Taming the Beast of Dynamic Resource Management in HPC

Dominik Huber[1]

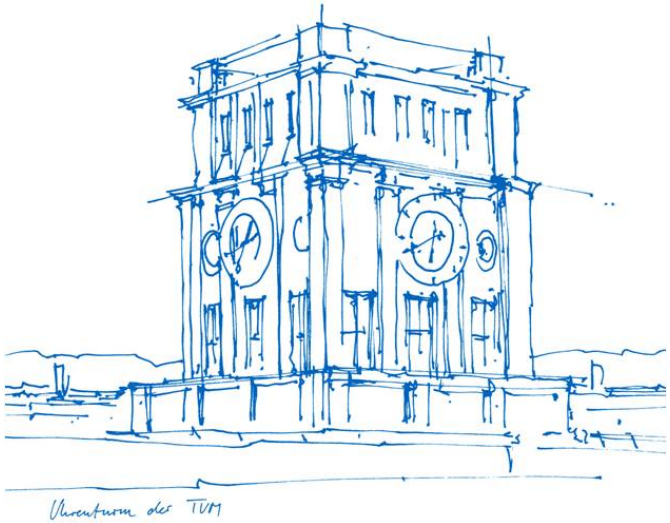
Supervisor: Martin Schreiber[2]

[1]Technical University of Munich

[2]Université Grenoble Alpes

Doctoral Showcase Supercomputing Conference 2025

St. Louis, MO, USA, 20.11.2025



OUTLINE

The Beast of Dynamic Resources in HPC

Dynamic HPC Software Stack
- Dynamic Processes with PSets
- Dynamic Applications and Libraries
- Dynamic Scheduling Optimization

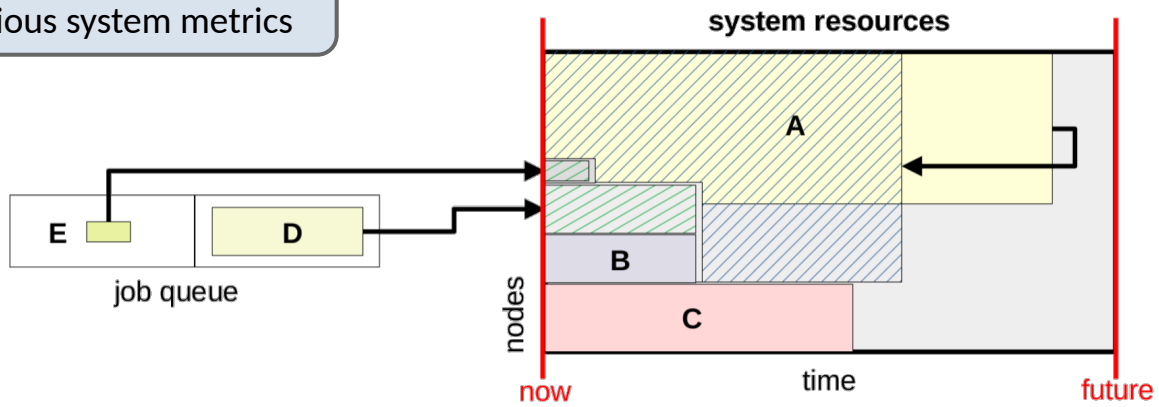
Performance Evaluation

The Beauty: Benefits of Dynamic Resources in the HPC

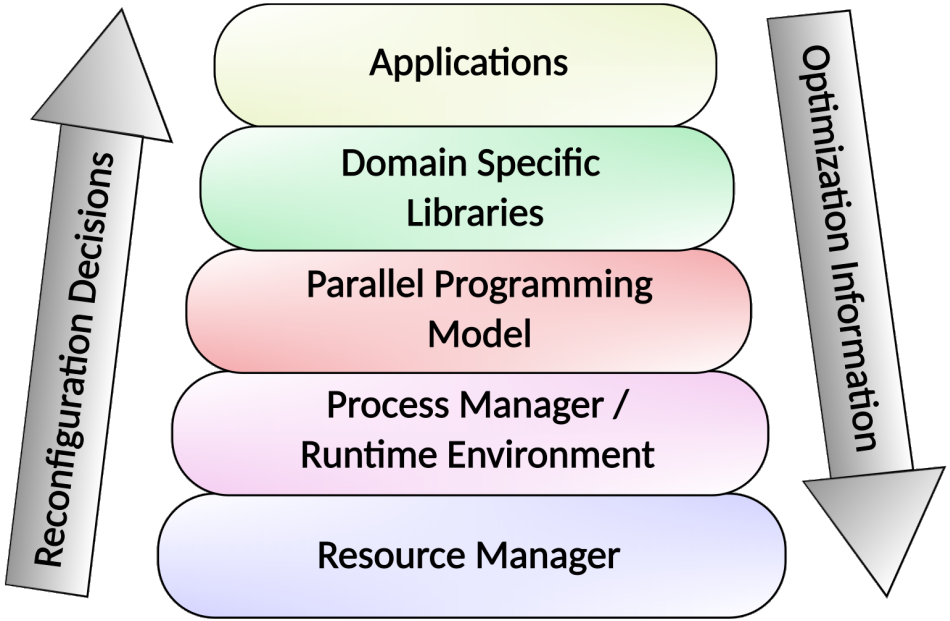
Dynamically adjusting resource allocations during job runtime!

More flexibility for resource management software to optimize various system metrics

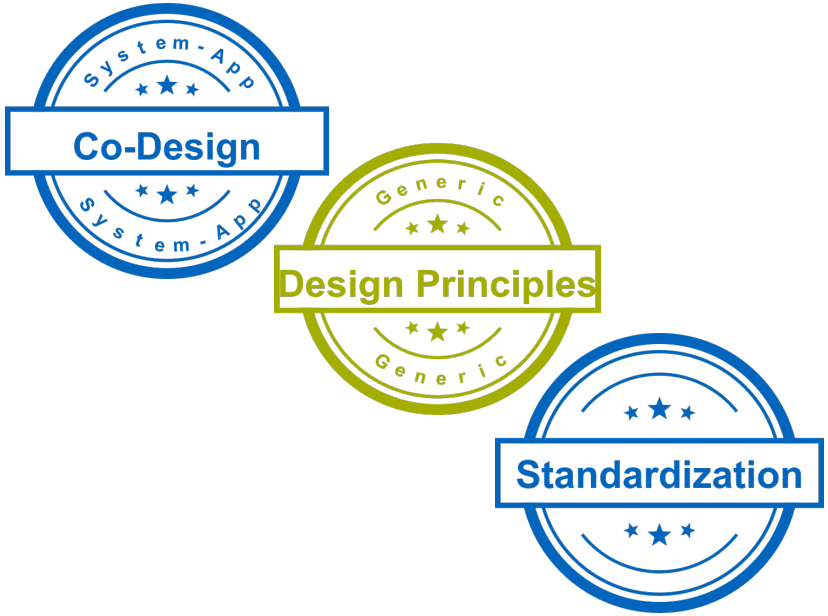
Better support for applications with dynamically varying resource requirements



The Beast: Dynamic Resources in the HPC Software Stack



Holistic approach required!



