Dynamic Resource Management for In-Situ Techniques using MPI-Sessions

EURO MPI

Yi Ju
Max Planck Computing & Data Facility
Garching near Munich, Germany

Dominik Huber
Technical University of Munich
Garching near Munich, Germany

Adalberto Perez
KTH Engineering Mechanics
Stockholm, Sweden

Philipp Ulbl
Max-Planck-Institut fur Plasmaphysik
Garching near Munich, Germany

Stefano Markidis
KTH Electrical Engineering and Computer Science
Stockholm, Sweden

Philipp Schlatter
Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany

Martin Schulz
Technical University of Munich
Garching near Munich, Germany

Martin Schreiber
Université Grenoble Alpes
Grenoble, France

Erwin Laure
Max Planck Computing & Data Facility
Garching near Munich, Germany

29.09.2024 Perth, Australia

This work has received funding from the European Union's Horizon 2020 JTI-EuroHPC research and innovation program under grant agreement number: 956748, ADMIRE, the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701, Time-X and under grant agreement No 101093261, Plasma-PEPSC.

Motivation

High Performance Computing (HPC) systems

- Rapid increase in computational capacity with heterogeneous computing recourse
- Relatively slow improvement of input/output (IO) subsystem
- Limited storage capacity

High Performance Computing (HPC) applications

Characteristics:

- Computationally expensive
- Requiring large storage for the results (tens of GB per simulation step)
- CPU underused by most GPU accelerated applications

Post-mortem data processing

Workflow:

- Simulation solver write results through IO subsystem to storage
- Data processor read the data through IO subsystem from storage

Disadvantage:

- Bottleneck in IO because of the IO bandwidth
- Limited frequency to preform data processing

In-situ data processing

Workflow:

- Data processer receive data from simulation solver without via IO subsystem and storage

Challenge:

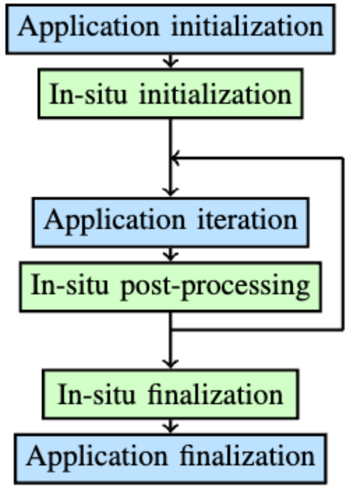
- Data processing could bring overhead to the simulation
- Data processing could influence the scalability of the simulation

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

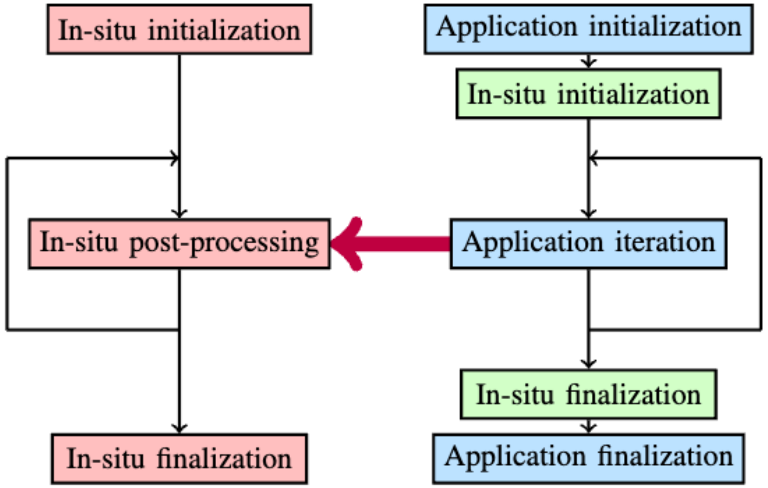
- Simulation waits until data processing finished



Asynchronous in-situ approach

Workflow:

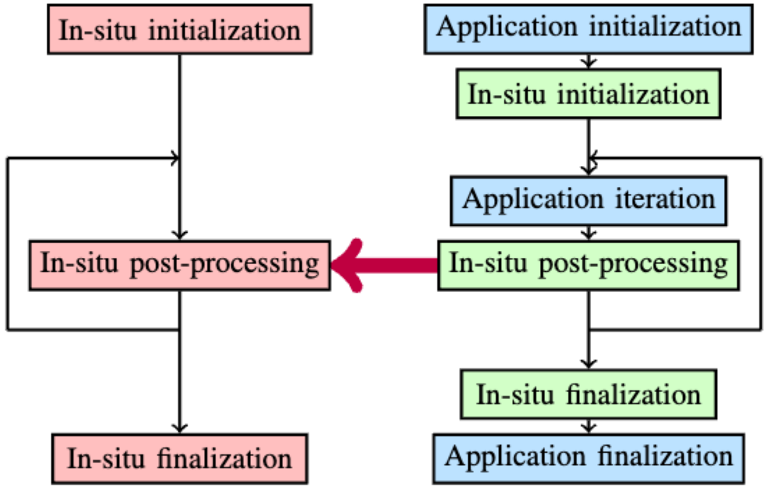
- Simulation sends data to separate computing resources and continues
- Data are processed concurrently



Hybrid in-situ approach

Workflow:

- First part of data processing is synchronous
- Second part of data processing is asynchronous



State-of-the-Art

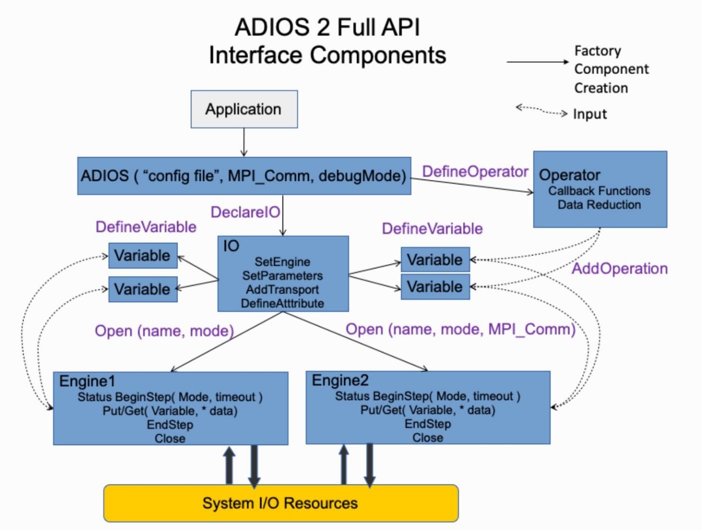
In-situ systems

- VisIt with Libsim
- ParaView with Catalyst
- SENSEI
- Adaptable IO System (ADIOS)

ADIOS

- Arbitrary data structure
- Runtime configuration
- Application programming interfaces (APIs) for multiple programming languages
- Operators such as lossless compression
- MPI-based data communication between arbitrary configuration

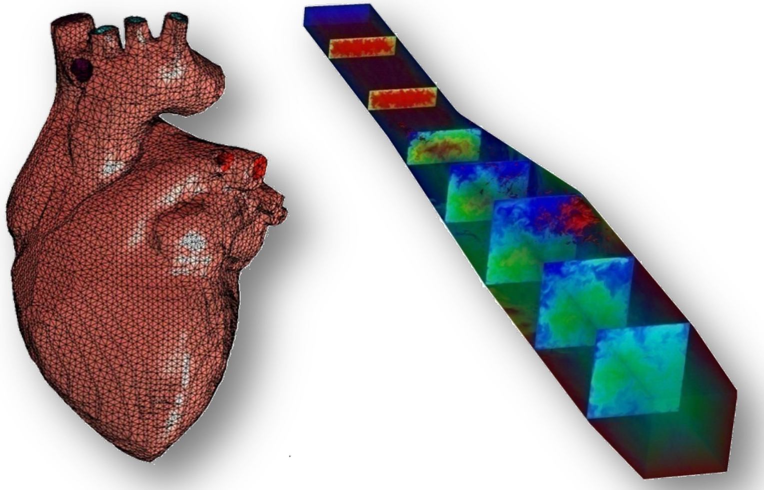
1



Simulation solver

Nek5000: [2]

