第１１回計算科学技術特論B(2026)
大規模地震シミュレーション２

東京大学
藤田航平

2026年6月25日（木）13:00－14:30

主催：高度情報科学技術研究機構(RIST)
　　次世代HPC・AI研究開発支援センター(HAIRDESC)
共催：東京大学物性研究所
後援：理化学研究所計算科学研究センター、
　　計算物質科学人材育成コンソーシアム(PCoMS)

内容・スケジュール

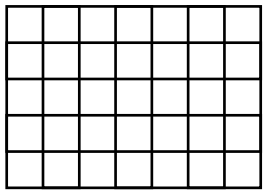
- 6/18 1300-
  - 地震シミュレーションの概要
  - 有限要素法の基礎
  - 大規模連立方程式の求解方法(共役勾配法)
  - 共役勾配法における並列化
  - 共役勾配法における前処理
  - 大規模地震シミュレーションの例

- 6/27 1300-
  - 前回の復習
  - CPU向け有限要素法の高速化
    - SIMDに適した有限要素法アルゴリズム・実装
  - GPU向けの有限要素法の高速化
    - 低精度演算・通信の活用
    - テンソルコアの活用
  - データ駆動型手法を活用した有限要素法の高速化
  - CPU-GPU協調計算

前回の復習

地震シミュレーション概要

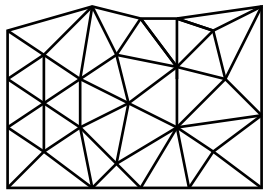
- 断層～地殻～地盤～構造物
  - 領域サイズ: 100 km
  - 空間分解能: 0.1 m
  - 複雑構造をもつ都市などを高精度で求解するため、有限要素法が使われることが多い
  - 都市規模の問題は数千億自由度規模の超大規模問題になるうえに、有限要素法ではランダムアクセスが主体となり計算効率が下がる傾向にあるため、高性能計算技術が必須



Example code:

```
do i=1,nx
do j=1,ny
a(i,j)=b(i,j)+b(i+1,j)+….
enddo
enddo
```

構造格子(差分法など)



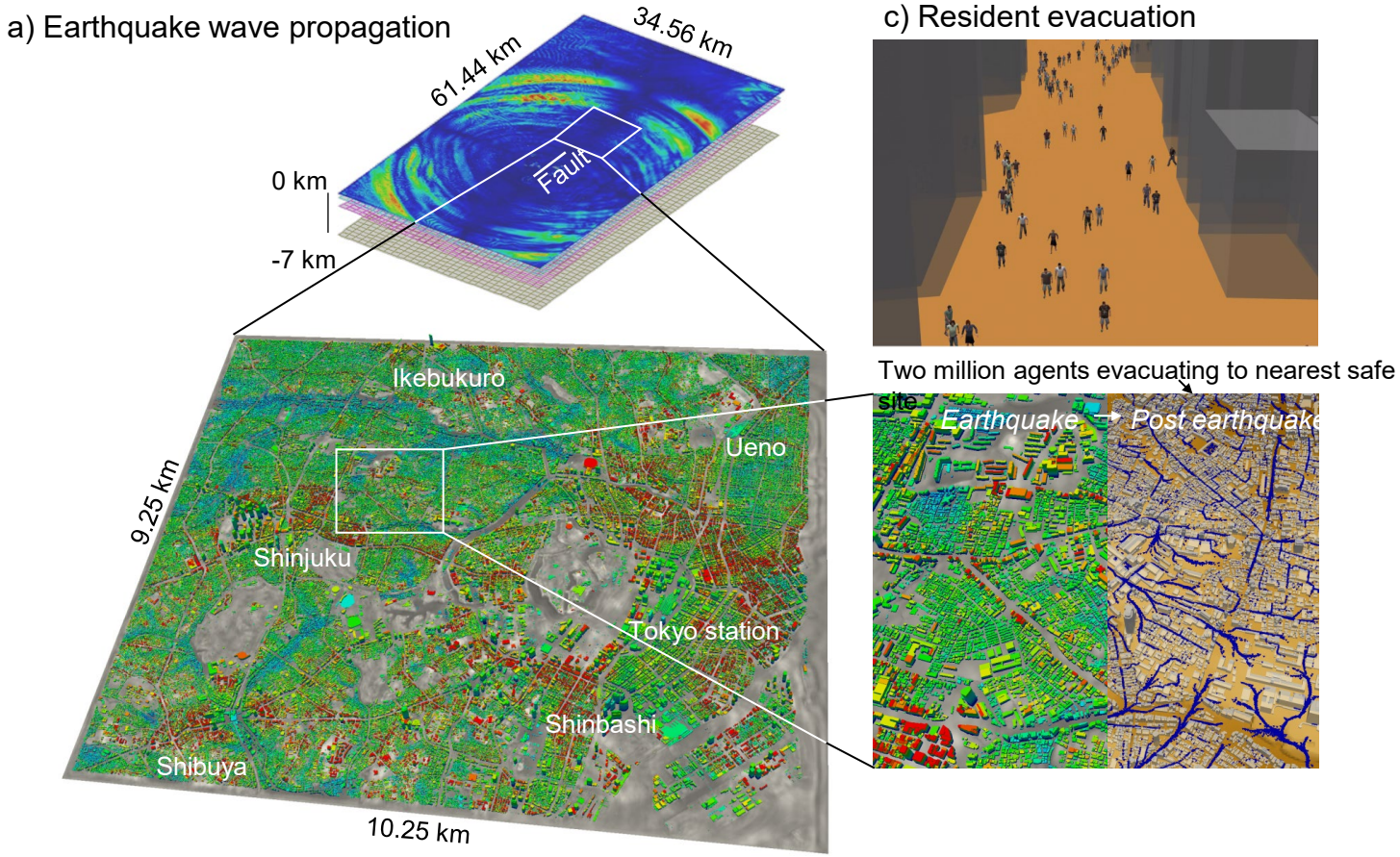
Example code:

```
do i=1,n
do j=ptr(i)+1,ptr(i+1)
a(i)=a(i)+b(index(j))
enddo
enddo
```

非構造格子(有限要素法など)

解析例

- ターゲット：断層～地殻～地盤～構造物～社会活動



京コンピュータ全系を使った
地震シミュレーション例

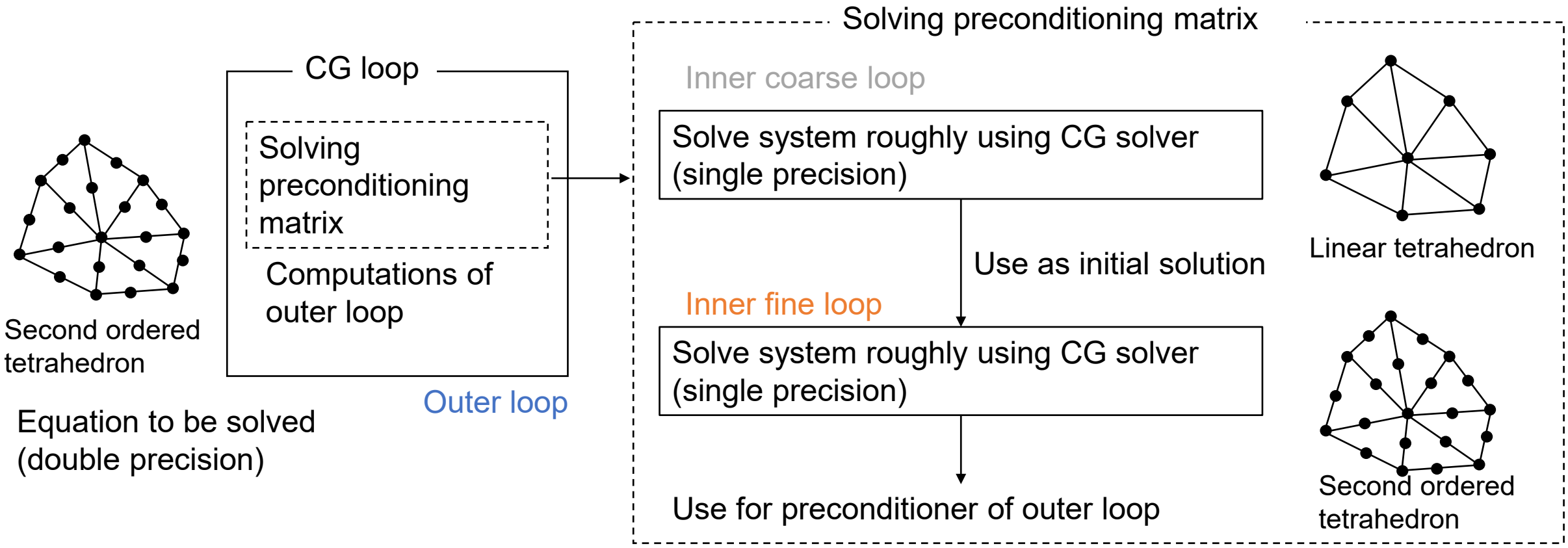
T. Ichimura et al., Implicit Nonlinear Wave Simulation with 1.08T DOF and 0.270T Unstructured Finite Elements to Enhance Comprehensive Earthquake Simulation, SC15 Gordon Bell Prize Finalist

有限要素法のポイント

- 有限要素法で数理問題を離散化することで、行列$\mathbf{A}$はスパース(疎)となる
  - 対称行列の行数をNとしたとき、
  - →行列の非零成分の数は$O(N)$になる
  - →行列ベクトル積のコスト・メモリ使用量は$O(N)$になる

- 行列ベクトル積コストが小さいため、行列ベクトル積を繰り返し使う反復法を使って解くことが多い
  - 反復法の前処理と組み合わせてアルゴリズムを構築

ソルバー例1：地震時の地盤増幅解析@京コンピュータ



- Solve preconditioning matrix roughly to reduce number of CG loops
  - Use geometric multi-grid method to reduce cost of preconditioner
  - Use single precision in preconditioner to reduce computation & communication

T. Ichimura et al., Implicit Nonlinear Wave Simulation with 1.08T DOF and 0.270T Unstructured Finite Elements to Enhance Comprehensive Earthquake Simulation, SC15 Gordon Bell Prize Finalist

前回までのまとめと今回の内容

- 有限要素法で数理問題を離散化することで、行列$\mathbf{A}$はスパース(疎)となる
- 行列ベクトル積コストが小さいため、行列ベクトル積を繰り返し使う反復法を使って解くことが多い

- いかに速く行列ベクトル積を実行できるか？
  →計算機アーキテクチャに沿った開発（CPU向け・GPU向け）
- そもそも行列ベクトル積回数を削減することはできないか？
  →データ駆動型手法との融合

SIMDに適した有限要素法アルゴリズム開発・実装の例

Kohei Fujita, Masashi Horikoshi, Tsuyoshi Ichimura, Larry Meadows, Kengo Nakajima, Muneo Hori, Lalith Maddegedara

Journal of Computational Science, 2020より

コア内並列

- SIMD (single instruction multiple data)
  - 一つの命令で複数のデータを操作
    - 命令数を減らすことができる
    - データが不連続であったり，データごとに異なる操作をする処理には適さない
  - 例：Intel AVX-512 (512 bit SIMD, 倍精度浮動小数点数8つをまとめて操作)
    - a512=_mm 512 _loadu_pd(&a[0]); // 連続した8要素をメモリからレジスタにロード
    - b512=_mm512_loadu_pd(&b[0]); // 連続した8要素をメモリからレジスタにロード
    - c512=a512+b512; // 8要素を加算
    - _mm512_storeu_pd(&c[0],c512); // 8要素をメモリの連続領域にストア
    - 参考： https://www.intel.co.jp/content/www/jp/ja/architecture-and-technology/avx-512-overview.html

ターゲット問題

- 陰解法による動的非線形低次非構造格子有限要素法
  - 都市の地震応答など、不均質な分布を持つ非線形物性・複雑形状を持つ領域の求解に適している
  - 多数回大規模線形連立方程式を求解する
    - 多数のランダムデータアクセスを含む

Solve for each of few thousand time steps:

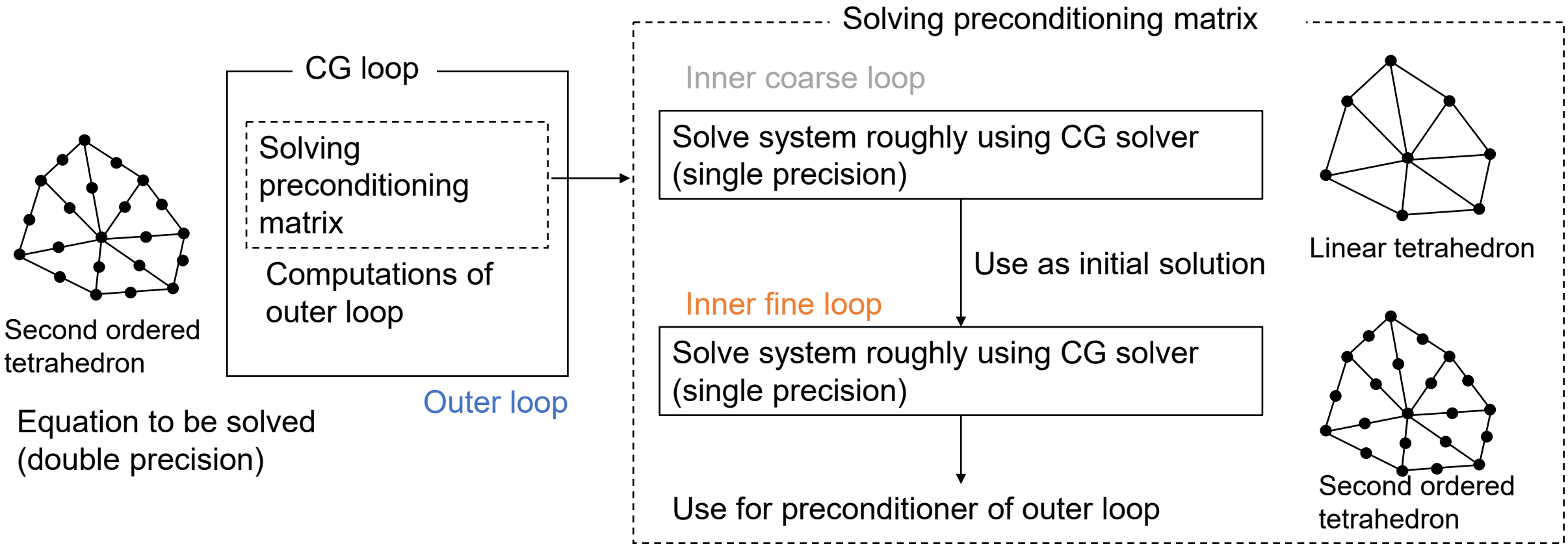
Unknown vector (up to trillion degrees of freedom)

$$\left(\frac{4}{dt^2}\mathbf{M}+\frac{2}{dt}\mathbf{C}^n+\mathbf{K}^n\right)\delta\mathbf{u}^n=\mathbf{F}^n-\mathbf{Q}^{n-1}+\mathbf{C}^n\mathbf{v}^{n-1}+\mathbf{M}\left(\mathbf{a}^{n-1}+\frac{4}{dt}\mathbf{v}^{n-1}\right)$$

