省メモリ技術と動的最適化技術による スケーラブル通信ライブラリの開発

九州大学:
南里豪志、本田宏明、薄田竜太郎、森江善之、小林泰三
富士通株式会社:
住元真司、安島雄一郎、志田直之、佐賀一繁、野瀬貴史
公益財団法人九州先端科学技術研究所:
柴村英智、曽我武史
大分大学:
高見利也
京都大学:
深沢圭一郎

2016年10月20日
CREST「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」領域会議
南里チーム資料

講演内容

- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- アプリケーション、ミドルウェアへの応用
  - 通信最適化、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- イベント・デモ

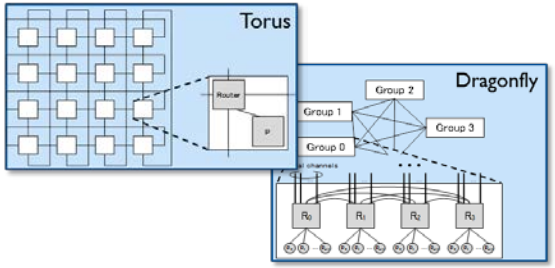
講演内容

- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- アプリケーション、ミドルウェアへの応用
  - 通信最適化、ACP + MPI、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- イベント・デモ

研究の狙い

- エクサスケール計算環境での利用に耐える
  「スケーラブルな通信ライブラリ」
  およびそれを活用したアプリケーションの開発

- 想定するエクサスケール計算環境：
  - ノード数：数十万ノード
  - プロセス数：数百万
  - メモリ量：10GB程度 / プロセス



- 通信ライブラリ、およびアプリケーションへの要求：
  通信性能と省メモリの両立による高スケーラビリティの達成

本プロジェクトで開発・公開するソフトウェア群

- ACP (Advanced Communication Primitives)ライブラリ
  - PGAS型省メモリ通信ライブラリ
- NSIM-ACE
  - ネットワークシミュレータ
- DMATP-MPI
  - 通信ライブラリの使用メモリ量解析ツール
- 各アプリケーション
  - Halo通信関数、OpenFMO、粒子系シミュレーション
- スクリプト言語向けインタフェース
  - Global Array on Ruby、ACP for Python

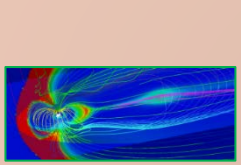
講演内容

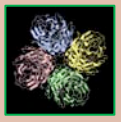
- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- 通信最適化に向けた取り組み
  - 通信最適化、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- イベント・デモ

ACP(Advanced Communication Primitives)ライブラリの構成

Languages/Applications

ACPライブラリを活用したスケーラブルな言語/アプリケーション
Python / Ruby / MHD / 粒子系シミュレーション / OpenFMO / ...





ACP通信ライブラリ

ACP中間層(Middle Layer)

Dataライブラリ

遠隔・分散データ構造の生成と操作を
省メモリ、低遅延で行うインタフェース
vector / list / deque / set / map /
workspace

Communicationライブラリ

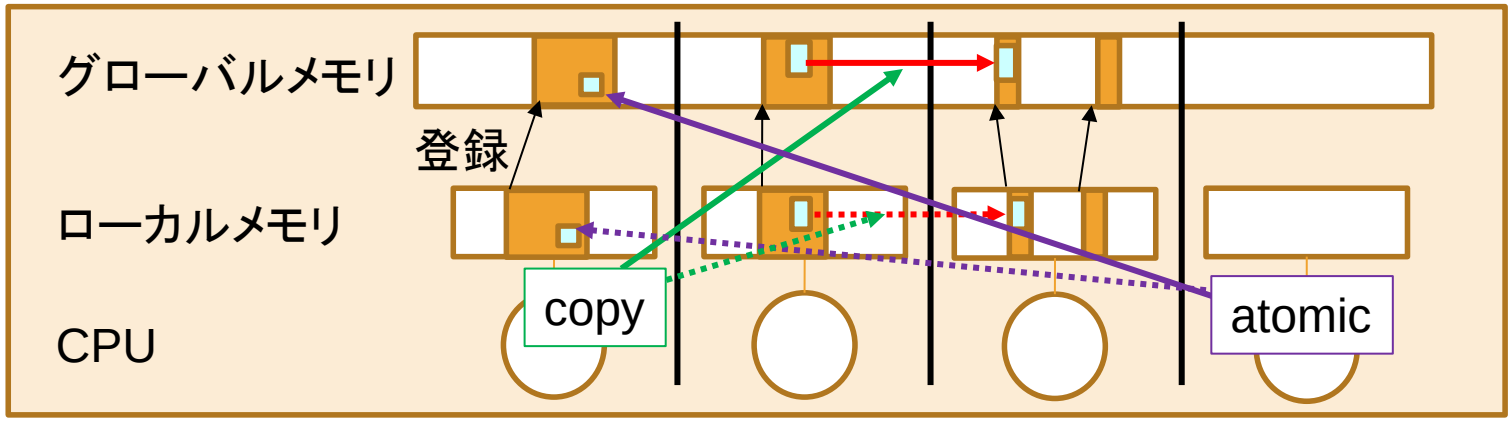
アプリケーションで必要とされる共通の通信
パターンを省メモリ低オーバヘッド実装した
インタフェース
channel / multi-channel / patterned comm.

ACP基本層(Basic Layer)

PGASモデルを採用した省メモリ低オーバヘッドの通信抽象化層
copy / atomic / memory registration / sync / ...

インターコネクトネットワーク （InfiniBand, Tofu, Ethernet)

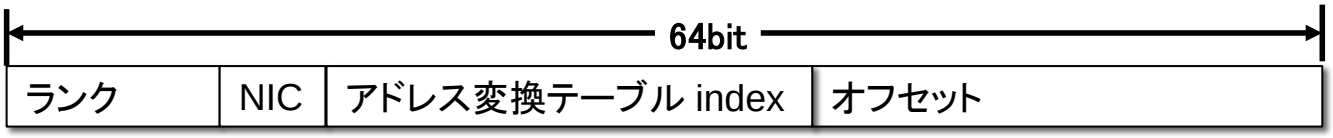
ACP基本層



- 64ビットグローバルアドレス
  - アドレス空間はプロセスに関連して、明示的に区画化される（PGAS）
- グローバルメモリ管理
  - 静的メモリ、動的メモリ
- グローバルメモリアクセス
  - 任意のプロセス間のデータ転送、4および8バイトの不可分操作
- メモリアクセス順序制御
  - 逐次実行、並列実行