- CPU version: Fortran
- GPU version: Fortran with OpenACC



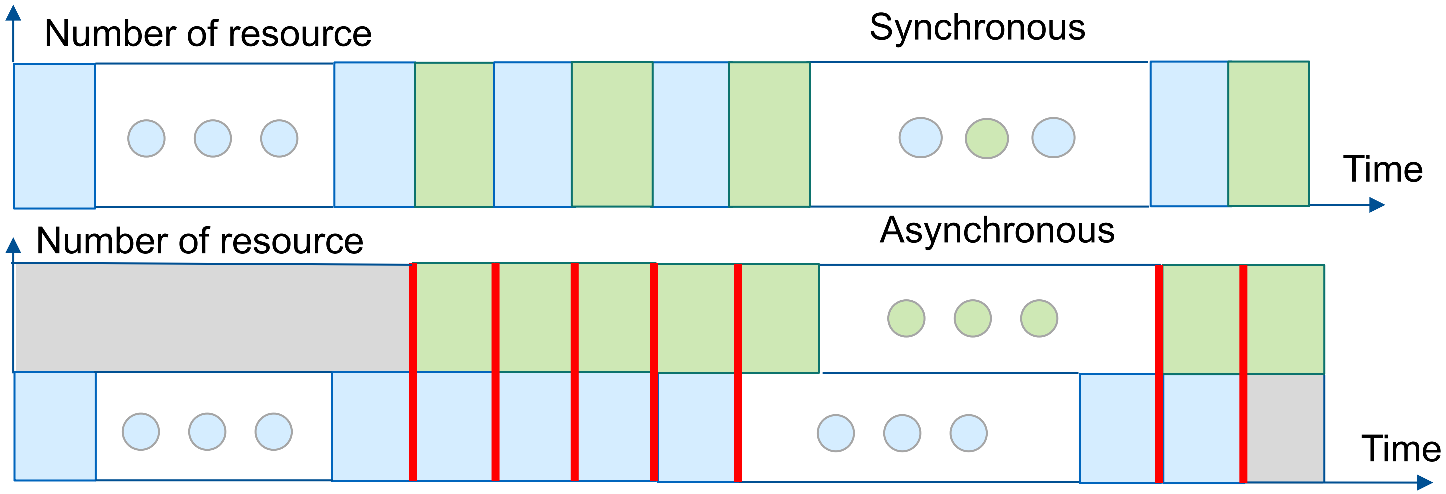
Characteristics:

- Direct Numerical Simulation (DNS) solver
- "Matrix-free"
- Scalability from "local domain"

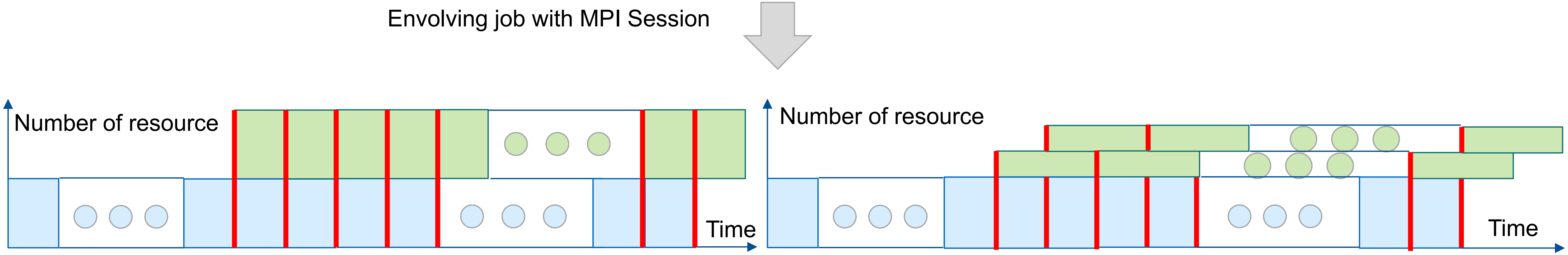
1: https://adios2.readthedocs.io/en/latest/components/components.html
2: https://github.com/Nek5000/Nek5000

CPU-Based Nek5000 with In-Situ Image Generation

However, it makes more sense to visualize the results after simulating for certain steps.



