IEEE CLUSTER 2025

Edinburgh, Scotland • 2–5 September

Tutorial: MPI Communication Performance Modeling

Ziheng Wang

Xi’an Jiaotong University

Hands-on Tutorial @ CLUSTER25

西安交通大學

XI'AN JIAOTONG UNIVERSITY

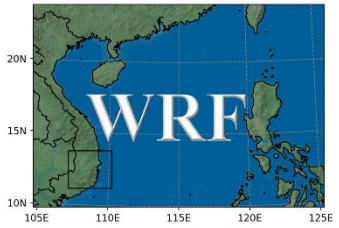
Outline

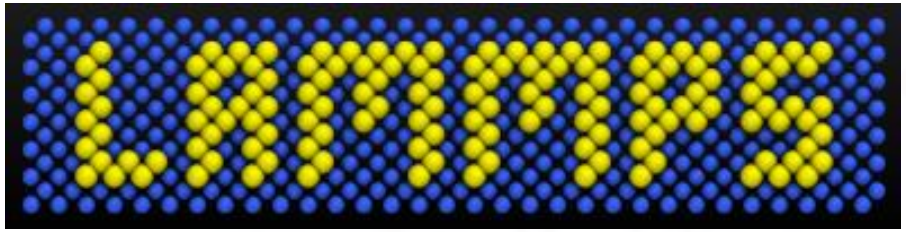
- **Introduction & Background**
- **Factors Contained in H-Lop**
- **Modeling of MPI with H-Lop**
- **Evaluation**
- **Hands-on Tutorial**
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

Outline

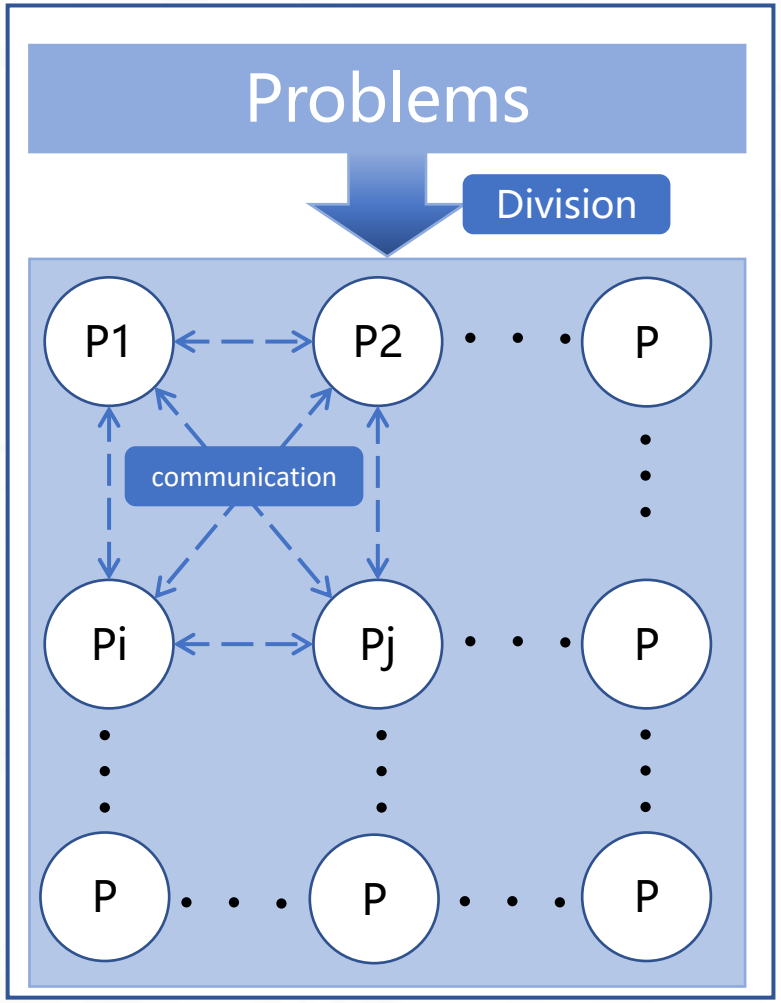
- **Introduction & Background**
- **Factors Contained in H-Lop**
- **Modeling of MPI with H-Lop**
- **Evaluation**
- **Hands-on Tutorial**
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

Communication in High Performance Computing





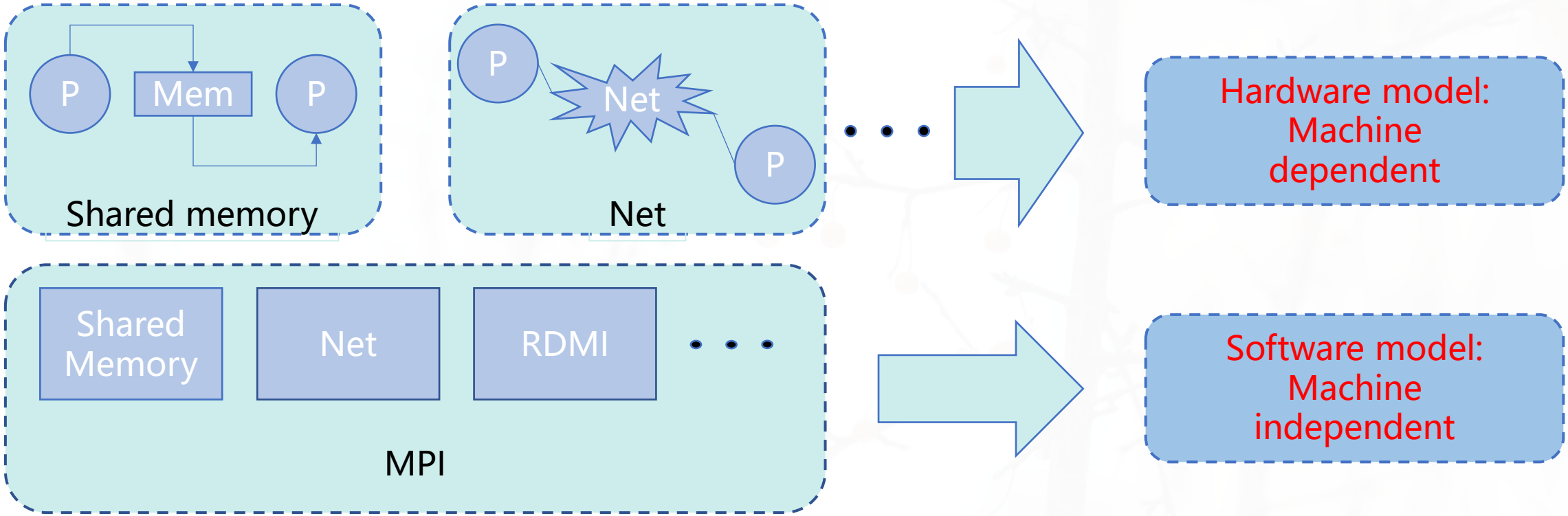
- Most HPC programs divide the computing load into many parts.
- Necessary information is transmitted between various parts through inter-process communication.



- Communication becomes one of the key!

