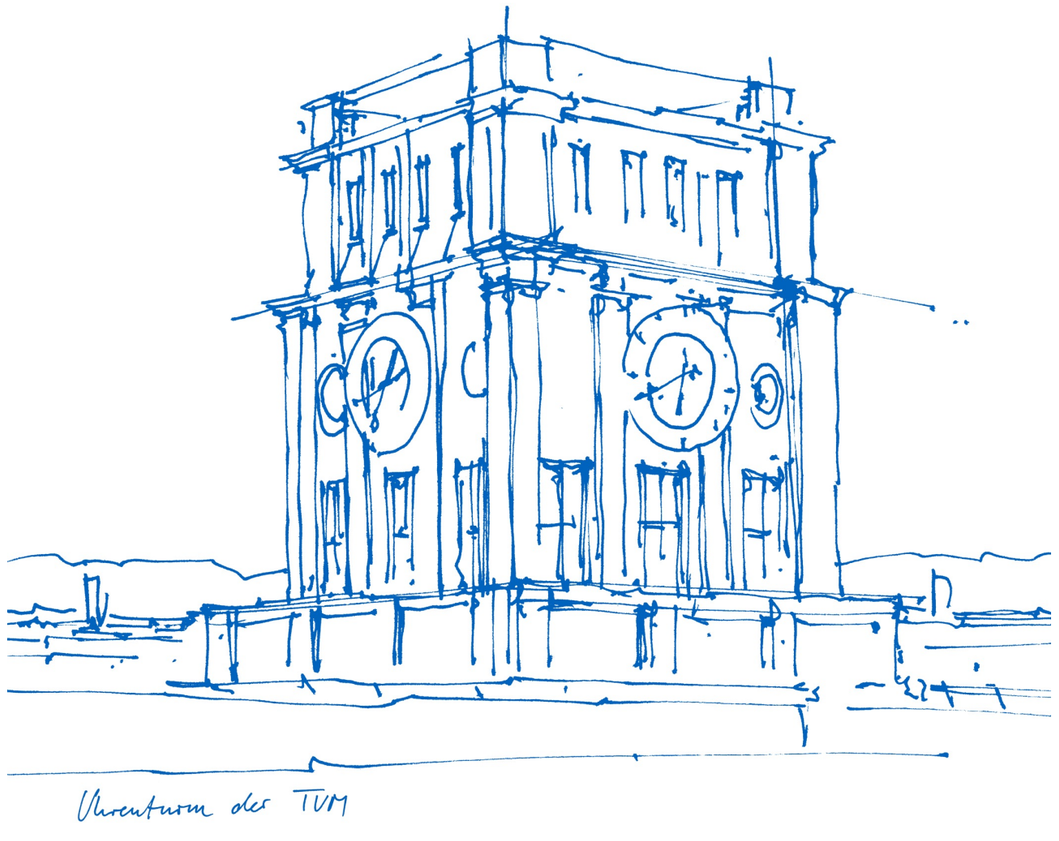
Adding Malleability to MPI: Opportunities and Challenges

Martin Schulz

Chair for Computer Architecture and Parallel Systems
Technical University of Munich (TUM)

ESPM2: 2022 ACM/IEEE 7th International Workshop on
Extreme Scale Programming Models and Middleware

Monday Nov. 14th, 2022 @ SC22



Systems must/will become more adaptive

New hardware architectures	require us to mix CPU with different accelerators
Power limits	require us to active manage power using power shifting
Costs for data movements	require short distances between components (i.e., on chip)
Varying workloads/flows	require us to integrate elements closely

Systems must/will become more adaptive

New hardware architectures	require us to mix CPU with different accelerators
Power limits	require us to active manage power using power shifting
Costs for data movements	require short distances between components (i.e., on chip)
Varying workloads/flows	require us to integrate elements closely

Result: Integrated Node Architectures

Adaptivity is key in the design of efficient integrated systems

… needed for efficient resource utilization in ”mixed compute” systems
… needed to counteract variability due to manufacturing differences
… needed to manage complex workflows in modern applications
… needed by new workloads especially in ML/DL/AI

Must be implemented across the entire system stack:
adaptive runtime resource management
in the hardware, middleware and the application, …

