マルチコア／マルチソケットノードにおけるメモリ性能のインパクト

研究代表者　朴　泰祐
筑波大学　システム情報工学研究科
taisuke@cs.tsukuba.ac.jp

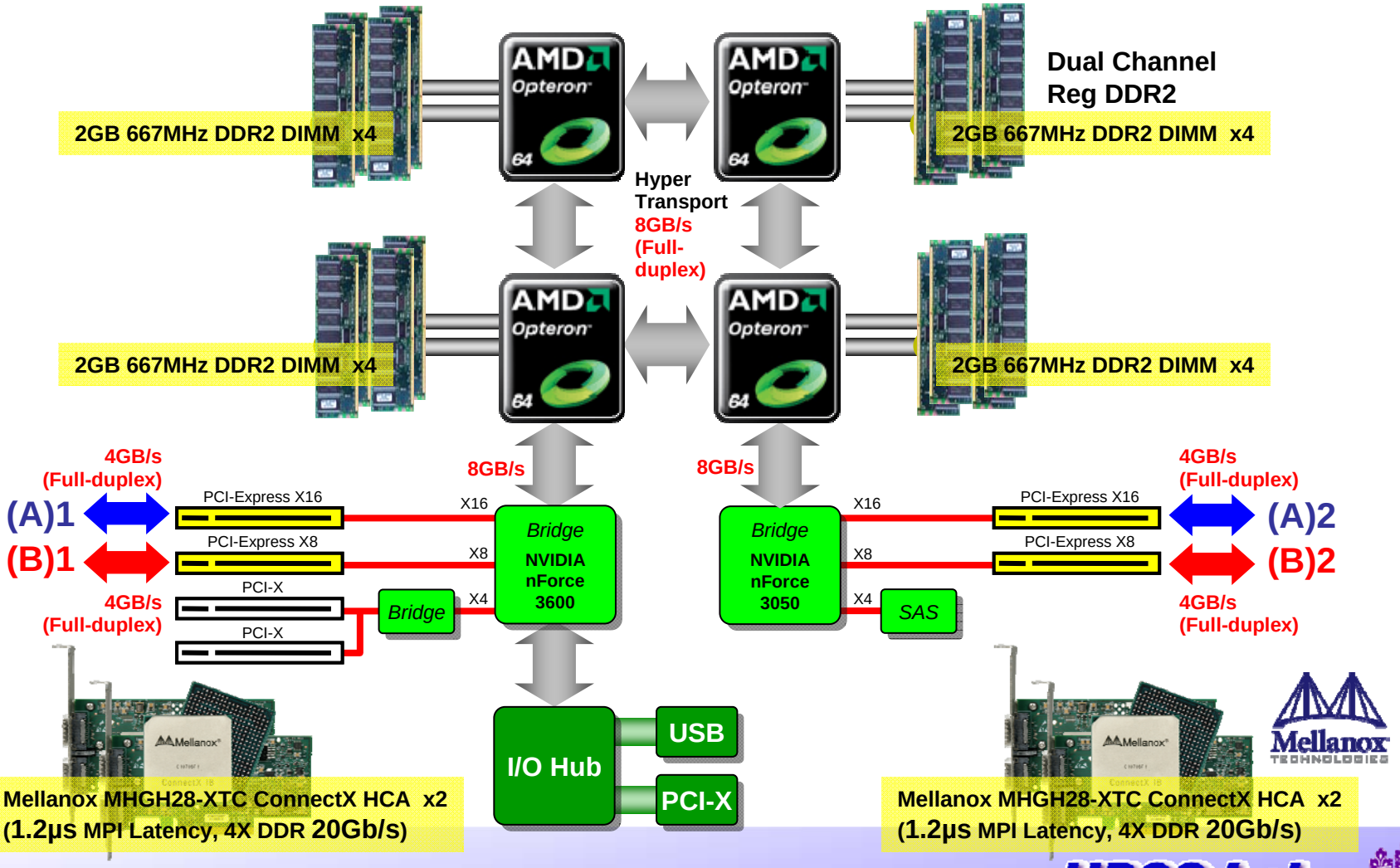
アウトライン

- 近年の高性能PCクラスタの傾向と問題
- multi-core/multi-socketノードとメモリ性能
- メモリバンド幅に着目した性能測定
- multi-link network性能評価
- まとめ

