Vol.2016-HPC-153 No.34
2016/3/3

スーパーコンピュータ「京」における MPI 通信性能の評価

北澤好人[†1]　黒田明義[†1]　南一生[†1]　庄司文由[†1]

スーパーコンピュータ「京」は 82,944 ノードから構成される超並列のシステムであり，アプリケーションを用いて高い実効性能を得るためには，MPI 通信関数の性能が極めて重要である．本稿では，Intel MPI Benchmark を使ったスーパーコンピュータ「京」における主要な MPI 通信のベンチマーク結果を報告する．「京」のネットワークトポロジを考慮して，1 次元，2 次元，3 次元のノード割当における測定を行い，性能を比較した．その結果，集団通信の殆どの関数では，3 次元のノード割当が優位であることがわかった．特に，メッセージサイズが大きい場合に，「京」の Tofu インターコネクトに最適化されたアルゴリズムが適用されることで，高い通信性能が得られた．更に，作業領域やノード形状の設定を変更することで，通信性能が高くなる場合があることがわかった．

1. はじめに

スーパーコンピュータ「京」（以下，「京」と記す）では，生命科学・医療・創薬，新物質・エネルギー創成，防災・減災，次世代ものづくり，物質・宇宙の起源をはじめとするの様々な分野のアプリケーションプログラムが実行され，成果を上げている[1]．実行されるアプリケーション・プログラムの並列規模は数千～数万であり，実行時間に対する通信処理の時間が占める割合が高くなるケースも多い．

本稿では，Intel MPI Benchmark を使ったスーパーコンピュータ「京」における主要な MPI 通信のベンチマーク結果を報告する．2 章では「京」の通信の概要を説明し，3 章では測定の内容と測定結果及び評価した内容を報告する．4 章では測定した結果の展開例として，「京」の運用における活用について報告し，5 章で全体をまとめる．

2. 「京」のネットワークの概要

本章では，「京」のノード間ネットワークの概要を説明する．

2.1 Tofu インターコネクトの概要

「京」では，ノード間の通信ネットワークを構成するために，Tofu(Torus fusion)インターコネクト[2][3]を採用し，システム全体で 82,944 ノードの構成の通信を実現している．各ノードには，データ転送のためのインターコネクト用の通信 LSI である ICC(Inter-Connect Controller)が装備されている．図 1 に ICC の構成を示す．ICC には，4 つの Tofu ネットワークインタフェース(TNI)と Tofu ネットワークルータ(TNR)が装備され，TNI により最大で 4 方向の同時通信が可能である．TNR は 10 本のリンクを持つ．10 本のリンクのうち，4 本が Tofu の基本単位である 12 ノード（Tofu 単位）内の 3 次元メッシュ/トーラスネットワークを担い，6 本が Tofu 単位間の 3 次元メッシュ/トーラスネットワークを担う．

2 つの 3 次元メッシュ/トーラスネットワークにより，6 次元メッシュ/トーラスネットワークを構成する．

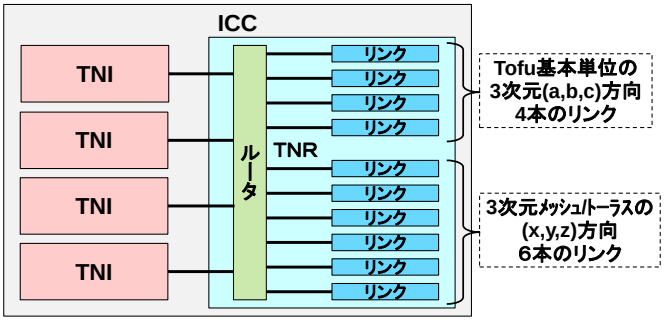


図 1　ICC

2.2 通信性能

各リンクの理論上の通信バンド幅は 5[GiB/s](双方向)である[2]．実際には MPI のオーバーヘッドがあるため，通信バンド幅の実効値は約 4.5[GiB/s]である．ただし，同時通信の場合はハードウェアの内部のバスの性能の制限により，1 ノードあたりの総通信バンド幅は最大で約 15[GiB/s]に律速される．