ACP基本層の実装（Tofu, Tofu2, IB, UDP）

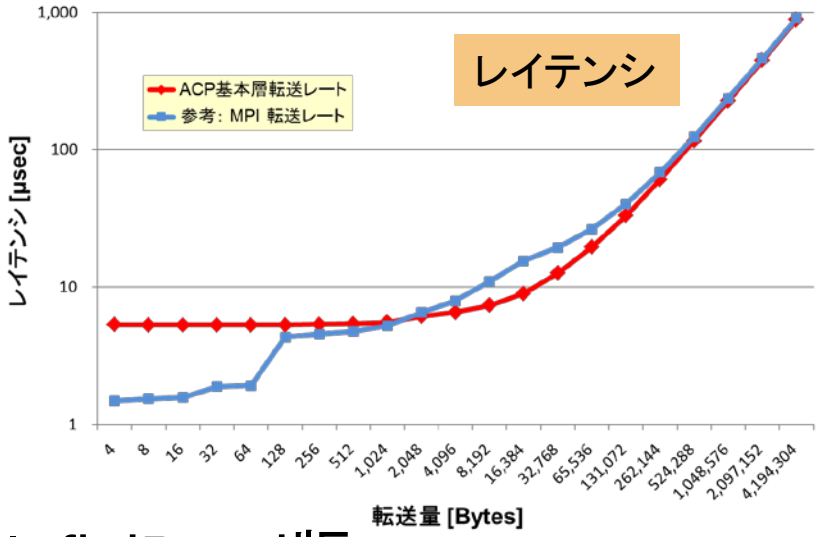
- 64bitグローバルアドレスへの情報圧縮
  - 例）InfiniBand

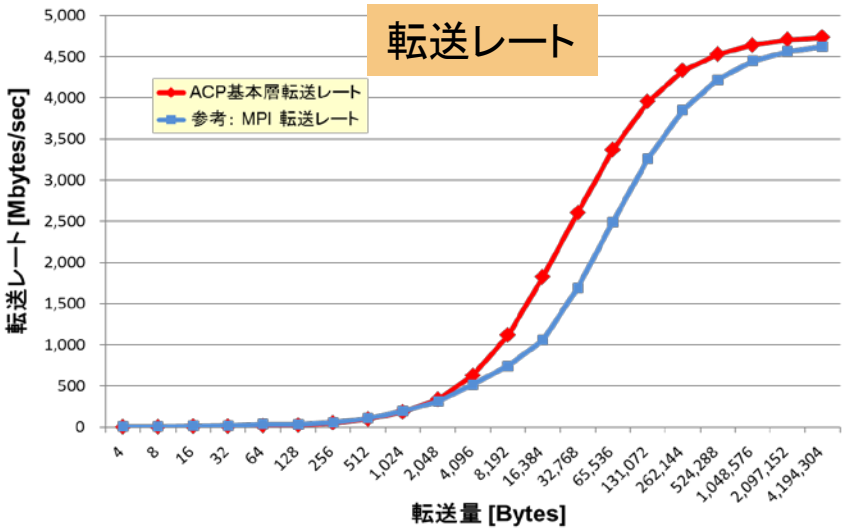


- 通信スレッドによる非同期非ブロッキングプロトコル処理
  - プロセス間コピー、不可分操作、メモリアクセス順序制御
  - 例）UDP

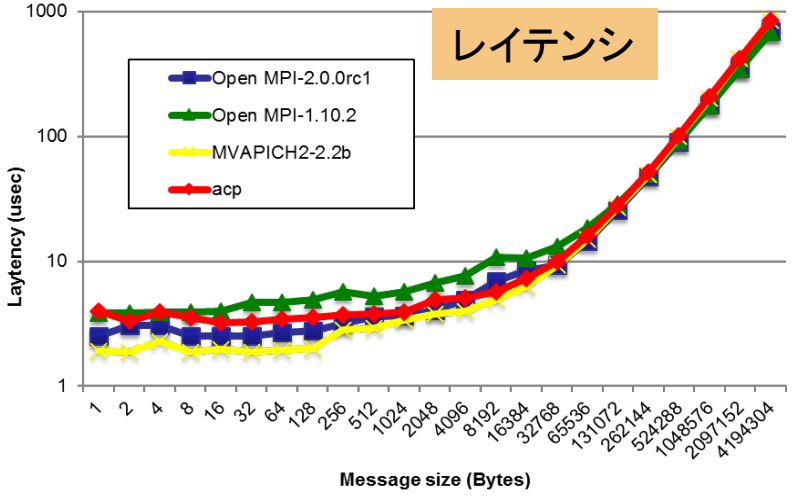
基本層通信性能

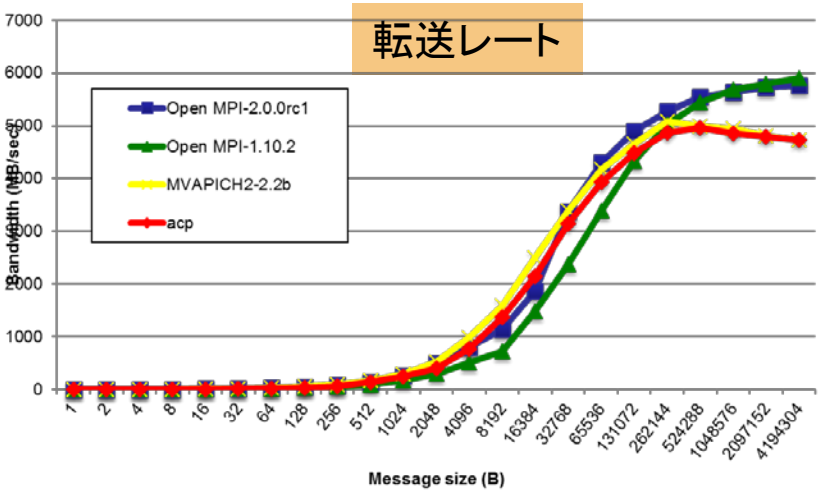
- Tofu版





- InfiniBand版

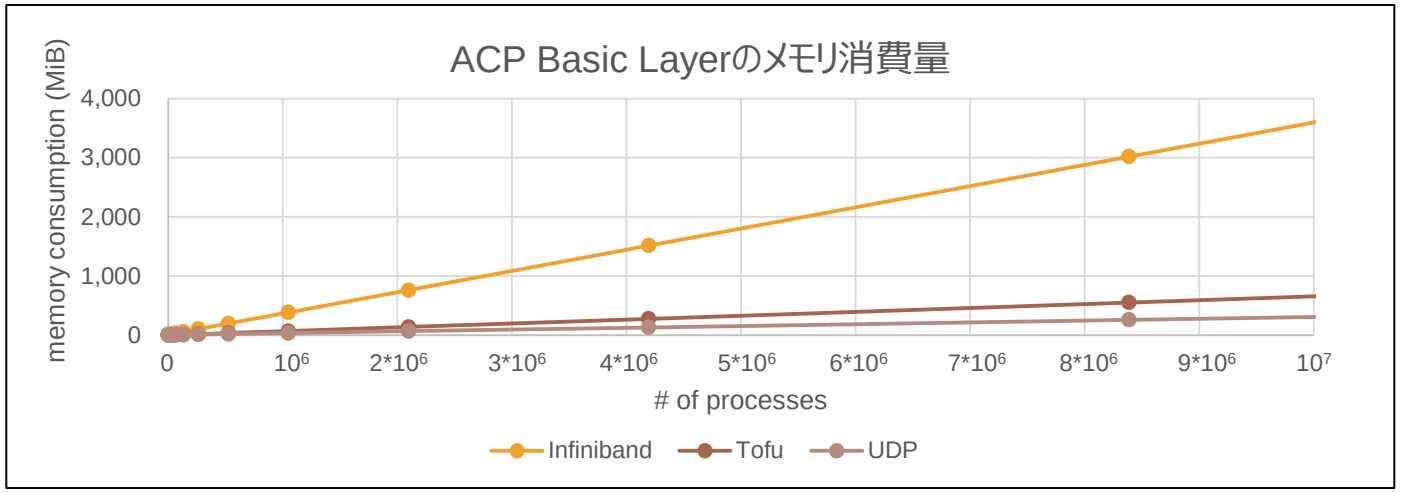




基本層メモリ使用量

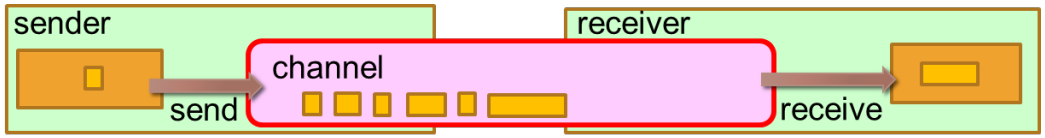
- 100万プロセスでの使用量

	InfiniBand	Tofu/Tofu2	UDP
ランク数に比例	359MiB@100万プロセス	66MiB@100万プロセス	31MiB@100万プロセス
メモリ登録数に比例		9KiB@128メモリ登録	
その他（固定分）	10MiB	512KiB	3MiB
合計	**約369MiB**	**約67MiB**	**約34MiB**

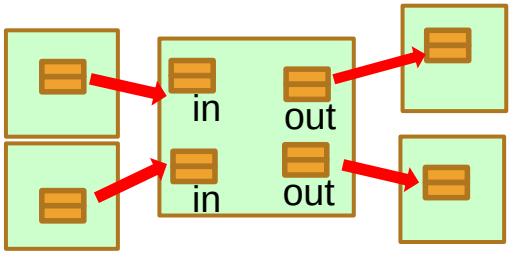


Communicationライブラリ

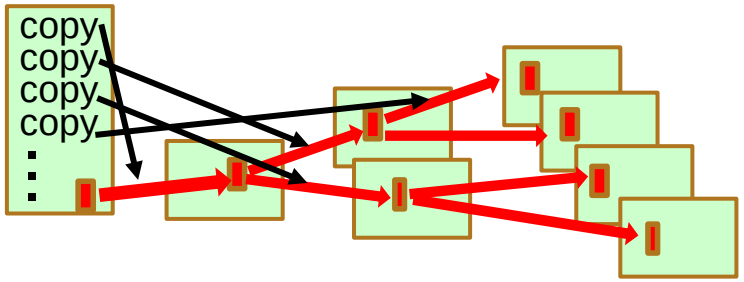
- 定型同期通信
  - 一対一、一対多、多対多
- Channelインタフェース
  - 一方向 Send/Recv通信
  - 役割（sender or receiver）に応じて必要最小限の通信用領域確保
  - 必要に応じて channel作成、解放



