PMv2, PMvX ライブラリを用いた PACS-CS向け高速通信ライブラリ

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

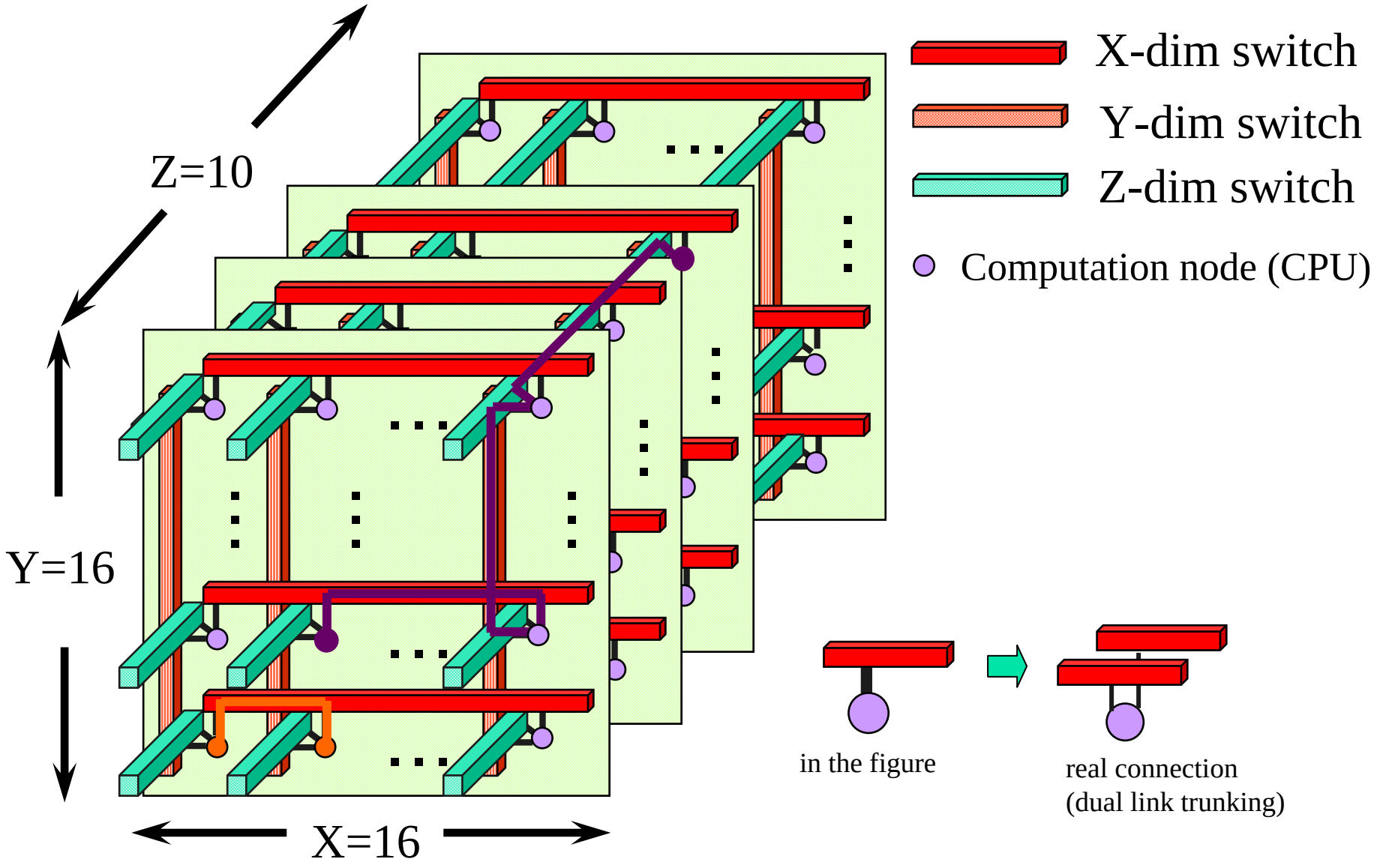
- 研究の背景
- PACS-CSにおける通信機構と問題点
- PMv2, PMvx通信機構
- QCDへの適用
- 研究の状況
- 今後の課題

