**Agenda**

- Open MPI Project / Community
- Roadmap
- Select organization project updates
  - Nvidia, Fujitsu, U. Tennessee, Cisco, others
- The (continuing) road to MPI-3
- Community questions
Open MPI Is…

• Evolution of several prior MPI’s
• Open source project and community
  ▪ Production quality
  ▪ Vendor-friendly
  ▪ Research- and academic-friendly
• MPI-2.1 compliant
Members, Contributors, Partners
Open MPI has 2 concurrent release series
- “Feature series” → v1.<odd>
- “Super stable series” → v1.<even>

Both are tested and QA’ed
- Main difference between the two is time
Feature Series

Development trunk

Branch to create Feature series

New features, enhancements

Transition to super stable

Bug fixes only

New branch, to become v1.7 / v1.8
v1.4 Series Sunset

- v1.4 is the current “super stable series”
- Likely to only have one more release
  - A few more bug fixes have crept in
  - v1.4.5 possibly in December

So long, v1.4 series!
v1.5 → v1.6 Transition

- ABI change since v1.4
- New features over the v1.5 series
  - Support for Mellanox “MXM” and offloaded collectives support (Voltaire)
  - ARM support
  - InfiniBand failover transport
  - WinVerbs support
  - Significant run-time scalability, robustness
  - …oodles of little improvements and fixes
v1.5 \rightarrow v1.6 Transition

- One more release in v1.5
  - Final MPI-2.2 functionality (no strong demand)
  - hwloc version bump
  - Stronger PMI support
  - Usual array of bug fixes, minor enhancements
- Aiming for December, 2011
  - US holiday schedule may force pushing to Jan
  - Transition to v1.6 a fixed time after that
- **ESTIMATE:** Q1 2012
Several upcoming v1.7 features discussed later in this presentation

ABI break from v1.5 / v1.6

Gating factors for v1.7 branch:
- v1.6 release
- Stability of new trunk features
- Have not yet elected v1.7 release managers

**ESTIMATE:** Q2 2012
Tremendous growth in CUDA adoption
Joined in April, 2011

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Alamos National Laboratory</td>
<td>Member</td>
</tr>
<tr>
<td>Mellanox Technologies</td>
<td>Member</td>
</tr>
<tr>
<td>Myricom, Inc.</td>
<td>Contributor</td>
</tr>
<tr>
<td>Nvidia</td>
<td>Contributor</td>
</tr>
<tr>
<td>Oak Ridge National Laboratory</td>
<td>Member</td>
</tr>
<tr>
<td>Oak Ridge National Laboratory</td>
<td>Member</td>
</tr>
<tr>
<td>Open Systems Laboratory</td>
<td>Member</td>
</tr>
<tr>
<td>Oracle</td>
<td>Member</td>
</tr>
<tr>
<td>Platform Computing</td>
<td>Contributor</td>
</tr>
</tbody>
</table>
Make Open MPI aware of CUDA

- Allow users to send and receive GPU buffers directly
- Hide complexity with the MPI stack
Make Open MPI aware of CUDA

- Stage data in host memory prior to MPI calls
  
  ```
  cuMemAlloc(devptr, size)
  kernel<<grid, block>>(devptr)
  hostptr = malloc(size)
  cuMemcpy(hostptr, devptr, size)
  MPI_Send(hostptr, ...
  ```

- Access device memory directly from MPI calls
  
  ```
  cuMemAlloc(devptr, size)
  kernel<<grid, block>>(devptr)
  MPI_Send(devptr, ...
  ```
Move GPU buffers within MPI

Original

New
Open MPI Plan

Three Phases
1. Add basic support - Done
2. Add registration of internal buffers - Done
3. Add interprocess memory support within a node – prototype working
Phase 1

- All changes were made in datatype and convertor code.
- Add new pointer in convertor that points to a memcpy routine.
- When MPI request is initialized, input buffer is queried and memcpy routine can be changed to CUDA routine, cuMemcpy.
- Modify opal_convertor_need_buffers() to return true if buffer is device memory (special flag added to convertor).
Phase 1 - Continued

- Code is enabled with `--with-cuda` and `--with-cuda-libdir`.
- Added to Open MPI trunk April, 2011
- [http://www.open-mpi.org/faq/?category=building#build-cuda](http://www.open-mpi.org/faq/?category=building#build-cuda)
- [http://www.open-mpi.org/faq/?category=running#mpi-cuda-support](http://www.open-mpi.org/faq/?category=running#mpi-cuda-support)
With these changes, we can support all the following APIs.

- MPI_Send, MPI_Recv, MPI_Isend, etc.
- MPI_Bcast, MPI_Gather, MPI_Scatter, etc.

No support for reductions or one-sided.

