Overview

- Problem
- Approach
  - Automation of Performance Optimization
  - Application to MPI
- Example Tuning Strategies
- Case Studies
- Related Research
- Future Research

Problem and Motivation

- MPI is pervasive
- Most users use library and cluster specific default parameters and settings
- Defaults by definition are not always optimal
- Workload studies show many jobs/applications may have MPI related performance issues.
- Most existing tools require in depth knowledge of MPI internals for productive use and/or major investment in effort.
- Easy to use, low overhead tool is needed.

Approach - Conceptual

1. Specify what is to be optimized
2. Specify the metrics needed to diagnosis the bottleneck and recommend the optimization
3. Define the algorithms for diagnosing bottlenecks and recommending optimizations
4. Determine the measurements needed to evaluate those metrics
5. Specify the instrumentation to obtain the measurements.

Approach - Operational

1. Get required data from measurements
2. Compute metrics from measurements
3. Diagnose bottlenecks and recommend optimizations using analyses based on metrics and execution environment parameters
4. Implement recommendations
5. Evaluate result of optimization recommendations
6. Go to 1

Previous Applications of Approach

- PerfExpert – Optimization (mostly) memory access at compute node level
- MACPO – Adding data structure measurements and metrics to PerfExpert diagnoses and recommendations.
- MACVEC – Application to optimization by enhancing vectorization.

All of these are intra-compute node optimizations.

