In-Situ/In-Transit Data Transformation Using Resource Efficiently

ADMIRE users day

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

Adalberto Perez
KTH Engineering Mechanics
Stockholm, Sweden

Laura Bellentani
CINECA
Casalecchio di Reno, Italy

Niclas Jansson
KTH Computer Science and Communication
Stockholm, Sweden

Dominik Huber
Technical University of Munich
Garching near Munich, Germany

Martin Schreiber
Université Grenoble Alpes
Grenoble, France

Stefano Markidis
KTH Electrical Engineering and Computer Science
Stockholm, Sweden

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

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

12.12.2023 Barcelona, Spain

This project has received funding from the European Union's Horizon 2020 JTI-EuroHPC research and innovation programme, with grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1

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

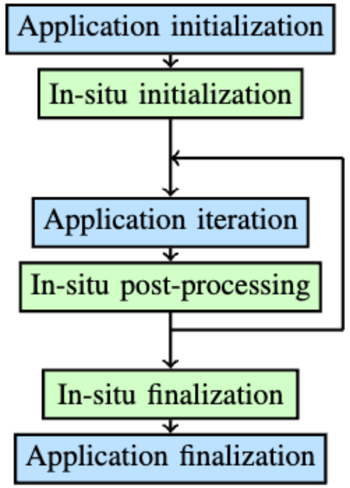
Challenge:
- Data processing could bring overhead to the simulation
- Data processing could influence the scalability of the simulation

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

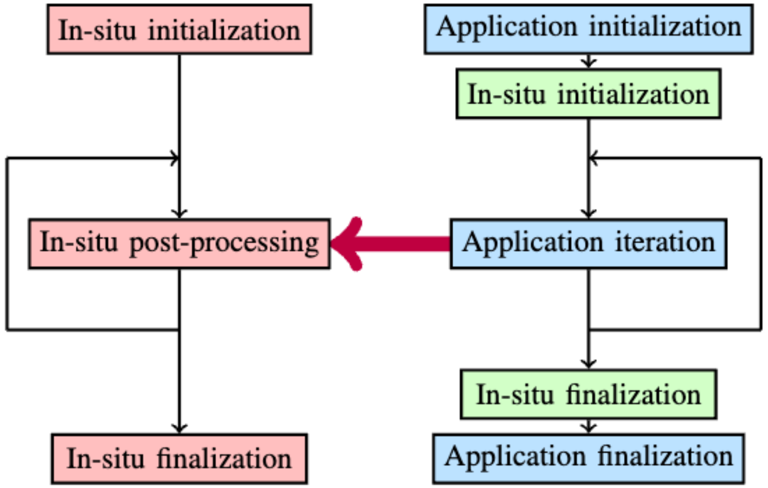
- Simulation waits until data processing finished



Asynchronous in-situ approach

Workflow:

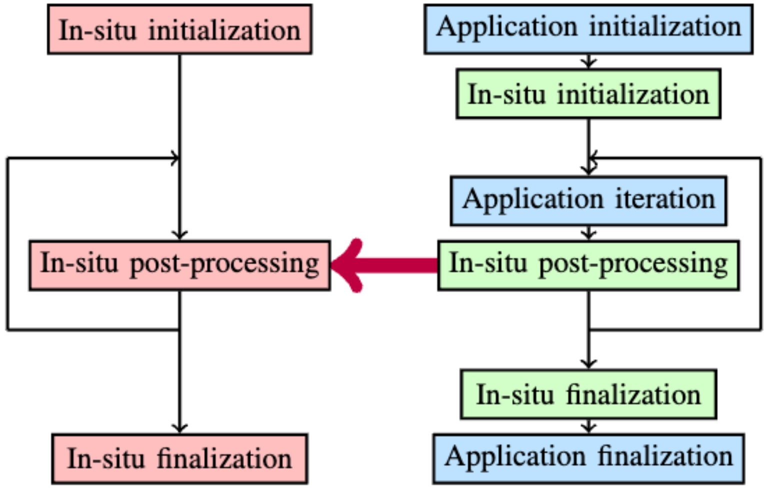
- Simulation sends data to separate computing resources and continues
- Data are processed concurrently

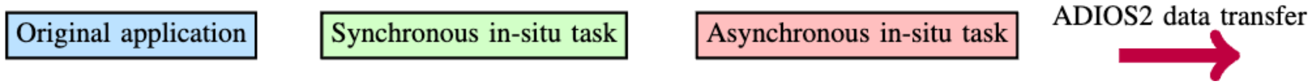


Hybrid in-situ approach

Workflow:

