GASNet 2

An Alternative High-Performance Communication Interface

U.C. Berkeley and LBNL

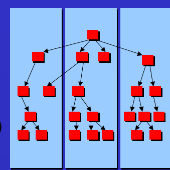
Christian Bell, Dan Bonachea, Wei Chen, Jason Duell, Paul Hargrove, Parry Husbands, Costin Iancu, Wei Tu, Mike Welcome, Kathy Yelick

http://upc.lbl.gov

upc@lbl.gov

Global Address Space Languages

- Global address space languages support:
  - Global pointers and distributed arrays
  - User controlled layout of data across nodes
  - Implicit reads & writes of remote memory (get & put)
- Single Program Multiple Data (SPMD) control
  - Similar to using threads, but with remote accesses
  - Global synchronization, barriers
- Languages: UPC, Titanium, Co-Array Fortran
- GASNet - A common communication system tailored for global address space languages



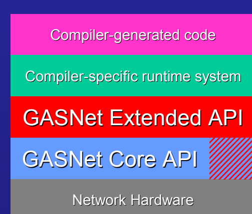
Distributed Data Structures

Supported Network Hardware

- High-performance network hardware support:
  - Quadrics QsNet I (Elan3) and QsNet II (Elan4) new!
  - Cray X1 - Cray shmem new!
  - SGI Altix - SGI shmem new!
  - Dolphin - SCI new! (work by Univ. of Florida - Su&Gordon)
  - 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 stack new!
  - MPI 1.1: portable implementation for other HPC systems

GASNet Core API

- Provides most basic required network primitives
- Implemented directly on each platform
  - Minimal set of network functions needed to support a working implementation
  - General enough to implement everything else
- Based on Active Messages, a lightweight RPC paradigm
  - Provides powerful extensibility mechanism
- Includes platform-independent job bootstrap & teardown



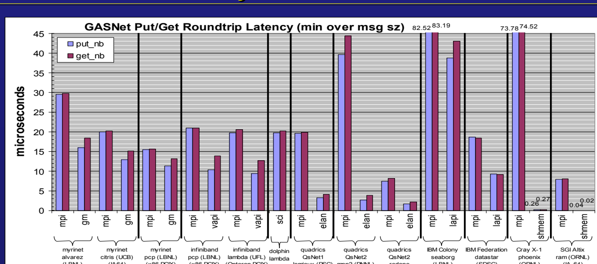
GASNet Goals

- **Language-independence**: support various GAS languages and compilers
  - UPC, Titanium, Co-array Fortran, possibly others..
  - Provide generic high-performance support for implementing GAS langs
  - Runtime system client provides language- or compiler-specific details, such as shared-pointer representation and memory allocation
- **Hardware-independence**: support a variety of parallel architectures & systems
  - CPU / architecture independence:
    - Clusters of uniprocessors or SMPs, integrated supercomputers
    - x86, Itanium, Opteron, Athlon, Alpha, PowerPC, MIPS, PA-RISC, SPARC, T3E, X-1, SX-6, …
  - OS / system software independence:
    - Implemented in ISO C, standard GNU configure toolset
    - OS's: Linux, FreeBSD, NetBSD, Tru64, AIX, IRIX, HPUX, Solaris, MS-Windows/Cygwin, Mac OSX, Unicos, SuperUX, …
    - Compilers: GCC, Portland Group C, Intel C, SunPro C, Compaq C, HP C, MIPSPro C, IBM VisualAge C, Cray C, NEC C, …
- **Ease of implementation on new hardware**
  - Infrastructure framework allows quick prototype implementations
  - Implementations can leverage performance features of hardware
- **Provide both portability & high performance**

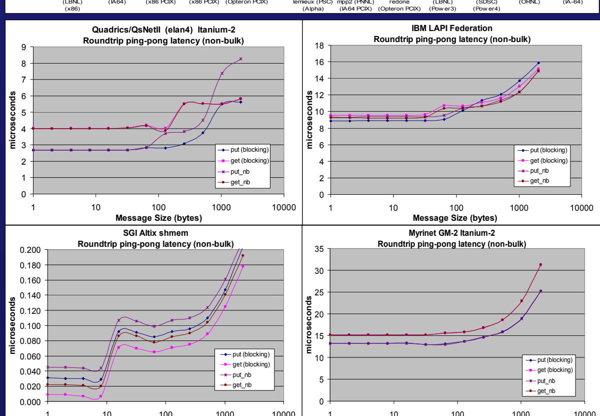
GASNet Extended API

- Wider interface that includes more complicated operations
  - puts and gets, split-phase barriers, collective operations, etc
- Semantics carefully chosen to perform well on modern hardware
  - Fully one-sided and non-blocking put & gets (often use zero-copy RDMA)
  - No tag matching, no ordering constraints, decouple data motion & sync
  - Delivers hardware peak bandwidth for large messages AND ultra-low latency/overhead for tiny (eg 8 byte) messages
- Semantics designed for parallel compiler code generation
  - Many flexible sync. mechanisms for non-blocking ops
  - Access to full remote virtual memory - no "pre-registration"
- Provide a reference implementation of the extended API in terms of the core API - upgrade path for quick prototyping
- Implementors can choose to directly implement any subset for performance - leverage hardware support for higher-level ops

