Process Management Interface – Exascale

PMIx 10^{18}

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration status
- Roadmap
- Malleable application support
- Wrap-Up/Open Forum

PMIx 10^{18}

PMIx – PMI exascale

Collaborative open source effort led by Intel, Mellanox Technologies, IBM, Adaptive Computing, and SchedMD.

New collaborators are most welcome!

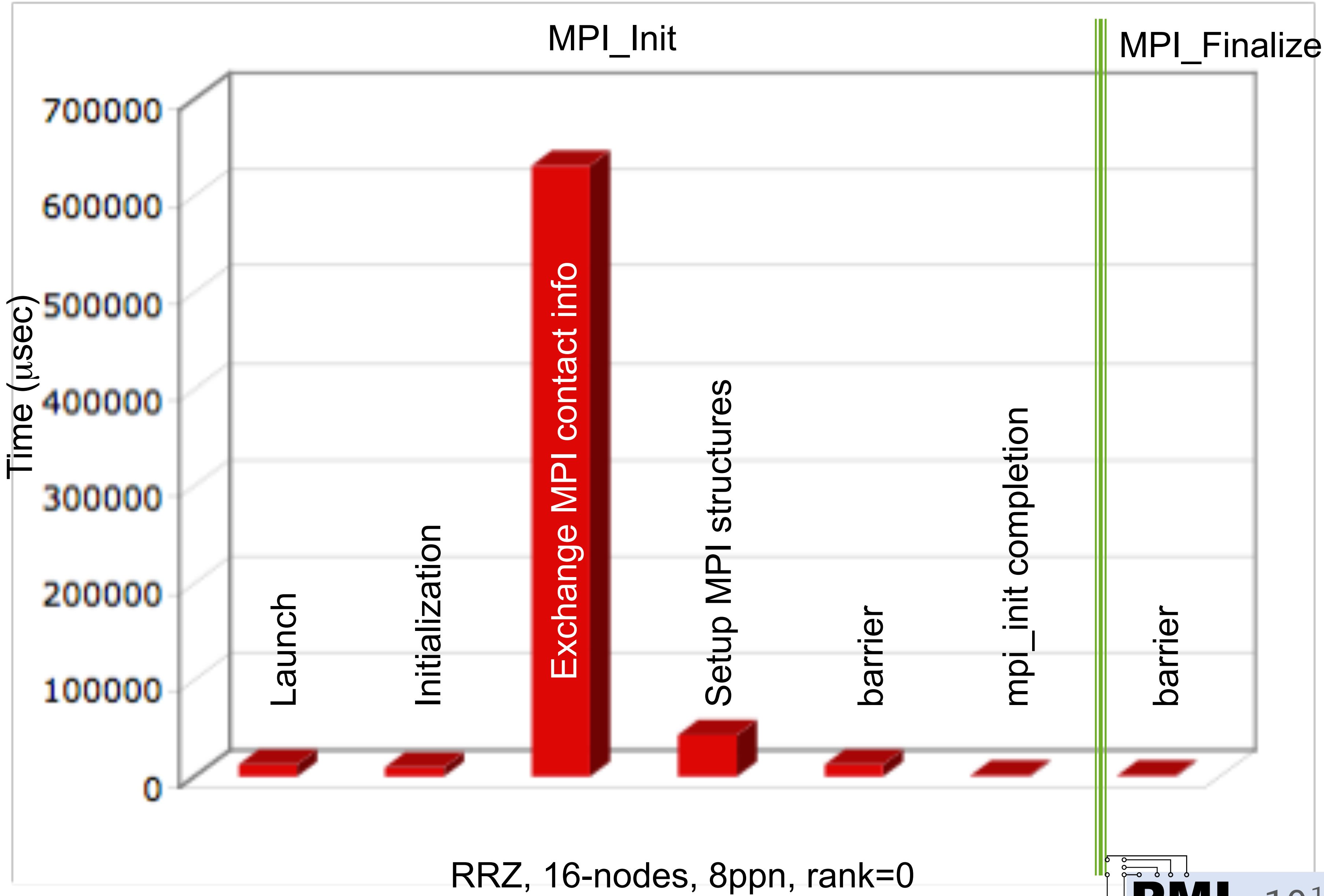
Contributors

- Intel
  - Ralph Castain
  - Annapurna Dasari
- Mellanox
  - Joshua Ladd
  - Artem Polyakov
  - Elena Shipunova
  - Nadezhda Kogteva
  - Igor Ivanov
- HP
  - David Linden
  - Andy Riebs
- IBM
  - Dave Solt
- Adaptive Computing
  - Gary Brown
- RIST
  - Gilles Gouaillardet
- SchedMD
  - David Bigagli
- LANL
  - Nathan Hjelmn

PMIx 10^{18}

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 exploding
  - Driven by need to efficiently exploit scale vs. resource constraints
  - Characterized by increased app-RM integration


MPI_Init
MPI_Finalize
Time (µsec)
700000
600000
500000
400000
300000
200000
100000
0
Launch
Initialization
Exchange MPI contact info
Setup MPI structures
barrier
mpi_init completion
barrier
RRZ, 16-nodes, 8ppn, rank=0

What Is Being Shared?

- Job Info (~90%)
  - Names of participating nodes
  - Location and ID of procs
  - Relative ranks of procs (node, job)
  - Sizes (#procs in job, #procs on each node)

