PCクラスタワークショップin 柏2025
2025年6月27日（金）15:30-16:00

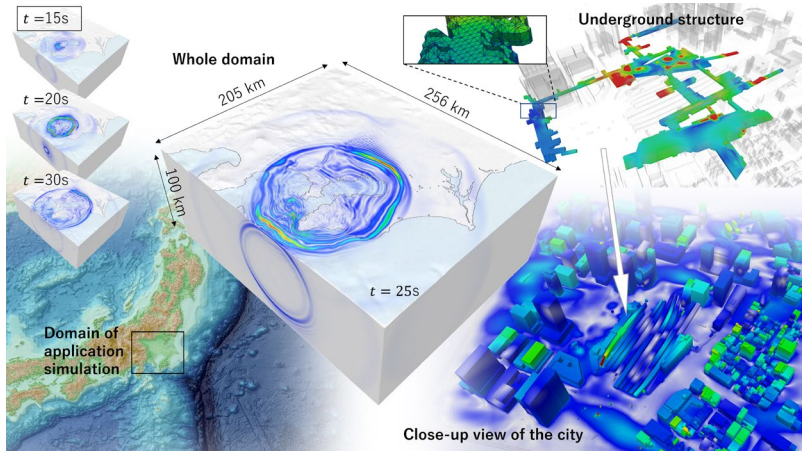
CPU-GPUの同時使用による波動シミュレーションの高速化・省エネルギー化

東京大学 地震研究所
藤田 航平

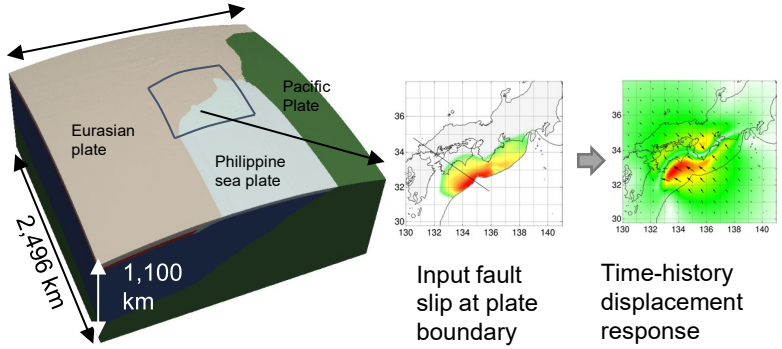
はじめに

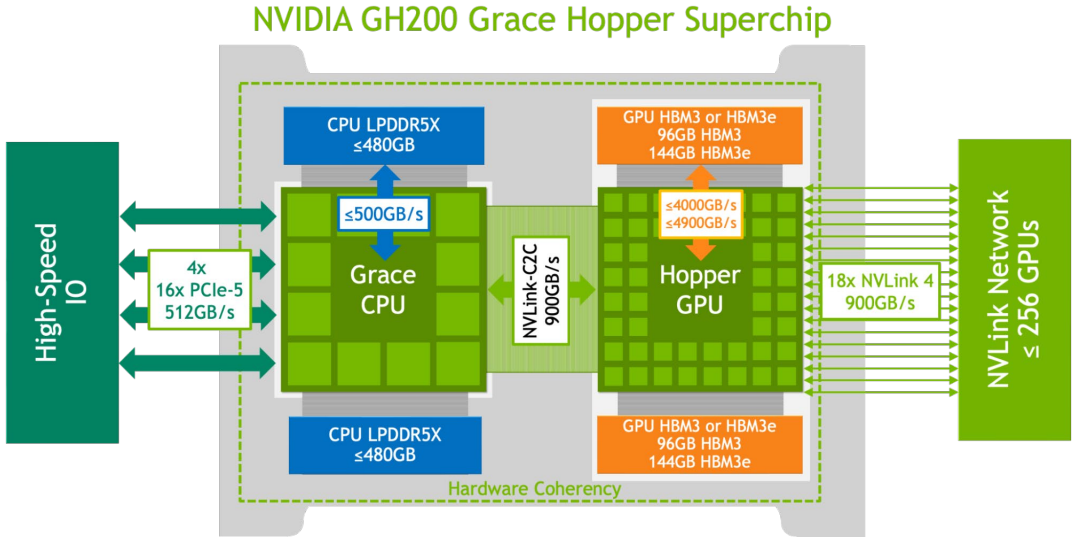
- 地震シミュレーションにおいては、偏微分方程式(PDE)に基づく広域・高分解能な時刻歴シミュレーションが必要
- 本発表ではGH200のCPUとGPUを同時に使用することで地震シミュレーションに要する時間とエネルギーの双方を削減する方法を説明

Seismic wave propagation



Crustal deformation

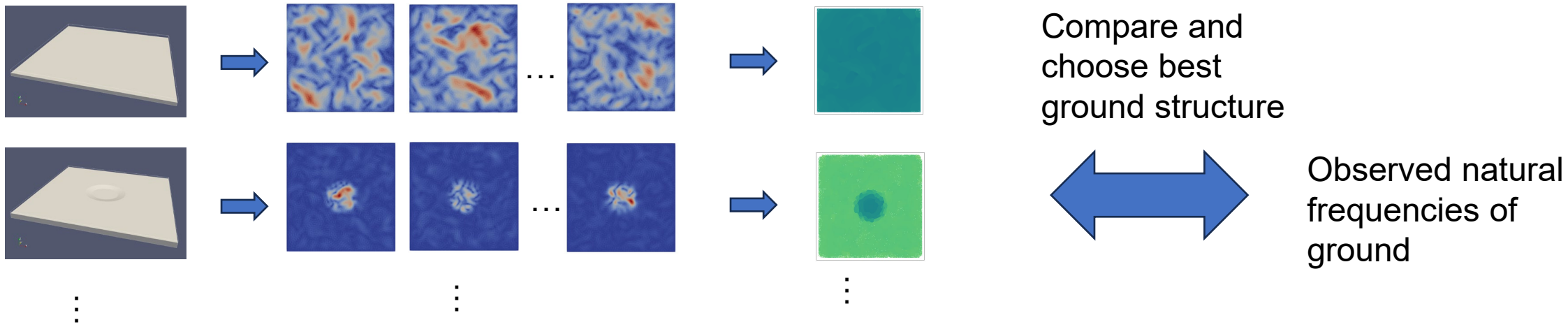




Data-driven part on memory rich CPU, Solver part on high-performance GPU, with fast CPU-GPU synchronization

シミュレーション例

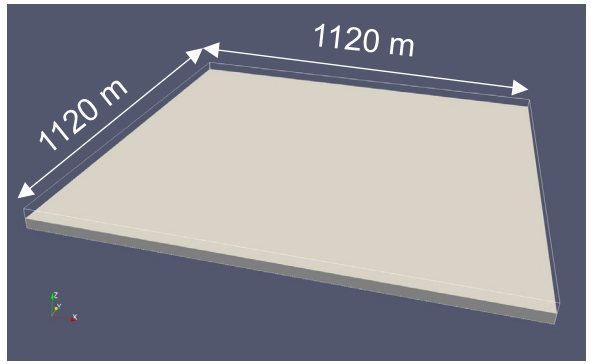
- 数値解析結果と観測データの比較により地盤特性を推定
  - 多数の候補地盤モデルについて、ランダム波入力に対する地盤振動を計算
  - 実際の観測結果に最も近い固有振動数を持つ地盤モデルを選ぶ



