Vol.2013-HPC-139 No.3
2013/5/29

京速コンピュータ「京」におけるCGPOP Miniappの性能評価

中尾 昌広[1,a)]　佐藤 三久[1,2,3]

概要：京速コンピュータ「京」において，海洋シミュレーションコード POP（Parallel Ocean Program）のミニアプリである CGPOP Miniapp の性能評価を行った．CGPOP Miniapp に対して性能解析を行い，その情報をもとにして京の性能を引き出すためのチューニングを行った．京の Tofu インターコネクトが持つ高機能バリア通信を利用する，三次元ジョブ形状で実行するといった工夫を行うことで，7445 ノードを用いて計測した場合，オリジナルの CGPOP Miniapp と比較して最大 1.15 倍の高速化を達成することができた．

1. はじめに

将来のエクサスケール規模の計算環境では，超高解像度モデルなどの導入によって，より高精度な気候変動予測が可能になると考えられている．そこで，多国間国際研究協力事業 [1] の研究プロジェクトの 1 つ「エクサスケール・コンピューティングによる精緻な気候シミュレーションの実現」では，エクサスケールに向けて各国のスーパーコンピュータを用いた研究が行われている．

本稿では，海洋シミュレーションコード Parallel Ocean Program（以下，POP）のミニアプリである CGPOP Miniapp（以下，CGPOP）[2] の性能評価を京速コンピュータ「京」（以下，京）の上で行い，その結果を示す．また，CGPOP に対して京の性能を引き出すためのチューニングも行う．そして CGPOP の結果を通して，今後のエクサスケールに向けた気候シミュレーションコードおよびプログラミングモデルについて考察する．

本稿の構成は以下の通りである．2 章では，対象アプリケーションである CGPOP について説明する．3 章では，京の性能を引き出すために行った CGPOP に対する最適化方法について述べ，CGPOP の性能を測定する．4 章では 3 章の結果をもとに考察を行う．最後に 5 章では，まとめと今後の課題について述べる．

[1] 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[2] 筑波大学 大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[3] 理化学研究所 計算科学研究機構
RIKEN Advanced Institute for Computational Science
[a)] mnakao@ccs.tsukuba.ac.jp

2. CGPOP Miniapp

まず，POP[3] について説明を行う．POP は Los Alamos National Laboratory が開発した海洋シミュレーションコードである．POP は，全球気候モデルとして広く用いられている Community Earth System Model（以下，CESM）[4] のコンポーネントの 1 つでもある．

CGPOP は POP version 2.0 のミニアプリである．CGPOP は POP と比較して短時間でコンパイルと実行が可能であり，またソースコードも簡潔であるという特徴がある．そのため CGPOP は，POP を新しいマシンへポーティングする際に必要となる性能チューニングを行う，新しい通信機構を導入した場合の性能変化について調べる，などの用途で用いられる．ソースコードの行数は，POP は約 71,000 行であるのに対し，CGPOP は約 3,000 行である．CGPOP は，POP の中で高並列時に性能に大きく影響を与える共役勾配法（Conjugate Gradient method）の箇所を主に抜き出したものである [5]．

POP の記述言語は Fortran90 と MPI ライブラリである．CGPOP の記述言語は，POP と同様に Fortran90 と MPI ライブラリで作成されたバージョンと，性能と生産性の向上のため，袖領域の交換を行うための通信部分を Coarray Fortran[6] で記述されたバージョンが存在する．さらに，データを格納するための配列の次元数や通信と計算の重複の有無などによって分けられた計 7 種類のバージョンが存在する．**表 1** に，CGPOP のそれぞれのバージョンの特徴をまとめたものを示す．

CGPOP には，共役勾配法の演算を行うための通信パターンが 2 種類ある．本稿ではそれぞれを **GlobalSum**,

表 1　CGPOP の種類

Name	袖通信の方法	配列の次元数	通信と計算の重複の有無
mpi1s1D	MPI（1-sided）	1	×
mpi2s1D_overlap	MPI（2-sided）	1	○
mpi2s1D	MPI（2-sided）	1	×
mpi2s2D	MPI（2-sided）	2	×
caf1D	Coarray（1-sided）	1	×
caf1D_sync_images	Coarray（1-sided）	1	○
caf2D	Coarray（1-sided）	2	×

