Runtime Coordination Based on PMIx

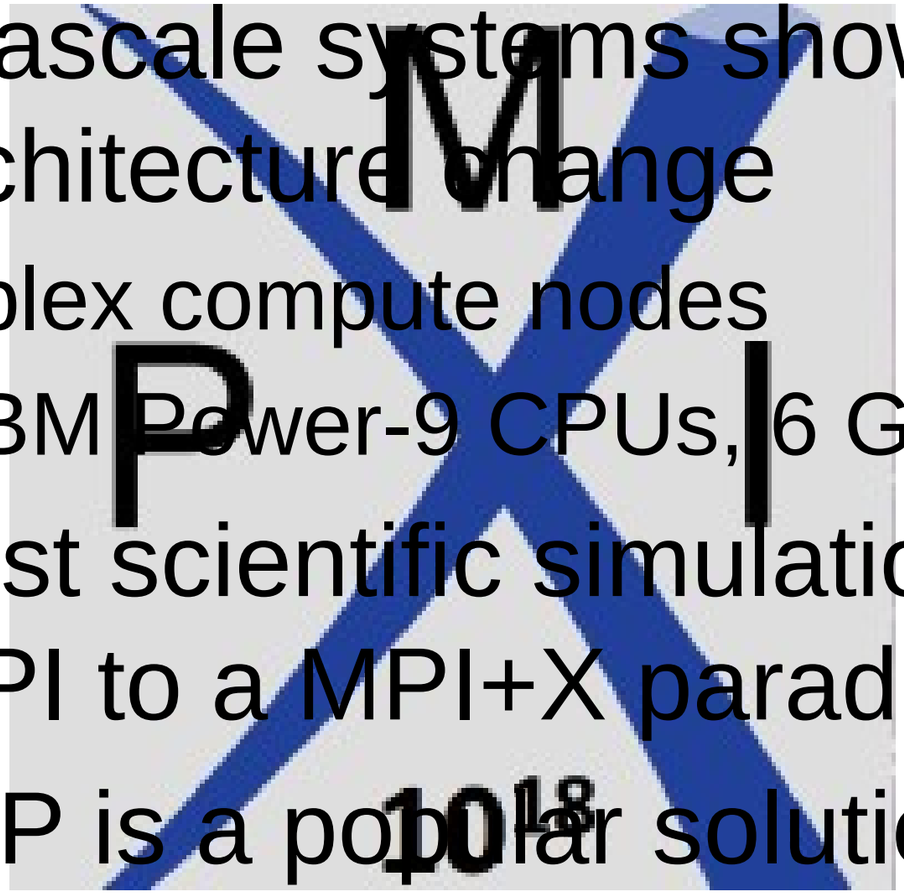
Geoffroy Vallee
Oak Ridge National Laboratory

OAK RIDGE National Laboratory

ECP EXASCALE COMPUTING PROJECT

Introduction

- Many pre-exascale systems show a fairly drastic hardware architecture change
  - Bigger/complex compute nodes
  - Summit: 2 IBM Power-9 CPUs, 6 GPUs per node
- Requires most scientific simulations to switch from pure MPI to a MPI+X paradigm
- MPI+OpenMP is a popular solution



