Adaptive MPI: Dynamic Runtime Support for MPI Applications

Sam White and Laxmikant V. Kale
University of Illinois at Urbana-Champaign

Abstract

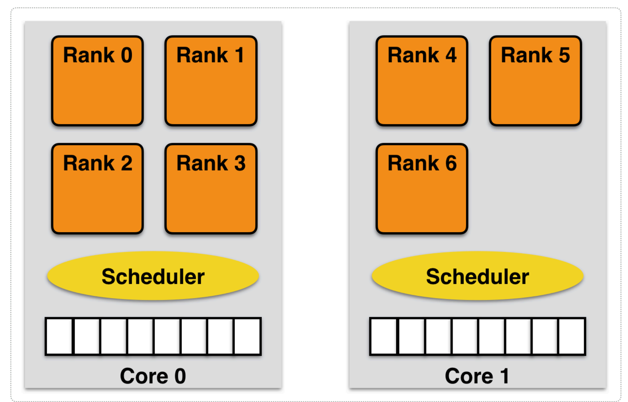
Adaptive MPI (AMPI) is an implementation of the MPI standard written on top of Charm++ and its adaptive runtime system. AMPI provides application-independent support for overdecomposition, dynamic load balancing, communication/computation overlap, and online fault tolerance.

AMPI programs are MPI programs without mutable global/static variables, or with them properly handled.

Execution Model

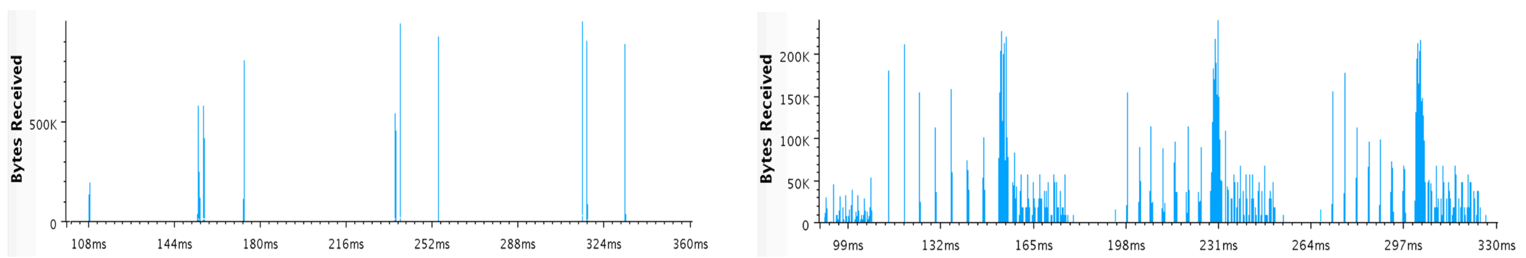
In AMPI, the ranks of *MPI_COMM_WORLD* are implemented as user-level threads (not OS processes):

- Can have multiple per core
- Fast to context switch
- Scheduled based on message delivery
- Migratable between address spaces at runtime



AMPI overlaps communication of one rank with computation of other ranks on the same core.

- Communication is spread over the timestep
- For LULESH, 8 ranks/core provides a 4x reduction in the peak network bandwidth needed

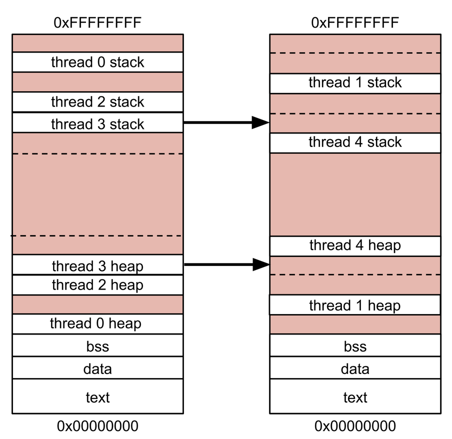


1 rank/core   8 ranks/core

Load Balancing

AMPI's Isomalloc memory allocator enables transparent migration of AMPI ranks and all their data.

