of the Bogoliubov Laboratory of Theoretical Physics and the Laboratory of Information Technologies supported by the JINR Directorate. The project is aimed at the significant speed-up of complex theoretical and experimental studies in the field of nuclear physics and condensed matter physics underway at JINR including the development of the NICA megaproject computing.

The platform consists of two elements: the education and testing polygon and the supercomputer "Govorun", combined by the unified software and information environment.

Heterogeneous platform "HybriLIT"

Heterogeneous platform “HybriLIT”

The supercomputer “Govorun” components

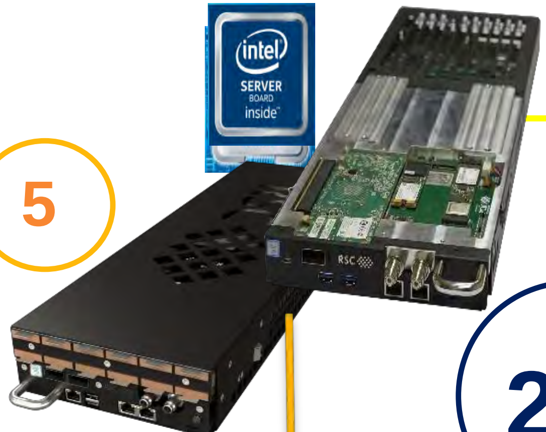
GPU-component based on NVIDIA DGX-1 Volta.

CPU-component based on the newest Intel architectures: Intel Xeon Phi and Intel Skylake processors

The supercomputer “Govorun” is designed to carry out resource-intensive and massive parallel calculations for the solution of a wide range of challenges facing JINR, which becomes possible due to the heterogeneity (presence of different types of computing accelerators) of the supercomputer computing architecture.

At present, the supercomputer is used to solve problems that require massive parallel calculations in various fields of nuclear physics and high energy physics, particularly in lattice quantum chromodynamics to study the properties of hadronic matter at high energy density and baryon charge and in the presence of supramaximal electromagnetic fields, for mathematical modeling of interactions of antiprotons with protons and nuclei using DPM, FTF and UrQMD+SMM generators developed at JINR and being of interest for the NICA-MPD experiment, as well as for solving applied problems.

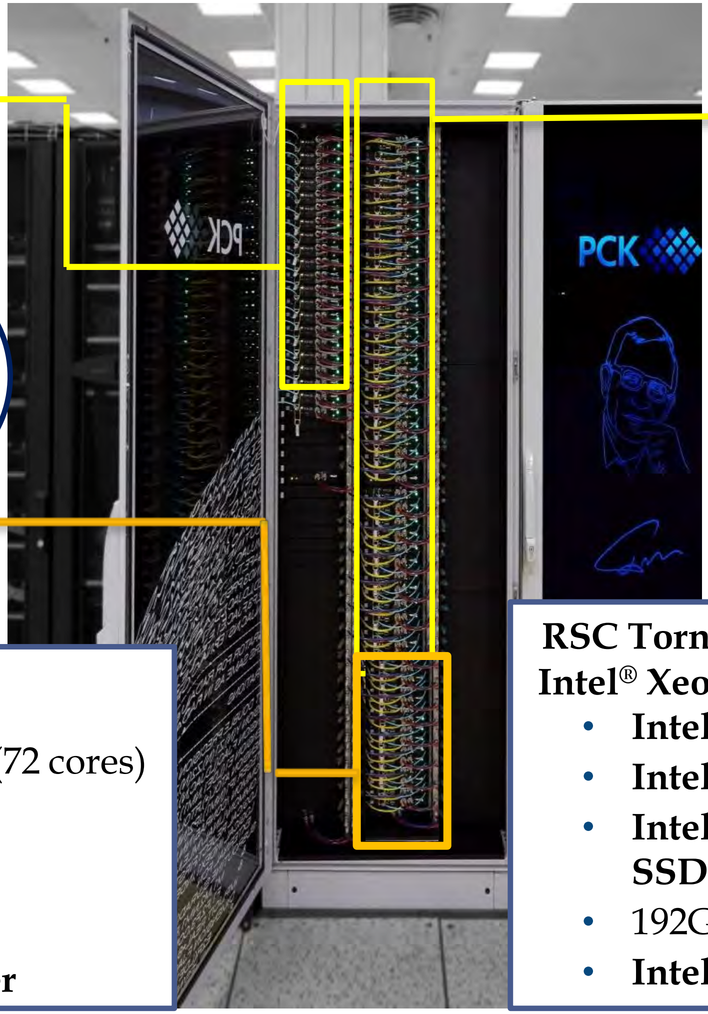
The CPU-component of the supercomputer "Govorun"



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:

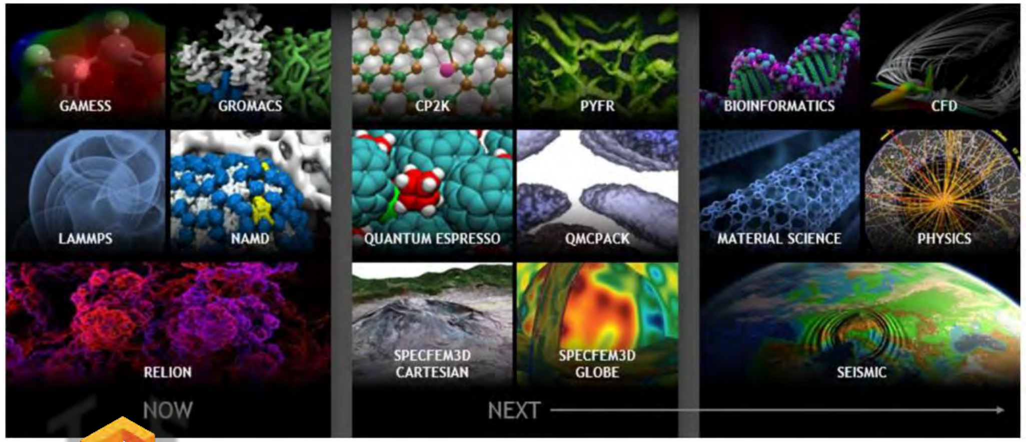
- **Intel® Xeon® Gold 6154** processors (18 cores)
- **Intel® Server Board S2600BP**
- **Intel® SSD DC S3520** (SATA, M.2), **2 x Intel® SSD DC P4511 (NVMe, M.2) 1TB**
- 192GB DDR4 2666 GHz
- **Intel® Omni-Path 100 Gb/s adapter**

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.

Heterogeneous platform "HybriLIT"

The GPU-component of the supercomputer "Govorun"

