OMPICollTune: Autotuning MPI Collectives by Incremental Online Learning

Sascha Hunold and Sebastian Steiner

Research Group for Parallel Computing, Faculty of Informatics
TU Wien (Vienna University of Technology)
Vienna, Austria

SC22 Dallas, TX | hpc accelerates.

MPI Collectives: A Very Brief Introduction

MPI - Message Passing Interface

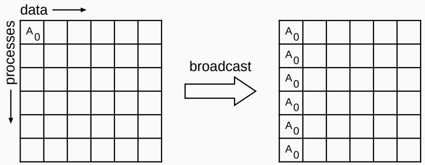
- specification of a communication interface
- all operations defined as functions
- all functions have defined semantics
- Several MPI libraries: Open MPI, MVAPICH, Intel MPI, Cray MPI, . . .

Collective Communication Operations

- Communication that involves a group of processes
- Collection of pre-defined optimized routines (common tasks)

Examples

- `MPI_Allreduce`
- `MPI_Alltoall`
- `MPI_Bcast`

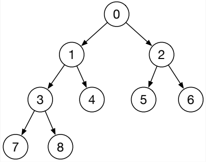


source: MPI: A Message-Passing Interface Standard Version 3.1

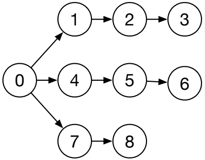
MPI Collectives under the Hood

- MPI collectives
  - defined semantics
  - possibly different implementations
- MPI libraries provide significant number of different algorithms for individual collectives such as MPI_Bcast or MPI_Allreduce
- parameterized MPI algorithms
  - example chain algorithm: 2 parameters in Open MPI
    - fan-out, how many chains?
    - segment size (for pipelining)

Algorithms for MPI_Bcast



binary tree algorithm



chain algorithm

The Problem

Problem Statement: Algorithm Selection and Configuration

Input

An instance of a collective communication problem P:

- a collective (e.g., `MPI_Bcast`),
- a message size (e.g., 1 MiB),
- a number of compute nodes, and
- a number of processor per compute node.

Output

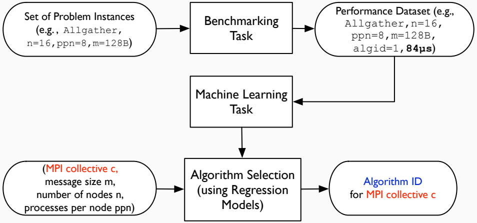
Return the fastest algorithm that solves the problem P.

Two problems to solve (selection and configuration):

1. Determine the best algorithm from the set of possible algorithms.
2. Determine the best parameters to configure this algorithm.

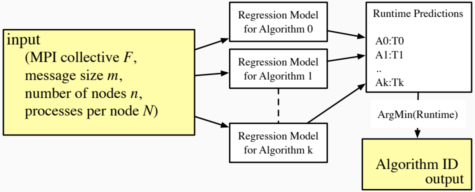
Status Quo: Machine Learning Based Collective Tuning

General Strategy



S. Hunold and A. Carpen-Amarie. "Algorithm Selection of MPI Collectives Using Machine Learning Techniques". In: *PMBS@SC*. 2018

Classification by Regression



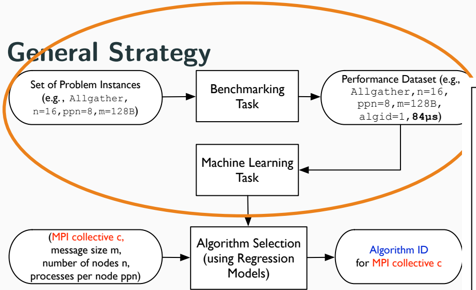
S. Hunold, A. Bhatele, G. Bosilca, and P. Knees. "Predicting MPI Collective Communication Performance Using Machine Learning". In: *IEEE CLUSTER*. 2020, pp. 259–269

Important Related Work

- M. Wilkins, Y. Guo, R. Thakur, P. Dinda, and N. Hardavellas. "ACCLAiM: Advancing the Practicality of MPI Collective Communication Autotuning Using Machine Learning". In: *IEEE CLUSTER*. 2022
- J. Pjesivac-Grbovic, G. Bosilca, G. E. Fagg, T. Angskun, and J. Dongarra. "MPI collective algorithm selection and quadtree encoding". In: *Parallel Computing* 33.9 (2007), pp. 613–623
- A. Faraj, X. Yuan, and D. K. Lowenthal. "STAR-MPI: self tuned adaptive routines for MPI collective operations". In: *ICS*. ACM, 2006, pp. 199–208

Status Quo: Machine Learning Based Collective Tuning

General Strategy



S. Hunold and A. Carpen-Amarie. "Algorithm Selection of Collectives Using Machine Learning Techniques". In: *PME* 2018

Classification by Regression

ng MPI
rning".

