Process Management Interface – Exascale
Agenda

• Since we last met…  
  ▪ Address some common questions  
  ▪ Outline PMIx standards process  

• PMIx v1.x release series  
  ▪ What has been included and planned  
  ▪ Review launch performance status  

• Roadmap  
  ▪ Features in the pipeline  
  ▪ Potential future features  

• Questions/comments/discussion
Charter?

- **Define**
  - set of agnostic APIs (not affiliated with specific model code base) to support application ↔ system mgmt software (SMS) interactions

- **Develop**
  - an open source (non-copy-left licensed) standalone “convenience” library to facilitate adoption

- **Retain**
  - transparent compatibility across all PMI/PMIx versions

- **Support**
  - the *Instant On* initiative

- **Work**
  - to define/implement new APIs for evolving programming models.
What Is PMIx?

- **Standardized APIs**
  - Four header files (client, server, common, tool)
  - Enable portability across environments
  - Support interactions between applications and system management stack
- **Convenience library**
  - Facilitate adoption
  - Serves as validation platform for standard
- **Community**
What Is PMIx?

- Standardized APIs
  - Four header files (client, server, common, tool)
  - Enable portability across environments
  - Support interactions between applications and system management stack
- Convenience library
  - Facilitate adoption
  - Serves as validation platform for standard
- Community

Not Required!
Required: Caveat

- Containerized operations
  - Require cross-boundary compatibility
  - Wireup library of containerized app must be compatible with the local resource manager
- Issues occur when migrating
  - Build under one environment using custom implementation
  - Move to another environment using different implementation
- Convenience library mitigates the problem
Why Not Part of MPI Forum?

• PMIx is agnostic
  ▪ No concept of “communicator”
  ▪ No understanding of MPI
  ▪ Used by non-MPI libraries

• Discussions underway
  ▪ Bring it into Forum process in some appropriate fashion
PMIx “Standards” Process

- Modifications/additions
  - Proposed as RFC
  - Include prototype implementation
    - Pull request to convenience library
  - Notification sent to mailing list
- Reviews conducted
  - RFC and implementation
  - Continues until consensus emerges
- Approval given
  - Developer telecon (2x/week)
PMIx Numbering

Version of Standard

Major . Minor . Release

Track convenience library revisions
Regression Testing?

- Limited direct capability
  - Run basic API tests on each PR
- Extensive embedded testing
  - Open MPI includes PMIx master, regularly updated
  - $20k^+$ tests run every night
    - Tests all spawn, wireup, publish/lookup, connect/disconnect APIs
    - Not 100% code coverage
Adoption?

- Already released
  - SLURM 16.05 (PMIx v1.1.5)
- Planned
  - IBM, Fujitsu, Adaptive Solutions, Altair, Microsoft
- Reference server
  - Provides surrogate support until native support becomes available
  - Supports full PMIx standard, limited by RM capabilities
  - Launches network of PMIx servers across allocation
Agenda

- Since we last met...
  - Address some common questions
  - Outline PMIx standards process

- PMIx v1.x release series (Artem Polyakov, Mellanox)
  - What has been included and planned
  - Review launch performance status

- Roadmap
  - Features in the pipeline
  - Potential future features

- Questions/comments/discussion
PMIx/UCX job-start use case

Hardware:
- 32 nodes
- 2 processors (Xeon E5-2680 v2)
- 20 cores per node
- 1 proc per core

Open MPI v2.1 (modified to enable ability to avoid the barrier at the end of MPI_Init)
- PMIx v1.1.5
- UCX (f3f9ad7)

* direct-fetch/async assumes no synchronization barrier inside MPI_Init.
PMIx/UCX job-start usecase

key exchange type:
- collective
- direct-fetch
- direct-fetch/async

MPI_Init (sec)

"allgatherv" on all submitted keys
Synchronization overhead
v1.2.0

• Extension of v1.1.5
  ▪ v1.1.5
    • Each proc stores own copy of data
  ▪ v1.2
    • Data stored in shared memory owned by PMIx server
    • Each proc has read-only access

• Benefits
  ▪ Minimizes memory footprint
  ▪ Faster launch times
• Server provides all the data through the shared memory
• Each process can fetch all the data with 0 server-side CPU cycles!
• In the case of direct key fetching if a key is not found in the shared memory – a process will request it from the server using regular messaging mechanism.
Server provides all the data through the shared memory
Each process can fetch all the data with 0 server-side CPU cycles!
In the case of direct key fetching if a key is not found in the shared memory – a process will request it from the server using regular messaging mechanism.
Shared memory data storage (synthetic performance test)

Hardware:
- 32 nodes
- 2 processors (Intel Xeon E5-2680 v2)
- 20 cores per node
- 1 proc per core

Shared memory data storage
(synthetic performance test)

- Open MPI v2.1
- PMIx v1.2
- 10 keys per process
- 100-element arrays of integers

https://github.com/pmix/master/tree/master/contrib/perf_tools
### Shared memory data storage (synthetic performance test) [2]

