Chapel over MPI-3

Pavan Balaji
Computer Scientist
Group Lead, Programming Models and Runtime Systems
Mathematics and Computer Science Division
Argonne National Laboratory
MPI: A Philosophical Perspective

- Debate in the community that MPI is too hard to program
  - We don’t disagree; it was never meant to be easy to program 😊

- A programming model has to pick a tradeoff between programmability, portability, and performance
  - MPI has chosen to be a high-performance and portable programming model
  - If we have to pick between losing performance/portability vs. losing programmability, we typically pick the latter
  - Focus has been on completeness and ability to help real and complex applications meet their computational needs

- *MPI’s goal is not to make simple programs easy to write, rather it is to make complex programs possible to write*
MPI as a Common Runtime System

- We want to encourage high-level libraries/languages on top of MPI to meet productivity goals or domain-specific optimizations.
- MPI was traditionally considered to be too “two-sided” in nature to be suitable to be a PGAS or HPCS runtime system.
- MPI-2 introduced one-sided communication, but the semantics were too restrictive to take advantage of modern hardware:
  - Cache-coherent systems
  - PUT/GET hardware
- MPI-3 introduced a significantly revamped one-sided communication interface:
  - Better support for cache-coherent hardware
  - Better communication and synchronization primitives
- The standard was released in September 2012.
- MPICH released a (full) implementation at SC 2012.
## Status of MPI-3 Implementations (as of 11/20/2013)

<table>
<thead>
<tr>
<th>Feature</th>
<th>MPICH</th>
<th>MVAPICH</th>
<th>Cray</th>
<th>Tianhe</th>
<th>Intel</th>
<th>IBM PE</th>
<th>IBM Platform</th>
<th>Open MPI</th>
<th>SGI</th>
<th>Fujitsu</th>
<th>MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB collectives</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Neighborhood collectives</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q3 ‘14</td>
<td>Q4 ‘13 (in nightly snapshots)</td>
<td>✔</td>
<td>Q1 ‘14</td>
</tr>
<tr>
<td>RMA</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q3 ‘14</td>
<td>Q1 ‘14</td>
<td>Q4 ‘13</td>
<td></td>
</tr>
<tr>
<td>Shared memory</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q2 ‘15</td>
<td>Q1 ‘14</td>
<td>Q4 ‘13</td>
<td></td>
</tr>
<tr>
<td>Tools Interface</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q2 ‘14</td>
<td>Q1 ‘14</td>
<td>Q1 ‘14</td>
<td>Q2 ‘15</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Non-collective comm. create</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q2 ‘15</td>
<td>Q4 ‘13 (in nightly snapshots)</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>F08 Bindings (MPI-3 + errata)</td>
<td>(Q1 ‘14)</td>
<td>(Q2 ‘14)</td>
<td>(Q2 ‘14)</td>
<td>(Q1 ‘14)</td>
<td>(Q1 ‘14)</td>
<td>(Q1 ‘14)</td>
<td>(Q3 ‘14)</td>
<td>(✔)</td>
<td>(Q1 ‘14)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Datatypes</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q2 ‘15</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Large Counts</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q2 ‘15</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td></td>
</tr>
<tr>
<td>Matched Probe</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Q1 ‘14</td>
<td>Q3 ‘14</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

Release dates are estimates and are subject to change at any time. Empty cells indicate no *publicly announced* plan to implement/support that feature.
Chapel over MPI-3

- Combine strengths of different runtime systems
  - Performance of MPI
  - Usability of PGAS languages
- Ability to leverage external, MPI-based libraries
  - Easy to program a Jacobi iteration in CHAPEL
  - Write one from scratch, or make use of PETSc, elemental, etc.
- Adopting a new language like CHAPEL is not easy for real-world applications
  - Many applications have an MPI dependence
  - Users slow to adopt if performance hinges on new runtime systems
- Chapel over MPI-3
  - Reduces duplication of runtime resources
  - MPI-3 provides a much better match to Chapel’s needs than MPI-1
MPI-3 Features in CHAPEL

- MPI Windows
  - Allocation handled by MPI Implementation
  - Exposed memory shared by all processes in a communicator
Chapel Tasks

- Traditionally done with Active Messages in GASNet
- No active messages in MPI (yet)
- Our model is to use a remote queuing model using MPI RMA atomic operations
  - Lock remote target
  - Enqueue task and arguments
  - Unlock target
- Partial asynchronous progress in this model
  - Enqueue operation can be asynchronous on a good MPI RMA implementation
  - The actual execution of the task requires the target process to see the enqueued operations and execute them