SI2-SSI: A Comprehensive Performance Tuning Framework for the MPI Stack

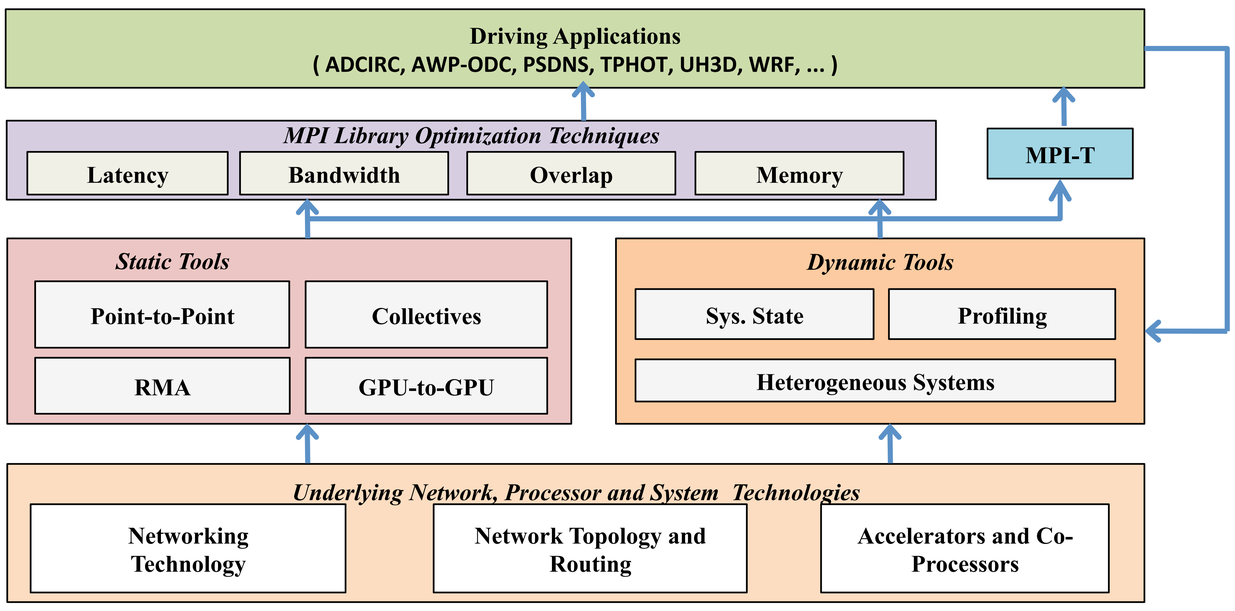
DK Panda (OSU), W. Barth (TACC), A. Majumdar (SDSC) and K. Tomko (OSC)

http://mvapich.cse.ohio-state.edu

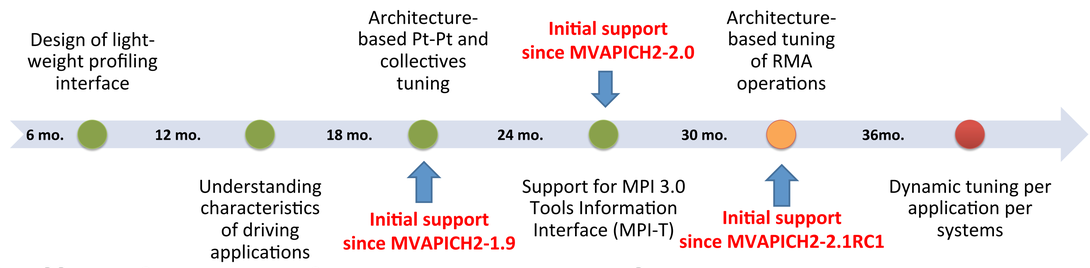
Research Challenges

- Can a set of tools be designed to optimize performance of an MPI library during installation?
- Can a set of dynamic tools with low overhead be designed to optimize performance on a per-user and per-application basis during production runs?
- How to configure MPI libraries on a given system to deliver optimal performance with applications?
- What kind of benefits (in terms of performance, scalability and memory efficiency) can be achieved using the proposed framework?

