Open MPI State of the Union (brief) and OpenFabrics Feedback

OpenFabrics 2009 Sonoma Workshop

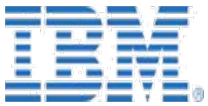
Jeff Squyres

CISCO

Agenda

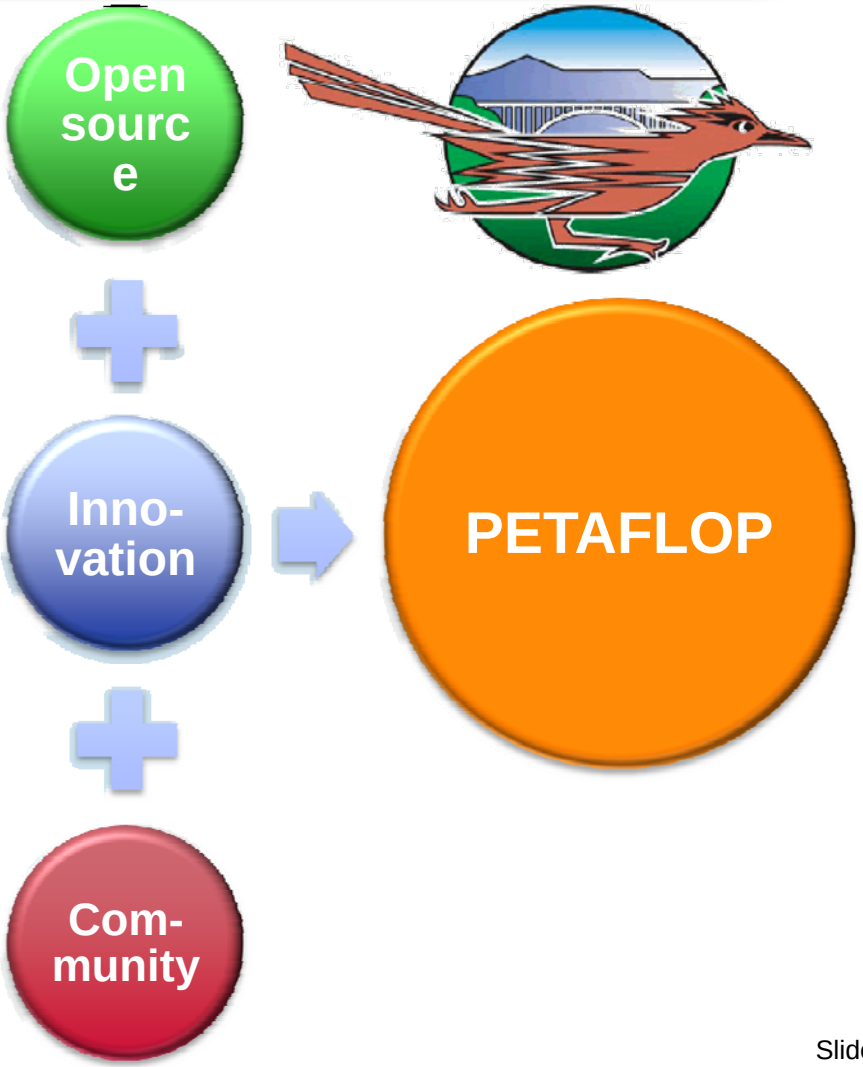
- Very brief – mainly for web archive
  - Open MPI Project / Community
  - Los Alamos Road Runner / Petaflop
  - Current Status
- OpenFabrics Feedback

16 Members, 8 Contributors, 2 Partners



Petaflop!!

