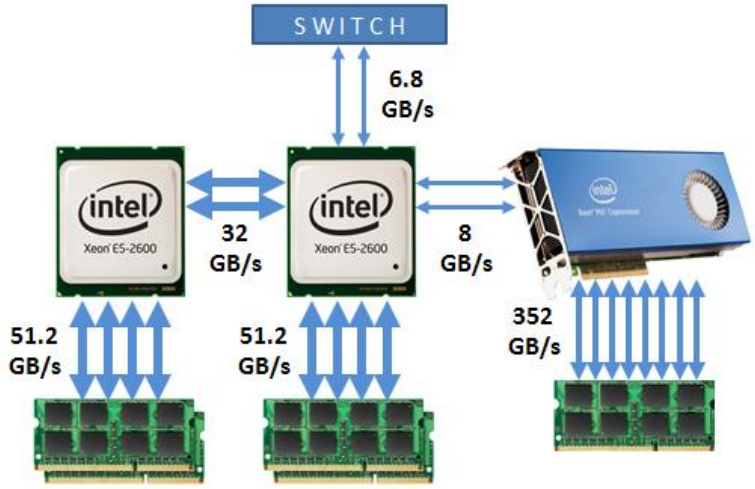
Optimizing rank-to-node mapping for MPI programs on large clusters

Victor Getmanskiy
Oleg Shapovalov
Efim Sergeev
Dmitry Kryzhanovsky

Singularis Lab, Ltd.
Volgograd State Technical University

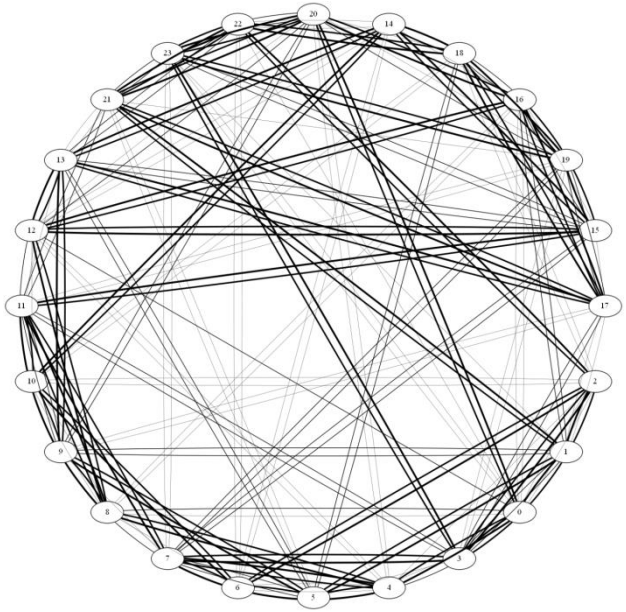
Problem

Different type and speed of interconnect between computing cluster cores



*Cornell Virtual Workshop
https://www.cac.cornell.edu/VW/Optimization/levelscomm.aspx

Non-uniform collective communication of MPI processes in HPC software



Communication graph of ZeusMP software

Is it possible to gain the performance of HPC application by minimization of communication time?

Optimization problem specification

Task graph: $G1(P, L), Li = (b_i, d_i), i = 1, \dots, N$

System graph: $G2(V, D), D_j = (l_j, b_j), j = 1, \dots, M$

l_i - latency

n_i – data exchange frequency

b_i - bandwidth

d_i – data volume

P – set of MPI processes

V – set of cores

Objective function:

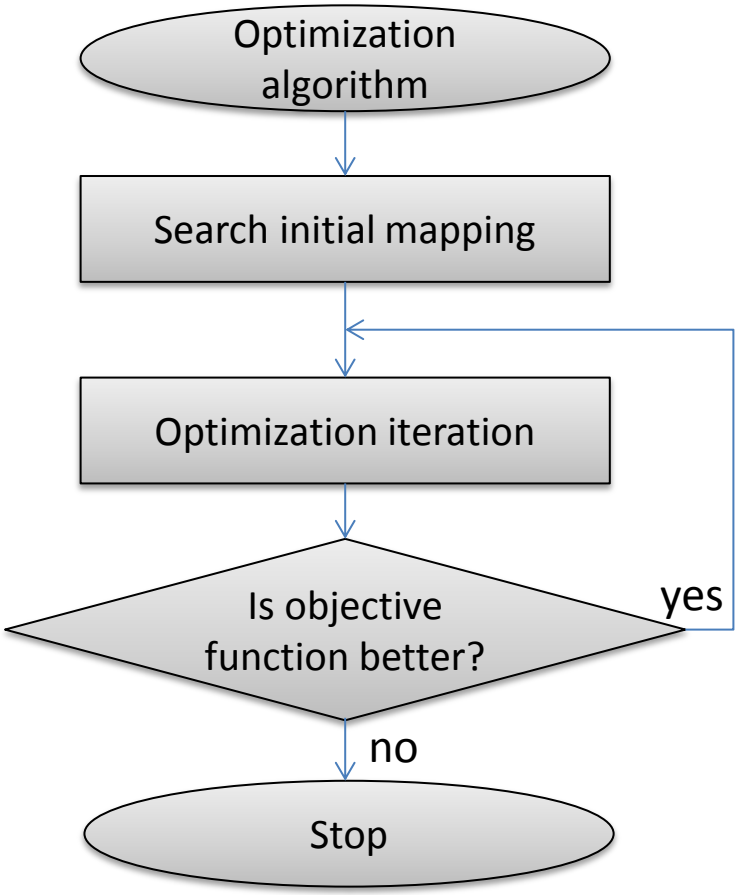
Searching task graph vertices mapping on system graph so that sum of edge weights function is minimal:

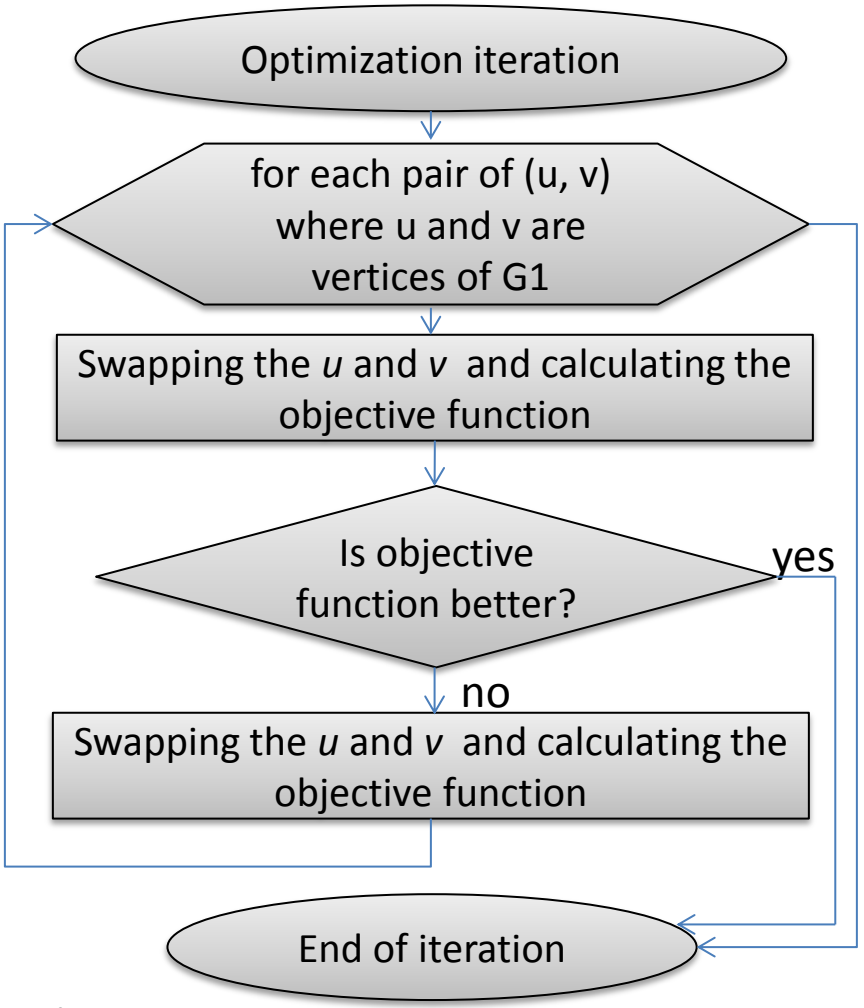
$$\min \sum_{k=1}^{N} T_k,\ T_k(D_i, L_j) = l_i * n_j + d_j / b_i,$$

$$G1 \to G2\ (P \to V): L_i \to D_j$$

* For the prototype the simplified formulation is used: task graph weight is an amount of exchanged data and system graph edges weight is data exchange “speed” ratio (Shared Memory = 1, IB = 2). The sum of the products of corresponding mapped edge weights is the objective function.

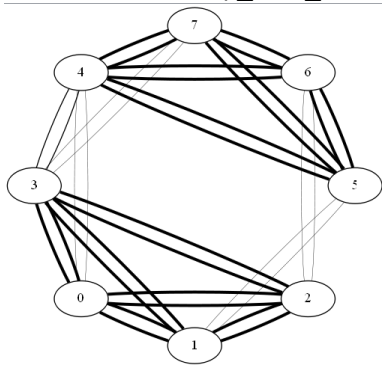
Optimization algorithm flowchart



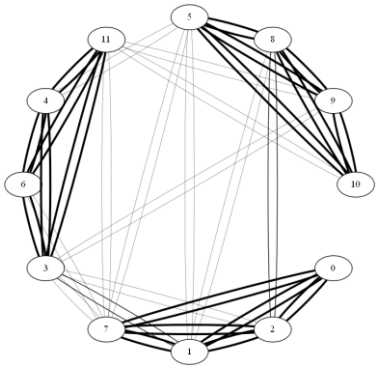


Samples of task and system graphs

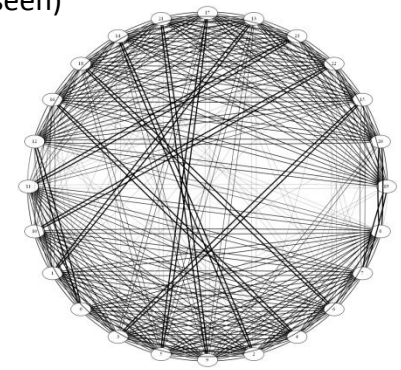
Synthetic test graph
(2 groups of 4 processes) based on Intel MPI statistics(I_MPI_STATS)



