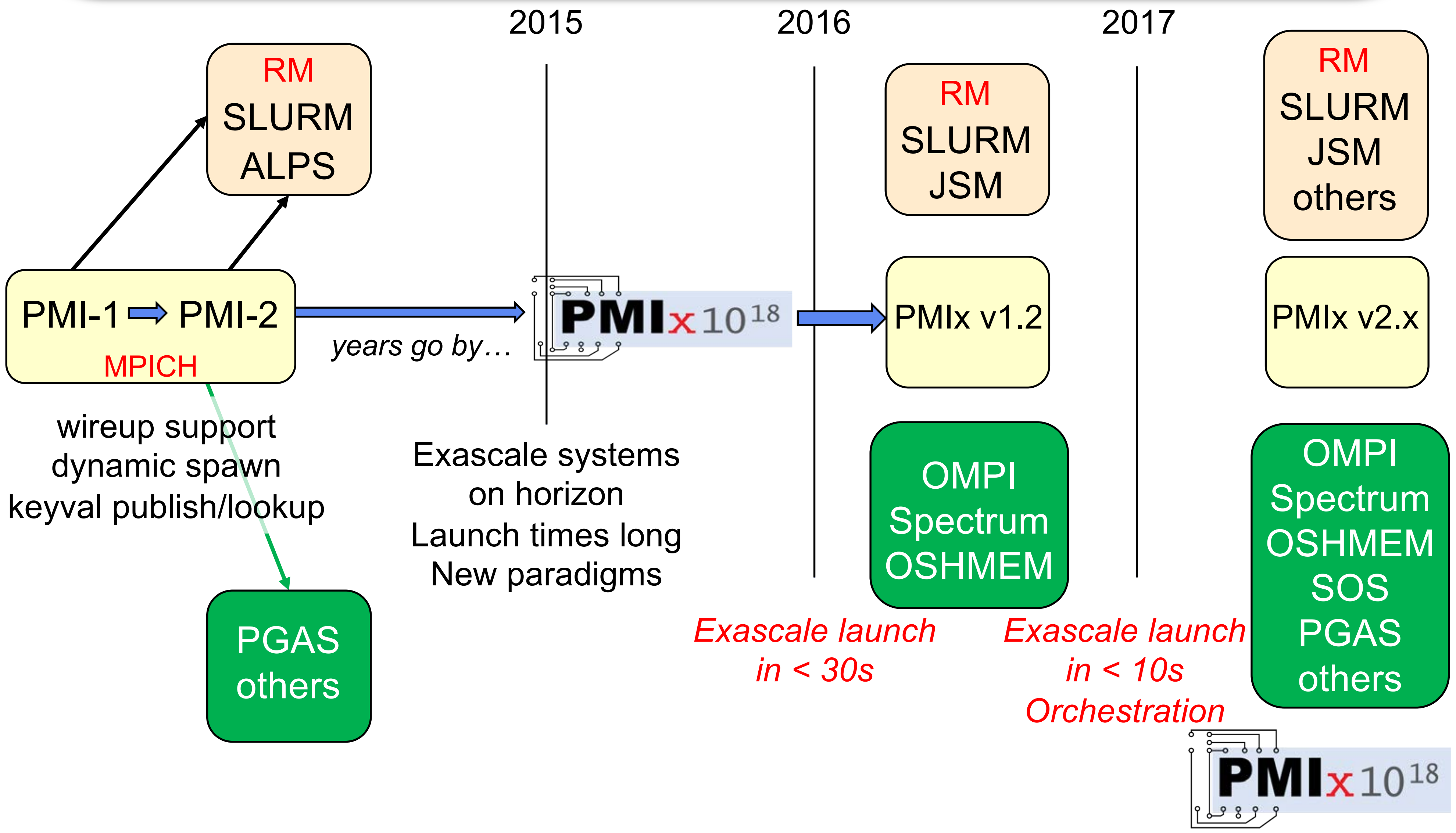
# PMIx: Process Management for Exascale Environments

