Cheetah: A Framework for Scalable Collective Operations

Project members

Richard L. Graham
Pavel Shamis
Joshua S. Ladd
Manjunath Gorentla Venkata

Computer Science and Mathematics Division
Oak Ridge National Laboratory

Work supported by US DOE ASCR FASTOS and Math/CS Institute EASI!

Need for Efficient Collectives

What are Collectives ?

A global communication operation performed over all processes of a parallel application

- A large percentage of High Performance Computing (HPC) application execution time is spent in global communication operations (collectives)
- Moving towards exascale systems, the time spent in collectives only increases (unless the applications change)
- Collectives are basic building blocks for many parallel programming languages and communication libraries

Collectives on Modern HPC Systems

Modern HPC systems have multiple data paths for communication, the collectives should take advantage of the system architecture to improve the performance of collectives

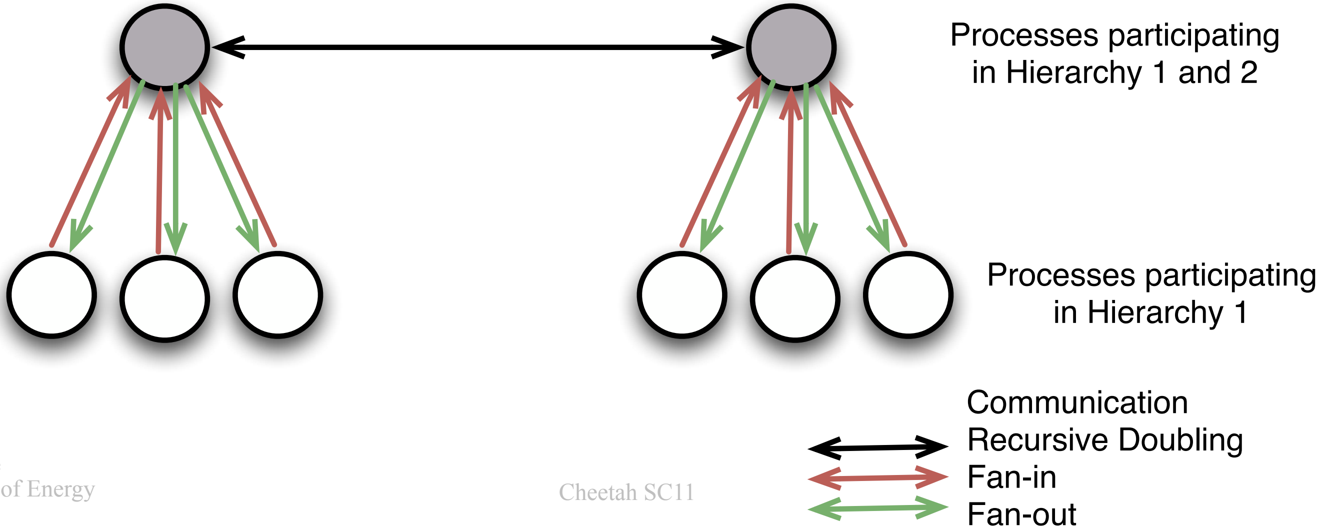
Previous Approaches

- Collectives optimized by replacing point-to-point communication by shared-memory communication
- Collectives optimized for a particular architecture
- Collectives with limited hierarchy support

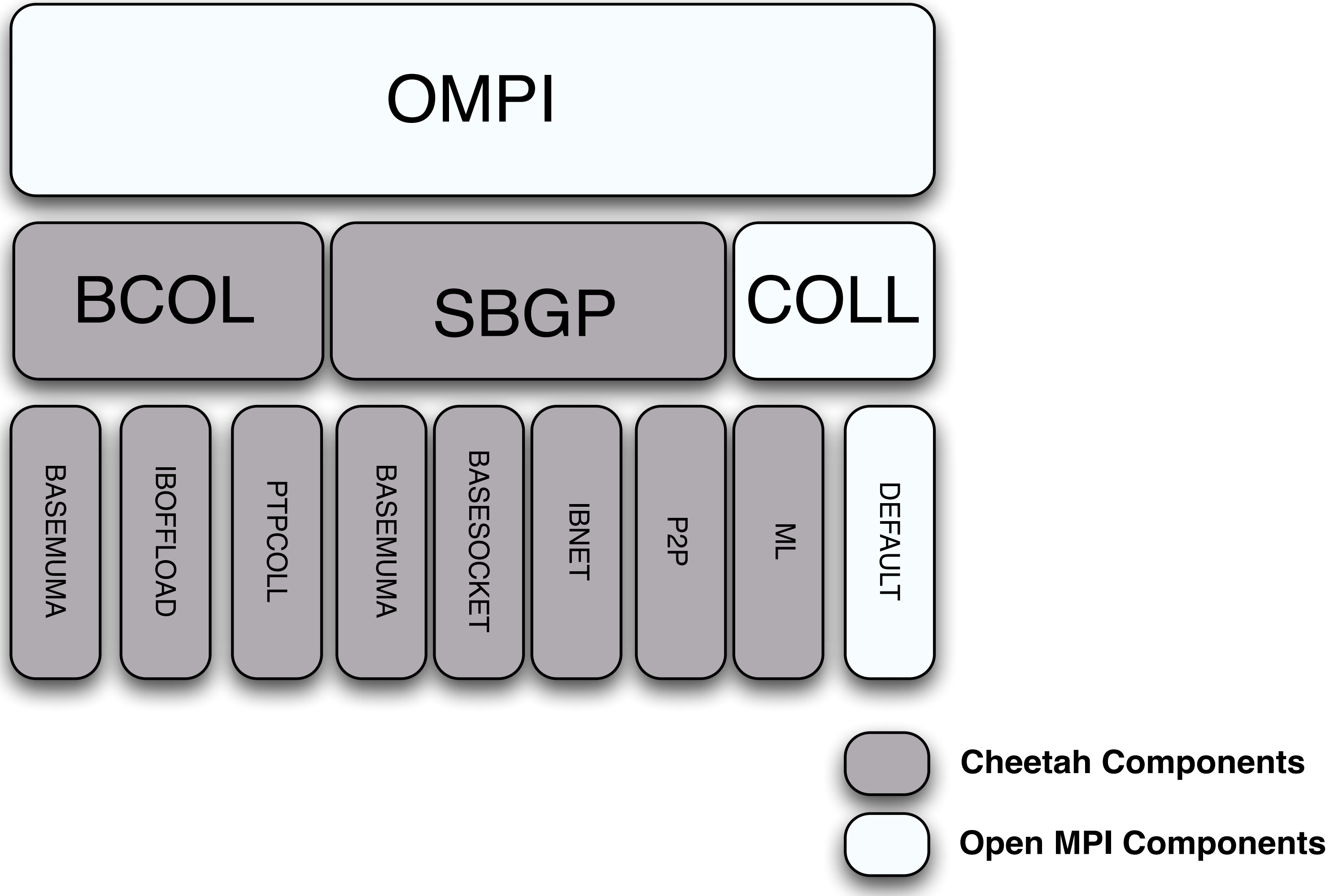
Cheetah: A Framework for Scalable Collective Operations

