
Enabling Low-Overhead Hybrid MPI/OpenMP Parallelism with MPC

Patrick Carribault, Marc Pérache and Hervé
Jourdren

CEA, DAM, DIF, F-91297 Arpajon France



- HPC Architecture: Petaflop/s Era
 - Multicore processors as basic blocks
 - Clusters of ccNUMA nodes
- Parallel programming models
 - MPI: distributed-memory model
 - OpenMP: shared-memory model
- Hybrid MPI/OpenMP (or *mixed-mode programming*)
 - Promising solution (benefit from both models for data parallelism)
 - How to *hybridize* an application?
- Contributions
 - Approaches for hybrid programming
 - Unified MPI/OpenMP framework (MPC) for lower hybrid overhead



- Introduction/Context
- **Hybrid MPI/OpenMP Programming**
 - Overview
 - Extended taxonomy
- MPC Framework
 - OpenMP runtime implementation
 - Hybrid optimization
- Experimental Results
 - OpenMP performance
 - Hybrid performance
- Conclusion & Future Work



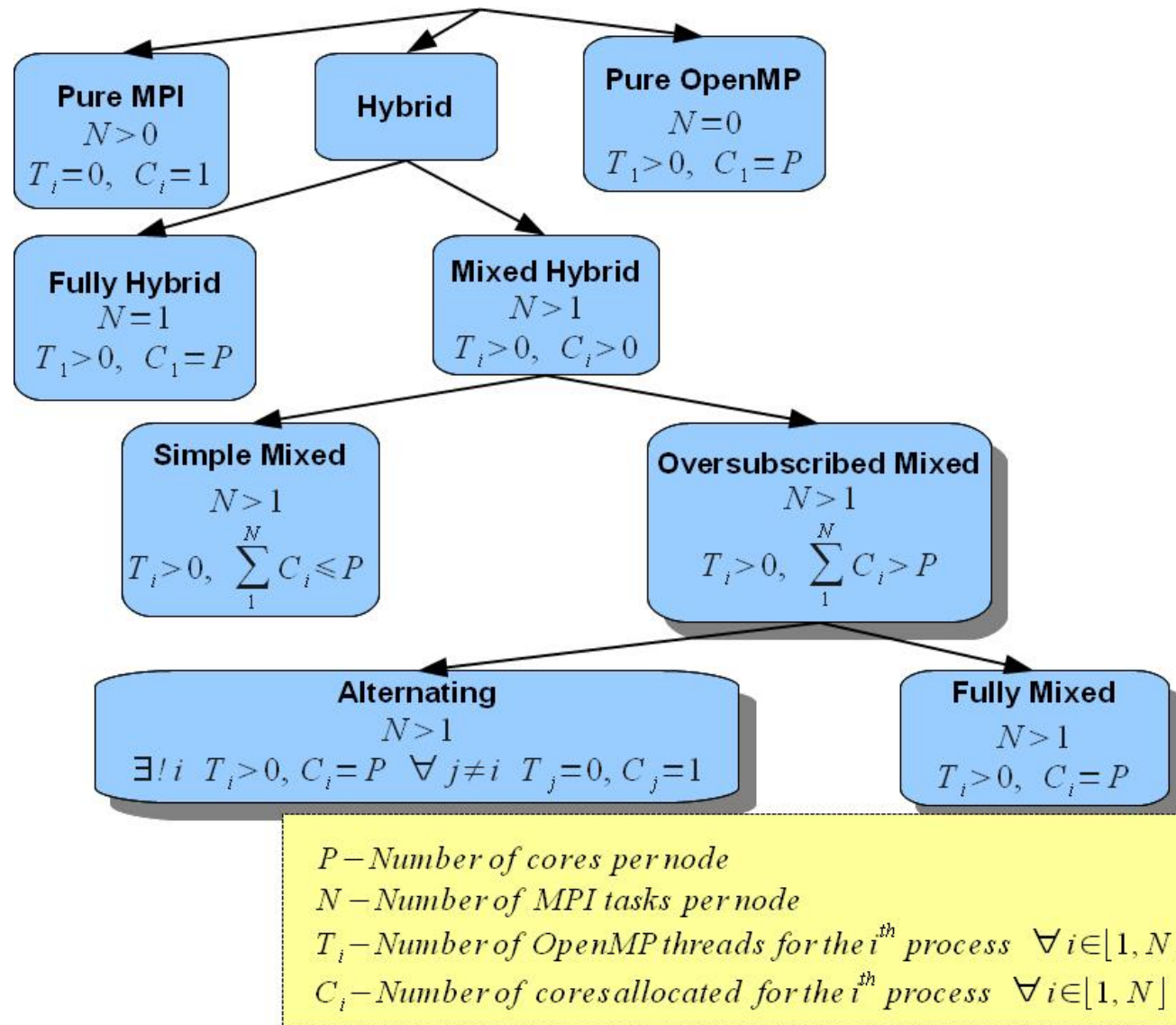
- MPI (*Message Passing Interface*)
 - Inter-node communication
 - Implicit locality
 - Useless data duplication and useless shared-memory transfers
- OpenMP
 - Fully exploit shared-memory data parallelism
 - No inter-node standard
 - No data-locality standard (ccNUMA node)
- Hybrid Programming
 - Mix MPI and OpenMP inside an application
 - Benefit from Pure-MPI and Pure-OpenMP modes



- Traditional Approaches
 - Exploit one core with one execution flow
 - E.g., MPI for inter-node communication, OpenMP otherwise
 - E.g., multi core CPU Socket-exploitation with OpenMP
- Oversubscribing Approaches
 - Exploit one core with several execution flows
 - Load balancing on the whole node
 - Adaptive behavior between parallel regions
- Mixing Depth
 - Communications *outside* parallel regions
 - Network bandwidth saturation
 - Communications *inside* parallel regions
 - MPI thread-safety

➔ Extended Taxonomy from [Heager09]

