T2K オープンスパコン（東大）チューニング連載講座番外編：Hybrid 並列プログラミングモデルの評価（Ⅱ）

中 島 研 吾

東京大学情報基盤センター

1. はじめに，ハードウェア諸元

前回〔1〕は，ノード（またはソケット）内に OpenMP，ノード（またはソケット）間に MPI を適用したいわゆる「Hybrid」並列プログラミングについて，

「有限要素法アプリケーションから得られる疎行列を，前処理付反復法で解く」

場合に，T2K（東大）1 ノード（16 コア）を使用した事例について紹介した．今回は，複数ノードを使用した事例について紹介する．利用した計算機は以下の 2 種類である：

- T2K オープンスパコン（東大）（Hitachi HA8000 クラスタシステム）（東京大学情報基盤センター）[1]
- Cray XT4（Franklin システム，アメリカ国立ローレンスバークレイ研究所）[2]

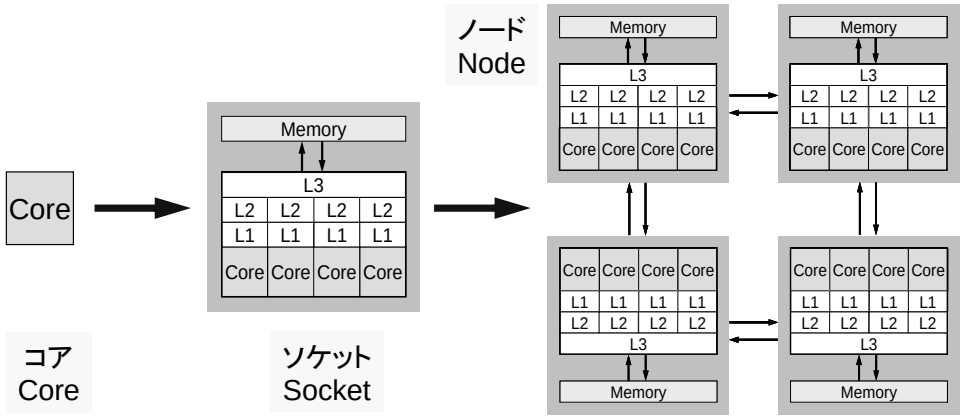


図1　T2Kオープンスパコン（東大）の各ノードの構成

T2K オープンスパコン（東大）（以下 T2K（東大））は〔1〕でも述べたように，各ノード上に 4 コアを有する AMD Opteron （2.3GHz）を 4 ソケット搭載している（合計 16 コア）（図 1 参照）． Cray XT4（Franklin）の各ノードは AMD Quad-Core Opteron（2.3GHz）1 ソケット 4 コアから構成され，T2K（東大）の 1 ソケット同様である．

T2K（東大）は全体で 952 ノード，15,232 コアから構成され，ピーク性能は 140.1 TFLOPS であるが，全体は 512，256，128，56 ノードから構成されるクラスタ群に分割されている．Cray XT4（Franklin）は 9,572 ノード，38,288 コアから構成され，ピーク性能は 352.2 TFLOPS であ

[1] http://www.cc.u-tokyo.ac.jp/service/intro/
[2] http://www.nersc.gov/nusers/systems/franklin/

る．

T2K（東大）の各ノードは Myrinet-10G インターコネクトで接続されている．Myrinet-10G は 1 リンク 1 方向あたり 1.25 GB/sec のバンド幅を持ち，双方向同時通信が可能である．前述の 4 つのクラスタ群のうち，合計 640 ノード（512＋128）は「タイプ A」と呼ばれ，1 ノードあたり 4 本の Myrinet-10G を搭載しているおり，ノード間通信バンド幅は 5.00 GB/sec，残り 312 ノード（256＋56）は「タイプ B」 と呼ばれ，2 本の Myrinet-10G を搭載しており，ノード間通信バンド幅は 2.50 GB/sec である．各クラスタ内では Myrinet-10G は「フルバイセクションバンド幅」が確保される方法で接続されており，クラスタ内の任意の半数のノードが同時に残り半分のノードにデータを送信してもネットワーク内での競合が発生しない．本稿では「タイプ A」のクラスタを使用している．

Cray XT4（Franklin）は Cray SeaStar2 に基づく 3D トラス構造によって各ノードが結合されており，各ノードの通信バンド幅は 45.6 GB/sec（6 方向に 7.6 GB/sec）である．

表 1 に T2K（東大），Cray XT4（Franklin）の諸元を示す．

表 1　T2K（東大），Cray XT4（Franklin）ハードウェア諸元

	T2K（東大）	Cray XT4（Franklin）
設置場所	東京大学情報基盤センター	ローレンスバークレイ国立研究所
コア数／ノード	16	4
総ノード数	952	9,572
総コア数	15,232	38,288
ピーク性能（TFLOPS）	140.1	352.2
ノード間インターコネクト	Myrinet-10G，多段スイッチ	Cray SeaStar2（3D トラス）
ノードあたり通信バンド幅（GB/sec）	5.00×双方向（タイプ A） 2.50×双方向（タイプ B）	45.6（＝7.60×6 方向）
通信レイテンシ（μsec）	2.00（Myrinet-10G 1 リンク）	5.00～6.00

2. アプリケーションの概要

本稿では前回に引き続いて，GeoFEM プロジェクト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群〔2〕のうちの三次元弾性静解析問題（Cube 型モデル）（図 2）を使用した．連立一次方程式の係数マトリクスの格納法として（a）CRS（Compressed Row Storage），(b) DJDS（Descending order Jagged Diagonal Storage）の 2 種類の方法が準備されているが，本稿ではスカラープロセッサ向けの CRS 法を使用した．

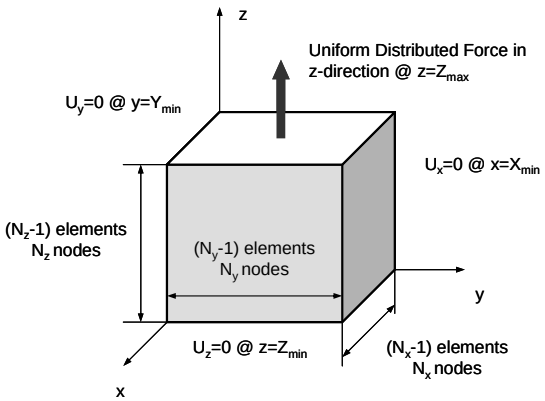


図 2　Cube 型ベンチマークの境界条件

係数行列が対称正定な疎行列となることから，SGS（Symmetric Gauss-Seidel）〔2〕を前処理手法とし共役勾配法（Conjugate Gradient，CG）法によって連立一次方程式を解いている（以下