- Cheetah collectives are implemented as a combination of multiple collective primitives
- Collective primitive is optimized for a particular data communication path (communication hierarchy)
- Collective primitives are progressed asynchronously and independently (when semantics permit)

Example: A n-level hierarchial Barrier is a combination of fan-in, fan-out and Barrier collective primitives


Processes participating in Hierarchy 1 and 2
Processes participating in Hierarchy 1
Communication
Recursive Doubling
Fan-in
Fan-out

Cheetah is Implemented as a Part of Open MPI

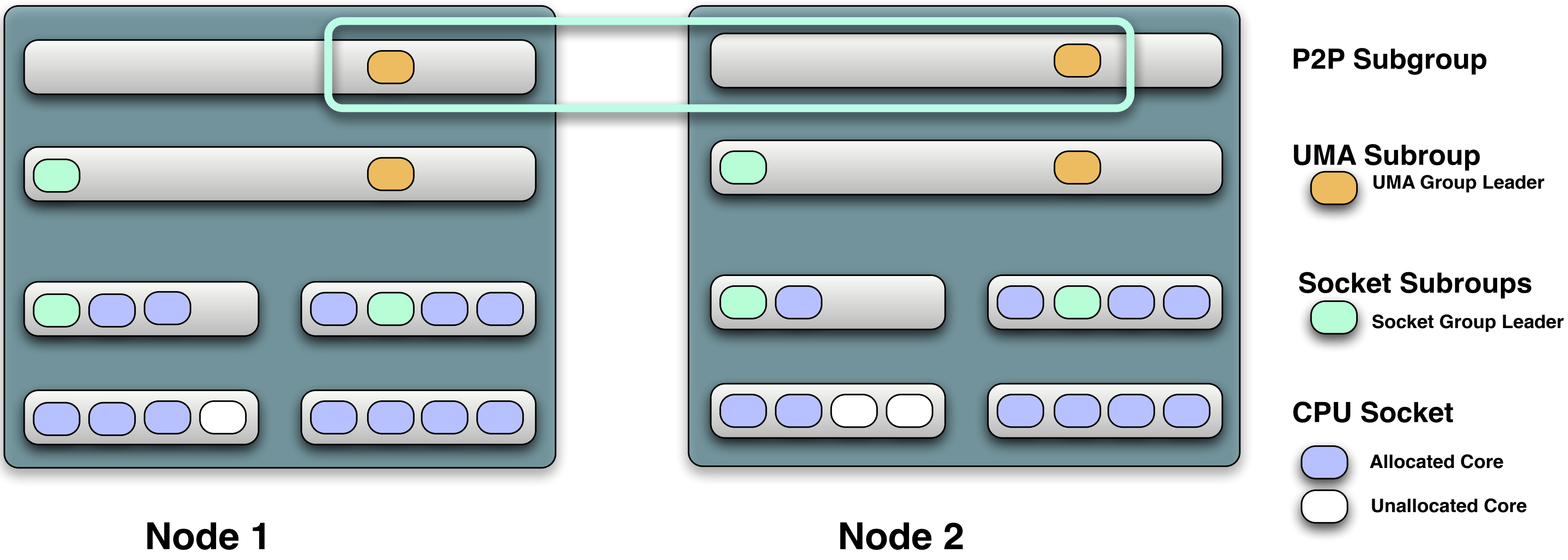


Cheetah Components and Functionality

- **Base Collectives (BCOL) – Implements basic collective primitives**
- **Subgrouping (SBGP) – Provides rules for grouping the processes**
- **Multilevel (ML) – Coordinates collective primitive execution, manages data and control buffers, and maps MPI semantics to BCOL primitives**
- **Schedule – Defines the collective primitives that are part of collective operation**
- **Progress Engine – Responsible for starting, progressing, and completing the collective primitives**

Grouping the Processes Based on the Communication Hierarchy

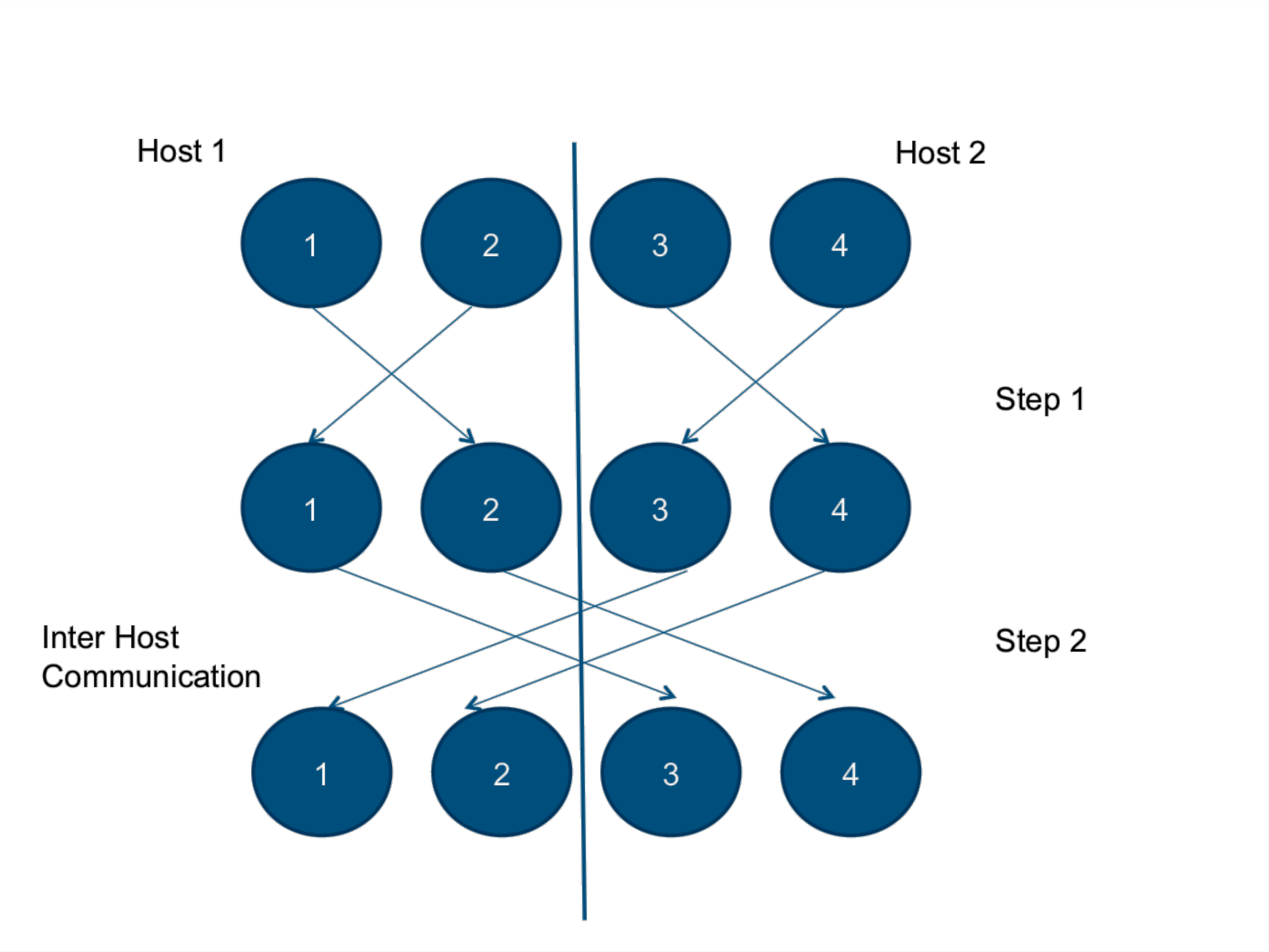
Grouping processes into UMA, Socket, and P2P subgroup