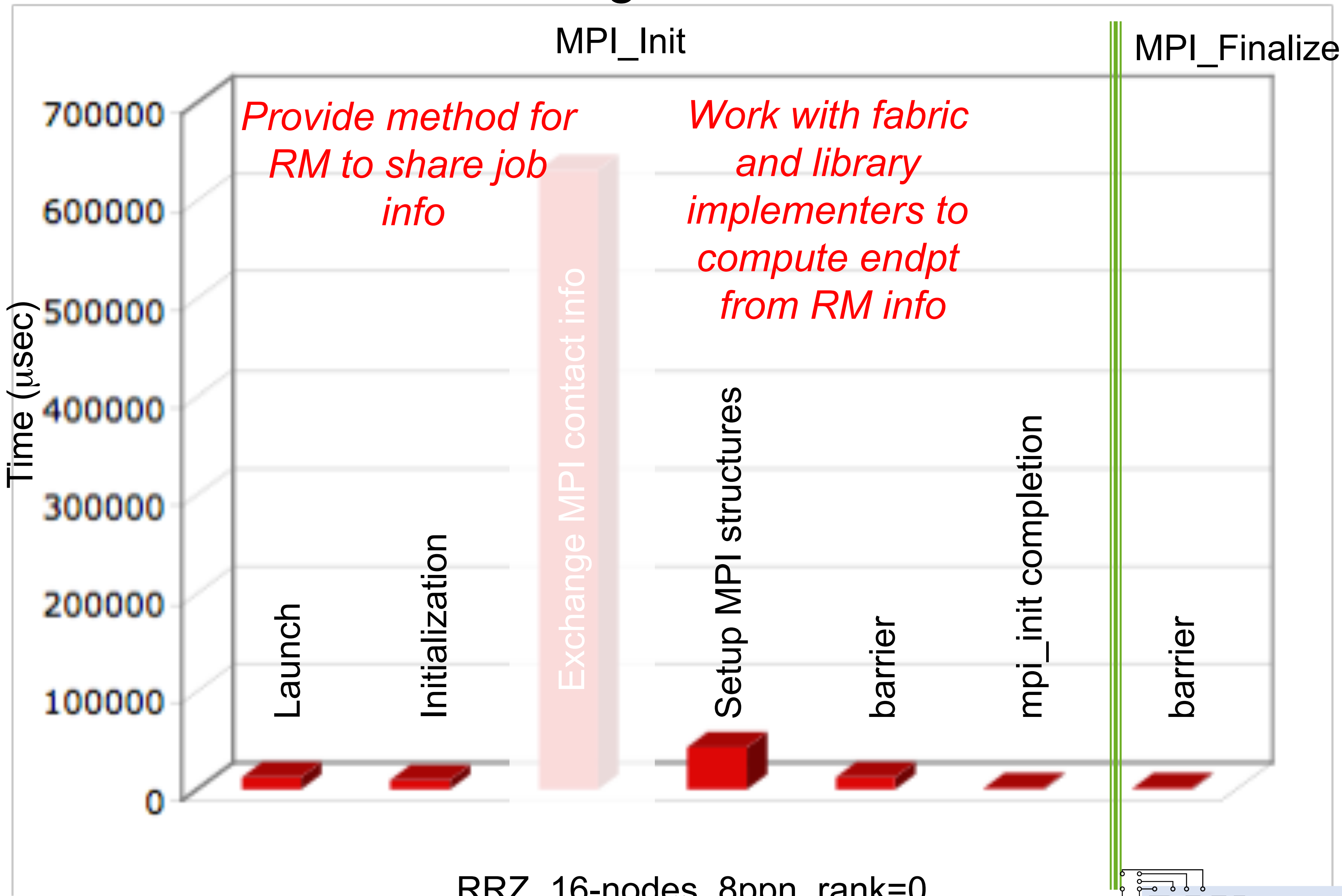
Known to local RM daemon

- Endpoint info (~10%)
  - Contact info for each supported fabric

Can be computed for many fabrics

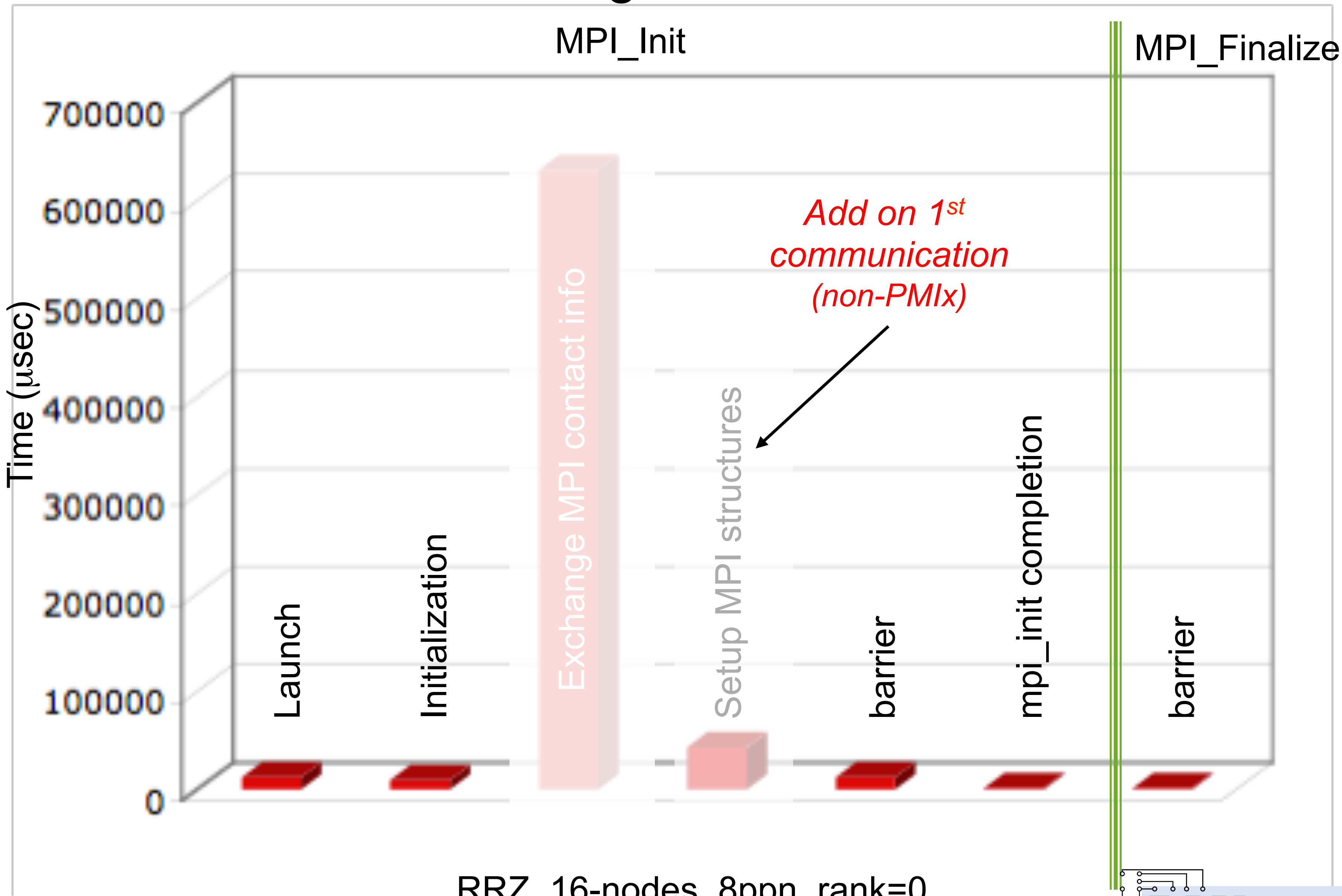
PMIx 10^{18}

Stage I



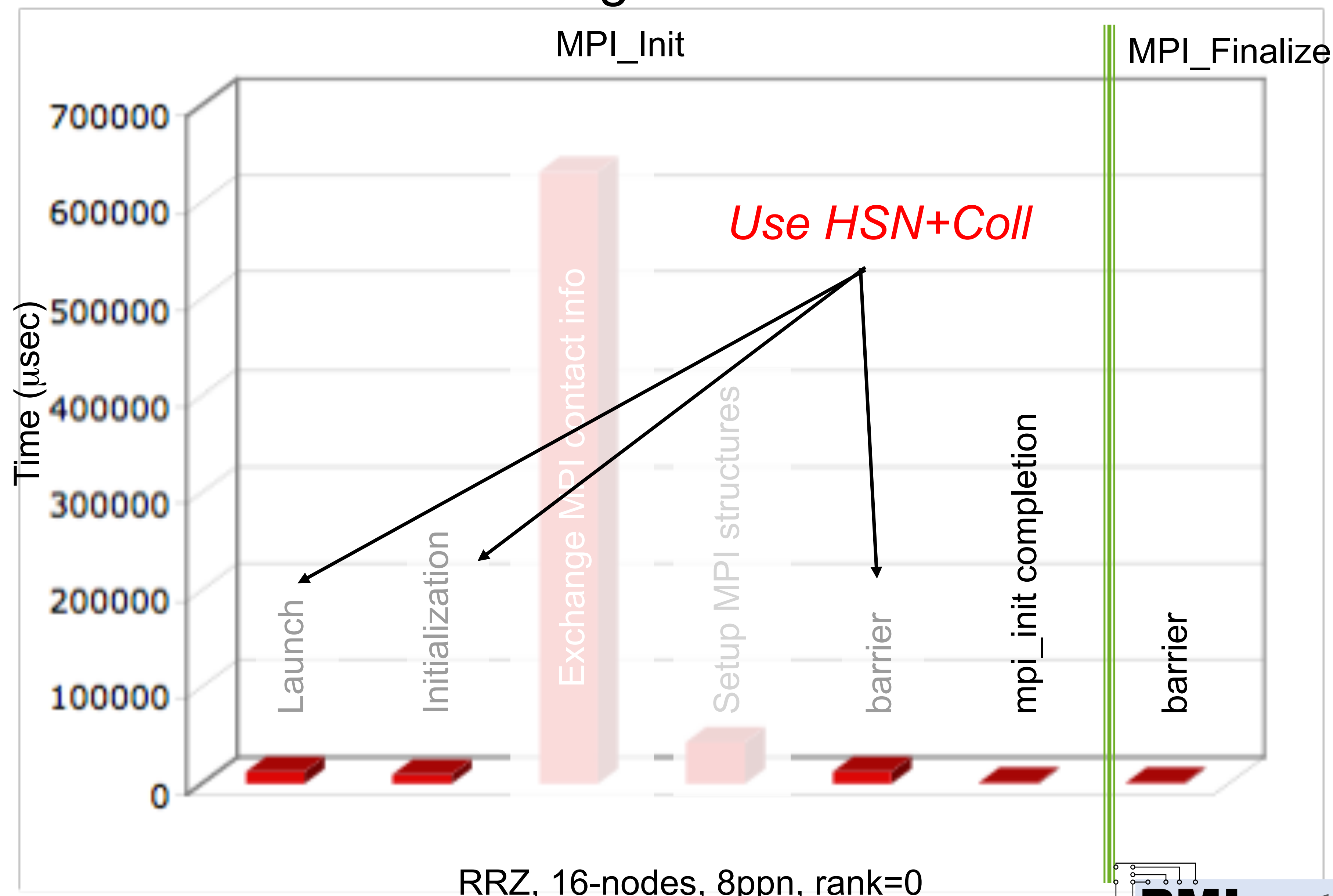
PMIx 10^{18}

Stage II



PMIx 10^{18}

Stage III



MPI_Init
MPI_Finalize
Use HSN+Coll
Time (µsec)
700000
600000
500000
400000
300000
200000
100000
0
Launch
Initialization
Exchange MPI contact info
Setup MPI structures
barrier
mpi_init completion
barrier
RRZ, 16-nodes, 8ppn, rank=0

Changing Needs

- 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}

Objectives

- Establish an independent, open community
  - Industry, academia, lab
- Standalone client/server libraries
  - Ease adoption, enable broad/consistent support
  - Open source, non-copy-left
  - Transparent backward compatibility
- Support evolving programming requirements
- Enable "Instant On" support
  - Eliminate time-devouring steps
  - Provide faster, more scalable operations

Today's Goal

- Inform the community
- Solicit your input on the roadmap
- Get you a little excited
- Encourage participation

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration/development status
- Roadmap
- Malleable/Evolving application support
- Wrap-Up/Open Forum

PMIx End Users

- OSHMEM consumers
  - In Open MPI OSHMEM:

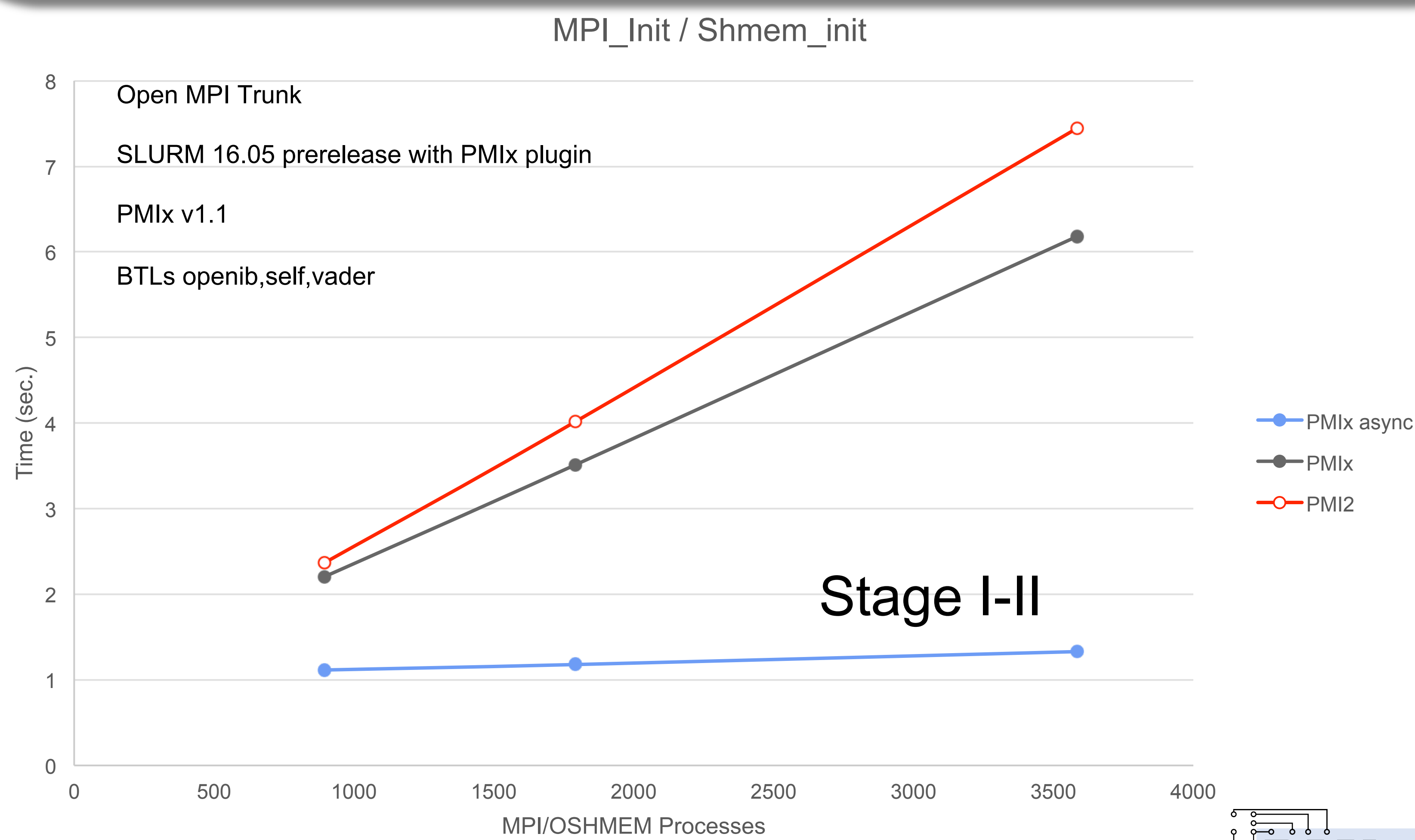
    $$shmem\_init = mpi\_init + C$$

    - Job launch scales as MPI_Init.
  - Data driven communication patterns
    - Assume dense connectivity
- MPI consumers
  - Large class of applications have spare connectivity
    - Ideal for an API that supports the direct modex concept

What's been done

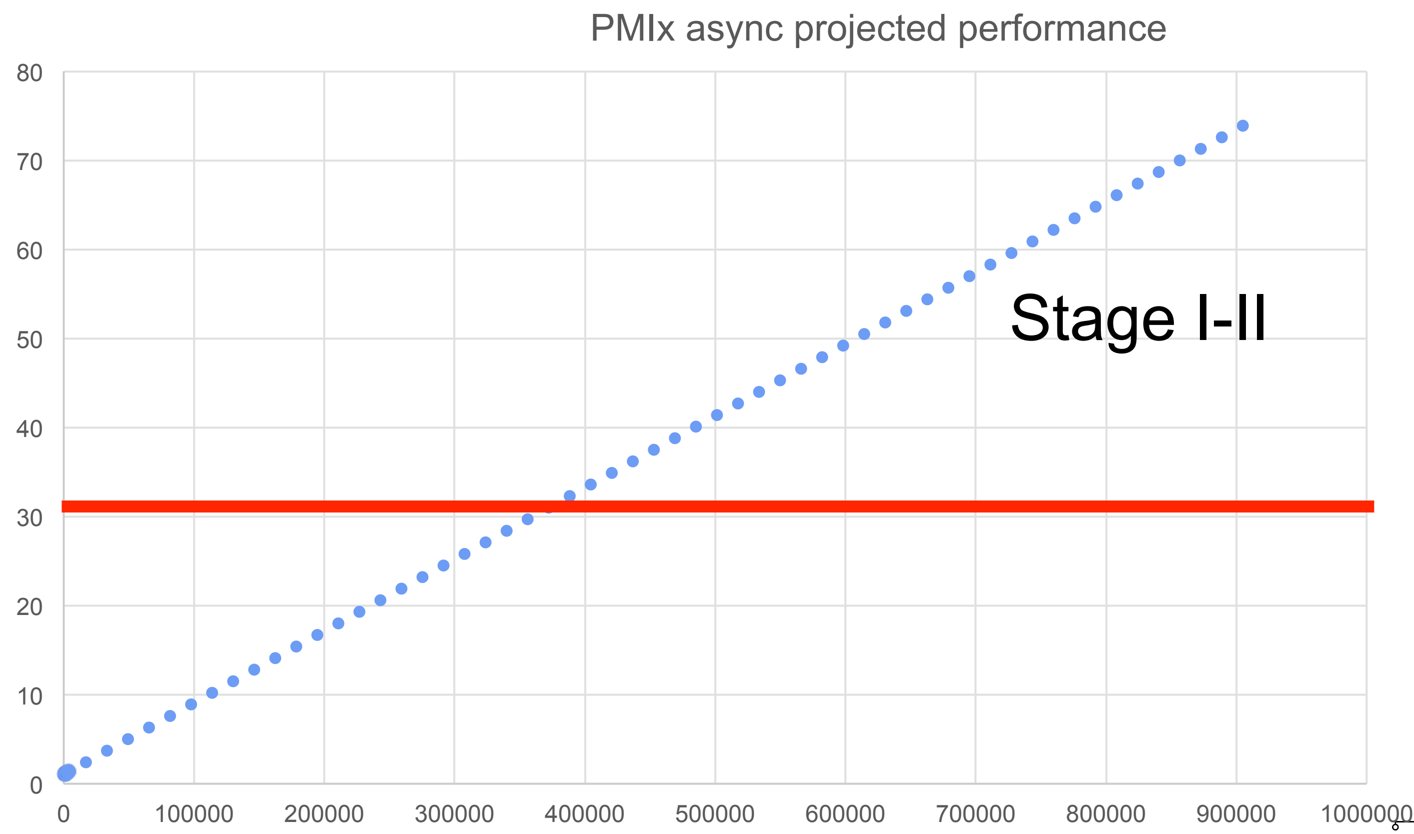
- Worked closely with customers, OEMs, and open source community to design a scalable API that addresses measured limitations of PMI2
    - Data driven design.
- Led to the PMIx v1.0 API
- Implementation and imminent release of PMIx v1.1
    - November 2015 release scheduled.
- Significant architectural changes in Open MPI to support direct modex
    - "Add procs" in bulk MPI_Init → "Add proc" on-demand on first use outside MPI_init.
    - Available in the OMPI v2.x release Q1 2016.
- Integrated PMIx into Open MPI v2.x
    - For native launching as well as direct launching under supported RMs.
    - For mpirun launched jobs, ORTE implements PMIx callbacks.
    - For srun launched jobs, SLURM implements PMIx callbacks in the PMIx plugin.
    - Client side framework added to OPAL with components for
        - Cray PMI
        - PMI1
        - PMI2
        - PMIx
        - backwards compatibility with PMI1 and PMI2.
- Implemented and submitted upstream SLURM PMIx plugin
    - Currently available in SLURM Head
    - To be released in SLURM 16.05
    - Client side PMIx Framework and S1, S2, PMIxxx components in OPAL
- PMIx unit tests integrated into Jenkins test harness

