

# Oakforest-PACSにおける 大規模CFD解析向け省通信型行列解法の開発

井戸村 泰宏

日本原子力研究開発機構

第11回JCAHPCセミナー（OFP運用終了記念シンポジウム）

「ありがとうOFP:京から富岳への狭間で咲いた大輪の花」

オンライン、2022年5月27日

## 共同研究者

- ◆ 伊奈拓哉、山下晋、小野寺直幸、山田進、真弓明恵、朝比祐一（JAEA）
- ◆ 今村俊幸（理研）、Yussuf Ali（OIST）、松本和也（会津大）

## 謝辞

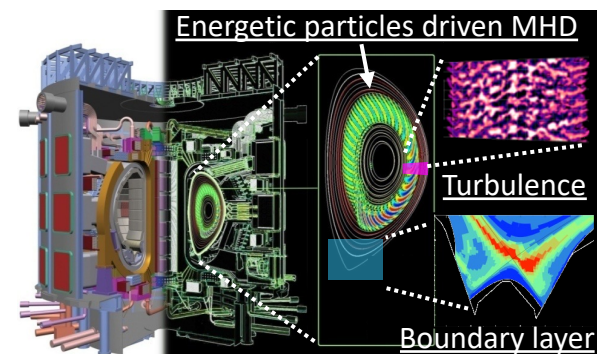
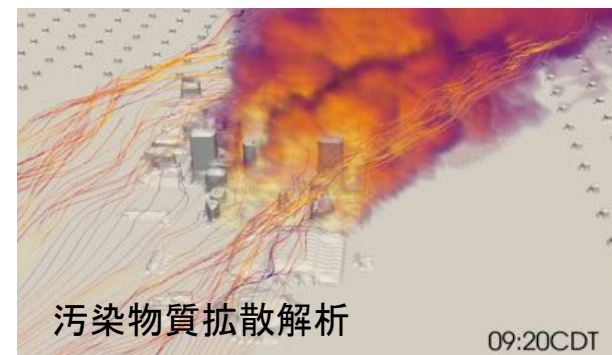
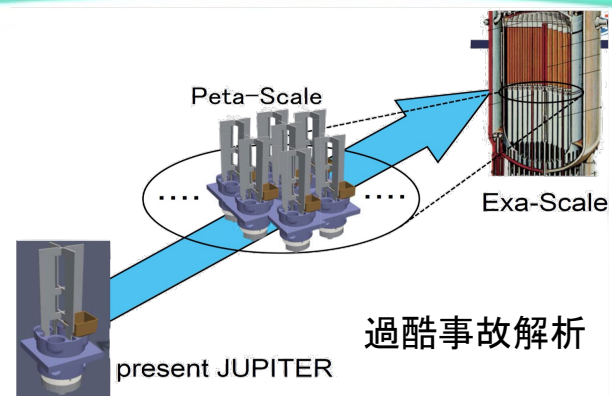
- ◆ 本研究はJCAHPC大規模HPCチャレンジ、文部科学省ポスト「京」重点課題⑥「革新的クリーンエネルギーの実用化」、「富岳」成果創出加速プログラム「核燃焼プラズマ閉じ込め物理の開拓」の支援の下に行われました。
- ◆ 本研究はOakforest-PACS@JCAHPC、富岳@理研を使用しました。

# JAEAにおける原子力流体解析

- 原子炉の安全性向上、廃炉に向けた熱流動解析
  - 単相流解析→JUPITER, SPIRAL, ASFRE,...
  - 混相流解析→JUPITER, TPFIT, ACE3D, ...
- 放射性物質の環境動態解析
  - 大気・海洋拡散解析→WRF, ROMS, WSPEEDI,...
  - 局所風況解析→LOHDIM-LES, CityLBM
  - 河川・湖沼解析, 港湾解析, ...
- 新型炉開発に向けた流体解析
  - 加速器駆動核変換システム設計→JUPITER
  - 高速炉熱流動解析→SPIRAL, SERAPHIM
  - 核融合プラズマ流体解析→GT5D



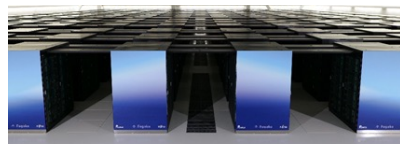
→エクサスケール原子力流体解析に向けた基盤技術開発

- メニーコア最適化、行列解法
- In-Situ可視化



核融合プラズマ解析

# 京からOakforest-PACS (OfP)を経て富岳へ

	京(2011) 	Oakforest-PACS(2016) 	富岳(2020) 
プロセッサ	SPARC64VIIIfx	KNL	A64FX
演算性能(Pflops)	11.3	25.0 ( 2.2x)	537.2 (48x)
電力性能比(Pflops/MW)	0.8	9.2 (11.5x)	14.8 (18x)
コア数/ノード	8	68 ( 8.5x)	48+2/4 ( 6x)
演算性能/ノード(Gflops)	128	3,046 ( 24x)	3,300 (26x)
メモリ帯域/ノード(GB/s)	64	480( 7.5x)	1024(16x)
メモリ/ノード(GB)	64	16(MCDRAM)/96(DDR4)	32(HBM2)
通信性能/ノード(GB/s)	5x4=20	12.5 ( 0.6x)	6.8x6=40.8 ( 2x)