Sparse symmetric positive definite matrix (changes every time step)

Known vector

ソルバー例1：地震時の地盤増幅解析@京コンピュータ



- Solve preconditioning matrix roughly to reduce number of CG loops
  - Use geometric multi-grid method to reduce cost of preconditioner
  - Use single precision in preconditioner to reduce computation & communication

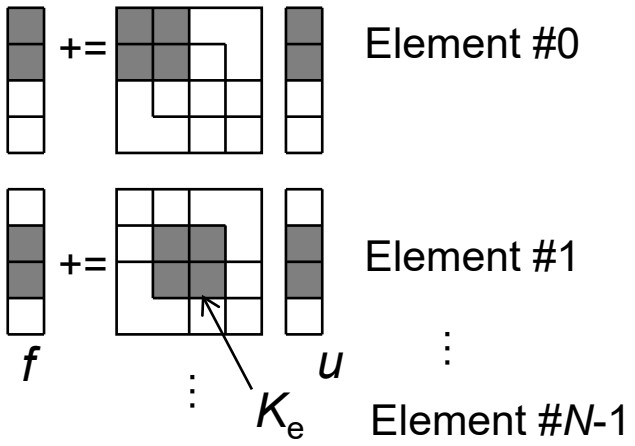
T. Ichimura et al., Implicit Nonlinear Wave Simulation with 1.08T DOF and 0.270T Unstructured Finite Elements to Enhance Comprehensive Earthquake Simulation, SC15 Gordon Bell Prize Finalist

計算機におけるSC14ソルバーの性能

- 京コンピュータ(理化学研究所)
  - 8コアCPU(SPARC64) x 82,944計算ノード
  - ピーク性能：10.6 PFLOPS、メモリバンド幅：5.3 PB/s
  - SC14ソルバーの性能：ピーク性能の**11.1%**
- Oakforest-PACS (東京大学・筑波大学)
  - 68コアCPU(Xeon Phi Knights Landing) x 8,208計算ノード
  - ピーク性能：25 PFLOPS、メモリバンド幅：4 PB/s
  - SC14ソルバーの性能：ピーク性能の**2.26%**

性能劣化の原因

- 計算機によりSIMD幅が異なる
  - 京コンピュータ: 倍精度演算で2 (単精度演算で2)
  - Oakforest-PACS:倍精度演算で16 (単精度演算で8)
- Element-by-Element法におけるランダムアクセスがwide SIMD計算機においてボトルネックとなる
  - 右辺ベクトルのランダムロード(*u*)
  - 左辺ベクトルへのランダム足しこみ(*f*)
  - これらはSIMDによる連続アクセスよりも低効率な命令で実装される

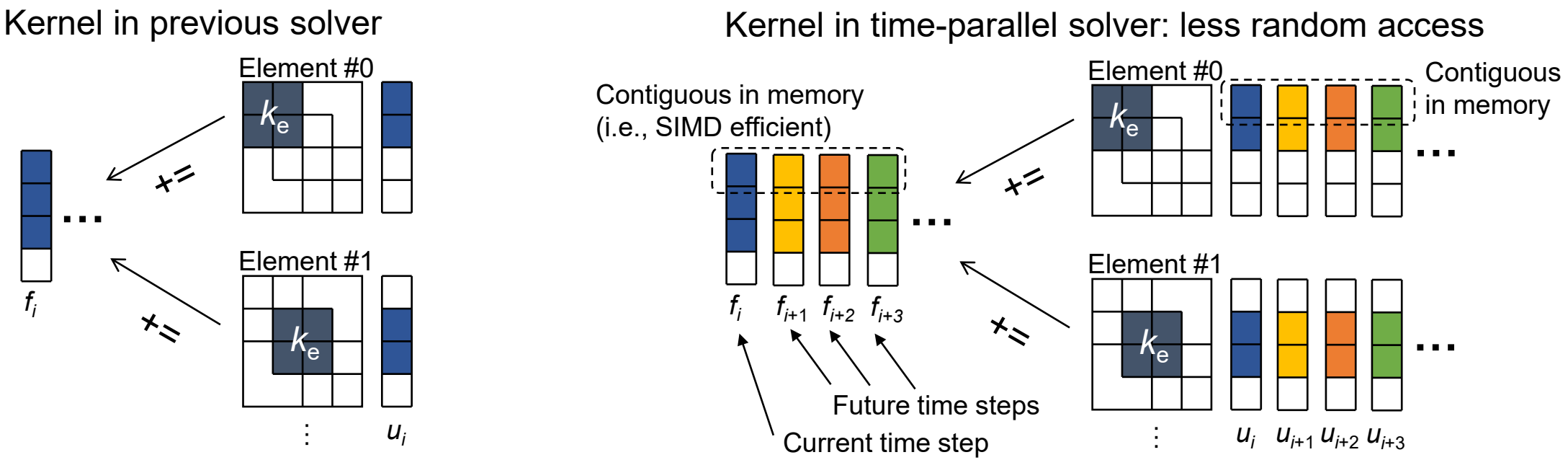


非構造有限要素法における、問題の「均質性」の利用

- 計算の均質性・連続性→高いデータアクセス性能に直結する
  - 差分法などの構造格子や、メッシュが構造化されている手法において高い計算性能が得られる理由の一つ
- 非構造有限要素においても、メッシュは時間方向に不変
  - この特性を使って、時間方向に並列に求解計算を実施することで、動的有限要素法の効率を改善

時間並列アルゴリズムの概要

- 複数の時間ステップを同時に求解することで、将来ステップの解を予測
  - 一反復当たりの計算コストを削減
  - 通常のソルバーを使った場合と全演算数はほぼ変わらないが、より高効率なカーネルを使用することができる



Tsuyoshi Ichimura, Kohei Fujita, Masashi Horikoshi, Larry Meadows, Kengo Nakajima, Takuma Yamaguchi, Kentaro Koyama, Hikaru Inoue, Akira Naruse, Keisuke Katsushima, Muneo Hori, Maddegedara Lalith, A Fast Scalable Implicit Solver with Concentrated Computation for Nonlinear Time-evolution Problems on Low-order Unstructured Finite Elements, 32nd IEEE International Parallel and Distributed Processing Symposium, 2018.

Kohei Fujita, Keisuke Katsushima, Tsuyoshi Ichimura, Masashi Horikoshi, Kengo Nakajima, Muneo Hori, Lalith Maddegedara, Wave Propagation Simulation of Complex Multi-Material Problems with Fast Low-Order Unstructured Finite-Element Meshing and Analysis, Proceedings of HPC Asia 2018 (**Best Paper Award**), 2018.

Standard solver algorithm

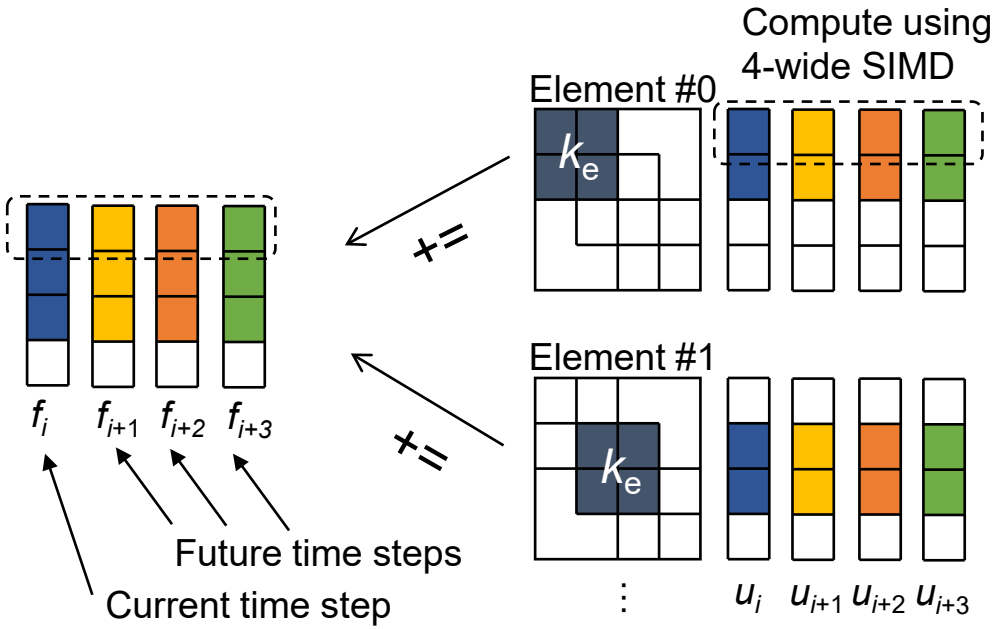
1: set $x_{-1} \leftarrow 0$
2: for($i = 0; i < n; i = i + 1$){
3: guess $\bar{x}_i$ using standard predictor
4: set b_i using x_{i-1}
5: solve $x_i \leftarrow A^{-1}b_i$using initial solution $\bar{x}_i$ **(Computed using iterative solver with SpMV kernel)**
6: }

Developed algorithm

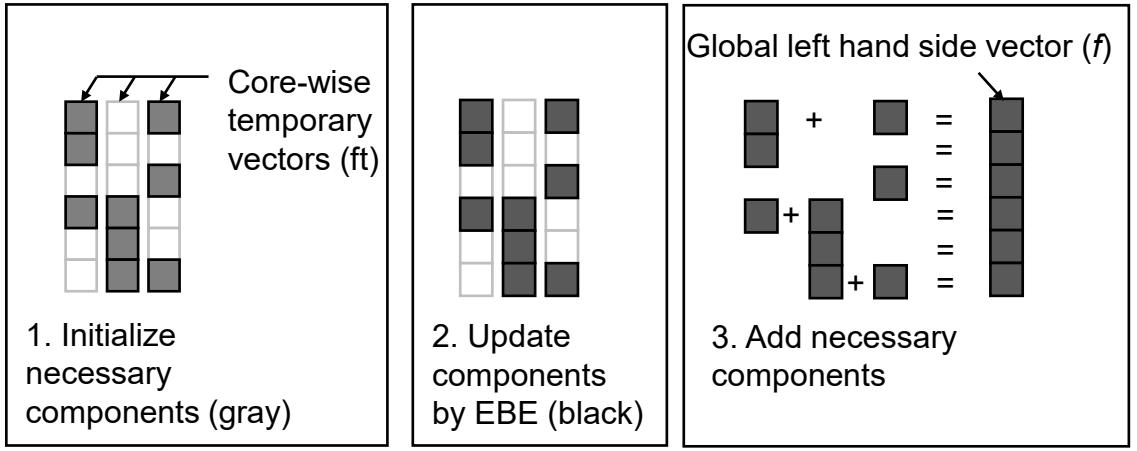
1: set $x_{-1} \leftarrow 0$ and $\bar{x}_i \leftarrow 0$ ($i = 0, \dots, m-2$)
2: for($i = 0; i < n; i = i + 1$){
3: set b_i using x_{i-1}
4: guess $\bar{x}_{i+m-1}$ using standard predictor
5: $b_j \leftarrow \bar{b}_j$
6: while ($\frac{|\mathrm{A}\bar{x}_i - b_i|}{|b_i|} > \epsilon$) do {
7: guess $\bar{b}_j$ using $\bar{x}_{j-1}$ ($j = i+1, \dots, i+m-1$)
8: refine solution $\{\bar{x}_j \leftarrow A^{-1}\bar{b}_j\}$ with initial solution $\bar{x}_j$ $(j = i, \dots, i+m-1)$ **(Computed using iterative solver with concentrated computation kernel)**
10: }
11: $x_i \leftarrow \bar{x}_i$
11: }

並列時間積分アルゴリズムのnaïveな実装 (*m*-ステップ)

- 各コアにてSIMDを用いて*m*本のベクトルを計算
  - ただし、実際の問題では*m* ≦4が一般的：SIMD幅をすべて使い切ることができない
- マルチコア間のデータ競合を回避するためにテンポラリのベクトルを確保
  - メニーコア計算機を使う場合は高コスト



Core-wise computation (in case of *m*=4)

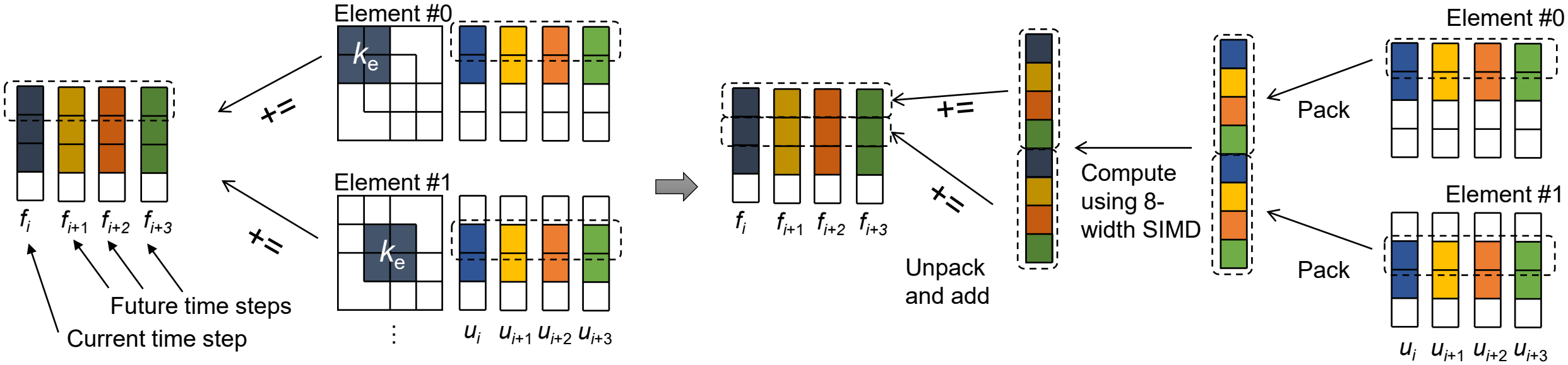


Many-core computation (in case of three cores)

