［大学 ICT 推進協議会　2017 年度年次大会論文集より］

反応・相変化を伴う多分散系混相流シミュレーションコードの最適化

佐々木　大輔 [1)]，加藤　季広 [2)]，磯部　洋子 [2)]，
笠原　弘貴 [3)]，渡部　広吾輝 [3)]，志村　啓 [3)]，奥野　航平 [3)]，
松尾　亜紀子 [3)]，江川　隆輔 [4), 5)]，滝沢　寛之 [4), 5)]，小林　広明 [5), 4)]

1) 東北大学 情報部情報基盤課
2) 日本電気株式会社
3) 慶應義塾大学
4) 東北大学 サイバーサイエンスセンター
5) 東北大学 大学院情報科学研究科

d-sasaki@cc.tohoku.ac.jp

Performance Optimization of Large-scale Simulation of Polydisperse Multiphase Flow with Reaction and Retastability

Daisuke Sasaki[1)], Toshihiro Kato[2)], Yoko Isobe[2)], Hirotaka Kasahara[3)], Hiroaki Watanabe[3)], Kei Shimura[3)], Kohei Okuno[3)], Akiko Matsuo[3)], Ryusuke Egawa[4), 5)], Hiroyuki Takizawa[4), 5)], Hiroaki Kobayashi [5), 4)]

1) Information Infrastructure Division of Information Department, Tohoku Univ.
2) NEC Corporation.
3) Keio University.
4) Cyberscience Center, Tohoku Univ.
5) Graduate School of Information Sciences, Tohoku Univ.

概要

東北大学サイバーサイエンスセンターでは，大規模科学計算システムを最大限活用するために計算科学者と本センターの計算機科学者が連携しながら，プログラムの高速化技法の研究・開発に取り組んでいる．本稿では，慶應義塾大学の研究グループが研究・開発している反応・相変化を伴う多分散系混相流シミュレーションコードのベクトル型スーパーコンピュータ SX-ACE における高速化について紹介する．スーパーコンピュータを利用することで，ワークステーションでの実行よりも圧倒的に大規模なモデルを高速に解析することが可能となり，粉塵爆発現象に対する安全防護技術の発展に貢献することが期待される．

図 1　スーパーコンピュータ SX-ACE

1　はじめに

東北大学サイバーサイエンスセンター(以下，本センター)は，全国共同利用機関として大規模科学計算システムの運用と，本システムを最大限に活用可能なコードの高速化技法や新しいシミュレーション技術の研究・開発に取り組んでいる．本センターでは 1997 年からさまざまな分野における実アプリケーションの高速化支援を計算科学分野の利用者との共同研究を通じて行っている[1]．また，センター独自の共同研究に加え，全国の情報基盤センター等と連携して，学際大規模情報基盤共同利用・共同研究拠点(JHPCN)や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(HPCI)を構成し，多様なニーズに応える計算環境の提供も行っている．以上のように本センターでは，利用者である計算科学者と本センターの計算機科学の専門家が密接に連携しながら，科学・工学の恒常的な進歩を支える共同研究，利用者コードの高速化支援活動を推進している．

本稿では，慶應義塾大学の研究グループが研究・開発している反応相変化を伴う多分散系混相流シミュレーションコードの高速化について紹介する．

2　スーパーコンピュータ SX-ACE

本センターに導入されているスーパーコンピュータ SX-ACE を図 1 に，諸元を表 1 に示す．SX-ACE の CPU はこれまでのベクトルプロセッサと同様に，高ベクトル演算性能と高メモリバンド幅を継承している．また，CPU は 4 コアで構成され最大ベクトル演算性能は 256 GFLOPS である．各コアには容量が 1,024 KB の ADB(Assignable Data Buffer)が搭載され，ADB とコア間のメモリバンド幅は 256GB/sec を有する．データが ADB 経由でアクセスされる場合は，データ転送速度と計算速度の比である B/F 値が 4 Bytes/FLOP となり，メモリ負荷の高いアプリケーションでも高い実行効率で実行可能となっている．また，単一コアで演算を実行する場合には，CPU-メモリ間の 256 GB/sec のメモリバンド幅をそのコアで占有して利用でき，4 Bytes/FLOP での実行が可能となっている．SX-ACE は高ベクトル演算性能と，高データ供給性能により，高い実行性能を達成することができる．