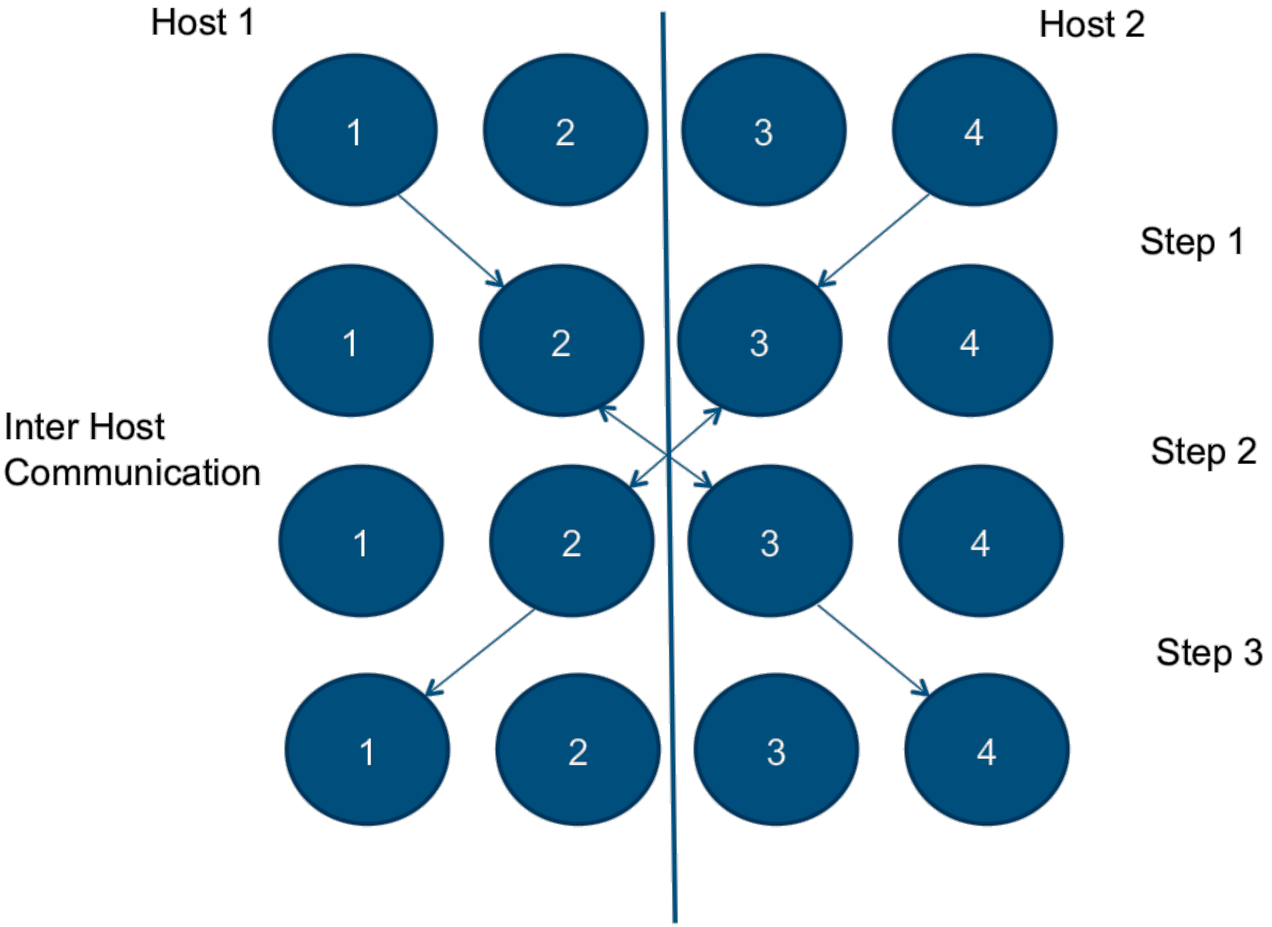
Hierarchical Collectives : n-level Hierarchical Barrier

- n-level Hierarchical Barrier is a combination of fan-in, fan-out, and recursive k’ing barrier (generalization of recursive doubling) primitives
  - processes participating in the collective operation are grouped into n hierarchies based on communication hierarchy
  - processes in top level hierarchy participate in recursive k`ing Barrier primitive
  - processes in n-1 levels participate in fan-in and fan-out primitives