OUTLINE

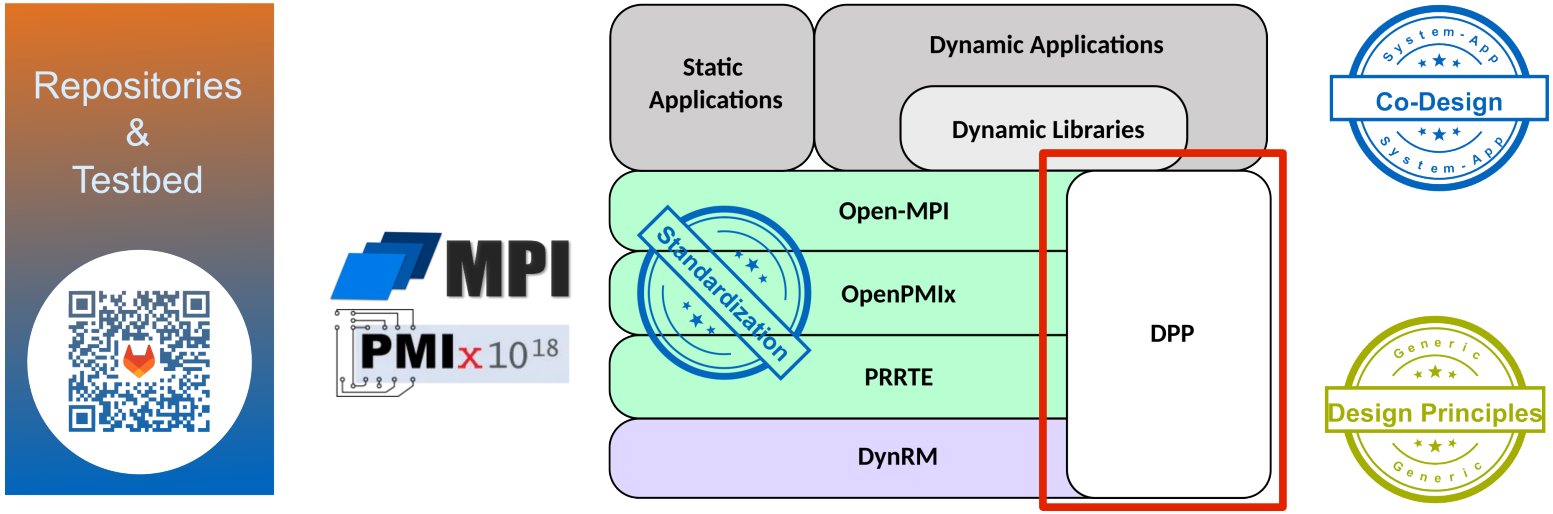
The Beast of Dynamic Resources in HPC

Dynamic HPC Software Stack

- **Dynamic Processes with PSets**
- Dynamic Applications and Libraries
- Dynamic Scheduling Optimization

Performance Evaluation

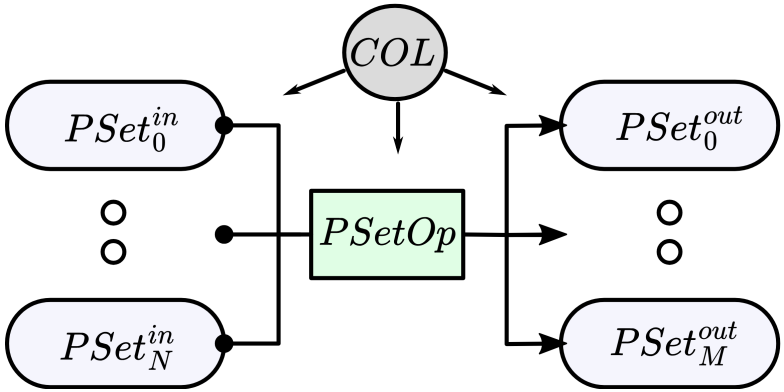
The Dynamic HPC Software Stack



Dynamic Processes with Process Sets (DPP)

1.Separation of concerns:
- Dynamic Process Management (DPM)
- Dynamic Resource Allocation (DRA)

2. DPM based on PSets and PSetOps

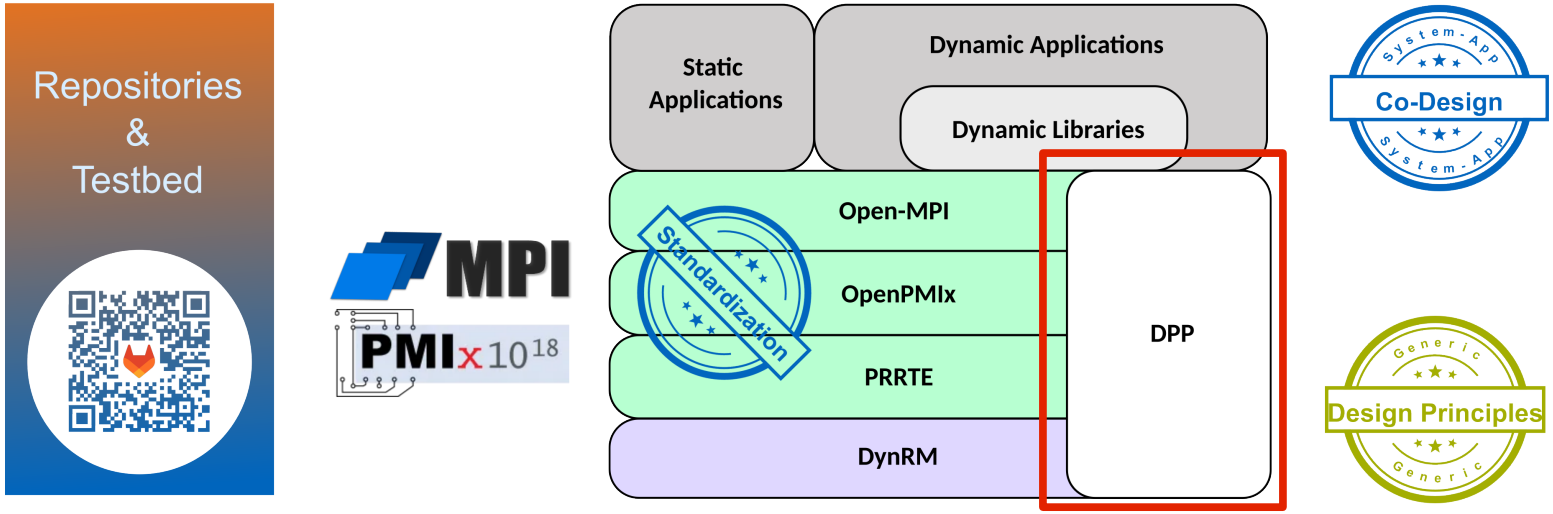


3. DRA based on Collaborative Optimization Language (COL)

Full paper

Design Principles of Dynamic Resource Management for High-Performance Parallel Programming Models

The Dynamic HPC Software Stack



OUTLINE

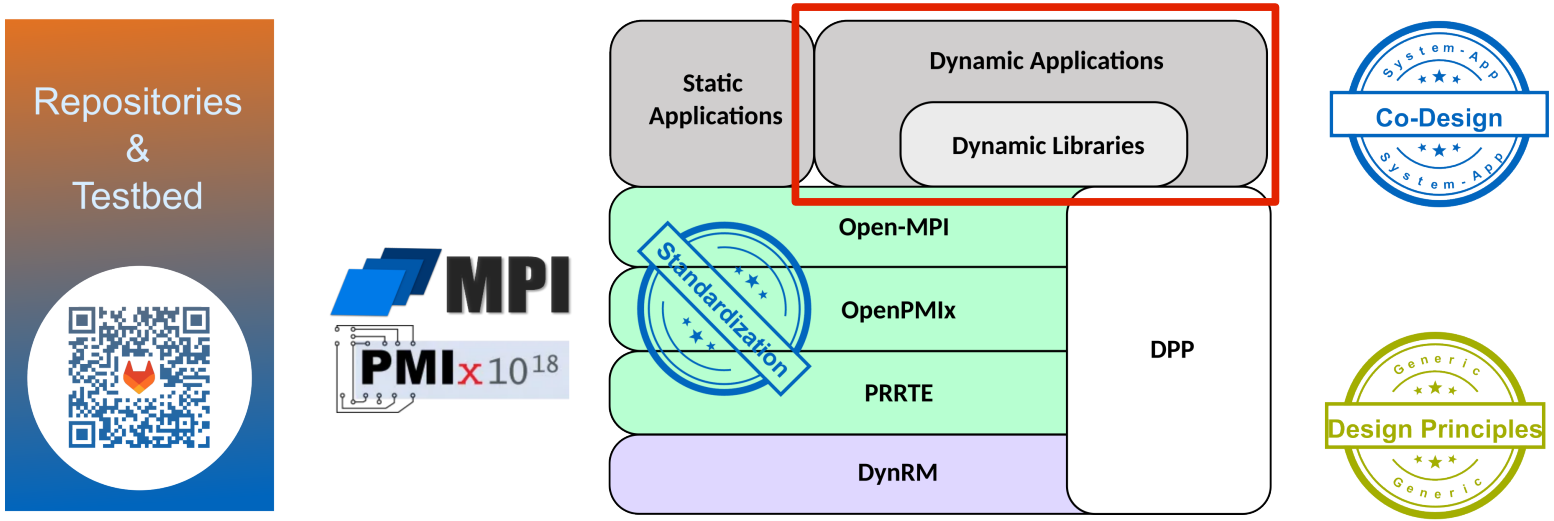
The Beast of Dynamic Resources in HPC

Dynamic HPC Software Stack

- Dynamic Processes with PSets
- **Dynamic Applications and Libraries**
- Dynamic Scheduling Optimization