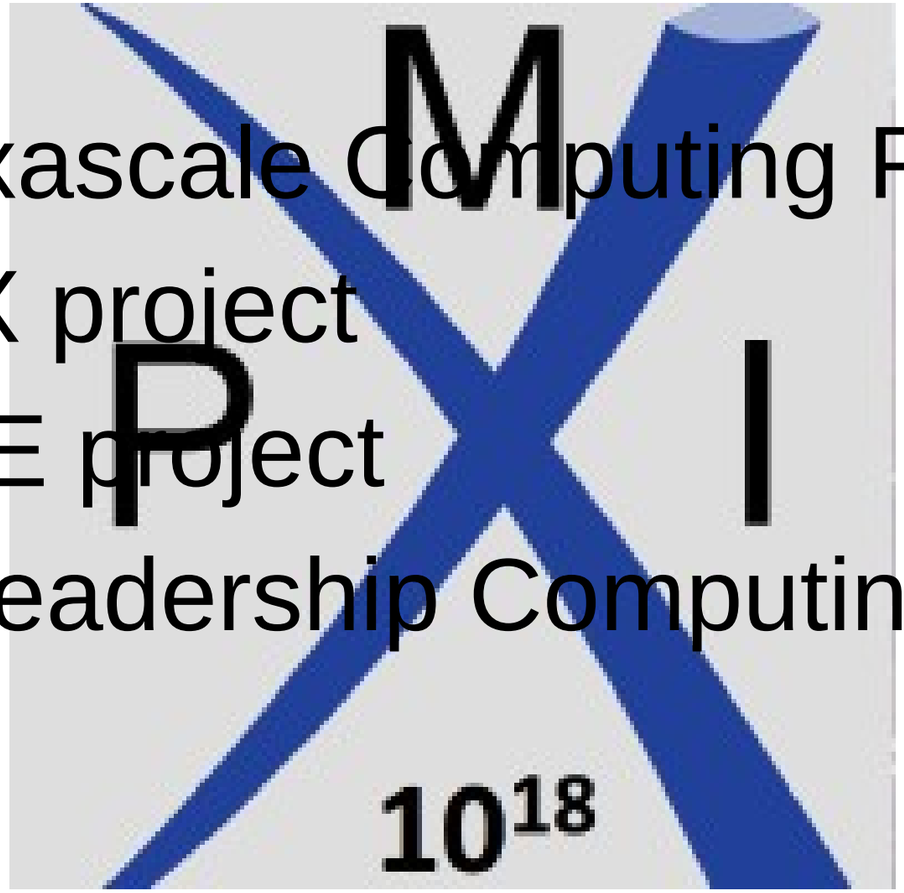
Challenges

- MPI and OpenMP are independent communities
- Implementations un-aware of each other
  - MPI will deploy ranks without any knowledge of the OpenMP threads that will run under each rank
  - OpenMP assumes by default that all available resources can be used

It is very difficult for users to have a fine-grain control over application execution

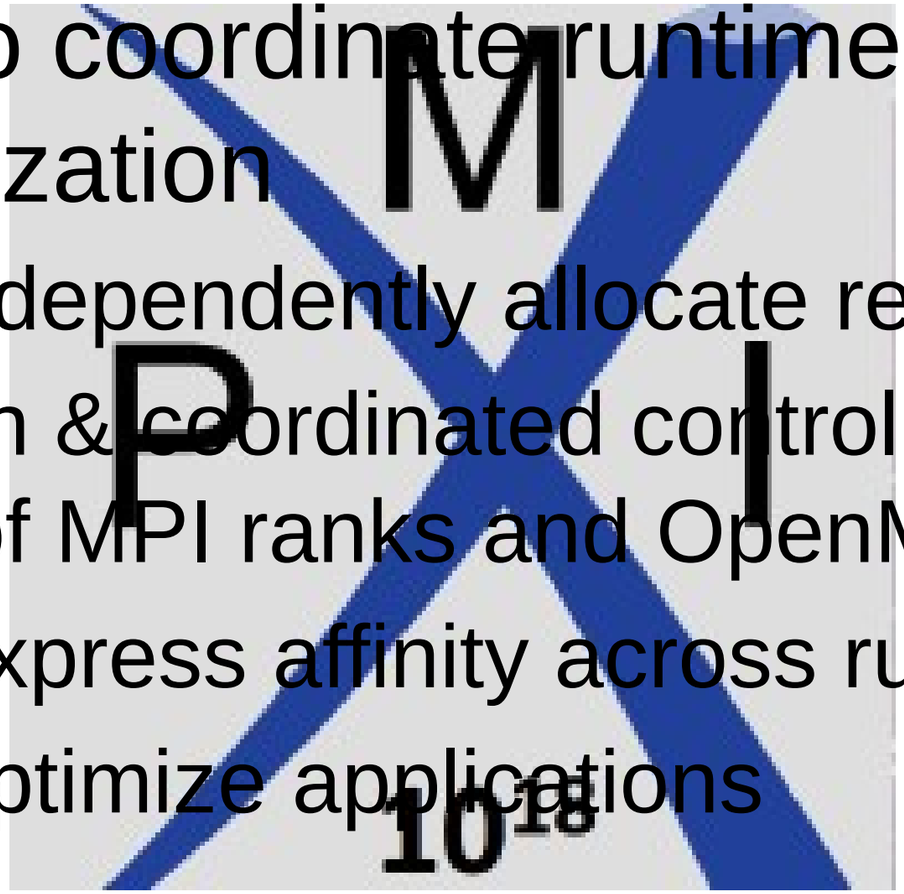
Context

- U.S. DOE Exascale Computing Project (ECP)
- ECP OMPI-X project
- ECP SOLLVE project
- Oak Ridge Leadership Computing Facility (OCLF)