Hybrid MPI/OpenMP Extended Taxonomy



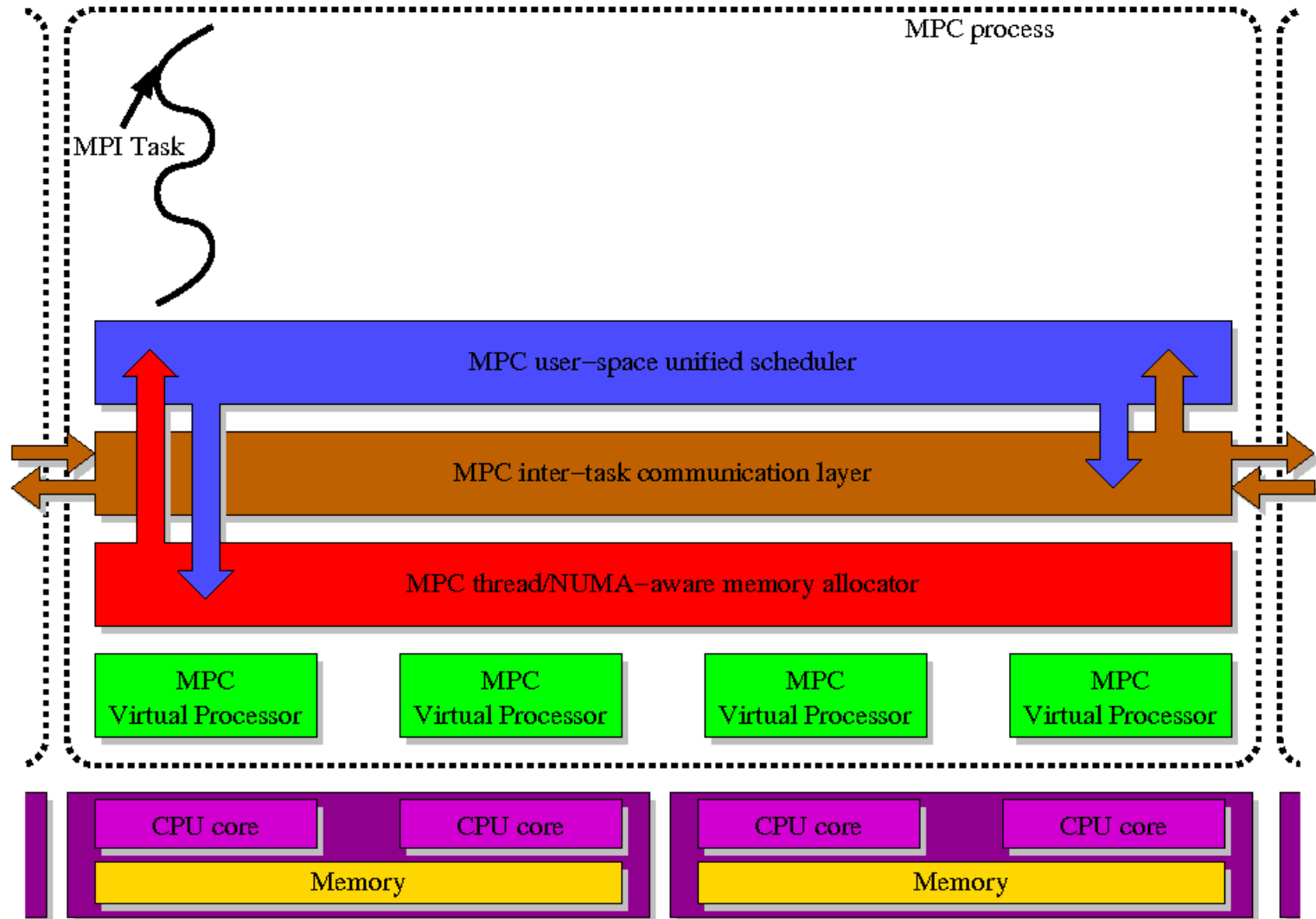


- User-level thread library [EuroPar'08]
 - Pthreads API, debugging with GDB [MTAAP'2010]
- Thread-based MPI [EuroPVM/MPI'09]
 - MPI 1.3 Compliant
 - Optimized to save memory
- NUMA-aware memory allocator (for multithreaded applications)
- Contribution: Hybrid representation inside MPC
 - Implementation of OpenMP Runtime (2.5 compliant)
 - Compiler part w/ patched GCC (4.3.X and 4.4.X)
 - Optimizations for hybrid applications
 - Efficient oversubscribed OpenMP (more threads than cores)
 - Unified representation of MPI tasks and OpenMP threads
 - Scheduler-integrated polling methods
 - Message-buffer privatization and parallel message reception

MPC's Hybrid Execution Model (*Fully Hybrid*)



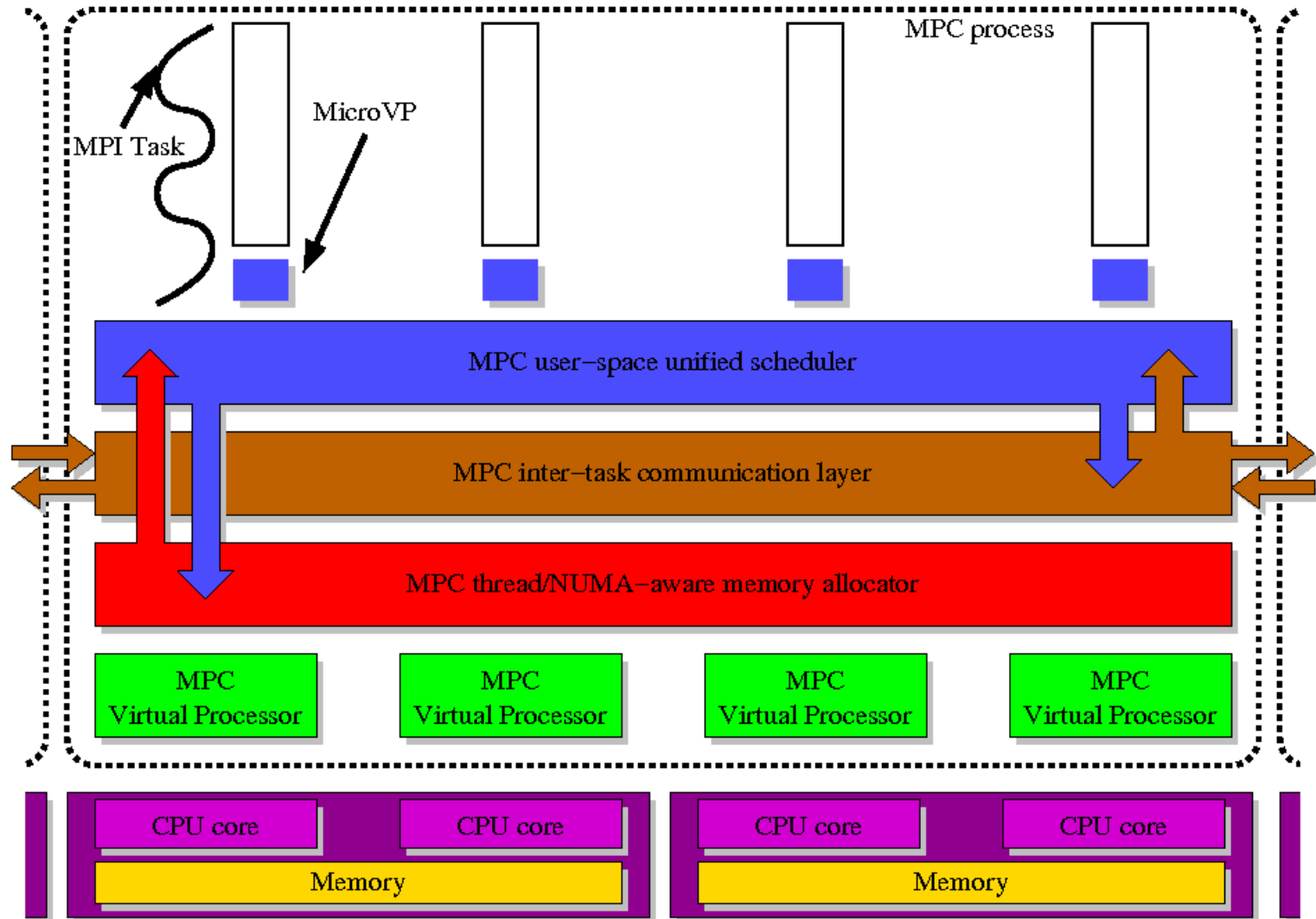
- Application with 1 MPI task per node



MPC's Hybrid Execution Model (*Fully Hybrid*)

