高性能・変動精度・高信頼性数値解析手法とその応用

Numerical Methods with High-Performance, Trans Precision and High Reliability

jh190042-NAH

研究代表者：中島研吾（東京大学情報基盤センター）

第12回JHPCNシンポジウム（オンライン）
2020年7月9日

ポストムーア時代に向けた低・変動精度演算による新計算原理の必要性

- 低・変動精度演算
  - 計算量・計算時間・消費電力・メモリ容量・I/O削減
- 科学技術計算の多くは倍精度（64bit）で実施されているが，条件の良い問題であれば，単精度（32bit）・半精度（16bit），混合精度でも可能
- 我々は不必要に高精度な計算を実施して貴重な時間やエネルギーを無駄にしている・・・かも知れない
- 昨今，低精度演算の活用は盛ん
  - 混合精度演算の研究，アプリへの適用は既に行われている
  - Approximate Computing：省電力のための研究
    - 元々は画像処理・機械学習
    - FPGAを含むハードウェア，システムソフトウェア，コンパイラ
  - ポストムーア時代へ向けた重要な技術の一つ

高性能・変動精度・高信頼性数値解析手法とその応用：背景と目的

JHPCN（学際大規模情報基盤共同利用・共同研究拠点）
2018/2019/2020年度共同研究課題

代表：中島研吾（東京大学情報基盤センター・理研R-CCS）
副代表：横田理央（東京工業大学）・市村強（東京大学地震研究所）

日米独 3ヶ国
16機関（産学官），30+人

- 通信最適化（Serial：メモリ階層，Parallel：分散）による**高性能**アルゴリズム
- **低・変動精度**演算の採用による省電力・省エネルギーの実現（半精度・単精度・倍精度・4倍精度）
- 精度保証による**高信頼性**アルゴリズムの実現
  - 数値計算による近似解（数値解）は様々な計算誤差を含み，計算結果の信頼性の観点から，数値解の正しさを数学的に保証する必要がある（精度保証）。
  - 低精度・変動精度使用時，悪条件問題には重要，実問題で現れる大規模疎行列・H行列への応用例は非常に少ない

中島研吾（東大）（代表）全体統括・高性能アルゴリズム
市村　強（東大）（副代表）高性能アルゴリズム・地震シミュレーション
横田理央（東工大）（副代表）高性能アルゴリズム・精度保証
藤田航平（東大）高性能アルゴリズム・地震シミュレーション
星野哲也（東大）高性能アルゴリズム・性能最適化
坂本龍一（東大）電力測定・自動チューニング（AT）
有間英志（東大）電力測定・AT
古村孝志（東大）地震シミュレーション
近藤正章（東大）精度保証・電力測定・AT
奥田洋司（東大）工学シミュレーション・精度保証
森田直樹（東大）工学シミュレーション・精度保証
荻田武史（東女大）精度保証・AT
田中一成（早大）精度保証・AT
尾崎克久（芝工大）精度保証・AT
河合直聡（東大）高性能アルゴリズム・量子科学シミュレーション
岩下武史（北大）高性能アルゴリズム
深谷　猛（北大）高性能アルゴリズム
塙　敏博（東大）性能最適化・電力測定
伊田明弘（東大）高性能アルゴリズム
片桐孝洋（名大）高性能アルゴリズム・精度保証・AT
大島聡史（名大）高性能アルゴリズム・性能最適化
櫻井刀麻（名大）精度保証・AT
八代　尚（環境研）大気シミュレーション
井上弘士（九大）精度保証・電力測定・自動チューニング
荒川　隆（RIST）大気シミュレーション
成瀬　彰（NVIDIA）精度保証・AT
堀越将司（インテル）精度保証・AT
Gerhard Wellein（FAU Erlangen-Nürnberg, Germany)高性能アルゴリズム・量子科学シミュレーション・SELL-C-σ
Achim Basermann（DLR, Germany）高性能アルゴリズム・量子科学シミュレーション
Osni Marques（Lawrence Berkeley National Laboratory, USA）高性能アルゴリズム・AT

目的(1/2)

- エクサスケールシステムにおける高性能数値アルゴリズム実現には，メモリ・ネットワークの階層の深化に対応した通信最適化(Serial，Parallel)，省電力・省エネルギー(以下「省電力」)に向けた検討が必要である。
- Approximate Computing(S. Mittal, A Survey of Techniques for Approximate Computing, ACM CSUR 48-4, 2016)は，低精度・混合精度演算の積極的活用により計算時間短縮，消費電力削減を図る試みであり，従来は画像認識等の計算精度の要求されない分野を対象としていたが，昨今は数値計算において半精度から四倍精度まで演算精度を動的に変動させる変動精度(Transprecision)も含めた研究が進められている。
- 数値計算による近似解(数値解)は様々な計算誤差を含み，計算結果の信頼性の観点から，数値解の正しさを数学的に保証する必要があり，低精度・混合／変動精度使用時，悪条件問題には重要であるが，実問題で現れる大規模疎行列・H行列への応用例はほとんどない。
- 本研究では，JHPCNシステム群の中で消費電力当たり計算性能(GFLOPS/W値)の高いシステムを主たるターゲットとして，以下を実施する:

目的(2/2)

① 疎行列演算，H行列演算，ステンシル演算等の代表的数値アルゴリズム，各アプリケーション(地震，大気科学，量子科学，構造力学)について，Serial・Parallel通信最適化に着目した高性能最適化手法を各システムに実装し，低精度・混合／変動精度演算について検討し，消費電力を測定する：**アルゴリズム開発・消費電力測定**

② 疎行列演算やH行列演算を対象として，特に悪条件問題における実用的な精度保証法を確立する。更に①の各アルゴリズム，アプリケーションについて所望の結果精度達成という条件下で，計算時間や消費電力を最小化する最適演算精度を自動チューニング技術により動的に制御する手法を確立する：**精度保証・自動チューニング**

③ 本研究によって開発された高性能・変動精度・高信頼性数値解法を，自動チューニング機構を有するアプリケーション開発・実行環境ppOpen-HPC(JST-CREST 2011-2018, https://github.com/Post-Peta-Crest/ppOpenHPC)及び(計算＋データ＋学習)融合を実現する革新的ソフトウェア基盤h3-Open-BDEC(科研費基盤S 2019-2023, http://nkl.cc.u-tokyo.ac.jp/h3-Open-BDEC/)に実装し，東大Oakforest-PACS(OFP)，東大Reedbush(RBH，RBL)，東大Oakbridge-CX(OBCX)，東工大Tsubame-3(TSB3)で公開する。将来的には「富岳」も含む他のセンターのスパコンへの導入も視野に入れる。

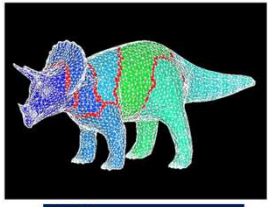
ppOpen-HPC

自動チューニング機構を有する
アプリケーション開発フレームワーク

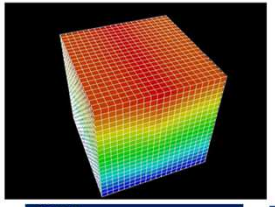
- JST/CREST＋DFG/SPPEXA（2011-2018）
- 7機関，50名以上が参加
- オープンソースソフトウェア
  - ✓ http://ppopenhpc.cc.u-tokyo.ac.jp/
  - ✓ https://github.com/Post-Peta-Crest/ppOpenHPC
  - ✓ 英語ドキュメント，MITライセンス

DFG Deutsche Forschungsgemeinschaft

JST 独立行政法人 科学技術振興機構 Japan Science and Technology Agency

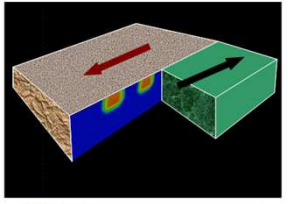


FEM
Finite Element Method

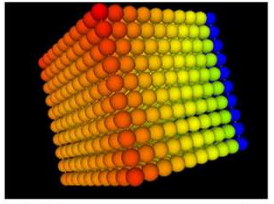


FDM
Finite Difference Method

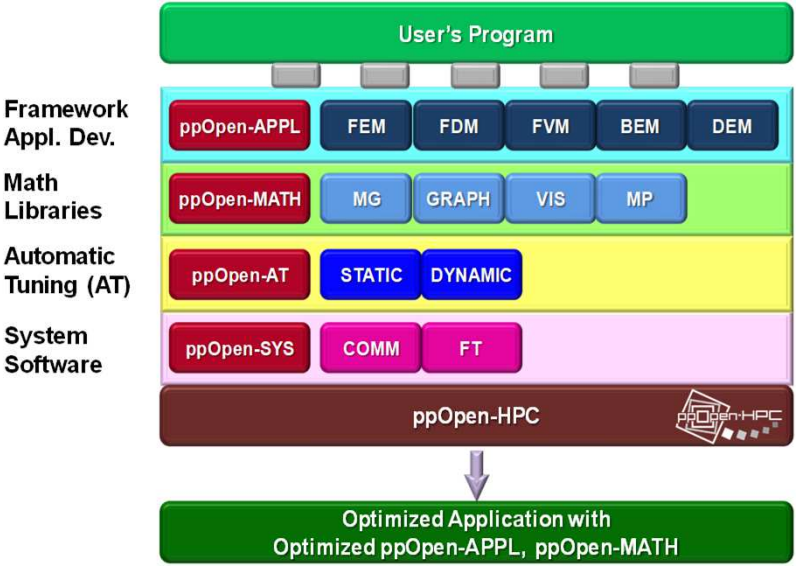
FVM
Finite Volume Method



BEM
Boundary Element Method

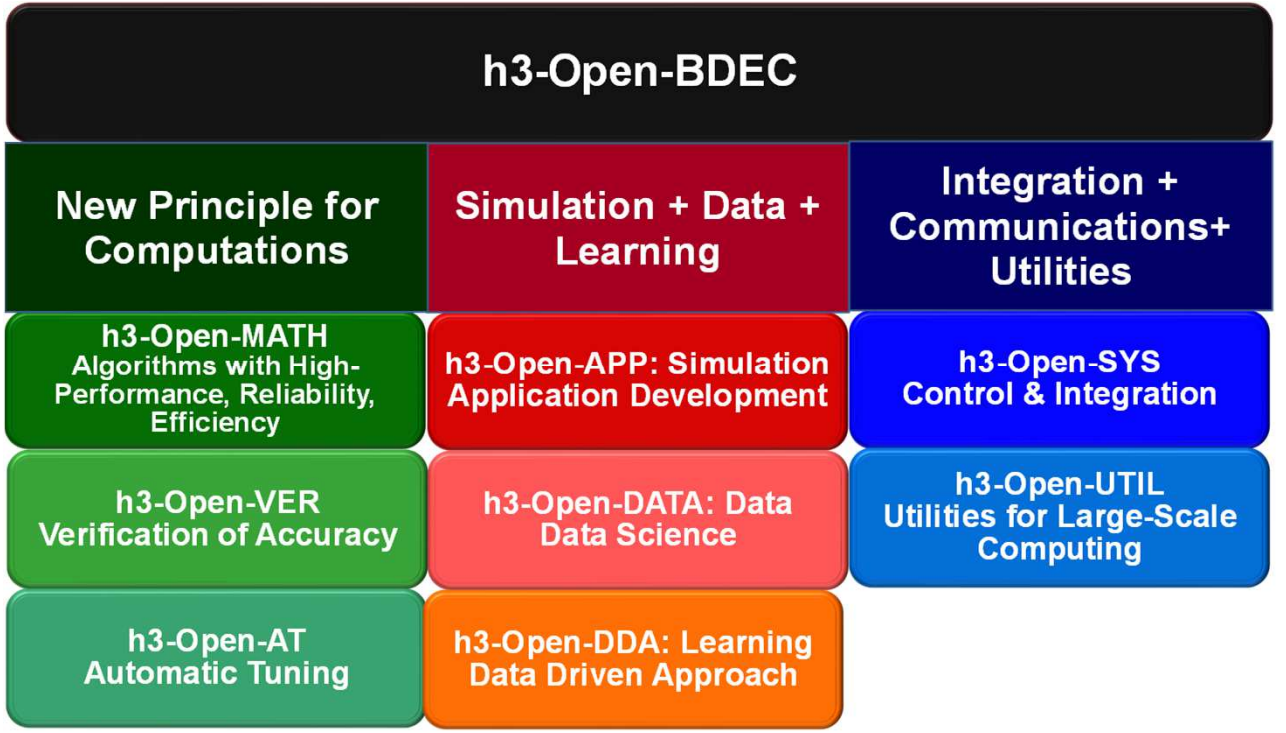


DEM
Discrete Element Method



h3-Open-BDEC

Extension of ppOpen-HPC for Integration of (Simulation + Data + Learning) in the Exa/Post Moore Era, Initial Target: BDEC@U.Tokyo



h3= Hierarchical, Hybrid, Heterogeneous

Big Data & Extreme Computing

研究の意義

- 本研究は，最先端のスパコン向けに開発された高性能数値アルゴリズムに対して，低精度・混合／変動精度演算を適用し，精度保証，そのための自動チューニング手法を開発する試みとしては初めてのものであり，様々なアプリケーションへの適用により，低精度・混合／変動精度演算の科学技術シミュレーションへの有効性を検証できる。
- 開発したアルゴリズム，アプリケーションの消費電力の直接測定によって，各計算の特性と低精度・混合／変動精度演算の有効性を消費電力の観点から検討可能となる。

本研究の沿革(1/3)

- 本研究は，計算科学・計算機科学・数値アルゴリズム分野の研究者の協力のもと，様々な数値アルゴリズムの最新アーキテクチャに向けた最適化，低精度・変動精度導入による高速化，消費電力節約，および精度保証に基づく最適精度選択のための自動チューニング選択手法の確立と実アプリケーションでの検証を目指したものである。
- 本研究では以下の各項目について研究開発を実施する
  - ステンシル計算(①構造格子，②半構造格子)
  - 疎行列演算(③一般行列，④悪条件問題，⑤Adaptive CG(局所前処理による))
  - ⑥H行列，⑦精度保証，⑧消費電力測定，⑨自動チューニング手法，の各項目についての研究開発を実施する。
- 本研究は2018年度から，3年計画として実施している。

研究実施項目の概要

実施項目	担当者	目標精度	概 要
①ステンシル計算：構造格子	深谷・岩下・古村・成瀬	半・単	• 三次元地震波動伝搬解析コード「Seism3D」（古村他） • 深谷・岩下の時空間タイリング手法による最適化 • 変動精度演算を適用した大規模シミュレーションによる性能検証，精度評価
②ステンシル計算：半構造格子	深谷・岩下・八代・荒川・成瀬	半・単	• 三次元全球大気解析コード「NICAM」（八代他） • 時空間タイリング手法，変動精度演算を適用した大規模シミュレーションによる性能検証，精度評価
③疎行列演算：ppOpen-MATH	岩下・中島・星野・塙・大島・奥田・森田・Wellein	半・単・倍	• ppOpen-HPC疎行列反復法ソルバー群（Pipeline法，Dynamic Loop Scheduling，hCGA法）（中島他） • SELL-C-σ，変動精度演算機能を適用 • 不均質場固体力学・地下水流れ，FrontISTR（奥田）〔2〕等による性能検証，精度評価
④疎行列演算：pK-Open-SOL（悪条件）	河合・伊田・中島・星野・塙・大島・Wellein・Basermann	半・単・倍・4倍	• 量子科学シミュレーションの固有値計算の過程で得られる悪条件問題を対象とする前処理付き反復法ソルバー • 悪条件問題向けのブロック化・対角シフト機能の他，世界初の分散並列リオーダリング手法を実装（河合他） • SELL-C-σ，変動精度演算機能を適用 • 量子科学分野の大規模シミュレーションによる性能検証，精度評価，悪条件問題における単精度実数演算適用の可能性について併せて検討
⑤疎行列演算：Adaptive CG	市村・藤田・堀越・成瀬	半・単・倍	• 三次元地震解析コード「GHYDRA」（市村・藤田）で使用されている手法，混合精度演算も導入済 • 半精度も含めた変動精度演算，大規模シミュレーションによる性能検証，精度評価
⑥H行列演算：HACApK	伊田・岩下・横田・星野・塙・大島・河合・Basermann	単・倍・4倍	• ppOpen-HPCの一環として開発（伊田・岩下） • 変動精度演算の実装，悪条件問題向けの前処理手法 • 電磁気学，地震発生サイクル等の大規模シミュレーションによる性能検証，精度評価 • 悪条件問題における単精度実数演算適用の可能性について併せて検討
⑦精度保証手法	荻田・片桐・尾崎・奥田・横田・井上・近藤・森田・堀越・成瀬		③～⑥を対象として，精度保証を適用可能な行列クラスを抽出し，悪条件問題における実用的な精度保証法を確立するとともに，数値解の精度を改善するための効率的な反復改良法（Iterative Refinement）を精度保証法と融合
⑧消費電力測定	近藤・井上・坂本・有間・塙		①～⑥を対象として，近藤，井上等によるJST-CREST「ポストペタスケールシステムのための電力マネージメントフレームワークの開発」の成果，知見を適用し，RAPL（Running Average Power Limit）等によりReedbush-L，東大情基センのクラスタ（IBM P9+NVIDIA V100）等を使用した電力測定を実施し，変動精度演算による消費電力削減効果を実証
⑨自動チューニング手法	片桐・荻田・横田・井上・近藤・有間		上記①～⑧の知見を元に，所望の精度で計算時間や消費電力を最小化する最適演算精度を，アプリケーション・係数行列の性質，問題サイズ，ハードウェア環境等に基づき自動チューニング技術によって動的に制御する手法を確立，ppOpen-AT（ppOpen-HPCの自動チューニング機構に実装