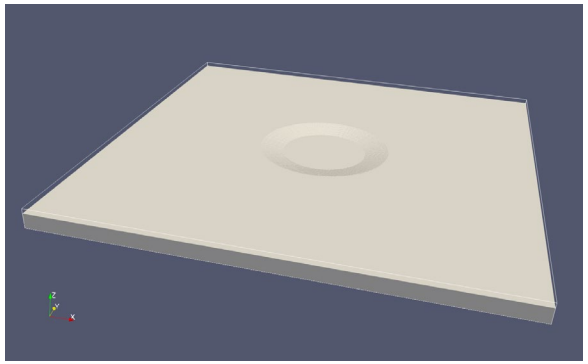
シミュレーション例

- 多ケース解析により地盤モデルの特性を計算することができる

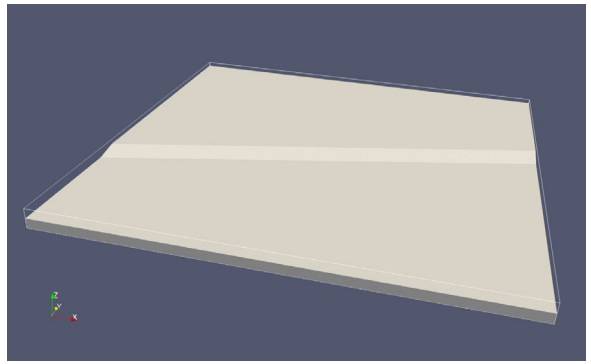
Candidate ground structures (showing the interface between soft and hard ground)



a) Horizontally stratified ground structure

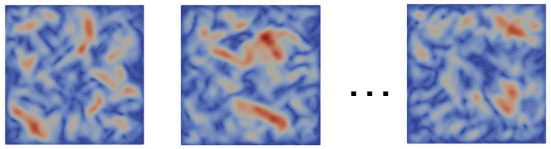


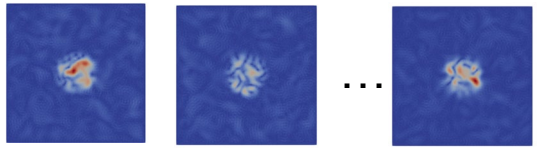
b) Ground structure with circular basin bedrock

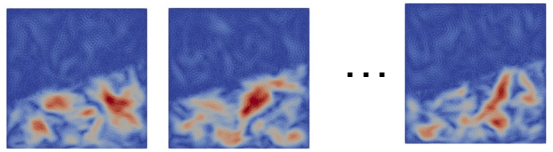


c) Ground structure with slanted bedrock

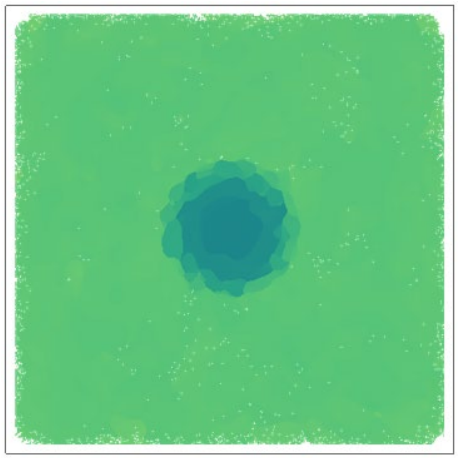
Computation of 16384-time step simulations ×8 random input cases