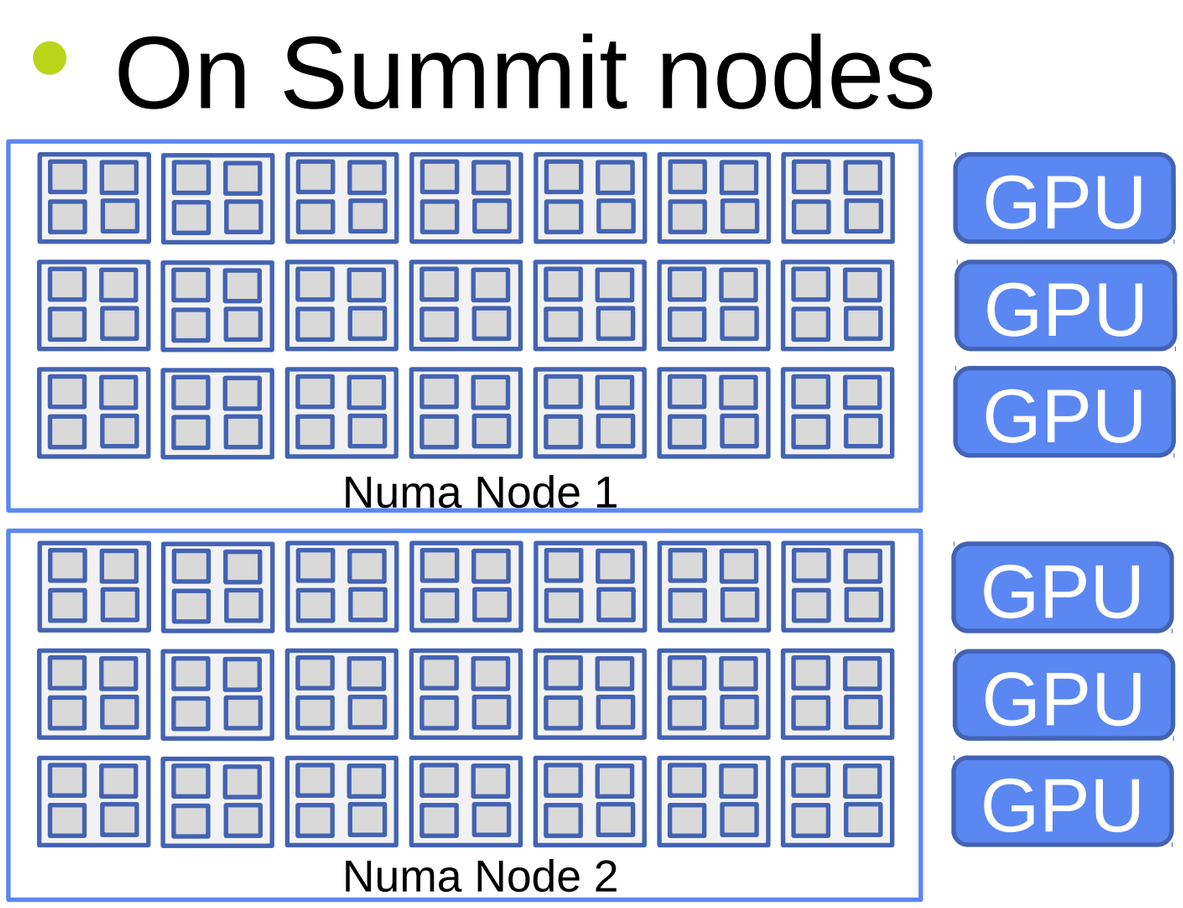
Challenges (2)

- Impossible to coordinate runtimes and optimize resource utilization
  - Runtimes independently allocate resources
  - No fine-grain & coordinated control over the placement of MPI ranks and OpenMP threads
  - Difficult to express affinity across runtimes
  - Difficult to optimize applications