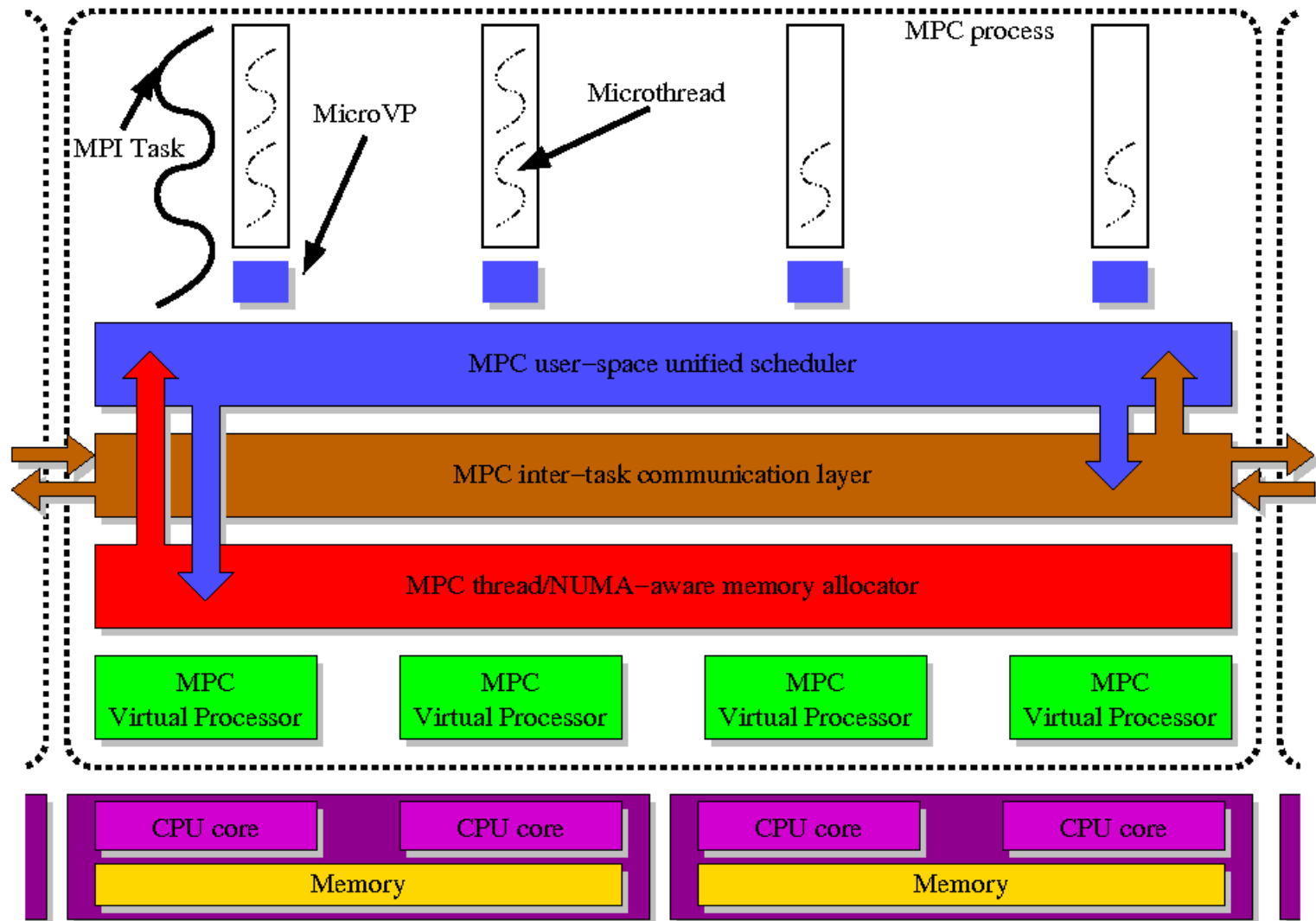


- Initialization of OpenMP regions (on the whole node)



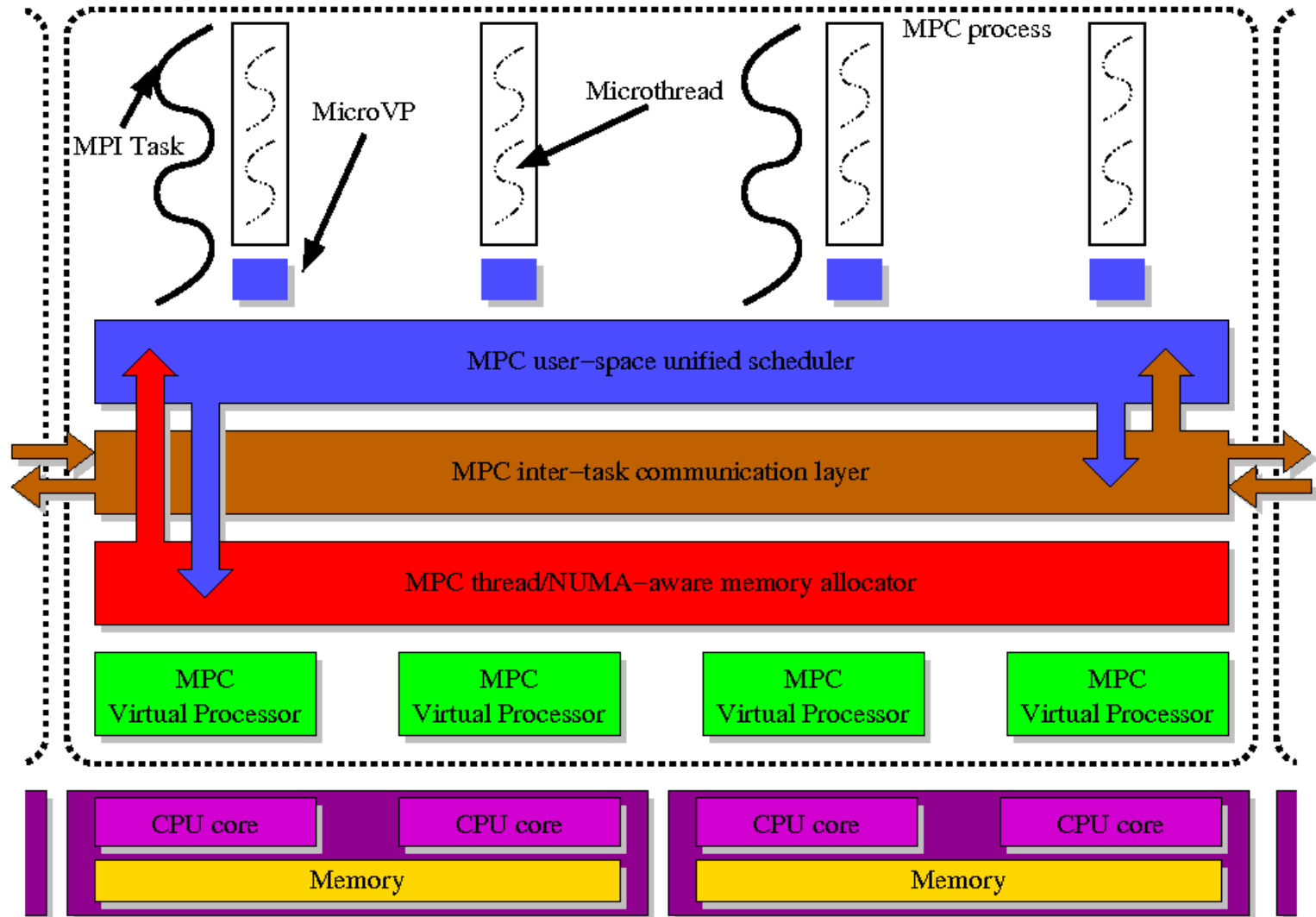
MPC's Hybrid Execution Model (*Fully Hybrid*)