研究の背景

- 超並列クラスタPACS-CS
  - メモリバンド幅、通信バンド幅重視の科学技術計算向けPCクラスタ
  - Single-core / node ⇒ 高い Byte/Flop 値
  - マルチリンクEthernetによる3次元ハイパクロスバ（3D-HXB）をネットワークシステムソフトウェアとして実装
    ⇒ 多次元 domain decomposition 等に適した高バンド幅通信

- 通信の最適化のために
  - 多次元同時転送⇒合計バンド幅の向上
  - Ethernetベースのソフトウェアによる multi-link bindingのため、メッセージ長が短いと性能が出にくい
  - トポロジを意識した集団通信（collective communication）
  - データのコピー回数を減らしてレイテンシ削減／バンド幅向上

PACS-CSのネットワーク構成
(3-D HXB network) 2560 nodes



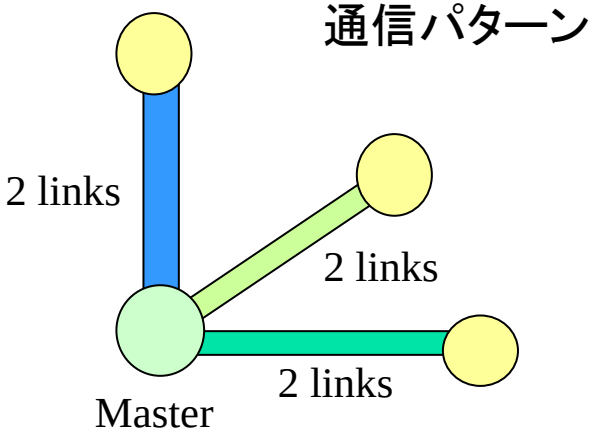
PACS-CSにおける通信機構：PMv2, PMvX

- PACS-CSのOS
  - Linux RedHat + SCore
  - 3D-HXBドライバ：PM/Ethernet-HXB
- PM/Ethernet-HXB
  - multi-link Ethernet上で3D-HXBネットワークを構成するためのネットワークドライバソフトウェア
  - ２つのバージョン
    - PMv2: 従来のSCore上のPMと上位互換性を持つPM/Ethernet-HXB
    - PMvX: 配列データのコピー回数を最小化する改良型PM/Ethernet-HXB

PM/Ethernet-HXB隣接通信性能：多次元 (PMv2)

	送信 MB/s (%)	受信 MB/s (%)	送受信 MB/s (%)
1次元	247.6 (99.0)	247.5 (99.0)	481.1 (96.2)
2次元	493.6 (98.7)	494.9 (99.0)	951.3 (95.1)
3次元	741.3 (98.8)	742.3 (99.0)	1401.3 (93.4)

