Using of the ITEP supercomputer

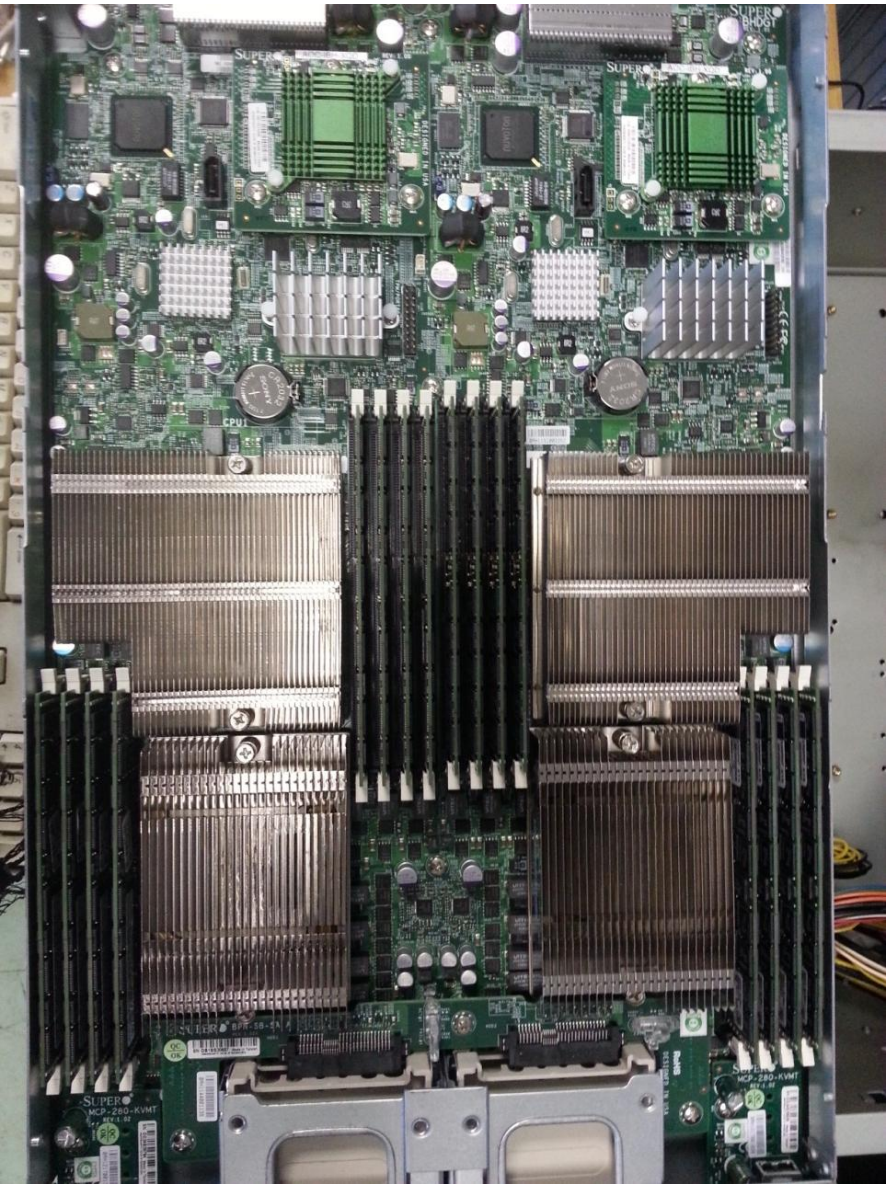
Luschevskaya E.V.

16 blade servers:
320 nodes, i.e. 640 CPU
i.e. 10240 cores

Each blade server has its own 72 Tb file server (36 disks,
each disk has 2 Tb)

Interaction between MPI cores is provided by the high-speed bus
Infiniband with the rate of 40 Gb/s.
OS: CentOS 6.X.

ITEP-FAIR supercomputer



Each blade has two motherboards (1 node has two CPU)
The motherboard has two 16-core processors
AMD Opteron 6272 (2.1 GGz).
The blade has 64 cores

Each processor has 8 GB x 4 =32 GB RAM (4 slots at the photo)

One core has 32 GB /16=2 GB RAM

GPU (graphics processing unit)