研究計画：3年計画（2018～2020年度）

- **1年目（FY.2018）**
  - Intel Xeon Phi（KNL）・NVIDIA P100向け最適化，実装及び精度評価
  - 1年目は電力測定も含めて「一通りやってみた」という感じ
- **2年目（FY.2019）**
  - 精度保証手法の確立
- **3年目（FY.2020）**
  - 自動チューニング手法確立
  - ppOpen-HPC等へ実装・公開

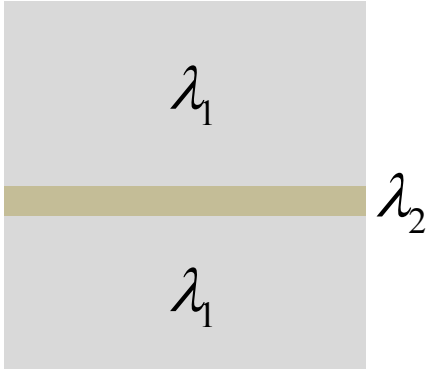
本研究の沿革(2/3)

- 2018年度は，上記①～⑩の項目について，(1)性能最適化，(2)多様な計算精度，(3)精度保証，(4)実アプリケーションを対象とした消費電力測定，を中心として予備的な検討を実施し，学会等で発表した他，ISC18，SC18，ISC19，SC19等の国際会議の展示ブースでも紹介した。
- 2018年度の段階で低精度・混合／変動精度演算をアルゴリズム，最適化，精度保証，消費電力まで含めて扱った研究事例はほとんどなかったが，SC19(2019年11月)では，低精度・混合／変動精度演算を数値計算に取り入れた研究事例が多数見られた。

三次元不均質ポアソン方程式　ICCG法　[KN 2018]

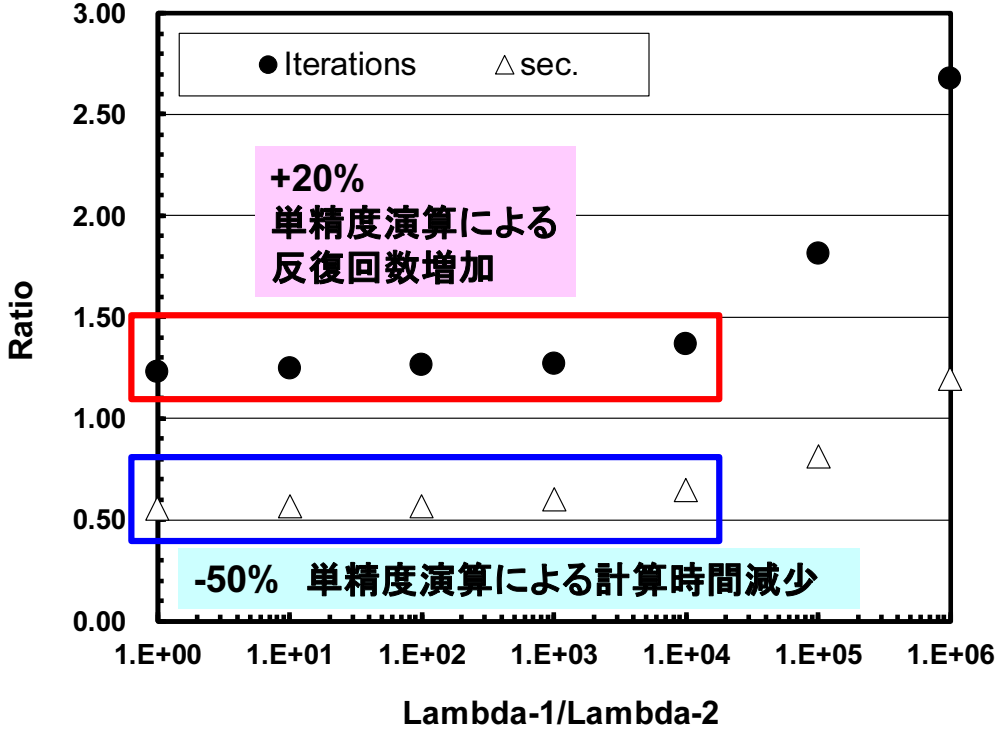
条件数(λ_1/λ_2)と反復回数・計算時間の関係

単精度／倍精度の比：少ないほど良い，Intel Xeon Broadwell-EP　1ノード



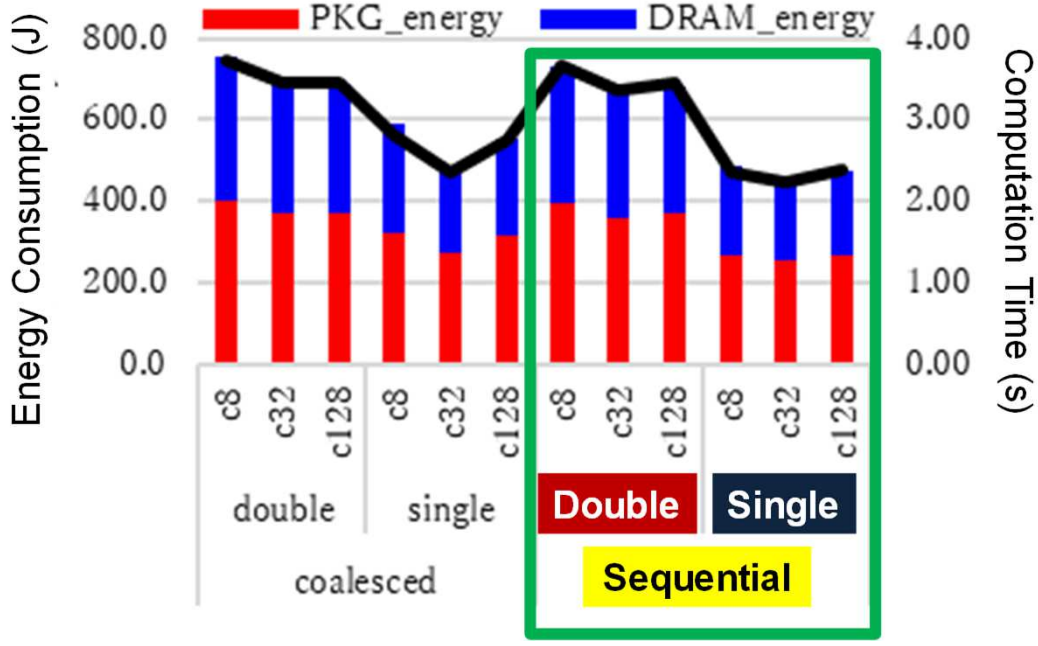
$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

Reedbush：
Intel Xeon Broadwell-EP
Single Node:
18 cores x 2 soc's



3D不均質ポアソン方程式　ICCG法：消費エネルギ測定（J）
$\lambda_1 = \lambda_2$：Reedbush：Intel Xeon Broadwell-EP

- 128^3 DOF
- Coalesced/Sequential
- 単精度・倍精度
- 色数：8, 32, 128
- **計算時間に比例した消費エネルギ（J）**
- **Watt値は単精度になると増加する場合もあり**
  - **計算密度増加**



[Sakamoto et al. 2018]

本研究の沿革(3/3):2019年度までに得られた知見

- 従来，倍精度演算が適用されていた科学技術計算に単精度・半精度・混合演算を適用し，反復改良法(Iterative Refinement)併用により，同等の精度の計算結果がより短時間で得られる場合がある
  - 一般に，計算時間短縮に比例して消費エネルギー(Joule)は減少する
- 悪条件問題では，低精度演算では正解が得られない場合がある。
  - 特に半精度演算は変数の範囲が限定されるため注意が必要であり，精度の要求されない反復法前処理等に適用するべきである。
- 従来のM疎行列向け精度保証手法(T. Ogita他, 2001)は，相対誤差上限の見積もりが厳しめであったため，より現実的な手法を開発し，悪条件問題への有効性が示された。
- 演算精度の影響は，問題規模・疎行列格納手法，アーキテクチャにより多様
- 局所的に演算精度を変更する手法の開発に着手し，有効性が示された。

2019年度の重点実施項目

- 反復改良法による低精度演算安定化（③）
- **大規模分散並列問題で局所的・動的に演算精度を変動させる手法の検討（③，④）**
- 単精度より低い精度の演算の有効性の検討（⑤）
- 疎行列・H行列を係数行列とする実問題向けの精度保証手法の開発（⑦）
- 様々な演算における消費電力の系統的測定（⑧）
- 最適精度自動選択へ向けての検討（⑦，⑧，⑨）

機械学習と混合精度・反復改良法を利用した連立一次方程式解法（奥田，森田，細川）：背景・目的

- 背景
  - 混合精度演算については既に多くの研究があるが，低精度利用時の精度保証の研究は実施されているものの，実問題での大規模疎行列への応用例はほとんどない
  - 混合精度を用いたIterative Refinementの手法も近年注目されているが，Iterative Refinementは必ずしもすべての行列に有効ではなく，Iterative Refinement内にある反復解法の収束判定値も適切な値を設定しないと効果が得られない。
  - 機械学習への期待から，疎行列を対象として最適な数値計算ライブラリ選択に関する研究が行われているが，Iterative Refinementへの応用例はない。
- 目的
  - 解の精度を改善するための反復改良法(Iterative Refinement)に着目し，有限要素解析で生成される大規模疎行列連立一次方程式の求解のための，高精度な混合精度演算手法を開発する

機械学習と混合精度・反復改良法を利用した連立一次方程式解法(奥田，森田，細川)：予備評価

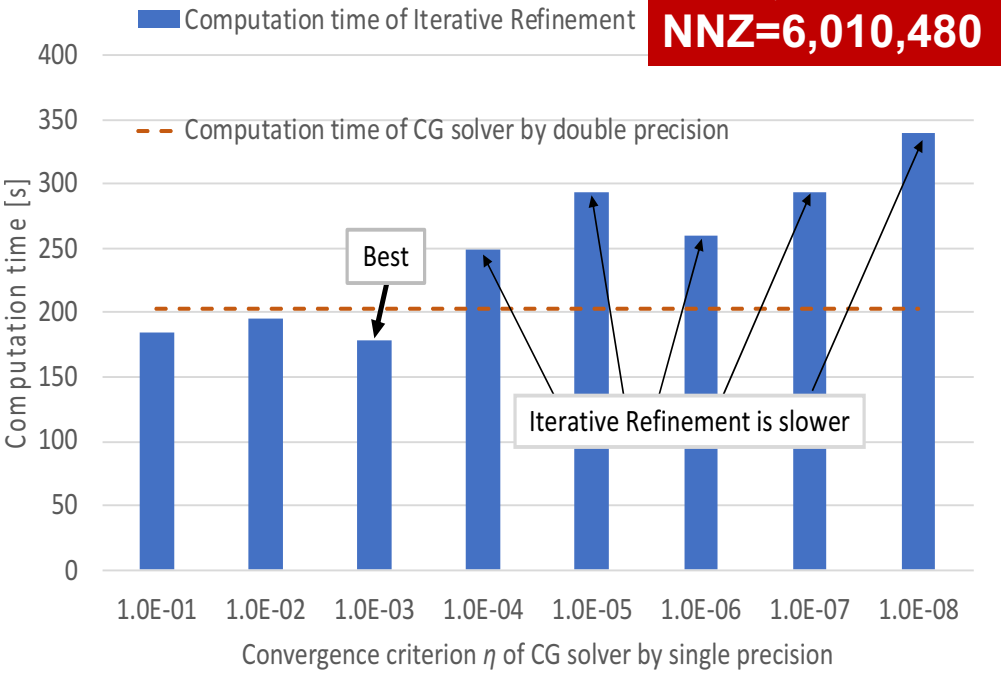
- 概要
  - Iterative Refinementのアルゴリズムではインナーループで単精度共役勾配法を実行するため，本研究の準備フェーズとして，単精度演算，倍精度演算それぞれで共役勾配法の計算を実行し，収束判定条件を変化させて反復回数・計算時間を計測した。
    - SuiteSparse Matrix Collectionより正定値対象の疎行列を選び，右辺ベクトルは厳密解がすべて1になるように定めた。
    - プログラム中で疎行列は全てCRS形式に変換して処理している。
    - 単精度演算を用いた共役勾配法の計算では，係数行列・右辺ベクトル共に単精度で計算している。
- 結果の特徴：3つのグループ
  - ① 反復回数は単精度も倍精度もほぼ等しく，計算時間が単精度の方が常に短い
  - ② 単精度の方が反復回数は多いが，収束判定値が緩いと単精度の方が計算時間が短い
  - ③ 倍精度が単精度よりも反復回数も少なく計算時間も短い

機械学習と混合精度・反復改良法を利用した連立一次方程式解法（奥田，森田，細川）：Iterative Refinement

Algorithm 2 Iterative Refinement

A を単精度行列に変換
$\boldsymbol{c}_0 = \boldsymbol{b}$ ($\boldsymbol{c}_i$ は単精度ベクトル)
for $i = 0 \ldots$ **do**
　単精度 CG ソルバ　▷ 収束判定値 η を設定し，
　　$A\boldsymbol{z}_i = \boldsymbol{c}_i$ の単精度の解 $\boldsymbol{z}_i$ を求める．
　$\boldsymbol{x}_0 = \boldsymbol{z}_0$, $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{z}_{i+1}$
　　▷ 倍精度 $\boldsymbol{x}$ のアップデート
　$\boldsymbol{r}_i = \boldsymbol{b}_i - A\boldsymbol{x}_i$
　if $\|\boldsymbol{r}_{i+1}\| < \epsilon$ **then**
　　Break　▷ 倍精度の解 $\boldsymbol{x}$ を得る．
　end if
　$\boldsymbol{c}_{i+1} = \boldsymbol{r}_{i+1}/\alpha_i$　▷ $\|\boldsymbol{r}\|$ を 1 にスケーリング
end for

ηの効果
N=83,334
NNZ=6,010,480

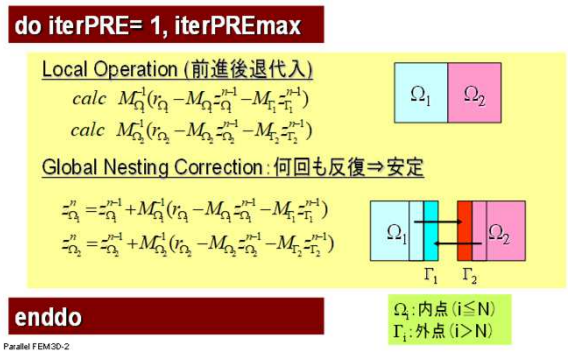


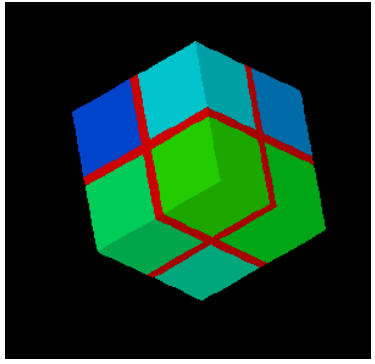
機械学習と混合精度・反復改良法を利用した連立一次方程式解法（奥田，森田，細川）：結論・展望

- Iterative Refinementは適切なインナーループの単精度CG法の収束判定値ηを設定しないと倍精度CG法に比べて良いパフォーマンスが得られないことがわかった。
- 今後は，機械学習を用いて行列の情報からηの値を設定することを考える予定である。