**Ralph H. Castain**, David Solt, Joshua Hursey, Aurelien Bouteiller

EuroMPI/USA 2017, Chicago, IL

# What is PMIx?


2015
2016
2017
RM
SLURM
ALPS
PMI-1 → PMI-2
MPICH
years go by…
PMIx 10^18
wireup support
dynamic spawn
keyval publish/lookup
PGAS
others
Exascale systems
on horizon
Launch times long
New paradigms
RM
SLURM
JSM
PMIx v1.2
OMPI
Spectrum
OSHMEM
Exascale launch
in < 30s
Exascale launch
in < 10s
Orchestration
RM
SLURM
JSM
others
PMIx v2.x
OMPI
Spectrum
OSHMEM
SOS
PGAS
others
PMIx 10^18

## 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

## The Community

intel

Mellanox TECHNOLOGIES

IBM

Altair

ICL UT

SchedMD

RIST

FUJITSU

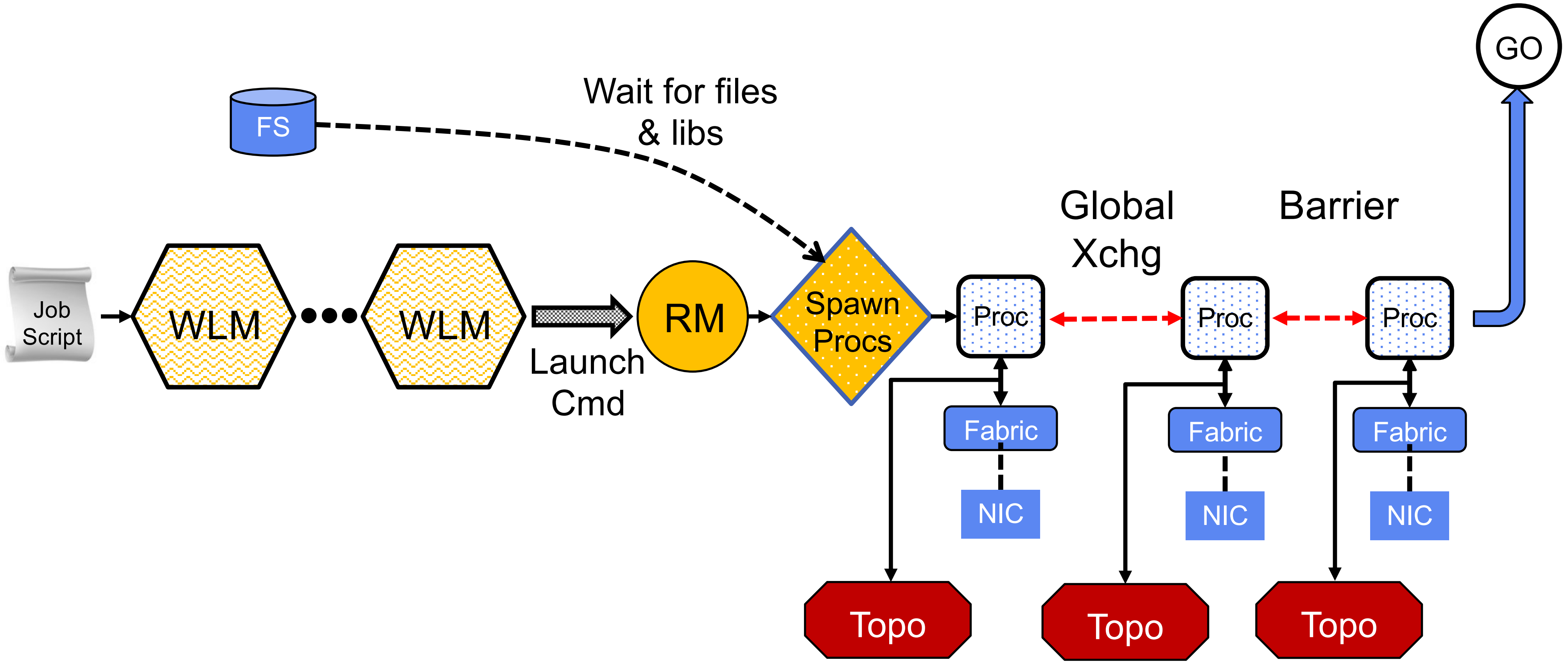
OAK RIDGE National Laboratory

Los Alamos NATIONAL LABORATORY EST. 1943

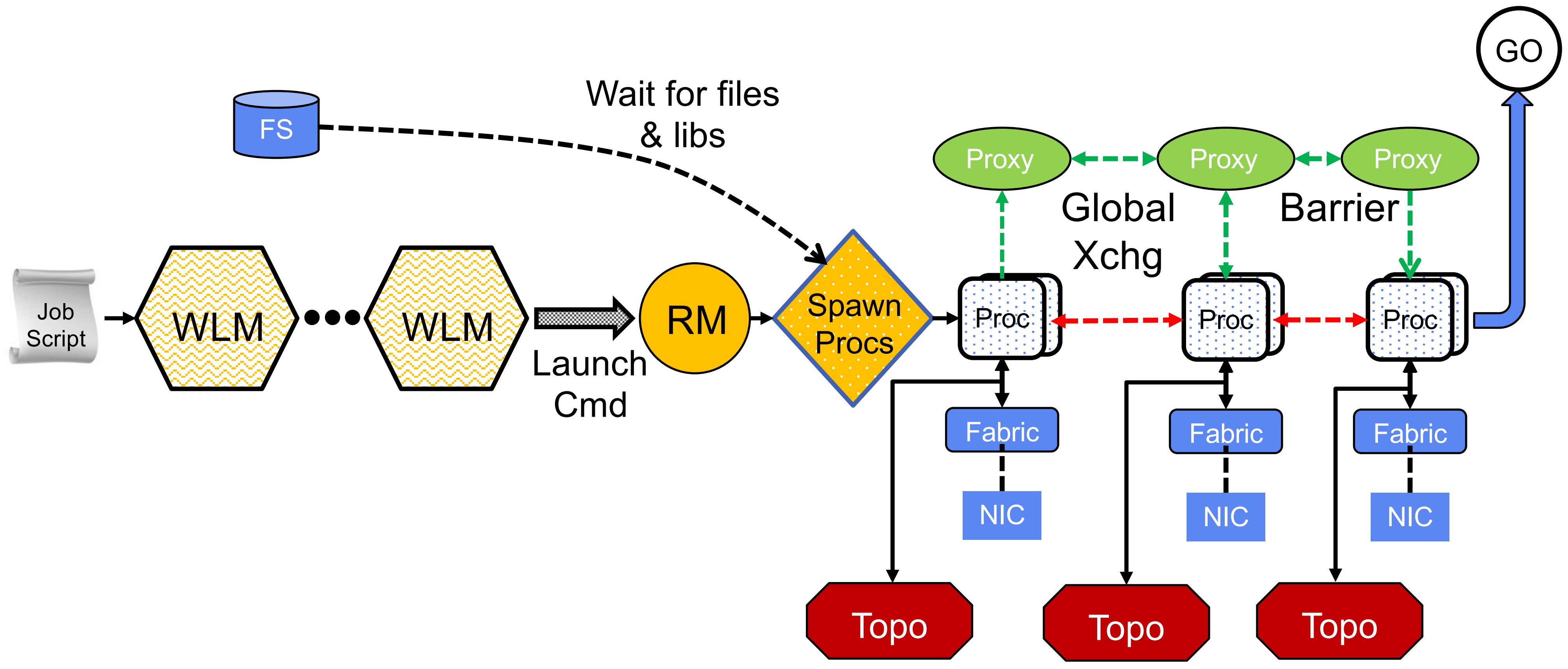
https://pmix.github.io/pmix
https://github.com/pmix