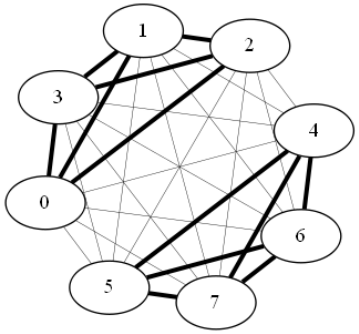
Synthetic test graph
(3 groups of 4 processes)



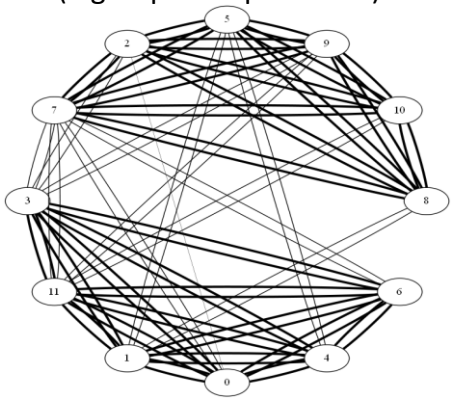
QBox graph
(2 groups of 12 processes can be seen)



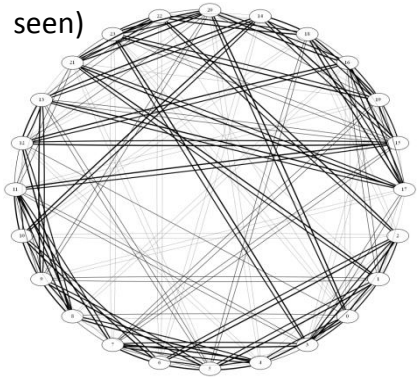
System graph
(cluster of 2 nodes with 4 cores in each)



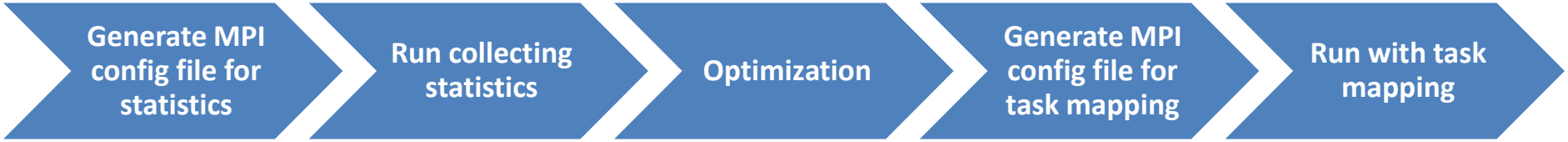
Synthetic test graph
(2 groups of 6 processes)



ZeusMP graph
(2 groups of 12 processes can be seen)



Launch and optimization methodology



Launch MPI application with config:

```
mpirun --configfile Config.txt
```

Sample Config.txt for collecting data exchange statistics on 4 cluster nodes:

```
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt -n 4 -host node1 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt -n 4 -host node2 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt -n 4 -host node3 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt -n 4 -host node4 /path/to/application
```

Custom optimizer (developed software)

Two-phase Optimizer based on greedy algorithm is developed:

Phase 1: Collect test launch statistics – launches application with small benchmark to collect exchange statistic and build the task graph

Phase 2: Optimize and generate config file with task mapping. Launch application with optimized config file.

Input data:

1. Host list, which is used to generate config file to collect communication statistics
2. Working directory
3. Application path and parameters
4. Extra parameters which will be included to the generated config file
5. **For phase 2 only:** Statistics file, which was generated by **test launch** of application with enabled I_MPI_STATS mode.

Output data:

1. Config file to collect exchange statistics (before test launch)
2. Problem graph in GraphViz representation (after test launch)
3. Config file to run application (after optimization)

Custom optimizer (developed software)

Usage example

Command line to generate config with “Optimizer” to collect communication statistics

Optimizer **-H hostlist.txt -A "/path/to/application" -S original.cfg** --extra="-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt -wdir /path/to/work/dir"

Command line to generate optimized config with “Optimizer”

Optimizer **-H hostlist.txt -A "/path/to/application" –O optimized.cfg** --extra="-env I_MPI_STATS 4 -wdir /path/to/work/dir " **--statsfile=stats.txt**

hostlist.txt – path to file with list of nodes name. **Content of hostlist.txt**:

node400 12

node402 6

node400 and node402 here are nodes name, 12 and 6 are numbers of cores on this nodes correspondingly.

–A “/path/to/application”. This path should point to MPI application to run.

In a field “—extra” additional parameters for mpirun program are given:

“-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat.txt” – this is needed to gather statistics from application run and a path “/path/to/stat/file.txt” should point to file to store statistics. Directory “/path/to/stat.txt” should be accessible for writing for the current user.

“-wdir /path/to/work/dir” – this option specify work directory for application. Usually it should point to directory with test data.

