AI Bridging Cloud Infrastructure上における3次元非構造格子有限要素解析の高速化検証

山口 拓真[1,a)] 市村 強[1] 藤田 航平[1] 堀 宗朗[1] ラリス ウィジャラトネ[1]

概要：近年，計算機性能の向上や観測データの蓄積によって高分解能での3次元有限要素解析が可能となりつつあり，より信頼性の高いシミュレーションの実現が期待されている．都市部を対象とした地盤震動解析では地上・地下の構造物と地盤をすべて考慮した解析を行うことが望ましいが，複雑形状を含んだ構造を解析対象とするため大自由度問題を解く必要があり，計算コストの増大が課題となる．この問題に対して，GPUを搭載したスーパーコンピュータを利用して計算時間の大幅な短縮及びスケーラビリティの改善が実現するソルバーが開発されている．一方で，実務においてはスーパーコンピュータを常時使用することは難しく，小規模から中規模のGPU計算機環境において高い性能を示すことがより重要となる．本論文ではAI Bridging Cloud Infrastructure (ABCI)の一部を対象として，提案された有限要素解析手法の性能分析を行う．他の一般的なソルバーとの性能の比較を通して，その有効性の検証を行う．

1. はじめに

これまで，計算機環境の発展に伴って地盤震動解析手法の高度化が進んできた．都市部の地盤震動解析に関しては地盤の揺れを解析し，その結果を用いて都市の構造物の揺れを計算するといった手法が一般的に用いられてきたが，このアプローチは複雑な都市構造物群が地盤に及ぼす影響を考慮することができなかった．計算機上で都市のモデルを構築し，それによる数値解析用モデルを構築し，地盤と都市構造物との相互作用を考慮した挙動を解析することができればより信頼性の高い被害評価につながると期待できる．一方で構造物を含む領域の解析は分解能の高さに影響されて大自由度問題となるため，計算コストの増大が課題となった．上記の課題を実現するため，[1]はGPUベースのスーパーコンピュータであるPiz Daint [2]及びSummit [3]を計算機環境として，良好なスケーラビリティが得られる高速な三次元有限要素ソルバーの開発を行った．その一方で，実務での地震被害推定においてスーパーコンピュータを常時使用することは容易ではない．この場合，$10^{1\sim2}$台のGPUを用いるような計算機環境においても，妥当な性能が得られることが重要となる．本論文では，[1]によって開発された有限要素ソルバーの，小規模計算機環境における性能評価を行う．AI Bridging Cloud Infrastructure (ABCI) [12]を用いた他のソルバーとの性能比較を通して，高度な前処理の他に，通信と計算のオーバーラップや精度変動演算の導入など，GPU計算のボトルネックとなりうる通信部分を考慮したアルゴリズムの設計が性能に影響を与えることを確認する．

2. 解析手法概要

解析対象領域は，$10^3 \times 10^3 \times 10^{1\sim2}$mの範囲であり，かつ，構造物が$10^{-2\sim1}$m程度の非常に複雑な幾何形状をもつ．そのため，非構造四面体二次要素を用いた非線形動的3次元有限要素法によって連続体の非線形時間発展問題を倍精度で解く．時間積分としてNewmark-β法($\beta = 1/4$, $\delta = 1/2$)を用いると，対象は以下の非線形時間発展問題となる．

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

ここで，$\delta\mathbf{u}$, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{a}$, $\mathbf{f}$は，各々，インクリメント変位，変位，速度，加速度，外力ベクトルである．また，$\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$は，整合質量，減衰，剛性マトリクスである．dt，nは時間刻み，タイムステップ数を表している．なお，$\mathbf{C}$としてRayleigh減衰マトリクスを用い，その要素減衰マトリクス$\mathbf{C}_e^n$は整合要素質量マトリクス$\mathbf{M}_e$及び要素剛性マトリクス$\mathbf{K}_e^n$を用いて与える．式(1)を解いて得られた$\delta\mathbf{u}^n$により時間ステップ毎に$\mathbf{q}^n = \mathbf{q}^{n-1} + \mathbf{K}^n\delta\mathbf{u}^n$, $\mathbf{u}^n = \mathbf{u}^{n-1} + \delta\mathbf{u}^n$, $\mathbf{v}^n = -\mathbf{v}^{n-1} + \frac{2}{dt}\delta\mathbf{u}^n$, $\mathbf{a}^n = -\mathbf{a}^{n-1} - \frac{4}{dt}\mathbf{v}^{n-1} + \frac{4}{dt^2}\delta\mathbf{u}^n$と

[1] The University of Tokyo, Bunkyo, Tokyo 113–0032, Japan
[a)] yamaguchi@eri.u-tokyo.ac.jp