Latency Performance

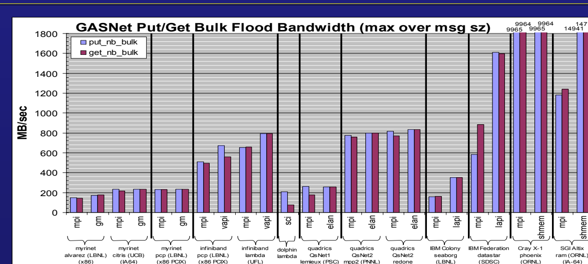


(down is good)

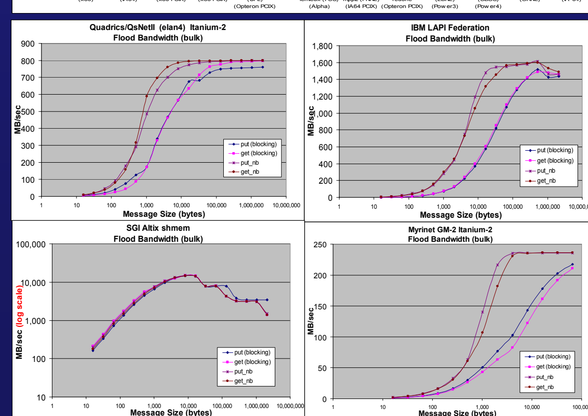


(down is good)

Bandwidth Performance



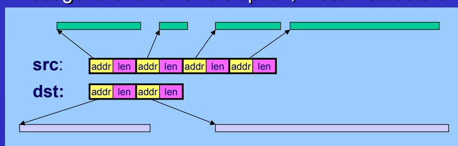
(up is good)



(up is good)

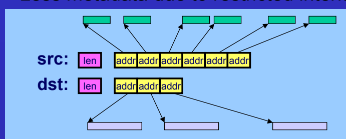
GASNet 2: Non-contiguous Accesses

- **Point-to-point non-contiguous put/get operations**
  - Allow message aggregation optimizations in the application and compiler
    - Transform fine-grained access patterns into bulk messages
    - Use available hardware support for offloading pack/unpack overheads
  - Leverage available network hardware support for scatter/gather RDMA
    - Expose them with a common interface for libraries & compilers
    - All fully non-blocking with flexible synchronization
- **Vector**: List of variable-length contiguous regions
  - Most general and flexible option, most metadata overhead



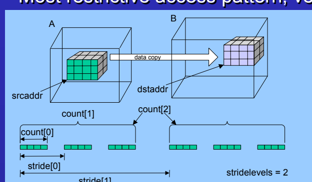
Useful for transferring bounding boxes or sparse array data, message coalescing

- **Indexed**: List of fixed-length contiguous regions
  - Less metadata due to restricted interface, better hardware support



Useful for transferring irregular set of array elements, inspector/executor optimizations, software pipelining

- **Strided**: Arbitrary rectangular section on an N-d dense array, for any N
  - Most restrictive access pattern, very little metadata overhead

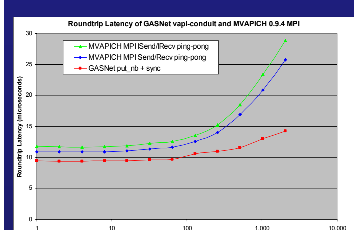


Useful for ghost value exchanges, supporting multi-D array libraries, transferring any rectangular section of a dense array

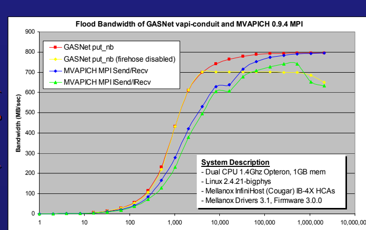
- Current status: Reference implementation avail for all networks using put & get
  - Implementation underway using GASNet Active Messages
    - Use AM operations to pack/unpack data, automatic algorithm selection
  - Implementations underway using native hardware/network support
    - Eg. Quadrics/Elan4 putv/getv, InfiniBand gather-send/scatter-recv, …

GASNet on InfiniBand

- Targets Mellanox VAPI interface
  - Vendor implementation of the InfiniBand Verbs w/minor extensions
- GASNet Core API: Active Messages
  - Based on Send/Recv operations, simple flow control
  - Uses an additional thread for improved responsiveness
- GASNet Extended API: puts and gets
  - Very thin, efficient layer over InfiniBand RDMA puts & gets
  - Simple record attached to each CQE for completion
  - Firehose provides dynamic memory registration
- Consistently outperforms MPI-over-InfiniBand
  - GASNet interface eliminates tag-matching & rendezvous overheads