近年の高性能PCクラスタの傾向と問題点

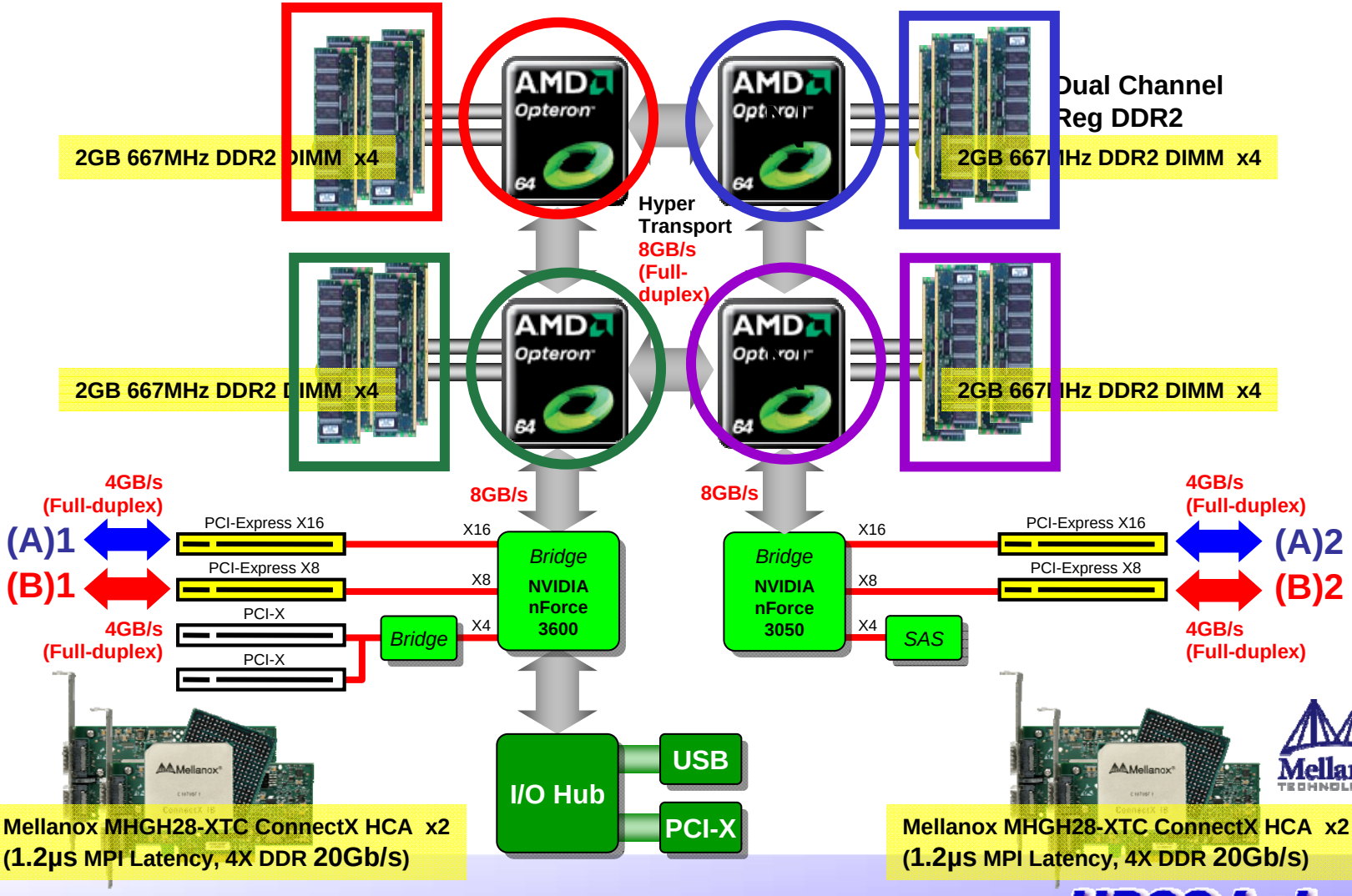
- ノード構成の傾向
  - CPUが4～6 core程度のmulti-core構成（以下、MC構成）となっている
  - ノード当たり複数ソケット（multi-socket: 以下、MS構成）となっている
- ネットワーク構成の傾向
  - Infinibandのような高性能ネットワークを大規模・多段Fat-Treeで構成
  - 複数リンクの平行結線によりネットワークバンド幅を増強（multi-rail、以下MR構成）
- ノード及びネットワーク性能の増強：MC-MS-MR構成
  ⇒実際の sustained performance を向上させるには様々な工夫が必要

T2K-Tsukuba計算ノードのブロックダイアグラム



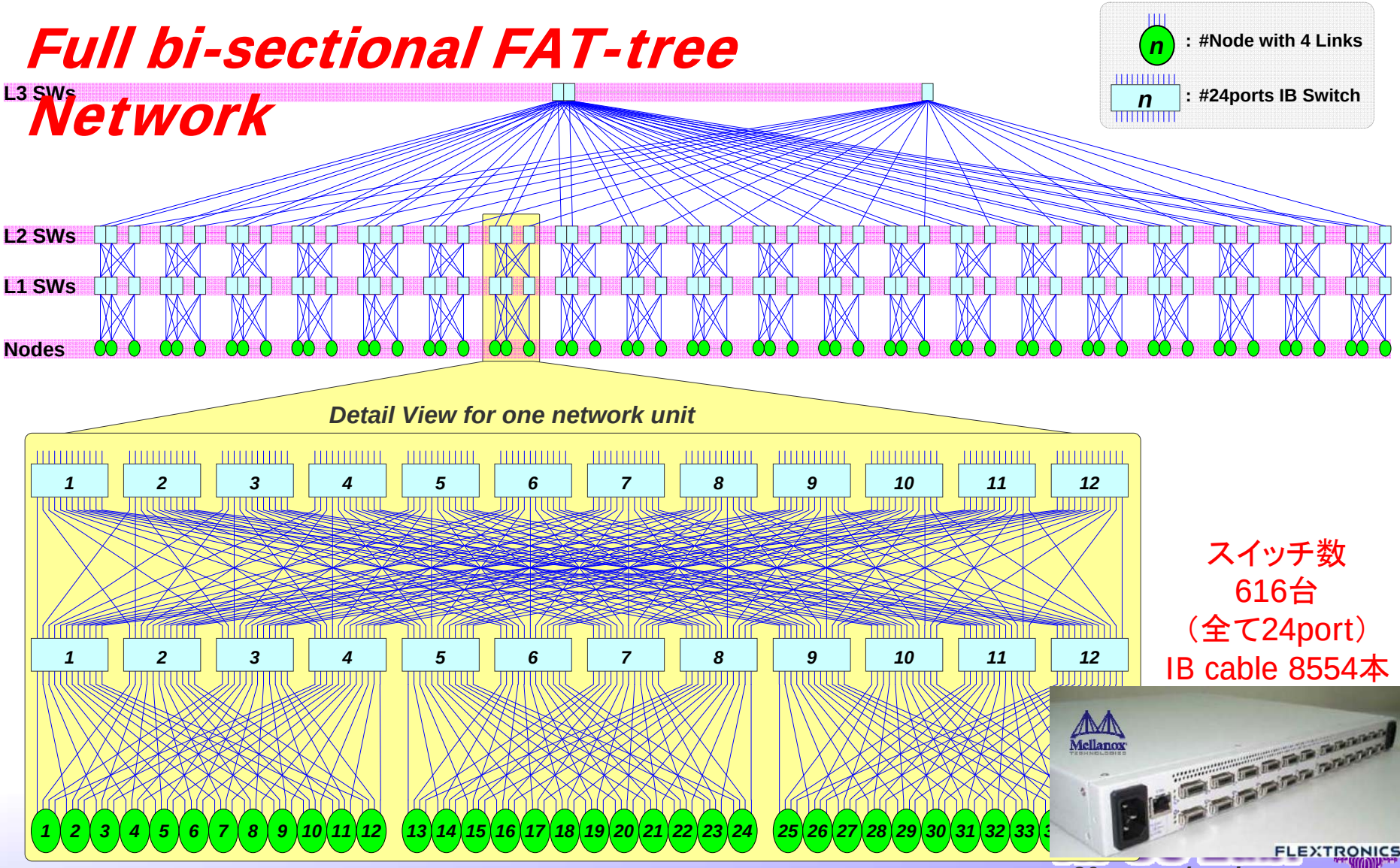
メモリマップとプロセスマップ

プロセス(コア)と参照データを近接メモリにマッピング可能(*numactl* 機能)