- Multi Channelインタフェース（実装中）
  - 固定バッファ間転送
  - ready命令、マルチバッファによるオーバラップ

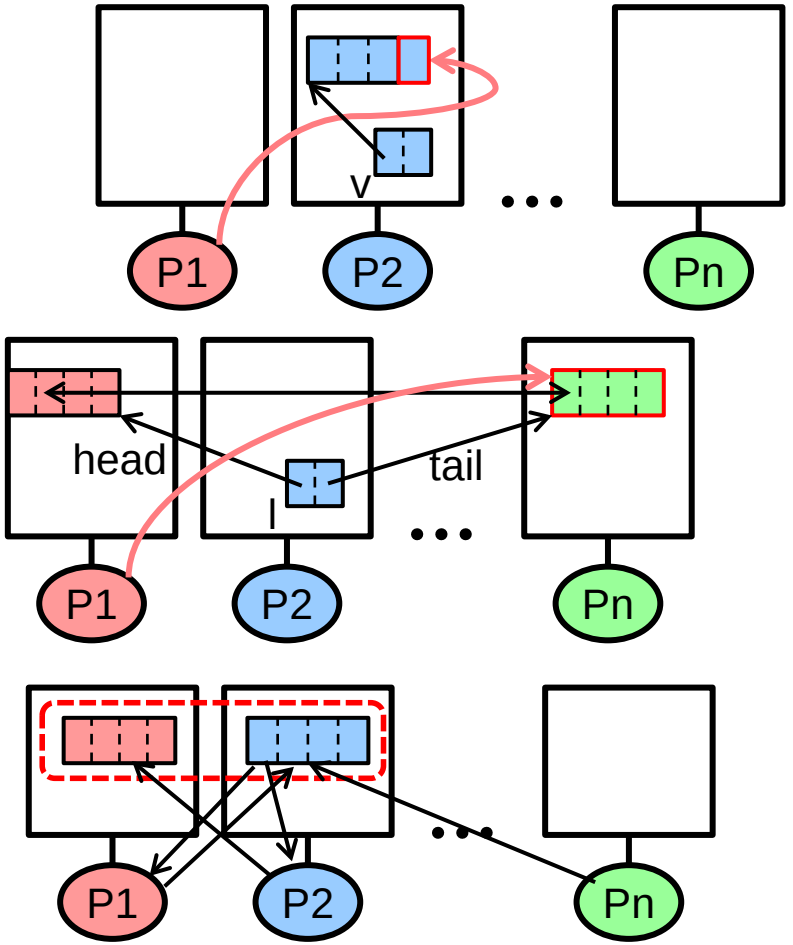


- 通信命令列インタフェース
  - 依存関係のある通信命令群の動的作成

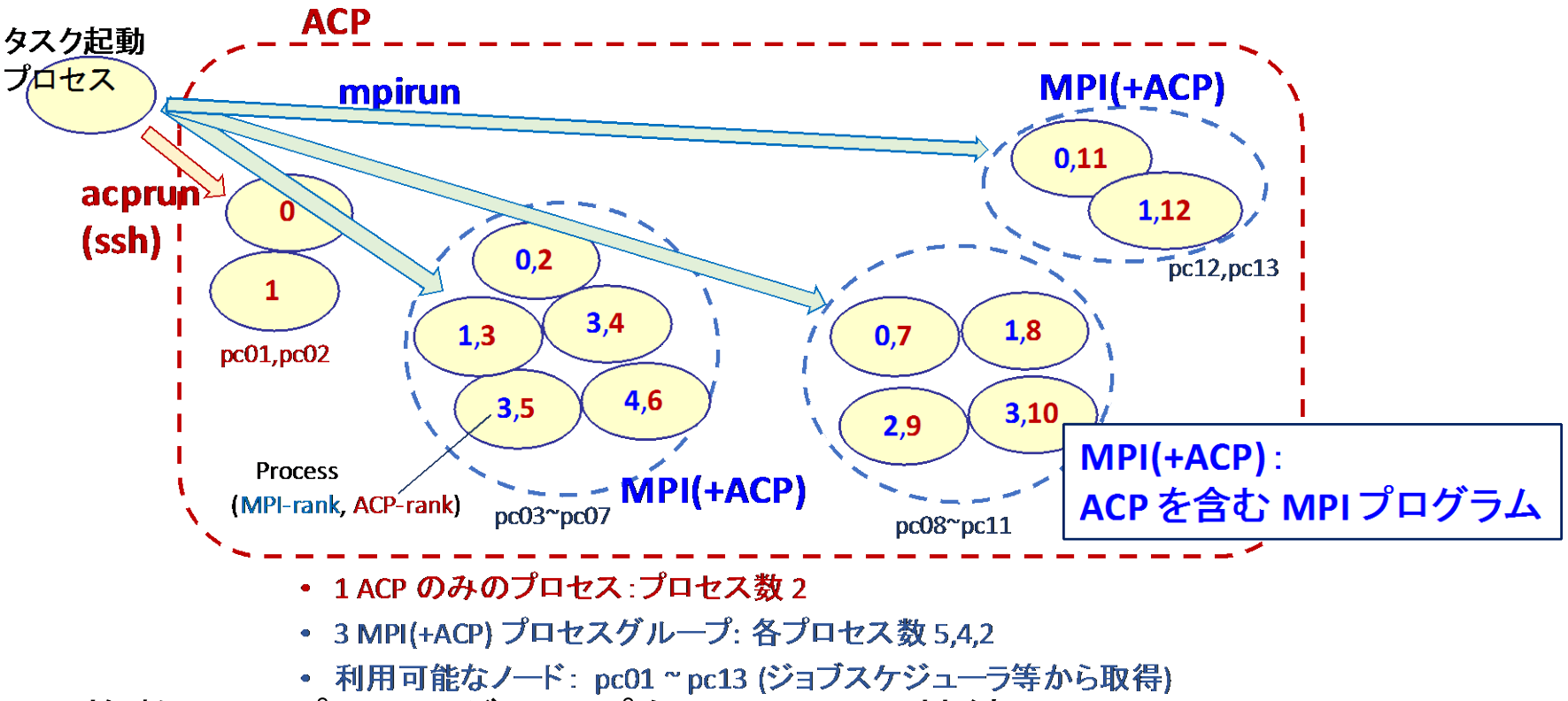


Dataライブラリ

- グローバルメモリアロケータ
  - サイズと割当プロセス番号を指定してメモリ割当て
- 遠隔データ構造: 可変長の連続データ
  - vector　　可変長一次元配列
  - deque　　双方向キュー
  - 先頭・末尾のデータ追加・削除が低コスト
- 分散データ構造: 要素単位で異なるプロセスに配置
  - list　　双方向リスト
  - set　　集合
  - map　　連想配列
- ワークスペース
  - データを分散格納する共有データストア



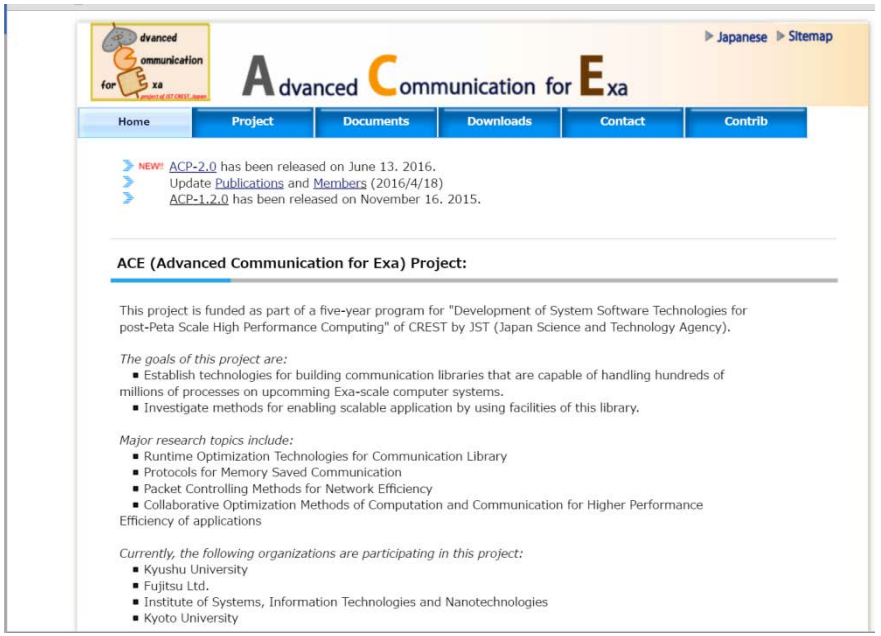
複数 MPI の ACP による連携 (ACP+MPI)



- 複数 MPI プロセスグループを ACP により接続
- 既存 MPI プログラムを活用しつつ，超高並列時の省メモリ性を確保
  - MPI-Split や MPI-Spawn，連成計算を対象
- mpirun と acprun を統合し起動する macprun コマンドをサポート

ACPライブラリの公開

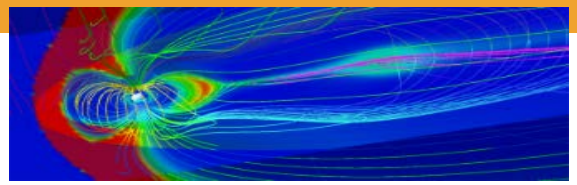
- 提供方法
  - tar ball をプロジェクト webサイト（http://ace-project.kyushu-u.ac.jp）で配布
- ライセンス
  - ソースコード：The BSD 3-Clause License
  - ドキュメント：CreativeCommons: 表示-継承
- ドキュメント
  - 配布 tar ball: OSS 準拠
  - Doc/Manuals：図入りの概要説明は pdf で準備。man3, 仕様書は doxygen で生成
- 告知・フィードバック
  - 告知は webサイト、bug 報告・修正情報は Mantis 上で提供
- ACP導入済みシステム
  - 九大FX10、東大FX10



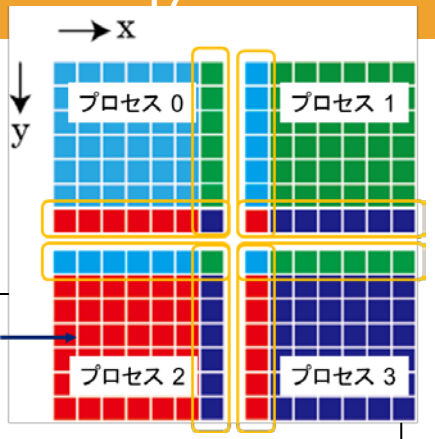
講演内容

- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- アプリケーション、ミドルウェアへの応用
  - 通信最適化、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- イベント・デモ

