Using the cluster "Sergey Korolev" for modelling computer networks

D Y Polukarov[1], **A P Bogdan**[1]

[1]Samara National Research University, Moskovskoe Shosse, 34A, Samara, Russia, 443086

e-mail: plkw@mail.ru

Abstract. Modelling large-scale networks requires significant computational resources on a computer that produces a simulation. Moreover, the complexity of the calculations increases nonlinearly with increasing volume of the simulated network. On the other hand, cluster computing has gained considerable popularity recently. The idea of using cluster computing structures for modelling computer networks arises naturally. This paper describes the creation of software which combines an interactive mode of operation, including a graphical user interface for the OMNeT++ environment, with a batch mode of operation more natural to the high-performance cluster, "Sergey Korolev". The architecture of such a solution is developed. An example of using this approach is also given.

1. Introduction

Modeling large-scale networks requires significant computational resources on a computer that produces a simulation. Many universities have at their disposal computational cluster architectures of various sizes. The aim of this current work is to improve the quality of network simulation by using cluster "Sergey Korolev" of Samara National Research University. OMNeT++ is used in this work to model computer networks. OMNeT++ is a discrete event simulation environment [1]. The primary application area of OMNeT++ is the simulation of communication networks. OMNeT++ has a number of examples in its composition. This simplifies the process of deploying and verifying the system on a cluster.

The use of the high performance cluster "Sergey Korolev" often dictates a batch (and not interactive) mode of operation [2], without no graphical interface being available. And although there are nevertheless some opportunities for interactive work using the cluster, and some graphical output [2] can be made available in relation to this, these possibilities are severely limited.

For simulation of computer networks, it is more convenient to use an interactive mode of operation alongside a graphical user interface. For these purposes, software should be created which is capable of interacting with the user via graphical tools. In such a case, the OMNeT++ system [1] can serve as a convenient shell. The user can use the OMNeT++ system models but make the necessary calculations using the cluster in batch mode. Subsequently, the results of the calculations are transmitted to the user's local computer.

2. Related works

In [3], the authors analyzed a set of INET models [4], in terms of the ability of such model networks, in OMNeT++, to run in parallel. The authors found several issues which prevented the parallel execution of INET models. They analyzed these problems and developed solutions to them – to allow parallel launching of INET models. A situational analysis (using the case-study method) showed the feasibility of this approach. Although there are some elements of the model range that have not yet been investigated, and the performance attained still leaves some room for improvement, the results show an acceleration due to parallelization across most configurations.

A number of papers are devoted to the use of clusters in scientific applications.

In [5], the authors implemented a fast installation of a virtual cluster in heterogeneous environments. To do this, they created a performance model that predicted the installation time of a virtual cluster. They proposed a resource selection policy based on the model made as well. They divided the entire installation of the virtual cluster into five logical steps and created a model for each step. Each step is represented as a linear combination of software and hardware parameters, including processor speed, disk I/O performance, and installation package size. An advanced virtual cluster installer was also made. Experiments using the advanced installer have shown the effectiveness of the selection policy based on the models created.

The work [6] is the result of efforts to support a data analysis cluster (DAC) with minimal effort from the local system administrator. The authors tried to enable the scientist to focus on analyzing the data and not on managing the cluster. They developed a tool that allows a scientist to deploy and restore entire clusters with minimal effort. Puppet CMS was used to implement the deployment algorithm. This allowed system administrators to spend less time deploying a cluster. The ATLAS Tier 3 Cluster is given as an example. Reducing the time spent deploying and maintaining the cluster will result in more scientific results being produced from the DACs.

There are works that offer novel algorithms for storing and processing data on clusters.

Article [7] contains an approach to distributed image storage. This is done to improve the efficiency of parallel image processing algorithms. The distributed image was defined as a set of overlapping fragments. This made it possible to avoid bottlenecks of parallel processing and overlapping associated with reading/writing image data. Since the efficiency of a distributed system for processing large-sized images is largely determined by data access methods in image processing software and the convenience of visualizing processing results. The authors analyzed the advantages and bottlenecks of popular parallel image processing systems. A novel approach to data organization in distributed systems for image processing and storage is proposed. This approach is based on the concept of a distributed image. The concept of the organization of distributed image data is proposed, which solves most of the current problems. Achieved the required level of fault tolerance of the distributed storage of image fragments.

