MPI Malleability
Extending a production MPI implementation

Dr. Sonja Happ – ParTec AG

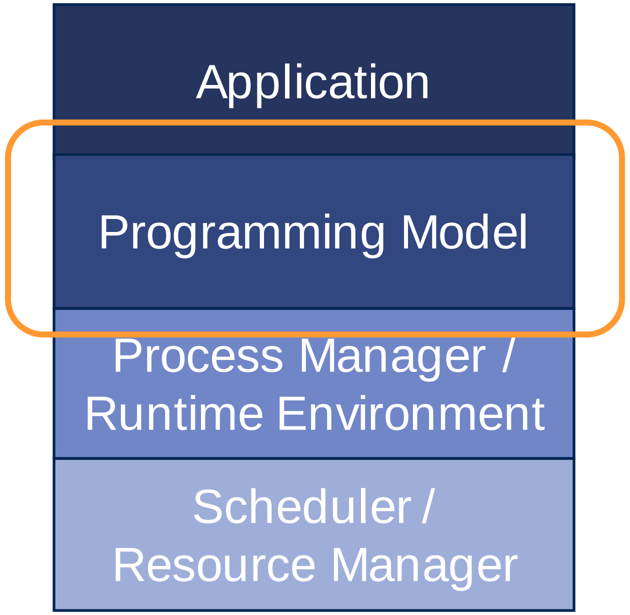
HiPEAC 2024

Computational Malleability in DEEP-SEA

- All layers of the HPC software stack involved
- DEEP-SEA enhances all layers with malleability features
  - Not all efforts can be covered here!

- Focus: MPI programming model
  - API and MPI library extensions
  - Interface with process manager

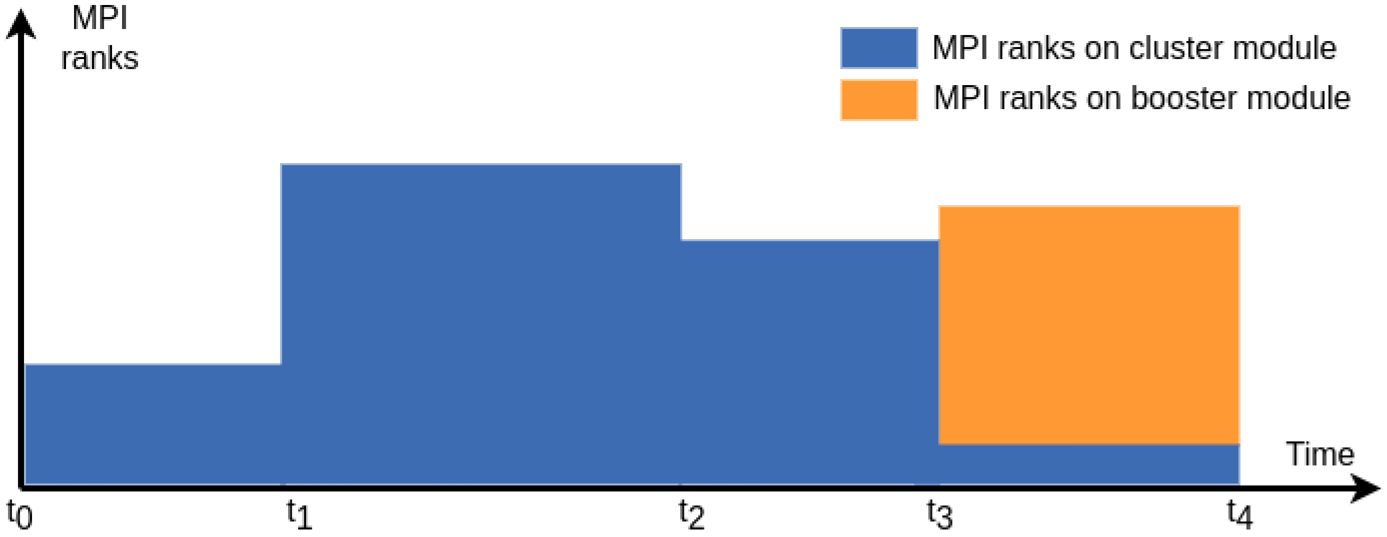
- Close collaboration with other EuroHPC19 projects