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





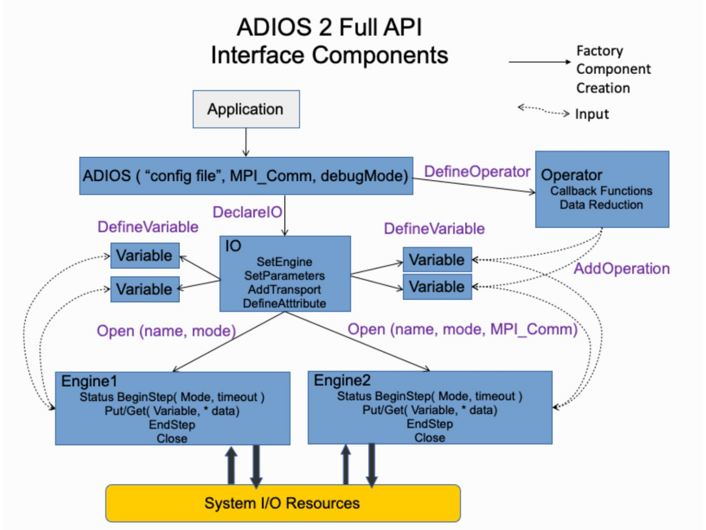
State-of-the-Art

In-situ systems

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

ADIOS

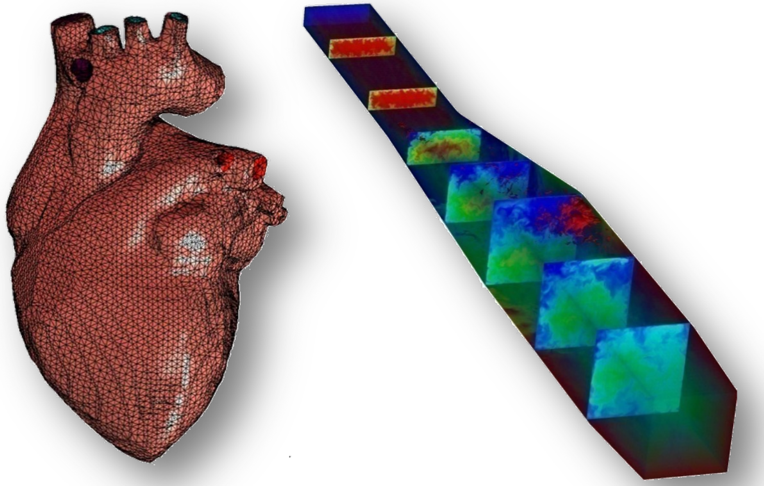
- Arbitrary data structure
- Runtime configuration
- Application programming interfaces (APIs) for multiple programming languages
- Operators such as lossless compression
- MPI-based data communication between arbitrary configuration [1]



Simulation solver

Nek5000: [2]

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



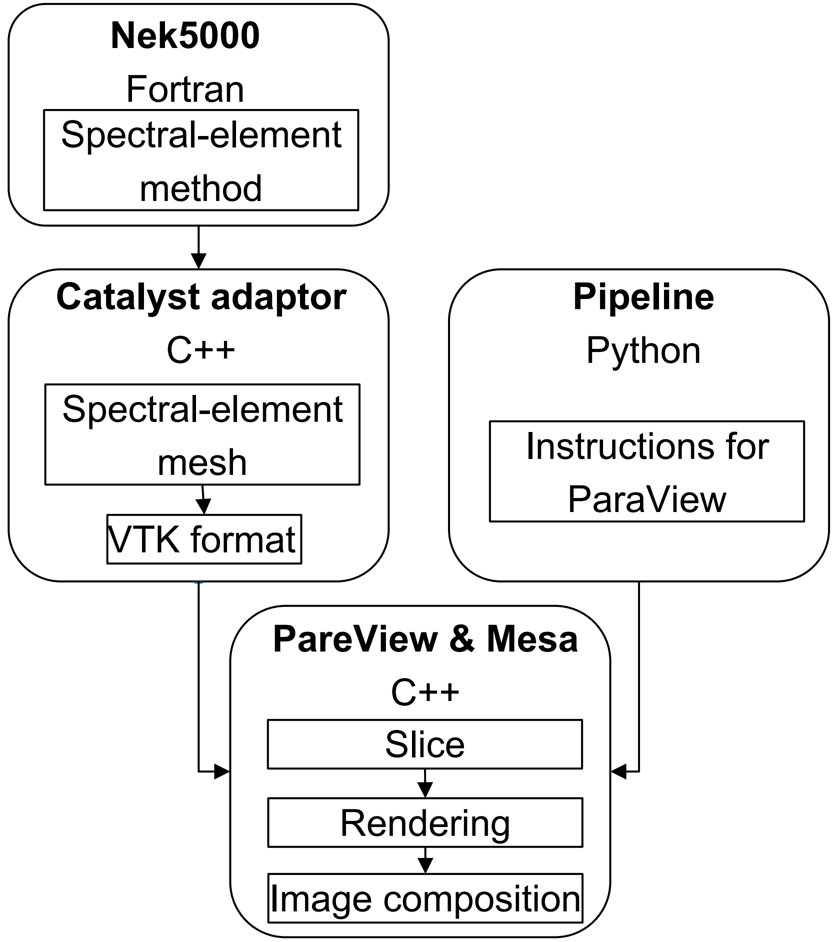
Characteristics:

- 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

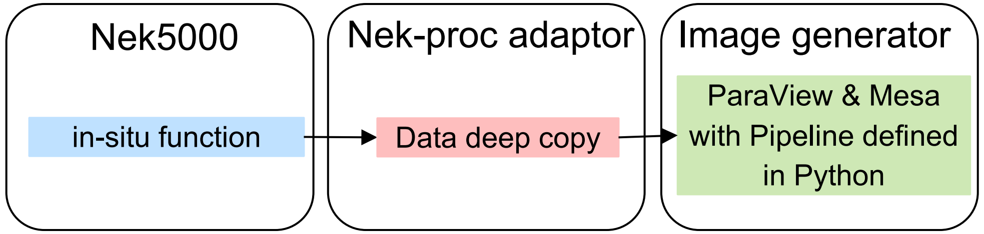
Nek5000 with Synchronous and Asynchronous Image Generation

Image generation [1]

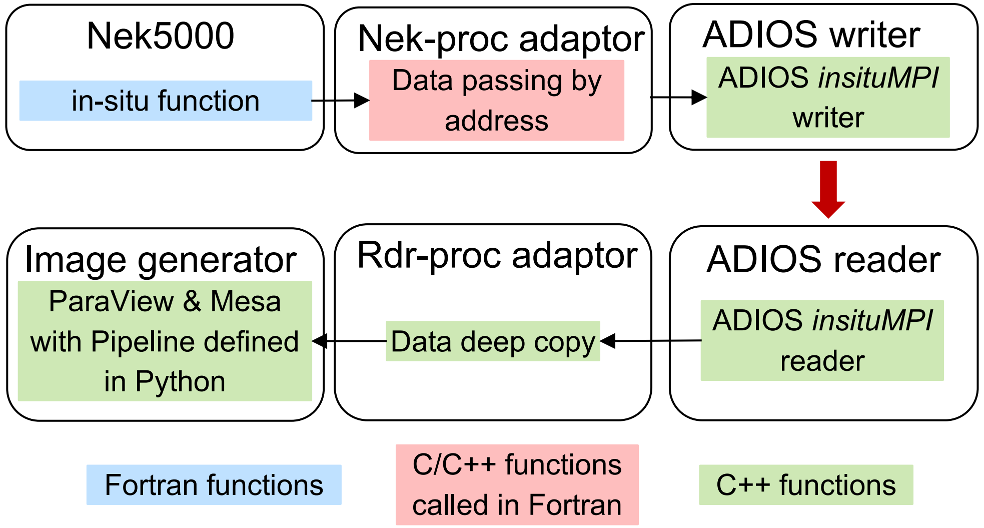


In-situ approach

- Nek5000 with synchronous image generation:



- Nek5000 with asynchronous image generation:



1: Original from "M. Atzori, W. Ko¨pp, S. W. Chien, D. Massaro, F. Mallor, A. Peplinski, M. Rezaei, N. Jansson, S. Markidis, R. Vinuesa et al., "In situ visualization of large-scale turbulence simulations in nek5000 with paraview catalyst," The Journal of Supercomputing, vol. 78, no. 3, pp. 3605–3620, 2022."