- Isomalloc reserves virtual memory space for each rank on every core
- Users just call *AMPI_Migrate*()
- AMPI collects load statistics
- LB strategies are runtime options
- Users can write custom strategies

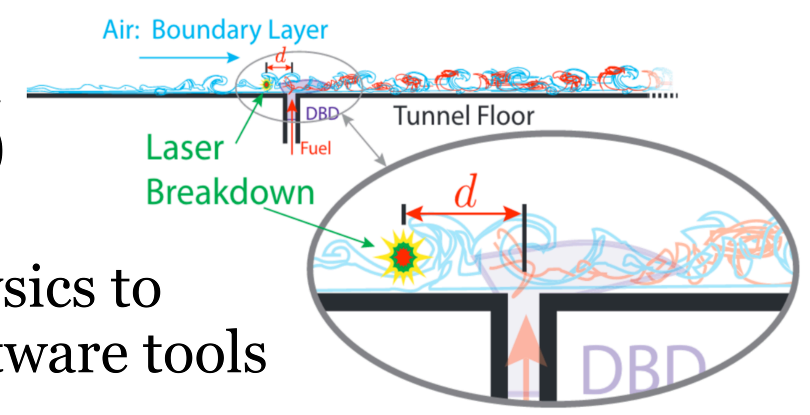


Applications

PlasComCM

Main simulation code for the PSAAPII Center for Exascale Simulation of Plasma-Coupled Combustion (XPACC).

- Challenge: multi-rate time integration needed to deal with multiple timescales (ns/us/ms)
- "Golden copy" approach: computationalists add new physics to the Fortran90 & MPI code, software tools can transform it but:
  - No new programming languages
  - Minimal changes to existing code



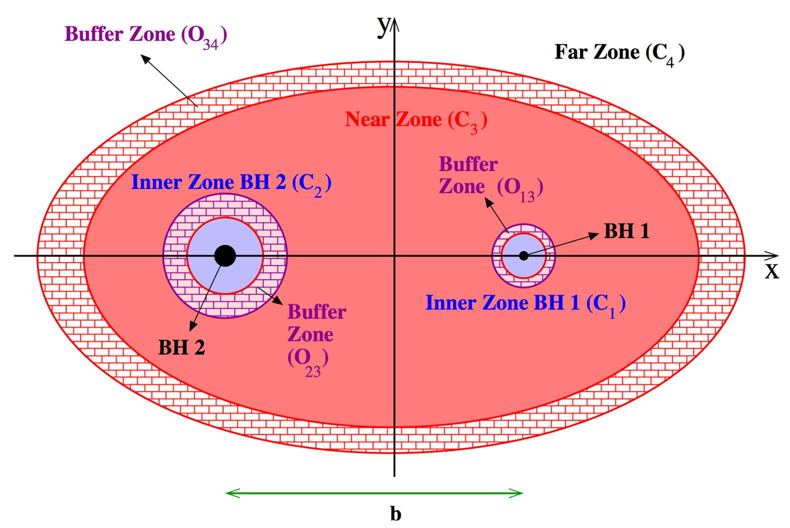
Ranks	NoLB	GreedyLB	RefineLB	DistribLB	MetisLB	ScotchLB	MetaLB
1024	1.00	---	---	---	---	---	---
4096	1.05	1.14	1.15	1.06	1.08	1.13	1.14 (GreedyLB)
8192	1.07	1.19	1.14	1.09	1.06	1.17	1.19 (GreedyLB)
16384	1.04	1.18	1.14	1.08	1.06	1.16	1.16 (ScotchLB)

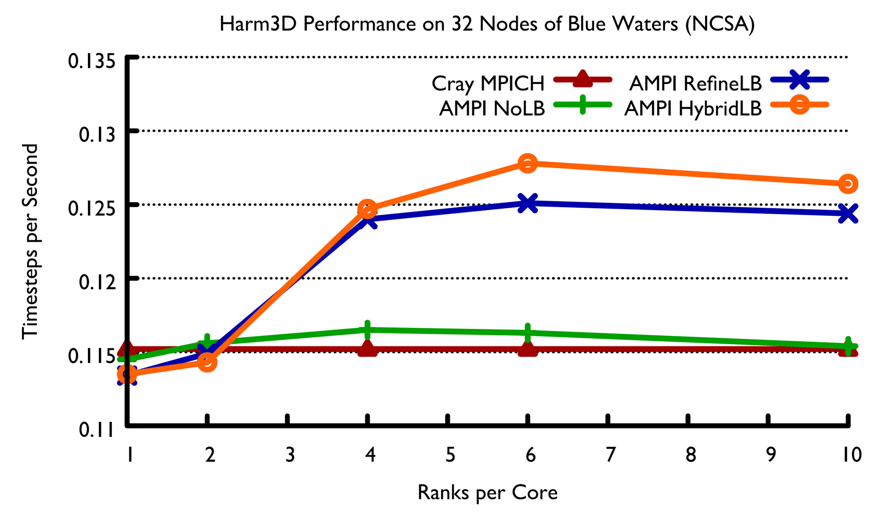
Above: PlasComCM simulation on 1024 cores of Quartz (LLNL) with different load balancing strategies. Speedups are normalized to 1 rank per core, no load balancer.