- Entering OpenMP parallel region w/ 6 threads



MPC's Hybrid Execution Model (*Simple Mixed*)

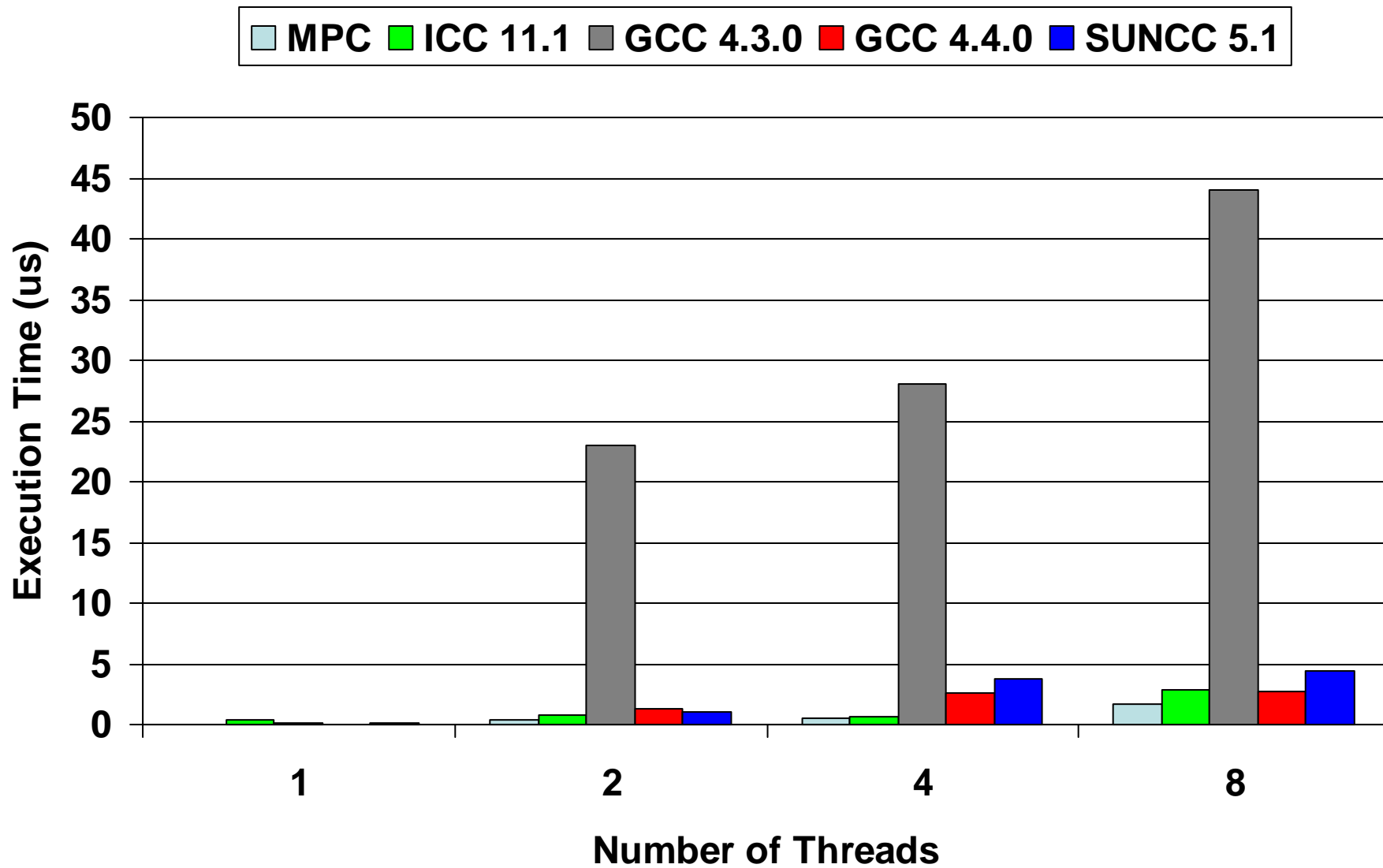
- 2 MPI tasks + OpenMP parallel region w/ 4 threads (on 2 cores)