2.3 6 次元ネットワーク

6 次元メッシュ/トーラスネットワークにおける物理的な座標軸を(x,y,z,a,b,c)と表記する[3]．a,b,c の 3 次元は，Tofu 単位である 2x3x2 の 12 ノードで構成され，筐体内の近い位置に配置される．一方，x,y,z の 3 次元は複数の筐体にまたがって構成される．「京」では，ジョブスケジューラの負荷を抑えるために，ノード割当てを Tofu 単位で実施している．この 3 次元＋3 次元の組み合わせにより，故障ノードを回避した通信や，ジョブ単位で 3 次元トーラスネットワークを構成することが可能となり[3]，高い可用性と多様な運用ニーズに対応できる柔軟性を実現している．ユーザは，ジョブを投入する際に，ノードを割り当てるための次元を指定することができる．ここではそれをノード形状と呼ぶ．ノード形状は 1 次元，2 次元，3 次元の種類があり，それぞれジョブ単位でトーラスネットワークを構成することが可能である．例えば，ノード形状を 3 次元に指定した場合は

†1 理化学研究所　RIKEN

6 次元の((xa), (yb), (zc))の組み合わせにより実現している．

「京」の MPI 環境は OpenMPI をベースに開発されたが，MPI の集団通信関数については，OpenMPI 由来の通信アルゴリズムの他に，Tofu インターコネクト向けに最適化した通信アルゴリズム（Tofu 専用アルゴリズム）を開発し，高い通信性能を実現している．例えば，Allreduce 通信に対して，3 次元トーラスを前提として，メッセージを 3 分割して 3 方向の同時通信により高い性能を実現するアルゴリズム Trinaryx3 を採用した[4][5]． OpenMPI のアルゴリズムと Tofu 専用アルゴリズムは, 通信時のメッセージサイズに応じて，最適な方が自動的に選択される．

3. MPI 通信の測定結果

3.1 測定内容

本節では，測定の内容として測定環境，測定プログラム，測定ケース，測定条件を説明する．

「京」では，コンパイラ，ライブラリ，ランタイム等をまとめて言語環境と呼び，言語環境全体としてバージョン管理している. 今回測定に用いた言語環境のバージョンは，K-1.2.0-16-3 である．

測定に用いたベンチマークプログラムは，IMB(Intel MPI Benchmarks)[6]の Ver.3.2.4 である．測定する通信関数としては，実際のアプリケーションで良く使われる MPI-1 規格の 16 種類の通信関数を選択した. 一対一通信から PingPong, PingPing, Sendrecv, Exchange を選択し，集団通信から Bcast, Allreduce, Reduce, Reduce_scatter, Allgather, Allgatherv,Gather, Gatherv, Alltoall, Alltoallv, Scatter, Scatterv を選択した．

測定においては,「京」で推奨されているハイブリッド並列を想定し，１ノードに 1 MPI プロセスを割り当て（ノード内はスレッド並列), 並列数とノード数が同一となるように設定した．並列数の規模として，小規模(384 ノード)・中規模(3072 ノード)・大規模(18432 ノード)の３種類と，ノード形状として，１次元・２次元・３次元の３種類の，計９ケースを測定した．測定に用いた９ケースの並列数およびノード形状を表 1 に示す. 表の中の":strict"はノードを割り当てる際に，システム側で回転することを許容しない指定である．これを指定しないと，実行の度に同じノード形状が割当てられるとは限らない．ここでは，ノード割当てが変わることにより、通信時間が変動する可能性を避ける意味で指定している．

表 1　測定ケース

ノード割当	１次元(1D)	２次元(2D)	３次元(3D)
小規模	384	64x6	8x6x8:strict
中規模	3072	48x64	16x6x32:strict
大規模	18432	128x144	48x12x32:strict

実行時に指定するオプションは次の通りである．IMB のオプションとして，メッセージサイズを 2^n (n=2～28)[Byte] となる 0～256[MiB]を指定し，広範囲のメッセージサイズの結果を取得した．IMB における使用メモリ量として 6[GiB]を設定し, Alltoall 通信や Gather 系の通信において測定可能なメッセージサイズの範囲を可能な限り大きく取るようにした．MPI の実行時オプションとして，集団通信の作業領域拡大のためのオプションの coll_tuned_prealloc_size には，デフォルト (6[MiB]) と 600[MiB]の２種類を設定して，それぞれを測定した．通信種類によってメッセージサイズの２倍の作業領域を使用することがあり, 最大の 256[MiB]の２倍以上の 600[MiB]を設定した．また，実行時に設定するラージページのサイズを最大の 256[MiB]とした．

通信性能を評価するための指標としては，メッセージサイズを通信時間で除算したスループットを用いた. 一般に，メッセージサイズが小さい場合はレイテンシやパイプライン転送の立ち上がり時間等の割合が大きくなるため，理論上の転送性能からはずれるが，メッセージサイズが大きい場合は理論値に近い転送性能を示すケースが多い．スループットの算出方法について, 一対一通信では IMB の出力値を使い，集団通信ではメッセージサイズを通信時間で割ることで算出した．