PMIx $10^{18}$

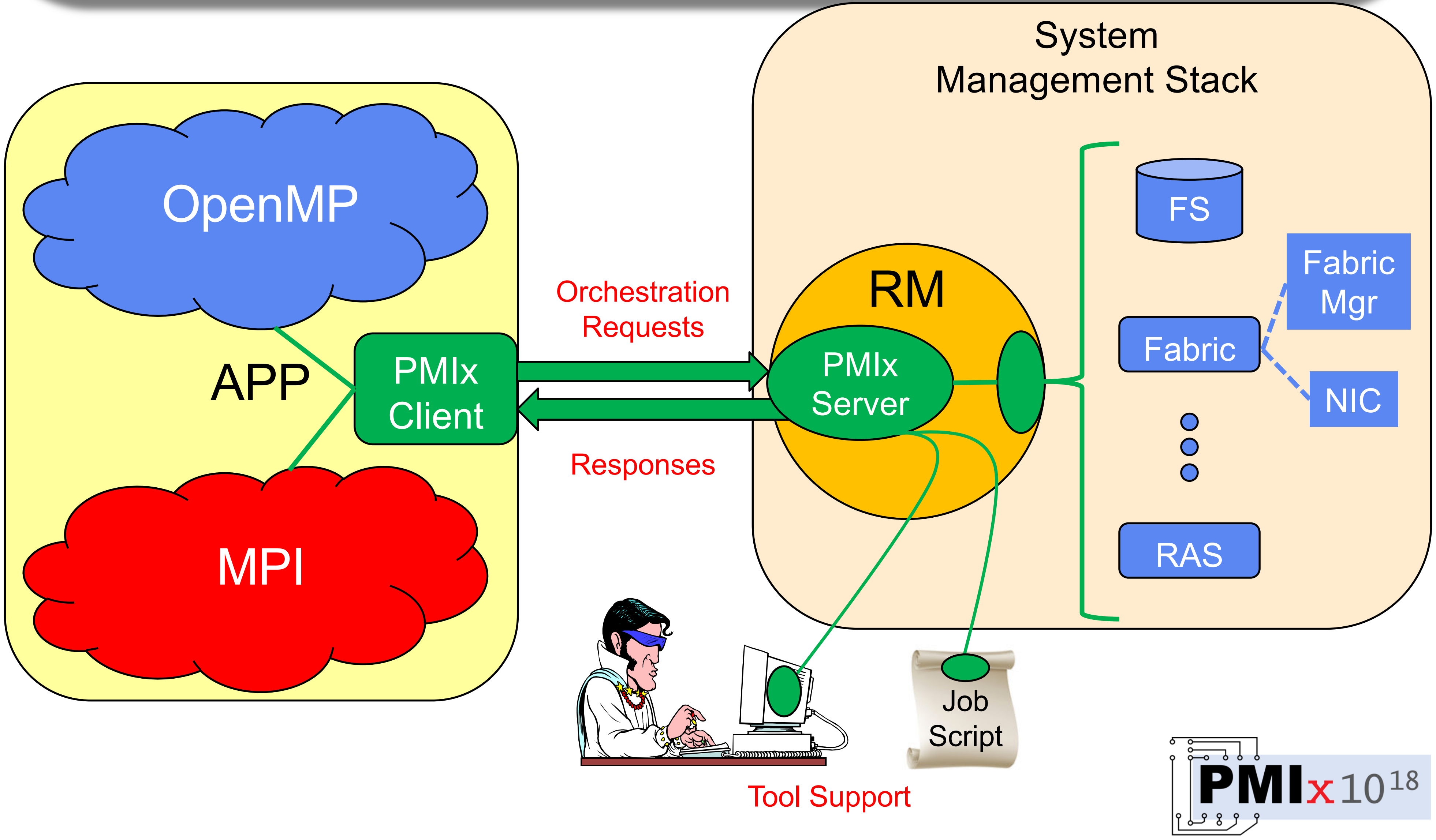
# Traditional Launch Sequence