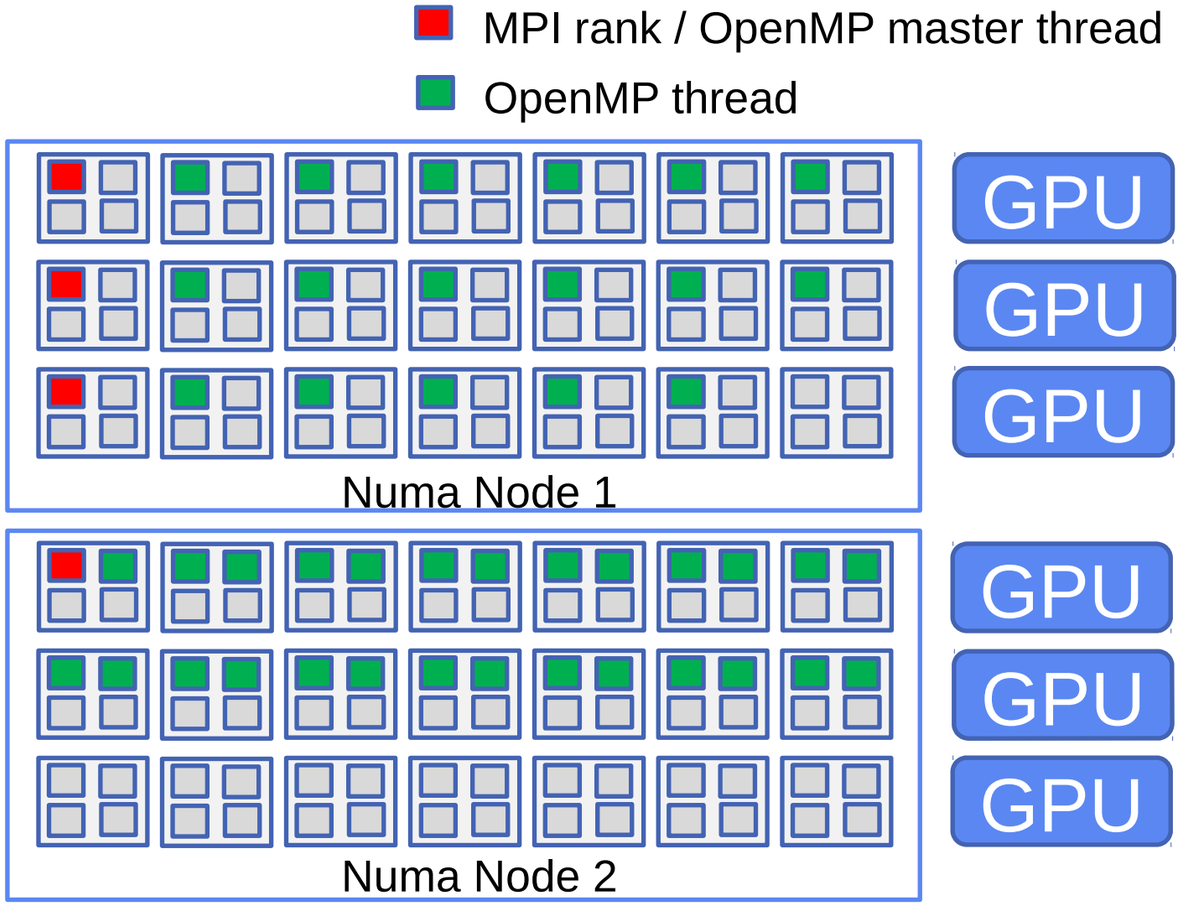


Illustration

- On Summit nodes



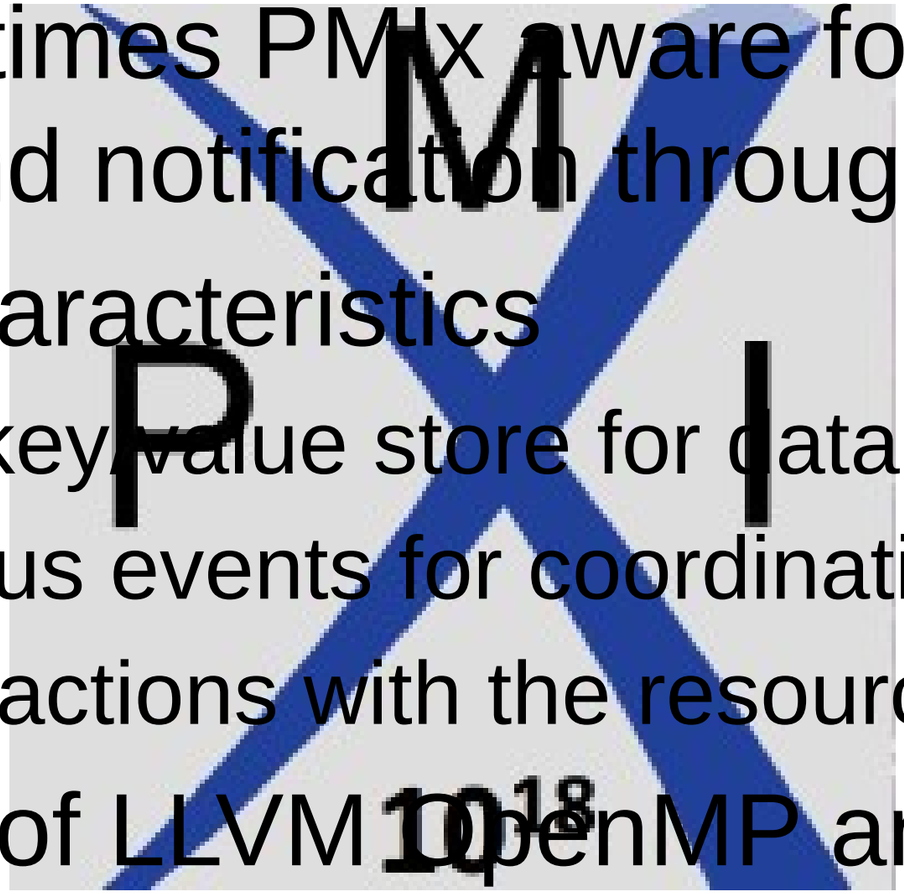
Node architecture



Desired placement

Proposed Solution

- Make all runtimes PMIx aware for data exchange and notification through events
- Key PMIx characteristics
  - Distributed key/value store for data exchange
  - Asynchronous events for coordination
  - Enable interactions with the resource manager
- Modification of LLVM OpenMP and Open MPI



