PMIx: Process Management for Exascale Environments

Ralph H. Castain, David Solt, Joshua Hursey, Aurelien Bouteiller
EuroMPI/USA 2017, Chicago, IL
What is PMIx?

PMIx is a protocol that provides a common interface for message passing and inter-process communication. It is designed to support a wide range of applications and operating systems, including MPICH, SLURM, and OSHMEM, among others.

PMIx has undergone several versions over the years, each with new features and improvements.

- **2015**: PMIx v1.2 was released, which included support for Exascale systems and improved launch times.
- **2016**: PMIx v2.x was introduced, with enhanced features like Exascale launch in under 10 seconds and improved orchestration.
- **2017**: Further developments in PMIx continue, with ongoing improvements and support for new paradigms.

PMIx provides wireup support, dynamic spawn, keyval publish/lookup, and other features to enable seamless communication across different systems and applications.

Exascale systems are on the horizon, and the launch times are expected to be significantly faster than in previous years.
Three Distinct Entities

- PMIx Standard
  - Defined set of APIs, attribute strings
  - Nothing about implementation
- PMIx Reference Library
  - A full-featured implementation of the Standard
  - Intended to ease adoption
- PMIx Reference Server
  - Full-featured “shim” to a non-PMIx RM
Traditional Launch Sequence

- **FS**: Wait for files & libs
- **WLM**: Launch Cmd
- **RM**: Spawn Procs
- **Global Xchg**
- **Barrier**
- **Proc**: Fabric
  - **NIC**: Topo

---

**PMI\times 10^{18}**
Newer Launch Sequence

FS → WLM → WLM → RM → Spawn Procs → Proc → Proc → Proc → GO

Wait for files & libs

Launch Cmd

Global Xchg

Barrier

Fabric

NIC

Topo

PMI x 10^{18}
PMIx-SMS Interactions

APP

PMIx Client

PMIx Server

RM

System Management Stack

FS

Fabric Mgr

Fabric

NIC

RAS

Job Script

Tool Support

OpenMP

MPI

Orchestration Requests

Responses

Tool Support

Job Script

PMIx x 10^{18}
PMIx Launch Sequence

*RM daemon, mpirun-daemon, etc.
PMIx/SLURM*

Performance papers coming in 2018!

*LANL/Buffy cluster, 1ppn  **PMIx Reference Server v2.0, direct-fetch/async
Similar Requirements

- Notifications/response
  - Errors, resource changes
  - Negotiated response
- Request allocation changes
  - shrink/expand
- Workflow management
  - Steered/conditional execution
- QoS requests
  - Power, file system, fabric

Multiple, use-specific libs?
(difficult for RM community to support)

Single, multi-purpose lib?
PMIx “Standards” Process

• Modifications/additions
  ▪ Proposed as RFC
  ▪ Include prototype implementation
    • Pull request to reference library
  ▪ Notification sent to mailing list

• Reviews conducted
  ▪ RFC and implementation
  ▪ Continues until consensus emerges

• Approval given
  ▪ Developer telecon (weekly)

Standards Doc under development!
Philosophy

• Generalized APIs
  ▪ Few hard parameters
  ▪ “Info” arrays to pass information, specify directives

• Easily extended
  ▪ Add “keys” instead of modifying API

• Async operations

• Thread safe

• SMS always has right to say “not supported”
  ▪ Allow each backend to evaluate what and when to support something
• Generalized APIs
  ▪ Few hard parameters
  ▪ “Info” arrays to pass information, specify directives

• Easily extended
  ▪ Add “keys” instead of modifying API

• Async operations
• Thread safe
• SMS always has right to say “not supported”
  ▪ Allow each backend to evaluate what and when to support something
Current Support

• Typical startup operations
  - Put, get, commit, barrier, spawn, [dis]connect, publish/lookup

• Tool connections
  - Debugger, job submission, query

• Generalized query support
  - Job status, layout, system data, resource availability

• Event notification
  - App, system generated
  - Subscribe, chained
  - Pre-emption, failures, timeout warning, …

• Logging (job record)
  - Status reports, error output

• Flexible allocations
  - Release resources, request resources
Event Notification Use Case

- Fault detection and reporting w/ULFM MPI
  - ULFM MPI is a fault tolerant flavor of Open MPI
- Failures may be detected from the SMS, RAS, or directly by MPI communications
- Components produce a PMIx event when detecting an error
- Fault Tolerant components register for the fault event
- Components propagate fault events which are then delivered to registered clients
In Pipeline

- Network support
  - Security keys, pre-spawn local driver setup, fabric topology and status, traffic reports, fabric manager interaction

- Obsolescence protection
  - Automatic cross-version compatibility
  - Container support

- Job control
  - Pause, kill, signal, heartbeat, resilience support

- Generalized data store

- File system support
  - Dependency detection
  - Tiered storage caching strategies

- Debugger/tool support++
  - Automatic rendezvous
  - Single interface to all launchers
  - Co-launch daemons
  - Access fabric info, etc.

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

Need to collaboratively define what we want to do with it

Project: https://pmix.github.io/pmix
Reference Implementation: https://github.com/pmix/pmix
Reference Server: https://github.com/pmix/pmix-reference-server