Supports both contiguous and non-contiguous datatypes.
Issues - Performance

- Each call to cuMemcpy incurs a 10us overhead.
- For IB and TCP, forcing usage of the pipelined send protocol can affect large message performance.
- For SM, overhead of cuMemcpy limits performance for large messages also.
Phase 2

- Register internal host buffers with cuMemHostRegister.
- Improved IB performance
- Allows possible change to asynchronous cuMemcpys in the MPI library.
- Added to Open MPI trunk August, 2011
Phase 3 – Improve on-node performance

CUDA 4.1 added new interprocess communication utilities.

- cuIpcGetMemHandle
- cuIpcOpenMemHandle
- cuIpcCloseMemHandle
- cuIpcGetEventHandle
- cuIpcOpenEventHandle
Remote GET for GPU memory

Process 0
MPI_Send(sendbuf, ...)

sendbuf

culpcGetMemHandle(&memHandle, sendbuf)

64 byte MemHandle Plus other Header stuff Needed for matching

Call upper layer to complete transaction

Process 1
MPI_Recv(recvbuf, ...)

recvbuf

rembuf

cuMemcopyAsync

culpcOpenMemHandle(&rembuf, memHandle)
cuMemcopyAsync(recvbuf, rembuf, size, 0)
cuEventRecord(event, 0)

PROGRESS:
If(cuEventQuery(event) == CUDA_EVENT_NOT_READY)
return;
else
  send_message_to_sender()
call upper layer to complete transaction
Memory Handles

- `culpcGetMemHandle` – 1 usec
- `culpcOpenMemHandle` – 100 usec

Therefore, cache the memory handles from remote processes and reuse them if the user reuses them. Similar to IB BTL.

Great benefit where user buffers are reused.
GPU to GPU within node

Comparison of various protocols

<table>
<thead>
<tr>
<th>Size</th>
<th>p2p</th>
<th>MPI-Pin</th>
<th>MPI-Unpin</th>
<th>No RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>0.9</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>16</td>
<td>4.0</td>
<td>1.7</td>
<td>1.7</td>
<td>0.5</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>3.4</td>
<td>3.3</td>
<td>0.9</td>
</tr>
<tr>
<td>64</td>
<td>12.0</td>
<td>6.7</td>
<td>6.5</td>
<td>1.8</td>
</tr>
<tr>
<td>128</td>
<td>32.0</td>
<td>13.3</td>
<td>13.0</td>
<td>3.7</td>
</tr>
<tr>
<td>256</td>
<td>63.0</td>
<td>26.5</td>
<td>25.8</td>
<td>7.4</td>
</tr>
<tr>
<td>512</td>
<td>125.0</td>
<td>52.7</td>
<td>51.2</td>
<td>14.8</td>
</tr>
<tr>
<td>1024</td>
<td>250.0</td>
<td>105.2</td>
<td>102.1</td>
<td>29.7</td>
</tr>
<tr>
<td>2048</td>
<td>448.0</td>
<td>206.5</td>
<td>201.8</td>
<td>58.5</td>
</tr>
<tr>
<td>4096</td>
<td>879.0</td>
<td>566.0</td>
<td>17.7</td>
<td>117.0</td>
</tr>
<tr>
<td>8192</td>
<td>1425.0</td>
<td>1014.7</td>
<td>35.1</td>
<td>230.9</td>
</tr>
<tr>
<td>16384</td>
<td>2192.0</td>
<td>1671.4</td>
<td>70.4</td>
<td>419.4</td>
</tr>
<tr>
<td>32768</td>
<td>2915.0</td>
<td>2469.3</td>
<td>139.0</td>
<td>434.5</td>
</tr>
<tr>
<td>65536</td>
<td>3458.0</td>
<td>3259.2</td>
<td>268.8</td>
<td>577.0</td>
</tr>
<tr>
<td>131072</td>
<td>3928.0</td>
<td>3878.5</td>
<td>523.7</td>
<td>691.3</td>
</tr>
<tr>
<td>262144</td>
<td>4170.0</td>
<td>4292.8</td>
<td>977.6</td>
<td>765.5</td>
</tr>
<tr>
<td>524288</td>
<td>4322.0</td>
<td>4527.3</td>
<td>1820.4</td>
<td>808.2</td>
</tr>
<tr>
<td>1048576</td>
<td>4399.0</td>
<td>4618.1</td>
<td>2713.4</td>
<td>829.0</td>
</tr>
<tr>
<td>2097152</td>
<td>4438.0</td>
<td>4731.5</td>
<td>3530.7</td>
<td>842.1</td>
</tr>
<tr>
<td>4194304</td>
<td>4457.0</td>
<td>4771.3</td>
<td>4119.7</td>
<td>845.6</td>
</tr>
</tbody>
</table>
Future

- More use of CUDA asynchronous copies
- Improved GPU to GPU memory communication between nodes.
- Better noncontiguous datatypes and collectives. (NVIDIA funding university research into this)
The ‘Super’ Computing Company
From Super Phones to Super Computers

Thanks
rvandevaart@nvidia.com
#1, baby!