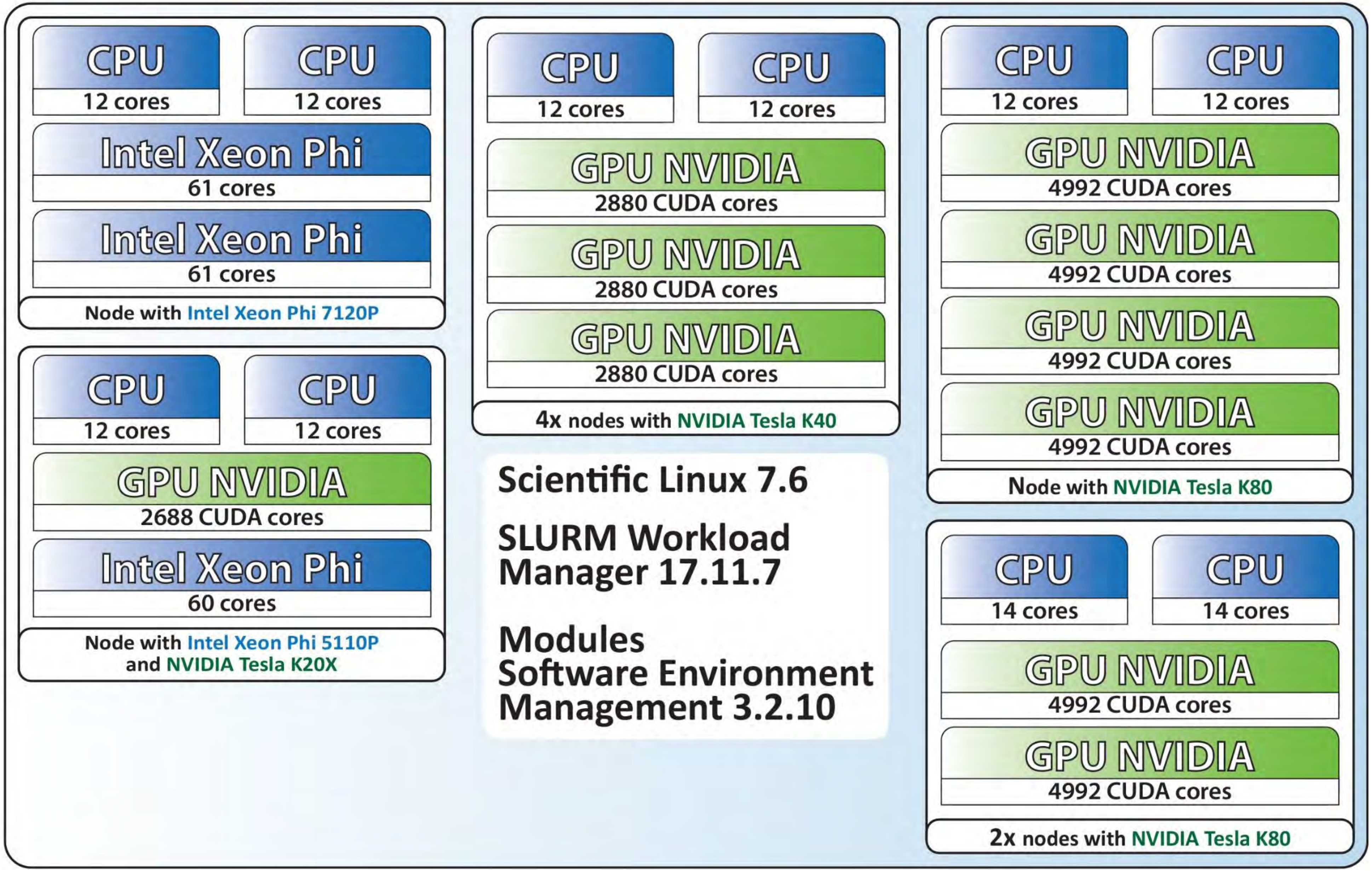
The GPU-component consists of 5 NVIDIA DGX-1 servers. Each server has 8 GPU NVIDIA Tesla V100 based on the latest architecture NVIDIA Volta. Moreover, one server NVIDIA DGX-1 has 40960 cores CUDA, which are equivalent to 800 high-performance central processors. A whole number of novel technologies are used in DGX-1, including the NVLink 2.0 wire with the bandwidth up to 300 Gb/s.



The GPU-component gives a users of the supercomputer a possibility to allow as massively parallel computation for general-purpose tasks using such technologies as CUDA and OpenCL, as well as use applications already adapted for this architecture. Also, GPU-component allow to use machine learning and deep learning algorithms for solving applied problems by neural network approach.

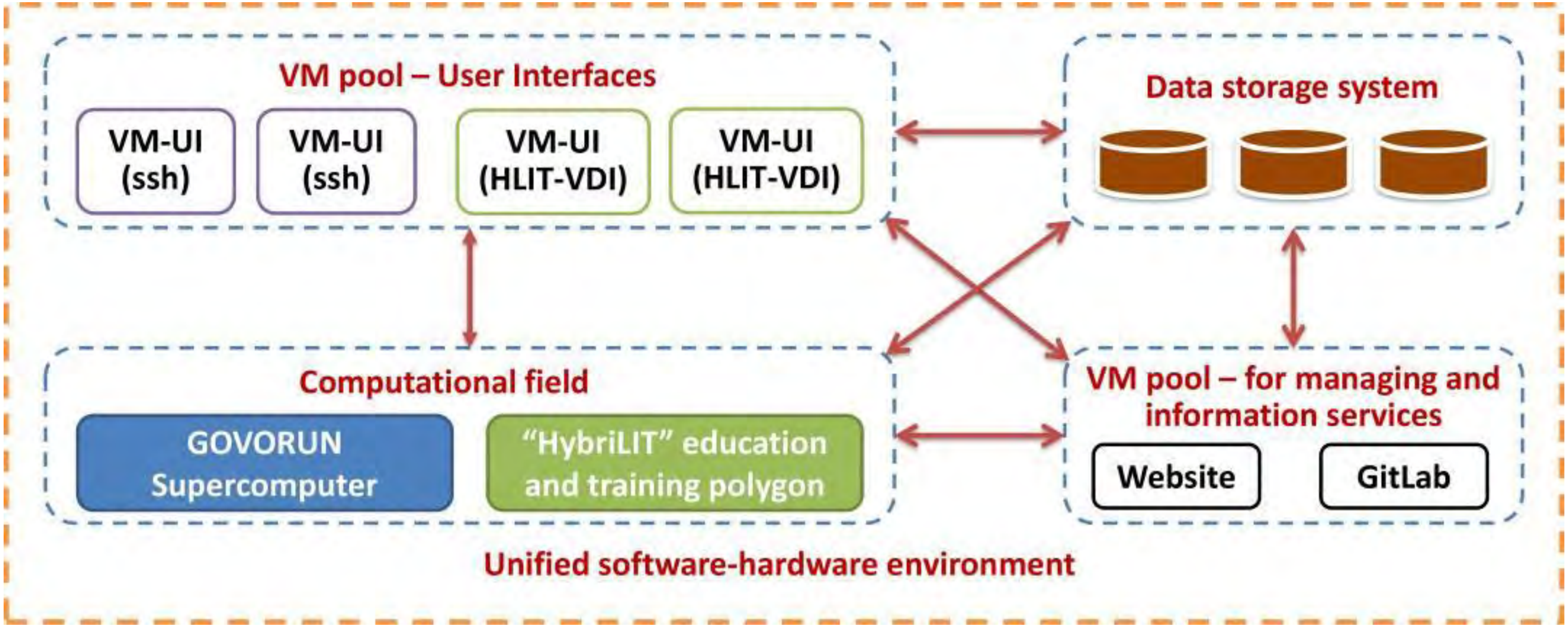
TensorFlow PYTORCH theano APACHE mxnet The Microsoft Cognitive Toolkit Caffe

Education and testing polygon "HybriLIT"



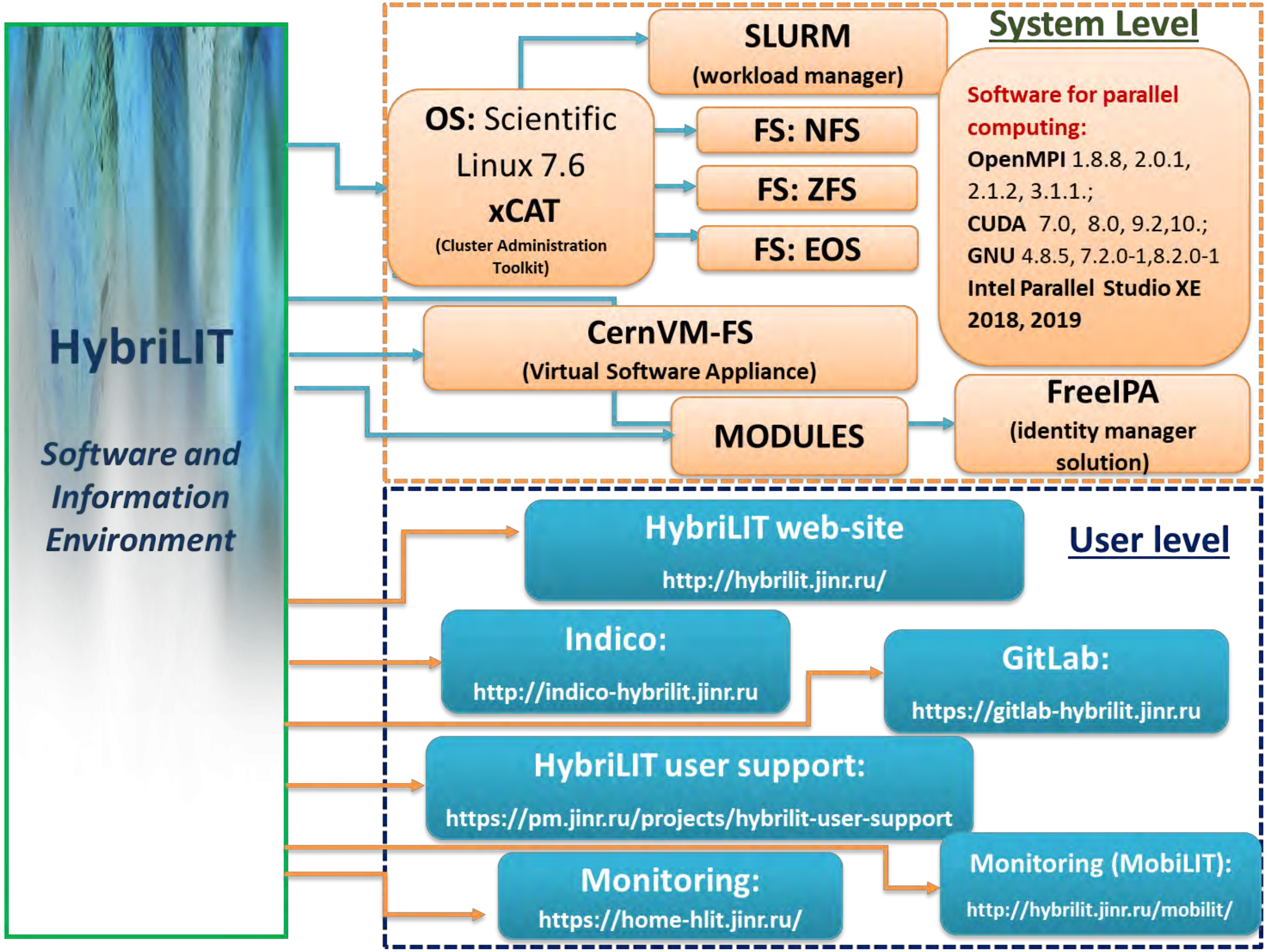
Heterogeneous platform "HybriLIT"