UpdateHalo と表す．**GlobalSum** は 3 要素の実数型変数の集合通信であり，`MPI_Allreduce` を用いて実装されている．**UpdateHalo** は袖領域の交換を行うための通信であり，表 1 のバージョンにより，`MPI_Isend()/Irecv()`，`MPI_Put()/Get()`，Coarray Fortran の PUT/GET 通信のいずれかを用いて実装されている．

CGPOP では，全球を 3600x2400 の矩形に分割して計算を行う．その際，隣接領域の情報が記述された Tile File を入力データとして用いる．Tile File にはブロックサイズ毎に異なるものが用意されており，そのブロックサイズによって推奨並列数は異なる．**表 2** に，ブロックサイズと推奨並列数との関係を示す．例えば，ブロックサイズが 225x150 の場合，3600x2400 の矩形は 16x16 の 256 個の小矩形に分割されるが，その中の陸地だけが存在する 28 個の小矩形は計算の対象外であるため，推奨並列数は 228 となる．

3. 性能評価

3.1 予備評価

本章では，CGPOP の京に対する最適化と性能評価を行った結果について述べる．まず，表 1 の中から性能評価に用いる CGPOP のバージョンを選択するため，京ではサポートされていない Coarray Fortran を利用したバージョン以外の性能について評価した．その結果，mpi2s1D_overlap が最も性能が高かったため，mpi2s1D_overlap に対して最適化を行うことにする．

本評価時の mpi2s1D_overlap の結果を**図 1** に示す．性能比較のために，エジンバラ大のスーパーコンピュータ HECToR（Cray XE6）の結果も同時に示す．京と HECToR のスペックは**表 3** の通りである．コンパイルオプションには，京では “-Kfast” を，HECToR では “-O3” を用いた．京におけるジョブの形状は一次元であり，また後述する最適化のため，1 ノードにつき 1 プロセスで実行している．それに対し，HECToR では，文献 [2] と同様にフラット MPI（1 ノードにつき 32 プロセス）で実行している．また CGPOP では，共役勾配法を行う回数をパラメータとして

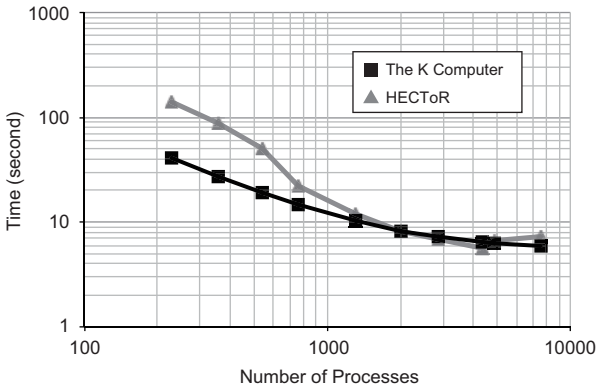

図 1　京と HECToR における CGPOP の性能評価

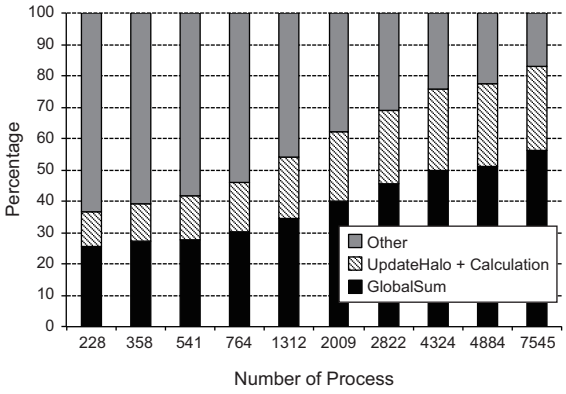

図 2　計算時間の内訳

設定できるため，この値は文献 [2] と同様に 226 回とした．

図 1 の結果から，京と HECToR では最良値はほとんど変わらないことがわかる．京の最良値は 7545 プロセス（7545 ノード）時の 5.88 秒であり，HECToR の最良値は 4324 プロセス（136 ノード）時の 5.63 秒であった．