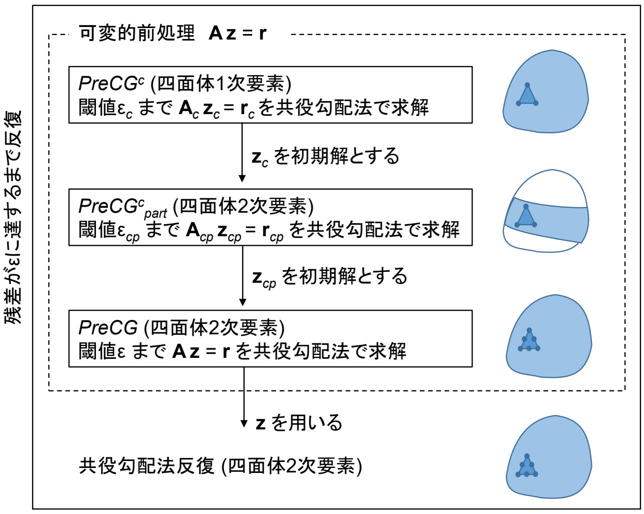

図 1　提案手法 [1] によるソルバーアルゴリズム．

して各ベクトルを更新しつつ非線形時間発展を解くこととなる．都市の非線形挙動解析へ向けた課題は，計算過程のほぼすべての計算コストである式 (1)，すなわち $\mathbf{A}^n\delta\mathbf{u}^n = \mathbf{b}^n$ を高速かつスケーラブルに解くことが出来る，最適なソルバーの開発である．従来法としては，$\mathbf{A}^n$ の 3×3 ブロック対角行列 [4] を前処理行列として用いる前処理付き共役勾配法と $\mathbf{A}^n$ をメモリ上に確保せずに on-the-fly で処理する Element-by-Element 法 [5]（EBE 法，後述）を組み合わせた *PCGE* を倍精度で実行する方法が挙げられる．また同種問題を解くことが可能なアルゴリズムとして [6] がある．これに対して，複雑な構造物を含んだ解析では格段に収束性が悪化することを考慮し，より積極的に収束性を改善するアルゴリズムを導入したものが [1] として位置づけられる．本論文では，以降 [6] によって提案されたソルバー，[1] によって提案されたソルバーをそれぞれ *SC14Solver*, *SC18Solver* と称する．

2.1 *SC18Solver* 構成

SC18Solver の構成は [1] に詳細が記されているが，その概要は図 1 に示されるものとなる．このソルバーは前処理部分 $\mathbf{r} = \mathbf{A}^n\mathbf{z}$ を共役勾配法によって大きな閾値で解く，可変的前処理 [7] を導入している．ここでは四面体二次要素を用いた $\mathbf{r} = \mathbf{A}^n\mathbf{z}$ の代替として四面体一次要素を用いた $\mathbf{r}_c = \mathbf{A}_c^n\mathbf{z}_c$ を設定しており，$\mathbf{A}_c^n$ に対して収束性を悪化させる一部領域 $\mathbf{A}_{cp}^n$ の抽出を行っている．前処理として $\mathbf{r} = \mathbf{A}^n\mathbf{z}$ を解くことを *PreCG*, $\mathbf{r}_c = \mathbf{A}_c^n\mathbf{z}_c$ を解くことを $PreCG^c$, $\mathbf{r}_{cp} = \mathbf{A}_{cp}^n\mathbf{z}_{cp}$ を解くことを $PreCG^c_{part}$ と呼ぶ．*SC18Solver* においては可変的前処理 $\mathbf{r} = \mathbf{A}^n\mathbf{z}$ に関して，初め四面体一次要素による連立一次方程式の求解 $PreCG^c$ を行い，得られた解を $PreCG^c_{part}$ の初期解として使用している．$PreCG^c_{part}$ で得られた解を四面体二次要素による求解部分 *PreCG* における初期解として計算を行うことによって，ベクトル $\mathbf{z}$ を得ることができる．このような構成を取ることによって，*PreCG* のみ，あるいは $PreCG^c$ と *PreCG* を用いて解く場合と比較すると収束性の改善が期待されることになる．

2.1.1 通信コストの削減

上述のアルゴリズムによって，ソルバーの演算量自体が削減されることが期待される．一方で，今回対象とするような近年の GPU 計算機環境の課題として，演算速度に対してメモリ転送及び他の GPU 間とのデータ転送が極めて遅いという傾向がある．CPU 向け大規模計算機環境を対象とした既往の研究では，可変的前処理には高い精度が必要ではないことに着目し，前処理部分に単精度変数を用いる精度混合演算が導入されてきた [6]．大規模 GPU 計算機環境においては通信コストが性能の大きなボトルネックになると予想されるため, データ転送量のさらなる削減および通信の隠蔽が計算の高速化に必須となる．そこで *SC18Solver* では大規模計算機環境上においてもスケーラビリティが確保できるよう，アルゴリズムの修正を行っている．