Application
Programming Model
Process Manager / Runtime Environment
Scheduler / Resource Manager

Malleability for MPI Applications

- Dynamic adaptation of number of MPI ranks
  - External constraints
  - Computational phases
  - Modular Supercomputing Architecture (MSA)

- Challenges
  - MPI interface for malleability
  - Exchange with process manager
  - Data re-distribution



Potential scenario:

t_0: Launch app with MPI ranks on cluster module

t_1: Expand app to use more ranks on cluster module (e.g. new app phase)

t_2: Shrink app to use less ranks on cluster module (e.g. energy constraint)

t_3: Shrink app to use much less ranks on cluster module and launch MPI ranks on booster module (e.g. new app phase)

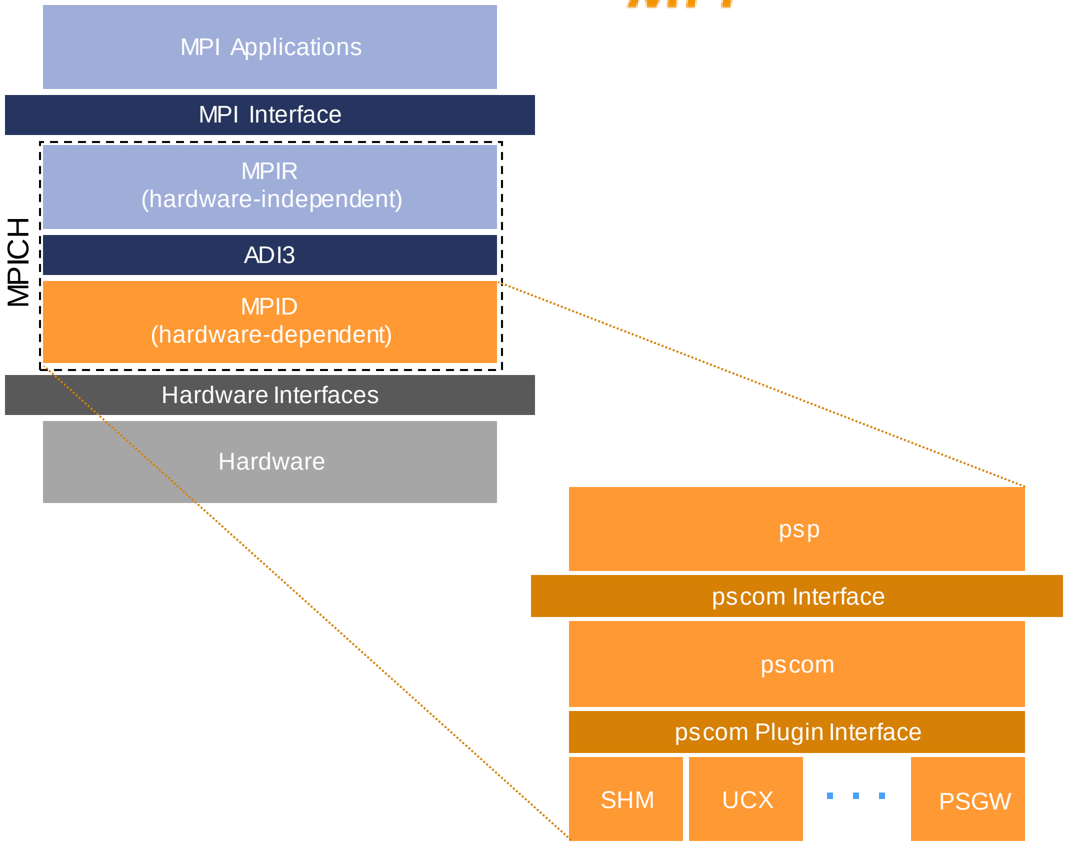
t_4: Finish app

ParaStation MPI

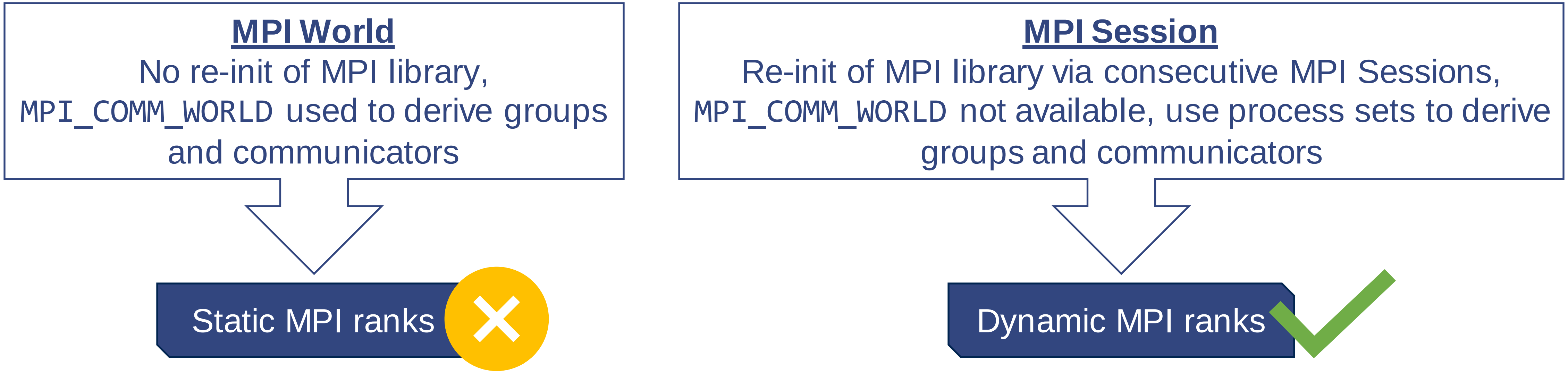
- Based on MPICH 4.1.2
  - Support MPICH tools for tracing, debugging, etc.
  - Integrates into MPICH on the MPID layer by implementing an ADI3 device
  - The PSP Device is powered by pscom – a low-level point-to-point communication library
  - Support the MPICH ABI Compatibility Initiative

- Support for various transports / protocols via pscom plugins
  - InfiniBand, Omni-Path, BXI, etc.
  - Concurrent usage of different transports

- Proven scalability



MPI Sessions for Malleability



MPI World
No re-init of MPI library, `MPI_COMM_WORLD` used to derive groups and communicators

Static MPI ranks ✗

MPI Session
Re-init of MPI library via consecutive MPI Sessions, `MPI_COMM_WORLD` not available, use process sets to derive groups and communicators

Dynamic MPI ranks ✓

- Exploit re-initialization ability of MPI Sessions
  - Re-init MPI library for changed processes
  - Update process sets during re-init to reflect changed MPI ranks

- Requirements for malleable MPI applications
  - Must use MPI Session model and process sets
  - Must not use MPI world model and `MPI_COMM_WORLD`