SGS／CG 法と呼ぶ）．SGS 前処理では，係数行列 A そのものが前処理行列として利用されるため ILU 分解は実施しない．三次元弾性問題では 1 節点あたり 3 つの自由度があるため，これらを 1 つのブロックとして取り扱っている．

3. 問題設定等

本稿では，図 2 で一辺の節点数が 144 の立方体の場合（総要素数：2,924,207，総節点数：2,985,984，総自由度数：8,957,952）について，全体問題サイズを固定し，コア数を 32 コアから 1,024 コアまで変化させる Strong Scaling について計算を実施した．前回同様：

- **Flat MPI**
- **Hybrid 4×4（HB 4×4）**： スレッド数 4 の MPI プロセスを 4 つ起動，T2K（東大）（図 1）の場合，各ソケットに OpenMP スレッド×4，ノード当たり 4 つの MPI プロセス
- **Hybrid 8×2（HB 8×2）**：スレッド数 8 の MPI プロセスを 2 つ起動，T2K（東大）（図 1）の場合, 2 ソケットに OpenMP スレッド×8, ノード当たり 2 つの MPI プロセス（Cray XT4（Franklin）については実施せず）
- **Hybrid 16×1（HB 16×1）**：1 ノード全体に 16 の OpenMP スレッド，1 ノード当たりの MPI プロセスは 1 つ（Cray XT4（Franklin）については実施せず）

の 4 種類の並列プログラミングモデルを適用した〔1〕．コンパイラとしては，T2K（東大）では日立コンパイラ（オプション：-Oss，Cray XT4（Franklin）で PGI はコンパイラ（オプション：-O3）を使用した．

SGS 前処理（前進後退代入時）に生じるデータ依存性を排除するためのデータの並び替え（reordering，リオーダリング）を導入した．レベルセットによる並べ替え法（level set reordering method）である Reverse Cuthill-McKee （RCM）法によって並び替えを施された節点に対して，更にサイクリックに再番号付けする Cyclic マルチカラー法（cyclic multicoloring，CM）を適用する手法（CM-RCM）を使用した．本稿では特に CM-RCM（色数 10）を用いた．

計算実施にあたっては，前回の結果に基づき，1 ノード（16 コア）の場合に最も高い性能を示した以下の組み合わせを適用した〔1〕：

- 最適な NUMA Policy の適用
- 図 3 に示すように同じスレッドで処理するデータを，メモリ上でなるべく連続に配置するように並び替える（Flat MPI は除く）
- 更に First Touch Data Placement を適用する（Flat MPI は除く）

表 2 は，各並列プログラミングモデルにおいて使用した NUMA Policy である．1 ノード実行時において最適な性能を示した NUMA Policy を適用した．

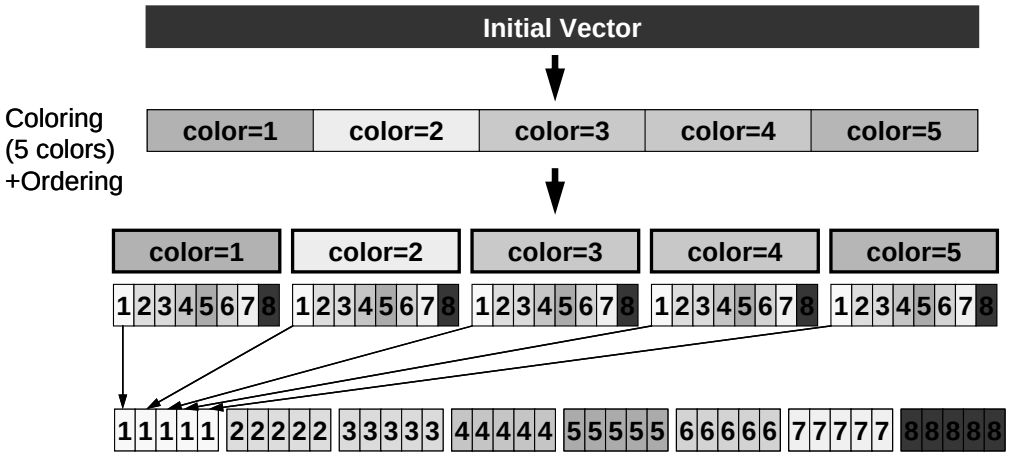


図 3　連続データアクセスのためのデータ再配置（5 色，8 スレッドの場合）

表 2　適用した NUMA Policy

Programming Model	Policy ID	Command line switches
Flat MPI	2	`--cpunodebind=$SOCKET --interleave=$SOCKET`
Hybrid　4×4	4	`--cpunodebind=$SOCKET --localalloc`
Hybrid　8×2	4	`--cpunodebind=$SOCKET --localalloc`
Hybrid　16×1	5	`--localalloc`

4. 計算結果：初期ケース

図 4 に T2K（東京），Cray XT4（Franklin）を 1,024 コアまで使用した場合の SGS／CG ソルバーの GFLOPS 値に基づく計算性能を示す．図 4（a）は GFLOPS 値の絶対値，図 4（b）は，T2K（東大），Cray XT4（Franklin）それぞれの Flat MPI 32 コアの場合の性能を 32.0 とした場合の性能上昇である．いずれの計算機でも HB 4×4 の性能が最も高く，図 4（b）によると 32⇒1,024 コアで，31.3⇒1,017（T2K（東大））， 31.9⇒1,110（Cray XT4（Franklin））， となり Flat MPI の 1,024 コアにおける相対性能（882：T2K（東大），888：Cray XT4（Franklin）を上回っている．それに対して，T2K（東大）における HB 8×2，HB 16×1 の性能上昇は比較的低く，32⇒1,024 コアでは，30.9⇒899（HB 8×2），30.7⇒837（HB 16×1）となっており，HB16×1 の性能は Flat MPI よりもむしろ低くなる．

