MPI Advisor

a Minimal Overhead Tool for MPI Library Performance Tuning

Esthela Gallardo and Pat Teller
Dept. of Computer Science, UT-El Paso
{egallardo5@miners.utep.edu, pteller@utep.edu}
and
Jerome Vienne, Leonardo Fialho and Jim Browne
Texas Advanced Computing Center (TACC), UT-Austin
{viennej@tacc.utexas.edu, fialho@tacc.utexas.edu, browne@cs.utexas.edu}

Overview

- Motivation
- Approach
- Tuning Strategies
- Future Research

Overview

- Motivation
- Approach
- Tuning Strategies
- Future Research

Motivation/1

- MPI is pervasive.
- Features to optimize performance are library dependent.
- Most users employ default library- and cluster-specific parameters.
- Many jobs may have MPI-related performance issues.
- Needed: an easy-to-use tool for non-experts to optimize MPI communication performance.

Motivation/2

Available MPI Performance Tools

	MPI Advisor (TACC/UTEP)	OPTO (PSTL/UH)	Atune (UIBK)	MPITune (Intel)	Periscope (TUM)
Single run	X				
Multiple libraries	X				
Basic privileges	X	X	X	X	
Inter-node optimization	X	X	X	X	X
Intra-node optimization				X	X
Message passing optimization	X	X	X	X	X
Does not require expert knowledge	X				

Motivation/3

TACC Performance Tools

PerfExpert	Optimization of (mostly) memory accesses at the compute-node level
MACPO	Addition of data structure measurements and metrics to PerfExpert diagnoses and recommendations
MACVEC	Application optimization via enhancement of vectorization

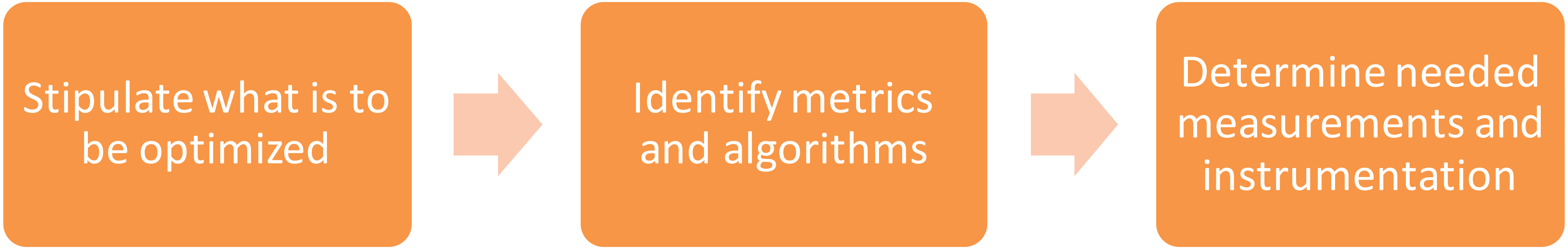
All of these are intra-compute node optimizations.