3.2 一対一通信の性能結果の特徴

本節では，測定した一対一通信の結果を元に通信性能の傾向と特徴を示す．

IMB の一対一通信には，単一転送と並列転送がある[6]. 単一転送を代表して PingPong，並列転送を代表して Sendrecv の性能を説明する．

(1) **PingPong**

PingPong は，2 つのランク間において，一方のランクからデータを Send 通信により送信し，もう一方のランクにて受信した後に Recv 通信によりデータを元のランクへ送り返すことで，データを往復させる[6]. PingPong は隣接通信でかつ単一転送のため通信経路の競合が無く，結果は割り当てノード形状に依存しない．スループットの結果を図 2 に示す．メッセージサイズが 256[MiB]の場合に約 4.5[GiB/s]に到達した．

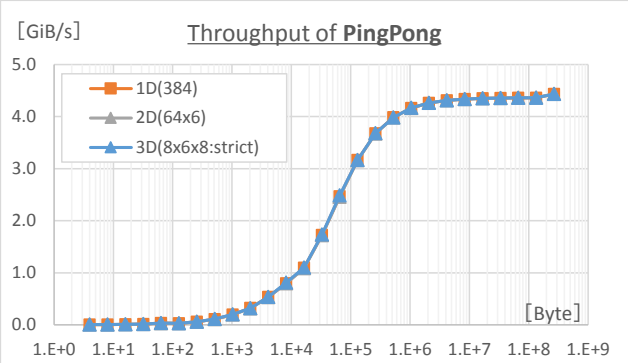


図 2　PingPong のスループット

(2) **Sendrecv**

Sendrecv では，下位のランクから上位のランクにデータを送信する．これを全てのランクが同時に実行するため，各ランクではデータの送信と受信が同時に発生する[6]．隣接通信が支配的であるため，結果はノード数に依存しない．スループットの結果を図 3 に示す．ノード形状が 1 次元の場合，メッセージサイズが 256[MiB]の場合に約 9.0[GiB/s]に到達した．

割り当てノード形状が 2 次元及び 3 次元の場合，1 次元の場合よりスループットが 1/2 程度と低くなった．性能が低下する原因は，Sendrecv におけるデータ転送が，ランクの 1 次元方向への転送のため，ノード形状の境界において前後のランクのノードが隣接にならず斜めの方向の通信となる場合があり，通信の衝突により通信性能の低下が発生したためである．通信の衝突が発生する例を図 4 に示す．トーラス形状の場合，ノード形状の境界となるノード③から④への通信が斜めの方向となり，ノード⑦を経由する．一方，ノード⑦から⑧への通信が④のノードを経由するため，同じ経路（⑦→④）を通り通信の衝突が発生する．また，トーラス形状の場合は斜めの方向の通信が 2 ホップとなるが，トーラスにならないメッシュ形状の場合はホップ数が増え，更に性能が低下する．尚，ステンシル計算等の隣接ノード間のみの通信のように斜めの方向の通信が無く，衝突が起きないケースにおいては，ノード形状が 2 次元及び 3 次元の場合も 1 次元と同様のスループットとなると見込まれる．

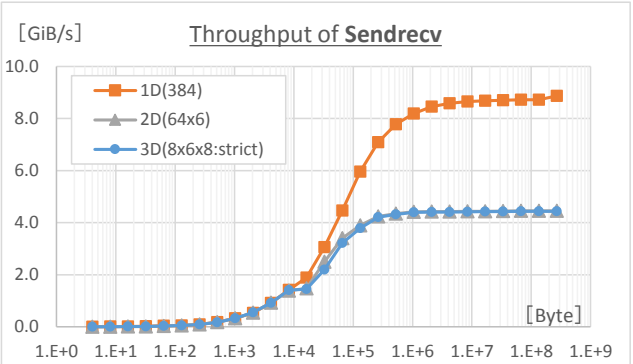


図 3　Sendrecv のスループット

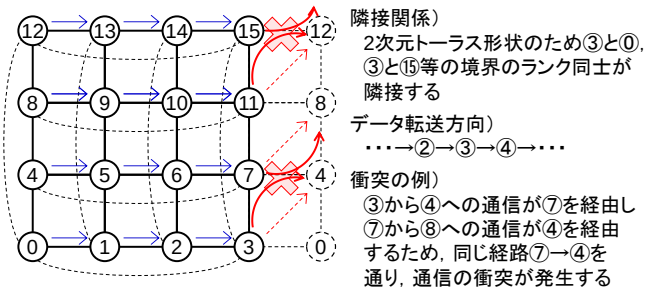


図 4　通信の衝突の例（2D トーラス形状）

3.3 集団通信の性能結果の特徴

本節では，測定した集団通信の結果を元に通信性能の傾向と特徴を示す．

集団通信は全ランクとの通信を行う．「京」における通信はノードあたり最大で 4 方向の同時通信が可能であり，インターコネクトバンド幅の実効値は最大で約 15[GiB/s]であるので，集団通信のスループット性能も最大で約 15[GiB/s]以下である．