本センターの SX-ACE は 2,560 ノードで構成されている．各ノード間は最大 4 GB/sec×2(双方向)で接続された 2 段ファットツリーネットワークで構成され，システム全体の性能は 706.6 TFLOPS となっている．利用者には，1 ジョブあたり最大 1,024 ノードの大規模な実行環境を提供している．

表 1 SX-ACE の諸元

CPUあたり	コア数	4個
	理論最大演算性能	276GFLOPS
	最大ベクトル演算性能	256GGLOPS
	ADB	1,024KB × 4
	メモリバンド幅(コア-ADB間)	256GB/s × 4
	メモリバンド幅(CPU-メモリ間)	256GB/s
ノードあたり	CPU数	1個
	メモリ容量	64GB
システムあたり	ノード数	2,560ノード
	CPU数	2,560個
	メモリ容量	160TB
	理論最大演算性能	706.6TFLOPS
	最大ベクトル演算性能	655.4TFLOPS

3　反応・相変化を伴う多分散系混相流シミュレーションコード

本コードでシミュレーションを行っている粉塵爆発とは，可燃性の固体粒子が一定条件下で空気中に浮遊している状態で，何らかの現象により発火し爆発する現象のことである．工場等のものつくり現場において，大規模な爆発を伴う火災が発生する危険がある．この爆発を伴う火災は，工場内で発生した大量の粉塵が浮遊・堆積し，静電気などにより発生した火花に引火することが原因で発生する．また，この火災で生じる衝撃波により堆積していた粉塵が舞い上がりさらに大規模な火災に発展する．

慶應義塾大学の研究グループでは，粉塵爆発現象について，粉塵からの可燃性ガスの揮発及び炭素個体の燃焼や気相燃焼，揮発ガス・既燃ガス・空気の混合を考慮しシミュレーションを行っている．また，爆発に伴って生じる衝撃波，爆風による粉塵の巻き上げについて解明するため，シミュレーションコードの開発に取り組んでいる．

4 コードの高速化

4.1 SX-ACEにおける初期性能

慶應義塾大学の研究グループが開発したプログラムの初期性能解析結果を表 2 に示す．性能解析には SX-ACE の簡易性能解析ツール FTRACE を用いた．また，測定データのグリッドサイズは 1,901×161 である．

初期性能解析の結果に基づき，全体の実行時間に対して約 90%を占めているサブルーチン A～C について高速化を行うことを検討する．また，将来の大規模並列実行を見据え，コードの並列化も検討する．

表 2　初期性能解析

サブルーチン名	実行時間	ベクトル化率	ベクトル長
subroutine A	8039.993	91.75	21.9
subroutine B	2314.420	82.60	18.8
subroutine C	1164.311	13.23	12.3
other subroutine:	1850.370	–	–
total	13397.172	88.92	26.2

以下では，本コードに適用した高速化について述べる．

4.2 サブルーチン A，B

表 2 からサブルーチン A, B はベクトル長が非常に短いことが分かる．ベクトルプロセッサは複数のデータに対して一括で演算を行うことができ，最内ループの反復(イテレーション)回数であるベクトル長が長いほど演算の効率が高くなる．SX-ACE ではベクトル長を 256 の時に演算効率が最大になる．

サブルーチン A, B の内部を詳細に解析したところ，両サブルーチンは同様のループ構造となっており，非効率なベクトル化が行われている箇所があった(図 2)．このループの反復回数は 10 であり，ベクトル化されているものの，SX-ACE の理想のベクトル長より非常に短く，ベクトル化の恩恵を受けていない．

```
+-----> do k = lkbgn, lkend
|+-----> do j = ljbgn, ljend
||
||V---> do i1 = 1, lsp
|||
|||
||V---- end do
||
|+----- end do
+------ end do
```