What about MPI communication optimization? ⇨ **MPI Advisor**

Overview

- Motivation
- Approach
- Tuning Strategies
- Future Research

Approach - Conceptual

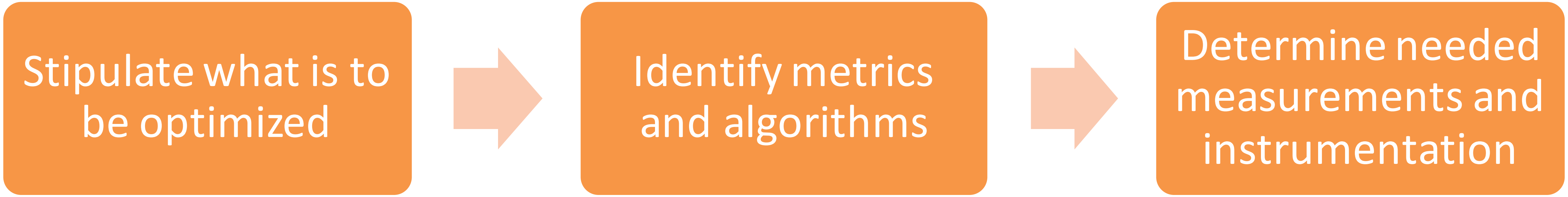


Stipulate what is to be optimized

Identify metrics and algorithms

Determine needed measurements and instrumentation

Approach - Conceptual

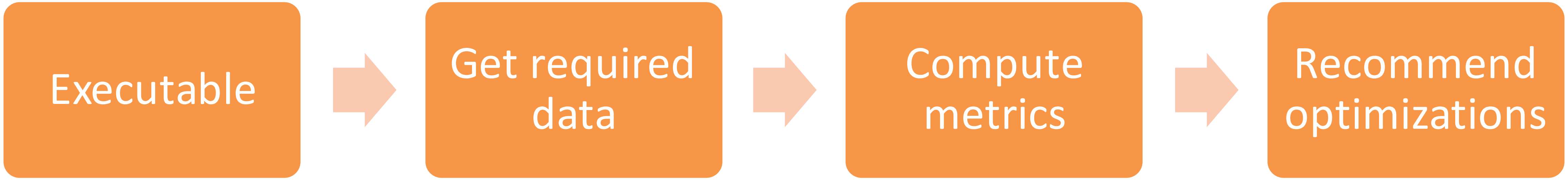


Stipulate what is to be optimized

Identify metrics and algorithms

Determine needed measurements and instrumentation

Approach - Operational



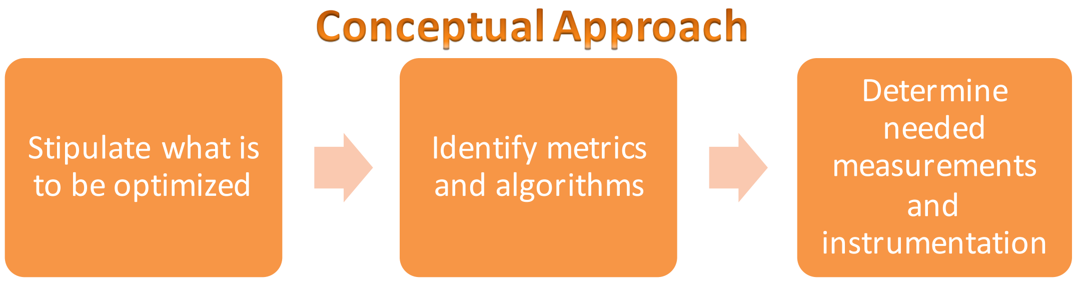
Executable

Get required data

Compute metrics

Recommend optimizations

MPI Advisor Functionality

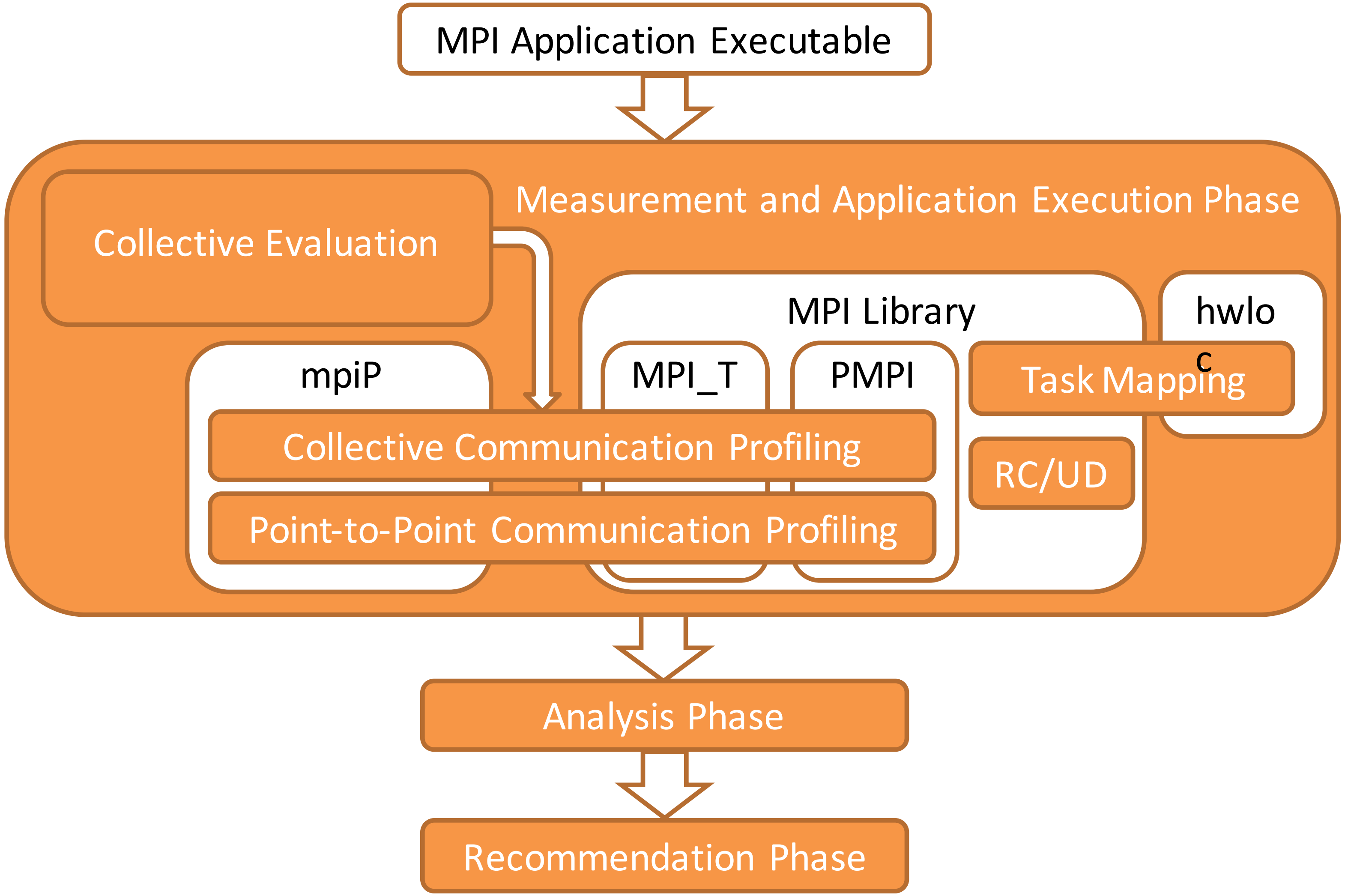

Conceptual Approach

Stipulate what is to be optimized
Identify metrics and algorithms
Determine needed measurements and instrumentation

Currently MPI Advisor:

- Executes all steps in the
- Implements
  1. measurements w/o user instrumentation,
  2. computation of metrics, and
  3. algorithms for diagnosing bottlenecks and recommending optimizations.
- Does all of this with
  – a *single run of the application* and
  – only a *few percent overhead*.

Workflow Diagram

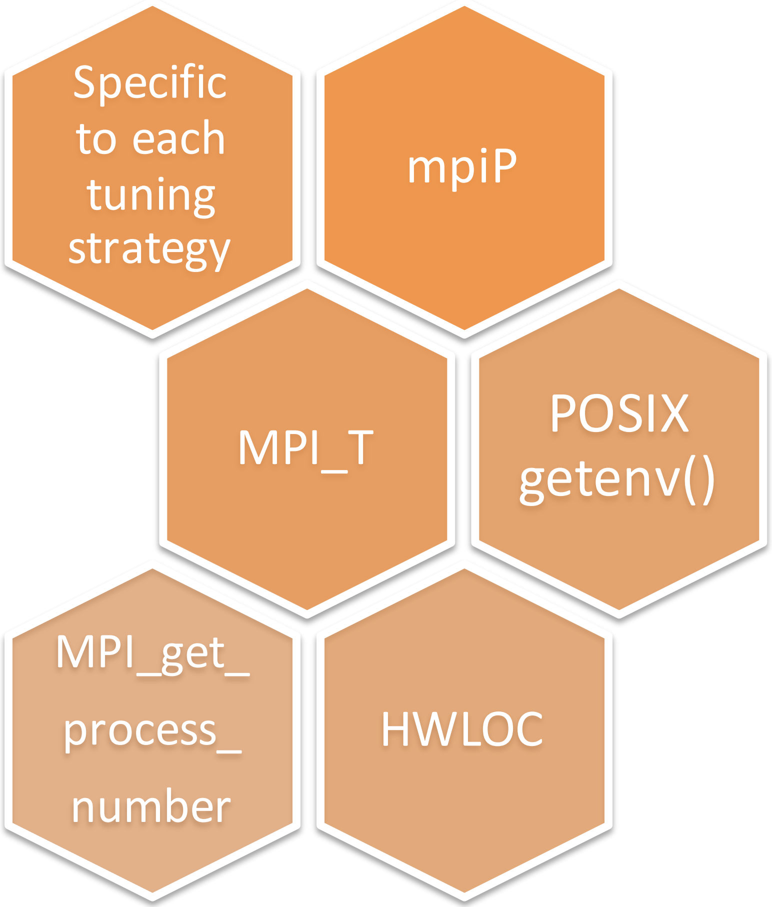

MPI Application Executable
Measurement and Application Execution Phase
Collective Evaluation
MPI Library
hwloc
mpiP
MPI_T
PMPI
Task Mapping
Collective Communication Profiling
RC/UD
Point-to-Point Communication Profiling
Analysis Phase
Recommendation Phase