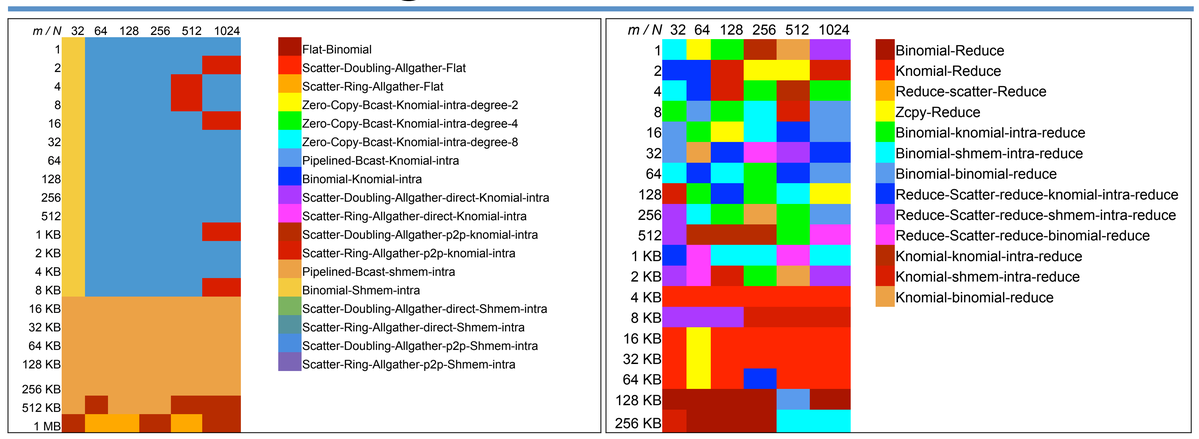
Proposed Framework



Project Milestones



Collective Tuning Framework

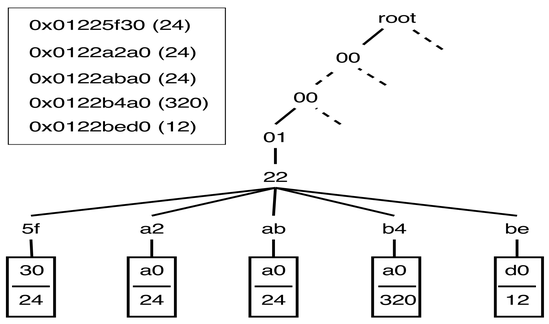


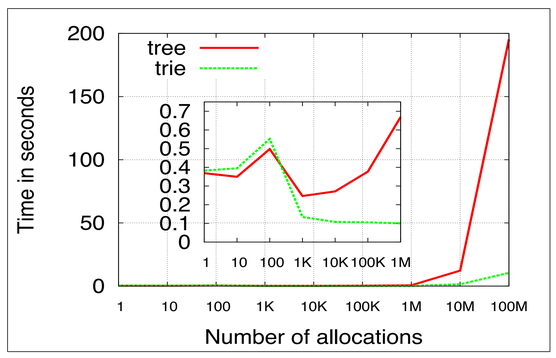
MPI_Bcast : Stampede (N= System size, m= Message size) **MPI_Reduce : Stampede (N= System Size, m= Message size)**

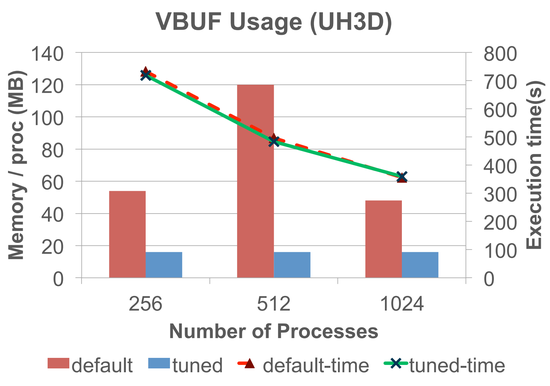
- For each system, for each configuration (message-size, system-size) the best algorithm for both intra-node and inter-node is dynamically selected
- Legend : if the algorithm is a two-level, then left side represents inter-node and right side represents intra-node schemes (ex: Pipeline-Bcast-shmem-intra)