Performance Evaluation

The Dynamic HPC Software Stack

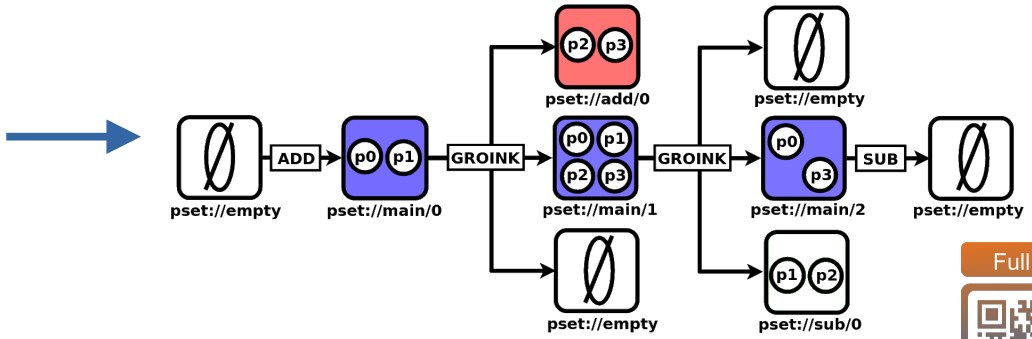


Dynamic Applications and Libraries

PetSc [1] Parallel numerical software library	**DynLAIK [13]** Data distribution for dynamicity and fault tolerance	**LibPFASST [3]** Parallel-in-time integration library
P4est [2] Adaptive Mesh Refinement library	**XBraid [4]** Multigrid parallel-in-time integration	**MPDATA3D [12]** 3D semi-Lagrangian multiscale fluid solver
DMR-API [6] Dynamic resources API for computational kernels	**AMT-GLB [11]** Asynchronous Many-Task Runtime System	**DPPInSitu [10]** Library for dynamic, asynchronous in-situ techniques

```
void main() {
   DMR_INIT(user_init_data(), recv_expand());
   DMR_Set_parameter(MIN, MAX, SWEET_SPOT);
   DMR_Inhibit_iter(n_iters);
   for(it = 0; it < ITERATIONS; it++) {
       DMR_RECONFIGURATION( send_expand(),
                            recv_expand(),
                            send_shrink(),
                            recv_shrink());

       compute();
   }
   DMR_FINALIZE(user_free_data());
}
```



Full paper

Bridging the Gap Between Genericity and Programmability of Dynamic Resources in HPC

OUTLINE

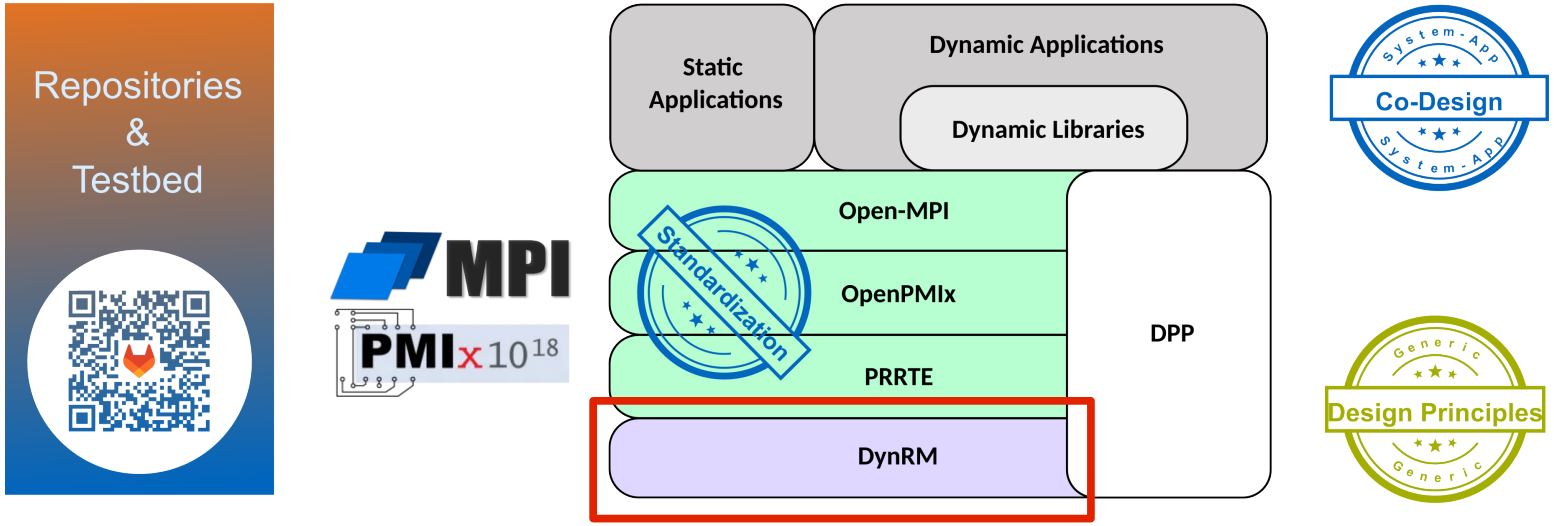
The Beast of Dynamic Resources in HPC

Dynamic HPC Software Stack

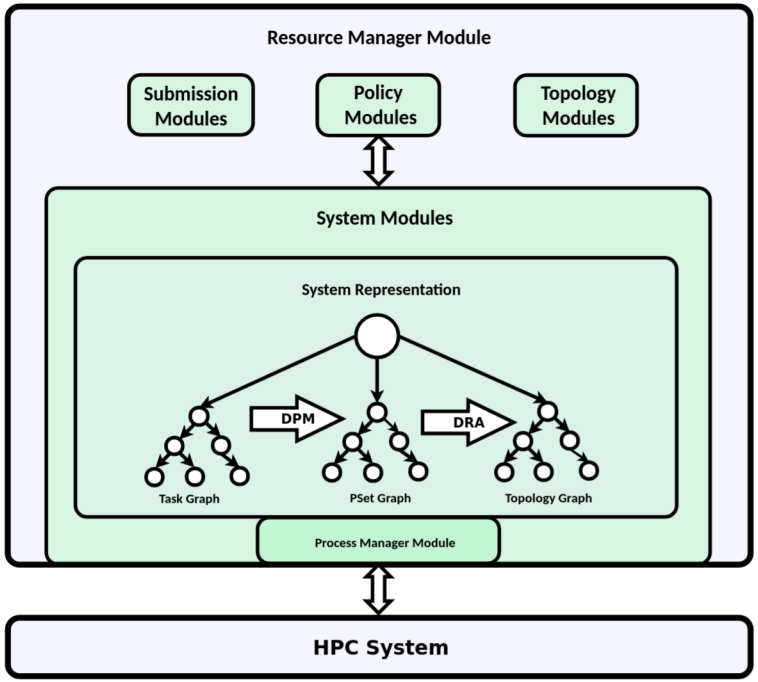
- Dynamic Processes with PSets
- Dynamic Applications and Libraries
- **Dynamic Scheduling Optimization**

Performance Evaluation

The Dynamic HPC Software Stack



DynRM: A modular, dynamic resource manager



Given: System Graphs

- G_{topo}: **System topology graph** describing the system resources
- G_{task}: **Task Graph** describing jobs and their dependencies
- G_{pset}: **PSet Graph** describing resource access of tasks via processes

Optimization Problem

Find *graph transformations* for G_{pset} that optimize the execution of G_{task} on G_{topo} according to some given optimization objective.