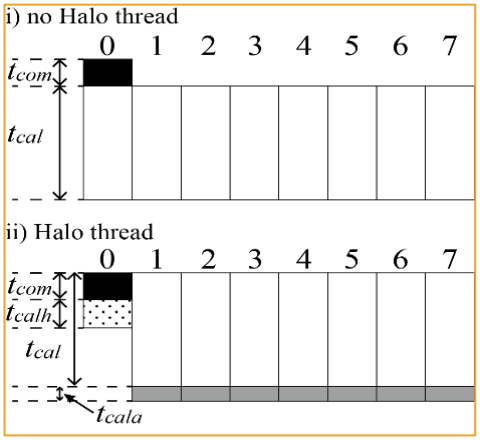
MHD計算における Halo通信の改良



地球磁気圏のシミュレーション



- Haloスレッド:
  - Halo通信とその通信に依存する計算を別スレッドで実行し、同期を削減

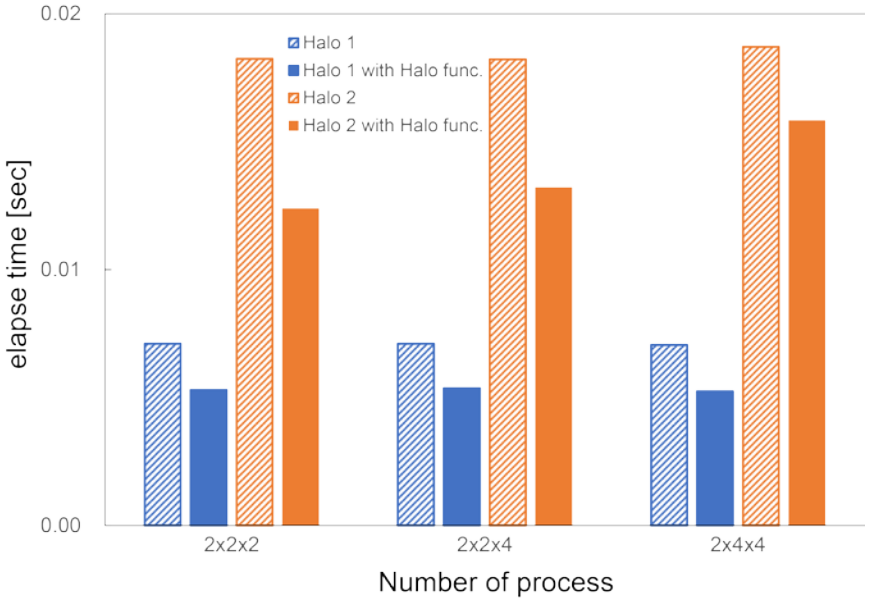


- Halo関数:
  - 通信ライブラリ非依存インタフェース
    - halo_init, halo_isend, halo_irecv、halo_wait
  - 通信, pack/unpackの実装効率化
  - Fortran wrapper

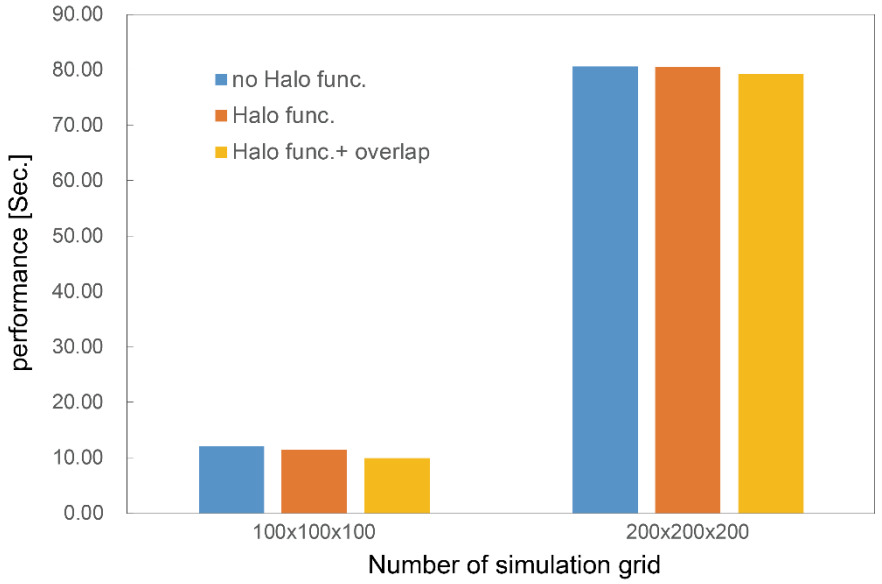
```
 call halo_init(f) ! Halo
!
!----Time evoluation---!
  do time = 1, 1000
!
!----Thread setting----!
!$OMP PARALLEL PRIVATE(myid,mylid,ks,ke,ii)
.
.
.
!----Halo thread----!
    if(myid == 0) then
      call boundary(f)    ! boundary setting
      do l = 1, 26
        call halo_irecv(f) ! Halo recieve
        call halo_isend(f) ! Halo send
      end do
!
      do l = 1, 26
        call halo_wait ! for receive
        do k = zs(l), ze(l)
          call mhd_calc(f) ! MHD calc. at Halo
        end do
      end do
!
      do l = 1, 26
        call halo_wait ! for send
      end do
!----Calc thread----!
    else
      do k = ks+1, ke-1
        call mhd_calc(f) ! MHD calc.
      end do
    end if
!
!$OMP END PARALLEL
.
.
.
  end do
```

FX100を利用した性能評価

Halo関数有り/無しの場合のHalo通信の性能



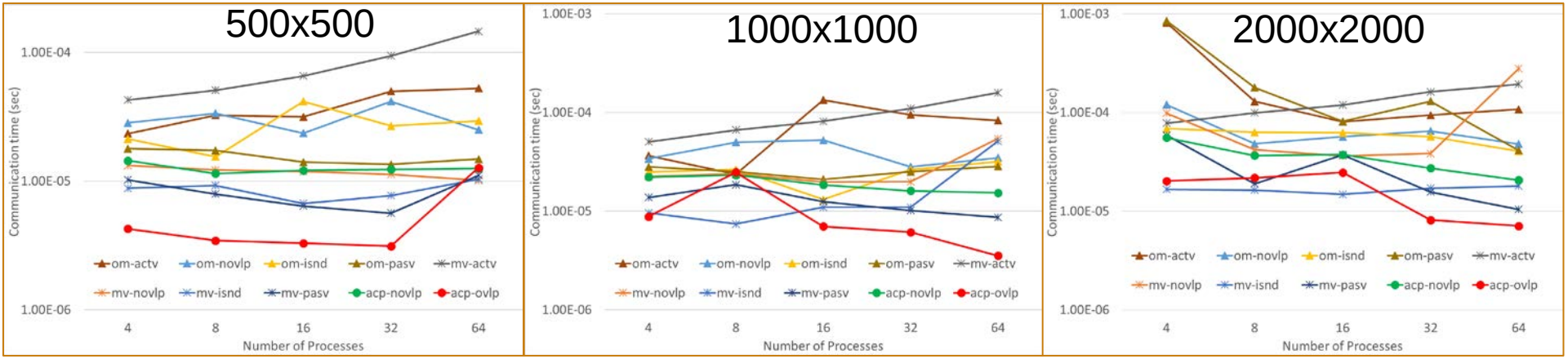
Halo関数有り無し、オーバーラップ時の計算性能



- MPI実装:
  - FX100でHalo関数を利用すると、通信が最大で1.3倍の性能向上
  - 計算時間全体では、Halo関数利用によりオーバラップ効果確認
- 今後: Halo関数の ACP実装
  - 省メモリ化、通信低オーバヘッド化

ACPによる Stencil計算の Halo通信性能予備評価

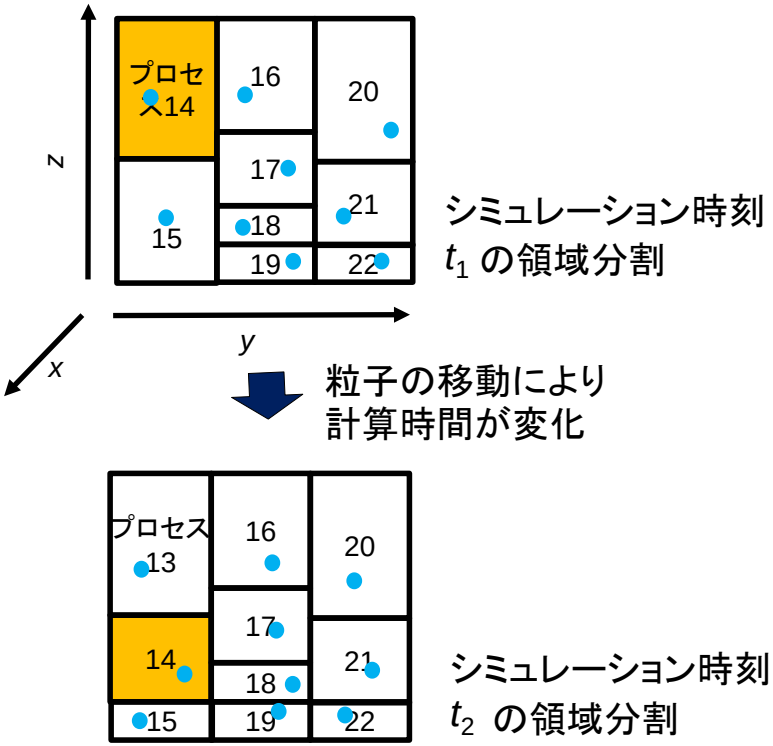
- 簡易版 2次元 stencil計算コード
  - 9点差分、1次元分割、pack/unpack
- Halo通信時間