- OfPは富岳と多くの共通点があり、技術実証環境として重要な役割を果たした
    - 50+コア、512bit SIMD演算のメニーコアCPUによる~3TFlopsの演算性能
    - MCDRAM/HBM2による広帯域メモリ
    - 演算性能向上と比べて通信性能向上は限定的
- エクサスケール計算における省通信技術の重要性を早期に認識できた

# ペタスケール時代の省通信技術

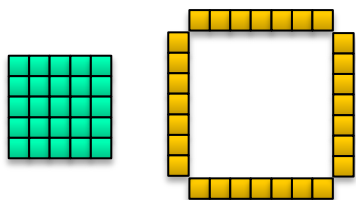
[Idomura, Int. J. HPC Appl. 2014]

## OpenMP通信スレッドによる通信オーバーラップ

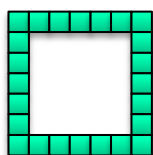
- 1対1通信 (同期, 非同期)
  - 袖通信
- 集団通信
  - データ転置
  - 縮約通信

### 袖通信

1) 袖通信と中心部の演算を同時処理

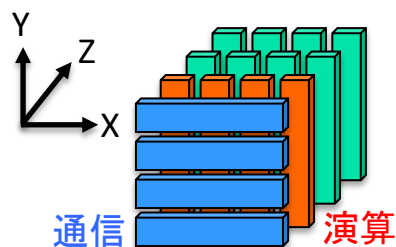


2) 表面部の演算

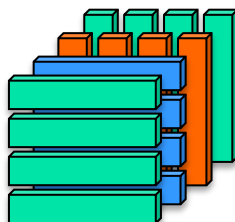


### データ転置

1) 異なるZにおける演算とデータ転置を同時処理

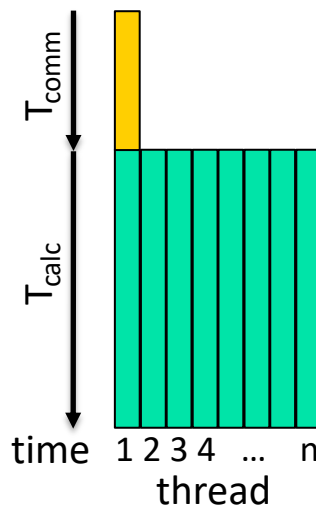


2) パイプライン処理



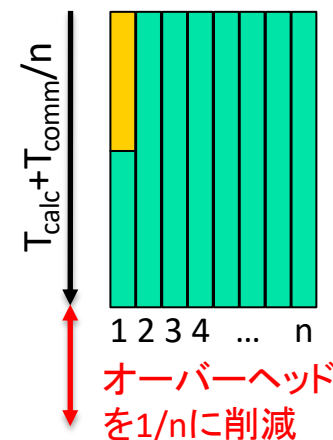
```
!$OMP PARALLEL
!$OMP MASTER
  call MPI communication
!$OMP END MASTER
!$OMP DO SCHEDULE(DYNAMIC)
  do
    call calculation
  enddo
!$OMP END PARALLEL
```

オーバーラップなし



■ calc. ■ comm.

オーバーラップ



→演算に比べて通信コストが小さく、オーバーラップが有効

# ペタスケール時代の省通信技術

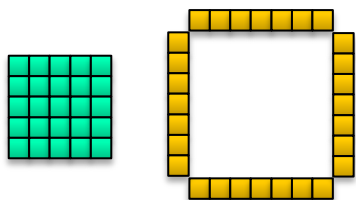
[Idomura, Int. J. HPC Appl. 2014]

## OpenMP通信スレッドによる通信オーバーラップ

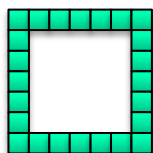
- 1対1通信 (同期, 非同期)
  - 袖通信
- 集団通信
  - データ転置
  - 縮約通信