T2K-Tsukubaのノード間ネットワーク構成

Full bi-sectional FAT-tree Network



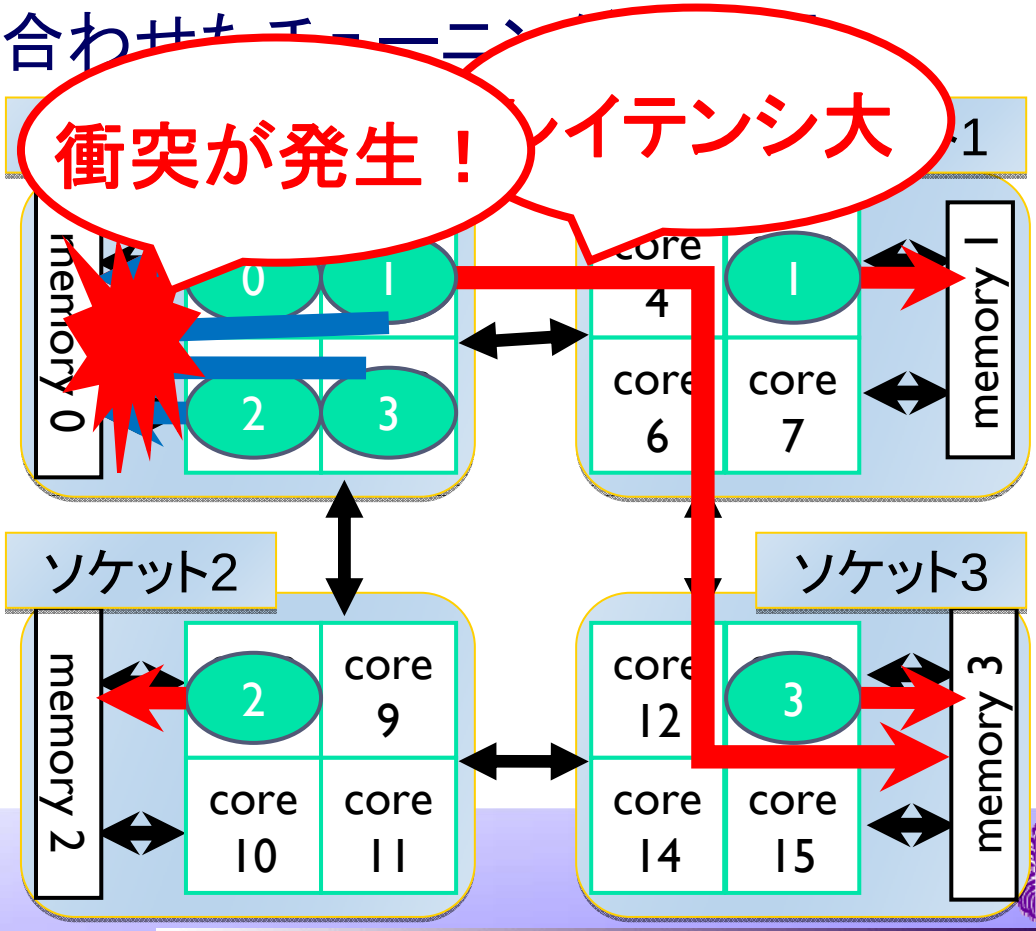
スイッチ数
616台
（全て24port）
IB cable 8554本

T2K-Tsukubaの計算ノード性能の問題点

- メモリバンド幅の不足
  - PACS-CS: 5.6GFLOPS, 6.4GB/s ⇒ 1.14 Byte/FLOP
  - T2K-Tsukuba: 147.2GFLOPS, 42.7GB/s⇒ 0.29Byte/FLOP
    ⇒PACS-CSに比べ約1/4のByte/FLOP性能しかない
- MC/MS構成により、メモリ階層が非常に複雑
  - B8000シリーズAMD quad-core Opteron (Barcelona)
    - 4つのcoreが各512KBの private L2 cache を持ち、さらに2MBの shared L3 cacheを持つ
    - 4つのCPU socketは共有メモリ結合だが構成はNUMA (Non-Uniform Memory Architecture)
- MC化はさらに進むが、メモリ性能が追いつかない！
  - DDR2->DDR3, FSBのさらなる向上でメモリも良くなっているがcore数はそれを上回る勢いで増える
    ⇒HPC的にどうか？
- 以上の背景の下、MC/MS構成ノードにおける演算性能とメモリ性能の特性を調べ、MC/MS環境のcoreの有効利用方法を探る

マルチソケット環境における並列化

- NUMA(Non-Uniform Memory Access)アーキテクチャ
  - 高いメモリアクセス性能
  - NUMAアーキテクチャに合わせたチューニング
  - ⇒NUMAコントロール (numactl)
- 並列化による性能向上
  - メモリバインド
  - プロセスのマッピング