“--statsfile=stats.txt” – this option specify where program “Optimizer” should search file with gathered statistics.

Launch and optimization methodology

The example of generated config for 4 nodes (node448, node449, node453, node454), each of them has 4 cores (content of original.cfg):

```
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 4 -host node448 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 4 -host node449 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 4 -host node453 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
/path/to/work/dir -n 4 -host /path/to/application
```

Launch and optimization methodology

The example of optimized config (content of optimized.cfg):

```
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 2 -host node448 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 2 -host /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 1 -host node449 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 2 -host node448 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 3 -host node449 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 4 -host node453 /path/to/application
-env I_MPI_STATS 4 -env I_MPI_STATS_FILE /path/to/stat/file.txt -wdir
        /path/to/work/dir -n 2 -host /path/to/application
```

Launch and optimization methodology

Config.txt for native mode launch on 2 Intel® Xeon Phi™ nodes:

```
-n 60 -host mic0 /home/test//path/to/application
-n 60 -host mic1 /home/test//path/to/application
```

Fragment of Config.txt after task mapping

```
-n 5 -host mic0 /home/test//path/to/application
-n 1 -host mic1 /home/test//path/to/application
-n 4 -host mic0 /home/test//path/to/application
-n 2 -host mic1 /home/test//path/to/application
-n 1 -host mic0 /home/test//path/to/application
-n 3 -host mic1 /home/test//path/to/application
-n 4 -host mic0 /home/test//path/to/application
-n 1 -host mic1 /home/test//path/to/application
-n 1 -host mic0 /home/test//path/to/application
-n 1 -host mic1 /home/test//path/to/application
-n 3 -host mic0 /home/test//path/to/application
-n 1 -host mic1 /home/test//path/to/application
-n 1 -host mic0 /home/test//path/to/application
-n 2 -host mic1 /home/test//path/to/application
-n 1 -host mic0 /home/test//path/to/application
-n 1 -host mic1 /home/test//path/to/application
-n 4……
```

Cluster configuration

«Tornado SUSU» Supercomputer

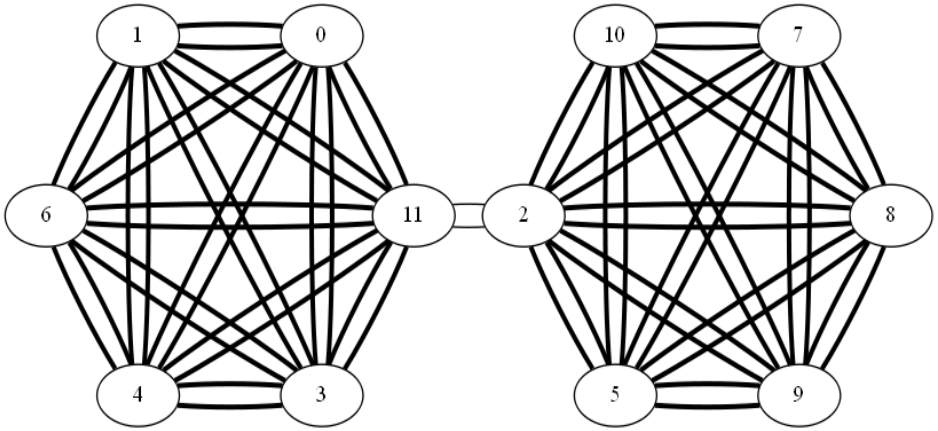
http://supercomputer.susu.ac.ru/en/computers/tornado/

Number of nodes/cores:	1-400/ 12-4800*
CPU:	Intel® Xeon™ X5680 (Gulftown, 6 cores 3.33 GHz) — 2 CPU per node
COPROCESSOR:	Intel® Xeon Phi™ SE10X (61 cores)
RAM:	24GB per node DDR3-1333
Interconnect:	InfiniBand™ QDR (40 Gbit/s)
Control network:	Gigabit Ethernet
OS	Linux Cent OS 6.2
Benchmarking Software	Coral QBox 1.40b
Build Tools	Intel® C++ Compiler XE 14.0.1 for Linux
MPI library	Intel® MPI Library 4.1.3.045

