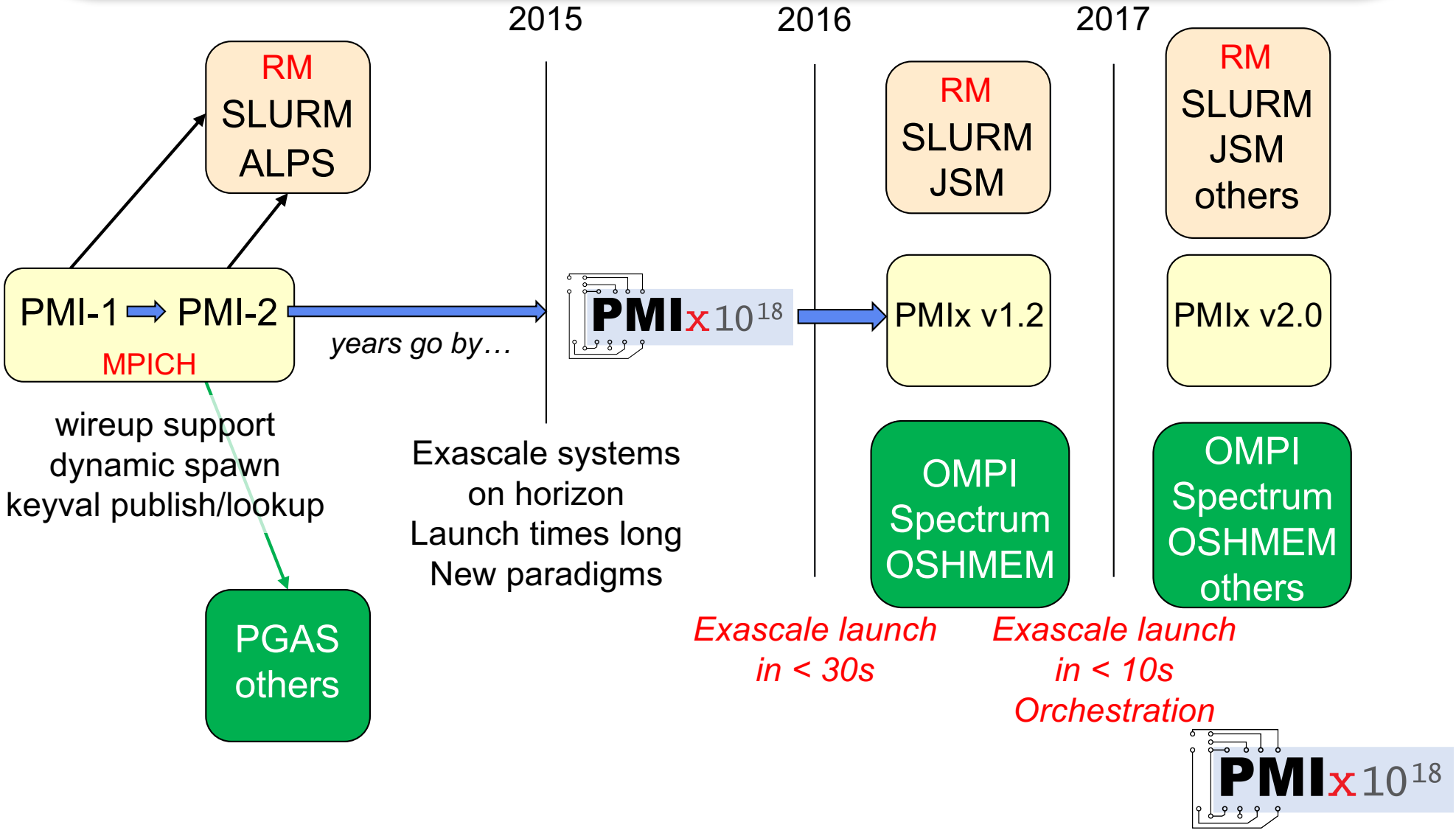
OpenMP – MPI Coordination

Issue

- Hybrid applications gaining popularity
  - Especially with many-core architectures
- Multiple programming libraries
  - Operate independently
  - No knowledge of presence of other libs or what those libs are doing
- Conflicts occur
  - Binding, resource utilization, …
- OpenMP/MPI WG
  - Prototype use of PMIx for coordination

What is PMIx?