Measurements

Execution-Environment Parameters

- Collected once – on installation.
- Collective Evaluation (CE) Script runs IMB and OMB benchmarks with each library and collects, among other data, the performance of each collective algorithm for different message sizes.

Application Measurements



Specific to each tuning strategy

mpiP

MPI_T

POSIX getenv()

MPI_get_ process_ number

HWLOC

Overview

- Motivation
- Approach
- **Tuning Strategies**
- Future Research

Currently Supported Tuning Strategies

Point-to-Point Protocol Threshold

- Eager vs. Rendezvous

Choice of Algorithms for Collective Operations

- Depends on system size, message size, and task properties

Mapping of MPI Tasks to Cores

- Map Task 0 to socket that shares the PCI Express bus with the HCA card
- Default mappings vs. custom mappings

Infiniband Transport: RC and UD

- Tradeoff between memory footprint and message size

Experimental Platform

- Stampede and Maverick clusters at TACC:
  - – Stampede: 6,400 dual-socket, 8-core Sandy-Bridge E5-2680 compute nodes, each with 32 GB of memory.
  - – Maverick: 132 dual-socket, 10-core Ivy-Bridge E5-2680v2 compute nodes, each with 256 GB of memory.
- MPI Libraries:
  - – Intel MPI 4.1.0.030
  - – MVAPICH2 1.9a2
  - – [OpenMPI 1.8.2]

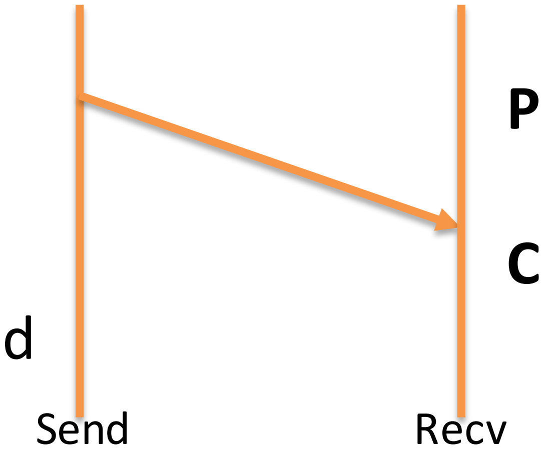
Point-to-Point Protocol Threshold: Description

Different "Send" Protocols

Eager

Pro: Decrease in sync delays and, thus, latency

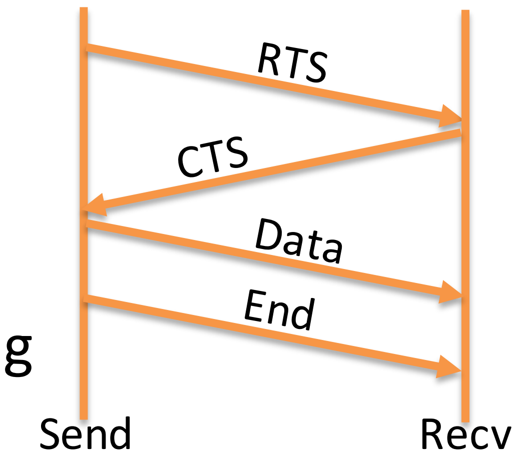
Con: Memory exhaustion if much buffering is required



Rendezvous

Pro: Prevents memory exhaustion

Con: Introduces delay due to handshaking



- MPI Advisor focuses only on increasing the switch point – *the eager threshold* – to reduce the use of the rendezvous protocol.
- Most jobs on Stampede have 5-10 GB of memory available during runtime, thus, increasing the threshold will not cause memory issues.

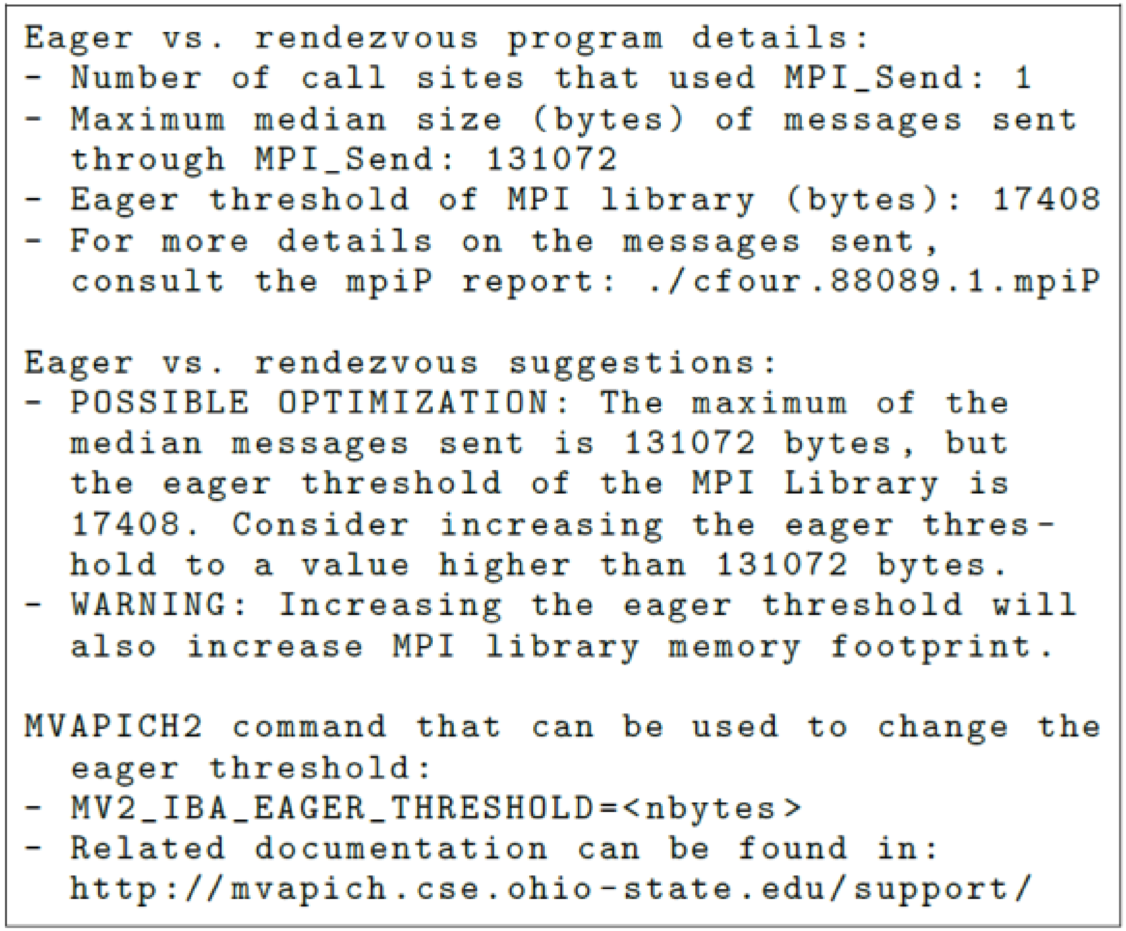
Point-to-Point Protocol Threshold: Implementation

