Vol.2018-HPC-167 No.26
2018/12/18

パイプライン型アルゴリズムによる並列共役勾配法の安定性評価

中島研吾[†1 †2]　荻田武史[†3]

パイプライン型アルゴリズムによる共役勾配法は，本来のアルゴリズムを保存しつつ，漸化式の適用によって計算順序を変更した手法であり，特に超並列環境では，MPI-3 のサポートする非同期集団通信関数によって，内積における集団通信と計算をオーバーラップさせることによって高いスケーラビリティを得られることが知られている。一方で，計算順序の変更により丸め誤差の伝播挙動が本来と異なるため，特に悪条件問題において収束が不安定となる場合がある。本研究では，悪条件問題におけるパイプライン型アルゴリズムによる共役勾配法の挙動を分析し，効率的で安定な計算手法について検討する。

Evaluations of Stability of Parallel Conjugate Gradient Methods based on Pipelined Algorithms

Kengo Nakajima[†1 †2 a)]　Takeshi Ogita [†3]

The Conjugate Gradient (CG) Method based on the *pipelined algorithm* introduces recurrent relations to the standard CG algorithm. Although the sequence of operations of such CG is different from that of the standard CG, the algorithm has not changed. The CG based on the pipelined algorithms with functions for asynchronous collective communications supported in the MPI-3 standard can hide overhead of global collective communications by overlapping communications and computations. On the other hand, the change of sequence of computation affects the convergence, because rounding errors may be propagated differently. This effect is significant in ill-conditioned problems. In the present work, we analyze behaviors of the CG based on pipelined algorithms in ill-conditioned problems, and investigate robust and efficient method for computation.

1. はじめに

本研究では，スケーラビリティの高い手法として注目されているパイプライン型アルゴリズムに基づく共役勾配法（Conjugate Gradient（CG） based on Pipelined Algorithms）の悪条件問題への適用時の安定性について検討する。

第 2 章ではパイプライン型アルゴリズムの概要について，著者等による先行研究〔1〕も交えて紹介する。第 3 章では不均質場における三次元定常熱伝導コードにおける求解部（ICCG 法）にパイプライン型アルゴリズムを適用した場合の効果を，倍精度，単精度，混合精度演算について評価する。第 4 章では，関連研究を紹介し，第 5 章では結論と将来展望について述べる。

2. パイプライン型アルゴリズム

2.1 背 景

差分法，有限要素法等の偏微分方程式の数値解法において，最も計算時間を要するプロセスは大規模な疎行列を係数行列とする連立一次方程式の求解であり，その最適化に向けて様々な試みがなされてきた。連立一次方程式の求解には共役勾配法（Conjugate Gradient，CG）に代表されるクリロフ部分空間法（反復法）が広く使用されているが，大規模並列計算においては通信によるオーバーヘッドが顕著となる。図 1 は連立一次方程式 Ax=b を前処理付き共役勾配法で解くアルゴリズムである：

```
1: r0:= b-Ax0; u0:= M^-1 r0; p0:= u0
2: for i=0 … do
3:     s:= Api
4:     α:= (ri,ui)/(s,pi)
5:     xi+1:= xi + αpi
6:     ri+1:= ri- αs
7:     ui+1:= M^-1 ri+1
8:     β:= (ri+1,ui+1)/(ri,ui)
9:     pi+1:= ui+1 + βpi
10: end do
```

図 1 前処理付き共役勾配法（Preconditioned Conjugate Gradient，PCG）のアルゴリズム（Alg.-1），Ax=b，M：前処理行列，赤字：疎行列ベクトル積（SpMV, Sparse Matrix Vector Multiply），青字：内積，緑字：前処理

分散メモリ型並列計算機上で並列計算を実施する場合は，①疎行列ベクトル積，②内積，③前処理で通信が発生する可能性がある。前処理手法によって通信パターンは異なるが，疎行列ベクトル積では隣接プロセスとの 1 対 1 通信（Point-to-Point Communication），内積では全プロセスとの集合通信（Collective Communication）が発生する。MPI を使用する場合は，前者は MPI_Isend，MPI_Irecv，後者は MPI_Allreduce 等が使用される。

通信によるオーバーヘッドを削減するために，通信回避・削減型アルゴリズム（Communication Avoiding/Reducing Algorithm）の研究開発が盛んに進められている。Matrix

†1 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
†2 理化学研究所計算科学研究センター
RIKEN Research Center for Computational Science (R-CCS)
†3 東京女子大学現代教養学部
School of Arts and Sciences, Tokyo Woman's Christian University
a) nakajima@cc.u-tokyo.ac.jp

Powers Kernel〔2〕に基づく *s*-step 法〔3〕では，*s*（*s*>1）反復分の通信を 1 反復で済ませることができ：

- 内積実施回数削減による集団通信オーバーヘッド削減
- 並列分散データの Halo 領域を大きくとって，*s* 回分の行列ベクトル積を，通信をしないで連続して実施する

ことが可能である。一般に *s* が大きくなると冗長計算が増加するとともに不安定性が増す。また *s*-step 法は，適用可能な前処理手法が限定されるという問題点がある。

パイプライン型共役勾配法（Pipelined CG Method）〔4〕は，*s*-step 法の元になったアルゴリズムに基づき，漸化式の適用によって，図 1 に示したオリジナルの前処理付き CG 法のアルゴリズムが変わらないように計算の順序を変更する手法である。内積の直後に疎行列ベクトル積，前処理などの計算量が多く，かつ直前で実施した内積の結果を使わないような処理を実行するように計算順序が変更される。Pipelined 法では，これに MPI-3 でサポートされている MPI_Iallreduce 等の非同期集団通信（Asynchronous Collective Communication）〔5〕を組み合わせることによって，集団通信と疎行列ベクトル積，前処理等の演算をオーバーラップすることができ，通信によるオーバーヘッドの隠蔽が可能となる。

疎行列ベクトル積は，限定された数の隣接プロセスとの通信であるが，集団通信は全プロセスに対する通信であり，大規模並列計算機システムにおいてノード数が増加すると，オーバーヘッドはより顕著となる。図 1 に示すオリジナルの前処理付き共役勾配法では，内積 1 回の通信量はスカラー変数 1 つ分であり，いわゆるレイテンシの効果が大きい。