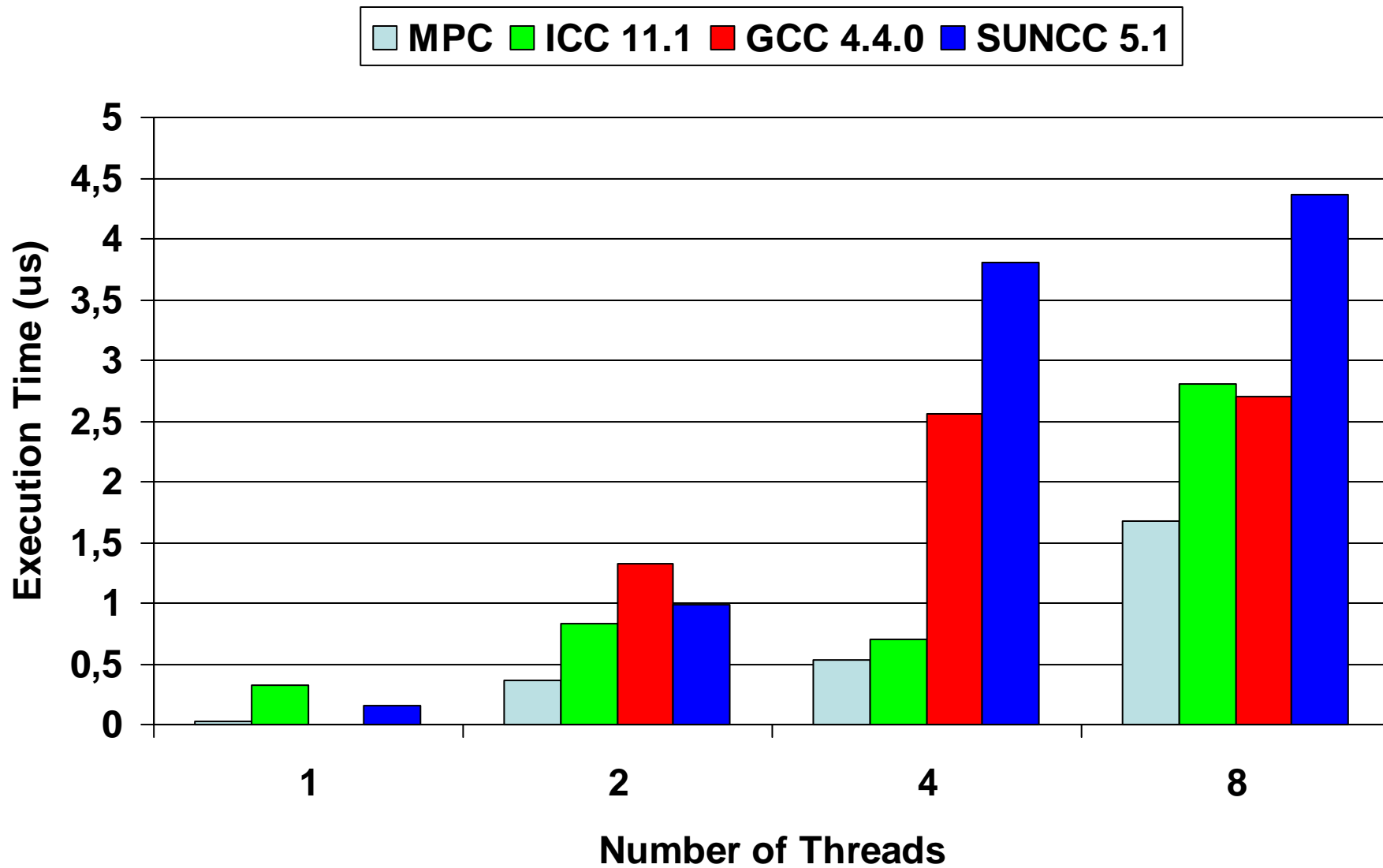


- Architecture
 - Dual-socket Quad-core Nehalem-EP machine
 - 24GB of memory/Linux 2.6.31 kernel
- Programming model implementations
 - MPI: MPC, IntelMPI 3.2.1, MPICH2 1.1, OpenMPI 1.3.3
 - OpenMP: MPC, ICC 11.1, GCC 4.3.0 and 4.4.0, SunCC 5.1
 - Best option combination
 - OpenMP thread pinning (KMP_AFFINITY, GOMP_CPU_AFFINITY)
 - OpenMP wait policy (OMP_WAIT_POLICY, SUN_MP_THR_IDLE=spin)
 - MPI task placement (I_MPI_PIN_DOMAIN=omp)
- Benchmarks
 - EPCC suite (Pure OpenMP/Fully Hybrid) [Bull *et al.* 01]
 - Microbenchmarks for mixed-mode OpenMP/MPI [Bull *et al.* IWOMP'09]