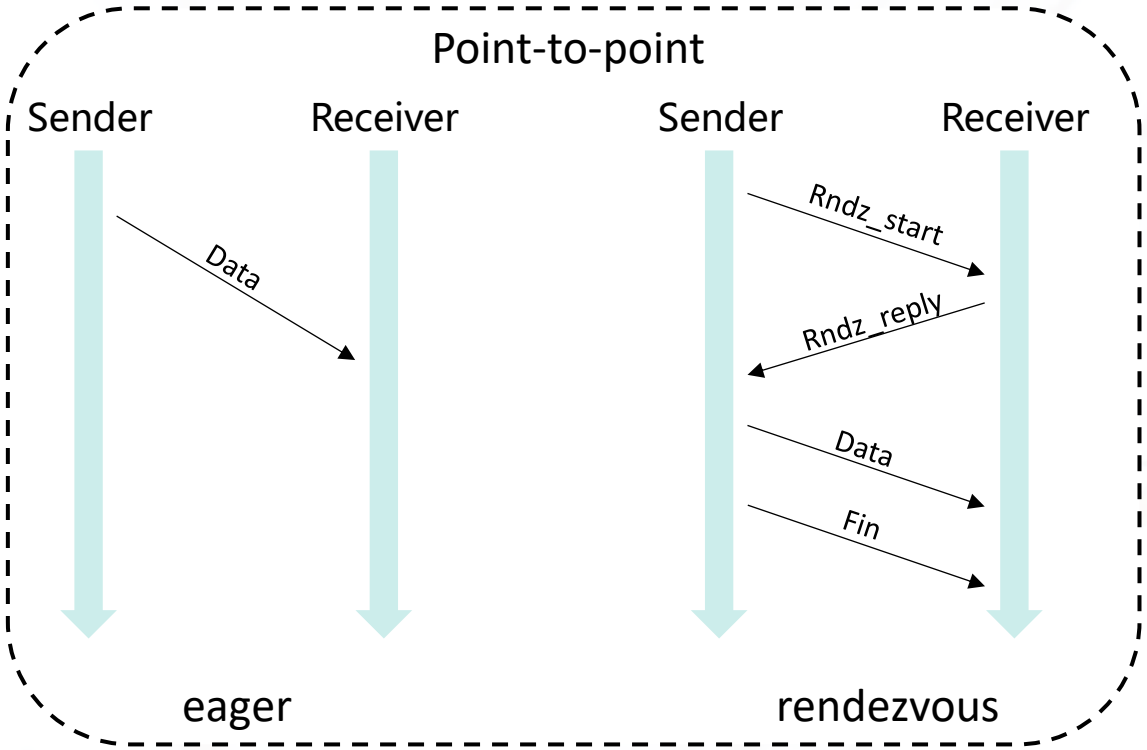
Communication performance modeling

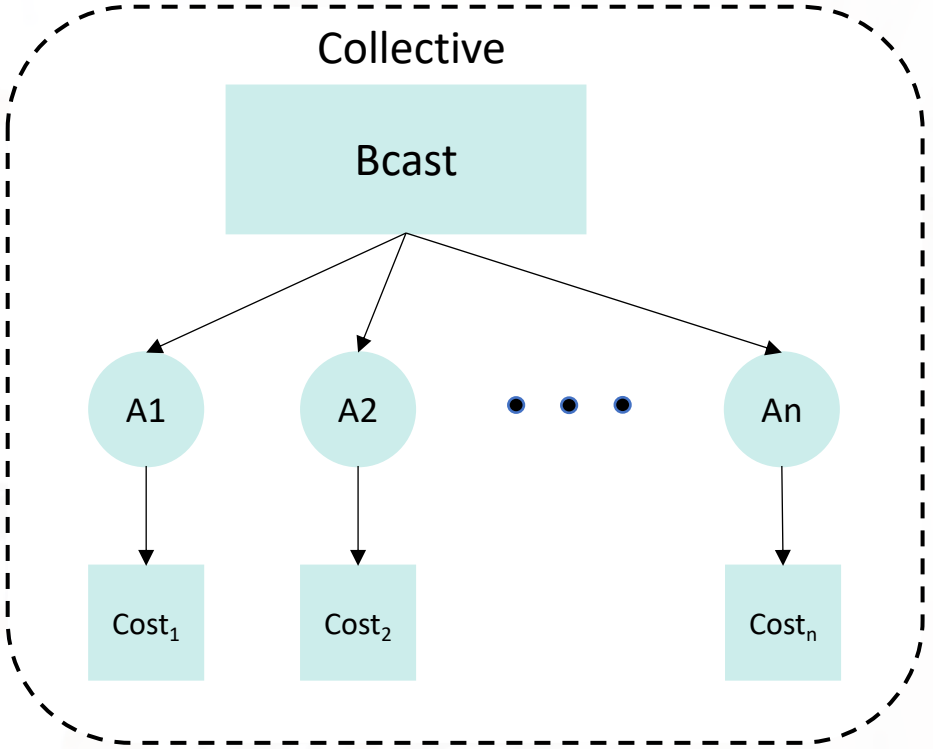
Definition: Estimation of communication performance, including bandwidth and latency

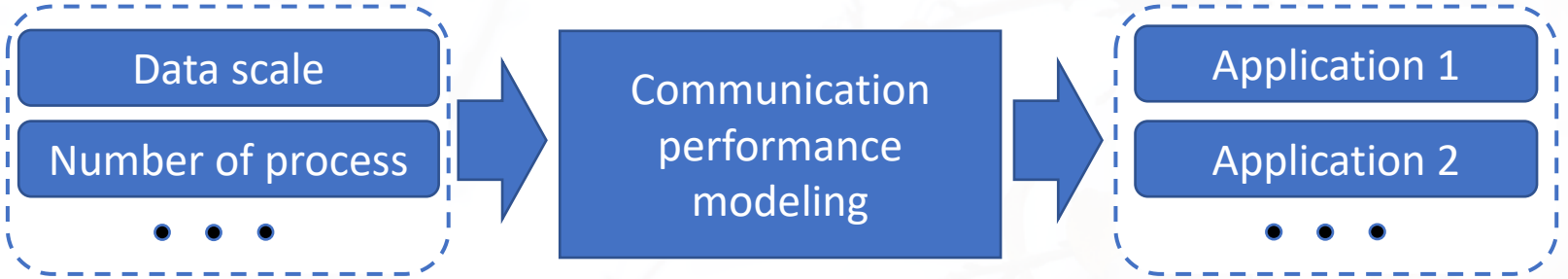


➢ Modeling of **MPI primitives** allows **machine independent** representation.

Influence factor of communication performance







- There are many factors that affect communication performance.

The application of performance model

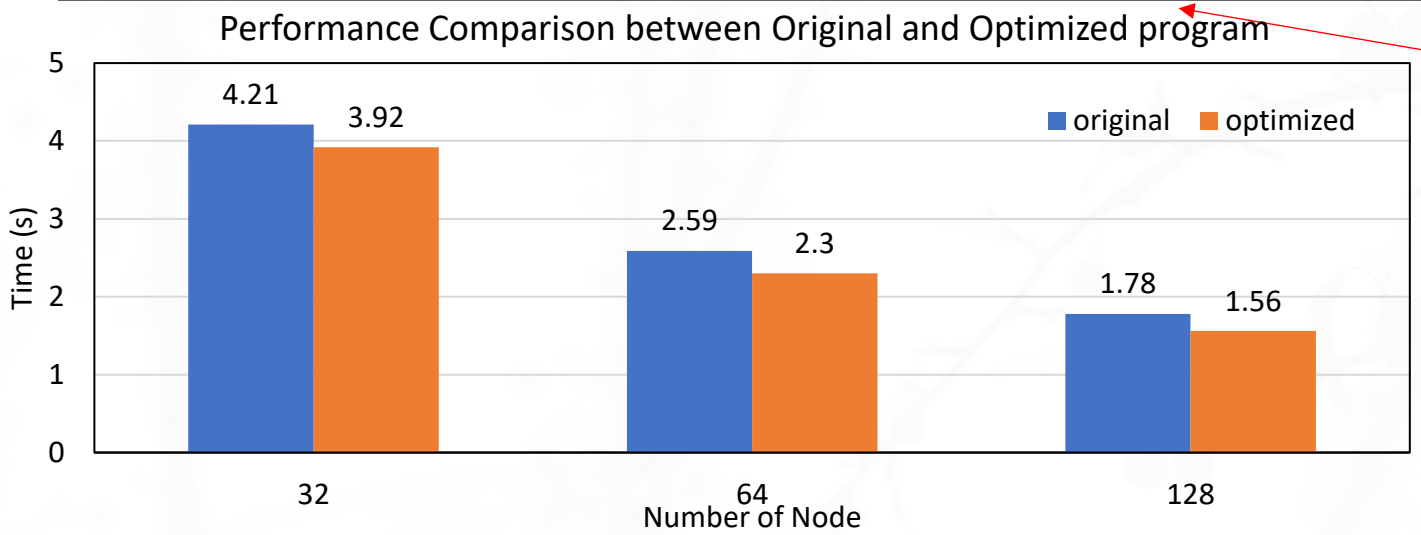
- **Performance Estimation**
  - Avoiding problems that are discovered only after deployment
- **Program Optimization**
  - Select the best hardware, protocols, and algorithms
- **Reduce costs**
  - minimize trial and error and improve resource utilization

```
1  export MPIR_CVAR_BCAST_INTRA_ALGORITHM=binomial
2  export MPIR_CVAR_CH3_EAGER_MAX_MSG_SIZE=32768
3  sbatch –N 4 –n 16 --distribution=file:task_dist.txt my_script.sh
```