著者等による先行研究〔1〕では，〔4〕に示されたパイプライン型共役勾配法，及び関連手法を，三次元有限要素法構造解析コードへ適用し，Reedbush-U（RBU）（東京大学情報基盤センター）〔5〕で計算を実施した。

先行研究〔1〕では，GeoFEM プロジェクト〔7,8〕で開発された並列有限要素法アプリケーションを元に整備した性能評価のためのベンチマークプログラム「GeoFEM／Cube」である。本ベンチマークは，均質場における三次元弾性静解析問題に関する並列前処理付き反復法による疎行列ソルバーの実行時性能を様々な条件下で計測するものである。プログラムは全て OpenMP ディレクティヴを含む Fortran90 および MPI で記述されている。三次元弾性静解析問題では係数行列が対称正定な疎行列となることから，前処理を施した共役勾配法（Conjugate Gradient，CG）法によって連立一次方程式を解いている。前処理手法としては，各 MPI プロセスの扱う領域に対して，局所化されたブロックヤコビ型 Symmetric Gauss Seidel（SGS）法を適用している〔7,8〕。各 MPI プロセス内の OpenMP による並列化には，並列性に優れたマルチカラー法（Multicoloring，MC）とより安定した収束を示す Reverse Cuthill-McKee （RCM）法を組み合わせ，RCM 法に Cyclic マルチカラー法（Cyclic Multicoloring，CM）を適用した CM-RCM(k)法を使用している〔8,9,10〕。局所 SGS 前処理法は，領域分割数を増加させると，反復回数が増加する傾向にあるため，収束を安定化させる手法として，領域間オーバーラップ領域に加法型 Schwartz 領域分割法（Additive Schwartz Domain Decomposition：ASDD）〔11〕を適用している。

2.2 パイプライン型アルゴリズム

本研究では，以下の 4 種類のアルゴリズムを使用する：

① オリジナル前処理付き共役勾配法（Alg.-1）（図 1）
② Chronopoulos/Gear アルゴリズム（Alg.-2）〔3,4〕
③ パイプライン型共役勾配法（Alg.-3）〔4〕
④ Gropp アルゴリズム（Alg.-4）〔12〕

本研究では，Alg.2～Alg.4 を総称して，「パイプライン型アルゴリズム」と呼ぶ。Chronopoulos/Gear アルゴリズム（Alg.-2）は *s*-step 法〔3〕において *s*=1 としたもので，下記のような漸化式（1）：

$$
\begin{aligned}
&s_i = Au_i + \beta_i s_{i-1},\ p_i = u_i + \beta_i p_i \Leftrightarrow s_i = Ap_i \\
&x_{i+1} = x_i + \alpha_i p_i \\
&r_{i+1} = b - Ax_{i+1} = b - Ax_i - \alpha_i Ap_i \\
&\quad = r_i - \alpha_i Ap_i = r_i - \alpha_i s_i
\end{aligned}
\qquad (1)
$$

を使用して，$s_i = Ap_i$ を陽に計算することなく求めている。結果として，図 8 に示すようなアルゴリズムが得られる。このアルゴリズムの特徴は，γ_i，δという内積処理を連続して実施できることにある（図 8 の 10 行目，11 行目）。したがって，実装上は MPI_Allreduce を 1 回呼ぶだけで済むため，全 MPI プロセスが関わる集合通信の回数を 1 回削減することができる：

```
1: r0:= b-Ax0; u0:= M^-1 r0; w0:= Au0
2: α0:= (r0,u0)/(w0,u0); β0:=0; γ0:= (r0,u0)
3: for i=0 … do
4:     pi:= ui + βi pi-1
5:     si:= wi + βi si-1
6:     xi+1:= xi + αi pi
7:     ri+1:= ri - αi si
8:     ui+1:= M^-1 ri+1
9:     wi+1:= Aui+1
10:    γi+1:= (ri+1,ui+1)
11:    δ:= (wi+1,ui+1)
12:    βi+1:= γi+1/ γi
13:    αi+1:= γi+1/ ( δ -  βi+1 γi+1/αi)
14: end do
```

図 2　Chronopoulos/Gear アルゴリズム（Alg.-2）〔3,4〕

Alg.-2 に以下の漸化式（2）を適用すると，図 3 に示すアルゴリズムが得られる。このアルゴリズムはパイプライン型

Chronopoulos/Gear アルゴリズム（前処理無し）である。

$$
\begin{aligned}
&u_i = r_i, w_i = Au_i = Ar_i \\
&Ar_{i+1} = Ar_i - \alpha_i As_i \Rightarrow w_{i+1} = w_i - \alpha_i As_i \\
&As_i = Aw_i + \beta_i As_{i-1} \Rightarrow z_i \left(= As_i = A^2 p_i\right) = Aw_i + \beta_i z_{i-1} \\
&q_i = Aw_i = A^2 r_i \\
&\quad = r_i - \alpha_i Ap_i = r_i - \alpha_i s_i
\end{aligned}
\qquad (2)
$$

このアルゴリズムの特徴は，Alg.-2 と同様に γ_i，δ という内積処理を連続して実施できるだけでなく（図 3 の 3 行目，4 行目），更にこれらの内積の値は 5 行目の疎行列ベクトル積では使用されず，6 行目以降で使用されるということである。この部分に，MPI-3 でサポートされている非同期集団通信機能を適用すれば，内積計算のための集合通信と疎行列ベクトル積をオーバーラップさせることが可能となる。

```
1: r0:= b-Ax0; w0:= Au0
2: for i=0 … do
3:     γi:= (ri,ri)
4:     δ:= (wi,ri)
5:     qi+1:= Awi+1
6:     if i>0 then
7:     βi:= γi/γi-1; αi:= γi/(δ - βi γi/αi-1)
8:     else
9:     βi:= 0; αi:= γi/δ
10:    end if
11:    zi:= qi + βi zi-1
12:    si:= wi + βi si-1
13:    pi:= ri + βi pi-1
14:    xi+1:= xi +αi pi
15:    ri+1:= ri - αi si
16:    wi+1:= wi - αi zi
17: end for
```