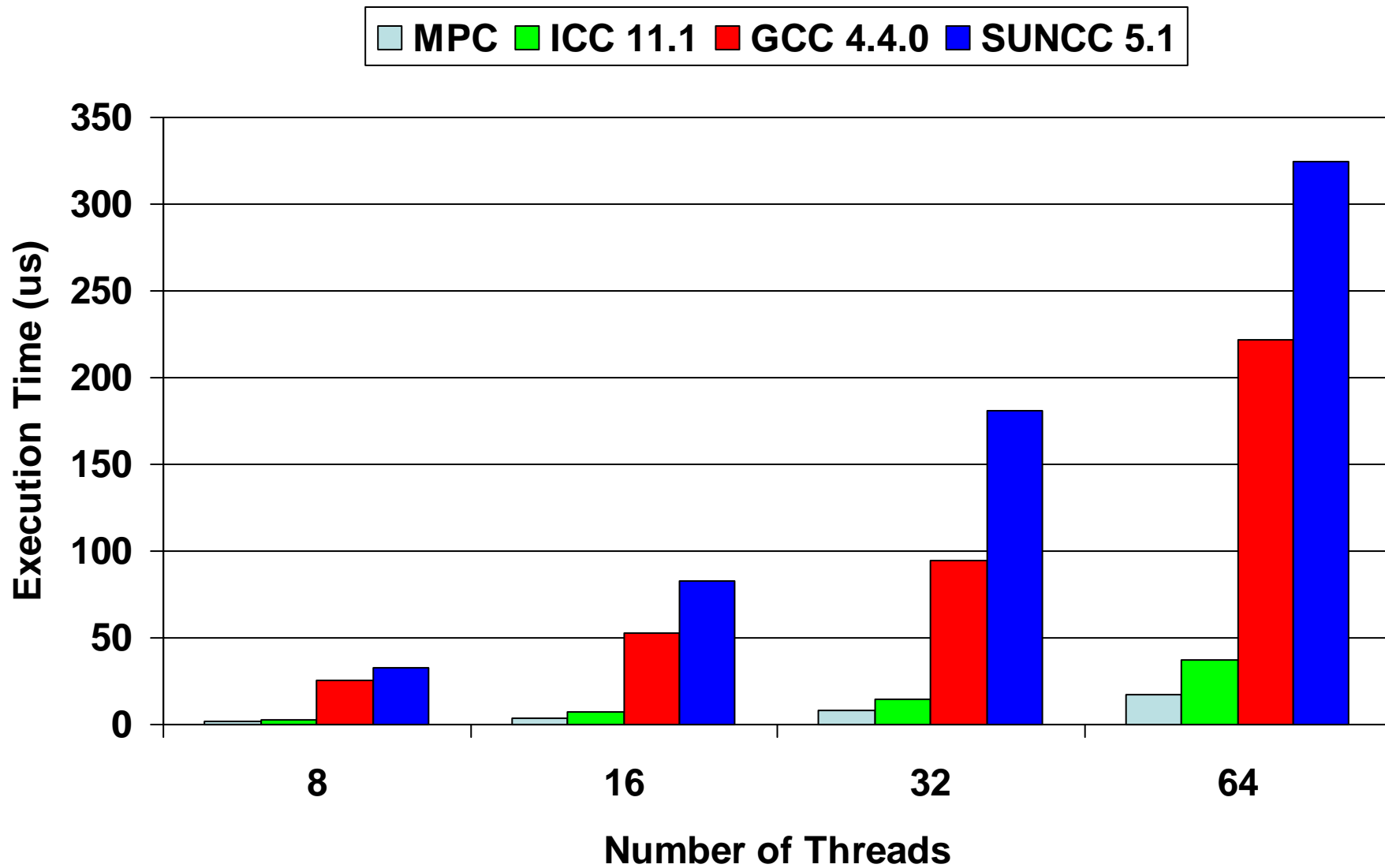
EPCC: OpenMP Parallel-Region Overhead



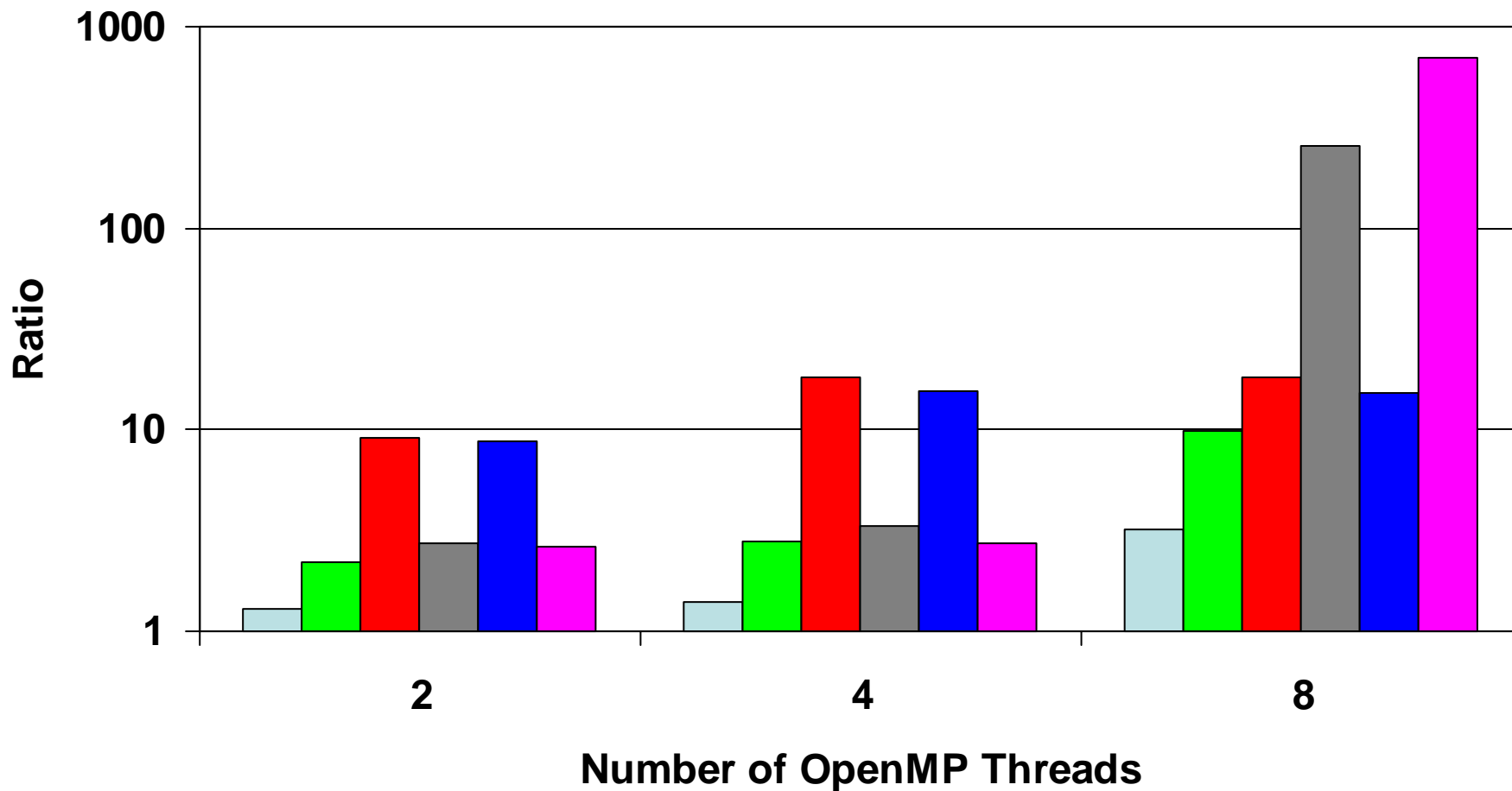
EPCC: OpenMP Parallel-Region Overhead (cont.)



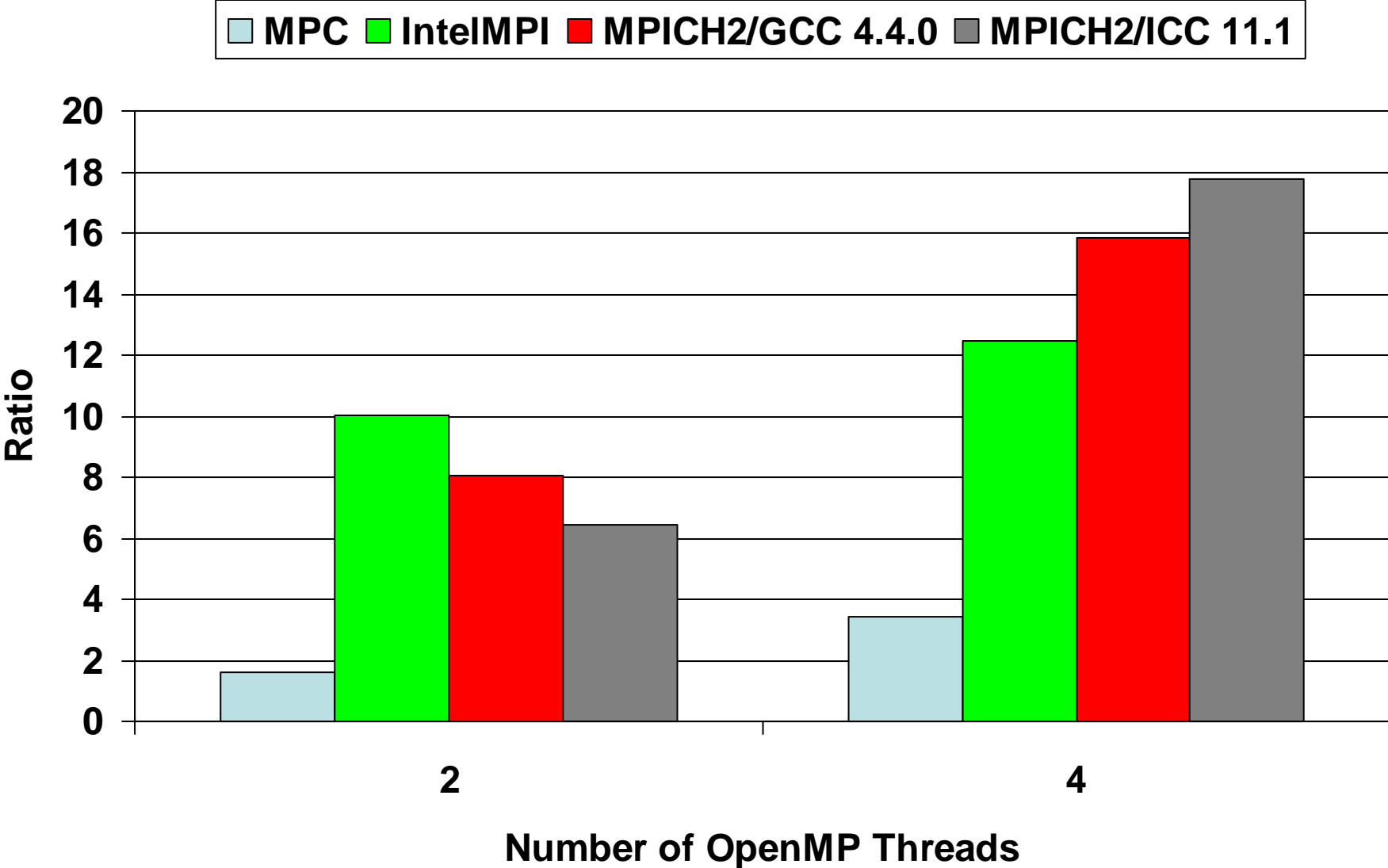
EPCC: OpenMP Parallel-Region Overhead (cont.)



