Interconnect Requirements for Partitioned Global Address Space (PGAS) Languages

Paul H. Hargrove

PHHargrove@lbl.gov

July 21, 2008

Based on a colleague's panel presentation, given October 2006:

PGAS 2006 Discussion Panel

Dan Bonachea
UC Berkeley / LBNL

What Improvements are Needed to Underlying Interconnect Hardware and Software to Support Better PGAS Application Performance Across the Mix of Commodity and Custom HPC Systems?

Some PGAS Characteristics

- PGAS allows shared-memory style programs on distributed memory machines
  - Code to access remote data same as local access (or nearly so)
- PGAS enables "irregular" applications
  - All RDMA ops are "unexpected messages"
  - Apps need not be "attentive" to the network
- Runtime implementations need RPC mechanism
  - For locks, memory allocation, etc.
  - "Active Messages" in GASNet

Wish list #1

- Fully one-sided RDMA access
  - No polling, interrupts, progress threads, etc.
- RDMA access to full VM space of peers
  - "Registration" for protection: OK
  - "Pinning" for translation: NOT OK
- Flow control for Active Messages
  - One-to-all and all-to-all "unexpected" messaging patterns
  - ULP solutions are "dirty"

Wish list #2

- **Scatter/gather support (offloaded)**
  - **Pack/unpack solutions are a poor use of host resources**

- **Collectives support (offloaded)**
  - **At least the basics: Bcast, Scatter, Gather**
  - **Non-blocking**

Wish list #3

- NIC-to-CPU cache coherence
  - Must include vector units, etc.
- Progress
  - E.g. for tree-based collective operations
  - Queue operations to be triggered remotely?
- Open API specifications
  - Open Source implementations even better

Dan Bonachea's "How NOT to build a system for PGAS":

(nearly?) every bullet represents something we've actually encountered.

How NOT to build a system for PGAS

- Don't implement reliable delivery of communication
  - reliability should be handled by application programmers!
- Don't provide any network flow control
  - contention is a fatal error! (so is all-to-one communication)
- Don't provide a job spawner
  - each admin should roll their own with ssh - they love that!
- Don't provide cache coherence between CPUs or NIC & CPU
  - apps should program "carefully" to avoid sharing a cache line anywhere
- Wrap all interconnect library calls in a critical section with a big lock
  - who'd want to use threads, or have CPU's share a NIC?
- Route all messages through the kernel on one or both sides
  - because our kernel is "fast"!!!
  - this way we can implement protection in software on the cheap!
- Require CPU interrupts in the critical path
  - only costs a few microseconds, right? (what's cache pollution?)
- Stall remote RMA's when passive-side CPU is inattentive to network
  - it'll never show up in the marketing microbenchmarks - who does FLOPS?
- Provide software interface that assumes clients will never use threads

How NOT to build a system for PGAS

- Don't expose low-level communication layer to open software
  - all system programmers should target MPI! (except for our employees)
- Only allow RDMA in kernel-level software/drivers
  - great for the parallel f/s, but you'd never need RDMA at user-level…
- Provide RDMA put, but not RDMA get
- Expose only blocking communication operations to software
  - we don't believe in overlap managed in software or compiler
- Require all RMA ops to be size/addr aligned at 64 bytes (512 bits)
  - smaller sizes r-m-w over network (what's a consistency model?)
- Require all shared memory to be registered before use
  - collectively, if possible.. and make it ridiculously slow (linked list of pages?)
- Prohibit dynamic (de)registration of shared memory
  - or clear all the data on a page to change its registration
- Only expose only a few MB of shared memory
  - why would you need more?
- Use an I/O bus slower than network link speed
  - users will appreciate our "forward-looking" design