Wide-SIMD CPUにおける効率的な計算方法

- ベクトルをパック・アンパックすることでSIMD幅をすべて使う

Example for time parallel kernel (*m* = 4) with 8 width FP32 SIMD architecture

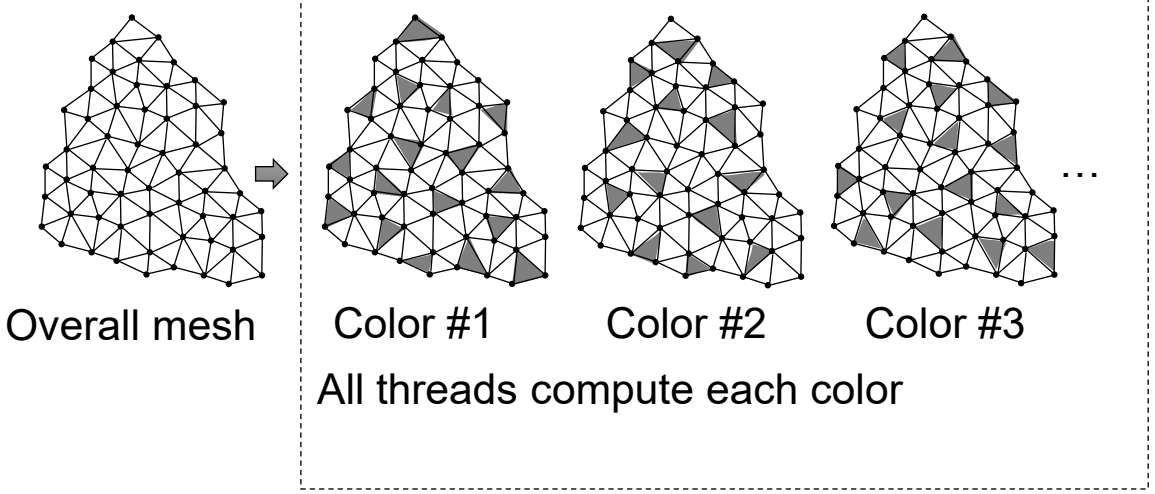


Naïve implementation: use only 4 out of the 8-width SIMD

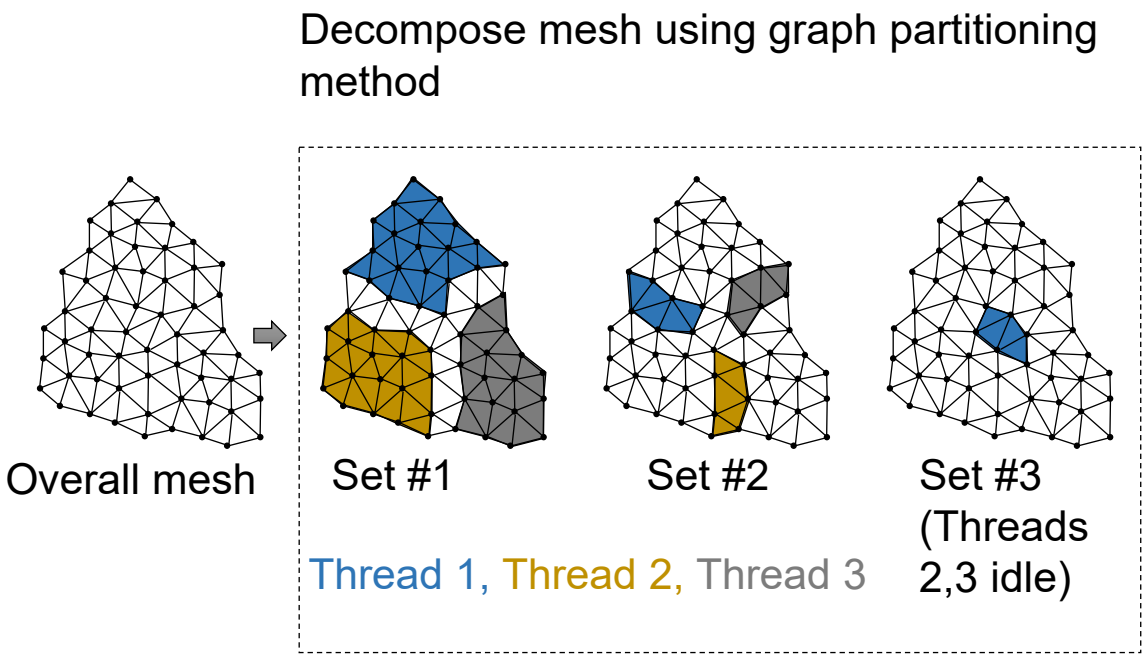
With packing: can use full SIMD width

メニーコア計算機向けのスレッド分割

- テンポラリ配列を使う必要がない
- グラフ分割法を使ってキャッシュの再利用を促進



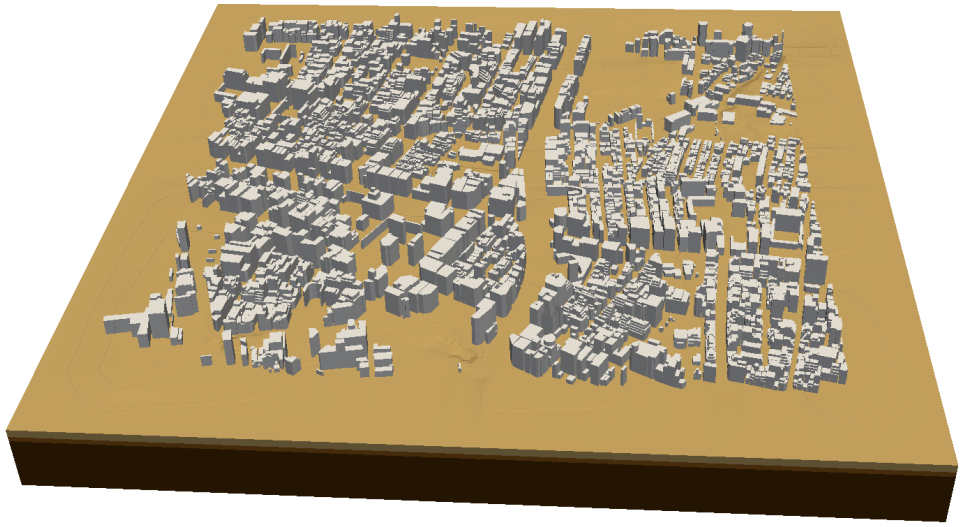
a) Standard coloring method



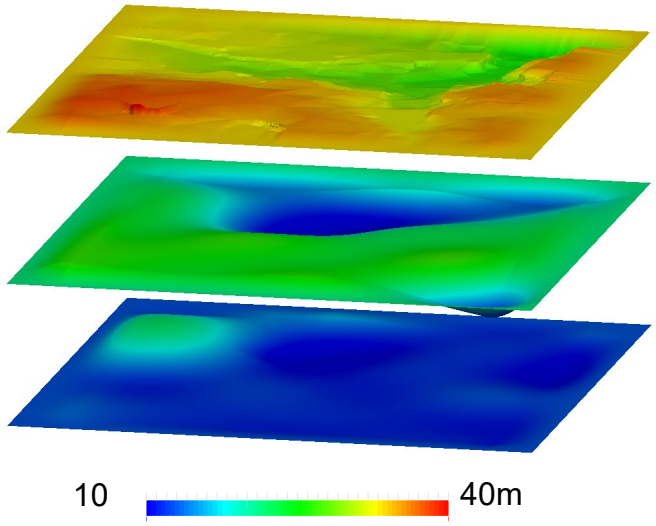
b) Developed thread partitioning method

Performance on actual urban earthquake simulation problem

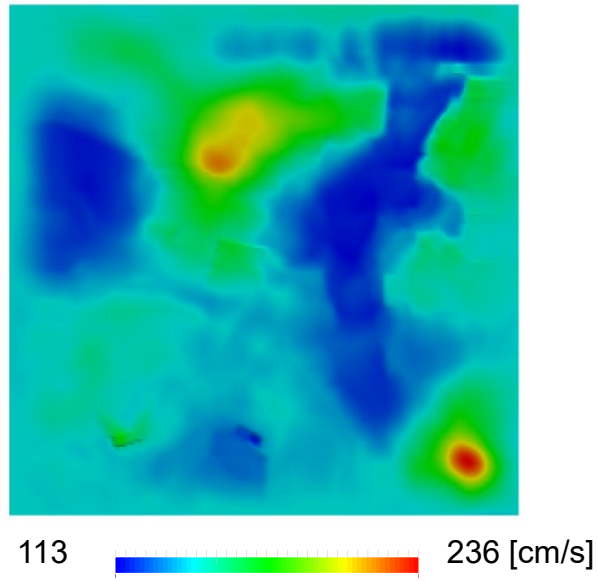
- Compute seismic shaking of 3 layered ground in central Tokyo



a) Model of 1.25 km x 1.25 km area of Tokyo with 4066 structures

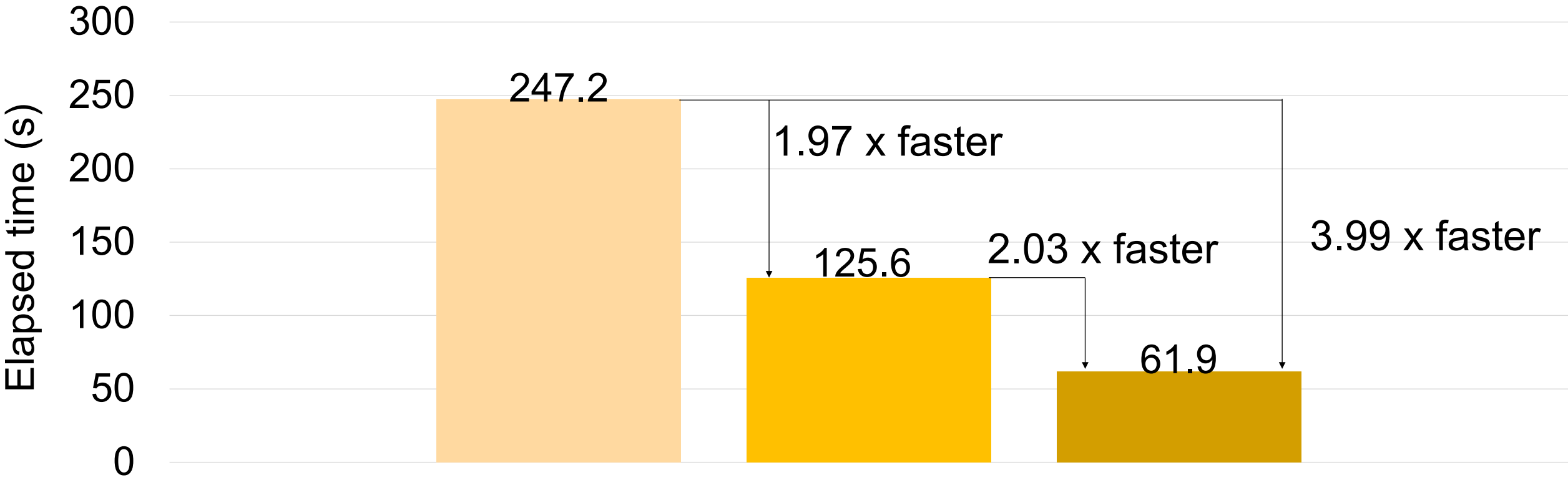


b) Elevation of interfaces of three soil layers



c) Response at ground surface (merged horizontal component of SI value)

Performance on actual urban earthquake simulation problem



Solver algorithm	SC14 solver (without time parallelism)	With time parallelism	With time parallelism
EBE kernel algorithm	Baseline (m=1)	Baseline (m=4)	Developed (m=4)

富岳向け計算手法の開発

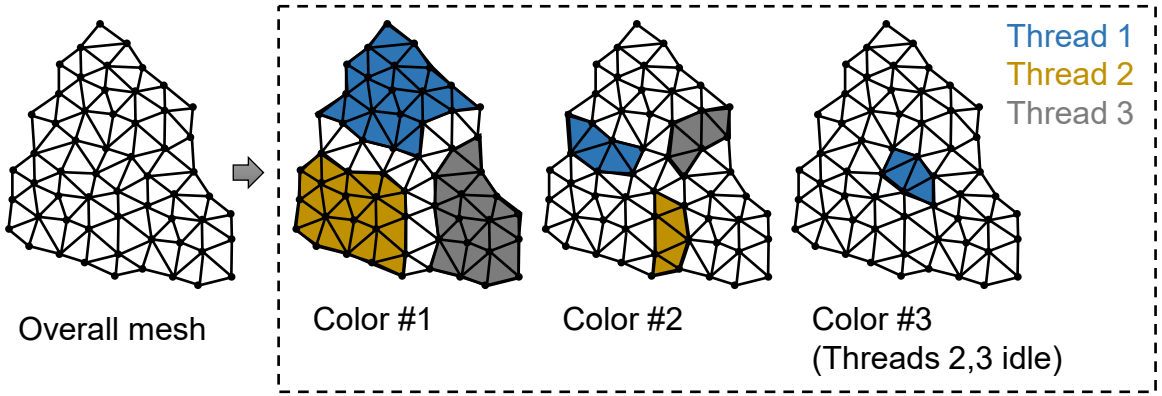
Kohei Fujita, Kentaro Koyama, Kazuo Minami, Hikaru Inoue, Seiya Nishizawa, Miwako Tsuji, Tatsuo Nishiki, Tsuyoshi Ichimura, Muneo Hori, Lalith Maddegedara, High-fidelity nonlinear low-order unstructured implicit finite-element seismic simulation of important structures by accelerated element-by-element method, Journal of Computational Science, 2020

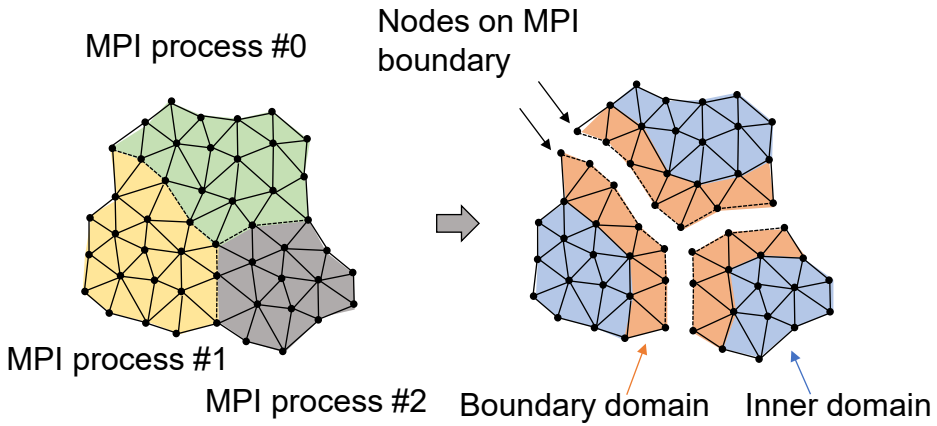
富岳上でのSC14ソルバーの性能

- 京コンピュータ(8コアCPU x 82944台)においてピーク性能の11.1%
- 富岳(48コアCPU x 158976台)においてピーク性能の1.5%
- 有限要素法においては行列ベクトル積におけるランダムデータアクセスがボトルネックになっている
  - いかにしてこれらのボトルネックを回避して有限要素法を高速化するか？
- 計算用コア数の増加、システム用のアシスタントコアの追加
  - これらも活用したい