局所的な変動精度適用（中島，河合）

- 国際会議WCCM 2018で，問題の条件に応じて局所的に演算精度を変動させる可能性について指摘があり，2019年度は，大規模分散並列問題で局所的・動的に演算精度を変動，動的に負荷分散を適用する手法の開発を検討
  - 領域分割に基づく並列有限要素法において，領域毎に異なる精度を適用した場合の検証を，並列有限要素法による三次元構造解析における加法シュワルツ法（Additive Schwartz Domain Decomposition（ASDD））に基づくICCG法ソルバーに対して実施





  - 規則形状を8分割した場合，6領域については倍精度，2領域については混合精度（前処理・加法シュワルツ法部分のみ単精度）の疎行列ソルバーを適用し，均質問題において，全8領域倍精度の場合と同じ反復回数で同じ解を得た。
  - 2020年度以降は不均質問題，動的負荷分散に継続して取り組む予定。

2019年度の重点実施項目

- 反復改良法による低精度演算安定化（③）
- 大規模分散並列問題で局所的・動的に演算精度を変動させる手法の検討（③，④）
- **単精度より低い精度の演算の有効性の検討（⑤）**
- 疎行列・H行列を係数行列とする実問題向けの精度保証手法の開発（⑦）
- 様々な演算における消費電力の系統的測定（⑧）
- 最適精度自動選択へ向けての検討（⑦，⑧，⑨）

変動精度演算用FP21のCUDA/OpenACC実装（市村，藤田）(1/2)

- 背景・概要
  - 比較的条件の良い問題では，倍精度演算(FP64)のかわりに単精度(FP32)を使うことによって計算時間を短縮できる場合があるが，機械学習などで使われる半精度(FP16)は有効桁数が3桁程度のため，実問題に使用することは困難である。
  - 本研究ではFP16と FP32の中間的な特性を持つデータ型であるFP21を定義し，有限要素解析における前処理において活用した。
- FP21
  - FP21は符号1ビット・指数部8ビット・仮数部12ビットの合計21ビットからなり，FP32(符号1ビット・指数部8ビット・仮数部23ビットの合計32ビット)と指数部のビット数が同一。
  - FP21はFP32と同じレンジを持つもののデータサイズが1/1.5となるため，メモリバンド幅ネックのカーネルにおいては実行時間が1/1.5になると期待される。
  - 現在の主要な計算機においてはFP21の演算器はサポートされていないため，変数のメモリへの格納時のみにFP21を使い，計算時にはFP21をFP32に変換して演算する。

変動精度演算用FP21のCUDA/OpenACC実装（市村，藤田）(2/2)

- 成果
  - 上記をCUDAにて実装しV100 GPUで計測した結果，想定通りの性能が得られ，FP21の有効性が示された。
  - 本研究ではさらに上記をOpenACCで実装し，ソースコードを公開することでより多くのアプリケーションでFP21を活用できるようにした。
  - FP21AXPY OpenACC source code, https://github.com/y-mag-chi/fp21axpy
  - 性能計測の結果，実際の地震アプリケーションに組み込んだ際にOpenACC版においてCUDA版と同等の性能が得られることを確認した〔Yamaguchi, Fujita, Ichimura et al. WACCPD 2019@SC19〕

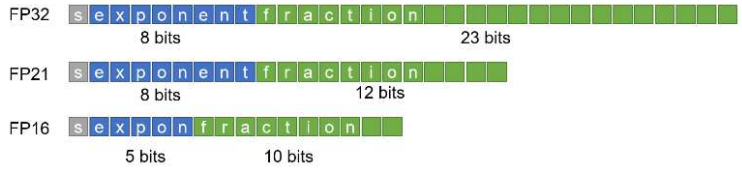


Table 2. Elapsed time for the entire solver measured on ABCI. The total elapsed time includes the output of the analysis results. Performance of the preconditioning solvers is summarized in order of their appearance in *CG* solver. The numbers of iteration in each solver are also shown in parentheses.

	CPU-based	baseline OpenACC	baseline CUDA	proposed	SC18GBF
Precision in $PreCG_c$, $PreCG_{cp}$, and $PreCG$	FP32	FP32	FP32	FP32/21	FP32/21/16
$PreCG_c$	161.4 s (6199)	14.89 s (6300)	14.21 s (6210)	9.79 s (4751)	7.47 s (4308)
$PreCG_{cp}$	69.9 s (28830)	15.94 s (28272)	12.20 s (28491)	13.22 s (28861)	8.98 s (26887)
$PreCG$	372.0 s (2674)	22.90 s (2729)	22.30 s (2735)	18.27 s (2575)	16.98 s (2797)
CG	83.9 s (91)	5.77 s (89)	4.57 s (89)	5.89 s (122)	8.32 s (129)
Other	94.8 s	7.21 s	7.73 s	8.66 s	5.99 s
Total	**781.8 s**	**66.71 s**	**61.02 s**	**55.84 s**	**47.75 s**
Speeding up ratio	1	11.7	12.8	14.0	16.4

2019年度の重点実施項目

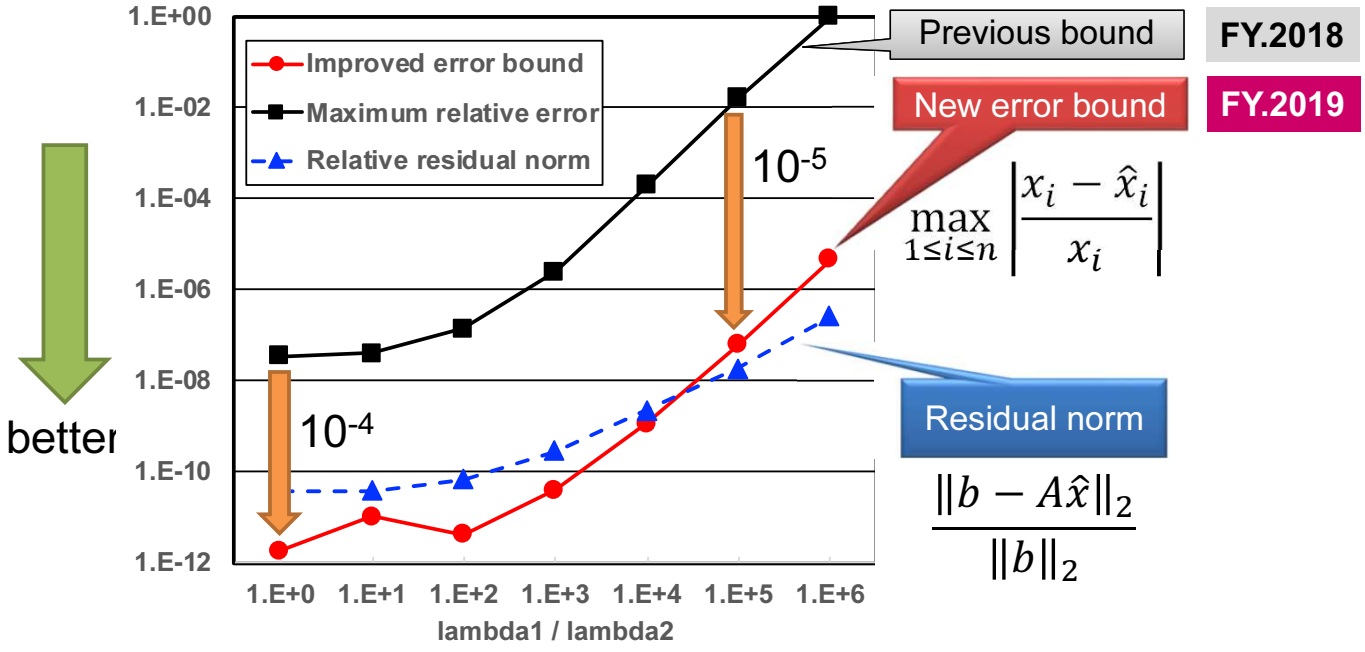
- 反復改良法による低精度演算安定化（③）
- 大規模分散並列問題で局所的・動的に演算精度を変動させる手法の検討（③，④）
- 単精度より低い精度の演算の有効性の検討（⑤）
- **疎行列・H行列を係数行列とする実問題向けの精度保証手法の開発（⑦）**
- 様々な演算における消費電力の系統的測定（⑧）
- 最適精度自動選択へ向けての検討（⑦，⑧，⑨）

低・変動精度演算の活用，精度保証（荻田，尾崎，中島）

- 実問題（悪条件問題）では精度保証が重要
  - 近年のApproximate Computingに関する研究で，精度保証にまで立ち入った例はない：計算できればOK
- これまでの精度保証研究は主として密行列対象
  - 疎行列・H行列の例は少ない：条件の良い問題が中心
  - 一般に精度保証には求解の数倍の時間を要する
- 本研究で目指す精度保証手法（対象：疎行列，H行列）
  - 高速・正確・悪条件に対応
  - 自動チューニングによる最適精度選択
- 荻田等による既存手法〔荻田・後・大石 2001〕
  - M行列向け（条件の良い行列）
  - 現実的な時間で精度保証が可能なことは確認されたが，悪条件問題向けにより精密な手法が必要（2018年度研究）⇒より精密な手法の開発（2019年度研究）（次頁）

Result (2-1): NX=NY=NZ=128

Vary λ_1/λ_2~cond between 1 and 10^6



Computed error bounds are significantly improved!

[Ogita & Nakajima, 2019]

Result (2-2): Computing time

- Elapsed

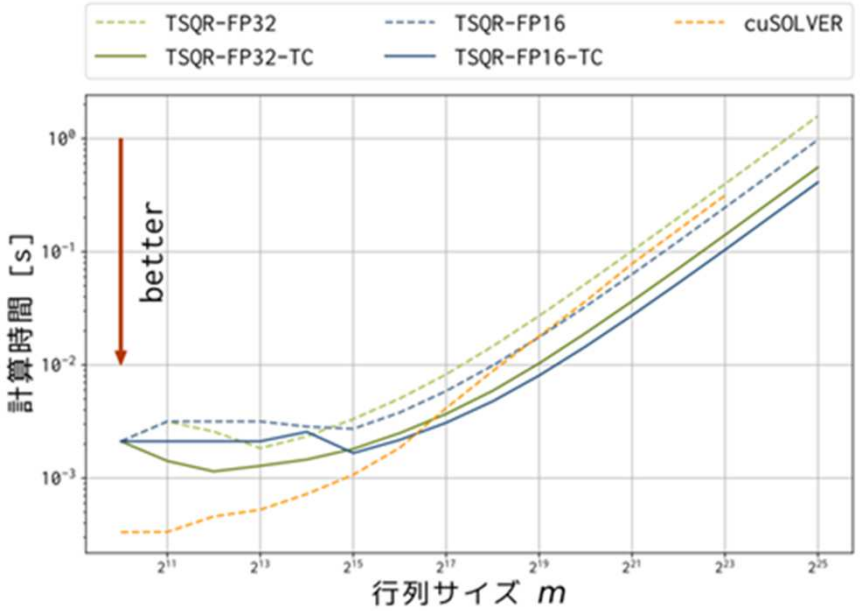
NX=NY=NZ=128 (n = 2,097,152)	Approximation (Solve $Ax = b$)	Verification (Solve $Ay = e$ and $Az = \hat{r}$)
$\lambda_1/\lambda_2 = 1$	3.75 (415)	4.47 (493 = 176 + 317)
$\lambda_1/\lambda_2 = 10^6$	6.83 (686)	7.85 (777 = 328 + 449)

NX=NY=NZ=256 (n = 16,777,216)	Approximation (Solve $Ax = b$)	Verification (Solve $Ay = e$ and $Az = \hat{r}$)
$\lambda_1/\lambda_2 = 1$	76.64 (903)	80.89 (964 = 387 + 577)
$\lambda_1/\lambda_2 = 10^6$	110.15 (1377)	120.07 (1504 = 639 + 865)

[Ogita & Nakajima, 2019]

H行列向け手法の高速圧縮手法の開発(横田)

- H行列計算の中で最も工夫を要するのは，密行列を低ランク行列に圧縮する部分である。
  - 圧縮には一般的にrandomized SVDが用いられ，その内部カーネルは複数のQR分解から構成される。



- 2018年度は多くの小さなQR分解をGPU上で並列に行うbatched QR分解を開発したが，2019年度はこれをTSQR(Tall & Skinny)に拡張し，任意の大きさの行列のrandomized SVDを行うためのQR分解の枠組みを開発した。
- NVIDIA V100上でFP32を用いた場合の計算時間はcuSOLVER による QR分解と比較して最大2.17倍高速となった

2019年度の重点実施項目

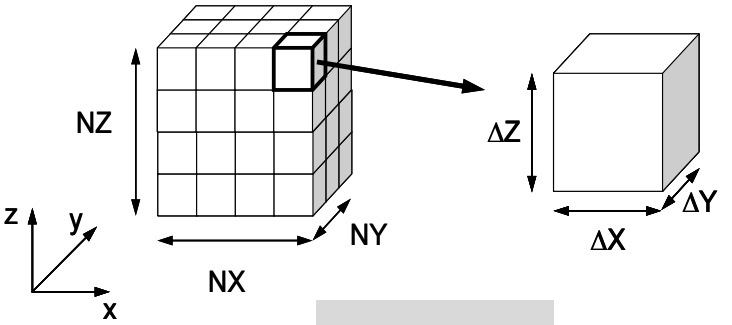
- 反復改良法による低精度演算安定化（③）
- 大規模分散並列問題で局所的・動的に演算精度を変動させる手法の検討（③，④）
- 単精度より低い精度の演算の有効性の検討（⑤）
- 疎行列・H行列を係数行列とする実問題向けの精度保証手法の開発（⑦）
- **様々な演算における消費電力の系統的測定（⑧）**
- 最適精度自動選択へ向けての検討（⑦，⑧，⑨）

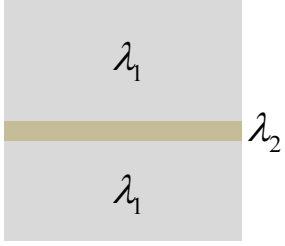
中島，坂本，星野，有間，塙，近藤，「低精度演算とアプリケーション性能」第174回HPC研究会（情報処理学会）（2020.5）

- 近年，科学技術計算において，低精度演算を積極的に活用することにより，計算時間を短縮する試みが活発に行われている。
- また，低精度演算による計算の精度を保証するための実用的手法についても研究が進められている。
- 本研究では，アプリケーションの実装方法，問題規模と低精度演算による性能改善の関係に注目し，様々なハードウェア環境下での検討を実施した。

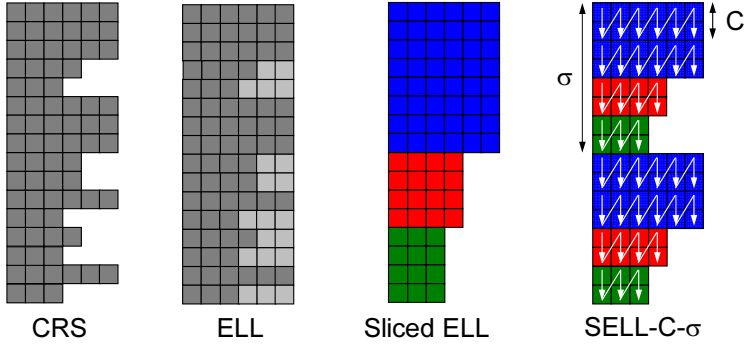
Poisson3D-OMP

- 有限体積法(FVM)，ポアソン方程式
  - 7点差分格子，内部データ構造は非構造格子型
    - $\lambda_1/\lambda_2 = 1.00$
  - 対称正定(Symmetric Positive Definite，SPD)
  - 前処理付きCG法：ICCG法
    - 〔大島他 SWoPP 2014 〕，〔中島 HPC-139・147・157〕，〔星野 HPC-158〕
- Fortran 90 + OpenMP
- OpenMPによるスレッド並列化：Reordering必要
  - MC/RCM/CM-RCM + Coalesced/Sequential
- 疎行列格納法
  - CRS, ELL (Ellpack-Itpack), SELL-C-σ
- 倍精度・単精度
- ループ構造
  - **Row-Wise**, Column-Wise