- プロセス数に対して行列サイズが小さい場合に ACPの効果大
  - 片側通信オーバヘッド: 低
  - オーバラップ率: 高

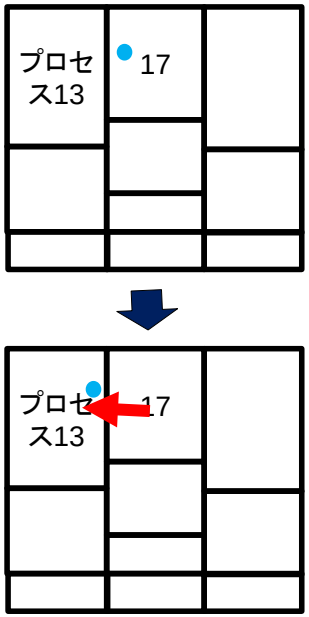
粒子系シミュレーションにおける不規則隣接通信の改良

■ 粒子系シミュレーション



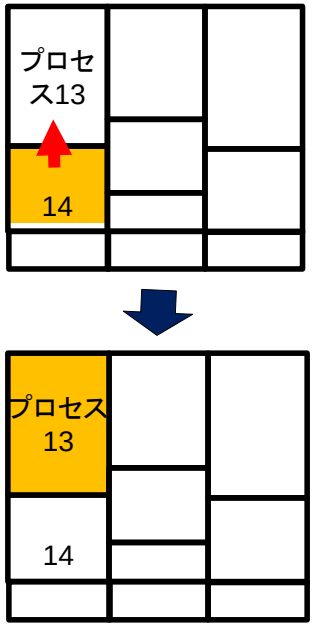
■ 粒子移動に伴う粒子データ通信

移動により担当プロセスが変わった粒子のデータを通信



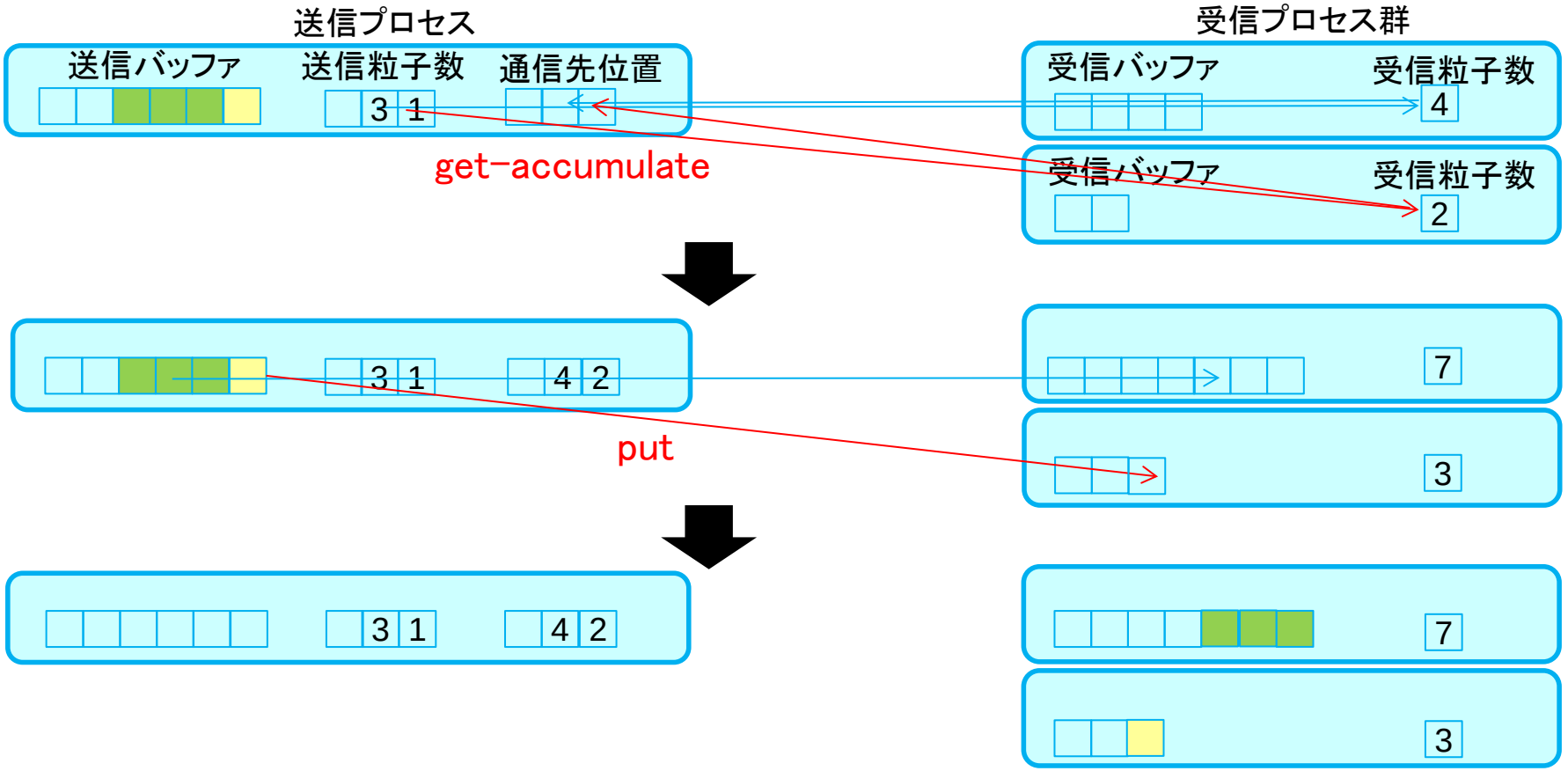
■ 領域分割に伴う粒子データ通信

領域分割により担当プロセスが変わった粒子のデータを通信



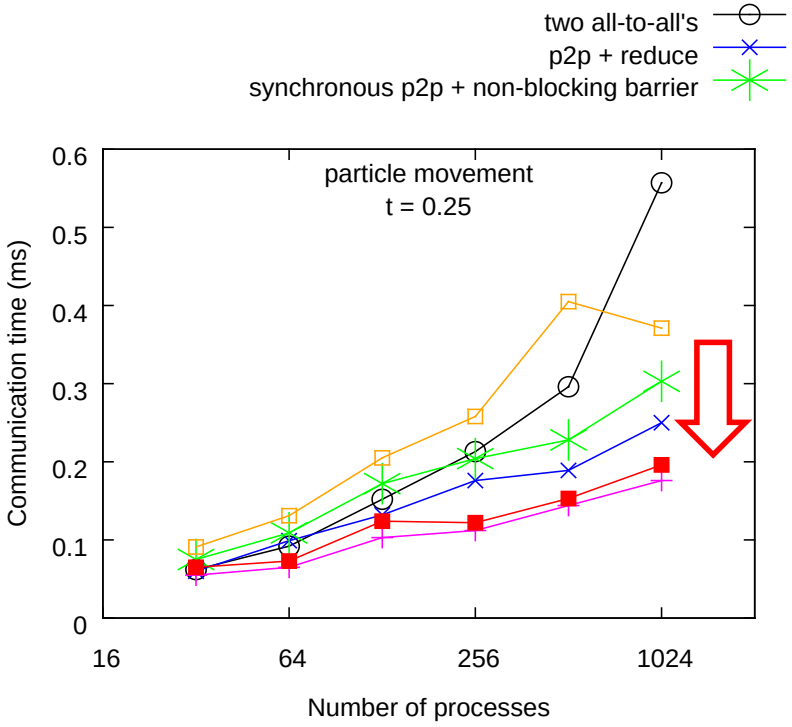
受信側は送信プロセスを予測できない→片側通信に適している

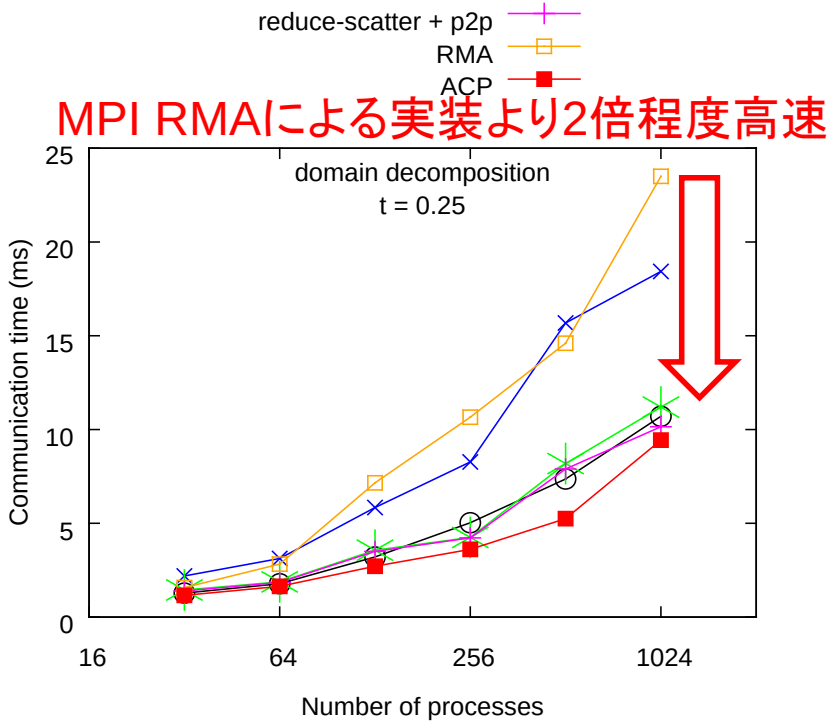
粒子データ通信の片側通信による実装



ACPによる実装とMPIの性能比較

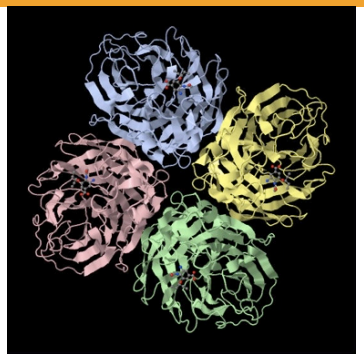
- 重力*N*体シミュレーションにおける粒子データ通信



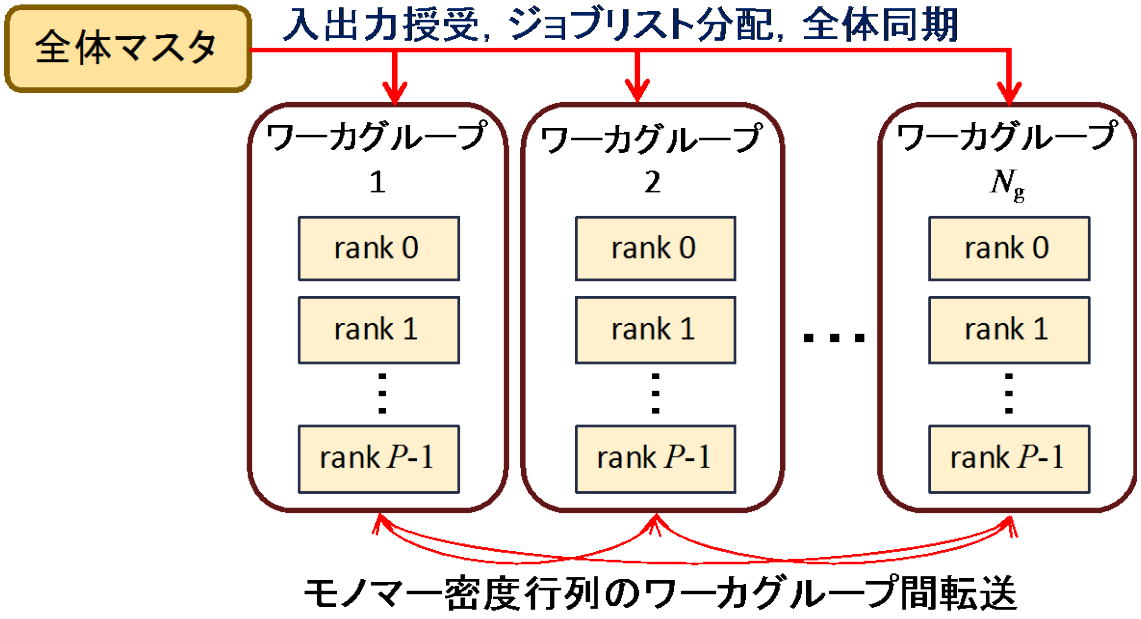


測定環境: Fujitsu PRIMERGY CX400　MPI: Intel MPI 5.1.3

OpenFMO

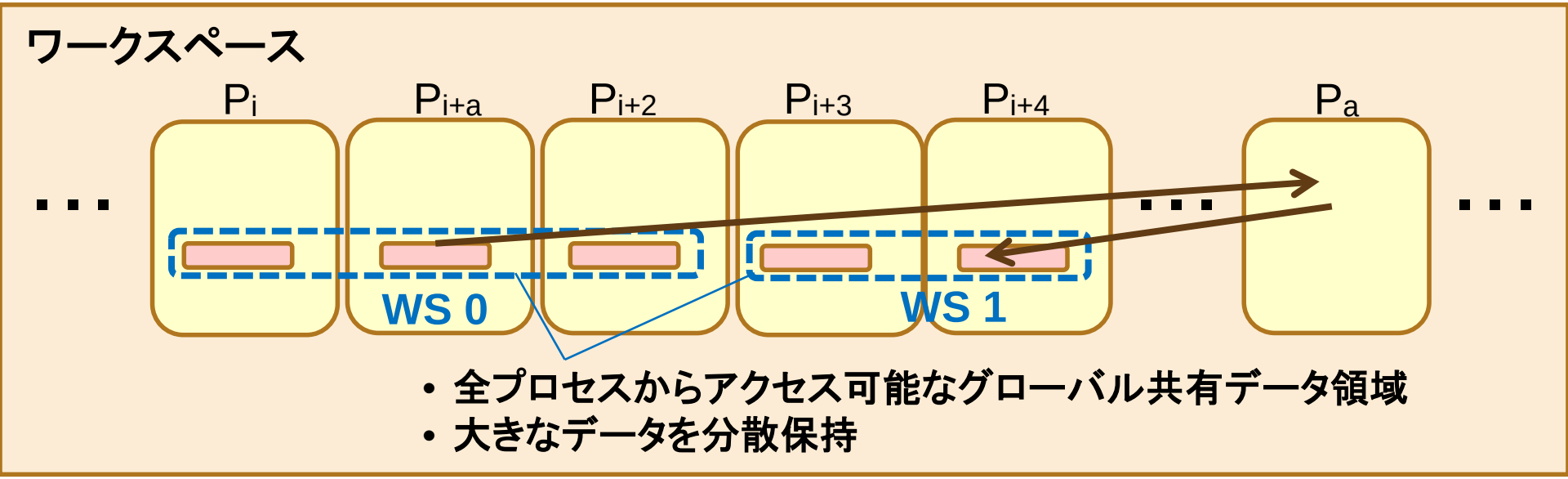


- 大規模分子の量子化学計算をタスク並列処理
- master / worker モデルの通信パターン
  - 粗粒度並列化部分
    - 全体マスター ⇔ ワーカグループ：ジョブ分配、同期
    - ワーカグループ間：データ共有
  - 細粒度並列化部分
    - ワーカグループ内：
      タスク分配、
      Bcast/Reduce

