PMIx: Storage Integration
Origin: Changing Landscape

Launch time limiting scale

Programming model & runtime proliferation

Legion

Hybrid applications

Model-specific tools

Container technologies

PMI \times 10^{18}
• Resolve launch scaling
  ▪ Pre-load information known to RM/scheduler
  ▪ Pre-assign communication endpoints
  ▪ Eliminate data exchange during init
  ▪ Orchestrate launch procedure
PMIx Launch Sequence

*RM daemon, mpirun-daemon, etc.
• Async event notification
• Cross-model notification
  ▪ Announce model type, characteristics
  ▪ Coordinate resource utilization, programming blocks
• Generalized tool support
  ▪ Co-launch daemons with job
  ▪ Forward stdio channels
  ▪ Query job, system info, network traffic, process counters, etc.
  ▪ Standardized attachment, launch methods
Sprinkle Some Magic Dust

• Allocation support
  - Dynamically add/remove/loan nodes
  - Register pre-emption acceptance, handshake

• Dynamic process groups
  - Async group construct/destruct
  - Notification of process departure/failure

• File system integration
  - Pre-cache files, specify storage strategies
Baseline Vision

• Tiered storage
  ▪ Parallel file system
  ▪ Caches at IO server, switches, cabinets, …
  ▪ Caches hold images, files, executables, libraries, checkpoints

• Bits flow in all directions
  ▪ Stage locations prior to launch
  ▪ Movement in response to faults, dynamic workflow, computational stages
Planned Support

System PMIx server

 Parse for dependencies

Current data map
Usage patterns
Authorization

Query

Retrieval time

User-specified caching, dependencies (data & libs), persistence

Job Script
Planned Support

System PMIx server → WLM

Job Script

Compute

Pre-stage (images, binaries, libs)
Allocation schedule

Allocate
Launch app
Planned Support

System PMIx server

WLM

Orchestration requests

Local PMIx Server

Compute

Job Script
Push Forward?

https://pmix.org  https://github.com/pmix
PMIx: Process Management for Exascale Environments

Ralph H. Castain\textsuperscript{a}, Aurelien Bouteiller\textsuperscript{b,1}, Joshua Hursey\textsuperscript{c}, David Solt\textsuperscript{c}

\textsuperscript{a}Intel, Inc.
\textsuperscript{b}The University of Tennessee, Knoxville
\textsuperscript{c}IBM

Abstract

High-Performance Computing (HPC) applications have historically executed in static resource allocations, using programming models that ran independently from the resident system management stack (SMS). Achieving exascale performance that is both cost-effective and fits within site-level environmental constraints will, however, require that the application and SMS collaboratively orchestrate the flow of work to optimize resource utilization and compensate for on-the-fly faults. The Process Management Interface - Exascale (PMIx) community is committed to establishing scalable workflow orchestration by defining an abstract set of interfaces by which not only applications and tools can interact with the resident SMS, but also the various SMS components can interact with each other. This paper presents a high-level overview of the goals and current state of the PMIx standard, and lays out a roadmap for future directions.


https://doi.org/10.1016/j.parco.2018.08.002