srun --mpi=xxx hello_world

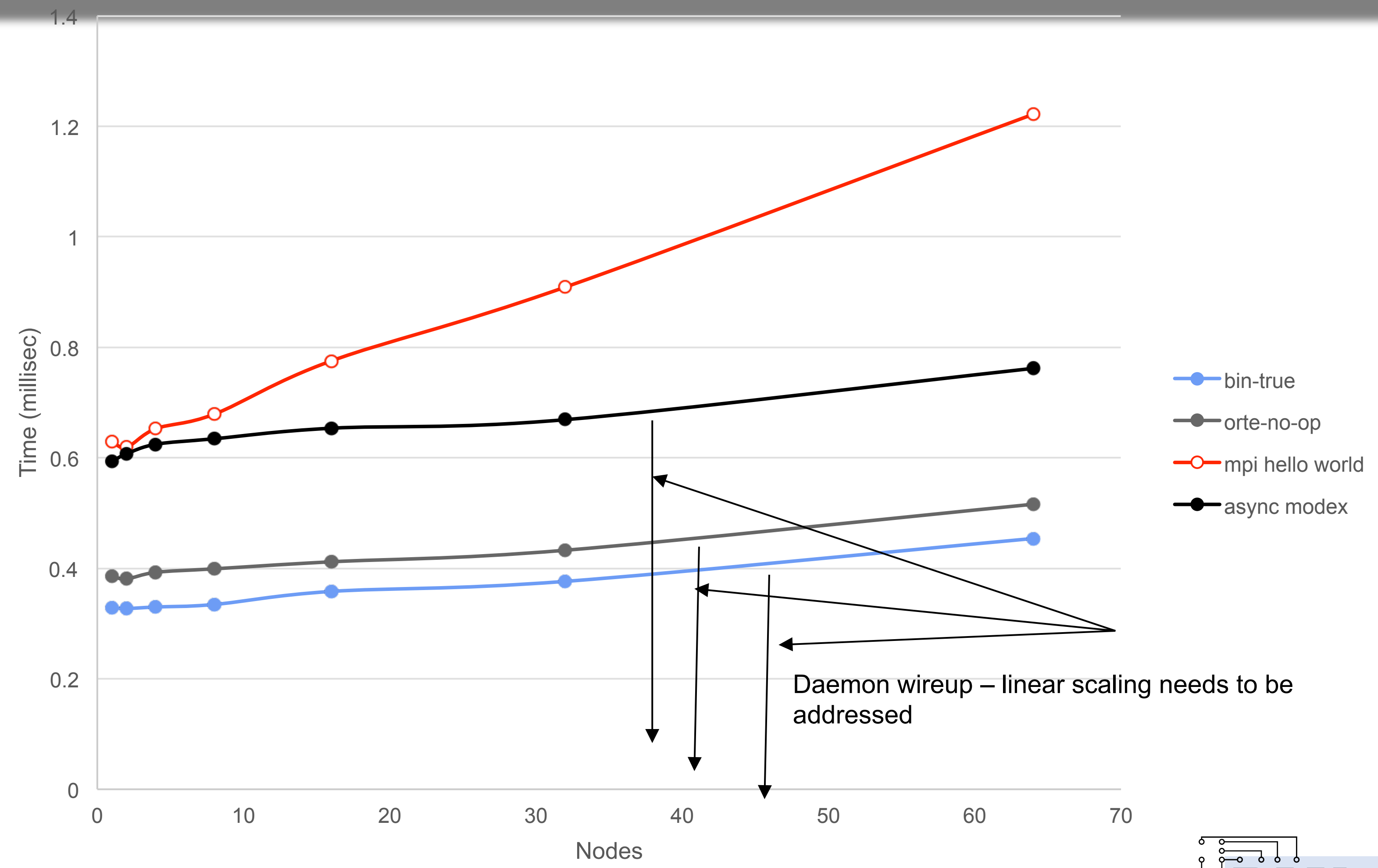


PMIx 10^{18}

srun --mpi=pmix ./hello_world



mpirun/oshrun ./hello_world



PMIx 10^{18}

Conclusions

- API is implemented and performing well in a variety of settings
  - Server integrated in OMPI for native launching and in SLURM as PMIx plugin for direct launching.
- PMIx shows improvement over other state-of-the-art PMI2 implementations when doing a full modex
  - Data blobs versus encoded metakeys
  - Data scoping to reduce the modex size
- PMIx supported direct modex significantly outperforms full modex operations for BTL/MTLs that can support this feature
- Direct modex still scales as O(N)
- Efforts and energy should be focused on daemon bootstrap problem
- Instant-on capabilities could be used to further reduce deamon bootstrap time

Next Steps

- Leverage PMIx features
- Reduce modex size with data scoping
- Change MTL/PMLs to support direct modex
- Investigate the impact of direct modex on densely connected applications
- Continue to improve collective performance
  - Still need to have a scalable solution
- Focus more efforts on the daemon bootstrap problem – this becomes the limiting factor moving to exascale
  - Leverage instant-on here as well

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration/development status
- Roadmap
- Malleable application support
- Wrap-Up/Open Forum

Client Implementation Status

- PMIx 1.1.1 released
  - Complete API definition
    - Future-proof API's with Info array/length parameter for most calls
    - Blocking/non-blocking versions of most calls
    - Picked up by Fedora, others to come
- PMIx MPI clients launched/tested with
  - ORTE (indirect) / ORCM (direct launch)
  - SLURM servers (direct launch)
  - IBM PMIx server (direct launch)

Server Implementation Status

- Server implementation time is greatly reduced through the PMIx convenience library
  - Handles all server/client interactions
  - Handles many PMIx requests that can be handled locally
  - Bundles many off-host requests
- Optional
  - RMs free to implement their own

Server Implementation Status

- Moab
  - Integrated PMIx server in scheduler/launcher
  - Currently integrating PMIx effort with Moab
  - Scheduled for general availability: no time set
- ORTE/ORCM
  - Full embedded PMIx reference server implementation
  - Scheduled for release with v2.0

Server Implementation Status

- SLURM
  - PMIx support for initial job launch/wireup currently developed & tested
  - Scheduled for GA: 16.05 release
- IBM/LSF
  - CORAL
    - PMIx support for initial job launch/wireup currently developed & tested w/PM
    - Full PMIx support planned for CORAL
  - Integration to LSF to follow (TBD)

PMIx 10^{18}

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration/development status
- Roadmap
- Malleable/Evolving application support
- Wrap-Up/Open Forum

PMIx 10^{18}

Scalability

- Memory footprint
  - Distributed database for storing Key-Values
    - Memory cache, DHT, other models?
  - One instance of database per node
    - "zero-message" data access using shared-memory
- Launch scaling
  - Enhanced support for collective operations
    - Provide pattern to host, host-provided send/recv functions, embedded inter-node comm?
  - Rely on HSN for launch, wireup support
    - While app is quiescent, then return to OOB

Flexible Allocation Support

- Request additional resources
  - Compute, memory, network, NVM, burst buffer
  - Immediate, forecast
  - Expand existing allocation, separate allocation
- Return extra resources
  - No longer required
  - Will not be used for some specified time, reclaim (handshake) when ready to use
- Notification of preemption
  - Provide opportunity to cleanly pause

PMIx 10^{18}

I/O Support

- Asynchronous operations
  - Anticipatory data fetch, staging
  - Advise time to complete
  - Notify upon available
- Storage policy requests
  - Hot/warm/cold data movement
  - Desired locations and striping/replication patterns
  - Persistence of files, shared memory regions across jobs, sessions
  - ACL to generated data across jobs, sessions

PMIx 10^{18}

Spawn Support

- Staging support
  - Files, libraries required by new apps
  - Allow RM to consider in scheduler
    - Current location of data
    - Time to retrieve and position
    - Schedule scalable preload
- Provisioning requests
  - Allow RM to consider in selecting resources, minimize startup time due to provisioning
  - Desired image, packages

Network Integration

- Quality of service requests
  - Bandwidth, traffic priority, power constraints
  - Multi-fabric failover, striping prioritization
  - Security requirements
    - Network domain definitions, ACLs
- Notification requests
  - State-of-health
  - Update process endpoint upon fault recovery
- Topology information
  - Torus, dragonfly, …

PMIx 10^{18}

Power Control/Management

- Application requests
  - Advise of changing workload requirements
  - Request changes in policy
  - Specify desired policy for spawned applications
  - Transfer allocated power to specifed job
- RM notifications
  - Need to change power policy
    - Allow application to accept, request pause
  - Preemption notification
- Provide backward compatibility with PowerAPI

PMIx: Fault Tolerance

- Notification
  - App can register for error notifications, incipient faults
    - RM will notify when app would be impacted
    - App responds with desired action
      - Terminate/restart job, wait for checkpoint, etc.
    - RM/app negotiate final response
  - App can notify RM of errors
    - RM will notify specified, registered procs
- Restart support
  - Specify source (remote NVM checkpoint, global filesystem, etc)
  - Location hints/requests
  - Entire job, specific processes

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration/development status
- Roadmap
- Malleable/Evolving application support
- Wrap-Up/Open Forum

Job Types

- Rigid
- Moldable
- Malleable
- Evolving
- Adaptive (Malleable + Evolving)

Job Type Characteristics

- Resource Allocation Type
  - Static
  - Dynamic

Who Decides	When it is decided	
	At job submission (static allocation)	During job execution (dynamic allocation)
User	Rigid	Evolving
Scheduler	Moldable	Malleable