3.3.1 Tofu 専用アルゴリズム適用による性能

集団通信において，「京」では Tofu 専用アルゴリズムが適用され性能が改善される場合がある．しかし，メッセージサイズが大きな場合に作業領域の不足のため性能が低下する場合があり，MCA パラメータ coll_tuned_prealloc_size により作業領域を大きくしておく必要がある．ここでは集団通信の Tofu 専用アルゴリズムが適用されたケースとして，Bcast，Allreduce，Reduce，Allgather の性能を説明する．

尚，coll_tuned_prealloc_size がデフォルト 6[MiB]の場合と 600[MiB]を指定した場合で，有意な差がある場合について，両方の結果を示す．

(1) **Bcast**

Bcast は，ルートランクから全ランクにデータを転送する[6]．スループットの結果を図 5 に示す．メッセージサイズが 16[KiB]～32[KiB]の辺りから，Tofu 専用アルゴリズムが適用され性能が向上した． Bcast では，ノード形状の次元の数だけ同時にデータを転送できるため，次元が上がると，スループットが向上する傾向にある．Bcast は基本的にノード数が増えるとパイプライン転送の各ステップにおける段数が増えるので，スループットが下がる傾向にある．例えば，ノード形状が 3 次元でメッセージサイズが 1[MiB]の時に，スループットが 3.6[GiB/s] (384 ノード)から 2.2[GiB/s] (3072 ノード)，1.5[GiB/s] (18432 ノード)へ下がった．

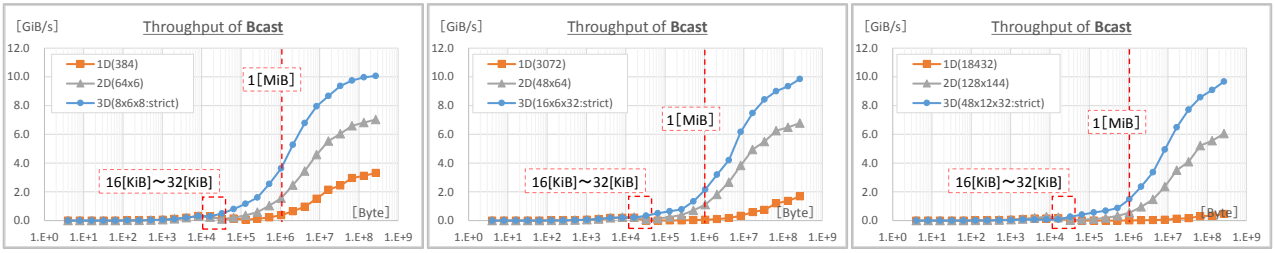


図 5　Bcast のスループット（384 ノード，3072 ノード，18432 ノード）

(2) **Allreduce**

Allreduce は，全ランクのデータを集めて集合演算を行い，その結果を全ランクに転送する．スループットの結果を図 6 に示す．Allreduce は，Reduce と Bcast の 2 段階で通信を行うため，スループットは，2 倍のメッセージサイズを通信時間で除算して算出した．ノード形状が 2 次元でメッセージサイズが 64[KiB]の辺り及び，ノード形状が 3 次元でメッセージサイズが 16[KiB]の辺りから，Tofu 専用アルゴリズムにより性能が向上する傾向にある．ノード形状が 2 次元の場合は 8[MiB]を超える辺りでアルゴリズムが変わるため性能が変化する．Bcast の場合と同様に，次元が上がると同時転送数が増えるため，スループットが向上する．

また，作業領域を使いながら集団通信を行うため，メッセージサイズが作業領域のデフォルト 6[MiB]を超えると作業領域が不足して性能が低下する場合があった．作業領域のサイズを MCA パラメータ coll_tuned_prealloc_size にて 600[MiB]と大きくすることにより，性能が改善して約 6[GiB/s]のスループットとなった．Bcast の通信性能より低い原因は，Reduce 処理の演算時間も合わせて通信時間としているためである．