### 袖通信

1) 袖通信と中心部の演算を同時処理

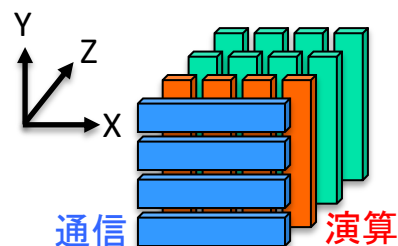


2) 表面部の演算

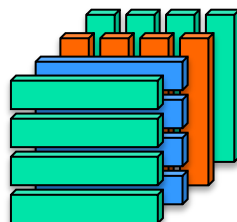


### データ転置

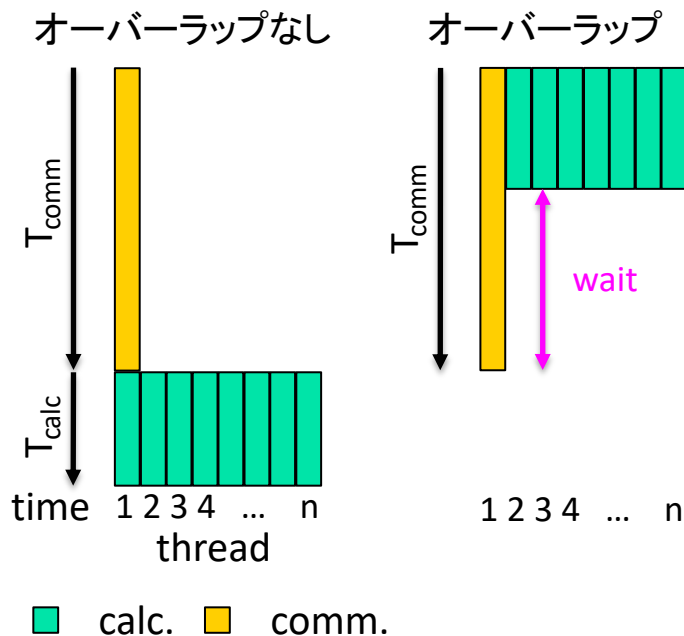
1) 異なるZにおける演算とデータ転置を同時処理



2) パイプライン処理



```
!$OMP PARALLEL
!$OMP MASTER
  call MPI communication
!$OMP END MASTER
!$OMP DO SCHEDULE(DYNAMIC)
  do
    call calculation
  enddo
!$OMP END PARALLEL
```



→演算加速によって通信と演算のコストが逆転し、通信を隠しきれなくなってきた

# エクサスケール時代の省通信型行列解法

流体解析コードに共通する大規模連立一次方程式 $Ax=b$ のクリロフ部分空間法

**Algorithm** Generalized Conjugate Residual method

**Require:**  $Ax = b$ , Initial guess  $x_0$

- 1:  $r_0 := b - Ax_0, p_0 := r_0$
- 2: **for**  $j = 0, 1, 2, \dots$  until convergence **do**
- 3:  $\alpha_j := \langle Ap_j, r_j \rangle / \langle Ap_j, Ap_j \rangle$
- 4:  $x_{j+1} := x_j + \alpha_j p_j$
- 5:  $r_{j+1} := r_j - \alpha_j Ap_j$
- 6:  $\beta_j := \langle Ap_j, Ar_{j+1} \rangle / \langle Ap_j, Ap_j \rangle$
- 7:  $p_{j+1} := r_{j+1} + \beta_j p_j$
- 8:  $Ap_{j+1} := Ar_{j+1} + \beta_j Ap_j$
- 9: **end for**



**Algorithm** Preconditioned CA-GMRES method

**Require:**  $Ax = b$ , Initial guess  $x_1$

- 1:  $\tilde{A} = D^{-1}A$  ( $D_{ii} = \max(|A_{i1}|, |A_{i2}|, \dots, |A_{in}|)$ )
- 2:  $d = D^{-1}b / |D^{-1}b|$
- 3:  $z = x / |D^{-1}b|$
- 4:  $B = [e_2, e_3, \dots, e_{s+1}]$
- 5: **for**  $i = 1, 2, \dots$  until convergence **do**
- 6:  $r := d - \tilde{A}z_i$
- 7:  $r' := M_{FP16}^{-1}r, \beta := \|r'\|, q := r' / \beta, \zeta := \beta e_1$
- 8: Compute SpMV's  $\hat{V} := [M_{FP16}^{-1}\tilde{A}q, \dots, M_{FP16}^{-1}\tilde{A}^s q]$
- 9:  $V := [q, \hat{V}]$
- 10: Compute QR factorization via ( $W := V^T V$ ;
- 11: Cholesky decomposition  $W = R^T R$ ;  $Q := VR^{-1}$ )
- 12:  $H := RBR^{-1}$
- 13:  $\bar{H} := \text{Givens rotation}(H) = G_1 G_2 \dots G_s H$
- 14:  $\bar{\zeta} := G_1 G_2 \dots G_s \zeta$
- 15:  $y := \bar{H}^{-1} \bar{\zeta}$
- 16:  $z_{i+1} = z_i + Qy = z_i + VR^{-1}y$
- 17: **end for**
- 18:  $x := |D^{-1}b|z$

クリロフ部分空間法の通信処理

- 行列ベクトル積 $Ax$ における袖通信
- 内積計算 $\langle x, y \rangle$ における集団通信

## ■ 2つの省通信化手法

手法1: 複数反復をまとめて処理する省通信クリロフ部分空間法により集団通信を削減

手法2: 省通信前処理Mによって問題を解きやすい形に変換し、反復(通信)回数を削減

$$Ax=b \rightarrow M^{-1}Ax=M^{-1}b \quad (M^{-1} \sim A^{-1} \text{に近づくほど反復回数を削減できる})$$

→問題の物理的特徴と計算機アーキテクチャに応じて最適な行列解法を設計



# 5次元プラズマ流体解析コードGT5D

[Idomura,CPC2008,SC11,NF2009,JCP2016,SC20]

- 5次元ボルツマン方程式 ( $R, \zeta, Z, v_{\parallel}, \mu = m_s v_{\perp}^2 / B$ )

$$\frac{\partial f_s}{\partial t} + \{f_s, H_s[\phi]\} = \sum_{s'} C_{s'}[f_s] + S_{source} + S_{sink}$$

$$\{f_s, H_s[\phi]\} = \frac{d\mathbf{R}}{dt} \cdot \frac{\partial f}{\partial \mathbf{R}} + \frac{dv_{\parallel}}{dt} \frac{\partial f}{\partial v_{\parallel}}$$

乱流電場と熱運動による移流

$$C_{s'}[f_s] = -\frac{\partial}{\partial \mathbf{v}} \cdot (\mathbf{D}_{1s'} f_s) + \frac{\partial^2}{\partial \mathbf{v} \partial \mathbf{v}} : (\mathbf{D}_{2s'} f_s)$$