Set algorithm guided with model result

Set threshold guided with model result

Processes are mapped to the specified nodes guided with model result

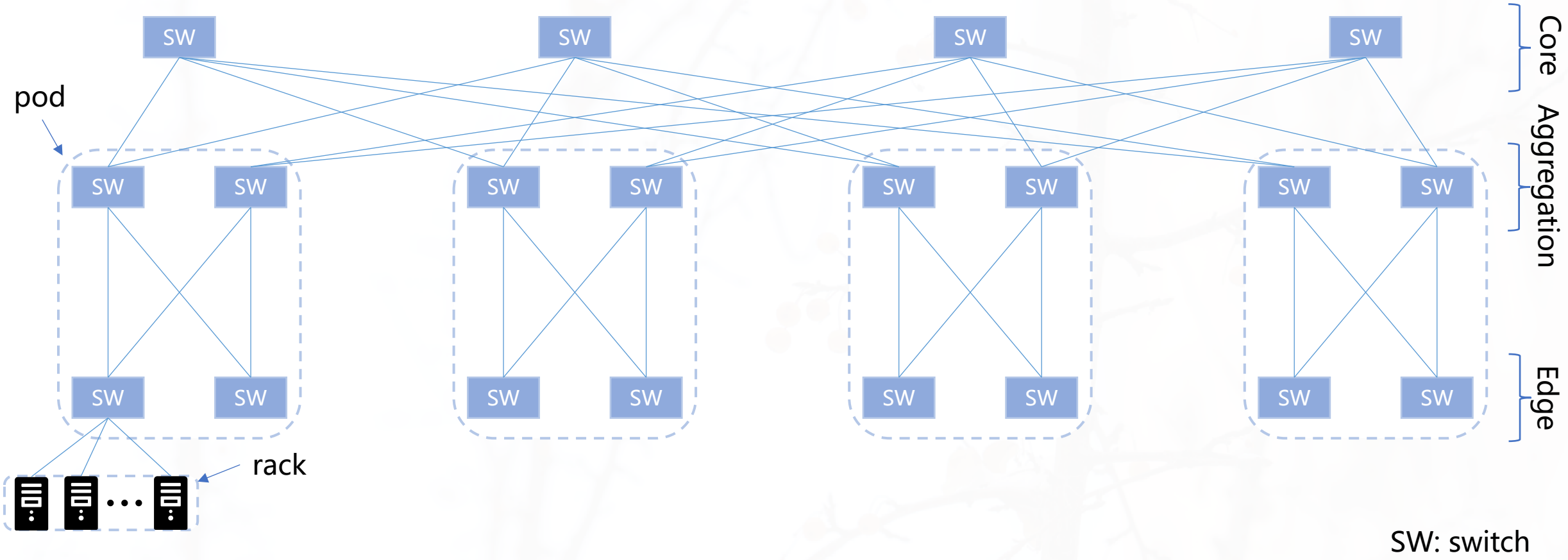


- The performance model can provide guidance for performance optimization.

Outline

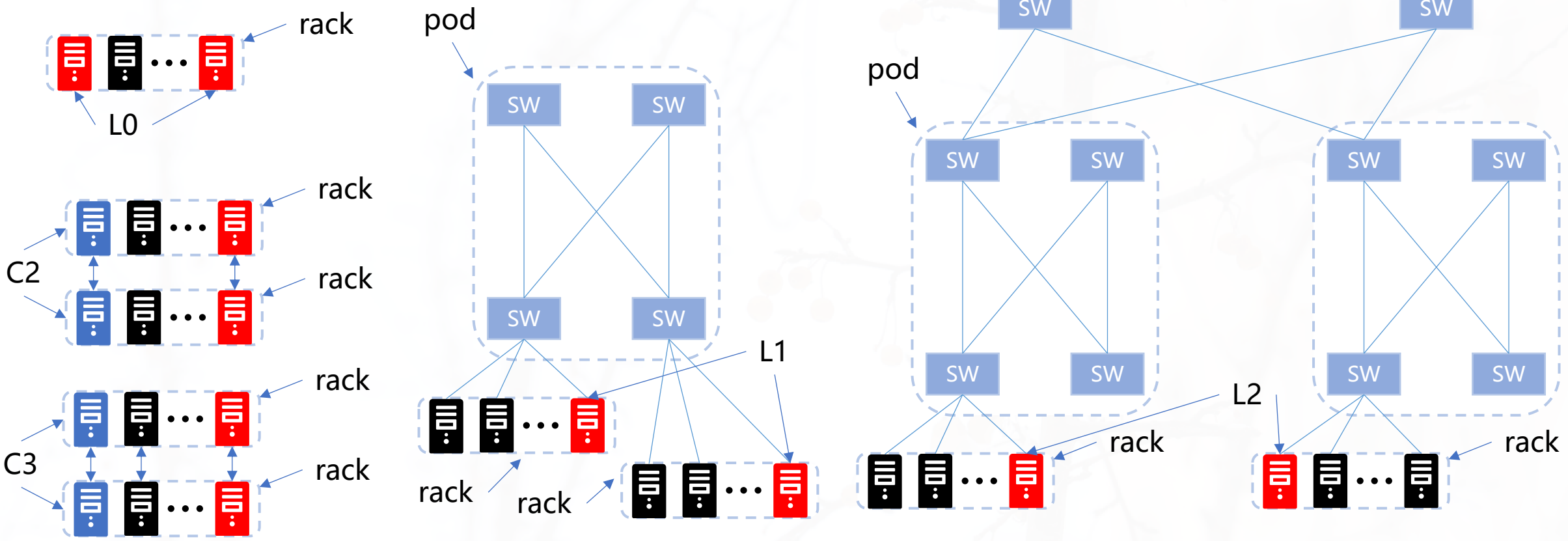
- **Introduction & Background**
- **Factors Contained in H-Lop**
- **Modeling of MPI with H-Lop**
- **Evaluation**
- **Hands-on Tutorial**
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

Typical supercomputer architecture: Three-tier fat tree structure



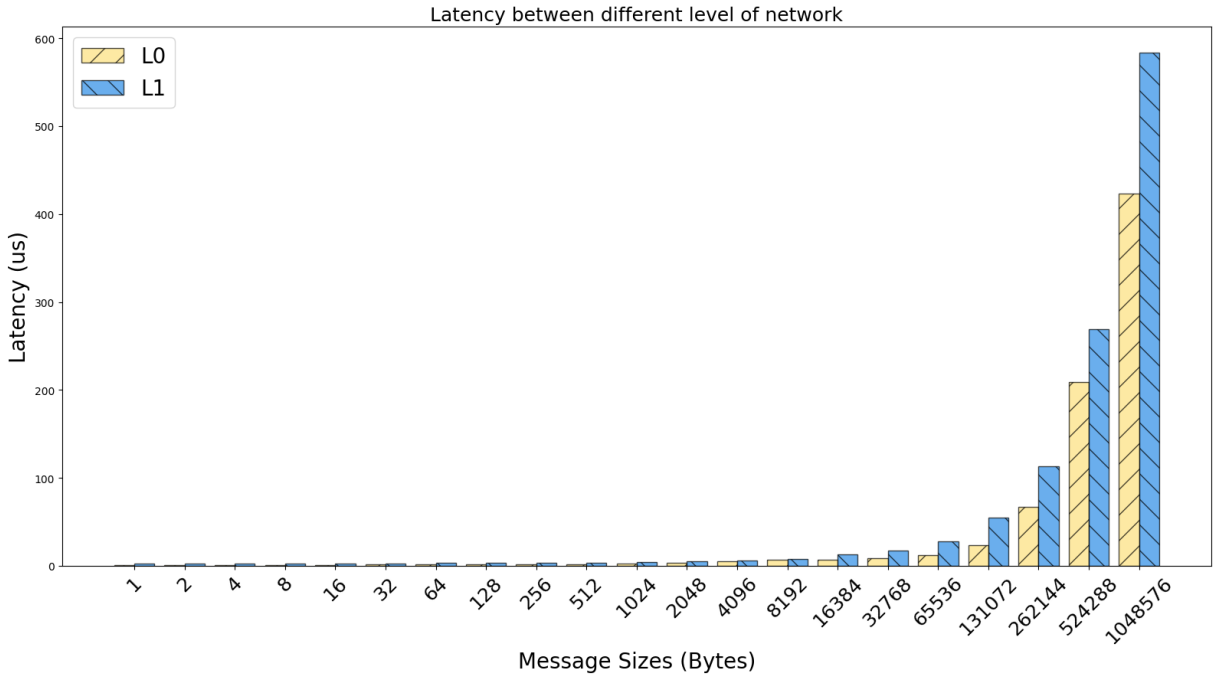
- Three-tier fat tree network leads to variable paths between nodes.
- Nodes in the same rack、in the same pod and in different pod.
- The latency and bandwidth vary in different situations.