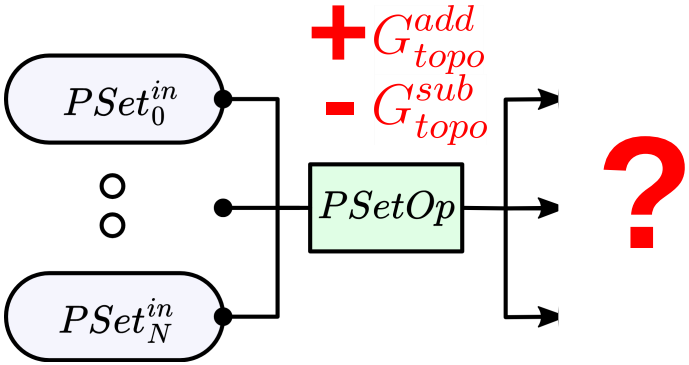
Graph-based, performance-aware dynamic scheduling

Edge Function: Contraint Function

Creates a finite set of possible instantiations of a PSetOp (*output space*).

$$C^{e_i}_{valid}(V_{in}, G^{sub}_{topo}, G^{add}_{topo}) \rightarrow \{V^1_{out}, \cdots, V^n_{out}\} =: \mathcal{V}_{out}$$

V_{in}	=	Input vertices
$G^{sub/add}_{topo}$	=	Topo vertices to be removed/added
V^i_{out}	=	i-th possible instantion of output vertices



Graph-based, performance-aware dynamic scheduling

Edge Function: Contraint Function

Creates a finite set of possible instantiations of a PSetOp (*output space*).

$$C^{e_i}_{valid}(V_{in}, G^{sub}_{topo}, G^{add}_{topo}) \rightarrow \{V^1_{out}, \cdots, V^n_{out}\} =: \mathcal{V}_{out}$$

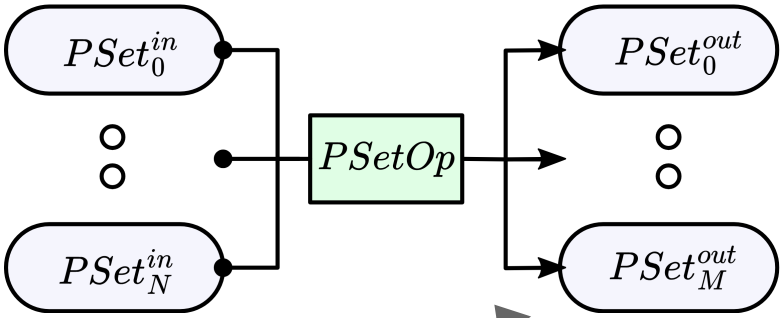
V_{in}	=	Input vertices
$G^{sub/add}_{topo}$	=	Topo vertices to be removed/added
V^i_{out}	=	i-th possible instantion of output vertices

Vertex Function PSet Model

Describes the performance of a (mapped) PSet.

$$M^{v_i}_{pset}(v) = Y, \quad Y \in R^n, \quad e.g., \quad S(n) = \frac{t_p + t_s}{t_s + \frac{t_p}{P}}$$

→ Pset Model Parameters can be inferred from monitoring data



Graph-based, performance-aware dynamic scheduling

Edge Function: Contraint Function

Creates a finite set of possible instantiations of a PSetOp (*output space*).

$$C^{e_i}_{valid}(V_{in}, G^{sub}_{topo}, G^{add}_{topo}) \rightarrow \{V^1_{out}, \cdots, V^n_{out}\} =: \mathcal{V}_{out}$$

V_{in}	=	Input vertices
$G^{sub/add}_{topo}$	=	Topo vertices to be removed/added
V^i_{out}	=	i-th possible instantion of output vertices

Vertex Function PSet Model

Describes the performance of a (mapped) PSet.

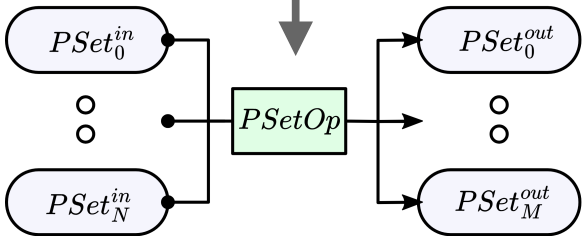
$$M^{v_i}_{pset}(v) = Y, \quad Y \in R^n, \quad e.g., \quad S(n) = \frac{t_p + t_s}{t_s + \frac{t_p}{P}}$$

→ Pset Model Parameters can be inferred from monitoring data

Edge Function: PSetOp Model

Describes performance change when applying PSetOp e_i.

$$M^{e_i}_{psetop}(e_i) = M^{e_i}_{psetop}(V_{in}, V_{out}) = M^{e_i}_{psetop}\begin{pmatrix}\begin{bmatrix} M^{v^{in}_1}_{pset}(v^{in}_1) \\ \vdots \\ M^{v^{in}_k}_{pset}(v^{in}_k) \\ M^{v^{out}_1}_{pset}(v^{out}_1) \\ \vdots \\ M^{v^{out}_l}_{pset}(v^{out}_l) \end{bmatrix}\end{pmatrix} \rightarrow Y, Y \in R^n$$



Graph-based, performance-aware dynamic scheduling

Edge Function: Contraint Function

Creates a finite set of possible instantiations of a PSetOp (*output space*).

$$C^{e_i}_{valid}(V_{in}, G^{sub}_{topo}, G^{add}_{topo}) \rightarrow \{V^1_{out}, \cdots, V^n_{out}\} =: \mathcal{V}_{out}$$

V_{in}	=	Input vertices
$G^{sub/add}_{topo}$	=	Topo vertices to be removed/added
V^i_{out}	=	i-th possible instantion of output vertices

Vertex Function PSet Model

Describes the performance of a (mapped) PSet.

$$M^{v_i}_{pset}(v) = Y, \quad Y \in R^n, \quad e.g., \quad S(n) = \frac{t_p + t_s}{t_s + \frac{t_p}{P}}$$

→ Pset Model Parameters can be inferred from monitoring data

Edge Function: PSetOp Model

Describes performance change when applying PSetOp e_i.

$$M^{e_i}_{psetop}(e_i) = M^{e_i}_{psetop}(V_{in}, V_{out}) = M^{e_i}_{psetop} \left(\begin{bmatrix} M^{v^{in}_1}_{pset}(v^{in}_1) \\ \vdots \\ M^{v^{in}_k}_{pset}(v^{in}_k) \\ M^{v^{out}_1}_{pset}(v^{out}_1) \\ \vdots \\ M^{v^{out}_l}_{pset}(v^{out}_l) \end{bmatrix} \right) \rightarrow Y, Y \in R^n$$

Objective Function: Local

Objective Function to choose a PSetOp instantiation.

$$\max_{V^*_{out}} f_{local}(x) = N_{local}(w^*_{local} \times M_{psetop}(V_{in}, V^*_{out})).$$