クーロン衝突

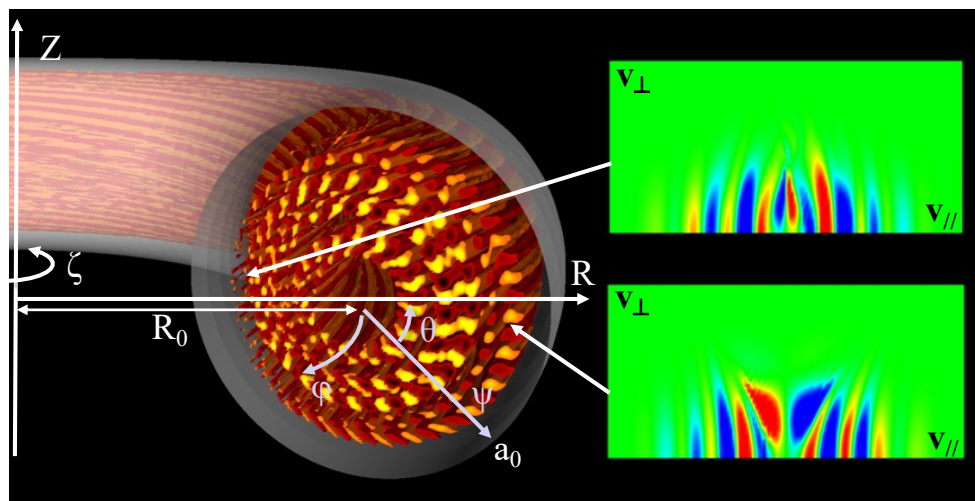
- 4次精度無散逸保存型差分 [Morinishi,JCP1998; Idomura,JCP2007]
- 2次精度半陰的ルンゲ・クッタ法 [Zhong,JCP1998]

→熱運動を含む線形4次元移流項に対する差分陰解法ソルバが8割以上のコスト

- 乱流電場のポアソン方程式 ( $\psi, \theta, \varphi$ )

$$-\nabla^2 \phi = \frac{1}{\epsilon_0} \sum_s q_s n_s [f_s]$$

- 1次元FFT ( $\varphi$ )
- 2次元有限要素法 ( $\psi, \theta$ )



悪条件でない線形4次元移流項の問題に対して前処理無しクリロフ部分空間法を適用

**Algorithm** Generalized Conjugate Residual method

**Require:**  $Ax = b$ , Initial guess  $x_0$

- 1:  $r_0 := b - Ax_0, p_0 := r_0$
- 2: **for**  $j = 0, 1, 2, \dots$  until convergence **do**
- 3:  $\alpha_j := \langle Ap_j, r_j \rangle / \langle Ap_j, Ap_j \rangle$
- 4:  $x_{j+1} := x_j + \alpha_j p_j$
- 5:  $r_{j+1} := r_j - \alpha_j Ap_j$
- 6:  $\beta_j := \langle Ap_j, Ar_{j+1} \rangle / \langle Ap_j, Ap_j \rangle$
- 7:  $p_{j+1} := r_{j+1} + \beta_j p_j$
- 8:  $Ap_{j+1} := Ar_{j+1} + \beta_j Ap_j$
- 9: **end for**

SpMV  
AXPY

A: 4次元移流項が与える非対称ブロック対角行列



**Algorithm** Communication-Avoiding GMRES method

**Require:**  $Ax = b$ , Initial guess  $x_0$

- 1:  $B = [e_2, e_3, \dots, e_{s+1}]$
- 2: **for**  $i = 0, 1, 2, \dots$  until convergence **do**
- 3:  $r := b - Ax_i, \beta := \|r\|, q := r/\beta, \zeta := \beta e_1$
- 4: Compute SpMVs or MPK  $\hat{V} = [Aq, \dots, A^s q]$
- 5:  $V := [q, \hat{V}]$
- 6: Compute QR factorization via ( $D := V^T V$ ;
- 7: Cholesky decomposition  $D = R^T R$ ;  $Q := VR^{-1}$ )
- 8:  $H := RB\bar{R}^{-1}$
- 9:  $\bar{H} :=$ Givens rotation( $H$ )=  $G_1 G_2 \dots G_s H$
- 10:  $\bar{\zeta} := G_1 G_2 \dots G_s \zeta$
- 11:  $y := \bar{H}^{-1} \bar{\zeta}$
- 12:  $x_{i+1} := x_i + Qy = x_i + VR^{-1}y$
- 13: **end for**

SpMV  
SYRK  
GEMV

CA-GMRES[Hoemmen, PhD2010]

## ■ 一般化共役残差法 (GCR) と一般化最小残差法 (CA-GMRES, s=22) の比較

	GCR	CA-GMRES
Number of SpMVs	598	639
All_reduce/iteration	1	1/s
Computation [Flop/grid]	47.0	84.8→59.7
Memory access [Byte/grid]	128.0	60.0→43.3
Elapse time on KNL [ns/grid]	0.52	0.35