- Los Alamos Road Runner
- #1 on Nov. 2008 Top500
  - 1.1 petaflops
- Powered by Open MPI
  - Significant community achievement



Current Status

- OFED 1.4[rcX] contains Open MPI 1.3.1
- Many performance enhancements
- OpenFabrics features
  - iWARP support
  - "Bucket" SRQ
  - XRC support
  - Message coalescing
  - Asynchronous error events
  - Automatic Path Migration (APM)
  - Improved processor + port binding

Release Methodology

- "Super stable" series
  - Even version numbers (1.4, 1.6, …)
  - Stable, tested
  - Time tested, mature
- Recommended for production sites
  - Small perturbations between releases

- Feature driven series
  - Odd version numbers (1.3, 1.5, …)
  - Stable, tested
- Recommended for feature-driven sites
  - Large perturbations between releases

OpenFabrics Feedback

- Summary of identified challenges from
  - Open MPI
    (including Sun ClusterTools)
  - HP MPI
  - Intel MPI
  - Platform MPI



- *<u>Items are listed in priority order</u>*
  - "→" marked items are suggestions / requests
  - "★" marked items are shared by at least 2 MPI's

OpenFabrics Feedback

★ 1. Memory registration: painful, dangerous
★ 2. fork() support inadequate
★ 3. Connection setup scalability problematic
4. Relaxed ordering verbs API support
★ 5. [Lack of] API portability
★ 6. Need reliable connectionless (scalability)
★ 7. Need better S/R registered memory utilztn.
★ 8. CMs are waaaay too complex

★1. Memory Registration

- Must be creative to deal with MR slowness
  - Pipelining, caching, etc.
  - *Dangerous* tricks to catch free, munmap, sbrk

→ Notify when virt. / phys. mapping changes

  - Pete Wyckoff proposed a way years ago
  - Other methods have also been discussed

→ Make MR / MD faster and better

  - Can MR / registration cache be hidden?
  - Can MR go away? (and still keep high perf.)

★2. fork() Support

- Still problematic
  - (Partial) Pages registered in parent not in child
  - Unfortunately, *many* user codes call fork()
  - Child dup's open device fd's upon fork()
    - Parent exits, connections should close
    - …but they don't because the device is still open

→ Child memory after fork() should behave exactly as it does without registered memory

→ Device fd's should be set to "close on exec"

★3. Connection Setup Scalability

- "All to all" RC connections at scale
  - Current in-band mechanisms do not scale
    - IB SM: cannot handle N^2 path record lookups
    - iWARP: preventing ARP floods requires extra setup
  - Requires out-of-band MPI info exchange

→ Need in-band, scalable CM

  - Should work "out of the box"
  - Should not require workarounds: dumping SM tables to files, preloading ARP caches, etc.

4. Relaxed Ordering API Support

- Platforms now using relaxed PCI ordering for memory bandwidth optimization
  - Precludes MPI's "eager RDMA" optimization
  - Can't poll memory to know when xfer complete
  - 30%+ latency difference to use send/receive

→ Simple verbs API change

  - Specify whether to use strict/relaxed ordering for individual memory registrations

★5. Stack / API Portability

- Windows API very different than Linux API
- Solaris API “close” to Linux API
  - OMPI uses side effects to determine portability

→ Standardize 1 API for all platforms / OSs

  - Standardize way for per-platform extensions

→ Keep slow, well-announced ABI changes

  - Innovation is good, ISVs need time to react
  - MPI needs to be able to adapt at run-time

★6. Reliable Connectionless

- Connection-oriented does not scale
  - N^2 use of resources
  - XRC helps, but is quite complex
- Datagram support in general is lacking
  - Mellanox HCAs need 2 UD QPs to get full BW

→ Need RD (or something like it)

  - Must still maintain RC-like high performance
  - Mix of hardware+middleware might work (MX)

★7. S/R Registered Memory Utilization

- S/R receiver buffer utilization can be poor
  - Hard to balance resource consumption (memory) between MPI and application
  - Frequent app complaint: “MPI takes too much memory”
  - Fixed size receive buffers ignore actual incoming size

→ Post a “slab” of memory for receives

  - Pack received messages more efficiently

★8. CM's (Far) Too Complex

- Effectively requires a progress thread for incoming connections
  - Or MPI implements all the timeout/retry code
- Significant ULP complexity required
  - OMPI: 2,800 LOC just for RDMA CM
  - OMPI: 2,300 LOC for *all of MX*

→ Want a higher-level API

- Middleware, kernel – doesn't matter
- Simple non-blocking connect and accept
- *Handle all connection progression*

Come Join Us!

http://www.open-mpi.org/