Understanding the Impact of Synchronous, Asynchronous, and Hybrid In-Situ Techniques in Computational Fluid Dynamics Applications

eScience Session 7: HPC and eScience

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

Adalberto Perez
KTH Engineering Mechanics
Stockholm, Sweden

Stefano Markidis
KTH Electrical Engineering and Computer Science
Stockholm, Sweden

Philipp Schlatter
KTH Engineering Mechanics
Stockholm, Sweden

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

14.10.2022 Salt Lake City, Utah, USA

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

Computational fluid dynamics (CFD)

Characteristics:

- Computationally expensive
- Requiring large storage for the results (tens of GB per simulation step)

In this study we use CFD as use case

Motivation

Computational fluid dynamics (CFD)

Characteristics:

- Computationally expensive
- Requiring large storage for the results (tens of GB per simulation step)

In this study we use CFD as use case

High Performance Computing (HPC) systems

- Rapid increment in computational capacity
- Relatively slow development in input/output (IO) subsystem
- Limit storage capacity

Motivation

Computational fluid dynamics (CFD)

Characteristics:
- Computationally expensive
- Requiring large storage for the results (tens of GB per simulation step)

In this study we use CFD as use case

High Performance Computing (HPC) systems

- Rapid increment in computational capacity
- Relatively slow development in input/output (IO) subsystem
- Limit storage capacity

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

Motivation

Computational fluid dynamics (CFD)

Characteristics:

- Computationally expensive
- Requiring large storage for the results (tens of GB per simulation step)

In this study we use CFD as use case

High Performance Computing (HPC) systems

- Rapid increment in computational capacity
- Relatively slow development in input/output (IO) subsystem
- Limit storage capacity

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, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished

Simulaion

Data processing

Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished

Simulaion

Data processing

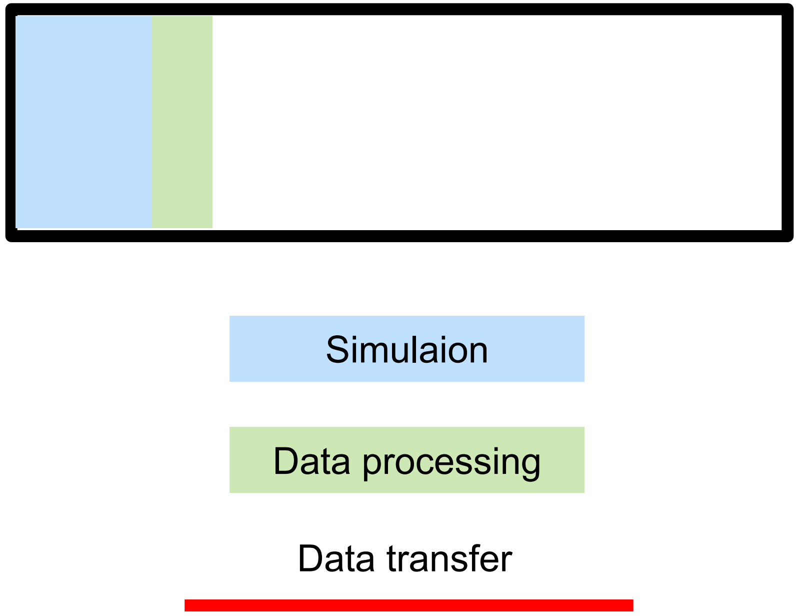
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished

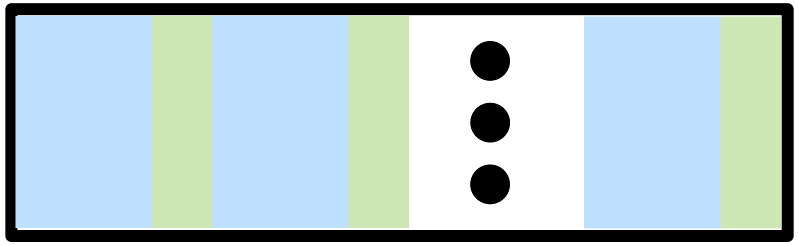


Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

Data processing

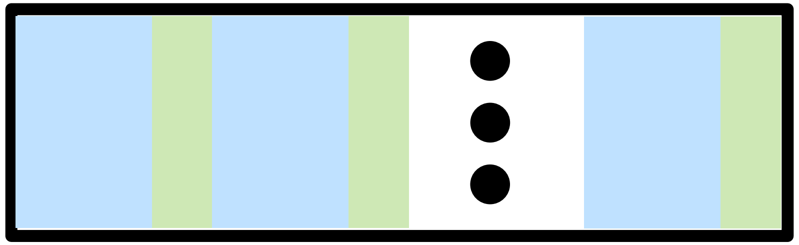
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

Data processing

Data transfer

Asynchronous in-situ approach

Workflow:

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

Simulaion

Data processing

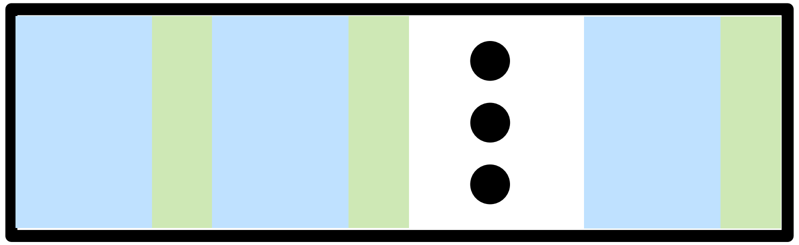
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

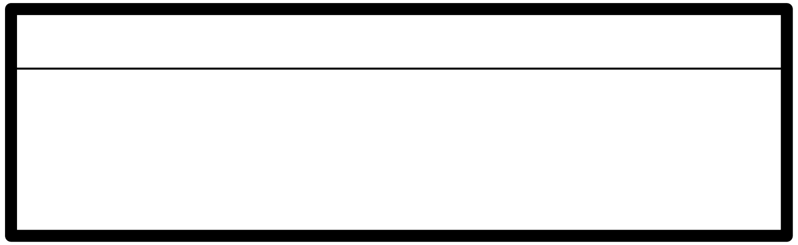
Data processing

Data transfer

Asynchronous in-situ approach

Workflow:

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



Simulaion

Data processing

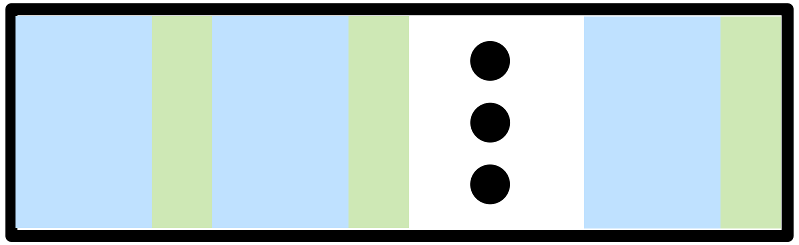
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

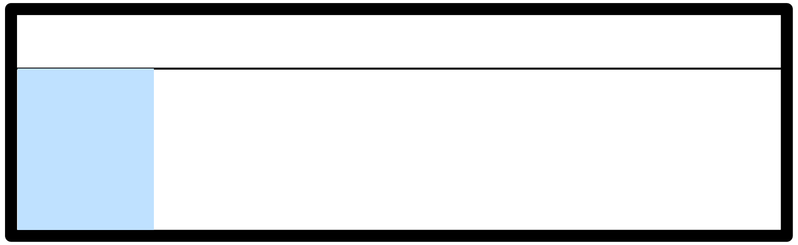
Data processing

Data transfer

Asynchronous in-situ approach

Workflow:

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



Simulaion

Data processing

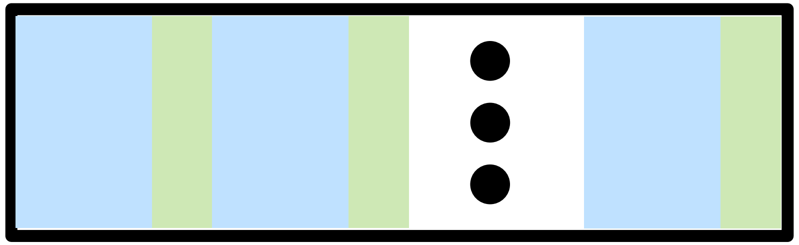
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

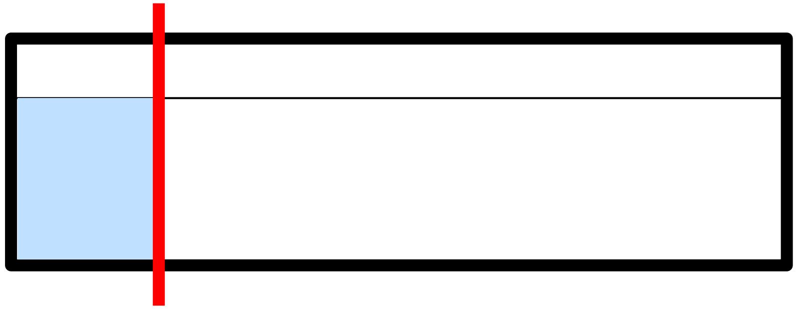
Data processing

Data transfer

Asynchronous in-situ approach

Workflow:

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



Simulaion

Data processing

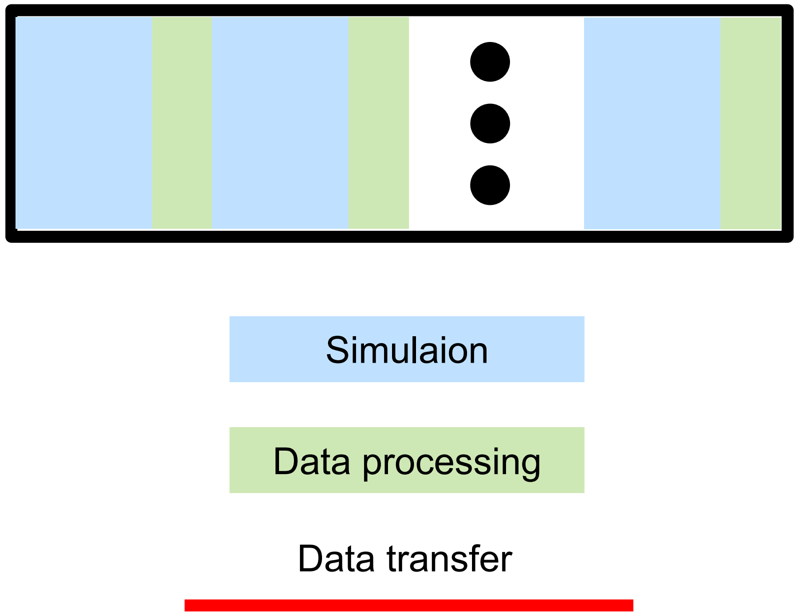
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

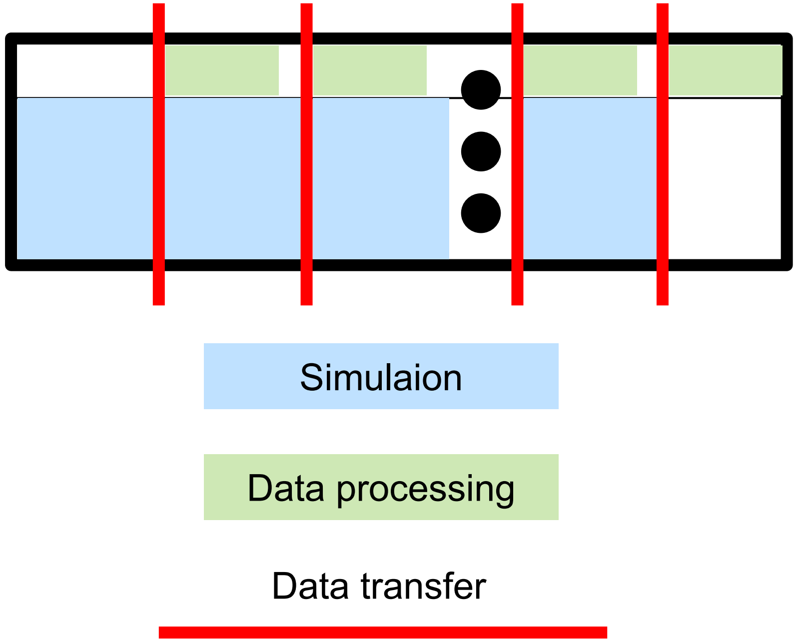
- Simulation waits until data processing finished



Asynchronous in-situ approach

Workflow:

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

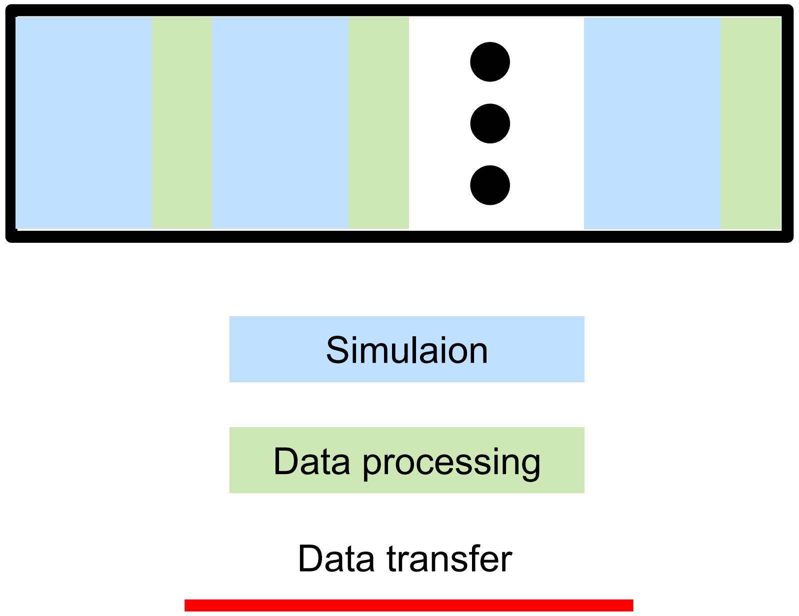


Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

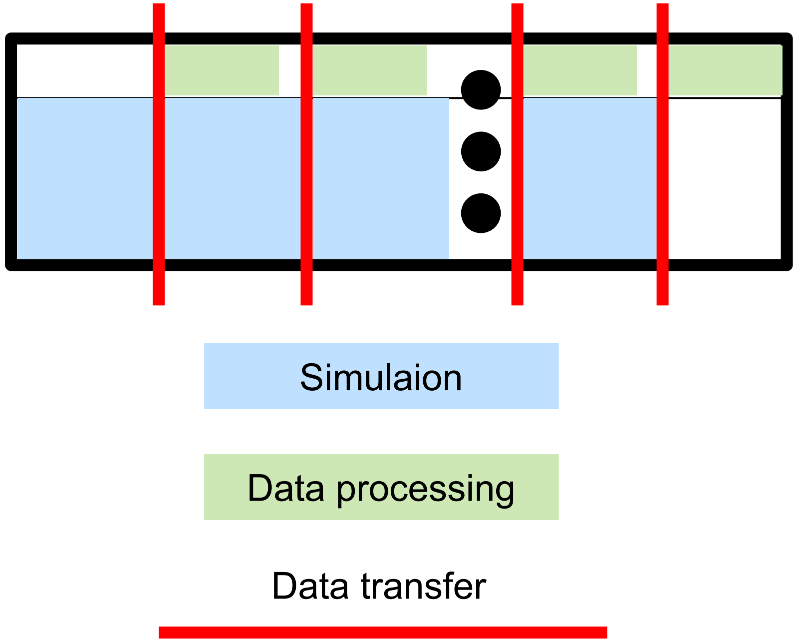
- Simulation waits until data processing finished



Asynchronous in-situ approach

Workflow:

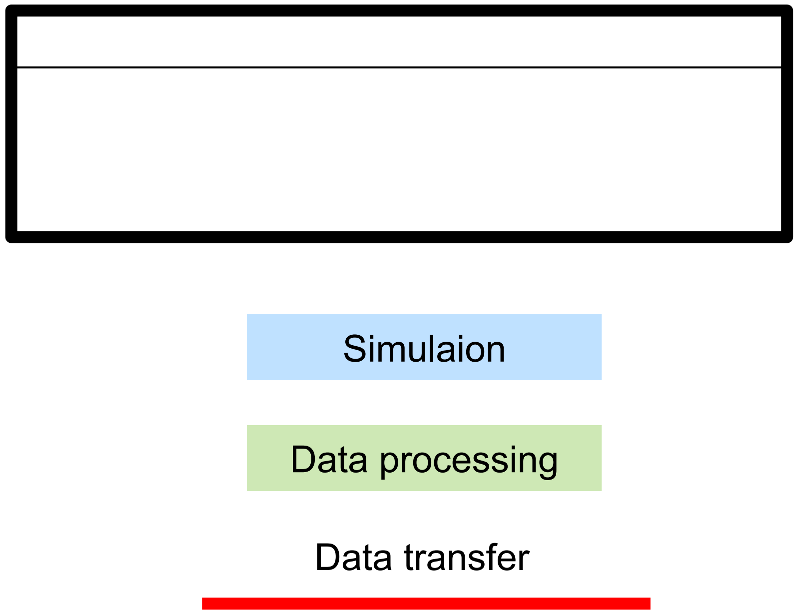
- Simulation sends data to separate computing resources and continues
- Data are processed concurrently