直交行列Qを計算しない  
最適化実装によりメモリ  
アクセスをさらに3割削減



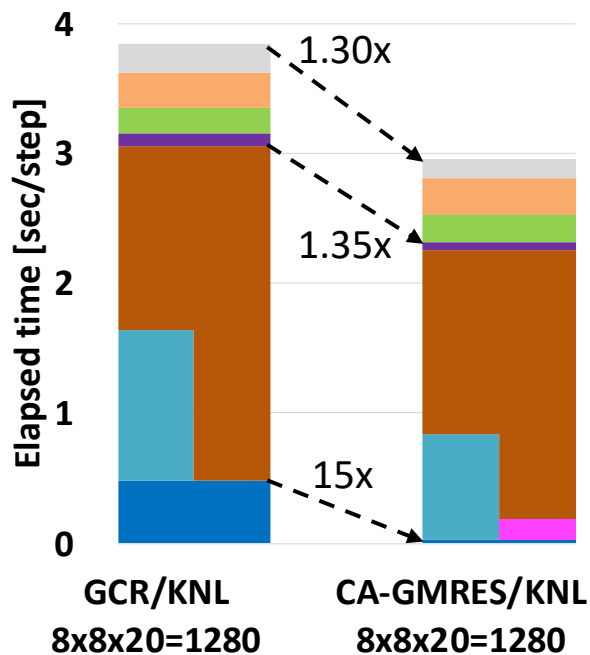
# Oakforest-PACS全系を用いたGT5Dの処理性能

[Idomura, ScalA2017]

## ■ GCRとCA-GMRESの比較

JT-60規模～60億格子@1280ノード

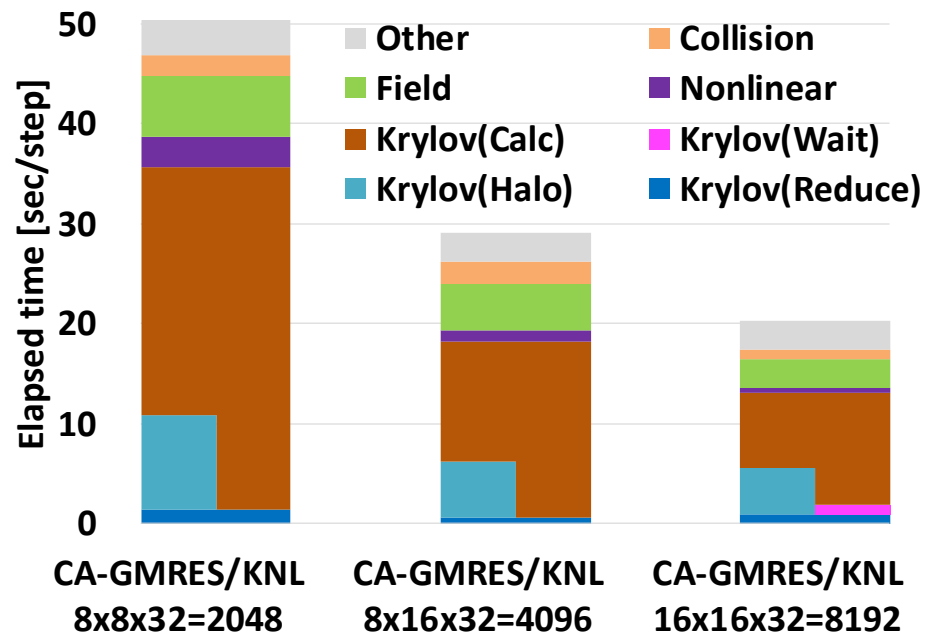
$(N_R, N_Z, N_\zeta, N_{v//}, N_\mu) = (320, 320, 32, 96, 20)$



## ■ 8192ノードまでの強スケーリング

ITER規模～1500億格子@2k～8kノード

$(N_R, N_Z, N_\zeta, N_{v//}, N_\mu) = (768, 768, 64, 128, 32)$



## ■ Oakforest-PACS全系規模の8192ノードまで良好な強スケーリング

- CA-GMRESによりGCRにおける集団通信のボトルネックを解消
- 密行列計算+省通信化によりソルバ全体で35%の性能向上

# 3次元多相多成分熱流動解析コードJUPITER

[Yamashita,NED2017]

## ■ 計算モデル

非圧縮流体モデル

$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \cdot \{ \mu [\nabla \mathbf{u} + (\nabla \mathbf{u})^T] \} + \mathbf{g} + \mathbf{F}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = -\frac{1}{\rho C_v} \nabla \cdot (\lambda \nabla T) + \frac{Q}{\rho C_v}$$

Volume of Fluid法

$$\frac{\partial \phi_l}{\partial t} + \nabla \cdot (\mathbf{u} \phi_l) = \phi_l \nabla \cdot \mathbf{u}$$

物理量 (S:固相、L:液相、G:気相)

$$Y(\phi_l) = \sum_l (Y_l^S \phi_l^L + Y_l^L \phi_l^L) + Y^G \left[ 1 - \sum_l (\phi_l^S + \phi_l^L) \right]$$

