Experiences with a Slurm and MPICH based Malleability Prototype

PMIx ASC Quarterly Meeting – 26.10.2021

Isaías Comprés – TUM

Motivation

- Algorithms with scalability ceilings that change during execution
  - Holding underutilized or not utilized resources
  - Increased scalability potential after launch

- Fixed allocations a limitation for schedulers
  - Less degrees of freedom for node utilization system goals
  - Idle nodes due to set of job requirements not fitting neatly

MPI Extension Overview

`MPI_Init_adapt``(...)`

- Initializes the library in invasive mode

`MPI_Probe_adapt``(...)`

- Probes the resource manager for adaptations

`MPI_Comm_adapt_begin``(...)`

- Provides a set of helper communicators

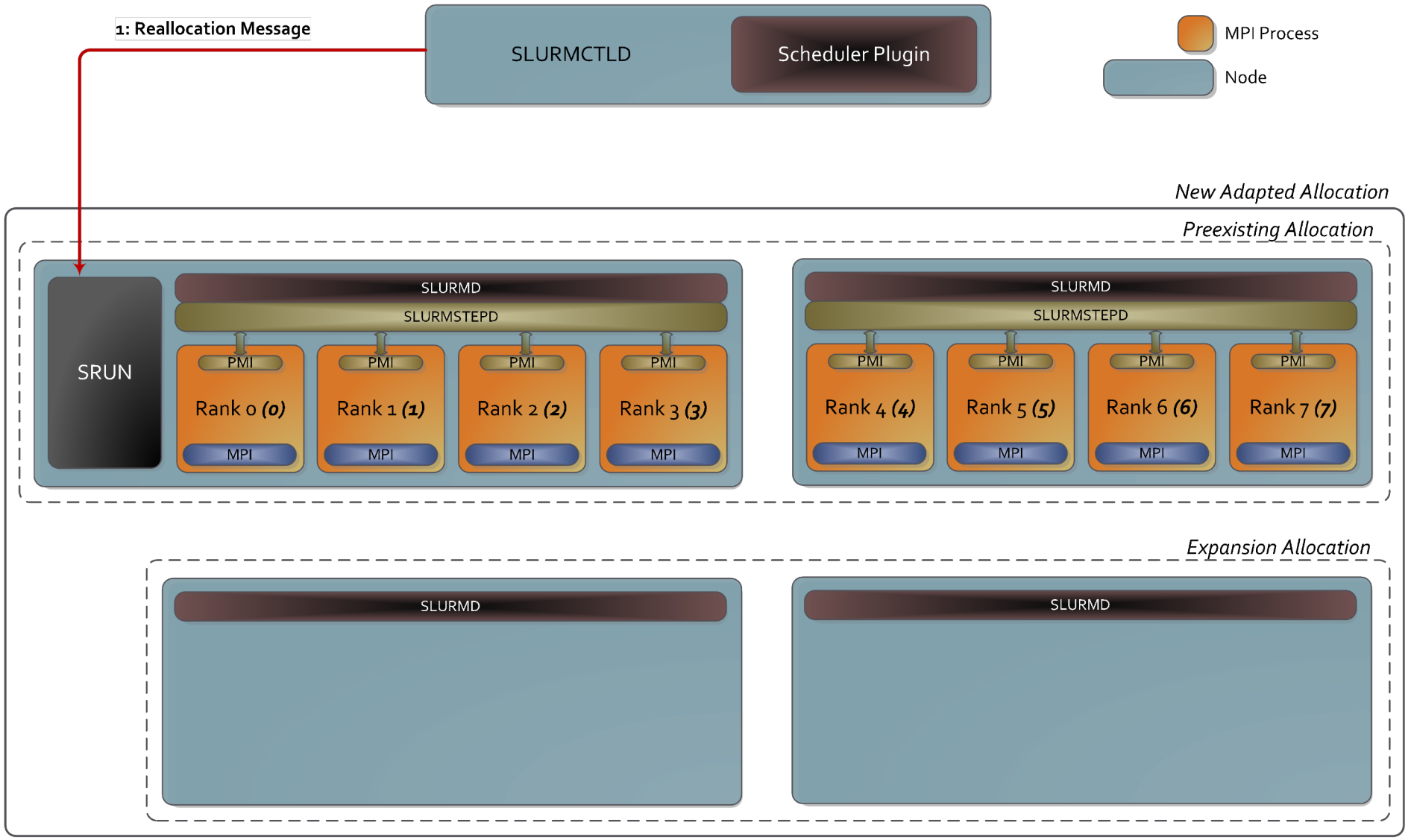
`MPI_Comm_adapt_commit``(...)`

- Sets adapted `MPI_COMM_WORLD`

Code Structure

```
MPI_Init_adapt(…, &status);
for (…){
    MPI_Probe_adapt(&adapt,…);
    if(adapt){
        MPI_Comm_adapt_begin(…);
        // redistribution code
        MPI_Comm_adapt_commit(…);
    }
    // compute and MPI code
}
```