図 2 サブルーチン A, B のオリジナルコード

両サブルーチンの最内ループは 3 重ループの最内ループとなっているため，最内ループの外側のループでベクトル化することで性能の改善が期待できる．今回は図 3 のように unroll 指示行を挿入し，最内ループを展開，最内ループの外側のループでベクトル化することで性能改善を試みる．

```
+-----> do k = lkbgn, lkend
|V----> do j = ljbgn, ljend
||
||       !cdir unroll=10
||*---> do i1 = 1, lsp
|||
||*---- end do
||
|V----- end do
+------ end do
```

図 3 サブルーチン A, B の高速化後のコード

高速化後の性能を表 3 に示す．指示行を挿入することでベクトル長がサブルーチン A で 21.9 から 225.9 に，サブルーチン B で 18.8 から 250.9 まで大きく改善され，サブルーチン A で約 44 倍，サブルーチン B で約 17 倍の性能向上を得た．

表 3　高速化の効果

		実行時間 [sec]	ベクトル化率	ベクトル長	加速率
subroutine A	before	8039.993	91.75	21.9	44.17
	after	182.020	99.18	225.9	
subroutine B	before	2314.420	82.60	18.8	16.96
	after	136.431	99.97	250.9	

4.3　サブルーチン C

表 2 からサブルーチン C はベクトル化率が低く，ベクトル長が短いことが分かる．サブルーチン C の内部を詳細に解析したところ，ループ内に依存関係があり，最内ループがベクトル化されていないことが分かった(図 4)．また，最内ループは外側のループ毎にループ長が異なり，かつそのループ長は短い．外側ループのループ長が長いため，外側ループを最内ループに移動することで性能改善が期待できる．

```
V------>     do np = 1, npmax
|            iend=vox(np)
|+----->       do ilp = 0, iend-1
||
|+-----        end do
|
V------      end do
```

図 4　サブルーチン C のオリジナルコード

そこで，ループの一部を分割し，最内ループのループ長のうち最大になるものを検索する．次に，最内ループの終端を検索した最大ループ長に変更し，ループ間の依存関係を解消する．最大ループ長に満たないループはマスクにより処理を行わないようにしている．依存関係が解消したため，ループ交換を行い，外側のループを最内側ループに移動することでループ長を長く確保し，ベクトル化率を高めた(図 5)．さらに，ベクトルプロセッサに搭載されている ADB を有効に活用できるように最内ループのブロック化を行う． SX-ACE は 1 命令で 256 要素を同時に演算できるため，ブロック長は 256 とした(図 6)．

```
V------>     do np = 1, npmax
|            iend=vox(np)
|            if(iend > maxiend) then
|              maxiend=iend
|            end if
V------      end do
+------>     do ilp = 0, maxiend-1
|V----->       do np = 1, npmax
|||              if(ilp < vox(np)) then
|||                  alx = ax(1,mp(np)) - ax(1,np)
                       :
                       :
|||              end if
|V-----        end do
+------       end do
```

図 5 サブルーチン C のベクトル化後のコード

```
             blksz=256
V------>     do np = 1, npmax
|            iend=vox(np)
|            if(iend > maxiend) then
|              maxiend=iend
|            end if
V------      end do
+------>     do ilp = 0, maxiend-1
|+----->       do np_ = 1, npmax, blksz
||V---->         do np = np_, min(np_+blksz-1,npmax)
|||              j=np-np_+1
|||              iend_(j)=vox(np)
||V----          end do
||V    >          do np = np_, min(np_+blksz 1,npmax)
|||              j=np-np_+1
|||              if(ilp < iend_(j)) then
|||                  alx = ax(1,mp(j)) - ax(1,np)
                       :
                       :
|||              end if
||V----          end do
|+-----        end do
+------       end do
```