(3) **Reduce**

Reduce は，全ランクのデータを集めて集合演算を行う．スループットの結果を図 7 に示す．ノード形状が 3 次元でメッセージサイズが 384 ノードで 8[KiB]～16[KiB]の辺り，3072 ノードと 18432 ノードで 16[KiB]～32[KiB]の辺りから，Tofu 専用アルゴリズムにより性能が向上する傾向にあった．

また，作業領域を使いながら集団通信を行うため，メッセージサイズが作業領域のデフォルト 6[MiB]を超えると作業領域が不足して性能が低下する場合があった．作業領域のサイズを MCA パラメータ coll_tuned_prealloc_size にて 600[MiB]と大きくすることにより，性能が改善して約 4[GiB/s]のスループットとなった．

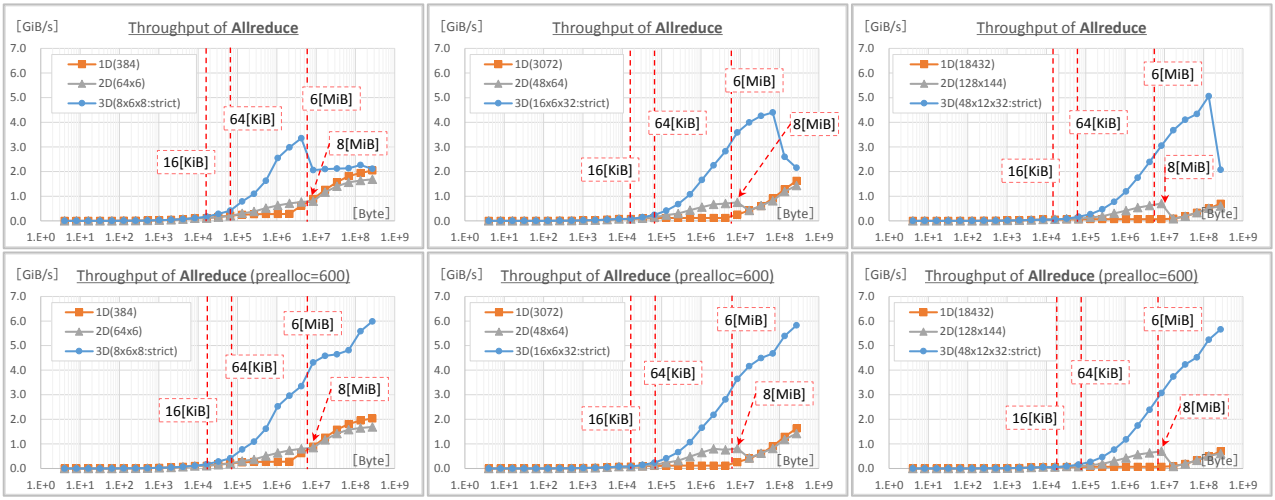


図 6　Allreduce のスループット（384 ノード，3072 ノード，18432 ノード）
（上段：作業領域デフォルト，下段：600MiB）

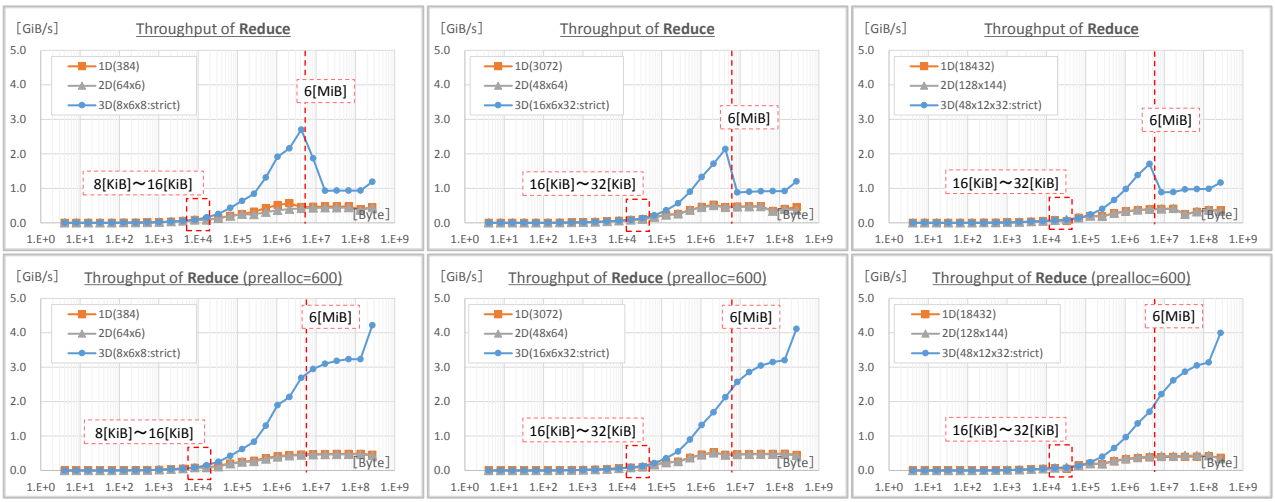


図 7　Reduce のスループット（384 ノード，3072 ノード，18432 ノード）
（上段：作業領域デフォルト，下段：600MiB）

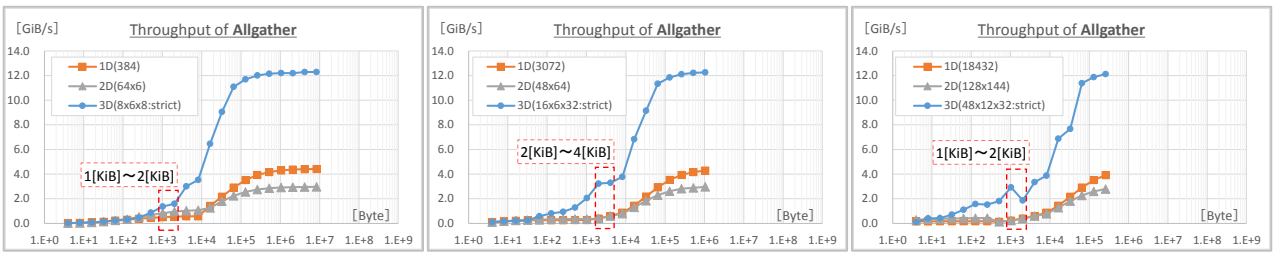


図 8　Allgather のスループット（384 ノード，3072 ノード，18432 ノード）

(4) Allgather

Allgather は，全ランクからルートランクへデータを集め，全ランクへ転送する．スループットの結果を図 8 に示す．全ランクからのデータ転送が発生するため，スループットはメッセージサイズにノード数を掛けたものを通信時間で除算して算出した．ノード形状が 3 次元で，メッセージサイズが 384 ノードで 1[KiB]～2[KiB]の辺り，3072 ノードで 2[KiB]～4[KiB]の辺り，18432 ノードで 1[KiB]～2[KiB]の辺りから，Tofu 専用アルゴリズムにより性能が向上する傾向にあった．

