PMIx: A *very* Brief Overview

Joshua Hursey

IBM

https://pmix.org

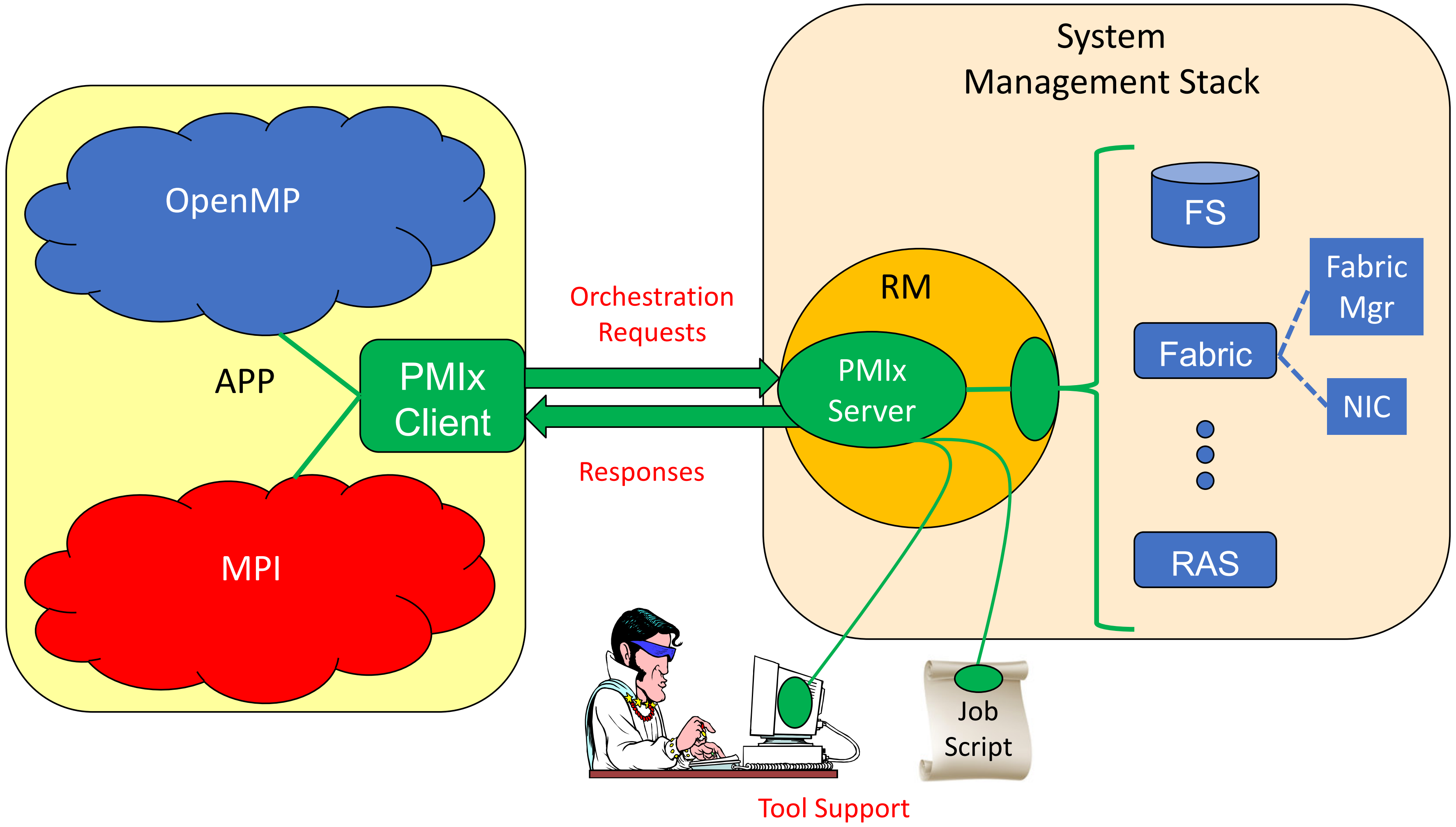
Overview

- Define a standard way for an application to communicate to the environment around it.
- Evolution from PMI-1 and PMI-2 (not a fork – maintain support)
  - Flexible APIs – key/value attributes and directives
  - New APIs for asynchronous events, dynamic resource management, alternative ‘fence’ behaviors, job-level data, …
  - Introduce a server-side of the interface to ease adoption, and provide a focused area for cross-version support
- Maintain a set of standard documents describing the syntax and semantics of the PMIx interface.
- Maintain reference implementations of the PMIx library (libpmix) and a reference runtime environment (PRRTE).

Working Model

- Interface with 4 dimensions:
  - Client side (inside app)
  - Server side (providing cross-node support, and interaction with SMS)
  - Tool side (external to job)
  - Cross-version support between the client and server (inside PMIx library implementation)

PMIx is a messenger, not a doer



This is the conceptual model used when discussing proposals to the PMIx standard

Strengths

- Job level information available after PMIx_Init()
  - No need to synchronize to get basic job information
- Flexible PMIx_Fence() behavior:
  - Full Modex: Fully exchange all PMIx_Put() data before returning
  - Direct Modex: Ensure all PMIx_Put() data is accessible before returning (might require an RPC during PMIx_Get() to ’pull’ data)
  - Sync: Simply synchronize processes without data exchange
- Allows for multiple programming models to interact through the PMIx interface.
- Cross-version support between the client and server is important for application contained container support models (e.g., Singularity)

Places for improvement

- Better facilitate documentation of Client requirements and RM provided features
  - What is the minimal set of PMIx’s interfaces/attributes does a RM need to support programming model X (e.g., OpenSHMEM, MPI, …)
- Community participation:
  - Handful of active participants, many more interacting on the sidelines
- Speed of development:
  - Concern that we are “moving too fast”
- Standard is too specific to the reference implementations
  - Continue to work to generalize the standard document.