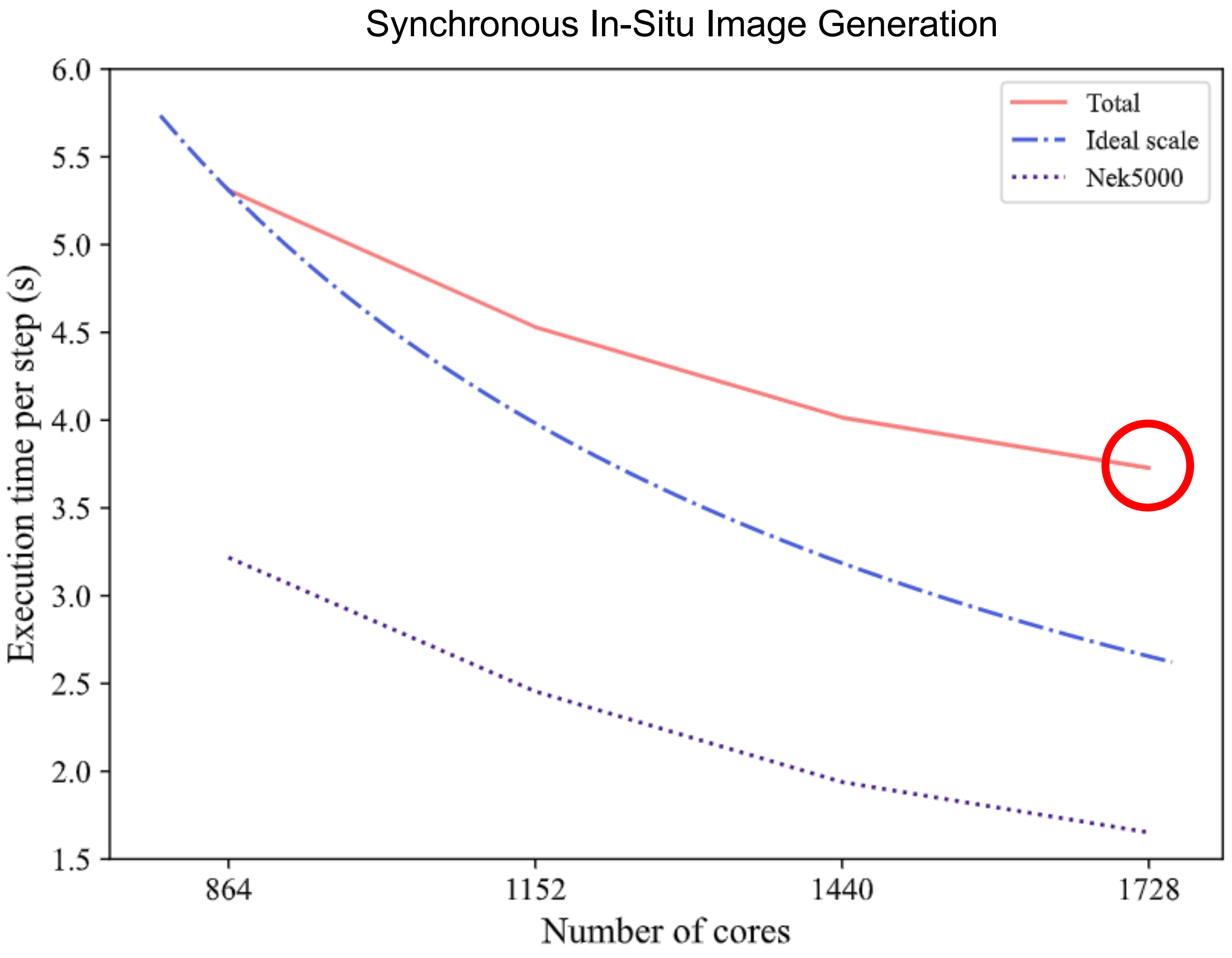
malleable data solutions for HPC

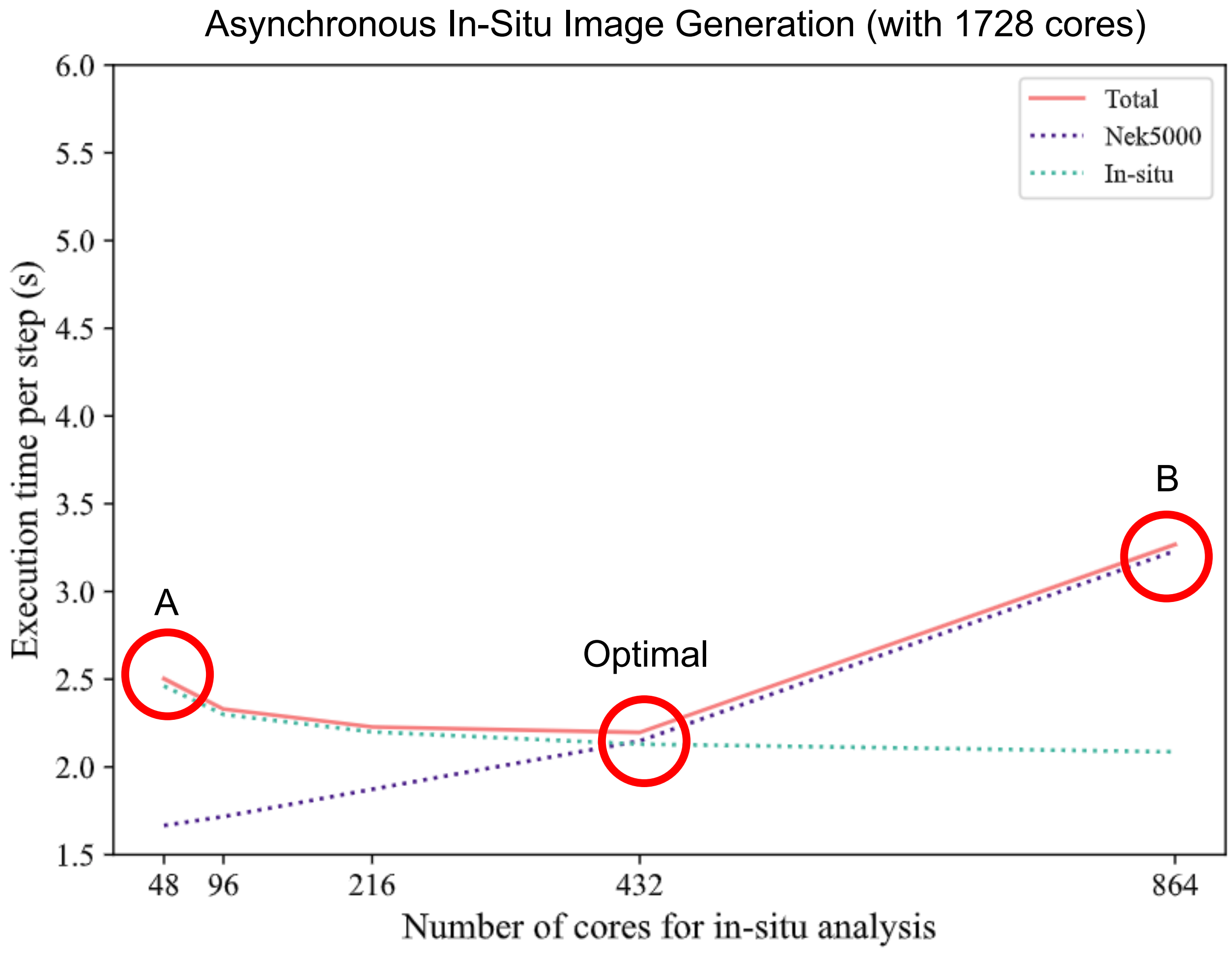
MPCDF MAX PLANCK COMPUTING & DATA FACILITY