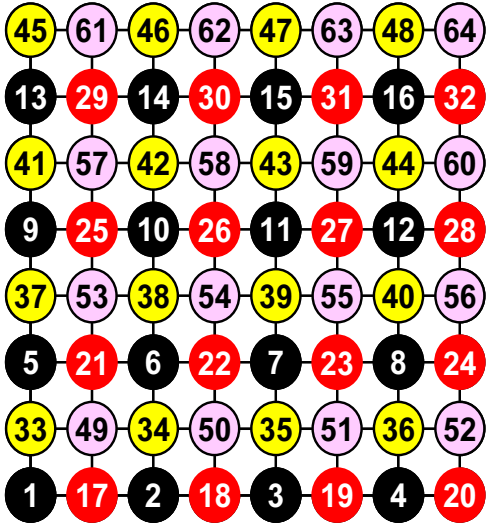


$$\nabla \cdot (\lambda \nabla \phi) + f = 0$$

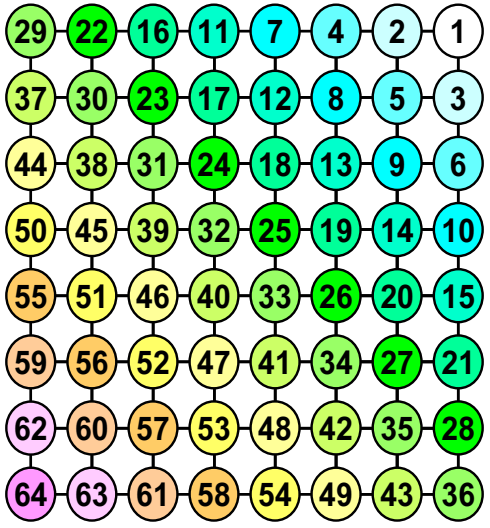


並列IC/ILUのためのReordering

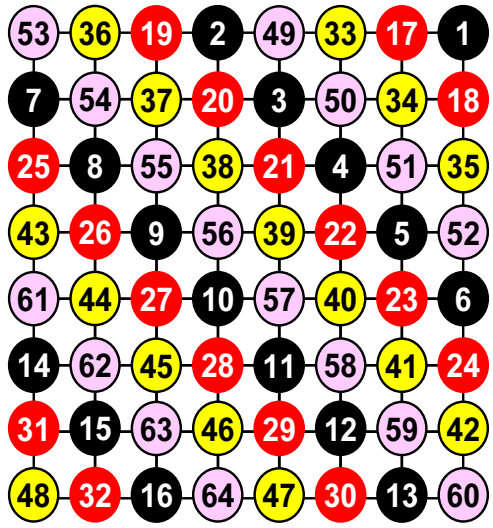
同じ色に属する要素は独立，お互いに依存性持たない⇒並列処理可能



MC (Color#=4)
Multicoloring
並列性能◎，収束△



RCM
Reverse Cuthill-Mckee
並列性能△，収束◎

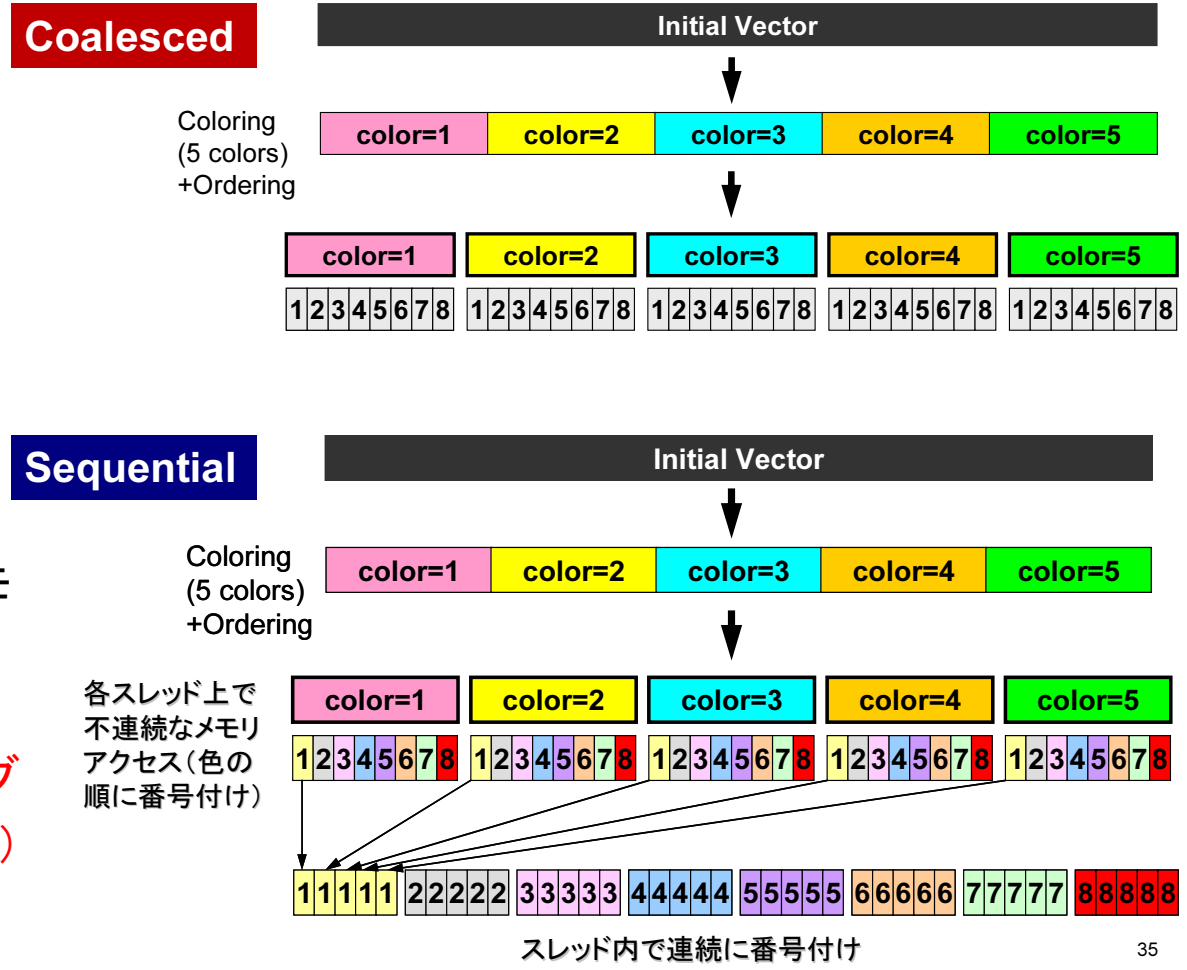


CM-RCM (Color#=4)
Cyclic MC + RCM
並列性能◎，収束◎

Coalesced, Sequential オーダリング

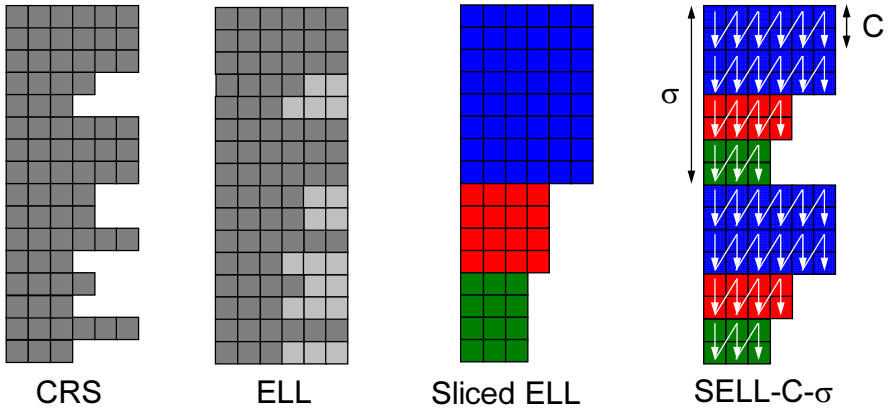
色順に並べ替えた後

- Coalesced
  - 色内連続番号
  - 各スレッドは不連続なメモリアクセス
- Sequential
  - データ順を再オーダリング
  - 計算単位（スレッドやコア）内で連続アクセス



問題設定

- 係数行列格納方法
  - CRS, ELL
  - SELL-C-σ
  - Optimized SELL-C-σ
    - [星野 HPC-158] **Barrier Free**, OFP向け
    - ベクトル長(C)＝ 8, 16
- Fortran+OpenMP/OpenACC
  - Intel Xeon Phi (Oakforest-PACS, OFP)
  - Intel Xeon CXL (Oakbridge-CX, OBCX)
  - Intel Xeon BDW (Reedbush-U, RBU)
  - NVIDIA P100, V100
- 倍精度・単精度
- 計算時間，消費電力(W)，消費エネルギー(J)測定



- 問題サイズ
  - Large: 256×256×256
  - Medium: 128×128×128
  - Small: 64× 64× 64
  - Tiny: 32× 32× 32
- 全ケースにCM-RCM(10) 適用

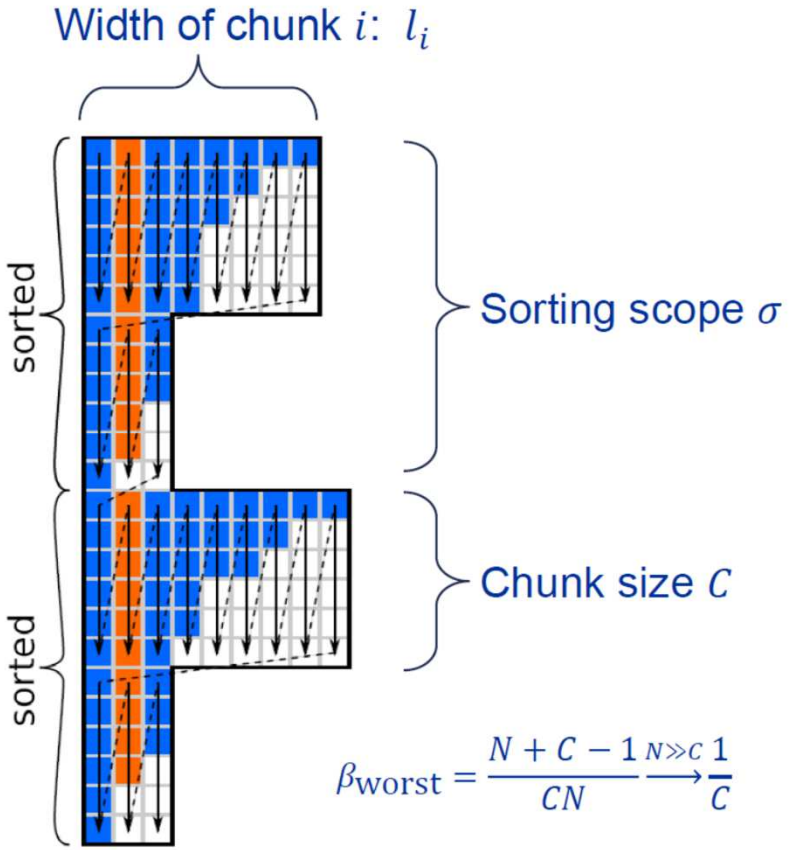
Constructing SELL-C-σ

[Kreutzer, Hager, Wellein 2014]

1. Pick chunk size C (guided by SIMD/T widths)
2. Pick sorting scope σ
3. Sort rows by length within each sorting scope
4. Pad chunks with zeros to make them rectangular
5. Store matrix data in "chunk column major order"

"Chunk occupancy": fraction of "useful" matrix entries

$$\beta = \frac{N_{nz}}{\sum_{i=0}^{N_c} C \cdot l_i}$$



SELL-6-12
β=0.66

前処理付き共役勾配法（PCG）（1/2）

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
    else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- 各ステートメントはループ
  - 行列ベクトル積
  - 内積
  - DAXPY
  - 前処理（前進後退代入）
- OpenMPによる並列化
  - 各ステートメントに!$omp parallel適用
  - 内積ではreduction
  - 各ステートメント毎にfork-join

（PCG）：OPT，Barrier Free（2/2）

```
Compute r(0)= b-[A]x(0)
!$omp parallel (…)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
    else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
End
!$omp end parallel
```

- 〔星野 HPC-158〕による最適化
- 反復法のループの前後に「!$omp parallel，!$omp end parallel」
- 「fork-join」が一回しか起こらない
- 「!$omp do」，「reduction」も除去

SELL-8-1（前進代入）：Row-Wise：OFPで最適

予稿のものはColumn-Wise

SCS: SELL-C-σ

```
!$omp parallel (…)
    do ic= 2, NCOLORtot-1
!$omp do
      do ip= 1, PEsmpTOT
        iq1= SMPindex((ip-1)*NCOLORtot + ic-1) + 1
        iq2= SMPindex((ip-1)*NCOLORtot + ic)
        ip0= (ip-1)*NCOLORtot + ic
        iq0= (iq1-1)*8
      do ib  = iq1, iq2
        ib0= (ib-iq1)*8
!$omp simd
      do is  = 1, 8
        i= iq0 + ib0 + is
        VAL= W(i,Z)
        do k= 1, 3
          VAL= VAL- AL(ib0+is,k,ip0)*W(itemL(ib0+is,k,ip0),Z)
        enddo
        W(i,Z)= VAL * W(i,DD)
      enddo
      enddo
      enddo
    enddo
!$omp end parallel
```

OPT

```
!$omp parallel (…)
(…)
    do ic= 2, NCOLORtot-1


      iq1= SMPindex((ip-1)*NCOLORtot + ic-1) + 1
      iq2= SMPindex((ip-1)*NCOLORtot + ic)
      ip0= (ip-1)*NCOLORtot + ic
      iq0= (iq1-1)*8
      do ib  = iq1, iq2
        ib0= (ib-iq1)*8
!$omp simd
      do is  = 1, 8
        i= iq0 + ib0 + is
        VAL= W(i,Z)
        do k= 1, 3
          VAL= VAL- AL(ib0+is,k,ip0)*W(itemL(ib0+is,k,ip0),Z)
        enddo
        W(i,Z)= VAL * W(i,DD)
      enddo
      enddo
!$omp barrier
    enddo
(…)
!$omp end parallel
```

SCS: SELL-C-σ

```
!C
!C +-------------+
!C | RHO= {r}{z} |
!C +-------------+
!C===
      RHO= 0.d0
!$omp parallel do private(i) reduction(+:RHO)
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo
!C===

!C
!C +-----------------------------+
!C | {p} = {z} if       ITER=1   |
!C | BETA= RHO / RHO1   otherwise |
!C +-----------------------------+
!C===
      if ( L.eq.1 ) then
!$omp parallel do private(i)
        do i= 1, N
          W(i,P)= W(i,Z)
        enddo
       else
        BETA= RHO / RHO1
!$omp parallel do private(i)
        do i= 1, N
          W(i,P)= W(i,Z) + BETA*W(i,P)
        enddo
      endif
!C===
```

$\rho = rz,\ p = z + \beta p$

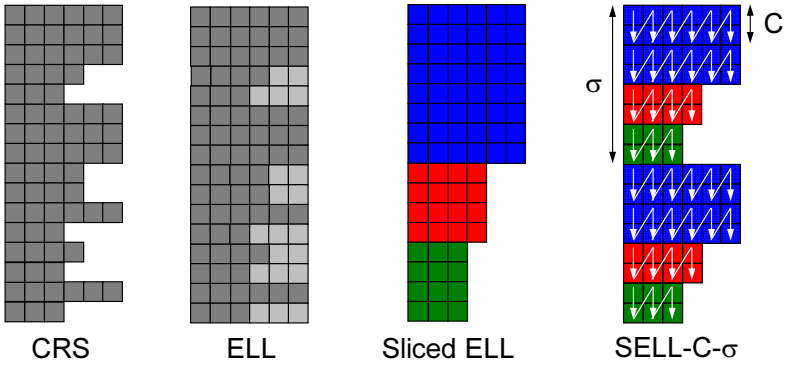
OPT

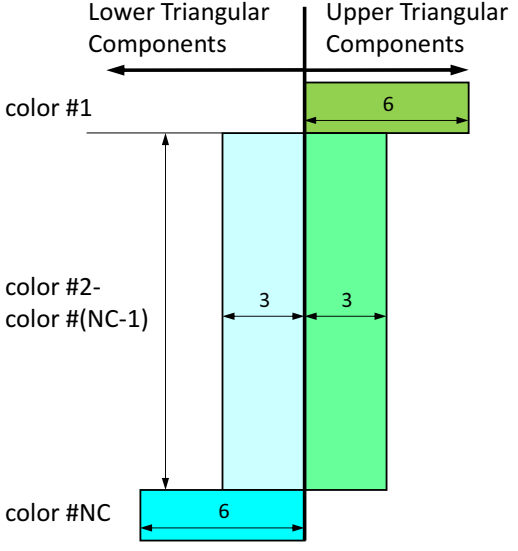
```
!C
!C +-------------+
!C | RHO= {r}{z} |
!C +-------------+
!C===
      W_RHO(ip)= 0.0d0
!$omp simd
      do i= (ip-1)*ls+1, min(ip*ls,N)
        W_RHO(ip)= W_RHO(ip) + W(i,R)*W(i,Z)
      enddo
      RHO= 0.d0
!$omp barrier
!$omp simd
      do i = 1, PEsmpTOT
         RHO= RHO + W_RHO(i)
      end do
!C===

!C
!C +-----------------------------+
!C | {p} = {z} if       ITER=1   |
!C | BETA= RHO / RHO1   otherwise |
!C +-----------------------------+
!C===
      if ( L.eq.1 ) then
        do i= (ip-1)*ls+1, min(ip*ls,N)
          W(i,P)= W(i,Z)
        enddo
!$omp barrier
       else
        BETA= RHO / RHO1
        do i= (ip-1)*ls+1, min(ip*ls,N)
          W(i,P)= W(i,Z) + BETA*W(i,P)
        enddo
!$omp barrier
      endif
!C===
```