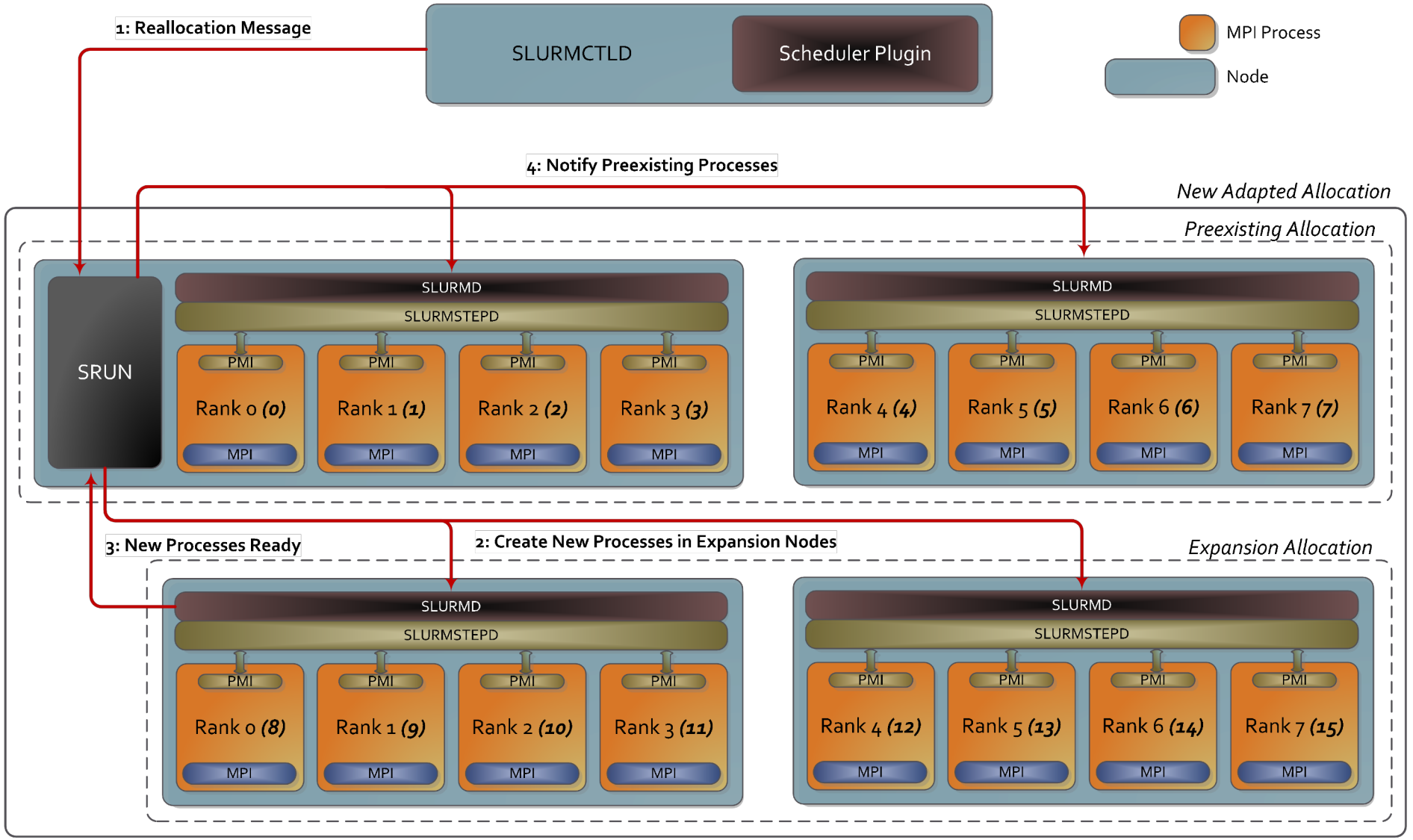
Adaptation Step 1



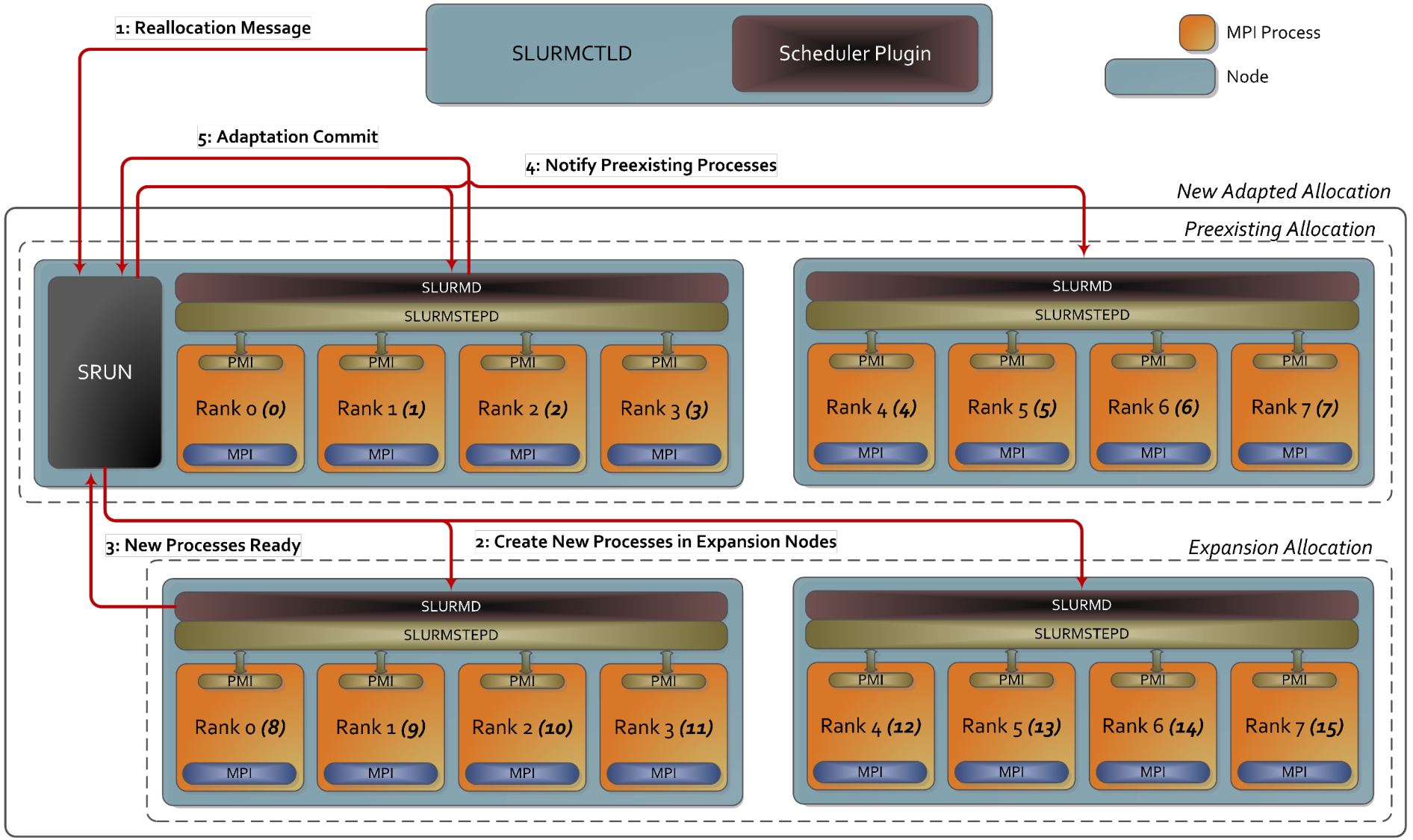
Adaptation Step 2

Adaptation Step 3

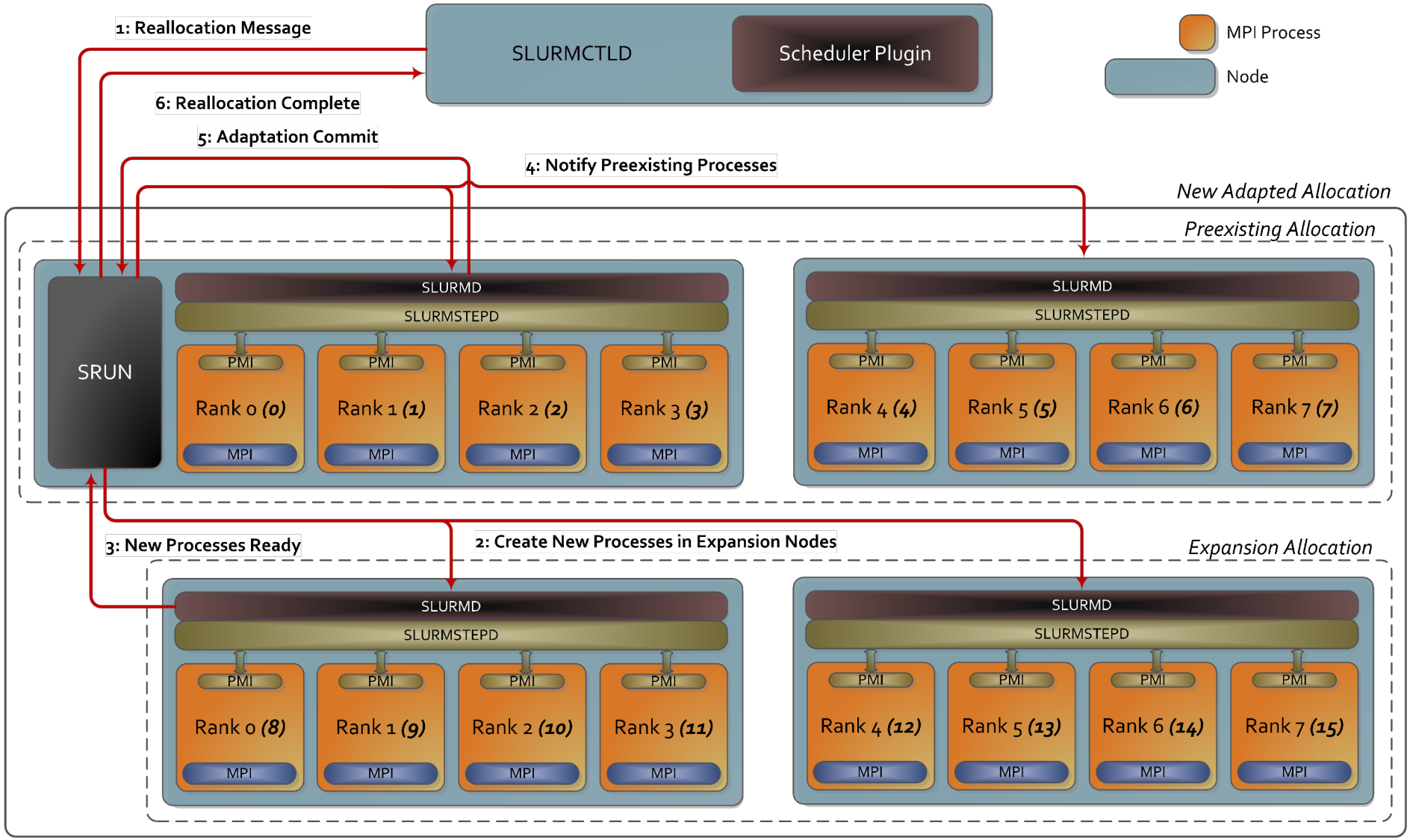
Adaptation Step 4



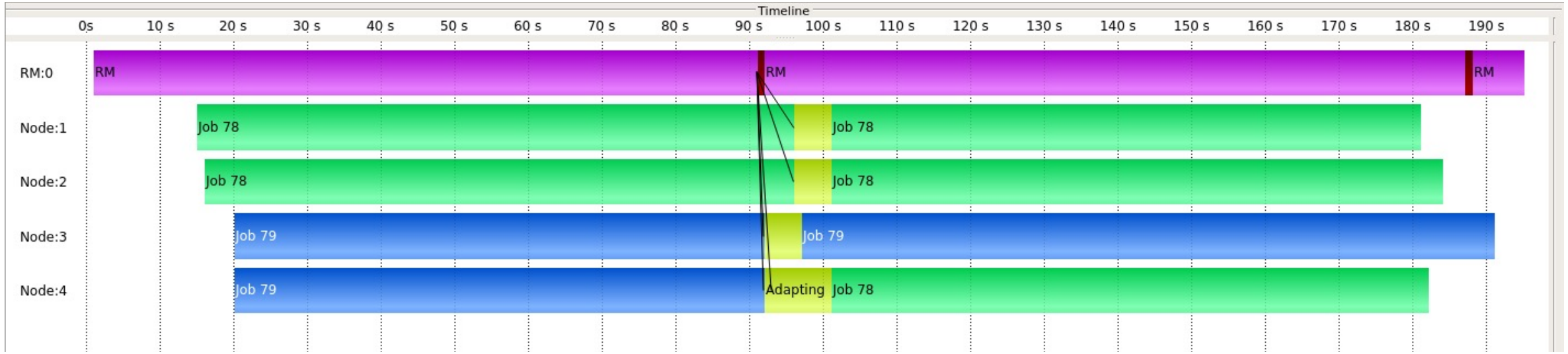
Adaptation Step 5



Adaptation Step 6



Adaptation Visualization Tool



PMIx Importance in Malleability

- Process creation and bootstrap occurs multiple times
  - In static allocations, it is only once

- Feedback and control mechanisms are necessary
  - In the static case, once the application starts, it runs to completion
  - In the malleable case, there is interaction between applications and RM components

- Malleability in multiple programming models
  - Some programming models are static today, due to MPI dependency
  - PMIx as a common interface with Tools and Workload Managers
  - Drop the MPI dependency in some cases

Malleability in PMIx

- We are looking at this topic in the Tools WG
  - Agreement in a transaction-like API so far
  - RM-driven and application-driven malleability
    - *In both cases the application initiates transactions to perform the resource exchange*
    - *The difference is only during negotiation*

- Prototype implementations within the DEEP-SEA project
  - Likely that the PMIx APIs are sufficient
  - Slurm, Open PMIx, MPICH and Open MPI in the stack

Summary

Alternative to MPI Dynamic Processes:

- Initialization, probing for adaptations, and transacction
- Allows for latency hiding designs (from preexisting processes)
- Overheads dominated by domain redistribution

DEEP-SEA European research project:

- Up to date software stack: Slurm, Open PMIx, Open MPI and MPICH
- Follow developments of the MPI Sessions WG malleability efforts

References:

https://doi.org/10.1145/2966884.2966917
https://doi.org/10.1145/3075564.3075585