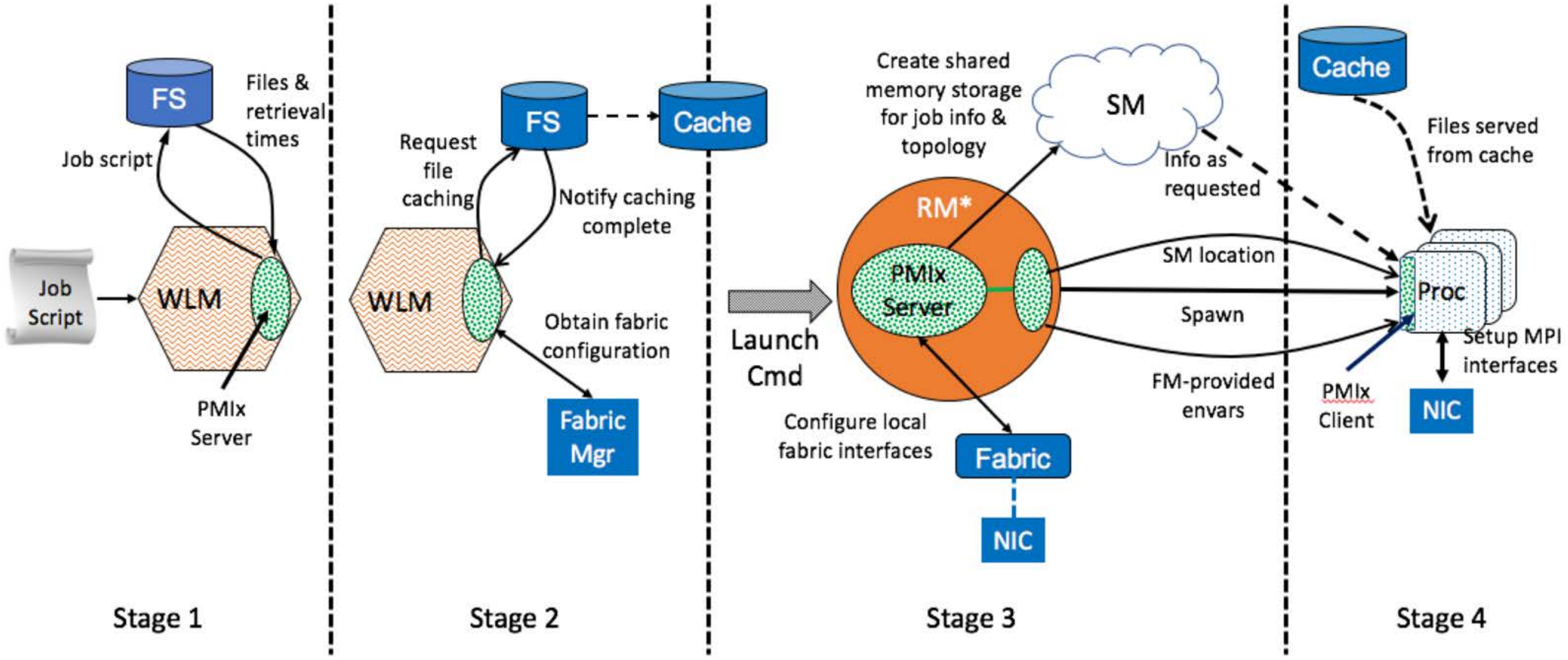
# Newer Launch Sequence



# PMIx-SMS Interactions



# PMIx