## ■ 数値計算法

- 3次元直交格子でMPI並列化
- 差分法(拡散:2次中心差分、移流:5次WENO)
- 半陰解法時間積分

→圧力ポアソンソルバが9割以上の計算コスト

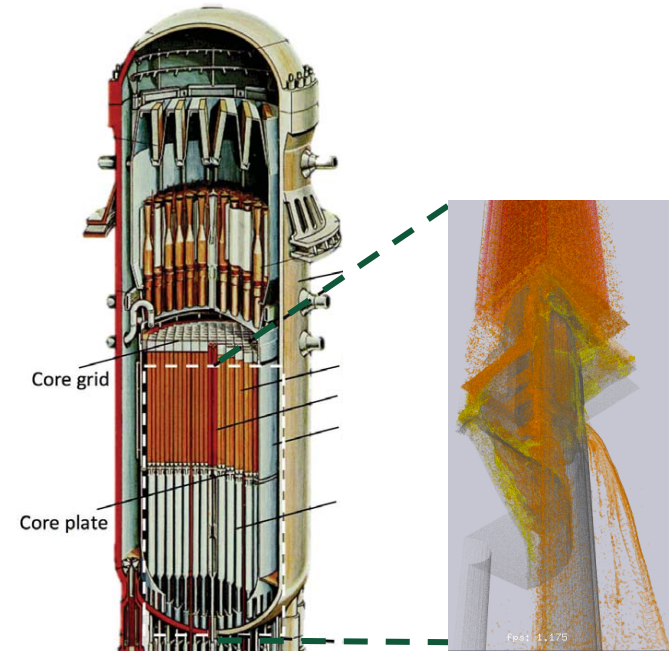
## ■ 圧力ポアソン方程式

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^* - \nabla \cdot \left( \frac{\Delta t}{\rho} \nabla p \right) = 0$$

- 直交格子の2次中心差分(7点ステンシル)
- 多相流の大きな密度比( $\sim 10^7$ )による悪条件問題
- 大規模なマルチスケール問題で更に条件が悪化

→適切な前処理の設計が必須

燃料溶融複雑系の過酷事故解析  
( $\text{UO}_2$ , Zry,  $\text{B}_4\text{C}$ , SUS, and Air)



# 省通信マルチグリッド前処理付き共役勾配(MGCG)法

[Idomura,ScalA2018]

- MG前処理によるCG法の反復回数削減によりAllReduceを削減
- 幾何的MG前処理の省通信化
  - All Reduceを含まない前処理付チェビシェフ反復(P-CI)をスムーザとして選択
    - 最もロバーストな収束特性を示すスムーザの一つ[Baker,SIAM2011]
    - 定常反復の近似解を $[\lambda_{\min}, \lambda_{\max}]$ でシフト規格化したチェビシェフ多項式で向上
    - 最大/最小固有値 $\lambda_{\min}, \lambda_{\max}$ の計算に省通信ランチョス法[Hoemmen,PhD2010]
  - 混合精度による計算量と袖通信の削減
    - 精度に敏感なクリロフ部分空間法(CG,CA-Lanczos)を倍精度処理
    - 精度要求が低いスムーザの反復計算(P-CI)を単精度処理

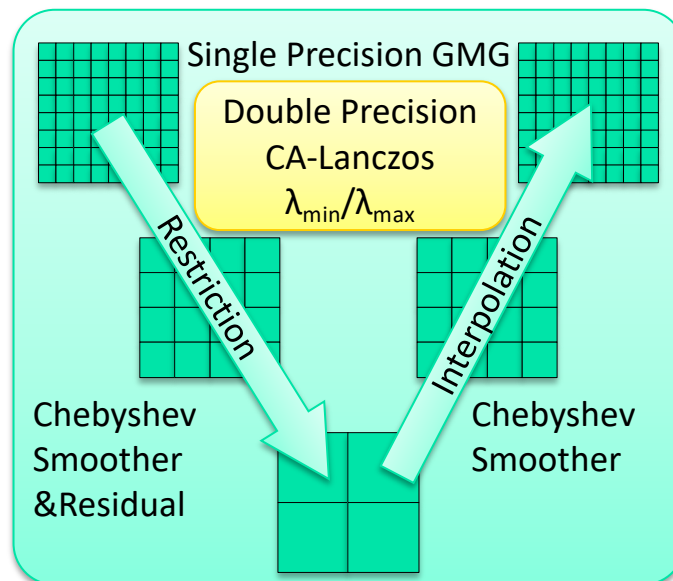
Algorithm 4 Preconditioned Chebyshev iteration (P-CI) method

Input:  $Ax = b$ , Initial guess  $x_0$ , Approximate minimum/maximum eigenvalues of  $A\tilde{M}^{-1}$ ,  $\lambda_{\min}, \lambda_{\max}$

Output: Approximate solution  $x_i$

- 1:  $d := (\lambda_{\max} + \lambda_{\min})/2, c := (\lambda_{\max} - \lambda_{\min})/2$
- 2:  $r_0 := b - Ax_0, z_0 := \tilde{M}^{-1}r_0, p_0 := z_0, \alpha_0 := 1/d$
- 3: for  $i = 1, 2, \dots$  until convergence do
- 4:  $x_i := x_{i-1} + \alpha_{i-1}p_{i-1}$
- 5:  $r_i := b - Ax_i$
- 6:  $z_i := \tilde{M}^{-1}r_i$
- 7:  $\beta_i := (\alpha_{i-1}c/2)^2$
- 8:  $\alpha_i := 1/(d - \beta_i/\alpha_{i-1})$
- 9:  $p_i := z_i + \beta_i p_{i-1}$
- 10: end for