Envolving job with MPI Session



Nek5000 simulation	Image generation	Idle

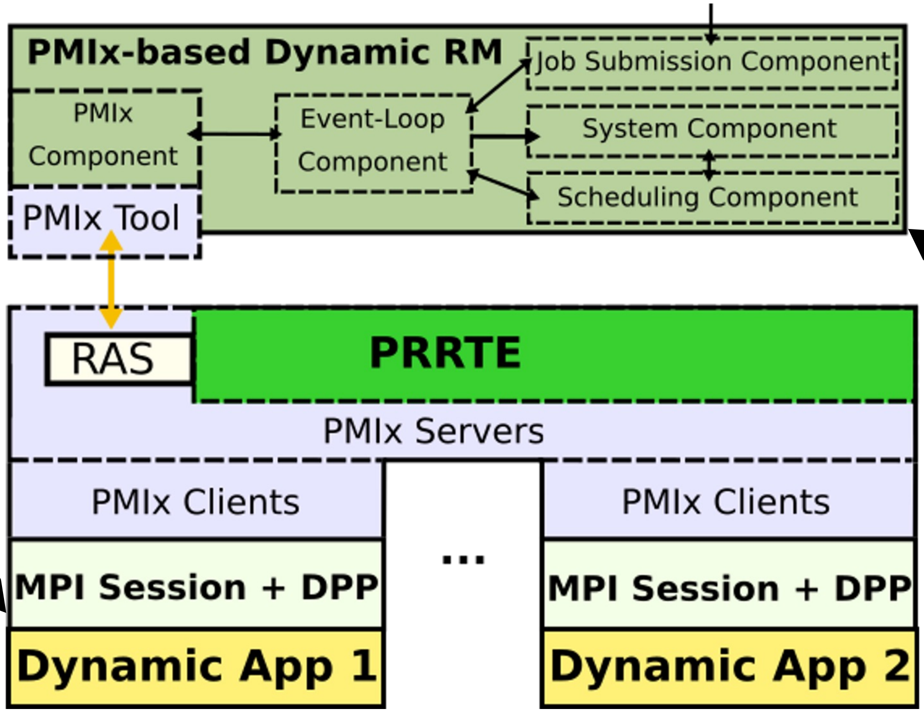
MPI Sessions and Dynamic Processes with PSets (DPP)

Goal: A generic application interface for dynamic resources

- Centered around processes, process sets and set operations
- Cooperative resource management between applications and Resource Manager (RM)

Prototype:

- OMPI
- OpenPMIx
- PRRTE
- PMIx-based RM

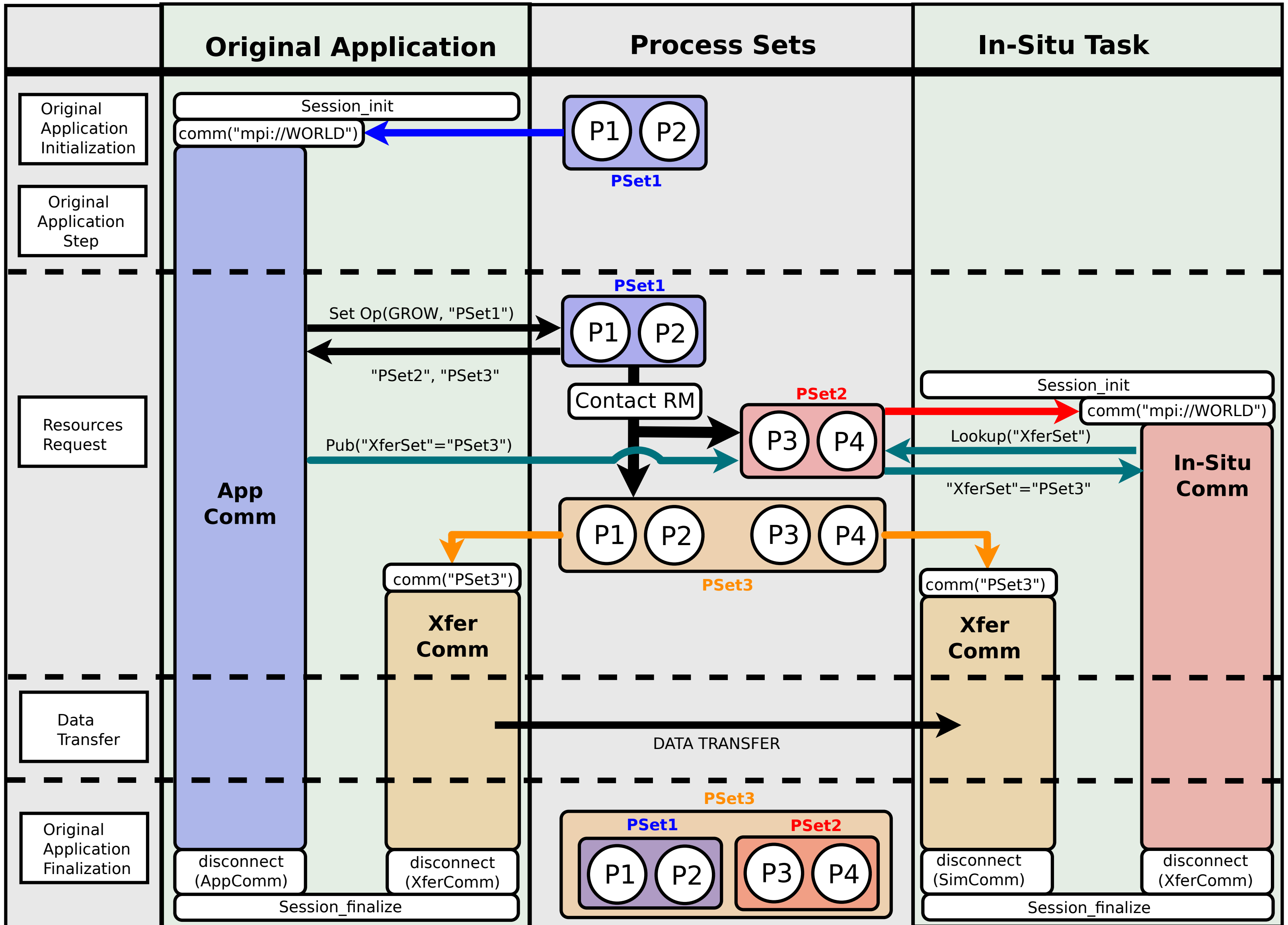


Optionally: plug in your own custom scheduling policy using Python

MPI Interface for writing dynamic applications