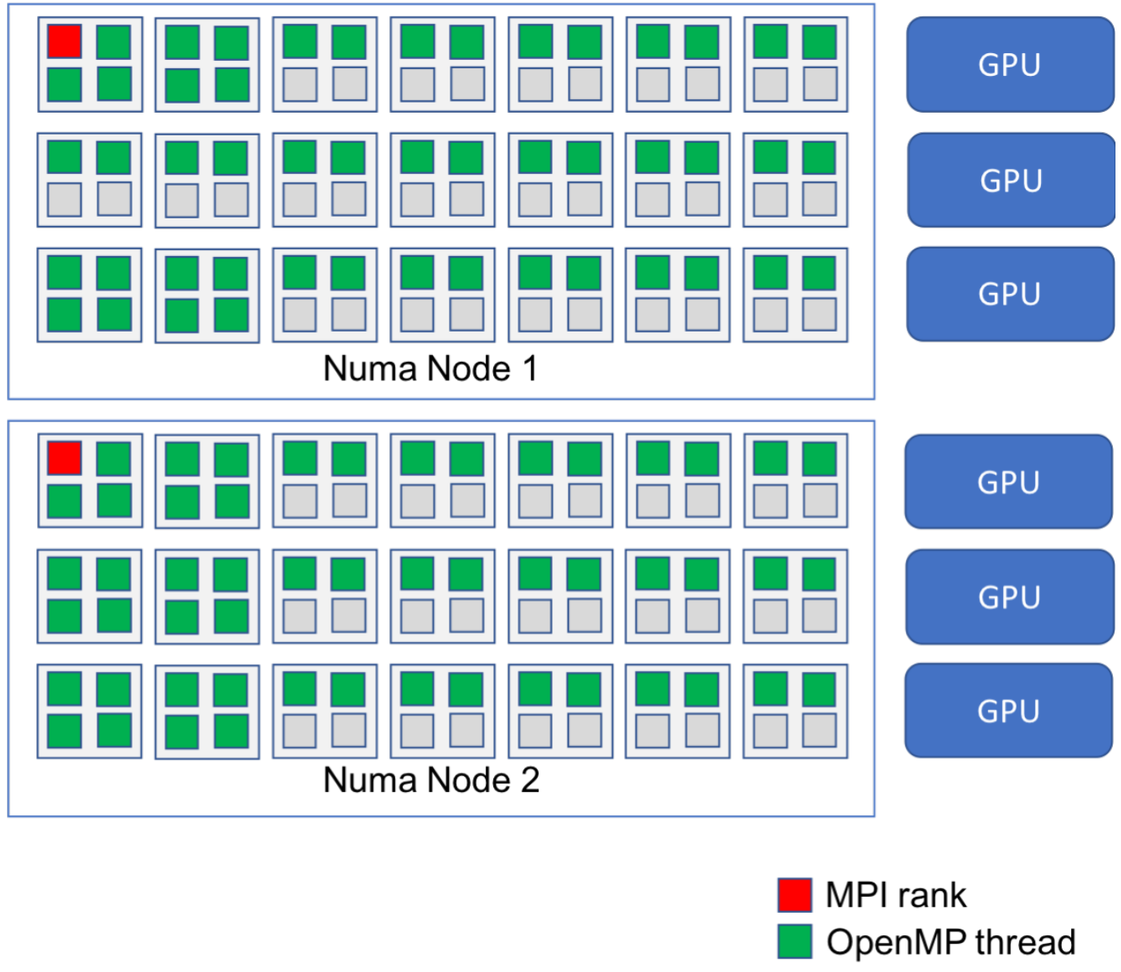
Two Use-cases

- Fine-grain placement over MPI ranks and OpenMP threads (prototyping stage)
- Inter-runtime progress (design stage)

Fine-grain Placement

- Goals
  - Explicit placement of MPI ranks
  - Explicit placement of OpenMP threads
  - Re-configure MPI-OpenMP *layout* at runtime
- Propose the concept of *layout*
  - Static layout: explicit definition of ranks and threads placement at initialization time only
  - Dynamic layouts: change placement at runtime