What about internode optimization – MPI Advisor

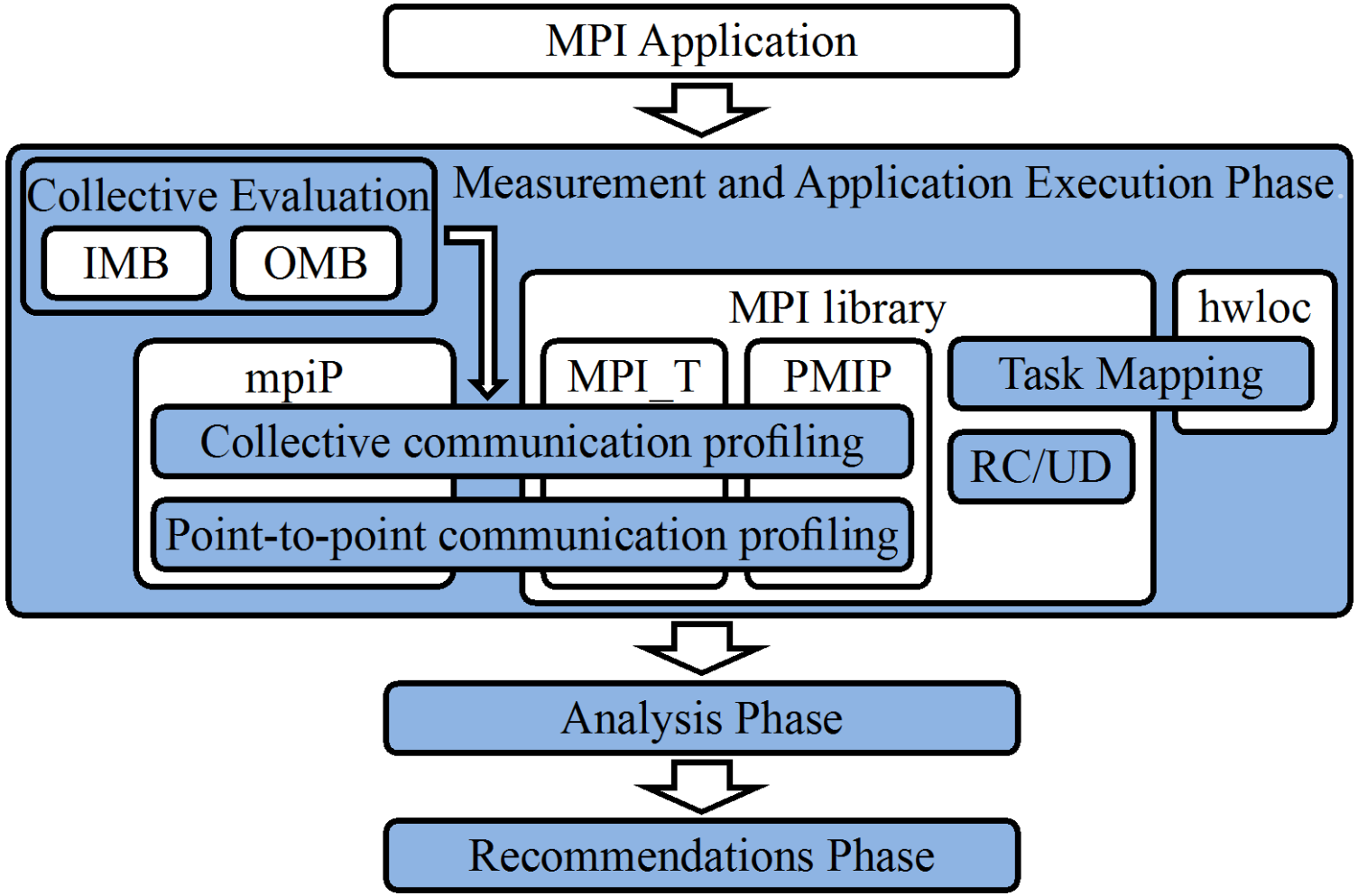
MPI Advisor Functionality

MPI Advisor currently:

1. Does all of the steps in the conceptual approach
2. Implements the measurements without requiring user instrumentation
3. Implements computation of the metrics
4. Implements the algorithms for diagnosing bottlenecks and recommending optimizations
5. Does all the above with a single run of the application and only a few percent overhead.

for a limited subset of the possible parameters, settings and choice of library calls.

Workflow Diagram



Measurements

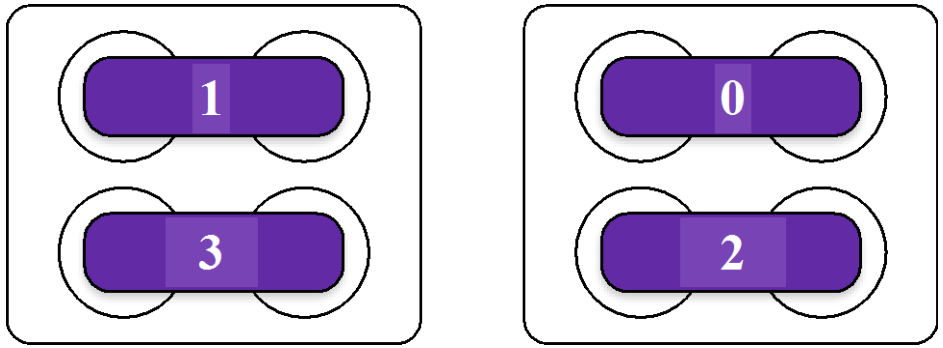
- Execution Environment Parameters – Once on Installation
  - Run IMB and OMB Benchmarks with each library and collect data on performance of each collective algorithm among other data
- Application Measurements
  - Specific to each tuning strategy
  - MPI_P
  - MPI_T
  - POSIX getenv()
  - MPI_get_processor_name()
  - HWLOC

Currently Implemented Optimizations

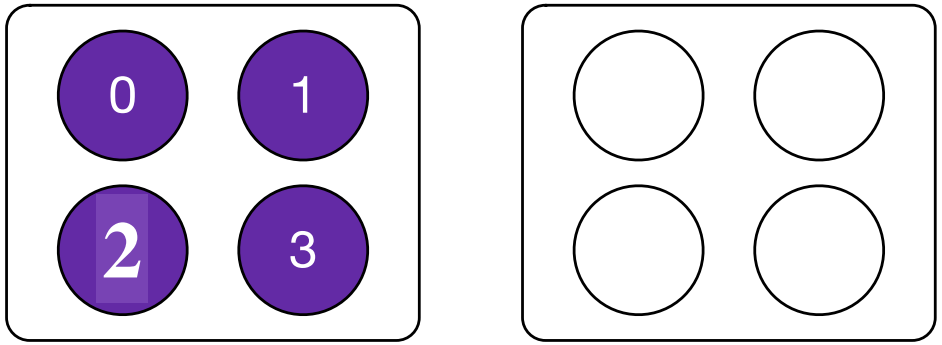
1. Point to Point Protocol Threshold
    1. Eager versus Rendezvous
2. Choice of Algorithm for Collective Operations
    1. Depends on system size, message size and task properties
    2. Default versus custom selected
3. Mapping of MPI Tasks to Cores
    1. Task 0 should be on HCA card
    2. Default mappings versus custom mappings
4. Selection between RC and UD Transport Protocols
    1. Memory and message size trade-off