Sliced ELL⇒三角分解

- 行列ベクトル積ほど簡単ではない
  - Sliced ELL，SELL-C-σを三角分解に適用した他の事例は恐らくない：Gauss-Seidel有
- Sliced　ELL：余計な計算を回避
  - HPCG：Gauss-Seidel　LUの区別ナシ
- スレッド並列：リオーダリング
  - 上下三角成分の数は変わる
  - 本研究で扱う問題の場合，Upper=3，Lower=3が基本であるが，MCを適用すると，自分の周囲が全部Upperとか Lowerの例が出てくる
  - CM，RCMの場合は元の大小関係が基本的に保たれる
- CM-RCM（本研究の事例（7点ステンシル）では）
  - 1色目：全部上三角成分：最大6
  - 2色目～（NCOLORtot-1）色目：上ー最大3，下ー最大3
  - （NCOLORtot）色目：全部下三角成分：最大6





ICCG法計算効率：大きいほど速い

ICCG法計算時間に基づく（反復回数増加考慮）

CRS・倍精度で無次元化（1.00）

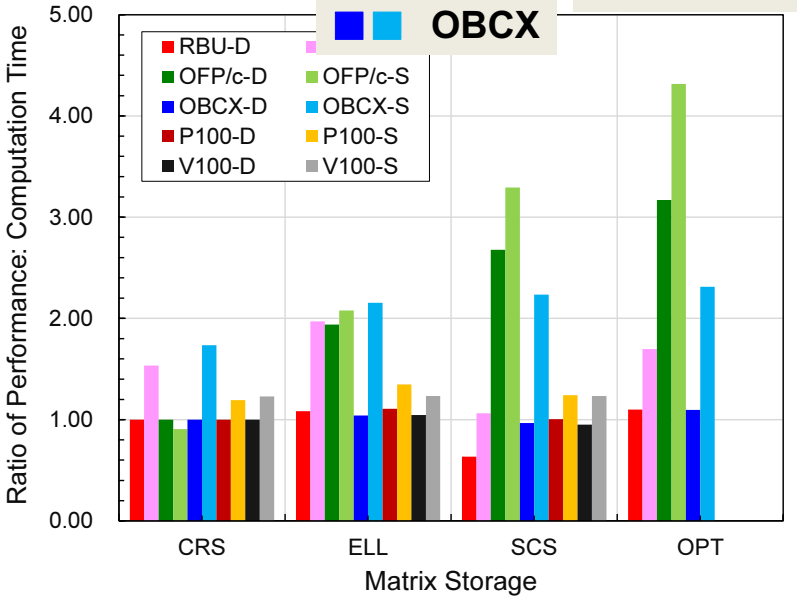
計算密度は上がる

- CRS
- ELL
- SCS: SELL-C-σ
- OPT: Optimized SCS

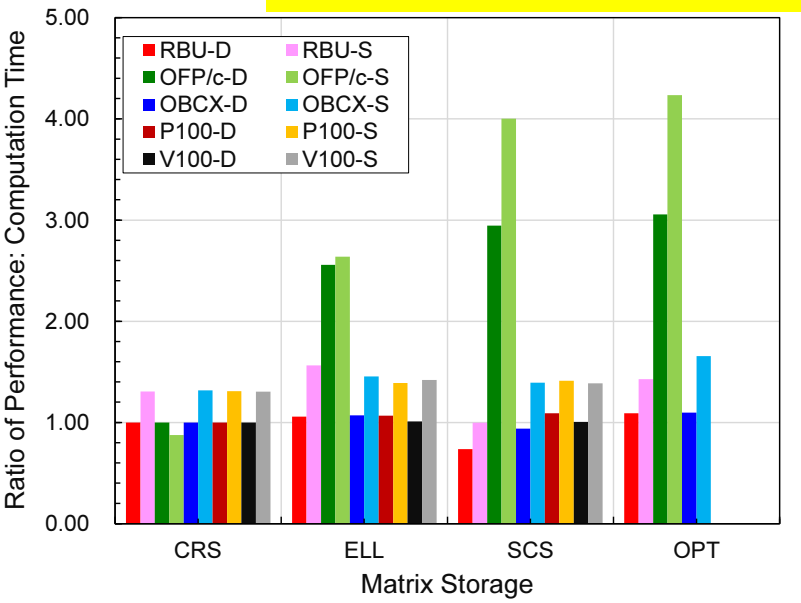
薄い色大：単精度の効果大

Medium

RBU　OFP　OBCX　P100　V100



Large



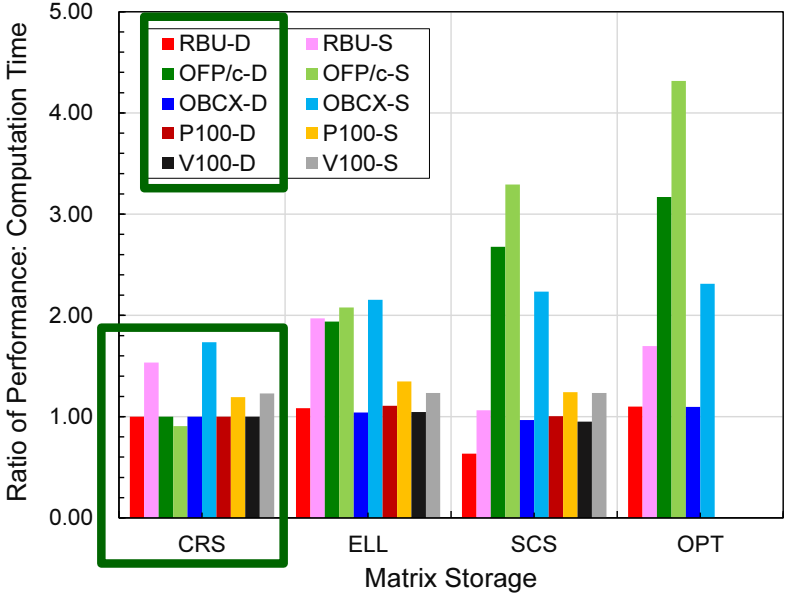
ICCG法計算効率：大きいほど速い

ICCG法計算時間に基づく（反復回数増加考慮）

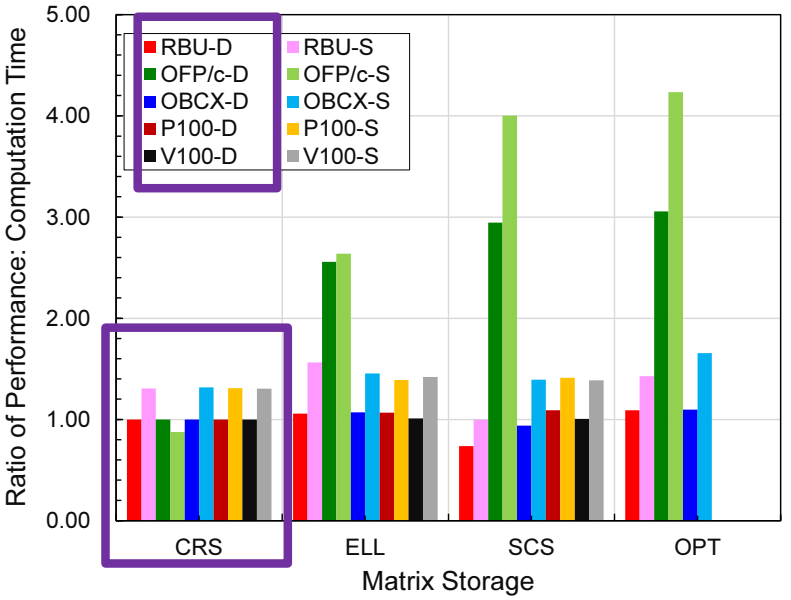
CRS・倍精度で無次元化（1.00）

・反復回数は倍精度⇒
単精度で20%増加

Medium



Large



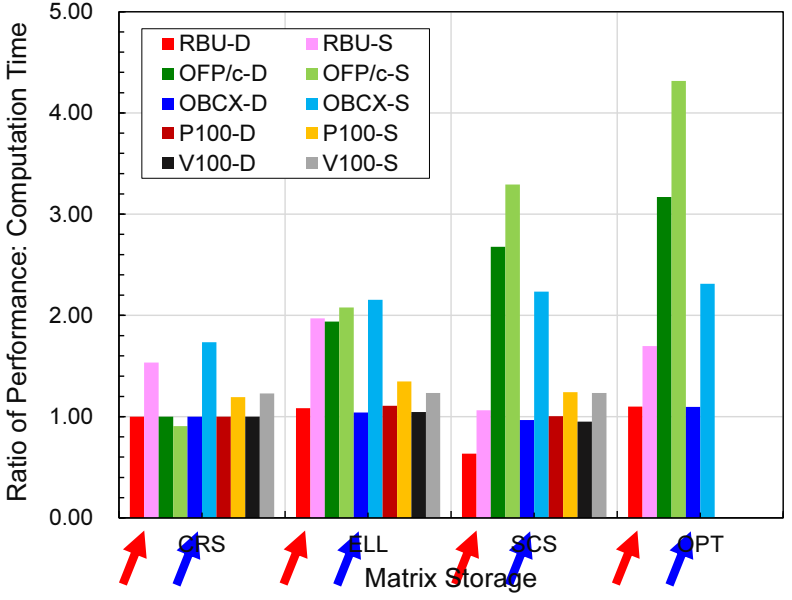
ICCG法計算効率：大きいほど速い

ICCG法計算時間に基づく（反復回数増加考慮）

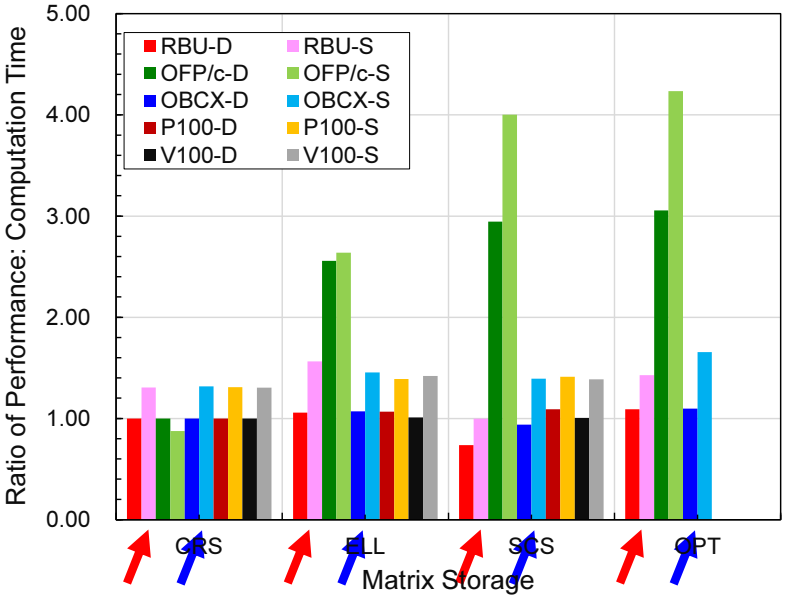
CRS・倍精度で無次元化（1.00）

- RBU，OBCXはCRS⇒ELLで性能向上するが（倍精度・単精度とも），SCSで一回遅くなる
- 倍精度⇒単精度の速度向上は顕著（特にMedium）
- RBU
- OBCX

Medium



Large



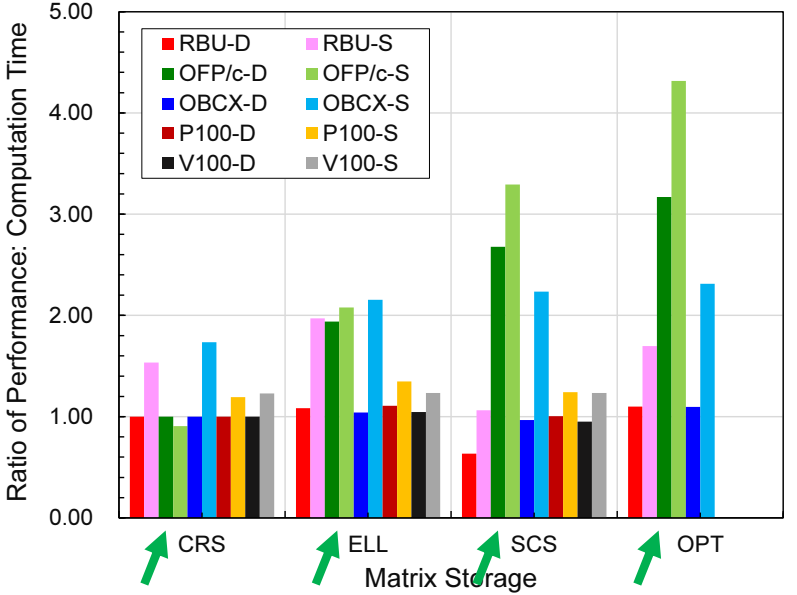
ICCG法計算効率：大きいほど速い

ICCG法計算時間に基づく（反復回数増加考慮）

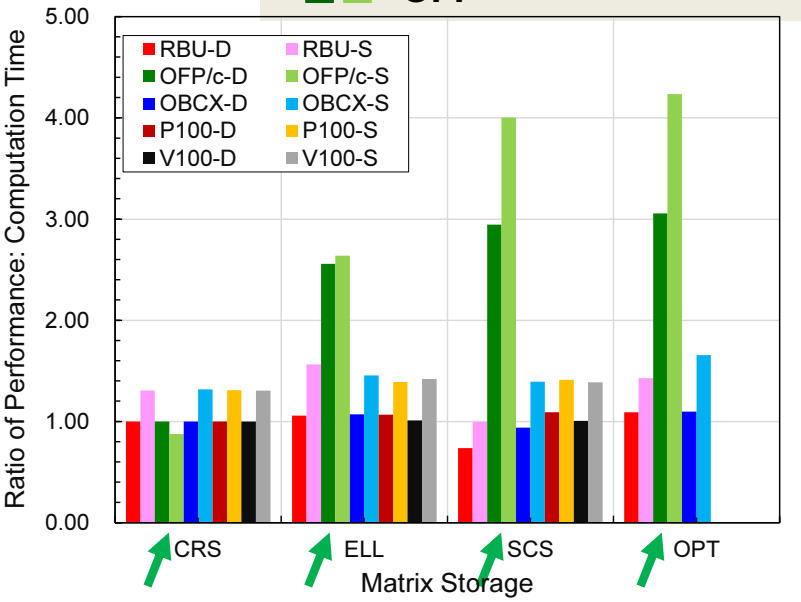
CRS・倍精度で無次元化（1.00）

- OFPではELL⇒SCS⇒OPTと順調に性能向上
- SCS，OPTは元々OFP向けに最適化
- 倍精度⇒単精度の速度向上は少ない（ベクトル化不十分，とくにCRS）
- ■■ OFP

