Optimizing Collective Communication for Petascale Supercomputers

http://upc.lbl.gov

Introduction

Collective communication
- Cooperative data-movement beyond one-to-one communication
- Common building blocks for many applications
- Key bottleneck of performance scalability

GASNet
- Portable high-performance communication primitives
- Used to implement partitioned global address space languages: e.g., UPC, Titanium, Co-array FORTRAN, and Chapel

Example Communication Topologies

Radix 2 k-nomial tree (binomial)

Binary Tree

Fork Tree

Radix 2 Dissemination

Collectives for PGAS Languages

Teams
- Thread-centric: Programmer explicitly specifies the threads that take part in the collective through a language level team construction API.
- Data-centric: Programmer only specifies the data for the collective. Runtime system then figures out where the data resides and performs the collective.

Synchronization modes
- Loose: Data movement can start and as soon as first thread enters collective and continue until last thread leaves the collective.
- Middle: Data movement into and out of local memory can occur only when the data-owner thread is in the collective operation.
- Strict: Data movement can start only after all threads have entered the collective and must finish before any thread leaves the collective.

Optimizations
- Non-blocking collective operations that facilitate overlapping communication and computation
- Network-specific optimizations for leveraging hardware features
- Automated performance tuning for accommodating different application characteristics on multiple platforms

Performance Auto-tuning

Offline tuning
- Optimize for platform common characteristics
- Minimize runtime tuning overhead

Online tuning
- Optimize for application runtime characteristics
- Refine offline tuning results

Micro-benchmarks

Applications

Conclusion

High Productivity
- Portable performance from multi-core PCs to petascale supercomputers
- Compact and clean UPC code

Scalable Performance
- 3-D FFT (communication intensive)
  - Weak scaling: 38% over MPI (16K cores)
  - strong scaling: 20% over MPI (16K cores)
- Numerical linear algebra: highly scalable performance up to 2X MPI/PBLAS