図 3　パイプライン型 Chronopoulos/Gear アルゴリズム（前処理無し）〔4〕

更に，これに以下に示す漸化式（3）を適用すると，図 4 に示すパイライン型共役勾配法（前処理付き）のアルゴリズム（Alg.-3）〔4〕が得られる：

$$
\begin{aligned}
&\gamma_i = (u_i, u_i)_M = (Mu_i, u_i) = (r_i, u_i) \\
&\delta_i = (M^{-1}Au_i, u_i)_M = (Au_i, u_i) = (w_i, u_i) \\
&M^{-1}r_{i+1} = M^{-1}r_i - \alpha_i M^{-1}s_i \Rightarrow u_{i+1} = u_i - \alpha_i q_i \quad \left(q_i = M^{-1}s_i\right) \\
&M^{-1}s_{i+1} = M^{-1}w_i + \beta_i M^{-1}s_i \Rightarrow q_{i+1} = M^{-1}w_i + \beta_i q_i \\
&Au_{i+1} = Au_i - \alpha_i Aq_i \Rightarrow w_{i+1} = w_i - \alpha_i Aq_i \\
&Aq_i = AM^{-1}w_i + \beta_i Aq_{i-1} \Rightarrow z_i = Am_i + \beta_i z_{i-1} \\
&\quad \left(m_i = M^{-1}w_i = M^{-1}Au_i = M^{-1}AM^{-1}r_i, z_i = Aq_i\right)
\end{aligned}
\qquad (3)
$$

このアルゴリズムも Alg.-2，図 3 に示すアルゴリズムの特性を継承しており，内積（3 行目，4 行目）の後に前処理，疎行列ベクトル積の計算があり，その後初めて内積の値を使用するため，集団通信と計算のオーバーラップが可能である。この他〔4〕で紹介されている Gropp アルゴリズム（Alg.-4）〔12〕は，Alg.-3 と良く似たアルゴリズムであるが，δ の導出法が異なっており，計算量も若干少なくなっている（図 5）。

```
1: r0:= b-Ax0; u0:= M-1r0; w0:= Au0
2: for i=0 … do
3:     γi:= (ri,ui)
4:     δ:= (wi,ui)
5:     mi:= M-1wi
6:     ni:= Ami
7:     if i>0 then
8:     βi:= γi/γi-1; αi:= γi/(δ - βi γi/αi-1)
9:     else
10:    βi:= 0; αi:= γi/δ
11:    end if
12:    zi:= ni + βi zi-1
13:    qi:= mi + βi qi-1
14:    si:= wi + βi si-1
15:    pi:= ui + βi pi-1
16:    xi+1:= xi +αi pi
17:    ri+1:= ri - αi si
18:    ui+1:= ui - αi qi
19:    wi+1:= wi - αi zi
20: end for
```

図 4　パイプライン型共役勾配法（前処理付き）のアルゴリズム（Alg.-3）〔4〕

```
1: r0:= b-Ax0; u0:= M-1r0; p0:= u0; s0:= Ap0;
γ0:= (r0,u0)
2: for i=0 … do
3:     δ:= (pi,si)
4:     qi:= M-1si
5:     αi:= γi/δ
6:     xi+1:= xi +αi pi
7:     ri+1:= ri - αi si
8:     ui+1:= ui - αi qi
9:     γi+1:= (ri+1,ui+1)
10:    wi+1:= Aui+1
11:    βi+1:= γi+1/γi
12:    pi+1:= ui+1 + βi+1 pi
13:    si+1:= wi+1 + βi+1 si
14: end for
```

図 5　Gropp アルゴリズム〔4,12〕

図 6 は Gropp アルゴリズム（図 5）の 3～5 行目を，Fortran，OpenMP，MPI で実装した例である。内積計算用集団通信関数として MPI_Allreduce ではなく MPI-3 でサポートされている非同期集団通信関数 MPI_Iallreduce を呼んでいる。内積の計算結果（δ）を使用する直前に MPI_Wait を呼んで同期しているため，集団通信と前処理計算部（4 行目）をオーバーラップすることが可能である。

```
!C> 3: δ:= (pi,si)
      DLO= 0.d0
!$omp parallel do private(i) reduction(+:DLO)
      do i= 1, 3*N
        DLO= DLO + P(i)*S(i)
      enddo
      call MPI_Iallreduce                          &
     &    (DLO, Delta, 1, MPI_DOUBLE_PRECISION,    &
     &     MPI_SUM, MPI_COMM_WORLD, request, err)

!C> 4: qi:= M-1si
   （前処理：省略）

   call MPI_Wait (request, status, err)

!C> 5: αi:= γi/δ
   Alpha= Gamma / Delta
```

図 6　Gropp アルゴリズムの実装例（非同期集団通信と計算のオーバーラップ）

表 1 は各アルゴリズムの計算量である．DAXPY はベクトルの定数倍の加減（例：$x_i = x_i + \alpha_i p_i$）である。Alg.3，Alg.4 はオリジナルの手法と比較して，計算量が増えているが，SpMV や前処理と比較すると計算量は少なく，無視できる。Alg.1～Alg.4 はアルゴリズム的には同じ計算内容であるが，計算順序が変わっているため，丸め誤差の伝播の仕方が異なっている。従って，悪条件問題の場合，アルゴリズムによって収束の履歴が異なる場合がある。〔4〕では，そのような事例が実際にあるため，50 反復に一回残差ベクトル（r_i）の値を $r_i = b - Ax_i$ によって補正している。先行研究〔1〕で対象としている問題では，そのような補正は不要で，各アルゴリズム全く同じ履歴，同じ回数で収束した。

表 1　各アルゴリズムの計算量（SpMV：疎行列ベクトル積，DAXPY：ベクトル定数倍加減），内積の（+1）は残差ベクトルのノルム計算

Alg.		SpMV	前処理	内積	DAXPY
1	Original CG	1	1	2+1	3
2	Chronopoulos/Gear	1	1	2+1	4
3	Pipelined CG	1	1	2+1	8
4	Gropp	1	1	2+1	5