尚，メッセージサイズやノード数が増加すると，メモリが不足して実行出来なくなるため，ノード数が増加するにつれて測定可能なメッセージサイズの上限が下がった．

3.3.2 メッセージサイズによるアルゴリズム・性能の変化

集団通信の中には，メッセージサイズが小さい時に，専用のアルゴリズムが適用され，性能が向上する場合がある．ここでは，Gather と Scatter の性能を説明する．

(1) Gather

Gather は，全ランクからルートランクへデータを転送する．スループットの結果を図 9 に示す．スループットはメッセージサイズにノード数を掛けたものを通信時間で除算して算出した．Gather では，すべての割り当てノード形状で Tofu 専用アルゴリズムが適用される．100[Byte]以下のメッセージサイズが小さい領域では，Tofu 専用アルゴリズムによって，スループットが向上した．

尚，メッセージサイズやノード数が増加すると，メモリが不足して実行出来なくなるため，ノード数が増加するにつれて測定可能なメッセージサイズの上限が下がった．

(2) Scatter

Scatter は，ルートランクから全ランクへデータを分散する．スループットの結果を図 10 に示す．スループットはメッセージサイズにノード数を掛けたものを通信時間で除算して算出した．256[Byte]以下のメッセージサイズにおいて，小メッセージサイズ向けのアルゴリズムが適用され，スループットが向上する傾向にある．なお，Tofu 専用アルゴリズムは Scatter では用意されていない．256[Byte]より大きいメッセージサイズではスループットが大きく低下するので，改善案を検討中である．

