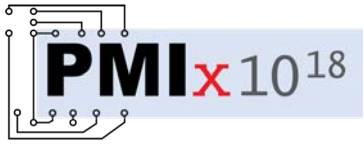
PMIx: Storage Integration

Agenda

- Brief overview of PMIx
  - What is PMIx?
  - Status
- Level set
  - Target vision of tiered storage
  - Role of workload manager
- PMIx-Storage integration
  - Related existing APIs
  - Possible extenstions

The Community

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



What Is PMIx?

- Standardized APIs
  - Four header files (client, server, common, tool)
  - Enable portability across environments
  - Support interactions between applications and system management stack
- Convenience library
  - Facilitate adoption
  - Serves as validation platform for standard
  - Plugin architecture to support proprietary plugins
- Community

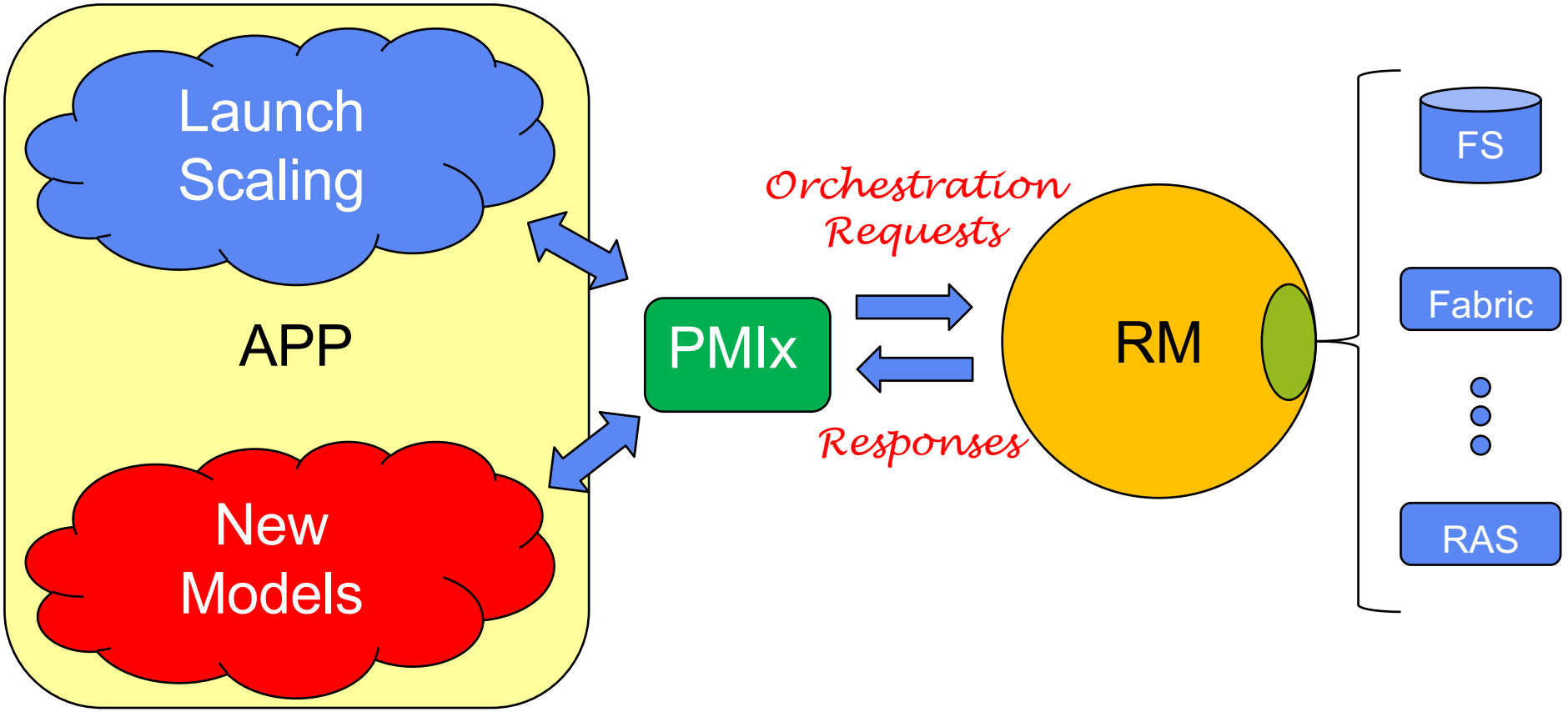
Motivation

- Exascale launch times are a hot topic
  - Desire: reduce from many minutes to few seconds
  - Target: $O(10^6)$ MPI processes on $O(10^5)$ nodes thru MPI_Init in < 30 seconds
- New programming models are emerging
  - Driven by need to efficiently exploit scale vs. resource constraints
  - Characterized by increased app-RM integration