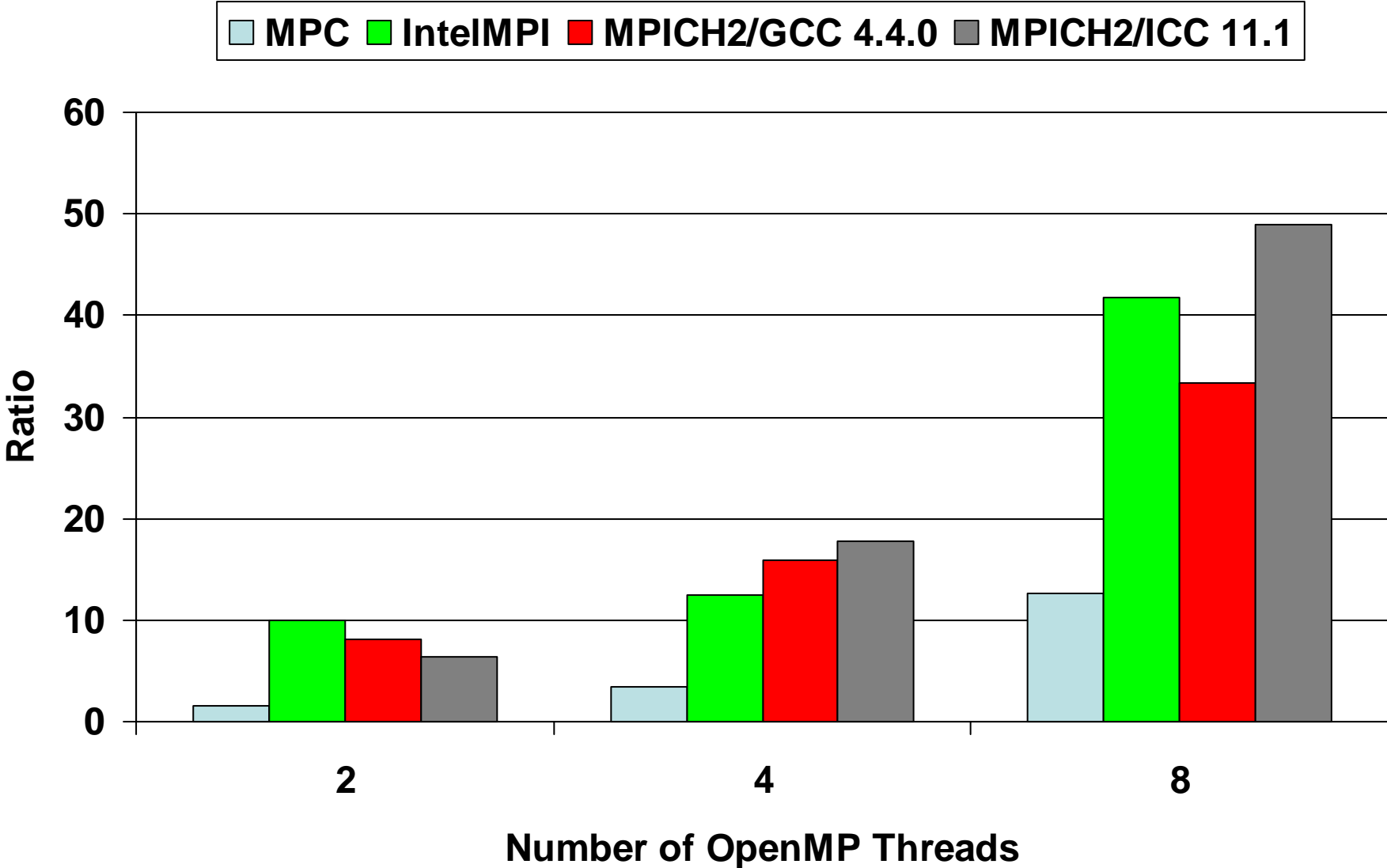
Hybrid Funneled Ping-Pong (1KB)