Communication Pattern of a Non-hierarchical Barrier



Communication Pattern of a Hierarchical Barrier



Cheetah’s Hierarchical Broadcast Algorithms

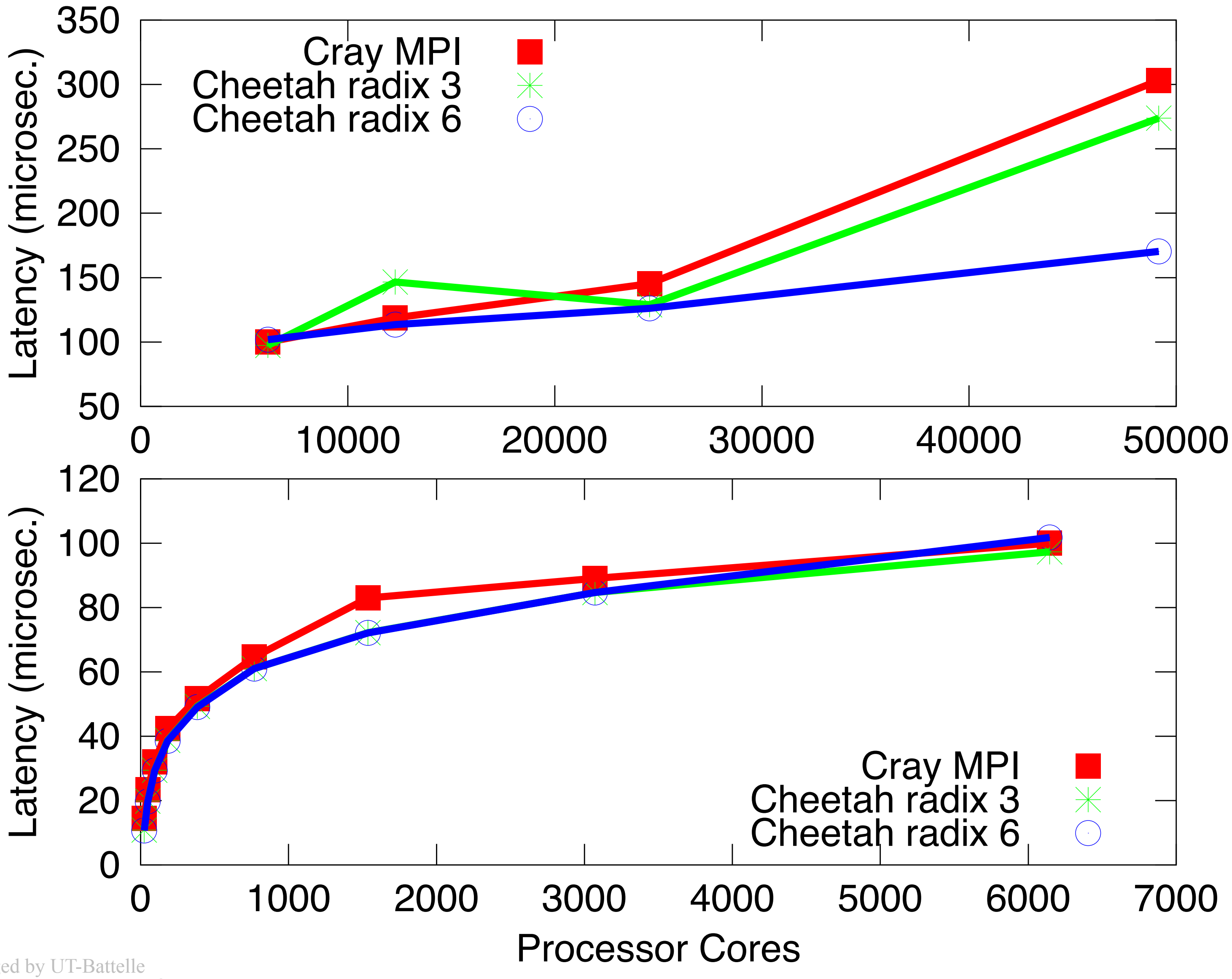
- Knownroot Hierarchical Broadcast
  - the primitives are ordered based on the source of data
  - the primitives are concurrently started after the execution of collective primitive with the source of broadcast
  - uses k-nomial tree for data distribution
- Unknowroot Hierarchical Broadcast
  - the primitives are not ordered and started simultaneously
  - the k-nomial tree for data distribution is built dynamically
- N-ary Hierarchical Broadcast
  - same as Knownroot algorithm but uses N-ary tree for data distribution
- Sequential Hierarchical Broadcast
  - the collective primitives are ordered sequentially
  - there is no concurrent execution

Experimental Setup

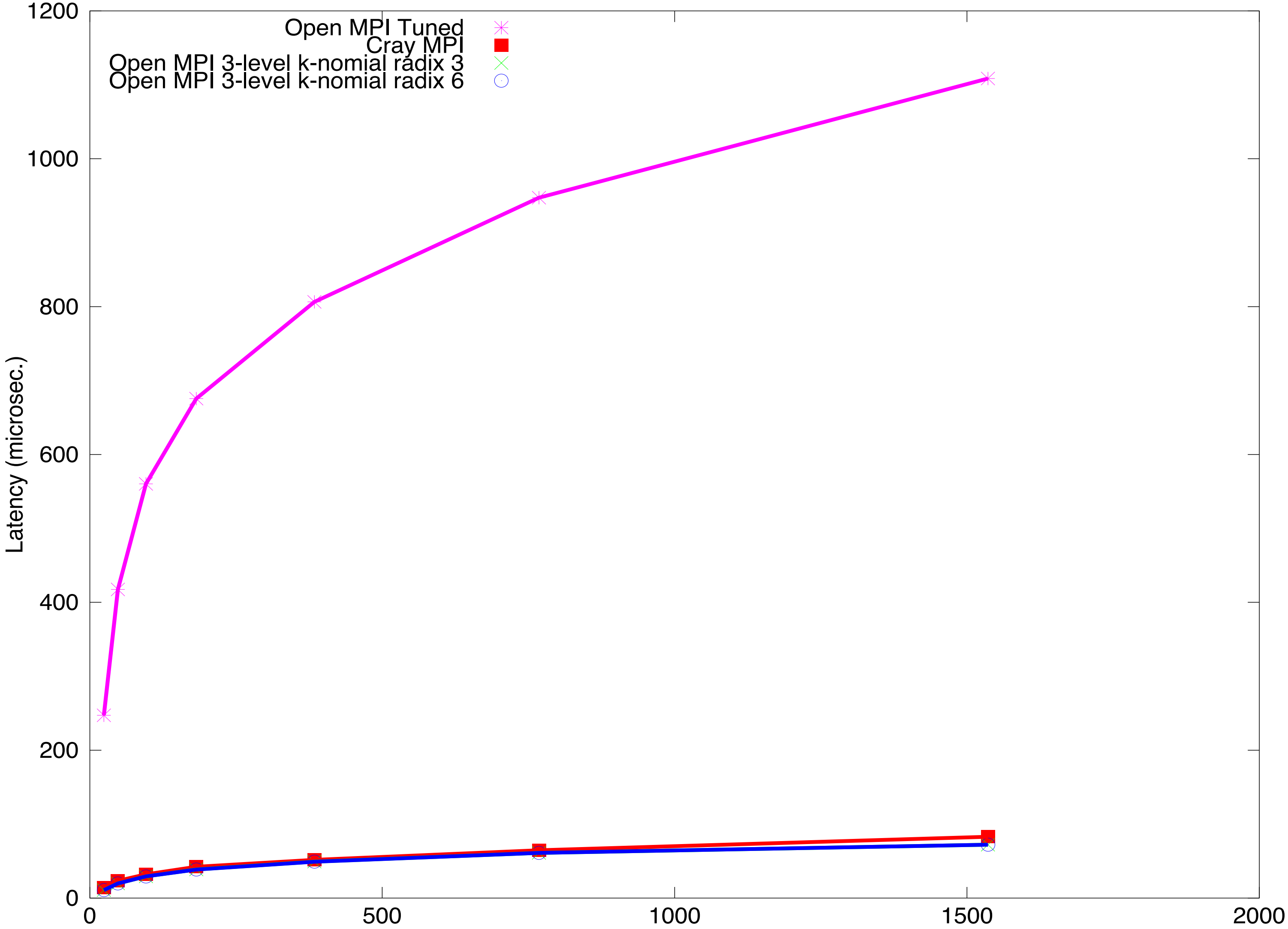
- Hardware :

  Jaguar (a Cray XT 5 supercomputer)
  - 18,688 Compute Nodes
  - 2.6 GHz AMD Opteron (Istanbul)
  - SeaStar 2+ Routers connected in a 3D torus topology
- Benchmarks :
  - Point-to-Point : OSU Latency and Bandwidth
  - Collectives :
    - Barrier in a tight loop
    - Broadcast in a tight loop