GPU 計算機環境に関しては従来の FP32/FP64 に加え FP16 [8] 演算機がサポートされている．FP16 を含めた精度変動演算によってデータ転送量が削減できれば計算の高速化に大きく寄与すると考えられる．その一方で，FP16 はダイナミックレンジが小さく，また精度が極めて低いことから一般的な科学数値計算で用いることは容易ではない．共役勾配法ソルバー部分に関しても計算対象とする配列全体を FP16 によって計算を行うと overflow/underflow の発生や低精度に伴う誤差の増加につながるため，ソルバーの収束性が失われると考えられる．そこで疎行列ベクトル積計算結果の袖通信部分において半精度変数を使用する．二つのプロセス間で通信されるベクトルデータを $\mathbf{y}_s$ とするとき，$\mathbf{y}_h \leftarrow \mathbf{y}_s/|\mathbf{y}_s|$ とし，ベクトル $\mathbf{y}_h$ とスカラー値 $|\mathbf{y}_s|$ を通信し，受信側は $\mathbf{y}'_s \leftarrow |\mathbf{y}_s|\mathbf{y}_h$ として復元する（$|.|$ には L infinity ノルムを使っている)．$\mathbf{y}_s$ と $\mathbf{y}'_s$ は等価とならないが，誤差は MPI 分割領域の境界部に局所化され，かつ，この結果は前処理にしか使われないために多くの問題で収束特性は変化せず，倍精度の最終結果に影響しない．これにより，FP64 を用いるソルバーに比べると隣接通信量を 1/4 に削減することができる．

また，GPU 間の袖通信および CPU-GPU 間のデータ転送を GPU 上の計算と同時に行うことによって通信時間の隠蔽を行っている．一般的に用いられる手法としては，[9] のように，疎行列ベクトル積の計算領域を袖領域と袖領域以外に分割するものが挙げられる．初めに通信が必要な袖領域の計算を行い，計算された値を他の GPU へ送受信する間に，転送が不要となる領域に関して同時にカーネルの計算を行うことができる．しかし，一部の GPU 計算機環境ではこのオーバーラップを用いても十分に通信時間を隠蔽することが難しいため，計算順序を組み替えた新しいア

For the i-th to $(i+3)$-th timesteps
$\mathbf{r} \Leftarrow \mathbf{A}\mathbf{x}$ and point-to-point comm. for $\mathbf{r}$
$\mathbf{r} \Leftarrow \mathbf{b} - \mathbf{r}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$
$\rho_a \Leftarrow 1$; $\alpha \Leftarrow 1$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$
synchronize ρ_b and γ by collective comm.
while $|\mathbf{r}_i|/|\mathbf{b}_i| > \epsilon$ **do**
  $\beta \Leftarrow -\gamma\rho_a/\alpha$
  $\mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$; $\mathbf{p} \Leftarrow \mathbf{z} + \beta\mathbf{p}$
  $\mathbf{q} \Leftarrow \mathbf{A}\mathbf{p}$ and point-to-point comm. for $\mathbf{q}$
  $\rho_a \Leftarrow (\mathbf{p}, \mathbf{q})$
  synchronize ρ_a by collective comm.
  $\alpha \Leftarrow \rho_b/\rho_a, \rho_a \Leftarrow \rho_b$
  $\mathbf{r} \Leftarrow \mathbf{r} - \alpha\mathbf{q}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$
  synchronize ρ_b and γ by collective comm.
end

Algorithm 1: 方程式 $\mathbf{A}\mathbf{x} = \mathbf{b}$ を解くための共役勾配法アルゴリズム．本論文では時間並列アルゴリズムの導入により，4 本のベクトルを同時に計算することを想定している．

ルゴリズムを用いている．提案手法のソルバーのベースである，共役勾配法をアルゴリズム 1 に示す．ここでは時間並列アルゴリズムの使用を想定している [10]．この方法では，有限要素法メッシュの接続情報は時間方向に変わらないことに着目し，複数時間ステップを並列で計算する．同時に解く時間ステップ数を m とした場合，反復法ソルバー一反復あたりの演算数は通常の方法の m 倍になるが，求めた将来時間ステップの解は反復ソルバーの高精度な初期解として使うことができるため，反復数を $1/m$ 程度まで減らすことができる．ここでは $m = 4$ であり，4 本のベクトルが同時に計算される．各反復につき疎行列ベクトル積は 1 回行われるため袖通信が 1 回必要となる．また係数の計算，およびそれに伴う MPI_ Allreduce の発行を除くと，他の計算はいずれもベクトルの加減算や内積計算などの単純なベクトル演算である．これらの演算において複数本のベクトルが同時に計算されていることに着目すると, アルゴリズム 2 のように，計算と通信の更なるオーバーラップを実現することが可能である．すなわち，4 タイムステップの求解を 2 ステップ ×2 組に分解し，一方のベクトル群で袖通信が必要となるとき，他方のベクトル群が他のベクトル演算を行うように計算の順序を組み替えることができる．これによって，通信を隠蔽できる計算の演算量が大幅に増加するため，スケーラビリティが改善すると考えられる．また，全体通信に関しては 1 反復あたりの MPI_ Allreduce の回数は変わらずに計算することが可能であるため，隣接通信コストのみが削減される．