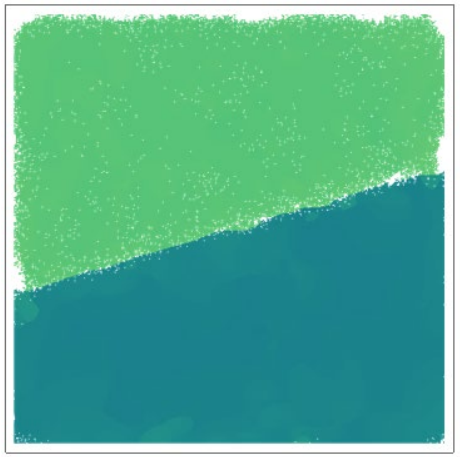


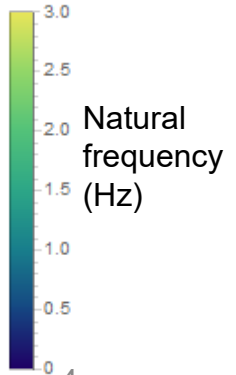




Computed natural frequencies of ground

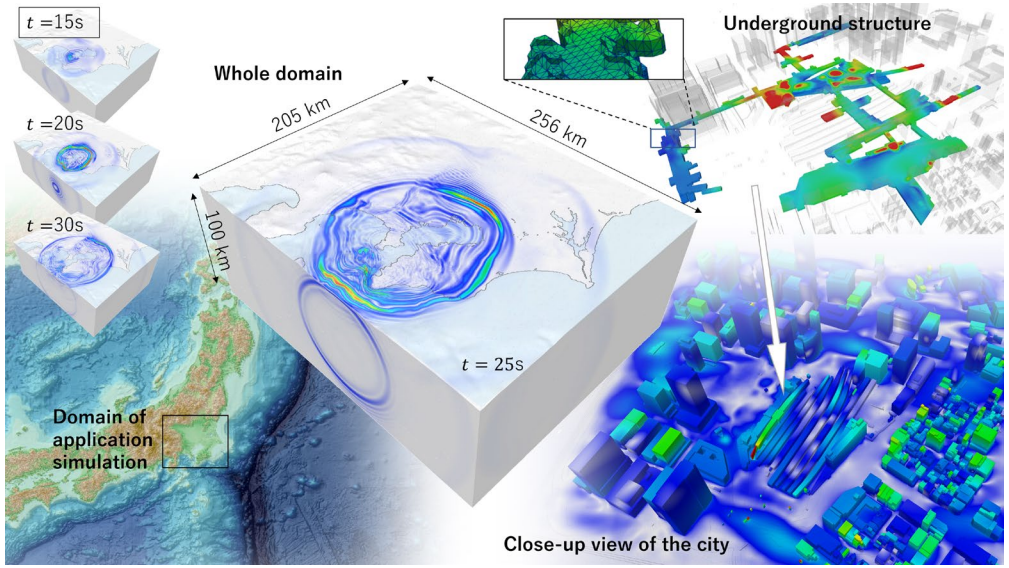




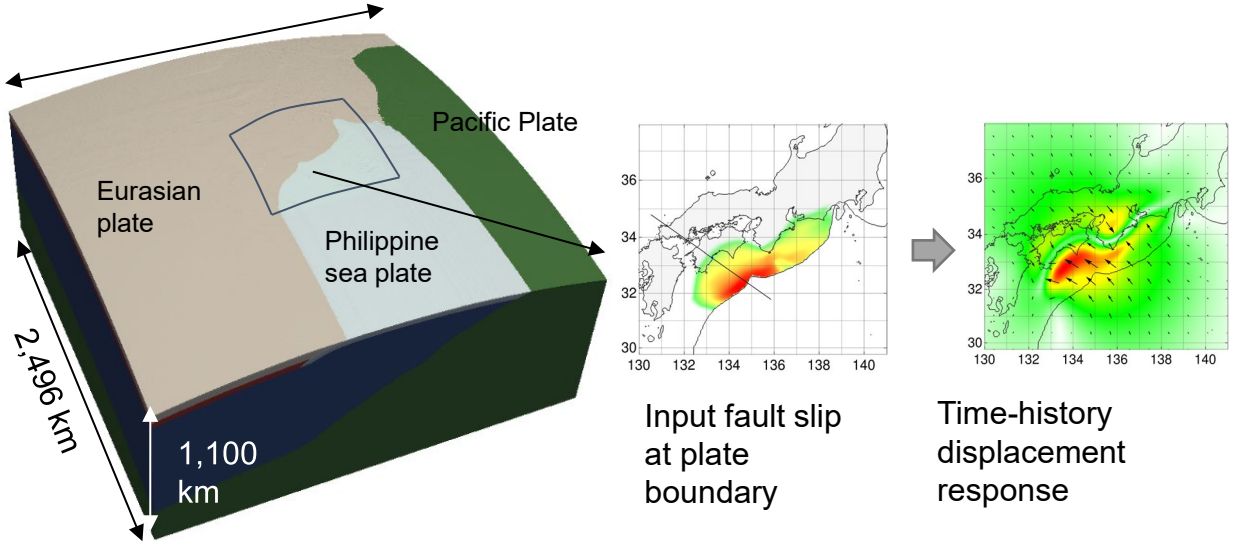


PDE求解におけるデータ駆動型手法の活用

- PDEに基づく時刻歴シミュレーションにおいて、過去の時刻歴データを用いて陰解法ソルバーにおける次のステップの初期解を予測
- ソルバーの反復回数が削減されることで、精度を落とさずに高速化を実現



Seismic wave propagation@full Fugaku (152352 nodes)
25-fold speedup from previous solver without data-driven method
Ichimura et al. HPC Asia 2022 [Best Paper]



Crustal deformation@Fugaku (73728 nodes)
Fujita et al. ScalaH 2022

PDE求解におけるデータ駆動型手法の活用

富岳A64FX
(CPU only)

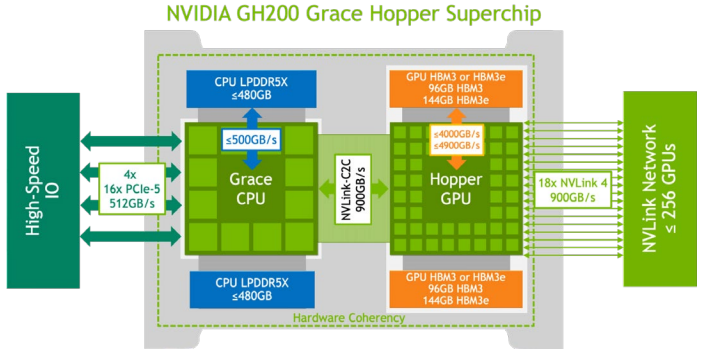
A100
(GPU only)

GH200
(CPU+GPU)

HPC Asia (Ichimura et al. 2022)
ScalaH@SC22 (Fujita et al. 2022)

ICCS 2023
(Murakami et al.)

GTC 2024, WACCPD@SC24
(Ichimura et al.)



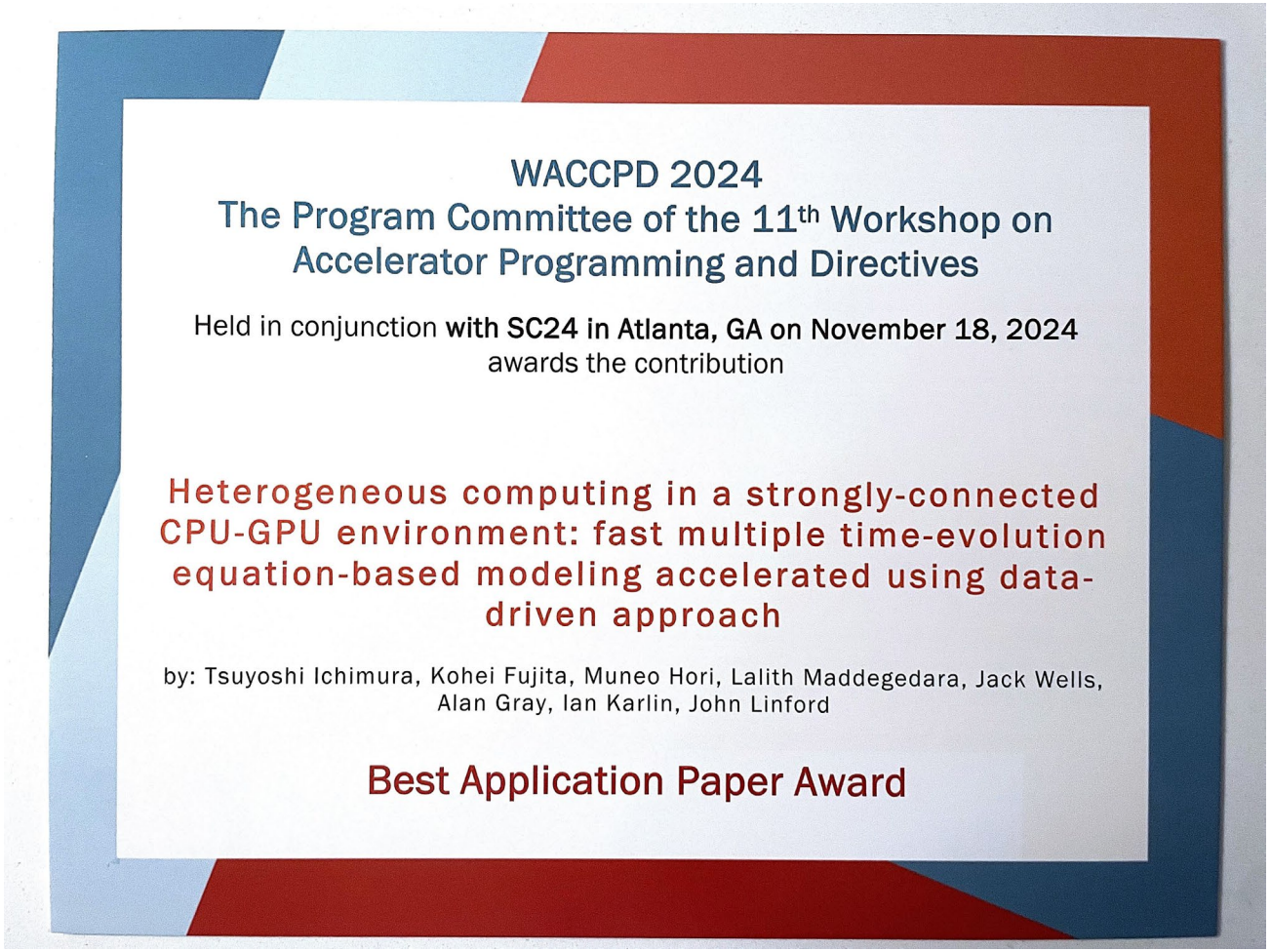
Data-driven part on memory rich CPU,
Solver part on high-performance GPU

- 本発表では、CPUとGPUを同時に使用してPDEに基づく波動シミュレーションを高速化する研究について説明

Heterogeneous computing in a strongly-connected CPU-GPU environment: fast multiple time-evolution equation-based modeling accelerated using data-driven approach

Tsuyoshi Ichimura, Kohei Fujita,
Muneo Hori, Maddegedara Lalith,
Jack Wells, Alan Gray,
Ian Karlin, John Linford