Mapping of MPI Tasks to Cores - Default Mappings

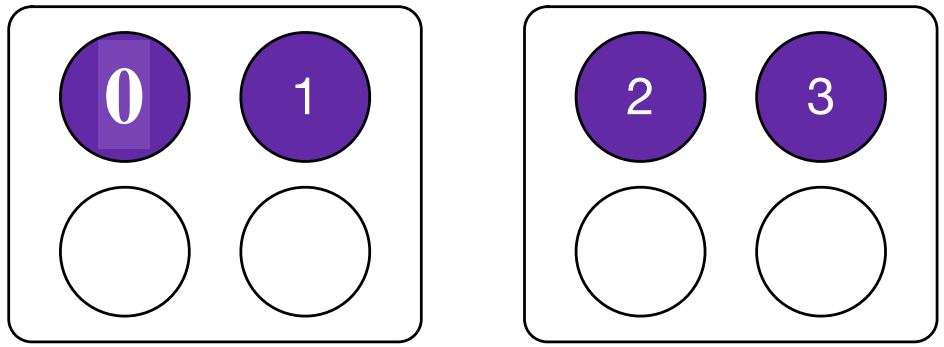
Default Mappings of 4, multi-threaded MPI Tasks for Three MPI Libraries on four-core chip nodes with two chips.



Intel MPI



MVAPICH2



Open MPI

Mapping of MPI Tasks to Cores

1. **Get Data**: Get (via getenv()) the value of OMP_NUM_THREADS, the number of MPI and OpenMP tasks per node, determine (via MPI_Get_processor_name()) the number of MPI tasks per node (MPI_Tasks_per_Node), and (via hwloc) the assignment of tasks/threads to cores, the number of cores on a node (Cores_per_Node), and the location of the HCA;
2. **Check Tasks-to-Cores Assignment:**
   If (all cores assigned to one MPI task) *then* **Check Value of OMP_NUM_THREADS 2**;
   Else **Check Value of OMP_NUM_THREADS 1**;
3. **Check Value of OMP_NUM_THREADS 1:**
   If (OMP_NUM_THREADS set) *then* **Check Number of Cores Used**;
   else output "Warning: Node Under-subscribed" and **Check Location of Rank Zero** ;
4. **Check Value of OMP_NUM_THREADS 2:**
   If (OMP_NUM_THREADS not set or = 1) *then* **Check Location of Rank Zero**;
   else output "Warning: Cores Over-subscribed" and **Check Location of Rank Zero**;
5. **Check Cores_per_Node:**
   If (MPI_Tasks_per_Node * OMP_NUM_THREADS < Cores_per_Node*) then* output "Warning: Node Under-subscribed" and **Check Location of Rank Zero**;
   If (MPI_Tasks_per_Node * OMP_NUM_THREADS > Cores_per_Node) *then* Output "Warning: Node Over-subscribed" and **Check Location of Rank Zero**;
   If (MPI_Tasks_per_Node * OMP_NUM_THREADS = Cores_per_Node) *then* **Check Location of Rank Zero**;
6. **Check Location of Rank Zero:**
   If (Rank Zero process close to HCA card) *then* output "Mapping is OK" and terminate
   else (output "Mapping Recommendation: ……." and terminate

Mapping of MPI Tasks to Cores – Case Study

HPCG Application with MVAPICH 2: Use Default Configuration - Two MPI Tasks with 8 OpenMP threads/task

Measurements

- Tasks are mapped to cores (Slide 11)
- Task 0 which does communication is mapped to socket 1 which is not connected to network.

Recommendations

- Map each task to different socket
- Map Task 0 to Socket 0 which is attached to network

Improvement

- 50% in GFLOPS/sec

Related Work

- Periscope – Complex to use and requires system level access privilege
- Atune – Autotuning – OpenMPI only
- OPTO – OpenMPI only Infiniband tuning
- Vtune – optimize library parameters for Intel MPI
- CrayPat – provides recommendations for rank ordering

Summary and Future Work

- MPI Advisor demonstrates automation of library and parameter level tuning of MPI codes
- Paper to appear in EuroMPI 2015
- MPI Advisor is an ongoing project
  - Additional Library and parameter selection strategies
  - Introduce important source code level optimizations
  - Expend to other MPI implementations
- Long Term Plan
  - Unified approach to optimization of multilevel parallelism.

THE UNIVERSITY OF TEXAS AT AUSTIN

TACC