

PEZY-SC3 に向けた PEZY-SC シリーズ向け OpenFOAM の実装と性能評価

菱沼 利彰 [1†] 黒澤 範行 [1]

[1] 株式会社 PEZY Computing

Abstract

我々は，連立一次方程式のソルバライブラリ pzSolverLA を MIMD 型メニーコアプロセッサ PEZY-SC2 上で実装し，それを基に OpenFOAM のソルバを PEZY-SC シリーズ向けに移植し，OpenFOAM の高速化を行っている．PEZY-SC2 は倍精度ピーク性能 4.1 TFlops，メモリバンド幅 100 GB/s で，性能がメモリバンド幅に制約を受けるアプリケーションでは性能が出にくい．一方，2019 年にリリース予定である PEZY-SC3 は，倍精度ピーク性能 19.7 TFlops，メモリバンド幅 1200 GB/s になる．これは PEZY-SC2 の 1984 コアモデルと比べピーク性能が約 7.1 倍，メモリバンド幅が約 15.6 倍で，連立一次方程式のソルバに必要な疎行列計算やベクトル計算の大幅な性能向上が見込める．そこで我々は連立一次方程式ソルバライブラリ pzSolverLA を PEZY-SC2 上で開発・性能評価を行い，その結果を基に PEZY-SC3 に向けた性能の見積もりを行った．PEZY-SC2 で OpenCAE 標準ベンチマークの問題 (サイズ約 2400 万次元) を解いた結果から PEZY-SC3 の性能を見積もると，50 タイムステップ全体で Intel Xeon CPU の約 4.5 倍の性能向上が得られることが推察された．

Keywords: OpenFOAM, PEZY-SC, Iterative Solver,Precondition, Performance Evaluation

1. はじめに

計算機の進化に伴って，より大規模な問題への応用が進んでいる．大規模な連立一次方程式のソルバを核とする数値解析は問題に合わせた離散化・解法・前処理や，高速化のための並列化が重要であり，流体解析の分野では，それらをもつ高機能な数値解析ソフトウェア OpenFOAM (Open source Field Operation And Manipulating) [1] の利用が拡大している．

コンピュータの高性能化を進める上での大きな課題として消費電力効率の向上が挙げられる．PEZY Computing 社と ExaScaler 社では，メニーコアプロセッサ PEZY-SC2 [2] を搭載したスーパーコンピュータシステムの開発を行っており，弊社の開発した Shoubu System B において，スーパーコンピュータの省エネルギーランキング Green500 [3] において，2017 年から 2018 年にかけて 3 期連続で 1 位に認定されるなどの成果を上げてきた．現在も新しいハードウェアの開発やそれらの上で動作する数値シミュレーション [4] などに取り組んでいる．

連立一次方程式のソルバは疎行列ベクトル積や内積などのベクトル計算が実行時間の大部分を占め，これらはメモリ性能がボトルネックとなる．一方，PEZY-SC2 はピーク性能に対してメモリバンド幅が低いため，性能がメモリバンド幅に制約を受けるアプリケーションでは性能が出にくい．

2019 年にリリース予定である PEZY-SC3 ではメモリ性能を大幅に向上させ，倍精度ピーク性能 19.7 TFlops，メモリバンド幅 1200 GB/s を予定している．そこで我々は連立一次方程式ソルバライブラリ pzSolverLA を PEZY-SC2 上で開発・性能評価を行い，その結果を基に PEZY-SC3 に向けた性能の見積もりを行った．

2. PEZY-SC2/SC3 のアーキテクチャ

表 1に我々が用いた PEZY-SC2 1984 コアモデル (以下，このモデルを PEZY-SC2 と呼ぶ)，及び開発中の PEZY-SC3 の諸元を示す．これは歩留まり向上のためにメモリバンド幅やコア数が若干少なくなっているモデルである．

PEZY-SC2 はホストとして Intel や MIPS などの CPU を用い，ホスト CPU から実行バイナリや必要なデータを転送することでプログラムを実行する．

PEZY-SC2 は MIMD (Multiple Instruction stream, Multiple Data stream) アーキテクチャを採用しており，コアあたり 8 つのスレッドを立てることができる．1984 個のコアが自身のプログラムカウンタをスレッド数本分もつことで各スレッドを別々に動作させることが可能で，GPU などと比べて条件分岐などによる性能劣化が起

† *E-mail address of corresponding author: hishinuma@pezy.co.jp*

Table 1　Features of PEZY-SC2 1984 core model / PEZY-SC3.