Eleventh Workshop on Accelerator Programming and Directives (WACCPD) 2024@SC24



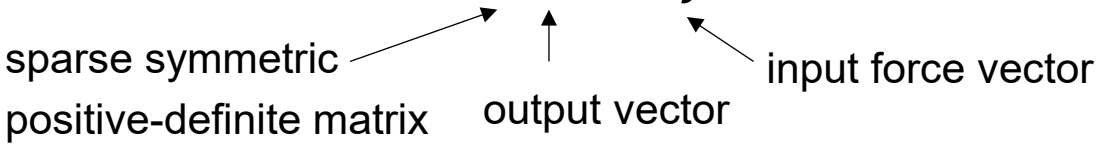
- CPUとGPUを同時使用する提案手法はGH200システム上でシミュレーションの実行時間と低エネルギー化を実現
- 大規模システムにおいても高い並列性能を実現

はじめに

- 近年の計算機特性を活用したPDEに基づく時刻歴シミュレーションの高速計算
  - 大容量のCPUメモリ：データ駆動型手法において用いる大容量データを保存可能
  - 高速なGPUと、プログラム開発環境の整備 (e.g., directive-based programming model)：PDEに基づく時刻歴シミュレーションの高速化においてGPUは広く使われている
- PDEに基づく時刻歴シミュレーションにおけるデータ駆動型手法の活用
  - CPUメモリに格納したデータを活用した計算高速化は可能だが、CPUの絶対性能はGPUよりは低い
  - その一方でGPUはメモリ容量が限られているため、問題規模を維持しながらデータ駆動型手法によりGPUベースの解析を高速化することは難しい

ターゲット問題とbaseline solver

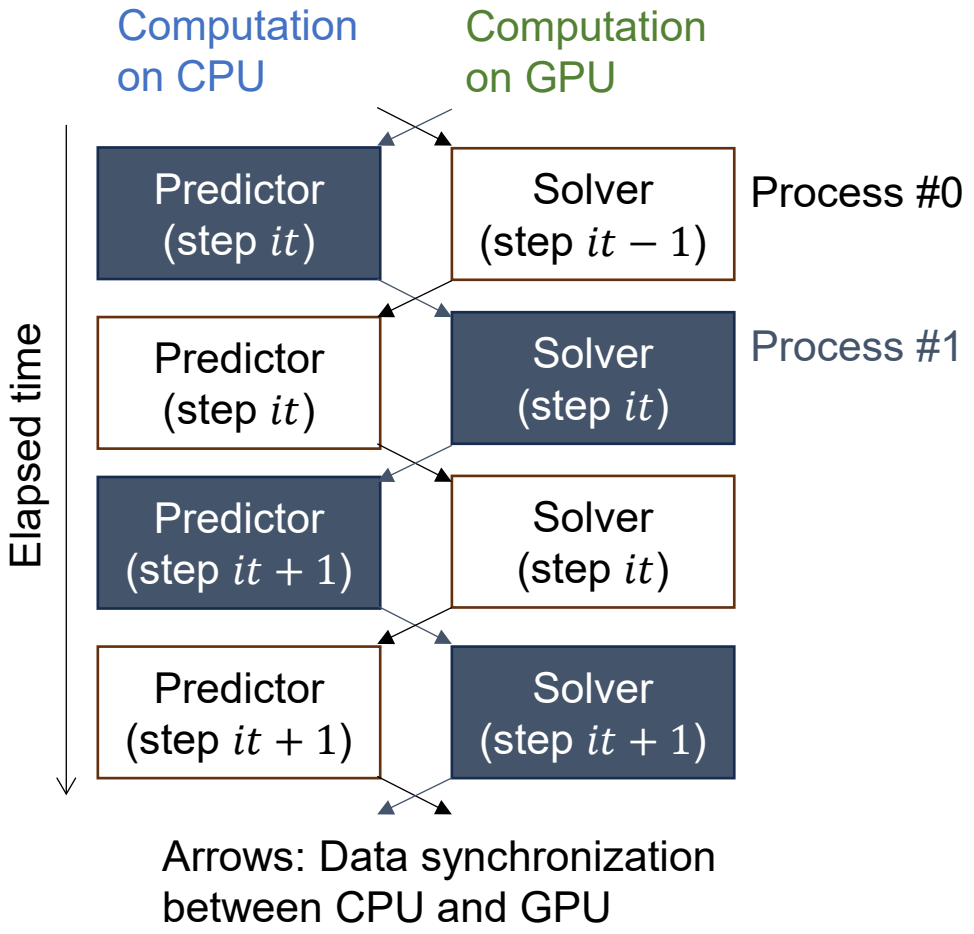
- CPU-GPU間が高速に接続されたCPU-GPU環境を用いて、多数ケースのPDEに基づく時刻歴シミュレーションを実施
  - CPU-GPU間が高速に接続されたシステム上でデータ駆動型手法を用いることで、大容量メモリを搭載したCPUと高性能GPUの両方を活用
  - 各時間ステップitにおいて連立方程式$\boldsymbol{A}\boldsymbol{x}^{it} = \boldsymbol{f}^{it}$を求解



- Baseline solver:
  - 問題規模が大きいため、反復法ソルバーを使用
  - 具体的には、Compressed Row Storage (CRS)形式で格納した行列に基づく疎行列ベクトル積を用いた共役勾配法を使用

Proposed concurrent CPU and GPU computing method

- 多数ケースのシミュレーションを同時に実行
- CPUにより次のタイムステップの解を予測(predictor)
- GPUにより線型方程式を反復求解(iterative solver)
- 高速なCPU-GPU interconnectを用いて、predictor/iterative solverの前後で初期解・求解結果を交換



Proposed concurrent CPU and GPU computing method (GPU part)

- GPUの性能を引き出すために、以下を実施
  - Matrix-free SpMV (sparse matrix-vector) multiplicationを活用
    - CRS形式で格納した全体行列$\boldsymbol{A}$をglobal memoryから読み出して疎行列ベクトル積$\boldsymbol{Ax}^{it}$を計算する代わりに、マトリクスフリー形式で疎行列ベクトル積を$\sum_e \boldsymbol{P}_e^{\boldsymbol{T}} \left(\boldsymbol{A}_e \left(\boldsymbol{P}_e \boldsymbol{x}^{it}\right)\right)$と計算

      要素行列

      全体節点番号から要素内のローカルな節点番号へのマッピング行列
    - 計算量は増えるものの、メモリアクセスは削減され、結果として演算性能の高いGPUにおいては計算速度が向上
  - Matrix-free SpMVを使うことでメモリ使用量も削減されるため、余ったメモリ容量を用いて多数ケースを同時に計算することが可能となる
    - これにより、疎行列ベクトル積計算を、より演算効率の出しやすいGSpMV (SpMV with multiple right-hand side vectors)に変換できる

$$\sum_e \boldsymbol{P}_e^{\boldsymbol{T}} \left(\boldsymbol{A}_e \left(\boldsymbol{P}_e \boldsymbol{x}^{it}\right)\right) \quad \Rightarrow \quad \sum_e \boldsymbol{P}_e^{\boldsymbol{T}} \left(\boldsymbol{A}_e \left(\boldsymbol{P}_e \{\boldsymbol{x}_0^{it}, \boldsymbol{x}_1^{it}, \boldsymbol{x}_2^{it}, \boldsymbol{x}_3^{it}\}\right)\right)$$

Proposed concurrent CPU and GPU computing method (CPU part)

- データ駆動型手法により、精度を担保しながらソルバーを高速化 @CPU
  - 反復ソルバーの初期解を以下のように予測

$$\bar{\boldsymbol{x}}^{it} = predictor(\boldsymbol{X}^{it}, \boldsymbol{F}^{it}, \boldsymbol{f}^{it}),$$