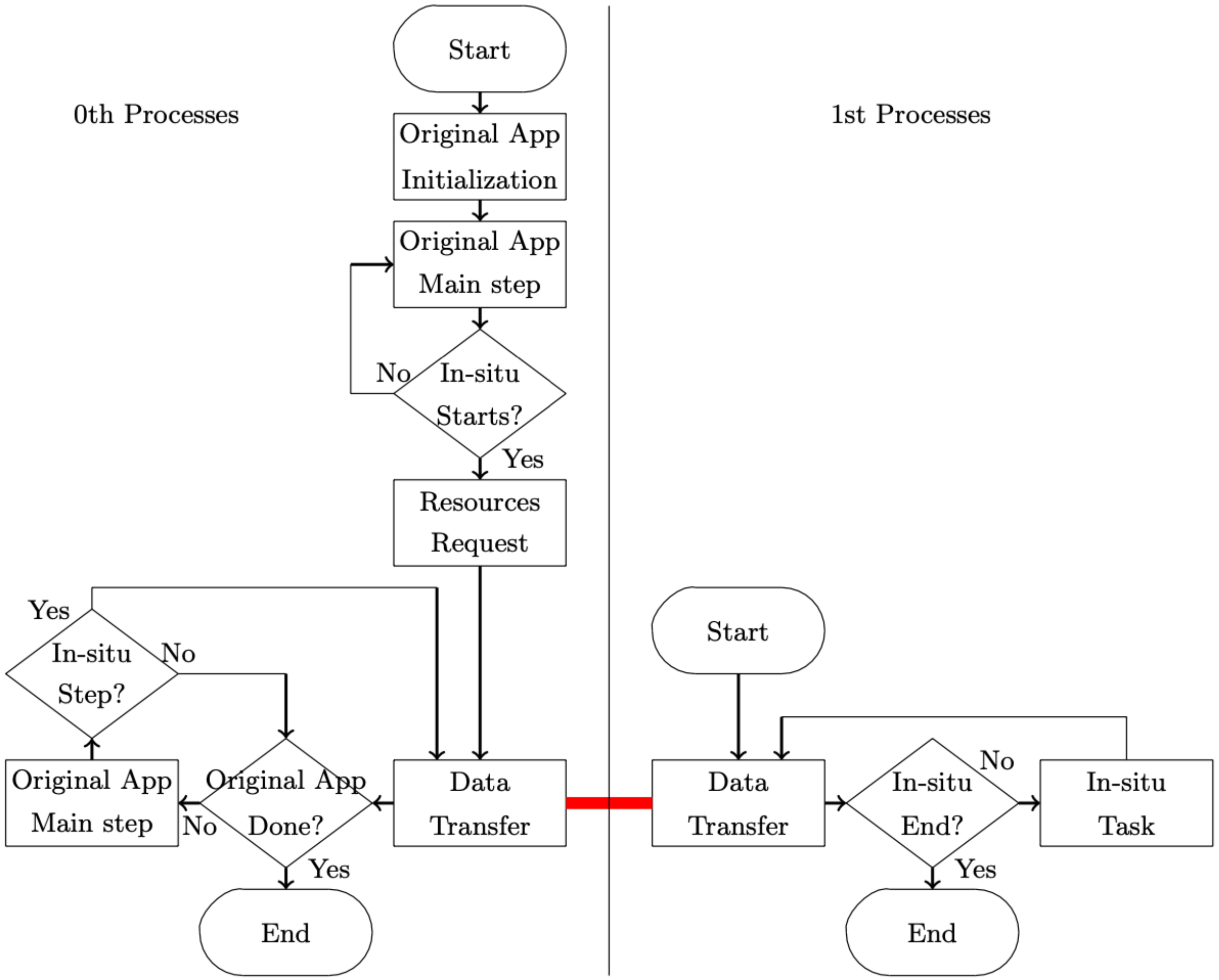
Design Principles of Dynamic Resource Management for High-Performance Parallel Programming Models (https://arxiv.org/abs/2403.17107)