	PEZY-SC2, 1984 core model (2017–)	PEZY-SC3 (2019)
Clock Freq.	0.7 GHz	1.2 GHz
PCIe I/F	PCIe Gen3 32 Lane (32 GB/s)	PCIe Gen4 48 Lane (96 GB/s)
DDR I/F	DDR4 64bit 2400MHz 4port (76.8 GB/s)	about 1200 GB/s
# of MIMD PE (core)	1984	4096
SIMD width	64 bit	128 bit
Peak (DP)	2.8 TFlops	19.7 TFlops (×2 SIMD)
Power Consumption	200 W (Peak)	500 W (Peak)

こりにくい．また，加算と乗算を 1 命令で行うことができる MAD 命令 ($d = a + b \times c$) が使用可能である．

PEZY-SC3 は PEZY-SC2 の演算性能・メモリ性能を向上させたアーキテクチャで，4096 の MIMD コアと 128 bit の SIMD 拡張命令を備え，周波数を上げることで PEZY-SC2 の 7.1 倍の理論性能で，メモリ性能は 1200 GB/s で PEZY-SC2 の 15.6 倍のメモリ性能をもつ．また，Last Level Cache (LLC) のサイズが 40 MB から 64 MB に向上するなどキャッシュメモリの強化も行われている．

3.　PEZY-SC シリーズ向け OpenFOAM の実装

3.1.　OpenFOAM の問題点と PEZY-SC シリーズ向け OpenFOAM の設計

OpenFOAM の連立一次方程式ソルバ・前処理をメニーコアアーキテクチャ上で効率よく動作させるには，以下の問題点がある：

1. 疎行列の格納形式が座標形式 (COO) [5] であるため，SpMV を並列化しにくい．
2. スレッド並列に対応しておらず，MPI を用いたプロセス並列でしか並列化がされていない．
3. 対角スケーリングを除く前処理に並列性がない．

1, 2 の問題点に対しては，疎行列の格納形式に行優先圧縮 (CRS) 形式 [5] を用いることで行列の次元分の並列性を確保した．このとき，COO 形式から CRS 形式への変換は各ソルバの呼び出し時に行うこととした．また，BiCG に代表される SpMV，転置 SpMV が必要な解法では CRS 形式の行列に加え列優先圧縮 (CCS) 形式の行列も作成して保持することにした．

3 の問題点に対して，CPU 上では OpenFOAM は各プロセスがもつローカルな行列に対する前処理を行うことでプロセス並列を行っているが，このとき各プロセス間で通信が必要な処理については無視しているため，並列度に従って前処理の効果が低下する．PEZY-SC2 は $1984 \times 8 = 15872$ のスレッド並列を行うため，同様の行列分割を行うと前処理の効果がほとんど得られないであろうことが予想される．そのため，今回は OpenFOAM が実装している不完全 Cholesky 分解法 (IC 法)，不完全 LU 分解法 (ILU 法) の実装の優先度を下げ，以下の二つの 2 つの前処理の実装を行った．

1 つめは Newton-Schulz-Hotelling 法 [6] を用いて得られる近似逆行列を前処理行列として用いる手法 (以下，Schulz 法) である．具体的には入力された行列 A とその対角行列 D を用いて，

$$\mathsf{A}^{-1} \approx \mathsf{D}^{-1} - \mathsf{D}^{-1}(\mathsf{A} - \mathsf{D})\mathsf{D}^{-1}$$

とした．これは RapidCFD[7] で IC 法・ILU 法の代わりに実装されている手法である．

2 つめは A-直交化に基づく近似逆行列 (AINV) 前処理 [8] である．疎行列の非ゼロプロファイルに従って生成した上三角行列の 対角 $+p$ に対して Fill-in を行う AINV(p) 法として実装した．これは並列性は確保できるが，計算量 N^2 の上三角行列に対する計算が発生する．これは他の前処理と計算量が大きく異なるため，本梗概の性能評価の項では扱わない．今後，対角ブロックに限定した計算を可能にしたり，分割数を入力できるインタフェースについて検討する予定である．

これらの連立一次方程式ソルバは全て行列の格納形式の変換や CPU とデバイス間の通信などを内包し，OpenFOAM から渡された COO 形式の行列やベクトルのデータをそのまま渡すことで，解を OpenFOAM 同様

Table 2 Functions of OpenFOAM for PEZY-SC series.