1. Use MPI_T to identify the value of the eager threshold.
2. Use mpiP performance data to determine the number and size of messages transmitted via send/receive operations.
3. Compute the median message size per call site, determine the maximum of these, and compare the maximum to eager threshold.
    - If the computed value is larger than the default, then MPI Advisor outputs its recommendations, instructions, and warnings.

Point-to-Point Protocol Threshold: Demonstration/1

- Application: CFOUR-based benchmark
  - Reads and writes fixed records in random order.
  - Messages are mainly point-to-point with sizes around 128 KB or less.
- Recommendation: Since the default eager threshold is 17 KB, MPI Advisor recommends increasing the eager threshold to more than 131,072 bytes.

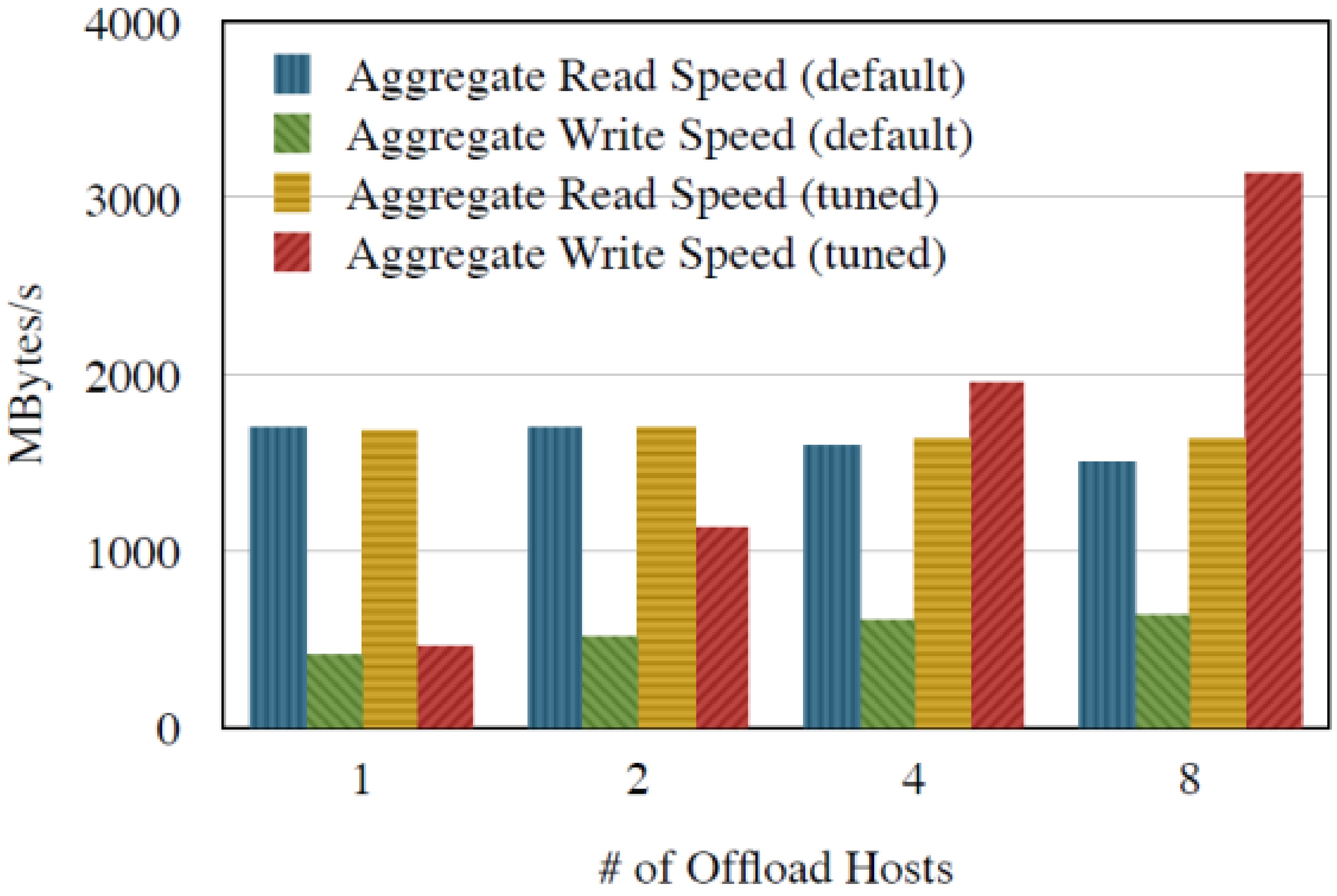
MPI Advisor Output



```
Eager vs. rendezvous program details:
- Number of call sites that used MPI_Send: 1
- Maximum median size (bytes) of messages sent
  through MPI_Send: 131072
- Eager threshold of MPI library (bytes): 17408
- For more details on the messages sent,
  consult the mpiP report: ./cfour.88089.1.mpiP

Eager vs. rendezvous suggestions:
- POSSIBLE OPTIMIZATION: The maximum of the
  median messages sent is 131072 bytes, but
  the eager threshold of the MPI Library is
  17408. Consider increasing the eager thres-
  hold to a value higher than 131072 bytes.
- WARNING: Increasing the eager threshold will
  also increase MPI library memory footprint.

MVAPICH2 command that can be used to change the
  eager threshold:
- MV2_IBA_EAGER_THRESHOLD=<nbytes>
- Related documentation can be found in:
  http://mvapich.cse.ohio-state.edu/support/
```

Point-to-Point Protocol Threshold: Demonstration/2

- Improvement: Running the CFOUR-based benchmark with an increased eager threshold of 256KB resulted in a ~5x improvement for write operations.