… and in the runtime & parallel programming models

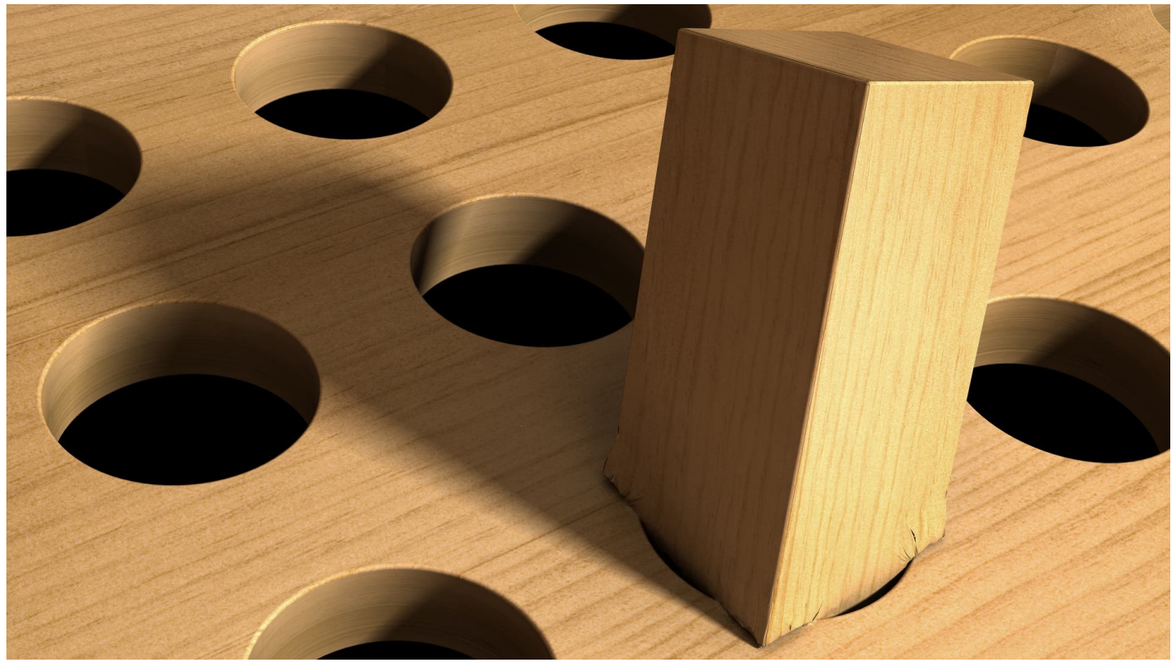
Enable applications to

- query changing resource usage
- grow and shrink resources
- Enable clean transition between resources

Across any kind of resources

- Nodes/Processes
- Hardware components
- Power, memory, …

- Driven by the runtime system and scheduler
- Driven by the application itself



Approach does not match current practice, which assumes:

- Simple, “homogeneous compute” nodes without any sharing of resources
- Fixed resource budgets across the life time of an application
- Worst case provisioning for many resources, including power

Application developers make assumptions based on this current practice!