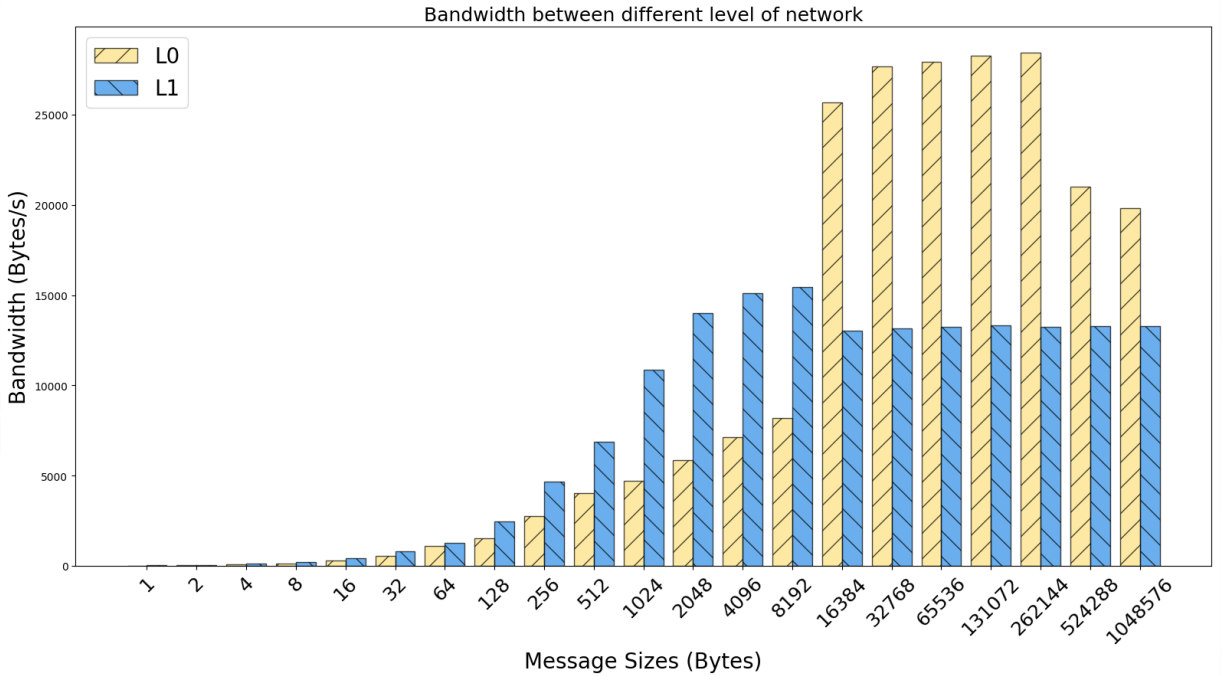
Factors in modeling



- We define the network levels (L0, L1, L2) and contention (C2, C3) as above.

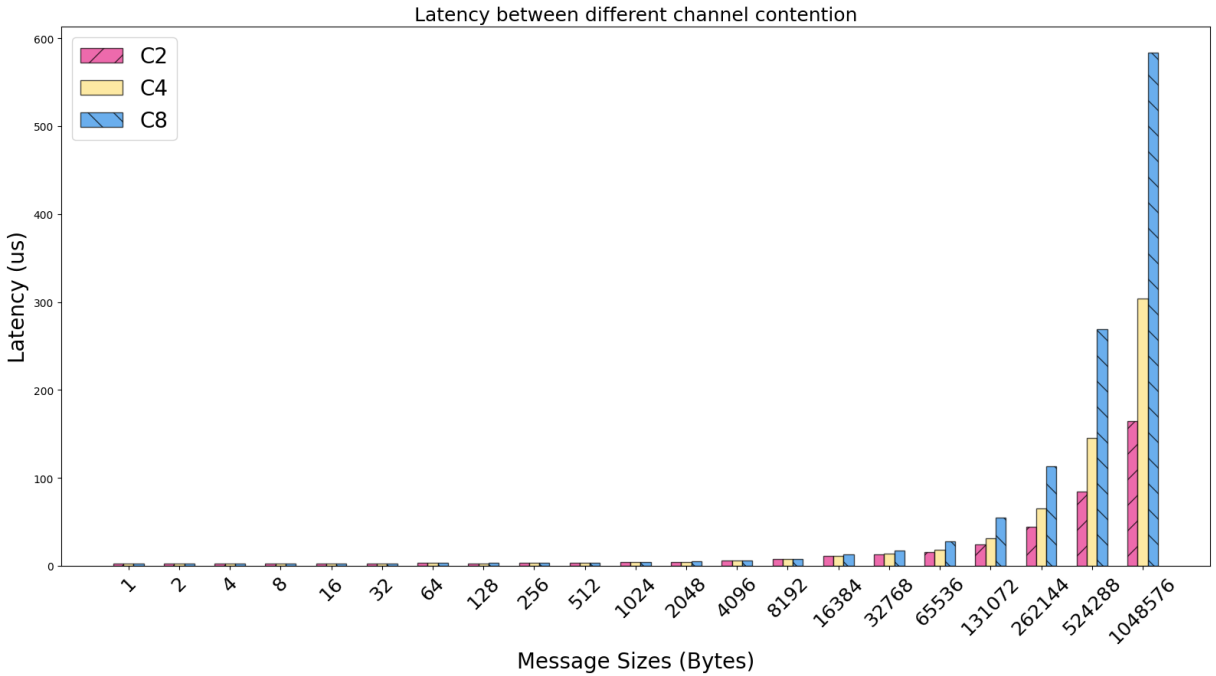
Network levels

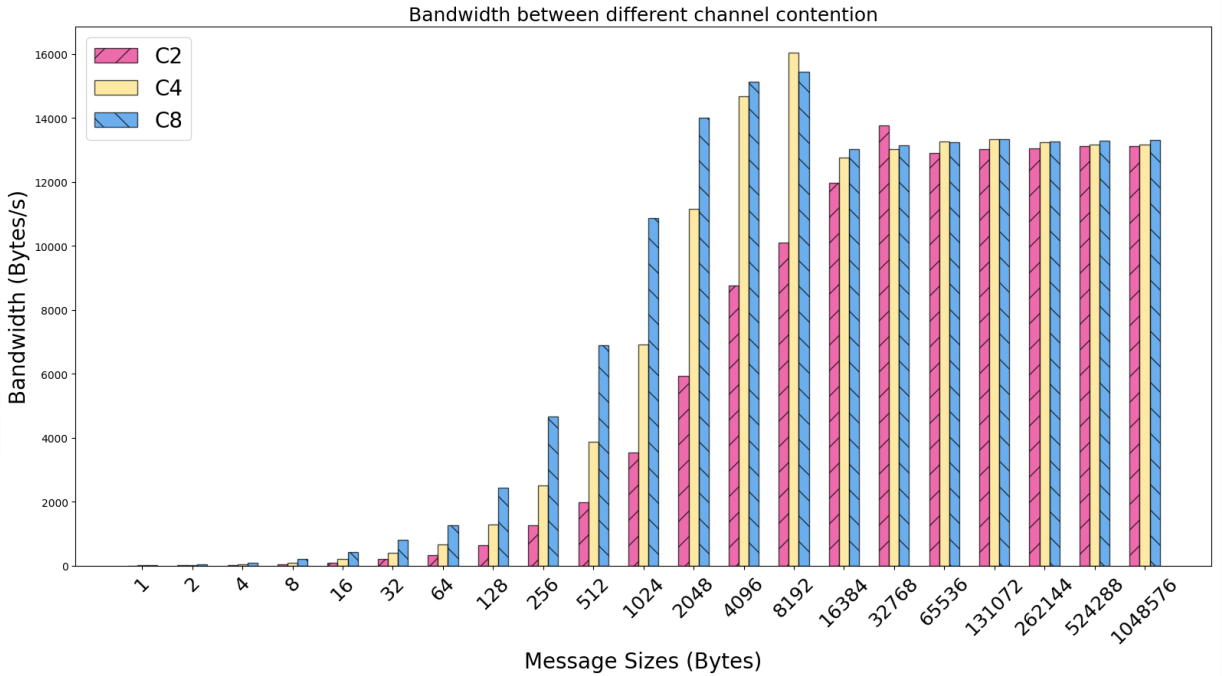




- It is shown that the relative position of nodes **has a certain impact** on the communication overhead between nodes.