Hybrid in-situ approach

Workflow:

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

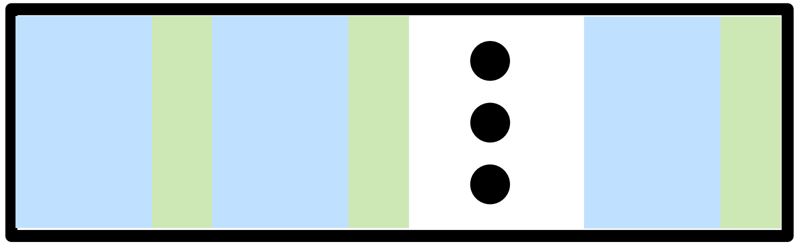


Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

- Simulation waits until data processing finished



Simulaion

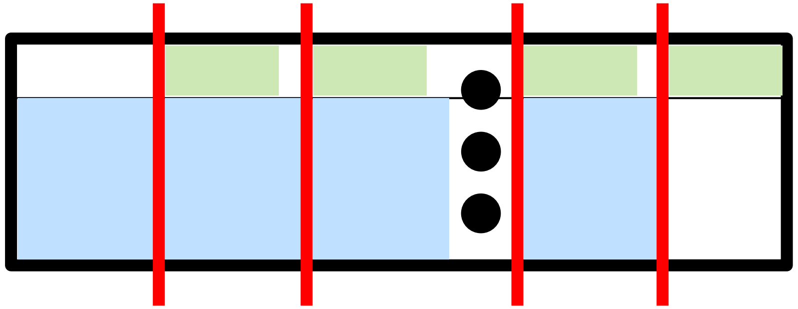
Data processing

Data transfer

Asynchronous in-situ approach

Workflow:

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



Simulaion

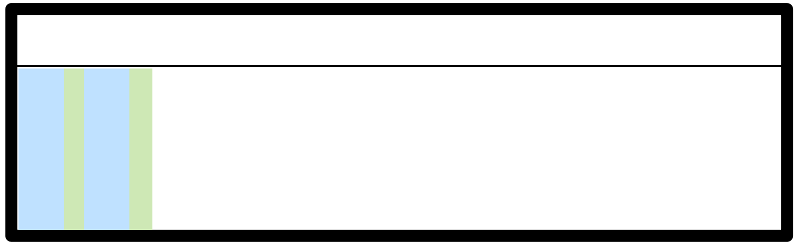
Data processing

Data transfer

Hybrid in-situ approach

Workflow:

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



Simulaion

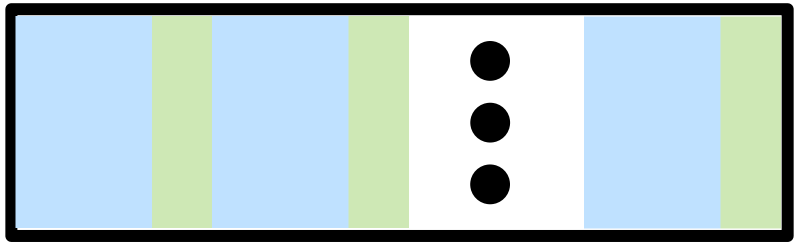
Data processing

Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:
- Simulation waits until data processing finished



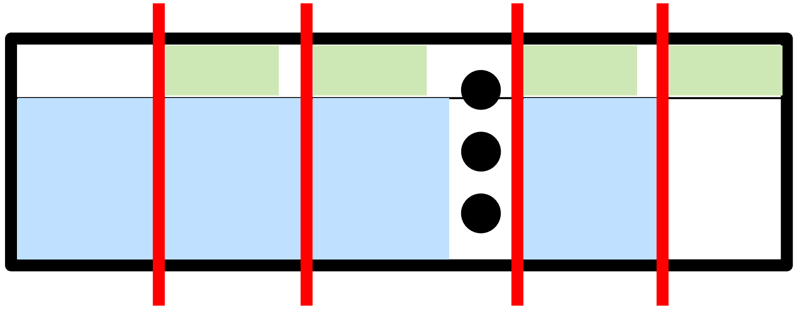
Simulaion

Data processing

Data transfer

Asynchronous in-situ approach

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



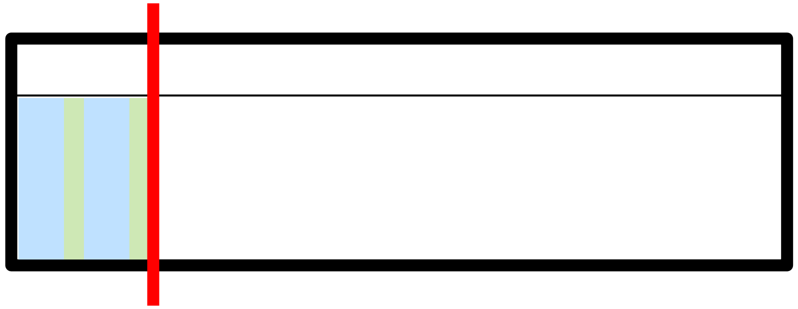
Simulaion

Data processing

Data transfer

Hybrid in-situ approach

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



Simulaion

Data processing

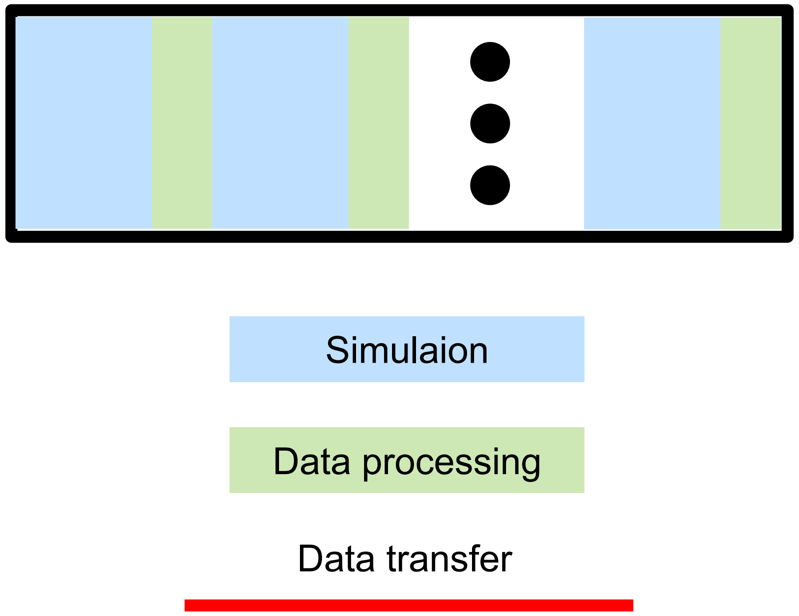
Data transfer

Synchronous, Asynchronous and Hybrid In-Situ Data Processing

Synchronous in-situ approach

Workflow:

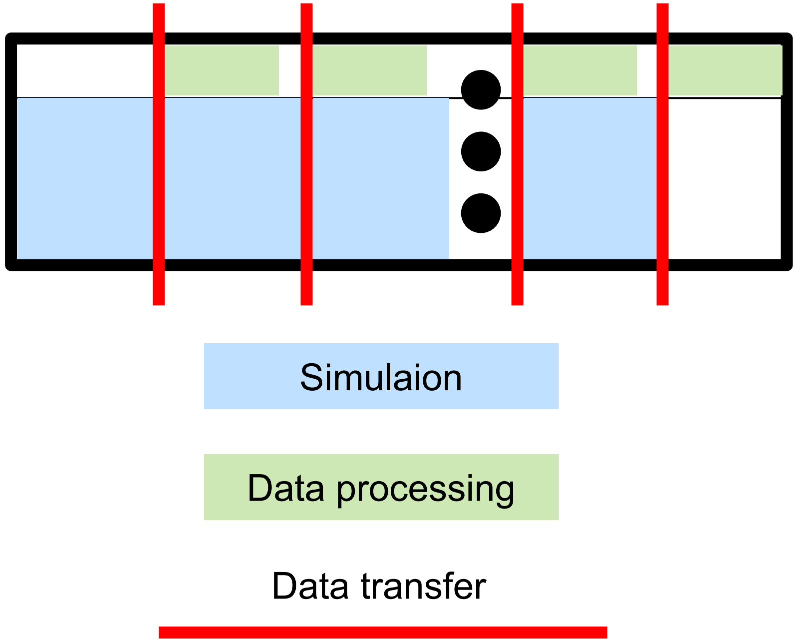
- Simulation waits until data processing finished



Asynchronous in-situ approach

Workflow:

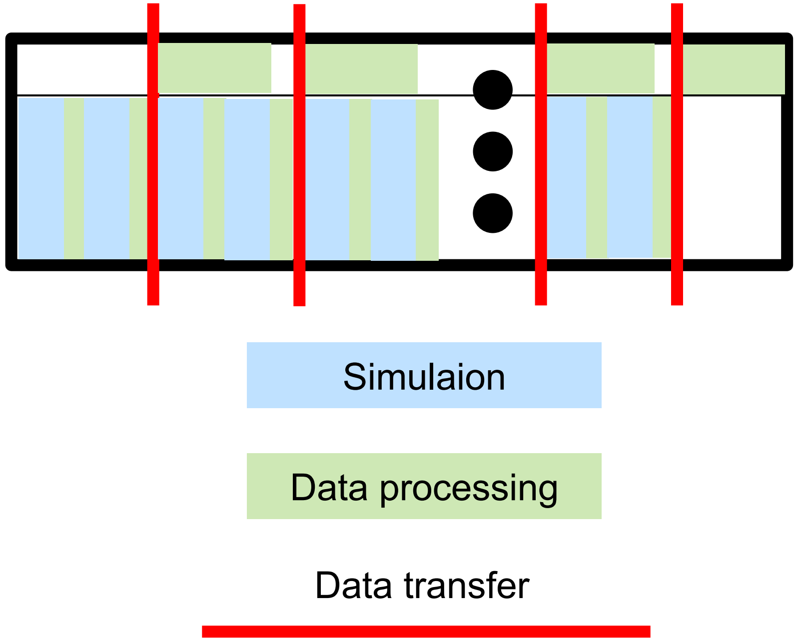
- Simulation sends data to separate computing resources and continues
- Data are processed concurrently



Hybrid in-situ approach

Workflow:

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

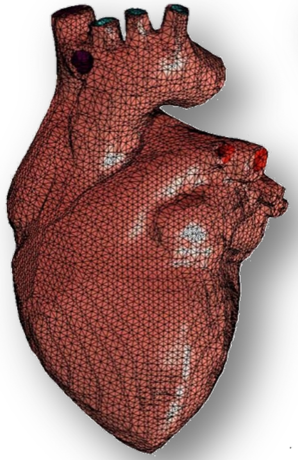


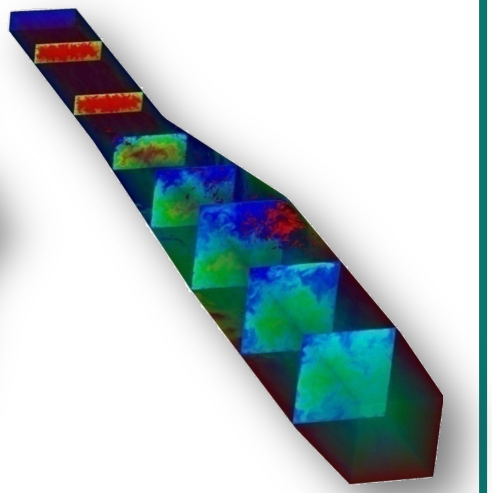
State-of-the-Art

Simulation solver

Nek5000:

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





Characteristics:

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

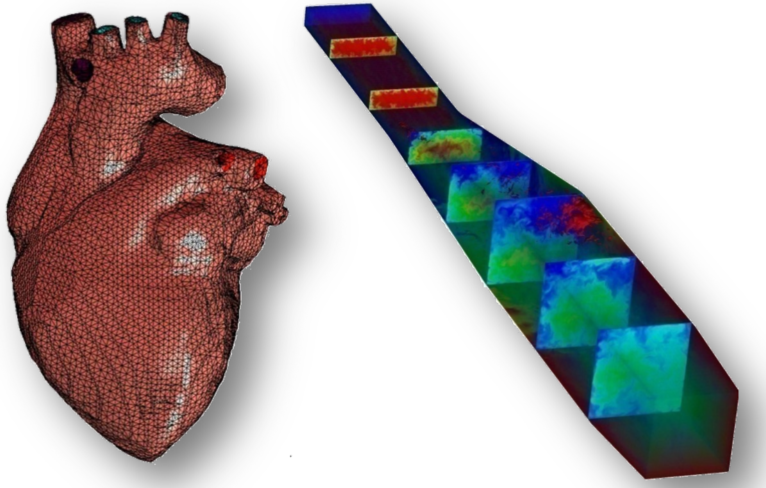
1: https://adios2.readthedocs.io/en/latest/components/components.html

State-of-the-Art

Simulation solver

Nek5000:

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



Characteristics:

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

In-situ systems

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

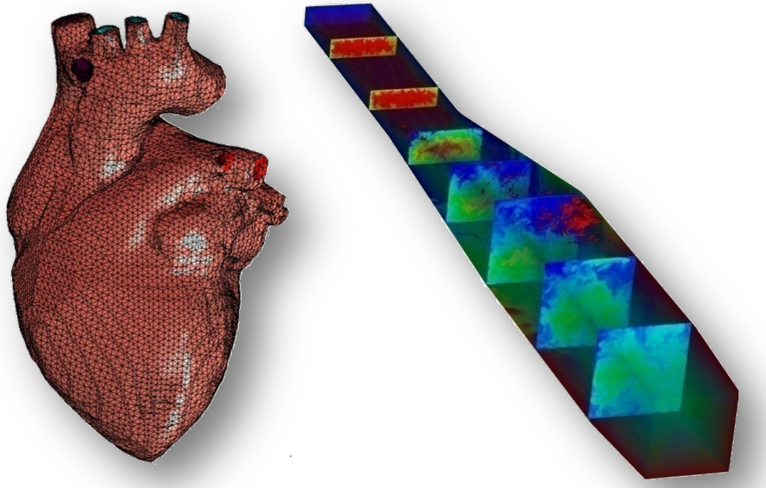
1: https://adios2.readthedocs.io/en/latest/components/components.html

State-of-the-Art

Simulation solver

Nek5000:

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



Characteristics:

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

In-situ systems

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

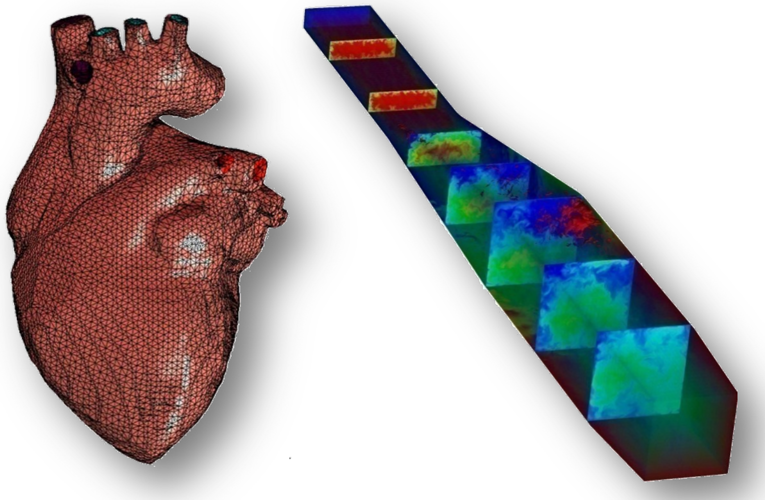
1: https://adios2.readthedocs.io/en/latest/components/components.html

State-of-the-Art

Simulation solver

Nek5000:

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



Characteristics:

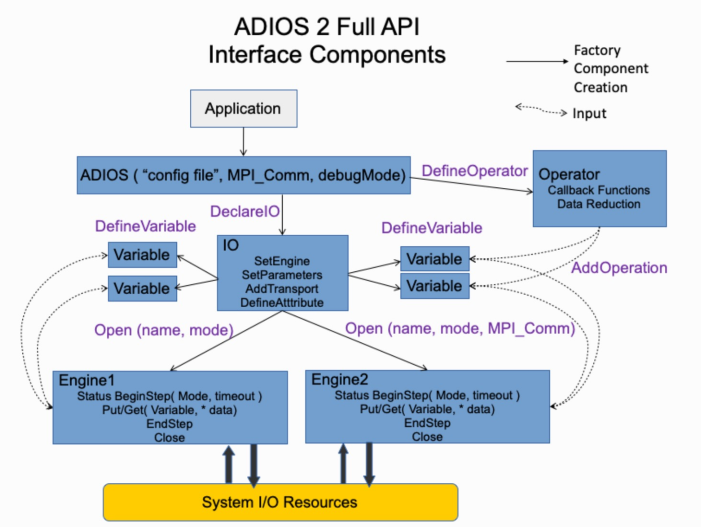
- Direct Numerical Simulation (DNS) solver
- “Matrix-free”
- Scalability from “local domain”

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]