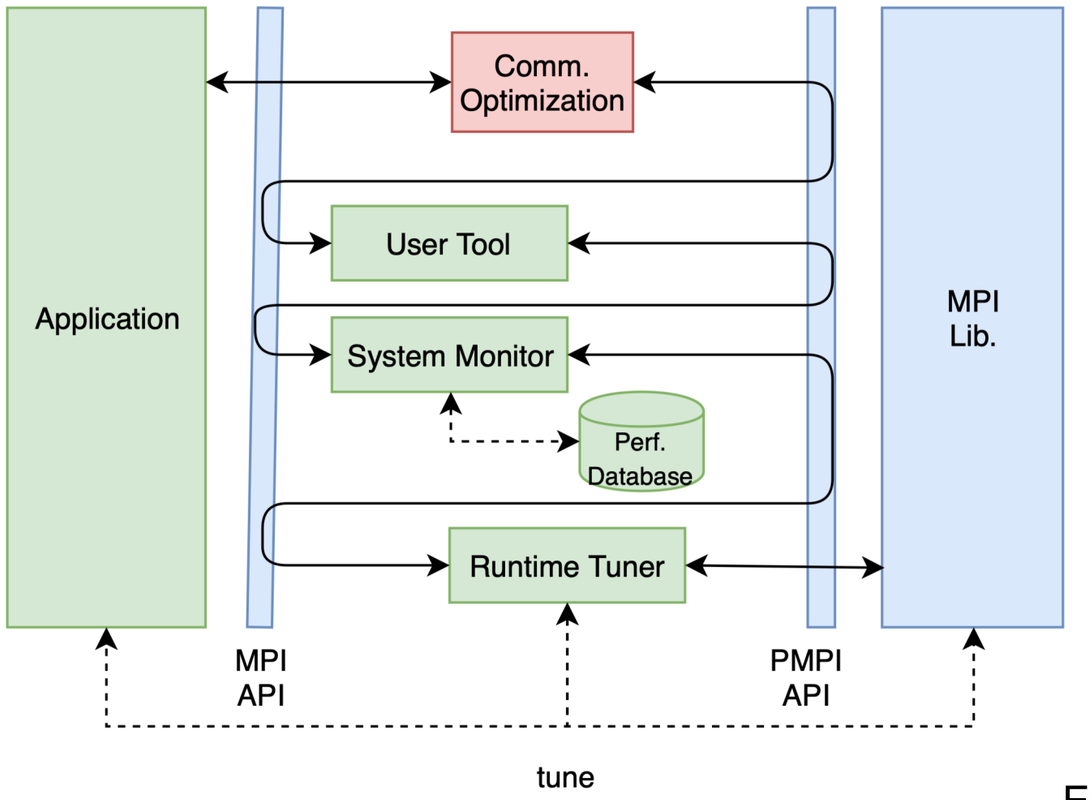
Breaking HPC Dogmas

One Node = On Job	On-Node Co-Scheduling	Effective use of complementary accelerators
Jobs have static resource usage from start to end	Dynamic Job Allocations	Adapt to external conditions (e.g., contention)
Worst case and fixed power distribution	The HPC PowerStack	Adaptive power steering on over-provisioned nodes

Providing a Dynamic Environment for Future Systems

Holistic Monitoring

- Must include whole system stack
- Developments like QMPI are crucial (MPI 5.0?)



Effective use of complementary accelerators

Adapt to external conditions (e.g., contention)

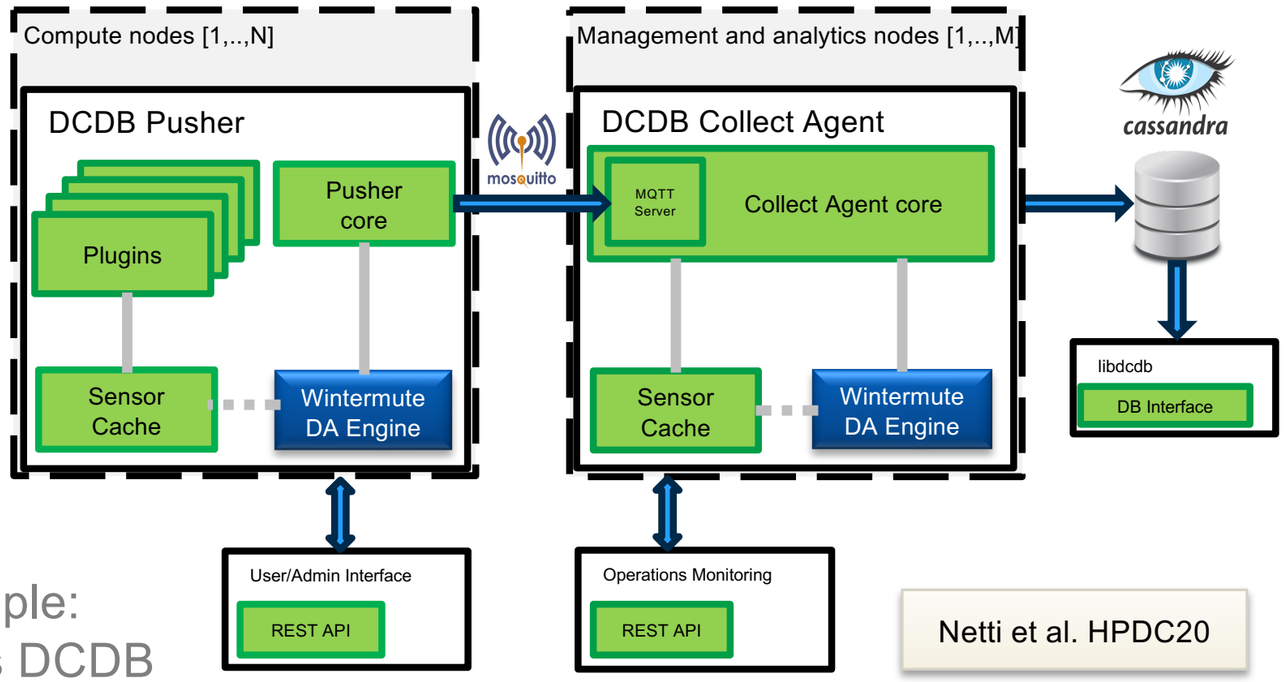
Adaptive power steering on over-provisioned nodes

