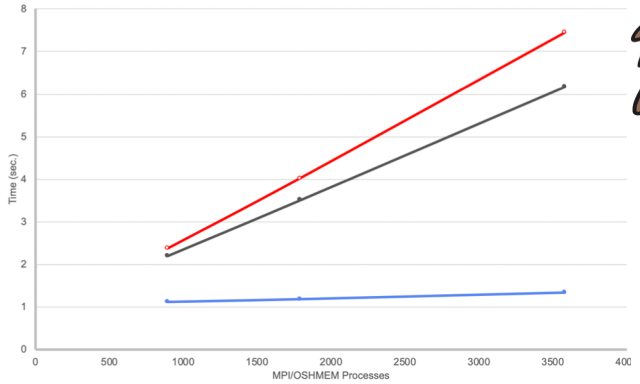
PMIx: Bridging the Container Boundary

Ralph H. Castain
Intel

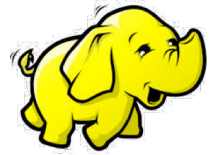
Origin: Changing Landscape

Launch time limiting scale



Programming model & runtime proliferation

Legion

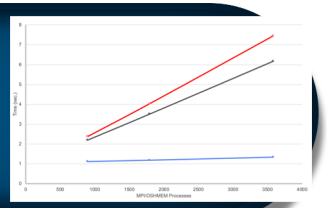


Hybrid applications

Model-specific tools

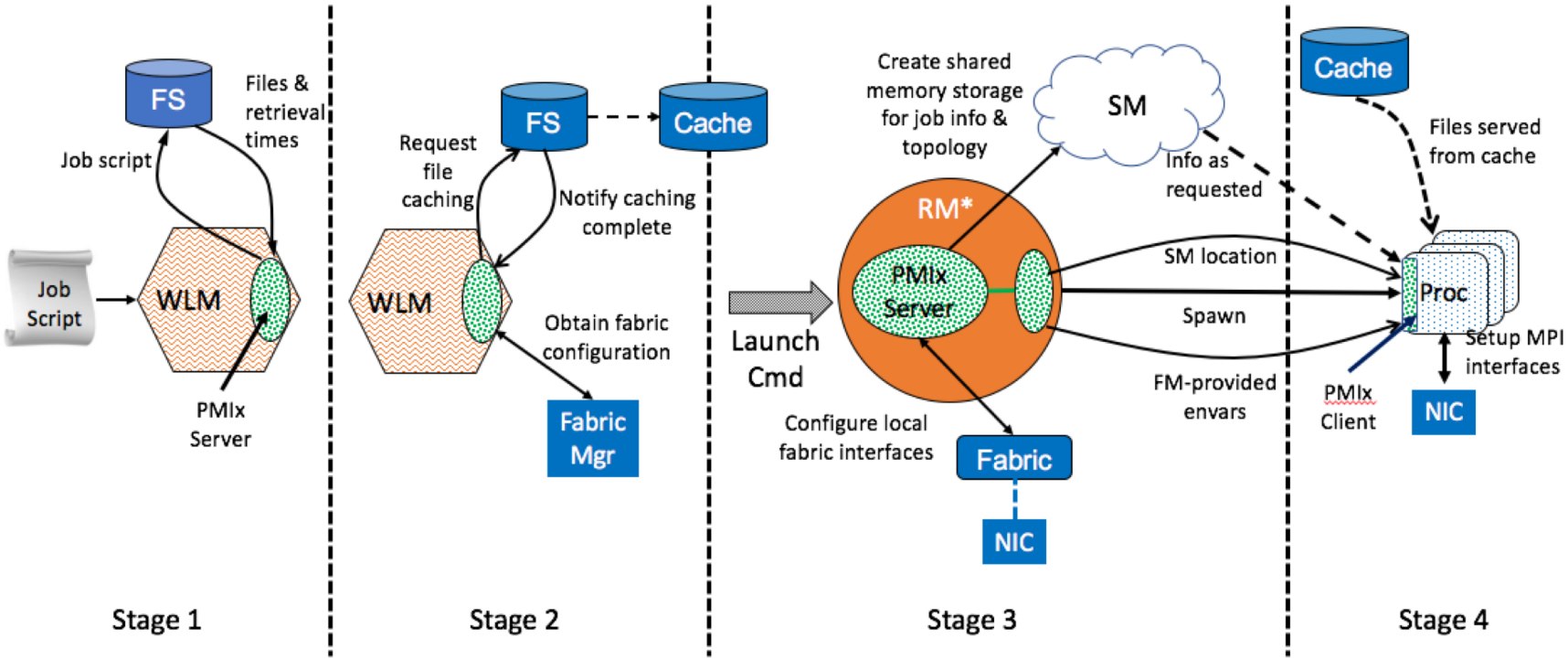
Container technologies

Start Someplace!



- 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.

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 RTE
  - Full-featured “shim” to a non-PMIx RM
  - Provides development environment

v3.1 just released!

Where Is It Used?

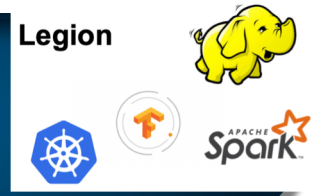
- Libraries
  - OMPI, MPICH, Intel MPI, HPE-MPI, Spectrum MPI, Fujitsu MPI
  - OSHMEM, SOS, OpenSHMEM, …
- RMs
  - Slurm, Fujitsu, IBM's JSM, PBSPro (2019), Kubernetes(?)
  - Slurm enhancement (LANL/ECP)

- New use-cases
  - Spark, TensorFlow
  - Debuggers (TotalView, DDT)
  - MPI
    - Re-ordering for load balance (UTK/ECP)
    - Fault management (UTK)
    - On-the-fly session formation/teardown (MPIF)
  - Logging information
  - Containers
    - Singularity, Docker, Amazon