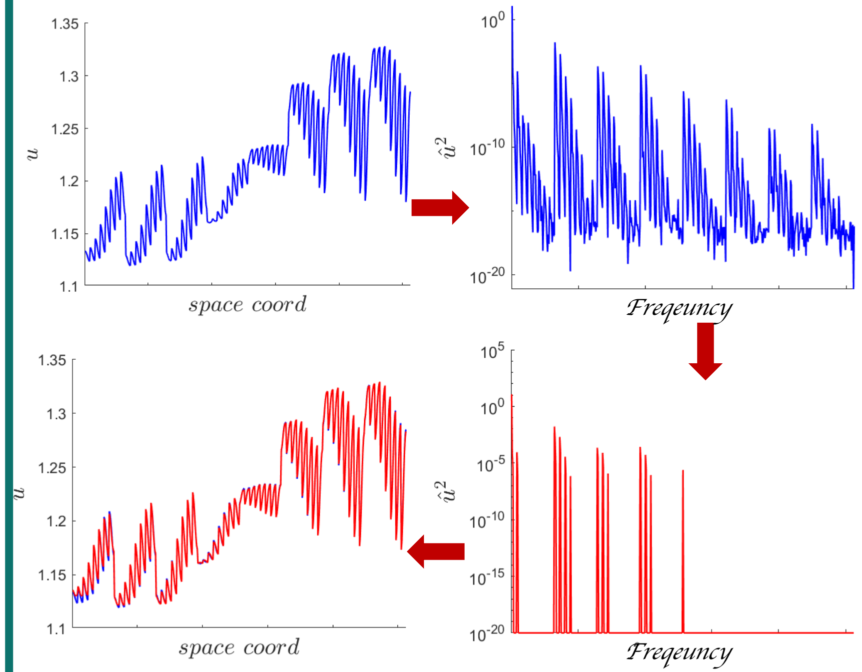


1: https://adios2.readthedocs.io/en/latest/components/components.html

Use Case: Lossy and Lossless Data Compression

Data compression

Lossy compression, physics-based method:
discard the data not associated with the most energetic flow motions [1]



Lossless compression:
ADIOS2 operator with runtime configuration

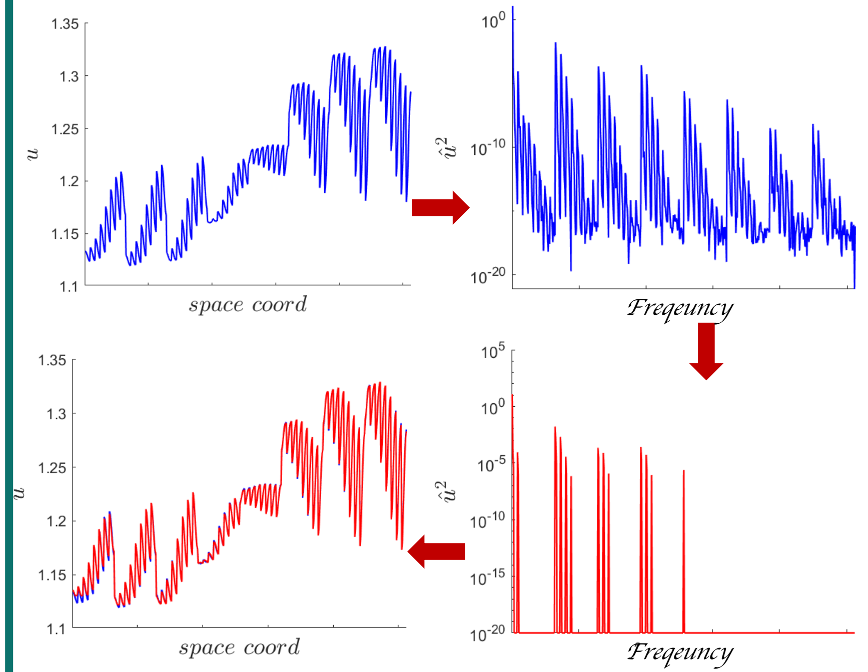
1: E. Otero et al., "Lossy data compression effects on wall-bounded turbulence: bounds on data reduction," Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

malleable data solutions for HPC

Use Case: Lossy and Lossless Data Compression

Data compression

Lossy compression, physics-based method: discard the data not associated with the most energetic flow motions [1]

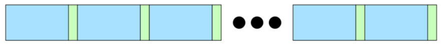


Lossless compression:

ADIOS2 operator with runtime configuration

In-situ approach

- Nek5000 with synchronous data compression:



Fortran functions

C/C++ functions called in Fortran

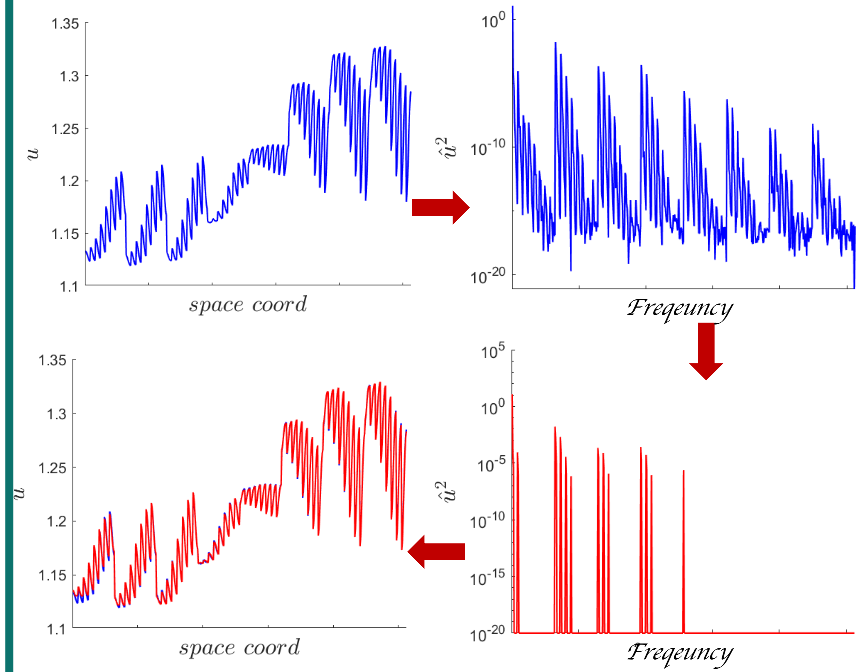
C++ functions

1: E. Otero et al., “Lossy data compression effects on wall-bounded turbulence: bounds on data reduction,” Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

Use Case: Lossy and Lossless Data Compression

Data compression

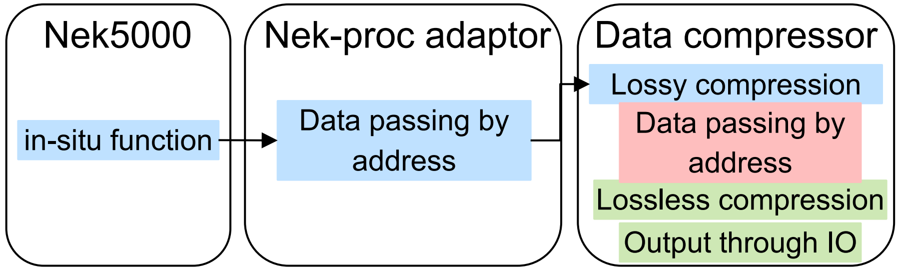
Lossy compression, physics-based method:
discard the data not associated with the most energetic flow motions [1]



Lossless compression:
ADIOS2 operator with runtime configuration

In-situ approach

- Nek5000 with synchronous data compression:



Fortran functions

C/C++ functions called in Fortran

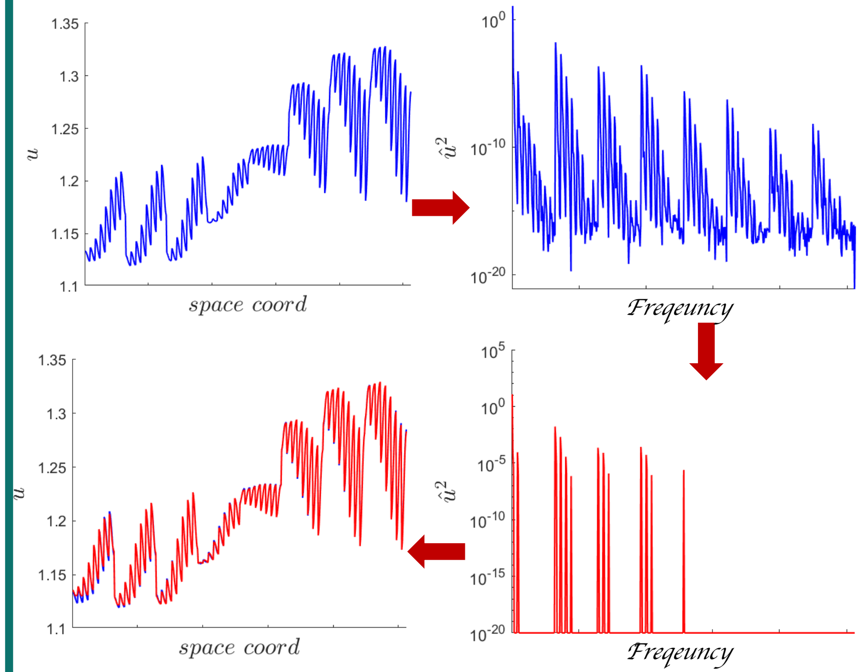
C++ functions

1: E. Otero et al., “Lossy data compression effects on wall-bounded turbulence: bounds on data reduction,” Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

Use Case: Lossy and Lossless Data Compression

Data compression

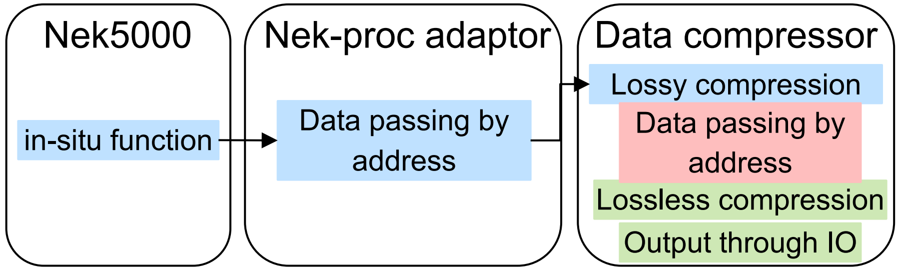
Lossy compression, physics-based method:
discard the data not associated with the most energetic flow motions [1]



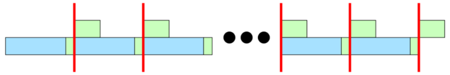
Lossless compression:
ADIOS2 operator with runtime configuration

In-situ approach

- Nek5000 with synchronous data compression:



- Nek5000 with hybrid data compression:



Fortran functions

C/C++ functions called in Fortran

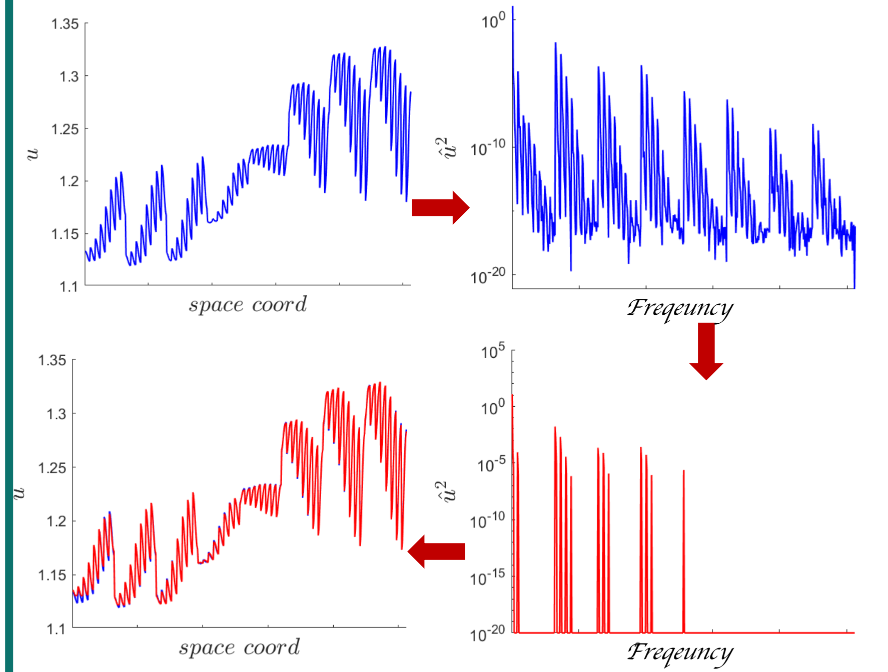
C++ functions

1: E. Otero et al., "Lossy data compression effects on wall-bounded turbulence: bounds on data reduction," Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

Use Case: Lossy and Lossless Data Compression

Data compression

Lossy compression, physics-based method: discard the data not associated with the most energetic flow motions [1]



Lossless compression:
ADIOS2 operator with runtime configuration

In-situ approach

- Nek5000 with synchronous data compression:

Nek5000: in-situ function → Nek-proc adaptor: Data passing by address → Data compressor: Lossy compression; Data passing by address; Lossless compression; Output through IO

- Nek5000 with hybrid data compression:

Nek5000: in-situ function → Nek-proc adaptor: Data selection → Data compressor: Lossy compression → Proc-wrtr adaptor: Data passing by address → ADIOS writer: ADIOS *insituMPI* writer

ADIOS reader: ADIOS *insituMPI* reader → Rdr-proc adaptor: Data passing by address → Data compressor: Lossless compression; Output through IO

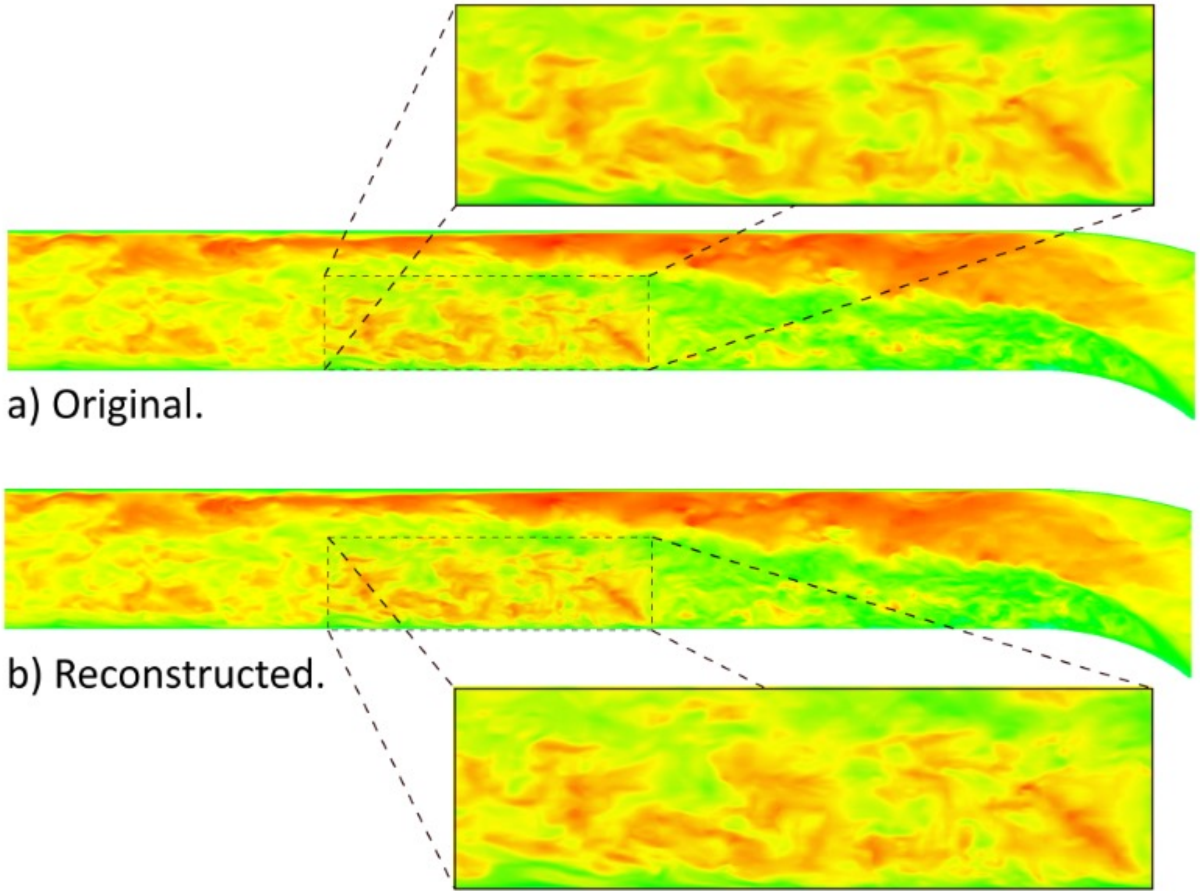
Fortran functions | C/C++ functions called in Fortran | C++ functions

1: E. Otero et al., “Lossy data compression effects on wall-bounded turbulence: bounds on data reduction,” Flow, Turbulence and Combustion, vol. 101, no. 2, pp. 365– 387, 2018.