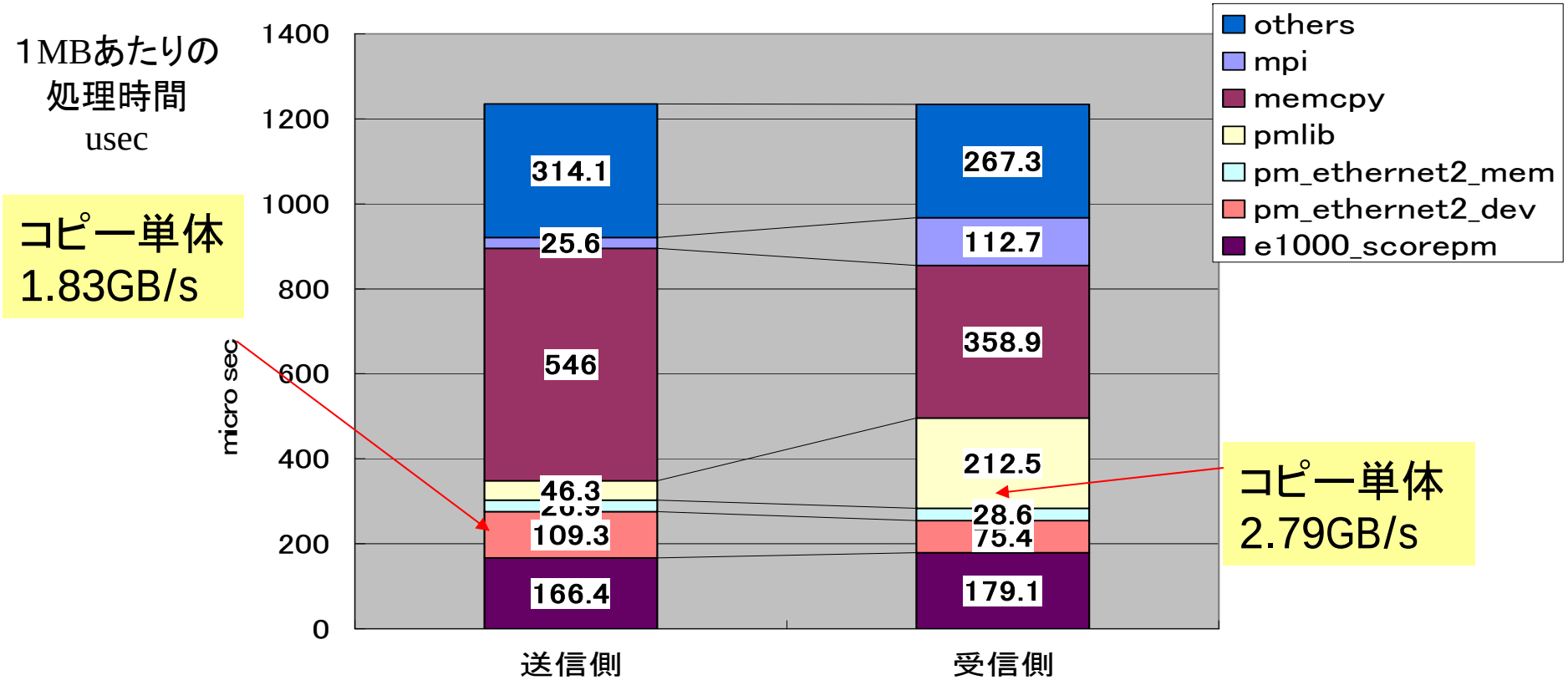
- 4ノードを用い3次元クロスバ通信性能を評価
- PMレベルでは93.4%以上の実効性能を実現



MPI通信性能解析:1次元単方向

- コピーの占める割合が大きい

iScopes(SACSIS2006)による解析結果



コピー単体
1.83GB/s

コピー単体
2.79GB/s

MPIバンド幅性能:810 MB/s

PMv2における問題点

- PMは本来、比較的低いレイテンシのネットワーク（Myrinet or Infiniband）を想定しているため、8KB程度のパケットで全ての通信を非同期に行う
- この上にMPIを実装した場合（例えばMPICH/PMやYAMPI/PM）、結局ユーザデータとMPIバッファ間のデータコピーが余計に発生し、メモリバスを圧迫する
- 解析の結果、メモリバス上でのデータコピーのオーバヘッドは無視できない
- 重要な通信はMPIレベルではなくPMレベルのまま実行したい
  ⇒YAMPI/PMはPMレベル通信との混在を許すように設計されている