A number of papers are devoted to the use of the OMNeT++ simulator on cluster architectures.

The article [8] describes the parallelization of OMNeT++ simulations with Xgrid. The necessary changes in OMNeT++ for the automatic creation of job description files for Xgrid are developed. This allows the user to easily set up a simulation for Xgrid. By processing the job specification file, the Xgrid controller then distributes the individual simulation runs to all available parallel computing machines. Smaller files, such as configuration files, may be included in the job description file. Large files should be read or written to network shares for increased performance. The actions developed accelerated simulation modeling, almost linear with respect to additional computing power, compared to conventional single-process computing.

The article [9] presents a scheme for cluster modeling of distributed systems based on Ethernet (RTEthernet) in real time. It relies on the OMNeT++ discrete event simulation environment associated with the ARM-based coprocessor. The presented approach allows us to associate the real RTEthernet subsystem with virtual components operating in discrete modeling that implement the required behavior for the subsystem. The authors estimated the performance limits of this approach with respect to latency and jitter when running a simulation on a Linux system with a real-time kernel fix. The results showed that synchronization requirements for cluster modeling of small RTEthernet networks can be achieved.

This paper demonstrates the concept of cluster modeling of RTEthernet systems. The computing platform is based on a standard PC with an RT-Linux kernel, on which RTEthernet models work in the OMNeT++ simulation environment and the ARM9 microcontroller as a coprocessor.

The evaluation showed that the platform provides sufficient performance for the latency requirements of distributed real-time systems in the 230μs range, which is limited and has a linear dependence on frame size. The observed jitter is below 40μs.

The article [10] reports on the novel parallel and distributed modeling architecture for OMNeT++, an open source discrete-event modeling environment. The main application area of OMNeT++ is network communication modeling. Support for the conservative PDES protocol (zero message algorithm) and relatively new simulation protocol has been implemented. The OMNeT++ PDES implementation has a modular and extensible architecture that makes it easy to add new synchronization protocols and new communication mechanisms, which also makes it an attractive platform for PDES research. The advantage of the implementation is that it has the principle of "separation of experiments from models" and, thus, allows simulation models to run in parallel without any changes. It is based on a new placeholder approach for instantiating the model on different logical processes (LPs).

So, to summarize, the above article describes an analysis of the parallelization problems inherent in the INET set of OMNeT++ models and demonstrates the possibility of INET parallelization, using a number of modifications to the models; this parallelization was primarily aimed at distributed initialization of the INET models.

The concept of distributed multistage initialization allows for the creation of a simulation model which supports distributed simulation execution. The tests showed that there is an acceleration of execution, due to parallelization, for the given example in most configurations, although there is still potential for further optimization. The implementation covers the most commonly used protocols, such as Ethernet, IPv4, TCP, and UDP, as well as the related models such as ICMPv4 or the application of UDP.

3. Deploy and run a project on a cluster

3.1 Deploying

According to the official documentation [2], the cluster is accessed via SSH-2, and SFTP is used to transfer files. Programs are run as part of batch processing tasks.

To deploy a network simulator on a cluster, the steps presented in Table 1 are necessary.

Table 1. Steps for deploying a network.

Deploying steps
• access the cluster; • upload the necessary files to the cluster; • create a task which build the OMNeT++ network simulator and wait for its completion (there is a version of OMNeT++ without an IDE, this is what was used here); • create a task which build the INET Framework; • create a test simulation model; • create a task which runs the simulation and wait for its completion; • download the result of the work.

However, interactive task execution is possible on the cluster - given access to a free node in the cluster. The simulation will be run as follows (in Figure 2).

It is worth noting that the main difficulties encountered when deploying a simulator on a cluster are to do with recreating the necessary environment and also there are the problems relating to the limited access available (all work is done in the console mode).