The Problem

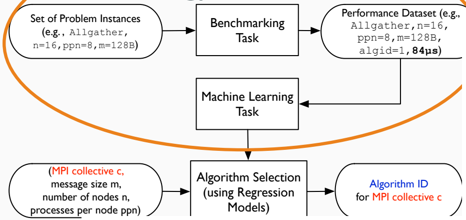
- **Which cases** to benchmark?
- case: (collective, number of compute nodes, processes per node, message size)
- some **cases** very slow and perhaps **irrelevant**, `MPI_Alltoall` with large messages

Important Related Work

- M. Wilkins, Y. Guo, R. Thakur, P. Dinda, and N Communication Autotuning Using Machine Lear
- J. Pjesivac-Grbovic, G. Bosilca, G. E. Fagg, T. Angskun, and J. Dongarra. "MPI collective algorithm selection and quadtree encoding". In: *Parallel Computing* 33.9 (2007), pp. 613–623
- A. Faraj, X. Yuan, and D. K. Lowenthal. "STAR-MPI: self tuned adaptive routines for MPI collective operations". In: *ICS*. ACM, 2006, pp. 199–208

Status Quo: Machine Learning Based Collective Tuning

General Strategy



Classification by Regression

S. Hunold
Collective
2018

ng MPI
rning".

Imp

The Problem

- Which cases to benchmark?
- case: (collective, number of compute nodes, processes per node,

ery slow and perhaps
PI_Alltoall with
s

The Idea

Let's build a model with the runtimes of collectives that we measure while executing HPC applications.

- collective algorithm selection and quadtree encoding". In: *Parallel Computing* 33.9 (2007), pp. 613–623
- A. Faraj, X. Yuan, and D. K. Lowenthal. "STAR-MPI: self tuned adaptive routines for MPI collective operations". In: *ICS*. ACM, 2006, pp. 199–208

Goals of this Work

Our Goals

Question

How to build a model to predict the best algorithm for specific collective communication problem

1. with low overhead and
2. with a high accuracy?

Our Goals

Question

How to build a model to predict the best algorithm for specific collective communication problem

1. with low overhead and
2. with a high accuracy?

Hypothesis

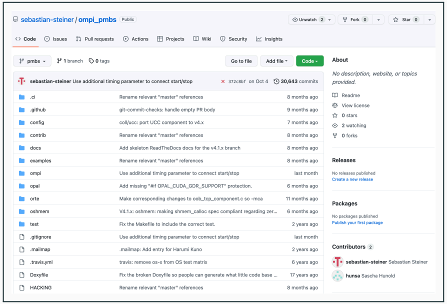
An efficient prediction model for collectives can be built from running HPC applications on a production system by

1. algorithm sampling
   - give every algorithm a chance, but perhaps not the same
2. process sampling
   - not all processes will have to participate (save storage)

Contribution: OMPICollTune

OMPICollTune: Online Tuning of MPI Collectives

- Extension of Open MPI `4.1.x` (fork)
- Intercepting MPI collectives (tracing) during application runtime
  - collect performance measurements
- Algorithm selection by probability distribution
  - probabilities updated with new performance data (after each `srun`, once per hour, once per day, ...)
  - slow algorithms get a smaller probability to be selected
- Very low overhead
  - sampling of performance stats can be bounded
    - e.g., record only the first 100 calls to `MPI_Allreduce`
    - e.g., record only on 16 of 10 000 processes