CPU-based Nek5000 with Synchronous and Asynchronous Image Generation

(45G VTK file for one image avoided)

1

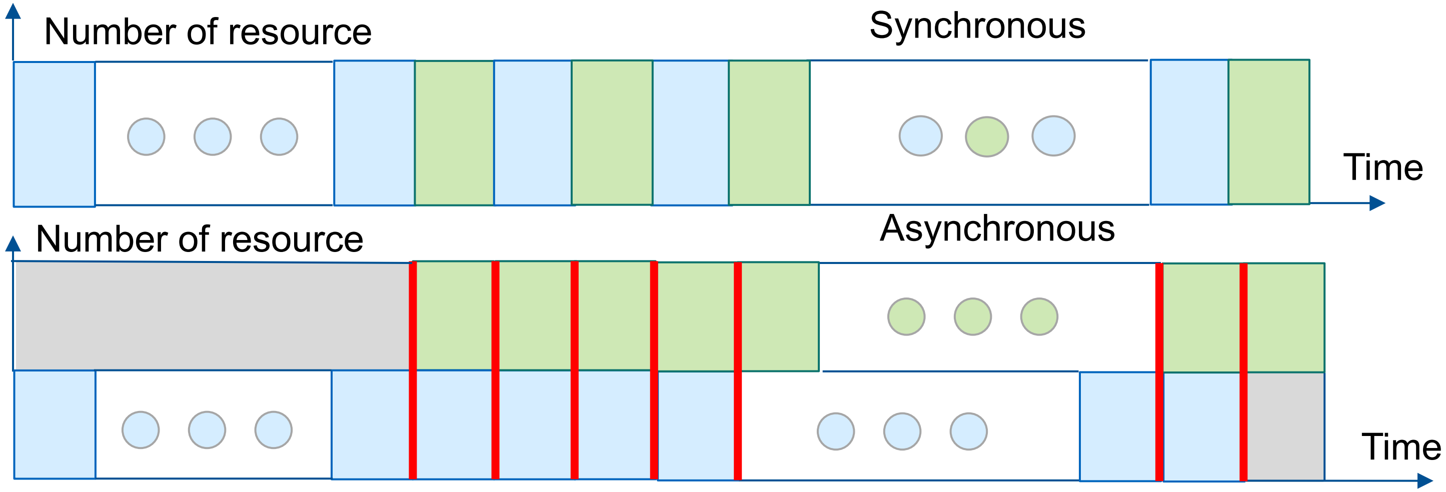




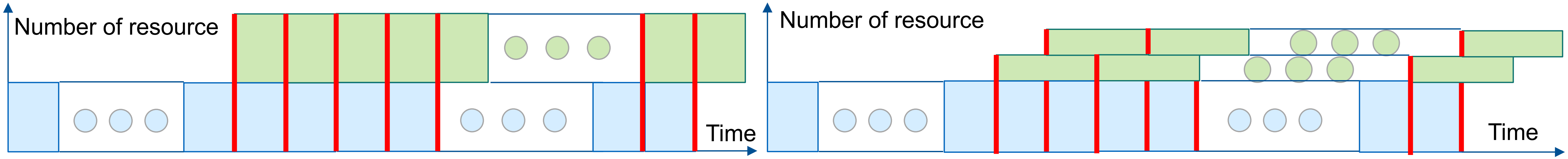
1: Execution time of Nek5000 with synchronous in-situ image generation every two steps on Raven supercomputer (left) and asynchronous in-situ image generation every two steps on 24 Raven nodes (right).