After initial testing, it was found that the INET Framework does not support parallel simulation work. To solve this problem, the proposals in "Parallel INET for OMNeT++" [11] were used.

However, this work has the disadvantage that it is based on the old versions of OMNeT++ and the INET Framework (from 2014), and so dictates a slightly different structure for the modules (changes in the algorithm are necessary).

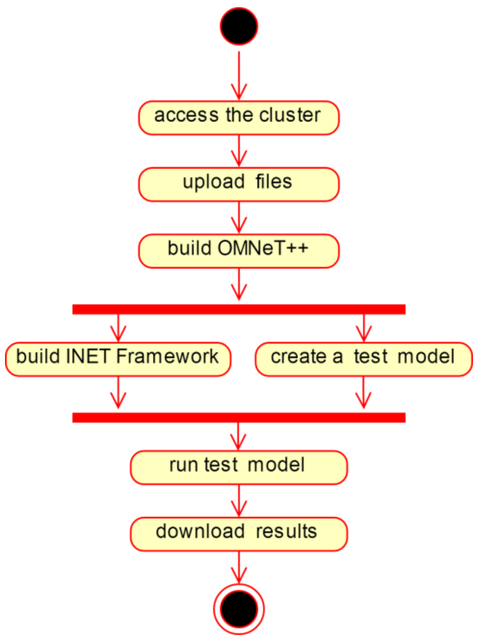
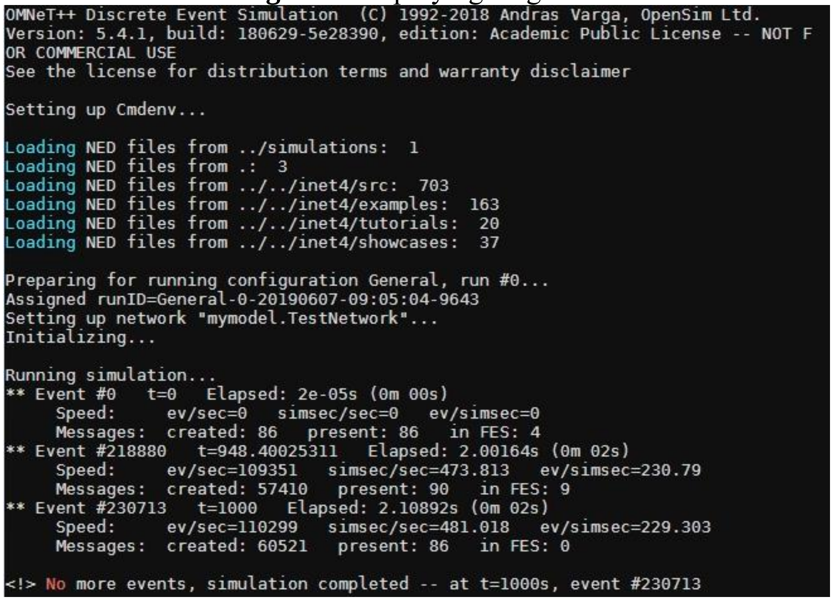


Figure 1. Deploying diagram.