<table>
<thead>
<tr>
<th>Nodes</th>
<th>procs</th>
<th>Messages (us)</th>
<th>Shmem (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>local key</td>
<td>remote key</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>8.8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>8.9</td>
<td>9.2</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>8.8</td>
<td>9.2</td>
</tr>
<tr>
<td>8</td>
<td>160</td>
<td>8.7</td>
<td>9.2</td>
</tr>
<tr>
<td>16</td>
<td>320</td>
<td>8.4</td>
<td>9.2</td>
</tr>
<tr>
<td>32</td>
<td>640</td>
<td>8.2</td>
<td>9.2</td>
</tr>
</tbody>
</table>

### Advantages:
- Stable timings for a separate key access (no difference between local and remote key access)
- Up to 30% improvement for the remote key fetch
- Significant CPU offload on the SMP systems with large core count.
Shared memory data storage (synthetic performance test) [3]

CL1 Hardware:
- 15 nodes
- 2 processors (Intel Xeon X5570)
- 8 cores per node
- 1 proc per core

CL2 Hardware:
- 64 nodes
- 2 processors (Intel Xeon E5-2697 v3)
- 28 cores per node
- 1 proc per core
PMIx Roadmap

RM Production Releases

In Pipeline

2014


1.1.3 1.1.4 1.1.5 1.2.0

Bug fixes Bug fixes

David Solt
IBM
Tool Support

- Tool connection support
  - Allow tools to connect to local PMIx server
  - Specify system vs application
Tool Support

- **Query**
  - Network topology
    - Array of proc network-relative locations
    - Overall topology (e.g., “dragonfly”)
  - Running jobs
    - Currently executing job namespaces
    - Array of proc location, status, PID
  - Resources
    - Available system resources
    - Array of proc location, resource utilization (ala “top”)
  - Queue status
    - Current scheduler queue backlog

Examples

Debuggers?
New Flexibility

- Plugin architecture
  - DLL-based system
  - Supports proprietary binary components
  - Allows multiple implementations of common functionality
    - Buffer pack/unpack operations
    - Communications (TCP, shared memory,...)
    - Security
Obsolescence Protection

- Plugin architecture
- Cross-version support
  - Automatic detection of client/server version
  - Properly adjust for changes in structures, protocols
  - Ensure clients always get what they can understand
  - Backward support to the v1.1.5 level
Notification

- Plugin architecture
- Cross-version support
- Event notification
  - System generated, app generated
  - Resolves issues in original API, implementation
  - Register for broad range of events
    - Constrained by availability of backend support
Logging

• Plugin architecture
• Cross-version support
• Event notification
• Log data
  ▪ Store desired data in system data store(s)
    • Specify hot/warm/cold, local/remote, database and type of database, …
  ▪ Log output to stdout/err
  ▪ Supports binary and non-binary data
    • Heterogeneity taken care of for you
RM Production Releases

- 1/2016: PMIx 1.1.3
  - Bug fixes

- 6/2016: PMIx 1.1.4
  - Bug fixes

- 8/2016: PMIx 1.1.5
  - Bug fixes

- 11/2016: PMIx 1.2.0
  - Shared memory datastore

Future

Ralph Castain
Intel
Future Features

Reference Server

- Initial version: DVM
  - Interconnected PMIx servers
  - High-speed, resilient collectives
    - bcast, allgather/barrier
- Future updates: "fill" mode
  - Servers proxy clients to host RM
  - Complete missing host functionality

Winter 2017
Future Features

Debugger Support

• Ongoing discussions with MPI Forum Tools WG
  ▪ Implement proposed MPIR2 interface
  ▪ Enhance scalability

• Exploit tool connection
  ▪ Obtain proctable info
  ▪ Use PMIx_Spawn to launch daemons, auto-wireup, localize proctable retrieval

• Extend available supporting info
  ▪ Network topology, bandwidth utilization
  ▪ Event notification

Winter 2017
Future Features

Network Support Framework

• Interface to 3\textsuperscript{rd} party libraries
• Enable support for network features
  ▪ Precondition of network security keys
  ▪ Retrieval of endpoint assignments, topology
• Data made available
  ▪ In initial job info returned at proc start
  ▪ Retrieved by Query

Spring 2017
Future Features

IO Support

• Reduce launch time
  ▪ Current practices
    • Reactive cache/forward
    • Static builds
  ▪ Proactive pre-positioning
    • Examine provided job/script
    • Return array of binaries and libraries required for execution

• Enhance execution
  ▪ Request async file positioning
    • Callback when ready
  ▪ Specify persistence options

Summer 2017
Future Features

Generalized Data Store (GDS)

• Abstracted view of data store
  - Multiple plugins for different implementations
    • Local (hot) storage
    • Distributed (warm) models
    • Database (cold) storage

• Explore alternative paradigms
  - Job info, wireup data
  - Publish/lookup
  - Log

Fall 2017
Open Discussion

We now have an interface library the RMs will support for application-directed requests

Need to collaboratively define what we want to do with it

Project: https://pmix.github.io/master
Code: https://github.com/pmix