Runtime Support for Irregular Computation in MPI-Based Applications

Xin Zhao*, Pavan Balaji† (Co-advisor) and William Gropp* (Advisor)

*University of Illinois at Urbana-Champaign, Champaign, IL, USA (xinzhao3, wgropp)@illinois.edu
†Argonne National Laboratory, Argonne, IL, USA balaji@mcs.anl.gov

Plan of Study

• Addressing scalability and performance limitations in massive asynchronous communication
• Tackling scalability challenges in MPI runtime
• Optimizing MPI runtime for fine-grained operations

Research Goal

• Study MPI’s suitability to irregular applications
• Find out where MPI lies on the spectrum of suitability
• Propose what if anything needs to change in MPI runtime and semantics to efficiently support irregular applications

Irregular Computations

• Some of them are organized around sparse data structures
• Large amount of outstanding asynchronous messages involved
• Data movement pattern is irregular and data-driven
• Increasing trend of applications migrated from regular computation model to irregular computation model, due to reasons of computation complexity, power consumption, etc.

Addressing Scalability and Performance Limitations in Massive Asynchronous Communication

• Virtual connection (VC) objects
• Problem: use O(P) memory per process to store VCs
• Solution: “Lazzy” initialization: initialize corresponding VCs when communicating with peers
• “On-demand” initialization: maintain fixed number of VCs that are recently used
• RMA window information storage
• Problem: use O(P) memory per process to store window information
• Solution:

Integrated Data and Computation Management

• Correctness semantics
• Interoperability: in SEPARATE window model, if AM and RMA, or AM and AM, or AM and local STORE, update on the same window, either on overlapping or on non-overlapping locations, result is undefined
• Memory consistency: MPI runtime must ensure memory consistency of RMA window
• Ordering: three orderings are imposed by default — AMs with different operations; AMs with the same operation; segments within AM
• Concurrency: by default, MPI runtime behaves “as if” AMs are executed in sequential order

Graph500 benchmark

128 (1000 nodes), 256 (1000 nodes), 512 (1000 nodes)

**Publications**


**Acknowledgements**

This material was based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Contracts DE-AC02-06CH11357, DE-AC02-06CH11357, DE-SC0004131.