Adoption Updates

- MPI use-cases
  - Re-ordering for load balance (UTK/ECP)
  - Fault management (UTK)
  - On-the-fly session formation/teardown (MPIF)
- MPI libraries
  - OMPI, MPICH, Intel MPI, HPE-MPI, Spectrum MPI, Fujitsu MPI
- Resource Managers (RMs)
  - Slurm, Fujitsu,
    IBM’s Job Step Manager (JSM),
    PBSPro (2019), Kubernetes(?)
- Spark, TensorFlow
  - Just getting underway
- OpenSHMEM
  - Multiple implementations

PMIx 10^{18}

Q&A

Useful Links:

General Information: https://pmix.org/

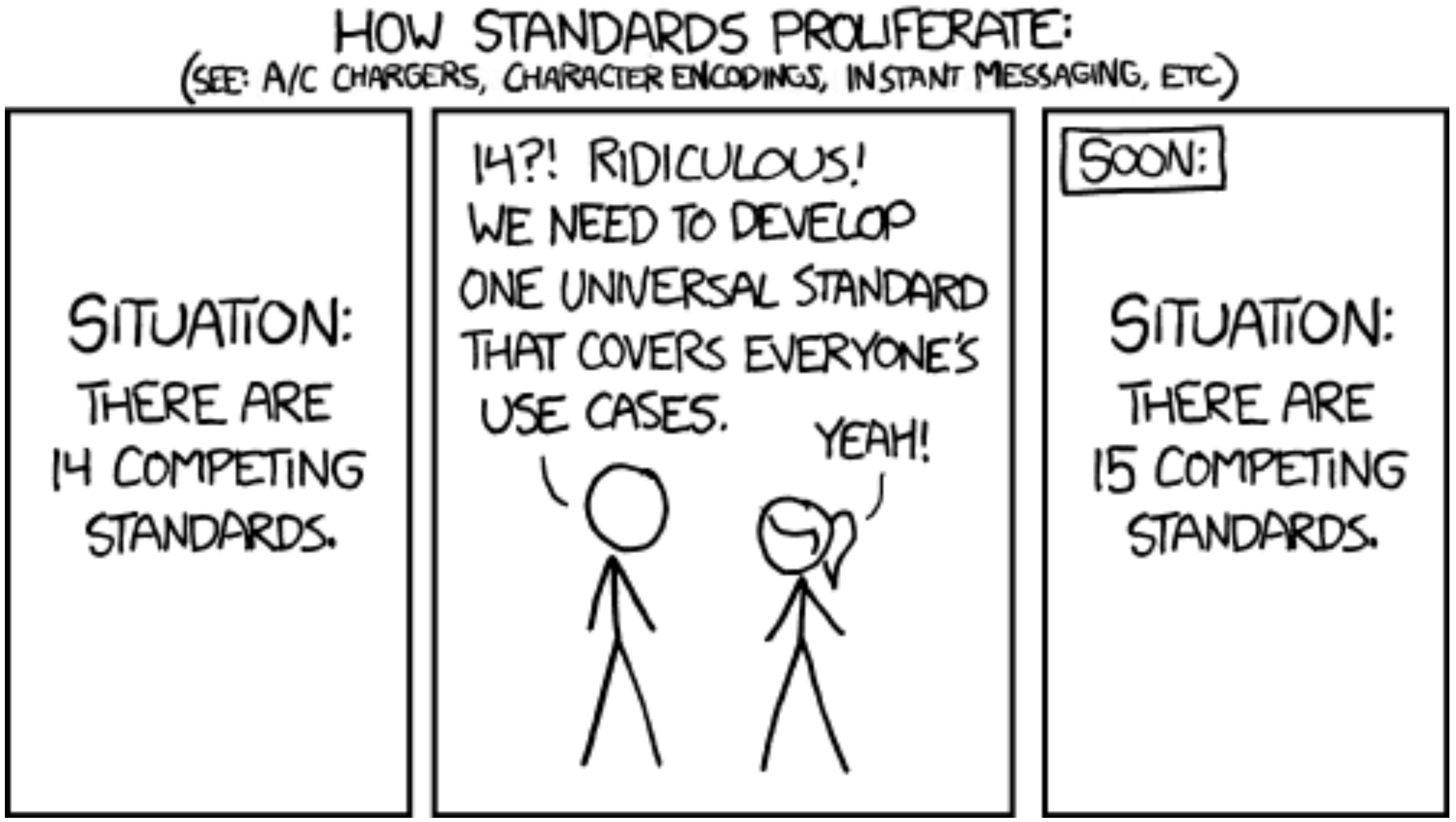
PMIx Standard: https://github.com/pmix/pmix-standard

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

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

Slack: pmix-workspace.slack.com

Ralph H. Castain, Joshua Hursey, Aurelien Bouteiller, David Solt,
“PMIx: Process management for exascale environments”, Parallel Computing, 2018.
https://doi.org/10.1016/j.parco.2018.08.002


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.