Build Upon It

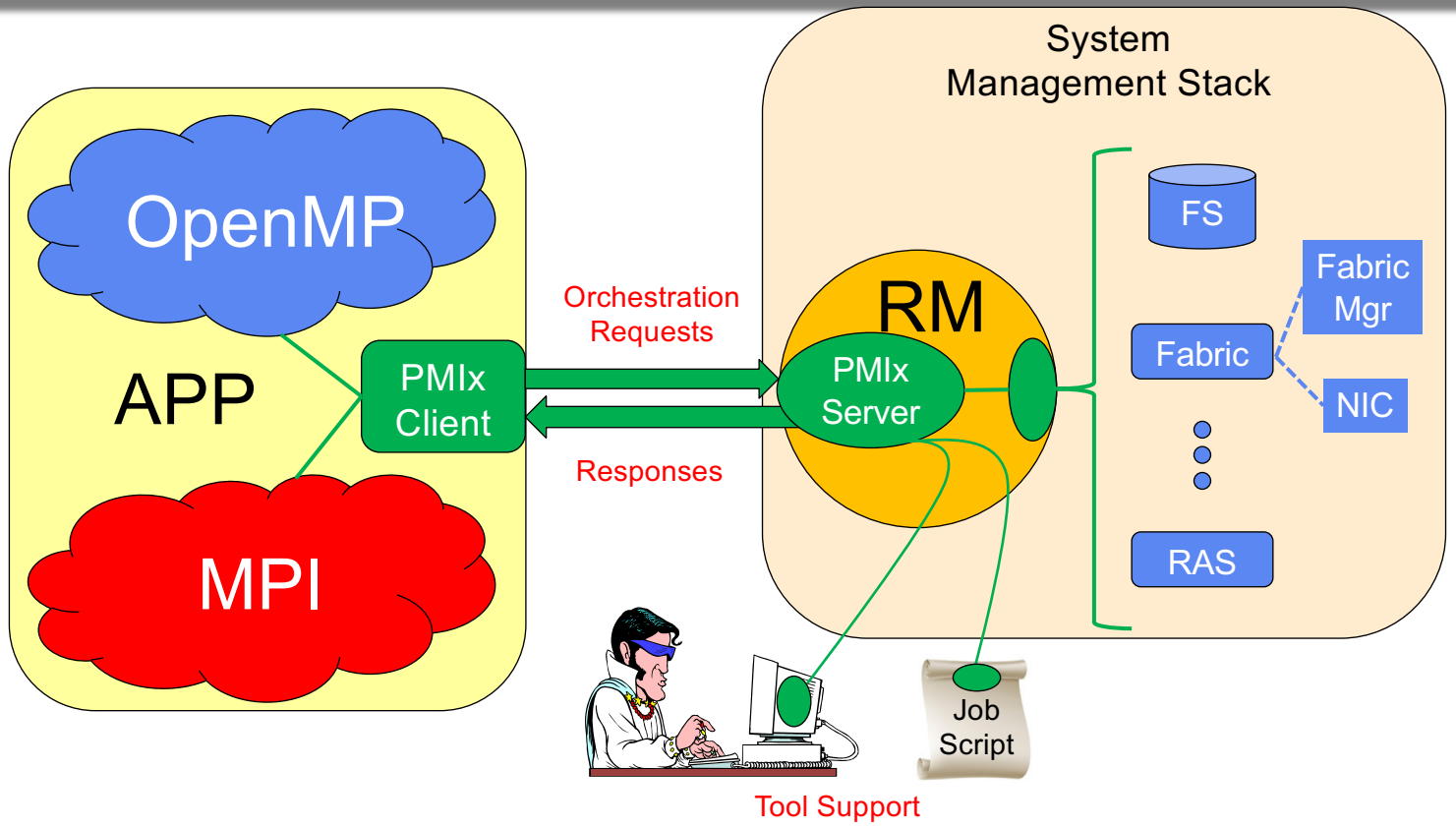
- Async event notification
- Cross-model notification
  - Announce model type, characteristics
  - Coordinate resource utilization, programming blocks
- Generalized tool support
  - Co-launch daemons with job
  - Forward stdio channels
  - Query job, system info, network traffic, process counters, etc.
  - Standardized attachment, launch methods