図 6　サブルーチン C の高速化後のコード

高速化後の性能を表 4 に示す．高速化によりベクトル化率が 12.23%から 98.91%に改善し，ベクトル長が 12.3 から 151.3 に伸びることで，サブルーチン単体で約 6 倍の性能改善を得ることができた．

表 4　サブルーチン C の高速化の効果

	実行時間 [sec]	ベクトル化率	ベクトル長	加速率
before	1164.311	13.23	12.3	6.11
after	190.607	98.91	151.3	

以上の高速化を適用後のコード全体の性能を表 5 に示す．シングルコア実行で高速化前に比較し約 17 倍の性能改善を得ることができた．

表 5　高速化後の性能

	実行時間 [sec]	加速率
before	13397.172	16.94
after	790.884	

4.4　並列化

本コードはベクトル化を行うことでシングルコアでの性能を高めてきた．さらなる性能の向上のためには，ネットワークに接続される複数のノードを利用し，多数のコアで計算をする並列化を検討する必要がある．並列化にはノード内のコア毎にプロセスを処理する MPI(Message Passing Interface)並列を行う Flat MPI を用いた．

本コードの計算領域は 1,901×161 の 2 次元空間で，この空間中に粒子が存在する．この 2 次元空間中の熱や圧力等の計算後，粒子の相互作用の計算を行う．並列化をするにあたり，計算領域を分割し分割後の各領域に MPI のプロセスを割り当て，各領域の熱や圧力等と存在する粒子に対して相互作用の計算を行うように並列化を行った．

図 7 に並列化後の性能を示す．2 次元分割を行うと計算の初期で粒子が存在しない領域が多数存在し，プロセス間のインバランスが大きくなるため，計算領域を x 方向(1,901)での 1 次元分割とする．高速化後のシングルコア実行に比較し，64 プロセス実行で約 9 倍の性能改善を得るにとどまった．実行後，時間が経過すると，各領域に存在する粒子数に偏りが発生し，プロセス間のインバランスにより，性能向上が抑制されていると考えらえる．今後，さらなる高速化のため，多数の粒子が存在する領域をさらに細かく分割することや，ノード内の並列化に OpenMP を用いた Hybrid MPI などの検討が必要である．

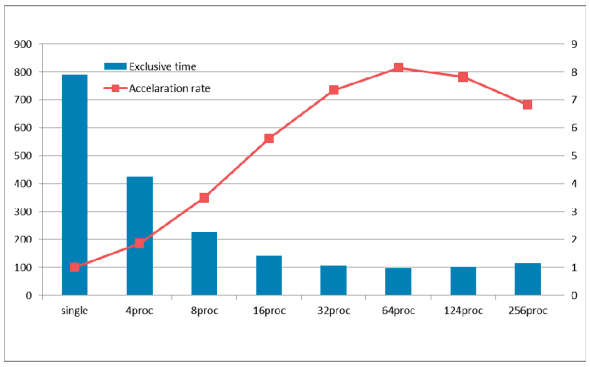

図 7　並列化性能

5　まとめ

本稿では反応・相変化を伴う多分散系混相流シミュレーションコードに，本センターが今までに取り組んできた高速化支援で得られた高速化技法を適用し高速化を行った事例について紹介した．本センターの提供しているスーパーコンピュータ SX-ACE の性能を最大限に引き出すためには，ベクトル化率を高く保つ必要がある．また，SX-ACE は従前のベクトル型スーパーコンピュータより，さらにノードを多数接続することで性能の向上を図る構成となっている．従って，今後の高速化支援においては，シングルコアでの高速化に加え，複数のノードのコアを利用する並列化を行うことで性能を引き出していくことが重要となる．

本コードは，高速化により数時間程度を要するシミュレーションを数分程度で計算することが可能となった．今後，さらなる大規模なモデルでシミュレーションを行うことで安全防護技術の発展に貢献していきたい．

参考文献

[1] 東北大学サイバーサイエンスセンター，高速化推進研究活動報告，第 6 号，p8-12，(2015)．