Solver	Preconditioner	Matrix Format
Preconditioned CG	None	COO
Preconditioned BiCG	Diagonal Scaling	CRS
Preconditioned BiCGStab	Schulz method	CCS
Jacobi method	AINV(p) methods	

の形式で出力する．

PEZY-SC シリーズ版 OpenFOAM は，OpenFOAM のもつメッシュ生成やモデリング部分は変更せず，連立一次方程式ソルバ・前処理部分に対して PEZY-SC シリーズ向けのソルバ・前処理を別の名前で新しく追加することで，もともとの OpenFOAM に大きな変更をせず，かつアーキテクチャ依存部分を独立させられるように設計した．

具体的には，様々なソルバ・前処理に共通して必要な内積や疎行列ベクトル積 (SpMV) などの処理を，PEZY 版 BLAS ライブラリ pzBLAS や疎行列ライブラリ pzsparse として実装し，これらを組み合わせた連立一次方程式のソルバライブラリ pzSolverLA を OpenFOAM の裏側で用いることができるようにした．OpenFOAM の変更はほとんど必要なく，pzBLAS や pzsparse の呼び出しを行うソルバ・前処理のソースコードの追加と，それらを切り替えるインタフェース用のソースコードへの追記のみでよい．

これらにより，(A) ビルド方法などはほぼ同様，(B) 実行時に PEZY 用のソルバ名を指定するのみで切替可能，(C) OpenFOAM のソルバの切り替えインタフェース以外のアップデートをそのまま適用できる．などの長所が得られた．

前処理は OpenFOAM の CPU 用実装，ソルバは PEZY-SC シリーズ向けなどのように組み合わせることもできるが，PEZY-SC シリーズ用のプログラム開発・実行環境 (pzSDK) は現状では 1 デバイス上での複数のプロセス立ち上げに対応していないため，CPU の前処理も 1 デバイスあたり 1 プロセスしか立ち上げられない．そのため MPI 並列を前提としている CPU 側では並列化による高速化効果が得られなくなる．

4. 数値実験

4.1. 実験環境

対象とした問題は，オープン CAE 学会チャネル流れベンチマークの問題 [9] の 2 つのサイズを対象とした．1 つめはメッシュサイズを $x = 240, y = 130, z = 96$ とした行列サイズが 2 995 299，非ゼロ要素数が 20 832 960 の “Mesh_3M，” 2 つめはメッシュサイズを $x = 480, y = 260, z = 192$ とした行列サイズが 23 961 600，非ゼロ要素数が 167 197 440 の “Mesh_24M” である．実験は 51 タイムステップ分行い，1 タイムステップ目を捨てた 2 から 51 タイムステップの合計時間に対して評価を行った．

実験環境は PEZY-SC2 1984 コアモデルで，ホスト CPU に Intel Xeon E5-2618L v3@2.3 GHz, 8 コア, メモリ 64 GB を用いた．OS は CentOS 7.2，ホストプログラムのコンパイラは gcc 4.8.5，カーネルプログラムのコンパイラは pzSDK4.1 + LLVM 3.6.2 である．

比較対象として，Intel Xeon E5-2683 v4@2.1 GHz, 16 コア, メモリ 64 GB を用いた．これはピーク性能 537.6 GFlops とメモリバンド幅 76.6 GB/s である．OpenFOAM のバージョンは v1806 で，実験ではコア数に合わせて MPI による 16 プロセス並列を行った．

本研究では，以下の 4 つの解法・前処理の組み合わせについて評価を行った．

(A) 速度線形ソルバは PBiCG 法・前処理なし，圧力線形ソルバは PCG 法・前処理なし．
(B) 速度線形ソルバは PBiCG 法・前処理は Diagonal，圧力線形ソルバは PCG 法・前処理は Diagonal.
(C) 速度線形ソルバは PBiCG 法・前処理はなし，圧力線形ソルバは PCG 法・前処理 Schulz 法 (PEZY-SC2/SC3 のみ).
(D) 速度線形ソルバは PBiCG 法・前処理は ILU 法，圧力線形ソルバは PCG 法・前処理 IC 法 (CPU のみ，Fill-in = 0).

Table 3 The Number of CG / BiCG Iterations in 50 time steps [x10^3].