Channel contention





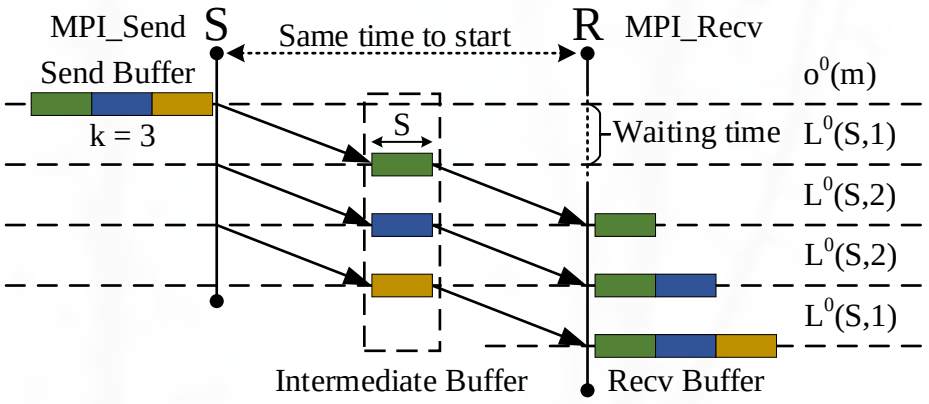
- Similar to the previous method, the contention of channels also has a significant impact on communication overhead.

Outline

- **Introduction & Background**
- **Factors Contained in H-Lop**
- **Modeling of MPI with H-Lop**
- **Evaluation**
- **Hands-on Tutorial**
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

Point-to-point modeling

- $\boldsymbol{L^c(m,\tau,C)}$: The time consumed for conducting a transfer.
- $\boldsymbol{o^c(m)}$: On channel c, the time the message transmission operation is called until the time when data begins to be injected into the channel.
- $\boldsymbol{p(m)}$: The time of packaging and unpacking non continuous stored messages.



Transmission Overhead:

$$\mathrm{T}_{p2p}^{0}(m) = o^0(m) + 2L^0(\mathrm{S},1,C) + 2L^0(\mathrm{S},2,C)$$

- Property 1, Linear Principle:

$$L^{\mathrm{c}}(A \times \mathrm{m},\tau,C) = A \times L^{\mathrm{c}}(m,\tau,C)$$

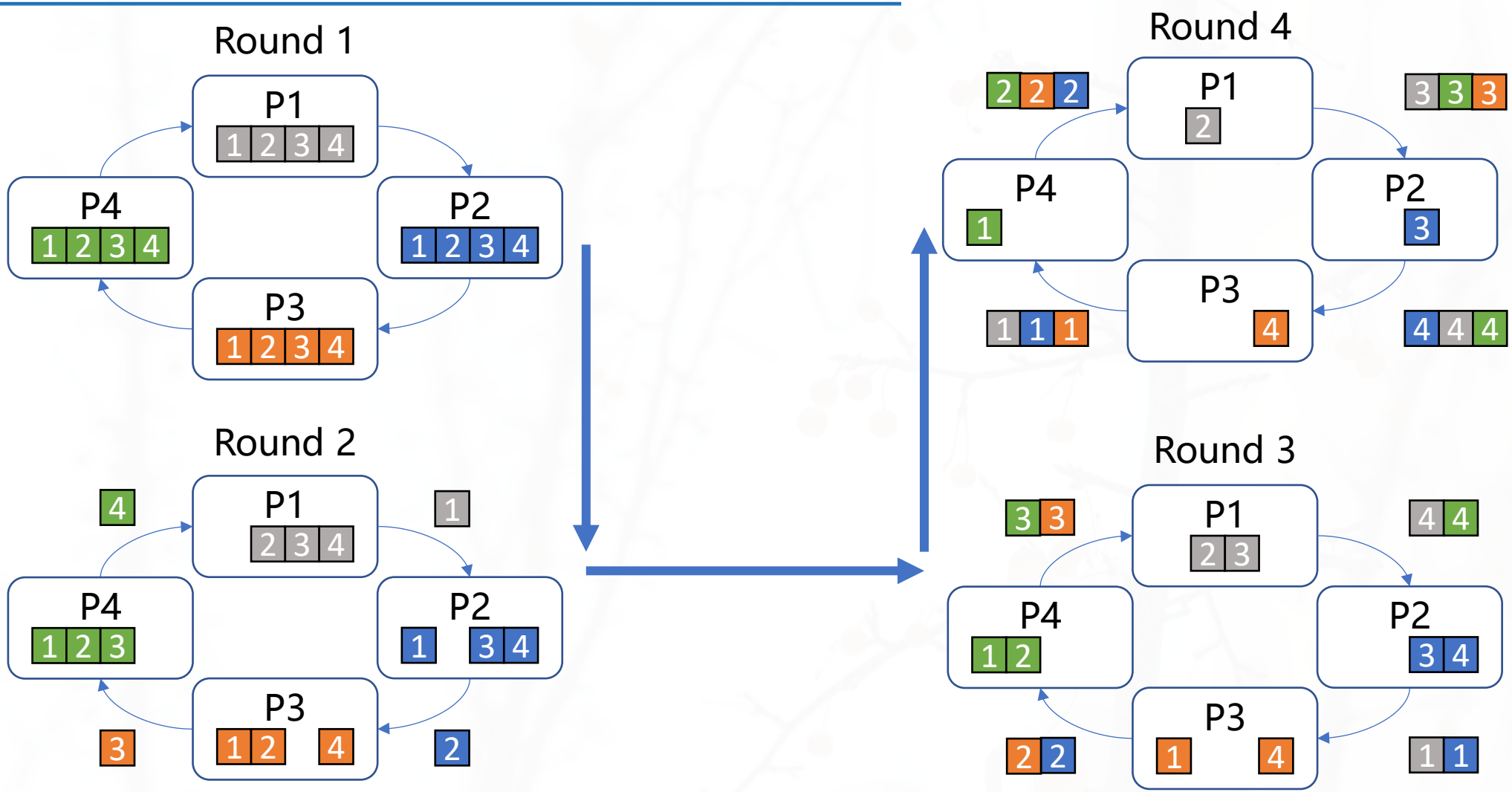
- Property 2, Frequency Unequal Principle:

$$L(m,1,C) \le L(m,\tau,C) \le \tau \times L(m,1,C)$$

- Property 3, Access and Reuse Principle:

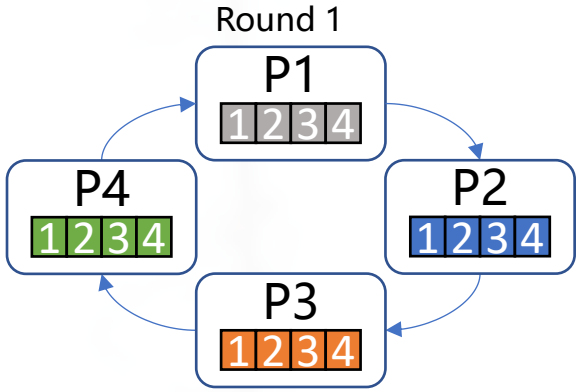
$$L(m,\tau,C_{\mathrm{r}}) \le L(m,\tau,0) \le L(m,\tau,C_{pn})$$

An example of data exchange in collective operation

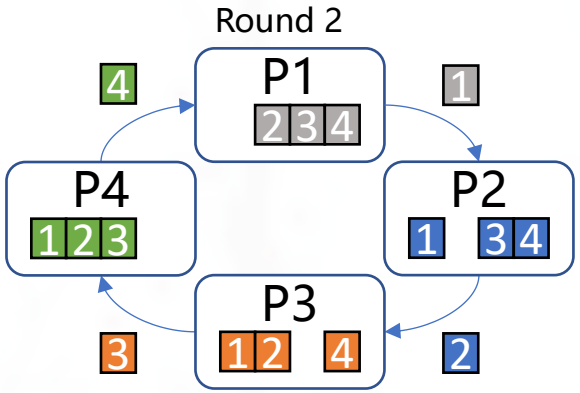


- We assume that each round of communication in collective communication is independent of each other.