2.3 計算機環境

先行研究〔1〕では東京大学情報基盤センターの Reedbush-U システム〔6〕を使用した。Reedbush-U の計算ノードは，CPU として Intel Xeon E5-2695v4 (Broadwell-EP) を 2 ソケット搭載している。CPU の各コアには，AVX2 命令に対応したベクトルユニットが 2 基ずつ搭載されている。これらの計算ノード 420 ノードは，InfiniBand-EDR によりフルバイセクションバンド幅を持つ Fat-tree 網で接続されている。Intel Compiler，Intel Parallel Studio に含まれる Intel MPI を使用した。表 2 は各ソケットの概要である。

表 2　Reedbush-U 各ソケットの概要

CPU	Intel Xeon E5-2695 v4（Broadwell-EP）
動作周波数（GHz）	2.10
コア数	18
理論演算性能（GFLOPS）	604.8
主記憶容量（GB）	128
メモリバンド幅性能（GB/sec, Stream Triad）	65.5
インタコネクト	Mellanox InfiniBand EDR 4x (100 Gbps) Full-bisection BW Fat-tree

2.4 計算結果

Reedbush-U（RBU）を使用した計算の概要は表 3 の通りである。1 ソケットの 18 コアのうち 16 コアを使用し，最小 2 ノード（4 ソケット, 64 コア）から最大 384 ノード（768 ソケット，12,288 コア）までの Strong Scaling 性能を評価した。

表 3　Reedbush-U における計算の概要

問題規模	Small：256×128×144 節点（14,155,776 DOF） Medium：256×128×288 節点（28,311,552DOF）
利用ノード数	最小 2 ノード，4 ソケット 最大 384 ノード，768 ソケット 1 ソケットあたり 16 コア使用，最大 12,288 コア Small：コア当り最小問題サイズ＝1,152 DOF/core Medium：同＝2,304 DOF/core
並列プログラミングモデル	Flat MPI：1 コアあたり 1MPI プロセス Hybrid：1 ソケットあたり 1MPI プロセス，16 スレッド
ノード内リオーダリング法	RCM
ASDD 反復数	2 回
CG 法収束条件	$\|r\|/\|b\| = \|b-Ax\|/\|b\| < 1.0\times10^{-8}$

ブロックヤコビ型局所 SGS 前処理を適用しているため，コア数が増加すると反復回数が増加する傾向があるものの，ASDD（加法型 Schwartz 領域分割法）の効果によって Hybrid の場合は 64 コアから 12,288 コアまで 10%程度の反復回数増加に留まっている〔1〕。図 7 は Hybrid の 384 ノード, 12,288 コアまでの速度向上率である。Flat MPI　2 ノード（4 ソケット，64 コア）の Alg.1 の性能を 64.0 としている。Small，Medium のいずれの場合にも Alg.1 の速度向上率が低く，Alg.2 がややそれより良く，Alg.3，Alg.4 は更に良く，両者の速度向上率はほぼ同じである。Medium での Alg.3，Alg.4 の速度向上率は 12,888 コアで 9,000 を超えており，約 75%の並列化効率が得られている。

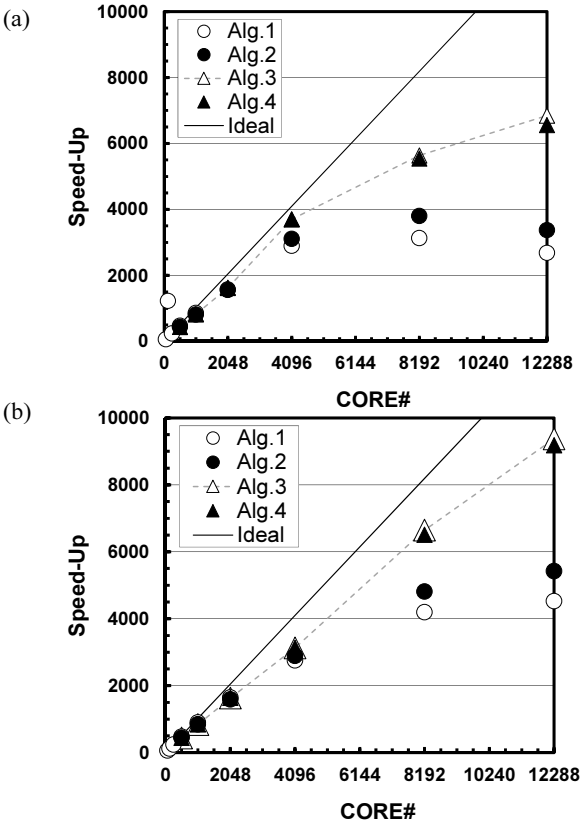

図 7 並列 CG 法の計算性能（Hybrid），Flat MPI　2 ノード（4 ソケット，64 コア）（Alg.1）の性能を 64.0 としたときの速度向上率（a）Small，（b）Medium

図 8 は Alg.3，Alg.4 において非同期集団通信関数（MPI_Iallreduce）を使った場合（IAR）と通常の MPI_Allreduce を使った場合（AR）と Alg.1 を比較したものである。非同期集団通信関数を使用しない場合は，Alg.1 とほぼ同じ挙動を示していることがわかり，非同期集団通信機能の適用により，通信と計算のオーバーラップを実現し，大幅な性能向上が得られていることがわかる。

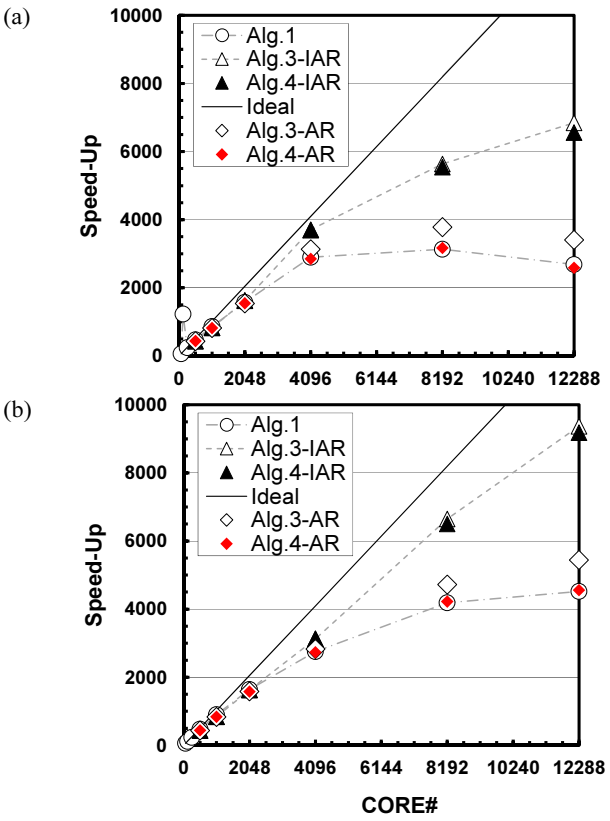

図 8　並列 CG 法の計算性能（Hybrid），Flat MPI　2 ノード（4 ソケット，64 コア）（Alg.1）の性能を 64.0 としたときの速度向上率（a）Small，（b）Medium, IAR : MPI_Iallreduce 使用，AR : MPI_Reduce 使用