Launch Sequence

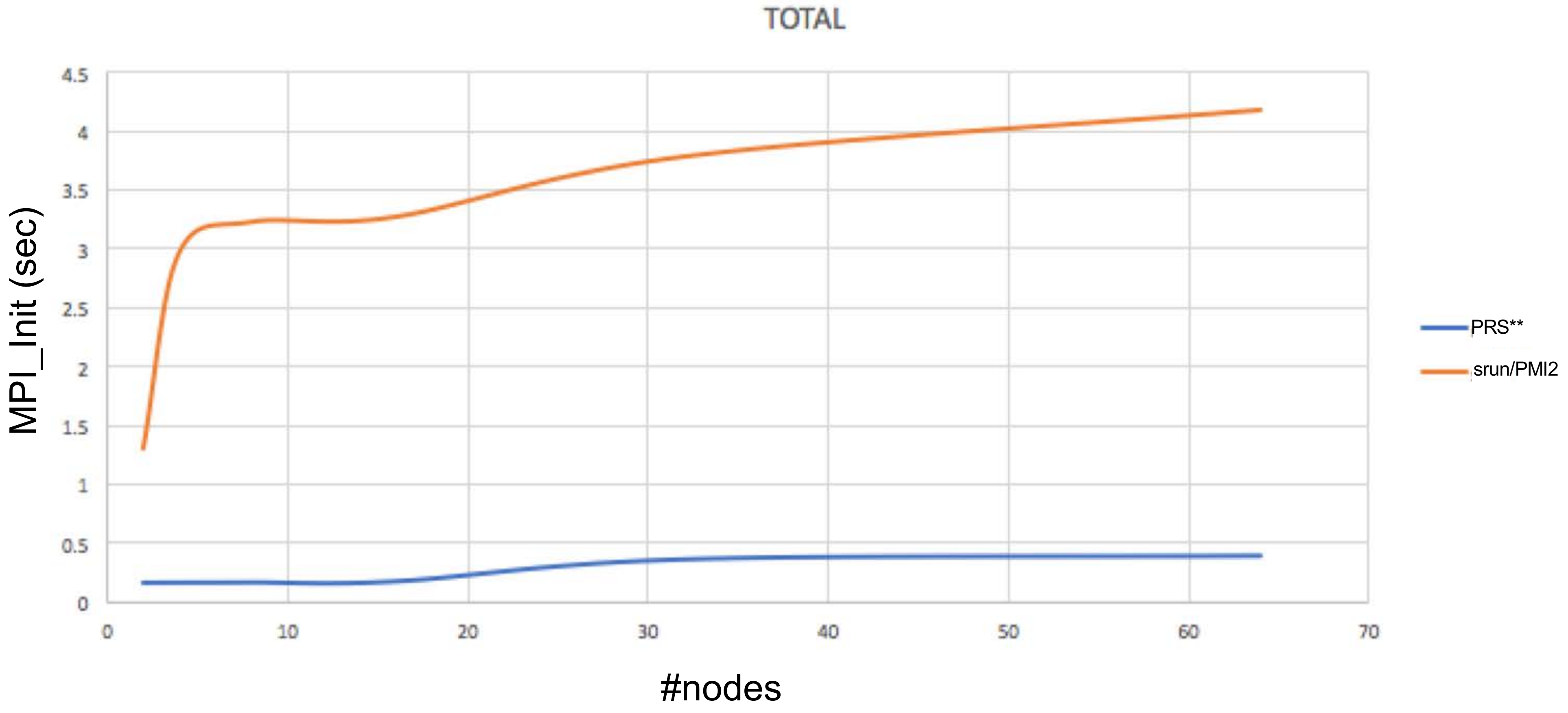


*RM daemon, mpirun-daemon, etc.