尚，メッセージサイズやノード数が増加すると，メモリが不足して実行出来なくなるため，ノード数が増加するにつれて測定可能なメッセージサイズの上限が下がった．

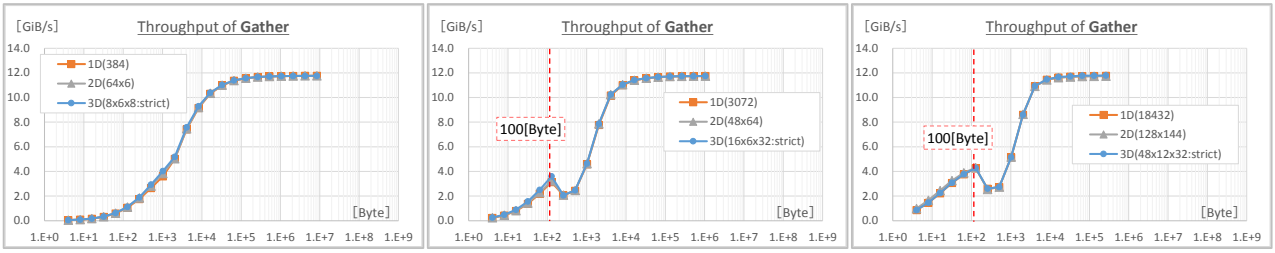


図 9　Gather のスループット（384 ノード，3072 ノード，18432 ノード）

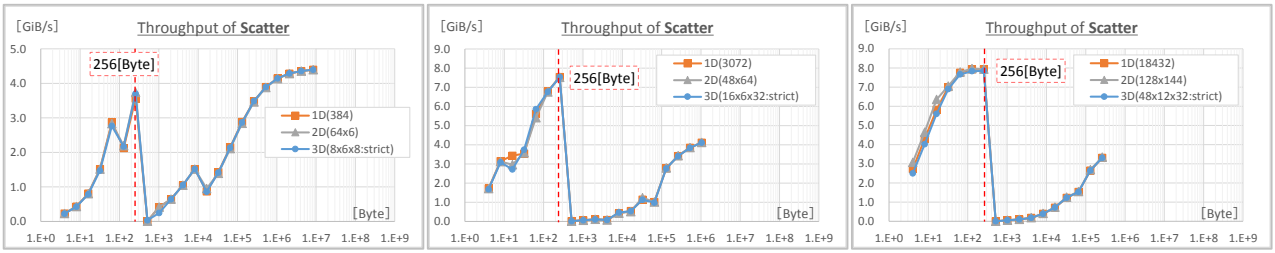


図 10　Scatter のスループット（384 ノード，3072 ノード，18432 ノード）

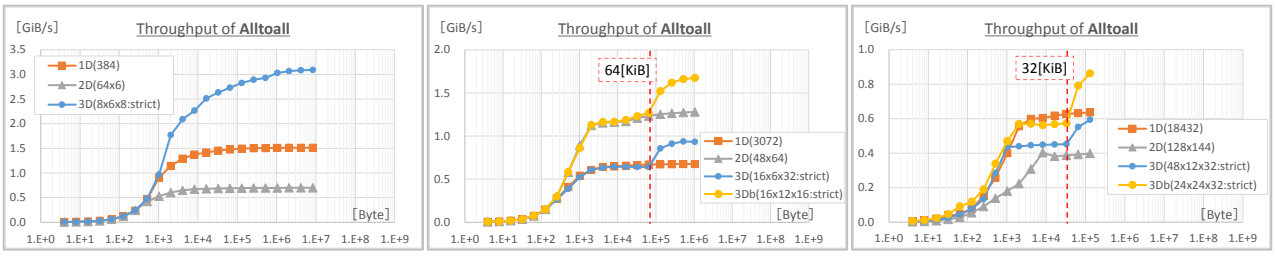
3.3.3 ノード形状による性能の違い

集団通信の中には，ノード形状により性能が変化する通信がある．ここでは Alltoall の性能を説明する．

(1) **Alltoall**

Alltoall では，各ランクが自分が持っているデータを他の全ランクへ転送する．スループットの結果を図 11 に示す．全ランクからのデータ転送が発生するため，スループットはメッセージサイズにノード数を掛けたものを通信時間で除算して算出した．すべての割り当てノード形状で Tofu 専用アルゴリズムが適用される．Alltoall の性能はバイセクションバンド幅に依存するため，Tofu の 6 次元のうちの a,b,c 軸を除いた物理 x,y,z 空間における形状が立方体に近いと，性能が高くなる．各ノード形状における物理 x,y,z 軸の大きさを図 11 に合わせて示す．3072 ノードでは，2 次元の物理 x,y,z 軸が 8x4x8 で，3 次元の物理 x,y,z 軸が 8x2x16 なので，2 次元の方が立方体に近く性能が高くなった．18432 ノードでは，1 次元の物理 x,y,z 軸が 12x16x8 で，3 次元の物理 x,y,z 軸が 24x4x16 なので，1 次元の方が立方体に近く性能が高くなった．3 次元においても，立方体に近いケースでは性能が高くなった．3072 ノードでは，物理 x,y,z 軸が 8x4x8 となる 3 次元(3Db)では立方体に近くなり性能が高くなった．18432 ノードでは，物理 x,y,z 軸が 12x8x16 となる 3 次元(3Db)では立方体に近くなり性能が高くなった．更に，3 次元では 2 種類のアルゴリズムが適用され，3072 ノードでは 64[KiB]，18432 ノードでは 32[KiB]を超えた辺りから性能が変化する．1 次元・2 次元・ strict 指定の無い 3 次元の場合は，ジョブを実行する度に割り当てられるノード形状が異なる可能性があるため，そのような場合はスループットが変動する．

尚，メッセージサイズやノード数が増加すると，メモリが不足して実行出来なくなるため，ノード数が増加するにつれて測定可能なメッセージサイズの上限が下がる．



384 ノードの物理 xyz 軸
- 1次元(1D):2x2x8
- 2次元(2D):1x2x16
- 3次元(3D):4x2x4

3072 ノードの物理 xyz 軸
- 1次元(1D) :2x8x16
- 2次元(2D) :8x4x8
- 3次元(3D) :8x2x16
- 3次元 b(3Db) :8x4x8

18432 ノードの物理 xyz 軸
- 1次元(1D) :12x16x8
- 2次元(2D) :24x8x8
- 3次元(3D) :24x4x16
- 3次元 b(3Db) :12x8x16

図 11　Alltoall のスループットと物理 xyz 軸（384 ノード，3072 ノード，18432 ノード）