3. 悪条件問題におけるパイプライン型アルゴリズム

3.1 高性能・変動精度・高信頼性数値解析手法とその応用

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のためには，通信最適化が必須であり，様々な提案がなされているが，本稿で扱うパイプライン型アルゴリズムは有効である。もう一つの技術的課題は消費電力，エネルギー（以下「消費電力」）である。ハードウェア的技術革新と共に，省電力・省エネルギー（以下「省電力」）のためのアルゴリズムの開発，実用化により，実計算時の消費電力の抑制が期待される。Approximate Computing〔13〕は，低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る。混合精度演算はその一種であり，既に多くの研究事例があるが，Approximate Computing では，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）の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・変動精度使用時，悪条件問題には重要である。昨今は，スパコンによる大規模計算向け精度保証の研究も実施されているが，実問題で現れる大規模疎行列・H 行列（階層化行列，密行列を低ランク近似等により階層化する手法）系への応用例はほとんどない。著者等はこのような背景に基づき，学際大規模情報基盤共同利用・共同研究拠点 2018 年度共同研究課題「高性能・変動精度・高信頼性数値解析手法とその応用（課題番号: jh180023-NAH）」〔14〕に取り組んでいる。当該研究は，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，半精度から倍精度，倍々精度までの広範囲をカバーする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新しい試みであり，開発された手法を様々なアプリケーションに適用することで，低精度を中心とした変動精度演算の科学技術シミュレーションへの有効性の検証を目的としている。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演算の有効性を消費電力の観点から明らかにすることを目指している〔15〕。

3.2 アプリケーションの概要

本研究では，著者等による先行研究〔10〕で使用されているプログラムに基づき検討を実施した。図 9 に示す差分格子によってメッシュ分割された三次元領域において，(4) に示す定常熱伝導方程式を解くアプリケーションを対象とし，導出された対称正定な疎行列を係数行列とする大規模連立一次方程式を不完全コレスキー分解前処理付き共役勾配法（ICCG）法によって解く〔10〕。

熱伝導率λの分布は図 10 に示すように，一層のみ$\lambda=\lambda_2$，他の部分は$\lambda=\lambda_1$とする。本節では$\lambda_2 \leqq \lambda_1$とし，$\lambda_1/\lambda_2$の比を様々に変化させた場合の計算を実施した。

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)=f \qquad (4)$$

$$\phi = 0 @ z = z_{\max}$$

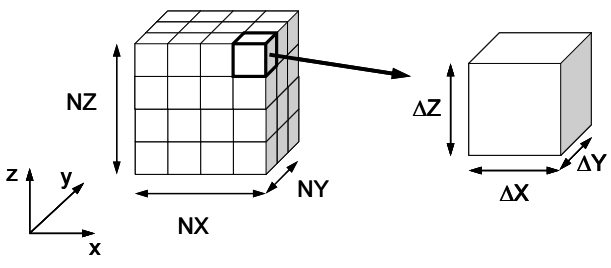

図 9　三次元ポアソン方程式ソルバーの解析対象

差分格子の各メッシュは直方体（辺長さはΔX，ΔY，ΔZ），
X，Y，Z 各方向のメッシュ数は NX，NY，NZ
形状は規則正しい差分格子であるが，プログラムの中では，一般性を持たせるために，有限体積法に基づき，非構造格子型のデータとして考慮する〔10〕。

先行研究〔10〕のプログラムの実数変数は全て倍精度であるが，本研究では，これを全て単精度にした場合，部分的に単精度演算を導入した混合精度演算の効果について検討する。

疎行列格納法として CRS（Compressed Row Storage）を適用し，番号付けを Sequential とした場合の計算を実施する〔10〕。問題サイズは NX=NY=NZ=128 とし，CM-RCM（20）（色数＝20 の CM-RCM 法）〔8,9,10〕とする。

3.3 計算結果：倍精度・単精度の比較

オリジナルの共役勾配法（Alg.1）に対して，λ_1/λ_2 を変化させ，倍精度，単精度により計算を実施した。本研究では，悪条件問題におけるパイプライン型アルゴリズムに基づく共役勾配法（Alg.2～Alg.4）の安定性，倍精度・単精度演算の挙動の評価が目的であるため，複数ノードではなく，Reedbush-U システム〔6〕の 1 ノードを使用して計算を実施し，OpenMP による並列化のみを適用した〔10〕。図 11 は，倍精度演算時の反復回数，計算時間を 1 とした場合の単精度演算時の反復回数，計算時間の比である。