Layout: Example

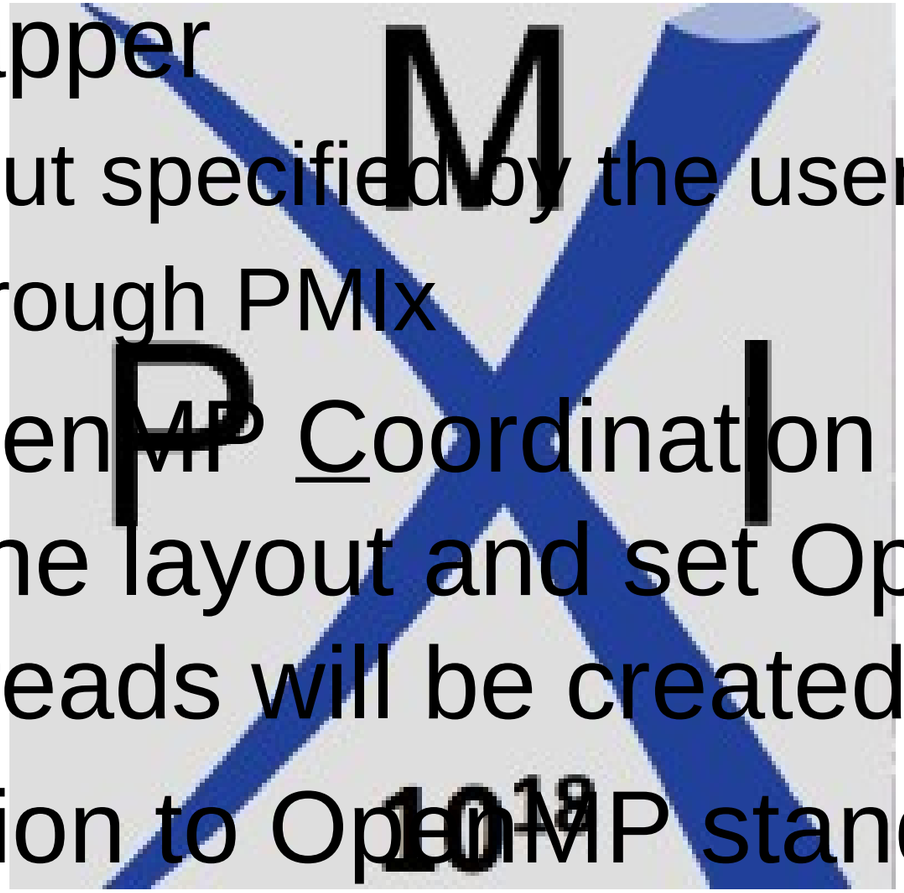


→

```
[MPI,-,Cores,[[0, -, -, -, -, -, -], [-, -, -, -, -, -, -],
                [-, -, -, -, -, -, -], [1, -, -, -, -, -, -],
                [-, -, -, -, -, -, -], [-, -, -, -, -, -, -]]
[OpenMP,MPI-0,HT,[[0,1,2,3],[4,5,6,7],[8,9,-,-],
    [10,11,-,-],[12,13,-,-],[14,15,-,-],[16,17,-,-],
    [18,19,-,-],[20,21,-,-],[22,23,-,-],[24,25,-,-],
    [26,27,-,-],[28,29,-,-],[30,31,-,-],[32,33,34,35],
    [36,37,38,39],[40,41,-,-],[42,43,-,-],[44,45,-,-],
    [46,47,-,-],[48,49,-,-]]
[OpenMP,MPI-1,HT, [[0,1,2,3],[4,5,6,7],[8,9,-,-],
    [10,11,-,-],[12,13,-,-],[14,15,-,-],[16,17,-,-],
    [18,19,21,22],[23,24,25,26],[27,28,-,-],
    [29,30,-,-],[31,32,-,-],[33,34,-,-],[35,36,-,-],
    [37,38,39,40],[41,42,43,44],[45,46,-,-],
    [47,48,-,-],[49,50,-,-],[51,52,-,-],[53,54,-,-]]
```

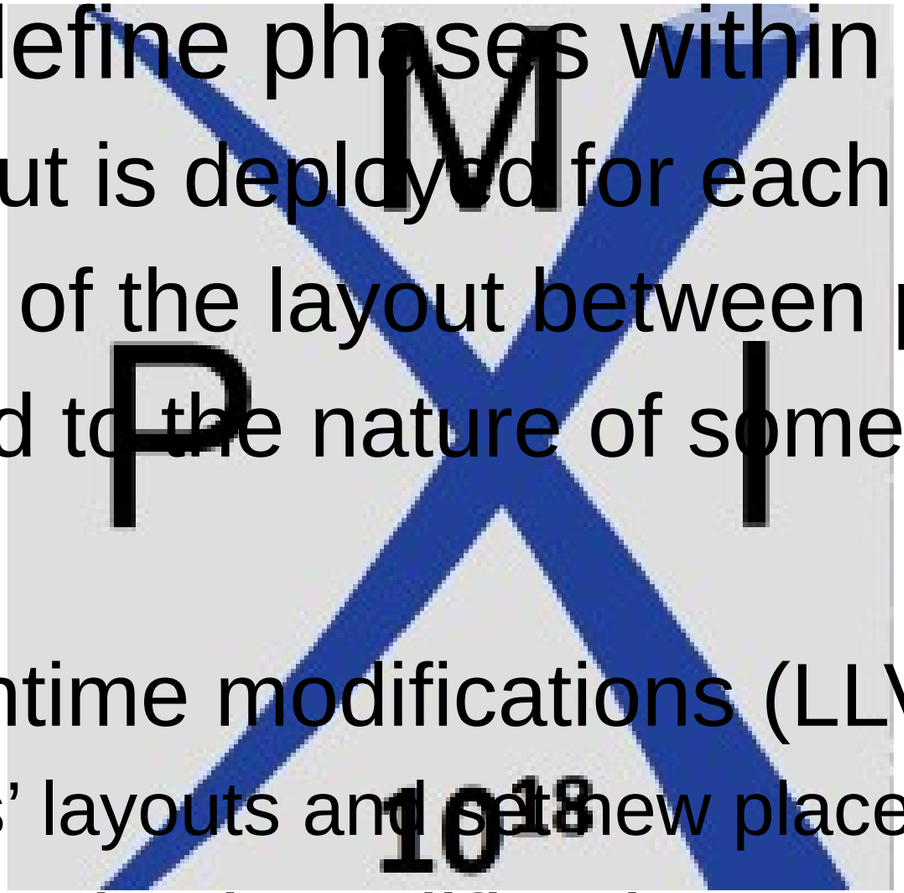
Static Layouts

- New MPI mapper
  - Get the layout specified by the user
  - Publish it through PMIx
- New MPI OpenMP Coordination (MOC) helper-library gets the layout and set OpenMP places to specify threads will be created placement
- No modification to OpenMP standard or runtime