CPU-based Nek5000 with Lossy and Lossless Data Compression

(with maximum allowed error $\varepsilon = 10^{-2}$ and compression ratio $c = 98\%$)

1

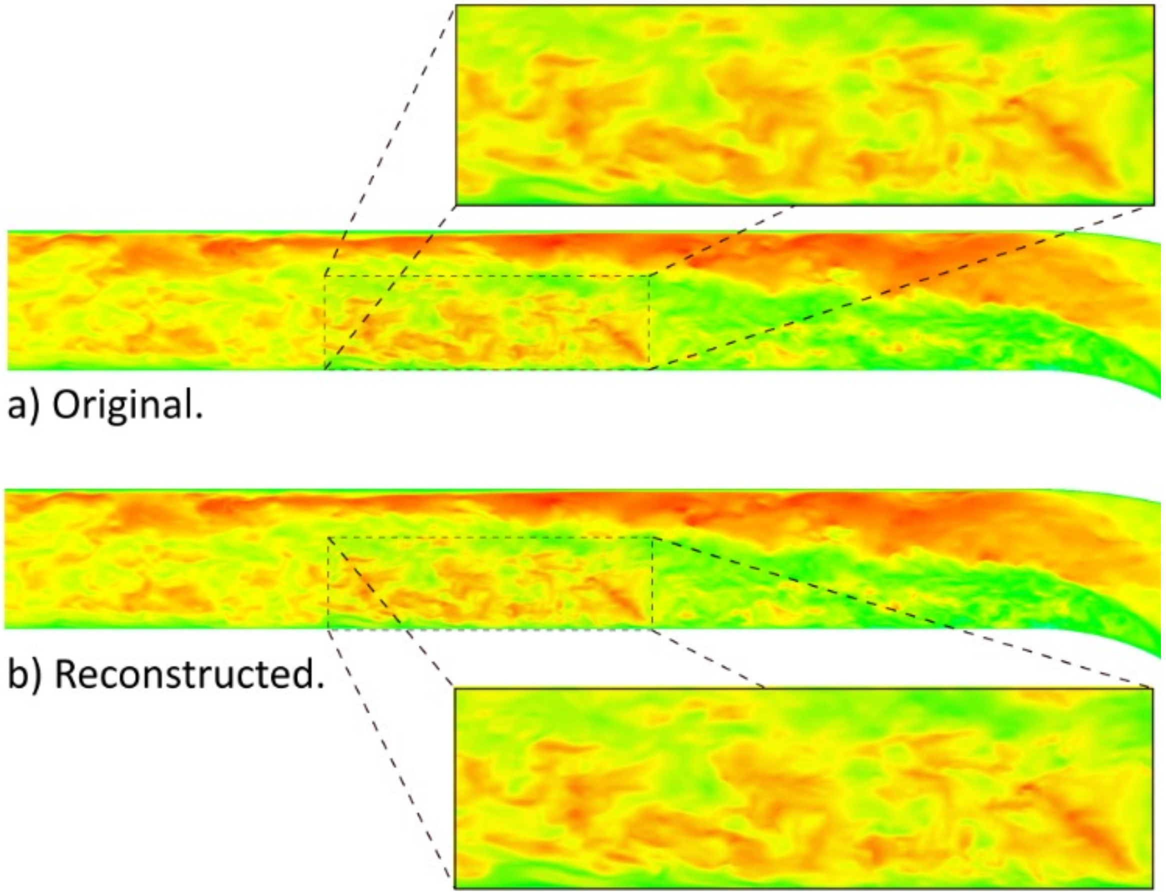


1: Slice of the velocity magnitude downstream from the bent section. a) is the original data set, while b) is the reconstruction of a field compressed with a maximum allowed error of 10^{-2}. 2: RMSE of a slice of the 3D field for a maximum allowed error of 10^{-2}. The error is shown per spectral element.

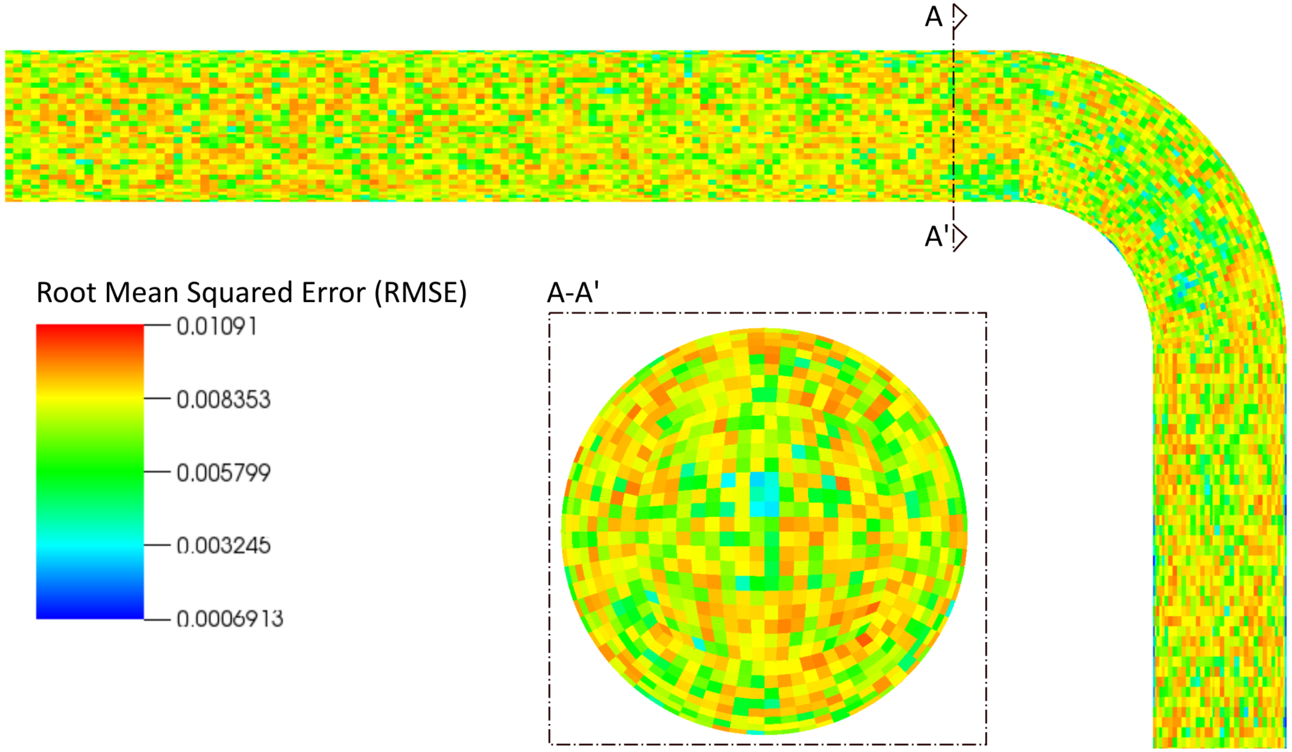
CPU-based Nek5000 with Lossy and Lossless Data Compression

(with maximum allowed error $\varepsilon = 10^{-2}$ and compression ratio $c = 98\%$)

1



2

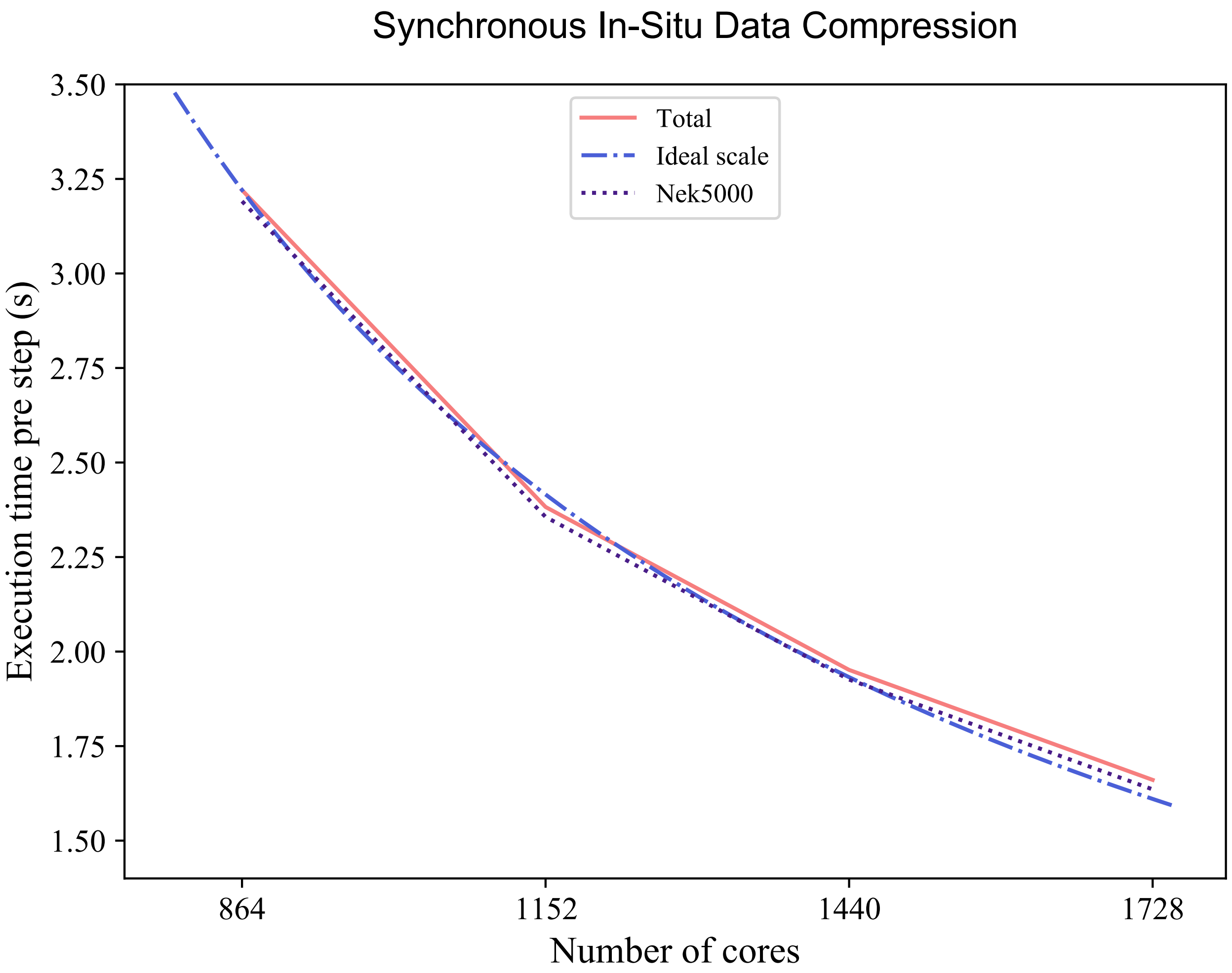


1: Slice of the velocity magnitude downstream from the bent section. a) is the original data set, while b) is the reconstruction of a field compressed with a maximum allowed error of 10^{-2}. 2: RMSE of a slice of the 3D field for a maximum allowed error of 10^{-2}. The error is shown per spectral element.

CPU-based Nek5000 with Synchronous and Hybrid In-Situ Data Compression

(with maximum allowed error $\varepsilon = 10^{-2}$ and compression ratio $c = 98\%$)

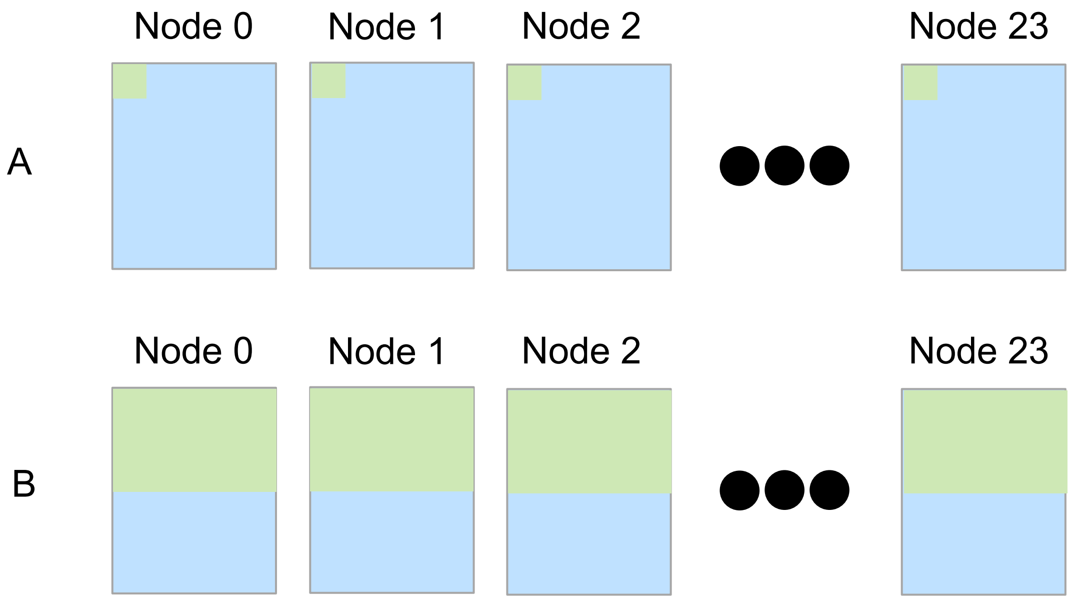
1



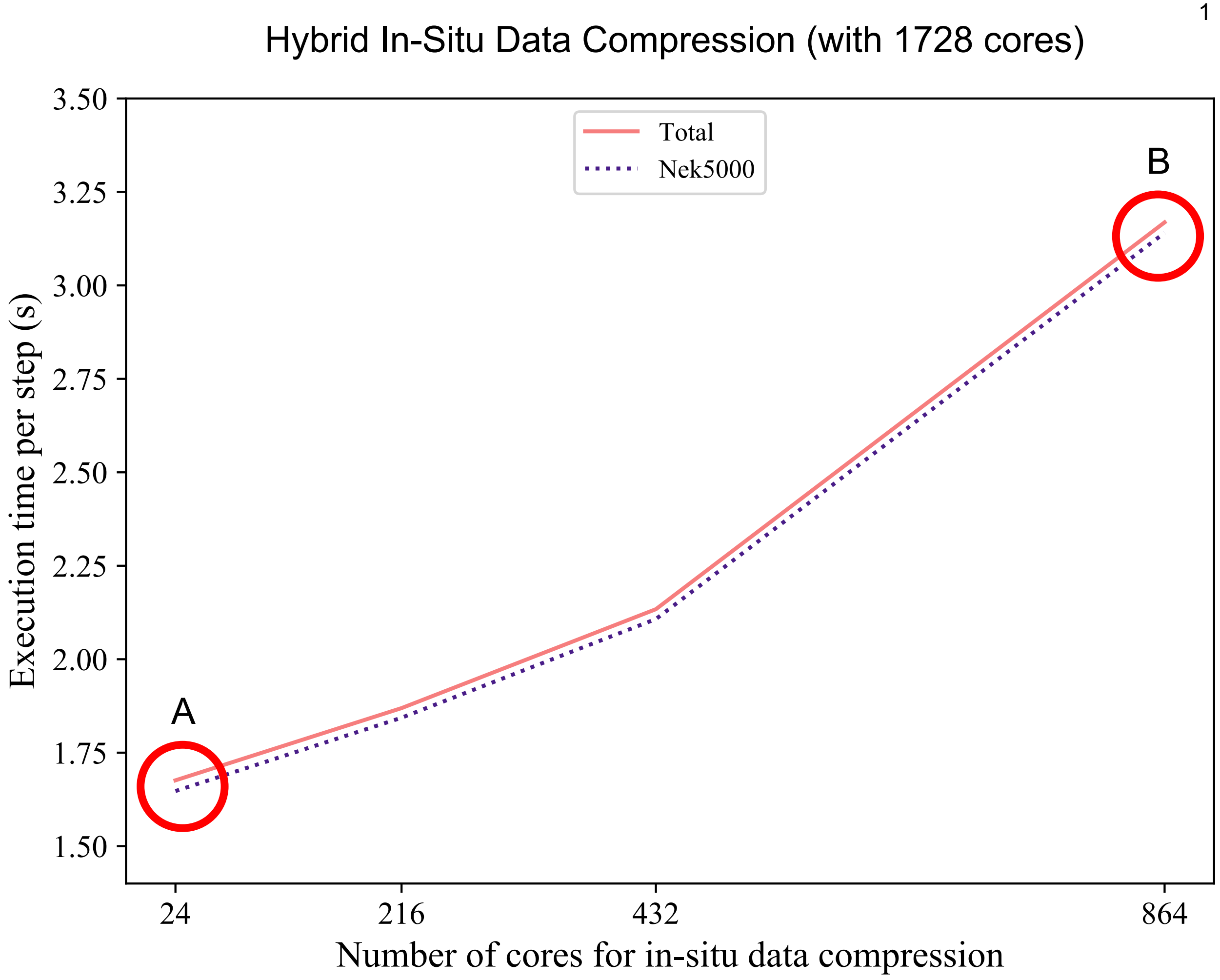
1: execution time of Nek5000 with synchronous in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on Raven supercomputer (left) and hybrid in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on 24 Raven nodes (right).