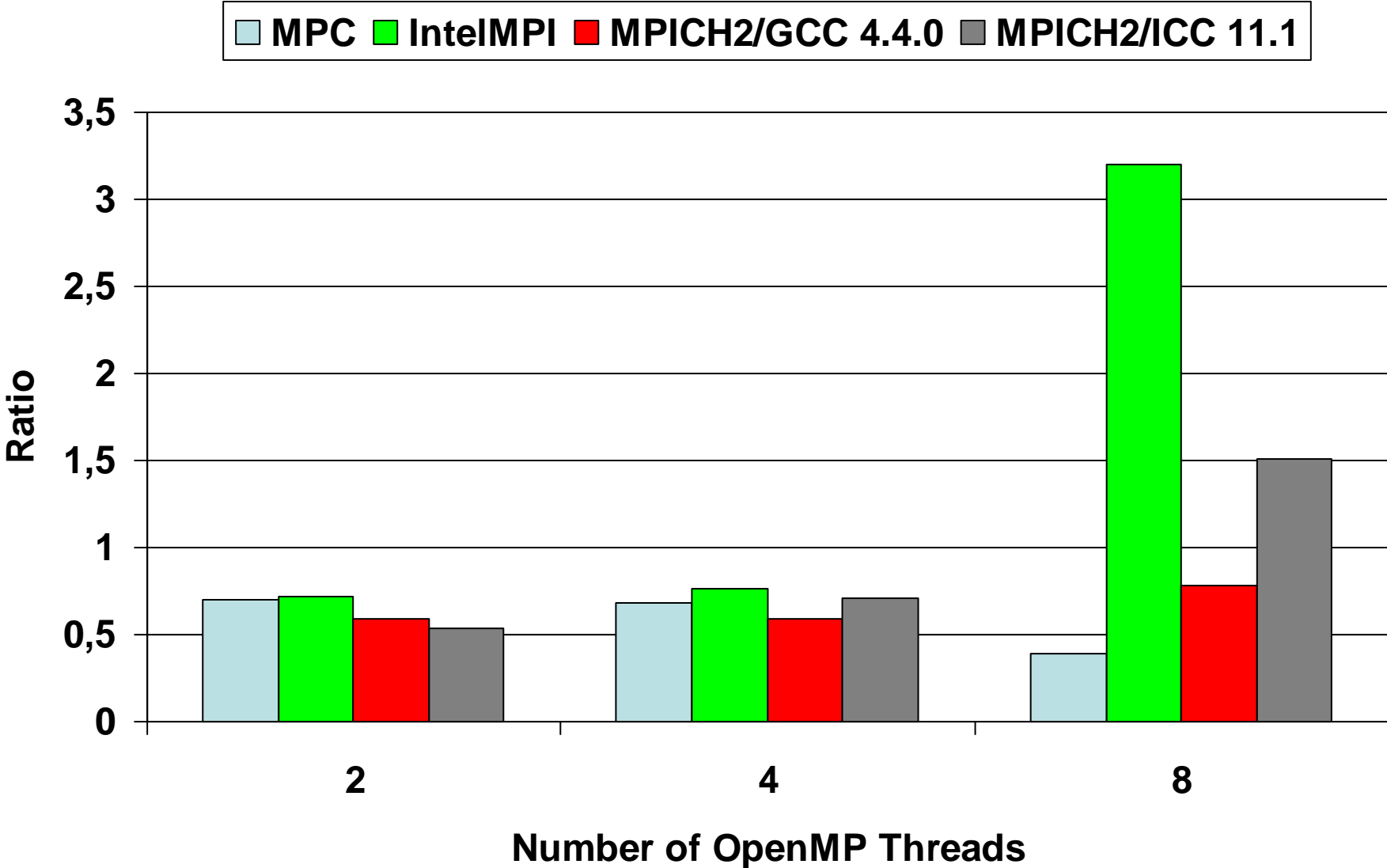
Hybrid Multiple Ping-Pong (1KB)



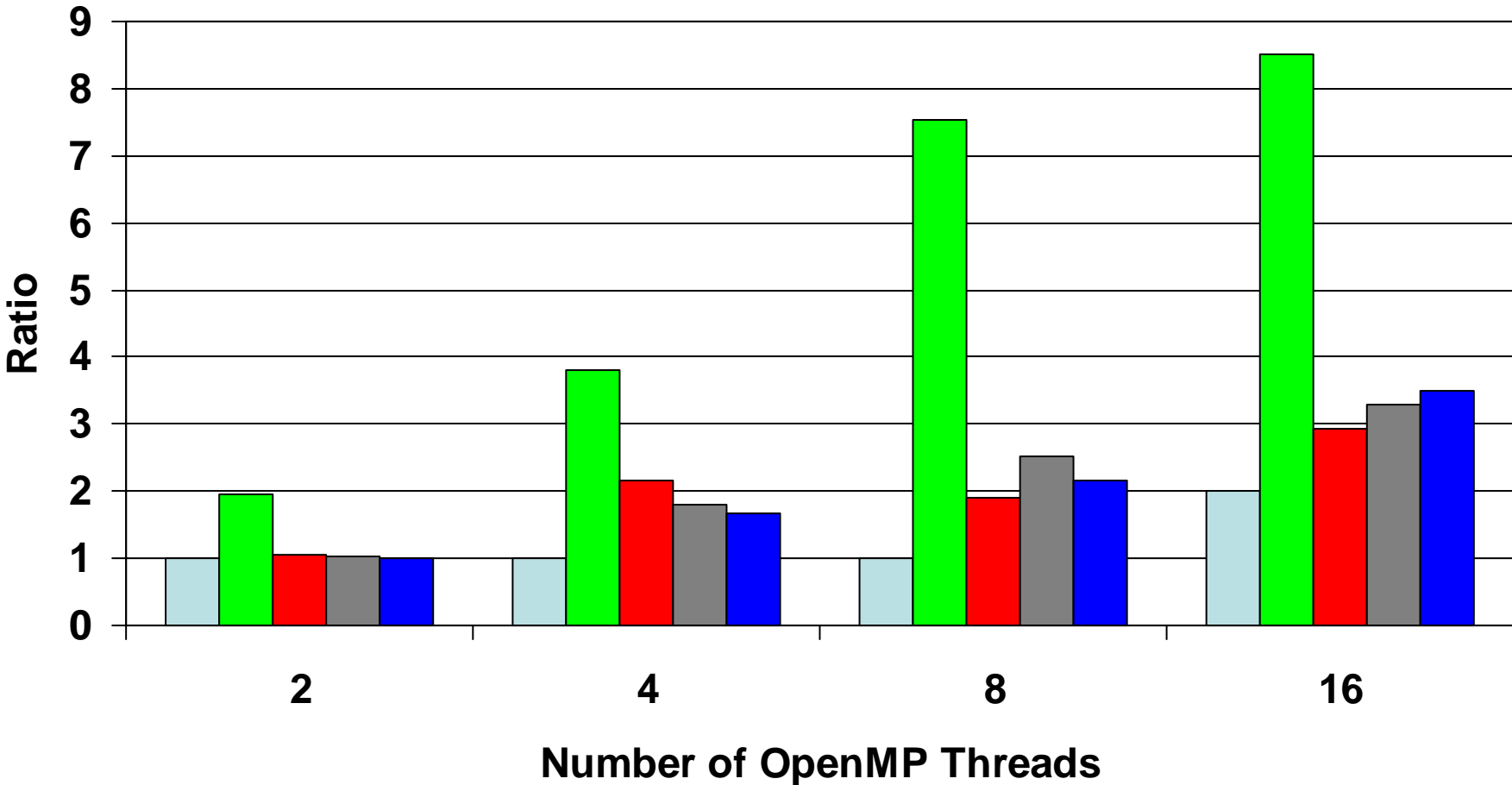
Hybrid Multiple Ping-Pong (1KB) (cont.)



Hybrid Multiple Ping-Pong (1MB)



Alternating (MPI Tasks Waiting)





- Mixing MPI+OpenMP is a promising solution for next-generation computer architectures → How to avoid large overhead?
- Contributions
 - Taxonomy of hybrid approaches
 - MPC: a Framework unifying both programming models
 - Lower hybrid overhead
 - Fully compliant MPI 1.3 and OpenMP 2.5 (with patched GCC)
 - Freely available at <http://mpc.sourceforge.net> (version 2.0)
 - Contact: patrick.carribault@cea.fr or marc.perache@cea.fr
- Future Work
 - Optimization of OpenMP runtime (e.g., NUMA barrier)
 - OpenMP 3.0 (tasks)
 - Thread/data affinity (thread placement, data locality)
 - Tests on large applications