計算機機構の活用

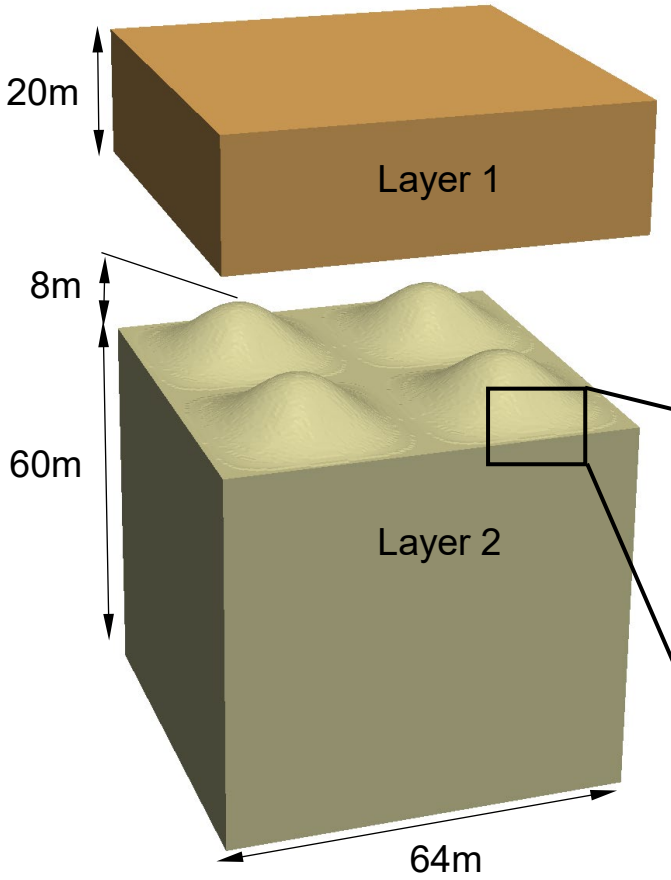
- 計算の連続アクセス化
  - SIMD演算器を有効活用するための計算の並び替え
- 多数コアの効率的活用
  - キャッシュ特性を考慮したマルチカラリング
- 計算と通信のオーバーラップ
  - アシスタントコアを活用して計算と通信を同時実行
- これらの工夫で主要計算部となる行列ベクトル積において13倍の高速化を実現



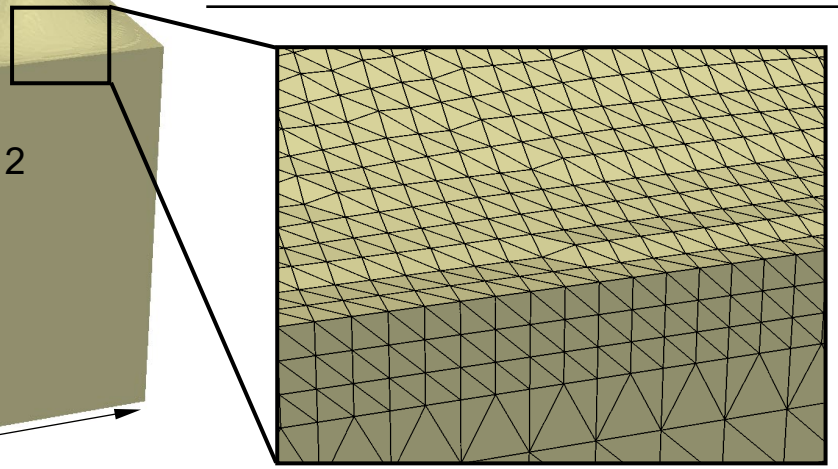


性能計測問題

- 2層の非線形地盤中の波動伝播解析

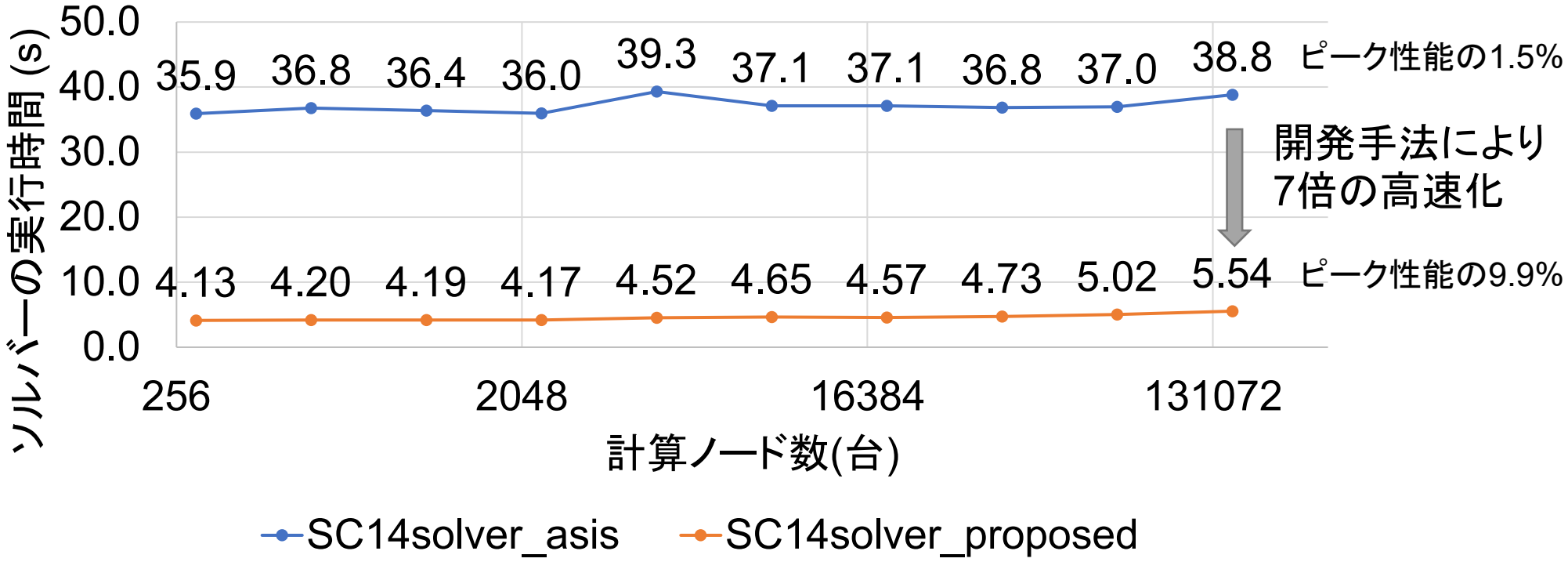


Layer	1	2
Vp (m/s)	700	2,100
Vs (m/s)	100	700
Density (kg/m^3)	1,500	2,100
Damping	0.25 (h_{max})	0.05
Strain Criteria	0.007	-



性能計測結果

- ウィークスケーリング(一台あたりの計算規模を一定として、計算機の台数を増やす)を計測：実行時間が増加しないのが理想的
- 富岳ほぼ全系に相当する147,456ノードにおいて、SC14ソルバーをそのまま実行した場合に比べて計算機機構を踏まえた手法により7倍速に
  - 京全系を使った場合の59倍速に相当



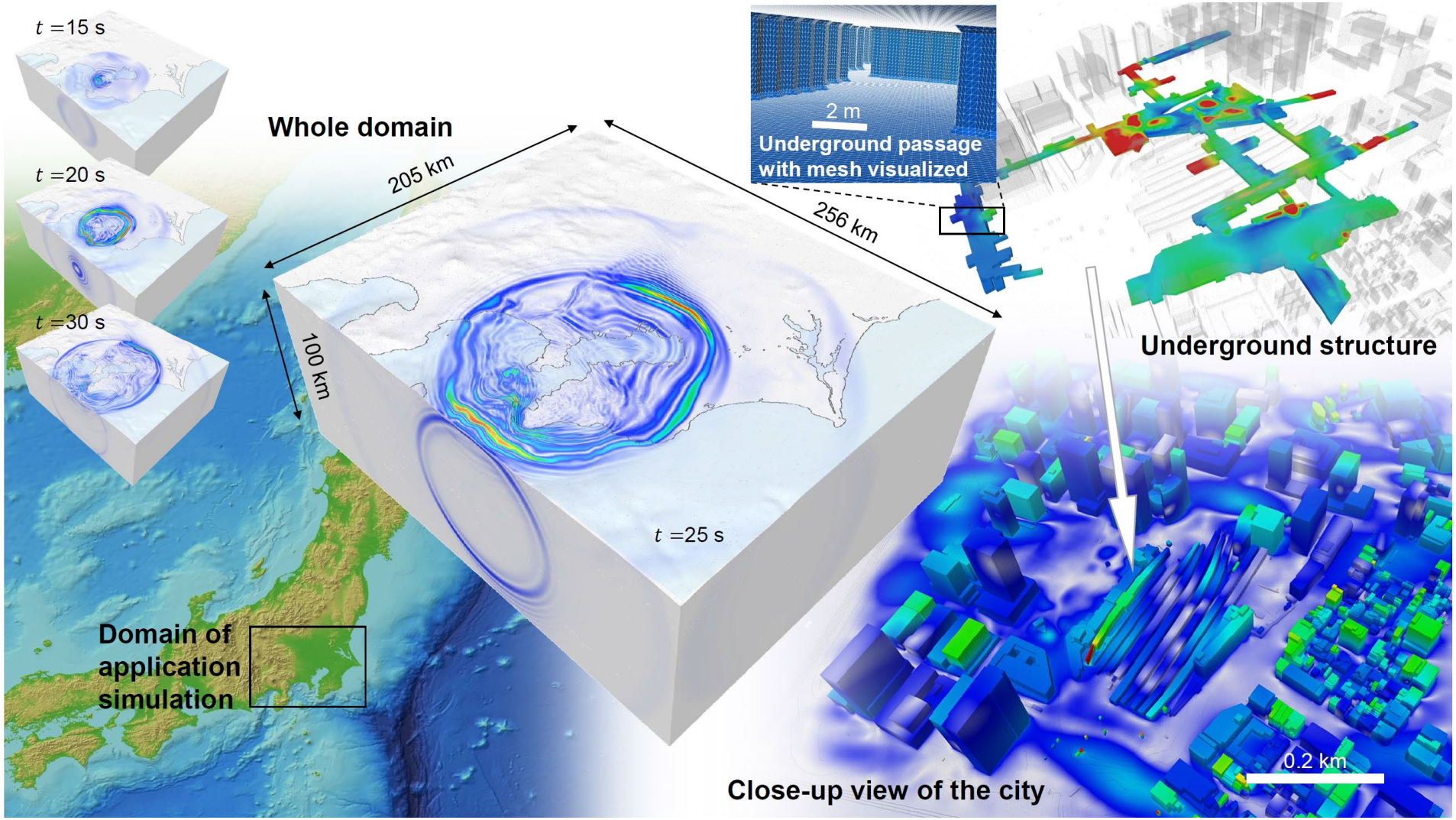
Data-driven approachとの融合

HPC Asia 2022 Best Paper: "152K-computer-node parallel scalable implicit solver for dynamic nonlinear earthquake simulation" / Tsuyoshi Ichimura and Kohei Fujita (The University of Tokyo, RIKEN); Kentaro Koyama (Fujitsu Ltd.); Ryota Kusakabe and Yuma Kikuchi (The University of Tokyo); Takane Hori and Muneo Hori (Japan Agency for Marine-Earth Science and Technology); Lalith Maddegedara (The University of Tokyo); Noriyuki Ohi, Tatsuo Nishiki, and Hikaru Inoue (Fujitsu Ltd.); and Kazuo Minami, Seiya Nishizawa, Miwako Tsuji, and Naonori Ueda (RIKEN)

断層から都市までの一体型の連成地震応答シミュレーション例

都市における構造物も陽に解像

最小要素サイズ: 12.5 cm



開発手法を富岳全系(152,352ノード(609,408 MPI processes × 12 OpenMP threads = 7,312,896コア)上で実行することでこれまでにない解像度で一体解析を実現

ターゲット問題

- 複雑形状・非均質物性領域中での非線形波動伝播シミュレーション
  - 複雑形状のモデル化のため非構造格子有限要素法が適している
  - 複雑形状の離散化のため局所的に小さい要素が生じる：極端に短い時間ステップを避けるため陰解法による求解が適している
- 陰解法による非構造格子有限要素法を活用
  - $$A^n \delta u^n = f^n \qquad (1)$$ を解くことで変位増分δu^nを求解し、変位・速度・加速度を $u^n \Leftarrow u^{n-1} + \delta u^n,\ \ v^n \Leftarrow -v^{n-1} + \frac{2}{dt}\delta u^n,\ \ \ a^n \Leftarrow -a^{n-1} - \frac{4}{dt}v^{n-1} + \frac{4}{dt^2}\delta u^n$ で更新
  - ここで$A^n \Leftarrow \left(\frac{4}{dt^2}M + \frac{2}{dt}C^n + K^n\right), f^n \Leftarrow F^n - Q^{n-1} + C^n v^{n-1} + M\left(a^{n-1} + \frac{4}{dt}v^{n-1}\right)$
- 解析コストのほぼすべてが式(1)の求解にかかるため、この式を高速・スケーラブルに求解するソルバーを開発

対象方程式の特性

- 領域サイズが$10^{5\text{-}6} \times 10^{5\text{-}6} \times 10^{4\text{-}5}$ mで最小要素サイズが10^{-1} mとなるため超大規模問題となる($10^{11\text{-}13}$自由度)
  - 省メモリ・スケーラブルな反復法ソルバーが必要
- 要素サイズ・物性が有限要素モデル内で大きく変化するため、分離型の地震シミュレーションに比べて収束性が悪化
  - 低精度演算を使いつつロバストなソルバーを設計するには工夫が必要に
- 疎行列の計算にはランダムアクセスが含まれるため、演算性能向上のためには工夫が必要となる

ソルバー設計

- 収束特性の悪い大規模問題を幅広い計算機アーキテクチャで求解可能なアルゴリズム設計
  - Equation-basedとdata-driven approachの高度な融合＋精度混合演算
  - 超並列環境でのスケーラビリティのための通信削減＋ロードバランス
- + アーキテクチャに適したカーネル設計による高効率計算

Equation-based and data-driven approach

- Equation-based approach (e.g., マルチグリッド法)
  - 低次モードを低コストで求解可能だが、高次モードの求解は高コストに
- Data-driven approach
  - 学習データサイズに応じてコストと精度が決まる(i.e., 大規模領域全域における高次モードまでの求解は超高コストとなるが、領域が小さければコストは抑えらえれる)
- これらの方法をうまく組み合わせることができないか？
  - Equation-based modelingで低次モードを求解し、data-driven methodにより小領域における高次モードを求解。これらを精度保証される求解スキーム内で活用