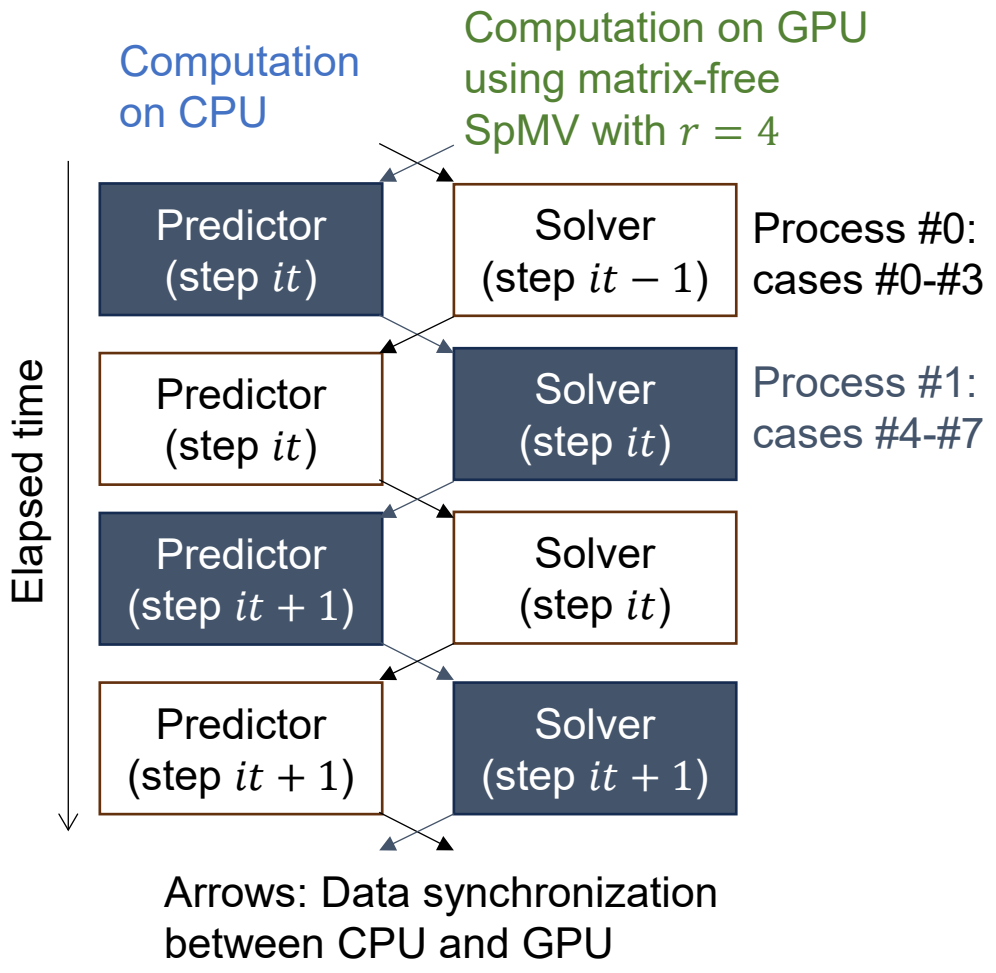
ここで、$\boldsymbol{X}^{it}, \boldsymbol{F}^{it}$は過去$s$時間ステップのデータ

$$\boldsymbol{X}^{it} = \left\{\boldsymbol{x}^{it-s}, \boldsymbol{x}^{it-s+1}, \ldots, \boldsymbol{x}^{it-1}\right\}, \boldsymbol{F}^{it} = \left\{\boldsymbol{f}^{it-s}, \boldsymbol{f}^{it-s+1}, \ldots, \boldsymbol{f}^{it-1}\right\}$$

  - 初期解を高精度で予測することで、ソルバーの反復数減少・総計算時間の短縮につながる
  - 時系列データは大容量となりGPUメモリに収まらないため、CPUにて計算

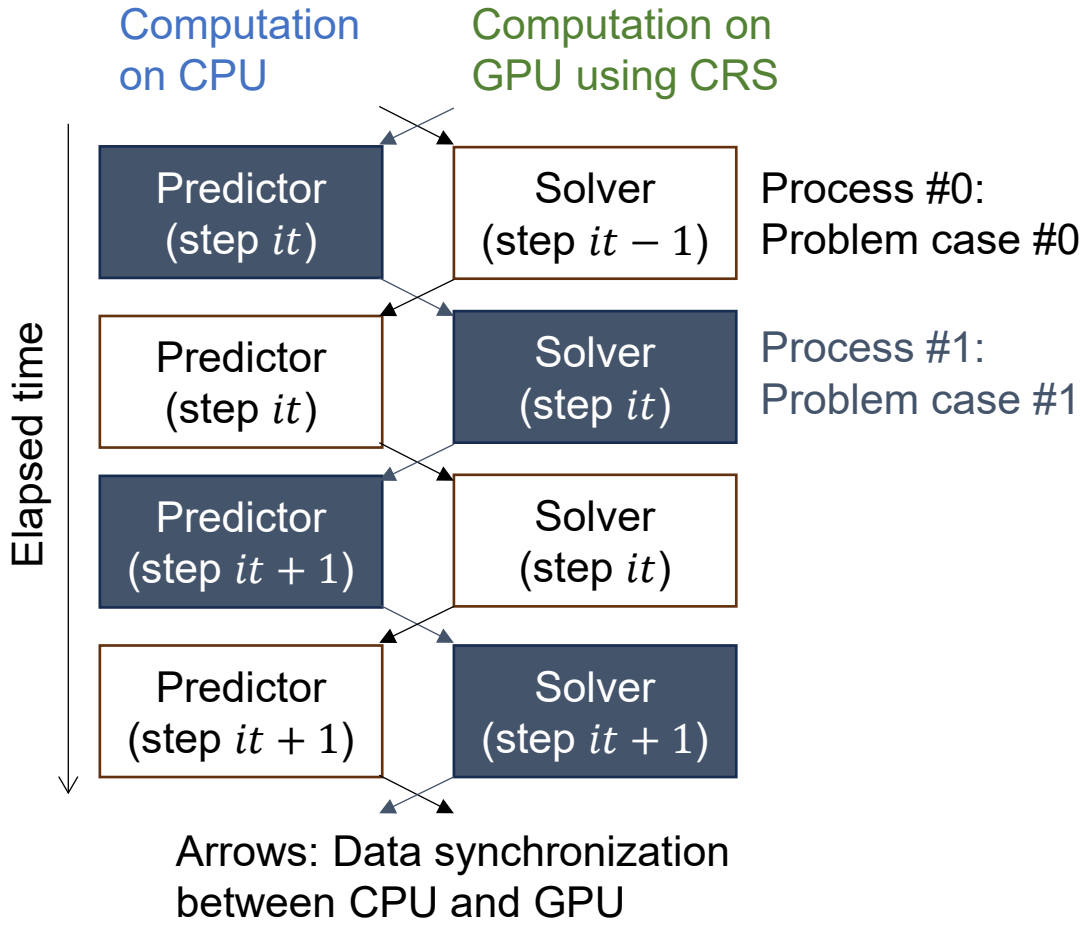
Proposed concurrent CPU and GPU computing method

- CPUとGPUを同時に用いることで$2r$ケースの問題を求解
  - CPUにおいて1セット(rケース)の問題の初期解をデータ駆動型手法により予測
  - GPUにおいてもう1セット(rケース)の問題を反復法ソルバーにより求解
  - Predictor/iterative solverの前後でCPU-GPU間で初期解・解析結果を同期
- 全実行時間を通してCPUとGPUを同時に活用
  - Predictorの学習/予測において用いるタイムステップ数sをシミュレーション中に動的に選択することで、predictor@CPUとsolver@GPUの実行時間を同程度とすることができる