Software and information environment of the HybriLIT



The unified software and information environment of the HybriLIT platform allows users to use the education and testing polygon is aimed at exploring the possibilities of novel computing architectures, IT-solutions, to develop and debug their applications, furthermore, carry out calculations on the supercomputer, which allows them to effectively use the supercomputer resources.

Software and information environment of the HybriLIT

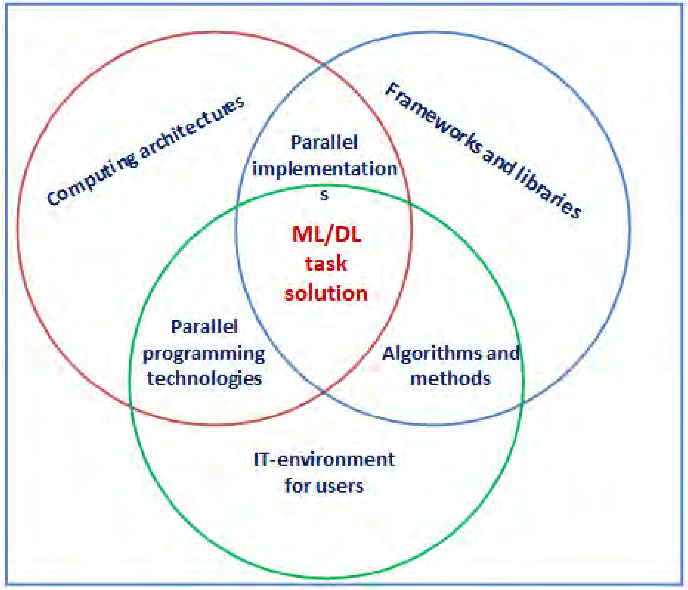


The unified software and information environment including the unified system level (the operation system, the job scheduler, file systems and software) as well as a set of services allowing users to quickly get the answers to their questions, jointly develop parallel applications, receive information about conferences, seminars and meetings dedicated to parallel programming technologies.

Heterogeneous platform "HybriLIT"

Implementation of the neural network approach, methods

To provide all the possibilities both for developing mathematical models and algorithms and carrying out resource-intensive calculations including graphics accelerators, which significantly reduce the calculation time, an ecosystem for tasks of machine learning and deep learning (ML/DL) and data analysis has been created and is actively developing for HybriLIT platform users.

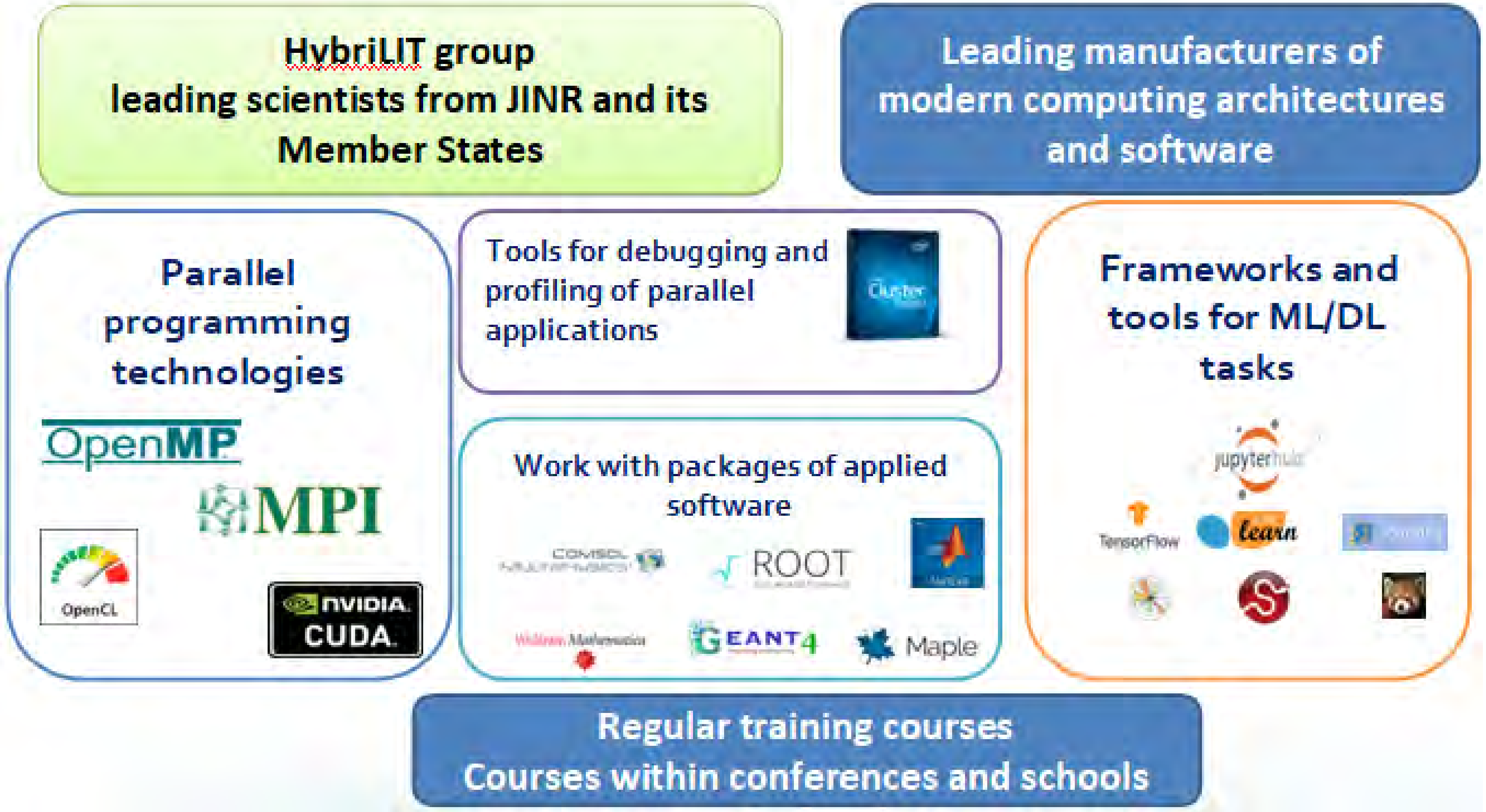


<u>The created ecosystem has two components</u>:

- the computing component is aimed at carrying out resource-intensive, massive parallel tasks of neural network training using NVIDIA graphics accelerators;
- the component for the development of models and algorithms on the *JupyterHub* basis, i.e. a multi-user platform for working with *Jupyter Notebook* (known as *IPython* with the possibility to work in a web browser), including several libraries and frameworks.