PMIx $10^{18}$

## 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 $10^{18}$

## 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)

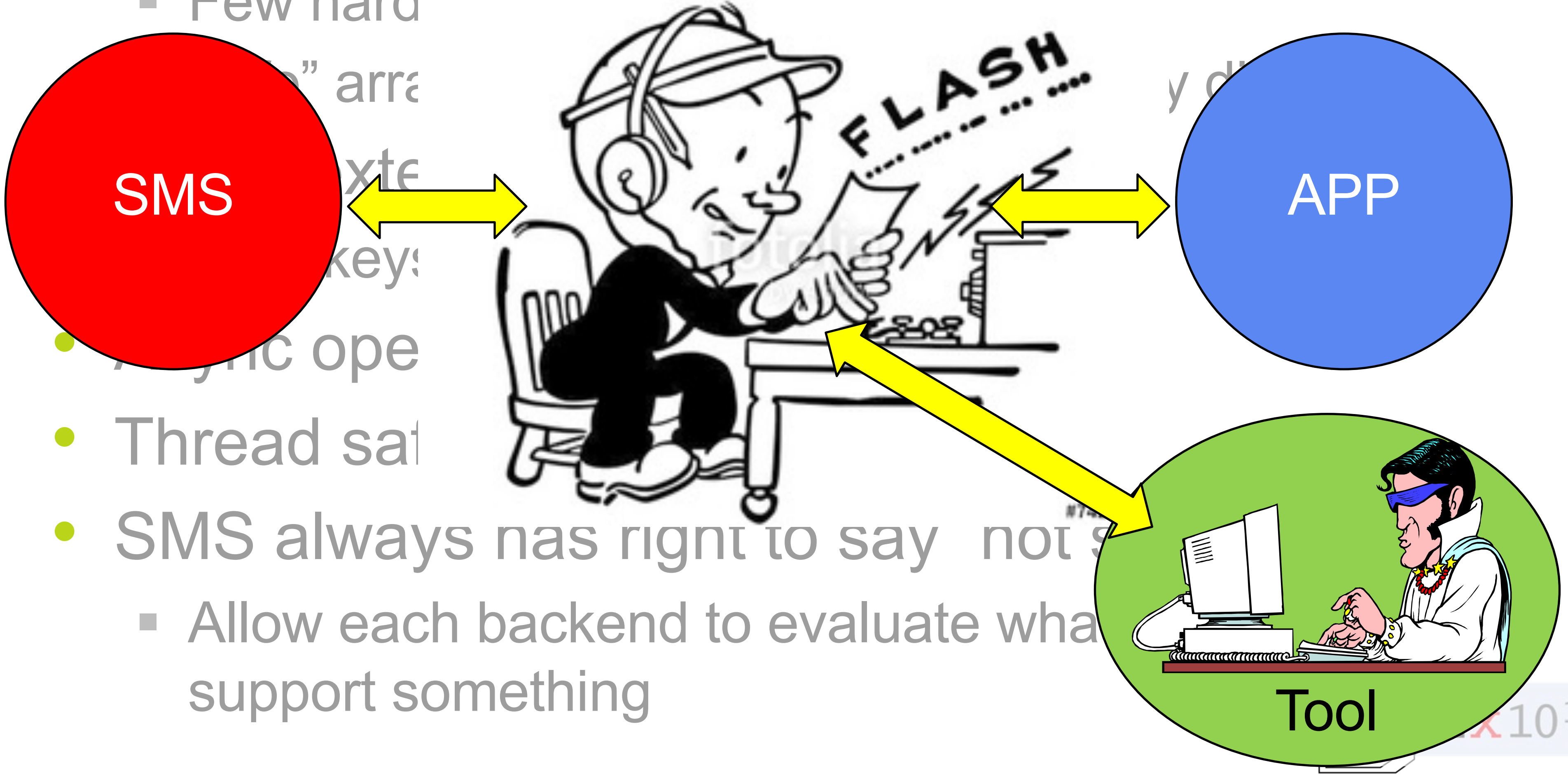
*Standards Doc under development!*

## 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

# Messenger not Doer

- Generalized APIs
  - Few hard parameters
  - ” arra y d
  - xte
  - keys
- ync ope
- Thread saf
- SMS always has right to say “not s
  - Allow each backend to evaluate wha support something

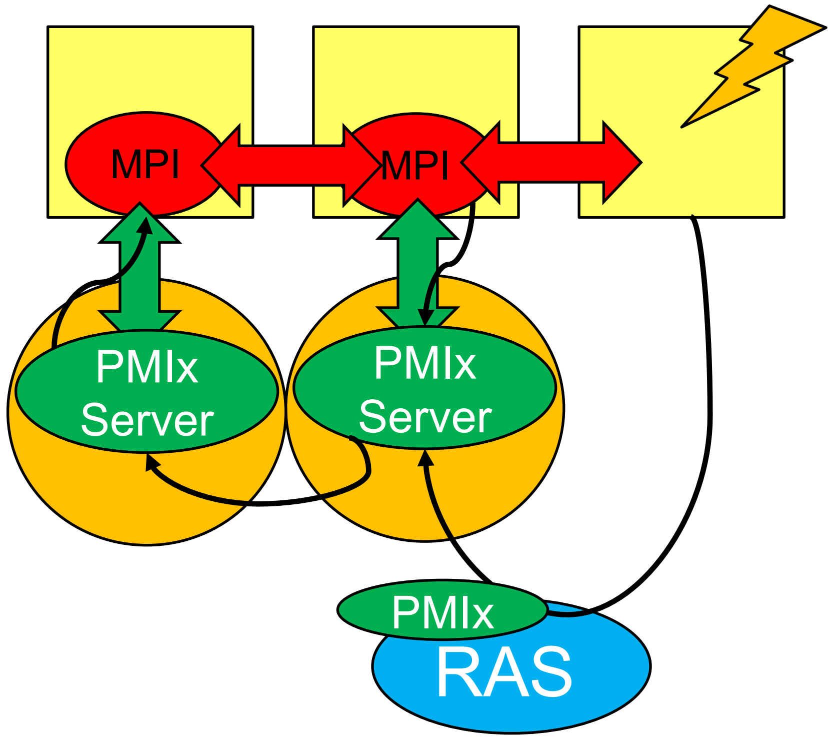


## Current Support

- 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

## Summary

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

PMIx $10^{18}$