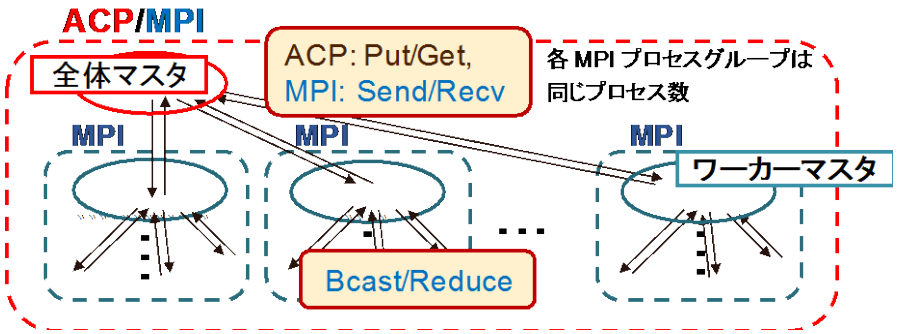


ACP の OpenFMOへの適用

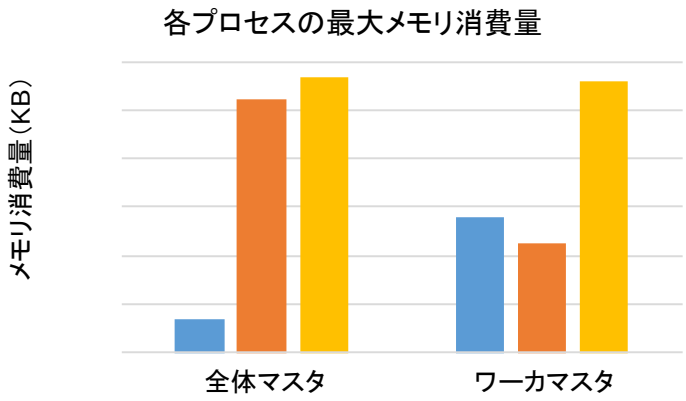
- 粗粒度並列化部分：
  - ACP + MPIによる省メモリワーカグループ実装
- 細粒度並列化部分：
  - ACPインタフェースを活用
    - チャネル、グローバルカウンタ、ワークスペース、Collective



メモリ消費量比較実験



- 超並列OpenFMO を模したプログラム
- 3種類の実装
  - ACP + MPI 実装
  - MPI-Spawn 実装
  - MPI-Split 実装



全プロセス数：1025
ワーカグループ内プロセス数: 128
ワーカグループ数：8

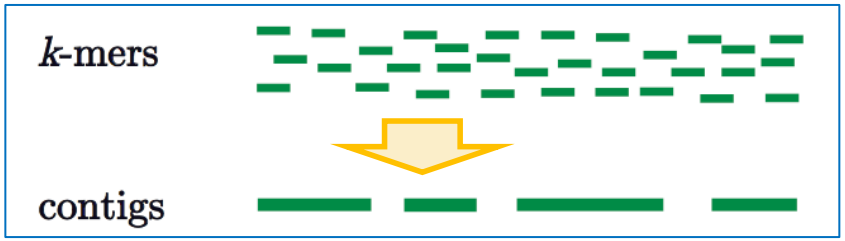
- メモリ消費量

  ACP+MPI < MPI-Spawn < MPI-Split

  ACP+MPI が最小
- 主要メモリ利用箇所
  - ACP+MPI
    - ワーカマスタ： MPI_Init, MPI_Bcast 関数が 36%, 41% を消費
  - MPI-Spawn
    - 全体マスタ: MPI_Spawn 関数が 91% を消費
  - MPI-Split
    - 全体マスタ, ワーカマスタ:
      両プロセスにおいて MPI_Init, MPI_Split 関数が 71%, 26% を消費

De novo ゲノムアセンブリ(遺伝子解析)

- ゲノムアセンブリ Trinity アプリケーション内の
  並列 Inchworm の Python 版プログラムを対象に ACP を適用
  - バイオインフォマティックスアプリ
  - Python プログラムとの DSL を含めたインターフェース開発
  - 大規模共有メモリ向けプログラムの分散メモリ環境での実装をサポート
    - ゲノムアセンブリプログラムにおける k-mer 辞書データを対象