2015
2016
2017
RM
SLURM
ALPS
PMI-1 ➡ PMI-2
MPICH
wireup support
dynamic spawn
keyval publish/lookup
PGAS
others
years go by…
PMIx 10 18
Exascale systems
on horizon
Launch times long
New paradigms
RM
SLURM
JSM
PMIx v1.2
OMPI
Spectrum
OSHMEM
Exascale launch
in < 30s
RM
SLURM
JSM
others
PMIx v2.0
OMPI
Spectrum
OSHMEM
others
Exascale launch
in < 10s
Orchestration
PMIx 10 18

The Community

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

PMIx 10^18

v1.2 Features

- Typical startup operations
  - Put, get, commit, barrier, spawn, [dis]connect, publish/lookup
- System/job data returned upon PMIx_Init
  - Topology, proc locations, bindings, …
  - Provided by host resource manager via PMIx
  - Full list: https://github.com/pmix/pmix/wiki/2.8-Pmix-Server-Data-Requirements

v2.0 Features

- Tool connections
  - Debugger, job submission, query
- Generalized query support
  - Job status, layout, system data, resource availability
- Logging
  - Status reports, error output
- Event notification
  - App, system generated
  - Subscribe, chained
  - Pre-emption, failures, timeout warning, …
- Flexible allocations
  - Release resources, request resources
- Job control
  - Pause, kill, signal, heartbeat, resilience support