CPU-based Nek5000 with Synchronous and Hybrid In-Situ Data Compression

(with maximum allowed error $\varepsilon = 10^{-2}$ and compression ratio $c = 98\%$)



1



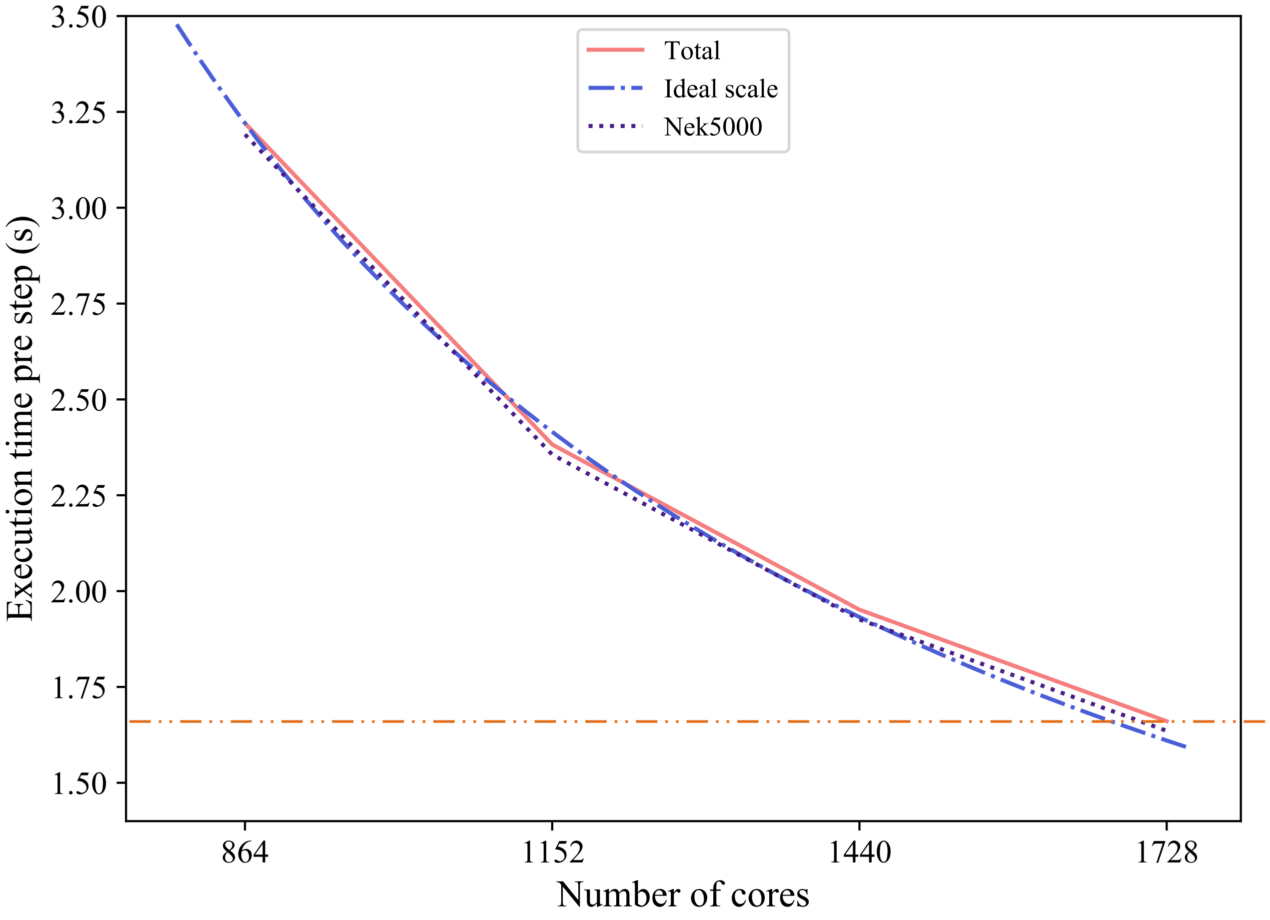
1: execution time of Nek5000 with synchronous in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on Raven supercomputer (left) and hybrid in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on 24 Raven nodes (right).

CPU-based Nek5000 with Synchronous and Hybrid In-Situ Data Compression

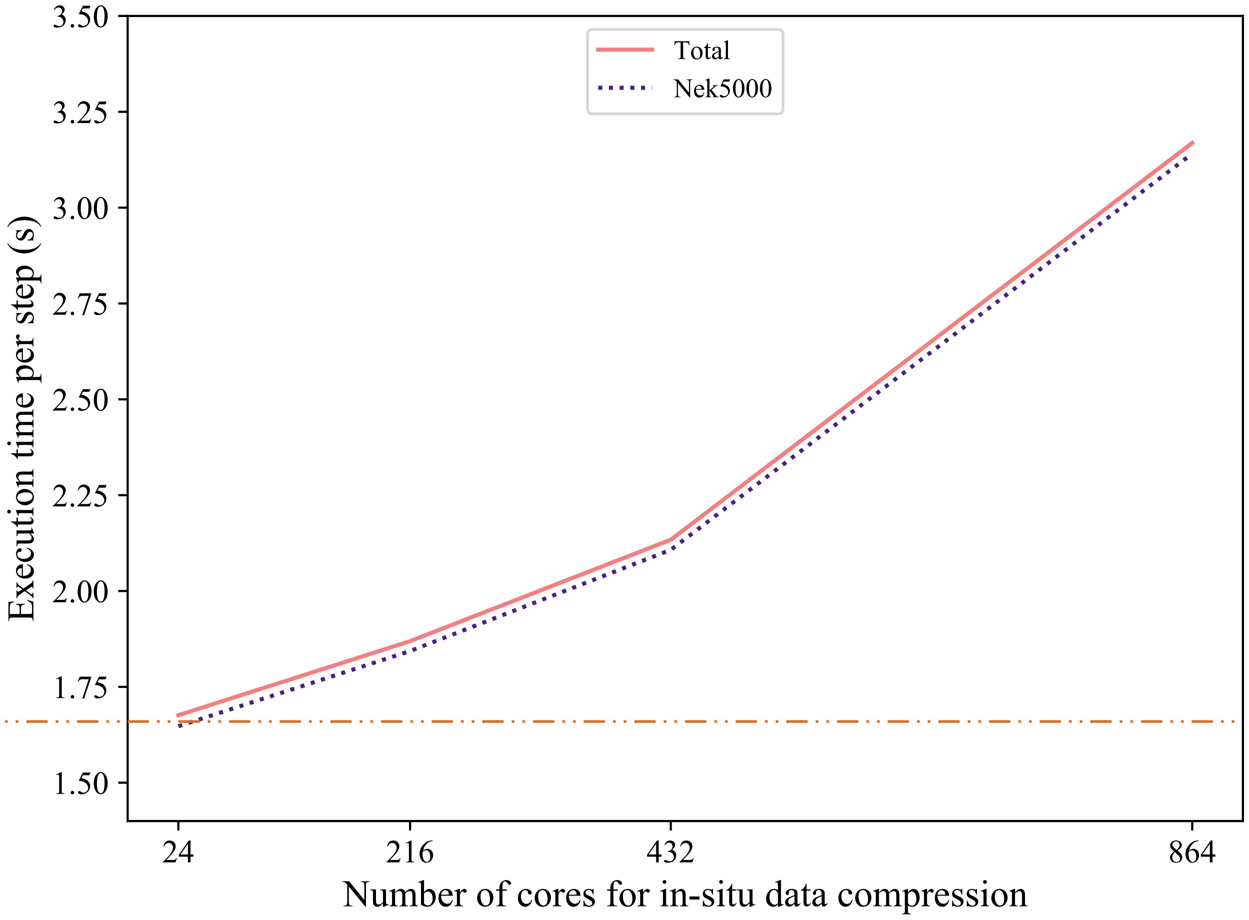
(with maximum allowed error $\varepsilon = 10^{-2}$ and compression ratio $c = 98\%$)

[1]

Synchronous In-Situ Data Compression

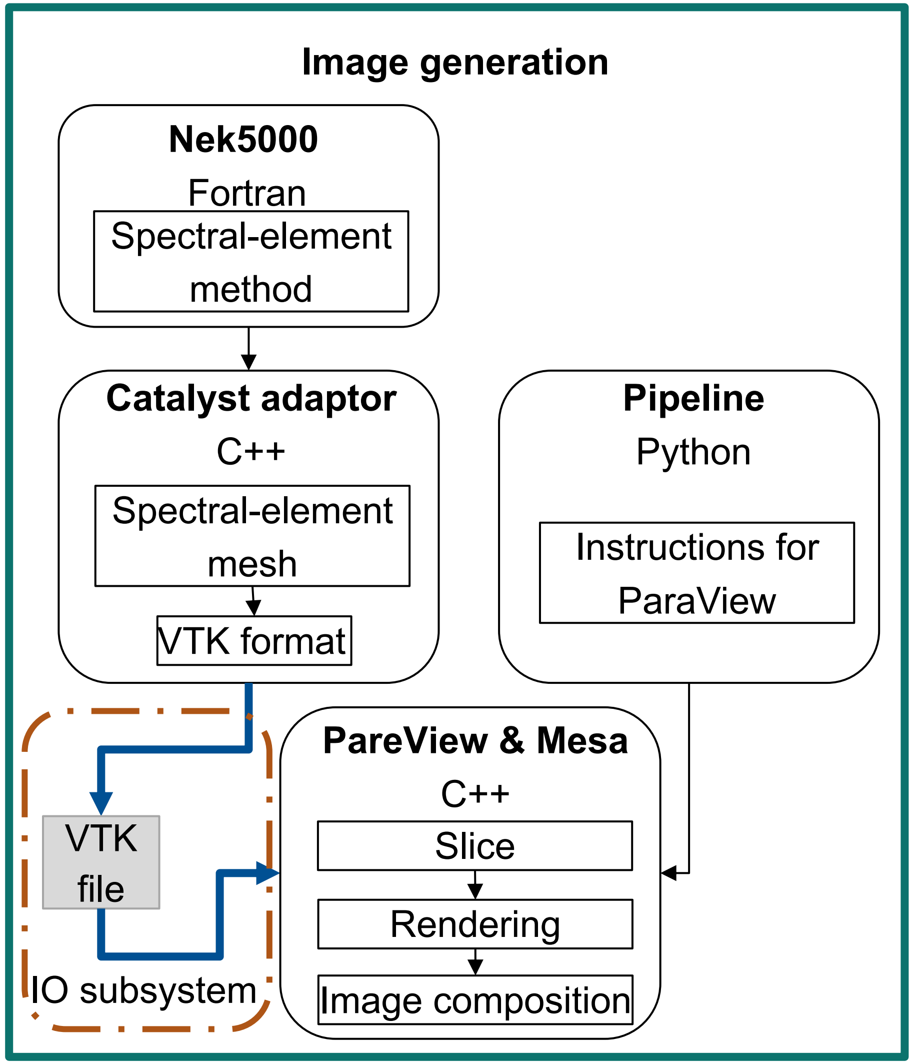


Hybrid In-Situ Data Compression (with 1728 cores)



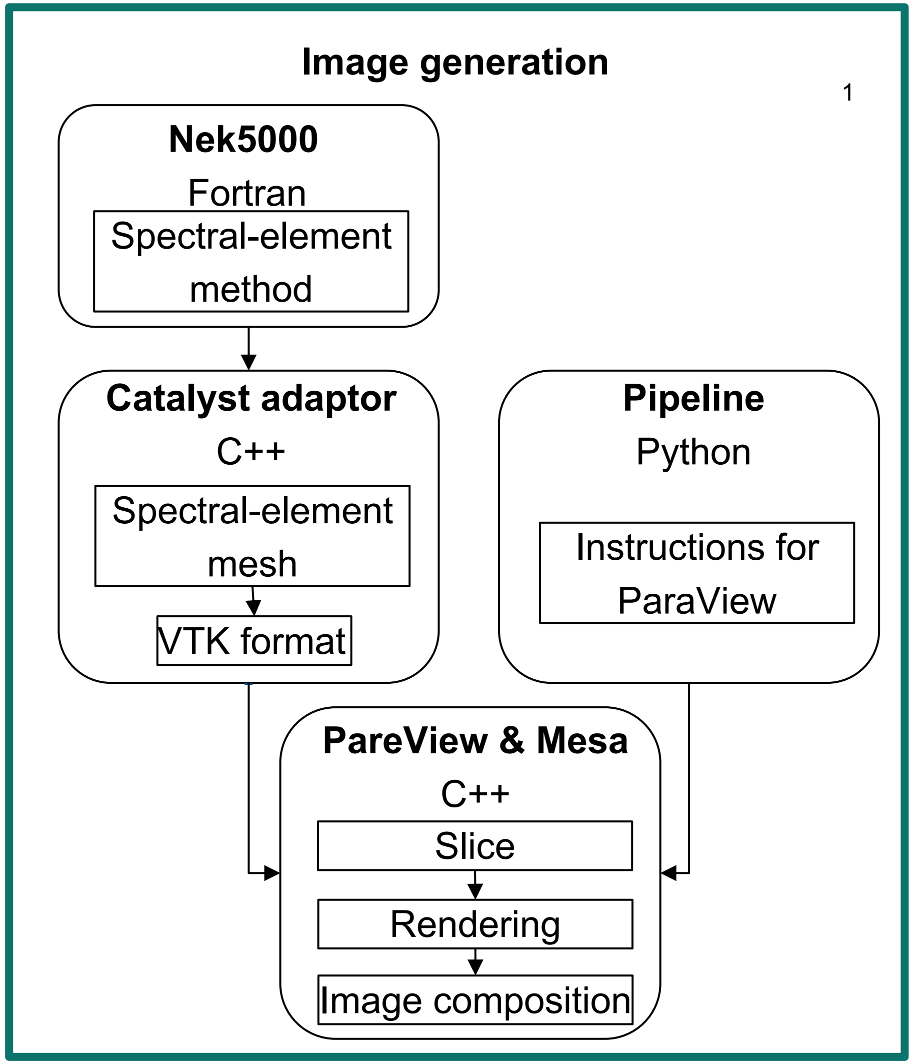
1: execution time of Nek5000 with synchronous in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on Raven supercomputer (left) and hybrid in-situ compression with lossy compression maximum allowed error ε = 10^{-2} on 24 Raven nodes (right).

Use Case: Image Generation with ParaView/Catalyst



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."

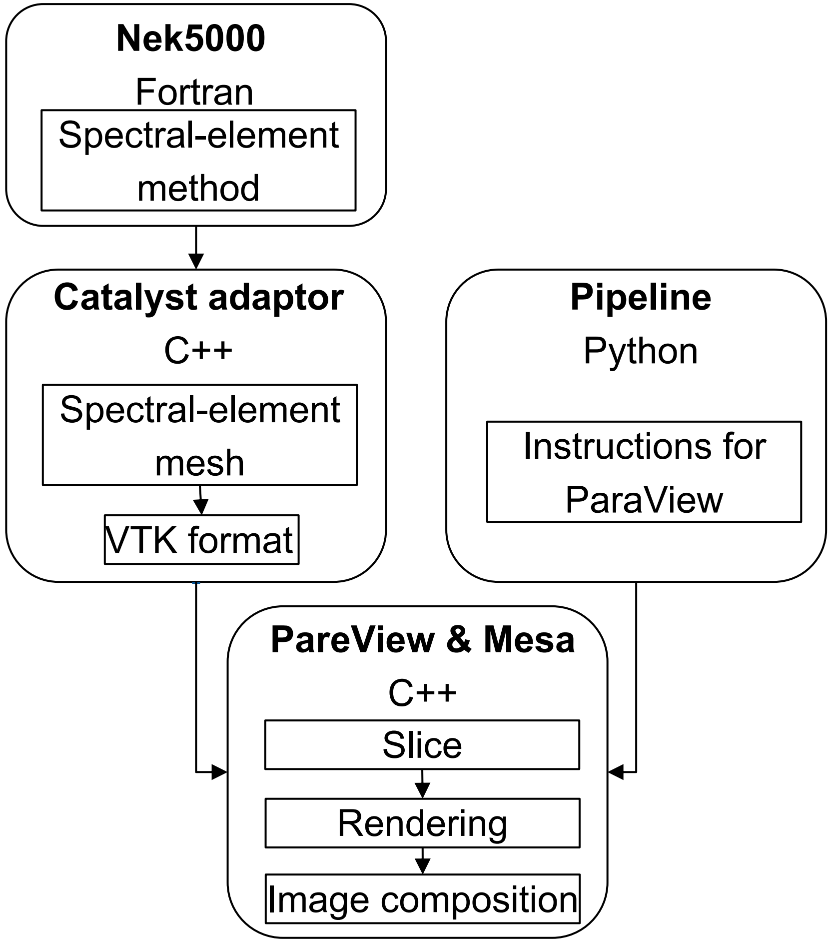
Use Case: Image Generation with ParaView/Catalyst



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."