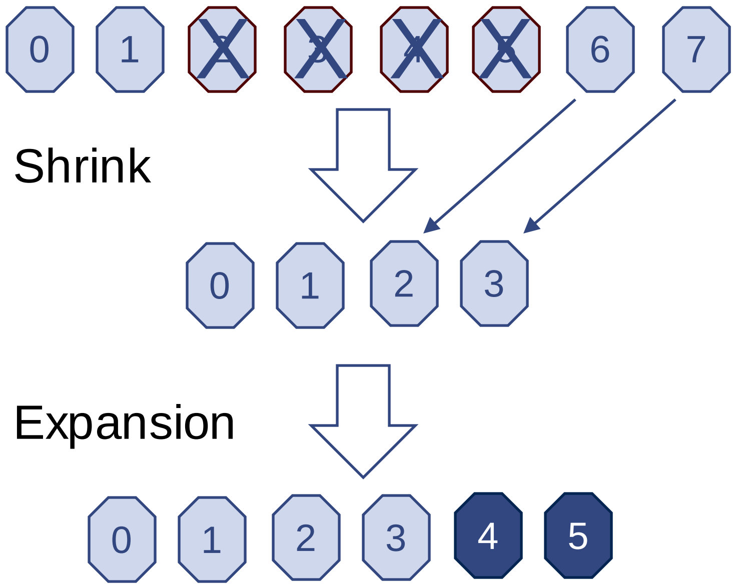
Getting ParaStation MPI ready for Malleability

- Enhancements for MPI Session implementation
  - Re-initialization of MPI library
  - Decoupling from MPI world model
  - Reference counting and checking on finalize
  - Error handling
  - PMIx Process Sets
- PMIx Spawn support

- All enhancements included in ParaStation MPI as of release 5.9.2-1
- Upstream: Many enhancements will be included in MPICH 4.2

MPI eXtensions for Malleability: Objectives

- Re-initialize MPI library
- Dense monotonic MPI rank numbering
- Allow users to
  - think in ranks and not nodes
  - select starting point
  - select ranks to exit for shrink
  - reuse utility functions
- Asynchronous resource allocation and spawning
- MPI-4 compatibility
- Rely on PMIx



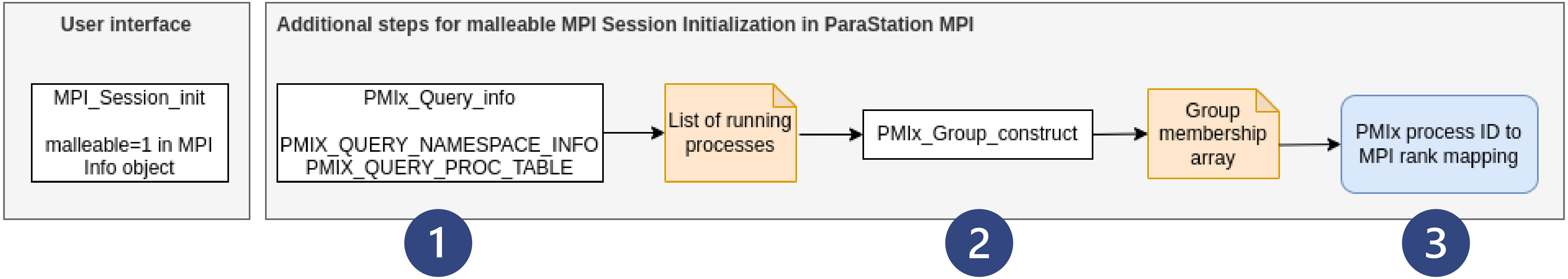
Shrink

Expansion

Proposal for 3 new MPI functions

Malleable Sessions with ParaStation MPI and PMIx

1. PMIx Query: Obtain information about all processes
2. Create PMIx Process Group for running processes
   - Process Group is destructed on `MPI_Session_finalize`
3. Obtain unambiguous mapping of PMIx process ID to MPI rank


User interface
MPI_Session_init
malleable=1 in MPI Info object
Additional steps for malleable MPI Session Initialization in ParaStation MPI
PMIx_Query_info
PMIX_QUERY_NAMESPACE_INFO
PMIX_QUERY_PROC_TABLE
List of running processes
PMIx_Group_construct
Group membership array
PMIx process ID to MPI rank mapping
1
2
3