時間並列アルゴリズムの利点として，複数のベクトルに同時にメモリアクセスを行うため，疎行列ベクトル積演算部分でのランダムメモリアクセスを軽減できることが挙げられる．逐次時間積分アルゴリズムと比較すると，演算数を同程度に保ったままランダムアクセス数の削減が可能であるため，近年の計算機においては計算時間の短縮が可能になる．ベクトルの本数が多いほどメモリアクセスの不規則性が減少するため，一度に計算されるベクトルの本数を 4 本から 2 本に変更することによって，カーネル単体での性能は低下する．多数の計算ノードを用いて計算する場合には通信性能がそれ以上の性能低下要因になることが予想されるので，大規模実行時には特にタイムステップの分割を行うことが有効になると考えられる．

2.1.2 演算の高速化

前章で述べた精度変動演算は演算部分にも導入可能であり，近年の GPU は倍幅半精度を用いることができれば理論演算性能が 2 倍となる．そこで *SC18Solver* では低精度変数を使用することによって各演算カーネルを高速化している．

対象とする行列，およびベクトルの各値に関しては，FP16 を適用できるレンジ以上のばらつきが存在するため行列やベクトル全体に対して FP16 を適用することは困難である．一方で要素単位で考えると，各要素の基底関数で展開される値のレンジは FP16 が定義する範囲内で取り扱えることが期待される．そこで行列ベクトル積において on-the-fly で要素行列を求め右辺ベクトルを掛け合わせる Element-by-Element 法 (EBE 法) におけるローカルな行列ベクトル積において FP16 を使う．

EBE 法では，行列ベクトル積 $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ を

$$\mathbf{y} \leftarrow \sum_e (\mathbf{Q}^{eT}(\mathbf{A}^e(\mathbf{Q}^e\mathbf{x}))) \tag{2}$$

と計算する．ここで，$\mathbf{A} = \sum \mathbf{Q}^{eT}\mathbf{A}^e\mathbf{Q}^e$ であり，$\mathbf{Q}^e$ は要素 e におけるローカル節点番号とグローバル節点番号のマッピング行列である．式 (2) をそのまま FP16 で実装すると変数の overflow/underflow が発生するため，$\mathbf{x}_h^e \leftarrow \mathbf{f}(\mathbf{Q}^e\mathbf{x}_s)$, $\alpha_s^e \leftarrow g(\mathbf{A}_s^e)$, $\beta_s^e \leftarrow h(\mathbf{Q}^e\mathbf{x}_s)$, として，

$$\mathbf{y}_s \leftarrow \sum_e \mathbf{Q}^{eT}\alpha_s^e\beta_s^e\mathbf{B}_h^e(\mathbf{x}_h^e)$$

と計算する．ここで，添字 s, h はそれぞれ FP32, FP16 の変数・関数を示す．$\mathbf{f}, g$ はベクトル $\mathbf{x}_h^e$ の成分，及び，関数 $\mathbf{B}_h^e(.)$ 内の計算で使われる変数の成分が 1 に近いように調整する関数であり，精度が無限であれば $\mathbf{A}^e\mathbf{x}^e = g(\mathbf{A}^e)h(\mathbf{x}^e)\mathbf{B}^e(\mathbf{f}(\mathbf{x}^e))$ を満たす．$\mathbf{y}_s$ への足しこみとスカラー値 α_s^e, β_s^e の計算は FP32 で行う必要があるが，主要計算部となる $\mathbf{B}_h^e(\mathbf{x}_h^e)$ は FP16 で行うことができるようになるため，FP32 に比べて FP16 の演算性能が高いシステムにおいては高速化が期待できる．このように高速化した EBE カーネルにおいては $\mathbf{y}_s$ へのランダム足しこみが実行時間の大部分を占めるようになる．