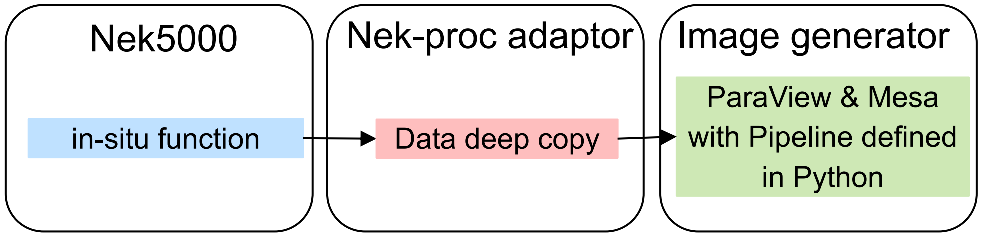
Use Case: Image Generation with ParaView/Catalyst

Image generation [1]



In-situ approach

- Nek5000 with synchronous image generation:



Fortran functions

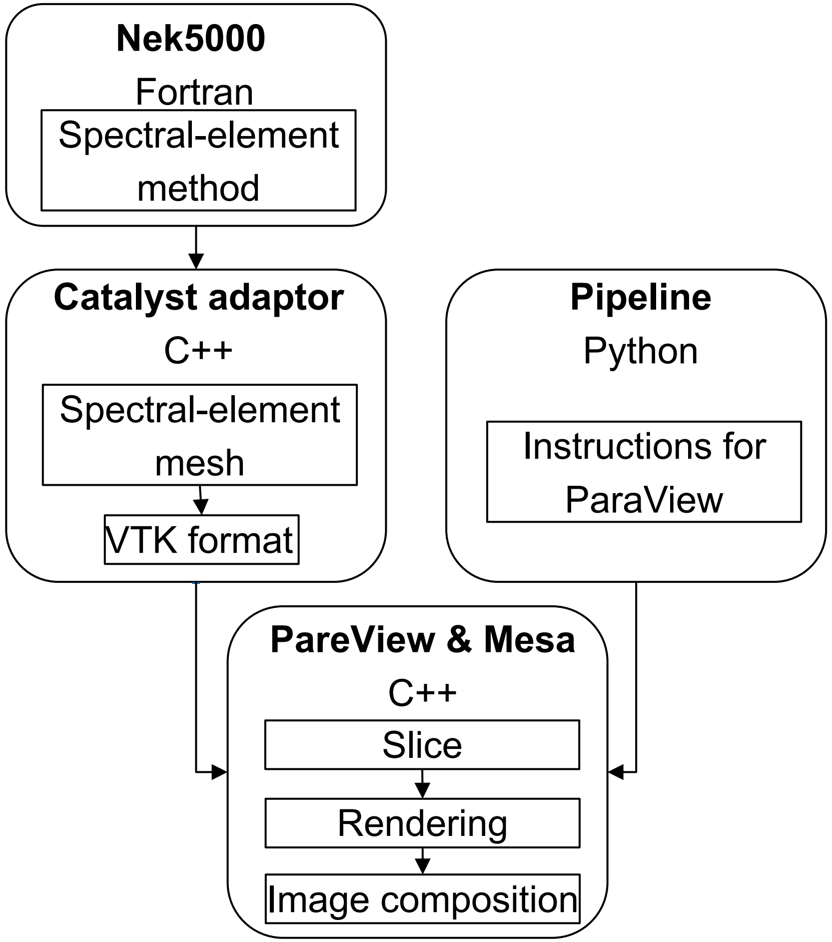
C/C++ functions called in Fortran

C++ functions

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."

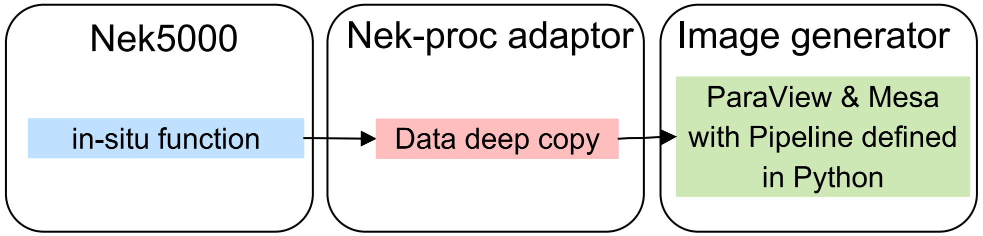
Use Case: Image Generation with ParaView/Catalyst

Image generation [1]

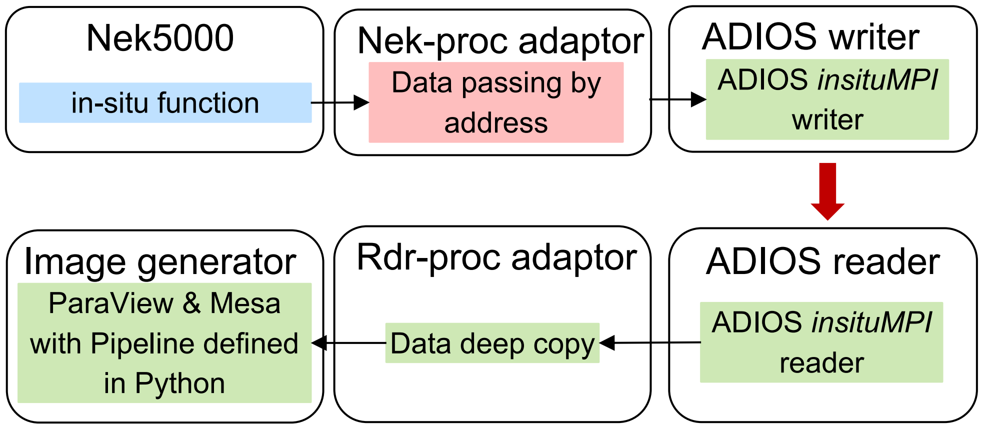


In-situ approach

- Nek5000 with synchronous image generation:



- Nek5000 with asynchronous image generation:



Fortran functions

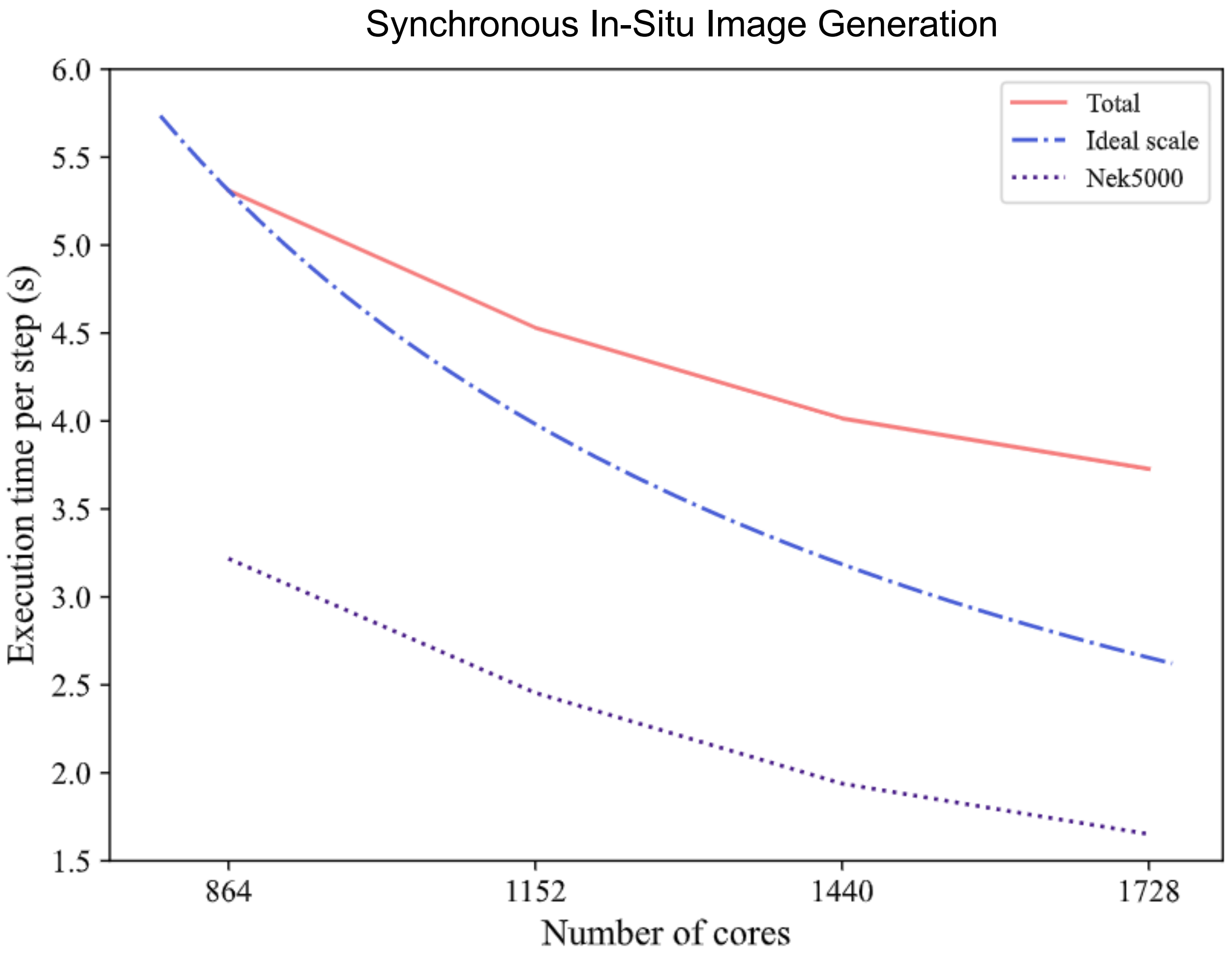
C/C++ functions called in Fortran

C++ functions

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."

CPU-based Nek5000 with Synchronous and Asynchronous Image Generation

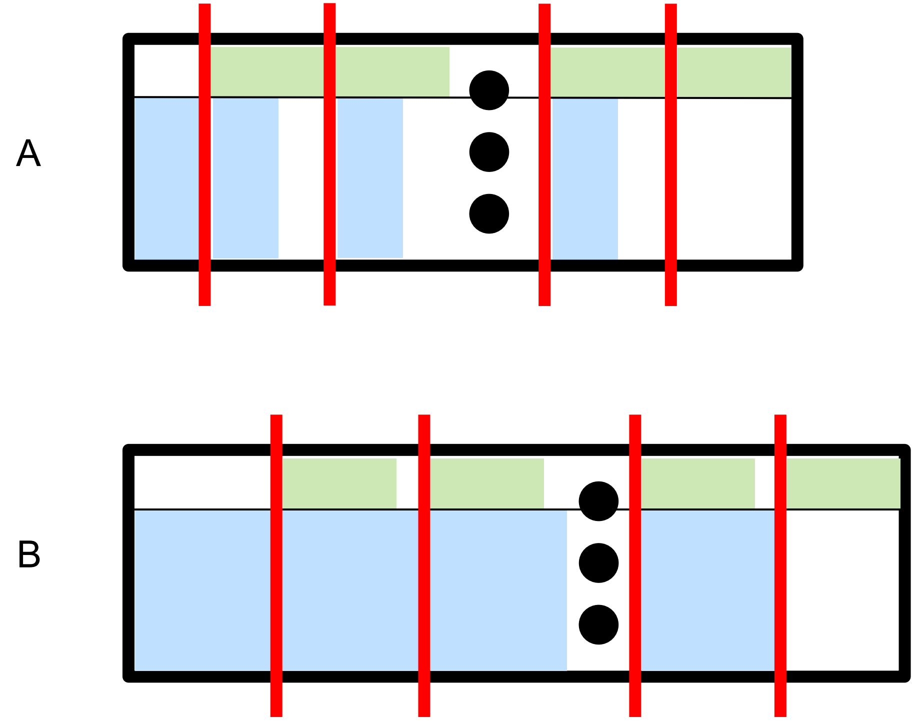
(45G VTK file for one image avoided) 1



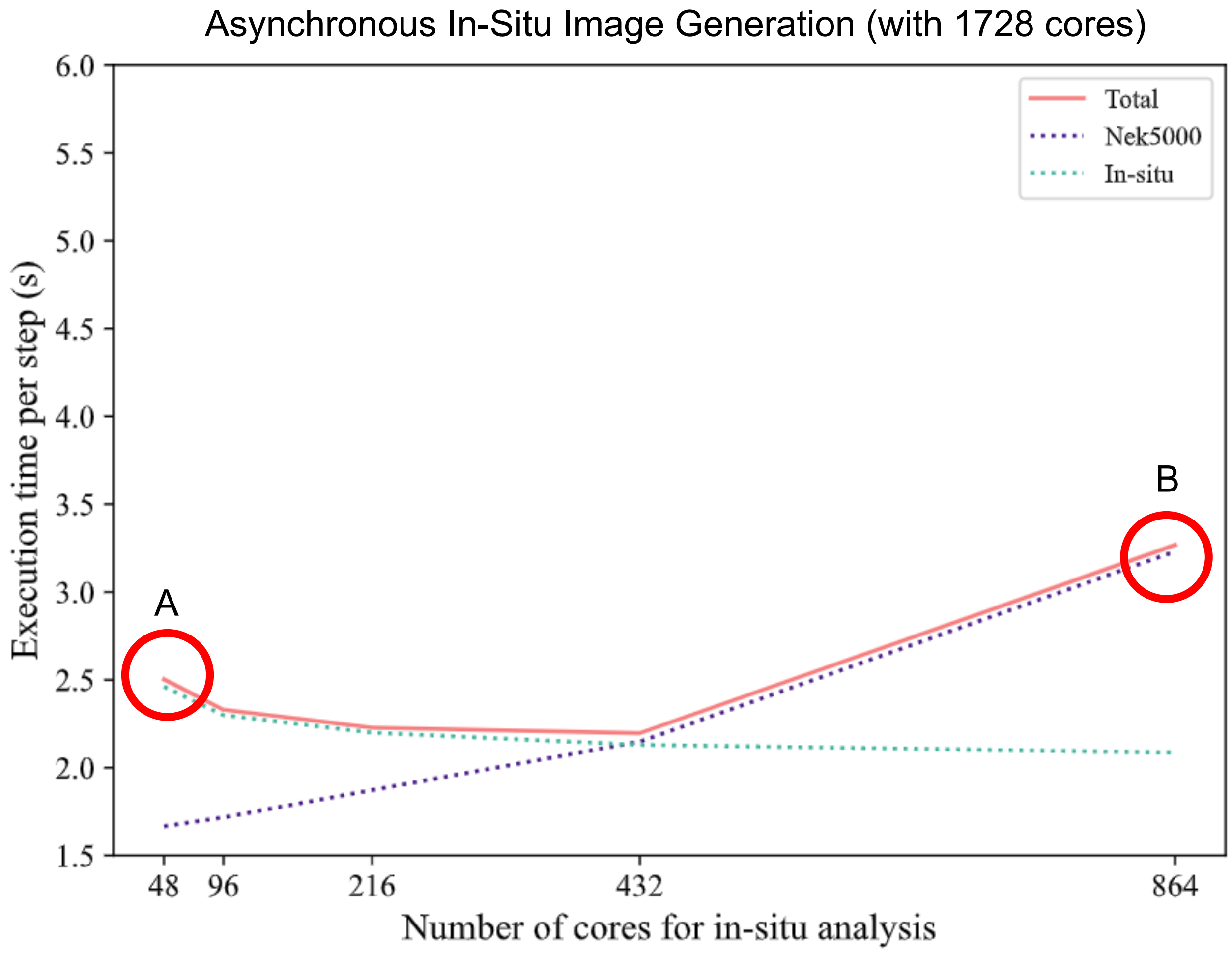
1: Execution time of Nek5000 with synchronous in-situ image generation every two steps on Raven supercomputer (left) and asynchronous in-situ image generation every two steps on 24 Raven nodes (right).