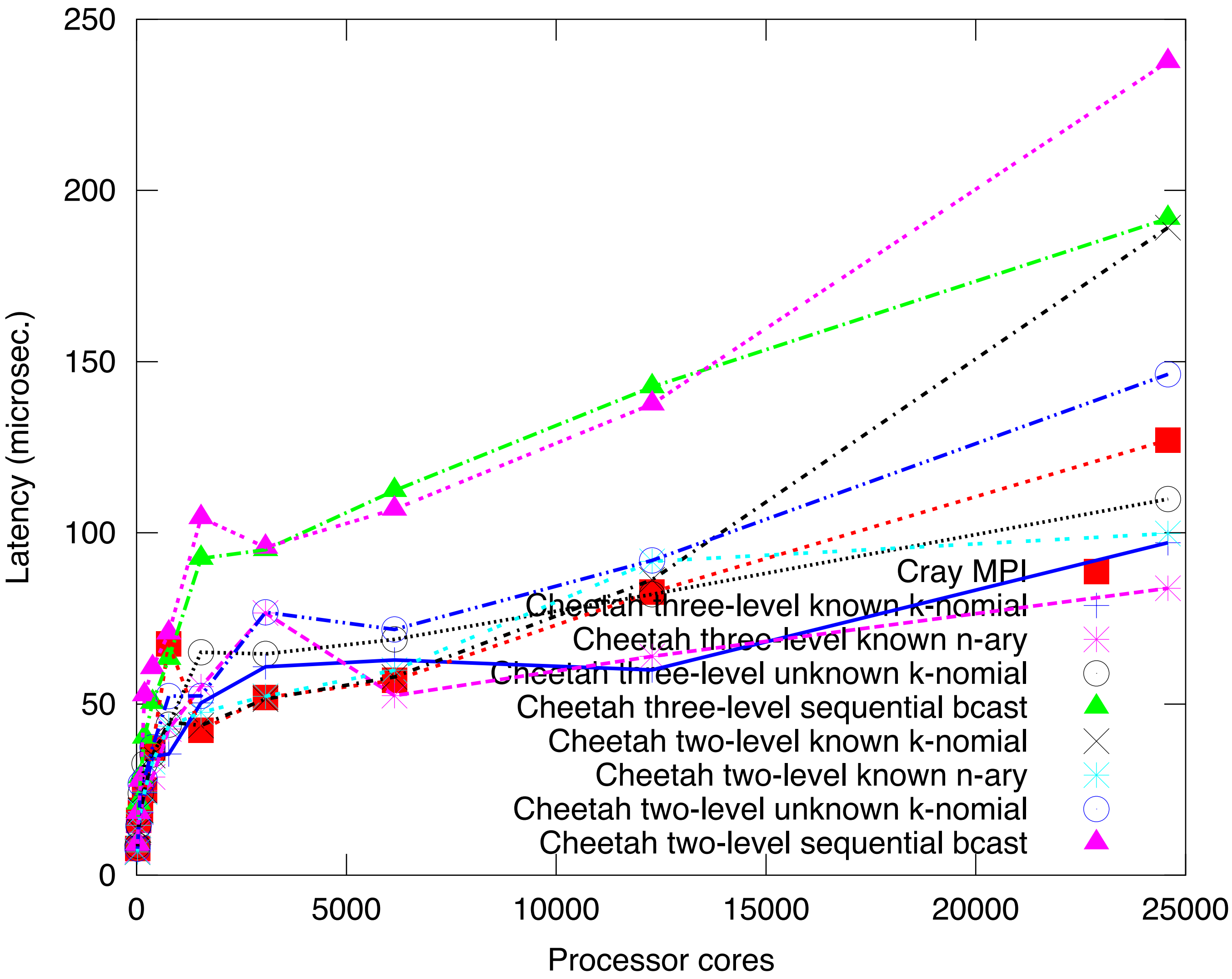
Cheetah's Barrier Collective Outperforms the Cray MPI Barrier by 78%



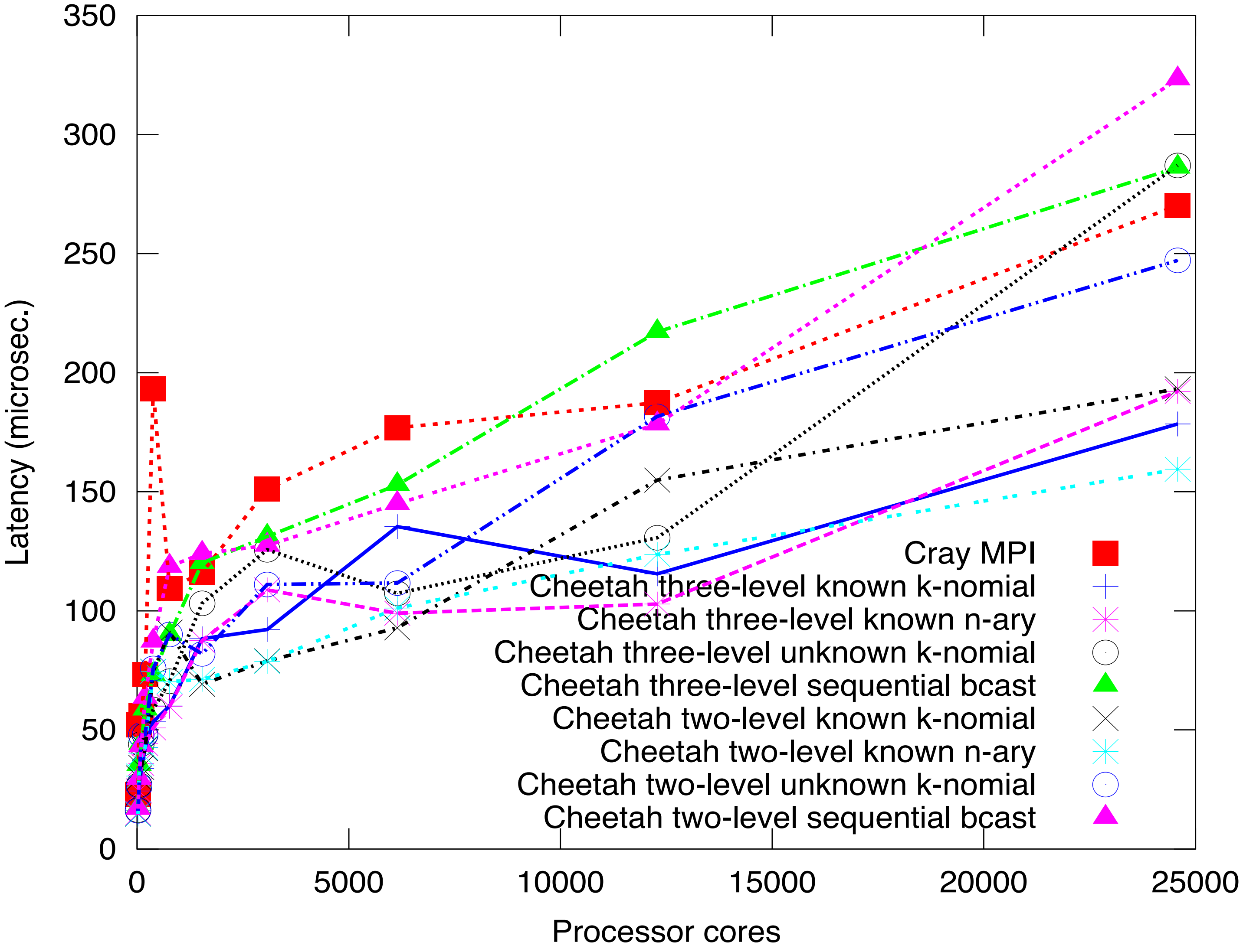
Cheetah's Barrier Collective Outperforms the Open MPI default Barrier