Harm3D

Solves the magnetohydrodynamics equations of motion in curved spacetime. Developed by Scott Noble at the University of Tulsa.

- Existing C & MPI code uses domain decomposition, no prior support for dynamic load balancing
- Future challenge: simulation of multiple accreting black holes suffers from load imbalance across ranks, varying over time
  - Buffer zone computations cost 3-4x more FLOPs than far zone, black holes move through the domain

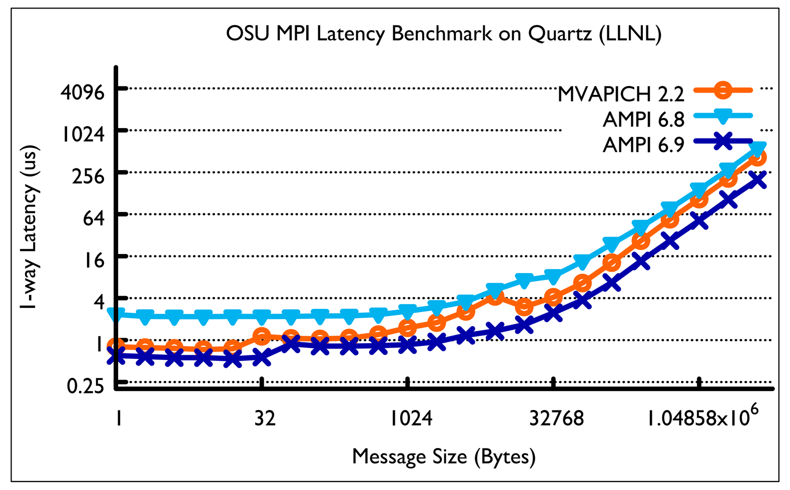




Shared Memory Messaging

AMPI optimizes for messages sent within the same process.

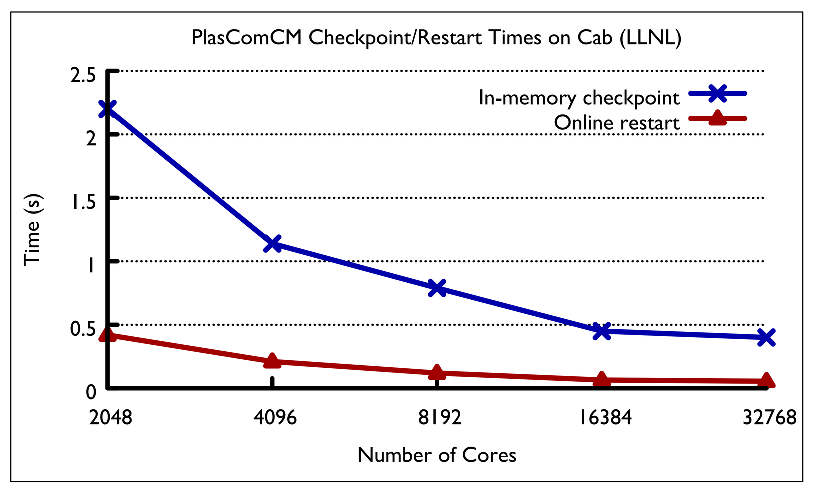
- Zero copy messaging: low latency, reduced memory footprint
- No NIC traffic for in-process sends
- Comm-aware load balancers try to co-locate ranks that communicate



Online Fault Tolerance

In AMPI, a checkpoint is simply a migration to storage.

- Storage can be parallel file system, SSDs, remote RAM, NVRAM, etc.
- AMPI automatically detects failures and restarts all ranks from last checkpoint online (no job restart)
- With Isomalloc, only user code needed: one call to *AMPI_Migrate*()



Conclusions

Performance

- AMPI optimizes communication based on locality
- Users can tune the number of ranks per core based on cache sizes, communication overlap, etc.
- Plug-in interface for dynamic load balancing strategies
- Checkpoint/restart-based fault tolerance schemes

Productivity

- No need to rewrite existing MPI applications for:
  - Dynamic load balancing
  - Latency tolerance
  - Hard fault resilience

Ongoing work

- Automatic global/static variable privatization via *Process-in-Process* library or *icc –fmpc-privatize*
- Further shared-memory awareness
- Compliance with the latest MPI-3.1 standard

This work was funded by US DOE Award Number DE-NA0002374 and US NSF Award Number OCI 07-25070.