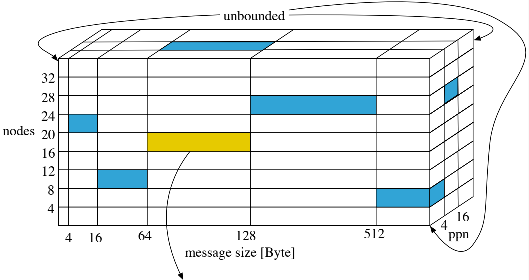


`https://github.com/sebastian-steiner/ompi_pmbs`

Approach

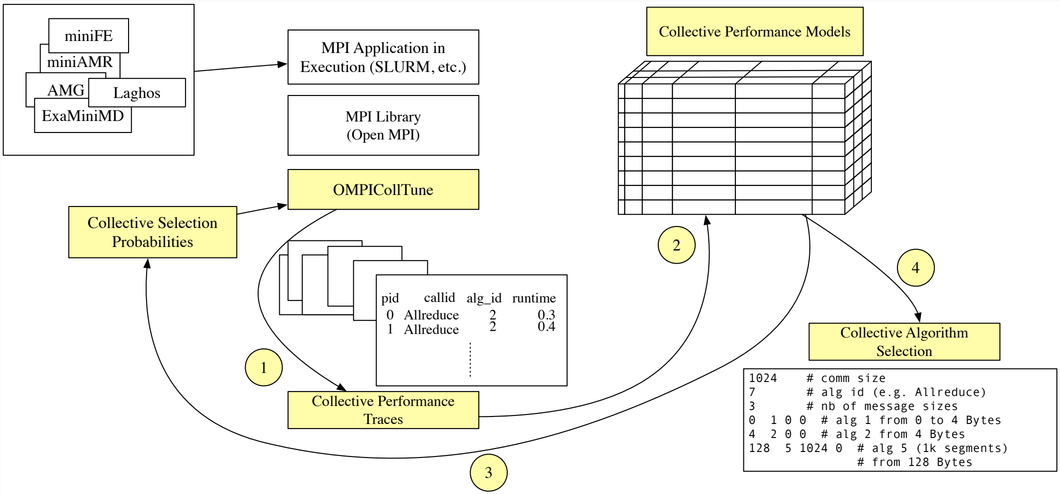
Performance Model

- 3D model: msize, nodes, ppn
- predefined dimensional cuts
  - e.g., message size in Bytes 1-10, 11-100, 101-1000
- each block holds a probability distribution for each collective (see illustration)
- prediction model across the 3D blocks
- whenever we get new data for a specific 3D block, the dataset and the prediction model are updated



AlgID	OMPI ID	SegSize	Prob
1	1	0	0.1
2	2	0	0.3
3	3	0	0.05
4	4	0	0.15
5	5	1KB	0.1
6	5	1MB	0.1
7	5	8MB	0.1
8	6	0	0.1

Online Tuning Approach



OpenMPI: MPI_Allreduce

- querying prediction model and conversion to Open MPI algorithm
- recording time stamps
- ompi/mca/coll/tuned/coll_tuned_allreduce_decision.c

```
if( AT_is_collective_sampling_enabled(MPI_ALLREDUCE) ) {
  // randomly select algorithm (incl. alg configuration)
  our_alg_id = AT_get_allreduce_selection_id(bufsize, commsize, operator)

  // translate algorithm and its configuration into OpenMPI
  AT_col_t our_alg = AT_get_allreduce_our_alg(our_alg_id);
  algorithm = our_alg.ompi_alg_id;
  segsize = our_alg.seg_size;

  AT_record_start_timestamp(MPI_ALLREDUCE, our_alg_id,
                            count * type_size, comm_size);
}
```

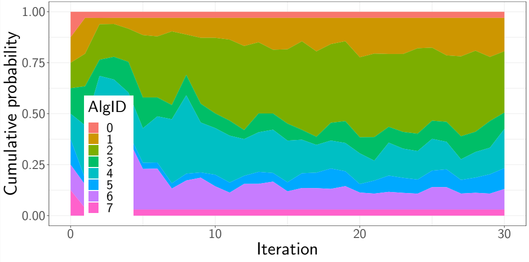
```
switch (algorithm) {
// ..
case (2):
 res = ompi_coll_base_allreduce_intra_nonoverlapping(..);
 break;
case (3):
 res = ompi_coll_base_allreduce_intra_recursivedoubling(..);
 break;
case (4):
 res = ompi_coll_base_allreduce_intra_ring(..);
 break;
// ..
}
if(AT_is_collective_sampling_enabled(MPI_ALLREDUCE))
        AT_record_end_timestamp(MPI_ALLREDUCE);
```

