High-Performance MPI Application Energy Consumption Profiling

Dmitry Kryzhanovskiy, Victor Getmanskiy

Singularis Lab, LLC

VolgSTU

The research was supported by Intel
and Nizhniy Novgorod Fund of Assistance for Education and Research

Outline

1. Task specifications
2. Energy consumption profiling
3. Packages and benchmarks
4. Difficulties and reefs
5. How to profile: methodology
6. Analysis results
7. Conclusions
8. Common results

Task specifications

General task: **Describe the methodology how to profile HPC MPI applications and analyze their power consumption**

1.Compare software tools for energy consumption.

2.Profile the selected packages, analyze the results and compare different MPI implementations.

3. Create master-classes to share the obtained knowledge.

Scope of power analysis

- Hardware analysis
  - analysis of computational system consumption
  - no analysis of single application consumption
- Software analysis at application level (Power Top)
  - analysis of single application consumption
  - detecting the consumption sources (IO, memory, HDD, CPU etc.)
- Software analysis at function level (Intel® VTune Amplifier)
  - analysis of consumption of single functions and units
  - detecting the functions (methods) requiring energy consumption optimization by the developer

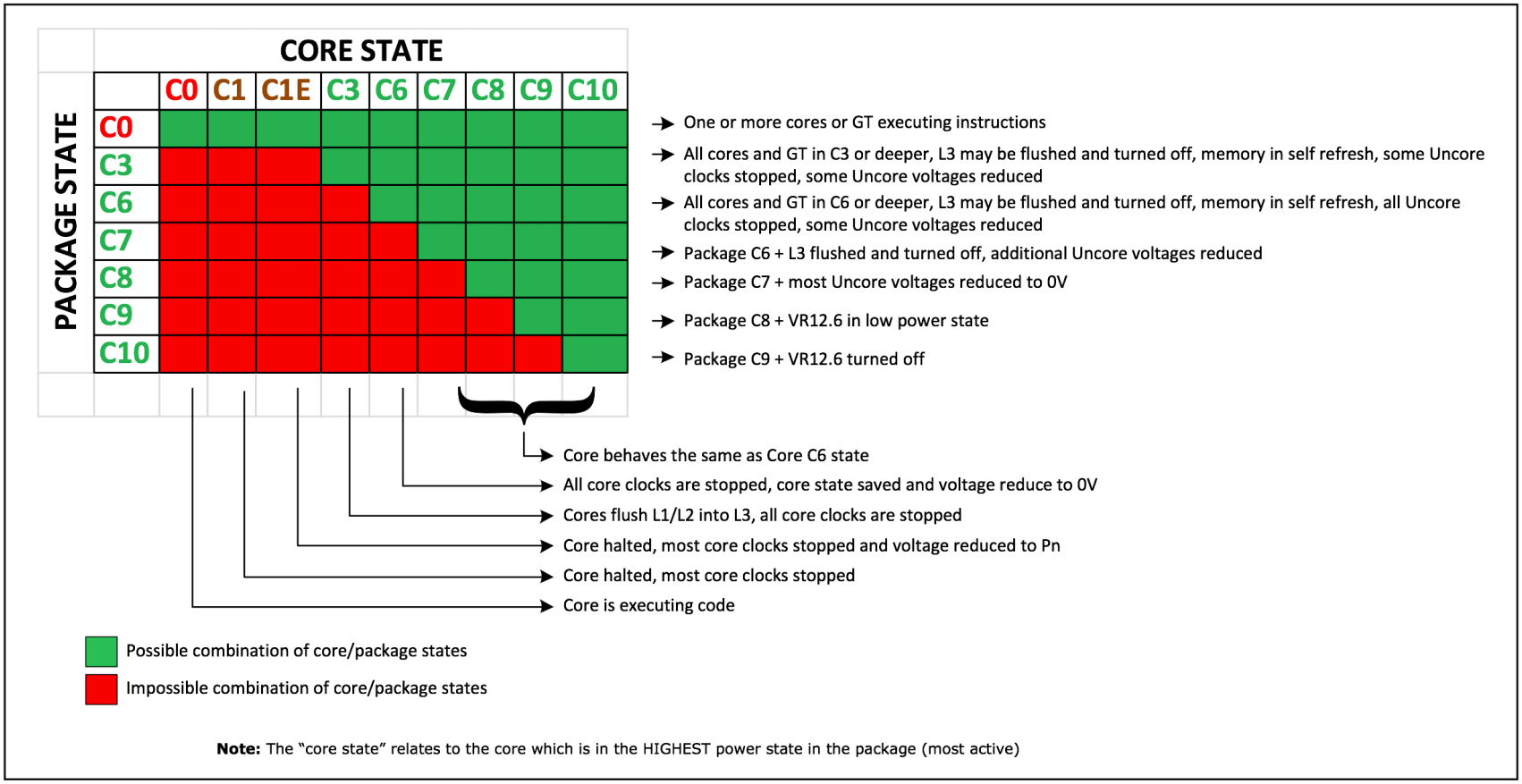
C-states of Intel Xeon E5