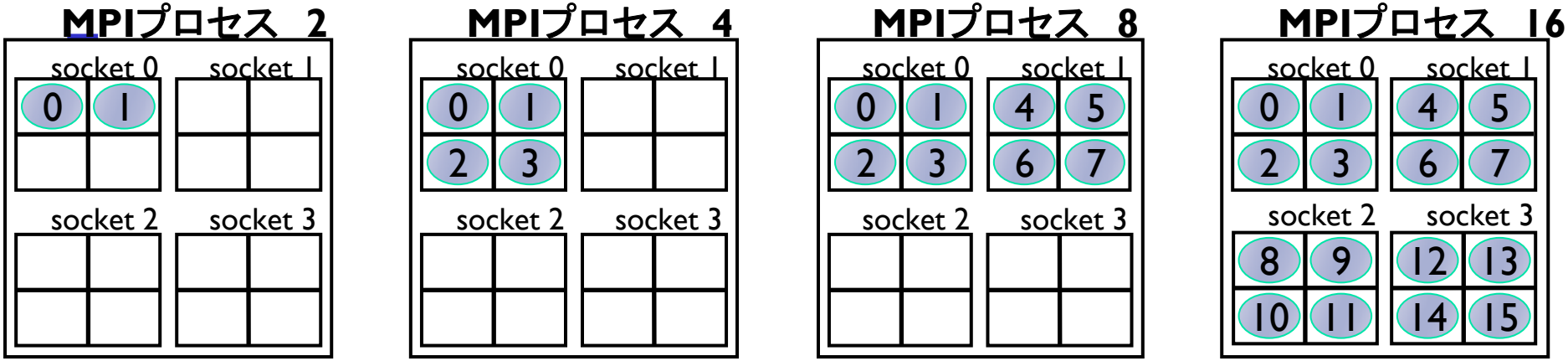
T2K-TsukubaにおけるMC/MSノード性能

- Byte/FLOPという尺度に着目し、synthetic benchmarkによりMC/MSにおけるプロセスマッピングとメモリ性能の関係を詳細評価

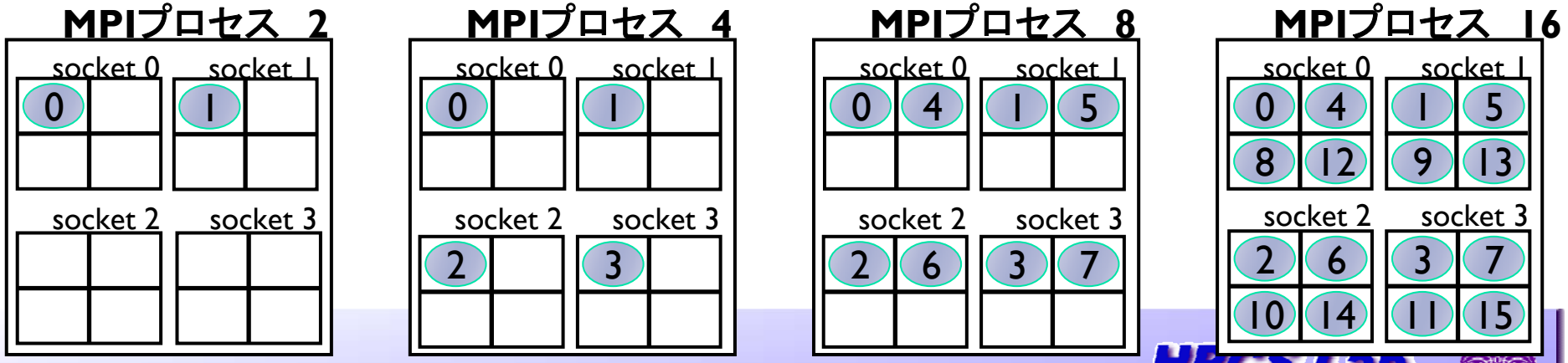
```
double a[N/p], b[N/p],  x1,  x2, y;   /* N is large enough */

for(i=0; i<N/p; i++){
     x1 = _mm_load_pd(&a[i]);   ← メモリアクセス1
     x2 = _mm_load_pd(&b[i]);
     y1 = x1;
     y2 = x2;
     y1 = _mm_mul_pd(y1,y2);
     y1 = _mm_mul_pd(y1,y2);    } 浮動小数点演算回数を調節
       ……                        （1, 2, 4, 8, 12, 16, 24回）
     _mm_store_pd(&c[i],  y1);
}
```

プロセスのマッピング: Linux numactl

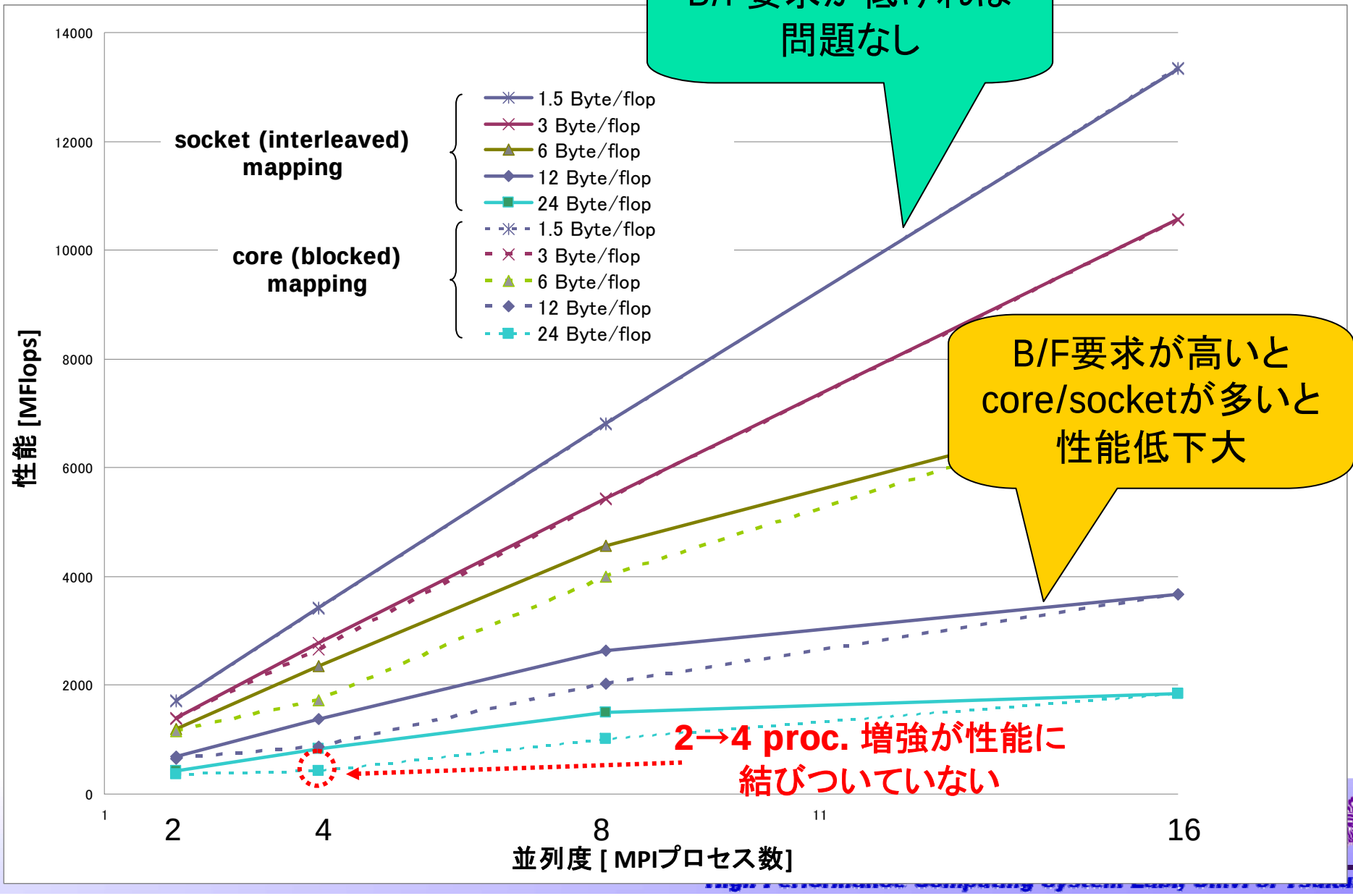
- numactl –physcpubind (core mapping: blocked)