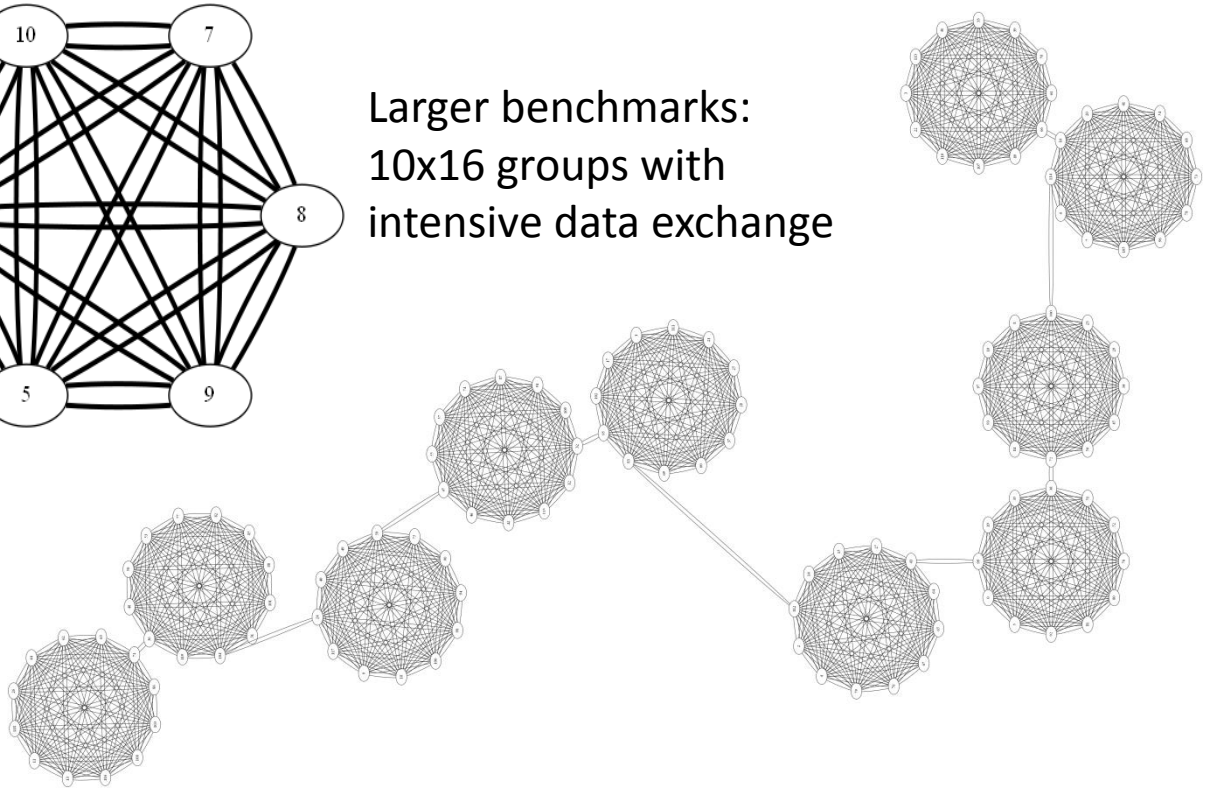
*All benchmark launches was performed with processes per node = 12

Synthetic test and results on InfiniBand and Ethernet

Synthetic MPI test: 2 groups of 6-processes communicating with each other in the group and weak inter-group communication

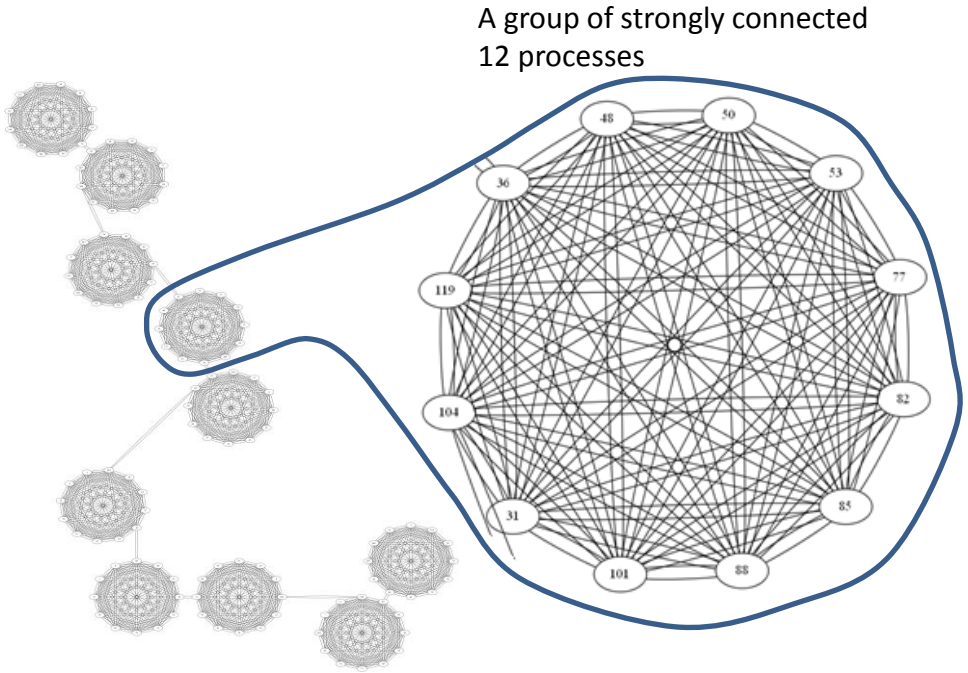


Larger benchmarks:
10x16 groups with intensive data exchange



Synthetic test and results on InfiniBand and Ethernet

Task graph (10 groups with 12 cores)



InfiniBand results, 10 nodes

$N_{iterations}$	100	1000	10000
V_{intra}	1000000	100000	10000
V_{inter}	1000	100	10
t, sec	1755	23.21	5.96
t_{opt}, sec	880	9.56	1.01
$S=t/t_{opt}$	2	2.42	5.9