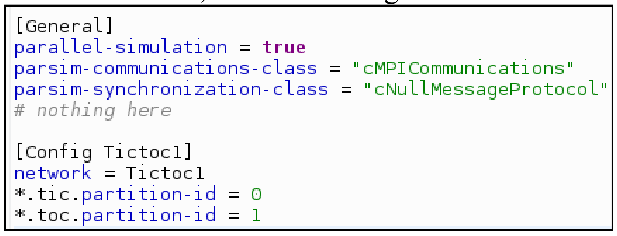


```
OMNeT++ Discrete Event Simulation  (C) 1992-2018 Andras Varga, OpenSim Ltd.
Version: 5.4.1, build: 180629-5e28390, edition: Academic Public License -- NOT F
OR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer

Setting up Cmdenv...

Loading NED files from ../simulations:  1
Loading NED files from .:  3
Loading NED files from ../../inet4/src:  703
Loading NED files from ../../inet4/examples:  163
Loading NED files from ../../inet4/tutorials:  20
Loading NED files from ../../inet4/showcases:  37

Preparing for running configuration General, run #0...
Assigned runID=General-0-20190607-09:05:04-9643
Setting up network "mymodel.TestNetwork"...
Initializing...

Running simulation...
** Event #0   t=0   Elapsed: 2e-05s (0m 00s)
     Speed:     ev/sec=0   simsec/sec=0   ev/simsec=0
     Messages:  created: 86   present: 86   in FES: 4
** Event #218880   t=948.40025311   Elapsed: 2.00164s (0m 02s)
     Speed:     ev/sec=109351   simsec/sec=473.813   ev/simsec=230.79
     Messages:  created: 57410   present: 90   in FES: 9
** Event #230713   t=1000   Elapsed: 2.10892s (0m 02s)
     Speed:     ev/sec=110299   simsec/sec=481.018   ev/simsec=229.303
     Messages:  created: 60521   present: 86   in FES: 0

<!> No more events, simulation completed -- at t=1000s, event #230713
```

Figure 2. Running the simulation in interactive mode.

3.2 Run on cluster

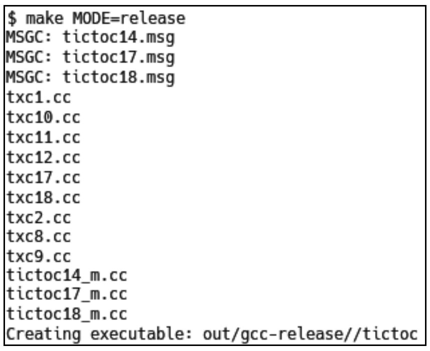
To work with MPI, the Tictoc1 system was selected; this consists of two modules. These modules should be assigned to different MPI nodes, as shown in Figure3.



```
[General]
parallel-simulation = true
parsim-communications-class = "cMPICommunications"
parsim-synchronization-class = "cNullMessageProtocol"
# nothing here

[Config Tictoc1]
network = Tictoc1
*.tic.partition-id = 0
*.toc.partition-id = 1
```

Figure 3. Configuring an MPI project.

Then the project in OMNeT++ is built on the local machine.



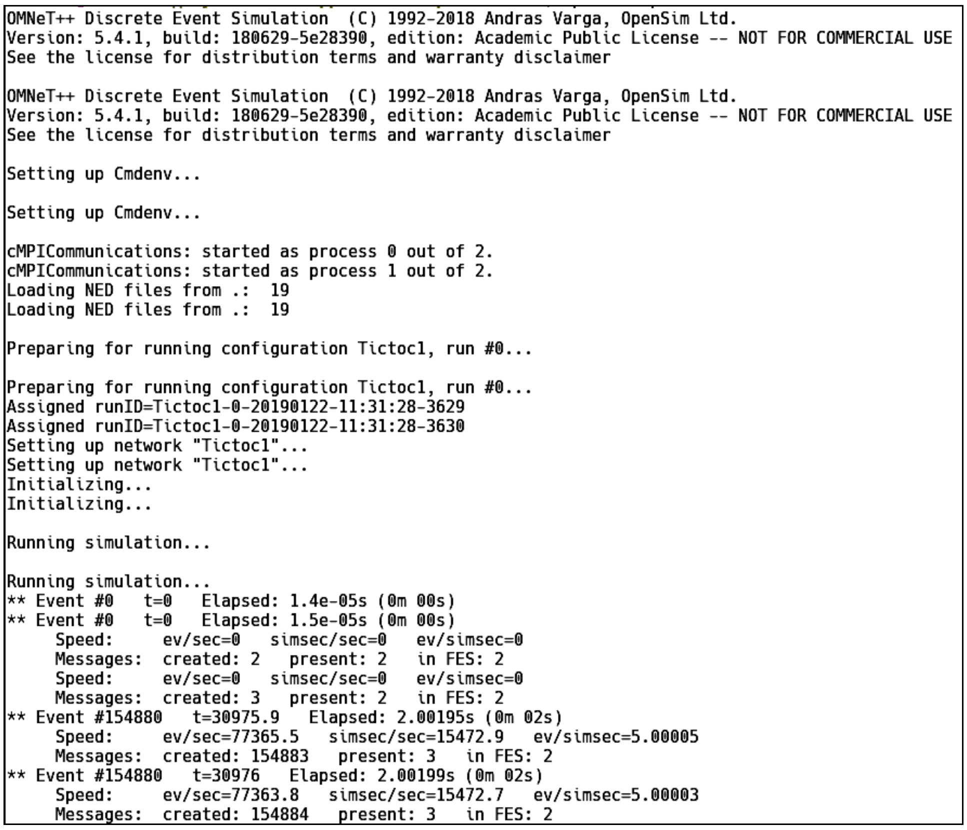
```
$ make MODE=release
MSGC: tictoc14.msg
MSGC: tictoc17.msg
MSGC: tictoc18.msg
txc1.cc
txc10.cc
txc11.cc
txc12.cc
txc17.cc
txc18.cc
txc2.cc
txc8.cc
txc9.cc
tictoc14_m.cc
tictoc17_m.cc
tictoc18_m.cc
Creating executable: out/gcc-release//tictoc
```