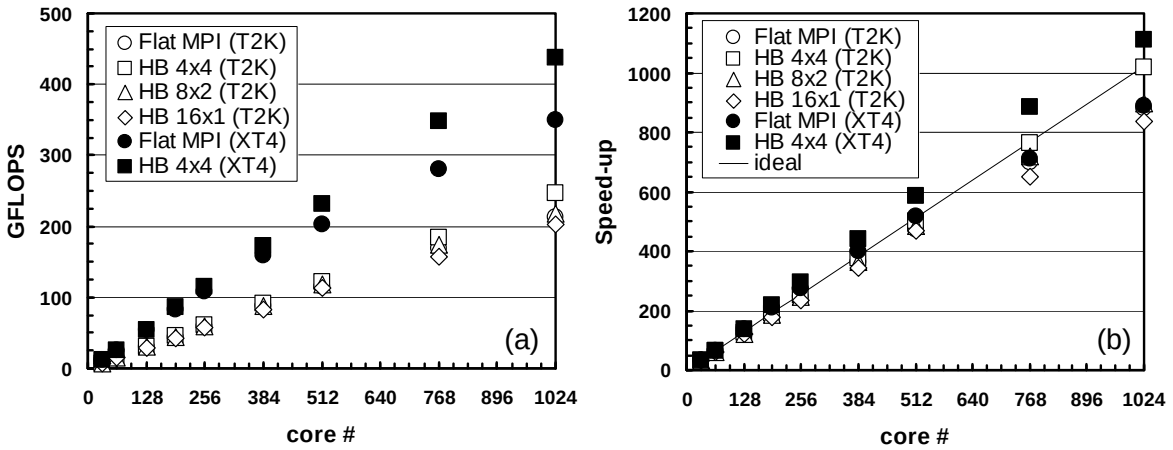


図 4　三次元弾性問題における SGS／CG ソルバーの計算性能（総自由度数:8,957,952, 32～1,024 コア使用），（a）GFLOPS 値　（b）Flat MPI 32 コアの性能を 32.0 とした場合の性能上昇

図 5 は Flat MPI, HB 4×4 について T2K（東大）と Cray XT4（Franklin）の性能を比較したものである．16 コアの場合と同じく，Cray XT4（Franklin）は T2K（東大）と比較して, 1.50 倍から 2.00 倍程度性能が高い〔1〕．コア数が 32 から 128 程度になると両者の差が大きくなり，相対的に Cray XT4（Franklin）の性能が高くなる．Strong Scaling の問題の場合はコア数（＝領域数）が増加すると，1 コアあたりの問題サイズが小さくなるため，よりキャッシュを有効に利用できるようになる．T2K（東大）は cc-NUMA アーキテクチャであるため各ソケットのキャッシュの Coherency をとる必要があるが，Cray XT4（Franklin）ではその必要がないため，1 コアあたりの問題サイズが少なくなったときにキャッシュをより有効に活用でき，性能増加がより顕著である（図 6 参照〔1〕）．

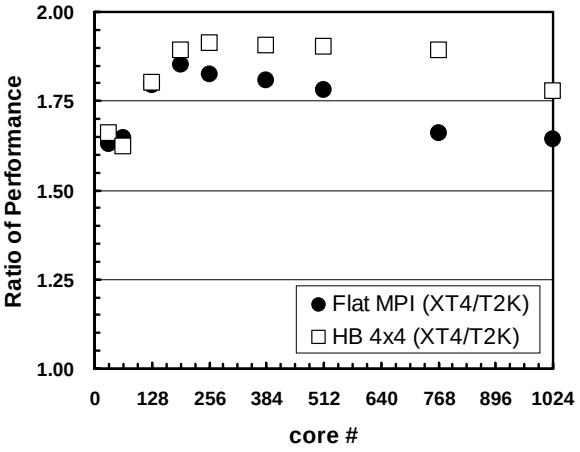


図 5　T2K（東大）に対する Cray XT4（Franklin）の性能比

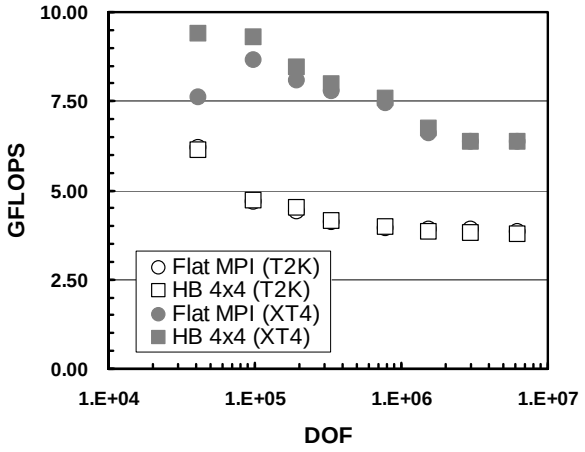


図 6　16 コアを使用した場合の T2K（東大）と Cray XT4（Franklin）における問題サイズと性能の関係〔1〕

疎行列を対象とした並列反復法においては, 通信レイテンシ（communication latency, 通信立ち上がりのオーバーヘッド）の影響がクリティカルである．この影響は MPI プロセスが増加するほど深刻となるため, ペタ／エクサスケールのシステムでは Hybrid 並列プログラミングモデルの導入によって，MPI プロセス数の爆発的な増加を少しでも抑制することが重要である. また疎行列を対象とした並列反復法は Memory-bound なプロセスである. 従って，メモリに負担をかけないように，できるだけ各コアあたりの問題規模を小さく抑えて，多くのコアを使って計算することが得策である．このような場合，〔1〕でも述べたように Hybrid 並列プログラミングモデルが有利となる可能性があり, 1,024 コアを利用した場合の計算結果によれば，HB 4×4 は T2K（東大），Cray XT4（Franklin）ともに 20%以上 Flat MPI より性能が高く，HB 8×2 ではわずかに Flat MPI より性能が高い．コア数を増加させた場合の，HB 4×4 の性能上昇が Cray XT4（Franklin）においてより顕著であるのは，図 5，図 6 に示した例と同じで，Cray XT4（Franklin）では各ソケットキャッシュの Coherency をとる必要がないため，1 コアあたりの問題サイズが少なくなったときにキャッシュをより有効に活用でき，性能増加がより顕著であるためである．