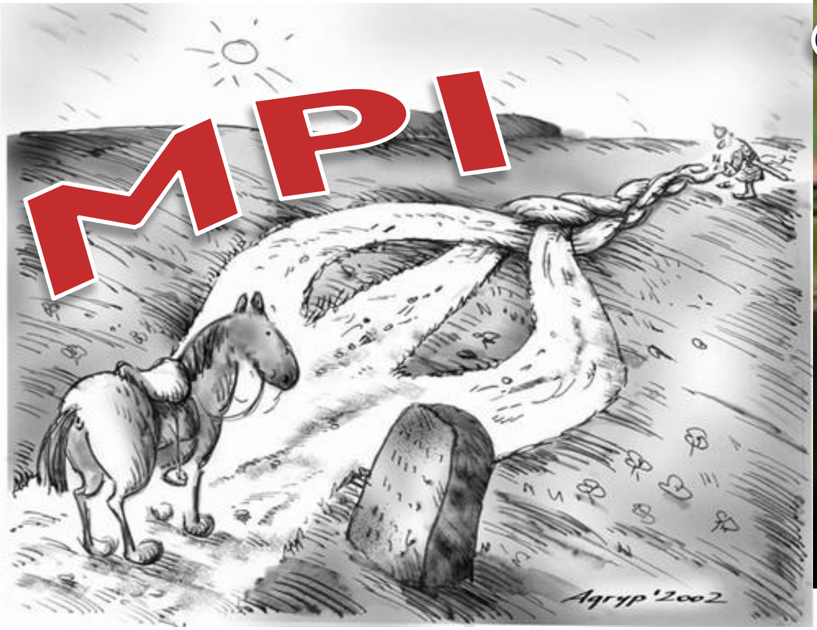
Educational program on the HybriLIT platform

The HybriLIT platform is used as a polygon for training students, post-graduate students and young scientists in the field of HPC. Training is conducted both by young staff members of the Institute and its Member States and by students of Dubna University. The main tool in the educational process is an education and testing polygon which is the basic platform for basic courses on “Computer System Architecture”, “High Performance Computing Technologies” and “Mathematical Models in Physics”; the number of students is about 200 people per academic year.



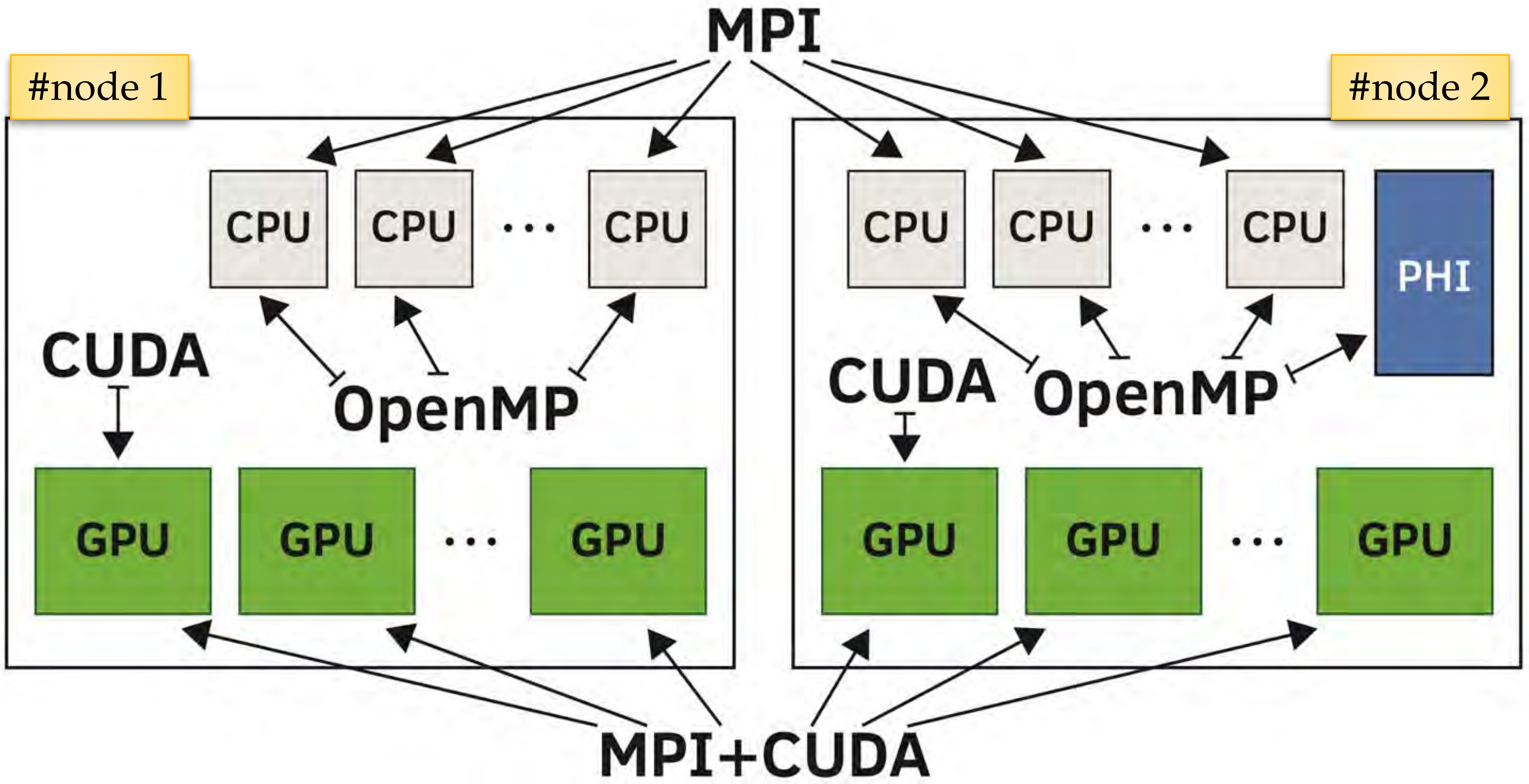
Heterogeneous platform “HybriLIT”

In the last decade novel computational facilities and technologies has become available: MPI – OpenMP – CUDA – OpenCL…



It is not easy to follow modern trends.
Modification of the existing codes or developments of new ones ?

Parallel technologies: levels of parallelism



How to control hybrid hardware:
MPI – OpenMP – CUDA - OpenCL ...

Heterogeneous platform "HybriLIT"

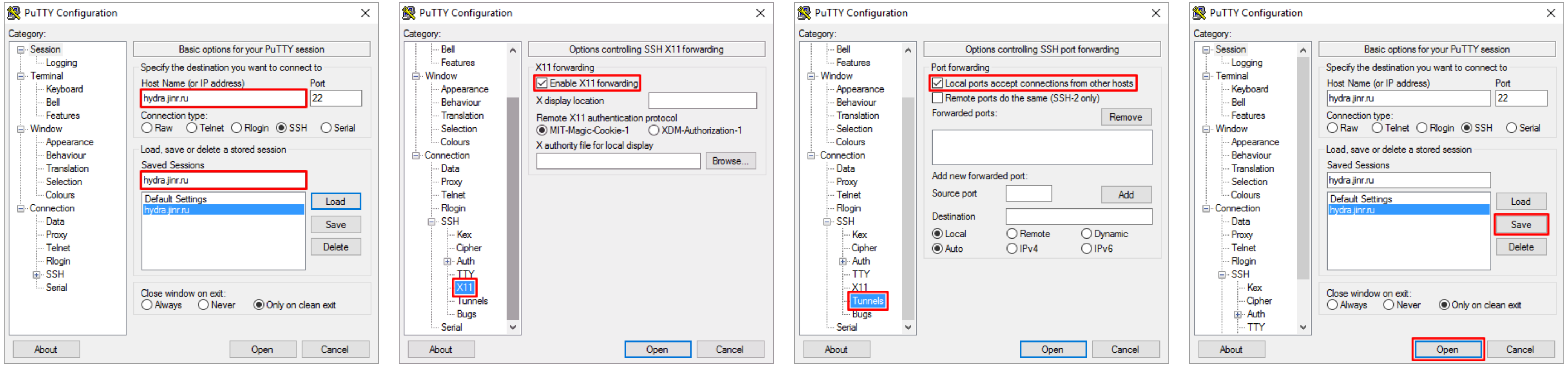
Getting started

Remote access to the cluster

For Windows users:

In order to connect from the Windows OS, it is necessary to use a special program – **SSH**-client, e.g. **PuTTY**.

1. To install PuTTY, it is necessary to copy **putty.exe** file to your computer and launch it.
2. Enter **hydra.jinr.ru** in the field **Host Name (or IP address)**
   Enter the preferable name for the current connection in the field **Saved Sessions**
3. To support remote graphical interface, choose **Connection > SSH > X11** and mark it in the field **Enable X11 forwarding**
4. Check if the field Local ports accept connection from other hosts is marked in **Connection > SSH > Tunnels**
5. After the chosen settings go back to the **Sessions** and click **Save** in order to save the current settings.
6. Click *Open* to connect HybriLIT and enter login and password.