- numactl –cpunodebind (socket mapping: interleaved)



プロセスとソケットのマッピングとメモリ性能



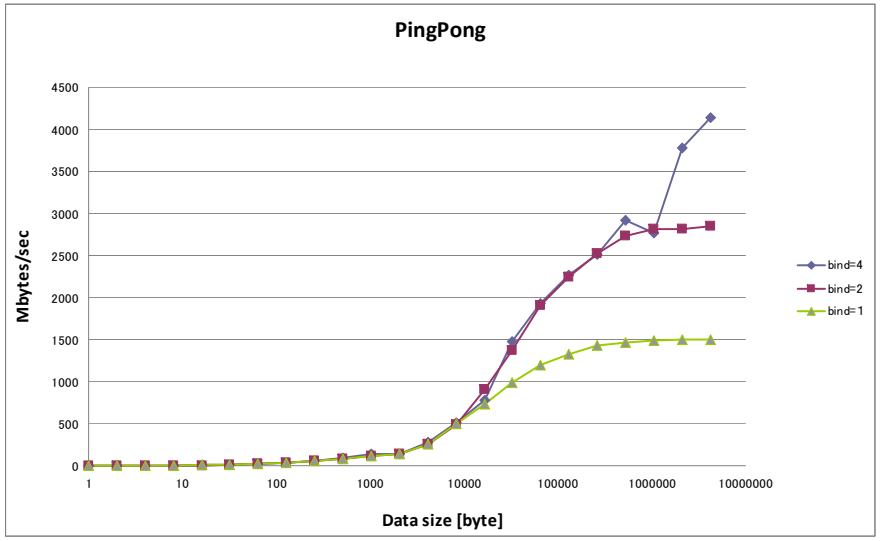
考察と今後の進展

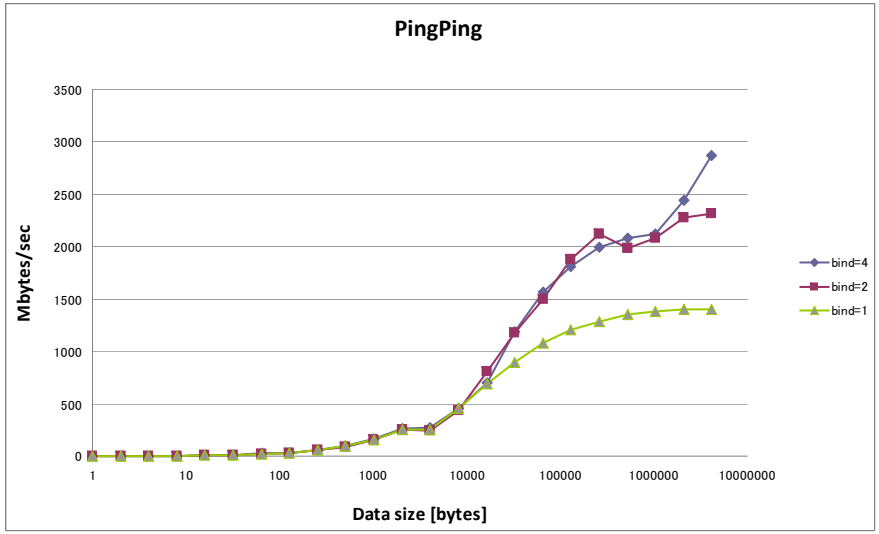
- numactlによるプロセスマッピングの重要性
  - socket mapping か core mapping かを慎重に検討する必要あり
  - メモリバンド幅要求による影響が強い
- メモリ性能限界
  - Byte/FLOPに基づく性能予測は重要
  - プロセス数の増加が必ずしも性能に結びつかない場合がある
- プロセス数またはスレッド数の制御
  - アプリケーションのメモリバンド幅要求に応じ、性能向上に結びつく利用コア数限界を求める
  - 「余剰コア」が発生した場合、これを有効利用する
    （例：通信スレッドへの割り当てによる全体性能の向上）

MR特性の予備評価

- T2K-TsukubaにおけるノードNode間通信のMRの数（何本のInfinibandを通信に用いるか）
- T2K-TsukubaにおけるFat-Treeネットワークの評価
- Intel MPI benchmarkによる性能評価
- 使用ノード数：2～128 nodes

pingpong, pingping性能 [バンド幅]