⇒特定アプリケーション向けにPMレベルでの通信ライブラリを実装

Comlib/PMv2: QCD向けPM通信ライブラリ

- QCDにおける基本的通信パターン
  - 3次元6方向の物理的に隣接したノード間同時転送
  - 小規模スカラー（double precision complex）のallreduce通信
- 従来のComlib/MPIはこれをMPIレベルで実装
  - isend/irecvを多次元方向に複数発行し、wait
  - MPI_allreduce等によりreductionを実行
- これらをプログラム上の互換性を保ちながらPMv2で実装
  - PMのパケット単位でメッセージを管理し、非同期通信を管理
  - PMレベルでallreduce等のcollective通信を実行
  - その他、ユーザが一般的なMPI通信をプログラムの他の部分で発行することを許す（MPIとの混在可能）

10000回の通信のスループットのヒストグラム：通信性能の安定性

スループット[MB/s]	通信のみ	通信＋計算
0-20	0	4
20-40	0	22
40-60	0	6
60-80	3	0
80-100	1	4
100-120	1	0
120-140	1	0
140-160	728	135
160-180	9230	8579
180-200	34	420
200-220	0	113
240-240	2	671
240-260	0	46

結論：

(1) 通信のみの場合より通信＋計算の場合の方が値が散るが平均性能としては大差はない。

(2) 全体として、各方法で約15分ずつの測定をしたが傾向は変化しなかった。

(3) よって、PMv2レベルでの通信劣化は見受けられない。

さらなる改良：PMvXへの対応

- PMv2ではMPIにおけるバッファコピーを1回減らせるが、8KB程度のパケットサイズ単位で通信が切られるため、多次元隣接転送等で通信skewが発生すると性能が大幅に劣化する可能性がある
- PMv2に対し、BufferArrayという概念を導入⇒PMvX
  - 基本パケットサイズは8KBで同じだが、これを多数連結したlinked listを作成可能⇒BufferArray
  - 1回の通信発行でBufferArray単位で通信の起動が可能
- QCD向けにComlib/PMvXを実装
  - 従来のComlibと完全互換
  - Comlib/PMv2で行っていた linked list buffer の管理をBufferArrayで置き換え、システム関数起動回数を大幅に削減

PMvXの主要関数

関数名	関数の概要
pmGetSendBufferArray	送信用pmBufferArrayの獲得
pmSendBufferArray	pmBufferArrayの送信
pmReceiveBufferArray	pmBufferArrayの受信
pmReleaseReceiveBufferArray	受信用pmBufferArrayの解放 （受信BufferArray毎に実行する）
pmReleaseSendBufferArray	送信用pmBufferArrayの解放
pmSendDoneBufferArray	送信用pmBufferArrayの送信完了確認

現状

- Comlib/PMvXをQCD-mult等のQCDベンチマークに適用し、動作が正しいことを確認
- Comlib/MPI vs Comlib/PMv2 vs Comlib/PMvX の完全な比較を実施しようとしたが時間切れ
- さらにPMvXのより有効な利用のため
  - MPIで実装されている主要collective通信（allreduce, alltoall等）をPACS-CSの3D-HXBトポロジを意識してアルゴリズムを最適化し、PMvXで実装
  - 一般のMPI collective通信では、ツリー構造での通信と線形構造での通信を組み合わせるが、適当な仮定を置いて切り替えている
    ⇒PACS-CSの3D-HXBを想定した最適化を行う
  - 基本アルゴリズムと一部実装はできているが未完成

まとめと今後の課題

- PACS-CSの3D-HXB/Ethernetを最大限有効利用するため、アプリケーション向けに最適化されたPMv2, PMvX直接利用ライブラリを開発
- Comlib/PMv2では通信性能の安定化が見られた
- Comlib/PMvXは評価途中
- その他のcollective通信をPMvXで実装する、より一般化されたComlibを実装中
- 研究においてQCDベンチマークコードの提供を受け、アプリケーションを意識したシステム開発を実施
- 次の機会でComlib/PMvXの評価とcollective通信のライブラリセットを完成させたい