Figure 4. Building the project on a local machine.

To check the functionality from the project directory via mpirun, the following command is issued:

```
$ mpirun -np 2 ./tictoc -m -u Cmdenv -c Tictoc1 omnetpp.ini.
```

The result of such a command should look like this:



```
OMNeT++ Discrete Event Simulation  (C) 1992-2018 Andras Varga, OpenSim Ltd.
Version: 5.4.1, build: 180629-5e28390, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer

OMNeT++ Discrete Event Simulation  (C) 1992-2018 Andras Varga, OpenSim Ltd.
Version: 5.4.1, build: 180629-5e28390, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer

Setting up Cmdenv...

Setting up Cmdenv...

cMPICommunications: started as process 0 out of 2.
cMPICommunications: started as process 1 out of 2.
Loading NED files from .:  19
Loading NED files from .:  19

Preparing for running configuration Tictoc1, run #0...

Preparing for running configuration Tictoc1, run #0...
Assigned runID=Tictoc1-0-20190122-11:31:28-3629
Assigned runID=Tictoc1-0-20190122-11:31:28-3630
Setting up network "Tictoc1"...
Setting up network "Tictoc1"...
Initializing...
Initializing...

Running simulation...

Running simulation...
** Event #0   t=0   Elapsed: 1.4e-05s (0m 00s)
** Event #0   t=0   Elapsed: 1.5e-05s (0m 00s)
     Speed:     ev/sec=0   simsec/sec=0   ev/simsec=0
     Messages:  created: 2   present: 2   in FES: 2
     Speed:     ev/sec=0   simsec/sec=0   ev/simsec=0
     Messages:  created: 3   present: 2   in FES: 2
** Event #154880   t=30975.9   Elapsed: 2.00195s (0m 02s)
     Speed:     ev/sec=77365.5   simsec/sec=15472.9   ev/simsec=5.00005
     Messages:  created: 154883   present: 3   in FES: 2
** Event #154880   t=30976   Elapsed: 2.00199s (0m 02s)
     Speed:     ev/sec=77363.8   simsec/sec=15472.7   ev/simsec=5.00003
     Messages:  created: 154884   present: 3   in FES: 2
```