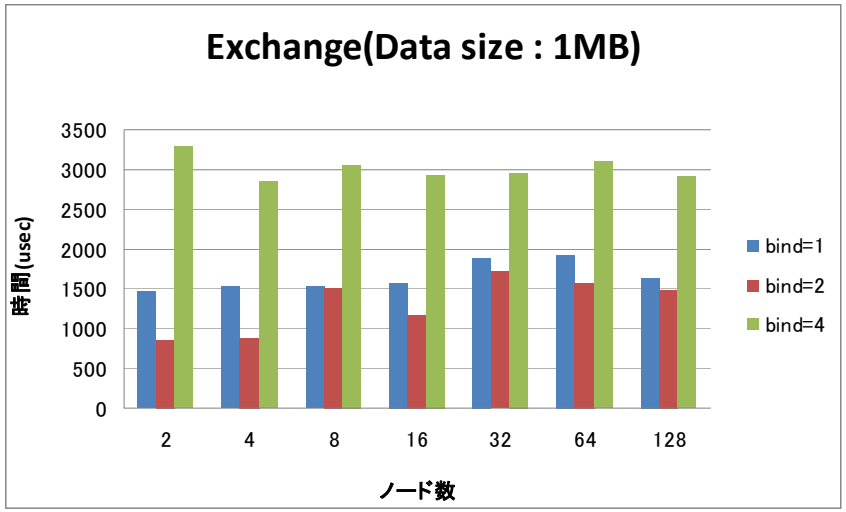


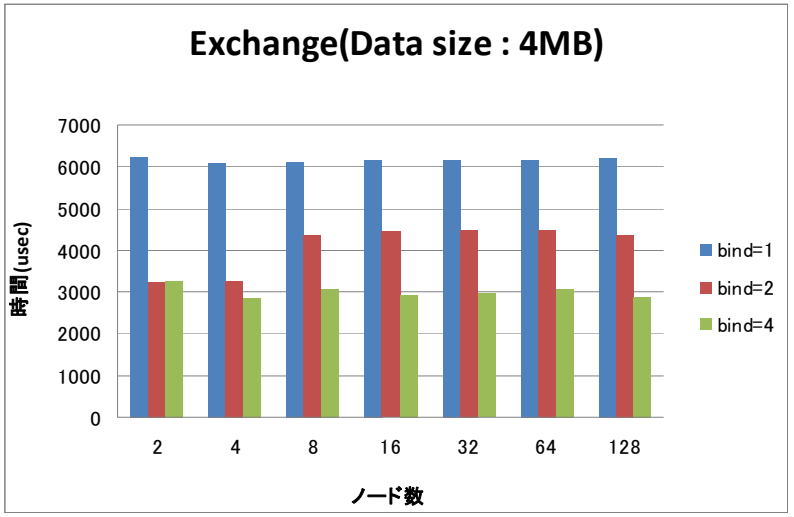
- MR=2 ⇒ MR=4の効果は小さい（データサイズがかなり大きくないと効果がない）
- pingpongとpingpingの性能は近い

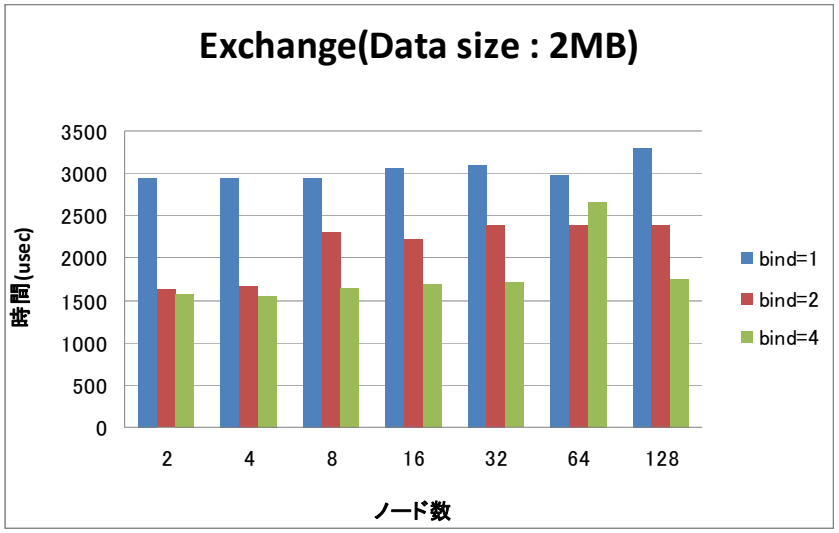
⇒PCI-Expressに十分なバンド幅があり、双方向通信でも高速

- （どちらかというと）multi-railは複数の通信ストリームに分散させて使った方がよいのではないか？

Exchange（隣接転送）性能 [時間]