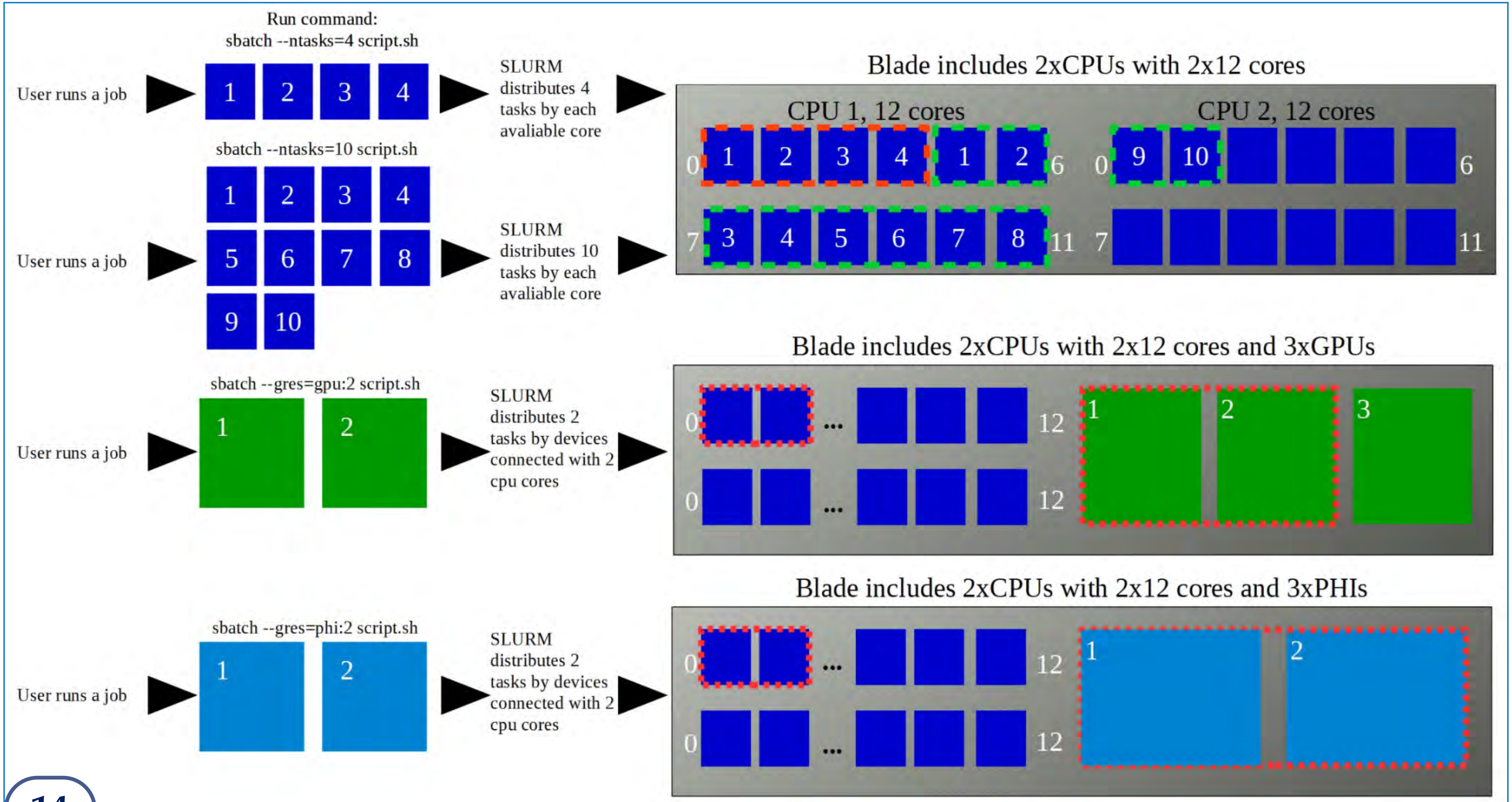
Remote access to the cluster for Linux users:

Launch the terminal, type in the command line:

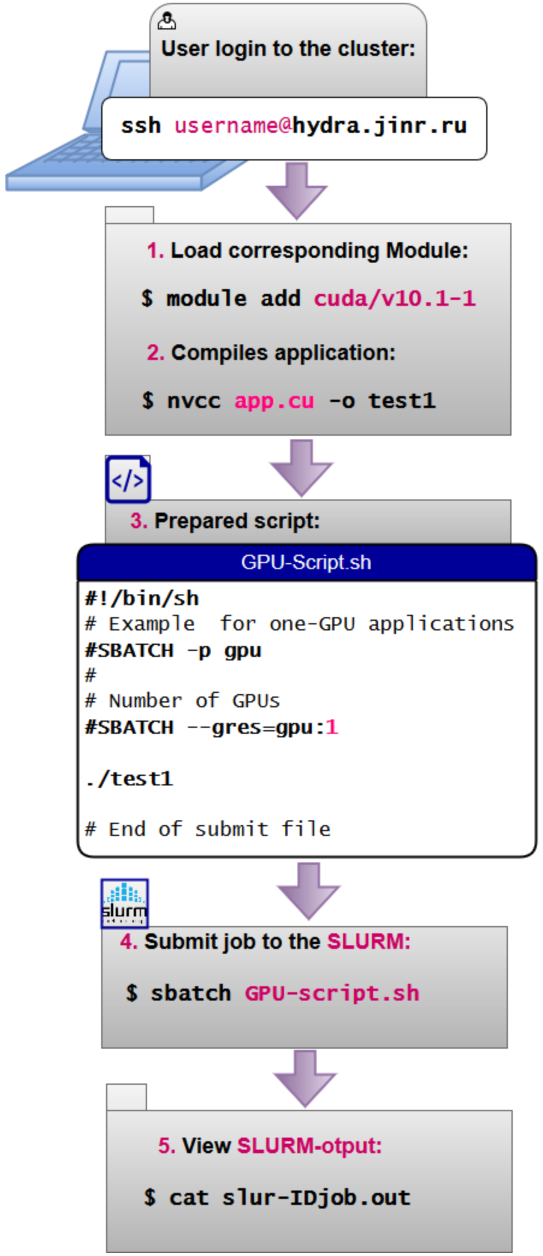
```
$ ssh username@hydra.jinr.ru
```

where **username** is your login

SLURM workload manager



5 basic steps to carry out computations on the platform



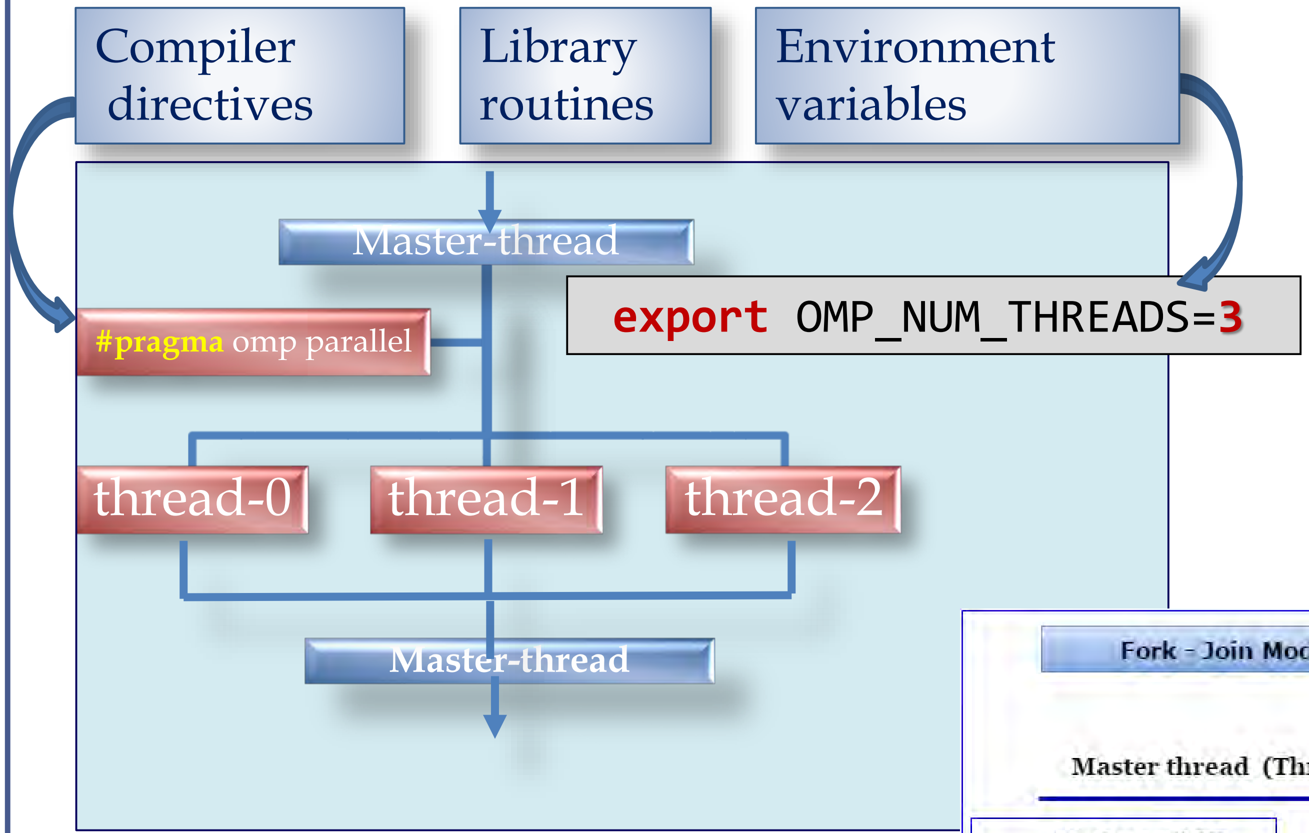
The list of main commands in the work with the modules:

Module commands	Description
`module avail`	list available modules
`module add <name of the module>`	load chosen module
`module list`	lists currently loaded modules
`module rm <name of the module>`	unloads module foo and removes all changes that it made in the environment
`module clear`	unloads all modules

The list of main SLURM commands:

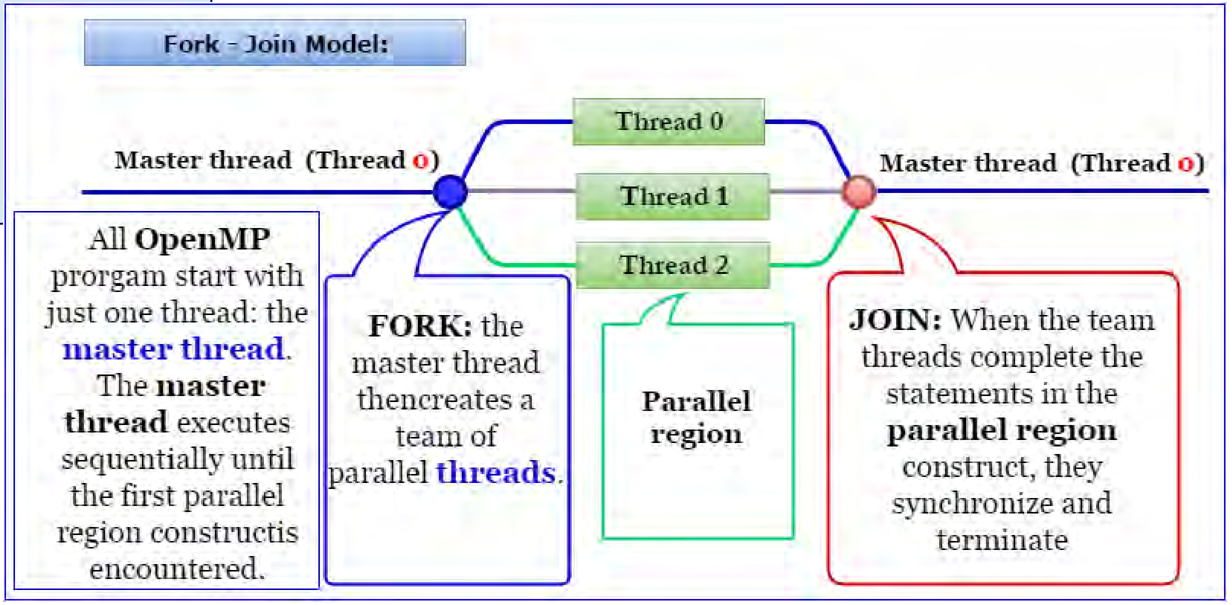
SLURM commands	Description
`sinfo`	reports the state of partitions and nodes managed by SLURM
`squeue`	reports the state of jobs
`sbatch <name of the script>`	submit a job script for later execution (the script typically contains one or more srun commands to launch parallel tasks)
`scancel`	cancel a pending or running job

OpenMP parallel programming technology



OpenMP (**Open** specifications for **M**ulti-**P**rocessing) is an API that supports multi-platform shared memory multiprocessing programming in **Fortran, C, C++.**

OpenMP Programming Model



OpenMP is an explicit (not automatic) programming model, offering the programmer full control over parallelization.

OpenMP parallel programming technology: example

Task: find the sum of two arrays $C = A + B$, the elements of which are determined by some functions.

Step 1.

$$\mathbf{A}: a_i = f(i) = \sin(81 \cdot i + 6) - 10\cos(15 \cdot i), \quad i = 0 \dots N;$$

$$\mathbf{B}: b_i = g(i) = \sin(12 \cdot i + 3) - 10\cos(13 \cdot i), \quad i = 0 \dots N;$$

Step 2.

$$\mathbf{C=A+B}: c_i = a_i + b_i, \quad i = 0 \dots N;$$

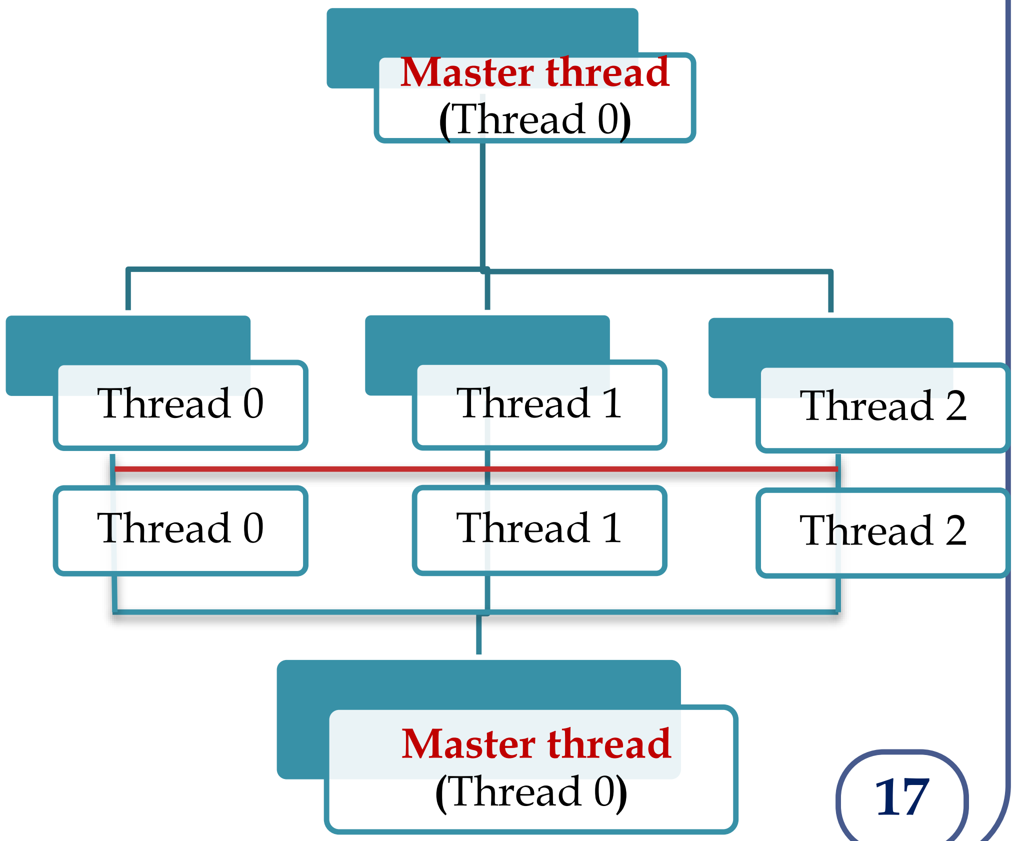
```
#include <stdio.h>
#include<math.h>
#include <omp.h>
#define Nvec  8000000
double f(int j);
double g(int j);
int main (int argc, char *argv[]) {
     double* A; double* B; double* C;
     A = (double*) malloc(Nvec*sizeof(double));
     B = (double*) malloc(Nvec*sizeof(double));
     C = (double*) malloc(Nvec*sizeof(double));
   // Step 1: initialization of A and B
#pragma omp parallel
{
    #pragma omp for
    for (i = 0; i < N; i++) {
           A[i] = f(i);
           B[i] = g(i);
     }
   //Step 2: C= A + B
    #pragma omp for
    for (i = 0; i < N; i++) {
           C[i] = A[i]+B[i]
    }
}
  free(A); free(B); free(C);
  return 0;
}
```

Compiler directive

Compiler directive

Compiler directive

task1-openmp.c



Compilation and launch of OpenMP-applications

```
$ module add intel/v2019.0.018
$ icc -qopenmp solver_Openmp.c -mkl -o test1
$ sbatch script_omp.sh
```

script_omp.sh

```
#!/bin/sh
# Example with 5 cores for OpenMP
#SBATCH -p tut
# Number of cores per task
#SBATCH -c 5
#Set OMP_NUM_THREADS to the same# value as -c
if [ -n "$SLURM_CPUS_PER_TASK" ]; then
   omp_threads=$SLURM_CPUS_PER_TASK
   else    omp_threads=1
fi
export OMP_NUM_THREADS=$omp_threads
export OMP_DISPLAY_ENV=TRUE
    ./test1
# End of submit file
```