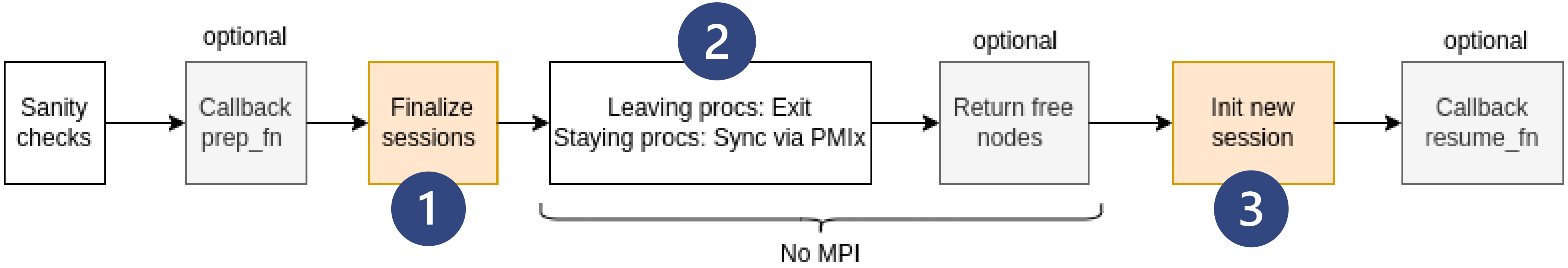
MPIX_Session_reinit

```
int MPIX_Session_reinit(int count, MPI_Session **session_array_in,
   MPI_Session *session_out, MPI_Info info,
   MPIX_Session_reinit_preparation_function *prep_fn, void *prep_userdata,
   MPIX_Session_reinit_resume_function *resume_fn, void *resume_userdata,
   int leave, MPI_Count nleave, int resize_allocation,
   MPI_Errhandler errhandler)
```

1. Finalize all existing MPI Sessions
2. Terminate processes that shall exit (shrink)
3. Initialize new MPI Session with updated rank-to-process mapping
   - Include new processes (expansion)
   - Exclude terminated processes (shrink)

Optional: Application-specific callbacks for session finalization and initialization

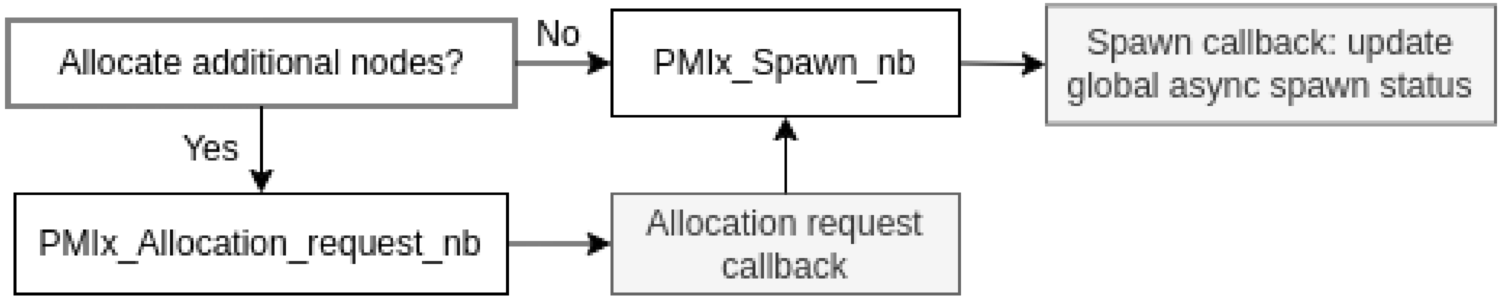
Optional: Return any free node(s)



MPIX_Spawn_async

```
int MPIX_Spawn_async(
   const char *command, char *argv[],
   int maxprocs, MPI_Info info,
   int root,int resize_allocation,
   MPI_Info *status)
```

- Trigger asynchronous spawn of new processes
- Optional: Allocate additional nodes and spawn processes on these nodes
- Most arguments like those of `MPI_Comm_spawn`
- Post-requirement: Call `MPIX_Session_reinit` to include additional processes in application progress


Allocate additional nodes?
No
PMIx_Spawn_nb
Spawn callback: update global async spawn status
Yes
PMIx_Allocation_request_nb
Allocation request callback