Modeling with H-Lop



$$\max_{pair \in \{pair_1\}} h\left\{\frac{P}{2^2}m, \tau, L^c, ctype\right\}$$



$$\max_{pair \in \{pair_2\}} h\left\{\frac{P}{2^2}m, \tau, L^c, ctype\right\}$$

- m: message size
- τ: number of segments
- L^c: time of transfer
- $ctype$: channel type

$$T_{Allreduce} = \Sigma_{i=1}^{\log_2 P} \max_{pair \in \{pair_i\}} h\left\{\frac{P}{2^i}m, \tau, L^c, ctype\right\}$$

- Where h is the basic model of point-to-point in our model.
- It is influenced by factors m, L^c, τ, and $ctype$.

Outline

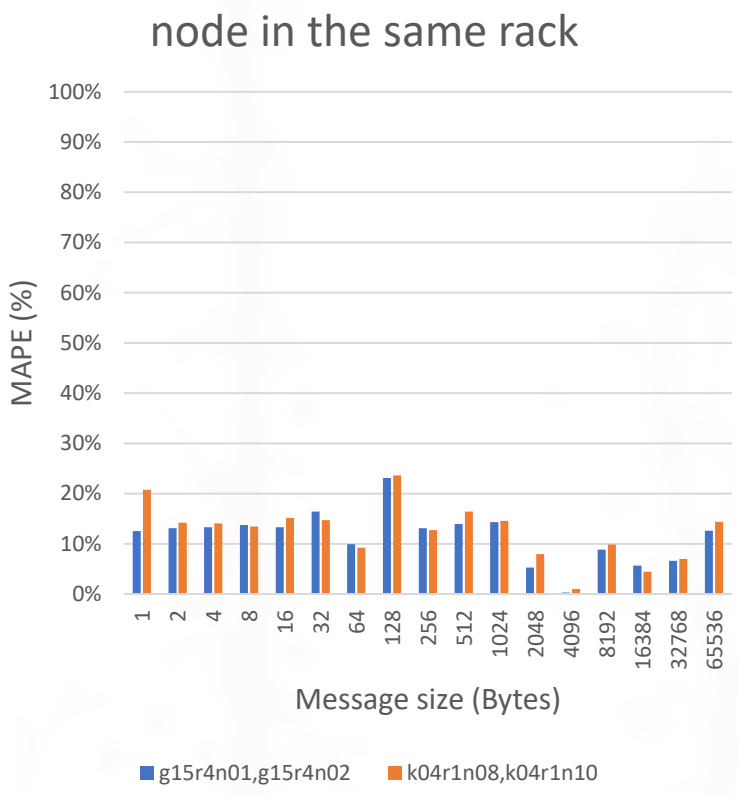
- **Introduction & Background**
- **Factors Contained in H-Lop**
- **Modeling of MPI with H-Lop**
- **Evaluation**
- **Hands-on Tutorial**
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

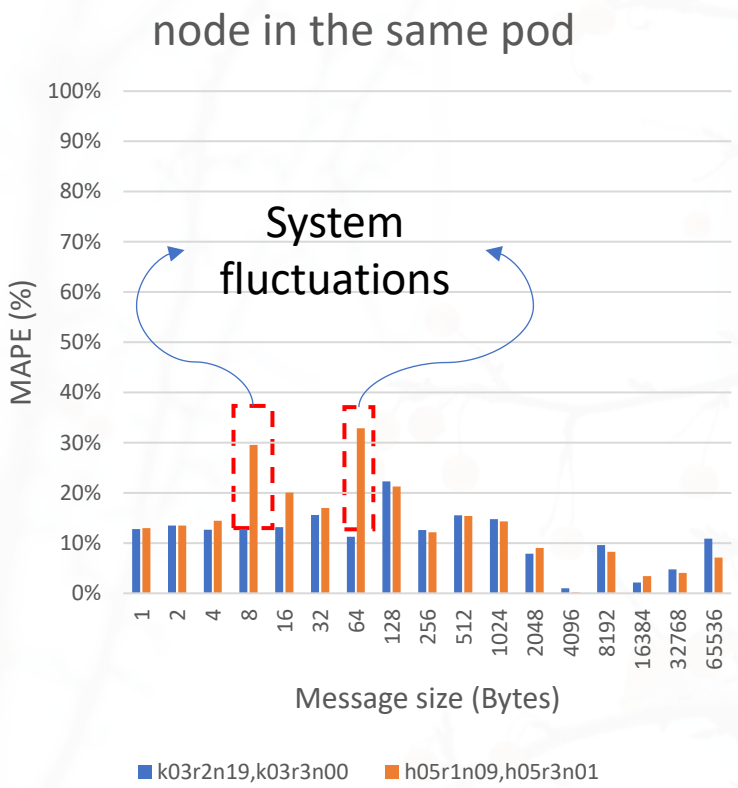
Experimental Setup

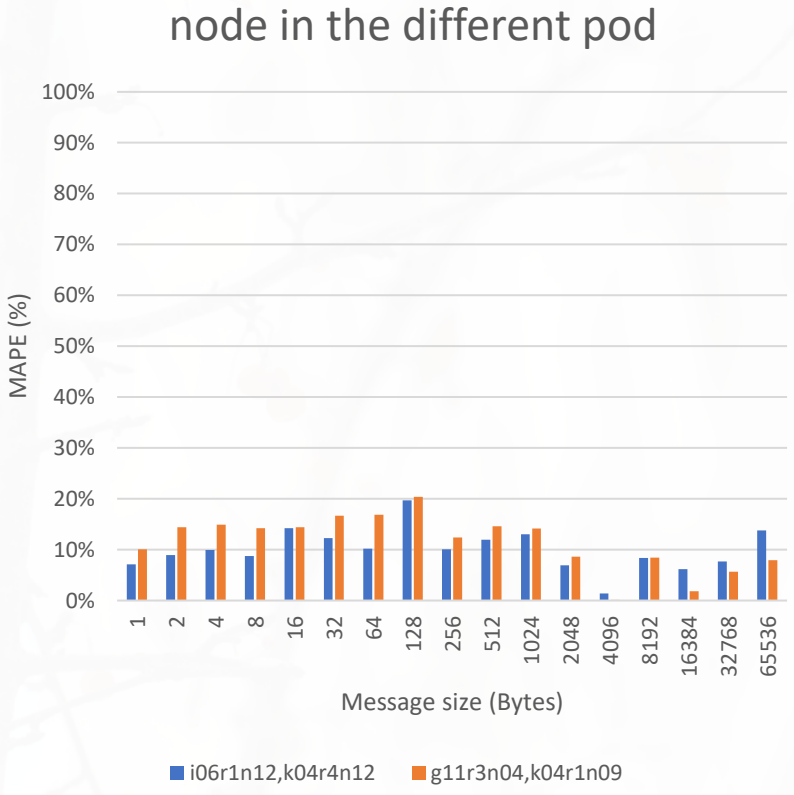
- **Server:**
  - 128 GB DDR4 memory & Hygon C86 7185 32-core Processor
  - CentOS Linux release 7.6.1810 (Core)
- **Dependency**
  - gflags 2.2.2
  - magic_enum 0.9.7
- **Representative programs**
  - OSU Micro-Benchmarks 7.3
- **Compiler**
  - gcc (conda-forge gcc 11.2.0-16) 11.2.0