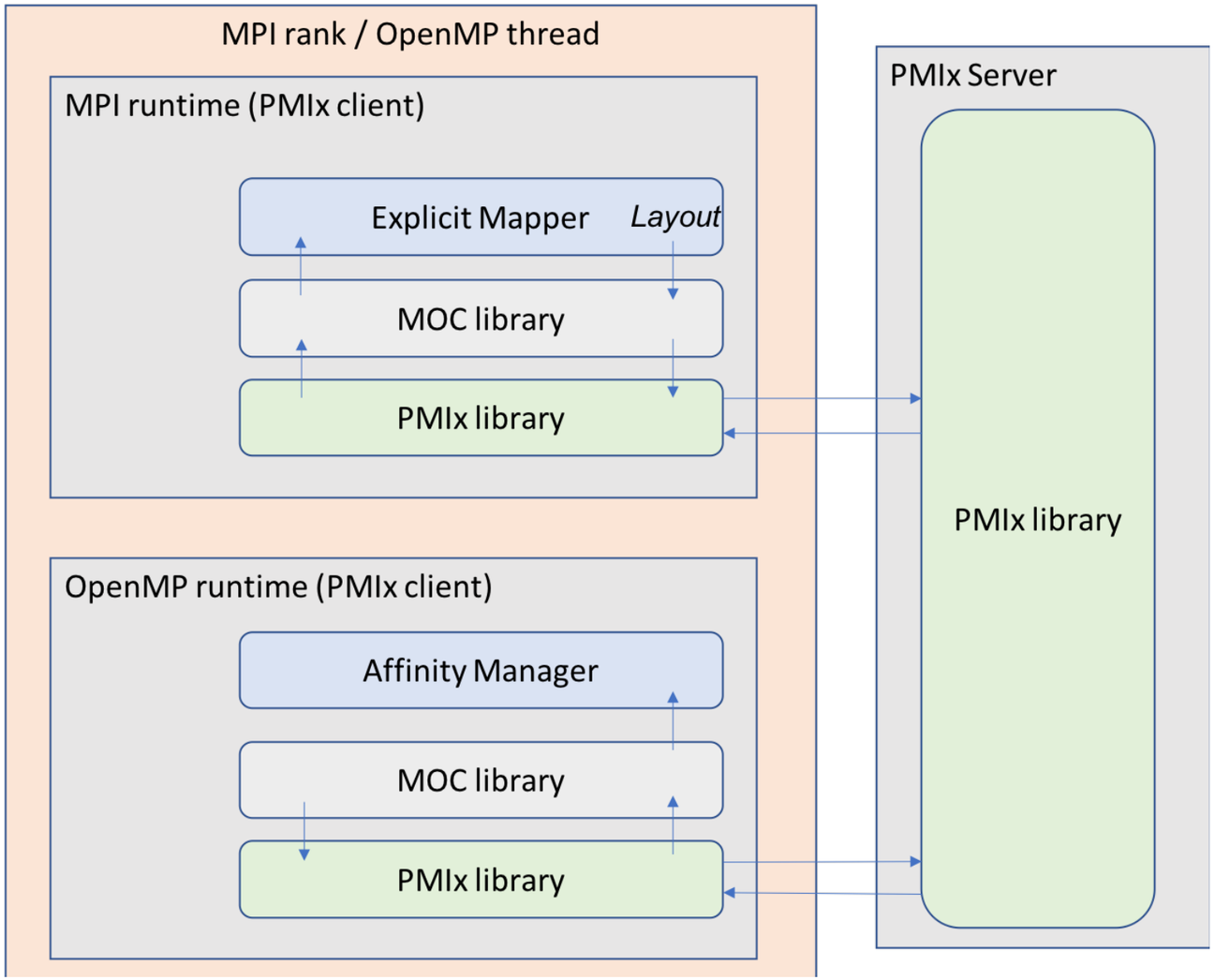


Dynamic Layouts

- New API to define phases within the application
  - A static layout is deployed for each phase
  - Modification of the layout between phases (w/ MOC)
  - Well adapted to the nature of some applications
- Requires
  - OpenMP runtime modifications (LLVM prototype)
    - Get phases' layouts and set new places internally
  - OpenMP standard modifications to allow dynamicity