- 10.51 petaflops
  - K “cranked it up to 11” (rounding up 😊)
Flexibility

• Support several backend runtimes
  ▪ Eventually with different levels of integrations
    • Notifiers / specialized logging services might not be available everywhere
  ▪ And different capabilities
    • MPI 2 dynamic processing or fault tolerance might be only partially supported in some environments.

• Open RTE, PMI, Hydra, local
Scalability

• Startup
  ▪ Gracefully handle many processes per node
  ▪ Minimize resource consumption while maximizing parallelism: build specialized network overlays


• Business card (Modex) exchange
  ▪ Use the network overlays to exchange the business cards of the participating processes
  ▪ Keep one single copy per node shared between all local processes
  ▪ Update the data asynchronously

- Self-adapting algorithms to evolve from any type of spanning tree toward BMG
- Good candidate for resilient runtime
Fault Tolerance
Correlated Set in Message Logging

**Coordinated C/R**
- A complete checkpoint is taken at specified time intervals
- In case of a failure all processes rollback to the last valid checkpoint
- The time to checkpoint strongly depends on the checkpoint support (I/O bandwidth)

**Uncoordinated C/R**
- A single checkpoint is taken at specified time intervals
- In case of a failure one process rollback to the last valid checkpoint
- The time to checkpoint barely depends on the checkpoint support (I/O bandwidth)
• **Hybrid** between coordinated and uncoordinated
• **Codependent failures** are defined as sets of processes prone to fail simultaneously (cores of a same node)
• Codependent processes use coordinated checkpoint: relieves the need for expensive sender-based logging
• Non codependent processes are still uncoordinated and benefit from faster recovery
Non deterministic events are still logged, but payload in a correlated set is not
MPI Forum Fault Tolerance Working Group

Define a set of semantics and interfaces to enable fault tolerant applications and libraries to be portably constructed on top of MPI.

- Application involved fault tolerance (not transparent FT)
  - Natural & Algorithm Based Fault Tolerance (ABFT)
- Fail-stop process failure:
  - MPI process permanently stops communicating with other processes.
- Two Complementary Proposals:
  - Run-Through Stabilization: *(Target: MPI-3.0)*
    - Continue running and using MPI even if one or more MPI processes fail
  - Process Recovery: *(Target: MPI-3.1)*
    - Replace MPI processes in existing communicators, windows, file handles
- Prototype in Open MPI is guiding proposal development

MPI Forum Fault Tolerance Working Group:
Open MPI Prototype of the Run-Through Stabilization Proposal

• Pt2Pt Overhead: NetPIPE over shared memory
  ▪ Latency: 0.84 to 0.85 microseconds (1.2%)
  ▪ Bandwidth: 8957 to 8920 Mbps (0.4%)

• Fault Aware Collective Performance
  ▪ MPI_Barrier & MPI_Bcast:
    Within 1% of fault-unaware, regardless of # failures
  ▪ MPI_Comm_validate_all: New fault tolerant agreement collective
    Within 3% of MPI_Allreduce() collective, log-scaling

• Prototype available to interested application developers
  ▪ Contact: Josh Hursey jjhursey@open-mpi.org
Point-to-point communications
Open MPI for Cray XE Systems

- uGNI and Vader BTLs provide point-to-point and shared-memory communication functionality
- uGNI BTL implements three protocols for Internode communication
  - Eager protocol for short message transfer
    - Send/Recv for short message (SMSG)
  - Rendezvous protocol for long message transfer
    - RDMA Read/Write for medium message (FMA)
    - Offloaded RDMA/Write for long message (BTE)
- Vader BTL provides protocols for Intranode communication
  - Single copy between source and destination buffers using Cray xpmem
  - Nemesis-style lock free fifos for small message delivery
Open MPI uGNI BTL Latency and Bandwidth (Preliminary Results)

Project members:
ORNLM: Richard Graham, Manjunath Gorentla Venkata
LANL: Samuel Gutierrez, Nathan Hjelm
SNL: Brain Barrett
### Adapting to NUMA architectures