Medium



Large



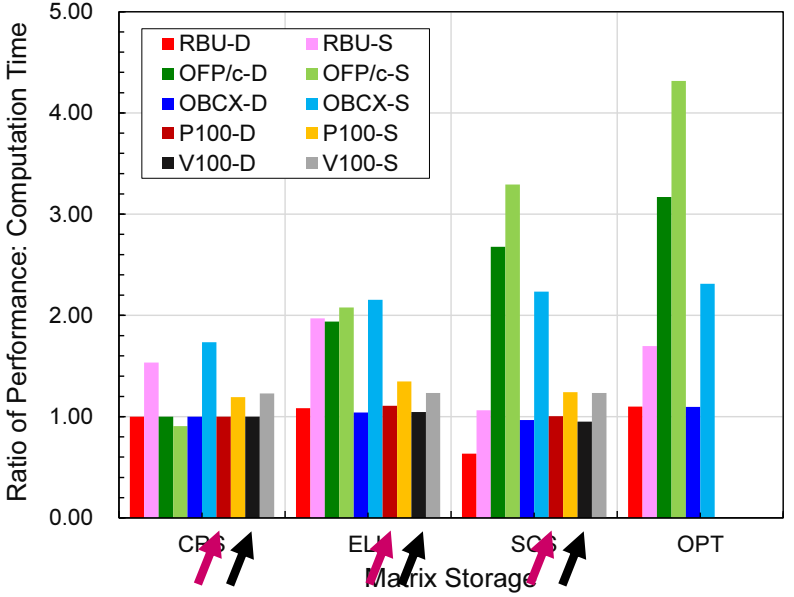
ICCG法計算効率：大きいほど速い

ICCG法計算時間に基づく（反復回数増加考慮）

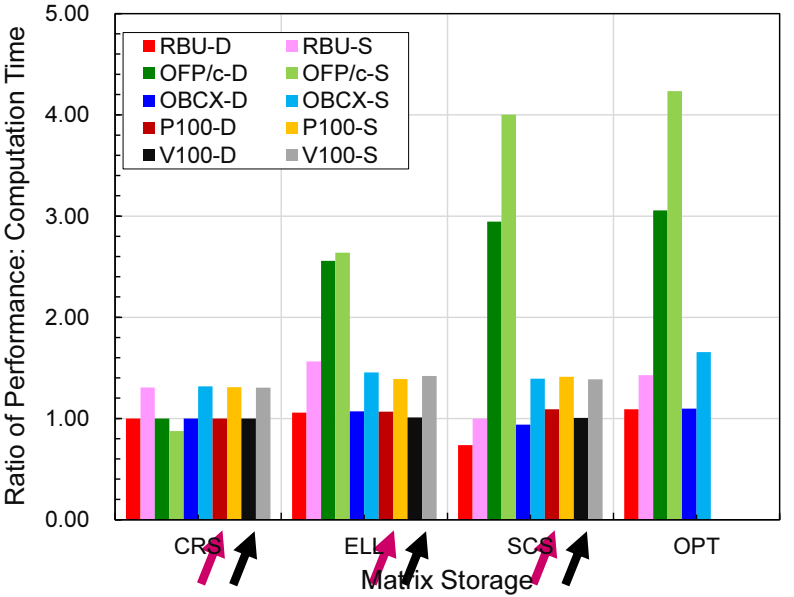
CRS・倍精度で無次元化（1.00）

- P100，V100では行列格納方法による差は少ない，ELLが比較的良い
- 倍精度⇒単精度の速度向上は20%程度
- P100
- V100

Medium



Large

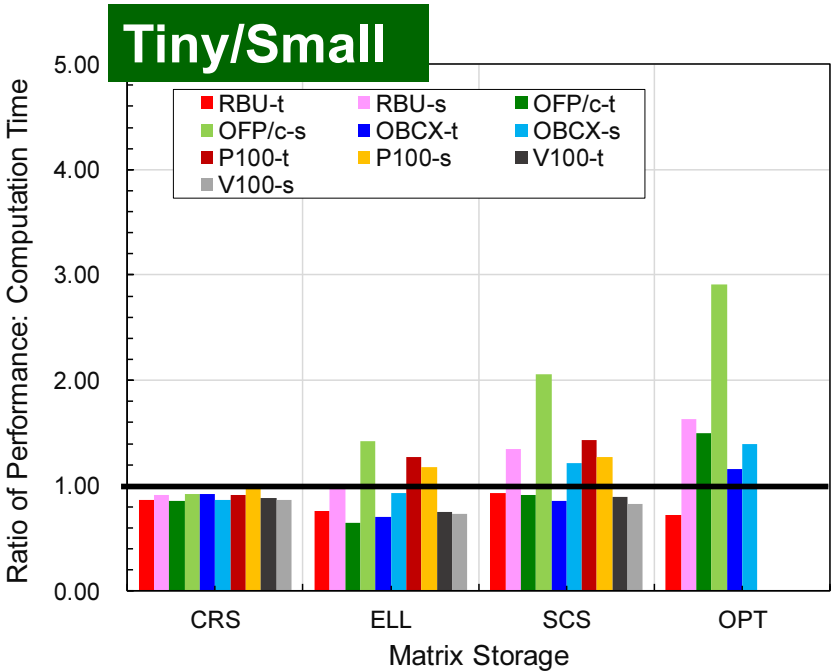


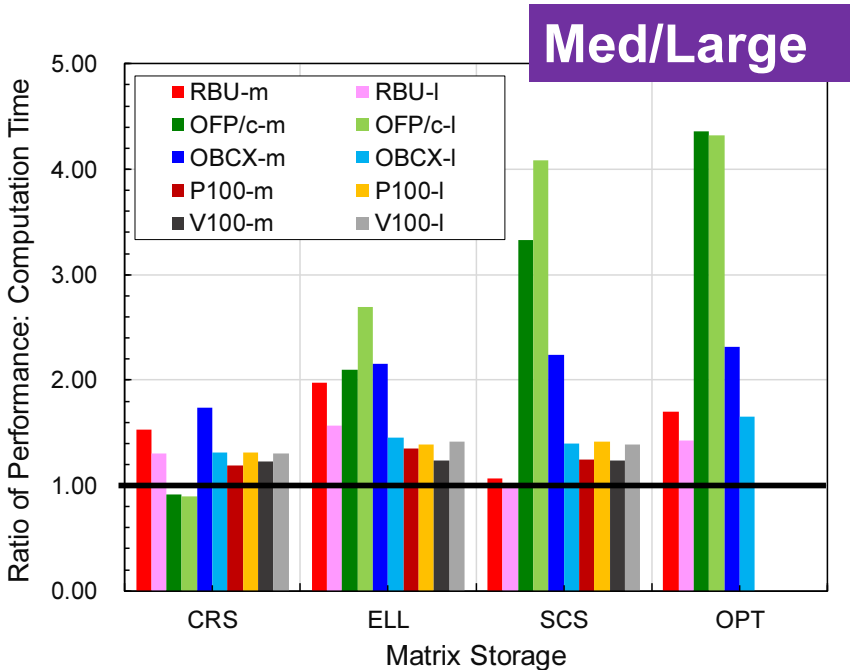
単精度計算効率

ICCG法計算時間（反復回数増加考慮）

CRS･倍精度で無次元化，大きいほど速い

- 反復回数は倍精度⇒単精度で20%増加
- OFP/CRSは単精度の効果無し（むしろ遅くなっている）
- 単精度による高速化の効果は問題サイズが大きい場合に顕著

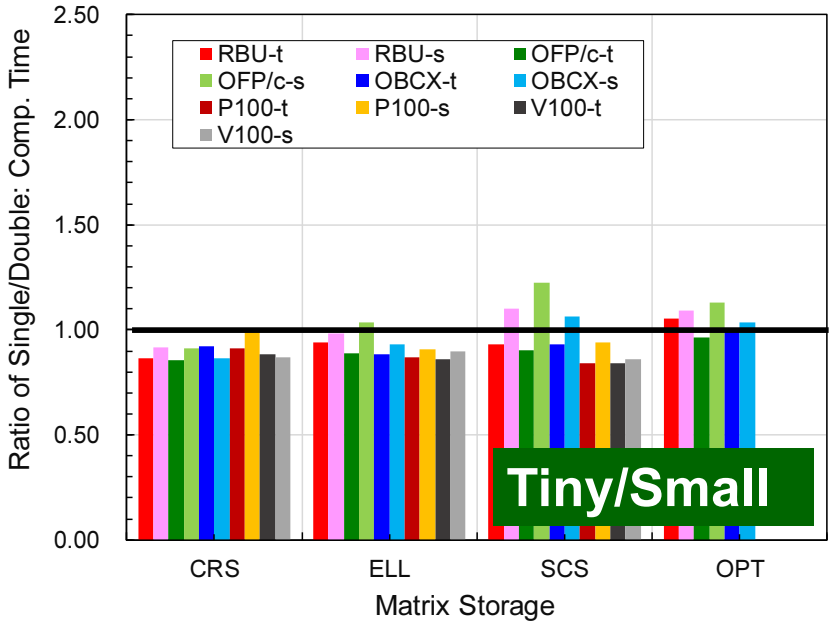


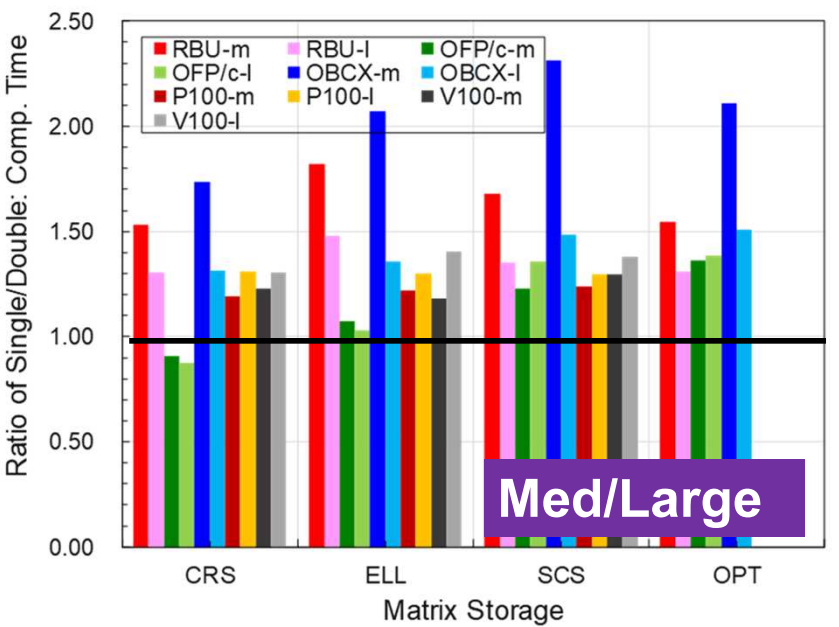


単精度と倍精度：計算時間比

ICCG法計算時間（反復回数増加考慮）
1.0より大きい⇒単精度速い

- 反復回数は倍精度⇒単精度で20%増加
- OFP/CRSは単精度の効果無し（むしろ遅くなっている）
- 単精度による高速化の効果は問題サイズが大きい場合，高演算密度で顕著
- RBU，OBCXのMediumサイズ■■

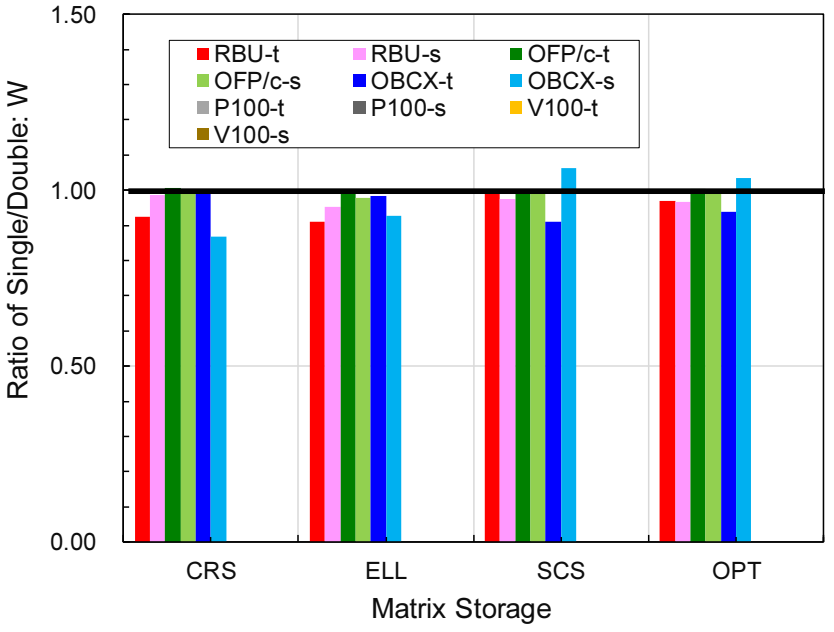


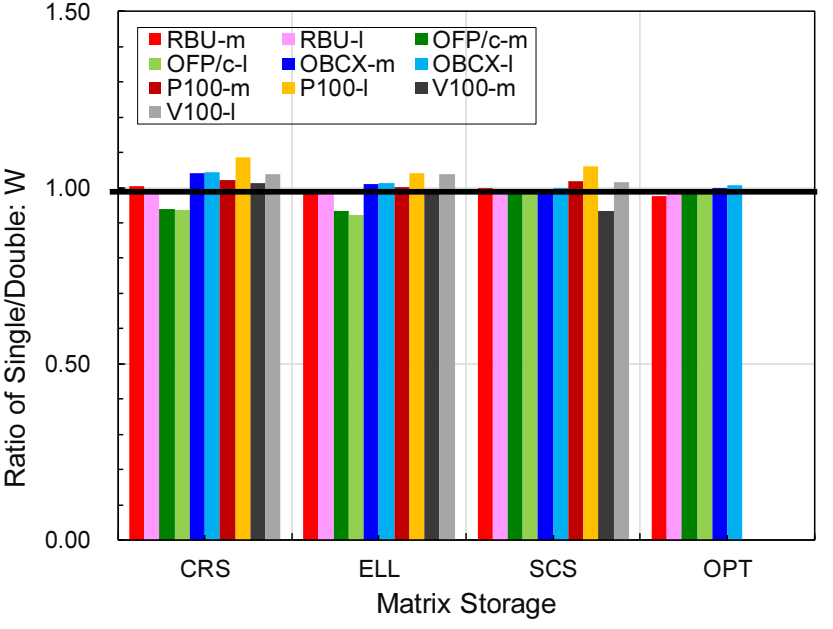


単精度と倍精度：消費電力比 Watt値

>1.0：単精度の消費電力多い（W）

- GPUではTiny/Small測定できず
- ほぼ同じ（~1.00）
- 単精度では計算密度が高まりW値が増えると言われているがそれほどでもない



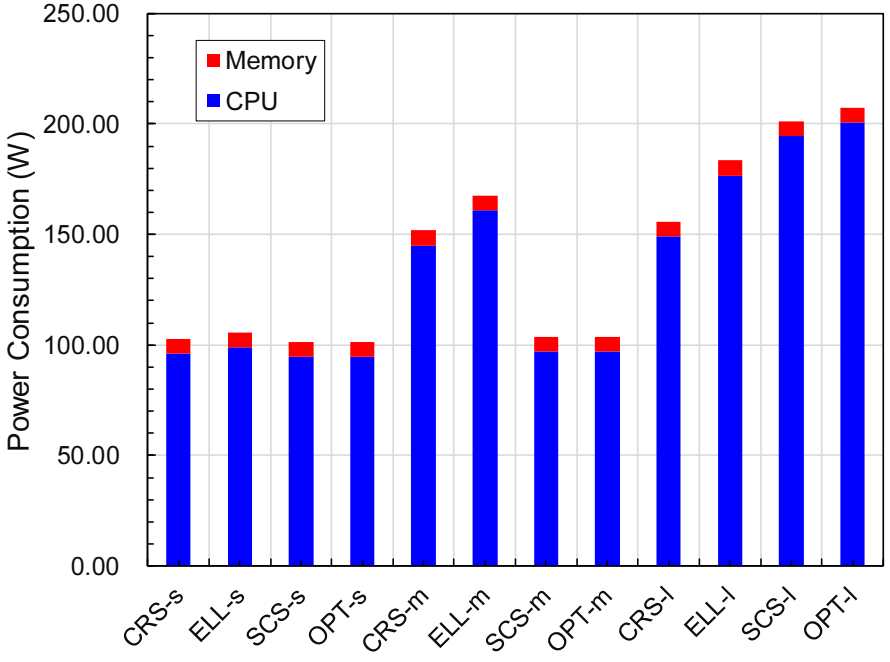


単精度・消費電力(W), Small/Medium/Large

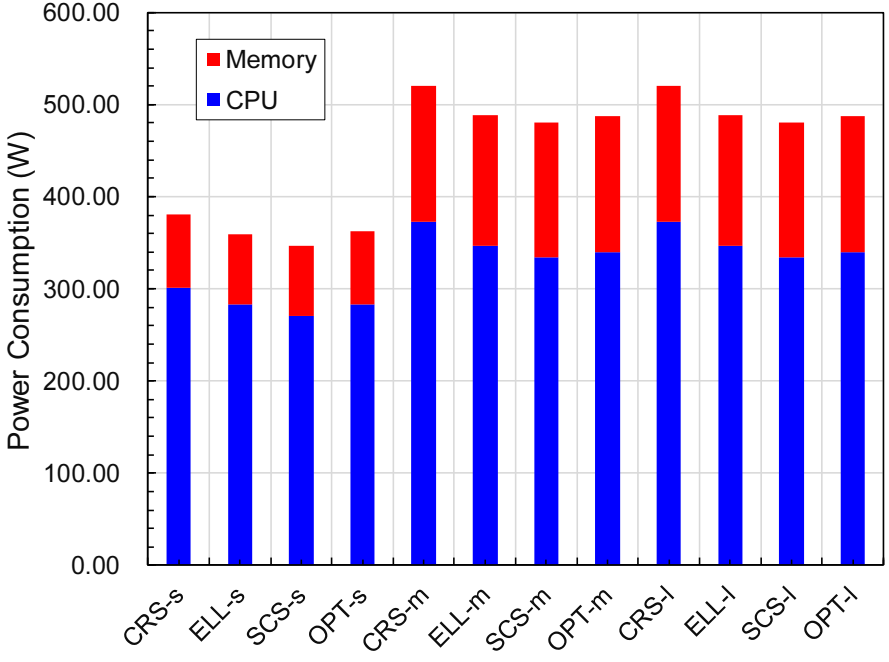
OFP:メモリ(DDR4)の消費電力は極めて少ない

Memory
Package(CPU)
キャッシュはPackage(CPU)