N_{local}	=	Local scalarization
w^*_{local}	=	Local metric wheights

Graph-based, performance-aware dynamic scheduling

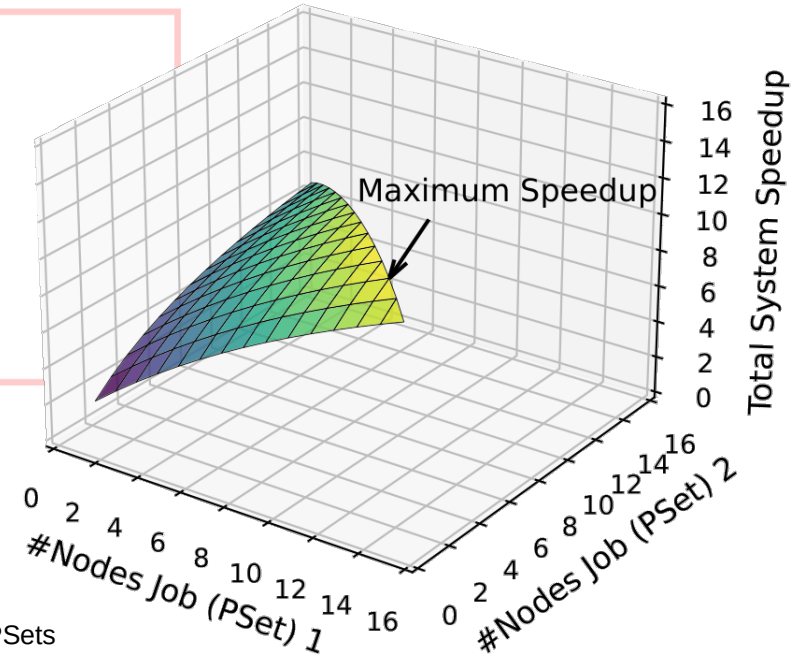
Objective Function: Global

Global Objective Function for dynamic scheduler.

$$\sum_i w_i \cdot N_{global}(M^i_{psetop}(V^i_{in}, V^i_{out})).$$

N_{global} = Global scalarization
w_i = Global PSetOp wheights

Full paper



Dynamic Resource Management in HPC systems using Dynamic Processes with PSets

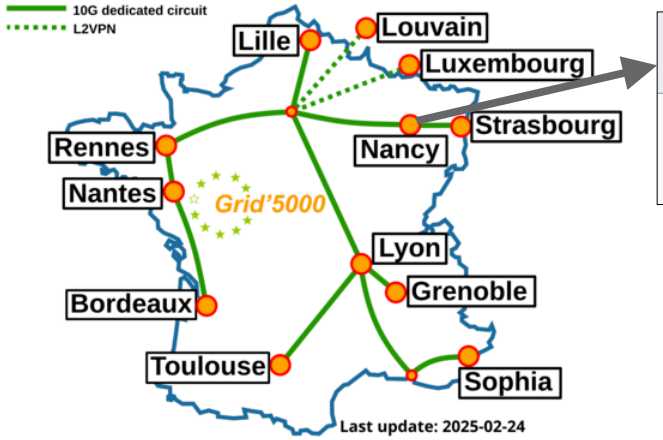
OUTLINE

The Beast of Dynamic Resources in HPC

Dynamic HPC Software Stack
- Dynamic Processes with PSets
- Dynamic Applications and Libraries
- Dynamic Scheduling Optimization

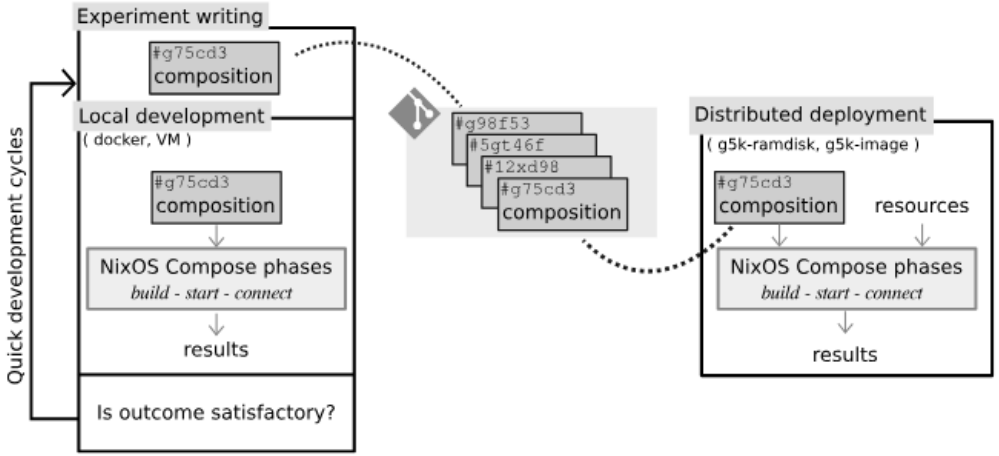
Performance Evaluation

Performance Evalution: Test System

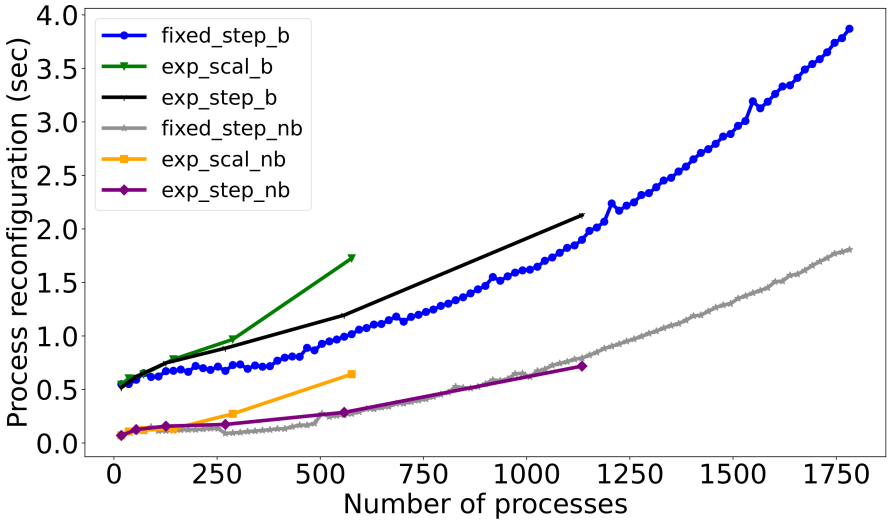


Nodes	CPU				Memory	Storage	Network
	#	Name	Cores	Architecture			
124	1	Intel Xeon Gold	18 cores/ CPU	x86_64	96 GiB	**480 GB SSD** + 960 GB SSD*	2 x 25 Gbps (SR-IOV)

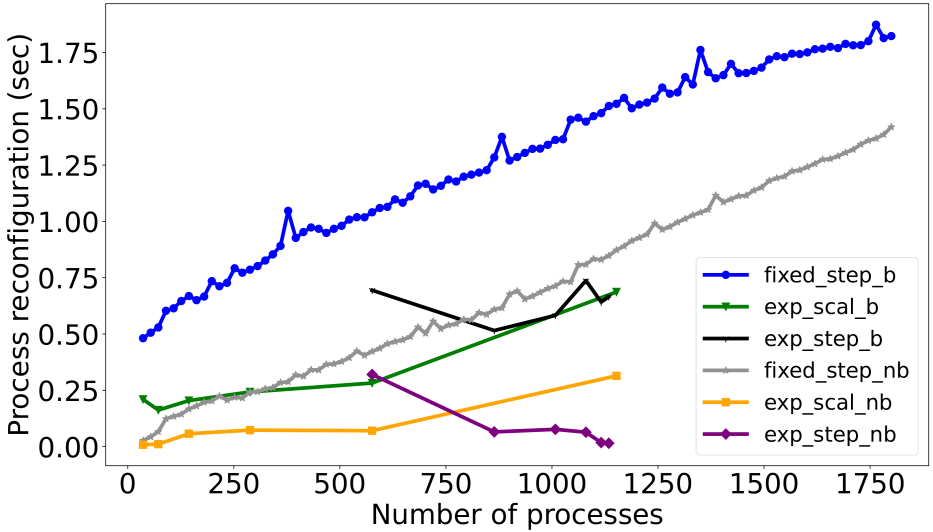
Bare-metal deployment of the *Dynamic Software Stack* using **NixOS-Compose**