CPU-Based Nek5000 with In-Situ Image Generation

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



Cooperation with Time-X [1]
Envolving job with MPI Session



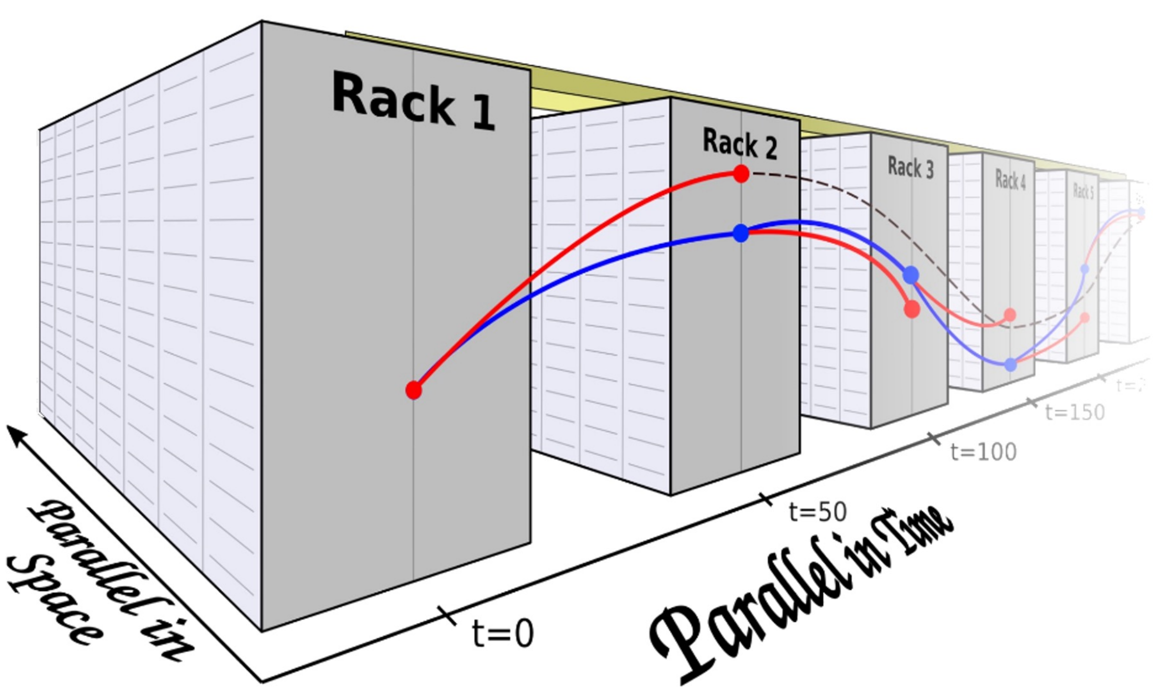
Nek5000 simulation	Image generation	Idle

1: https://time-x.eu/

Time-X: TIME parallelization for eXascale computing and beyond

Technical Univ. of Munich
Université Grenoble Alpes

- **Main idea:** time direction as an **additional direction for parallelization** of PDE solvers



Time-X - Interdisciplinary research:
- Mathematical Theory
- Computational Science
- Application Development

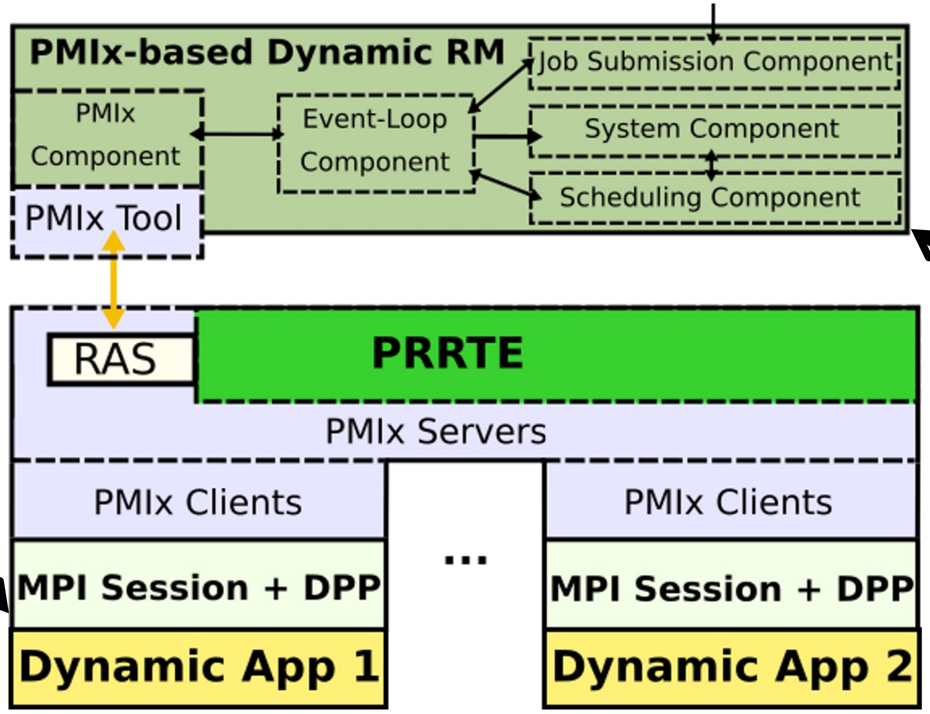
Time-X @ TUM: Adaptive PinT methods
- Dynamically change number of parallel timesteps
- Requires application interface for dynamic resources → **DPP**