Proposed heterogeneous computing method in reduced form

- 一部の問題では、matrix-free SpMVを用いることが難しい場合がある
- その場合、提案手法は、CRSベースの反復法ソルバーを用いて2ケースの問題を解くこととなる



数値実験:ターゲット問題

- 線形動的弾性問題における性能を評価する

$$\rho\ddot{\boldsymbol{u}} - (\nabla \cdot \boldsymbol{c} \cdot \nabla) \cdot \boldsymbol{u} = \boldsymbol{f}$$

  - $\boldsymbol{u}$: displacement, ρ: density, $\boldsymbol{c}$: elasticity tensor, $\boldsymbol{f}$: outer force
- 四面体二次要素による離散化・Newmark-β法による時間積分を適用すると、対象問題は以下となる

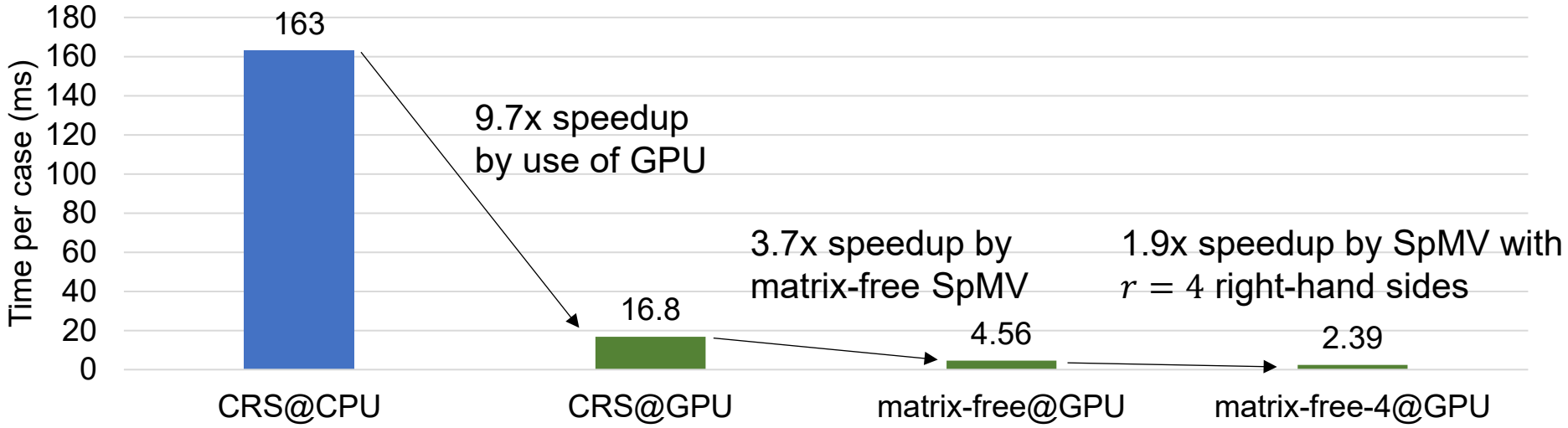
$$\left(\frac{\boldsymbol{M}}{dt^2} + \frac{\boldsymbol{C}}{dt} + \boldsymbol{K}\right)\boldsymbol{u}^{it} = \boldsymbol{f}^{it} + \boldsymbol{C}\boldsymbol{v}^{it-1} + \boldsymbol{M}\left(\boldsymbol{a}^{it-1} + \frac{4}{dt}\boldsymbol{v}^{it-1}\right)$$

- この式を各時間ステップitにおいて以下のように解く
  - (CPUまたはGPUにおける) baseline method: block-CRSに基づく疎行列ベクトル積カーネルを用いた共役勾配法
  - 提案手法:multi-vector matrix-free SpMVに基づく共役勾配法において、文献[1]のデータ駆動型手法により予測した初期解を使用
  - Baseline/提案手法ともに同じ前処理を使用 (3x3 block Jacobi preconditioning)
  - OpenACC (GPU computing), OpenMP (CPU computing), 及びMPIを用いて実装

[1] Tsuyoshi Ichimura et al., HPC Asia ’22, best paper award

GH200 1ノードにおける行列ベクトル積カーネル性能

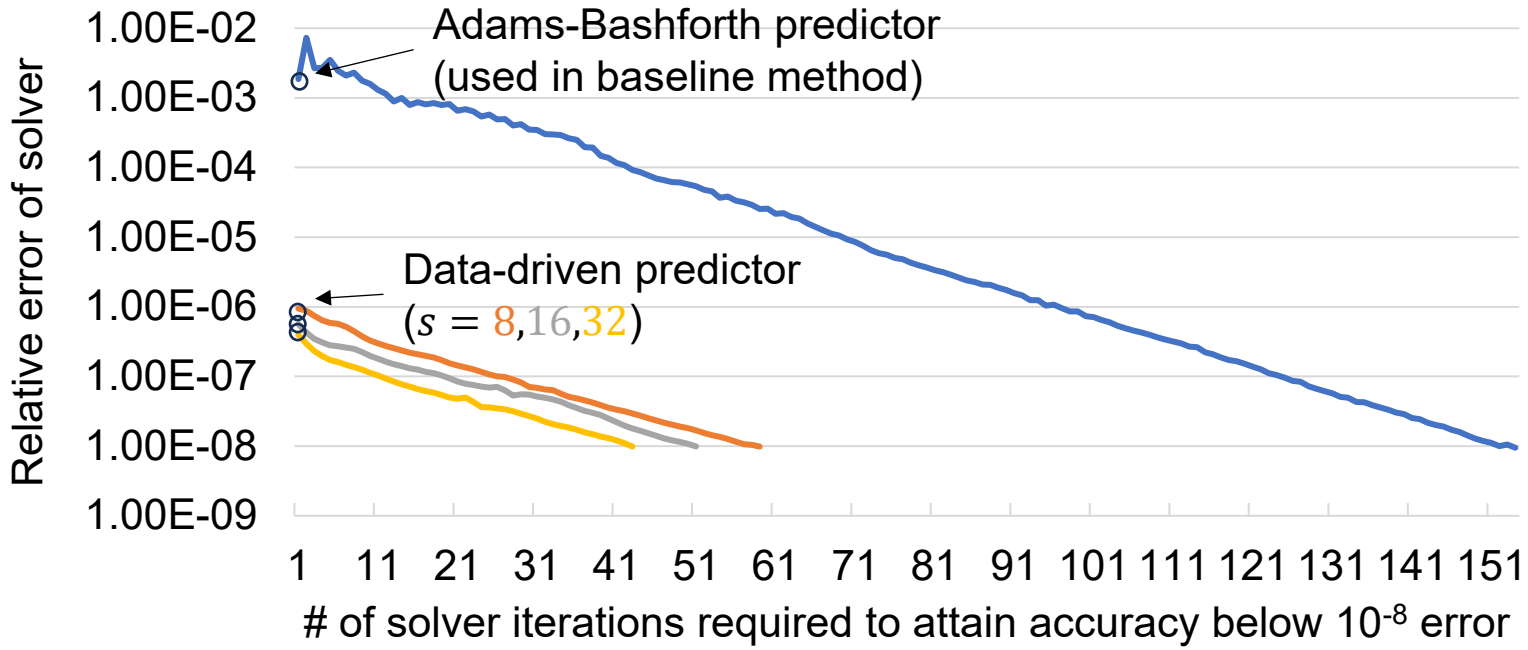
- 以下の構成のGH200 1ノードにおいて性能を評価
  - CPU: 72-core ARMv9a Grace with 480 GB (384 GB/s) memory
  - GPU: H100 (34 TFLOPS) with 96 GB (4000 GB/s) memory