PMIx 10^{18}

A Deal Is Struck

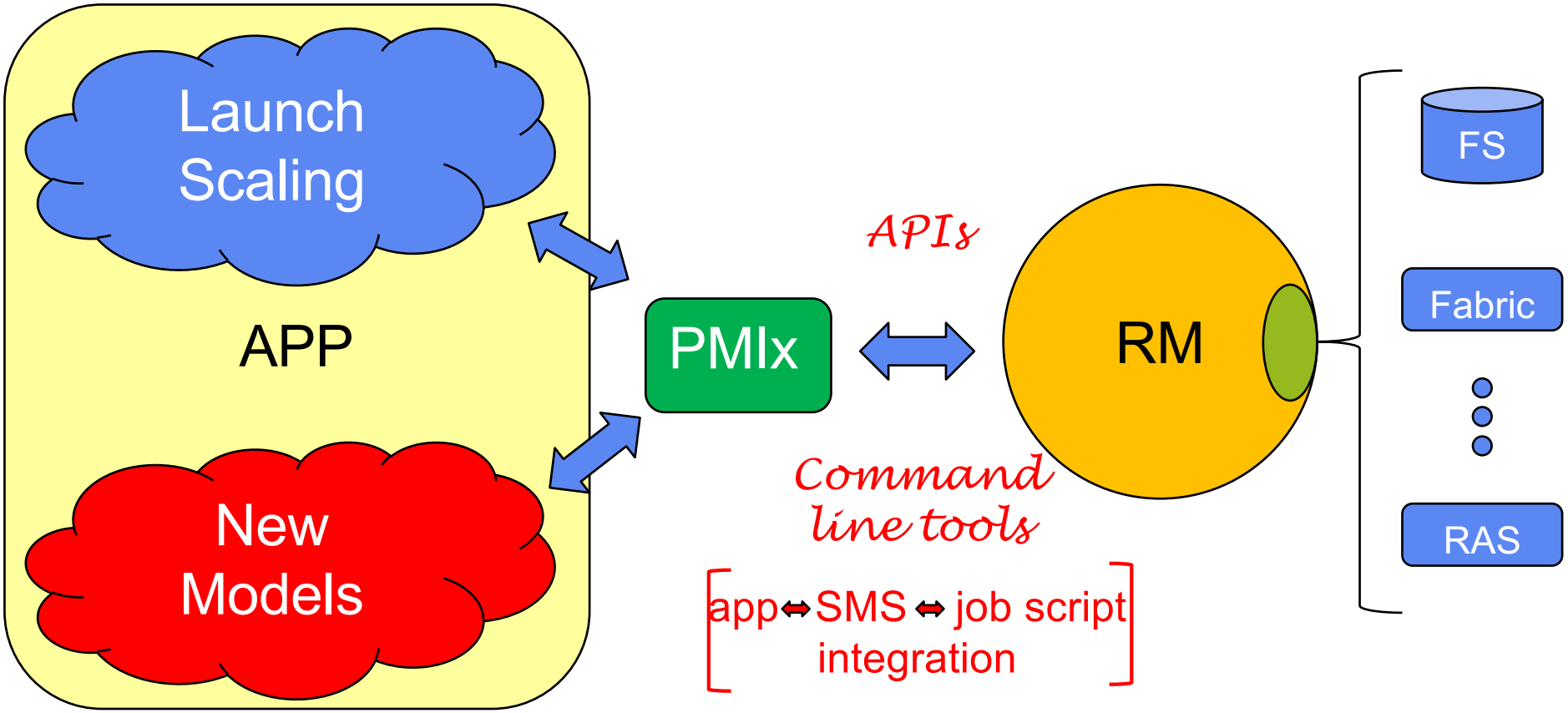
Instant On ⇔ Scope



Easier to add another callback than to support an additional library/community

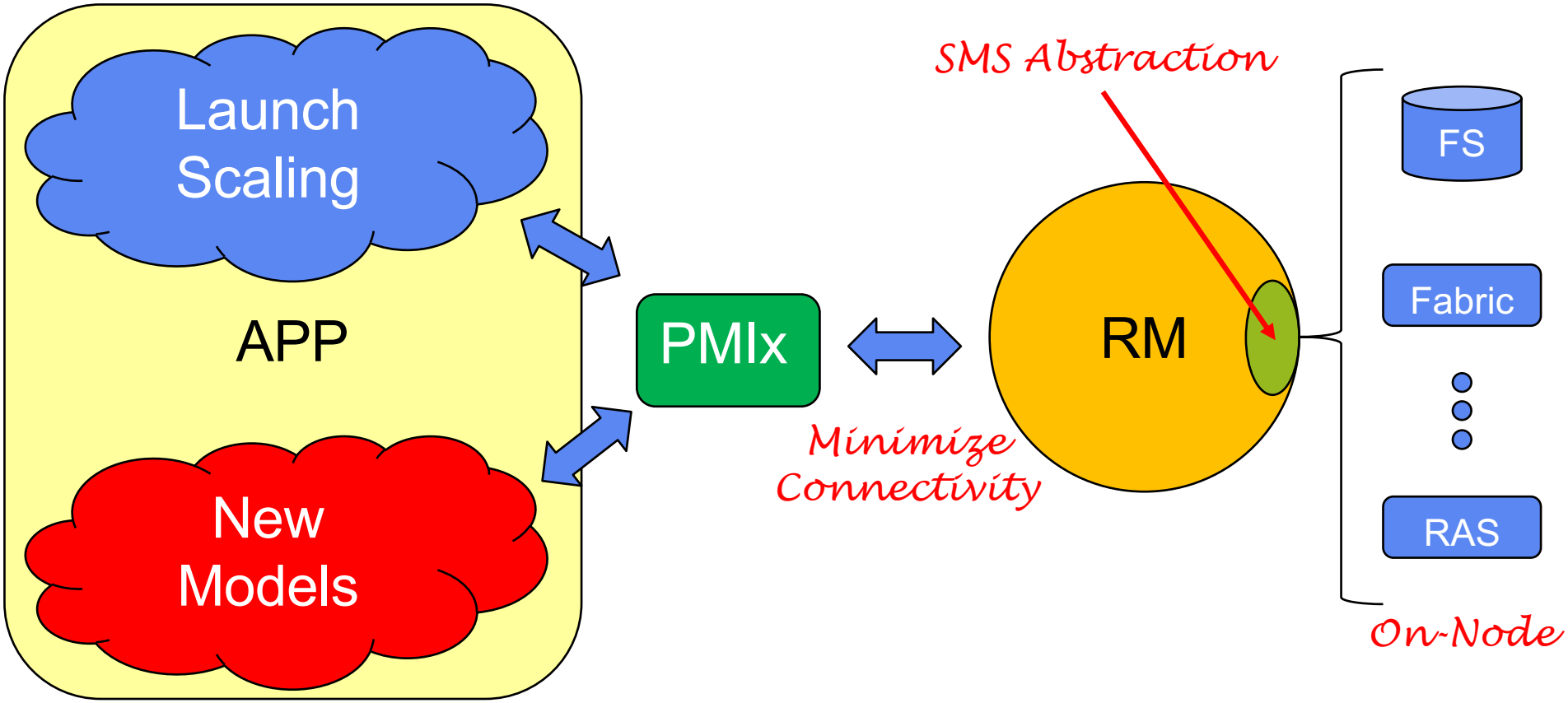
A Deal Is Struck

Instant On ⇔ Scope



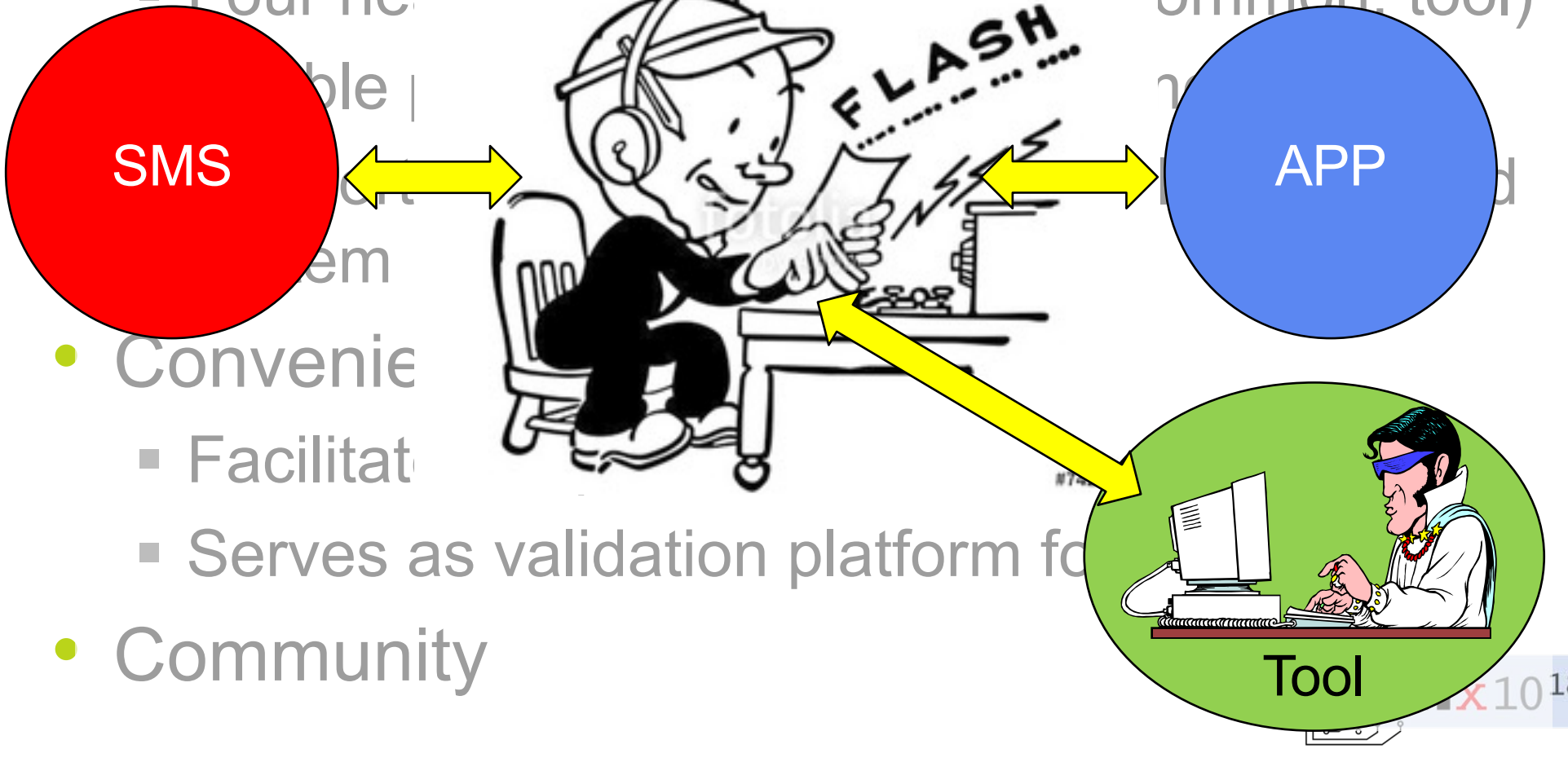
A Deal Is Struck

Instant On ⇔ Scope



Messenger not Doer

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



PMIx "Standards" Process

- Modifications/additions
  - Proposed as RFC
  - Include prototype implementation
    - Pull request to convenience library
  - Notification sent to mailing list
- Reviews conducted
  - RFC and implementation
  - Continues until consensus emerges
- Approval given
  - Developer telecon (2x/week)

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

RM Adoption

- Already released
  - SLURM 16.05 (PMIx v1.1.5)
  - IBM-CORAL
- Planned (2017)
  - IBM-LSF, Fujitsu, Adaptive Solutions, Altair, Microsoft
- Reference server
  - Provides surrogate support until native support becomes available
  - Supports full PMIx standard, limited by RM capabilities

