Experiences Implementing Partitioned Global Address Space (PGAS) Languages on InfiniBand

Paul H. Hargrove (LBNL) with Dan Bonachea and Christian Bell

http://gasnet.cs.berkeley.edu

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
Outline

- Background
- GASNet vapi-conduit / ibv-conduit
- RDMA Put/Get
- Active Messages (RPC)
- Asynchronous Progress Threads
- Memory Registration
Background – PGAS & GASNet

• **Partitioned Global Address Space (PGAS) Languages**
  - Examples
    - Unified Parallel C (UPC), Titanium and Co-Array FORTRAN
  - Shared memory style programming
  - “Global pointers” as a language concept
  - Explicit memory affinity for global pointers

• **Global Address Space Networking (GASNet)**
  - Language-independent library for PGAS network support
  - Designed as a compilation target, not for end users
  - Project of Lawrence Berkeley National Lab and the University of California Berkeley (P.I. Kathy Yelick)
**Background – GASNet API**

- **GASNet “Core” API**
  - Active Message (RPC) interface
  - Minimum requirement for a new port – “Reference Extended” implements Extended via Core

- **GASNet “Extended” API**
  - Remote Put and Get operations
  - Blocking and Non-blocking (multiple variants)
    - Implicit (“region” based) or Explicit (“handle” based)
    - Initiation of Puts with or without local completion
GASNet – vapi- and ibv-conduits

• The network-specific code in GASNet is a “conduit”
• InfiniBand support began with Mellanox VAPI
  • “vapi-conduit”
• Later Open Fabrics verbs “ibv” support added
  • “ibv-conduit”
• Same source code supports both APIs via a thin layer of macros (and some #ifdef’s)
• Very little (if any) beyond VAPI 1.0 features
**RDMA Put and Get**

- **Initiator** provides everything needed to complete one-sided communication
  - Local address and length; remote node and address
- **GASNet** needs just a thin layer over InfiniBand RDMA_WRITE and RDMA_READ
  - Uses inline send when possible
  - Uses `wr_id` to connect CQE to GASNet op for completion
  - Uses semaphore (try_down/up) to control SQ/CQ depth
  - TO DO: suppress CQEs when possible
- **Wish List**: verbs-level CQ depth management?
Active Messages (RPC)

- RPC mechanism based on Berkeley AM
  - Request with optional reply – no other comms
  - Used by language runtimes (locks, memory alloc, etc.)
- Primary channel uses SEND_WITH_IMM
  - Credit-based flow control (we never see RNR)
  - TO DO: Utilize SRQ and revisit flow control
- Secondary channel uses RMDA_WRITE
  - Based on success with similar optimization in MVAPICH
  - No CQE – poll in memory (csum based, not “last byte”)
  - For bounded number of “hot peers” only
- Wish list: SEND w/ lower latency
Asynchronous Progress Threads

• Polling-base progress may not service AMs for long periods of time
  • Bad for apps when memory allocation or locks involved
  • Bad for memory registration rendezvous (next section)
• Initial design used EVAPI_set_comp_eventh()
  • Never found “well behaved” app that benefited
  • “Network attentive” apps saw performance decline
  • TO DO: progress thread not implemented yet for ibv
• Wish List: ibv_req_notify_cq_timed()?
  • Event when CQE remains unserviced “too long”
Memory Registration – “FIREHOSE”

• An algorithm for distributed management of memory registration
  • Exposes one-sided, zero-copy RDMA as common case
  • Degrades gracefully to rendezvous as working set grows
• Used in gm, vapi/ibv, lapi and (soon) portals
Memory Registration

- Registration is required (Protection)
  - Need Protection = access/Rkey/Lkey
- As a ULP we don’t need “pinning” (Translation)
  - Source of many woes
- Dynamic registration is costly
  - Cost in time motivates aggressive caching/reuse
  - Roughly as much code as for RDMA and AMs
- Wish List: non-pinning memory registration
  - Associate access/Rkey/Lkey with address range
  - Lazy translation – ideally w/ page allocation
Summary