加算部分におけるメモリアクセスの不規則性を軽減させ

For the i-th and $(i+1)$-th timesteps	**For the $(i+2)$-th and $(i+3)$-th timesteps**
$\mathbf{r} \Leftarrow \mathbf{Ax}$ and point-to-point comm. for $\mathbf{r}$	$\mathbf{r} \Leftarrow \mathbf{Ax}$ and point-to-point comm. for $\mathbf{r}$
$\mathbf{r} \Leftarrow \mathbf{b} - \mathbf{r}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$	$\mathbf{r} \Leftarrow \mathbf{b} - \mathbf{r}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$
$\rho_a \Leftarrow 1$; $\alpha \Leftarrow 1$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$	$\rho_a \Leftarrow 1$; $\alpha \Leftarrow 1$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$
synchronize ρ_b and γ by collective comm.	synchronize ρ_b and γ by collective comm.
	$\beta \Leftarrow -\gamma\rho_a/\alpha$
	$\mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$; $\mathbf{p} \Leftarrow \mathbf{z} + \beta\mathbf{p}$
	$\mathbf{q} \Leftarrow \mathbf{Ap}$ and point-to-point comm. for $\mathbf{q}$
while $\lvert\mathbf{r}_i\rvert/\lvert\mathbf{b}_i\rvert > \epsilon$ **do**	**while** $\lvert\mathbf{r}_i\rvert/\lvert\mathbf{b}_i\rvert > \epsilon$ **do**
	$\rho_a \Leftarrow (\mathbf{p}, \mathbf{q})$
synchronize ρ_b and γ by collective comm.	synchronize ρ_a by collective comm.
$\beta \Leftarrow -\gamma\rho_a/\alpha$	$\alpha \Leftarrow \rho_b/\rho_a$; $\rho_a \Leftarrow \rho_b$
$\mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$; $\mathbf{p} \Leftarrow \mathbf{z} + \beta\mathbf{p}$	
$\mathbf{q} \Leftarrow \mathbf{Ap}$ and point-to-point comm. for $\mathbf{q}$	$\mathbf{r} \Leftarrow \mathbf{r} - \alpha\mathbf{q}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$
$\rho_a \Leftarrow (\mathbf{p}, \mathbf{q})$	
synchronize ρ_a by collective comm.	synchronize ρ_b and γ by collective comm.
$\alpha \Leftarrow \rho_b/\rho_a$; $\rho_a \Leftarrow \rho_b$	$\beta \Leftarrow -\gamma\rho_a/\alpha$
	$\mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$; $\mathbf{p} \Leftarrow \mathbf{z} + \beta\mathbf{p}$
$\mathbf{r} \Leftarrow \mathbf{r} - \alpha\mathbf{q}$; $\mathbf{z} \Leftarrow \mathbf{M}^{-1}\mathbf{r}$; $\rho_b \Leftarrow (\mathbf{z}, \mathbf{r})$; $\gamma \Leftarrow (\mathbf{z}, \mathbf{q})$	$\mathbf{q} \Leftarrow \mathbf{Ap}$ and point-to-point comm. for $\mathbf{q}$
end	**end**

Algorithm 2: 本手法で用いる，方程式 $\mathbf{Ax} = \mathbf{b}$ を GPU 計算機環境上で解くための共役勾配法アルゴリズム．左右で同じ行にある演算及び操作は同時に行うことができる．例えば，左右の係数に関する集団通信はベクトル 4 本分に対して一度に実行される．

るために，GPU の shared memory を有効利用することで L2 への atomic add を削減する方法を実装している．各スレッドによって計算される要素マトリクスとベクトルの乗算結果は一度 shared memory に格納され，shared memory 内で同じ節点に加算される値をまとめてから全体ベクトルへの加算を行う．この手法は [11] によって提案されたものを，地盤震動解析向けに拡張したものである．

以上に示される計算を倍幅半精度を用いて GPU 上で行う．時間並列アルゴリズムを使用しているため 1 要素につき複数ベクトルが一度に計算されており，これらの中からベクトル 2 本を取り出して倍幅半精度変数に格納しても計算が可能である．一方で，GPU の演算性能を決定する要素としては，各スレッドのレジスタ使用量と shared memory 使用量が代表的なものとして挙げられる．1 スレッドが複数の要素ベクトルを対象として計算を行う場合，前述したスレッド毎の計算結果を縮約するための shared memory 使用量が増加するため，発行できるスレッド数が減少し，カーネルの性能を低下させる可能性がある．そのため 1 本の要素ベクトルを 1 スレッドで計算する形をとる．図 2 のように要素ベクトルを 2 分割し，該当する要素マトリクスの部分を適切に掛け合わせることによって計算結果を得る．

FP16 は局所的に正規化が可能となる EBE カーネルに関しては導入可能であるが，そのほかのベクトル演算に関しては局所化の利用は困難であり，ダイナミックレンジの広い変数の使用が必要となる．一方で，収束性が担保されれば変数の精度は低くても良い．これを踏まえて，図 3 に示

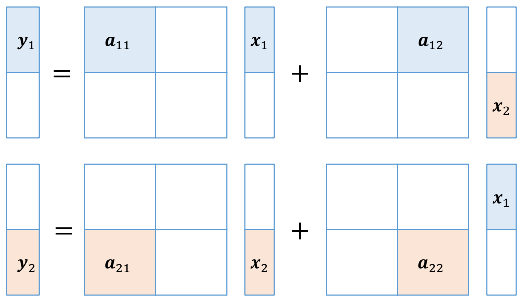

図 2 倍幅半精度による要素マトリクスベクトル積 $\mathbf{y} = \mathbf{Ax}$ の概略図．ただし，$\mathbf{x} = (\mathbf{x}_1 \ \ \mathbf{x}_2)$, $\mathbf{y} = (\mathbf{y}_1 \ \ \mathbf{y}_2)$ である．各段の上下が倍幅半精度によって 2 要素ずつ同時に計算される．右辺第 2 項の計算にあたっては，倍幅半精度データ型の上位ビットと下位ビットを入れ替える命令が存在するため，これを用いて右辺ベクトルを並び替えながら乗算を行う．