OFP-c (Flat-MCDRAM)



OBCX

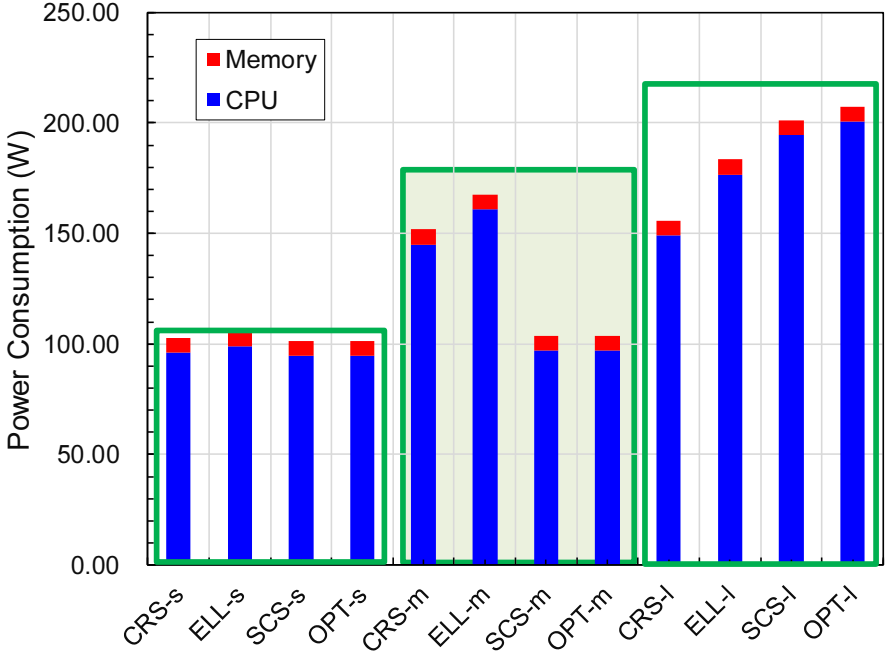


単精度・消費電力(W), Small/Medium/Large

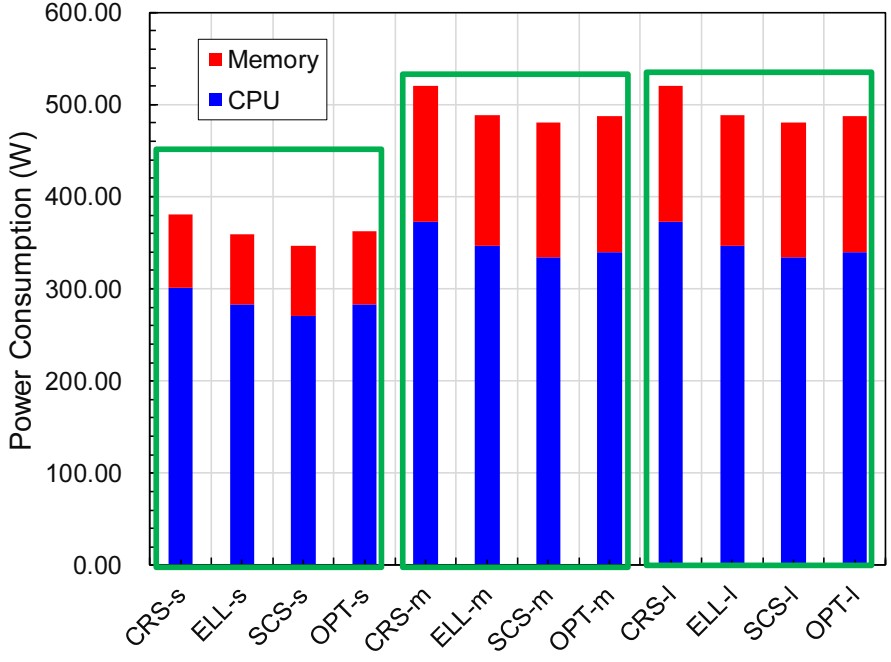
OFP/cの挙動は不審:演算密度高いSCS,OPTで低い

問題規模が大きくなるとW値も上がる, 特にOBCXのメモリ

OFP/c (Flat-MCDRAM)



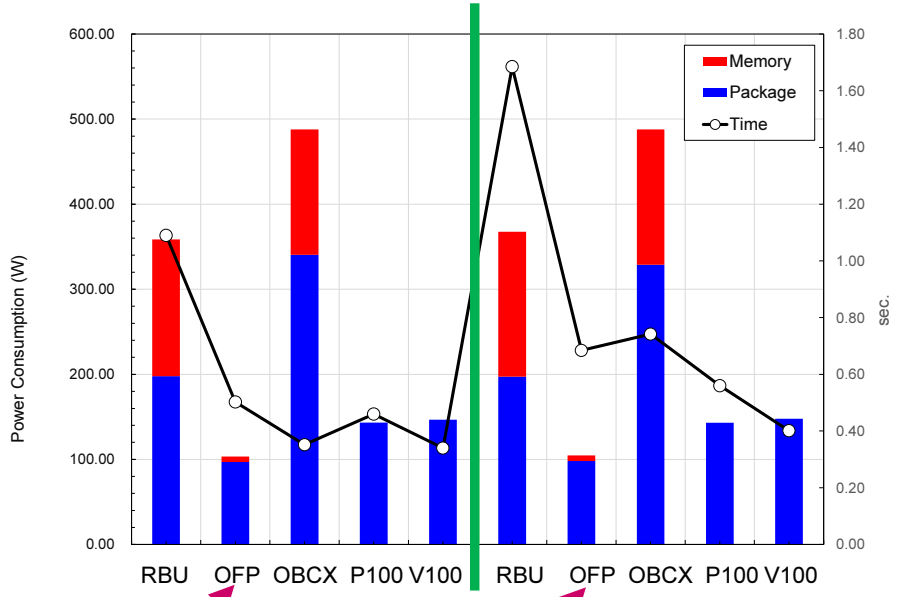
OBCX

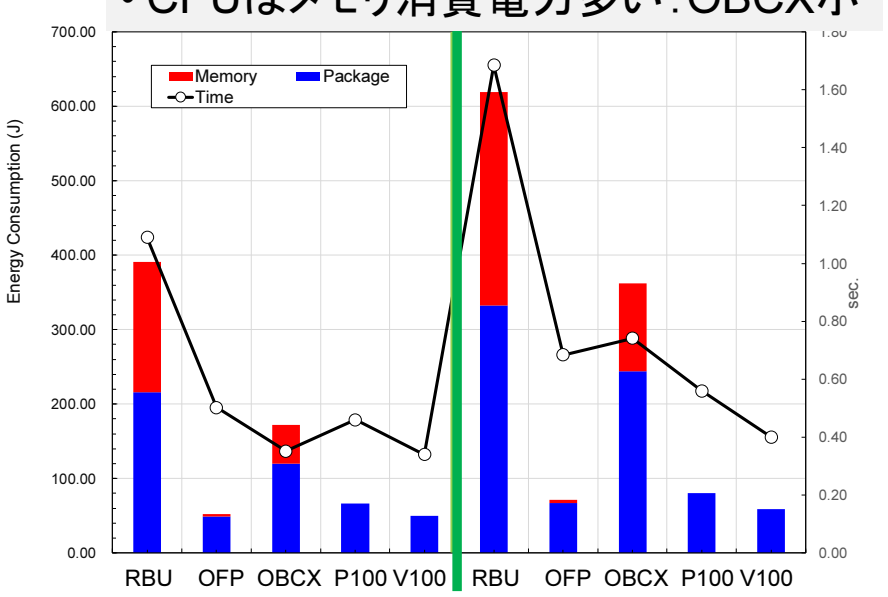


計算時間・消費電力(W)・消費エネルギー(J)

Medium，各HWの最適ケース
OFPのWatt値小，Largeでは200W程度

- P100⇒V100：35-40%高速
- OFPはP100より若干遅い
- CPUはメモリ消費電力多い：OBCX小





Single　Double

Power (W)

Memory
Package（CPU）

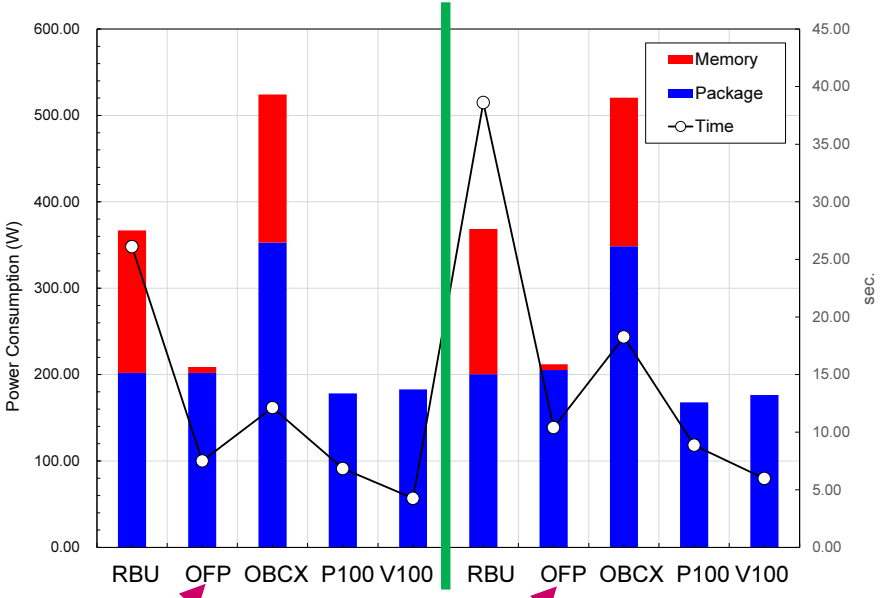
Single　Double

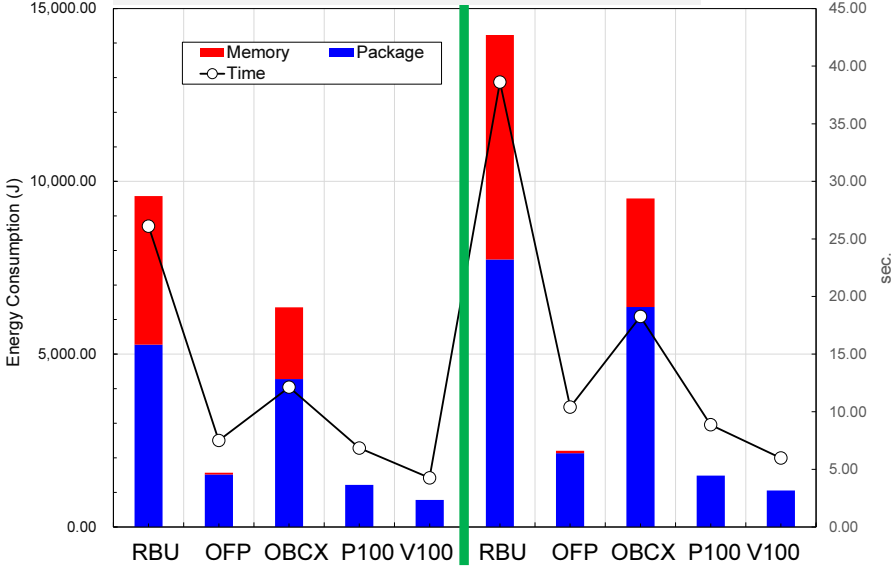
Energy (J)

計算時間・消費電力(W)・消費エネルギー(J)

Large，各HWの最適ケース
OFPのWatt値，Largeでは200W程度

- **P100⇒V100：45-60%高速**
- OFPはP100より若干遅い
- W値は若干上昇





Single　Double　　Single　Double

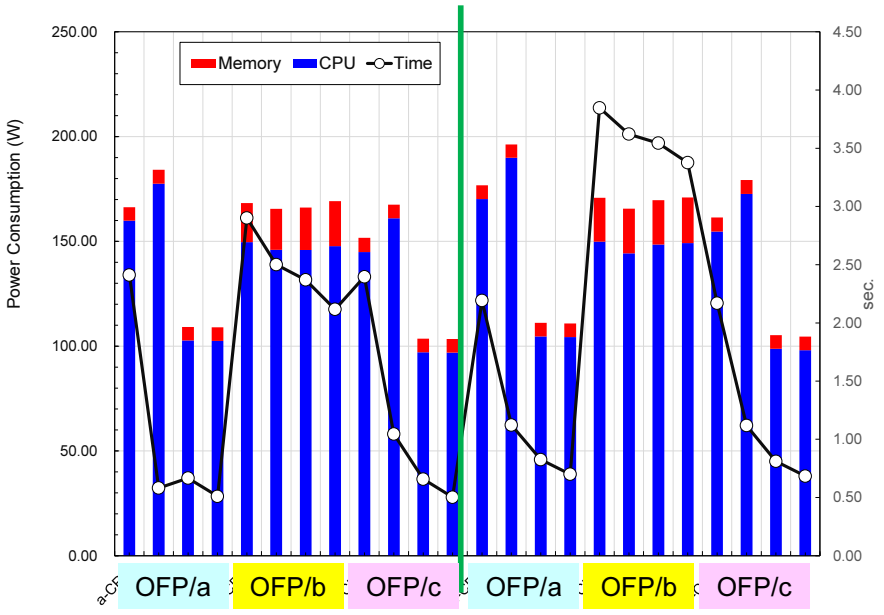
Power (W)

Memory
Package（CPU）

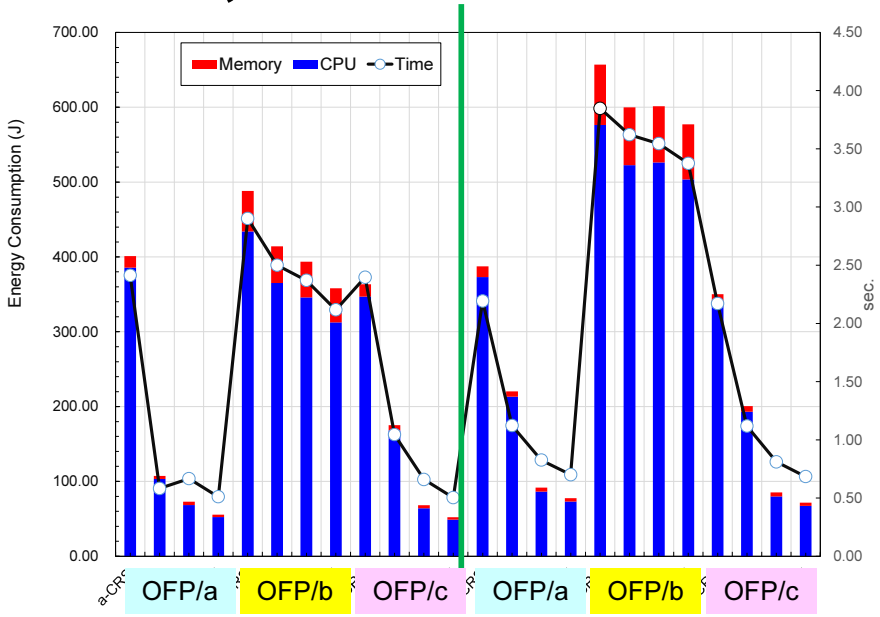
Energy (J)