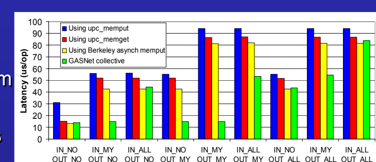
(down is good)



(up is good)

GASNet 2: Collective Operations

- **Collective interface specifically designed for GAS 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
- **Design includes many collectives features not found in MPI-2:**
  - Fully non-blocking collectives
    - Well defined and deadlock-free semantics, even with teams
  - Exploit global address knowledge when available
    - Allows RDMA-based impl. with no rendezvous and no eager buffering costs
  - Explicit consistency flags provide 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)
  - Aggregation hints allow coalescing of messages & synchronization
    - For any set of consecutive collective calls, even of different varieties
  - SMP-friendly threading-aware interface
    - Runtime libraries can eliminate copies by exposing thread layout information
- **Reference implementation of data movement collectives**
  - Supports all GASNet networks & platforms
  - Includes hooks for network-specific optimizations and tuning
- **Work in Progress**
  - Completing specification & design of computational collectives
  - Improvements to data movement collectives
    - Applying a variety of tree-based algorithms
    - Implementing RDMA-based synchronization algorithms
  - Network-specific tuning & optimization
    - Algorithm selection based on hardware perf. characteristics & network state
    - Leverage hardware collective support (eg. Quadrics hardware broadcast)
  - Collectives research
    - Platform for experimenting with new collective ops proposed for UPC & Titanium
    - Automatic perf tuning
      - Find best algrthm/params
      - Online at runtime
      - ATLAS-style offline

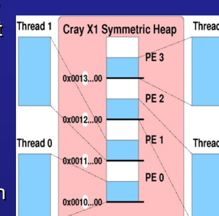


(down is good)

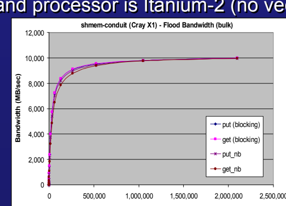
Small Broadcast Performance w/ varying sync mode
Point-to-Point UPC vs. Direct GASNet collective

GASNet on Cray X1 / shmem

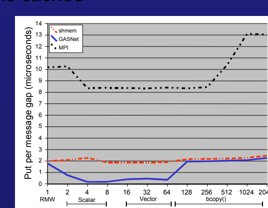
- Uses shmem for implementing core API Active Messages
- Uses hardware's native global memory support for put/get
  - Outperforms both MPI & shmem for small messages
  - Operates directly on hardware global pointers
- Neither MPI nor shmem can fully exploit the hardware capabilities for fine-grained communication on the X1
  - Library interfaces prevent crucial vectorization
  - gasnet_put/get fully inlined - allows caller vectorization
- shmem-conduit also supports SGI Altix
  - similar global system, but remote memory is cached and processor is Itanium-2 (no vectors)



Cray X1 memory centrifuge



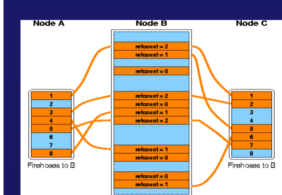
(up is good)



(down is good)

shmem-conduit/X1 small put performance
(get performance similar)

Firehose Memory Registration Library



Firehose Algorithm
for distributed management
of DMA registration

- Ideal memory registration strategy for global address space languages on pinning-based network hardware (eg Myrinet, Infiniband, Dolphin)
  - C. Bell and D. Bonachea. "*A New DMA Registration Strategy for Pinning-Based High Performance Networks*" CAC 2003
- Exposes one-sided/zero-copy RDMA over **entire VM** as common case
  - Common-case performance of *Pin-Everything* (without drawbacks)
  - Degrades to *Rendezvous*-like behavior for the uncommon case
- Amortizes cost of registration/synch over many operations, using temporal/spatial access locality to avoid repinning costs
- Cost of handshaking and registration negligible when working set fits in physical memory, degrades gracefully beyond
- Shares registration state between threads on SMP to maximize hit rate
  - Fast optimistic concurrency control protocol between threads

Approach	Zero-copy	One-sided	Full VM avail	Description, Pros and Cons
Hardware-based (eg.Quadrics)	✓	✓	✓	Hardware manages everything No handshaking or bookkeeping in software Hardware complexity and price, Kernel modifications
Pin Everything	✓	✓	✗	Pin all pages at startup or when allocated (collectively) Total usage limited to physical memory May require a custom allocator
Bounce Buffers	✗	✗	✓	Stream data through prepinned bufs on one/both sides Mem copy costs (CPU consumption, cache pollution, prevents comm. & computation overlap) Messaging overhead (metadata & handshaking)
Rendezvous	✓	✗	✓	Round-trip message to pin remote pages before each op Registration costs paid on every operation
Firehose	✓ common case	✓ common case	✓	Common case: All the benefits of hardware-based Uncommon case: Messaging overhead (metadata & handshaking)

Survey of Approaches to Memory Registration for HPC NICs