される変数を前処理部分で使用する．以降ではこのデータ型を FP21 と呼ぶ．この変数は FP32 と同等のダイナミックレンジを持っているため，overflow/underflow の発生を抑えることが可能である．またこの変数は FP32 変数の fraction 部後半を取り除くことによって得られるため，データ型の変換に伴う演算コストが極めて小さい．このデータ型はハードウェア上ではサポートされていないため，データ格納にのみ使用する．すなわち，この変数は FP21 としてメモリに格納されており，変数を使用する計算に際しては FP21 から FP32 への変換を行ったのち演算に用いる．演算後は，FP32 で得られる結果を FP21 に変換してからメモリに格納する．このデータ型が対象とする演算はメモリバンド幅律速となる演算であるので，read/write の対象

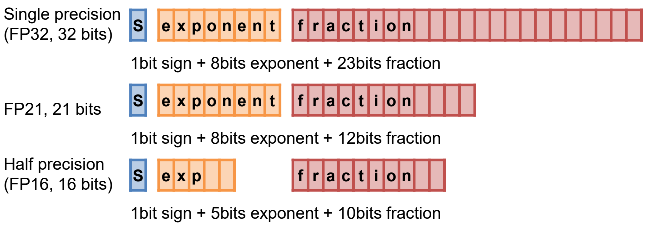

図 3 FP21 の概要図．各セルが 1bit に相当している．

となるメモリ量が減少することによって計算時間が短縮されると期待できる．

対象とする有限要素ソルバーは 3 次元の問題を対象としているため，x, y, z の 3 成分が 1 つの節点に対して存在する．そのため，64bit 配列の 1 要素ごとに 1 節点の値である FP21 の 3 要素を対応付けすることができ，これらの要素へのメモリアクセスを容易に取り扱うことができる．なお，EBE カーネルの計算にあたっては atomic 演算が使用されているが，これは実数では FP32 及び FP64 にしかハードウェア上のサポートがないため，EBE カーネルの出力ベクトルとしては FP21 でなく FP32 を用いる必要がある．よって共役勾配法の可変的前処理内では，FP21 によるベクトルと FP32 によるベクトルが混在する形となる．

3. 性能測定

SC18Solver ソルバーでの性能を，ABCI [12] を用いて測定する．各計算ノードには Intel Xeon Gold 6148 CPU (20 コア) が 2 台，NVIDIA V100 GPU が 4 台搭載されている．ノード間は InfiniBand EDR $\times$ 2 によって接続されている．GPU 使用時には各 MPI プロセスに 1GPU を割り当てて計算を行う．

3.1 問題設定

性能測定として使用する問題は， [1] と同様に 2 層の地盤内にコンクリートを模した固い層が存在するものである．地盤の一層目は歪に応じて剛性テンソルが変化する非線形物性とし，地盤の二層目は基盤層を模擬した線形物性とする．具体的には，z を鉛直方向として地表面 z=0m を水平にとったとき，$-5.55\text{m} < z < 0\text{m}$ または $-40\text{m} < z < -7.22\text{m}$ を満たす領域が地盤台一層，$-80\text{m} < z < -40\text{m}$ を満たす領域を地盤第二層とし，コンクリート層は中間の $-7.22\text{m} < z < -5.55\text{m}$ に位置する．このような領域を x, y 方向に複製した周期的な問題設定を使うことで弱スケーリングの測定に必要なモデル群を作成する．周期的な問題ではあるものの，実際の都市の問題と似たロードバランス特性となるように METIS [13] を利用してモデルを並列計算用に分割している．このモデルの底面に入力波として $dt = 0.01$ s の地震波（1995 年日本の兵庫県南部地震で観測された地震動 [14]）を入力し，25 時間ステップの求解にかかった時間を測る．本論文中では，

表 1 性能計測用モデルセット． 使用する GPU 台数が MPI プロセス数に等しい．

モデル	GPU 数	自由度	自由度/GPU
W-1	8	49,553,703	6,194,212
W-2	16	98,928,603	6,183,037
W-3	32	197,500,311	6,171,884
W-4	64	394,643,727	6,166,308
W-5	128	788,574,375	6,160,737