Data-driven initial solution estimator

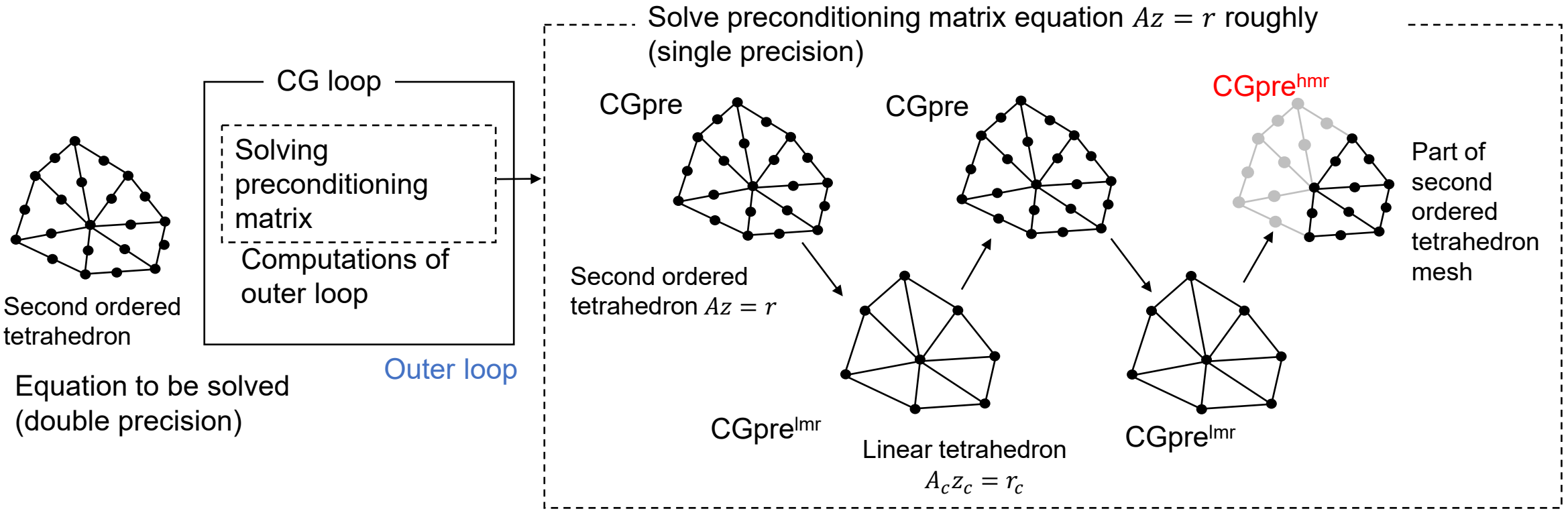
- 過去に求解したデータを使って高次モードを含む初期解を推定
  - $X^{n-1} = AX^{n-2}$により、時間発展行列A を推定。ここで、直前のsタイムステップのデータ$x^n = \delta u^n - \delta u^n_{adam}$により$X^{n-1} = [x^{n-1}, x^{n-2}, \ldots, x^{n-s}]$と設定
  - Equation-based predictor (δu^n_{adam}: Adams-Bashforth法により推定)により、data-driven methodにおいて精度低下につながるtransient-modeを求解:

$$\delta u^n_{ini} = \delta u^n_{adam} + A\left(\delta u^{n-1} - \delta u^{n-1}_{adam}\right)$$

- 計算は局所化されるため、大規模システムにおいて高いスケーラビリティが期待される
  - 時間発展行列AはMPI領域内をさらに再分割した小領域内で構築・評価可能

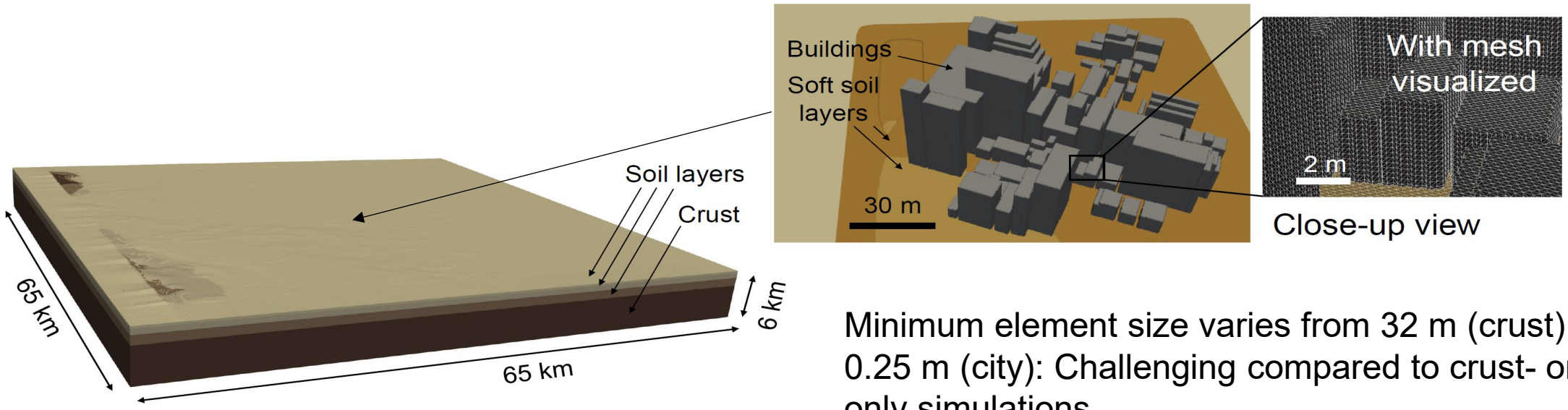
マルチグリッド法による前処理

- Data-driven methodによる初期解推定により高次モードの大部分は求解される
  - 結果として、特に収束性が悪い領域に誤差が集中
- これら誤差が大きい領域を集中的にrefineするW-cycle multi-grid preconditionerを用いる
- 富岳のメニーコア・SIMDアーキテクチャに適した行列ベクトル積カーネル(EBEカーネル)と組み合わせることで高い演算性能が期待される



性能測定問題

- 実際の地殻・地盤・構造物を模したモデルを使用
  - モデル下部からの地震動入力に対する非線形応答を求める
  - 主要動が構造物に到達する401-600ステップでのソルバー性能を測定
- 直前の16ステップでのデータを使って初期解を推定し、都市部をマルチグリッドでの選択的refinement領域として設定
- PCGE^{K} (京コンピュータ向けにSC14論文で開発されたEBEカーネルを使った共役勾配法ソルバー)と性能を比較



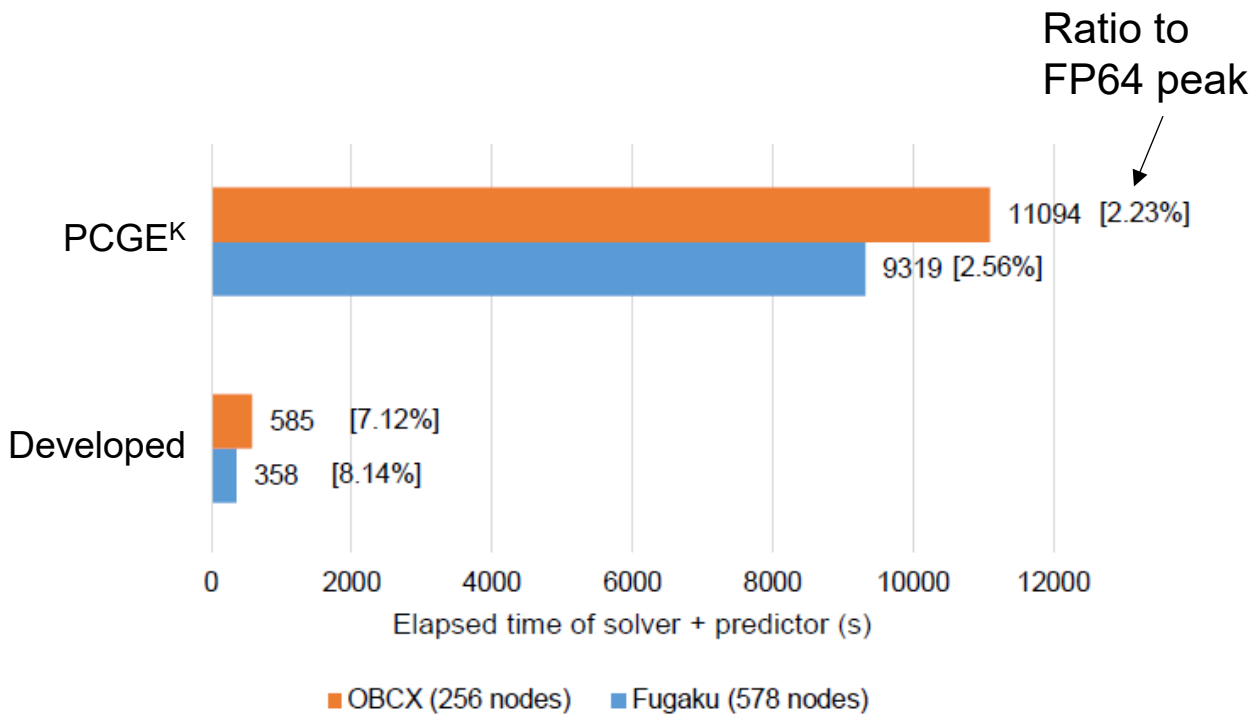
Minimum element size varies from 32 m (crust) to 0.25 m (city): Challenging compared to crust- or city-only simulations

性能測定環境

- Fugaku@Center for Computational Science, RIKEN
  - Single 48-core Fujitsu A64FX/node (3.07 TFLOPS FP64 peak & 1024 GB/s memory bandwidth)
  - Total 158,976 compute nodes
- Oakbridge-CX@Information Technology Center, The University of Tokyo
  - Dual 28-core Intel Cascade Lake Xeon/node (4.84 TFLOPS FP64 peak & 281 GB/s memory bandwidth)
  - Total 1,368 compute nodes

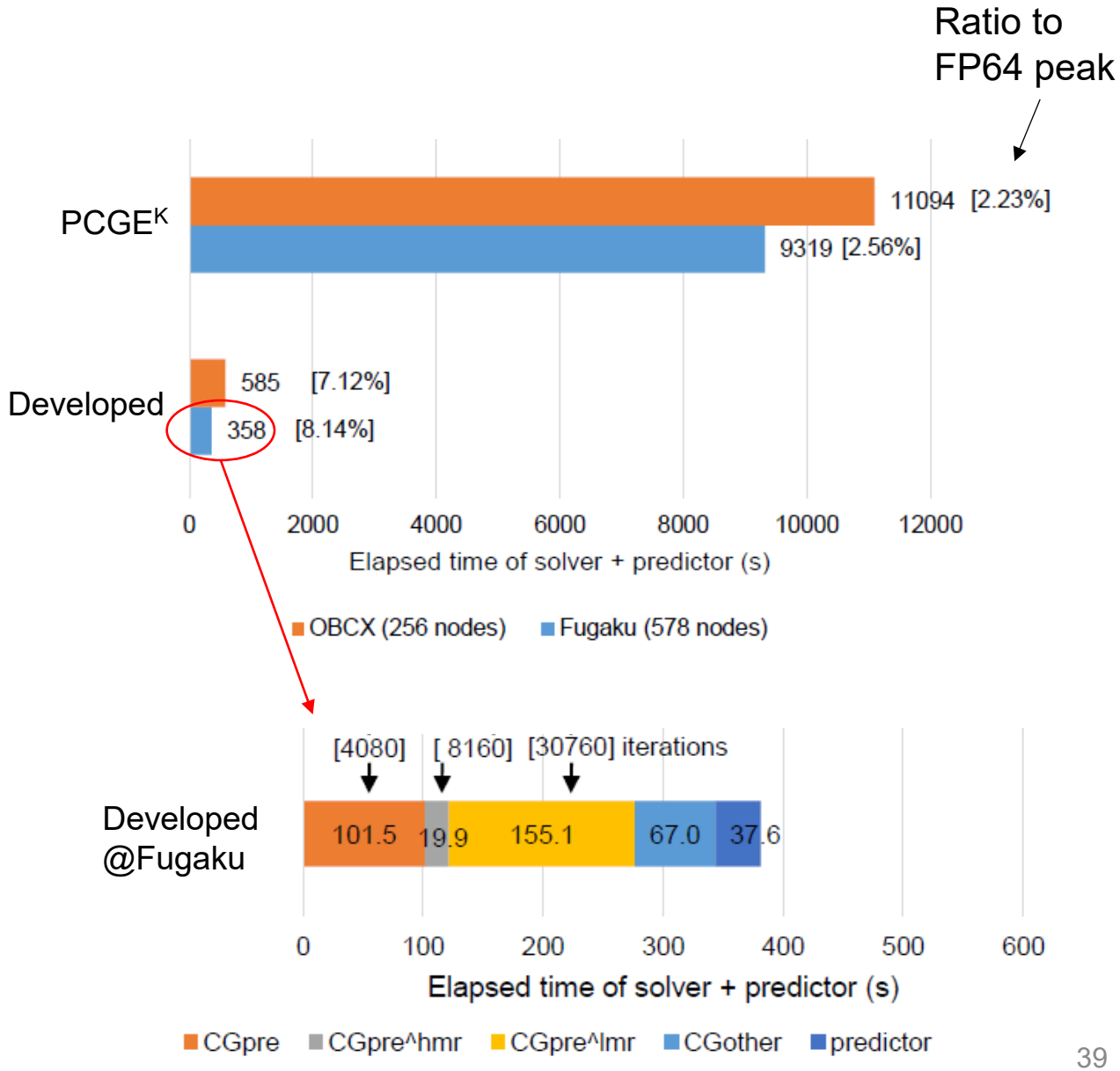
Oakbridge-CXと富岳での実行時間

- 以下の計算環境で性能計測を実施
  - 256 Oakbridge-CX compute nodes (1.24 PFLOPS FP64 peak, 72.1 TB/s memory bandwidth)
  - 578 Fugaku compute nodes (1.77 PFLOPS FP64 peak, 591 TB/s memory bandwidth)
- 従来手法比でOakbridge-CX上で18.9倍、富岳上で26.0倍の高速化を実現
  - AI＋マルチグリッド法によるソルバーアルゴリズム開発により浮動小数点演算数を1/8.16に削減
  - 計算機アーキテクチャに即したカーネル開発により浮動小数点演算効率を2%から7-8%に改善