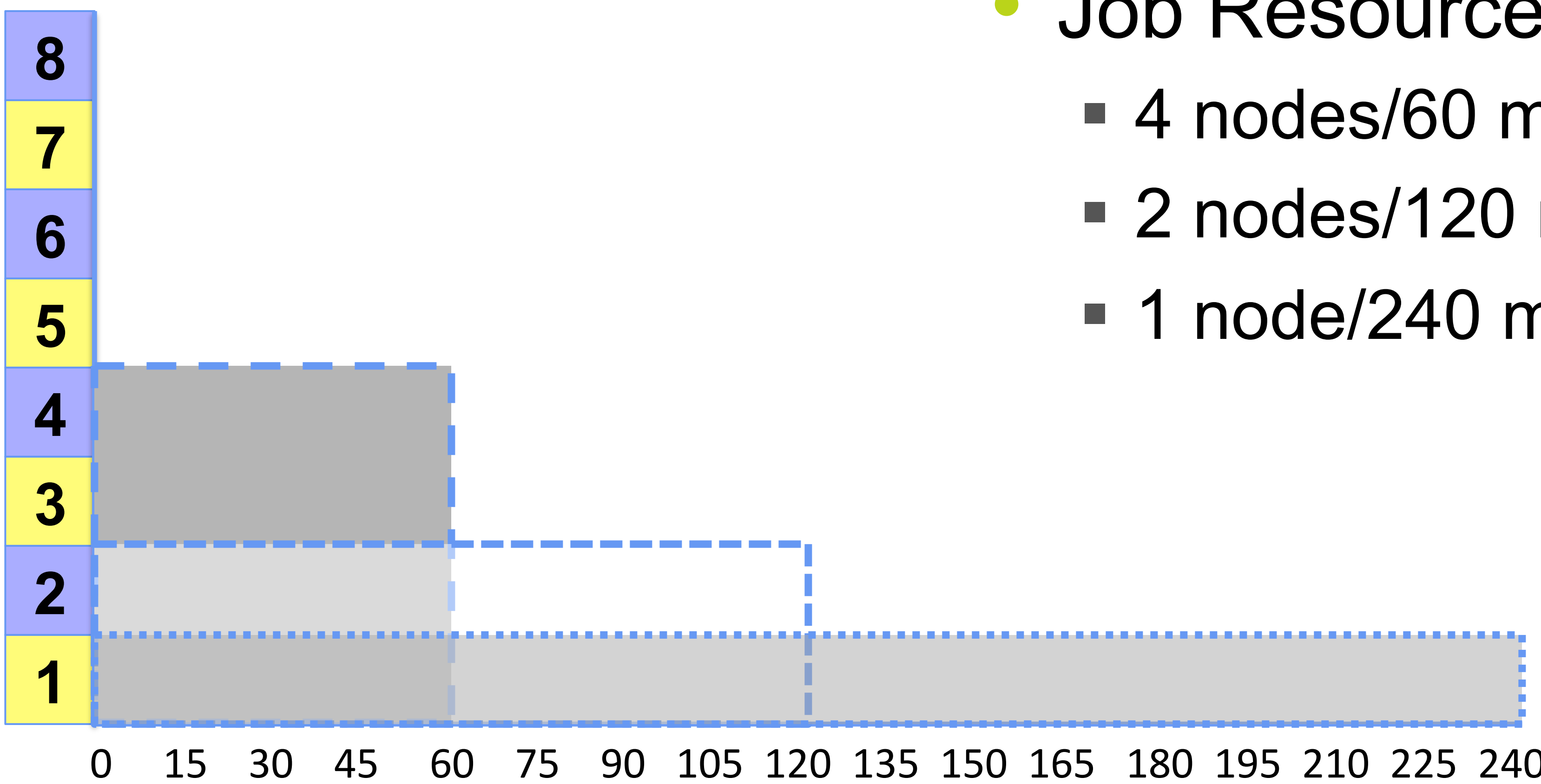
PMIx10^{18}

Rigid Job

- Job Resources
  - 4 nodes
  - 60 minutes

PMIx 10^{18}

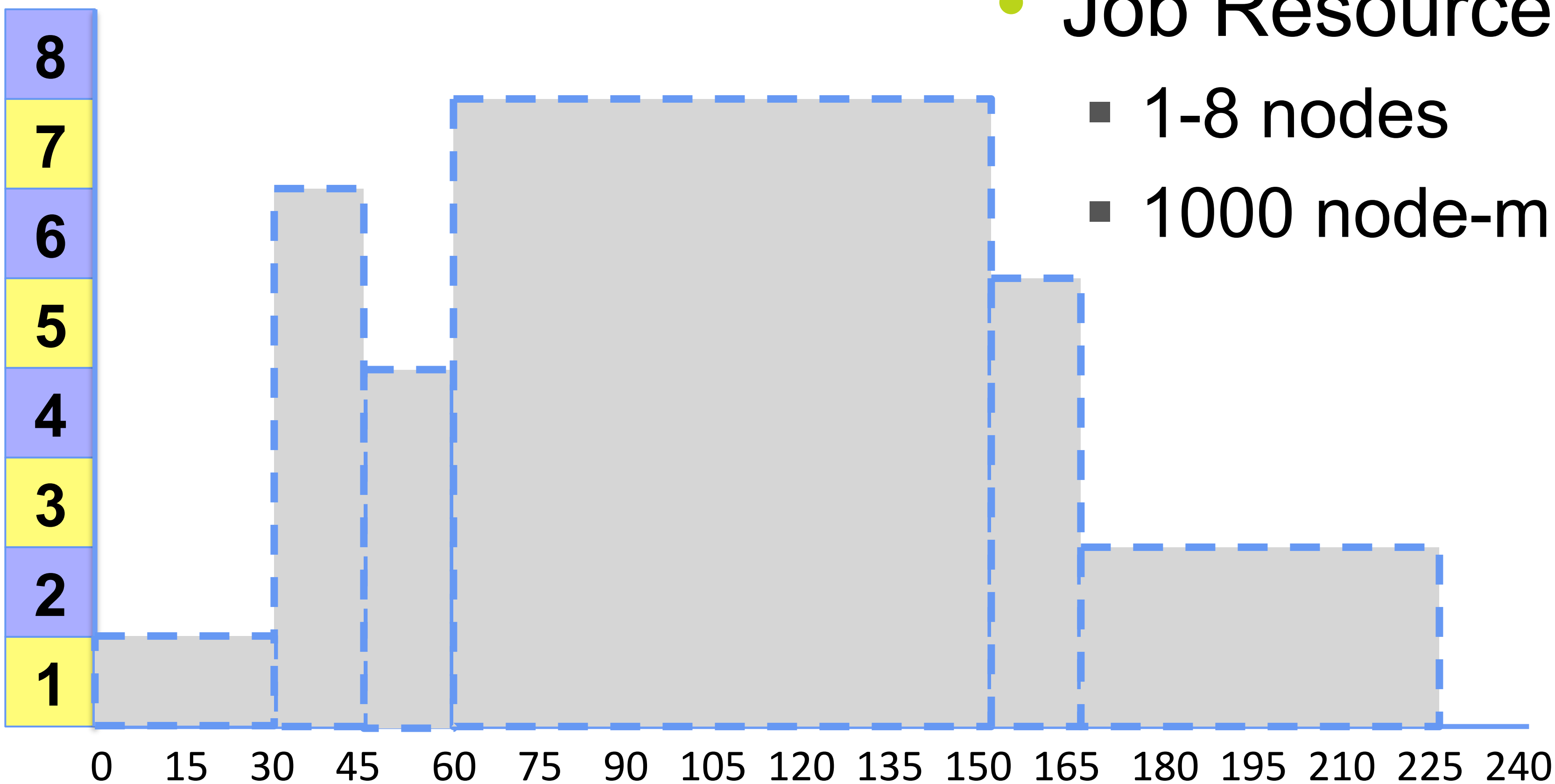
Moldable Job



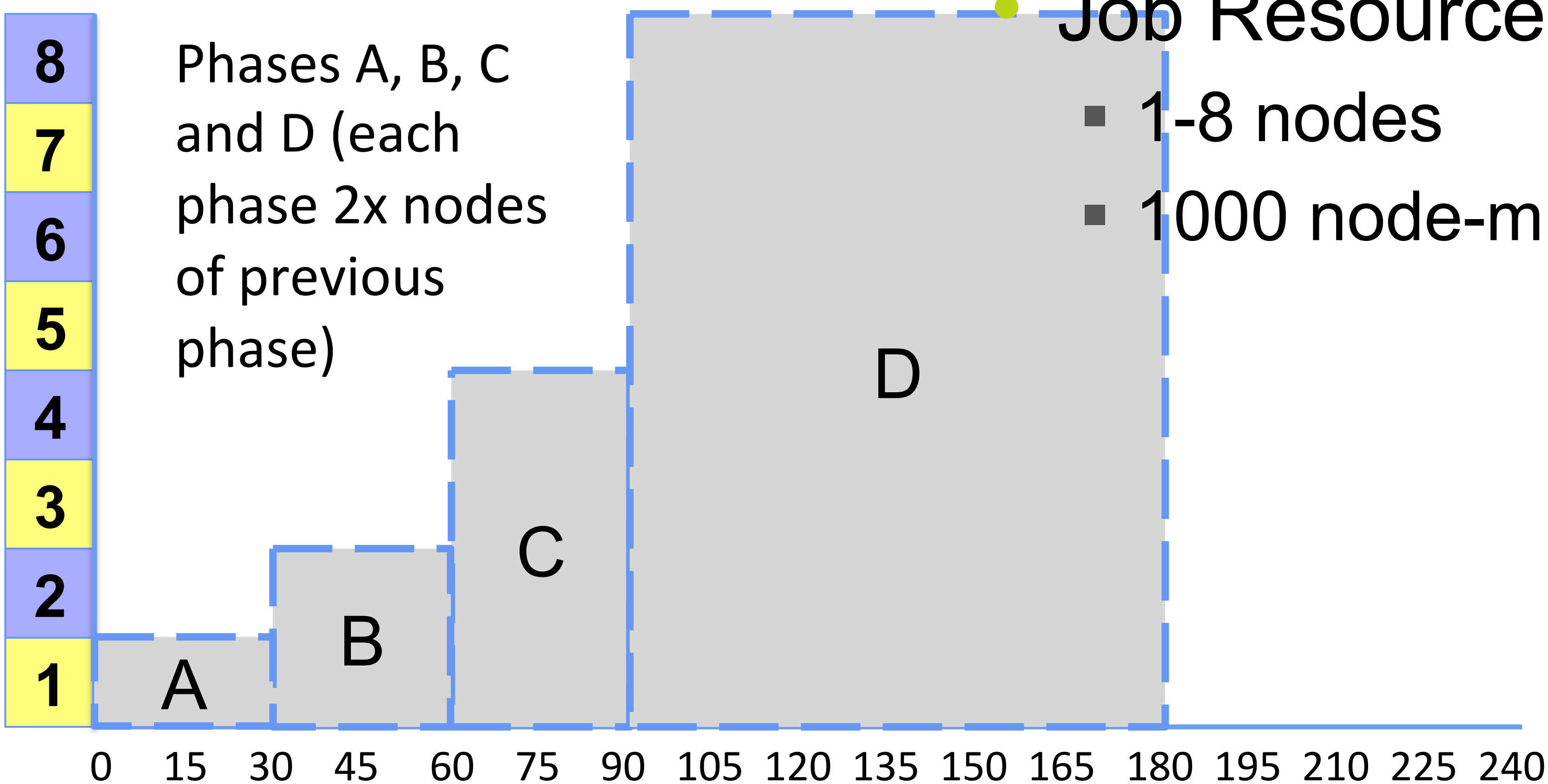
- Job Resources
  - 4 nodes/60 min
  - 2 nodes/120 min
  - 1 node/240 min

Malleable Job

- Job Resources
  - 1-8 nodes