MPIX_Spawn_status

```
int MPIX_Spawn_status(MPI_Info *status)
```

- Obtain global status of asynchronous spawn
  - Async. spawn is available?
  - Process is spawned?
  - Status of global spawn operation?
  - Additional keys specific for ongoing or complete async spawn operation

- Future work: Relay external/ system constraints to MPI application via key value pairs, for example
  - Number of processes that can be spawned
  - Specific processes that should exit for shrink

Status and outlook

Status

- Testing with PMIx Reference Runtime Environment (PRRTE)
- Single-node tests successful
- Multi-node tests ongoing

Outlook

- Deployment on research HPC system (DEEP @JSC)
- Code review and release
- Discussion with MPI Forum

Thank you!

Dr. Sonja Happ

sonja.happ@par-tec.com

www.par-tec.com

ParTec AG
Possartstr. 20, 81679 Munich
Germany

www.deep-projects.eu

@DEEPprojects

@deep-projects

EuroHPC
Joint Undertaking

SPONSORED BY THE
Federal Ministry
of Education
and Research

fwo

Proyecto PCI2021-121958 financiado por:

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ
ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΕΠΕΝΔΥΣΕΩΝ
ΓΓΕΚ
ΓΕΝΙΚΗ ΓΡΑΜΜΑΤΕΙΑ
ΕΡΕΥΝΑΣ ΚΑΙ ΚΑΙΝΟΤΟΜΙΑΣ

MINISTERIO
DE CIENCIA
E INNOVACIÓN

Financiado por
la Unión Europea
NextGenerationEU

Plan de Recuperación,
Transformación y
Resiliencia

AGENCIA
ESTATAL DE
INVESTIGACIÓN

bpifrance

Swedish
Research
Council

The DEEP Projects have received funding from the European Commission's FP7, H2020, and EuroHPC Programmes, under Grant Agreements n° 287530, 610476, 754304, and 955606.

The EuroHPC Joint Undertaking (JU) receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Spain, Greece, Belgium, Sweden, United Kingdom, Switzerland.

Simplified example: Expansion

```c
int main(int argc, char *argv[])
{
    […] /* Init variables */
    /* Set malleable parameter in info object */
    MPI_Info_set(sinfo, "malleable", "1");
    /* Init session */
    MPI_Session_init(sinfo, MPI_ERRORS_ARE_FATAL, &session);
    […] /* Create MPI Objects and do work...*/
    if (!spawned) {
       /* Trigger the spawn */
       MPIX_Spawn_async((char *) "./my_app", MPI_ARGV_NULL,
                        /* spawn 2 processes */ 2, MPI_INFO_NULL,
                        /* rank 0 is the root */ 0, 0, &spawn_status);
       /* Wait for spawned processes to become ready */
       MPI_Info_get(spawn_status, "spawn_x_status", MPI_MAX_INFO_VAL, status, &found);
       while (strncmp(status, "complete", MPI_MAX_INFO_VAL) != 0) {
              MPIX_Spawn_status(&spawn_status);
              MPI_Info_get(spawn_status, "spawn_x_status", MPI_MAX_INFO_VAL, status, &found);
       }
       […] /* Clean-up old MPI objects */
       /* Re-init */
       sessions_for_reinit[0] = &session;
       MPIX_Session_reinit(1, sessions_for_reinit, &session_NEW, MPI_INFO_NULL,
                           MPIX_SESSION_REINIT_PREP_FN_NULL, NULL,
                           MPIX_SESSION_REINIT_RESUME_FN_NULL, NULL, 0, 0, 0, MPI_ERRORS_ARE_FATAL);
       […] /* Create new MPI objects and continue with work */
       MPI_Session_finalize(&session_NEW);
    } else {
       /* Spawned process, finalize session */
       MPI_Session_finalize(&session);
    }
}
```

Trigger async spawn

Know that new processes are available

Re-init complete, new processes are included