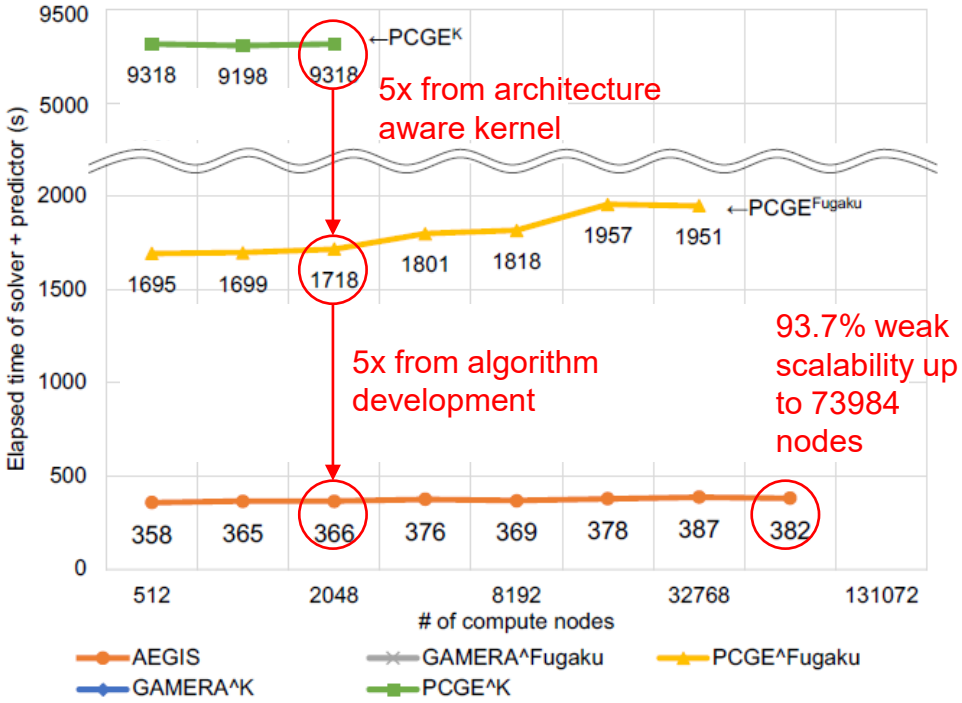
高速化の要因

- 通常の初期解推定手法(相対誤差10^{-3}, Adams-Bashforth in PCGE^K)に対し、提案手法により初期解の誤差が10^{-5}まで低減
  - 反復数がPCGEの62345回から 4080 + 8160 + 30760 = 43000回に削減
- W-cycle multi-gridによる反復あたりのコスト削減
  - Cgpre^{lmr}の自由度は元のメッシュの1/8、CGpre^{hmr}の自由度は元のメッシュの1/11.5
- 適切なカーネルアルゴリズム構築により、浮動小数点演算性能も改善
  - EBE@PCGEでは2.6% of FP64 peak
  - EBE@developed solverでは12.3% of FP64 peak



富岳上でのweak scaling

- 大規模問題の計測においてはPCGE^{K} は遅すぎるため、PCGE^{Fugaku} (本研究で富岳用に開発したEBEカーネルを使ったPCGE)と性能を比較
  - PCGE^{K}からの高速化率(25.4倍)のうち、5倍がカーネル開発、5倍がソルバーアルゴリズムから
- 富岳73984ノード(1.2兆自由度)までウィークスケーリング効率93.7%を実現
  - ほぼ一定の反復数＋均質なロードバランス＋通信削減により高いスケーラビリティを実現
  - 1.2兆自由度 x 16時間ステップ = 19.2兆自由度のデータを使った学習・推定により高い高速化率を実現



# of compute nodes	AEGIS			PCGE
	# of CG_{pre} Iterations	# of CG_{pre}^{hmr} iterations	# of CG_{pre}^{lmr} iterations	total # of iterations
578	4080	8160	30760	62345
18496	4150	8300	31332	62454
36992	4180	8360	31208	63169

GPU向けの有限要素法の高速化：低精度演算の活用

Tsuyoshi Ichimura, Kohei Fujita, Takuma Yamaguchi, Akira Naruse, Jack C. Wells, Thomas C. Schulthess, Tjerk P. Straatsma, Christopher J. Zimmer, Maxime Martinasso, Kengo Nakajima, Muneo Hori, Lalith Maddegedara

SC18 Gordon Bell Prize Finalist

より（資料：山口拓真氏提供）

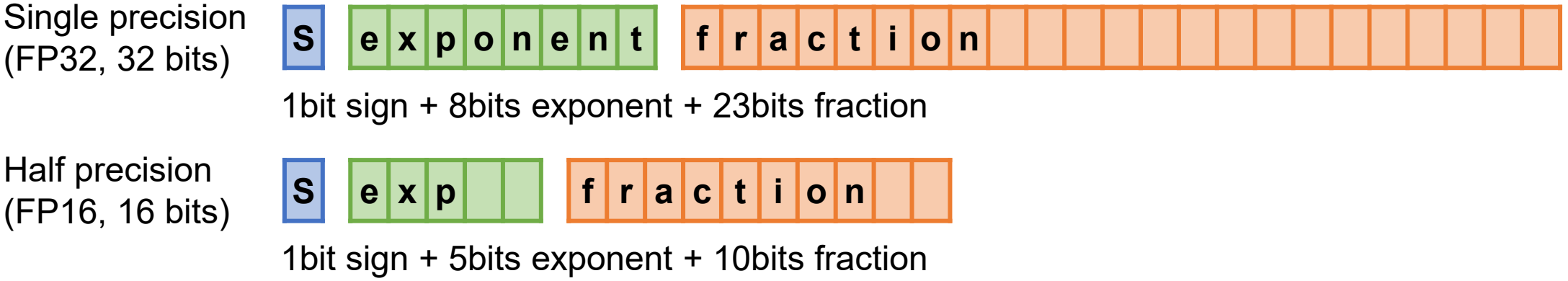
Porting to GPU systems

- Communication & memory bandwidth relatively lower than K computer
- Reducing data transfer required for performance
  - We have been using FP32-FP64 variables
  - Transprecision computing is available due to adaptive preconditioning

	K computer	**Summit**
CPU/node	1 × SPARC64 VIIIfx	2 × IBM POWER 9
GPU/node	-	6 × NVIDIA V100 GPU
Peak FP32 performance/node	0.128 TFLOPS	93.6 TFLOPS
Memory bandwidth/node	64 GB/s	5400 GB/s
Inter-node throughput	5 GB/s in each direction	25 GB/s

Introduction of FP16 variables

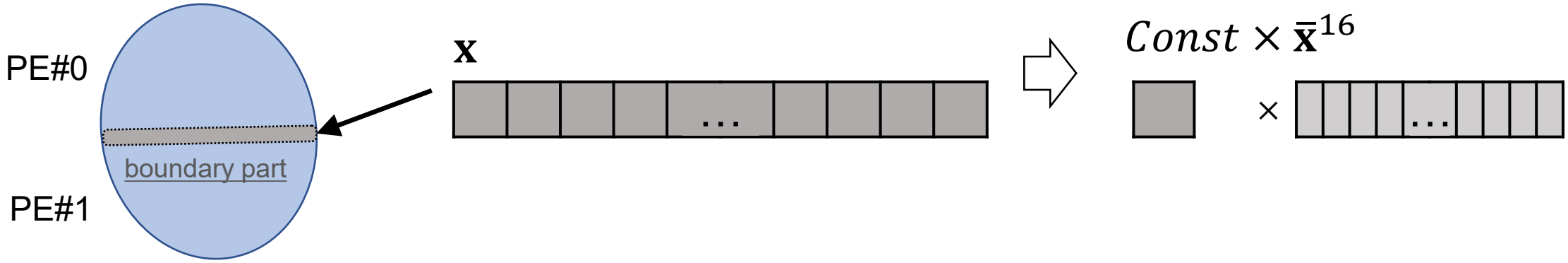
- Half precision can be used for reduction of data transfer size



- Using FP16 for whole matrix or vector causes overflow/underflow or fails to converge
  - Smaller exponent bits → small dynamic range
  - Smaller fraction bits → no more than 4-digit accuracy

FP16 for point-to-point communication

- FP16 MPI buffer only for boundary part
  - To avoid overflow or underflow, Original vector $\mathbf{x}$ is divided into one localized scaling factor $Const$ and FP16 vector $\bar{\mathbf{x}}^{16}$
- Data transfer size can be reduced
- $Const \times \bar{\mathbf{x}}^{16}$ does not match $\mathbf{x}$ exactly, but convergence characteristic is not changed for most problems



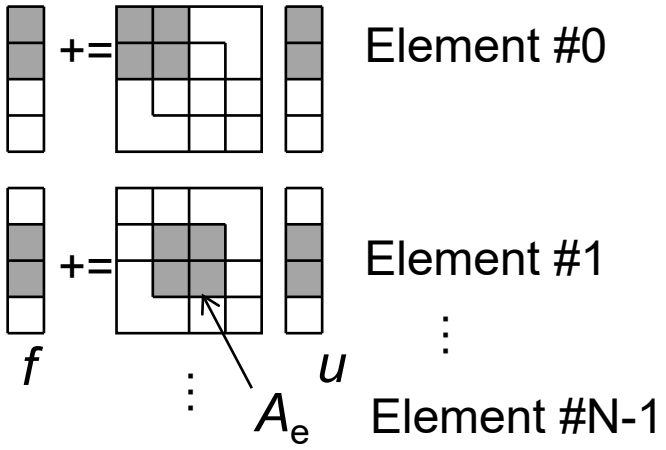
FP16 computation in Element-by-Element method

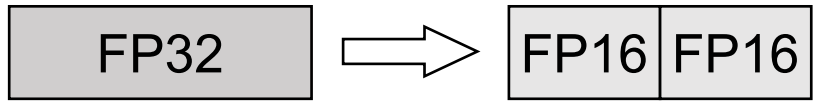
- Matrix-free matrix-vector multiplication
  - Compute element-wise multiplication
  - Add into the global vector
- Normalization of variables per element can be performed
  - Enables use of doubled width FP16 variables in element wise computation
  - Achieved 71.9% peak FP64 performance on V100 GPU
- Similar normalization used in communication between MPI partitions for FP16 communication

Element-by-Element (EBE) method

$f = \Sigma_e P_e A_e P_e^{\mathsf{T}} u$

[A_e is generated on-the-fly]





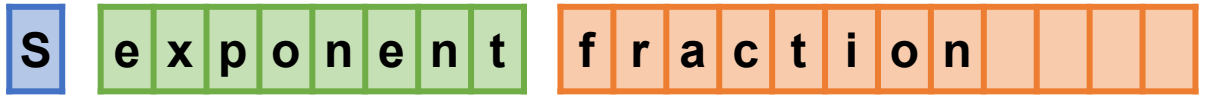
Introduction of custom data type: FP21

- Most computation in CG loop is memory bound
  - However, exponent of FP16 is too small for use in global vectors
- Use FP21 variables for memory bound computation
  - Only used for storing data (FP21 × 3 are stored into 64bit array)
  - Bit operations used to convert FP21 to FP32 variables for computation

Single precision
(FP32, 32 bits)

1bit sign + 8bits exponent + 23bits fraction

(FP21, 21 bits)



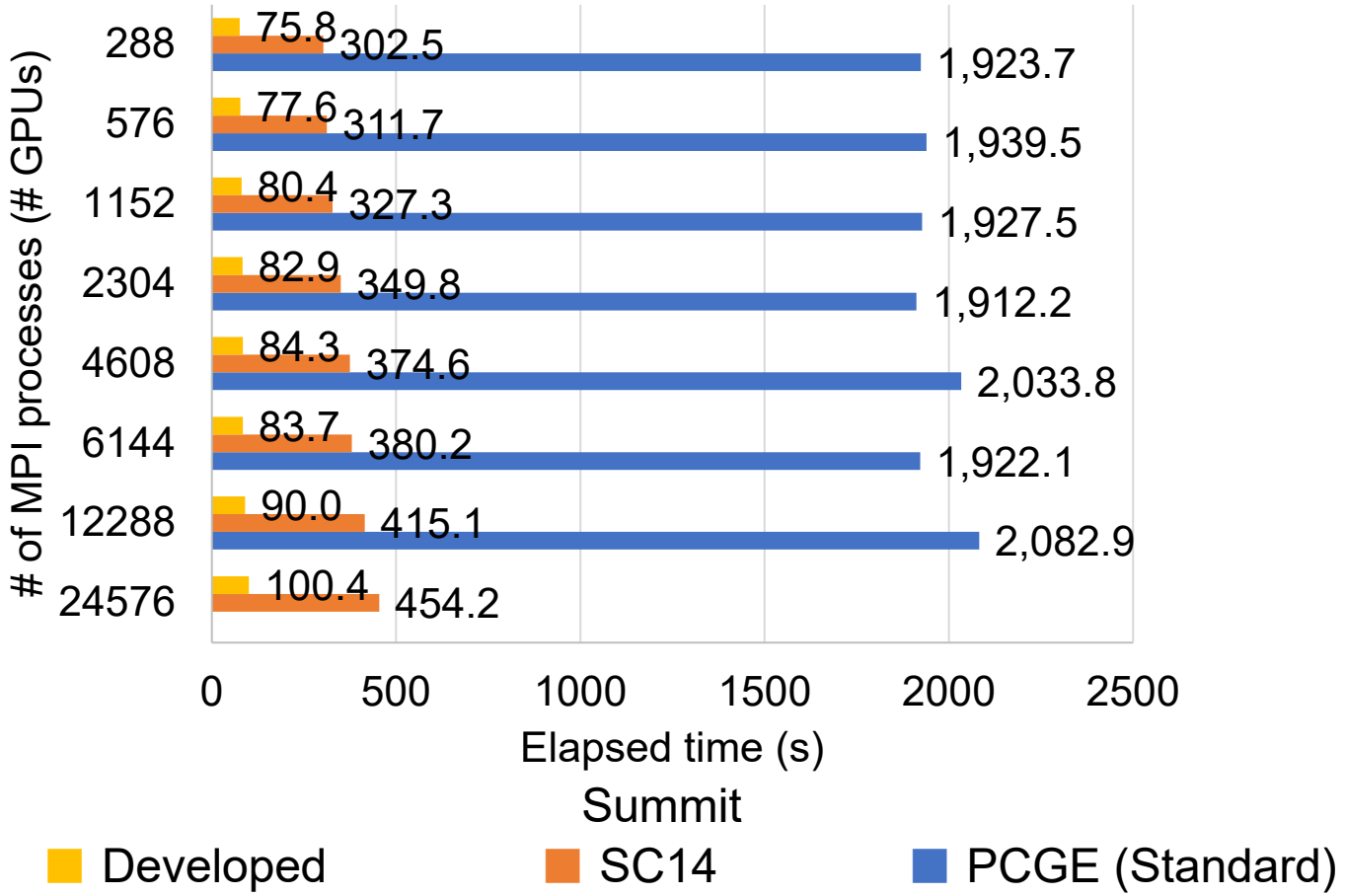
1bit sign + 8bits exponent + 12bits fraction

Half precision
(FP16, 16 bits)

1bit sign + 5bits exponent + 10bits fraction

Performance on Summit

- Developed solver demonstrates higher scalability compared to previous solvers
- Leads to 14.7% (nearly full Summit) peak FP64 performance