  - 1000 node-min



Evolving Job

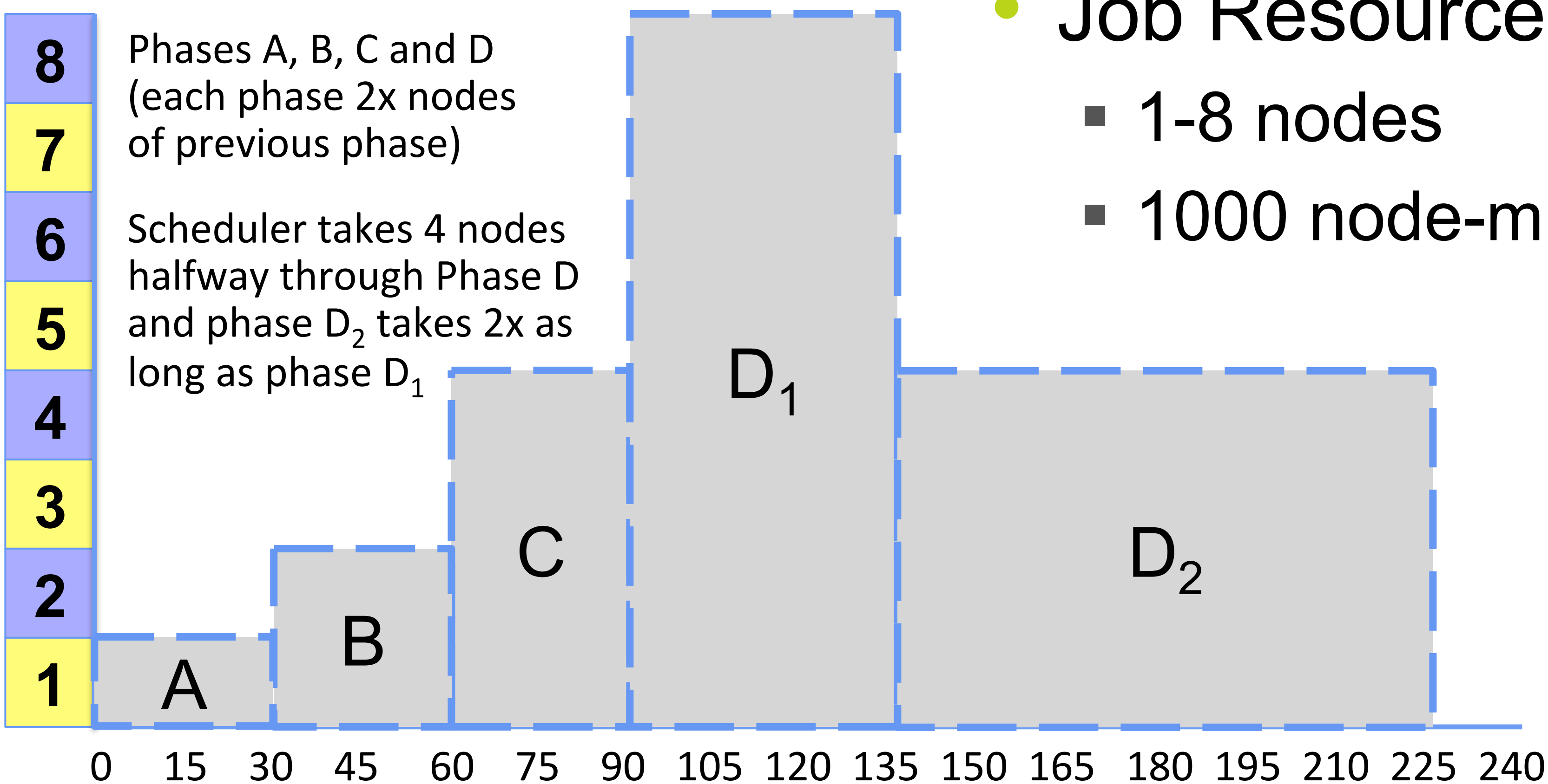


- Job Resources
  - 1-8 nodes
  - 1000 node-min

PMIx 10^{18}

Adaptive Job

- Job Resources
  - 1-8 nodes
  - 1000 node-min



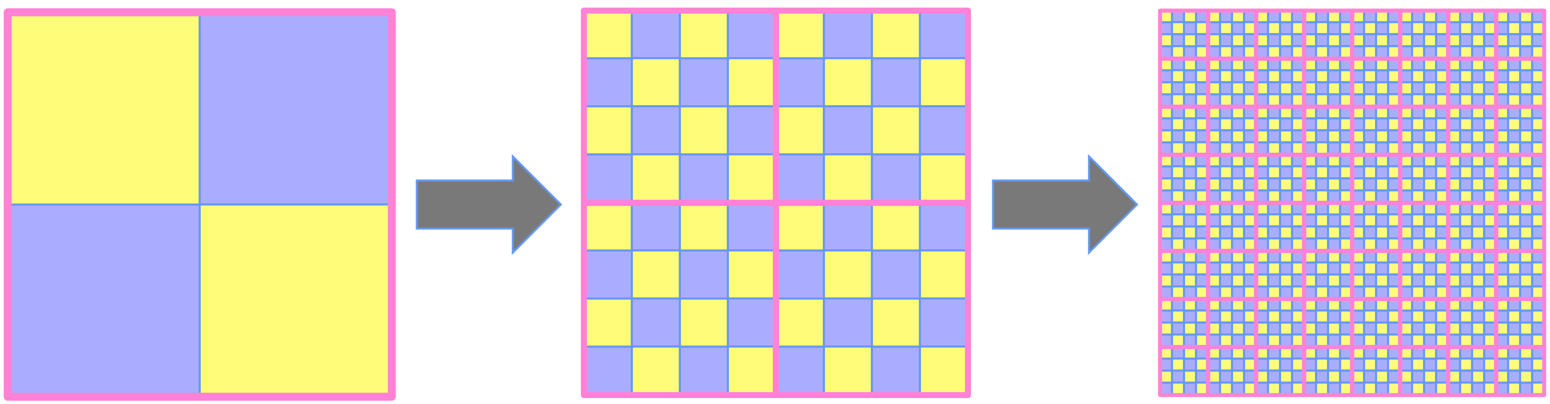
Motivations

- New Programming Models
- New Algorithmic Techniques
- Unconventional Cluster Architectures