5. 計算結果：HB 8×2，16×1 性能改善

4. に示したように，Strong Scaling 問題において，ノード数を増加させた場合，疎行列ソルバーのような Memory-bound なプロセスでは，Flat MPI と比較して，Hybrid 並列プログラミングモデルの性能が高く，特に HB 4×4 でその効果が顕著であった．一方，T2K（東大）におい

ては，HB 8×2，16×1 ではその効果は顕著ではなく，特に HB 16×1 では 1,024 コア使用時において Flat MPI よりも性能が低いことがわかった. Hybrid 並列プログラミングの利点の一つは，利用コア数が増加した場合に MPI のプロセス数を減らすことが可能になることであり，HB 8×2，16×1 のようにノード内のスレッド数の多い場合の性能を高めることが重要である．本節では特に HB 8×2，16×1 の性能改善に注目する．

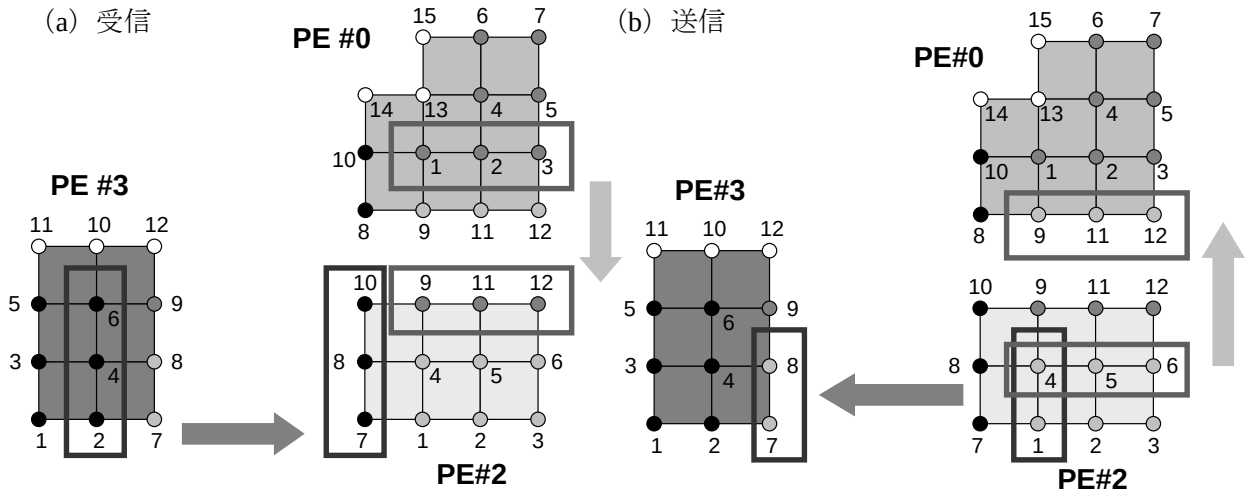


図 7　並列有限要素法による領域間送受信〔2〕

本稿で対象としている並列有限要素法のようなアプリケーションでは，図 7 に示すように領域間で通信が発生する〔2〕．FORTRAN，OpenMP，MPI によるハイブリッド並列プログラミングモデルを適用する場合，送信部分は図 8 に示すようになる（三次元弾性問題（1 節点に 3 自由度がある））〔2〕．境界点（Boundary Nodes）〔2〕における変数ベクトル（X）の値を送信バッファ（WS）に代入（メモリーコピー）し送信を行っている．受信時には逆に受信バッファから変数ベクトルへのメモリーコピーが必要となる．

```
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
!$omp parallel do private (ii)
        do k= istart+1, istart+inum
              ii   = 3*NOD_EXPORT(k)
          WS(3*k-2)= X(ii-2)
          WS(3*k-1)= X(ii-1)
          WS(3*k  )= X(ii  )
        enddo
!$omp end parallel do

        call MPI_Isend (WS(3*istart+1), 3*inum,MPI_DOUBLE_PRECISION,    &
     &                  NEIBPE(neib), 0, SOLVER_COMM, req1(neib), ierr)
      enddo
```

図 8　領域間通信：送信部分〔2〕

図 8 の例では，変数ベクトルの値を送信バッファに格納する部分について，OpenMP によって並列化しているが，この代入操作は領域あたりの自由度数を n とすると $n^{2/3}$ のオーダーの演算である．本稿で対象としている Strong Scaling 問題では，コア数，すなわち領域数が増加するにつれてループ長が短くなるため，OpenMP のオーバーヘッドの影響が大きくなる．この影響は当然，MPI プロセス数あたりの OpenMP スレッド数が大きい HB 8×2，16×1 の場合に顕著となることが推定される．そこで図 9 に示すように，領域間送受信プロセスにおける送受信バッファと変数ベクトルのメモリーコピー実施を 1 スレッドで実施することにより，OpenMP オー

バーヘッドの影響を除去することを試みた．

結果を図 10 に示す．図 10（a）は図 4（b）に相当するが，T2K（東大）の HB 8×2，16×1 を図 9 に基づき OpenMP ディレクティヴを削除した場合の結果に差し替えてある．図 10（b）はディレクティヴ削除前後の各コア数における HB 4×4，8×2，16×1 の性能比である．コア数が 512 以下ではほとんど影響が無いが，1,024 コアでは HB 8×2，16×1（T2K（東大））の性能は 10%程度改善している（図 10（b））．32⇒1,024 コアでの性能上昇は，30.9⇒997（HB 8×2），30.7⇒919（HB 16×1）となっており，図 4（b）に示した Flat MPI の性能上昇（32.0⇒882）を上回っている．HB 4×4 の場合は T2K（東大），Cray XT4（Franklin）ともに OpenMP ディレクティヴ省略による性能変化は無いため，図 8，図 9 のいずれを使用しても良いことになる．従って Hybrid 並列プログラミングモデルを使用する場合，領域間通信では図 9 に示したように送信・受信バッファへの代入計算は並列化せず，単一スレッドで実施した方が効果的である．

```
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart

        do k= istart+1, istart+inum
              ii   = 3*NOD_EXPORT(k)
          WS(3*k-2)= X(ii-2)
          WS(3*k-1)= X(ii-1)
          WS(3*k  )= X(ii  )
        enddo

        call MPI_Isend (WS(3*istart+1), 3*inum,MPI_DOUBLE_PRECISION,    &
     &                  NEIBPE(neib), 0, SOLVER_COMM, req1(neib), ierr)
      enddo
```