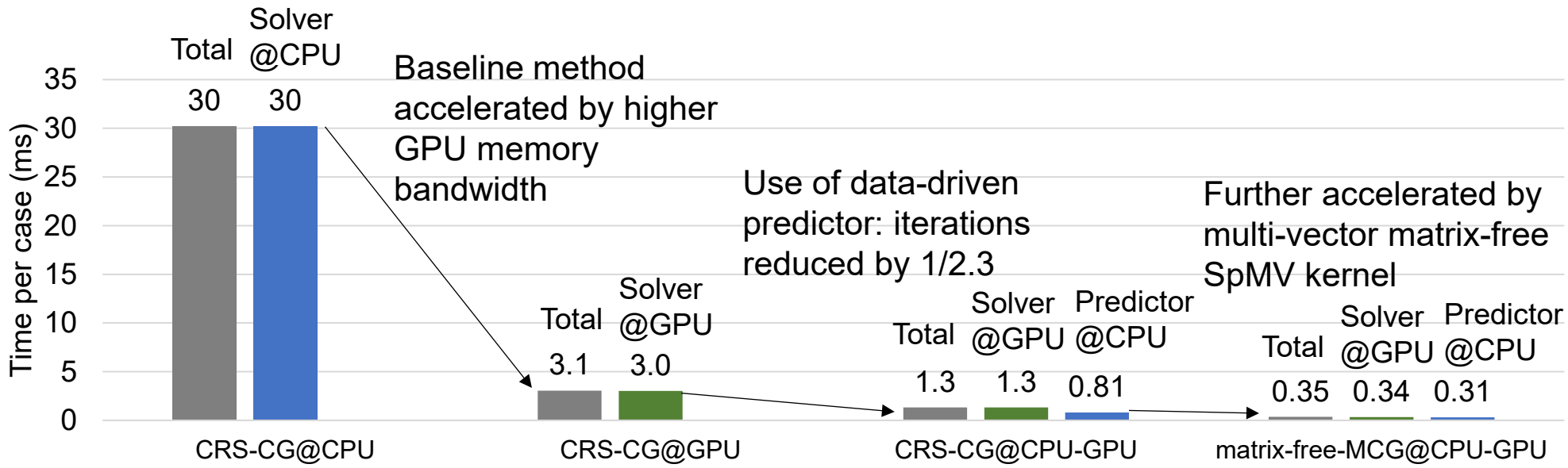
Memory Bandwidth (% to peak)	0.21 TB/s (**55%**)	2.0 TB/s (**51%**)	0.58 TB/s (15%)	0.51 TB/s (13%)
TFLOPS (% to peak)	0.049 TFLOPS (1.4%)	0.47 TFLOPS (1.4%)	9.5 TFLOPS (28%)	18 TFLOPS (53%)

データ駆動型手法の性能

- 初期解の精度向上がソルバーの反復数削減につながっている
  - より多くのステップ数sを用いることで、predictorによる初期解精度が改善し反復数もより少なくなる



GH200 1ノードにおけるアプリケーション性能(概要)



Relative speedup	1.0	10	26	86
Relative energy usage	32	7.0	3.2	1.0

- 86-fold speedup and 32-fold reduction in energy from using only CPU
- 8.7-fold speedup and 7.0-fold reduction in energy from using only GPU

GH200 1ノードにおけるアプリケーション性能(詳細)

- Baseline CPUとGPUでは、メモリバンド幅比程度で実行時間が1/10に減少。総電力が2倍程度となるため、エネルギー使用量は約1/5に
- CPUとGPUの同時使用により電力はGPUのみを用いた場合と比べて増加。その一方で、データ駆動型手法により、反復数が152から66～68に減少することで、全実行時間は1/2.3に削減。結果としてエネルギー使用量1/2に
- Multi-vector matrix-free SpMVカーネルによりさらに実行時間・エネルギー使用量が4倍改善