Ethernet results, 4 nodes

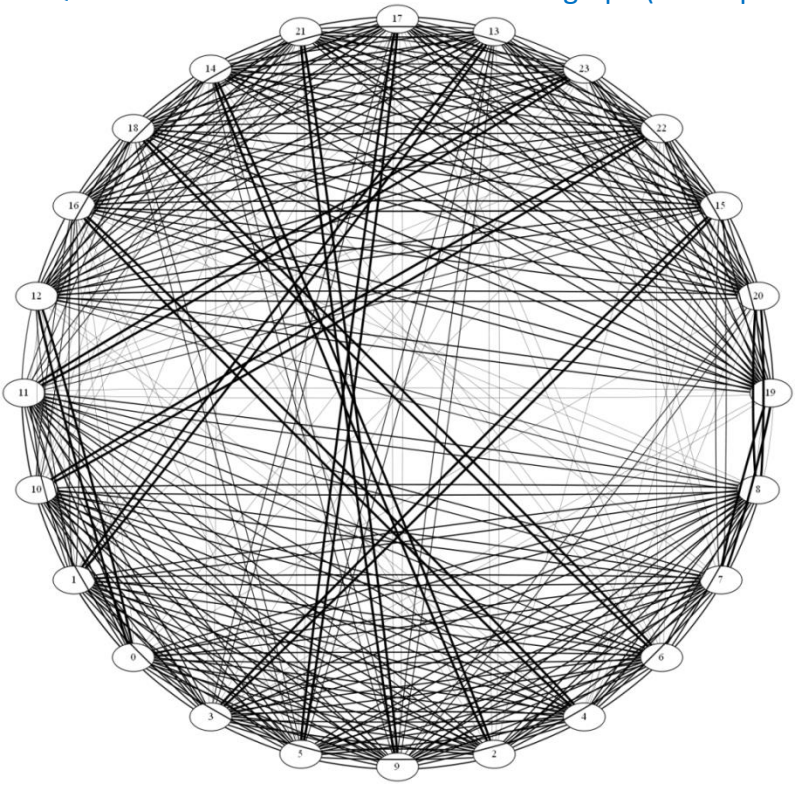
$N_{iterations}$	1000
V_{intra}	100000
V_{inter}	100
t, sec	962
t_{opt},sec	41
$S=t/t_{opt}$	23

$N_{iterations}$ – number of iterations,
V_{intra} – intra node exchange data size (number of doubles),
V_{inter} – inter node exchange data size (number of doubles),
t – initial time (sec.),
t_{opt} – optimized time (sec.)

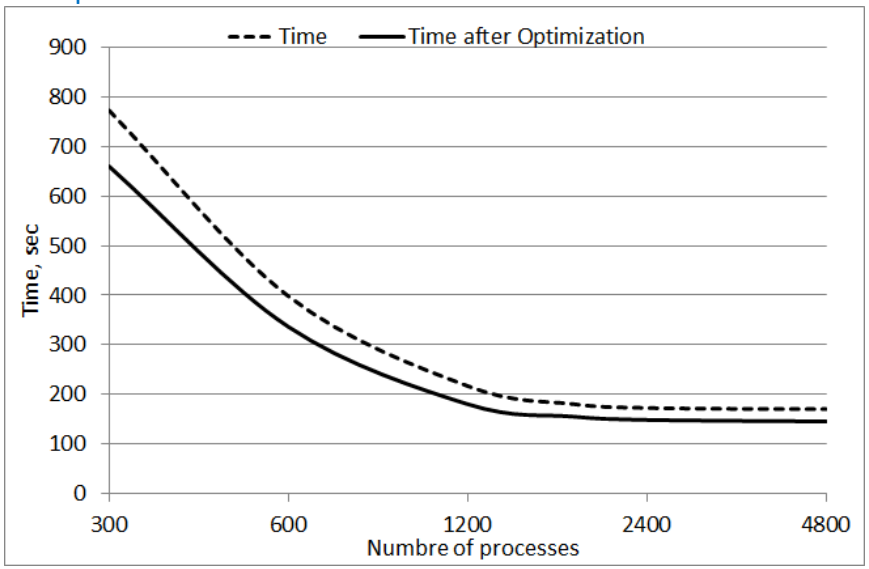
Summary: A synthetics benchmark with tight connected processes gives a good speedup on 10 nodes connected by InfiniBand and even better speedup on 4 nodes connected by 1GB Ethernet network

Synthetic test and results on InfiniBand and Ethernet

QBox non-uniform communication task graph (2 x 12 processes)