<table>
<thead>
<tr>
<th>CPU</th>
<th>locality</th>
<th>btl_eager_limit</th>
<th>pipe_size</th>
<th>use_knem</th>
<th>DMA_min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tigerton</td>
<td>No shared L2</td>
<td>2k</td>
<td>0.5 * L1 size</td>
<td>true</td>
<td>2MB</td>
</tr>
<tr>
<td>Nehalem EP</td>
<td>No shared L2</td>
<td>4k</td>
<td>0.5 * L1 size</td>
<td>false</td>
<td>0</td>
</tr>
<tr>
<td>Tigerton</td>
<td>Shared L2</td>
<td>2k</td>
<td>L1 size</td>
<td>true</td>
<td>4MB</td>
</tr>
</tbody>
</table>
Fig. 3. Bandwidth of the ping-pong test for vanilla MPICH2, vanilla OpenMPI and multi-tuned Open MPI
Collective communications
Barrier – Comparison with Native MPI

![Graph showing latency vs. processor cores for different MPI implementations.](image-url)
Large-Scale Broadcast Performance: OMPI vs Native MPI large message 16 MBytes
Non-blocking Bcast Overlap – IB CORE-Direct

![Graph showing non-blocking broadcast overlap vs message size in bytes.](image-url)
Architecture aware collective

Bcast Aggregate BW on parapluie (27 nodes, 24 cores/node, 20 G IB)

Reduce Aggregate BW on parapluie (27 nodes, 24 core/node, 20 G IB)
Cisco 1st Gen. Ethernet MPI Transport Technology Preview

- Demo in Cisco booth (#1317)
  - New Open MPI BTL (point-to-point transport)
  - Ethernet NetPIPE latency: 5.17us
- Using Linux VFIO technology
  - NOTE: VFIO is not upstream yet
- This is not RoCE, not iWARP
- Cisco 2nd generation NIC coming “soon”
  - Latency will be significantly lower than 5.17us
Processor Affinity

- Core counts are rising
- Users are asking for powerful, flexible affinity controls
  - Bind processes to an entire sockets
  - Bind processes to half the cores in a socket
  - Bind processes to a NUMA locality
  - …etc.
- Joint work between Cisco, Oracle, ORNL
Processor Affinity

- Processor affinity revamp
  - Overview presented at SC’10 SoU BOF
  - Took a loooong time to implement
- Branched for this work last year
  - Just folded first major part back to SVN trunk
  - More coming soon (still testing)
- Slated for v1.7
  - We need real-world feedback
Processor Affinity

- `mpirun` reads from compute nodes
  - Sockets, cores, threads, caches, NUMA, etc.
  - Maps MPI processes according to what exists
- Useful for:
  - Dissimilar head node
  - Heterogeneous

`Compute nodes`
Processor Affinity

• Clarified, fixed mpirun affinity options
  ▪ --map-by <entity>
  ▪ --bind-to <entity>

• New options for flexible mapping / binding
  ▪ Inspired by Blue Gene XYZ specification
  ▪ --map <letter sequence>
  ▪ --bind <letter sequence>
  ▪ Letters for thread, core, socket, NUMA node, caches, server node
The (Continuing) Road to MPI-3

Jeff Squyres

CISCO
MPI-3 Prototyping Work

- MPI-3 has a "freely available implementation" requirement
  - Much work being prototyped in Open MPI
  - Will help speed our final implementation
MPI-3 Prototyping Work

• New Fortran ‘08 bindings
  ▪ Compile-time sub. parameter type safety
  ▪ Unique types for MPI handles
  ▪ Safe non-blocking MPI functionality (when compilers support it)

• Better “use mpi” implementation
  ▪ …except for gfortran 😞

• Craig Rasmussen (Los Alamos National Labs), Jeff Squyres (Cisco)
MPI-3 Prototyping Work

- **MPI_MPROBE**
  - Matched probe
  - Helpful for threaded MPI apps
  - Helpful for upper-level bindings (e.g., Python)
- Almost ready to be folded back to SVN trunk
- Brian Barrett (Sandia National Labs)
MPI-3 Prototyping Work

- Run-through stabilization prototype
  - Gracefully allow for process failure(s)
  - New MPI API functions
  - Adapt underlying MPI run-time to not automatically kill the entire job
  - Define what happens in the MPI layer
- Josh Hursey (Oak Ridge National Labs)
MPI-3 Prototype Work

- New one-sided / RMA chapter
  - Implementation on Portals
  - Tweaking of infrastructure for other underlying transports
- Almost ready to be folded back to SVN trunk
- Brian Barrett (Sandia National Labs)
MPI Forum = Needs Feedback

• MPI Forum BOF tonight
  - 5:30pm, TCC 301/302
  - Slides to be posted on meetings.mpi-forum.org

• PLEASE send your feedback
  - Many of the Forum are implementers
  - Need real world user feedback

• Next face-to-face meeting:
  - Cisco, San Jose, CA, USA, Jan. 9-11, 2012
• Community questions
  ▪ Feedback: http://www.open-mpi.org/sc2011
Come Join Us!

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