Heterogeneous platform "HybriLIT"

Heterogeneous platform "HybriLIT"

MPI parallel programming technology

MPI: Message Passing Interface

Data transfer operation:

- MPI is based on data transfer operations.
- Among MPI functions there are:
  - *point-to-point* operations between two processes,
  - *collective* communication actions for simultaneous interaction of several processes.

```
#include <stdio.h>
#include <mpi.h>

 int main(int argc, char **argv){
     int  size;
     int rank;

    MPI_Init(&argc,&argv);
    MPI_Comm_size (MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 printf(“I am process number:%d form %d\n", rank, size);

MPI_Finalize();

return 0;}
```

MPI: Concepts

- **Processes**
- **Group of processes**
- **Communicator**

Each process in a communicator is identified by its **rank**

MPI COMM WORLD – name of the default MPI communicator:
MPI_Comm_size (MPI_COMM_WORLD, &**size**);
MPI_Comm_rank(MPI_COMM_WORLD, &**rank**);

MPI parallel programming technology: example

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char **argv){
   int  size;
   int rank;

   MPI_Init(&argc,&argv);

   MPI_Comm_size (MPI_COMM_WORLD, &size);

   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   printf("Hello world!  I am process number: %d form %d\n", rank, size);

   MPI_Finalize();

   return 0;
}
```

Initializes the MPI execution environment

How many processes?

The rank value is between 0 and size-1

Terminate the MPI computation and clean up

MPI parallel programming technology

rank = 0

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char* argv[]){
    int numprocs, rank;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    printf("Numprocs is %d; Hello from %d\n",
                                numprocs, rank);
    MPI_Finalize();
}
```

```
Numprocs is 6; Hello from 0
Numprocs is 6; Hello from 1
Numprocs is 6; Hello from 5
Numprocs is 6; Hello from 3
Numprocs is 6; Hello from 4
Numprocs is 6; Hello from 2
```

rank = 0

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char* argv[]){
    int numprocs, rank;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,
                        &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,
                            &rank);
    printf("Numprocs is %d; Hello
       from %d\n", numprocs, rank);
    MPI_Finalize();
}
```

rank = 1

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char* argv[]){
    int numprocs, rank;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,
                        &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,
                            &rank);
    printf("Numprocs is %d; Hello
       from %d\n", numprocs, rank);
    MPI_Finalize();
}
```

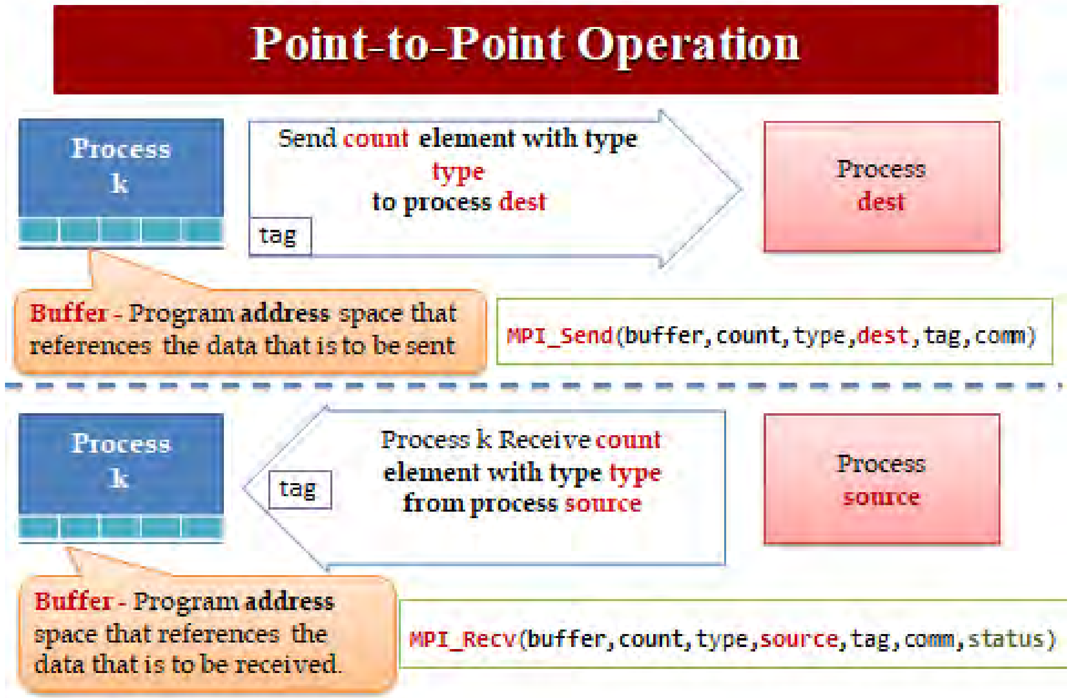
rank = numprocs - 1

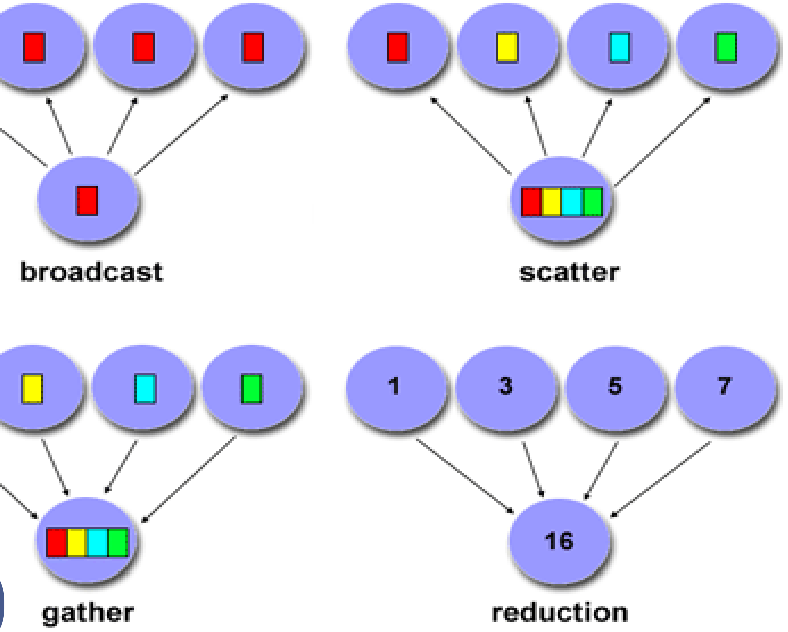
```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char* argv[]){
    int numprocs, rank;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,
                        &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,
                            &rank);
    printf("Numprocs is %d; Hello
       from %d\n", numprocs, rank);
    MPI_Finalize();
}
```

MPI parallel programming technology

The 6 most important MPI commands:

`MPI_Init`() – initiate a MPI computation
`MPI_Comm_size`() – how many processes participate in a given MPI communicator?
`MPI_Comm_rank`() – which one am I? (A number between 0 and size-1)
`MPI_Send`() – send a message
MPI_Recv() – receive a message
`MPI_Finalize`() – terminate the MPI computation





Collective Communication

`MPI_Barrier`() – barrier synchronization
`MPI_Bcast`() – broadcasting
`MPI_Gather` ()– gathering of data from all process to the master process
`MPI_Scatter`() – scattering of data from master process to all processes in group
`MPI_Reduce`(), `MPI_AllReduce`() – global operations (*MPI_MIN, MPI_MAX, MPI_SUM, MPI_PROD* etc)

Intel: compilers, tools for developing, debugging and profiling parallel applications, mathematical library …

Intel Inspector
Memory and Threading Checking

Intel Aplication Performance Snapshot
Tool for application profiling

Intel Advisor
Vectorization Optimization and Thread Prototyping

Intel Math Kernel Library
Optimized Routines for Science, Engineering, and Financial

Intel Data Analytics Acceleration Library
Optimized for Data Analytics & Machine Learning

Intel® Integrated Performance Primitives
Image, Signal, and Compression Routines

Intel Threading Building Blocks
Task-Based Parallel C++ Template Library

Source: https://software.intel.com/en-us/articles/intel-parallel-studio-xe-release-notes