Performance Evalution: Reconfiguration Overhead

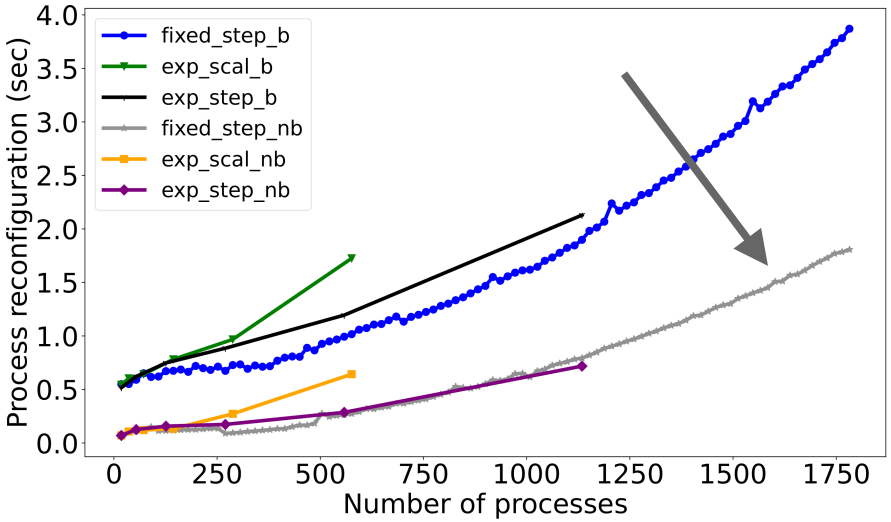


a) Expansion

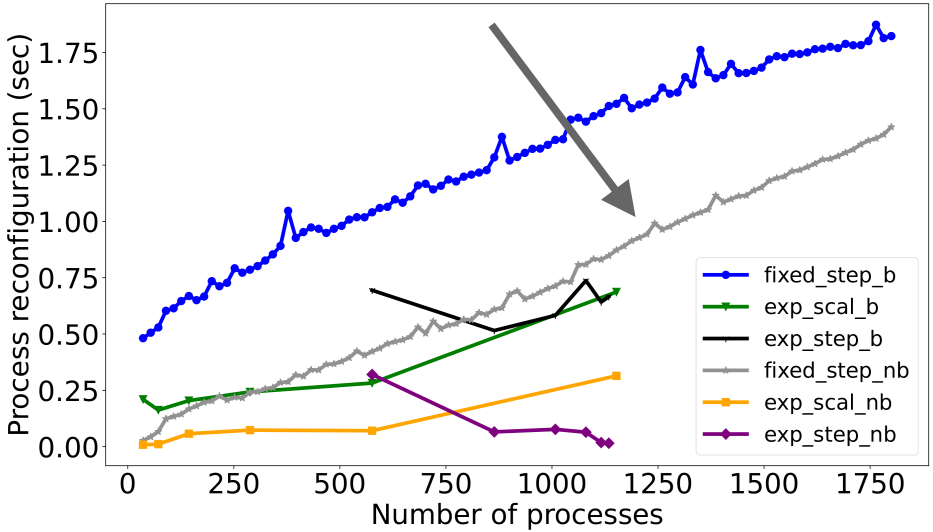


b) Shrinking

Performance Evalution: Reconfiguration Overhead

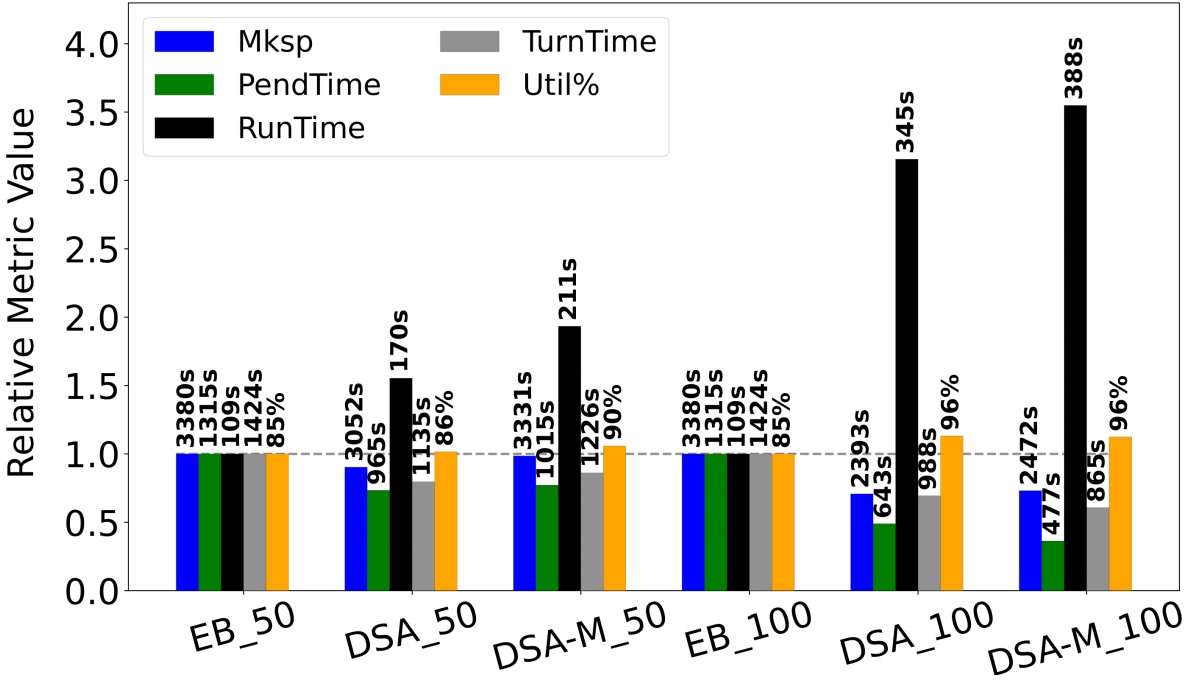


a) Expansion

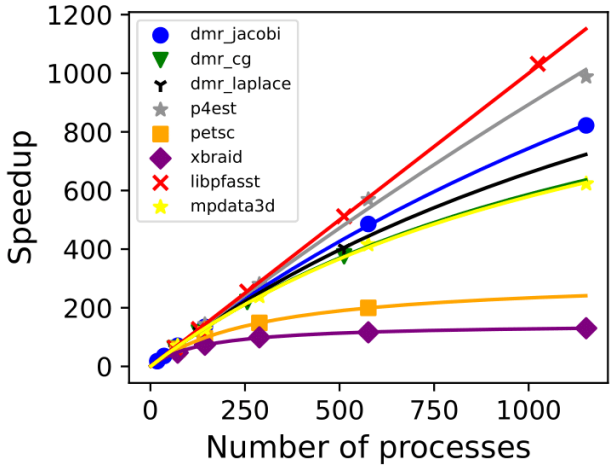


b) Shrinking