Sprinkle Some Magic Dust

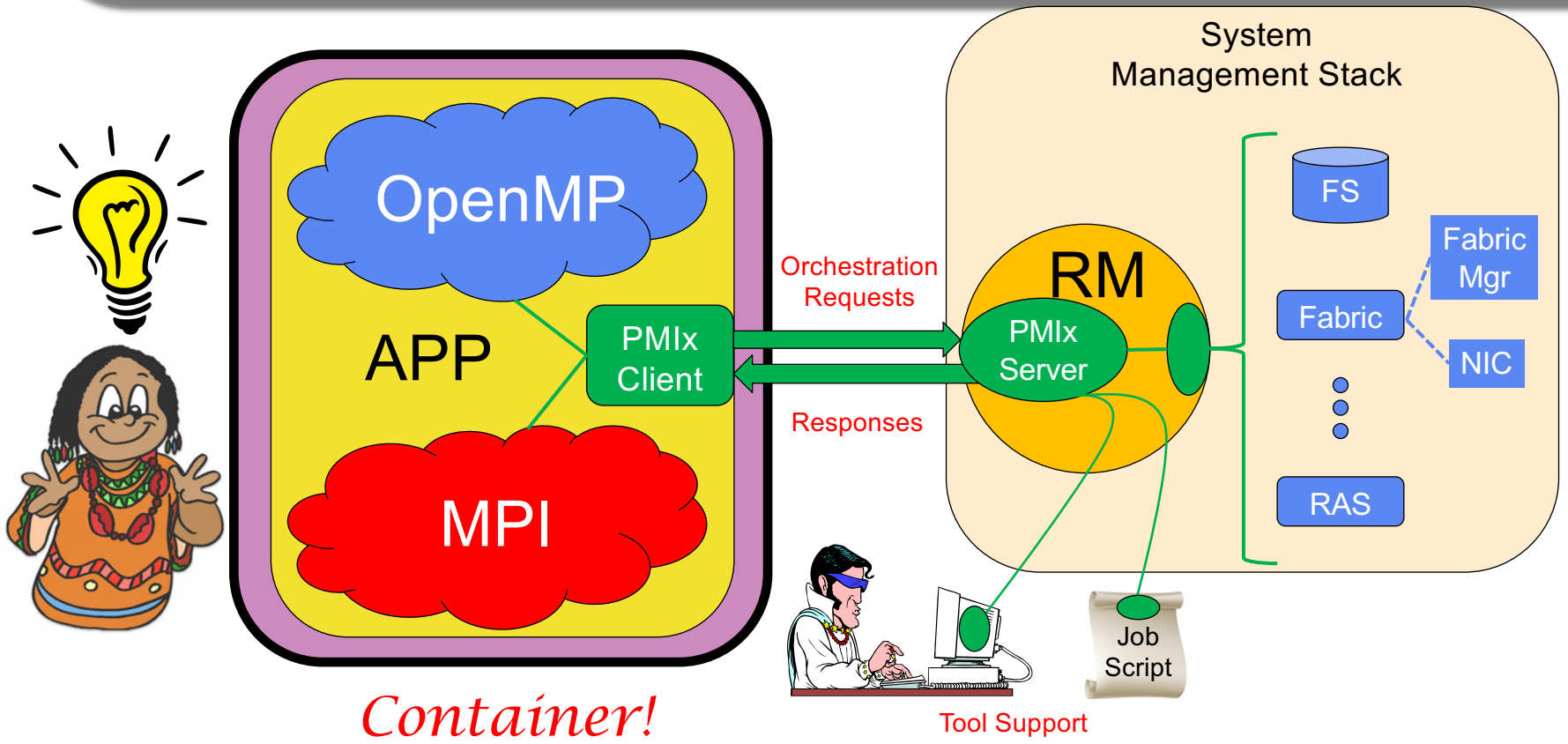


- Allocation support
  - Dynamically add/remove/loan nodes
  - Register pre-emption acceptance, handshake
- Dynamic process groups
  - Async group construct/destruct
  - Notification of process departure/failure
- File system integration
  - Pre-cache files, specify storage strategies

PMIx-SMS Interactions



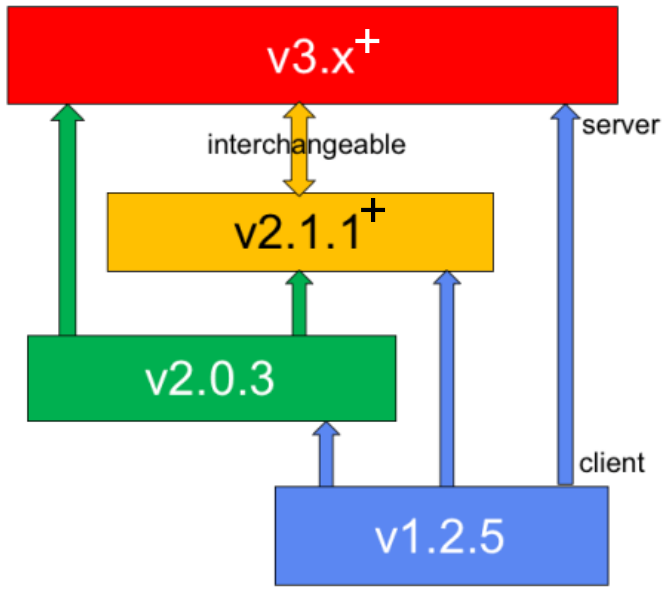
PMIx-SMS Interactions



Container Issues

- Version tracking across container boundary
  - Different pieces moving at different rates
- Container managers vs HPC schedulers
  - Dynamic, service related vs static, application focus
- Uneven adoption rates
  - Different environments adopt features at different times, different combinations