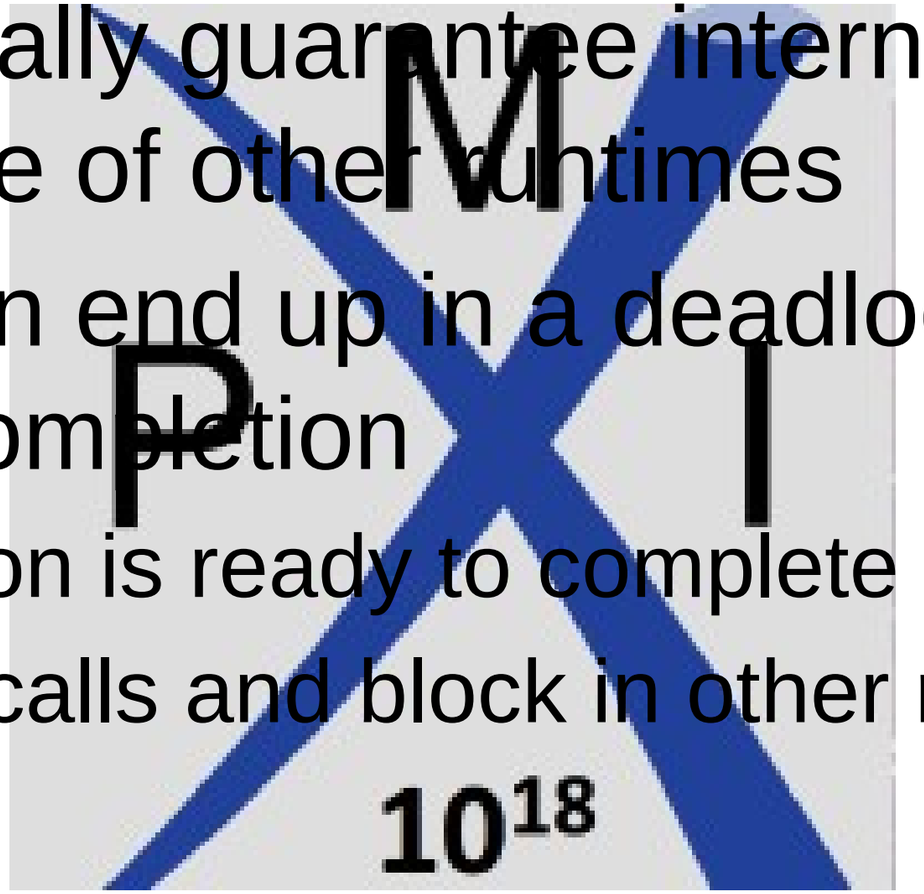
Architecture Overview



- MOC ensures inter-runtime data exchanged (MOC-specific PMIx keys)
- Runtimes are PMIx-aware
- Runtimes have a layout-aware mapper/affinity manager
- Respect the separation between standards, resource manager and runtimes

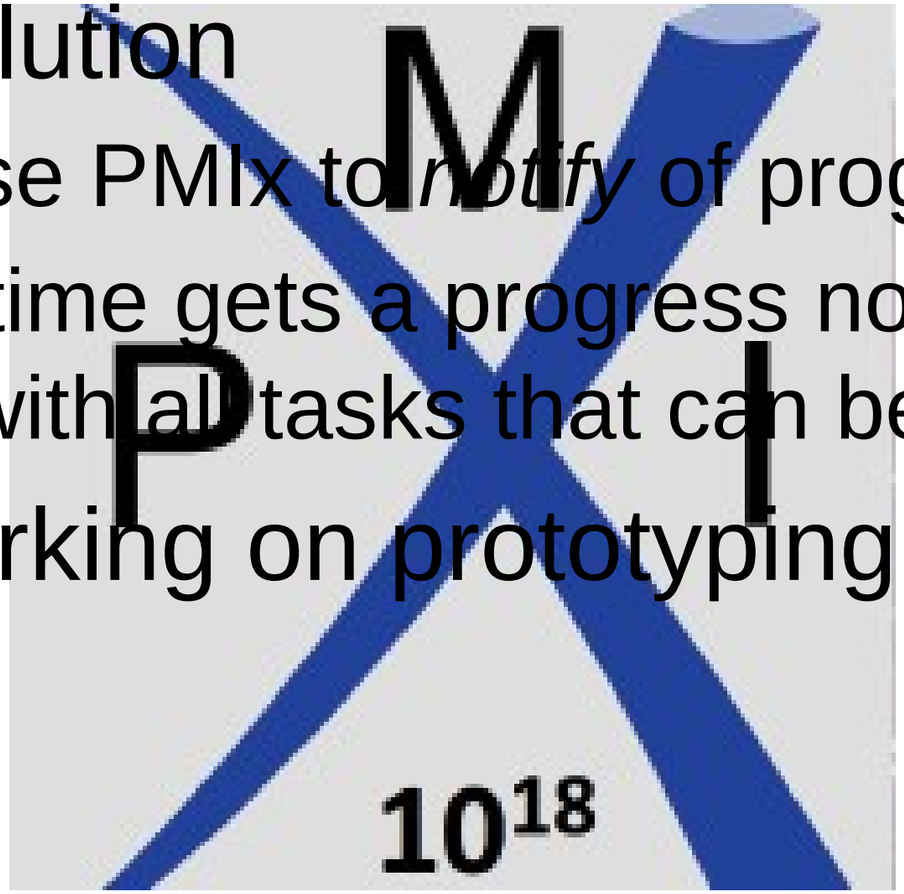
Inter-runtime Progress

- Runtime usually guarantee internal progress but are not aware of other runtimes
- Runtimes can end up in a deadlock if mutually waiting for completion
  - MPI operation is ready to complete
  - Application calls and block in other runtimes



Inter-runtime Progress (2)

- Proposed solution
  - Runtimes use PMIx to *notify* of progress
  - When a runtime gets a progress notification, it yields once done with all tasks that can be completed
- Currently working on prototyping



Conclusion

- There is a really need for making runtimes aware of each other
- PMIx is a privileged option
  - Asynchronous / event based
  - Data exchange
  - Enable interaction with the resource manager
- Opens new opportunities for both research and development (e.g. new project focusing on QoS)