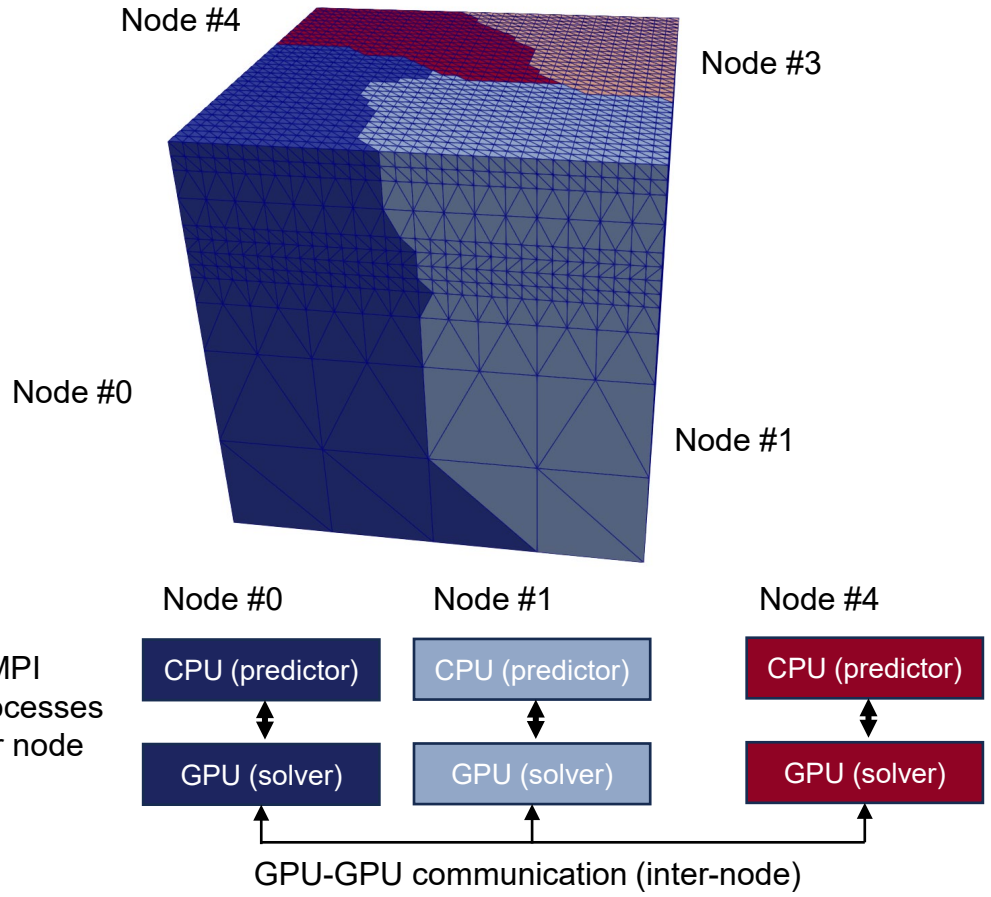
Power including CPU, memory, GPU

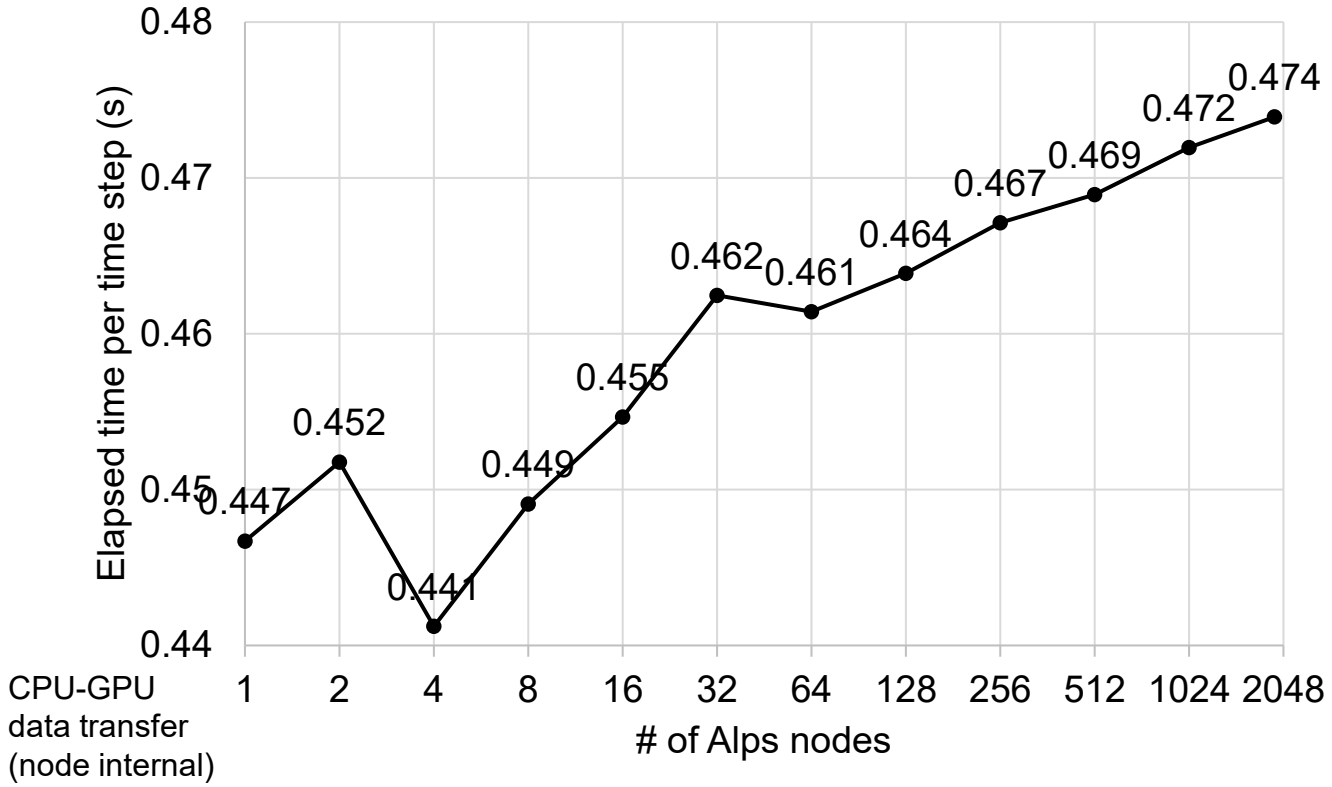
	CPU memory usage	GPU memory usage	Total elapsed time per case	Elapsed time for solver per case	Elapsed time for predictor per case	Solver iterations per time step	Relative speedup	Module power (GPU power)	Total energy per time step per case
CRS-CG@CPU	56.9 GB	-	30.4 s	30.2 s	No predictor	152	1.00	327 W (76 W)	9944 J
CRS-CG@GPU	104 GB	44.9 GB	3.05 s	3.03 s	No predictor	152	9.96	709 W (608 W)	2163 J
CRS-CG@CPU-GPU	178 GB	57.8 GB	1.17 s	1.33 s	0.812 s	66.6	26.1	858 W (604 W)	1001 J
EBE-MCG@CPU-GPU	340 GB	60.5 GB	0.352 s	0.336 s	0.310 s	68.8	86.4	877 W (652 W)	309 J

Computed on GPU　Computed on CPU

Weak scaling on Alps@CSCS

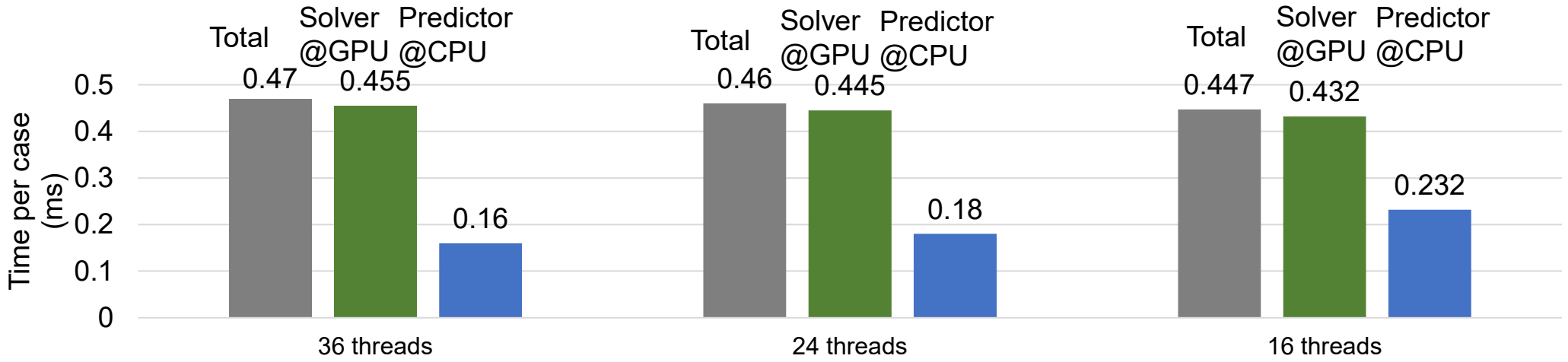
- Predictor･solver間のデータ転送は各ノード内部で行われるため、多数ノードまで高い並列性能が得られる
- 94.3% weak scalability to 1920 Alps nodes (7680 GH200 modules)





Method flexible for systems with power caps and varying memory size, CPU or GPU capabilities

- 例：CSCS Alpsでは、モジュールあたり617 Wの電力制約があるため、CPUとGPUを同時に高負荷で実行することはできない
  - 通常、CPU電力が優先され、余った電力枠にGPU電力が収まるようGPU周波数が調整される
  - ここではCPUのOpenMPスレッド数を変更することでCPUとGPUに割り振られる電力の割合を変更



Relative speedup	1.0	1.02	1.05
Relative energy usage	1.05	1.03	1.0

CPUのOpenMPスレッド数を減らすことで、CPUの電力消費が時間的に分散され、
経過時間の短縮とエネルギー削減につながる

まとめ

- CPUとGPUが密に結合されたシステムにおいてPDEに基づく時刻歴シミュレーションを高速化する手法を開発
  - データ駆動型手法とCPU-GPU間の高速インターコネクトを用いることで、大容量CPUメモリと高性能GPUの両方を活用
- シミュレーションのスループット、及び、エネルギー消費量の双方を、シミュレーション精度を落とすことなく改善
  - 86-fold speedup and 32-fold reduction in energy from using only CPU
  - 8.7-fold speedup and 7.0-fold reduction in energy from using only GPU
- 94% weak scalability up to 1920 CSCS Alps compute nodes
- 電力制約やコンピュータ・アーキテクチャの特性に合わせたパラメータチューニングが可能