CPU-based Nek5000 with Synchronous and Asynchronous Image Generation

(45G VTK file for one image avoided)



[1]



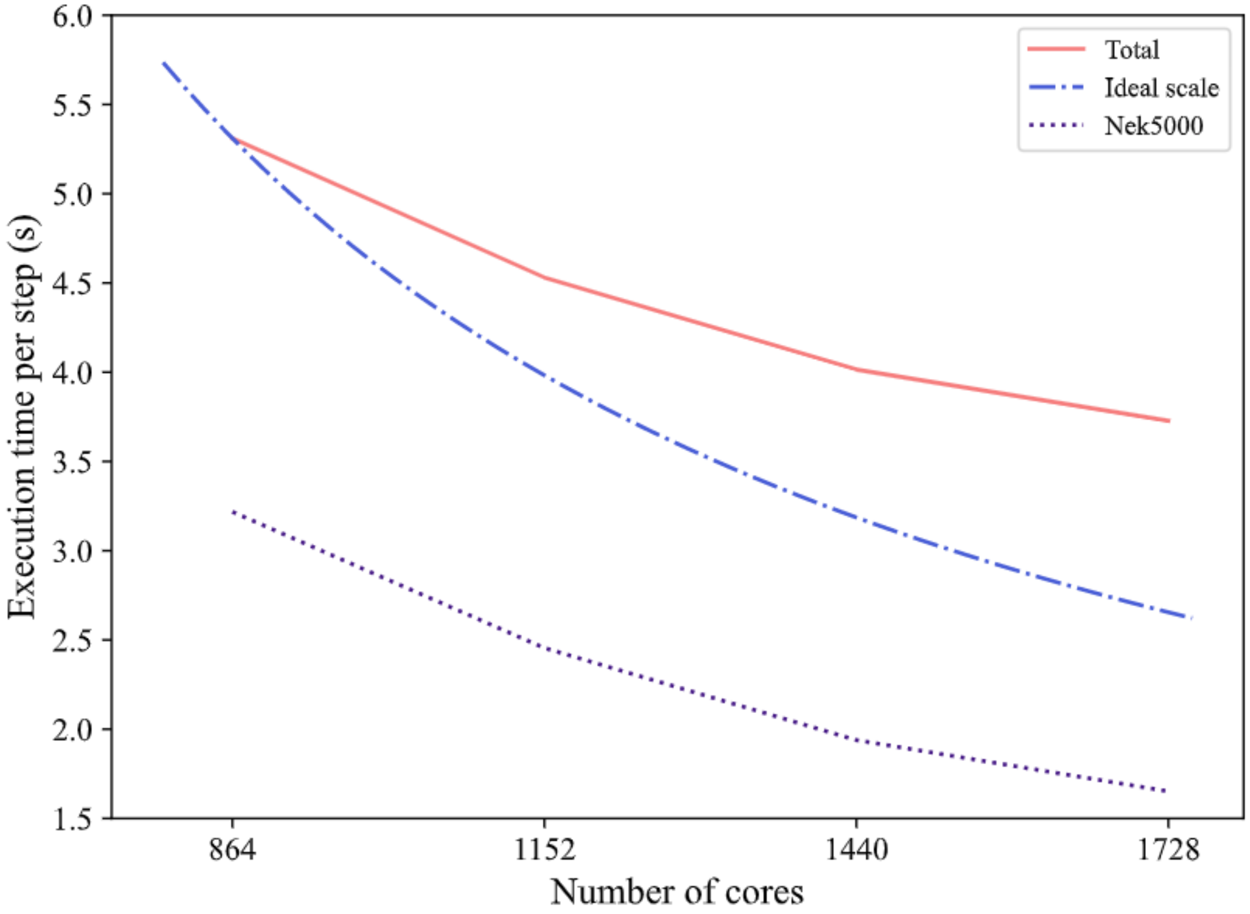
1: Execution time of Nek5000 with synchronous in-situ image generation every two steps on Raven supercomputer (left) and asynchronous in-situ image generation every two steps on 24 Raven nodes (right).

CPU-based Nek5000 with Synchronous and Asynchronous Image Generation

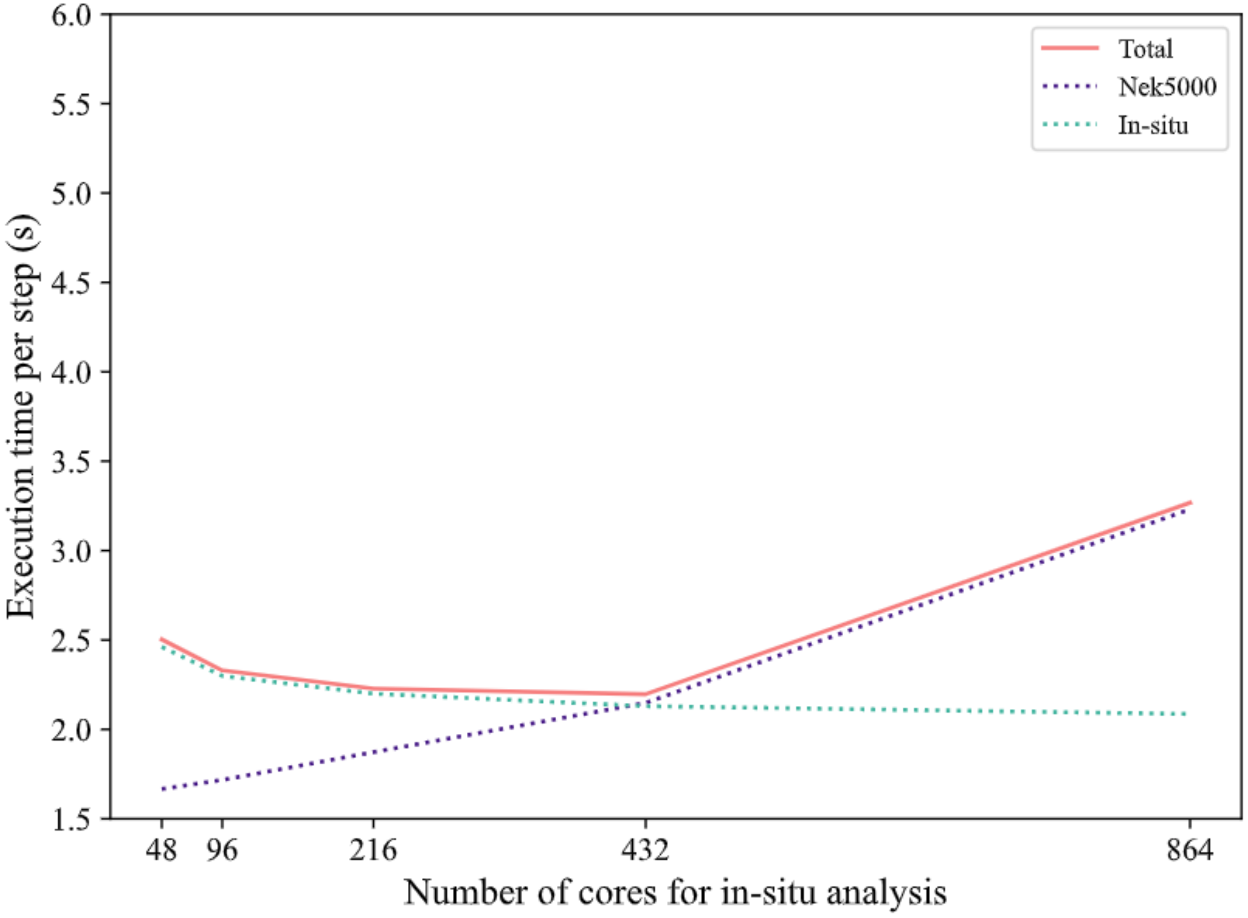
(45G VTK file for one image avoided)

1

Synchronous In-Situ Image Generation



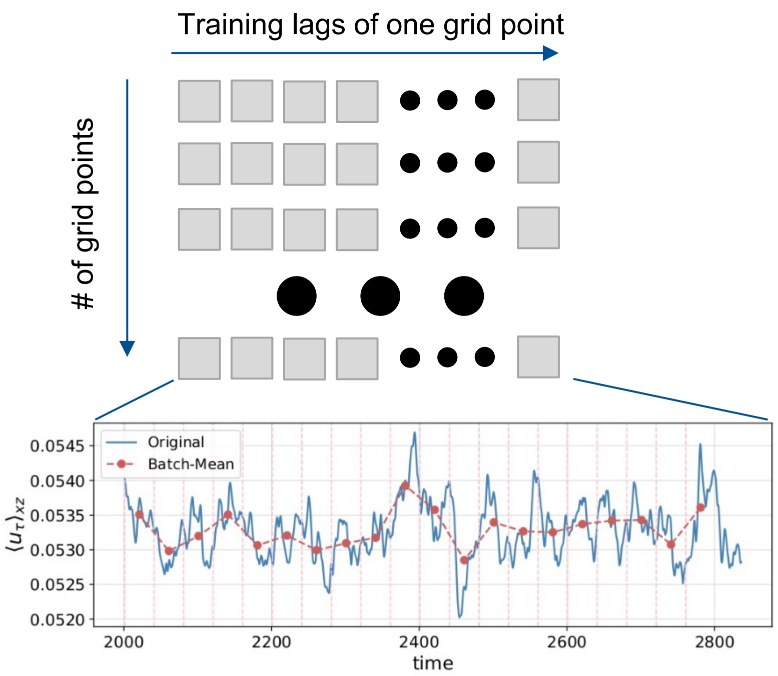
Asynchronous In-Situ Image Generation (with 1728 cores)



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).

Use Case: Uncertainty Quantification

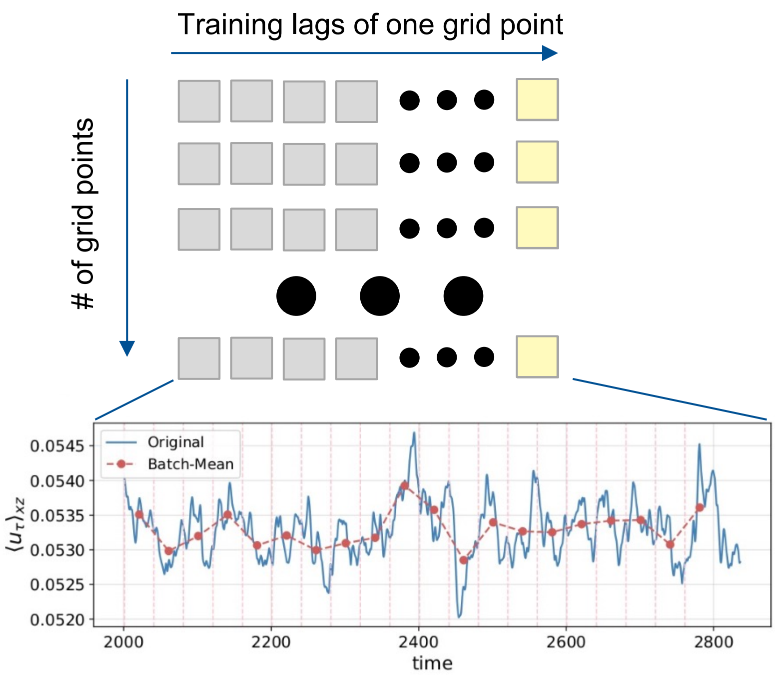
Uncertainty Quantification



malleable data solutions for HPC

Use Case: Uncertainty Quantification

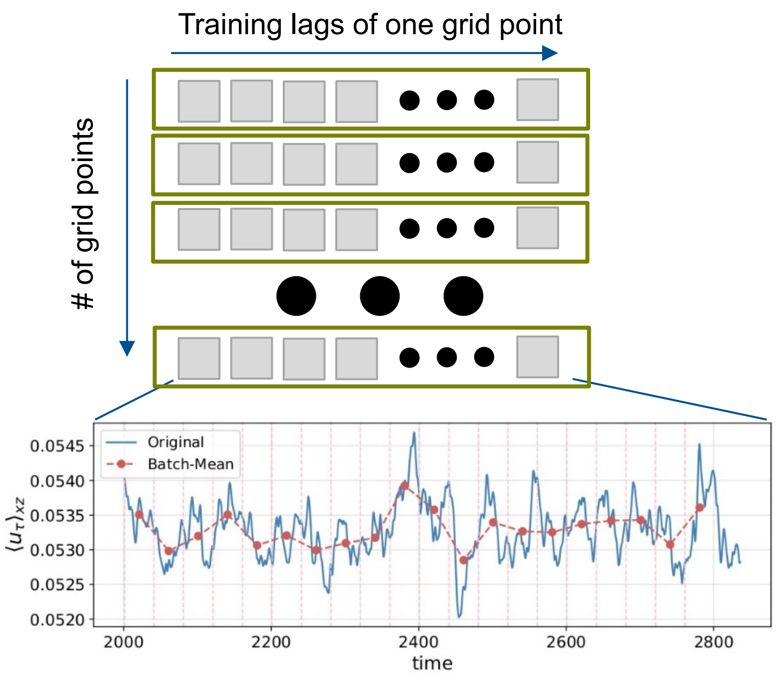
Uncertainty Quantification



Frequent training lag update

Use Case: Uncertainty Quantification

Uncertainty Quantification

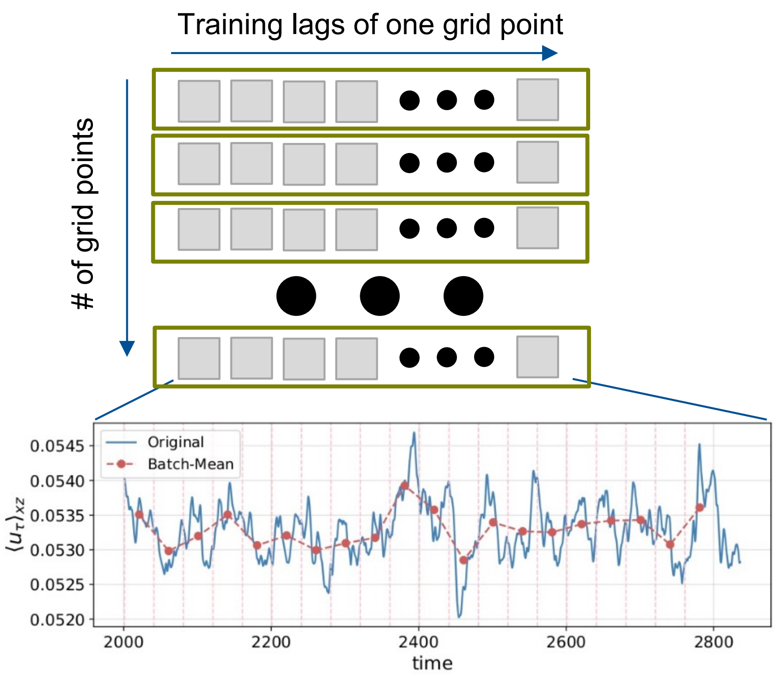


Frequent training lag update

Expensive model and uncertainty update

Use Case: Uncertainty Quantification

Uncertainty Quantification

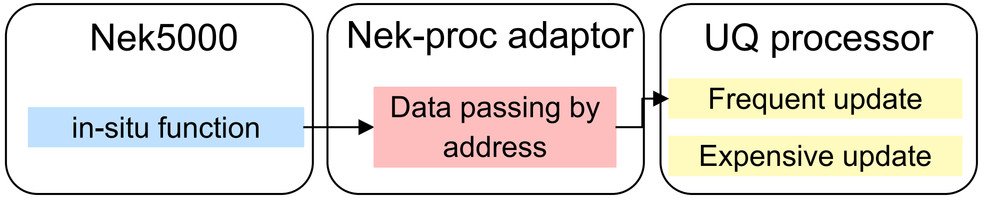


Frequent training lag update

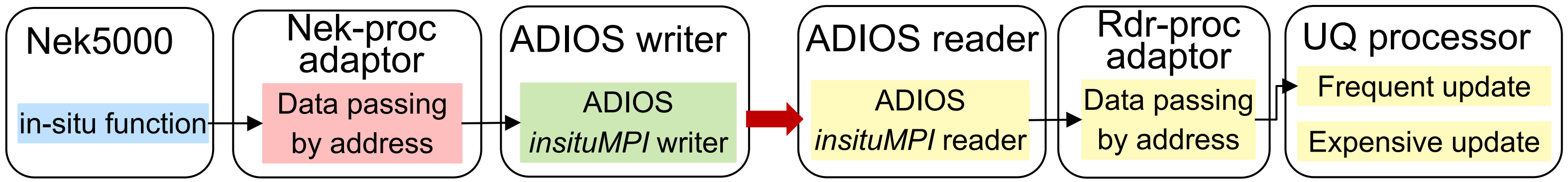
Expensive model and uncertainty update

In-situ approach

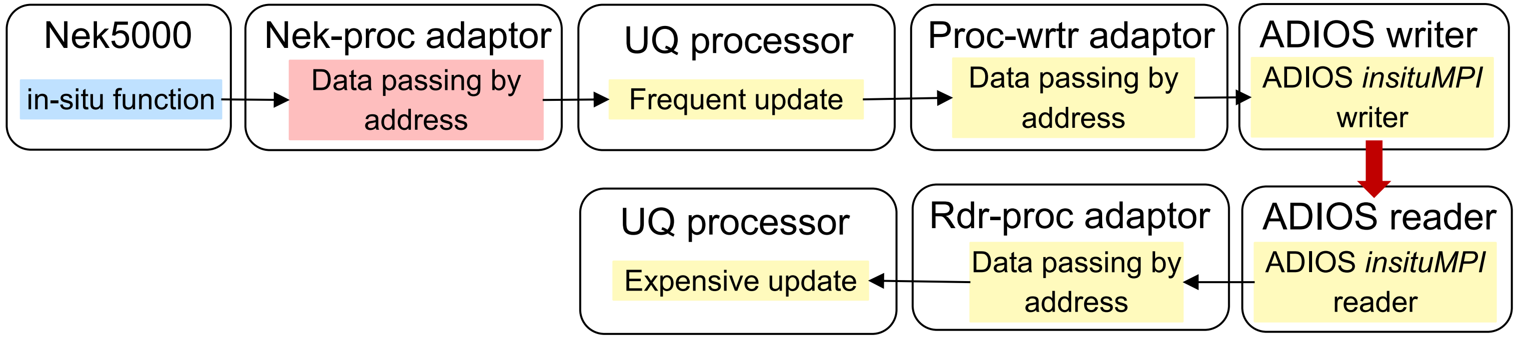
- Nek5000 with synchronous uncertainty quantification:



- Nek5000 with asynchronous uncertainty quantification:



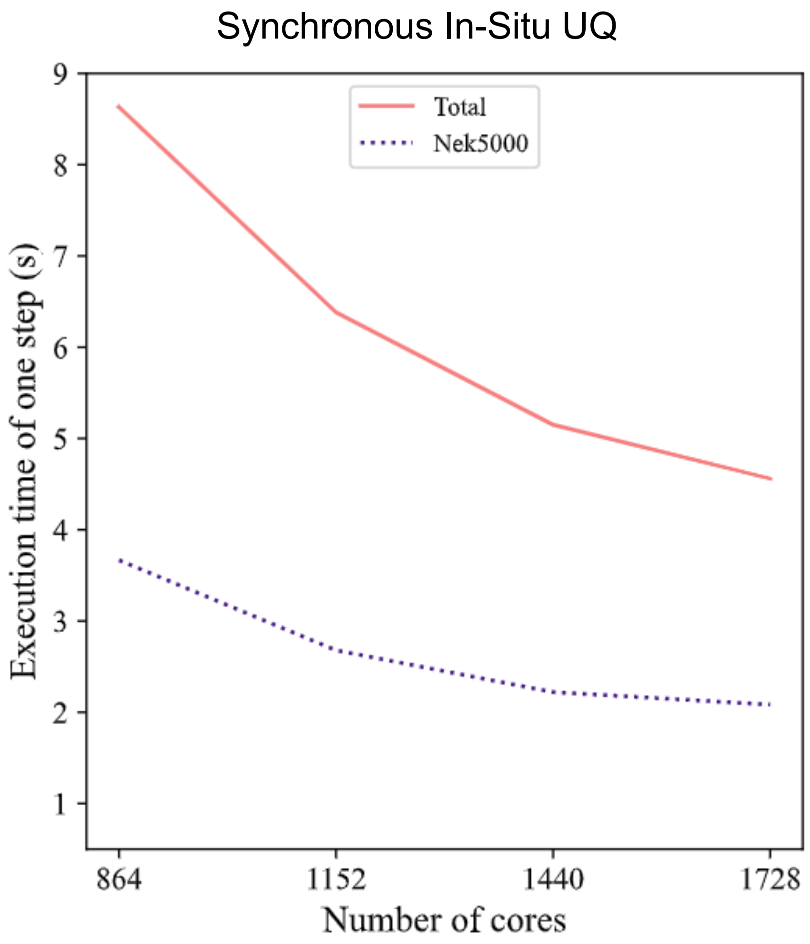
- Nek5000 with hybrid uncertainty quantification:



Fortran functions | C/C++ functions called in Fortran | Python functions | C++ functions

CPU-based Nek5000 with In-Situ Uncertainty Quantification (UQ)

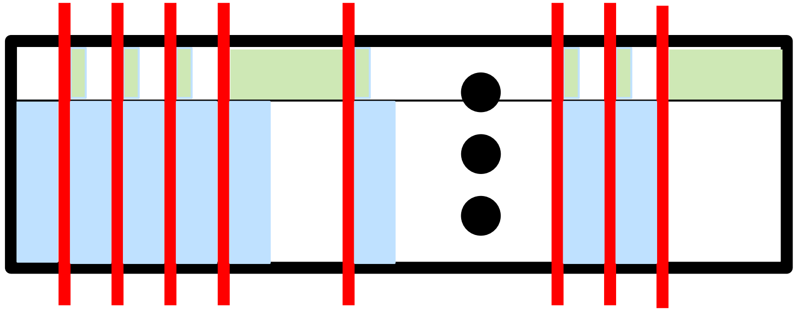
1



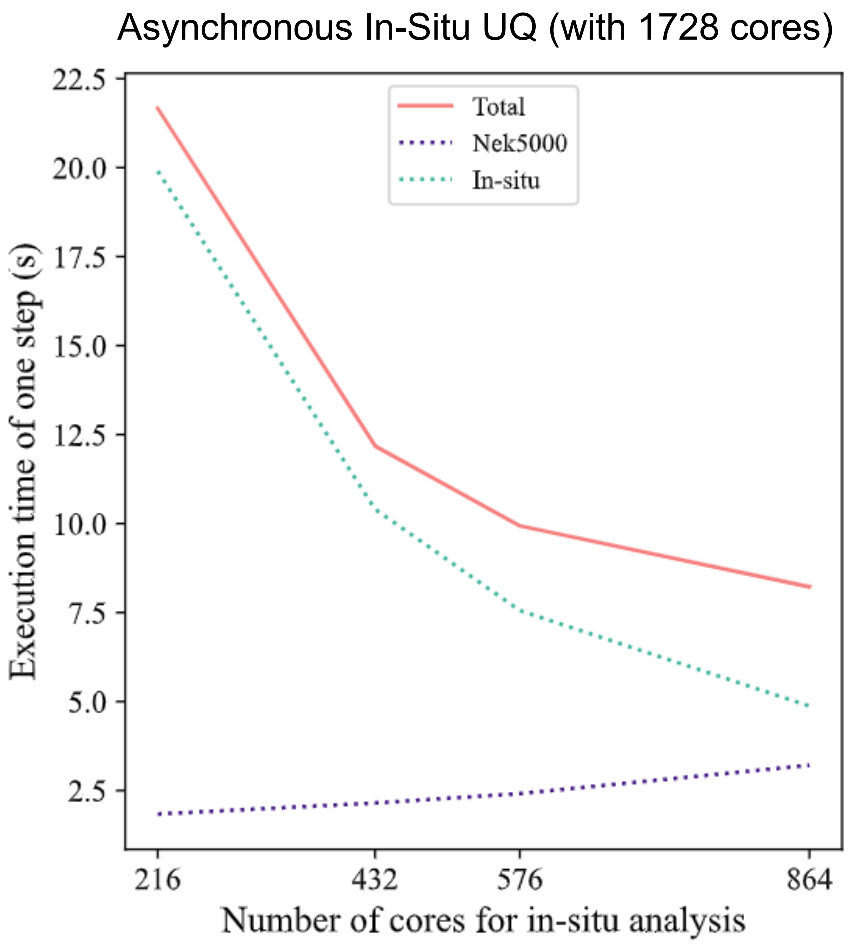
1: Execution time of Nek5000 with synchronous in-situ uncertainty quantification (left), asynchronous in-situ uncertainty quantification on 24 Raven nodes (middle) and hybrid in-situ uncertainty quantification on 24 Raven nodes (right).

CPU-based Nek5000 with In-Situ Uncertainty Quantification (UQ)

Asynchronous



1

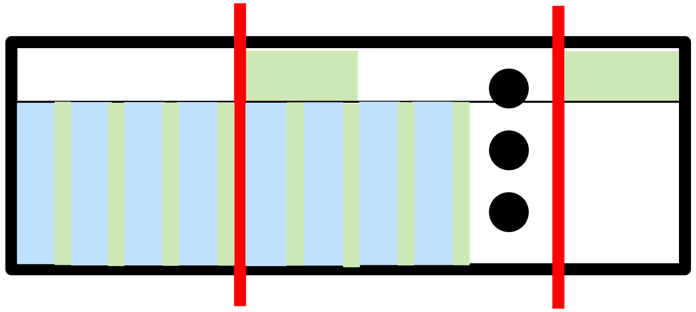


1: Execution time of Nek5000 with synchronous in-situ uncertainty quantification (left), asynchronous in-situ uncertainty quantification on 24 Raven nodes (middle) and hybrid in-situ uncertainty quantification on 24 Raven nodes (right).

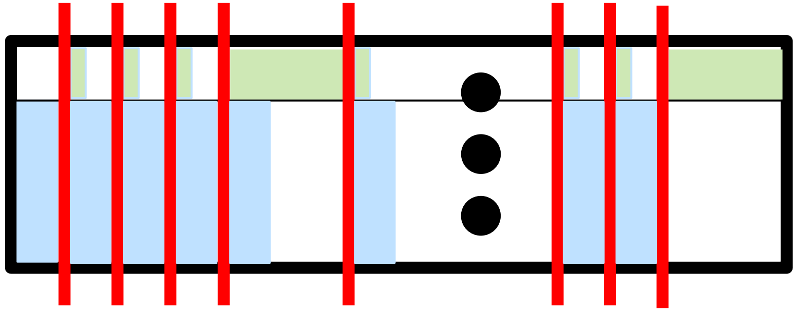
CPU-based Nek5000 with In-Situ Uncertainty Quantification (UQ)

[1]

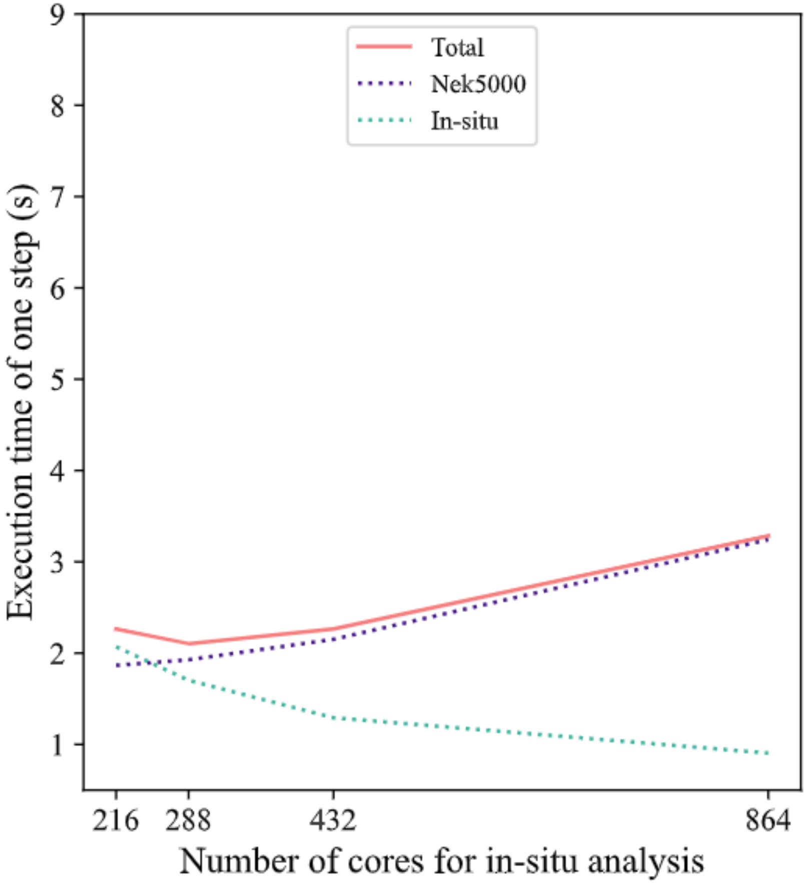
Hybrid



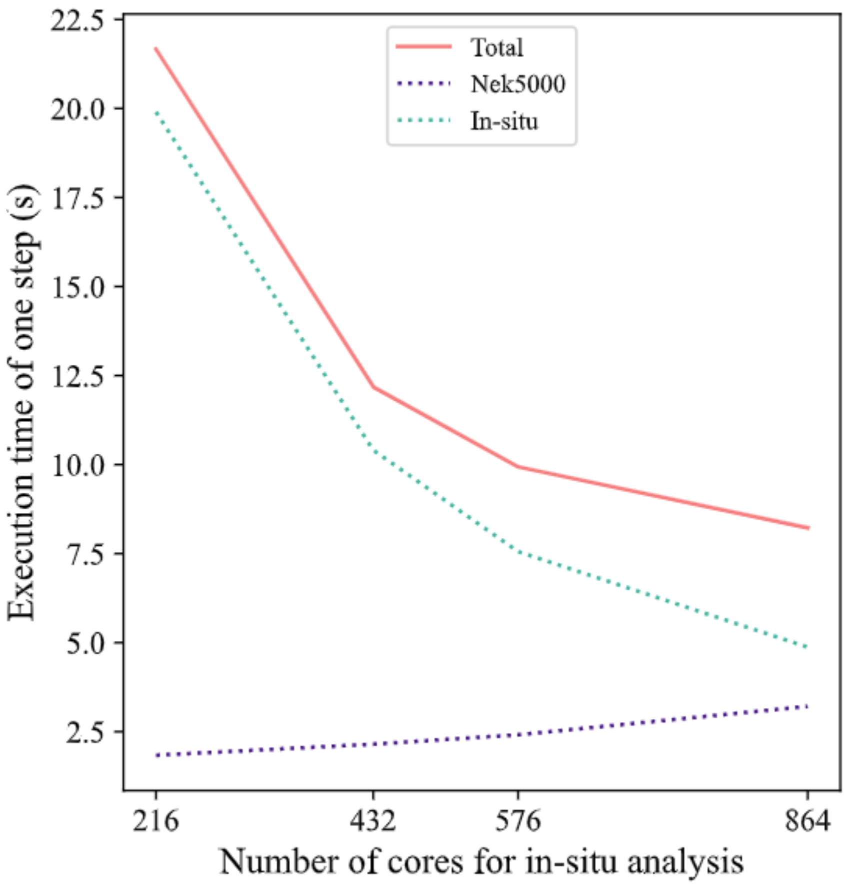
Asynchronous



Hybrid In-Situ UQ (with 1728 cores)



Asynchronous In-Situ UQ (with 1728 cores)

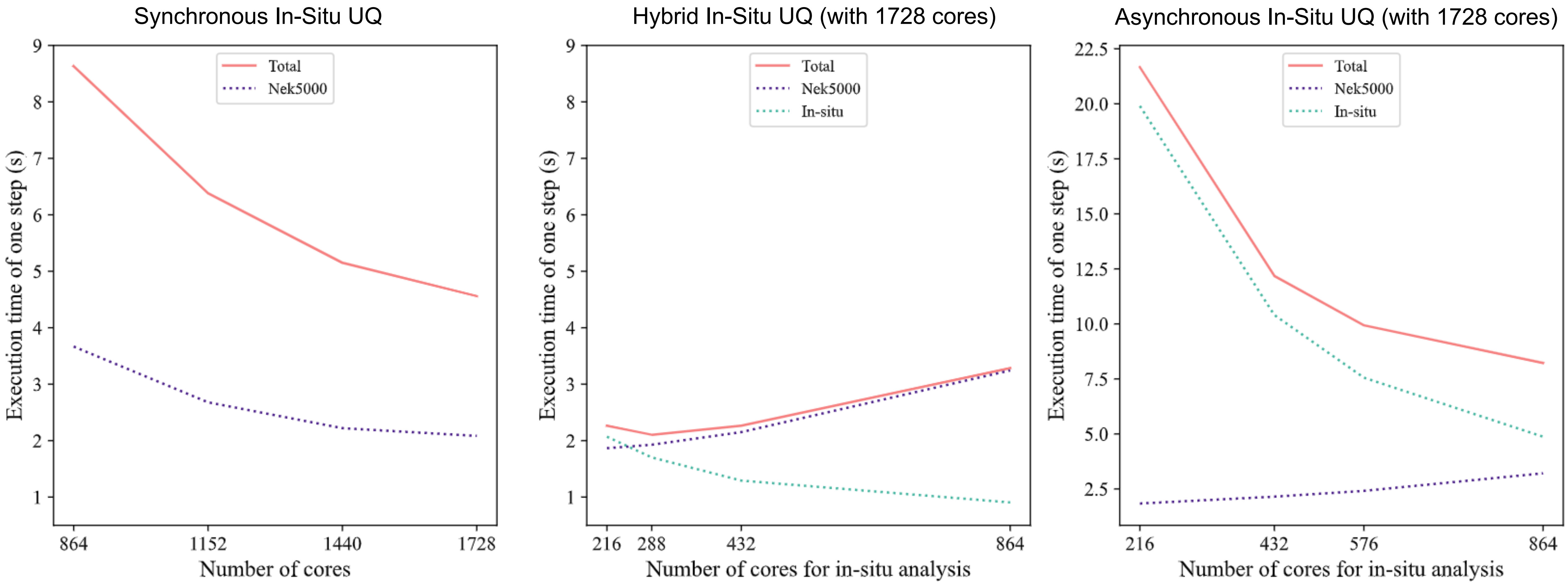


1: Execution time of Nek5000 with synchronous in-situ uncertainty quantification (left), asynchronous in-situ uncertainty quantification on 24 Raven nodes (middle) and hybrid in-situ uncertainty quantification on 24 Raven nodes (right).

malleable data solutions for HPC

CPU-based Nek5000 with In-Situ Uncertainty Quantification (UQ)

[1]



1: Execution time of Nek5000 with synchronous in-situ uncertainty quantification (left), asynchronous in-situ uncertainty quantification on 24 Raven nodes (middle) and hybrid in-situ uncertainty quantification on 24 Raven nodes (right).

Summary

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

- The synchronous in-situ data compression is preferred because of its low computational cost.
- 45GB VTK file for each in-situ step is avoided by in-situ techniques.
- The asynchronous in-situ image generation is preferred because of the optimal computing resource allocation to minimize the overhead from the MPI collective communication.
- The hybrid in-situ uncertainty quantification is preferred because of the more efficient computing resources usage

Outlook

- In-situ tasks to GPU based simulation
- In-situ tasks to exasacle simulation
- Performance model of in-situ techniques
- Dynamic computing resources allocation

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

Thank you for your attention!

Understanding the Impact of Synchronous, Asynchronous, and Hybrid In-Situ Techniques in Computational Fluid Dynamics Applications

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

- The synchronous in-situ data compression is preferred because of its low computational cost.
- 45GB VTK file for each in-situ step is avoided by in-situ techniques.
- The asynchronous in-situ image generation is preferred because of the optimal computing resource allocation to minimize the overhead from the MPI collective communication.
- The hybrid in-situ uncertainty quantification is preferred because of the more efficient computing resources usage

Outlook

- In-situ tasks to GPU based simulation
- In-situ tasks to exasacle simulation
- Performance model of in-situ techniques
- Dynamic computing resources allocation

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