Version Tracking

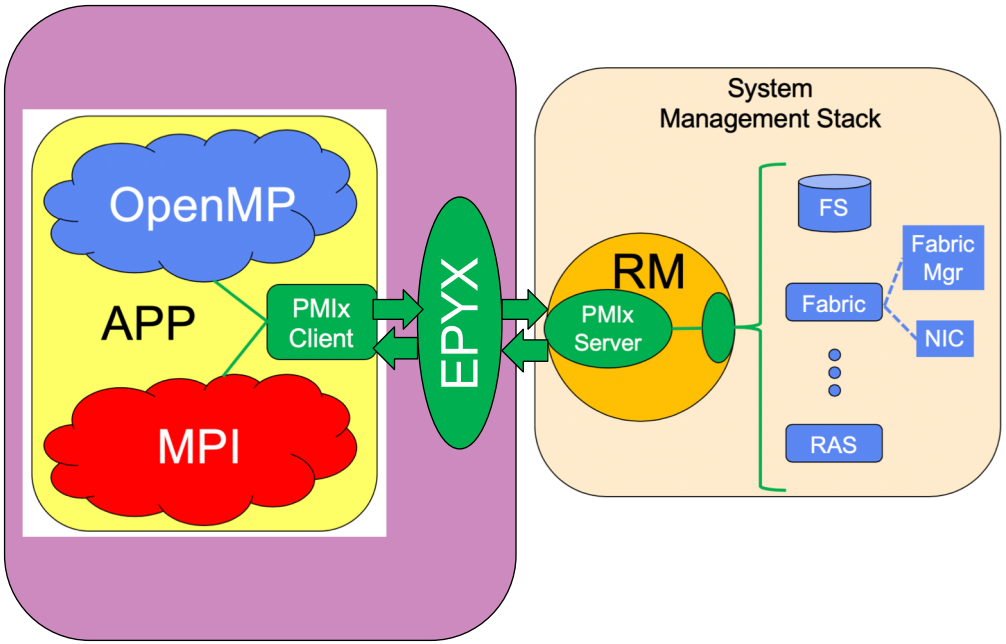


- Auto-negotiate messaging protocol
- Client starts
  - Envar indicates server capabilities
  - Select highest support in common
  - Convey selection in connection handshake
- Server follows client's lead
  - Per-client messaging protocol
  - Support mix of client versions

Container Issues

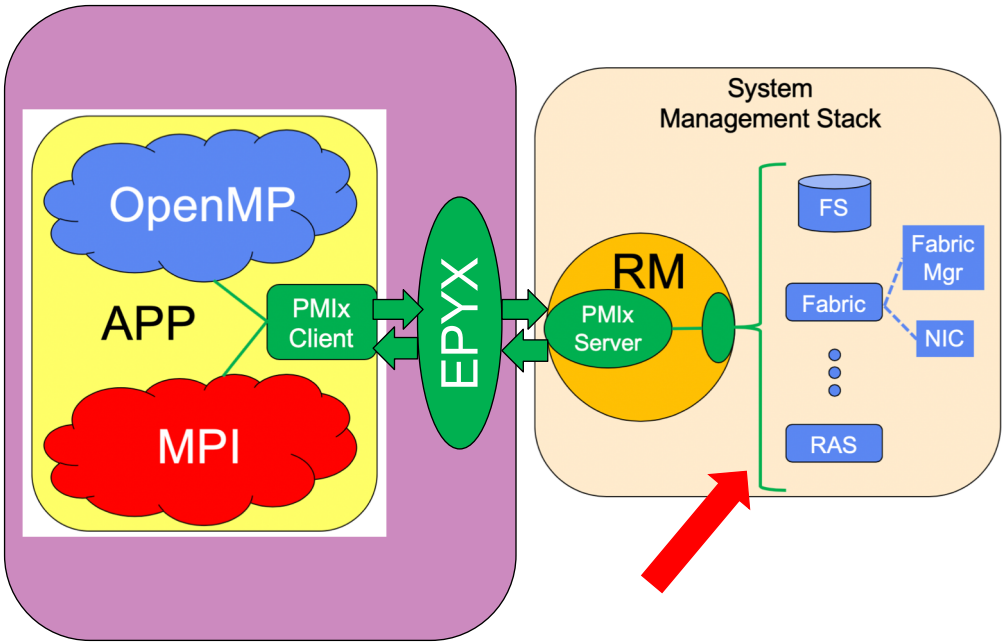
- Version tracking across container boundary
  - Different pieces moving at different rates
- Uneven adoption rates
  - Different environments adopt features at different times, different combinations
- Container managers vs HPC schedulers
  - Dynamic, service related vs static, application focus
  - Mismatched capabilities