Experimental Results

Iterative Improvement: miniAMR

- run miniAMR with 32×32 processes
  - same problem instance
- trace `MPI_Allreduce` and update probability distribution based on recorded performance
- Algorithms 0 and 7 fade out
  - `basic_linear` and `rabenseifner`
- highest selection probability: Algorithm 2

Progress of selection probabilities of each algorithm

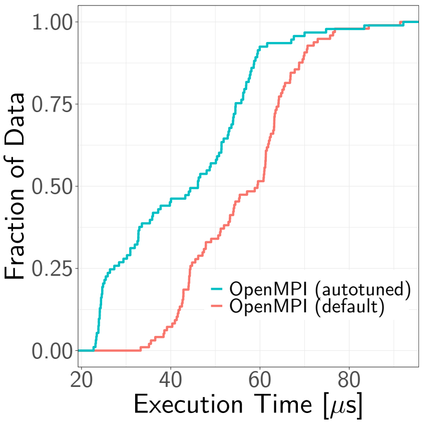


`MPI_Allreduce`; 32×32 processes

Compare Tuned Algorithms in Benchmark

Comparing ECDF of `MPI_Allreduce`

- query performance model with unseen instance: 24×32 processes
- OMPICollTune Alg ID 2 maps to Open MPI algorithm 3
  - `recursive_doubling` (no segmentation)
- default decision logic in Open MPI 4.1.x
  - Open MPI algorithm 2
  - `nonoverlapping` (internally Reduce+Bcast)
- ReproMPI benchmark to compare runtimes of 100 calls to `MPI_Allreduce`



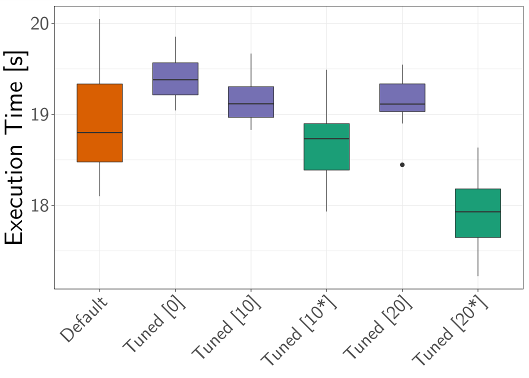
48-Byte messages

S. Hunold and A. Carpen-Amarie. "Reproducible MPI Benchmarking is Still Not as Easy as You Think". In: *IEEE Trans. Parallel Distrib. Syst.* **27.12 (2016), pp. 3617–3630.** DOI: 10.1109/TPDS.2016.2539167

Incremental Online Learning

Overhead Analysis

- per iteration: 20 different runs of miniAMR
- Default: using default Open MPI selection logic
- `Tuned[x]`: runtime after x updates of the model
    - uses probability distribution to select algorithm
- `Tuned[x*]`: runtime after x updates of the model
    - select only the algorithm that has highest probability



Thank you

References

[1] A. Faraj, X. Yuan, and D. K. Lowenthal. "STAR-MPI: self tuned adaptive routines for MPI collective operations". In: *ICS*. ACM, 2006, pp. 199–208.

[2] S. Hunold, A. Bhatele, G. Bosilca, and P. Knees. "Predicting MPI Collective Communication Performance Using Machine Learning". In: *IEEE CLUSTER*. 2020, pp. 259–269.

[3] S. Hunold and A. Carpen-Amarie. "Algorithm Selection of MPI Collectives Using Machine Learning Techniques". In: *PMBS@SC*. 2018.

[4] S. Hunold and A. Carpen-Amarie. "Reproducible MPI Benchmarking is Still Not as Easy as You Think". In: *IEEE Trans. Parallel Distrib. Syst.* 27.12 (2016), pp. 3617–3630. DOI: 10.1109/TPDS.2016.2539167.

References (2)

[5] J. Pjesivac-Grbovic, G. Bosilca, G. E. Fagg, T. Angskun, and J. Dongarra. "MPI collective algorithm selection and quadtree encoding". In: *Parallel Computing* 33.9 (2007), pp. 613–623.

[6] M. Wilkins, Y. Guo, R. Thakur, P. Dinda, and N. Hardavellas. "ACCLAiM: Advancing the Practicality of MPI Collective Communication Autotuning Using Machine Learning". In: *IEEE CLUSTER*. 2022.