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**
- Clusters and supercomputers have increasing core numbers and are more heterogeneous
- Explicit data movement becomes more important to performance
- There is growing interest in high-performance for non-traditional scientific applications: machine-learning, data/graph analytics
- Message-Passing Interface (MPI) is being used, but the performance is not ideal – especially with high thread counts

**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 with high thread contention
  - Cooperative scheduling techniques improve latency by 3x
  - Advanced lock with unbounded-bias improves message rate by 4x
- Implementations are being incorporated into MPICH [mpich/pull/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 facing primitives
- LCI improves the state-of-the-art performance for graph frameworks
  - D-Galois: deals with issues with flow-control and data management
  - Gluon: deals with issues with heterogeneity in computing architecture

**CONCLUSIONS**
- MPI performance is lagging behind due to the changes in architecture and usage patterns
- Performance of message-passing can be improved with better data structures and relaxation in semantics
- LCI represents a clean ground-up design, very low-overhead and highly integrated with threads
- FULT is a thread scheduling technique and library for scalable communication synchronization
- Future work: a standard LCI API, new microbenchmarks, integration MPI + OpenMP

**CONTRIBUTIONS**
- Study and evaluation of MPI semantics and performance for emerging applications and architectures
- Design and Implementation of LCI, a low-level and efficient communication interface targeting multi-threaded, event-driven, heterogeneous frameworks
- Development of new thread synchronization and scheduling techniques for efficient inter-operation between threads and communication runtimes

**FULT/PPL: Fast synchronizations for communication [ICPP’18, ESPM’15]**
- Schedule/de-scheduling tasks quickly is needed for distributed events:
  - Communication server receives messages and signals waiting threads
  - Signal/wait performance is critical for the performance of communication with large number of threads.

**FULT** is a Fast User-Level Threading scheduling technique:
- Each work queue 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 upto 6x vs Argobots and Qthreads.

**ACKNOWLEDGEMENTS**

**CONTACTS AND LINKS**

Hoang-Vu Dang: hdang8@illinois.edu
LCI: https://github.com/danghvu/LCI
UIUC-HPC: https://github.com/uisc-hpc
D-Galois: http://iss.ices.utexas.edu/?p=projects/galois

---

**Graphs and Diagrams**

**Captions set in a serif style font such as Times, 18 to 24 size, italic style.**

---

**CONTACTS AND LINKS**

Hoang-Vu Dang: hdang8@illinois.edu
LCI: https://github.com/danghvu/LCI
UIUC-HPC: https://github.com/uisc-hpc
D-Galois: http://iss.ices.utexas.edu/?p=projects/galois