非構造格子有限要素法における Tensor Coreの活用

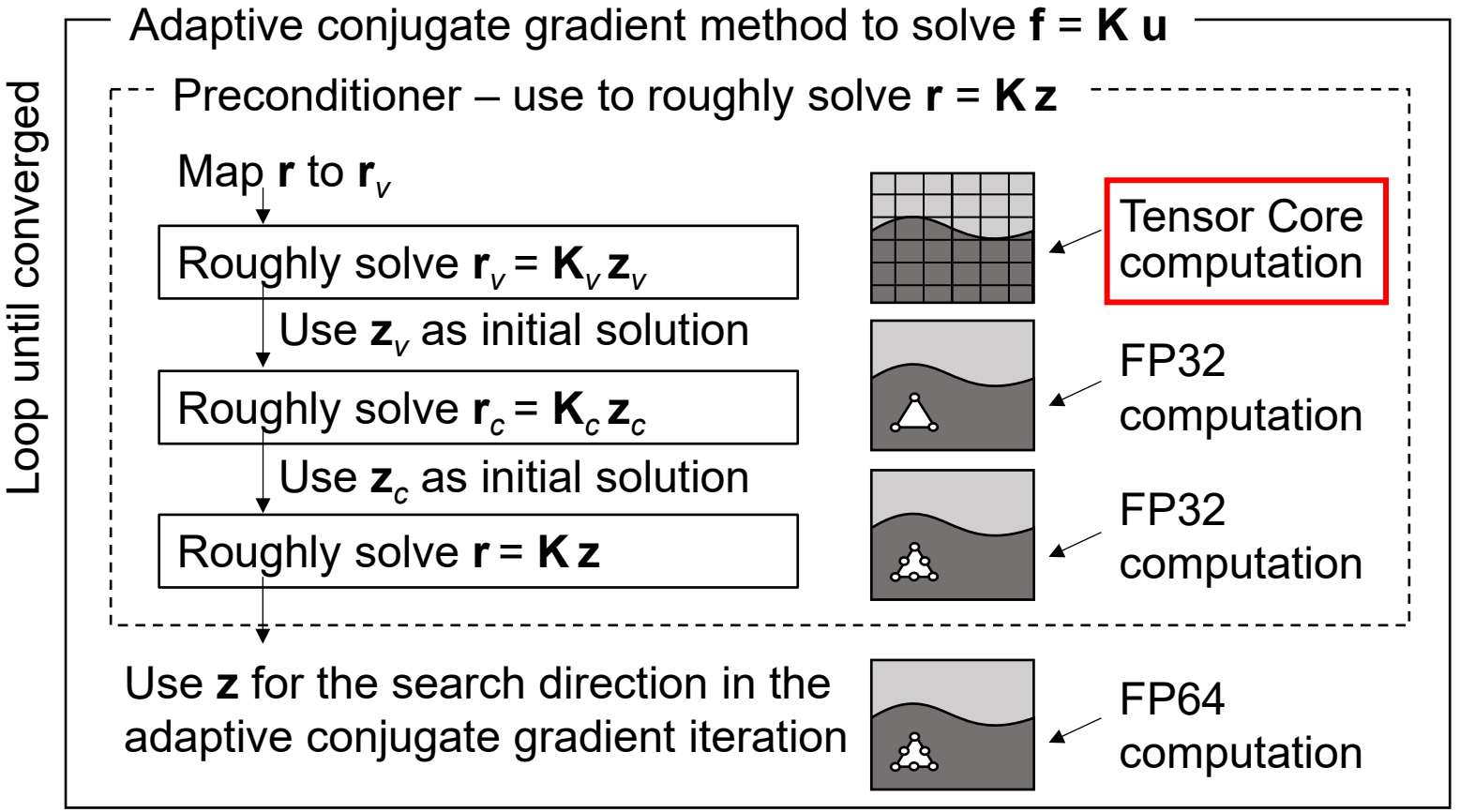
Tsuyoshi Ichimura, Kohei Fujita, Takuma Yamaguchi, Akira Naruse, Jack C. Wells, Christopher J. Zimmer, Tjerk P. Straatsma, Takane Hori, Simone Puel, Thorsten W. Becker, Muneo Hori, and Naonori Ueda.

416-PFLOPS fast scalable implicit solver on low-ordered unstructured finite elements accelerated by 1.10-ExaFLOPS kernel with reformulated AI-like algorithm: For equation-based earthquake modeling

SC19 Research Poster

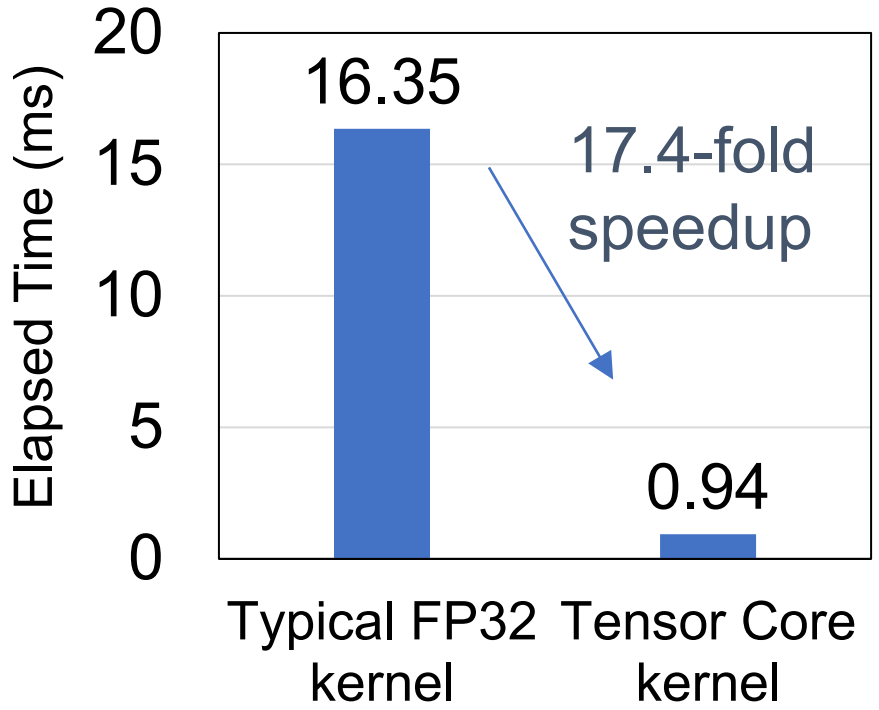
非構造格子有限要素法におけるTensor Coreの活用

- NVIDIA GPUに備わるTensor Coreを使うことができれば、行列行列積を超高速計算可能になる
- その一方で、非構造格子有限要素法の主要計算は疎行列計算になるため、Tensor Coreをそのまま適用することは難しい
- 前処理に構造格子を使う非構造格子有限要素法を開発することでTensor Coreを活用

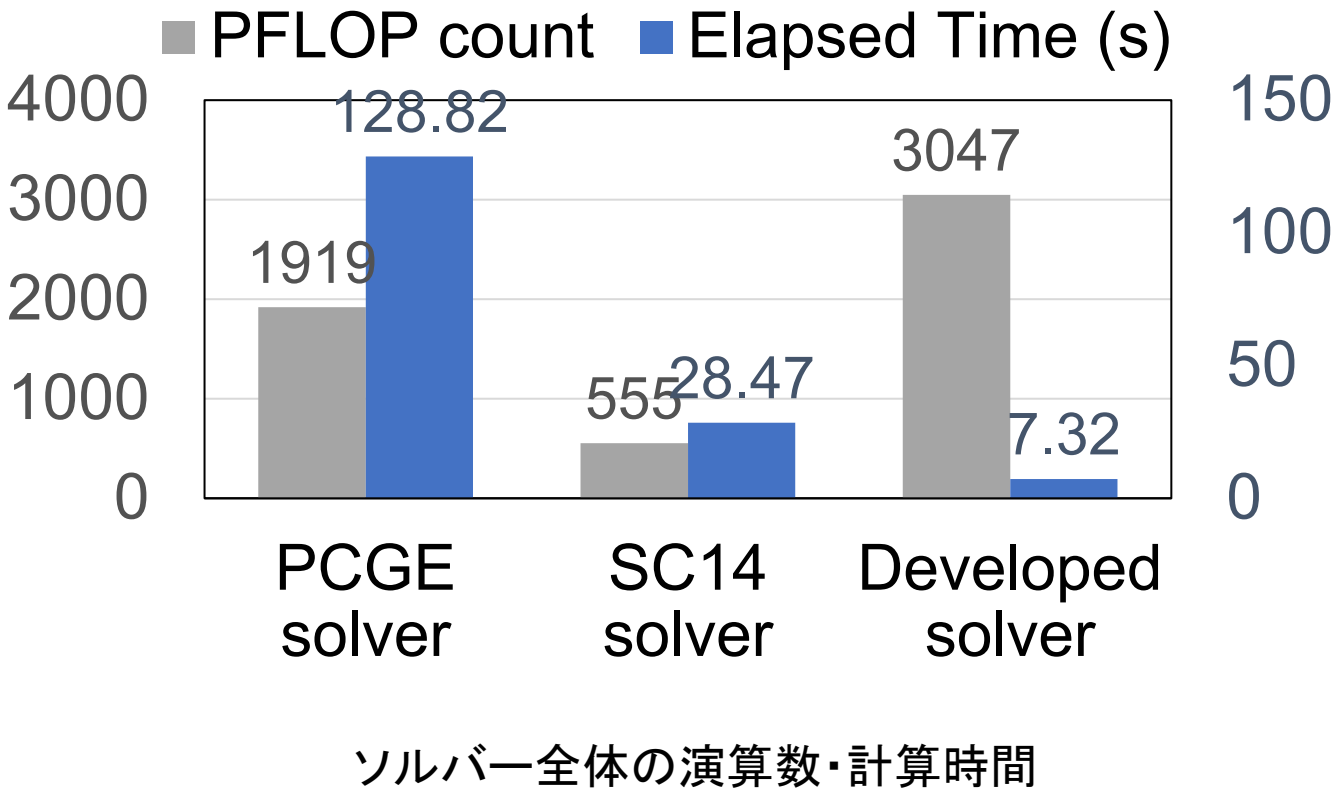


非構造格子有限要素法におけるTensor Coreの活用@Summit

- ソルバー全体の演算数は増加するものの、Tensor Coreによる高速演算により高速化



構造格子による前処理部の疎行列ベクトル積計算



ソルバー全体の演算数・計算時間

CPU-GPU協調解析

Tsuyoshi Ichimura, Kohei Fujita, Muneo Hori, Lalith Maddegedara, Jack Wells, Alan Gray, Ian Karlin, and John Linford. 2025.

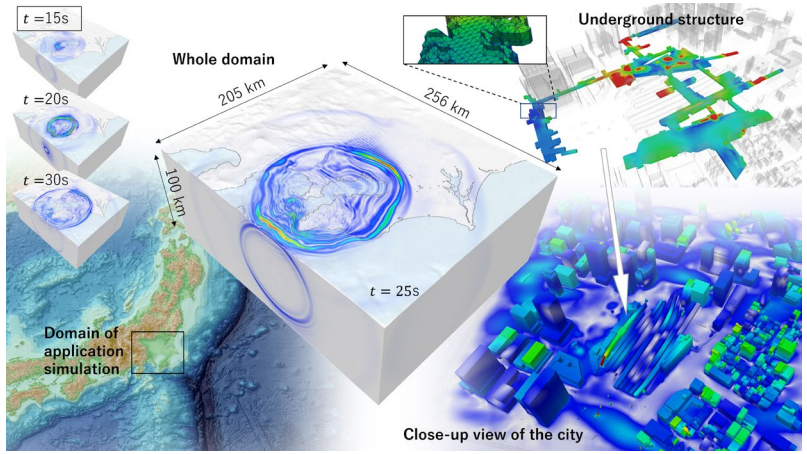
Heterogeneous computing in a strongly-connected CPU-GPU environment: fast multiple time-evolution equation-based modeling accelerated using data-driven approach.

In Proceedings of the SC24 Workshops: WACCPD Eleventh Workshop on Accelerator Programming and Directives

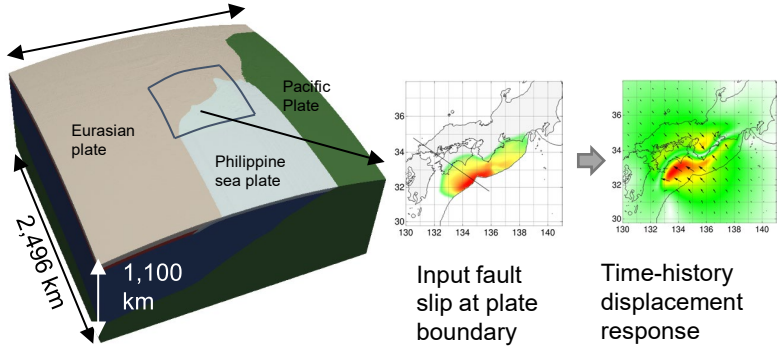
データ駆動型手法を用いた高速多数回波動場シミュレーション@CPUとGPUによるヘテロジニアス環境

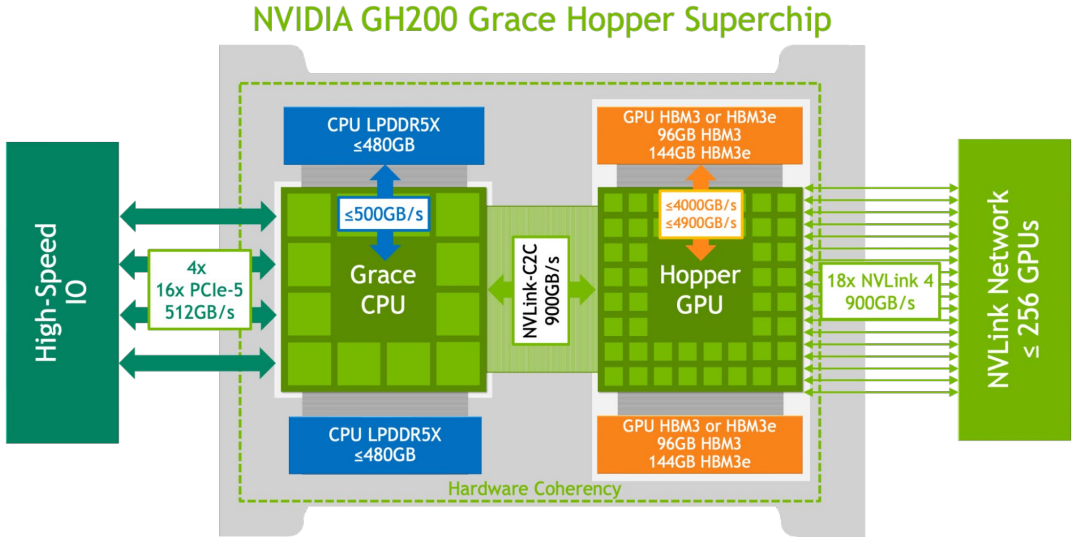
- 本研究では、CPUとGPUを同時に活用することで多数ケースの時刻歴シミュレーションのtime-to-solutionとenergy-to-solutionの双方を削減する手法を開発
- データ駆動型手法を活用：
  - 時刻歴シミュレーションにおける過去の求解結果を用いることで、次タイムステップの初期解を推定
  - これにより反復法ソルバーの反復数を削減し、精度劣化なくシミュレーションを高速化