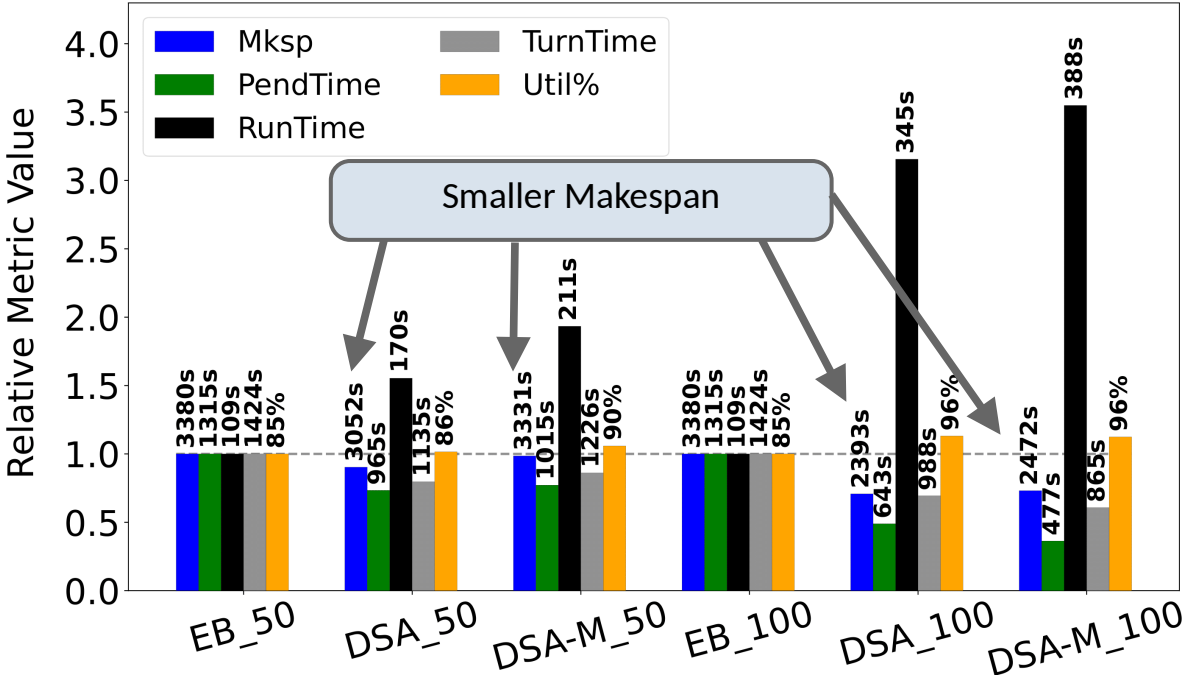
Performance Evalution: Scheduling



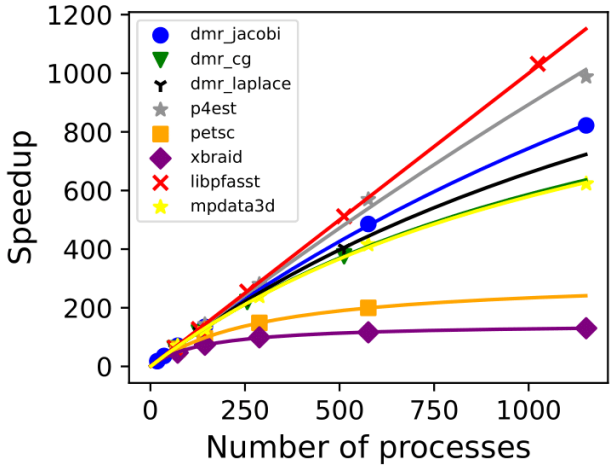
- Feitelson Workload Model
- 100 jobs
- 100 nodes



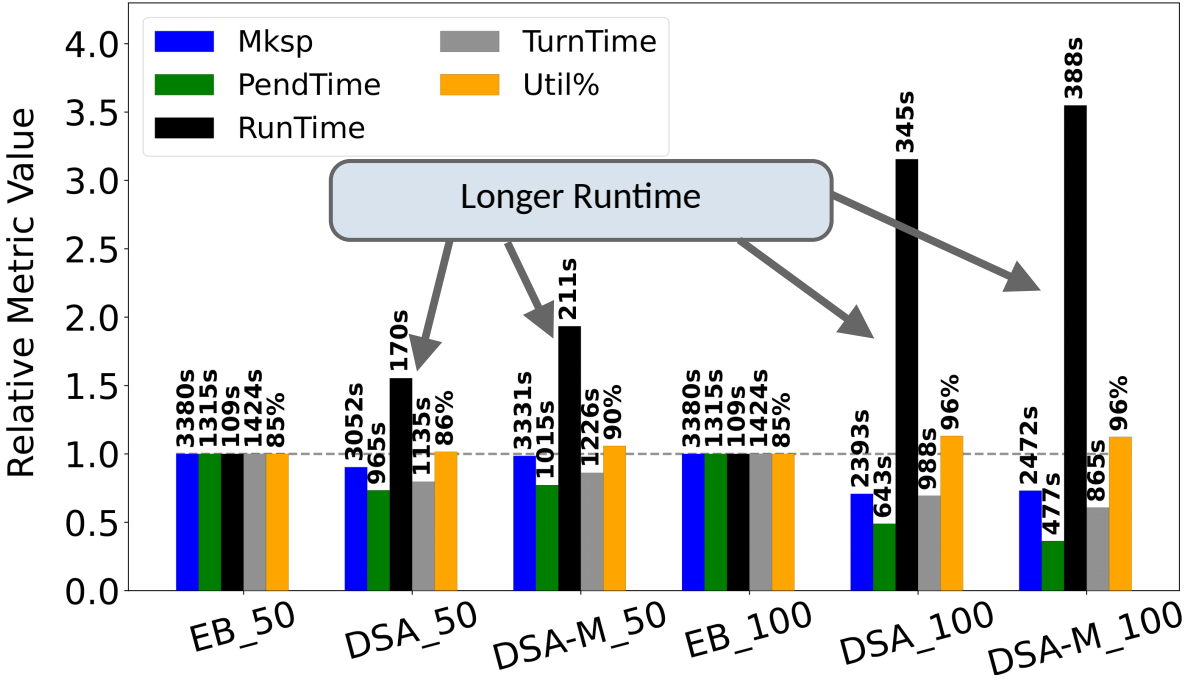
Performance Evalution: Scheduling



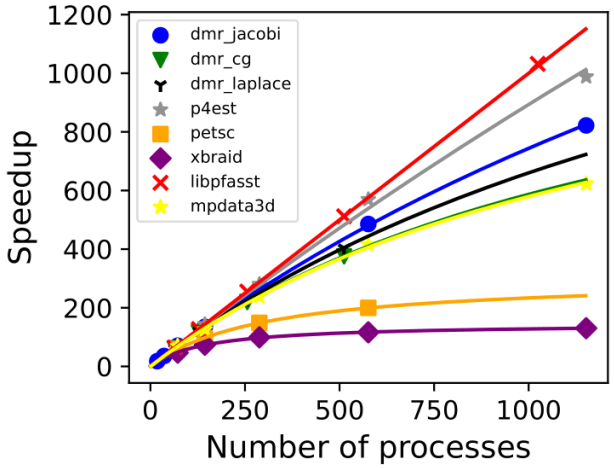
- Feitelson Workload Model
- 100 jobs
- 100 nodes