Time-X: Dynamic Processes with PSets (DPP)

Technical Univ. of Munich
UGA Université Grenoble Alpes

Goal: A generic application interface for dynamic resources
- Centered around processes, process sets and set operations
- Cooperative resource management between applications and RM

Prototype:
- OMPI
- OpenPMIx
- PRRTE
- PMIx-based RM

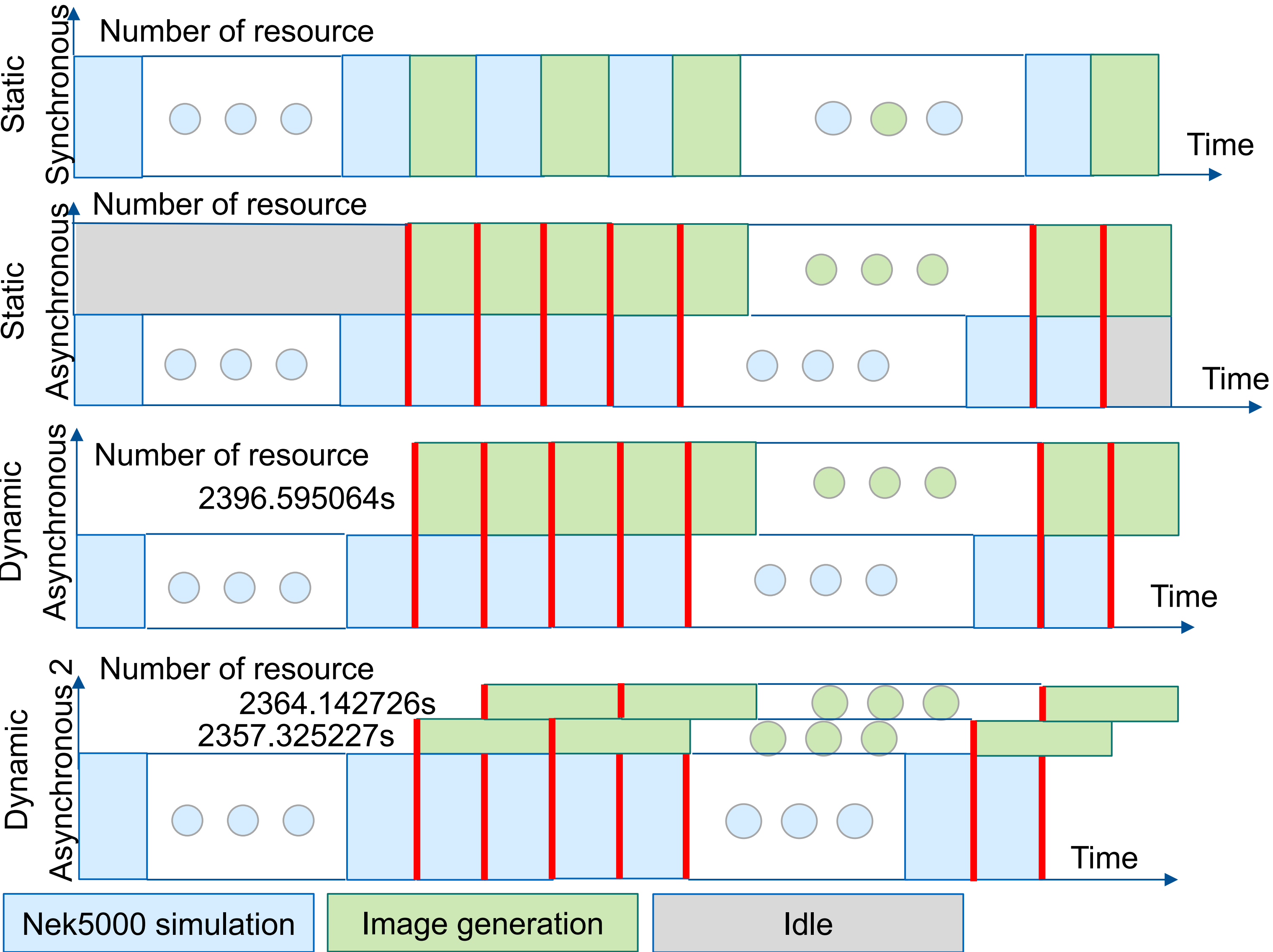


Optionally: plug in your own custom scheduling policy using Python

MPI Interface for writing dynamic applications

Adaptive CPU-Based Nek5000 with In-Situ Image Generation

2002 steps, start to do in-situ at 1002 every 2 step



24 nodes for Nek5000 and synchronous image generation
Total time: 5567.10s
In-Situ time: 2005.55s
Resource usage: 133610.40 s nodes