計算時間・消費電力(W)・消費エネルギー(J)

OFP, Med.: a:cache, b:flat/DDR, c:flat/MCDRAM



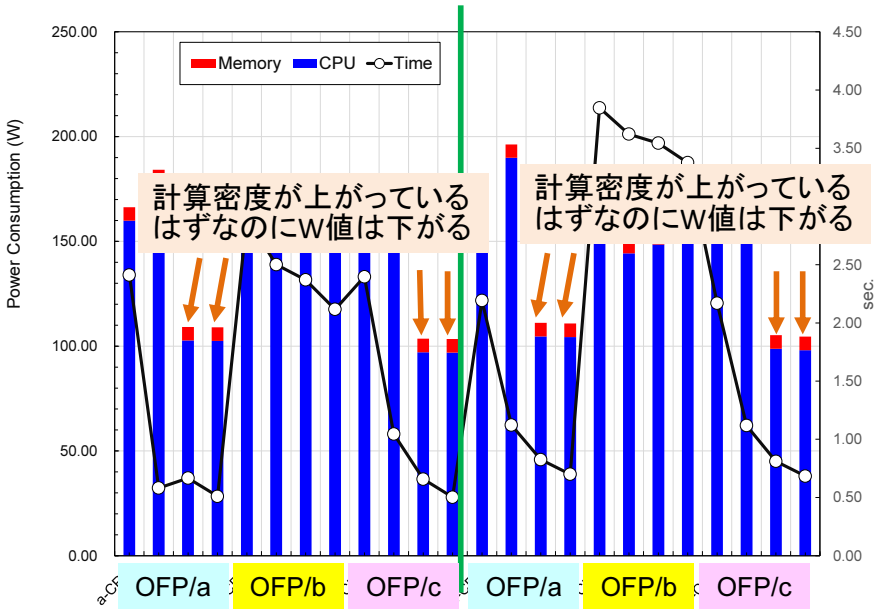
Power (W)

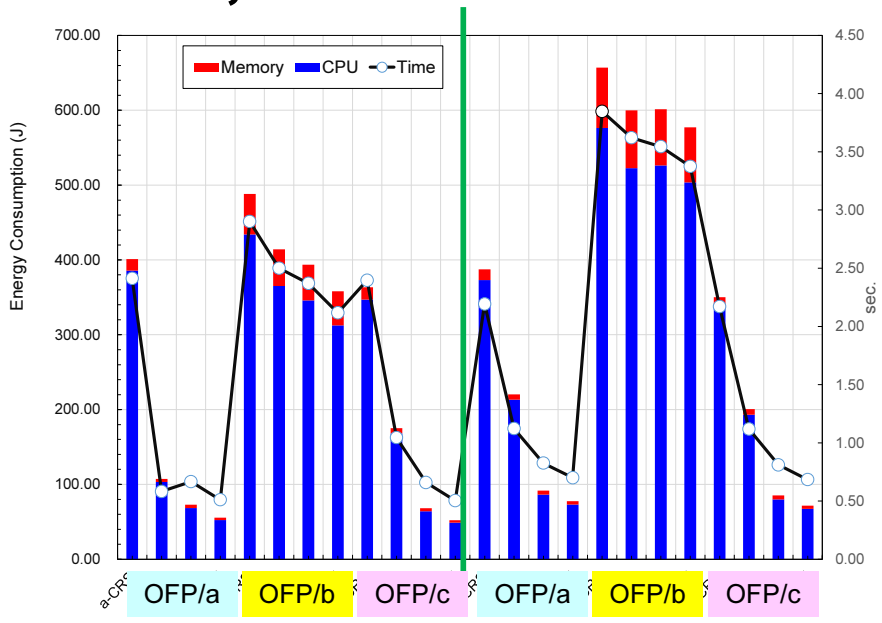


Energy (J)

計算時間・消費電力(W)・消費エネルギー(J)

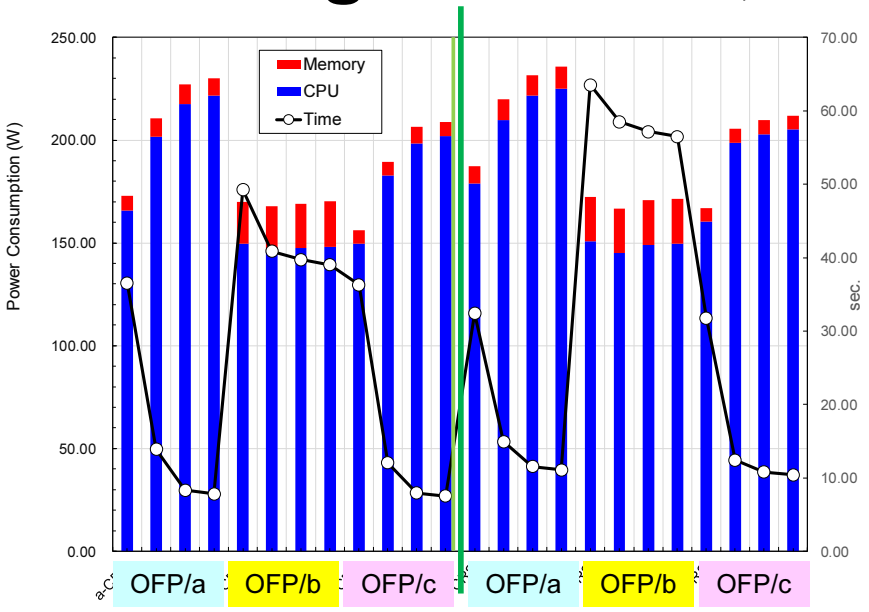
OFP, Med.: a:cache, b:flat/DDR, c:flat/MCDRAM





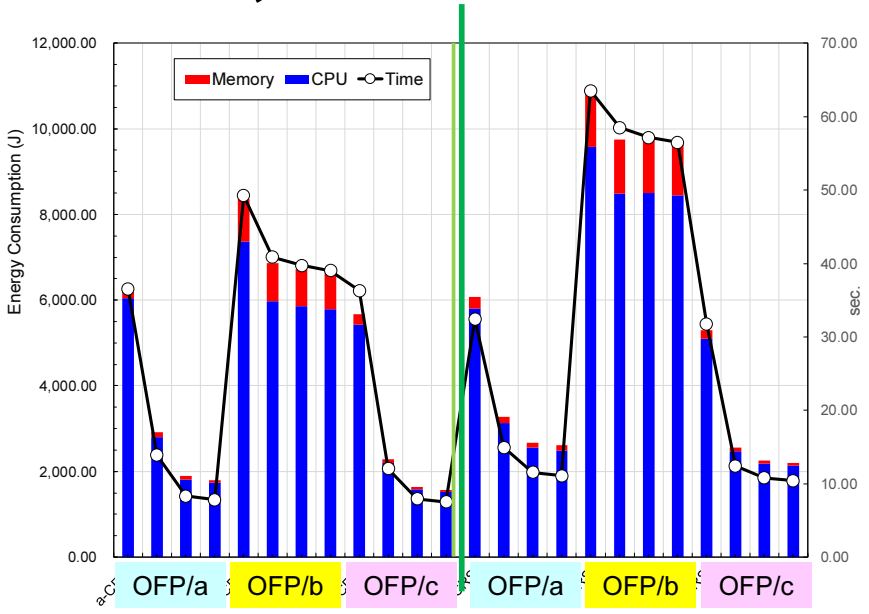
計算時間・消費電力(W)・消費エネルギー(J)

OFP, Large: a:cache, b:flat/DDR, c:flat/MCDRAM



Single Double

Power (W)

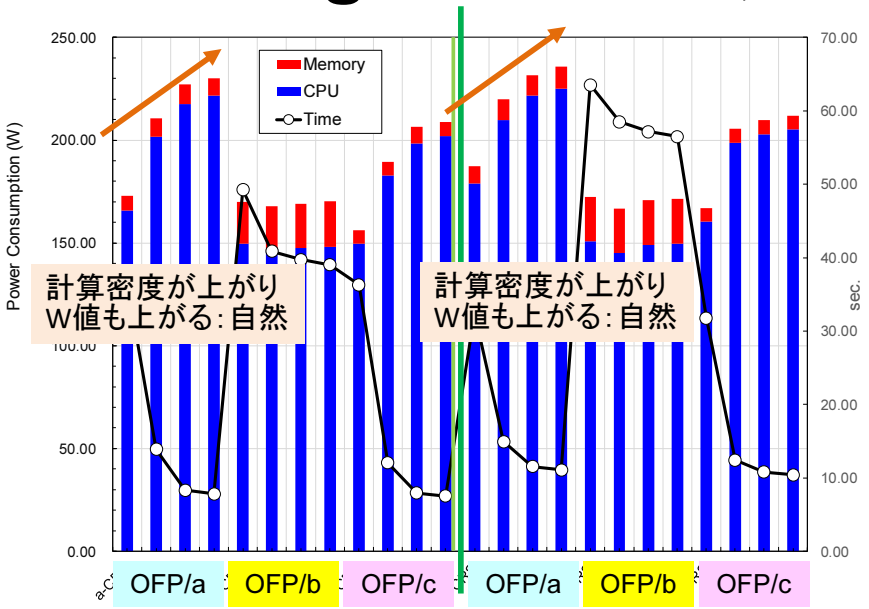


Single Double

Energy (J)

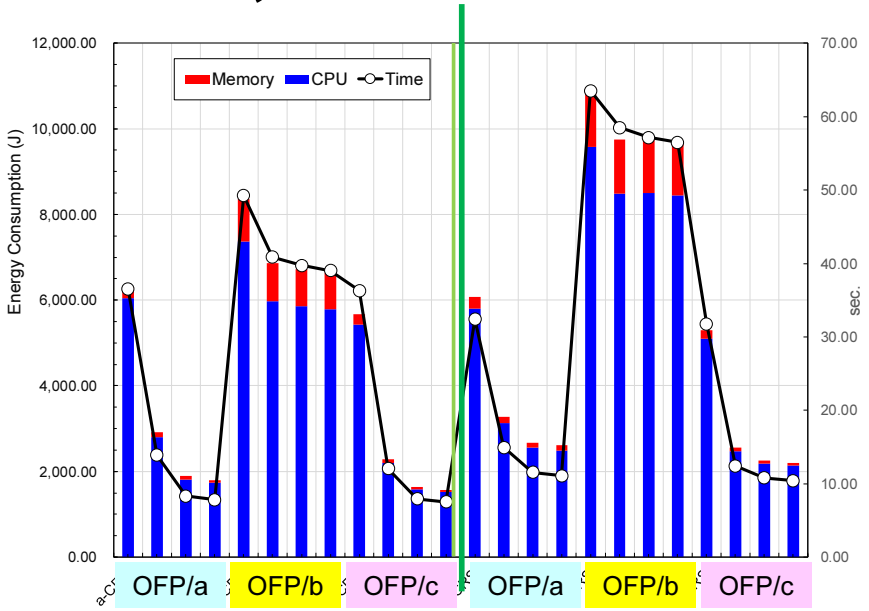
計算時間・消費電力(W)・消費エネルギー(J)

OFP, Large: a:cache, b:flat/DDR, c:flat/MCDRAM



Single Double

Power (W)



Single Double

Energy (J)

OFPの挙動

- a: Cacheモード
- b: Flatモード，DDR4のみ使用
- c: Flatモード，MCDRAMのみ使用
- Cacheモード（OFP/a）とFlat/MCDRAM（OFP/c）はほぼ同じ挙動である
  - これらのケースでは基本的にCPU部分に収納されているMCDRAMが主として使用されるため，メモリ部分の電力消費はほとんどない．
  - Flat/MCDRAM（OFP/c）は若干W少ない
- 問題サイズが大きくなると消費電力（W）はむしろOFP/bが他と比べて小さい
  - メモリ消費電力はRBU，OBCXと比較すると少ない
- MCDRAMとDDRのメモリバンド幅の比は6:1程度であるため，本研究で扱っている疎行列ソルバーのようなmemory-boundなアプリケーションでは，その影響は顕著
  - OFP/c-OPTとOFP/b-OPTの計算性能の比は倍精度の場合5倍程度である

まとめ(1/2)

- 著者等が先行研究において開発した有限体積法におけるICCGソルバーをターゲットとし，実装方法，問題規模と低精度演算による性能改善の関係に注目し，5(＋1)種類のハードウェア環境下での検討を実施した
- 実装方法(疎行列格納形式)，問題規模，ハードウェア環境によって，低精度演算を使用することにより，計算時間，消費電力(W)，消費エネルギー(J)への様々な効果があることがわかった
  - 問題規模大，SIMD化・ベクトル化が進んでいる場合(高演算密度)の効果大
  - OFP，GPU(P100，V100)はIntel Xeon(RBU，OBCX)と比較して消費電力，消費エネルギー共に小さい
- SELL-C-σは，OFPを除くと必ずしも有効でない
- Barrier-Free〔星野 HPC-158〕は有効
  - ELLへのBarrier-Free

まとめ(2/2)

- 今後は同様な検討を他のアプリケーションについても実施する予定である
  - 混合精度(例:前処理のみ単精度)
- FX700での測定
- GPUを使用する場合は，CPU使用電力(＋アイドル時電力(下表))も考慮した評価が必要

アイドル時電力	Package (W)	Memory (W)	TDP (W)
RBU (BDW)	60.25	6.65	120
OBCX (CLX)	50.37	8.66	205
Rome	88.50	?	225
P100	32.46	-	250
V100	36.80	-	300

2019年度研究成果

- 国際会議論文(査読付き)　8
  - ICCS 2019
  - SC19
    - ポスター: 3(うち一つはBest Poster Candidate)
    - WS: 2
  - HPC Asia 2020
- 国際会議発表　12
  - 招待講演:3
  - APCOM 2019 Keynote Talk
- 国内会議発表　15
- ソフトウェア公開
  - FP21AXPY OpenACC source code, https://github.com/y-mag-chi/fp21axpy

① Fujita, K., Horikoshi, M., Ichimura, T., Meadows, L., Nakajima, K., Hori, M., Maddegedara, L., Development of Element-by-Element Kernel Algorithms in Unstructured Implicit Low-Order Finite-Element Earthquake Simulation for Many-Core Wide-SIMD CPUs, Proceedings of ICCS 2019, Lecture Notes in Computer Science 11536, 267-280, 2019（査読付国際会議）

② Ogita, T., Nakajima, K., Verified solutions of large sparse linear systems arising from 3D Poisson equation in HPC environments, European Numerical Mathematics and Advanced Applications Conference 2019, 2019

③ Yamaguchi, T., Fujita, K., Ichimura, T., Naruse, A., Lalith, M., Hori, M., GPU implementation of a sophisticated implicit low-order finite element solver with FP21-32-64 computation using OpenACC, Proceedings of 6th WACCPD in conjunction with SC19, 2019（査読付国際会議）

④ Ootomo, H., Yokota, R., TSQR on TensorCores, Research Poster for SC19, 2019 (Best Poster Candidate, 査読付国際会議)

⑤ Sakamoto, R., Kondo, M., Fujita, K., Ichimura, T., Nakajima, K., The Effectiveness of Low-Precision Floating Arithmetic on Numerical Codes: A Case Study on Power Consumption, ACM Proceedings of HPC Asia 2020, 2020（査読付国際会議）

⑥ Ooi, R., Iwashita, T., Fukaya, T., Ida, A., Yokota, R., Effect of Mixed Precision Computing on H-matrix Vector Multiplication in BEM Analysis, ACM Proceedings of HPC Asia 2020, 2020（査読付国際会議）

まとめ・将来展望

- 2019年度末時点で，ほぼ当初の予定通り目標を達成
- 2020年度は各アルゴリズムの，演算精度，最適化，アーキテクチャの他，問題規模も考慮して消費電力・エネルギーへの体系的な影響評価を継続して実施する他，低精度・混合／変動精度演算に関する研究開発を，これまでの研究成果を元に継続して実施し，自動チューニング手法確立を目指す。
- 精度保証手法については，疎行列演算に加えて，H行列，偏微分方程式解法向け手法の研究開発も実施する（2019年度途中から検討を開始）。

- 2020年度詳細は「jh200037-NAH」へ