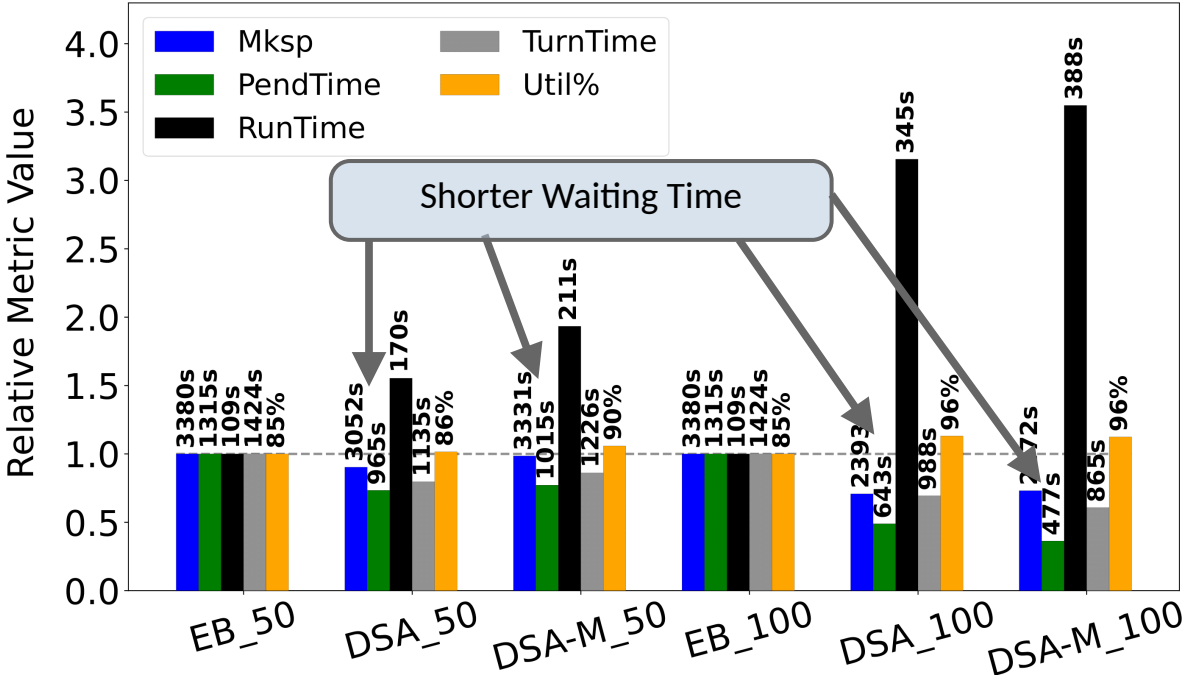
Performance Evalution: Scheduling



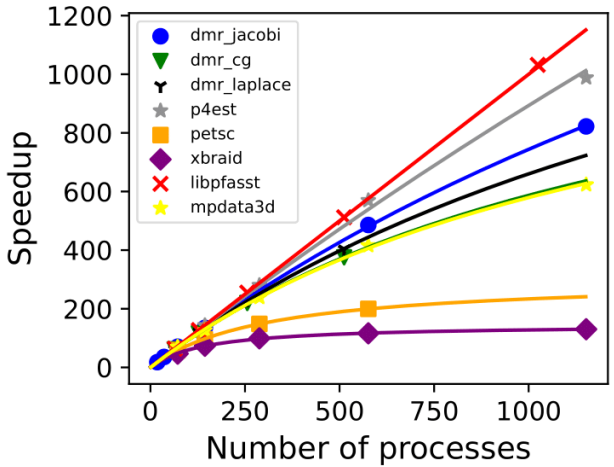
- Feitelson Workload Model
- 100 jobs
- 100 nodes



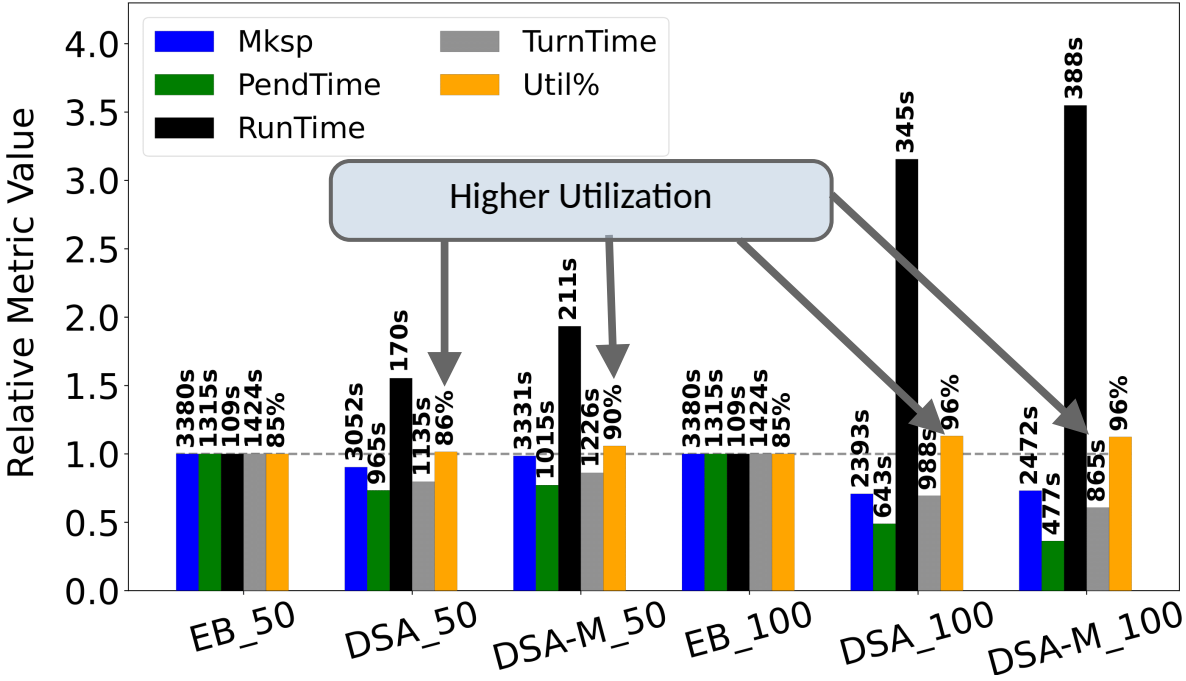
Performance Evalution: Scheduling



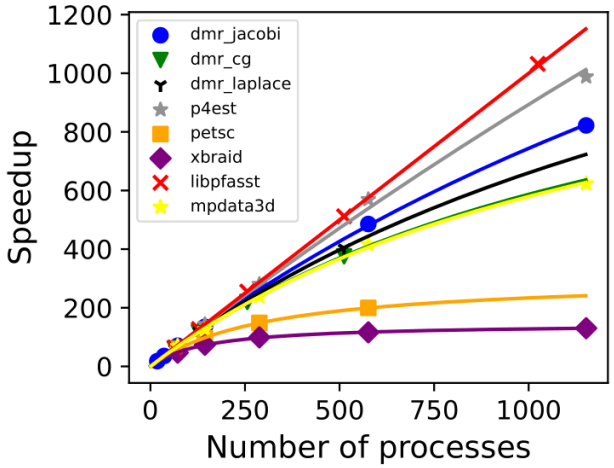
- Feitelson Workload Model
- 100 jobs
- 100 nodes



Performance Evalution: Scheduling



- Feitelson Workload Model
- 100 jobs
- 100 nodes



Papers & Portfolio

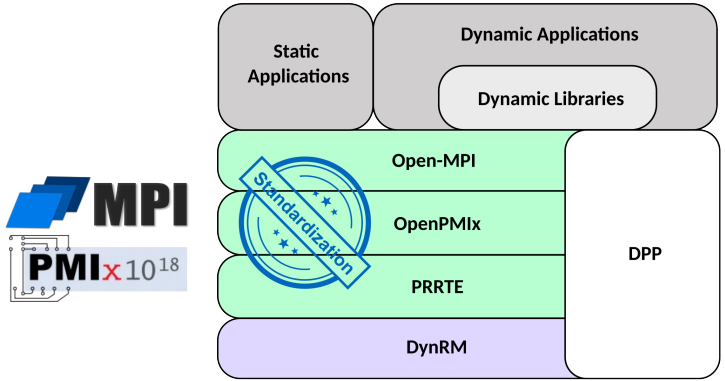
Dominik Huber
domi.huber@tum.de

Summary

We developed a new, **generic DRM design**, which:

- ☑ is **applicable to standard system software** (MPI, PMIx, ..)
- ☑ covers a **divers set of HPC applications** and libraries
- ☑ enables **fine-grained performance-aware dynamic scheduling**, and
- ☑ achieves **improvement of key system metrics**, e.g. throughput and utilization

Repositories & Testbed



Co-Design

Design Principles

This research was partially supported and funded by:

TIME-X
SEANERGYS
Energy Efficient Exascale
DEEP-SEA
SPONSORED BY THE
Federal Ministry of Education and Research
EuroHPC Joint Undertaking