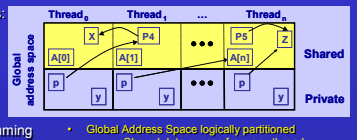
Optimized Collectives for PGAS Languages with One-Sided Communication

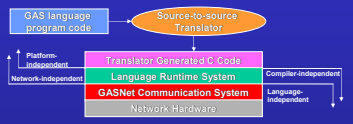
Dan Bonachea, Rajesh Nishtala, Paul Hargrove, Mike Welcome, Kathy Yelick

Partitioned Global Address Space Languages

- Partitioned Global Address Space (PGAS) Languages:
  - Global pointers and distributed arrays
  - User controlled layout of data across nodes
  - Communicate using implicit reads & writes of remote memory
- Languages: UPC, Titanium, Co-Array Fortran
  - Productivity benefits of shared-memory programming
  - Competitive performance on distributed memory
  - See PGAS Booth #342
- Use Single Program Multiple Data (SPMD) control
  - Fixed number of compute threads
  - Global synchronization, barriers, collectives
- Exploit fast one-sided communication
  - Individual accesses and bulk copies
  - Berkeley implementations use GASNet



- Global Address Space logically partitioned
  - Shared data access from any thread
  - Private data local to each thread
- Pointers are either local or global (possibly remote)

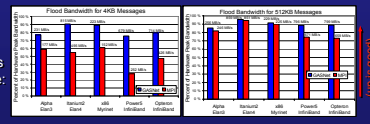


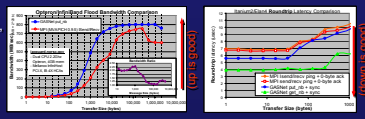
GASNet Portability

- Native network hardware support:
  - Quadrics QsNet I/II (Elan3/Elan4)
  - Cray X1 - Cray shmem
  - SGI Altix - SGI shmem
  - Cray XT3 - Cray Portals (new!)
  - Dolphin - SCI
  - InfiniBand - Mellanox VAPI
  - Myricom Myrinet - GM-1 and GM-2
  - IBM Colony and Federation - LAPI
- Portable network support:
  - Ethernet - UDP: works with any TCP/IP
  - MPI 1: portable impl. for other HPC systems
- Berkeley UPC, Titanium & GASNet highly portable:
  - Runtimes and generated code all ANSI C
    - New platform ports in 2-3 days
    - New network hardware 2-3 weeks
  - CPU's: x86, Itanium, Opteron, Athlon, Alpha, PowerPC, MIPS, PA-RISC, SPARC, T3E, X-1, SX-6, …
  - OS's: Linux, FreeBSD, NetBSD, Tru64, AIX, IRIX, HPUX, Solaris, MS-Windows/Cygwin, Mac OSX, Unicos, SuperUX, Catamount, BlueGene, …

High Performance

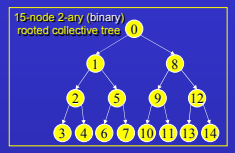
- Communication micro-benchmarks show GASNet consistently matches or outperforms MPI
- One-sided, lightweight communication semantics
  - eliminates matching & rendezvous overheads
  - better match to hardware capabilities
  - leverages widely-available support for remote direct memory access
  - Active Messages support provides extensibility

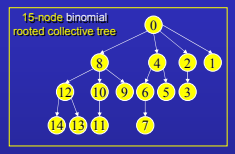


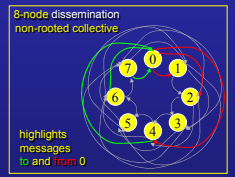


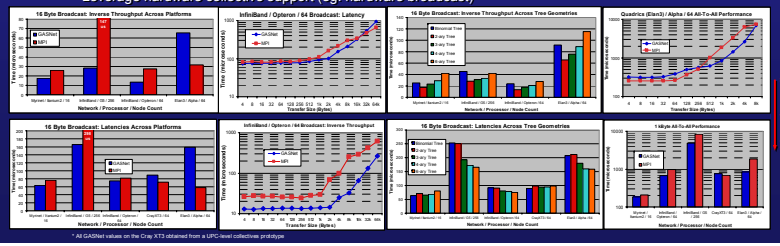
Optimized Collectives in GASNet

- **Collective interface specifically designed for PGAS Languages**
  - Data movement: **Broadcast, Scatter, Gather, Gather-All, Transpose**
  - Computational: **Reduce, Prefix-Reduce**
  - Superset of collective support in UPC and Titanium languages
    - Extensible to variable-contribution and teams-based subset collectives
  - Achieves performance not obtainable from language-level implementations
- **Interface includes many collectives features not found in MPI-2:**
  - Fully non-blocking collectives
    - Allows overlap of latency with computation and other communication
  - Exploit global address knowledge when available
    - Allows RDMA-based impl. - no rendezvous or eager buffering costs
  - Explicit consistency flags for detailed control over data sync. enforcement
    - Sync-free collectives: data not produced/consumed in current phase
    - Per-thread sync: data has affinity to producer or consumer (MPI style)
    - Global sync: barrier-like data sync (more efficient than full barrier)
- **Implementation features**
  - **Collective geometry cache**
    - Communication geometries lazily built and reused
    - Extensible infrastructure to allow for future auto-tuning features
  - **Distributed scratch space management**
    - Active message based to avoid over synchronization
    - Supports multiple overlapped collectives of any variety
  - **Enables network-specific tuning & optimization**
    - Algorithm selection based on hardware characteristics & network state
    - Eventually enable auto-tuning – find best algorithm & params
    - Leverage hardware collective support (eg. hardware broadcast)









* All GASNet values on the Cray XT3 obtained from a UPC-level collectives prototype

GASNet on the Cray XT3

- GASNet Put/Get operations implemented over Portals Put/Get
  - Remote access region covered by Portals Memory Descriptor
  - Portals Events used for GASNet operation completion
  - Put/Get injection throttled to prevent local event queue overflow
  - No remote event generation
  - Local Put source and Get destination regions:
    - copied through pre-pinned bounce buffers for small messages
    - pin/un-pin of local region for messages >= 1 KB
- GASNet Active Message layer currently prototyped over MPI
  - Port to native Portals-based AM is underway
- UPC LU Application:
  - Compliant with but less than 1/2 code size of MPI-based HPL
  - System tuned DGEMM for high floating point performance
  - Novel multi-threading (user-space co-routines) that yield on long latency communication, using a highly asynchronous algorithm
- UPC FT benchmark:
  - Each thread uses non-blocking Put to send message as soon as local FFT complete - overlap computation with communication
- Bulk, slab-based and pencil-based implementations - Slab best on XT3