Adaptive Mesh Refinement

- Granularity

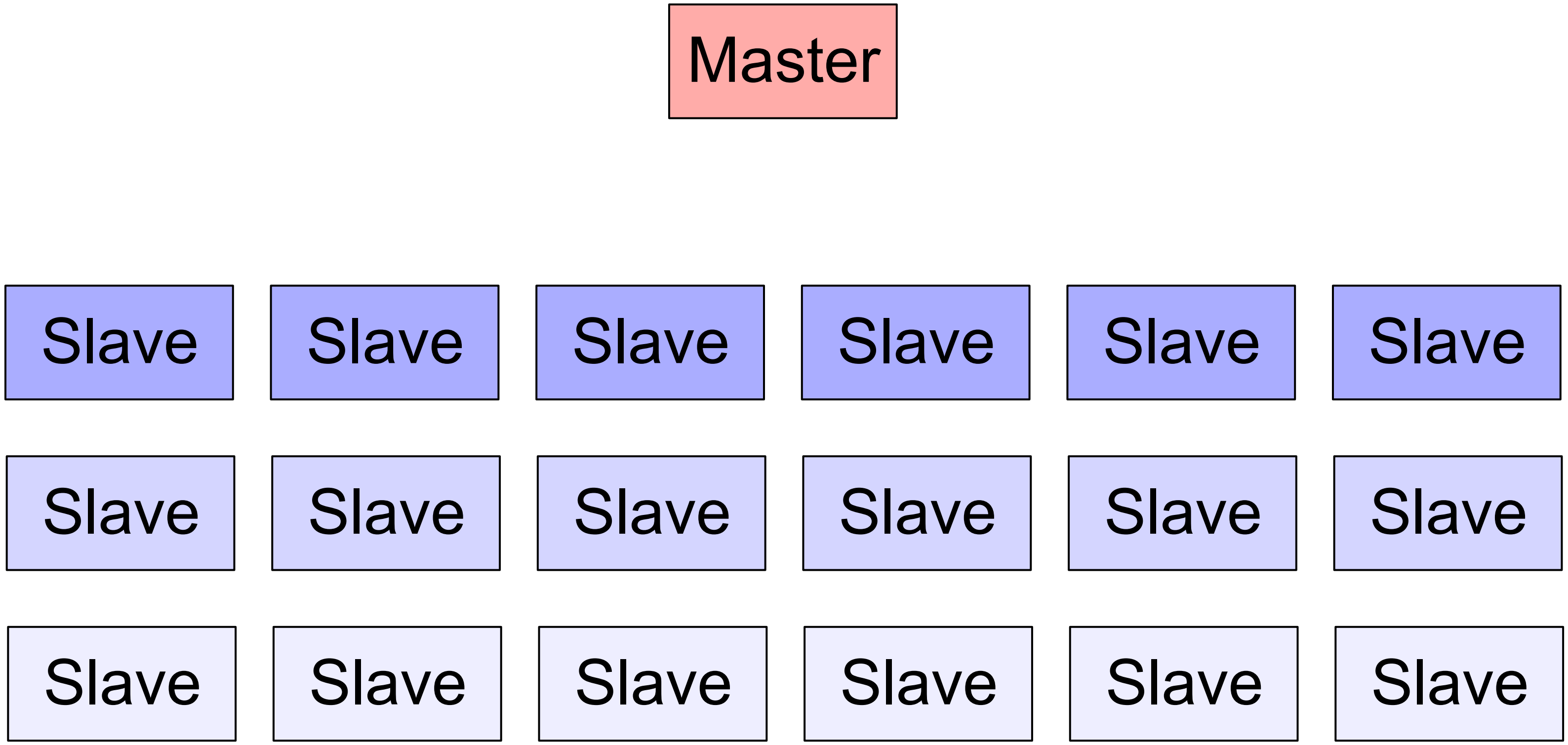
Coarse ➔ Medium ➔ Fine



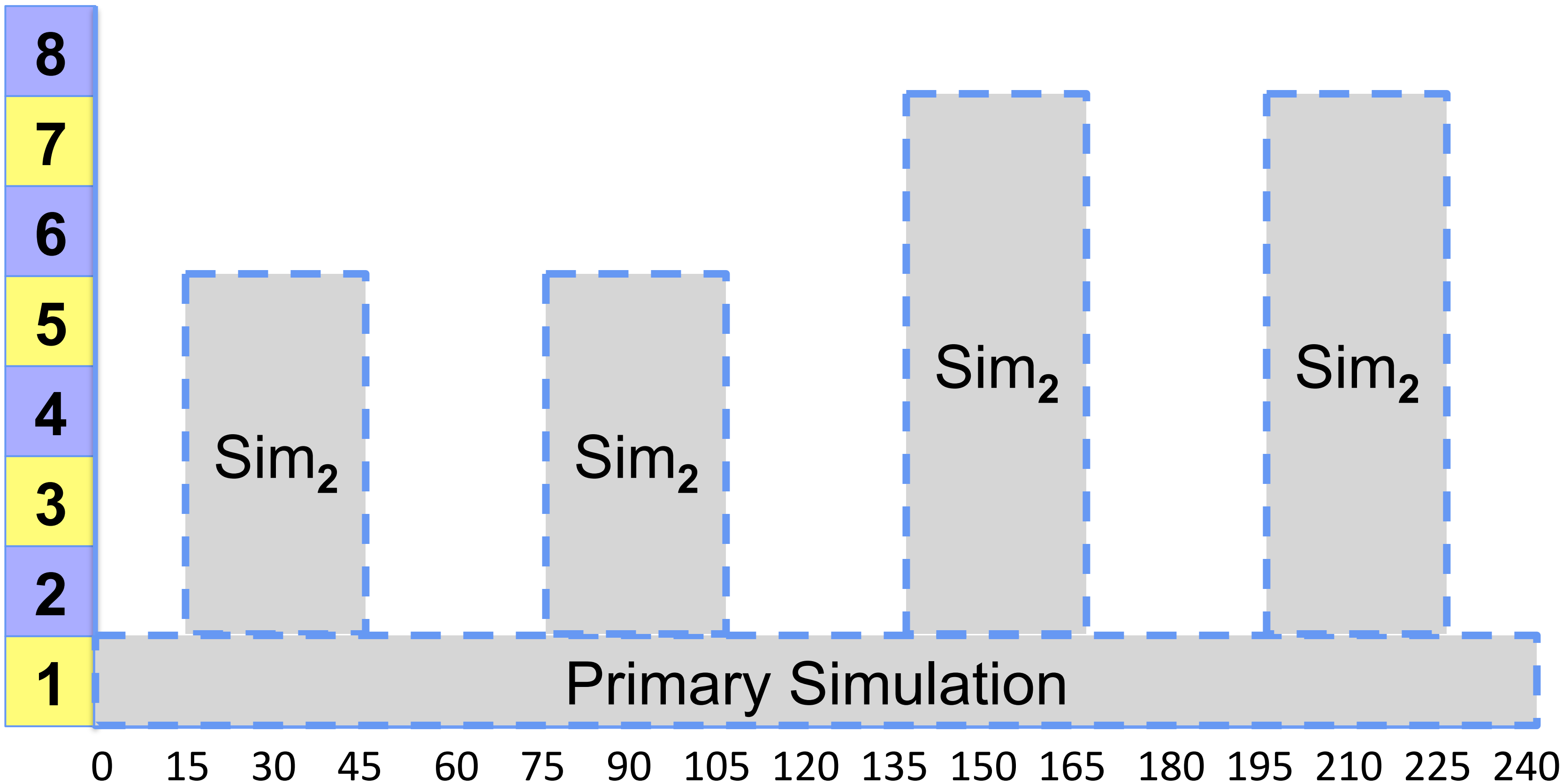
- Node Allocation

Few (1) ➔ Some (4) ➔ Many (64)

Master/Slaves



Secondary Simulations



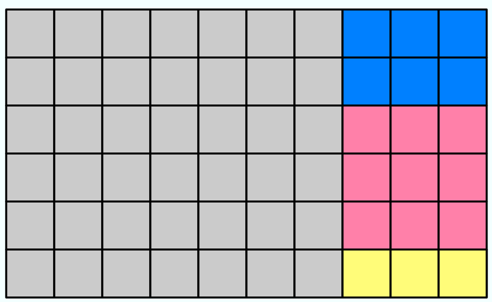
Unconventional Architectures

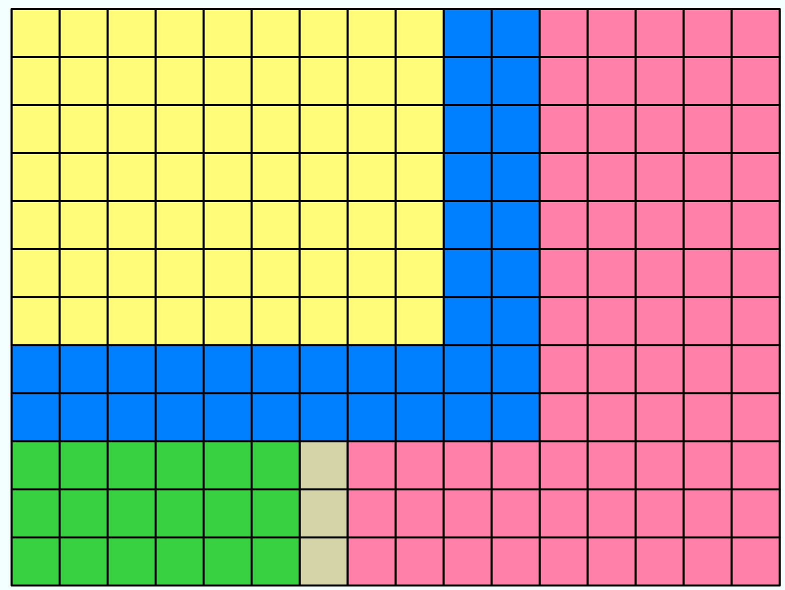
- Cluster Booster

Same Network Domain

Many-core "Booster"

Multi-core "Cluster"