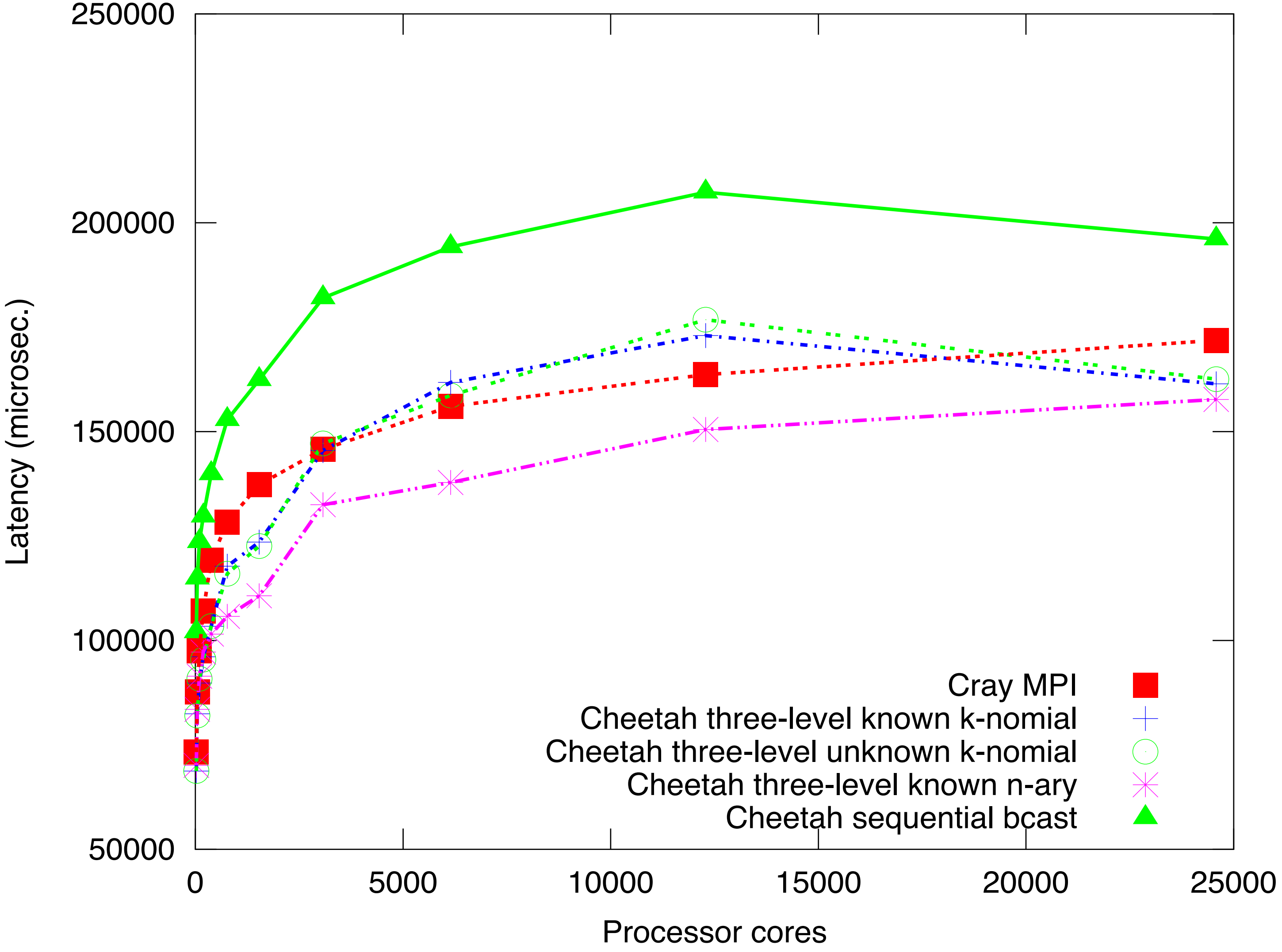
Cheetah's N-ary Broadcast Outperforms the Cray MPI Broadcast by 52% for 1 byte message