図 9　領域間通信：送信部分，OpenMP ディレクティヴ省略

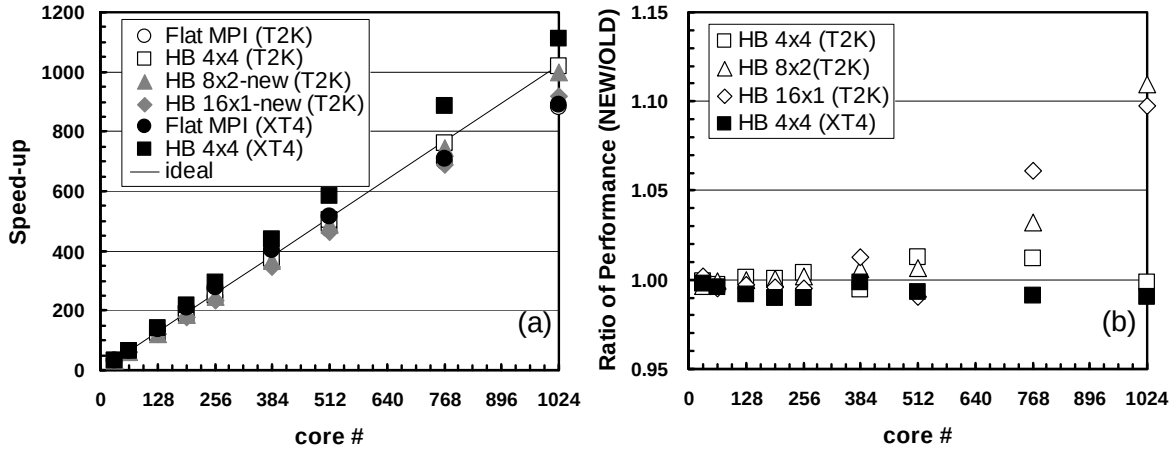


図 10　三次元弾性問題における SGS／CG ソルバーの計算性能（総自由度数：8,957,952，32～1,024 コア使用），（a）Flat MPI 32 コアの性能を 32.0 とした場合の性能上昇，（b）OpenMP ディレクティヴ省略による領域間通信性能改善前後の性能比（値が 1.0 より大きいと，OpenMP ディレクティヴ省略によって演算速度が向上していることを示す，HB 8×2，16×1 で効果高い）

6. 計算結果：Myrinet-10G リンク数の変更

1. で示したように，T2K（東大）の各ノード間は Myrinet-10G（リンク当たり通信バンド幅:1.25 GB/sec）で結合されており，本稿で使用されている「タイプ A」のクラスタでは，Myrinet-10G の 4 本のリンクから構成されている.「タイプ A」のクラスタではデフォルトでは 4 本のリンクを全て利用する（ノード間ピーク通信バンド幅 : 5.00 GB/sec）．また，環境変数「MX_BONDING」（デフォルト値＝4）を指定することにより，実行時にノード間通信で使用するリンク数を変更することができ，例えば「MX_BONDING=1」とすると，1 本のリンクしか使用しない．また，MX_BONDING の指定に関わらず，ノード間通信量が 32KB を下回る場合のみ，自動的に 1 本のリンクのみ利用するように切り替わるような設定となっている（図 11 参照）.

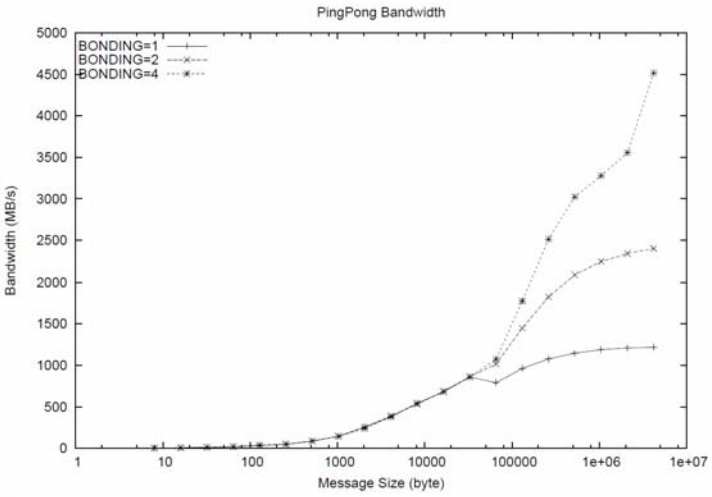


図 11　T2K（東大）ノード間実効通信バンド幅と Myrinet-10G リンク数（BONDING）の関係
メッセージサイズが 32KB より小さい場合は
自動的に BONDING=1 となる
（松葉浩也助教（東京大学情報基盤センター）による測定）

図 12 は，T2K（東大）において，これまでのデフォルト設定（MX_BONDING=4）と「MX_BONDING=1」の場合の性能比較である．なお，HB 4×4，8×2，16×1 の場合は図 9 に示した OpenMP ディレクティヴ削除を実施している．HB 8×2，16×1 の場合は MX_BONDING の値による性能の影響はほとんど無く, HB 16×1 では 1,024 コアの場合には若干低下している．それに対して, Flat MPI, HB 4×4 では性能が増加し, 特に Flat MPI では 1,024 コアの場合には，10%程度の上昇が見られ（図 12（b））, 32⇒1,024 コアでの性能上昇は，32.0⇒965 となり，MX_BONDING=4 の場合の性能上昇（32.0⇒882）を上回っており, コア数が増加した場合には，ノード間通信バンド幅が低い設定の方がむしろ性能が改善されていることがわかる.