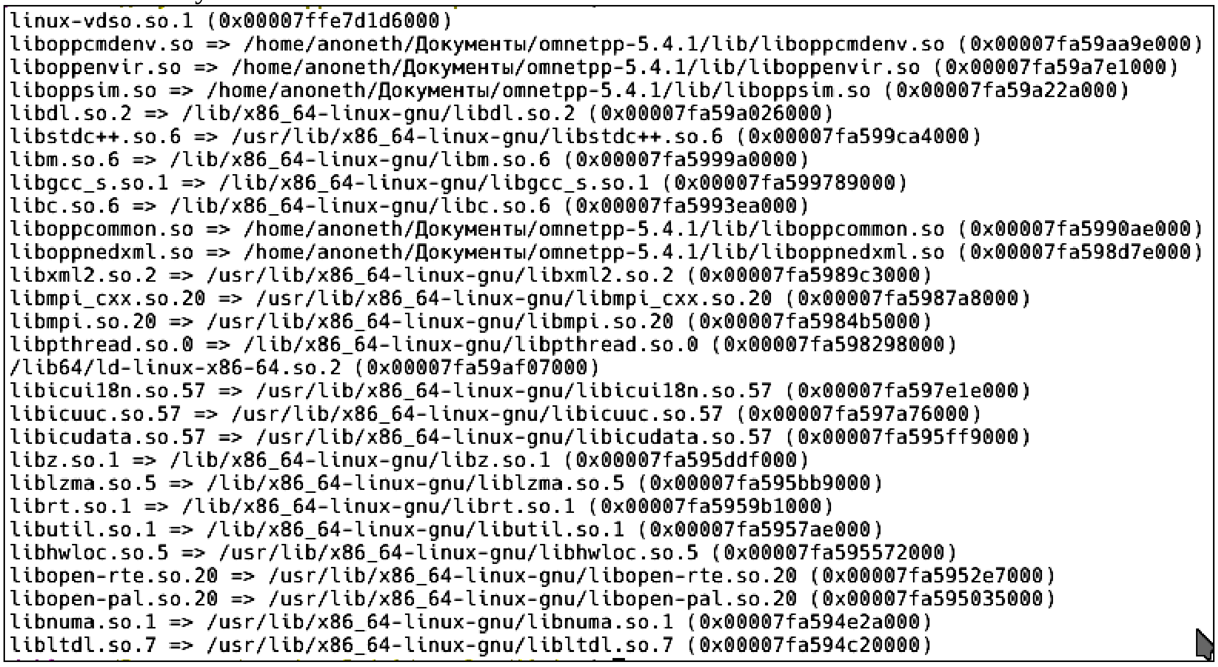
Figure 5. Test result.

In order to execute the simulation on the supercomputer, three files must be copied from the project folder to the working folder (the latter on the supercomputer).

Table 2. Files of test model.

Testing model structure
• omnet.ini;
• tictoc;
• tictic1.ned.