3.2 予備評価の考察

図 1 における計算時間の内訳を**図 2** に示す．**UpdateHalo** については，通信と計算のオーバラップが行われているため，図 2 中の該当の項目名は「UpdateHalo+Calculation」としている．図 2 中の項目名の「Other」は，主に CPU による計算時間である．

図 1 から，高並列時は **GlobalSum** が全体の時間を大きく占めることがわかる．そのため，本章では **GlobalSum** の最適化を中心に行う．

3.3 GlobalSum の最適化

3.3.1 高機能バリア通信の利用

2 章で述べた通り，**GlobalSum** は 3 要素の実数型変数の `MPI_Allreduce` 通信である．京では `MPI_Allreduce` などの集合通信を行う際，Tofu インターコネクトのハードウェア機能として提供される高機能バリア通信を利用することで，その高速化を行うことができる [7]．しかしながら，高機能バリア通信を利用するためには，複数の制約条

表 2 ブロックサイズと推奨並列数

ブロックサイズ	225x150	180x120	144x96	120x80	90x60	72x48	60 x40	48x32	45x30	36x24
推奨並列数	228	358	541	764	1312	2009	2822	4324	4884	7545

表 3 計算環境

	The K computer	HECToR
CPU	SPARC64 VIIIfx 2.0GHz, 8Cores/Socket	AMD Opteron Interlagos 2.3GHz x 2, 16Cores/Socket
Memory	DDR3 SDRAM 16GB, 64GB/s/Socket	DDR3 SDRAM 32GB, 42.7GB/s/Socket
Network	Torus fusion 6D mesh/torus network, 5GB/s	Gemini interconnect 3D torus network, 8GB/s
Fortran Compier	Fujitsu Fortran Compiler Version K-1.2.0-11	Cray Fortran Compiler Version 8.1.4
Comm. Library	Fujitsu MPI Version K-1.2.0-11	Cray-mpich2 Version 5.6.1

```
vlen = 3
call MPI_ALLREDUCE(local_vector, global_sum_vector_dbl_time, vlen, ...)
```



```
vlen = 3
do i= 1, vlen
  call MPI_ALLREDUCE(local_vector(i), global_sum_vector_dbl_time(i), 1, ...)
end do
```

図 3 コード変更

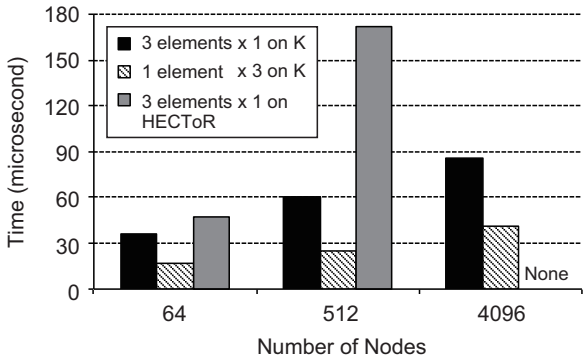

図 4 転送要素数が異なる場合の MPI_Allreduce に要する時間

件を満たす必要がある．その制約条件の中で，CGPOP と関係している条件は下記の 2 つである．

- 各ノードにおけるバリアゲートの利用数を超えないこと
- 転送データは 1 要素であること

前者の条件については，例えば 1 ノードに複数のプロセスを実行する場合，並列数によっては問題が生じる．そのため，今回の実験では，1 ノードにつき 1 プロセスのみを実行することにした．後者の条件については，3 要素を 1 回の命令で転送していた箇所を，**図 3** のように 1 要素ずつ転送するようにコード変更を行うことで対応した．

ここで，高機能バリア通信の利用が CGPOP における MPI_Allreduce の通信時間に与える影響について調べる．オリジナルの CGPOP のように 3 要素のデータを 1 回で転送する場合と，図 3 のように 1 要素のデータを 3 回に分けて転送する場合との性能比較を行った．転送されるデータの型は CGPOP と同様に，実数型（8 バイト）である．**図 4** に結果を示す．比較のために，HECToR の結果も示している．京と HECToR の両方において，1 ノードにつき 1 プロセスで実行している．HECToR には京のような集合通信のためのハードウェアサポート機構は存在しないため，3 要素のデータを 1 回で転送する場合のみを示している．また，HECToR は実行ノード数の制限のため 4096 ノードの結果はない．