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
  - Status reports, error output

In Pipeline

- Network support
  - Security keys, pre-spawn local driver setup, fabric topology and status, traffic reports
- Obsolescence protection
  - Automatic cross-version communication compatibility
- Flexible allocations
  - Release resources, request resources
- Job control
  - Pause, kill, signal, heartbeat, resilience support
- Generalized data store

Launch Scaling

- Eliminate initialization collectives
  - Stage I
    - RM has info – provide it at startup
    - Fetch info at first message
  - Stage II
    - Compute endpoints from RM info
  - Stage III
    - Correctly recover from unexpected messages prior to local process start
    - Use fabric for launch message
- Executable/library pre-staging
  - File system cache to switch-local NVRAM

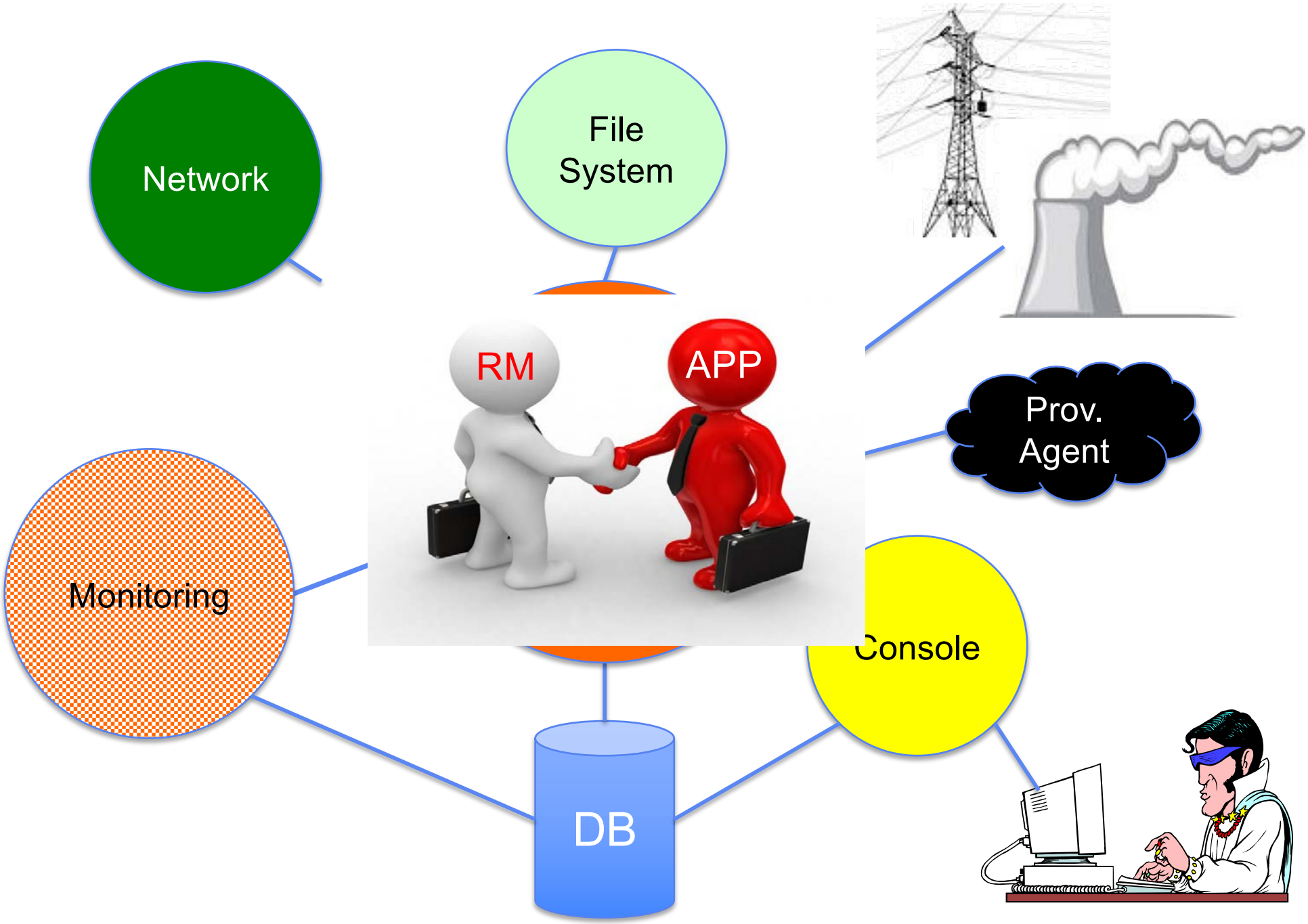
Complete or Scheduled

Where we came in…

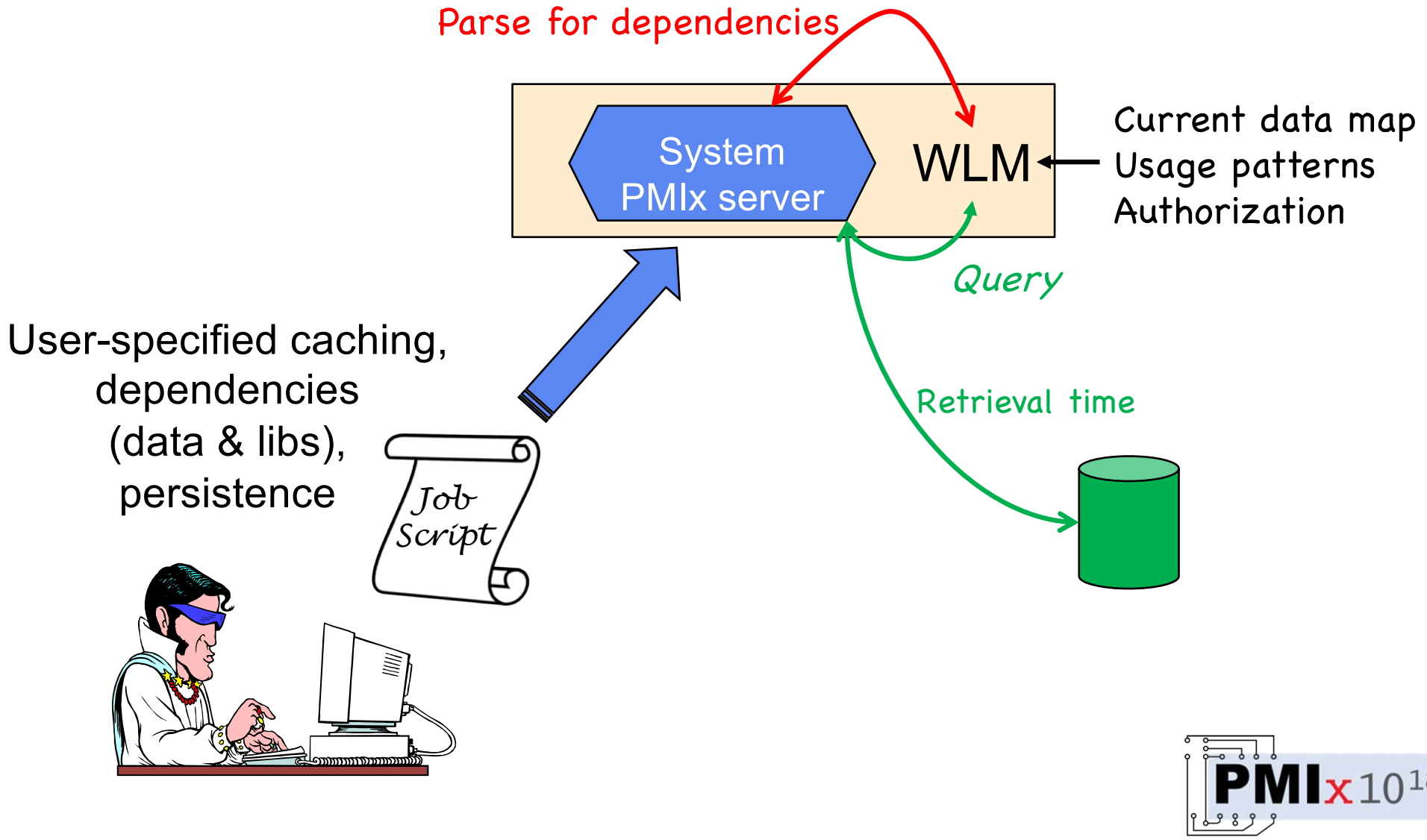
Baseline Vision

- 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

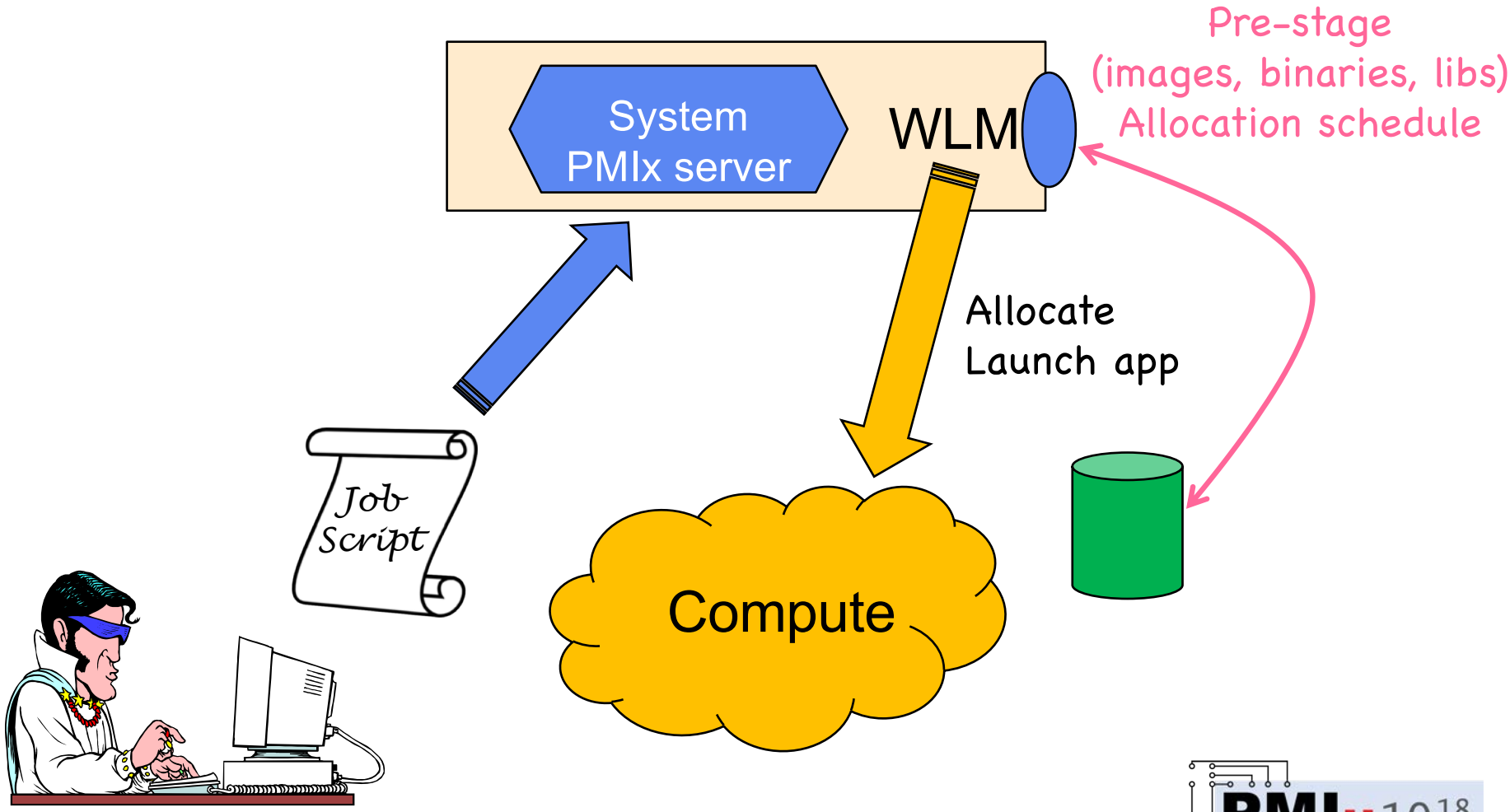
RM ⟹ Orchestrator



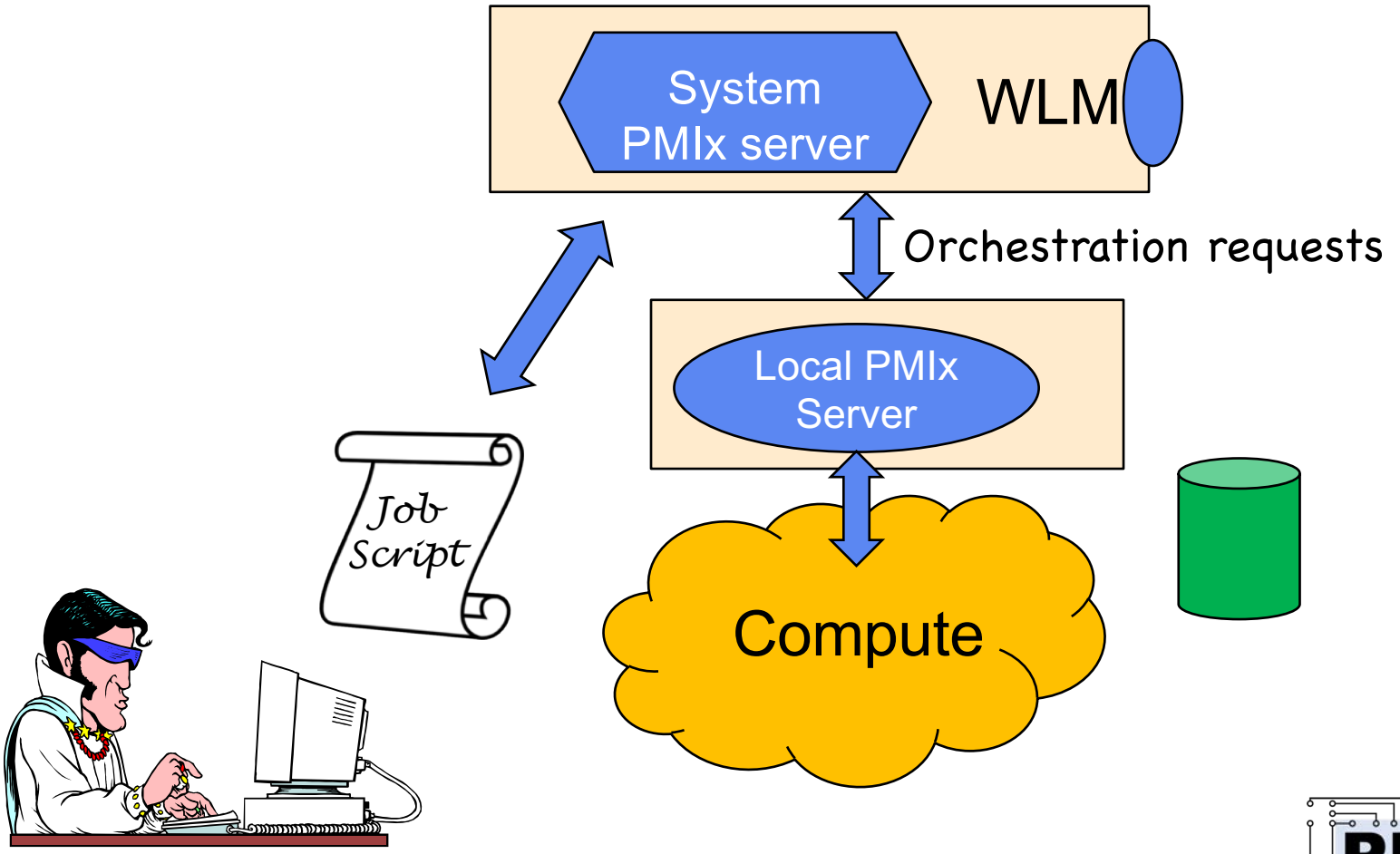
Planned Support



Planned Support



Planned Support



File System Integration Plans

- Tool support
  - Communicate initial requests
  - Job script/app coordination
- Dependency detection
- Abstraction to local subsystem libraries
- Extend existing functions
  - Query
  - Job control
  - Allocate
- New APIs?

Summary

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

Offer: collaboratively define what we want to do with it
New APIs? New keys?

Thank You!

PMIx 10^18