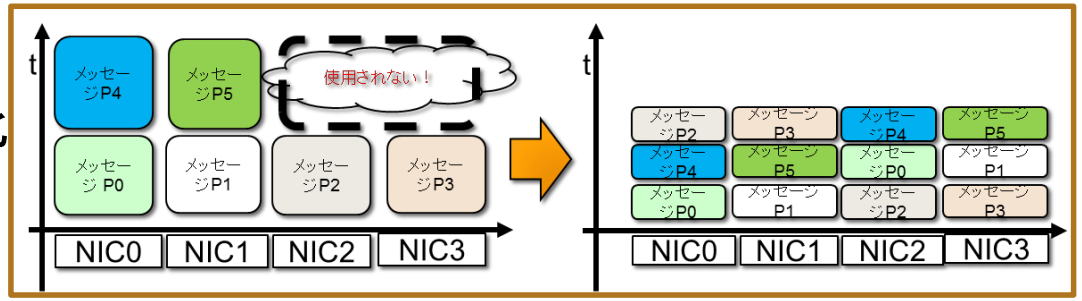


ゲノムアセンブリ操作 (k の長さの遺伝子断片の接続)

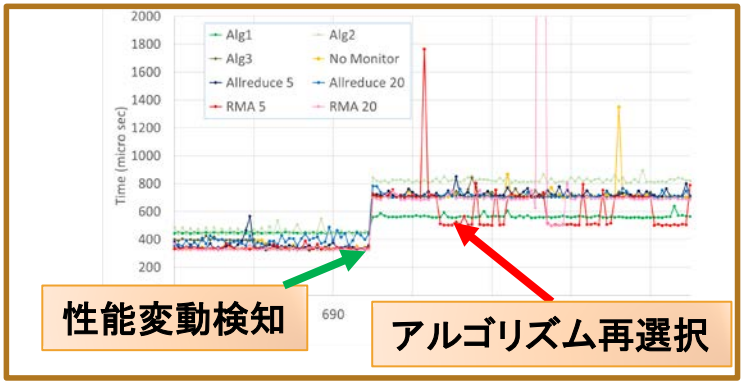
- 九州大学医学部大川研究室との共同研究

動的通信最適化技術

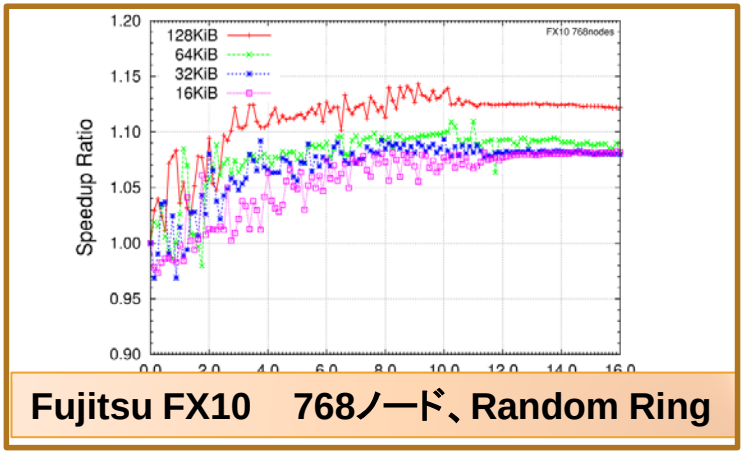
- 複数NICによる隣接通信
  - NIC毎の通信量自動均等化



- 集団通信アルゴリズム選択
  - 片側通信による性能モニタリングを用いた低オーバヘッド動的アルゴリズム選択技術

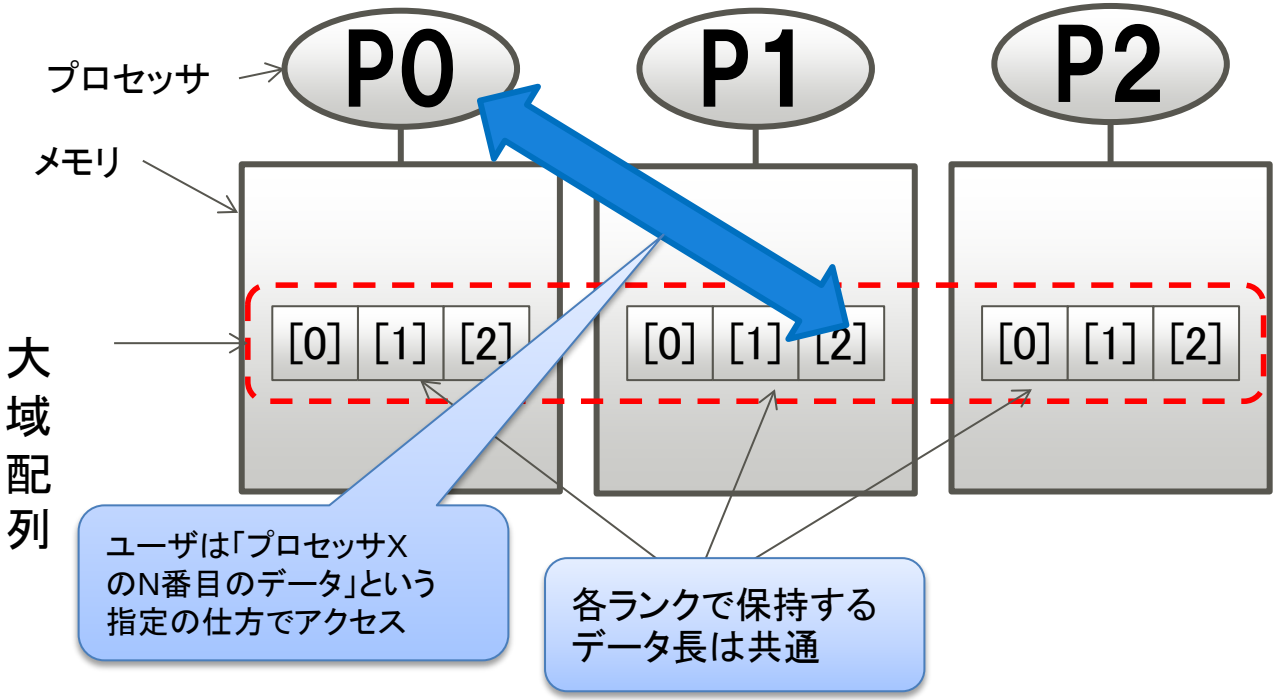


- パケットペーシング
  - 指定する間隔（パケット間ギャップ）を空けてパケットをネットワークに送出
    - パケットの停止・再送遅延を抑制



Ruby / Python向けグローバル配列

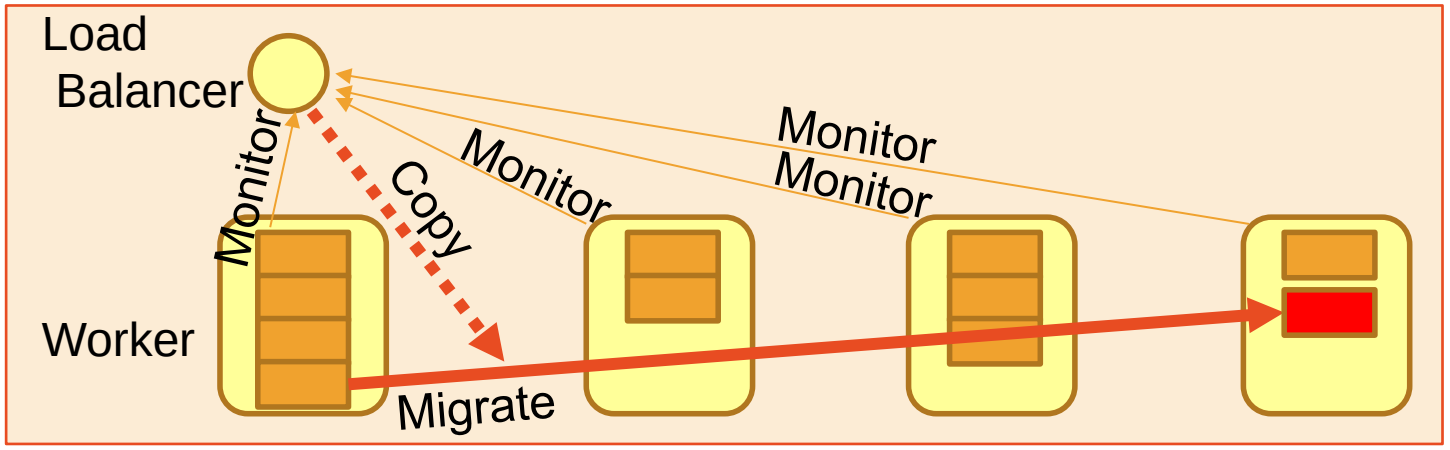
- Ruby/Pythonは文法が習得しやすく、既に多数のユーザがいる
- FortranのCoArray機能やXcalable MPのローカルビューモデルに準ずるインターフェースを提供



タスク並列言語処理系への適用

- ACPの非同期 remote-to-remoteコピー、不可分操作をworker間のタスク移動やデータコピーに利用

- 適用例: Load Balancer
  - 負荷バランスを監視し、タスクを適宜移動して均等化
  - workerと非同期に動作するため、workerは自分のタスクに集中
  - タスクが参照するデータのプリロードにもACPコピーを適用可



講演内容

- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- アプリケーション、ミドルウェアへの応用
  - 通信最適化、ACP + MPI、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- ACP普及に向けた活動