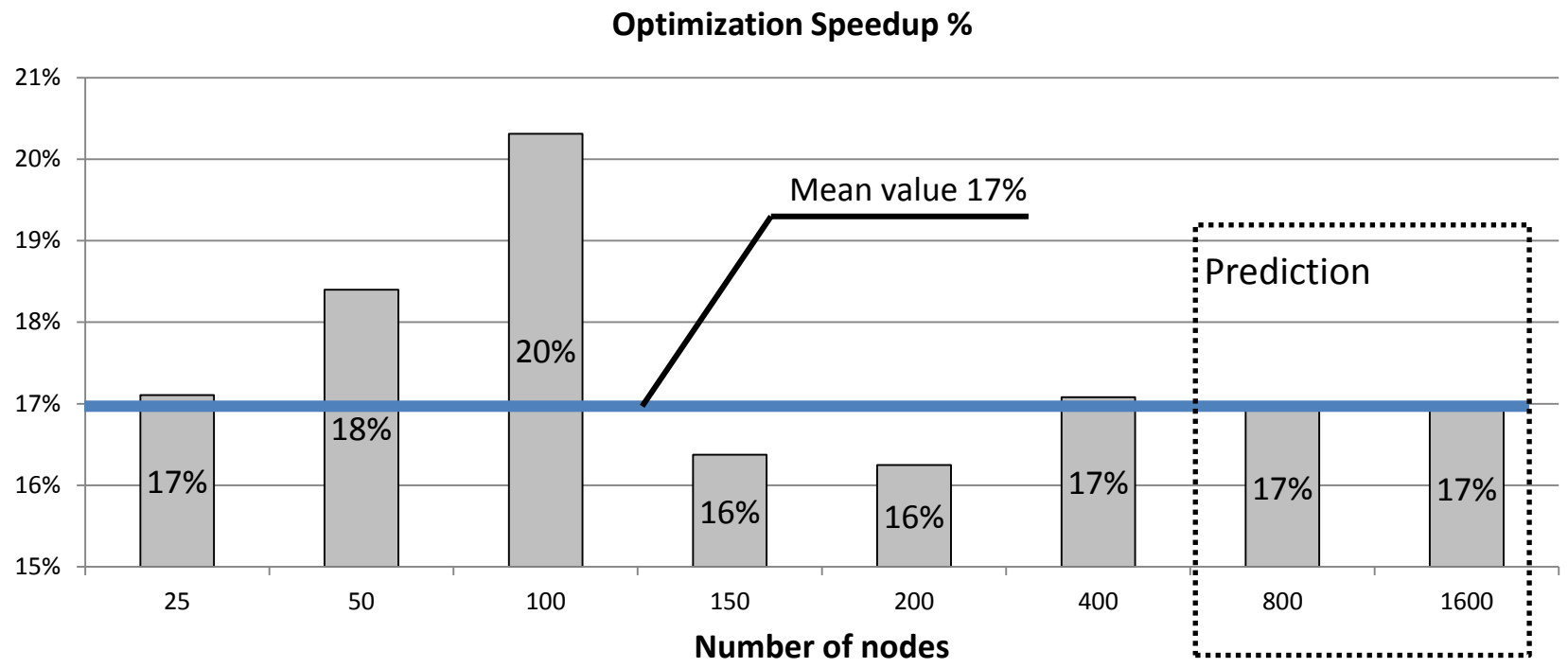
Collected run time results on SUSU Torando cluster for up to 4800 parallel processes



Parallel tasks	QBox grid dimension	$T, sec.$	$T_{opt}, sec.$	S
300	25x12	773,5	660,5	1,171083
600	50x12	398,3	336,4	1,184007
1200	100x12	216,8	180,2	1,203108
1800	150x12	180,5	155,1	1,163765
2400	200x12	172,4	148,3	1,162508
4800	400x12	170	145,2	1,170799

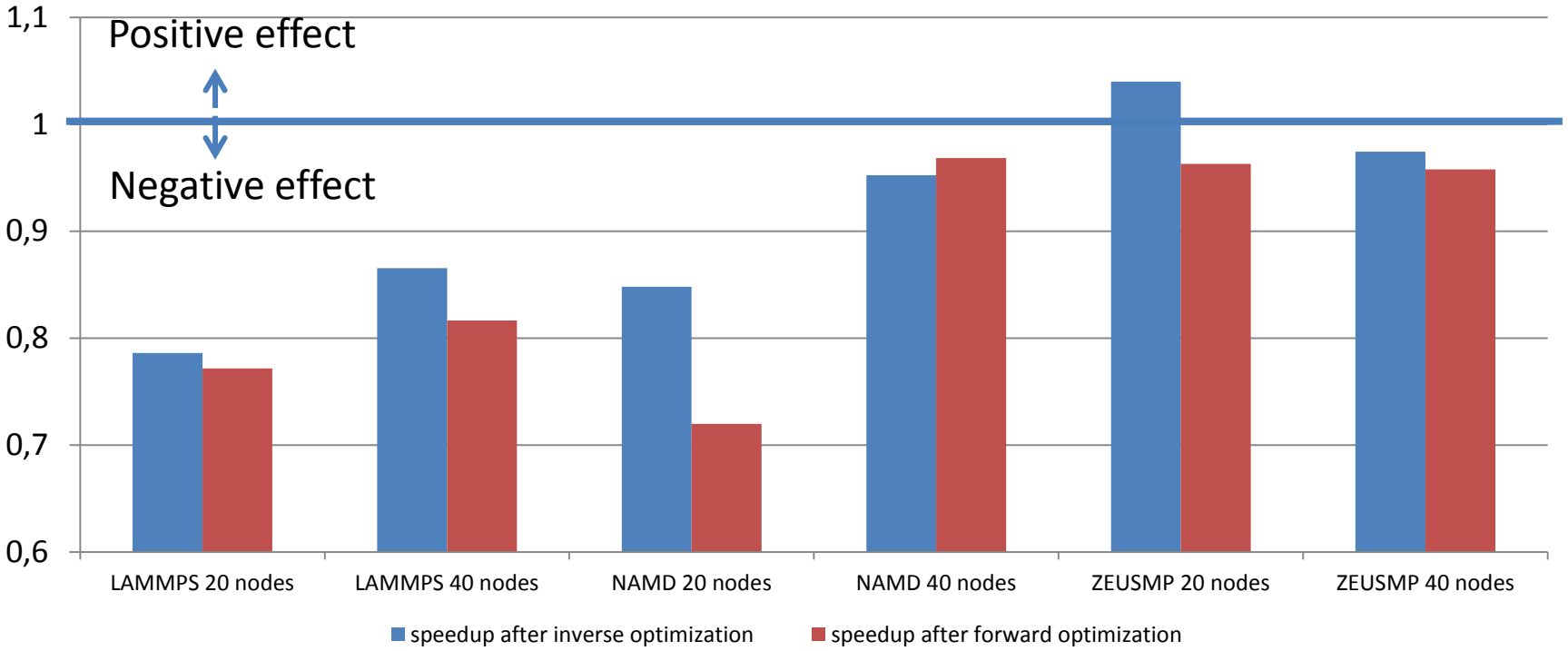
Summary: The considered Qbox benchmark has a good scalability up to 100 nodes. On greater number of nodes efficiency decreases. On larger task scalability is better (benchmarks in paper [9]) and it is possible to get better results.