24 nodes for Nek5000 (3/4 of cores per socket) and asynchronous image generation (1/4 cores per socket)
Total time: 4450.62s
In-Situ time: 40.9199s
Resource usage: 106814.88 s nodes

18 nodes for Nek5000 and 6 nodes for asynchronous image generation
Total time: 4837.34s
In-Situ time: 124.05s
Resource usage: 101712.62 s nodes

18 nodes for Nek5000 and 1 node (2 sockets) for asynchronous image generation
Total time: 5155.73s
In-Situ time: 451.839s
Resource usage: 95601.45 s nodes

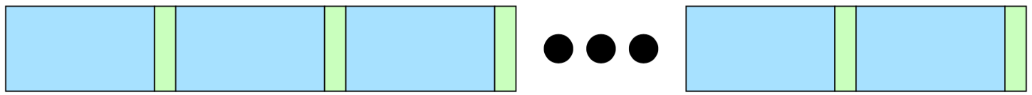
Performance Model of In-Situ Techniques

Performance model of original application: $f(\vec{x}) = f_{setup} + n\, f_{main} + f_{final}$

Performance model of in-situ task: $g(\vec{x}) = g_{setup} + n\, g_{main} + g_{final}$

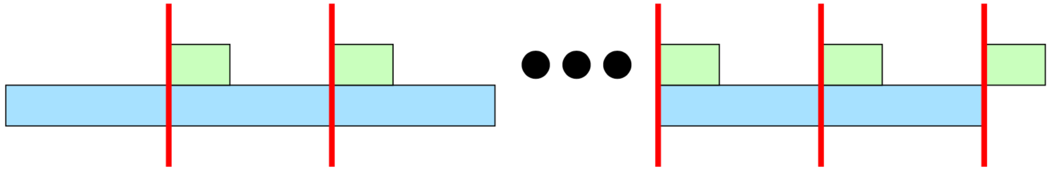
Freqency of in-situ task: υ

Performance model of synchronous in-situ technique:



$$\psi(\vec{x}) = f_{setup} + g_{setup} + n\, f_{main} + \upsilon n\, g_{main} + f_{final} + g_{final}$$

Performance model of asynchronous in-situ technique:



$$\psi(\vec{x}) = \max\left(f_{setup} + {}^{1}/_{\upsilon}\, f_{main}, g_{setup}\right) + (\upsilon n - 1)\max\left({}^{1}/_{\upsilon}\, f_{main}, g_{main}\right) + \max\left(f_{final}, g_{main} + g_{final}\right) + \psi_{comm}$$

Performance Model of In-Situ Techniques

Performance models of original application and of in-situ task are generated with Extra-P. [1]

We use coefficient of determination R^2 to evaluate the performance models derived.

Case study	R^2 (Synchronous)	R^2 (Asynchronous)
CPU-based QE with data compression	0.9780	0.9770
GPU-based QE with data compression	0.9180	0.8840
CPU-based Nek5000 with data compression	0.9983	0.9988
CPU-based Nek5000 with image generation	0.9994	0.9940
CPU-based NEKO with data compression	0.9993	0.9896
CPU-based NEKO with image generation	0.9986	0.9982
GPU-based NEKO with data compression	0.9838	0.9451
GPU-based NEKO with image generation	0.9945	0.9472

https://github.com/extra-p/extrap

In-Situ/In-Transit Data Transformation Using Resource Efficiently

Approaches

- The synchronous in-situ approach: simulation waits until data process finished
- The asynchronous in-situ approach: simulation sends data to separate computing resources and continues, while data are processed concurrently
- The hybrid in-situ approach: the first part of data process is synchronous; the second part of data process is asynchronous.

Case study

- CPU-based Nek5000 with asynchronous in-situ image generation shows that poor scalability of image generation makes asynchronous approach more beneficial.
- Adaptive CPU-based Nek5000 with asynchronous in-situ image generation shows that in-situ technique with dynamic resource allocation can reduce the resource usage.
- Performance model of in-situ techniques could be derived from performance models of original applications and in-situ tasks generated from Extra-P and it proved to be accurate with CFD applications Nek5000 and NEKO and Molecular Dynamic application Quantum Espresso with in-situ tasks.

Outlook

- Deep learning training as in-situ task
- In-situ tasks to exascale simulation

HORIZON 2020 EuroHPC Joint Undertaking
This project has received funding from the European Union's Horizon 2020 JTI-EuroHPC research and innovation programme, with grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1