EPYX



- PMIx relay daemon/server
- Integrated into container
- Sense what SMS supports
  - From nothing to everything
- Supported requests
  - Relay requests/responses
- Unsupported requests
  - Execute internally
  - Return “not supported”

EPYX

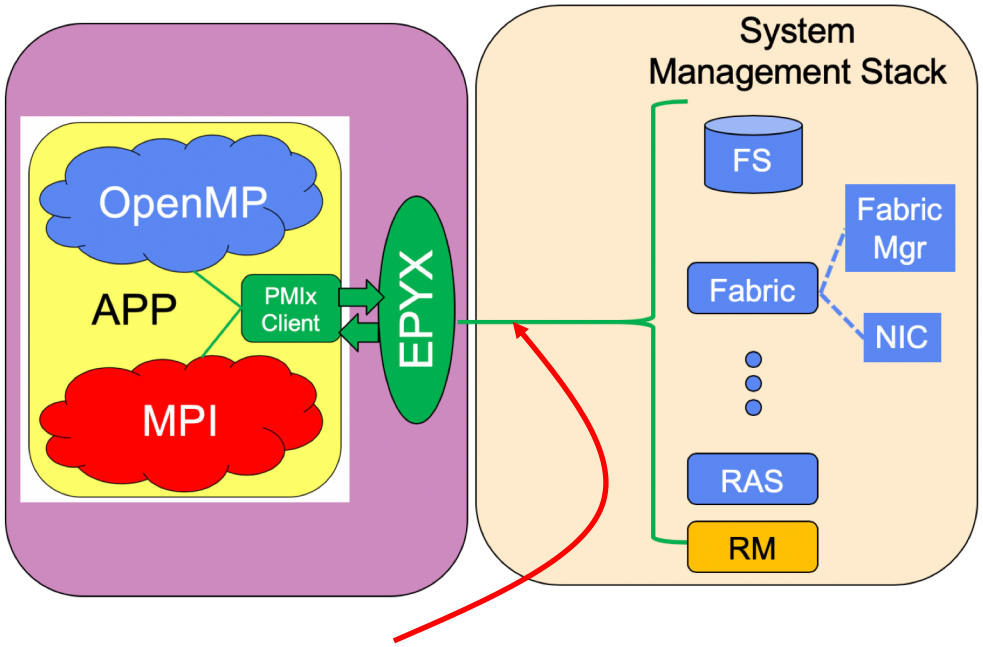


*RM can perform request, but doesn't have PMIx backend support for it

- PMIx relay daemon/server
- Integrated into container
- Sense what SMS [illegible] ports
  - From nothi[illegible]ning
- Suppo[illegible]quests
  - Rela[illegible]equests/responses
- Unsupported requests
  - Execute internally
  - Return "not supported"

Capable?

