Fast and generic concurrent message-passing

Hoang-Vu Dang, Advisor: Prof. Marc Snir
Department of Computer Science, College of Engineering, University of Illinois at Urbana-Champaign

MOTIVATIONS

• Cluster and supercomputer architecture grows increasingly more cores, more heterogeneous
• Data movements become more expensive and cache coherency does not scale for implicit communication
• Growing interest in high-performance networking in non-traditional scientific computing applications: machine-learning, data/graph analytics
• Message-Passing Interface (MPI) is being used, but the performance is not ideal

MPI performance and analysis [EuroMPI’16 (best-paper), CCGrid’17]

Case study and implementation with MPICH 3.1 performance with threads:
• MPI_THREAD_MULTIPLE performs poorly in high thread contentions
• Cooperative scheduling techniques improve latency by 3x
• Advanced lock with unbounded-bias improves message rate by 4x
• Implementations are being incorporated into MPICH [mpich/pal/3068]

Design and implementation of message-passing point-to-point:
• MPI relaxation of wildcard matching
• Efficient low-contention tag-matching using hash-table
• Dedicated communication server minimizes data movement
• User-Level Tasking minimizes thread synchronizations


LCI: generic and low-overhead communication interface [IPDPS’18, PLDI’18]

LCI design principles are to decouple:
• producer-consumer matching: tag, un-tag, one-sided, two-sided
• completion events and progress: completion queue, completion signal
• fatal-error and recoverable errors: retry when recoverable
• high-level, low-level features: maintains simple network primitives

LCI extends the state-of-the-art performance for graph frameworks:
• D-Galois: issues with flow-control and data management
• Gluon: issues with heterogeneity in computing architecture for graphs

[IPDPS’18] Hoang-Vu Dang, Roshan Datathri, Gurinder Gill, Alex Brooks, Niko Dryden, Andrew Lenhart, Loc Hoang, Keshav Pingali, and Marc Snir: “A lightweight communication runtime for distributed graph analytics.”

CONCLUSIONS

• MPI performance is lagging behind due to the changes in architecture and usages
• Performance of message-passing can be improve with better concurrent data structures and some relaxations in semantics.
• LCI represents a clean ground-up design, highly integrated with threads
• FULT is a technique for scalable communication synchronization.
• Future work: a standard API for LCI, release of FULT software package, integration to OpenMP

THREAT: Fast synchronizations for communication [ICPP’18, ESPM2’15]

Signal/Wait for scheduled/de-scheduling tasks on distributed events:
• Communication server receives messages and signal a waiting thread
• Optimizing signal/wait improves performance of communication with high number of threads

FULT is a Fast User-Level Threading scheduling technique:
• Each work queues of a worker is a bit-vector
• Hierarchical bit-vectors for millions of tasks per node
• Load-balancing using work-stealing, highly scalable synchronizations
• Performance improvement up to 6x vs Argobots and Qthreads.

[ICPP’18] Hoang-Vu Dang, and Marc Snir, “FULT: A Fast User-Level Thread Scheduling using bitvectors”