Multi-core jobs dynamically burst out to parallel "booster" nodes with accelerators

Apps, RTEs and Archs

Applications

- Astrophysics
- Brain simulation
- Climate simulation
- Flow solvers (QuadFlow)
- Hydrodynamics (Lulesh)
- Molecular Dynamics (NAMD)
- Water reservoir storage/flow
- Wave propagation (Wave2D)

RTEs

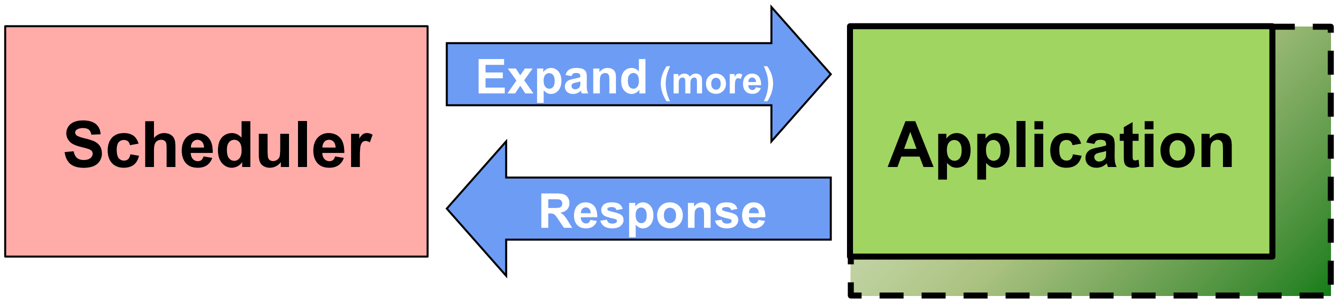
- Charm++
- OmpSs
- Uintah
- Radical-Pilot

Architectures

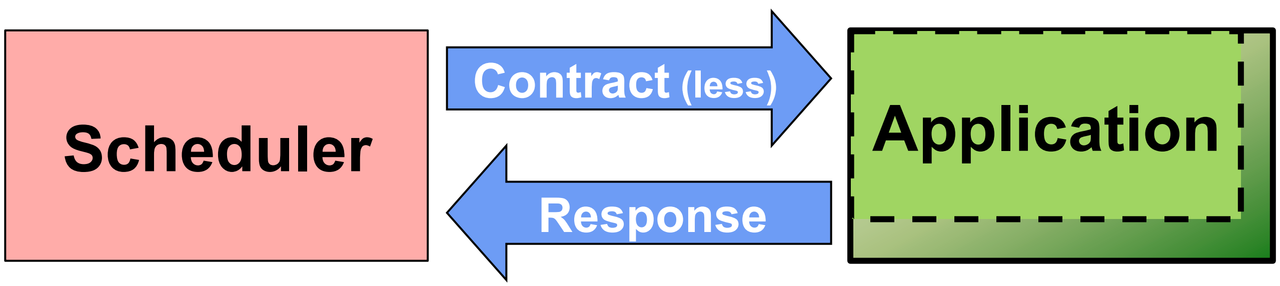
- EU DEEP/DEEP-ER

Scheduler/Malleable Job Dialog

- Expand resource allocation



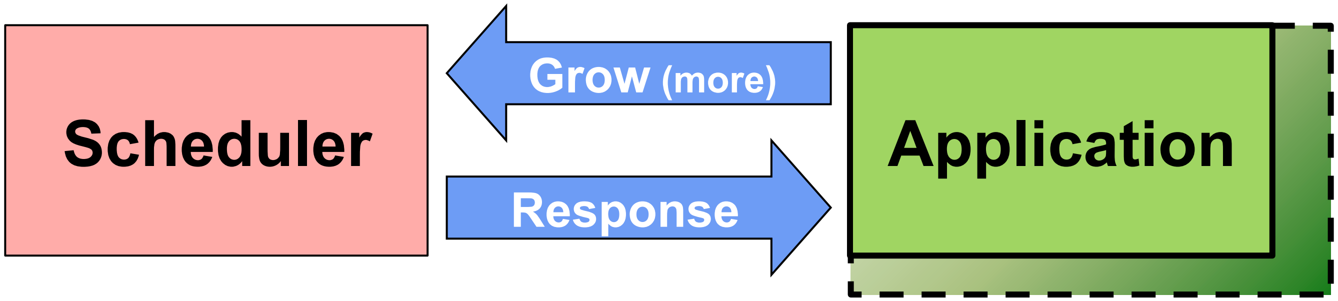
- Contract resource allocation



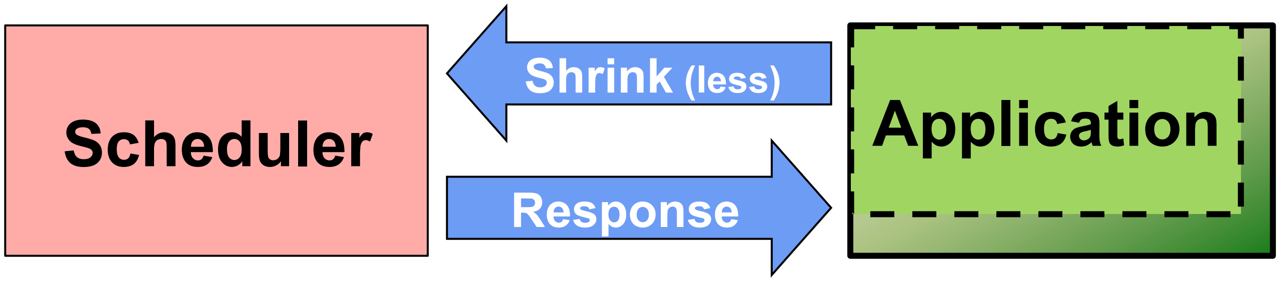
PMIx 10^{18}

Scheduler/Evolving Job Dialog

- Grow resource allocation

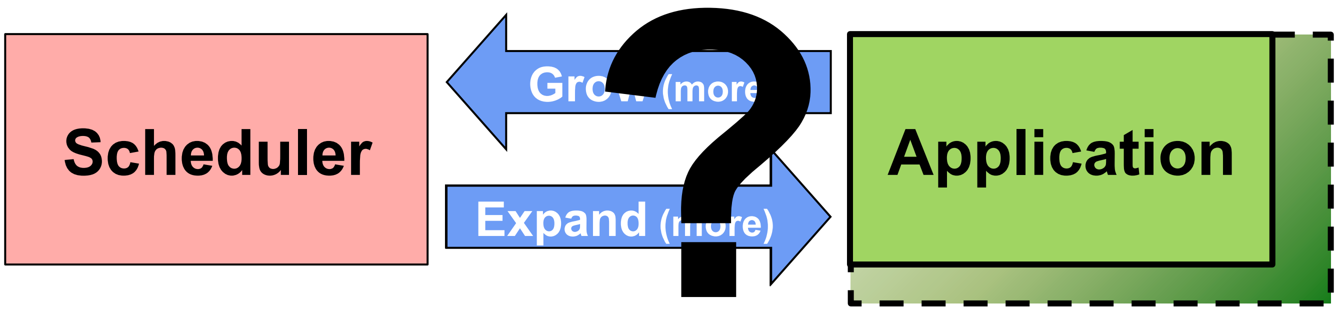


- Shrink resource allocation



Adaptive Job Race Condition

- Reason for naming convention
  - Prevent ambiguity and confusion



Need for Standard API

- MPI: standard API for parallel communication
- Need standard API for application / scheduler resource management dialogs
  - Same API for applications
  - Scheduler-specific API implementations
- Scheduler and Malleable/Evolvable Application Dialog (SMEAD) API
  - Make part of PMIx
  - Need application use cases

Interested Parties

- Adaptive Computing (Moab scheduler, TORQUE RM, Nitro)
- Altair (PBS Pro scheduler/RM)
- Argonne National Laboratory (Cobalt scheduler)
- HLRS at University of Stuttgart
- Jülich Supercomputing Centre (DEEP-ER)
- Lawrence Livermore National Laboratory (Flux scheduler)
- Partec (ParaStation)
- SchedMD (Slurm scheduler/RM)
- TU Darmstadt Laboratory for Parallel Programming
- UK Atomic Weapons Establishment (AWE)
- University of Cambridge COSMOS
- University of Illinois at Urbana-Champaign Parallel Programming Laboratory (Charm++ RTE)
- University of Utah SCI Institute (Uintah RTE)

URLs

- Need your help to design a standard API!
  - Malleable/Evolving Application Use Case Survey
  - http://goo.gl/forms/lq85y3SkV3 (Google Form)

- Info on adaptive job types and scheduling
  - 4-part blog about malleable / evolving / adaptive jobs and schedulers
  - http://www.adaptivecomputing.com/series/malleable-and-evolving-jobs/

Agenda

- Overview
  - Introductions
  - Vision/objectives
- Performance status
- Integration/development status
- Roadmap
- Malleable/Evolving application support
- Wrap-Up/Open Forum

Bottom Line

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

<u>For any programming model</u>

MPI, OSHMEM, PGAS,…

PMIx 10^{18}

Contribute or Follow Along!

- **Project: https://pmix.github.io/master**
- **Code: https://github.com/pmix**

Contributors/collaborators are welcomed!

PMIx 10^{18}