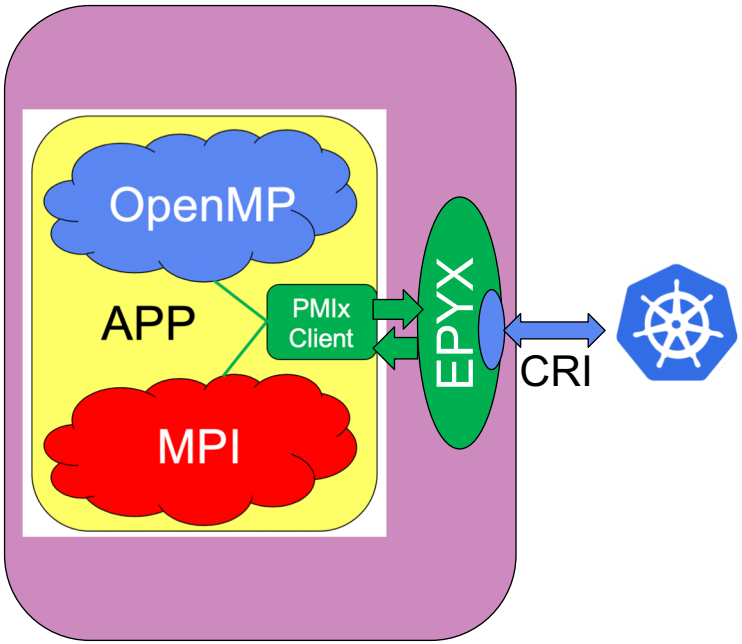
EPYX: Filling the Gaps



Who writes these drivers?

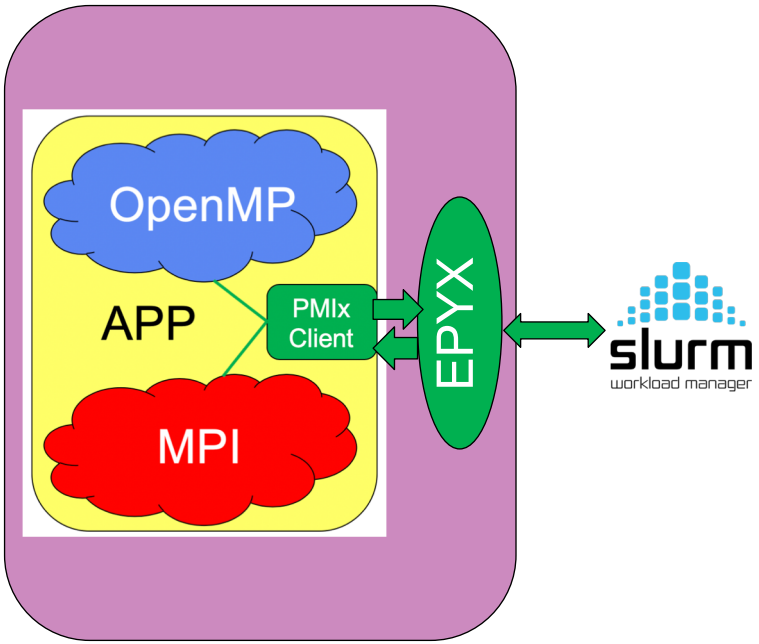
- Call intrinsic APIs to execute PMIx requests from client
- Treat the RM as an equal member of SMS
- Pros
  - Allows more transparent movement of containers across systems
  - Reduces obstacles
- Cons
  - Reduces pressure on SMS vendors to integrate