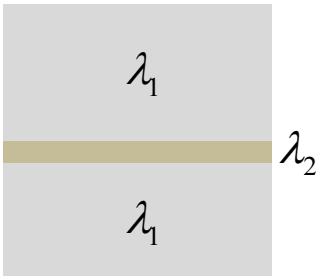

図 10　熱伝導率λの分布（$\lambda_2 \leqq \lambda_1$）

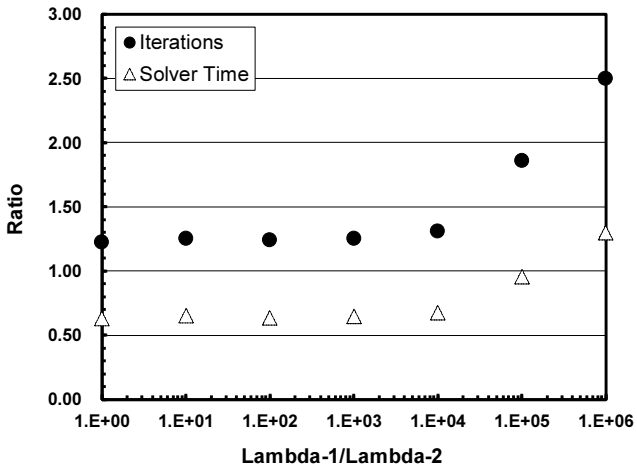

図 11　Alg.1：単精度／倍精度演算時の反復回数比・計算時間とλ_1/λ_2 の関係（NX=NY=NZ=128）（倍精度の場合を 1 とする）

λ_1/λ_2（条件数に比例）が大きくなるに従って単精度演算時の反復回数が相対に大きくなっていることがわかる。単精度演算の反復回数が安定している$\lambda_1/\lambda_2=10^4$ までは，計算時間の単精度／倍精度比は 0.60 程度であるが，それより大きい場合は，単精度演算の反復回数が増加して，倍精度演算の場合より計算時間が増えている。図 12 は，図 9 における，Bottom 面と Top 面の代表点の計算結果の誤差である。倍精度演算の結果が正しいものとして，単精度演算結果との誤差を示している。$\lambda_1/\lambda_2=10^6$ では誤差が約 10 に達し，正しい解から大きな差がある。

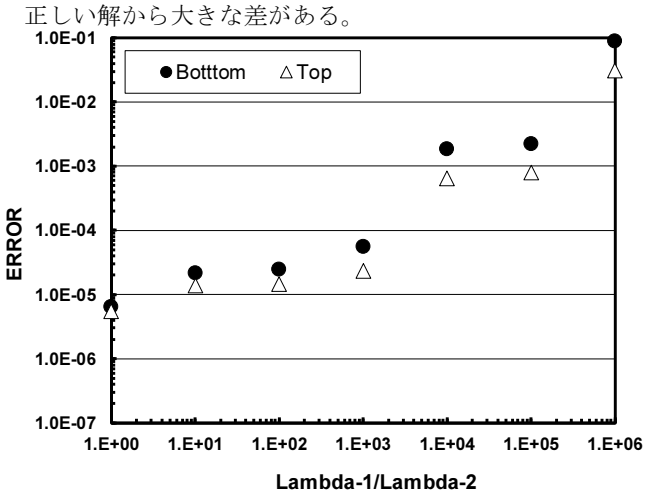

図 12　Alg.1：単精度／倍精度演算時の代表点（Bottom 面，Top 面）（図 3）計算結果相対誤差とλ_1/λ_2 の関係（(NX=NY=NZ=128)

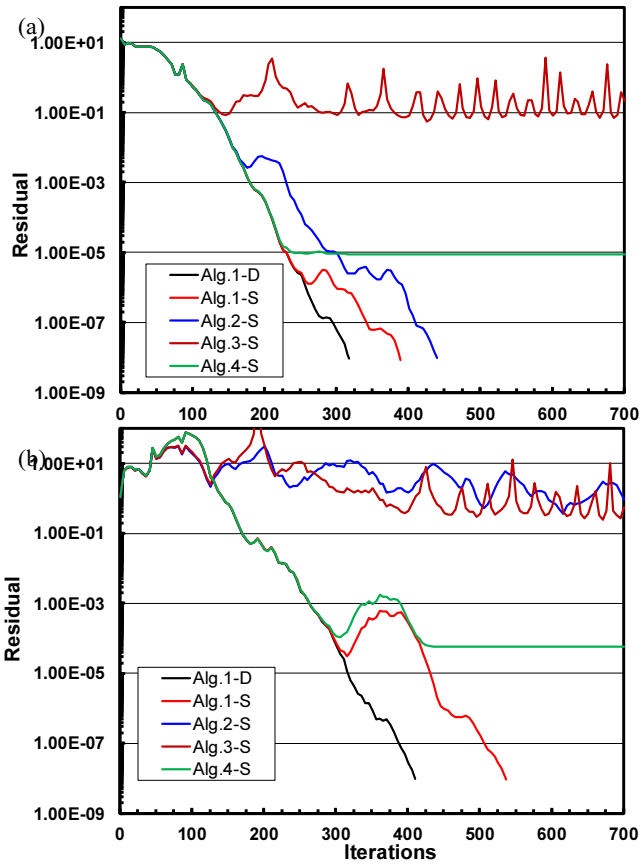

図 13　各アルゴリズムの収束履歴（NX=NY=NZ=128），-D：倍精度演算，-S：単精度演算，(a)$\lambda_1/\lambda_2=10^0$，(b)$\lambda_1/\lambda_2=10^4$

続いて，Alg.2～Alg.4 を同じ問題に適用した場合の結果について述べる。図 13 は$\lambda_1/\lambda_2=10^0, 10^4$ の場合について各アルゴリズムを適用した場合の残差ベクトルノルムの収束履歴である。倍精度演算（-D）の場合は各アルゴリズムで同じ収束履歴が得られているが，単精度演算（-S）の場合は各種法において異なっている。Alg.1（オリジナルのアルゴリズム）が最も安定であるが，Alg.2，Alg.3 は不安定である。特に Alg.3（パイプライン型共役勾配法）は不安定で，単精度演算を適用した場合には条件数の比較的小さい $\lambda_1/\lambda_2=10^0$ の場合でも残差履歴が停滞している。Alg.4（Gropp アルゴリズム）は Alg.2，Alg.3 と比較して安定しており，残差は停滞しているがほぼ正解に近い結果が得られている。

3.4 計算結果：混合精度の導入

続いて倍精度演算に対して，前処理部分（不完全コレスキー分解，前進後退代入）にのみ単精度演算を適用した混合精度演算について検討を実施した。図 14 は$\lambda_1/\lambda_2=10^0, 10^4$ の場合について各アルゴリズムを適用した場合の残差ベクトルノルムの収束履歴である。全般的に収束は改善し，Alg.1 の場合はいずれのケースでも倍精度の場合とほぼ同じ反復回数で収束している。また単精度では収束しなかった$\lambda_1/\lambda_2=10^4$ の場合における Alg.2 も収束している。