	CPU				PEZY-SC2			
	Mesh_3M		Mesh_24M		Mesh_3M		Mesh_24M	
	CG	BiCG	CG	BiCG	CG	BiCG	CG	BiCG
(A)	92.0	0.69	179.3	0.78	91.3	0.69	179.4	0.77
(B)	90.7	0.68	181.1	0.79	93.1	0.68	183.3	0.77
(C)	none	none	none	none	58.8	0.69	116.4	0.78
(D)	14.0	0.53	37.4	0.46	none	none	none	none

4.2. PEZY-SC3 の性能見積もり方法

我々は，それぞれのソルバ・前処理の処理時間をボトルネック要因の異なる 4 つのパートに分けて計測することで，PEZY-SC2 の実験結果を基に PEZY-SC3 の性能を見積もることにした．

4 つのパートの概要と性能の見積もりモデルは以下の通りである．

Kernel PCG / PBiCG 法の計算カーネル．内積や SpMV，転置 SpMV などから成る．全ての演算はメモリ性能がボトルネックになっているため，PEZY-SC2/SC3 メモリ性能の比から 15.6 倍になることが期待できる．

Convert OpenFOAM の COO 形式から CRS / CCS 形式への変換．各タイムステップの最初に必要な処理．メモリデータのコピー，及びソートを核とし，メモリ性能がボトルネックになっているため，PEZY-SC2/SC3 メモリ性能の比から 15.6 倍になることが期待できる．

Comm. PEZY-SC2 とホスト CPU 間のデータ転送処理．PEZY-SC3 と CPU 間の転送速度はシステム構成に依存するが，これに関しては現状では未定のため，PEZY-SC2 と等しく 1.0 倍になるとした．

Other ホストでの配列初期化やカーネル起動のオーバーヘッドなどから成る．カーネル起動時間などは PEZY-SC2 と等しくかかると考えて 1.0 倍とした．

4.3. 実験結果

それぞれの問題における 50 タイムステップ全体の処理時間を図 1に，50 タイムステップ全体における PCG / PBiCG 法の反復回数を表 3に示す．

Mesh_3M の問題において，以下の結果が得られた：

- 並列化による演算順序の違いで反復回数が異なるが，同じ前処理における PEZY-SC2 の反復回数は 3 % 以内の差に収まった．
- (B) の Diagonal は (A) 前処理なしと比べて反復回数が増え，この問題では有効ではない．
- IC / ILU 前処理の収束改善効果が非常に高く，反復回数が約 85 % 減少．
- PEZY-SC2 の前処理としては Schulz 法が有効で，反復回数が約 35 % 減少．
- PEZY-SC2 は (A) において，CPU の処理時間を 1 としたとき，Kernel / Convert / Comm. / Other の時間がそれぞれ 0.89 / 0.31 / 0.18 / 0.09 で，全体の処理時間は 1.47．
- PEZY-SC3 は (A) において，CPU の処理時間を 1 としたとき，Kernel / Convert / Comm. / Other の時間がそれぞれ 0.06 / 0.02 / 0.18 / 0.10 で，全体の処理時間は 0.35．
- CPU で最も高速な (D) と PEZY-SC3 で最も高速な (C) を比べると，PEZY-SC3 は CPU の 1.08 倍の時間．

Mesh_24M の問題において，以下の結果が得られた：

- 並列化による演算順序の違いで反復回数が異なるが，同じ前処理における PEZY-SC2 の反復回数は 1 % 以内の差に収まった．
- (B) の Diagonal は (A) 前処理なしと比べて反復回数が増え，この問題では有効ではない．
- IC / ILU 前処理の収束改善効果が非常に高く，反復回数が約 79 % 減少．
- PEZY-SC2 の前処理としては Schulz 法が有効で，反復回数が約 36 % 減少．