Choice of Algorithms for Collective Operations Description

- For each collective operation there are several algorithms provided by each MPI library that implement the operation.
- Any algorithm's performance depends on several parameters (e.g., system size, message size, architecture, etc.).
- Expert recommendations regarding the algorithm to use for collective operations could result in better application performance as compared to the MPI library's auto-selection strategy.

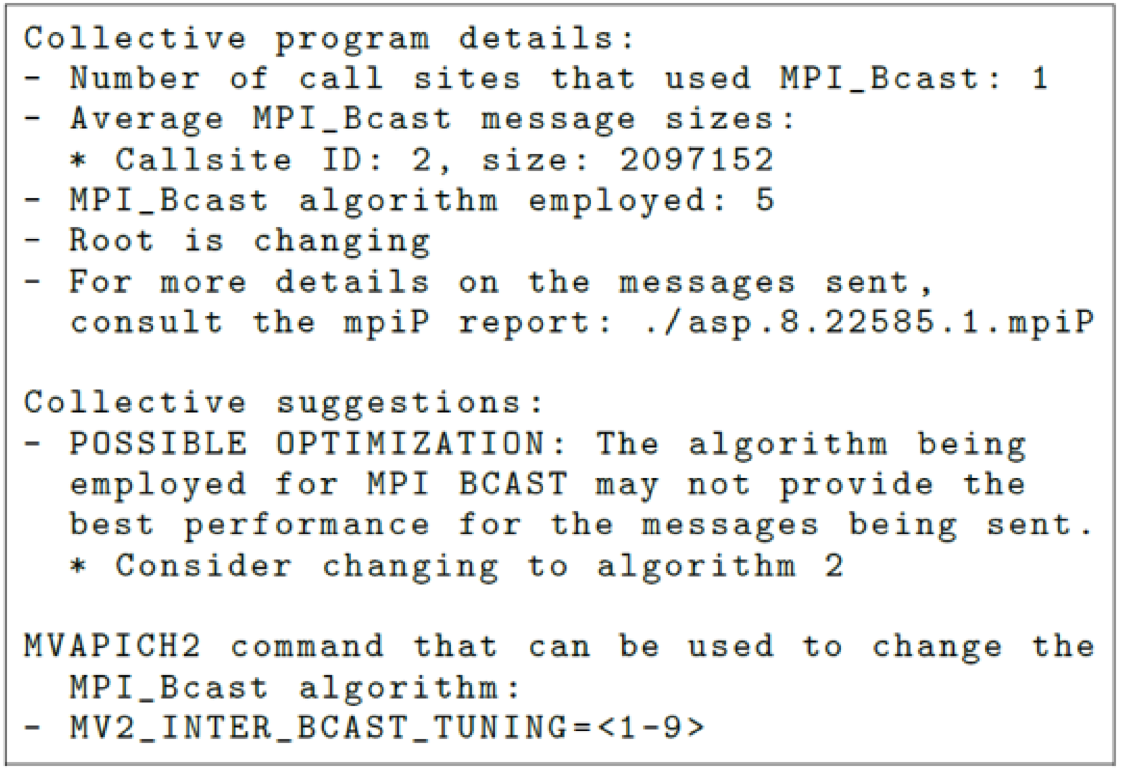
Choice of Algorithms for Collective Operations Implementation

1. Use MPI_T to identify the collective algorithm that is employed.
2. Use mpiP to record the execution time and message size of each collective operation employed by the application.
3. Determine the algorithm to use for each message size of interest by referencing a table (generated by the CE script) that includes the execution times of every collective operation algorithm for a large set of message sizes.
4. If there are collective operations for which the application should change algorithms, MPI Advisor outputs related recommendations.

Choice of Algorithms for Collective Operations Demonstration/1

- Application: ASP
  - Parallel version of the Floyd-Warshall algorithm used to solve the all-pairs shortest-path problem.
  - Mainly uses MPI_Bcast.
  - Changes the root for each iteration.
- Recommendations: MPI Advisor recommends changing the algorithm used for MPI_Bcast.

MPI Advisor Output



```
Collective program details:
- Number of call sites that used MPI_Bcast: 1
- Average MPI_Bcast message sizes:
  * Callsite ID: 2, size: 2097152
- MPI_Bcast algorithm employed: 5
- Root is changing
- For more details on the messages sent,
  consult the mpiP report: ./asp.8.22585.1.mpiP

Collective suggestions:
- POSSIBLE OPTIMIZATION: The algorithm being
  employed for MPI BCAST may not provide the
  best performance for the messages being sent.
  * Consider changing to algorithm 2

MVAPICH2 command that can be used to change the
  MPI_Bcast algorithm:
- MV2_INTER_BCAST_TUNING=<1-9>
```

Choice of Algorithms for Collective Operations Demonstration/2

- Improvement: By changing the MPI_Bcast algorithm, the performance of ASP improved by 8.3%.