Estimation on benchmark

MAPE (Mean Absolute Percentage Error) is used to present estimation error







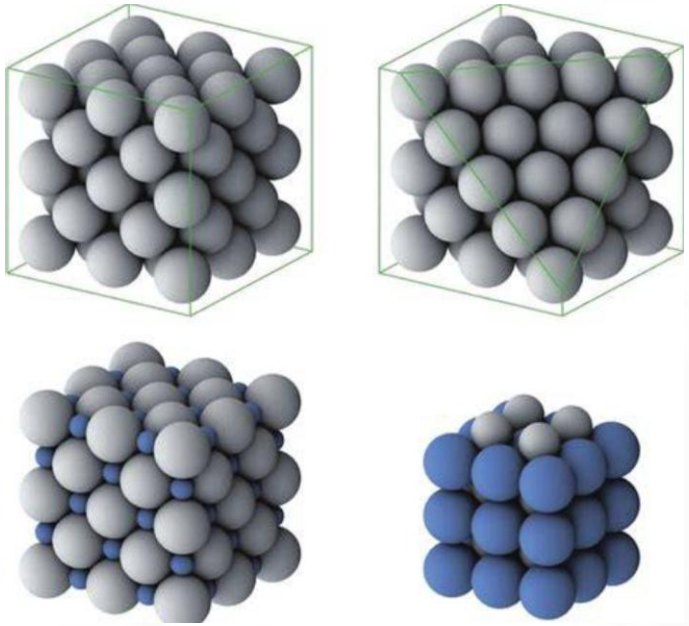
- Percentage error of nodes located in the same rack.
- The average error in this situation is about 11.5%.

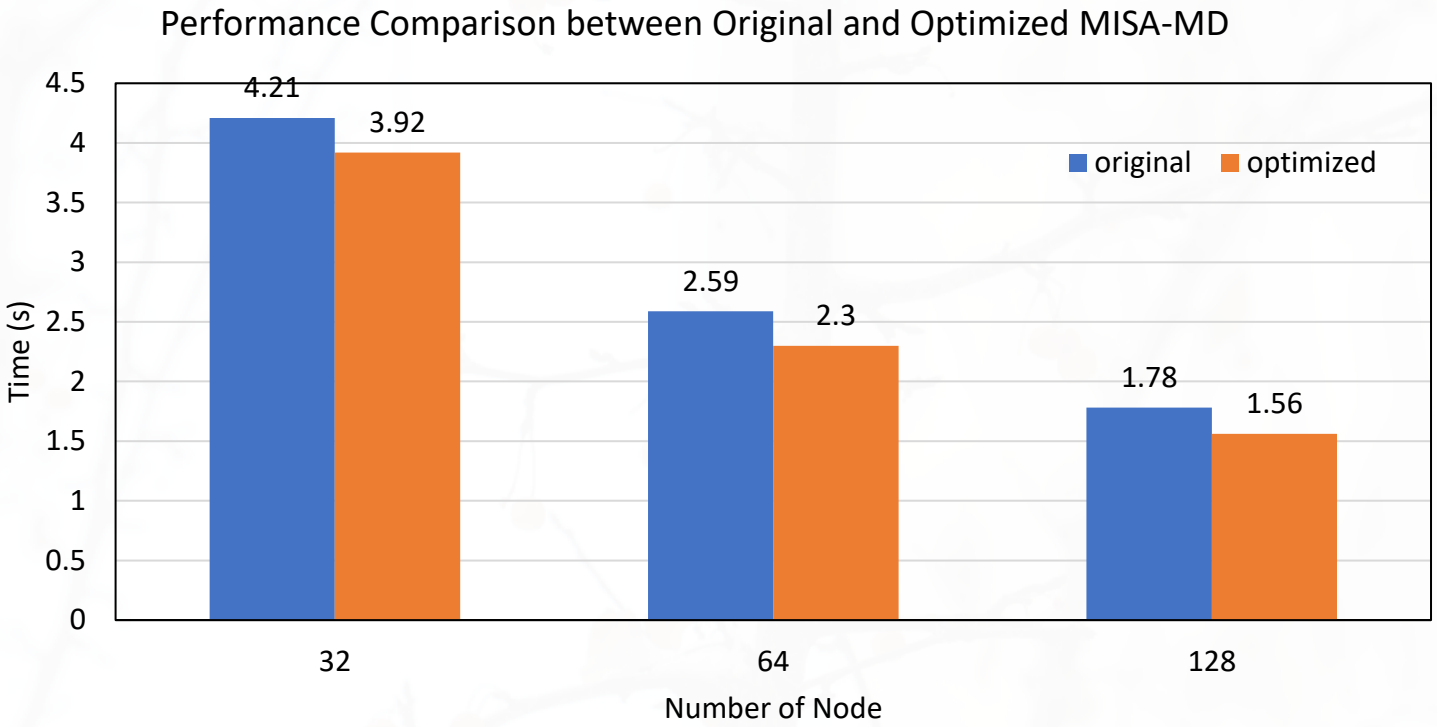
- Percentage error of nodes located in the same pod.
- The average error in this situation is about 14.2%.

- Percentage error of nodes located in different pods.
- The average error in this situation is about 12.0%.

Estimation on the application

MISA-MD is a Molecular simulation software





- The number of processes for each node is 16.
- The speedup after optimization are 7.40%, 12.61% and 14.10%, respectively.

Outline

- Introduction & Background
- Factors Contained in H-Lop
- Modeling of MPI with H-Lop
- Evaluation
- Hands-on Tutorial
  - Installation
  - Case Study – Parameter collection
  - Case Study – Operation broadcast
  - Case Study – Operation scatter

Installation - OSU Micro-Benchmarks

- **Install with source code:**
  - Dependencies: git, g++>=11.2.0, make, cmake>=3.12
  - Source Code:
    ```
    wget https://mvapich.cse.ohio-state.edu \
    /download/mvapich/osu-micro-benchmarks-7.3.tar.gz
    ```
  - Compilation Instruction:
    ```
    cd osu-micro-benchmarks-7.3 && \
     ./configure CC=mpicc CXX=mpicxx --prefix=/path/to/install
    ```
  - Configure the Path:
    ```
    export osu_home=/path/to/insatll/libexec \
    /osu-micro-benchmarks/mpi/collective
    ```
- **Install in tutorial cluster:**
  - Source Code:
    ```
    cp –r /public/home/dtune/shared_folder \
    /osu-micro-benchmarks-7.3 .
    ```
  - Compilation Instruction:
    ```
    cd osu-micro-benchmarks-7.3 && \
     ./configure CC=mpicc CXX=mpicxx –prefix=.
    ```
  - Configure the Path:
    ```
    export osu_home=./libexec/osu-micro-benchmarks \
    /mpi/collective
    ```

Installation – H-Lop

- **Install with source code:**
  - Dependencies: git, g++>=11.2.0, make, cmake>=3.12
  - Source Code:
    ```
    git clone https://github.com/tututu-dual/hlop.git
    ```
  - Compilation Instruction:
    ```
    cd H-Lop && install_dir=/path/to/install bash build.sh
    ```
  - Configure the Path:
    ```
    export HLOP=${install_dir}/bin/hlop
    ```
- **Install in tutorial cluster:**
  - Source Code:
    ```
    cp –r /public/home/dtune/shared_folder/H-Lop .
    ```
  - Compilation Instruction:
    ```
    cd H-Lop && install_dir=. bash build.sh
    ```
  - Configure the Path:
    ```
    export HLOP=${install_dir}/bin/hlop
    ```
- **Instruction to modeling with target configuration:**
  - H-Lop:
    ```
    $HLOP --op=<op> --algo=<algo> --pf=<pf> --nl=<nl> \
    --ppn=<ppn> --msz=<msz>
    ```