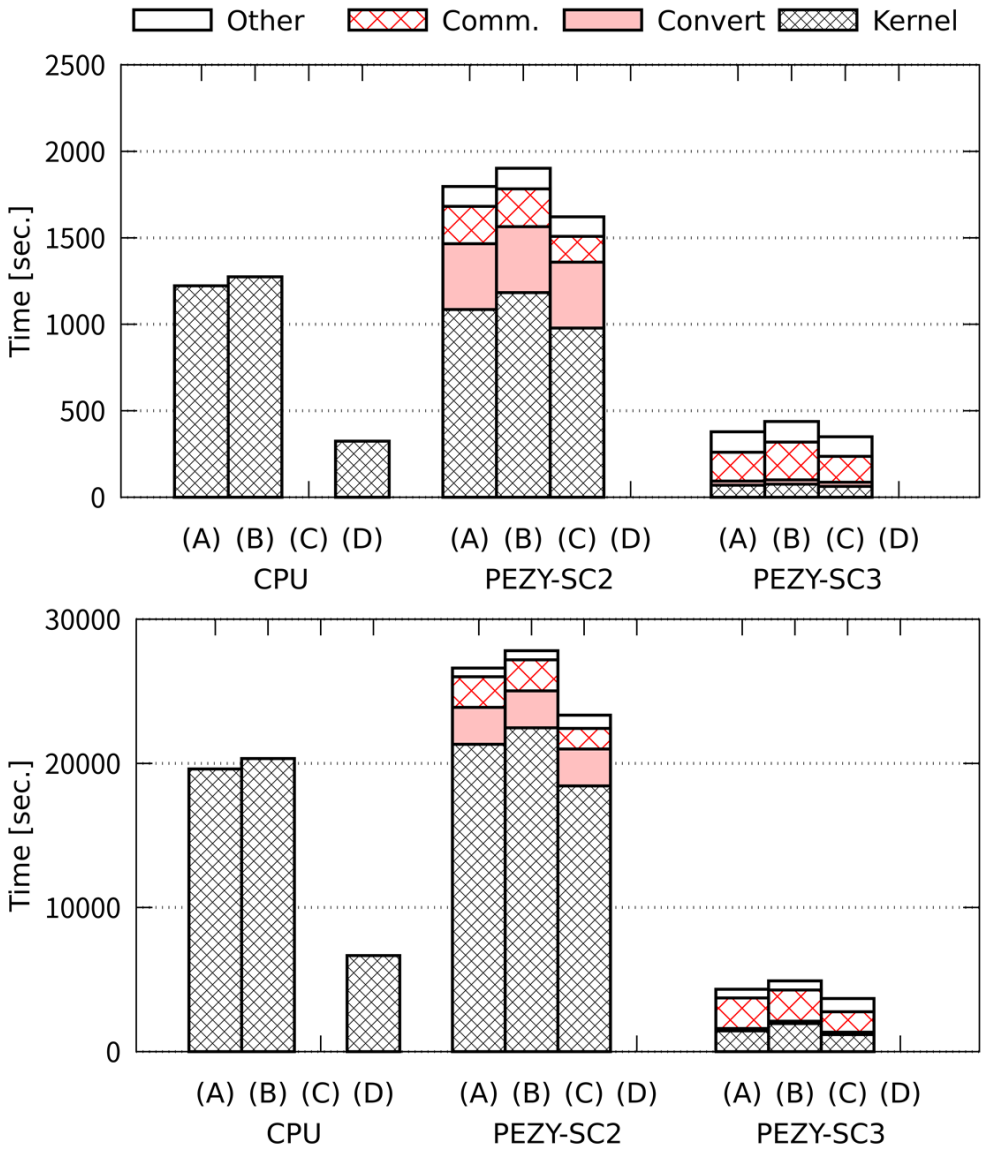

Fig. 1 The Total Elapsed Time of 50 Time steps (up: Mesh_3M, bottom: Mesh_24M).
(A): None / None, (B): Diagonal / Diagonal, (C): Schulz / None, (D): IC / ILU.

- PEZY-SC2 は (A) において，CPU の処理時間を 1 としたとき，Kernel / Convert / Comm. / Other の時間がそれぞれ 1.09 / 0.13 / 0.11 / 0.03 で，全体の処理時間は 1.36.
- PEZY-SC3 は (A) において，CPU の処理時間を 1 としたとき，Kernel / Convert / Comm. / Other の時間がそれぞれ 0.07 / 0.01 / 0.11 / 0.03 で，全体の処理時間は 0.22.
- Mesh_3M はベクトル 1 本であればキャッシュメモリに格納できたが，Mesh_24M ではできないため Kernel の処理時間が増大.
- CPU で最も高速な (D) と PEZY-SC3 で最も高速な (C) を比べると，PEZY-SC3 は CPU の 0.55 倍の時間.

5. まとめと考察

本研究では，連立一次方程式のソルバライブラリ pzSolverLA 及び，それをベースとした PEZY-SC シリーズ向け OpenFOAM の開発・性能評価を行った.