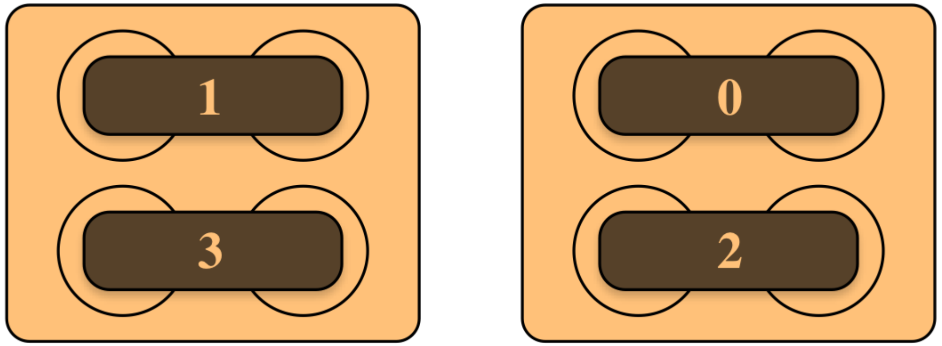
MVAPICH2 Default	MVAPICH2 Tuned	Intel MPI Default
24.45 sec	22.41 sec	22.38 sec

Mapping of MPI Tasks to Cores: Description

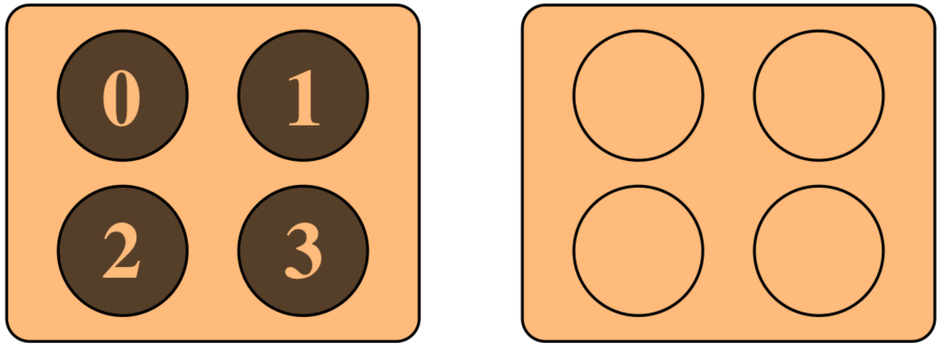
- Each MPI library provides its own default strategy for mapping tasks to sockets and cores.
- There is no single best mapping for all applications – the mapping is application dependent.
- The default mapping by MVAPICH2 and Open MPI does not deliver best performance for hybrid applications.
- Identifying and effecting a suitable mapping requires knowledge regarding the node architecture and the related parameter settings.

Mapping of MPI Tasks to Cores: Default Mappings

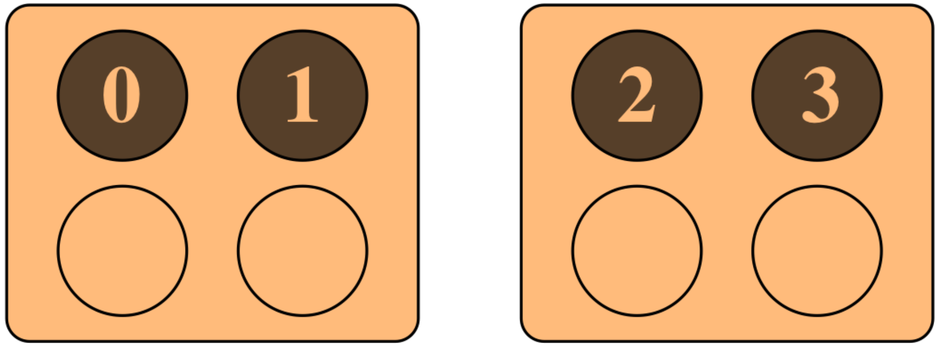
Default mappings of 4 multi-threaded MPI Tasks inside a dual-socket node equipped with two 4-core processors.



Intel MPI 4.1



MVAPICH2 1.9a2



Open MPI 1.8.2

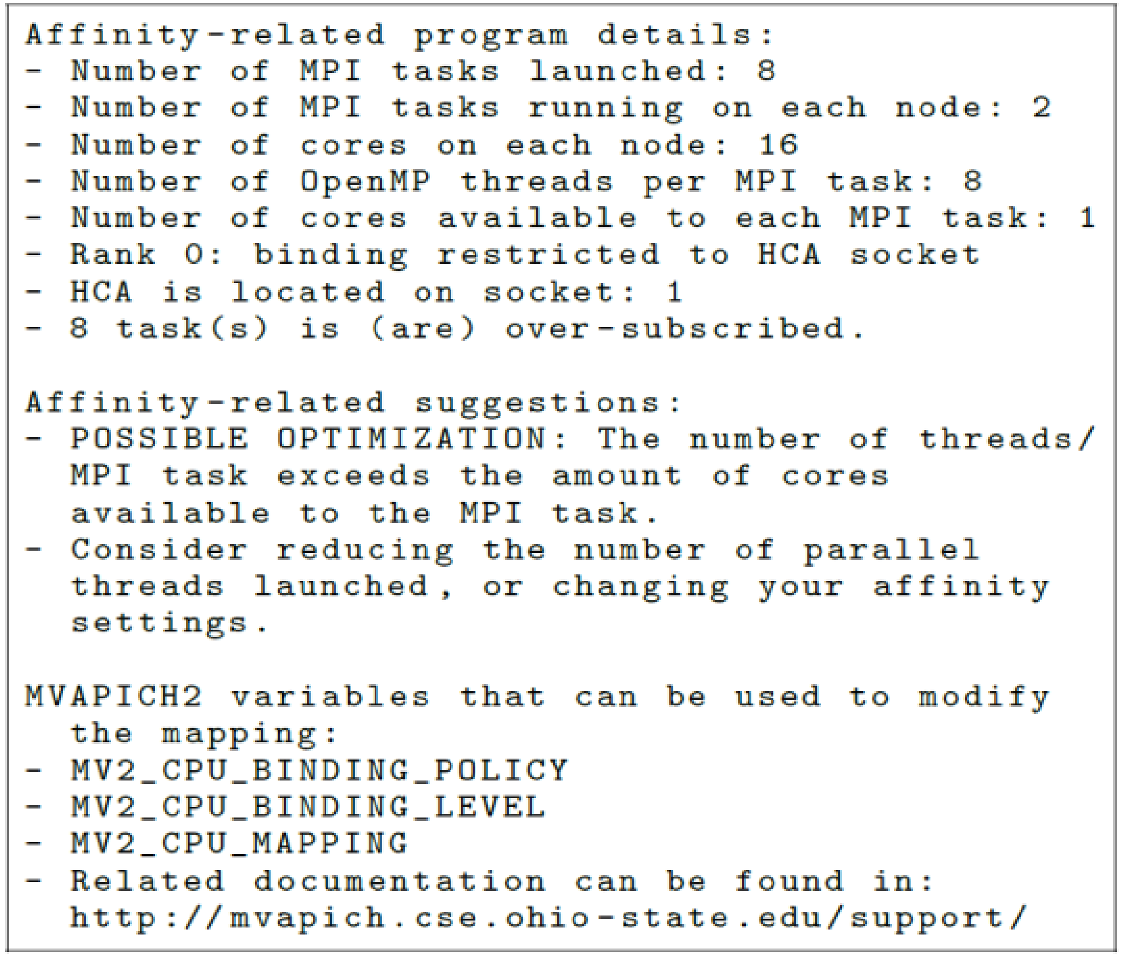
Mapping of MPI Tasks to Cores: Implementation

1. Use HWLOC to expose node architecture and current mapping.
2. Check that there is no node, core, and/or task over- or under-subscription.
3. Check that the Rank 0 process is on the socket that shares the PCI Express bus with the HCA card.
4. If any of the previous conditions exist, MPI Advisor prints warnings, details about problems with current mapping, and suggests changing it.