no All\_Reduce



# Oakforest-PACS全系を用いたJUPITERの処理性能

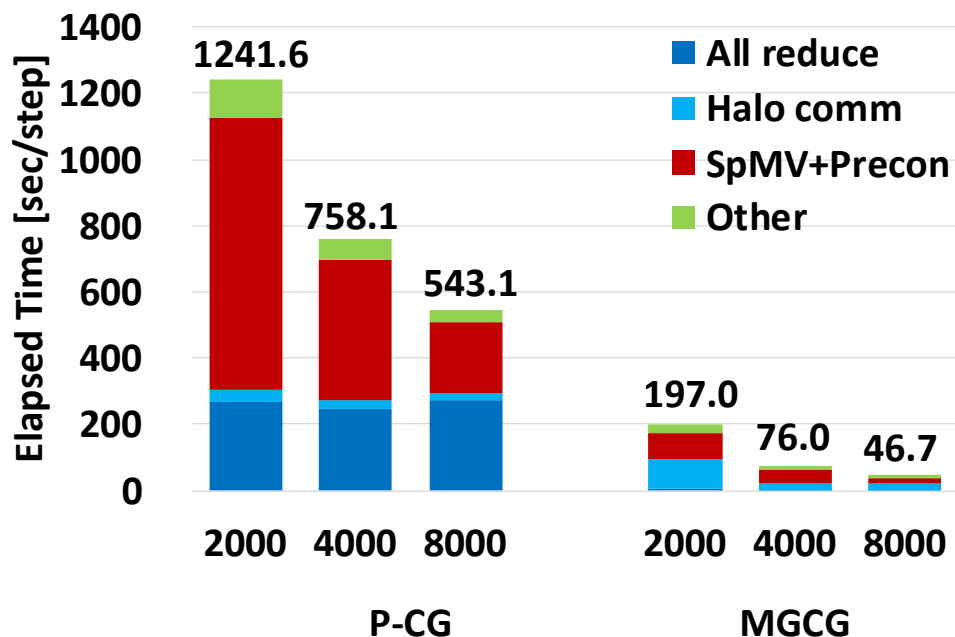
[Idomura, ScalA2018]

- 900億格子問題における8000ノードまでの強スケーリング

固相/液相4成分 ( $\text{UO}_2$ , Zry,  $\text{B}_4\text{C}$ , SUS)+気相、3200x2000x14160格子、条件数 $\sim 6 \times 10^9$

P-CG: ブロックヤコビILU前処理付きCG法

MGCG: 省通信マルチグリッド前処理付きCG法



sec/step	P-CG	MGCG	Ratio
SpMV+Precon	218.3	17.9	12.2x
Other	32.6	9.6	3.4x
Halo comm	21.4	17.2	1.2x
All reduce	270.8	1.9	142.5x
Total	543.1	46.7	11.6x

8000ノードにおけるコスト分布  
P-CG : 26475 反復  
MGCG: 32 反復 (Lv7 x 50step/Lv)

- Oakforest-PACS全系規模の8000ノードまで良好な強スケーリング

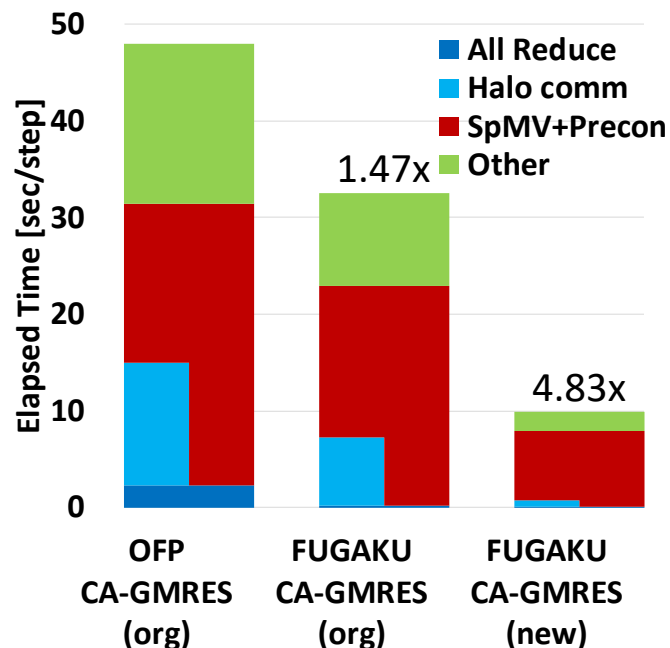
- 反復回数を1/800に削減し、集団通信 (All reduce) のコストを1/100以下に抑制
- 収束特性の改善により差分計算 (SpMV) のコストを1/10以下に削減
- 8000ノードにおいて11.6xの性能向上

# 富岳におけるGT5DとJUPITERの処理性能

[Idomura,SC20, Ina,ScalA2021]

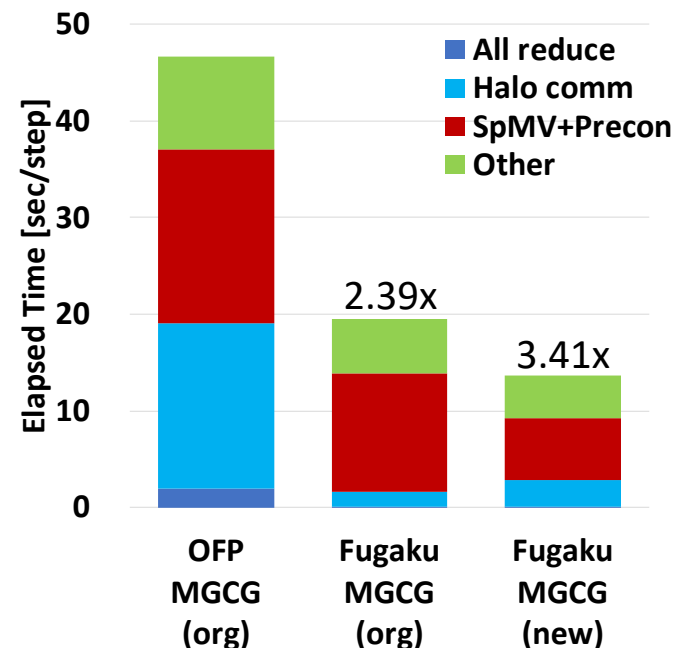
## ■ GT5D (CA-GMRES)