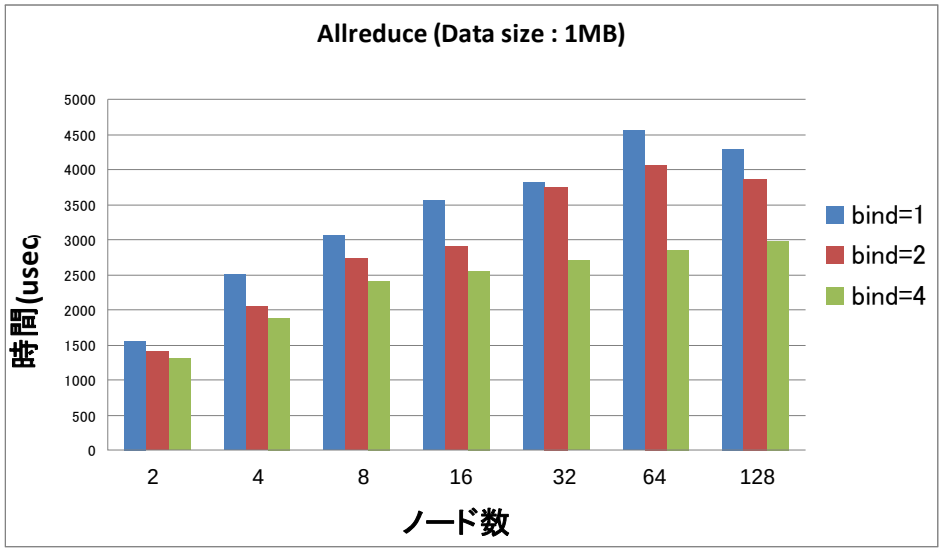


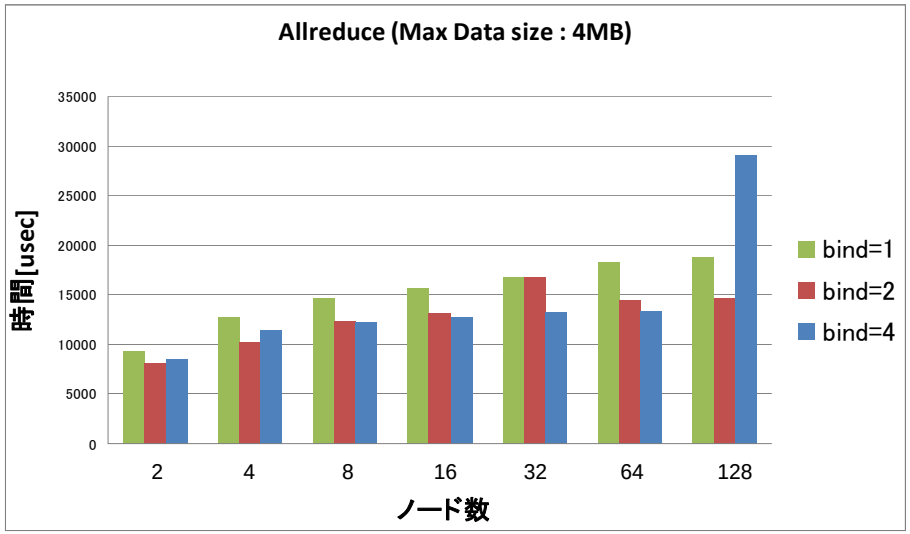


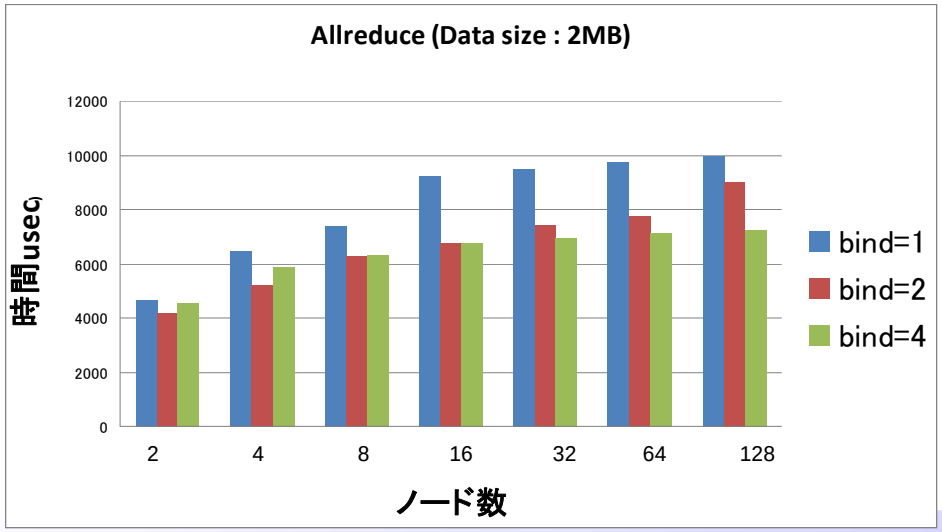


- 1MBの時のデータがおかしい？
- データサイズが大きいとMR=4の効果が高く出る
- Fat-treeであることで、ノード数が増加しても通信性能に影響しない

Allreduce性能 [時間]

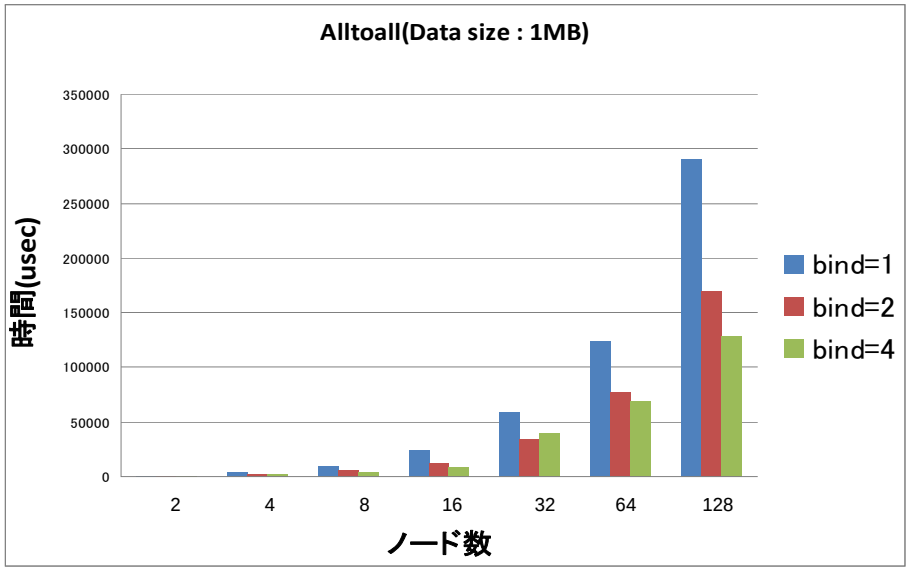


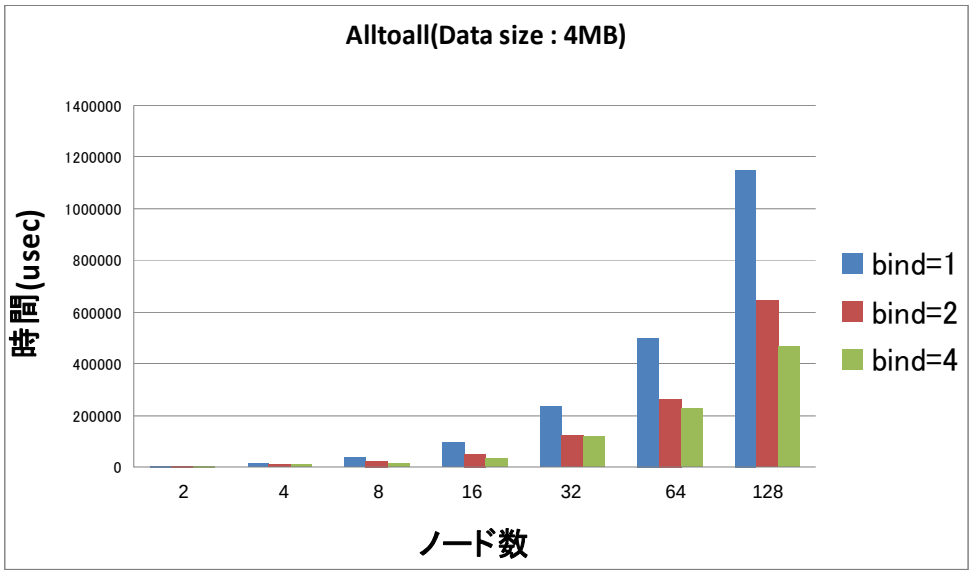


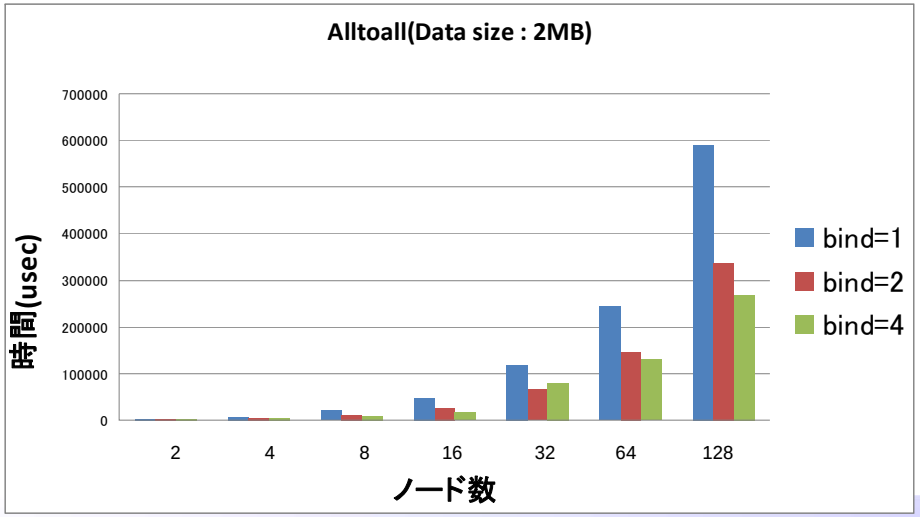


- データサイズが小さい場合はMR数増加で通信性能改善
  4MB時にMR=4で性能劣化
  ⇒通信バッファ不足？
- ノード数増加に対し、logオーダー程度の通信時間
  ⇒Fat-Treeが有効に働いている