Mapping of MPI Tasks to Cores: Demonstration/1

- Application: HPCG
  - Alternative ranking of the TOP500 list.
  - Launched with default configuration: 2 MPI tasks with 8 OMP threads/tasks.
- Recommendation: MPI Advisor recommends modifying the mapping; in this case, it is better to place each task on a different socket.

MPI Advisor Output



```
Affinity-related program details:
- Number of MPI tasks launched: 8
- Number of MPI tasks running on each node: 2
- Number of cores on each node: 16
- Number of OpenMP threads per MPI task: 8
- Number of cores available to each MPI task: 1
- Rank 0: binding restricted to HCA socket
- HCA is located on socket: 1
- 8 task(s) is (are) over-subscribed.

Affinity-related suggestions:
- POSSIBLE OPTIMIZATION: The number of threads/
  MPI task exceeds the amount of cores
  available to the MPI task.
- Consider reducing the number of parallel
  threads launched, or changing your affinity
  settings.

MVAPICH2 variables that can be used to modify
  the mapping:
- MV2_CPU_BINDING_POLICY
- MV2_CPU_BINDING_LEVEL
- MV2_CPU_MAPPING
- Related documentation can be found in:
  http://mvapich.cse.ohio-state.edu/support/
```

Mapping of MPI Tasks to Cores: Demonstration/2

- Improvement: Modifying the tasks-to-cores mapping of the hybrid version of HPCG so that each task is placed on a different socket increased its performance from 26.05 GFLOPS/sec to 38.85 GFLOPS/sec.

Default Mapping	Recommended Mapping
26.05 GFLOPS/sec	38.85 GFLOPS/sec

Infiniband Transport: Description

- Increasing the number of MPI tasks increases the amount of memory required by each task.
- Infiniband provides mechanisms to reduce an application’s memory footprint, but they are not enabled by default.
- Reliable connection (RC) and unreliable datagram (UD) are the the most commonly used Infiniband transports.
- RC connections are initiated during startup, while UD only establishes connections as required.

Infiniband Transport: RC vs UD

Characteristic	RC	UD
Scalability	n^2	n
Corrupt data detected	Yes	Yes
Delivery guarantee	Yes	No
Ordering guarantee	Yes	No
Data loss detection	Yes	No
Error recovery	Yes	No
Send/RDMA write	Yes	No
Receive/RDMA read	Yes	No
Max message size	1 GB	MTU

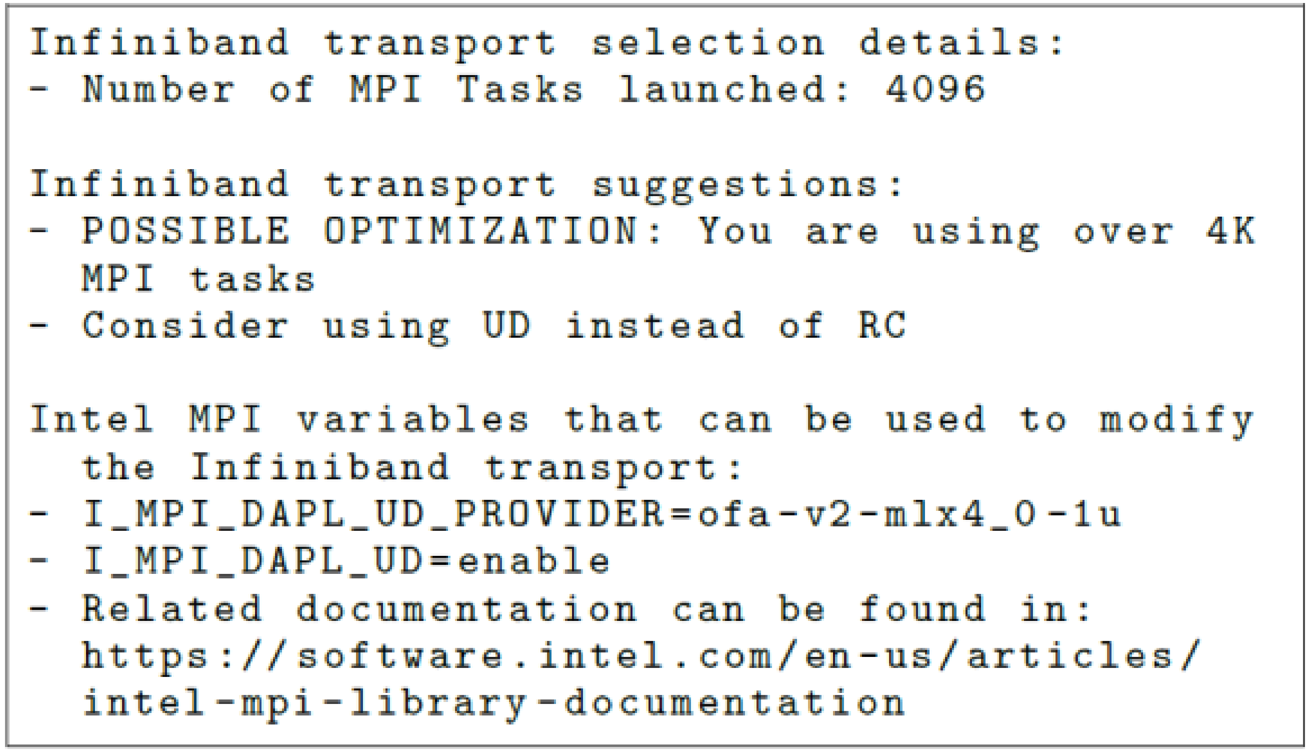
Infiniband Transport: Implementation

1. Assume that the input application is using RC by default.
2. Call MPI_Comm_size() to identify the number of tasks employed by the application.
3. If the number of tasks employed exceeds 4K, MPI Advisor recommends that UD be employed and provides instructions on how to make the change.