1000億格子@1440ノード



## ■ JUPITER (MGCG)

900億格子@8000ノード



## ■ 同アルゴリズムで1.5x~2.4xの性能向上

- メモリ帯域比2.1倍に対して演算処理で約1.5倍の性能向上
- 通信帯域比3.3倍に対して通信処理で10倍以上の性能向上

OfP(OmniPath): FatTree, All Rduce(16B)/500μsec@8192nodes

富岳(TofuD): 6Dtorus, All Rduce(16B)/66μsec@27648nodes

## ■ 富岳向けに開発した半精度前処理によってさらに3.4x~4.8xの性能向上



# まとめ

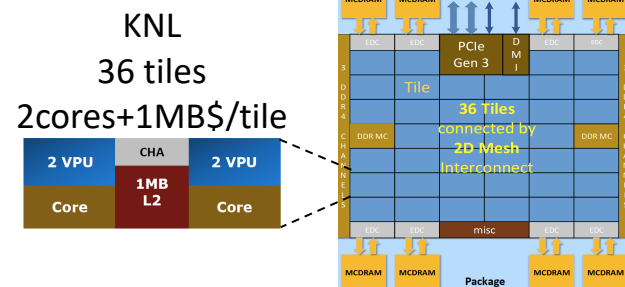
Oakforest-PACSを活用して富岳向けの省通信型行列解法を開発した

- OfP全系を駆使した1000億格子規模の大規模原子力流体解析を実現
  - GT5D：省通信GMRES法によってGCR法から1.35xの性能向上
  - JUPITER：省通信MG前処理付きCG法によってP-CG法から11.6xの性能向上
- 上記ソルバを富岳に移植し高性能計算を実現
  - メモリ帯域幅、通信性能により同アルゴリズムで最大2.4xの性能向上
  - 半精度前処理を追加することでさらに最大4.8xまで性能向上

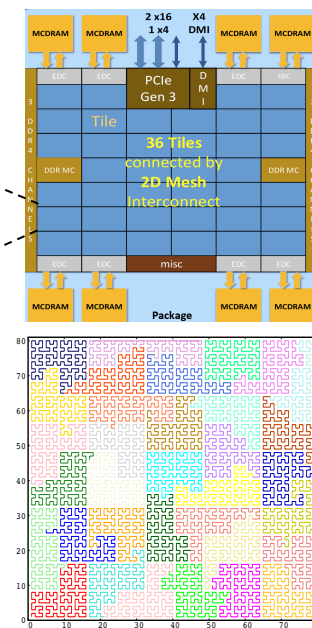
## ■ OfPと富岳の違い

- 半精度演算を活用することで性能向上が可能
- 12コア/CMG構成となりスレッド並列化が容易
- 8GB/CMGなのでメモリの並列化要求が厳しい

cf.OfP:16GB(MCDRAM)+96GB(DDR4)=112GB/node



空間充填曲線による動的スレッド割付  
→ブロック分割から  
1.9x性能向上



# OfPを用いて得られた主な成果

## ■ 省通信型行列解法

- [1] Y. Idomura, T. Ina, A. Mayumi, S. Yamada, K. Matsumoto, Y. Asahi, and T. Imamura, 2017 IEEE/ACM 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), pp.1-8.
- [2] Y. Idomura, T. Ina, A. Mayumi, S. Yamada, T. Imamura, Lecture Notes in Computer Science, vol. 10776, 257–273 (2018).
- [3] Y. Idomura, T. Ina, S. Yamashita, N. Onodera, S. Yamada, and T. Imamura, 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), pp.17-24.
- [4] Y. Ali, N. Onodera, Y. Idomura, T. Ina, T. Imamura, 2019 IEEE/ACM 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), pp.1-8.
- [5] Y. Idomura, T. Ina, Y. Ali, T. Imamura, in 2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1318-1330 .
- [6] T. Ina, Y. Idomura, T. Imamura, S. Yamashita, N. Onodera, 2021 IEEE/ACM 12th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), pp.1-8.

## ■ In-Situ可視化

- [7] T. Kawamura, T. Noda, Y. Idomura, Supercomputing Frontiers and Innovations 4, 43-54 (2017).
- [8] T. Kawamura, Y. Idomura, Journal of Visualization volume 23, 695–706 (2020).

## ■ 関連するオープンソースソフトウェア

反復解法ライブラリPARCEL: <https://ccse.jaea.go.jp/software/PARCEL/>  
In-Situ可視化ツールIn-Situ PBVR: [https://ccse.jaea.go.jp/software/In-Situ\\_PBVR/](https://ccse.jaea.go.jp/software/In-Situ_PBVR/)