図 4 から，1 要素のデータを 3 回に分けて転送した場合の方が，3 要素のデータを 1 回で転送した場合よりも 2.18〜2.40 倍高速であることがわかった．さらに HECToR と比較すると，京の方が 64 ノードの場合は 2.85 倍，512 ノードの場合は 6.86 倍高速であることがわかった．

3.3.2 ジョブ形状の最適化

京では，ジョブ形状をユーザが指定することができる．指定できる形状は，一次元，二次元，三次元のいずれかであり，それぞれ仮想的なトーラストポロジを構成することが可能である．京における MPI_Allreduce は，Reduce + Bcast アルゴリズムを使用したパイプライン転送によって実装されているため [8]，ホップ数が最も少なくなる三次元の形状が最も性能が高いと考えられる．

ここで，ジョブの形状が MPI_Allreduce に与える影響について調べる．各ジョブの形状において，1 要素のデータを 3 回に分けて転送を行い，その時間の測定を行った．結果を**図 5** に示す．図 5 より，ジョブの形状の次元数が増えるほど，性能が向上することがわかる．一次元ジョブ形状と三次元ジョブ形状の結果を比較すると，64 ノードを用いた場合は 1.12 倍，4096 ノードを用いた場合は 1.34 倍高速化することがわかった．

3.3.3 再評価

3.3.1 節と 3.3.2 節で説明した高速化手法を用いて，mpi2s1D_overlap の再評価を行う．再評価の方法として，図 1 において最も性能が高かった 7545 ノードを利用した場合において，それぞれの最適化手法を適用させる．ここでジョブの形状については，三次元形状，かつ各次元の大きさは近い値の方が良いと考えられる．しかしながら，7545=3x5x503 であるため，そのままのノード数で三次元形状を構成すると 1 つの次元が突出して大きい値になってしまう．そこで，7600=19x20x20 の形状でノードを確保