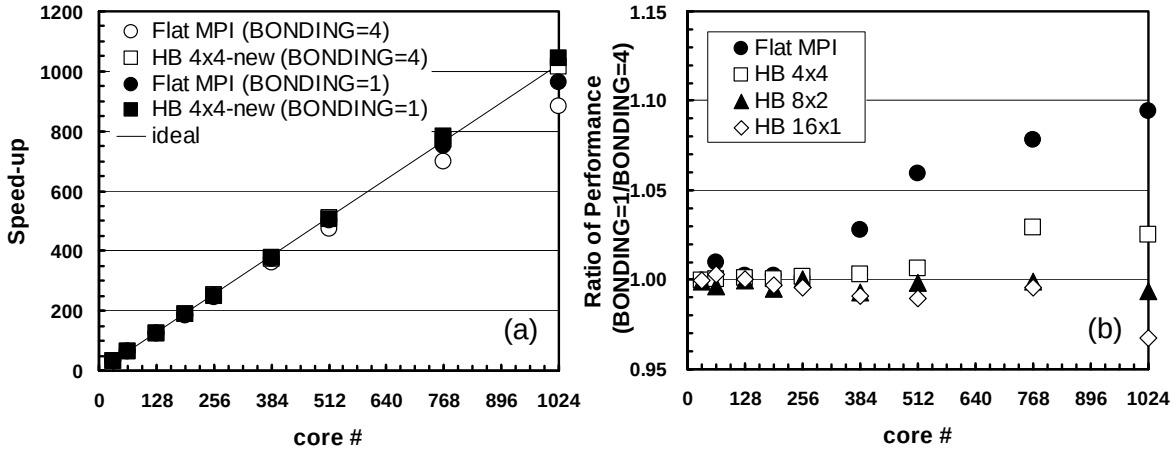


図 12　三次元弾性問題における SGS／CG ソルバーの計算性能（総自由度数：8,957,952，32～1,024 コア使用）（T2K（東大））,（a）Flat MPI 32 コア（MX_BONDING=4，デフォルト値）の性能を 32.0 とした場合の性能上昇,（b）MX_BONDING=1 の場合の MX_BONDING=4 の場合に対する性能上昇（値が 1.0 より大きいと，MX_BONDING=1 によって演算速度が向上していることを示す，HB 4×4，8×2，16×1 では図 9 に示した OpenMP ディレクティヴ削除を実施）

7. MPI ベンチマーク

（1）概要

6.で示したように，T2K（東大）において Strong Scaling 問題でコア数を増加させた場合，Flat MPI 並列プログラミングモデルを使用すると，MX_BONDING=1 とした場合の方が MX_BONDING=4 の場合よりも性能が改善される可能性のあることがわかった．

その効果について定量的な検討を行うため，通信性能測定のための MPI ベンチマークを実施した．並列疎行列ソルバーに現れる通信を含むプロセスをモデル化した，以下の 2 つのベンチマークを作成し，T2K（東大）の 32～512 コアを使用して評価を実施した：

- Allreduce
- 領域境界 1 対 1 通信

疎行列反復法のプロセスは：

- 内積
- DAXPY（ベクトルの定数倍の加減）
- 行列ベクトル積
- 前処理

から構成される．並列計算時に DAXPY では通信は発生せず．また，本稿では SGS 前処理を使用しているため，領域間通信は発生しない〔2〕．通信が発生するのは内積，行列ベクトル積である．上記の 2 つのベンチマークはそれぞれ，①Allreduce：内積，②領域間 1 対 1 通信：行列ベクトル積，に対応している．

各ベンチマークは，①Default（MX_BONDING 指定なし），②MX_BONDING=4，③MX_BONDING=1 の 3 ケースについてそれぞれ実施した．

（2）Allreduce

このベンチマークテストは，疎行列反復法における内積演算で使用される MPI_Allreduce の性能を測定するベンチマークであり，図 13 に示すように，倍精度実数の合計を求める計算を各ループ内で 5 回実施している．T2K（東大）の 2～32 ノード，32～512 コアを使用して，32～512 プロセスの Flat MPI に基づき評価を実施した．

```
      do iter= 1, ITERMAX

        WS0= 1.d0
        call MPI_Allreduce (WS0, WS, 1, MPI_DOUBLE_PRECISION,          &
     &                      MPI_SUM, MPI_COMM_WORLD, ierr)
        call MPI_Allreduce (WS0, WS, 1, MPI_DOUBLE_PRECISION,          &
     &                      MPI_SUM, MPI_COMM_WORLD, ierr)
        call MPI_Allreduce (WS0, WS, 1, MPI_DOUBLE_PRECISION,          &
     &                      MPI_SUM, MPI_COMM_WORLD, ierr)
        call MPI_Allreduce (WS0, WS, 1, MPI_DOUBLE_PRECISION,          &
     &                      MPI_SUM, MPI_COMM_WORLD, ierr)
        call MPI_Allreduce (WS0, WS, 1, MPI_DOUBLE_PRECISION,          &
     &                      MPI_SUM, MPI_COMM_WORLD, ierr)
      enddo
```

図 13　MPI ベンチマーク（Allreduce）の実行内容

ITREMAX=5,000 として計算時間を測定し平均をとって, 1 回の MPI_Allreduce の計算時間を算出した結果を図 14 に示す. MX_BONDING=4 と default の場合の差異は無く, いずれも MX_BONDING=1 の場合と比較して時間を要している. コア数が増加すると，全体的に計算時間は増加し，かつ MX_BONDING=1 と他の 2 ケースの差は広がる傾向にあり，512 コアの場合で約 2 倍となっている．本ベンチマークでは，メッセージとしては 8 Byte の倍精度実数を一個送っているだけなので, 通信レイテンシ(立ち上がりのオーバーヘッド）を測定しているのと同等である.