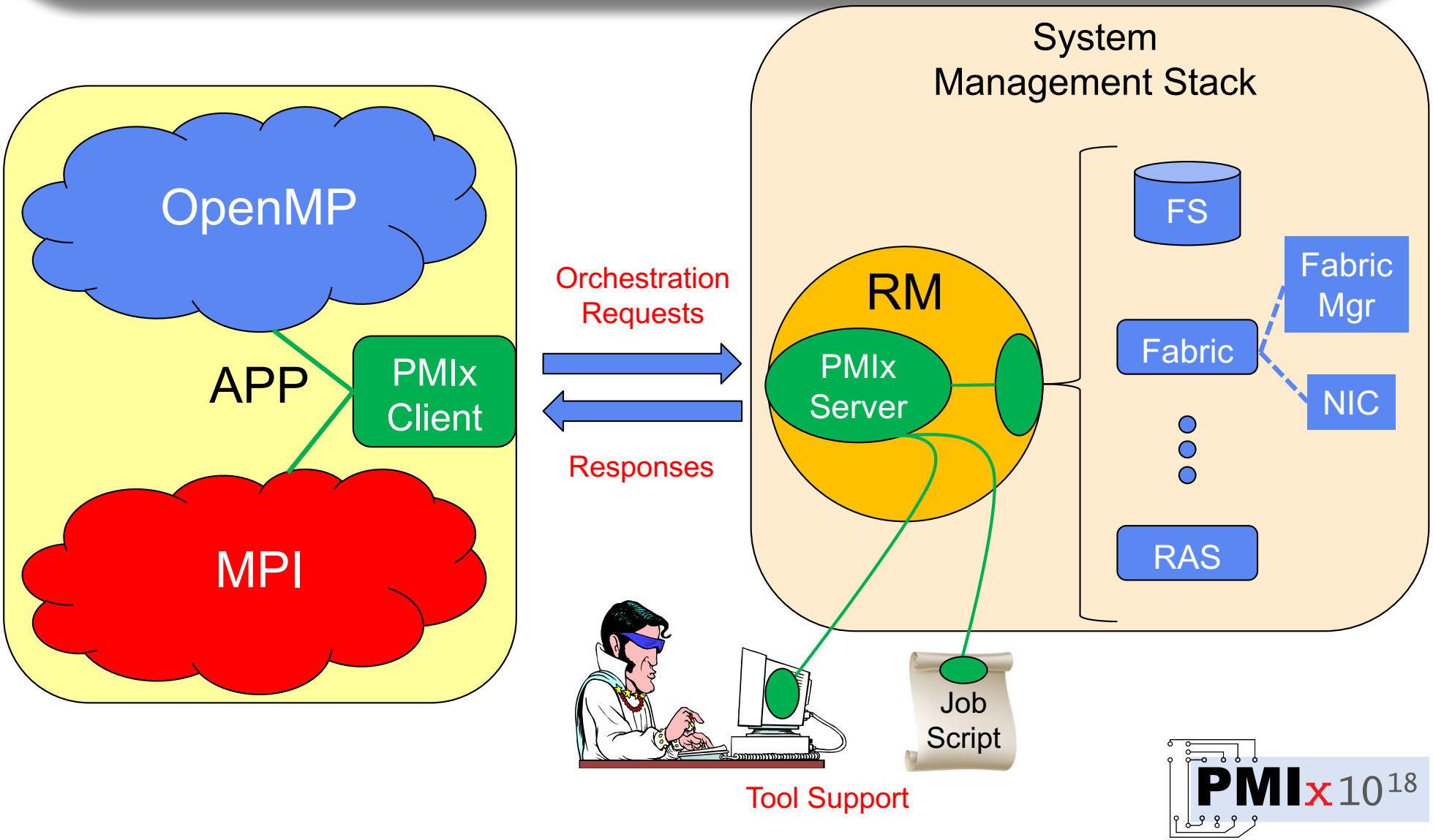
2Q2017

v3.0 Planned Features

- Network support
  - Security keys, pre-spawn local driver setup, fabric topology and status, traffic reports
- File system support
  - Dependency detection
  - Tiered storage caching strategies
- Inter-library coordination