• PGAS Put/Get map well to RDMA Read/Write
  • Queue the RDMAs, reap the completions
  • The 64-bit wr_id links completions back to GASNet ops
  • Need to manage CQ space

• AM/RPC support fits less well
  • Like the MPI implementers, we work around the latency of CQE generation on receiver
  • Async progress not yet seen to be helpful with the current notification facilities

• Memory registration
  • Like the MPI implementers, we devote far too much code to this
  • Must cache registrations to amortize their costs
  • Wish registration didn’t imply pinning
BACKUP SLIDES...
## Memory Registration Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Zero-copy</th>
<th>One-sided</th>
<th>Full VM avail</th>
<th>Description, <strong>Pros</strong> and <strong>Cons</strong></th>
</tr>
</thead>
</table>
| Hardware-based (eg. Quadrics) | ✓         | ✓         | ✓             | Hardware/firmware manage everything  
No handshaking or bookkeeping in software  
Hardware complexity and price, Kernel modifications                                                                                                                                                                         |
| Pin Everything            | ✓         | ✓         | X             | Pin all pages at startup or when allocated (collectively)  
Total usage limited to physical memory  
May require a custom allocator                                                                                                                                                                                                |
| Bounce Buffers            | X         | X         | ✓             | Stream data through pre-pinned bufs on one/both sides  
Mem copy costs (CPU consumption, cache pollution, prevents comm. & computation overlap)  
Messaging overhead (metadata & handshaking)                                                                                                                                                                                   |
| Rendezvous                | ✓         | X         | ✓             | 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)                                                                                                                                                                                                     |
**Firehose: Conceptual Diagram**

- **Basic Idea:** Use AM to delegate control over registration to the RDMA initiators

- A and C each control a share of pinnable memory on B
- A and C can freely "pour" data through their firehoses using RDMA to/from anywhere in the memory they map on B
- Use AM to reposition firehoses
- Refcounts used to track number of attached firehoses (or local pins)
- Support lazy deregistration for buckets w/ refcount = 0 to avoid re-pinning costs
Summary of Firehose Results

- Firehose algorithm is an ideal registration strategy for GAS languages on pinning-based networks
  - Performance of Pin-Everything (without the drawbacks) in the common case, degrades to Rendezvous-like behavior for the uncommon case
  - Exposes one-sided, zero-copy RDMA as common case
  - Amortizes cost of registration/synch over many ops, uses temporal/spatial locality to avoid cost of repinning
  - Cost of handshaking and registration negligible when working set fits in physical memory, degrades gracefully beyond
Vapi-conduit Performance Nov. 2004

![Graph showing bandwidth (KB/s) vs. size (bytes) for different applications.]

- **Titanium**
- **UPC (shared source)**
- **UPC (private source)**

(up is good)
InfiniBand Multi-QP (puts)
InfiniBand Multi-QP (gets)

Bandwidth (KBytes/s)

Size (bytes)

get_nbi_bulk 1qp
get_nbi_bulk 2qp

(up is good)
GASNet vs. MPI on InfiniBand (Jul ‘05)

![Graph comparing GASNet and MPI performance](image.png)

- **gossip_put_nbi_bulk**
- **gossip_put_bulk**
- **MPI Flood**
- **MPI Ping/Ack**

**Relative BW (put_nbi_bulk/MPI_Flood):**

- 1.0
- 1.2
- 1.4
- 1.6
- 1.8
- 2.0
- 2.2
- 2.4

**Size (bytes):**

- 10
- 100
- 1,000
- 10,000
- 100,000
- 1,000,000
- 10,000,000

**Bandwidth (KB/s):**

- 0
- 10,000
- 100,000
- 1,000,000
- 10,000,000

**Notes:**

- Up is good
- Implementing PGAS on InfiniBand