HPC on Container Mgrs



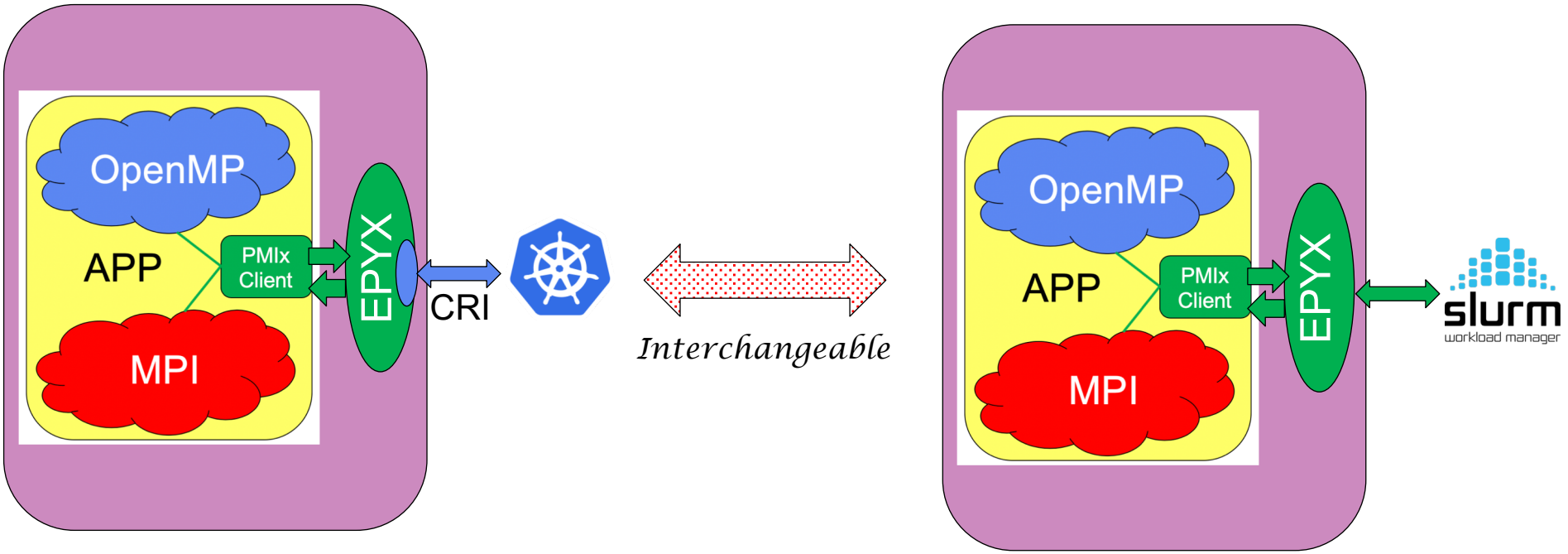
- Allocation request
  - Stabilize allocation for some period of time
- Event notification
  - Handshake need to break commitment
  - Notify when restored
  - Use new FT/Sessions methods for flexible members

Services on HPC Systems



- Job control API
  - Register as a service
  - Request auto-restart, multiple replicas
  - Setup parallel duplicate IO streams
- IO Forwarding APIs
  - Construct data flows between processes
- Storage APIs
- Publish/Lookup APIs
  - Service discovery, rendezvous

Why Enable This?



Summary

- Avoid having to write entire runtimes just to do something a little different
- Portability (HPC ⟺ Service Mgrs)
- Generalized tools
- Scalable operations
- Async event notification
- Full system orchestration

Come Join Us!

Slack: pmix-workspace.slack.com https://pmix.org https://github.com/pmix

Q&A

Useful Links:

General Information: https://pmix.org/

PMIx Library: https://github.com/pmix/pmix

PMIx Reference RunTime Environment (PRRTE): https://github.com/pmix/prrte

PMIx Standard: https://github.com/pmix/pmix-standard

Slack: pmix-workspace.slack.com

Overview Paper

PMIx: Process Management for Exascale Environments

Ralph H. Castain[a], Aurelien Bouteiller[b,1], Joshua Hursey[c], David Solt[c]

[a] *Intel, Inc.*
[b] *The University of Tennessee, Knoxville*
[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.

Ralph H. Castain, Aurelien Bouteiller, Joshua Hursey, David Solt, “PMIx: Process management for exascale environments”, Parallel Computing, 2018.

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