Cheetah's N-ary Broadcast Outperforms the Cray MPI by 67% for 2 KB message

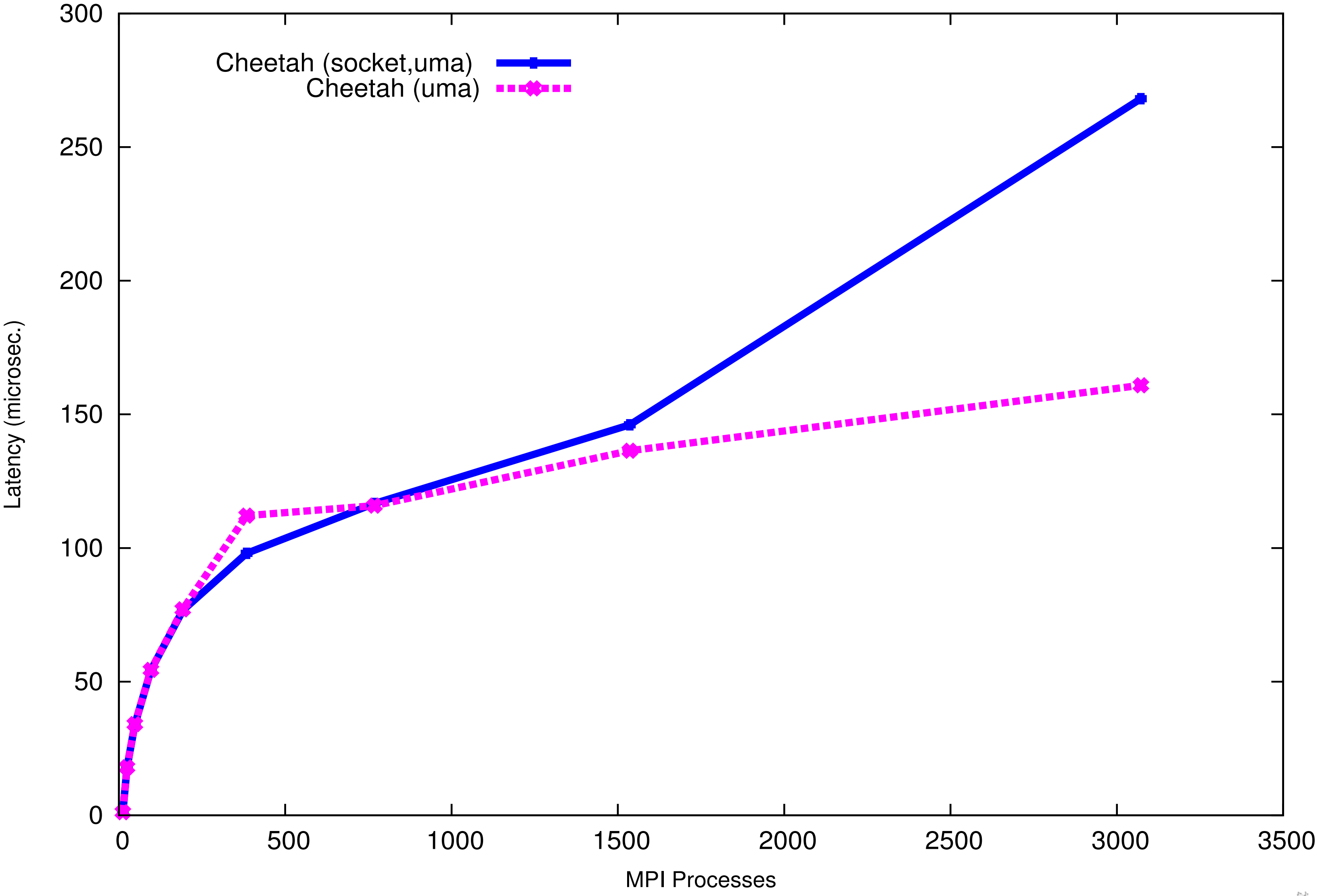


Cheetah's N-ary Broadcast Outperforms the Cray MPI by 8% for 16 MB message

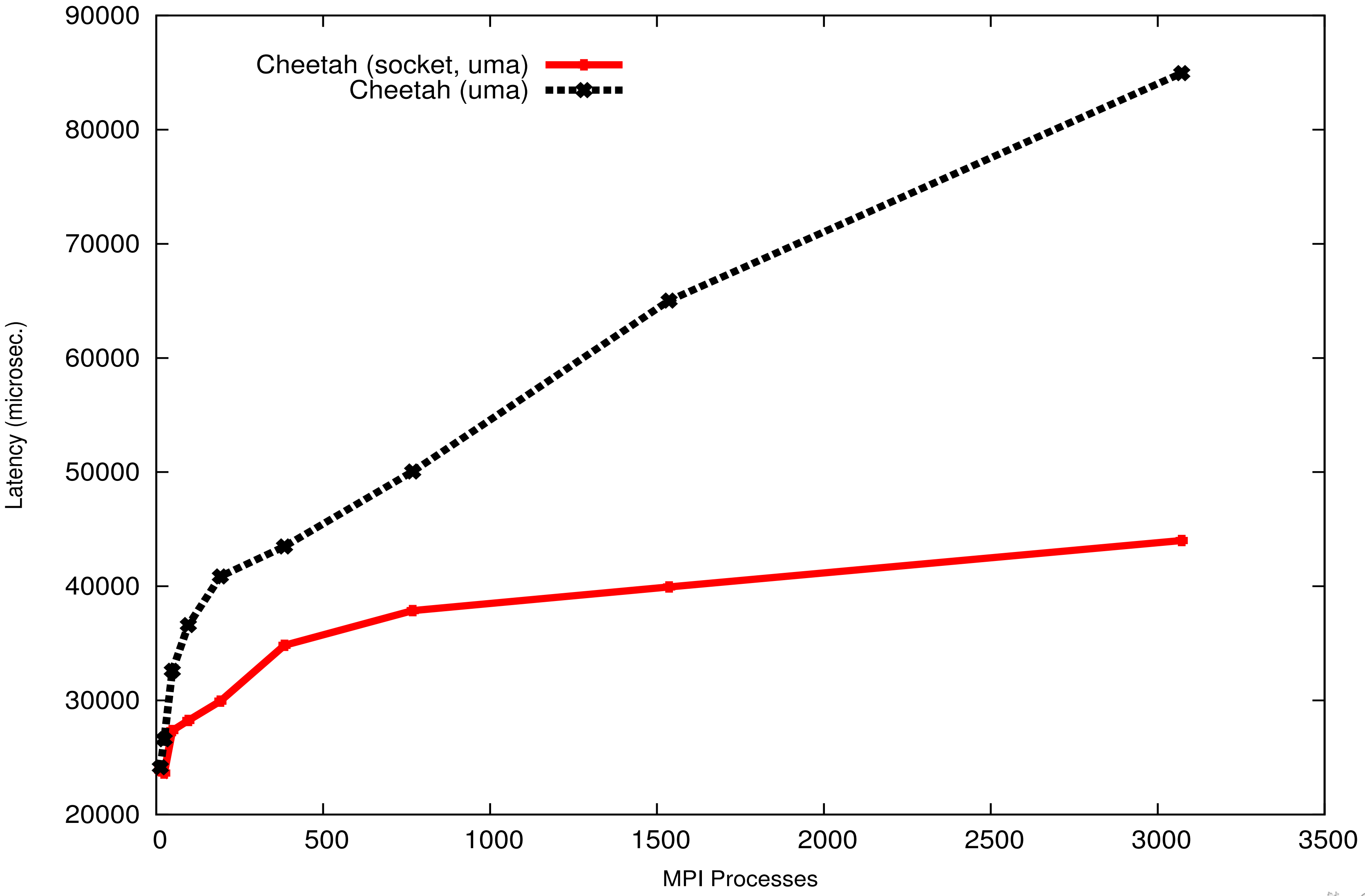


Cheetah’s Collective Performance with Different Levels of Hierarchy

Cheetah's two-level hierarchical Barrier outperforms three-level Barrier

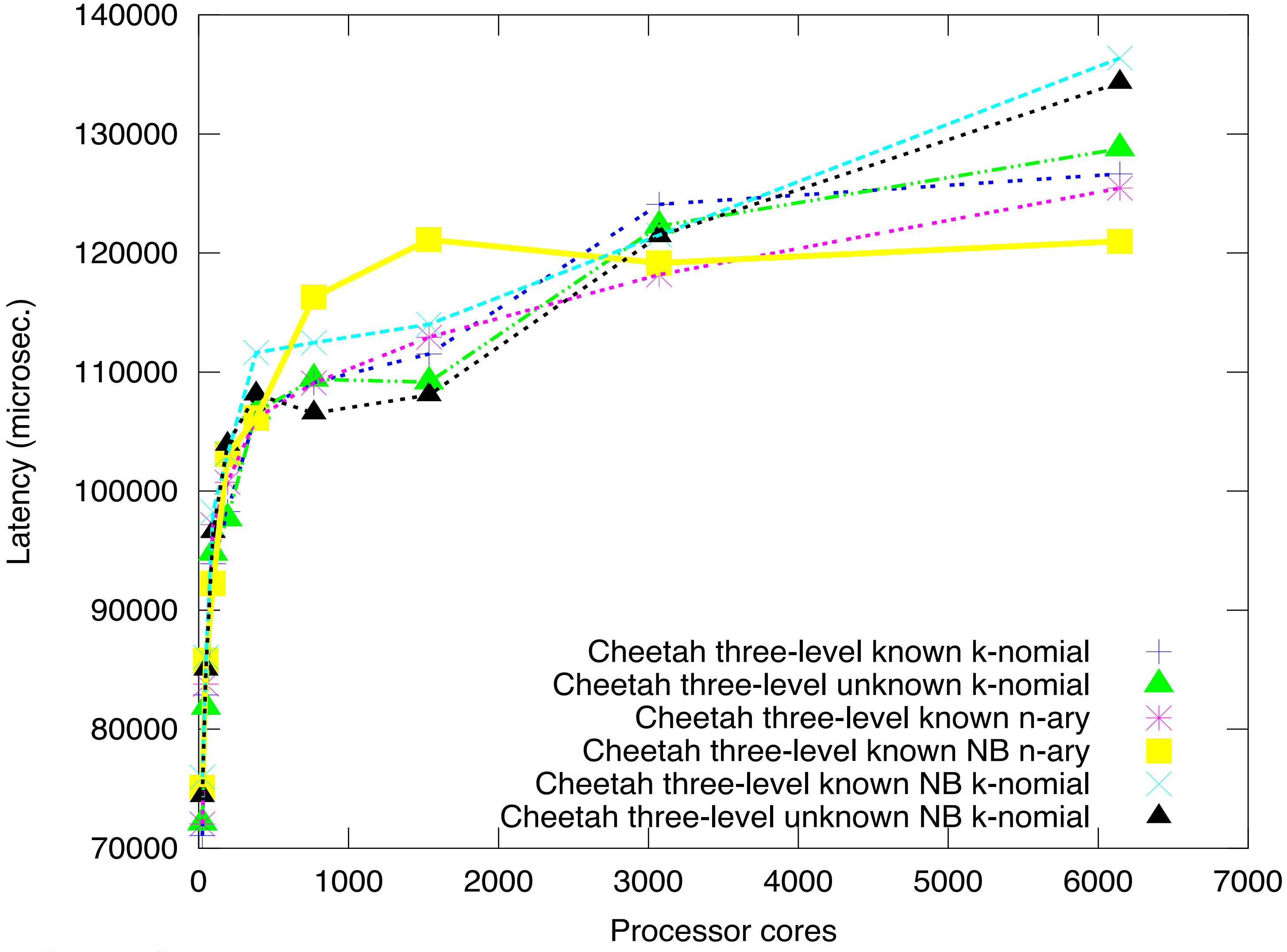


Cheetah's three-level hierarchical Broadcast outperforms two-level Broadcast

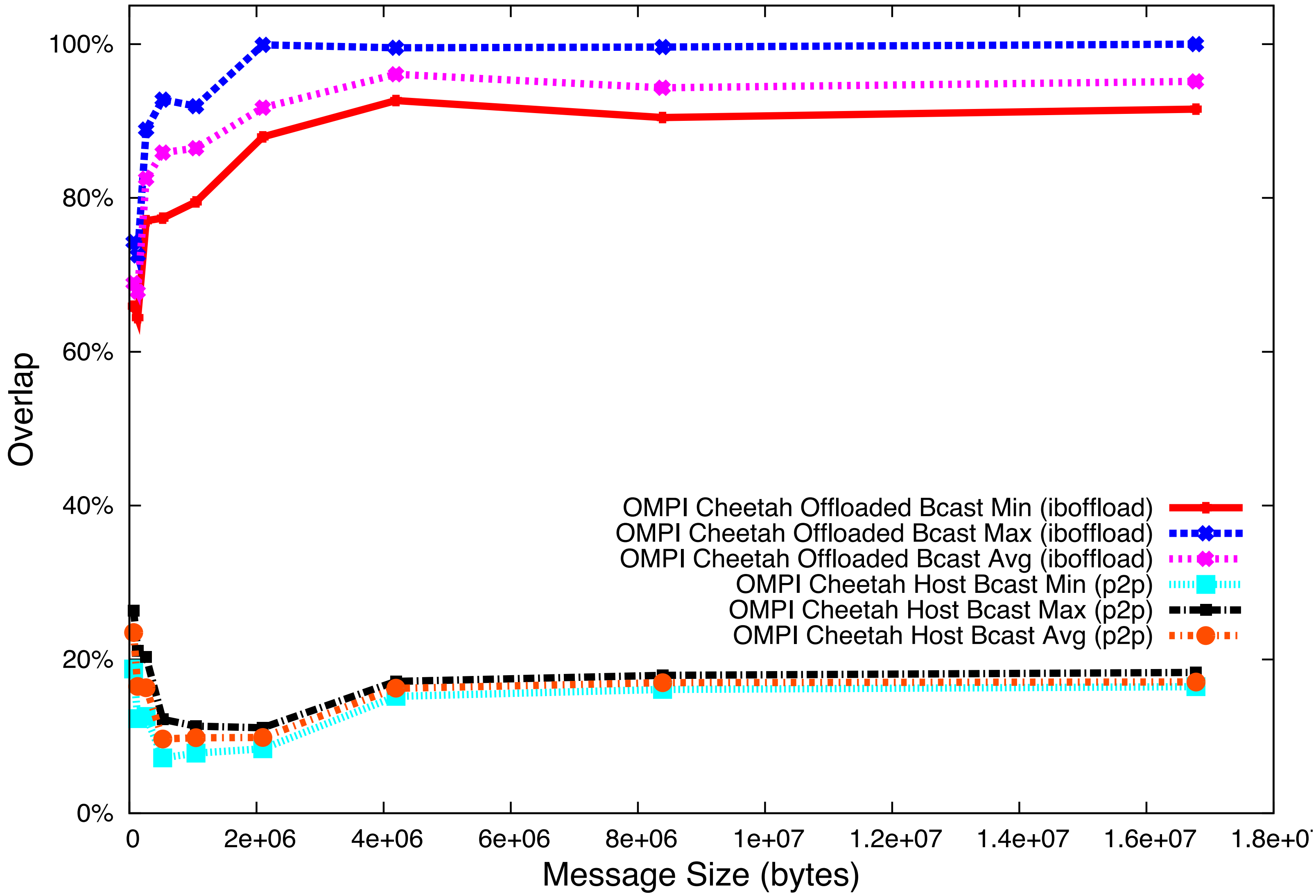


Cheetah's Non-blocking Collective Performance and Overlap

Cheetah's Non-blocking Collectives Performance is Comparable to Blocking Collectives



Cheetah's offloaded Broadcast Collective provides 96% Overlap



Summary

- Cheetah's medium message Broadcast outperforms the Cray MPI Broadcast by 67%
- Cheetah's Barrier outperforms the Cray MPI's Barrier by 78%
- No one-fixed hierarchy configuration suits all collectives
- Cheetah's non-blocking collective performance is similar to its blocking collective performance, and offloaded Cheetah collective provides 96% overlap
- The key to the performance and scalability of the collective operations
  - Collectives customized for communication hierarchies
  - Concurrent execution of sub-operations
  - Scalable resource usage techniques
  - Asynchronous semantics and progress

Contact

Richard L. Graham

Application Performance Tools
Computer Science and Mathematics Division
(865) 356-3469
rlgraham@ornl.gov