Memory Footprint Profiling / Tuning with MPI3 MPI-T

Trie-based out-of band design for MPI_T memory profiling

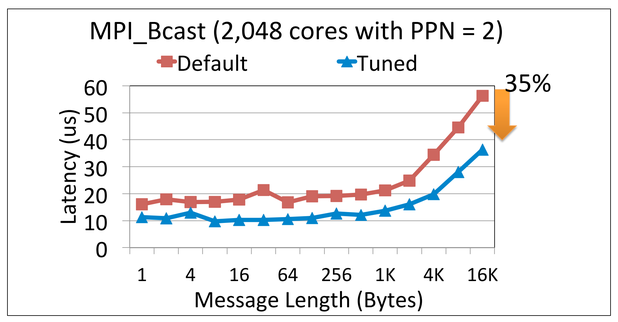


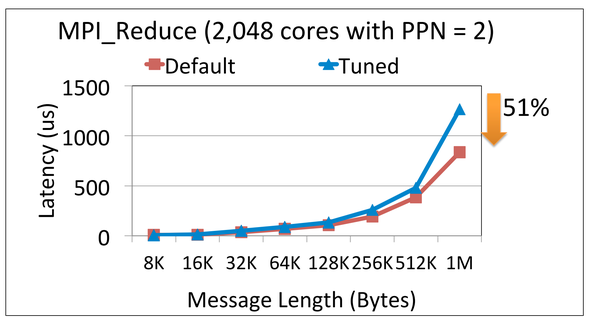


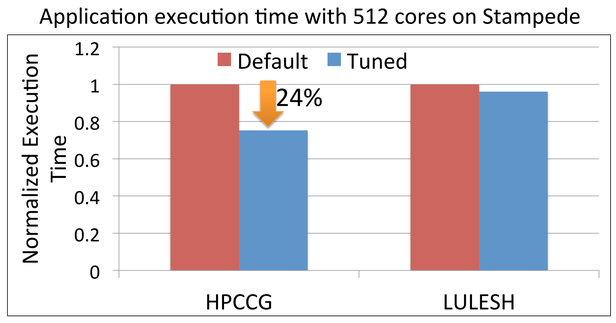


- **Trie** design achieves 10X improvement compared to **Tree** based design in capturing and profiling memory allocation. Thus introduces a negligible overhead on application execution time [EUROMPI 2014]
- For **UH3D** application, MPI-T assisted analysis attributed memory usage to statically-established RC connections
- Optimal selection of UD threshold leads to 7.5X reduction in memory usage
- Total execution time with tuned version was consistently equal, or lesser than, default