2019 年にリリース予定である PEZY-SC3 は，倍精度ピーク性能 19.7 TFlops，メモリバンド幅 1200 GB/s に

なる．これは PEZY-SC2 と比べピーク性能が約 7.1 倍，メモリバンド幅が約 15.6 倍で，連立一次方程式のソルバに必要な疎行列計算やベクトル計算の大幅な性能向上が見込める．そこで我々は PEZY-SC2 の実験結果を基に PEZY-SC3 の性能見積もりを行い，PEZY-SC3 において数値解析を行うことの有用性を示した．

メニーコアプロセッサでは，OpenFOAM の並列化の難しい行列格納方法や前処理の実装が問題点となる．これに対し，並列性のある CRS / CCS 形式へ各ソルバの呼び出し時に行うことで，並列性の高い高速なソルバを実現した．並列性化の難しい IC / ILU 前処理は Schulz 法や AINV(p) 法を実装することで，並列性を確保しつつ条件の悪い問題への適用性を高めることを試みた．

また，PEZY-SC シリーズ向け OpenFOAM は，OpenFOAM に PEZY-SC シリーズ用のソルバを追加したものになっているため，ビルド方法や実行方法は従来の OpenFOAM とほぼ同様の形で実装できた．

実験の結果から，以下のことがわかった：

- PEZY-SC2 は Kernel だけで比較すれば CPU と同程度だが,Convert, Comm., Other などが原因で CPU の約 1.3 から 1.5 倍の時間がかかる．
- PEZY-SC3 は PEZY-SC2 で全体の約 60 から 80 % を占める Kernel の時間と，約 10 から 21 % の時間を占める Convert がメモリ性能の向上により 15.6 倍の高速化が見込めるため，CPU の 3 から 5 倍高速．
- 問題サイズが大きければ計算時間の高速化効果が及ぼす影響が大きいこと，反復回数が多く必要なことから，Convert や Other のコストが相対的に小さくなり，PEZY-SC3 で 5 倍の高速化効果が得られた．
- 問題サイズが小さく，少ない反復回数で収束する問題では Comm. や Convert の時間が大部分を占め，大きいサイズの問題と比べて高速化効果が小さく 3 倍程度の高速化に留まる．

前処理なしでの比較から PEZY-SC3 は，並列性のある前処理であれば問題サイズに応じて 2.8 倍から 4.5 倍程度の高速化効果が得られた．一般的に前処理の並列性と効果はトレードオフであることから，悪条件でない問題であれば，今回の実験と同程度の高速化効果は得られると考えられる．

これらの結果から，今後前処理などの実装を増やしていくことにより，PEZY-SC3 は大規模な数値解析に対して有用であることが推察された．

今後はより悪条件な問題や大規模な問題に対応するための実装を進めていく必要があると考えている．Schulz 法や Jacobi 法を緩和法とした AMG 法や，マルチカラーオーダリングを施した IC / ILU 法の実装を考えている．

より大規模な問題への適用という観点では，MPI によるプロセス並列化の実装を行う予定である．MPI 並列した際には MPI のプロセス間の通信と，CPU と PEZY-SC2 間の通信をオーバーラップして隠蔽できるので，さらなる高速化が見込めると考えている．

参考文献

[1] OpenFOAM, https://www.openfoam.com/.
[2] PEZY Computing, PEZY-SC2 モジュール & プロセッサ, https://www.pezy.co.jp/products/.
[3] Green500, https://www.top500.org/green500/.
[4] H. Tanaka, et al., “Automatic Generation of High-Order Finite-Difference Code with Temporal Blocking for Extreme-Scale Many-Core Systems”, ESPM2 2018 (in conjunction with SC18), pp. 1–8, 2018. 11 (Accepted).
[5] R. Barrett, et al., “Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods”, SIAM, pp. 57–65 (1994).
[6] G. Schulz, “Iterative Berechnung der Reziproken matrix”, Z. Angrew. Math. Mech. 13 (1933), 57–59.
[7] RapidCFD, https://sim-flow.com/rapid-cfd-gpu/.
[8] M. Benzi, C. D. Meyer, M. Tůma, “A sparse approximate inverse preconditioner for the conjugate gradient method”, SIAM Journal on Scientific Computing 17.5 (1996): 1135–1149.
[9] オープン CAE 学会チャネル流れベンチマーク, https://gitlab.com/OpenCAE/OpenFOAM-BenchmarkTest.