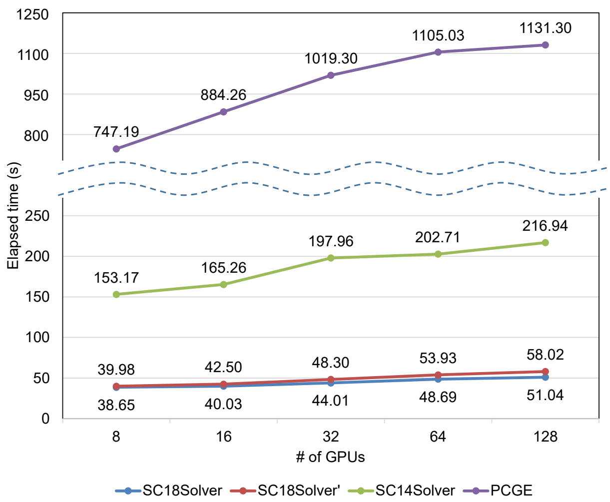

図 4 ABCI における弱スケーリング計測結果．

以下全てで，側面及び底面には半無限吸収境界条件を適用する．なお，任意の構成則を利用できるが，本論文では地盤の非線形構成則として修正 RO モデル [15] と Masing 則 [16] を用いている．

この問題において，提案手法である *SC18Solver* と一般的な解析手法である *PCGE*，及び *SC14Solver* を比較する．なお，計算コストの内訳は，*SC18Solver* から前処理 $PreCG^c_{part}$，FP16/FP21 演算，FP16 通信，時間並列アルゴリズムを除くと *SC14Solver* とほぼ等価になると解釈できる．*PCGE* は *SC14Solver* を FP64 演算・通信にし $PreCG^c$，$PreCG$ を除いたものとほぼ等価となる．また，*SC18Solver* から FP16/FP21 演算，FP16 通信，時間並列アルゴリズムによる通信の隠蔽を除いたものを *SC18Solver*$'$ として，全計算時間に対する通信時間と，低精度変数の影響を確認する．

ソルバーの閾値は全問題で $\epsilon = 1.0 \times 10^{-8}$ としている．なお，*SC18Solver* の各前処理 $PreCG^c$，$PreCG^c_{part}$，$PreCG$ の各閾値は 0.7, 0.05, 0.25 としており，*SC14Solver* の前処理の閾値は文献 [6] の通りとしている．すべての計測時間には解析結果のファイル出力を含んでいる．また，表 1 に示されるように，本論文では ABCI の 1 ラックまでを測定対象としている．

3.2 測定結果

各ソルバーにおける計算時間の推移は図 4 に示されてい

る．まず一般的な共役勾配法ソルバーである *PCGE* に関しては，8 GPU 使用時のモデル W-1 においてソルバー求解部計算時間は 747.19 秒であるのに対して，128 GPU 使用時のモデル W-5 においては 1131.30 秒となっており，最小サイズのモデルに対して 66.0%の性能となっている．なお，モデル W-5 におけるソルバー反復回数は [CG]=[134137] である．

SC14Solver はこれに可変的前処理を加えることによって必要な反復回数を *PCGE* よりも削減している．モデル W-5 におけるソルバー反復回数は [CG, $PreCG^c$, $PreCG$]=[351, 61957, 14389]，計算時間は 216.94 秒で，スケーラビリティに関しては，W-5 時は W-1 時の 70.1%の性能となっている．

これに前処理 $PreCG^c_{part}$，時間並列アルゴリズムを加えたものが $SC18Solver'$ に相当する．高度な前処理によって計算時間が削減されているものの，$PreCG^c_{part}$ の自由度が他の $PreCG^c, PreCG$ と比較すると 1/10 以下になるため，計算量に対して通信コストが相対的に増加することが問題となる．スケーラビリティは *SC14Solver* よりもわずかに悪化しており，W-5 時は W-1 時における性能の 68.9%となっている．

$SC18Solver'$ に FP16/FP21 による通信・計算を導入し，時間並列アルゴリズムの計算順を並び替えたものが *SC18Solver* である．$SC18Solver'$ と比較してスケーラビリティが改善しており，W-5 時は W-1 時の 75.7%の性能となっている．このことから，比較的小規模の計算機環境においても GPU 計算のボトルネックである通信部分を考慮したアルゴリズム開発が重要であることがわかる．またベクトル演算部分に低精度変数を使用しているため，計算時間も 128GPU 使用時において 51.0 秒まで減少している．この性能は *PCGE*, *SC14Solver*, $SC18Solver'$ ソルバーのそれぞれ 22.2 倍，4.25 倍，1.13 倍高速といえる．また，*SC18Solver* と $SC18Solver'$ の反復回数を比較することによって，低精度変数の使用が収束性に与えた影響を確認することができる．モデル W-5 において，*SC18Solver* の反復回数は [CG, $PreCG^c$, $PreCG^c_{part}$, $PreCG$]=[119, 4305, 27029, 2895] に対して $SC18Solver'$ では [CG, $PreCG^c$, $PreCG^c_{part}$, $PreCG$]=[97, 5189, 27420, 2844] であったため，今回対象とする問題セットに対しては FP16 及び FP21 は収束性に大きな影響がないことを確認することができる．なお，*SC18Solver* においても十分なスケーラビリティが得られているとは言い難いが，これは並列数の少ない有限要素モデルの隣接通信は隣接するプロセス数が少なく通信コストが減少することと，前処理部以外の袖通信に関しては十分に隠蔽ができないことが影響している．