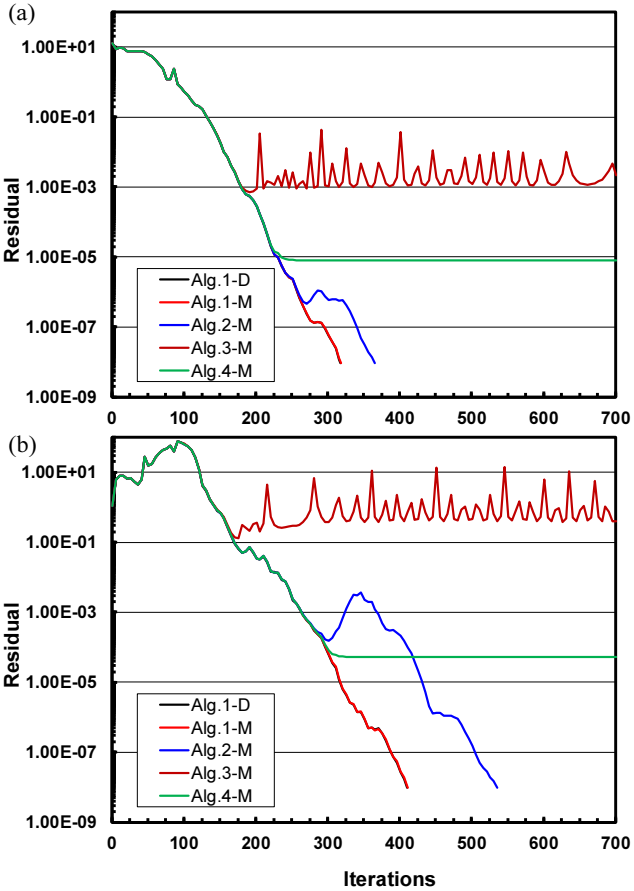

図 14　各アルゴリズムの収束履歴（NX=NY=NZ=128），-D：倍精度演算，-M：混合精度演算，（a）$\lambda_1/\lambda_2=10^0$，（b）$\lambda_1/\lambda_2=10^4$

図 15 は，$\lambda_1/\lambda_2=10^0, 10^4$ の場合について図 3 における Bottom 面における代表点における計算結果を比較したものである。Alg.1・倍精度の場合を基準としている。収束していない場合は，700 回反復した値を使用している。混合精度演算導入によって，Alg.2，Alg.3 の計算精度が向上していることがわかる。

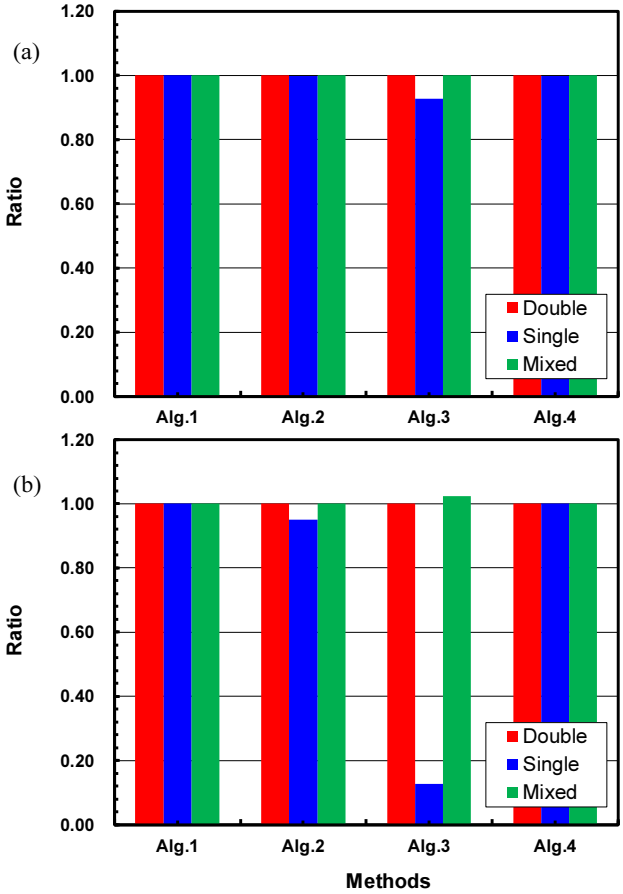

図 15　Bottom 面代表点（図 3）における計算結果の比較，Alg.1・倍精度演算の場合を基準（NX=NY=NZ=128），（a）$\lambda_1/\lambda_2=10^0$，（b）$\lambda_1/\lambda_2=10^4$

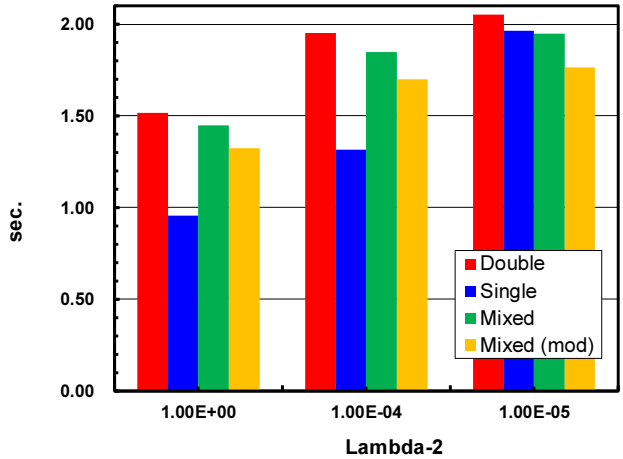

図 16　Alg.1 の計算時間（ICCG 法による求解部），Mixed（mod）は前処理部分の単精度化による効果のみを考慮した計算時間（NX=NY=NZ=128）

図 16 は Alg.1 において，倍精度（Double），単精度（Single），混合精度（Mixed）による ICCG 法による求解部の計算時間

である。混合精度演算と倍精度演算の差は前処理部分の計算時間のみであるが，混合精度演算時には倍精度演算の場合と比較して行列ベクトル積の計算時間が増加する傾向が見られたため，これを補正したのが Mixed（mod）である。$\lambda_2=10^{-4}$ までは単精度演算（Single）が最も速いが，$\lambda_2=10^{-5}$ では混合精度演算が最速である。