EuroMPI19, Elis et al.

Providing a Dynamic Environment for Future Systems

Holistic Monitoring

- Must include whole system stack
- Developments like QMPI are crucial (MPI 5.0?)
- Coupled with in-situ operational data analytics



Netti et al. HPDC20

Example:
LRZ's DCDB

Effective use of complementary accelerators

Adapt to external conditions (e.g., contention)

Adaptive power steering on over-provisioned nodes

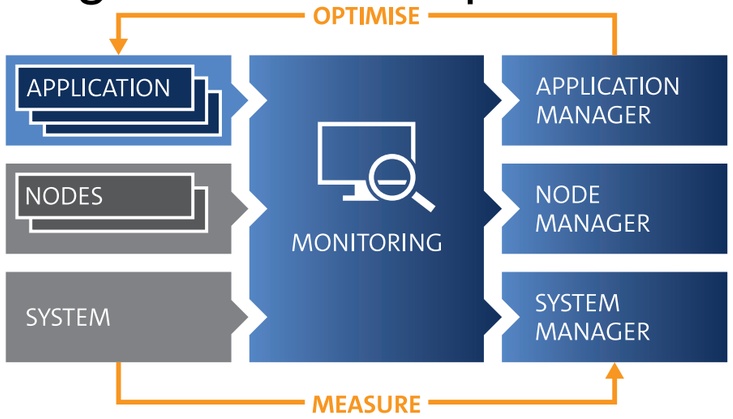
Providing a Dynamic Environment for Future Systems

Holistic Monitoring

- Must include whole system stack
- Developments like QMPI are crucial (MPI 5.0?)
- Coupled with in-situ operational data analytics

Adaptive System Management

- Closed system loop for optimization
- Extensions to resource managers
- Active power shifting between components and nodes



Effective use of complementary accelerators

Adapt to external conditions (e.g., contention)

Adaptive power steering on over-provisioned nodes

EU Grant #956560
BMBF #16HPC039K
REGALE

Providing a Dynamic Environment for Future Systems

Holistic Monitoring

- Must include whole system stack
- Developments like QMPI are crucial (MPI 5.0?)
- Coupled with in-situ operational data analytics

Adaptive System Management

- Closed system loop for optimization
- Extensions to resource managers
- Active power shifting between components and nodes

Adaptive Software Stack

- Consequences: more adaptivity across the system!
- Need evolution in standard models, like OpenMP & MPI
- Domain specific libraries and frameworks
- Should be unified with fault tolerance!