MPI Sessions and Dynamic Processes with PSets (DPP)

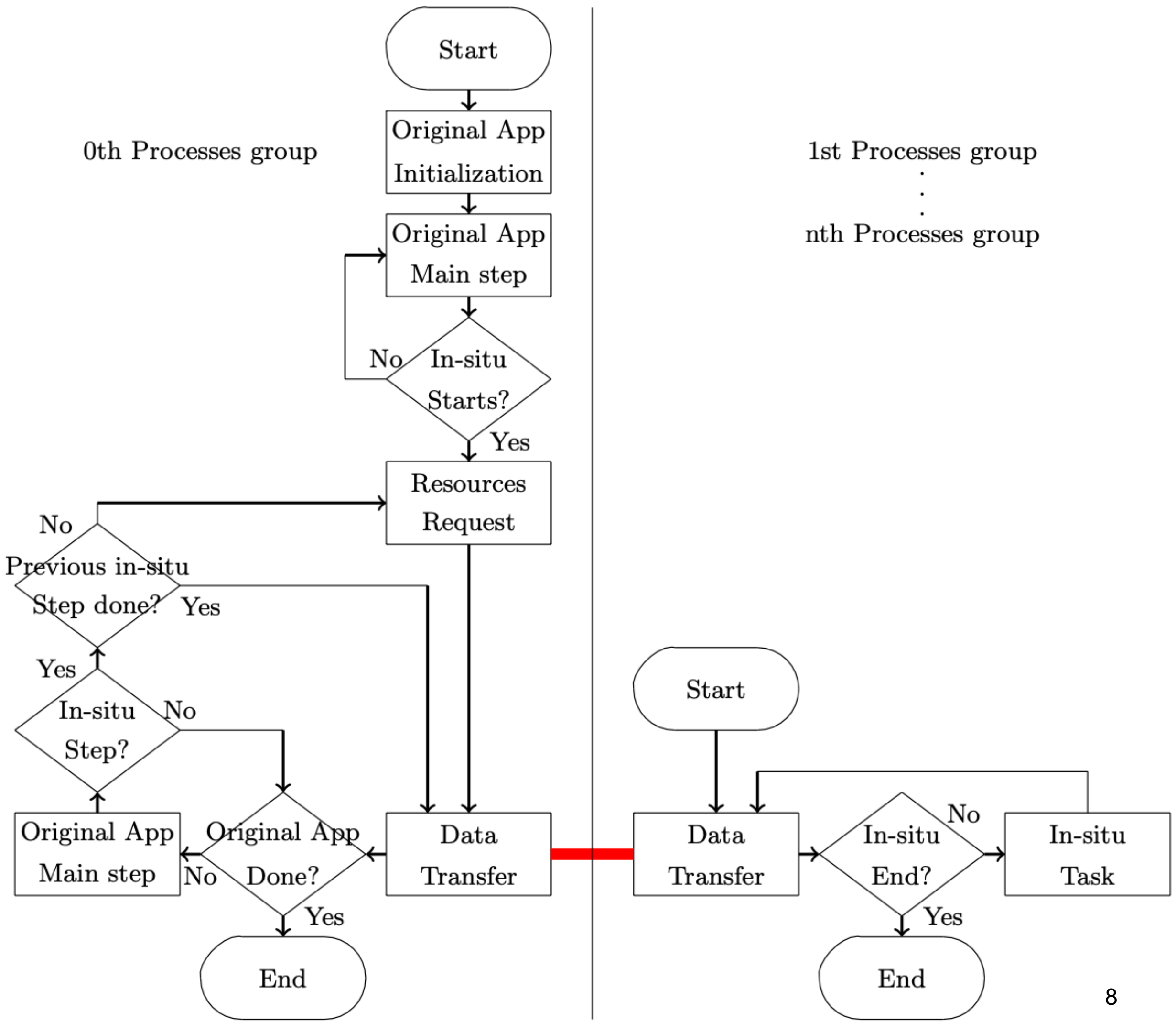


The workflow of the dynamic in-situ technique

Basic dynamic in-situ technique

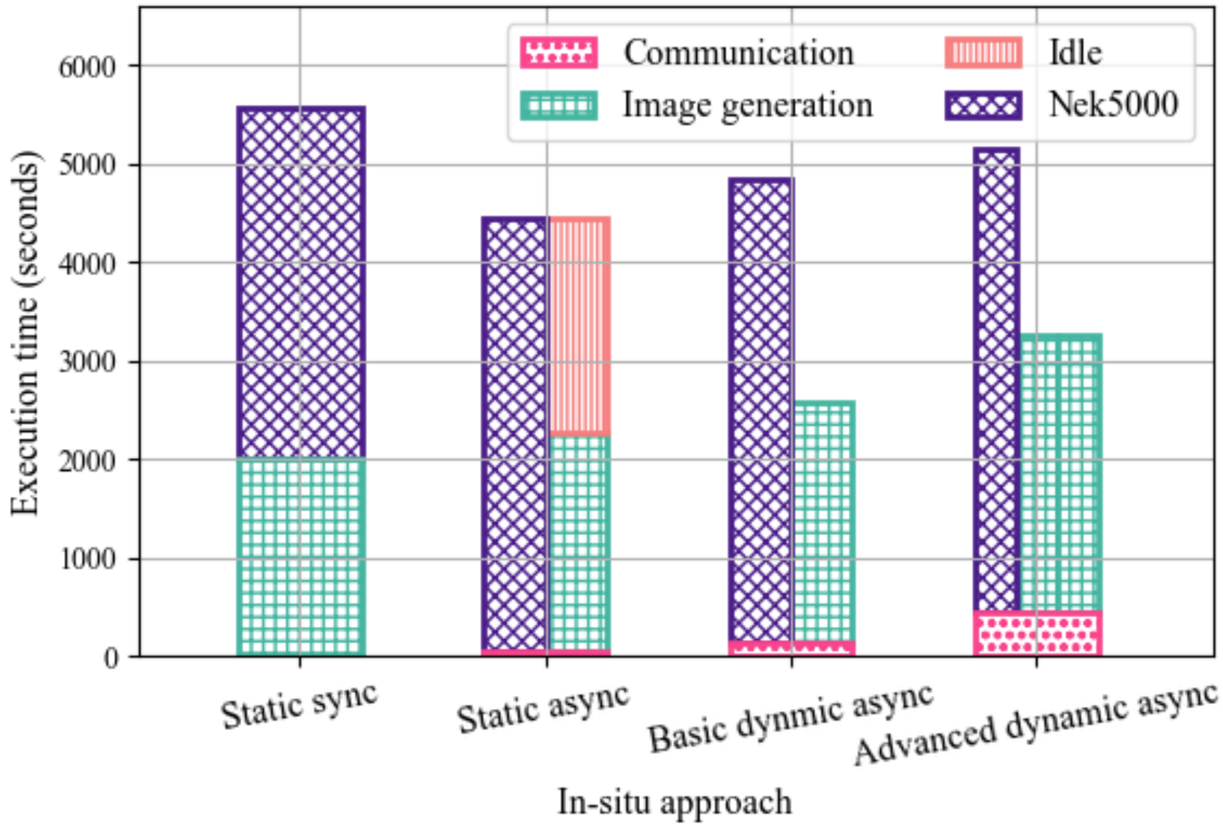


Advanced dynamic in-situ technique

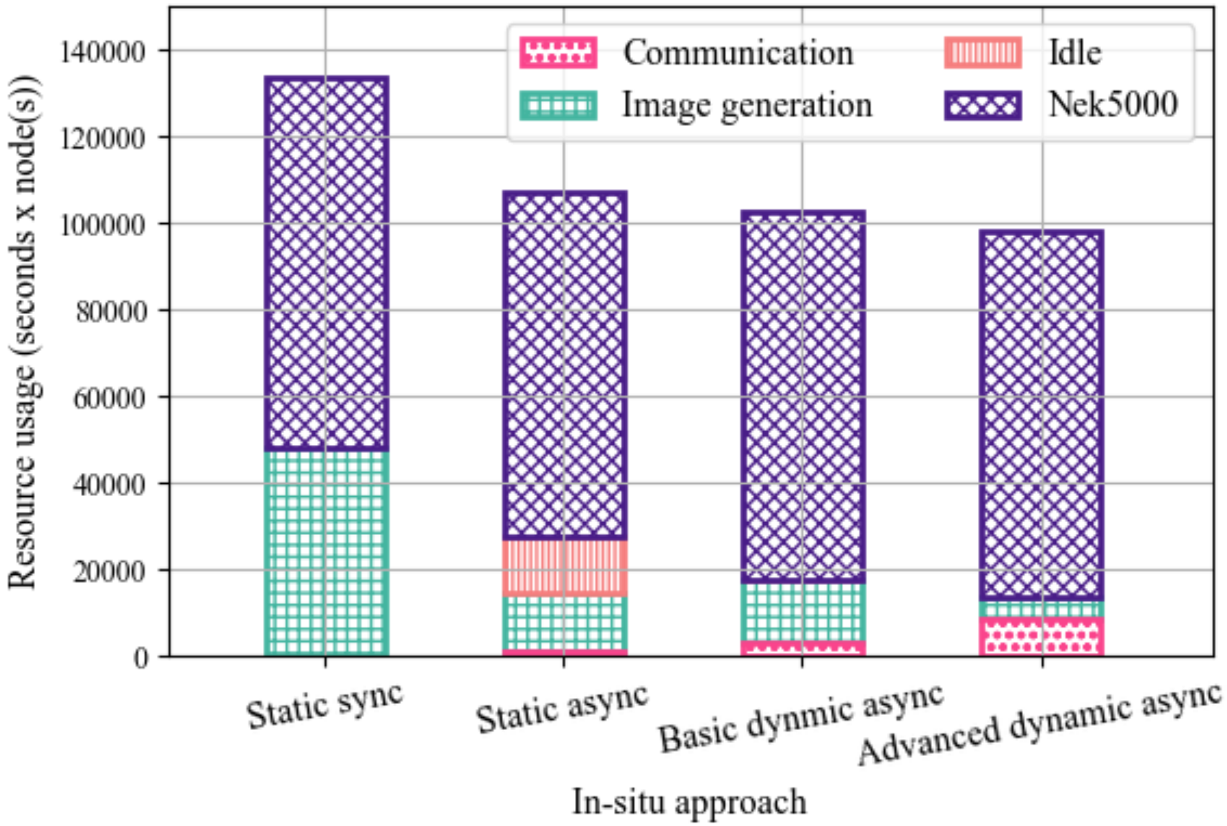


CPU-Based Nek5000 with In-Situ Image Generation

1



2



1: Execution time of Nek5000 with in-situ image generation.2: Resource usage of Nek5000 with in-situ image generation.

Dynamic Resource Management for In-Situ Techniques using MPI-Sessions

Approaches

- With MPI-Session and DPP, in-situ technique can be combined with dynamic resource management.
- The basic dynamic in-situ technique adds once resources.
- In the advanced dynamic in-situ technique, instead of the user specifying the additional resources, the program automatically asks for a small group of resources and uses them for the first step of the in-situ task.

Case study

- Although the Nek5000 with the static asynchronous in-situ image generation has the shortest execution time, the dynamic asynchronous in-situ approaches take a shorter time than the static synchronous approach.
- Dynamic asynchronous approaches have lower resource usage because they avoid idle time..
- The advanced dynamic asynchronous in-situ image generation does not need any performance model and can ask for just enough resources during runtime automatically and, therefore, has the lowest resource usage.

Outlook

- Dynamic in-situ techniques enabling the MPI binding of processes
- A shared computing cluster with dynamic resource management policies in place

This work has received funding from the European Union's Horizon 2020 JTI-EuroHPC research and innovation program under grant agreement number: 956748, ADMIRE, the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701, Time-X and under grant agreement No 101093261, Plasma-PEPSC.