Case Study – Parameter collection

➢ Collect parameter in our model:

```
cp param-template.sh param-coll.sh
vim param-coll.sh
bash param-coll.sh
```

```
#!/bin/bash
NTASK=32
PPN=16
JOBNAME=param
echo "START $JOBNAME WITH NTASK=$NTASK
nowdate=$(date +%Y_%m_%d_%H_%M_%S)
echo $nowdate

mkdir -p log

sbatch << END
#SBATCH --job-name=${JOBNAME}
#SBATCH --ntasks-per-node=${PPN}
#SBATCH --ntasks=${NTASK}
#SBATCH --partition=normal
#SBATCH --output=log/%j.log
for j in {1..${PPN}}; do
    CMD=(
        "mpirun"
            "-np \$((j * 2))"
            "-ppn \${j}"
            "-bind-to core"
            "\$(pwd)/mpi/pt2pt/osu_multi_lat"
            "-m 4:65536"
            "-x 5"
            "-i 100"
    )
    \${CMD[@]} | tail -n 15 | awk '{print(\$2)}' | paste -sd,
    sleep 3
done
END
```

Modify the Job Scale

Modify the JOB NAME

Modify the EXE PATH

Case Study – Parameter collection

0.4611,0.4257,0.425,0.4258,0.4258,0.4427,0.445,0.904,0.6437,0.6005,0.7705,1.0593,1.684,1.8764,5.1287,6.0015,8.07
1.3922,1.3653,1.358,1.366,1.3574,1.5219,1.5477,2.2364,1.986,2.0981,2.6757,3.3221,4.7471,7.6438,7.6862,10.1846
1.3411,1.3113,1.3006,1.2962,1.2947,1.45,1.4562,2.2011,1.9171,1.9821,2.5512,3.1564,4.3933,6.5762,8.0031,10.3123
1.5261,1.5037,1.5,1.4963,1.4989,1.7144,1.684,2.2154,2.0533,2.1392,2.5929,3.1593,4.318,6.3915,8.0457,10.7031
1.4229,1.3912,1.3894,1.3853,1.3912,1.5447,1.55,2.1986,1.9041,1.9847,2.4893,3.0,4.0614,5.8029,8.2785,11.3746
1.4429,1.3874,1.3877,1.3941,1.3893,1.5553,1.5605,2.2825,1.985,2.0579,2.5769,3.0814,4.1093,5.6,8.8562,11.82
1.4893,1.4406,1.4333,1.4315,1.432,1.5562,1.5807,2.2315,1.9306,1.9971,2.4973,2.9915,4.0214,5.4607,8.6715,11.5162
0.4724,0.4392,0.4367,0.4338,0.4359,0.4463,0.4527,0.85,0.5862,0.6536,0.794,1.0239,1.5819,1.8672,5.1527,6.0171,8.16
0.5936,0.564,0.567,0.5681,0.565,0.6047,0.6076,1.06,0.7938,0.8633,1.0244,1.2744,1.7986,2.1487,5.3246,6.2715,8.2338
0.8363,0.8086,0.8088,0.8086,0.8024,0.8933,0.8894,1.2769,1.0393,1.2187,1.59,1.9646,2.8636,4.1971,6.0357,7.1123
0.9346,0.9154,0.9021,0.9029,0.9033,1.0479,1.0286,1.6673,1.42,1.4436,1.8092,2.2307,3.1407,3.87,6.1407,7.1707
1.029,0.996,0.9936,0.9906,0.996,1.181,1.1412,1.74,1.3888,1.4727,1.83,2.238,3.1386,3.5964,6.146,7.2215,11.6477
1.0006,0.9706,0.954,0.9557,0.9622,1.1193,1.1033,1.655,1.4433,1.4494,1.8254,2.2377,3.16,3.8423,6.3946,7.8323
1.3383,1.308,1.3043,1.3007,1.3071,1.4573,1.4585,1.9362,1.668,1.7831,2.3214,3.0518,4.8585,7.4769,7.0067,8.3786
1.3081,1.2794,1.2838,1.2769,1.2794,1.4371,1.4406,2.1527,1.896,2.0357,2.646,3.3279,4.8446,7.9331,7.6373,9.6531

part of cost in a round

basic parameter of point-to-point in the model

```
[INFO] comm_pair{ src: g15r1n06{8, 0}; dst: g15r1n06{9, 1}; }
[INFO] contention: 2
[INFO] L0_0_2: 112.197
[INFO] cost: 112.197
```

Case Study – Operation broadcast

➢ Testing the cost of operation broadcast:

```
cp op-template.sh bcast.sh
vim bcast.sh
bash bcast.sh
```

```
#!/bin/bash
NTASK=16
PPN=8
JOBNAME=bcast
CMD=(
     "mpirun"
     "-np $NTASK"
     "-ppn $PPN"
     "-bind-to core"
     "$(pwd)/mpi/collective/osu_bcast"
     "-m 4:65536"
     "-x 5"
     "-i 100"
)
mkdir -p log

echo "START $JOBNAME WITH NTASK=$NTASK"
nowdate=$(date +%Y_%m_%d_%H_%M_%S)
echo $nowdate

sbatch << END
#SBATCH --job-name=${JOBNAME}
#SBATCH --ntasks-per-node=${PPN}
#SBATCH --ntasks=${NTASK}
#SBATCH --partition=normal
#SBATCH --output=log/%j.log
export MPIR_CVAR_BCAST_INTRA_ALGORITHM=nb
export MPIR_CVAR_IBCAST_INTRA_ALGORITHM=sched_binomial
${CMD[@]}
END
```

Modify the Job Scale

Modify the JOB NAME

Modify the EXE PATH

Modify the ENV VAR

Case Study – Operation broadcast

```
srun: ROUTE: split_hostlist: hl=g15r1n[05-06] tree_width 0
# OSU MPI Broadcast Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
4                       2.48
8                       2.49
16                      2.47
32                      2.86
64                      3.38
128                     5.38
256                     3.81
512                     7.09
1024                    7.56
2048                    7.87
4096                   10.06
8192                   11.51
16384                  33.27
32768                  53.10
65536                  73.96
```

```
$HLOP --op=BCAST \
--algo=binomial \
--pf=DF \
--nl=g15r1n[05-06] \
--ppn=8 \
--msz="4,8,16,32,64,128,256,
512,1024,2048,4096,8192,16384,
32768,65536"

…
PPN=8
JOBNAME=bcast
…
```

Case Study – Operation scatter

➢ Testing the cost of operation scatter:

```
cp op-template.sh scatter.sh
vim scatter.sh
bash scatter.sh
```

```
#!/bin/bash
NTASK=16
PPN=8
JOBNAME=scatter
CMD=(
     "mpirun"
     "-np $NTASK"
     "-ppn $PPN"
     "-bind-to core"
     "$(pwd)/mpi/collective/osu_scatter"
     "-m 4:65536"
     "-x 5"
     "-i 100"
)
mkdir -p log

echo "START $JOBNAME WITH NTASK=$NTASK"
nowdate=$(date +%Y_%m_%d_%H_%M_%S)
echo $nowdate

sbatch << END
#SBATCH --job-name=${JOBNAME}
#SBATCH --ntasks-per-node=${PPN}
#SBATCH --ntasks=${NTASK}
#SBATCH --partition=normal
#SBATCH --output=log/%j.log
export MPIR_CVAR_SCATTER_INTRA_ALGORITHM=nb
export MPIR_CVAR_ISCATTER_INTRA_ALGORITHM=sched_binomial
${CMD[@]}
END
```

Modify the Job Scale

Modify the JOB NAME

Modify the EXE PATH

Modify the ENV VAR

Case Study – Operation scatter

```
srun: ROUTE: split_hostlist: hl=g15r1n[05-06] tree_width 0
# OSU MPI Scatter Latency Test v7.3
# Datatype: MPI_CHAR.
# Size       Avg Latency(us)
4                       2.48
8                       2.49
16                      2.47
32                      2.86
64                      3.38
128                     5.38
256                     3.81
512                     7.09
1024                    7.56
2048                    7.87
4096                   10.06
8192                   11.51
16384                  33.27
32768                  53.10
65536                  73.96
```

```
$HLOP --op=SCATTER \
--algo=binomial \
--pf=DF \
--nl=g15r1n[05-06] \
--ppn=8 \
--msz="4,8,16,32,64,128,256,
512,1024,2048,4096,8192,16384,
32768,65536"

…
PPN=8
JOBNAME=scatter
…
```