4. おわりに

本手法では，GPU 計算機環境を対象とした三次元非構造格子有限要素ソルバーの高速化手法における ABCI 上での性能分析を行った．提案されている *SC18Solver* は高度な可変的前処理に精度変動演算，及び通信隠蔽アルゴリズムを組み込むことによって GPU をベースとしたスーパーコンピュータ上で良好なスケーラビリティを得るように設計されたものである．今回の性能分析によって，これらのアルゴリズムの導入は小規模計算機環境においても 10%以上の性能改善につながることが確認できた．ABCI のほぼ 1 ラックを使用した有限要素解析において，*SC18Solver* は従来のソルバー *PCGE* および *SC14Solver* と比較してそれぞれ 22.2 倍，4.25 倍高速となった．

なお，今回使用した FP16 および FP21 は従来の変数である FP32，FP64 と比較して精度が極めて低いため，問題設定によっては解が収束しない現象が確認されている．今後の課題としては，このような精度変動演算を伴うソルバーに関する収束性の検証が必要である．

参考文献

[1] T. Ichimura, K. Fujita, T. Yamaguchi, A. Naruse, J. Wells, T. Schulthess, T. Straatsma, C. Zimmer, M. Martinasso, K. Nakajima, M. Hori, and L. Maddegedara. “A fast scalable implicit solver for nonlinear time-evolution earthquake city problem on low-ordered unstructured finite elements with artificial intelligence and transprecision computing,” Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC’18), IEEE Press, 2018, p. 49.
[2] Piz Daint, [Online]. https://www.cscs.ch/computers/piz-daint/
[3] Summit, [Online]. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/
[4] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.
[5] J. M. Winget and T. J. Hughes, ”Solution algorithms for nonlinear transient heat conduction analysis employing element-by-element iterative strategies,” Computer Methods in Applied Mechanics and Engineering, vol. 52(1–3), 1985, pp. 711–815.
[6] T. Ichimura, K. Fujita, S. Tanaka, M. Hori, M. Lalith, Y. Shizawa, and H. Kobayashi, “Physics-based urban earthquake simulation enhanced by 10.7 BlnDOF x 30 K time-step unstructured FE non-linear seismic wave simulation,” Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC’14), IEEE Press, 2014, pp. 15–26.
[7] G. H. Golub and Q. Ye, “Inexact conjugate gradient method with inner-outer iteration,” SIAM Journal on Scientific Computing, vol. 21(4), 1997, pp. 1305–1320.
[8] D. Zuras, et al., “IEEE Standard for Floating-Point Arithmetic,“, IEEE Std 754-2008, 2008, pp. 1–70.
[9] Micikevicius, Paulius. “3D finite difference computation on GPUs using CUDA,” Proceedings of 2nd workshop on general purpose processing on graphics processing units, ACM, 2009, pp. 79–84.
[10] K. Fujita, K. Katsushima, T. Ichimura, M. Horikoshi, K. Nakajima, M. Hori, and L. Maddegedara, “Wave Propagation Simulation of Complex Multi-Material Prob-

lems with Fast Low-Order Unstructured Finite-Element Meshing and Analysis," Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2018), ACM, 2018, pp. 24–35.
[11] T. Yamaguchi, K. Fujita, T. Ichimura, A. Glerum, Y. van Dinther, T. Hori, O. Schenk, M. Hori, and L. Wijerathne, "Viscoelastic Crustal Deformation Computation Method with Reduced Random Memory Accesses for GPU-Based Computers," In International Conference on Computational Science, Springer, 2018, pp. 31–43.
[12] AI Bridging Cloud Infrastructure, [Online]. https://abci.ai/en/about_abci/computing_resource.html/
[13] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM Journal on scientific Computing, vol. 20(1), 1998, pp. 359–392.
[14] Strong ground motion of The Southern Hyogo prefecture earthquake in 1995 observed at Kobe JMA observatory, Japan Meteorological Agency, [Online]. https://www.data.jma.go.jp/svd/eqev/data/kyoshin/jishin/hyogo_nanbu/dat/H1171931.csv
[15] I. M. Idriss, R. Dobry, and R. D. Sing, "Nonlinear Behavior of Soft Clays during Cyclic Loading," Journal of the Geotechnical Engineering Division, vol. 104(ASCE 14265), 1978, pp. 1427–1447.
[16] G. Masing, "Eigenspannungen und Verfestigung beim Messing," Proceedings of the 2nd International Congress of Applied Mechanics, 1926, pp. 332–335.