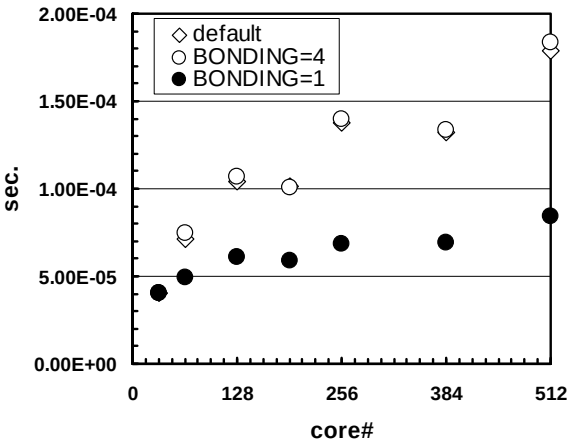
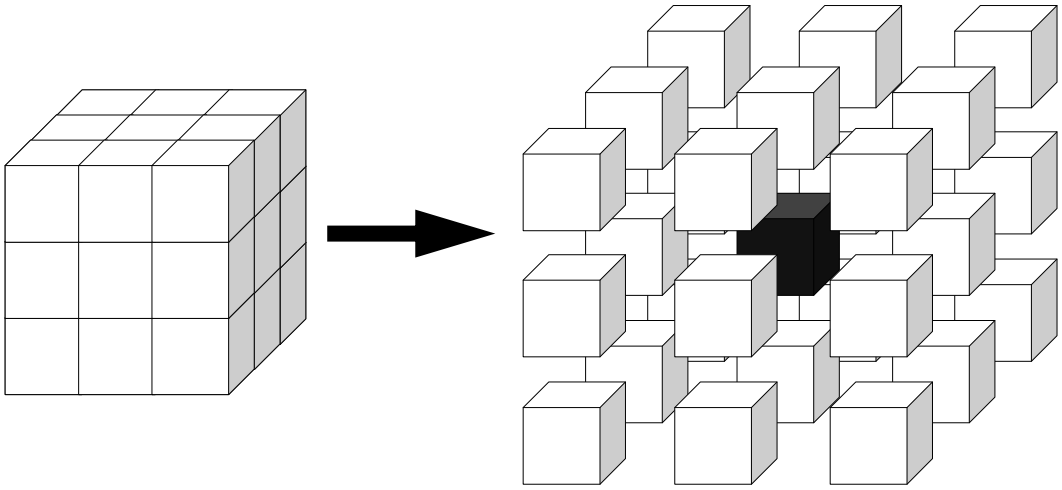


図 14　MPI ベンチマーク（Allreduce）結果（計算時間），T2K（東大）2～32 ノード（32～512 コア）使用，Flat MPI，MX_BONDING の影響

（3）領域境界 1 対 1 通信

このベンチマークテストは，疎行列反復法の行列・ベクトル積の計算において領域境界で発生する図 7 のような 1 対 1 通信をモデル化したものである. 本ベンチマークでは図 15 に示すような立方体の集合において，各立方体と隣接する立方体間で大きさ $3\times N^2$ の倍精度実数のメッセージの送信・受信を実施し，その時間を測定するものである．隣接する立方体の数は最大 26 であるが（図 15（b）参照），立方体の位置によって，7，11，17，26 と異なっている.



（a）立方体集合　　（b）立方体領域に隣接する立方体（最大 26 個）

図 15　MPI ベンチマーク（領域境界 1 対 1 通信）計算モデルの概要

MPI ベンチマーク（領域境界 1 対 1 通信）は図 16 に示すように，MPI_Isend，MPI_Irecv を使用した非ブロッキング通信に基づいている．N= 8，16，32，48 の場合ついて，T2K（東大）の 2～32 ノード，32～512 コアを使用して，32～512 プロセスの Flat MPI に基づき評価を実施した．隣接領域数が 26 個の場合のプロセスあたりのメッセージ送信量は，それぞれ約 40KB（N=8），160KB（N=16），640KB（N=32），1.44MB（N=48）である．各 5,000 回（図 16 の ITERmax=5,000）の計算を実施して，一回の平均計算時間によって評価した.

```
      do ic  = 1, CASEtot
        N= 3*LENGTH(ic)*LENGTH(ic)

      do iter= 1, ITERMAX

      do neib= 1, NEIBPEtot
        call MPI_Isend (WS(N*(neib-1)+1), NN, MPI_DOUBLE_PRECISION,     &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req1(neib),    &
     &                  ierr)
      enddo

      do neib= 1, NEIBPEtot
        call MPI_Irecv (WR(N*(neib-1)+1), NN, MPI_DOUBLE_PRECISION,     &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD, req2(neib),    &
     &                  ierr)
      enddo

      call MPI_Waitall (NEIBPETOT, req2, sta2, ierr)
      call MPI_Waitall (NEIBPETOT, req1, sta1, ierr)

      enddo

      enddo
```