CPU **C-states** are core power states requested by the Operating System Directed Power Management (OSPM) infrastructure. C1-Cn states describe states where the processor clock is inactive and different parts of the processor are powered down.

C0	Full on
C1/C1E	Auto-halt
C3	Deep Sleep
C6/C7	Deep Power Down

C-states (Haswell Mobile Processor)

Processor Package and Core C-States



Software tools

	OS	CPU models	CPU consumption	Consumption of the other devices	Linkage with the code
Power Top	Linux, Solaris	Core2Duo …	Modelled	Modelled	No
Joulemeter	Windows	Core2Duo …	Modelled	Modelled	No
Intel Power Gadget	Windows, Linux, OS X	Core-i, XEON (Sandy Bridge) …	Direct measurement	–	No
XCode 5 Power Profiler	OS X	Core-i, XEON (Sandy Bridge) …	Combined indices	–	Statistics for single threads
Intel ® VTune Amplifier	Windows, Linux	Core-i, XEON (Sandy Bridge) …	Direct measurement	–	Yes

Intel® VTune Amplifier is the most comfortable tool for developers

Packages and benchmarks

Open source and popular in industry HPC applications:

- **GAMESS** – computational quantum chemistry.
- **GROMACS** – molecular dynamics for modelling physicochemical processes.
- **LAMMPS** – simulation of the classical molecular dynamics of particle system.
- **NAMD** – an object oriented code for modelling molecular dynamics of big biomolecular systems.
- **OpenFOAM** – computational hydrodynamics and work with fields (scalar, vector, tensor).