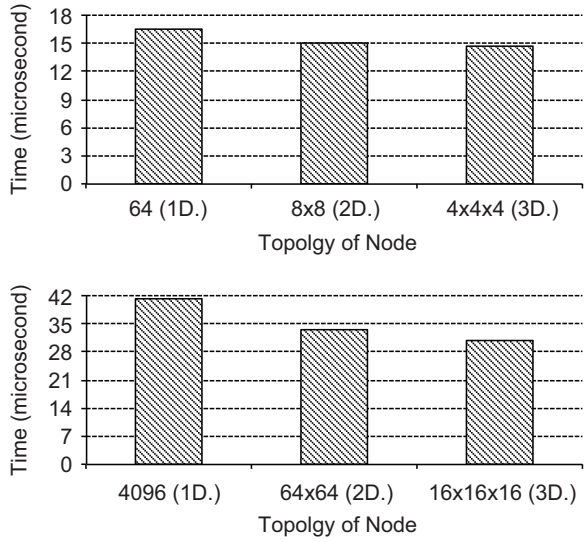

図 5 各ジョブ形状における MPI_Allreduce に要する時間（上グラフは 64 ノード利用時．下グラフは 4096 ノード利用時の結果）

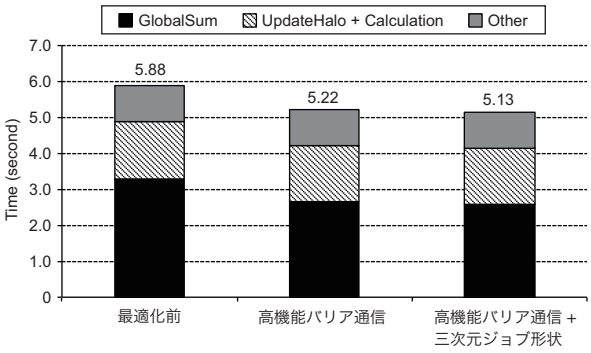

図 6 CGPOP の再評価

し，その形状の中に 7545 プロセスを割り当てることにした．再評価の結果および実行時間の内訳を**図 6** に示す．

図 6 から，高機能バリア通信と三次元ジョブ形状を組合せた場合が最も性能が高く，オリジナルの場合と比較して 1.15 倍高速化することがわかった．なお，**GlobalSum** のみに関すると，高機能バリア通信を適用した場合は 1.24 倍，高機能バリア通信と三次元ジョブ形状を組合せた場合は 1.27 倍の高速化であった．3.3.1 節と 3.3.2 節で示した結果ほど **GlobalSum** の高速化が実現できなかった理由は，各プロセスの計算時間は不均等であり，かつ各結果の **GlobalSum** の時間には MPI_Allreduce を行うための通信待ち時間も含まれるからであると考えられる．

3.4 スレッド並列化

mpi2s1D_overlap 中のループ文のスレッド並列可能な箇所に対して OpenMP を用いたスレッド並列を行った．各ノード 1 プロセス 8 スレッドで実行した場合とスレッド化を行わない場合（図 1 の京の結果と同じ）との結果を**図 7** と**図 8** に示す．図 8 は，図 7 のそれぞれの結果の比（スレッド並列化を行わない場合の実行時間/スレッド並列化

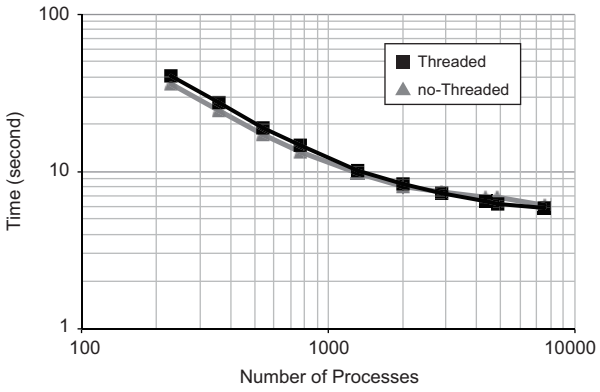

図 7 スレッド並列化の結果

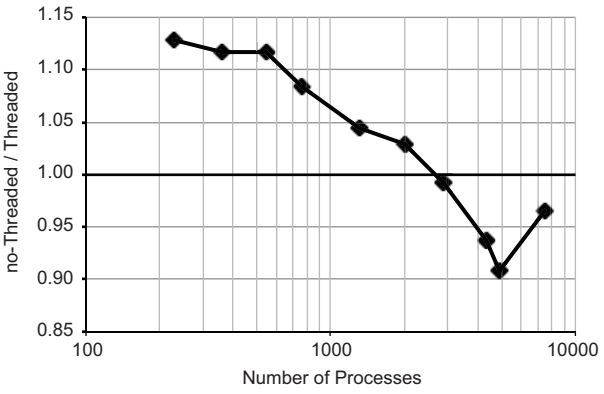

図 8 スレッド並列化による性能向上比

を行った場合の実行時間）を表したものであり，縦軸の値が 1.00 以上の場合はスレッド化を行った方が性能が高いことを示している．

図 7 の結果から，スレッド化を行った場合と行わない場合の性能はほぼ同じであることがわかる．また図 8 の結果から，高並列時はスレッド並列を行った方が性能が低くなる場合があることがわかる．それらの理由は，mpi2s1D_overlap にはスレッド化可能な箇所が少なく，かつ強スケーリング問題であるため，高並列時は低並列時と比較して，計算に要する時間が短く，スレッド生成コストの方が高くなってしまったからであると考えられる．

4. 考察

CGPOP では，共役勾配法を用いているため，内積計算のために MPI_Allreduce の操作が必要になる．共役勾配法を用いることで，Δt の値を大きく設定できるため，総計算時間が小さくなることが期待できる．しかしながら，京のように MPI_Allreduce に対するハードウェアサポートがあったとしても，エクサスケール計算環境ではよりノード数が増えることが予測されるため，MPI_Allreduce のような全体通信は性能のボトルネックになると考えられる．共役勾配法の代わりに，例えば隣接通信のみで計算を行える陽解法のアプローチを試みる必要があると考えられる．