Alltoall性能 [時間]

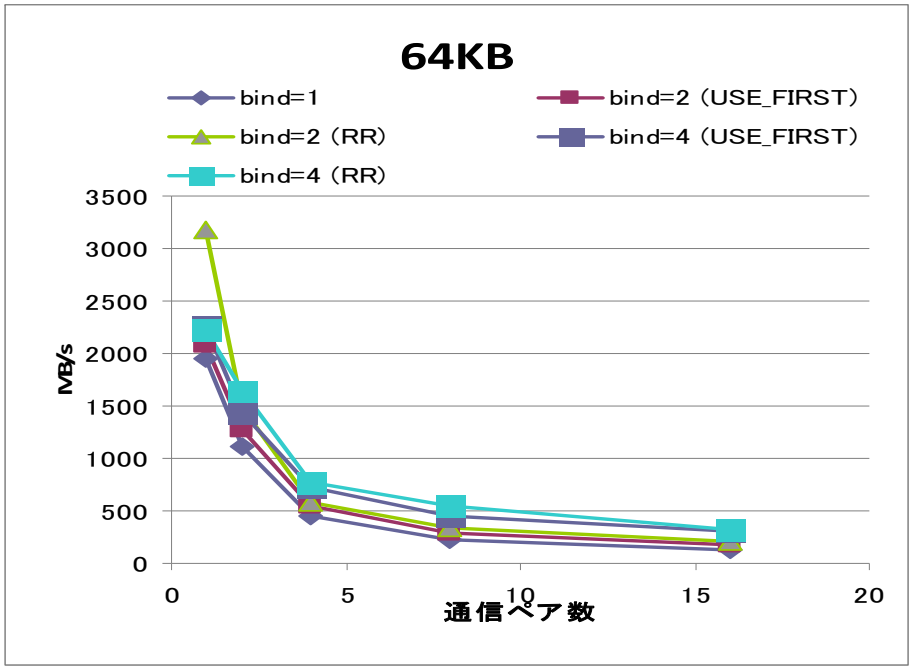


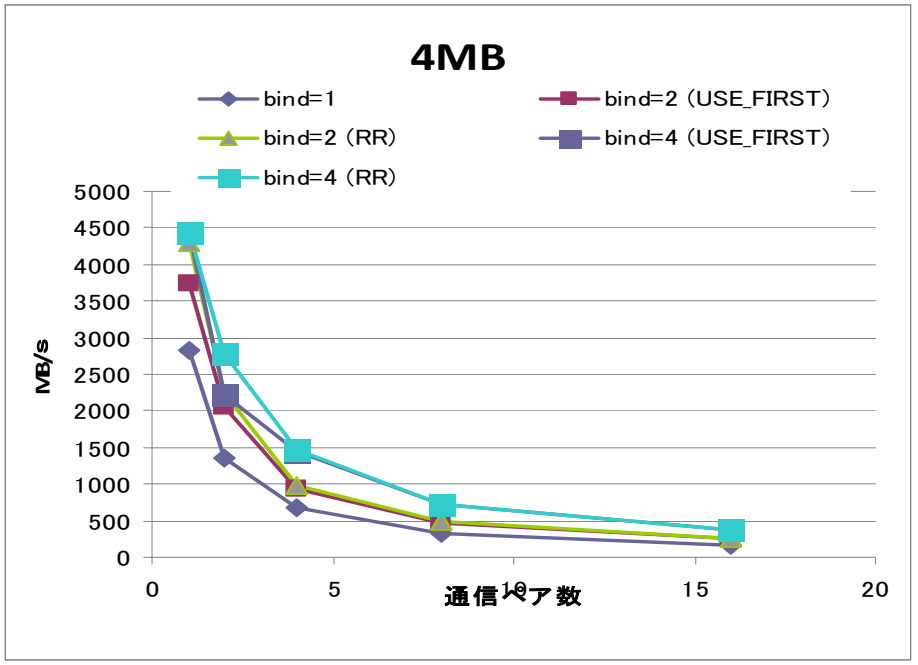




- どの場合でもMR数増加が性能改善に貢献
- 最も多数のメッセージ通信が行われるが他のcollective通信より性能が安定している
  ⇒Fat-Treeが有効に働いている

ノードを跨ぐ多数通信ペアの性能 [バンド幅]





- 2ノード間の通信ペア数を変化させたsendrecv通信（双方向同時）
- 通信ペア数が増えれば、MR数を増やすよりもMR=1で複数通信ストリームを同時並行実行した方が効率が良いはず

⇒ MR=4が常に良い

- T2K-Tsukubaで用いているMVAPICH2の設定、パラメータ選択に問題があるのでは？

まとめ

- 現状のT2K-Tsukubaにおいて
  - point-to-point, collective のどちらの通信でも Infiniband のMR (binding) は概ね効果がある
  - 一部の負性特性領域・ケースがあるが、MR=2 or MR=4 としておいた方が全体の通信性能が向上する
  - point-to-pointとcollectiveではMRの効き方に違いがあるため、アプリケーションで重要な通信の種類とデータサイズに応じ、最適なMR数を見つける必要があるのでは？

- 現状の問題点と今後の課題
  - ノード上に複数MPIプロセスがあり、同時に多数の通信を行う場合、MR=1で複数ストリームを同時に処理できていない？
    ⇒いつでもMR=2 or MR=4とした方が「とりあえず」性能が高い
  - 今後、MVAPICH2の実装とパラメータ設定を詳細に調査
  - MC/MS/MRという複雑な構造における性能最適化のため、coreとrailの使い方、パラメータ設定を（半）自動化するようなシステムを作りたい