5 GPU servers; 1 server containes

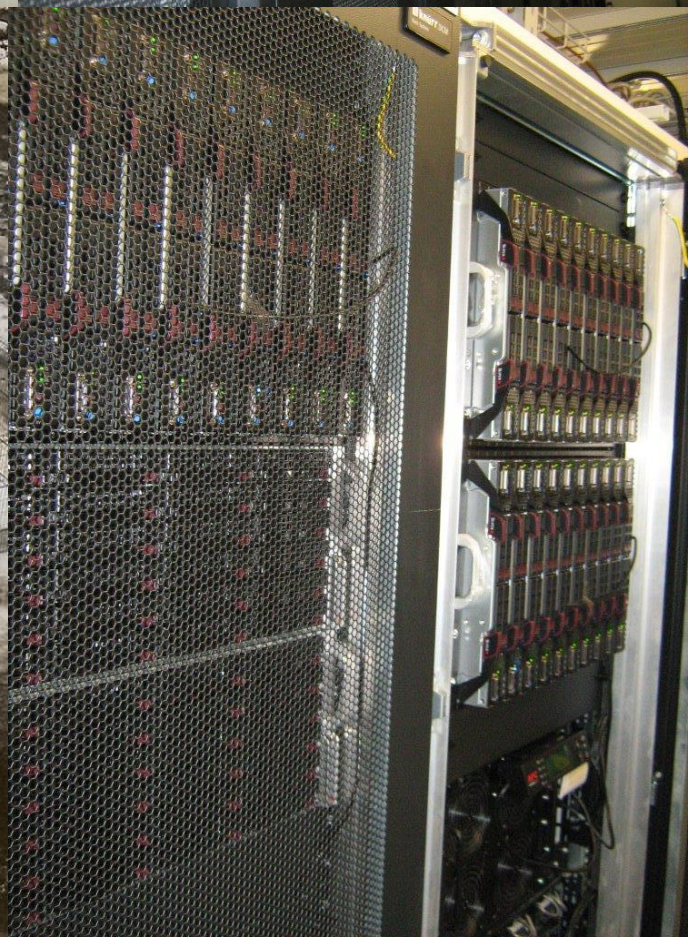
8 GPU NVIDIA Tesla K20X 6GB
8 CPU Intel Xeon E5-2650

Each GPU server consists of 4 nodes, i.e. motherboards.
Peak performance :
1300 Gflops for double precision calculations
3950 Gflops for single precision calculations


Allen-Bradley
PanelView 300 Micro
Temperature
Outlet -0.3 °C
Inlet 6.3 °C
Outside -10.9 °C
F1
F2
F3
F4


1


JÆGGI



Cooling system

Access to the supercomputer

- **UNIX system:**

  ssh login_ID@rrcmpi.itep.ru

  or

  ssh login_ID@rrcmpi-a.itep.ru

  password : #####

- **Windows :**

  any ssh client, for example PuTTY or WinSCP

Data storage

Save, read and write the data in

/home/clusters/rrcmpi !!!!! (the free space is appr. 800 TB now, LUSTRE file system)

Compile and run programs from

Disk rrcmpi-dsk-01 ; path: /home/clusters/01 ; quota: 244.2 GB

Disk rrcmpi-dsk-02 ; path: /home/clusters/02 ; quota: 4.841 TB

The size of the home directory is limited

Home directory: **/home/rrcsrv/login_ID,** command **quota**

Example: **[login_ID@rrcmpi-a ~]$ cd /home/clusters/rrcmpi**

[login_ID@rrcmpi-a ~]$ mkdir login_ID

[login_ID@rrcmpi-a ~]$ cd login_ID

Activation of MPI

To run parellel tasks the content of the file **.bashrc** (you can find this file in the home directory) should be the following

```
# .bashrc
if [ -f /etc/bashrc ]; then
      . /etc/bashrc
Fi
if ( module > /dev/null 2>&1 ); then
     module add openmpi-x86_64
fi
```

If it differs from the specified, it must be changed , one can use **mcedit, ed, vim.**

Compilation of programs

Compilers: mpicc, mpic++, mpif77, mpif90, gcc, g++, gfortran

- Use library libamdlibm, installed to **/opt/gcc-4.6/lib64.**

  There are two files in this directory: **libamdlibm.so** is for dynamic linking, **libamdlibm.a** is for the static one.

  Binary programm can be run in the environment

  LD_PRELOAD= /opt/gcc-4.6/lib64/libamdlibm.so yourprogram

- There is an optimized version of libraries **BLAS и LAPACK от AMD:**

  ACML-5.1.0, ACML-5.2.0 (см. /opt/acml5.X.0)

Installed software

- CentOS 6.6
- gcc 4.4.7 (is not recommended)
- gcc 4.6.3 (recommended, because it can optimize the code for processors installed on supercomputer), /opt/gcc-4.6/
- python 2.6.6
- perl 5.10.1
- acml 5.1.0 (5.2.0), /opt/acml5.1.0 (/opt/acml5.2.0)
- CUDA 4.2 for GPU applications, /opt/cuda