Intel Math Kernel Library.
LAPACK : Solving Systems of Linear Equations

$$\boldsymbol{A} \cdot Y = B$$

$$\begin{pmatrix} \boldsymbol{D} & \boldsymbol{U} & 0 & 0 & 0 & 0 \\ \boldsymbol{L} & \boldsymbol{D} & \boldsymbol{U} & 0 & 0 & 0 \\ 0 & \boldsymbol{L} & \boldsymbol{D} & \boldsymbol{U} & 0 & 0 \\ 0 & 0 & \boldsymbol{L} & \boldsymbol{D} & \boldsymbol{U} & 0 \\ 0 & 0 & 0 & \boldsymbol{L} & \boldsymbol{D} & \boldsymbol{U} \\ 0 & 0 & 0 & 0 & \boldsymbol{L} & \boldsymbol{D} \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

?gtsv
Computes the solution to the system of linear equations with a tridiagonal coefficient matrix A and multiple right-hand sides.

```
lapack_int LAPACKE_dgtsv (
int matrix_layout ,
lapack_int n ,
lapack_int nrhs ,
double * dl ,
double * d ,
double * du ,
 double * b ,
 lapack_int ldb );
```

Compilation

```
$ module add intel/v2019.0.018
$ icc tridiagonal_solver.c -mkl -qopenmp -o test1
```


https://software.intel.com/en-us/mkl-developer-reference-c-lapack-linear-equation-driver-routines

CUDA programming model

Definitions: ***device*** = GPU; *host* = CPU; ***kernel*** = function that runs on the device

Parallel parts of an application are executed on the device as ***kernels***

- One ***kernel*** is executed at a time
- Several threads execute each ***kernel***

```
kernel <<< dim3 dG, dim3 dB, nSgMem, nStream  >>> ( args )
```

- dim3 **dG** – number of blocks dimension (grid)
- dim3 **dB** – dimension of blocks
- **nShMem** – the amount of shared memory to be allocated for each block
- **nStream** – the stream number

dim3 is an integer vector type that can be used in CUDA code.

```
dim3 dG(512);  // 512x1x1
dim3 dB(1024, 1024);  // 1024x1024x1
```

Function Type Qualifiers

Declaration specifier	Callable from	Executed on
`__global__`	host	device
`__host__`	host	host
`__device__`	device	device

CUDA application: GPU vectors sum

$$c_i = a_i + b_i, i = 0, \dots, N-1$$

Serial code

```
void sum(int N, int *a, int *b,
                         int *c){
   int tid = 0;
   while(tid < N){
      c[tid] = a[tid] + b[tid];
      tid += 1;
   }
}
```

CUDA code

```
__global__ void kernel(int N, int *a,
                         int *b, int *c){
   int tid = threadIdx.x + blockIdx.x *
                              blockDim.x;
   while(tid < N){
      c[tid] = a[tid] + b[tid];
      tid += blockDim.x * gridDim.x;
   }
}
```

Compilation and execution CUDA applications

Add module

$ module add cuda/v10.1-1

Compilation

$ nvcc -arch=compute_35 deviceInfo.cu -o test1

script_CUDA.sh

```
#!/bin/sh
#SBATCH -p tut
# Number of GPU per node
#SBATCH --gres=gpu:1
# Set time of work (in minutes)
#SBATCH -t 60
# Connect module for CUDA
module add cuda/v10.1-1
./test1
# End of submit file
```

Run in batch mode

$ sbatch script_CUDA

Ecosystem for ML/DL, data analysis tasks (basic libraries and tools)

scikit-learn – machine learning library for Python programming language.

NumPy – is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object sophisticated (broadcasting) functions tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities

SciPy – is a Python-based ecosystem of open-source software for mathematics, science, and engineering. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

matplotlib – is a Python 2D plotting library which produces publication quality figures

pandas – Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.

TensorFlow – is a free and open-source a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

Ecosystem for ML/DL tasks

Development component

VM with JupyterHub
https://jhub.jinr.ru

VM:
CPU: 24 Cores
RAM: 32 GB

Computation component

Servers with NVIDIA Volta & Intel Xeon Gold
https://jhub2.jinr.ru

Dell Volta specs:

GPU 4x Nvidia Volta V100-SXM2 *NVLink* 32Gb HBM2
CPU 2x Intel Xeon Gold 6148 CPU @ 2.40GHz 20 Cores / 40 Threads
RAM 512 GB DDR4 2666MHz
SSD 2*240 GB

Heterogeneous platform "HybriLIT"

Ecosystem for tasks of machine learning, deep learning and data analysis

Work within the ML/DL ecosystem

To get started you need to:

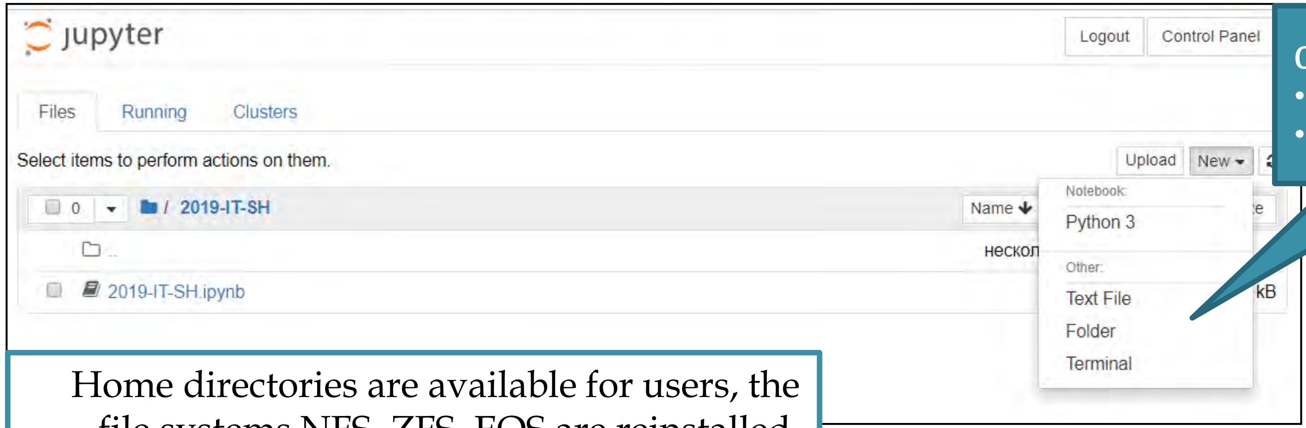
1. Log in with your HybriLIT account in GitLab: https://gitlab-hybrilit.jinr.ru/
2. Enter the components (the authorization is done via GitLab):

GitLab

Development component (without graphics accelerators)	Component for carrying out resource-intensive calculations (with graphics accelerators NVIDIA) NVIDIA
https://jhub.jinr.ru	**https://jhub2.jinr.ru**

Jupyter Notebook

After the authorization the **Jupyter Notebook** interactive environment opens:



Create:
- **Folder:**
- **File Python 3**

Home directories are available for users, the file systems NFS, ZFS, EOS are reinstalled.

SLURM submission script

Submission script for MPI application (Intel realization)

```
#!/bin/sh
#Partition for run;
#SBATCH -p tut
#Number of MPI tasks;
#SBATCH -n 8
#Set time of work (in minutes);
#SBATCH -t 60
#Connect module for Intel MPI
module add intel/v2019.0.018
#Submit a job for execution;
mpirun ./exec_mpi
```

Submission script for MPI application (GNU realization)

```
#!/bin/sh
#Partition for run;
#SBATCH -p tut
#Number of MPI tasks;
#SBATCH -n 8
#Set time of work (in minutes);
#SBATCH -t 60
#Connect module for OpenMPI
module add openmpi/v3.1.3
#Submit a job for execution;
mpirun ./exec_mpi
```

Submission script for OpenMP application

```
#!/bin/sh
#Partition for run;
#SBATCH -p tut
#Number of cores per task;
#SBATCH -c 5
#Number of MPI tasks;
#SBATCH -n 1
#Set OMP_NUM_THREADS to the same value as -c;
export OMP_NUM_THREADS=5
#Set time of work (in minutes);
#SBATCH -t 60
#Submit a job for execution;
./exec_openmp
```

Submission script for CUDA and application

```
#!/bin/sh
#Partition for run;
#SBATCH -p tut
#Number of GPU per node;
#SBATCH –gres=gpu:1
#Set time of work (in minutes);
#SBATCH -t 60
#Connect module for CUDA
module add cuda/v10.1-1
#Submit a job for execution;
./exec_cuda
```

JOINT INSTITUTE FOR NUCLEAR RESEARCH
http://jinr.ru
141980, Dubna, Moscow region, Joliot-Curie 6

LABORATORY OF INFORMATION TECHNOLOGIES
http://lit.jinr.ru
Phone: (+7 49621) 64-019
Fax: (+7 49621) 65-145

MULTIFUNCTIONAL INFORMATION AND COMPUTING COMPLEX
https://micc.jinr.ru

HETEROGENEOUS PLATFORM HYBRILIT IN LIT JINR
http://hlit.jinr.ru