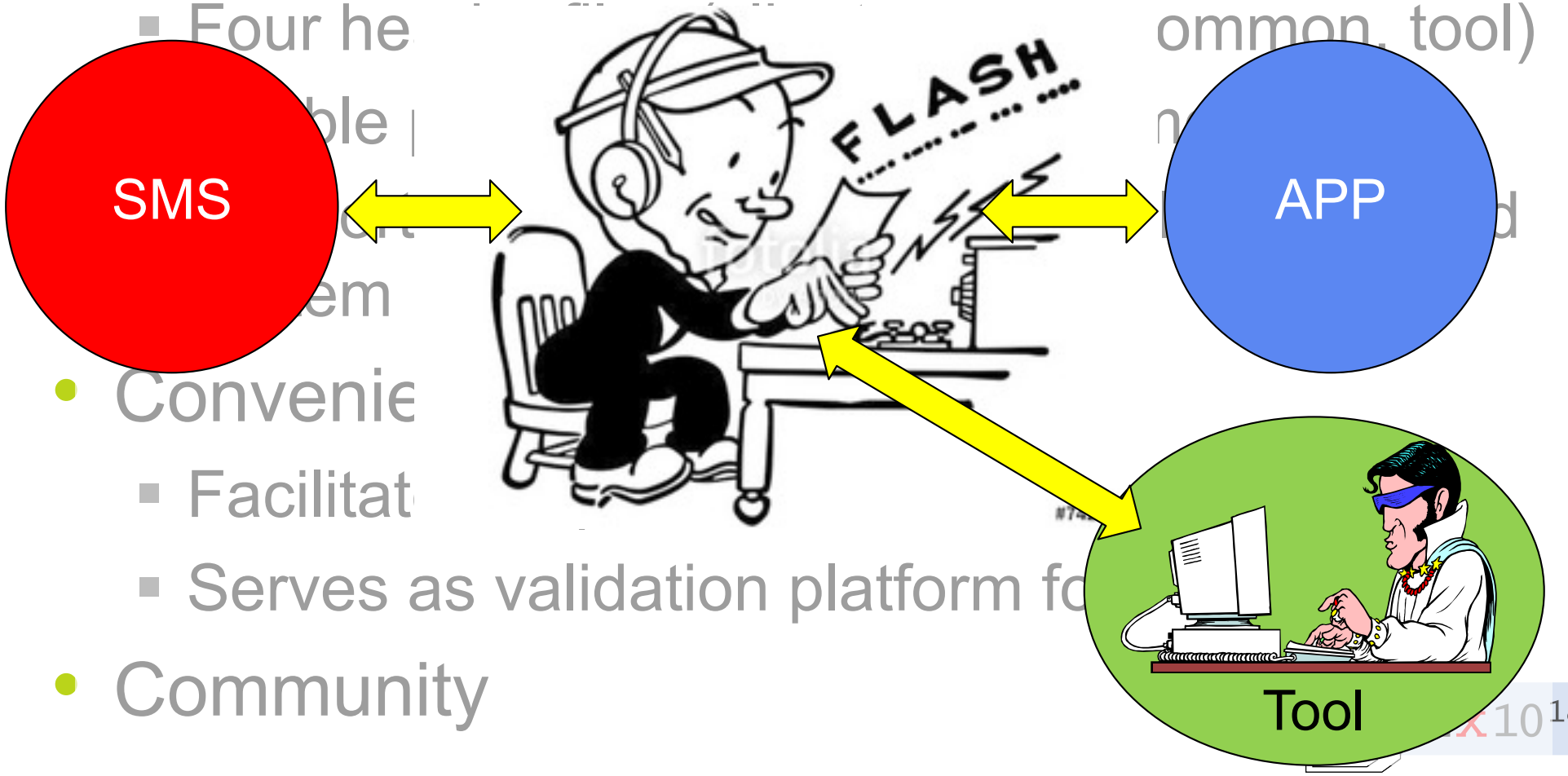
4Q2017

PMIx Integration Architecture



Messenger not Doer

- Standardized APIs
  - Four he ommon, tool)
  - ble
  - em
- Convenie
  - Facilitat
  - Serves as validation platform fo
- Community



Scheme

- Each library
  - Declares itself upon initialization
    - Model, implementation, version, …
  - Registers event handler
    - Receive async notice of other declarations
    - Async updates as published by other libs
  - Order is irrelevant
    - Events cached and delivered upon registration
- Coordinate during execution via events
- Opportunity
  - Utilize PMIx features for OpenMP itself

Initialization

```
pmix_info_t *info;
size_t ninfo;
pmix_status_t code = PMIX_MODEL_DECLARED;

/* declare ourselves */
ninfo = 4;
PMIX_INFO_CREATE(info, ninfo);
PMIX_INFO_LOAD(&info[0], PMIX_PROGRAMMING_MODEL, "EXAMPLE", PMIX_STRING);
PMIX_INFO_LOAD(&info[1], PMIX_MODEL_LIBRARY_NAME, "FOOL", PMIX_STRING);
PMIX_INFO_LOAD(&info[2], PMIX_MODEL_LIBRARY_VERSION, "1.2.3", PMIX_STRING);
PMIX_INFO_LOAD(&info[3], PMIX_THREADING_MODEL, "NONE", PMIX_STRING);
if (PMIX_SUCCESS != (rc = PMIx_Init(&myproc, info, ninfo))) {
    fprintf(stderr, "PMIx Init failed: %s\n", PMIx_Error_string(rc));
    exit(1);
}
PMIX_INFO_FREE(info, ninfo);

/* register a handler specifically for when models declare */
active = -1;
ninfo = 1;
PMIX_INFO_CREATE(info, ninfo);
PMIX_INFO_LOAD(&info[0], PMIX_EVENT_HDLR_NAME, "APP-MODEL", PMIX_STRING);
PMIx_Register_event_handler(&code, 1, info, ninfo,
                            model_callback, model_registration_callback, pointer_to_object);
/* do whatever */
        .
        .
        .
/* when registration completes */
PMIX_INFO_FREE(info, ninfo);
```

Callback when declarations received

Called back when registration completes

object returned on registration callback

Callback

- Receives
  - Copy of all info provided by declarer
  - size_t identifier of the event handler
  - Name of the event handler, if assigned
  - Provided object, if given
- Multiple handlers registered for same code(s)
  - Called in order (defined in registration)
  - Results of each handler aggregated and provided to later handlers

Summary

Working group:

Wiki: https://github.com/pmix/pmix/wiki/7.1-openmp

Minutes: https://github.com/pmix/pmix/wiki/4.3-openmp

Prototype code:

OpenMPI master: https://github.com/open-mpi/ompi

Example app:

https://github.com/open-mpi/ompi/blob/master/orte/test/mpi/xlib.c

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

Need to collaboratively define what we want to do with it