図 16　MPI ベンチマーク（領域境界 1 対 1 通信）の実行内容

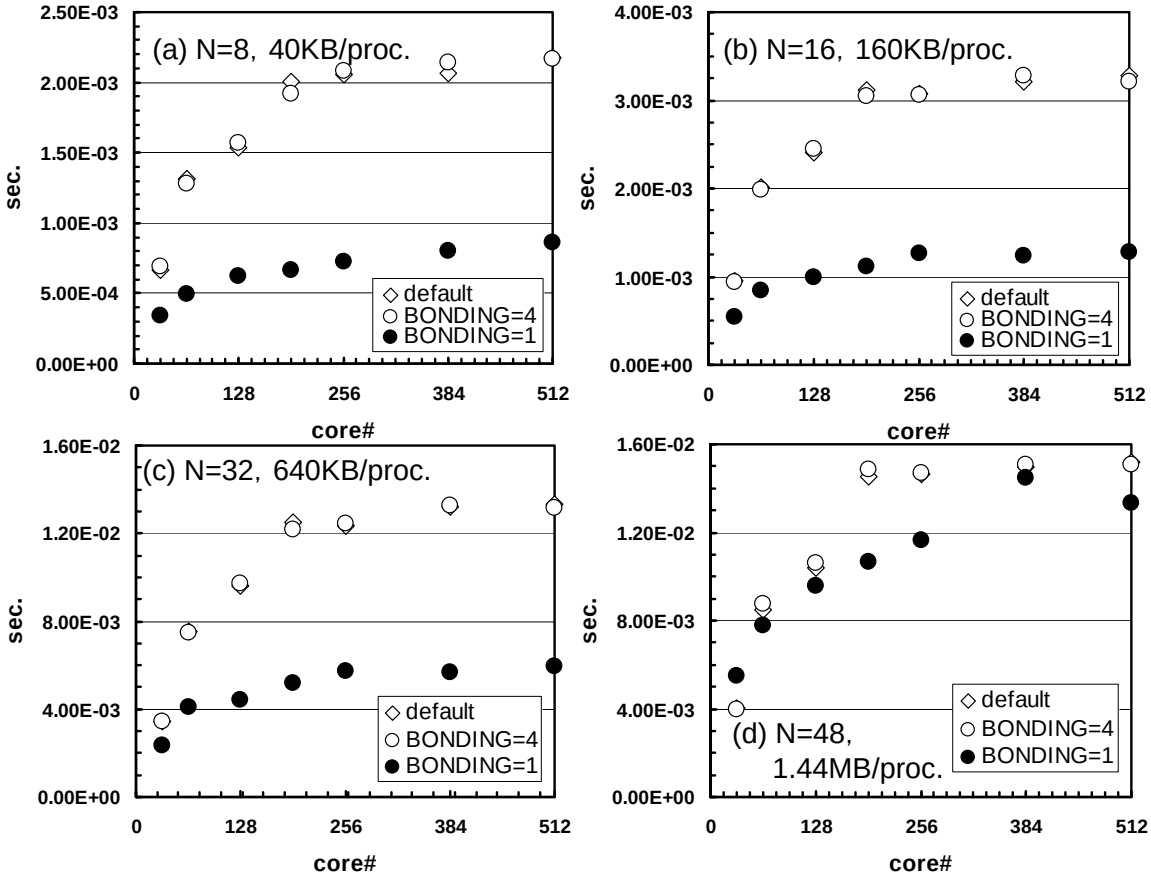


図 17　MPI ベンチマーク（領域境界 1 対 1 通信）結果（計算時間），T2K（東大）2～32 ノード（32～512 コア）使用，Flat MPI，MX_BONDING の影響

図 17 に計測結果を示す．MX_BONDING=4 と default の場合の差異は無く，いずれも MX_BONDING=1 の場合と比較して時間を要している．コア数が増加すると，計算時間が増加し，MX_BONDING=1 と他の 2 ケースの差は広がる傾向にあるが，256 コア以上では各ケースとも計算時間はほぼ一定である．N=48 の場合は，MX_BONDING=1 の場合と他の場合の計算時間は，ほぼ同様になる．

（4）MX_BONDING の効果

2 種類のベンチマーク結果より，以下のような知見が得られる：

- 集合通信，1 対 1 通信ともに，MX_BONDING=1 の場合の方が，コア数が多い場合の通信の立ち上がりのオーバーヘッドが少ない．
- 1 対 1 通信では，通信メッセージサイズが小さい場合には，前項で述べたコア数が多い場合の通信の立ち上がりのオーバーヘッドが支配的であるため，MX_BONDING=1 の方が性能が良いが，メッセージサイズが増えるにつれて MX_BONDING=4（デフォルト）の優位性が増す．
- ベンチマークは Flat MPI を対象としたものであるが，Hybrid 並列プログラミングモデルでは，同じコア数を利用する場合でも，MPI プロセス数が減少して通信の立ち上がりのオーバーヘッドの影響がより少なくなり，プロセス間のメッセージサイズも大きくなる．従って，Flat MPI の場合と比較すると MX_BONDING=4 がより優位である（図 12（b）参照）．

8. まとめ

2 回にわたって，T2K（東大）において：

「有限要素法アプリケーションから得られる疎行列を，前処理付反復法で解く」

アプリケーションについて，1 ノード（16 コア），複数ノードにおける事例を紹介し，Hitachi SR11000/J2，Cray XT4（Franklin）との比較を実施し，T2K（東大）の利用にあたって，以下のような知見を得ることができた：

① コア数が多く，コア当たりの問題規模が小さい場合には Hybrid 並列プログラミングモデル，特に各ソケットに MPI プロセスを割り当てる HB 4×4 が有効である．
② Hybrid 並列プログラミングモデルでは，各スレッドにおけるメモリアクセスが連続となるような，並び替えが有効である．
③ Hybrid 並列プログラミングモデルでは，領域間通信のための送信，受信バッファ代入のループは OpenMP による並列化によって，却って性能が低下する場合もあるため，並列化せず 1 スレッドで実行する方が安全である．
④ 特に Flat MPI を利用する場合，コア数が多く，コア当たりの問題規模が小さく，通信レイテンシの影響が顕著と考えられる場合には MX_BONDING=1 とした方がデフォルト設定（MX_BONDING=4 または 2）よりも性能が高い場合がある．
⑤ 以上は特定のアプリケーションの特定の条件下での結果から得られた知見である．利用者の個々のアプリケーションに適用される場合には，充分注意されたい．ベンチマーク等によって事前に挙動を把握しておくことをお勧めする．

筆者はこれまで，主として T2K（東大）上で Hybrid 並列プログラミングモデルを最適化することを目的とした研究を実施してきたが，各コアに MPI プロセスを発生させる「Flat MPI」は，T2K（東大）上では実は「Flat」ではない．ソケット内，ノード内，ノード間の三階層の通信の

特性を考慮した，データ配置，最適化などは，今後検討するべき課題であることを付記しておく．

謝　辞

本稿の執筆にあたって，有用な助言とデータを提供してくれた松葉浩也助教（東京大学情報基盤センター）に深く感謝する．

参　考　文　献

[1] 中島研吾（2009）T2Kオープンスパコン（東大）チューニング連載講座番外編：Hybrid並列プログラミングモデルの評価（Ⅰ），スーパーコンピューティングニュース（東京大学情報基盤センター）11-4
http://www.cc.u-tokyo.ac.jp/publication/news/VOL11/No4/200907tuning.pdf

[2] Nakajima, K.（2003） Parallel Iterative Solvers of GeoFEM with Selective Blocking Pre-conditioning for Nonlinear Contact Problems on the Earth Simulator. ACM/IEEE Proceedings of SC2003

付録：MX_BONDINGの指定方法

実行時に環境変数MX_BONDINGの値を変えるには，例えば以下のようにバッチ処理スクリプトに記入すればよい．この行を省略すると，「タイプA」のノードでは「MX_BONDING＝4（ノード間ピーク通信性能：5.00 GB/sec)」，「タイプB」のノードでは「MX_BONDING＝2（同：2.50 GB/sec)」となる．

```
#@$-r test
#@$-q parallel
#@$-N 64
#@$-J T4
#@$-e err
#@$-o test.lst
#@$-lM 28GB
#@$-lE 01:00:00
#@$-s /bin/sh
#@$

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=4
export MX_BONDING=1
mpirun ./a.out
```