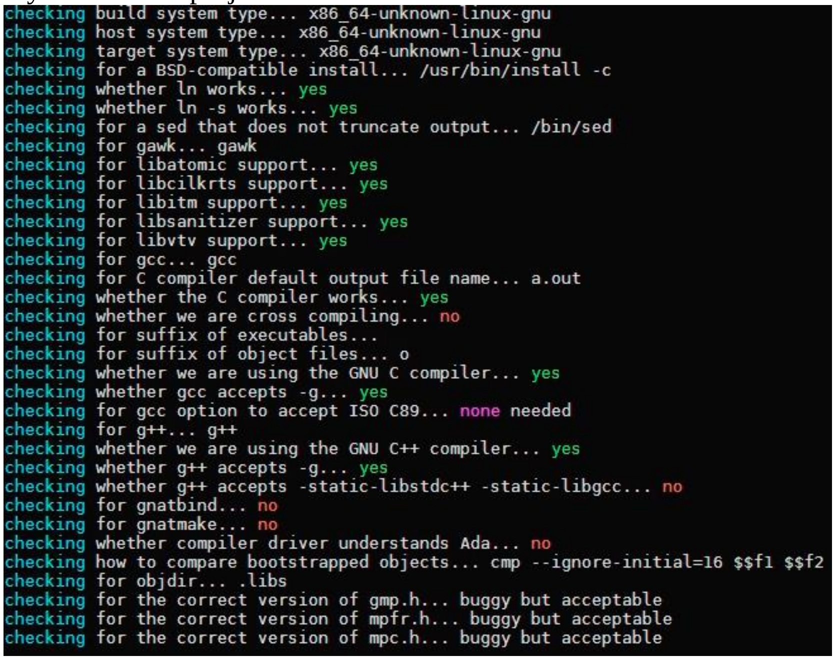
Next, the folder containing the project itself and the folder containing the source files must be copied over in order that the necessary libraries can be collected together into a working folder on the supercomputer, so that all can then be compiled (the libraries, and the project itself). The list of required libraries can be viewed using `ldd %executable_file_name%`, usually the necessary libraries will be only those that are in the OMNeT++ folder.



```
linux-vdso.so.1 (0x00007ffe7d1d6000)
liboppcmdenv.so => /home/anoneth/Документы/omnetpp-5.4.1/lib/liboppcmdenv.so (0x00007fa59aa9e000)
liboppenvir.so => /home/anoneth/Документы/omnetpp-5.4.1/lib/liboppenvir.so (0x00007fa59a7e1000)
liboppsim.so => /home/anoneth/Документы/omnetpp-5.4.1/lib/liboppsim.so (0x00007fa59a22a000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fa59a026000)
libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007fa599ca4000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fa5999a0000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007fa599789000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa5993ea000)
liboppcommon.so => /home/anoneth/Документы/omnetpp-5.4.1/lib/liboppcommon.so (0x00007fa5990ae000)
liboppnedxml.so => /home/anoneth/Документы/omnetpp-5.4.1/lib/liboppnedxml.so (0x00007fa598d7e000)
libxml2.so.2 => /usr/lib/x86_64-linux-gnu/libxml2.so.2 (0x00007fa5989c3000)
libmpi_cxx.so.20 => /usr/lib/x86_64-linux-gnu/libmpi_cxx.so.20 (0x00007fa5987a8000)
libmpi.so.20 => /usr/lib/x86_64-linux-gnu/libmpi.so.20 (0x00007fa5984b5000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007fa598298000)
/lib64/ld-linux-x86-64.so.2 (0x00007fa59af07000)
libicui18n.so.57 => /usr/lib/x86_64-linux-gnu/libicui18n.so.57 (0x00007fa597e1e000)
libicuuc.so.57 => /usr/lib/x86_64-linux-gnu/libicuuc.so.57 (0x00007fa597a76000)
libicudata.so.57 => /usr/lib/x86_64-linux-gnu/libicudata.so.57 (0x00007fa595ff9000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007fa595ddf000)
liblzma.so.5 => /lib/x86_64-linux-gnu/liblzma.so.5 (0x00007fa595bb9000)
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007fa5959b1000)
libutil.so.1 => /lib/x86_64-linux-gnu/libutil.so.1 (0x00007fa5957ae000)
libhwloc.so.5 => /usr/lib/x86_64-linux-gnu/libhwloc.so.5 (0x00007fa595572000)
libopen-rte.so.20 => /usr/lib/x86_64-linux-gnu/libopen-rte.so.20 (0x00007fa5952e7000)
libopen-pal.so.20 => /usr/lib/x86_64-linux-gnu/libopen-pal.so.20 (0x00007fa595035000)
libnuma.so.1 => /usr/lib/x86_64-linux-gnu/libnuma.so.1 (0x00007fa594e2a000)
libltdl.so.7 => /usr/lib/x86_64-linux-gnu/libltdl.so.7 (0x00007fa594c20000)
```

Figure 6. Required dependencies.

Next, it necessary to build the project on the cluster.



```
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether ln works... yes
checking whether ln -s works... yes
checking for a sed that does not truncate output... /bin/sed
checking for gawk... gawk
checking for libatomic support... yes
checking for libcilkrts support... yes
checking for libitm support... yes
checking for libsanitizer support... yes
checking for libvtv support... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking whether g++ accepts -static-libstdc++ -static-libgcc... no
checking for gnatbind... no
checking for gnatmake... no
checking whether compiler driver understands Ada... no
checking how to compare bootstrapped objects... cmp --ignore-initial=16 $$f1 $$f2
checking for objdir... .libs
checking for the correct version of gmp.h... buggy but acceptable
checking for the correct version of mpfr.h... buggy but acceptable
checking for the correct version of mpc.h... buggy but acceptable
```

Figure 7. Building project on a cluster.