Infiniband Transport : Demonstration/1

- Application: SMG2000
  - Parallel semi-coarsening multi-grid solver.
  - Can be run with different node counts.
- Recommendation: MPI Advisor recommends using UD when over 4K tasks are used.

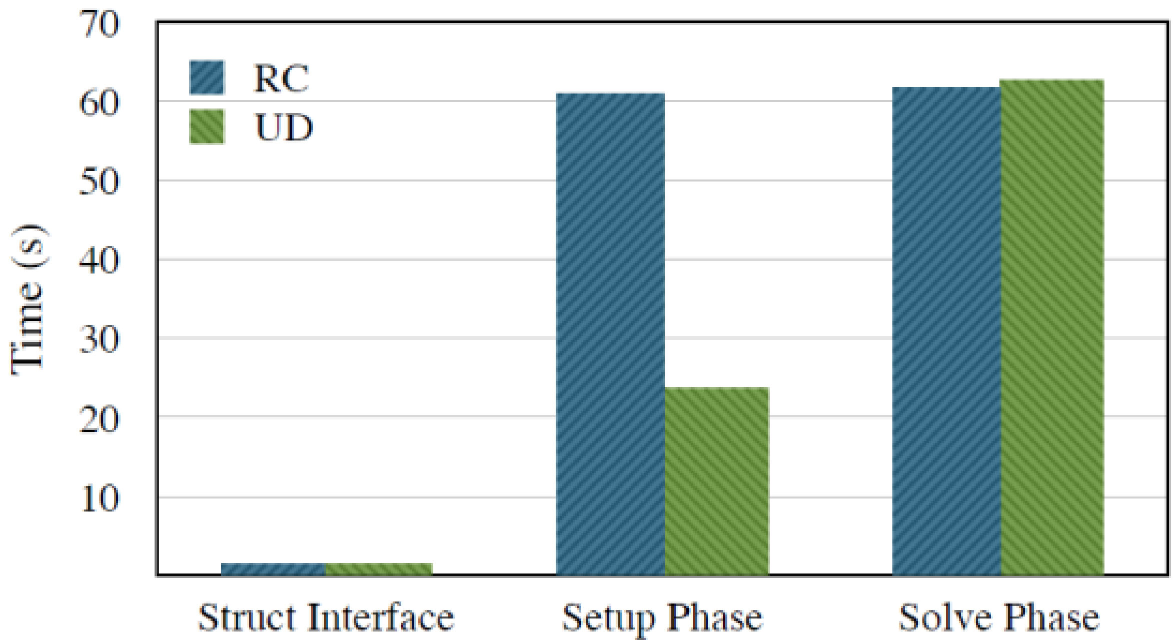
MPI Advisor Output

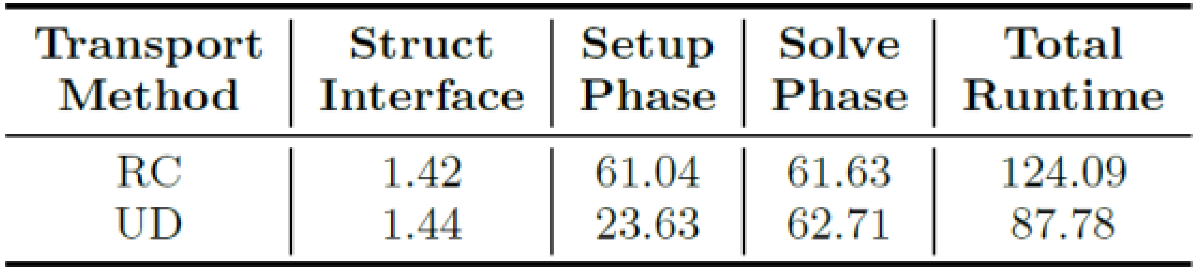


```
Infiniband transport selection details:
- Number of MPI Tasks launched: 4096

Infiniband transport suggestions:
- POSSIBLE OPTIMIZATION: You are using over 4K
  MPI tasks
- Consider using UD instead of RC

Intel MPI variables that can be used to modify
  the Infiniband transport:
- I_MPI_DAPL_UD_PROVIDER=ofa-v2-mlx4_0-1u
- I_MPI_DAPL_UD=enable
- Related documentation can be found in:
  https://software.intel.com/en-us/articles/
  intel-mpi-library-documentation
```

Infiniband Transport : Demonstration/2

- Improvement:
  - SMG2000’s global performance was improved by 29%.
  - SMG2000’s setup phase was improved by 61%.





Transport Method	Struct Interface	Setup Phase	Solve Phase	Total Runtime
RC	1.42	61.04	61.63	124.09
UD	1.44	23.63	62.71	87.78

Summary

- MPI Advisor demonstrates automation of library- and parameter-level tuning of MPI codes.
- The use of MPI Advisor via selection of:
  - Point-to-Point Threshold: Increased write speeds of a CFOUR-based benchmark by ~5x.
  - Collective Operation Algorithm: Improved ASP performance by 8.3%.
  - Tasks-to-Cores Mapping: Increased HPCG GFLOPS/sec from 26.05 to 38.85.
  - Infiniband Transport (RC vs. UD): Improved SMG2000 overall performance by 29%.
- MPI Advisor only requires a *single run* of each executable.

Overview

- Motivation
- Approach
- Tuning Strategies
- Future Research

Future Research

- MPI Advisor is an ongoing project; plans include:
  - Additional library- and parameter-selection strategies
    - *Contact us with suggestions!*
    - *Be a beta user!*
  - Introduction of important source-code level optimizations
  - Expansion to other MPI implementations
    - *Developers: Work with us to support other MPI implementations by exposing more information via MPI_T!*
- Long-Term Plan
  - Unified approach to optimization of multilevel parallelism.

Acknowledgements

- This work was partially funded by the National Science Foundation Stampede grant (through the Texas Advanced Computing Center, award #UTA13-000072).
- The authors are especially grateful to:
  - Todd Evans for providing Stampede monitoring data,
  - Brice Goglin for his guidance on using HWLOC, and
  - The NBC laboratory at OSU for increasing the number of variables exposed through MPI_T for MVAPICH2.

Merci!