また，CGPOP の袖通信は通信先を指定するローカル

ビューのアプローチで記述されている．よりコードを簡潔に記述するには，通信先を指定せずに袖通信が行えるグローバルビューのアプローチが有効であると考えられる．しかし，例えば並列言語 XcalableMP[9], [10] はグローバルビューの袖通信に対応しているが，CGPOP のように隣接ノード同士の関係が規則的でない場合の袖通信には対応していない．XcalableMP を CGPOP に対応させるためには，隣接ノードをユーザが柔軟に指定できるように拡張を行う必要がある．

5. まとめと今後の課題

本稿では，京における CGPOP の性能チューニングおよび性能評価を行った．性能チューニングについては，CGPOP の中で高並列時に最も時間割合が大きい **GlobalSum** の最適化を主に行った．京の Tofu インターコネクトが持つ高機能バリア機構を利用できるようにコード変更を行い，さらに三次元ジョブ形状で実行を行うことにより，7445 ノードを用いた場合，オリジナルの CGPOP と比較して最大 1.15 倍の高速化を達成した．

今後の課題としては，下記のことが挙げられる．

- **GlobalSum** の次に時間割合の大きい袖通信 **UpdateHalo** の最適化を行う．具体的には，表 1 中の Coarray を用いて袖通信を行っているバージョンを，Coarray Fortran の上位互換である XcalableMP を用いて動作させることを考えている．現在，XcalableMP は京が持つ RDMA エンジンを用いた高速片側通信を行えるように拡張作業が進められている．この機能を用いることで, 表 1 中の片側通信の MPI ライブラリを用いた場合よりも XcalableMP によって Coarray を用いた場合の方が高速に実行できることが期待できる．
- 京で用いられている CPU である SPARC64 VIIIfx に対する最適化，例えばセクタキャッシュの利用，レジスタ数を考慮したループ分割などを行う．これらの最適化を行うことで，各プロセスの計算時間を少なくすることができ，さらに **GlobalSum** および **UpdateHalo** 時の通信待ち時間も減少することができると考えられる．
- 本稿で評価を行った CGPOP はミニアプリであるため，エクサスケール計算環境に向けたより深い知見を得るには，実アプリケーションである POP，または CESM のスケーラビリティを京の上で評価する必要がある．

謝辞 本研究は，日本学術振興会・多国間国際研究協力事業「エクサスケール・コンピューティングによる精緻な気候シミュレーションの実現」の支援によって行われました．

参考文献

[1] 日本学術振興会多国間国際研究協力事業：事業概要. http://www.jsps.go.jp/j-bottom/01_b_gaiyo.html.
[2] Stone, A. I., Dennis, J. M. and Strout, M. M.: Evaluating Coarray Fortran with the CGPOP Miniapp, *Proceedings of the Fifth Conference on Partitioned Global Address Space Programming Models (PGAS)* (2011).
[3] Jones, P.: Parallel Ocean Program (POP) user guide, Technical Report Technical Report LACC 99-18, Los Alamos National Laboratory (2003).
[4] CESM: Community Earth System Model. http://www.cesm.ucar.edu.
[5] Stone, A., Dennis, J. and Strout, M. M.: The CGPOP Miniapp, Version 1.0, Technical Report Technical Report CS-11-103, Colorado State University (2011).
[6] Numwich, R. and Reid, J.: Co-Array Fortran for parallel programming, Technical report ral-tr-1998-060, Rutherford Appleton Laboratory (1998).
[7] Fujitsu: *Parallelnavi for MP10 V1.0* (2013).
[8] 松本 幸, 安達知也, 住元真司, 南里豪志, 曽我武史, 宇野篤也, 黒川原佳, 庄司文由, 横川三津夫：MPI_Allreduce の「京」上での実装と評価，情報処理学会論文誌. コンピューティングシステム， Vol. 5, No. 5, pp. 152–162 (2012).
[9] Nakao, M., Lee, J., Boku, T. and Sato, M.: Productivity and Performance of Global-View Programming with XcalableMP PGAS Language, The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (2012).
[10] XcalableMP Specification Working Group: XcalableMP Specification Version 1.1 (2012). http://www.xcalablemp.org/spec/xmp-spec-1.1.pdf.