Then you need to create, in the working folder on the supercomputer, the task file *.pbs.
Sample file content is as follows:

```
#!/bin/bash
#PBS -N helloOMNeT                //name of task
#PBS -A  helloOMNeT
#PBS -l walltime=00:01:00         //required time
```

```
   #PBS -l nodes=1:ppn=2            //number of nodes and processes
   #PBS -j oe                       //error stream to standard output
   cd $PBS_O_WORKDIR
   export PATH=$PBS_O_PATH
   # Loading Intel MPI v4 environment
   module load impi/4
   export I_MPI_DEVICE=rdma
   export I_MPI_DEBUG=0
   export I_MPI_FALLBACK_DEVICE=disable
   # command run task taking into account the features of the environment
   mpirun  -r  ssh  -machinefile  $PBS_NODEFILE  -np  $PBS_NP  ./tictoc  -m  -u
Cmdenv -c Tictoc1 omnetpp.ini.
```

4. Future work
We plan to test the work of OMNeT++ on the cluster "Sergey Korolev" using the project INET Framework. It is also planned to organize the possibility of full interactive work in the graphical interface of the modeling environment. To illustrate the effectiveness of the method used, it is planned to measure the performance of applications running on a cluster.

5. Conclusion
In this paper, we showed the development of software which allows us to combine an interactive mode of operation of the OMNeT++ modeling system and the batch mode of the cluster “Sergey Korolev”. in Via initial testing, it was discovered that the way by which networks are created in the course of simulation does not work in Parallel INET for OMNeT++. As a result, it was decided to create a network for parallel simulation using a Python script, and for ordinary simulation using a module created for OMNeT++. These results can be used to solve the other, related, problems [12].

6. References

[1] OMNeT++ Discrete Event Simulator URL: https://omnetpp.org/ (23.01.2019)

[2] Instructions for working with computer systems at SCC of Samara University *Supercomputer center of Samara University* URL: http://hpc.ssau.ru/node/19/ accessed 23.01.2019

[3] Enabling Distributed Simulation of OMNeT++ INET Models URL: https://arxiv.org /pdf/1409.0994.pdf (23.01.2019)

[4] INET Framework URL: https://inet.omnetpp.org/ (23.01.2019)

[5] Yamasaki S, Maruyama N and Matsuoka S 2007 Model-based resource selection for efficient virtual cluster deployment *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing* p 6

[6] Hendrix V, Benjamin D and Yao Y 2012 Scientific Cluster Deployment and Recovery–Using puppet to simplify cluster management *Journal of Physics: Conference Series* **396(4)** 042027

[7] Kazanskiy N L, Popov S B 2011 Distributed storage and parallel processing for large-size optical images *Proceedings of SPIE* (*Optical Technologies for Telecommunications)* **8410** 84100I DOI: 10.1117/12.928441

[8] Seggelmann R, Rüngeler I, Tüxen M and Rathgeb E P 2009 Parallelizing OMNeT++ simulations using xgrid *Proceedings of the 2nd International Conference on Simulation Tools and Techniques* (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) p 69

[9] Karfich O, Bartols F, Steinbach T, Korf F and Schmidt T C 2013 A hardware/software platform for real-time ethernet cluster simulation in OMNeT++ *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques* (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) 334-337

[10] Varga A, Sekercioglu A Y 2003 Parallel simulation made easy with OMNeT++ *15th European Simulation Symposium*

[11] Parallel INET for OMNeT++ URL: https://redmine.comsys.rwth-aachen.de/redmine/ projects/parallel-inet/ (01.05.2019)

[12] Nikitin V S, Semenov E I, Solostin A V, Sharov V G and Chayka S V 2016 Modeling the" smartlink connection" performance *Computer Optics* **40(1)** 64-73 DOI:10.18287/2412-6179-2016-40-1-64-73

Acknowledgments
This work was undertaken as a component of the project part of the state task of the Ministry of Science and Higher Education of the Russian Federation No. 2.974.2017/4.6.