4. 関連研究

Alg.3（パイプライン型共役勾配法）を提案した，W. Vanroose 等のグループは継続して手法の改善に取り組んでいる。Alg.1～Alg.4 について丸め誤差の挙動に関する詳細な分析を実施している他〔16〕，計算による残差を真の残差に置き換える最適なタイミングを求める手法を提案している〔17〕。また，パイプラインを深くとった Deep Pipelined CG 法（p(ℓ)-CG）〔18〕を提案しているが，前処理付き反復法では深いパイプラインの効果が得られていない。これらの研究は全て倍精度実数演算を対象としており，単精度演算，混合精度演算への適用はまだ行われていない。

Alg.4（Gropp アルゴリズム）についても継続して研究が実施されている〔19〕。

5. まとめ

本研究では，スケーラビリティの高い手法として注目されているパイプライン型アルゴリズムに基づく共役勾配法の悪条件問題への適用時の安定性について検討した。不均質場における三次元定常熱伝導コードにおける求解部（ICCG 法）にオリジナルの CG 法アルゴリズム（Alg.1），Chronopoulos/Gear アルゴリズム（Alg.2）〔3，4〕，パイプライン型共役勾配法（Alg.3）〔4〕及び Gropp アルゴリズム（Alg.4）〔4,12〕を適用した場合の安定性に関する効果を，倍精度，単精度，混合精度演算について評価した。

Alg.2～Alg.4 のいわゆるパイプライン型アルゴリズムに基づく共役勾配法では，オリジナルの CG 法アルゴリズムと比較して，丸め誤差の伝播の仕方が異なり，一般に不安定であることが知られている。単精度演算の適用により，全般的により不安定となることがわかった。しかしながら，前処理部分に単精度演算，他の部分に倍精度演算を適用する混合精度演算導入により，全般的に安定性の改善が見られることがわかった。

今後は Iterative Refinement の導入，単精度演算⇒倍精度演算への反復中での切り替え，真の残差導入等により，単精度演算によるパイプライン型アルゴリズムによる共役勾配法の安定化に継続して取り組んで行く。パイプライン型アルゴリズムによる共役勾配法は，本来，大規模分散並列環境における計算の効率化のために考案されたものである。本研究では，主としてアルゴリズムの安定性に着目し，1 ノードでの検討を中心に実施したが，先行研究〔1〕で実施したような大規模分散並列環境での検討も継続して実施する。

謝辞

本研究の一部は、学際大規模情報基盤共同利用・共同研究拠点、および、革新的ハイパフォーマンス・コンピューティング・インフラ（JHPCN-HPCI)）の支援によるものです（課題番号: jh180023-NAH、課題名：高性能・変動精度・高信頼性数値解析手法とその応用)。

参考文献

1) 塙敏博，中島研吾，大島聡史，星野哲也，伊田明弘，パイプライン型共役勾配法の性能評価，情報処理学会研究報告（2016-HPC-157-6），2016

2) Demmel, J., Hoemmen, M., Marghoob, M., Yelick, K., Avoiding Communication in Sparse Matrix Computations, IEEE Proceedings of 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS 2008), 2008

3) Chronopoulos, A.T., Gear, C.W., *s*-step iterative methods for symmetric linear systems, Journal of Computational and Applied Mathematics 25-2, 153-168, 1989

4) Ghysels, P., Vanroose, W., Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing 40, 224-238, 2014

5) MPI: A Message-Passing Interface Standard Version 3.0, Message Passing Interface Forum, http://mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf, 2012

6) 東京大学情報基盤センター（スーパーコンピューティング部門），http://www.cc.u-tokyo.ac.jp/

7) GeoFEM：並列有限要素法による固体地球シミュレーションプラットフォーム，http://geofem.tokyo.rist.or.jp

8) Washio, T., Maruyama, K., Osoda, T., Shimizu, F., Doi, S., Efficient implementations of block sparse matrix operations on shared memory vector machines. Proceedings of The 4th International Conference on Supercomputing in Nuclear Applications (SNA2000), 2000

9) Nakajima, K., Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator, ACM/IEEE Proceedings of SC2003, 2003

10) 中島研吾，拡張型 Sliced-ELL 行列格納手法に基づくメニィコア向け疎行列ソルバー，情報処理学会研究報告（2014-HPC-147-3），2014

11) Smith, B., Bjϕrstad, P., Gropp, W., *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996

12) Gropp, W., Update on libraries for Blue Waters, http://jointlab.ncsa.illinois.edu/events/workshop3/pdf/presentations/Gropp-Update-on-Libraries.pdf>, 2010

13) Mittal, A., A Survey of Techniques for Approximate Computing, ACM CSUR 48-4, 2016

14) 学際大規模情報基盤共同利用・共同研究拠点 2018 年度共同研究課題（課題番号: jh180023-NAH、課題名：高性能・変動精度・高信頼性数値解析手法とその応用）: https://jhpcn-kyoten.itc.u-tokyo.ac.jp/ja/abstract.php?ID=jh180023-NAH

15) 坂本龍一，近藤正章，中島研吾，藤田航平，市村強，HPC アプリケーションにおける低精度演算の積極的利用による電力効率改善の検討，情報処理学会研究報告（2018-HPC-167-19）， 2018（in press）

16) Cools, S., Yetkin, E.F., Agullo, E., Giraud, L., Vanroose, W., Analysis of rounding error accumulation in CG to improve the maximal attainable accuracy of pipelined CG, Technical report. University of Antwerp Dept. Mathematics and Computer Science and INRIA Bordeaux, 2016
17) Cools, S., Yetkin, E.F., Agullo, E., Giraud, L., Vanroose, W., Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined Conjugate Gradient method, SIAM Journal on Matrix Analysis and Applications (SIMAX), 39 (1), pp. 426-450, 2018
18) Cornelis, J., Cools, S., Vanroose, W., The communication-hiding Conjugate Gradient method with deep pipelines, submitted to SIAM Journal on Scientific Computing (SISC), 2018 (pre-print)
19) Eller, P.R., Gropp, W., Scalable Non-blocking Krylov Solvers for Extreme-scale Computing, ACM Student Research Competition, SC18 (The International Conference for High Performance Computing, Networking, Storage and Analysis), 2018