Difficulties and reefs

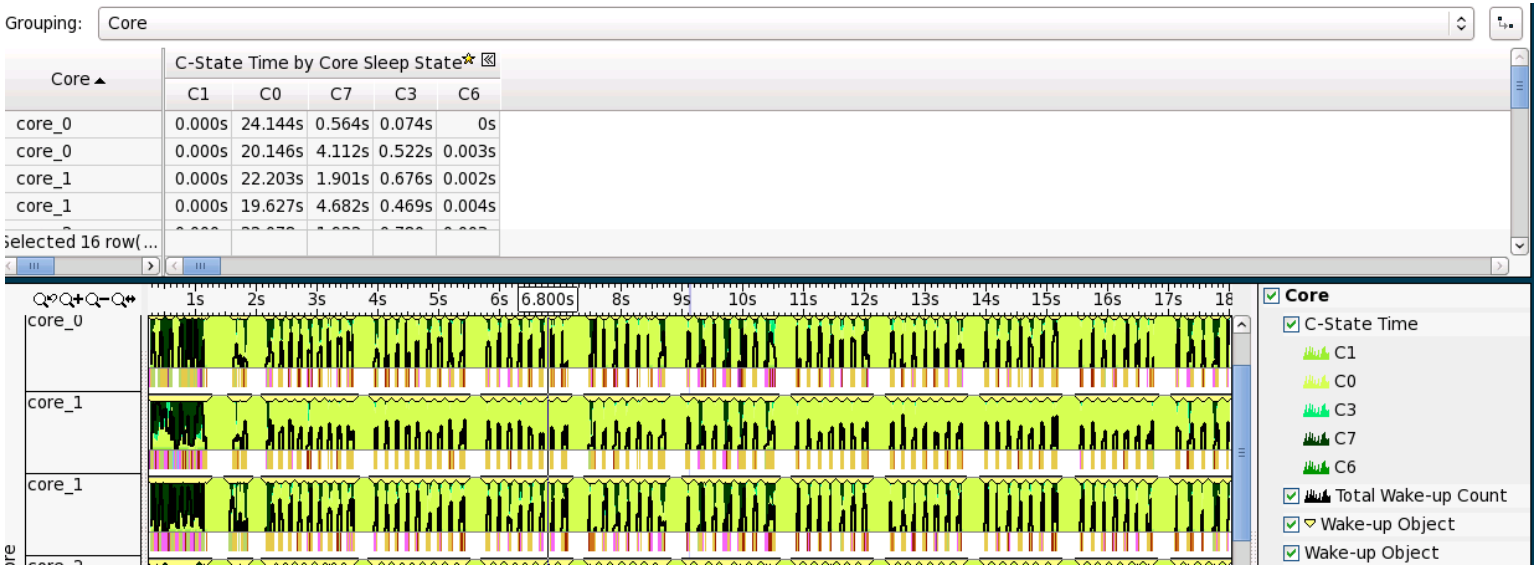
1. Building the packages with various implementations of MPI.
2. Need to balance between the size of task (sample), the size of application profile and its level of detail.
3. Problems with detecting units under dynamic linking and detecting functions (methods) inside units.

Difficulties and reefs

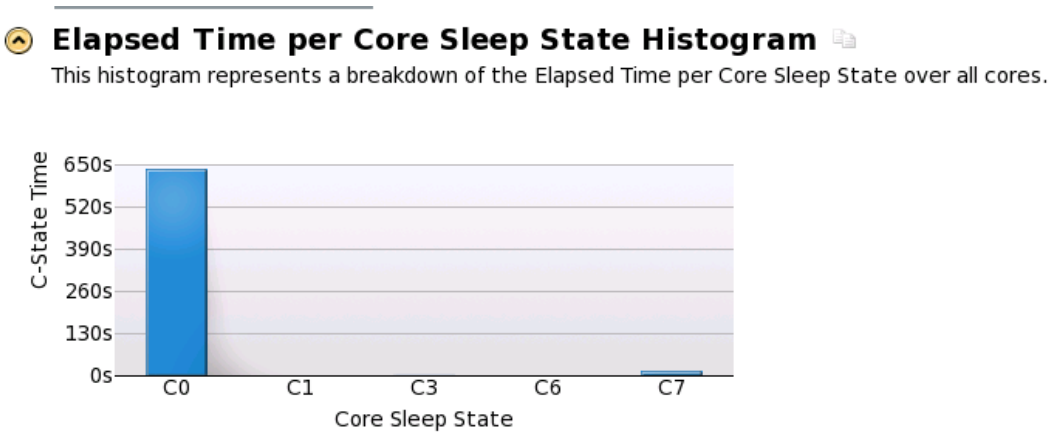
5. Collecting PMU events is limited with only one profiling process.
6. Detecting the mode of profiling for one and all processes and the necessity in analysis of MPI scheduler energy consumption.
7. Adjusting hardware for maximum scalability of the applications (NUMA, hyper-threading).