Collective Tuning for MPI+OpenMP Programming

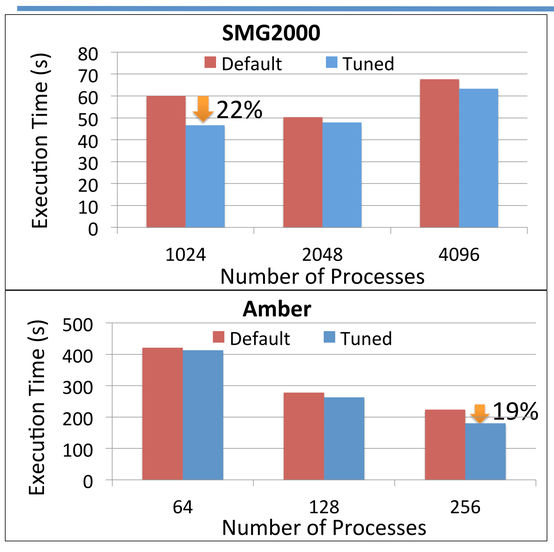






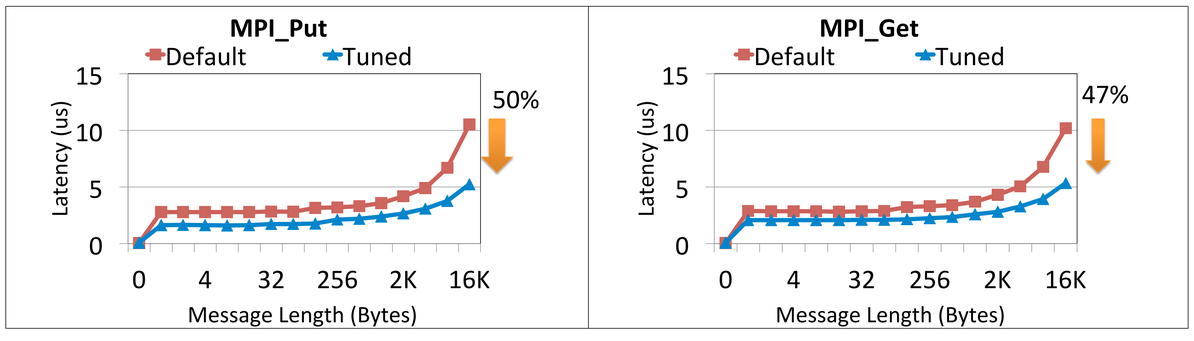
- Partial subscription nature of hybrid MPI +OpenMP programming requires a new level of collectives tuning
- For PPN=2 (Processes Per Node), the tuned version of the MPI_Bcast and MPI_Reduce shows 35% and 51% improvement respectively on 2,048 cores (128 nodes)
- With LULESH and HPCCG applications, on 512 cores (8 OpenMP threads per MPI processes), 4% and 24% improvement respectively

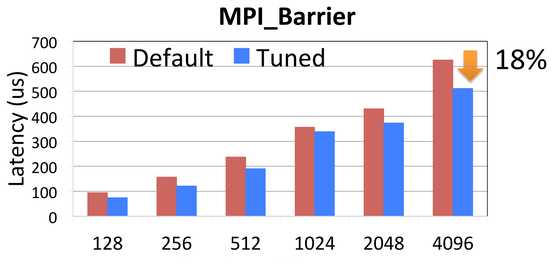
Application Case Studies: SMG2000 and Amber



- All experiments are performed on Stampede system
- **UD-based transport protocol selection** benefits the SMG2000 application execution time by 22% and 6% on 1,024 and 4,096 cores, respectively
- Increasing the **Eager threshold** for Amber Application avoids the synchronization of Rendezvous protocol and thus yielding to a better communication computation overlap. This results in 19% improvement on 256 processes
- This tuning was inferred through analysis of MPI-T counters data

Architecture-based MPI Communication Tuning





- Performance of MPI-3 RMA operations as well as collectives heavily rely on network and processor characteristics
- MVAPICH2 library uses a tuning framework to select optimal collective algorithms and/or the best RMA protocol based on system characteristics yielding best performance
- On systems like Stampede, up 18% improvement is shown from use of tuned Barrier collectives at 4,096 processes
- Tuning RMA operations for a given systems reduces the latency of Put and Get operations by 50% and 47% with 16KB messages

Future Work and Research Dissemination

- Leverage the light-weight profiling interface to dynamically tune applications on systems
- Creating a Best Practice page under the MVAPICH2 Website to disseminate this information.
- Over 235K downloads from the project website (from 2,325 organizations across 75 countries) as of Feb'15

• OSU/OSC (Award #1148371) - Lead Institution
• TACC (Award #1148424)
• SDSC (Award #1147926)