Seismic wave propagation



Crustal deformation

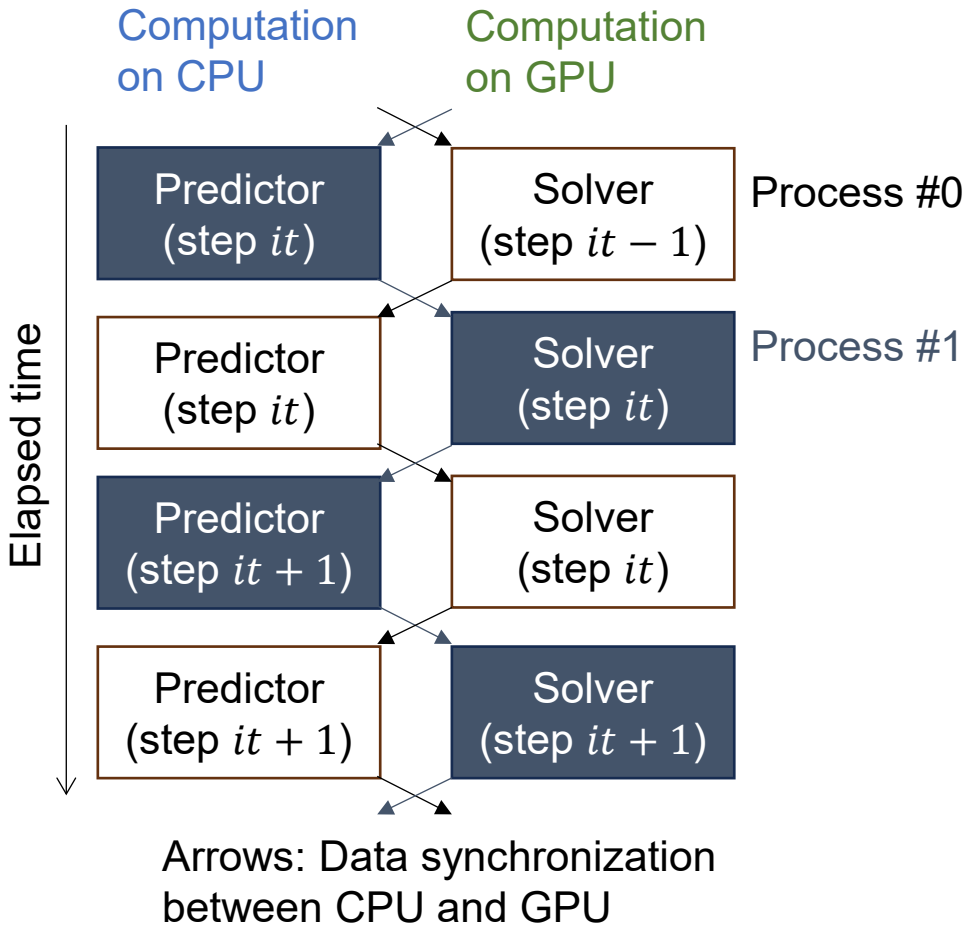




Data-driven part on memory rich CPU, Solver part on high-performance GPU, with fast CPU-GPU synchronization

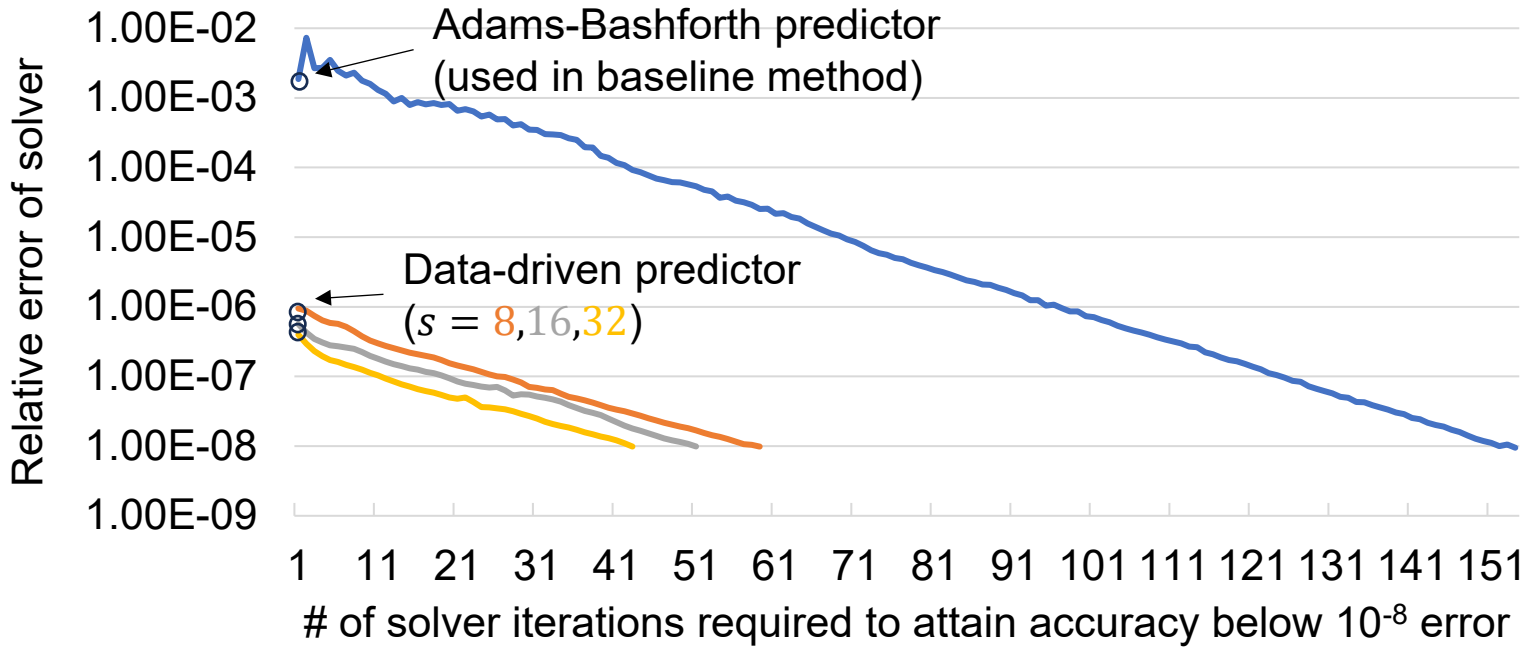
提案手法

- 多数ケースのシミュレーションを同時に実施
- Predictor@CPU：過去の求解データを活用することで、次のタイムステップの解を推定
- Solver@GPU：反復法ソルバーにより求解
- 高速な CPU-GPUインターコネクトにより predictor/solver結果を高速に同期



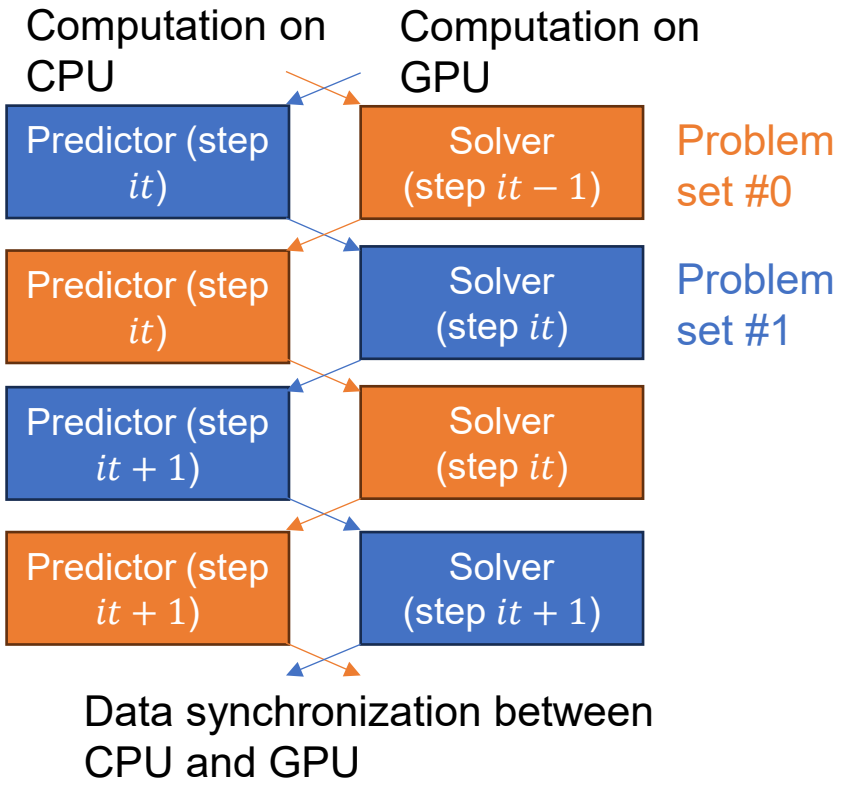
Predictor@CPU

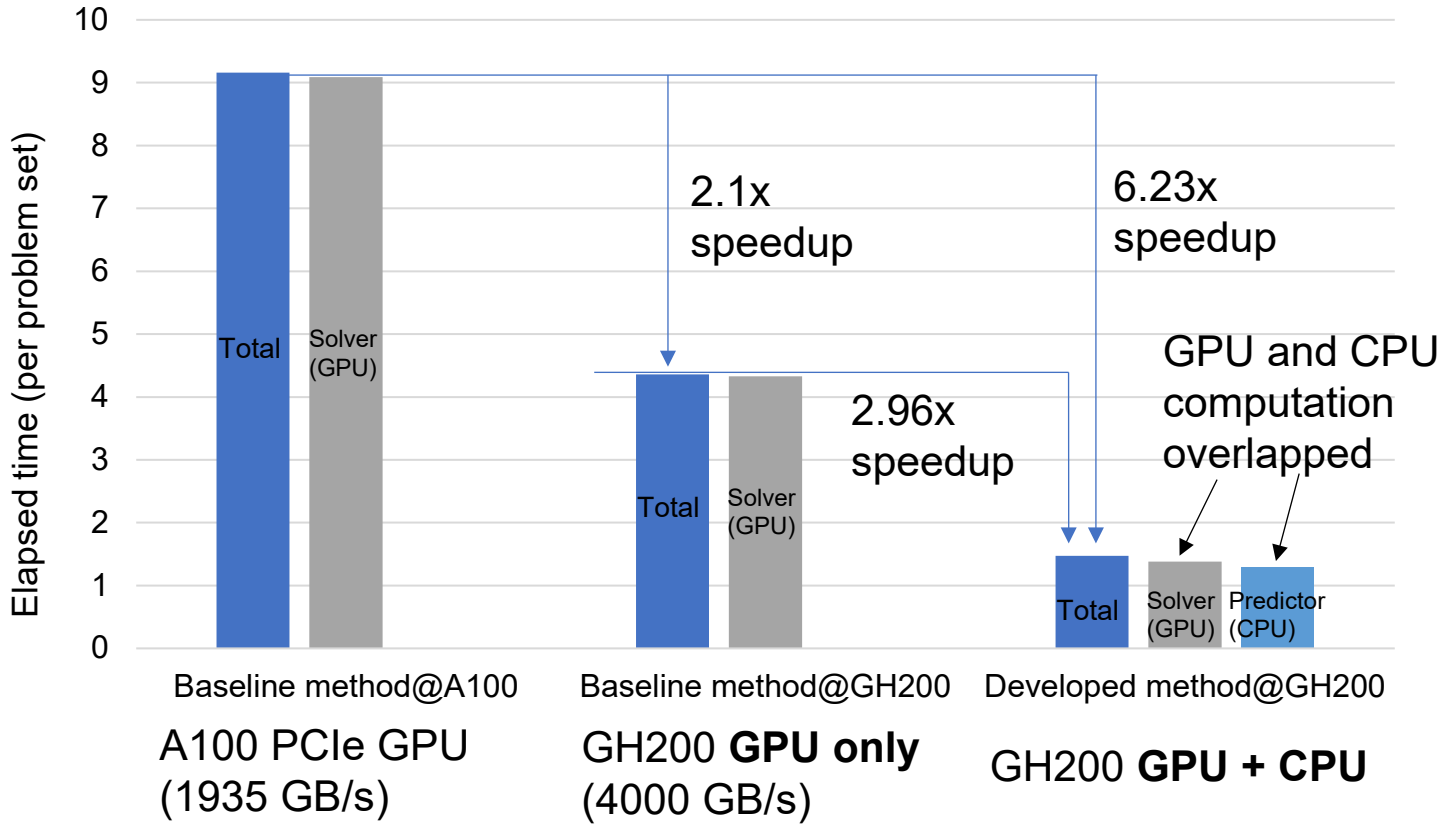
- 過去のs時間ステップのデータ$\boldsymbol{X}^{it} = \{\boldsymbol{x}^{it-s}, \boldsymbol{x}^{it-s+1}, \ldots, \boldsymbol{x}^{it-1}\}, \boldsymbol{F}^{it} = \{\boldsymbol{f}^{it-s}, \boldsymbol{f}^{it-s+1}, \ldots, \boldsymbol{f}^{it-1}\}$を用いることで、次のステップの解を$\bar{\boldsymbol{x}}^{it} = predictor(\boldsymbol{X}^{it}, \boldsymbol{F}^{it}, \boldsymbol{f}^{it})$と予測
- 初期解予測精度の向上により反復法ソルバーの反復数が削減される
- 多数ステップのデータはGPUメモリに格納できないため、大容量のCPUメモリを活用しCPUで計算



GH200における性能計測結果

- CPUとGPUの双方を使うheterogeneous computingにより計算効率を向上





Tsuyoshi Ichimura, Kohei Fujita, Muneo Hori, Maddegedara Lalith, Jack Wells, Alan Gray, Ian Karlin, John Linford,
Workshop on Accelerator Programming and Directives (WACCPD) 2024@SC24

まとめ

まとめ

- 有限要素法は複雑形状問題の求解にたけているが、ランダムアクセスが多く含まれるため、性能を出すには並列計算機の特徴に合わせたアルゴリズム開発・実装開発が重要
  - CPU向け：SIMDの活用のためのカーネルレベルの実装・求解アルゴリズムの再設計
  - GPU向け：低精度演算の活用、Tensor Coreの活用
  - また、CPUとGPUの双方を活用
  - データ駆動型アルゴリズムとの融合
- 開発手法はランダムアクセス系の他の手法にも応用可能と期待