Effective use of complementary accelerators

Adapt to external conditions (e.g., contention)

Adaptive power steering on over-provisioned nodes

Towards a Dynamic Programming Environments

EuroHPC project DEEP-SEA

- Goal: comprehensive EU SW Stack for Exascale
- Focus on heterogeneous machines
  - On-node and Cross-node
- From driver level to programming abstractions
- Central theme: Adding malleability at all levels of the entire software stack

EU Grant #955606
BMBF #16HPC014
DEEP-SEA

SPONSORED BY THE
Federal Ministry
of Education
and Research

Impact on MPI

- Currently, support for moldability, but not malleability
- Need to go beyond ”Spawn” abstraction

- Need to break MPI_COMM_WORLD abstraction
- Add/remove MPI processes on the fly
- Enable interaction with runtime data

BUT: maintain MPI “look & feel”

Part of a Solution: MPI Sessions

Instead of MPI_Init / MPI_COMM_WORLD:

1. Get local access to the MPI library

   Get a Session Handle

2. Query the underlying run-time system

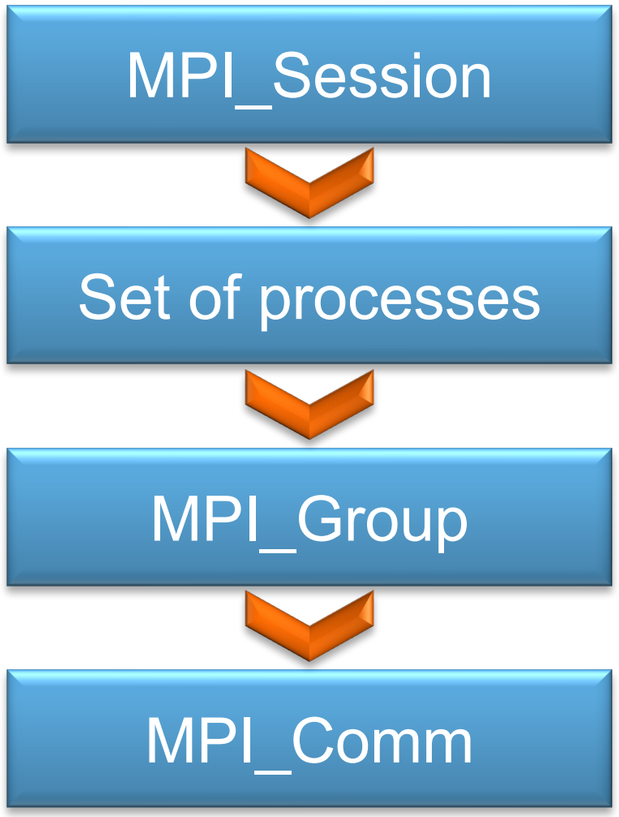
   Get a "set" of processes

3. Determine the processes you want

   Create an MPI_Group

4. Create a communicator with just those processes

   Create an MPI_Comm



MPI Sessions

What does this do?

- Deliver runtime information to the MPI library
- Enable resource isolation between sessions
- Eliminate the static resource MPI_COMM_WORLD

Where do Process Sets Come From?

- Two predefined sets: **`mpi://WORLD`** and **`mpi://SELF`**
- Runtimes can provide system configurations, like **`location://rack/17`**
- Users can specify process sets, like **`app://ocean`**

Intended Usage Patterns

- Scalable initialization on subsets of processes
- Separate library initialization
- Separation of application components (ice, ocean, atmosphere, …)

Building on top of MPI Sessions

Option 1: MPI Session form ”MPI Bubbles”

- “All resources that are derived from a set of resources across a set of MPI processes”
- Implicitly derived from MPI application using sessions
- Within an MPI bubble, normal MPI
- Can invalidate and recreate new bubbles
- New sessions can have new process sets
- Need some kind of versioning
- ➢ **Medium granularity**

Option 2: Process sets can change

- Names are local to MPI Sessions
- Enable process sets to grow or shrink
- Ability for the runtime to “tell” something to the application
- New routines to manipulate process sets
- Agreement protocol/versioning to agree on new set
- ➢ **Fine granularity**

