**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

**Supported Network Hardware**

- High-performance network hardware support:
  - Quadrics QsNet I (Elan3) and QsNet II (Elan4) new!
  - Cray X1 - Gray shmem new!
  - SGI Altix - SGI shmem new!
  - Dolphin - SCI new! (work by Univ. of Florida - Sub&Cordon)
  - 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 unprocessors 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, Solaris, HP-UX, Windows, Cygwin, Mac OSX, Unices, SuperUX, ...
- **Ease of implementation on new hardware**
  - Infrastructure framework allows quick prototype implementations
  - Implementors can choose to directly implement any subset for performance - leverage hardware support for higher-level ops

**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

**Latency Performance**

- GASNet Put/Get Roundtrip Latency (min over msg sz)
- GASNet Core API
- GASNet Extended API
- Network Hardware

**Bandwidth Performance**

- GASNet Put/Get Bulk Flood Bandwidth (max over msg sz)
- GASNet Core API
- GASNet Extended API
- Network Hardware
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

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

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

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 APIs: 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

GASNet on Cray X1 / shmem

- Uses shmem for implementing core API Active Messages
- Uses hardware 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 central vectorization
  - gasnet_put/get fully inline - allows caller vectorization
  - shmem-conduit also supports SGI Altix
  - similar global system, but remote memory is cached and processor is titanium-2 (no vectors)

Firehose Memory Registration Library

- 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

Survey of Approaches to Memory Registration for HPC NICs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Zero-copy</th>
<th>One-sided</th>
<th>Full VM</th>
<th>Description, Pros and Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware-based (e.g. Quadrics)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Hardware manages memory: No hardw. choosing or benefit from software. Hardware completely and price. Kernel modifications are not required.</td>
</tr>
<tr>
<td>Pin Everything</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Pin all pages at startup or when allocated (default). Total usage limited to physical memory. May require a custom daemon.</td>
</tr>
<tr>
<td>Source Buffers</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Stream data through preinitialized buffers on on-chip buffers. Stream &amp; data paths (DMA vs. software). Message overheads overheard (create &amp; de-queue).</td>
</tr>
<tr>
<td>Rendezvous</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Round-trip message to pin remote pages before each I/O or computation.</td>
</tr>
<tr>
<td>Firehose</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Common case: All the benefits of hardware-assisted unmanaged case. Memory overhead (create &amp; de-queue).</td>
</tr>
</tbody>
</table>

Firehose Algorithm for distributed management of DMA registration