4. 展開

今回の測定結果については，「京」の MPI 環境のリファレンスデータとして，ユーザに提供すると共に，言語環境をアップデートした場合の動作検証及び性能検証に活用している．

(1) **測定結果の提供**

ユーザの利便性を向上させるために，測定結果として，今回測定した 16 種類の通信関数の性能の傾向と測定結果データを提供している．限られた条件における結果ではあるが，通信性能のリファレンスデータとして，アプリケーションの開発や最適化に役立つことを期待している．

(2) **システム検証**

測定結果は，随時蓄積し，「京」の言語環境をアップデートした場合に過去の結果と比較することで，動作検証や性能検証に活用している．

5. まとめと今後の課題

「京」における MPI 通信性能として IMB を用いた性能を測定した．集団通信では，Tofu 専用アルゴリズムが適用されることで性能が向上するケースが多いことがわかった．更に，作業領域のサイズやノード形状によって性能が変化する場合があり，実行時に留意する必要があることがわかった．

また，測定結果は定期的にアップデートし，リファレンスデータとしてユーザに提供すると共に，動作検証や性能検証に活用していく予定である．

謝辞 本報告に際し，富士通株式会社の佐治隆行氏には，IMB の測定方法に関して、高科勝俊氏には測定結果の確認について，それぞれ多大な協力を得た．両氏に感謝すると共に，理化学研究所計算科学研究機構運用技術部門，富士通株式会社 SE の諸氏に感謝します．本論文の結果は，理化学研究所計算科学研究機構が保有するスーパーコンピュータ「京」によるものです．

参考文献

1) HPCI 戦略プログラム（戦略 5 分野）について
http://www.aics.riken.jp/jp/science/spire/

2) Toyoshima, T., ICC: An interconnect controller for the Tofu interconnect architecture., Hot Chips 22 (2010).

3) 安島雄一郎，井上智宏，平本新哉，清水俊幸：スーパーコンピュータ「京」のインターコネクト Tofu, FUJITSU.63,3(05,2012),pp.260-264, http://www.fujitsu.com/downloads/JP/archive/imgjp/jmag/vol63-3/paper05.pdf

4) 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，情報処理学会誌，コンピューティングシステム，Vol.5,No.5,152-162(Oct.2012).

5) Tomoya Adachi, Naoyuki Shida, Kenichi Miura, Shinji Sumimoto, Atsuya Uno, Motoyoshi Kurokawa, Fumiyoshi Shoji, and Mitsuo Yokokawa.:The Design of Ultra Scalable MPI Collective Communication on the K Computer.,2013 Comput. Sci., 28(2-3):147-155, May 2013.

6) Intel® MPI Benchmarks User Guide and Methodology Description, https://software.intel.com/en-us/articles/intel-mpi-benchmarks