Difficulties: profiling mode for all / one process

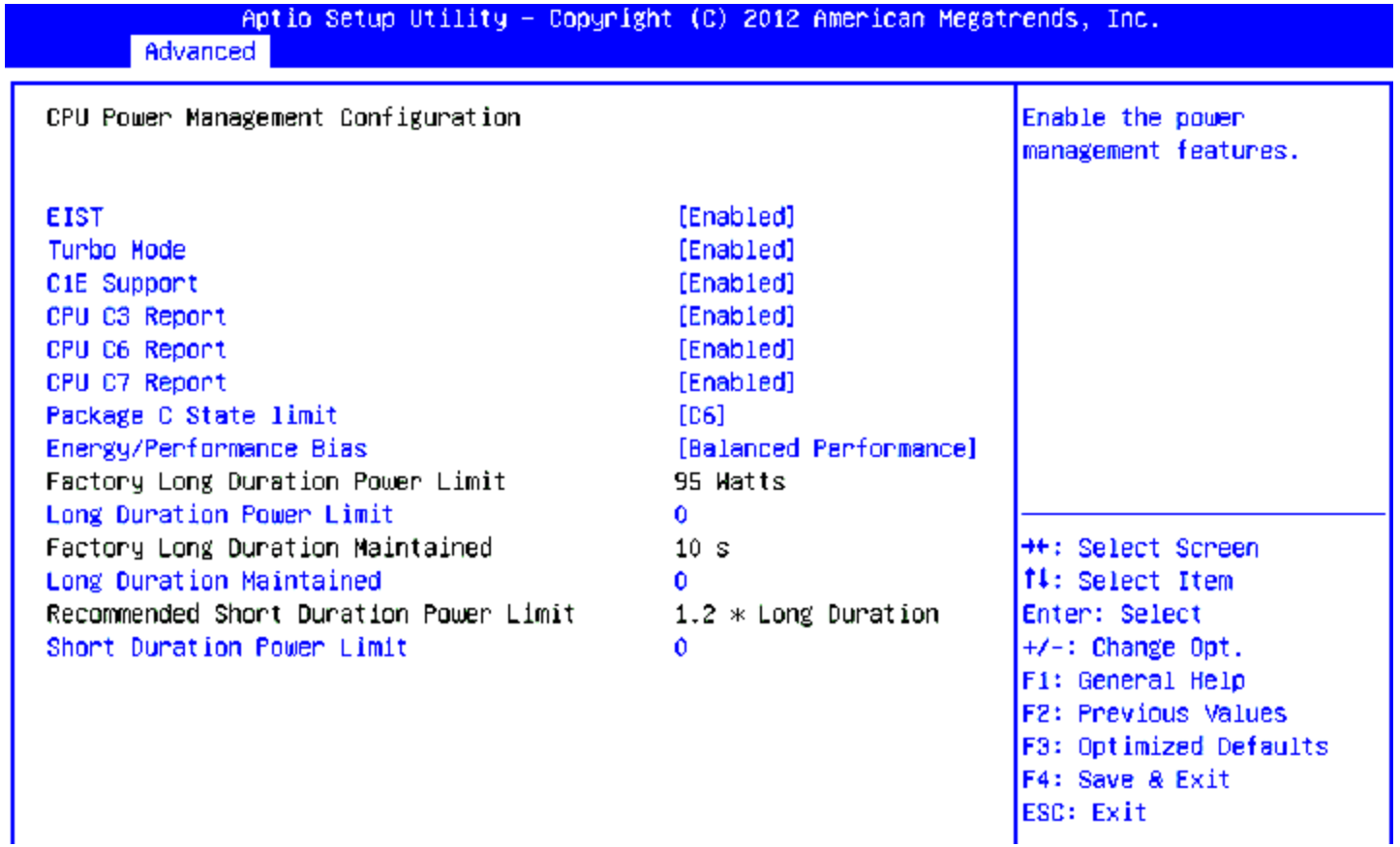
NAMD



LAMMPS

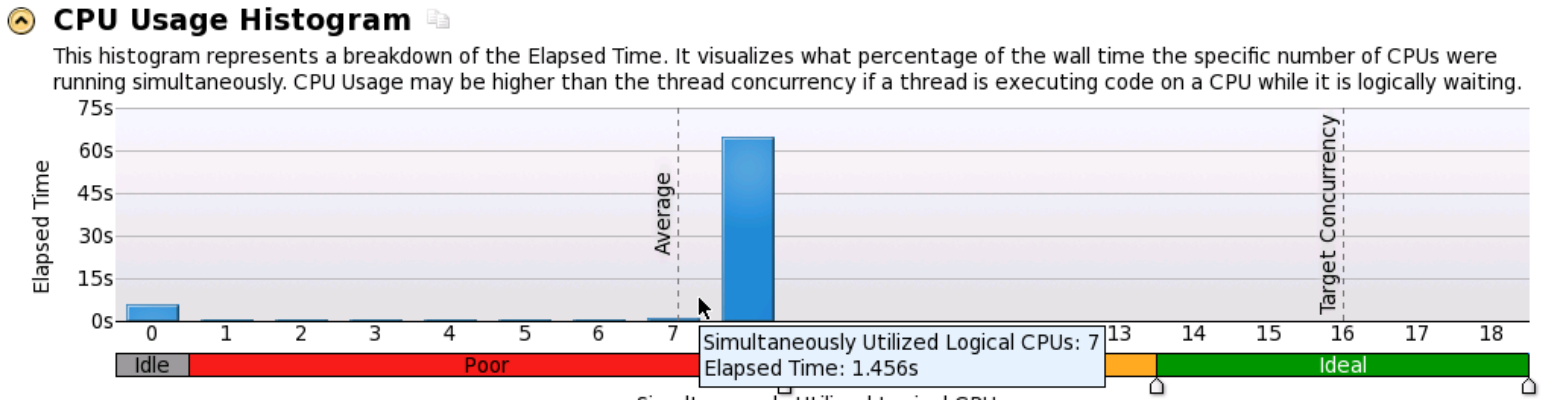


Difficulties: adjusting hardware (NUMA, hyberthreading)

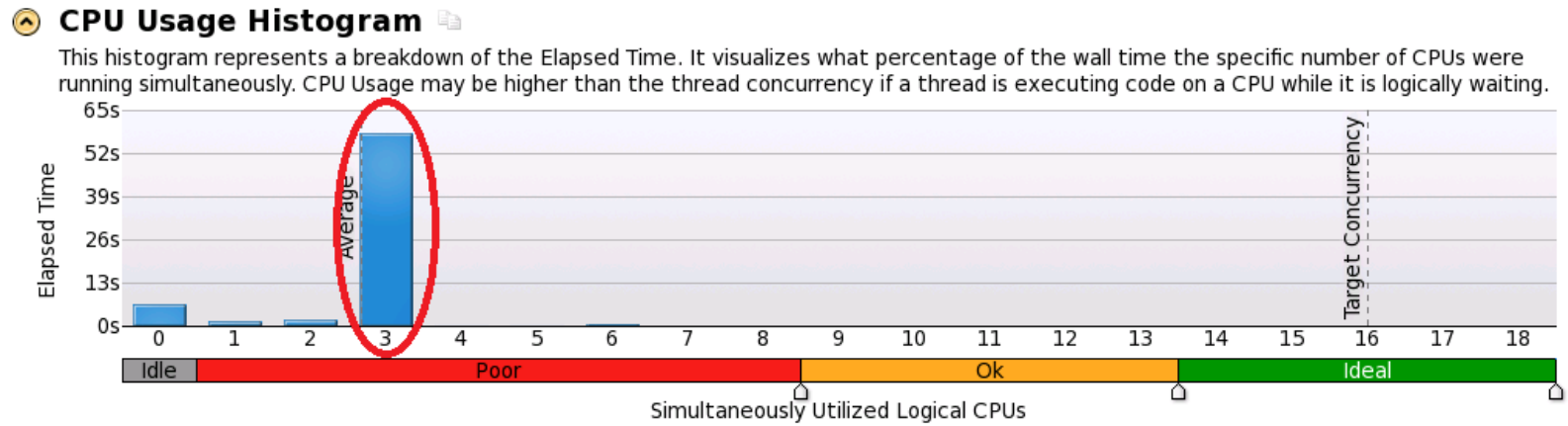


Difficulties: adjusting hardware (NUMA, hyberthreading)

Without NUMA



With NUMA



How to profile

Methodology:

1. Use static linking.
2. Profile all CPUs at node.
3. Find a relative part of the scheduler energy consumption.
4. Profile one of the processes being run.
5. Detect the MPI library functions in the application unit.
6. Estimate the energy consumption relative parts of the application functions and MPI library.