Open Issues

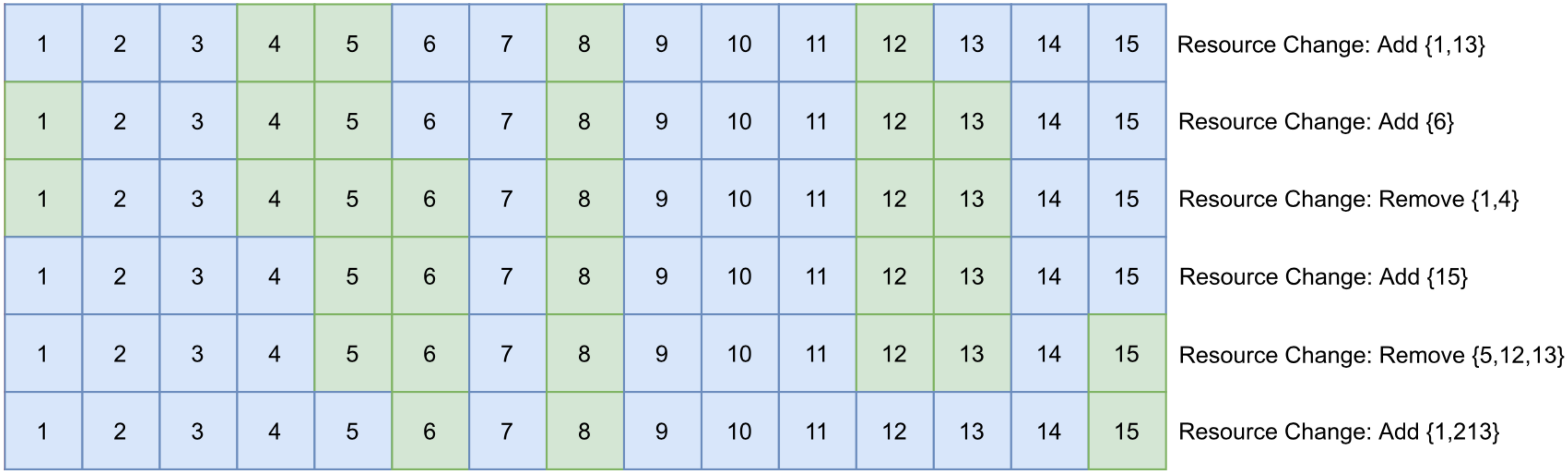
Resource change detection APIs
- Do we want/need notification vs. Polling?
- How to version process sets?
- How to avoid full comparisons of all process sets?

Resource description APIs
- Which resources should be requested?
- Which resources are available?
- Should this be part of MPI or external?

Handshake negotiation APIs
- How to request a change?
- How to inform applications of a change?
- How to capture agreement?
- Central manager or collective?

Connection to fault tolerance proposals ?!?!

Impact of Malleability



Resources can come and go

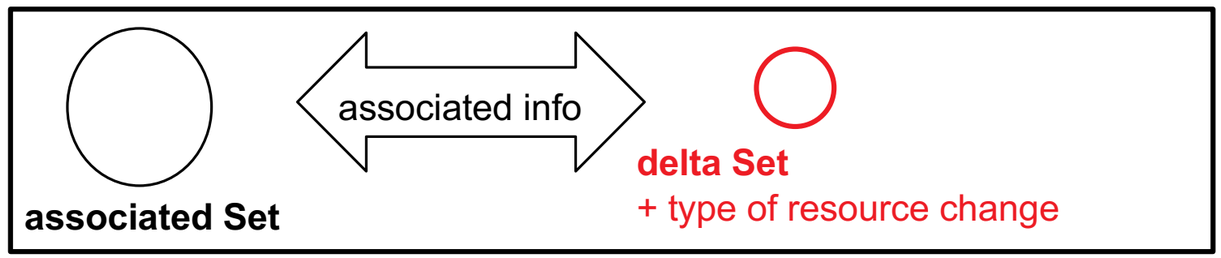
Final set can be completely different than starting

From: Jan Fecht, TUM – published at HPCMALL 2022

Example API (Fine Grained / Single Manager)

Step 1: Query resource

Get information on what has changed or request a change




```
int MPIX_Session_Dyn_Recv_res_change(

        session,                IN:     Session handle
        associated_pset,        IN:     Name of the associated PSet
        res_change_type,        OUT:    Type of the resource change (MPI_RC_NONE/ADD/SUB)
        delta_pset,             OUT:    Name of delta PSet, describing the changing resources
        included_flag           OUT:    Flag indicating if this process is included in the delta PSet
);
```


From: Dominik Huber, TUM – published at EuroMPI 2022

Example API (Fine Grained / Single Manager)