FAIRsoft

- CMAKE 2.8.9
- GTEST 1.6.0
- GSL 1.15
- GLPK 4.39
- BOOST 1.51.0
- PLUTO 5.37
- PATHIA 6 (416)
- PATHIA 8 (165)
- ROOT 5.34.01
- Application Monitoring API for C++ 2.2.7
- MonALISA Web Service Client (using gsoap) 2.7.10
- Geant 4 9.5.1
- Geant 3.21 (v1-14)
- Geant4_VMC (r 617)
- VGM (r 715)

Installed in the /opt/fairsoft

For CBMRoot, PandaRoot, etc.

export SIMPATH=/opt/fairsoft

Run the program

qstat , **qsub** file.sh, where the file.sh is the BASH script,

kill the task with **qdel <Job id>** , **qstat –f <Job id> - task information**

```
[luschevskaya@rrcmpi-a ~]$ qstat -q

server: rrcmpi-a-srv.itep.ru

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
medium             --   23:59:59 23:59:59   --    0   0 47   D R
main               --      --       --     --    0   0 --   D R
mpi                --   430080:0 720:00:0   --    0   0 95   D R
short              --   04:49:59 04:49:59   --    0   0 47   D R
kepler             --   720:00:0 720:00:0   --    9   0 40   E R
long               --   199:59:5 199:59:5   --    0   0 95   D R
mem8gb             --   200:00:0 200:00:0   --    0   0 23   D R
mem4gb             --   500:00:0 500:00:0   --    0   0 47   D R
gpu                --   720:00:0 720:00:0   --    0   0  3   E R
xxl                --   2160:00: 2160:00:   --    0   0 47   D R
                                              ----- -----
                                                  9     0

[luschevskaya@rrcmpi-a ~]$
```

Example of the BASH script for single-processor task

```
#!/bin/bash
#PBS -N 20x20_B3.500
#PBS -q long
#PBS -o /home/clusters/rrcmpi/luschevskaya/IMPROVED/L20x20/out.out
#PBS -e /home/clusters/rrcmpi/luschevskaya/IMPROVED/L20x20/err.err

cd   /home/clusters/rrcmpi/luschevskaya/IMPROVED/
/home/clusters/02/luschevskaya/IMPROVED/su2-impr.exe  –i /home/clusters/rrcmpi/luschevskaya/IMPROVED/info001.dat  –o /home/clusters/rrcmpi/luschevskaya/IMPROVED/configuration001.dat
exit 0
```

Examples

1. single-processor task : qsub file.sh
2. run MPI program

qsub –q mpi –l nodes=n:ppm=m

the total number of processes m x n

processes at each node m <=32 (number of cores at a node)

If there are no special requirements for the distribution of processes across the nodes, then you should indicate ppm=1

Inside the MPI script a program runs by the following command:

mpirun --mca btl ^tcp your_mpi_programm

option "--mca btl ^tcp" directs the MPI messaging to Infiniband.

BASH scripts for execution of MPI codes

File job.sh

```
#!/bin/bash

qsub -q mpi -l nodes=5 -d /home/clusters/02/ulybyshev/test/log_errors
    /home/clusters/02/ulybyshev/test/beta_run.sh
```

File beta_run.sh:

```
#!/bin/bash

mpirun --mca btl ^tcp /home/clusters/02/ulybyshev/test/programs/calc
    /home/clusters/02/ulybyshev/test/logs/
    /home/clusters/02/ulybyshev/test/logs/init_files/constants.txt
    /home/clusters/02/ulybyshev/test/logs/init_files/index_conf.txt
```

Running: ./job.sh

The file submit_b0_7p7.sh :

```
qsub -q mpi -l nodes=64 `pwd`/hmc_b0_7p7.sh
```

where the file hmc_b0_7p7.sh runs the tasks

```
mpirun /home/clusters/01/vborn/NC/src/hmc/hmc_chiral_test -np
64  /home/clusters
/01/vborn/NC/Confs/Confs16t6b7.7m0.024m0.053B0/latest.par
```

Execution:

```
./submit_b0_7p7.sh
```

Example of compilation and running of C parallel code

- [kolosov@rrcmpi tmp]$ cat > hello.c <<EOF

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

int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);
  printf("Hello World! from process %d out of %d on %s\n", rank, numprocs,
processor_name);
  MPI_Finalize();
}
EOF
```

```
[kolosov@rrcmpi tmp]$ mpicc -O3 -o hello hello.c

[kolosov@rrcmpi tmp]$ cat > test.sh <<EOF
#!/bin/bash

echo  Starting batch on `hostname` on `date`

mpirun --mca btl ^tcp xhpl

EOF
[kolosov@rrcmpi tmp]$ chmod +x test.sh
[kolosov@rrcmpi tmp]$ qsub -q mpi -l nodes=240 test.sh
```