Adjusting the environment

.bashrc

```
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
          . /etc/bashrc
fi
# User specific aliases and functions
export PATH=/usr/gcc/gcc470/bin:$PATH
export LD_LIBRARY_PATH=/usr/gcc/gcc470/lib64:/usr/gcc/gcc470/lib:$LD_LIBRARY_PATH
source /opt/intel/compilers/composerxe/mkl/bin/mklvars.sh intel64
source /opt/intel/compilers/composerxe/bin/compilervars.sh intel64
```

.bashrc.namd_IntelMPI

```
module unload MPI-3/MVAPICH2
module unload MPI-3/OpenMPI
module unload MPI-3/MPICH
module load MPI-3/IntelMPI
export PATH=/var/local/INTEL/install/intelmpi-icc/third-
party/src/NAMD_2.9_Source/Linux-x86_64-ics-2013-SP1:$PATH
export PATH=/var/local/INTEL/src/NAMD_2.9_Source/charm-6.4.0/bin:$PATH
```

How to run the profiling

```
#!/bin/bash
DIR=$(pwd)
WORKING_DIR=$DIR/NAMD/small

# remove profile output folder
$DIR/RemoveDir.sh $PROFILE_DIR

# rewrite bashrc
(cat $DIR/.bashrc) > ~/.bashrc
(echo source $DIR/.bashrc.namd_IntelMPI) >> ~/.bashrc
source ~/.bashrc

# run
cd $WORKING_DIR
mpirun -genv I_MPI_FABRICS=shm:tcp -genv I_MPI_WAIT_MODE=1 \
-host node31 -n 1 -wdir=$WORKING_DIR amplxe-cl -collect advanced-hotspots -knob
collection-detail=stack-sampling -r $PROFILE_DIR - namd2 apoa1 : \
-host node31 -n 15 -wdir=$WORKING_DIR namd2 apoa1 : \
-host node30 -n 16 -wdir=$WORKING_DIR namd2 apoa1
```

Test hardware and run configuration

Hardware:

- 2 nodes with 2 Intel XEON E5 CPUs 8-cores per node (16 cores per node)
- 128Gb RAM per node
- 1Gb LAN

Run configuration:

- Maximal power (32 cores), no NUMA, no HT
- Single-node mode (16 cores), no NUMA, no HT

Analysis results

Energy consumption relative parts for the application functions and MPI library

Application	Energy Core	Energy Pack	Energy DRAM
namd2	1091474032	1351496528	36065280
Without MPI	873696888	1081513344	28925454
	Energy Core	Energy Pack	Energy DRAM
Total Power	1171367632	1450679136	38622192
Relative Power Consumption	Energy Core	Energy Pack	Energy DRAM
mpiexec	0%	0%	0%
MPI	19%	19%	18%
namd2	75%	75%	75%

Conclusions

1. Static MPI linking with one process profiling provides more correct results.
2. Running the package on several cluster nodes provides more correct and representative results.
3. Running the package on several cluster nodes under one process profiling is possible for Intel MPI, MPICH, MVAPICH and Open MPI with some differences in the MPI scheduler commands.

Conclusions

5. Analyzing energy consumption on cluster, you need to take into account that CPU is in the state C0 the most time. Switching between the states happens only if intensive data exchange takes place (as it is in NAMD).
6. Energy consumption of the application functions and MPI library depends on the used MPI implementation and the task you are solving (its sizes).

Common results

- The methodology how to profile applications for power consumption has been described and master-classes on this topic have been created.
- The benchmark parameters for more representative power profiling have been adjusted.
- The distribution of energy consumption for different MPI implementations and for single MPI functions has been estimated for several HPC packages.

Contacts

Thank you for your attention!

- Dmitry Kryzhanovskiy (Singularis Lab)
  - dmitry.kryzhanovsky@singularis-lab.com
- Victor Getmanskiy (Singularis Lab)
  - victor.getmanskiy@singularis-lab.com
- Dmitry Zavyalov (VolgSTU)
  - sinegordon@gmail.com
- Vitaliy Kamnev (VolgSTU)
  - kamnevv@gmail.com