Step 2: Define exact resource request

Manipulate the process set using new set of APIs

$$\bigcirc \circ \bigcirc = \bigcirc \, , \circ \in \{\cup, \cap, \setminus, \ldots\}$$

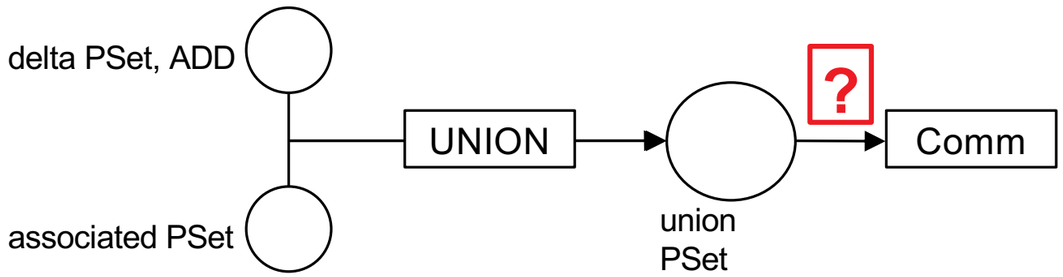
Example: $S_{assoc} \cup S_{delta} = S_{union}$

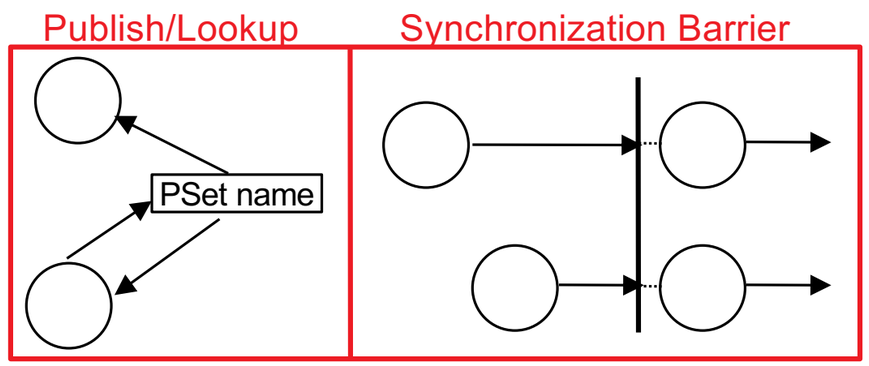
```
int MPIX_Session_pset_create_op (
        session,            IN:     Session handle
        pset_operation,     IN:     Requested set operation: UNION, DIFF, INTERSECT
        pset1,              IN:     First argument to the set operation (PSet name)
        pset2,              IN:     Second argument to the set operation (PSet name)
        pset_result         OUT:    Result of the set operation (PSet name)
);
```


From: Dominik Huber, TUM – published at EuroMPI 2022

Example API (Fine Grained / Single Manager)

Step 3: Finalize Change






```
/* Collective over the UNION of the associated Pset(s) and the delta Pset */
MPIX_Session_Dyn_integrate_res_change (
        session          IN:     Session handle
        info,            IN:     MPI Info object (optional)
        delta_pset,      IN:     Name of the delta PSet of the resource change
        provider_flag,   IN:     Flag indicating if this process provides the name of a PSet
        pset_name_buf,   IN/OUT: Buffer for a PSet name (can be NULL if provider_flag == false)
        terminate        OUT:    Flag indicating if this process needs to terminate
)
```


From: Dominik Huber, TUM – published at EuroMPI 2022

Future System will (have to be) more malleable!

New system architectures

- Dynamic environments and workloads
- Constrained resources like power
- New and diverse workloads

Direct impact on programming models and this includes in particular MPI

- Away from static resources
- Interactions with runtime systems
- Missing: negotiation APIs

MPI Sessions can be a/the basis

- Process sets to interact with runtime
- Creation of communicators from scratch
- Granularity?
- Central vs. Collective?

MPI www.mpi-forum.org

DEEP-SEA
EU Grant #955606
BMBF #16HPC014

TIME-X
EU Grant #955701
BMBF #16HPC050

REGALE
EU Grant #956560
BMBF #16HPC039K

SPONSORED BY THE
Federal Ministry of Education and Research