QBox benchmarking results



Summary: Forward optimization (mapping intensive data exchange to shared memory) improves performance for 16-20%. The prediction for >1000 nodes is the same 16-20%. Inverse optimization (mapping intensive data exchange to InfiniBand) is better in custom cases (paper [2]) and can be achieved only considering network topology.

Benchmarking results for LAMMPS, NAMD, ZeusMP



Summary: Optimization is ineffective for various reasons for other tested HPC software benchmarks (LAMMPS, NAMD, ZeusMP). The cause is mostly bad scalability of benchmarks and initial sequential mapping best optimized for computations so remapping is ineffective.

Efficiency analysis of the proposed method for HPC software

Optimization with positive effect is possible:

1. Software with non-uniform collective communication between large amount of parallel processes (using MPI functions MPI_AllToAll, MPI_AllGather, MPI_AllScatter, etc. in source code) in case of existing processes groups with tight communications
2. For any number of nodes in case of slow interconnect (Ethernet)
3. For large number of nodes (>10 nodes) in case of fast interconnect (InfiniBand)
4. In case of large amount of data exchanged

Efficiency analysis of the proposed method for HPC software

Optimization with negative effect is possible:

1. For HPC software with low scalability on large amount of cores (LAMMPS, NAMD)
2. For software with uniform communications (LINPACK, etc.)
3. For software run on small number of nodes (< 10 nodes) and InfiniBand interconnect
4. For software where initially consequent mapping of the processes gives maximum performance of communications (e.g. grid decomposition)
5. For software with small run time

Possible next steps

1. Complication of the objective function taking into account the bandwidth and latency
2. Adding a network topology to system graph
3. Adding new system graph levels (e.g. coprocessor Intel® Xeon Phi™)
4. Benchmarking on larger amount of software
5. Researching the prediction possibility of performance gain by initial run on smaller number of nodes

Intel MPI 5.1 Beta static tuning

UI

- mpitune --rank-placement --hostfile-in hosts.in -a \"mpiexec.hydra ... my_app\“

Preparation phase

- collecting statistics then extraction of data transfers
- measurement of latencies by forked “mpiexec.hydra -machinefile hosts.in topo_graph_gen“

Optimization and result

- permutation of host list, time-limit for suboptimal solution (10 sec. by default)
- result is hostfile-out and configuration file

Usage

- mpiexec.hydra -tune file.cfg… my_app
- mpiexec.hydra -configfile file.cfg… my_app
- mpiexec.hydra -machinefile hostfile-out… my_app

References

[1]Intel MPI Reference Manual // http://software.intel.com/sites/products/documentation/hpc/ics/impi/41/lin/Reference_Manual/index.htm

[2]Shared Memory Communication vs InfiniBand // http://www.nsc.liu.se/~pla/blog/2013/09/12/smp-vs-infiniband/

[3] Hu Chen, et al. MPIPP: an automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters

[4] QBox summary // https://asc.llnl.gov/CORAL-benchmarks/Summaries/QBox_Summary_v1.2.pdf

[5] NAMD // http://www.ks.uiuc.edu/Research/namd

[6] LAMMPS // http://lammps.sandia.gov/

[7] Zeus MP // http://www.netpurgatory.com/zeusmp.html

[8]Cornell Virtual Workshop https://www.cac.cornell.edu/VW/Optimization/levelscomm.aspx

[9] Large-Scale First-Principles Molecular Dynamics simulations on the BlueGene/L Platform using the Qbox code. François Gygi, Robert K. Yates, Juergen Lorenz, Erik W. Draeger, Franz Franchetti, Christoph W. Ueberhuber, Bronis R. de Supinski, Stefan Kral, John A. Gunnels, and James Sexton. SC, page 24. IEEE Computer Society, (2005)

Thank you for your attention

www.singularis-lab.com

SINGULARIS LAB

software development