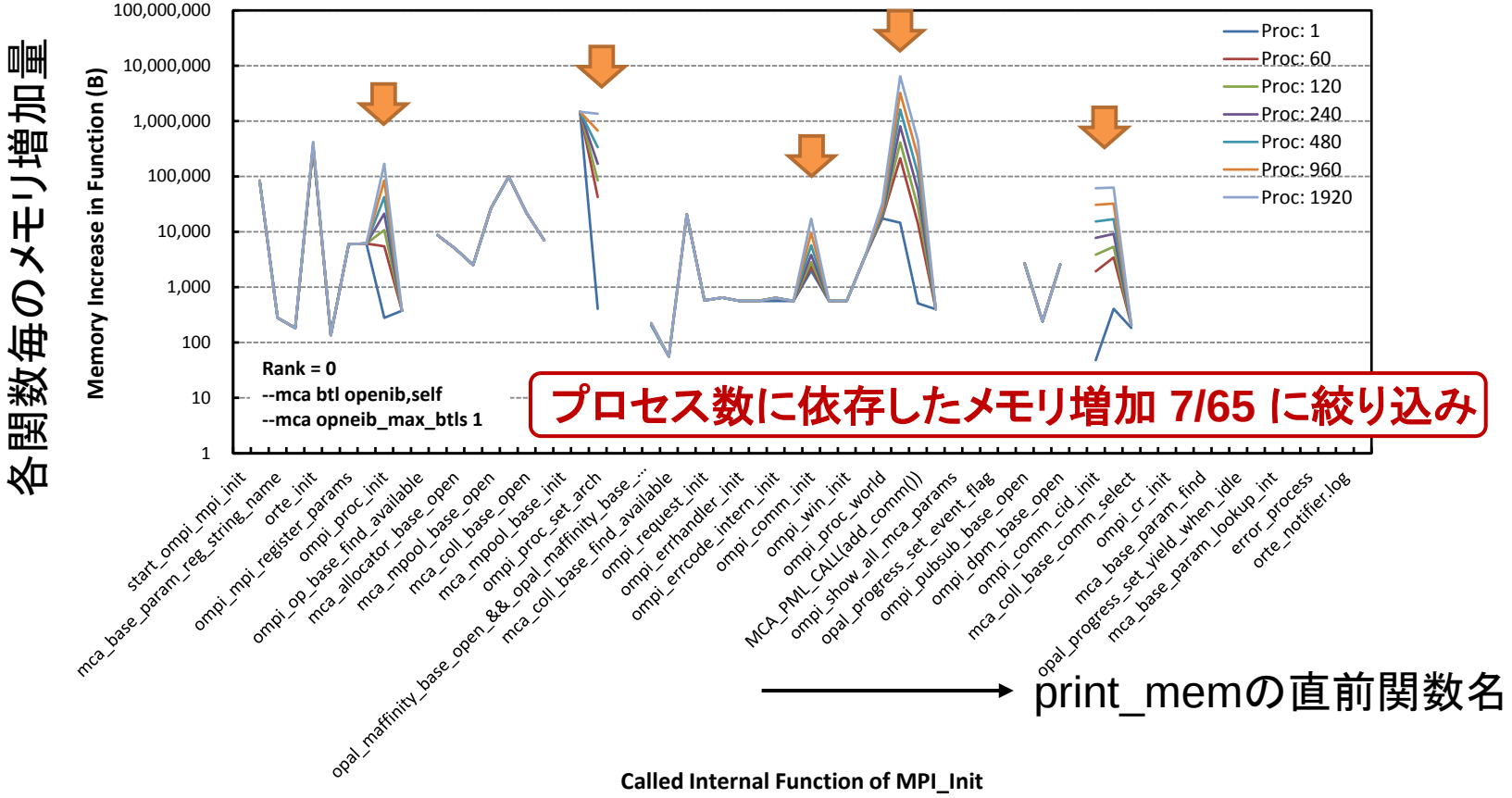
DMATP-MPI (Dynamic Memory Allocation Tracing Profiler for MPI)

- 機能
  - ライブラリ毎のメモリ使用量
  - メモリ使用量分析
  - メモリスナップショット(詳細分析)
- 基本動作
  - 初期化処理
    - malloc系関数のhook
    - 結果表示関数の登録
  - 分類・集計処理（hook関数）
    - オリジナル関数呼び出し
    - 動的メモリ獲得・開放量の取得
    - 動的メモリの分別，分類・集計
  - 結果出力
    - カテゴリ毎の集計値

DMATP-MPIの動作

DMATP-MPIによるOpen MPI1.4.6 解析結果

- MPI_Init関数内メモリ使用量調査(スナップショット利用)
- ompi_mpi_init 関数について print_mem を65個所に挿入
- プロセス数を変更(1 ～ 1920)して測定



Open MPIコミュニティに解析結果を開示、MPIライブラリの省メモリ化に貢献(2.0.0)

NSIM: インターコネトシミュレータ

Input files

- Interconnect Configuration File
- MGEN program (Evaluation program)
- Rank-Node Conversion table file

NSIM

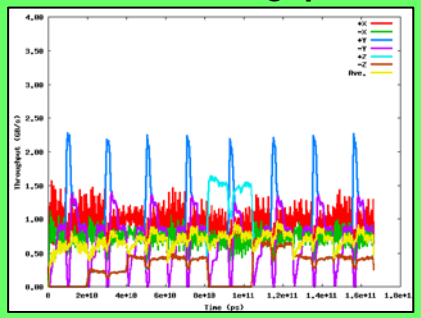
Output files

Simulation Example of an Alltoall on 4K-node 3D-Torus

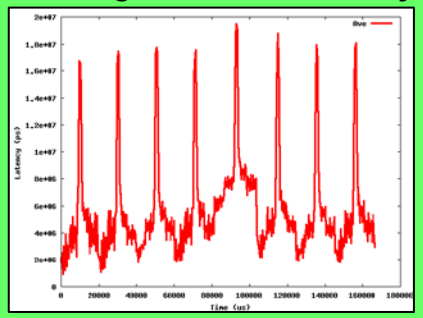
Performance Information

```
MaxSimTime                  166712548000 [Ticks]
SumArrivedPackets           71268352
SumArrivedPacketSize        139652431872 [Bytes]
SumSizeTransfer             1397206548480 [Bytes]
SumNumLinks                 12288
EffectiveBandwidth          8380.932121 [GBytes/10^12Ticks] (SumSizeTransfer/M
EffectiveLinkBandwidth      0.682042 [GBytes/10^12Ticks/Links] (SumSizeTransfer/
IdealLinkBandwidth          4.000000 [GBytes/10^12Ticks/Links]
LinkBandwidthEfficiency     17.051050 [%] (EffectiveLinkBandwidth/IdealLinkBand
MinSimStartTime             1290126348.243068
MaxSimEndTime               1290127795.179783
SimDuration                 1446.936715
SumLinkBusyTime             382164407752000 [Ticks]
LinkBusyRate                18.655236 [%] (CumulLinkBusyTime/SumNumLinks/MaxSim
SumNumWaitPacketsInRt       372972599
SumNumThruPacketsInRt       340059081
CumulPacketWaitTimeInRt     328675643082000 [Ticks]
SumNumWaitPacketsInNi       25432757
SumNumThruPacketsInNi       45835595
CumulPacketWaitTimeInNi     93268741998000 [Ticks]
CumulNetworkLatency         370885137344000 [Ticks]
AveragePacketWaitTime       881232.680265 [Ticks] (CumlPacketWaitTimeInRt/SumNum
AverageAllPacketWaitTime    460955.175346 [Ticks] (CumlPacketWaitTimeInRt/(SumNu
:
```

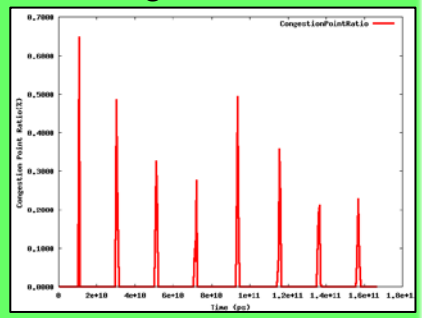
Link Throughput



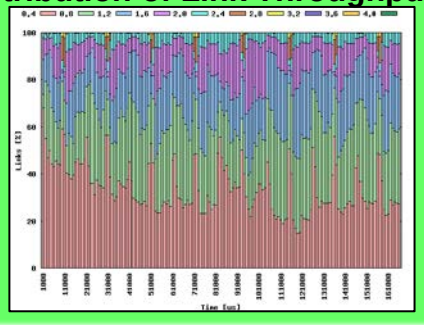
Average Network Latency



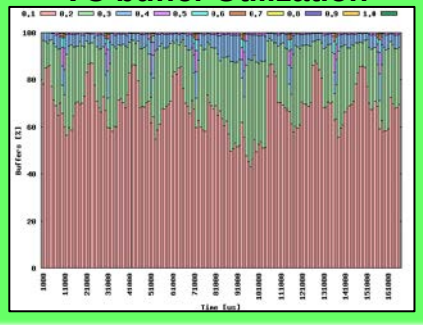
Congestion Rate



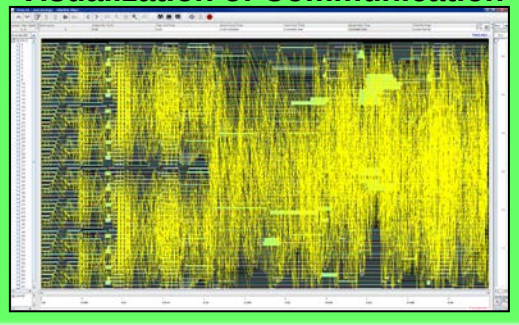
Distribution of Link Throughput



VC buffer Utilization



Visualization of Communication

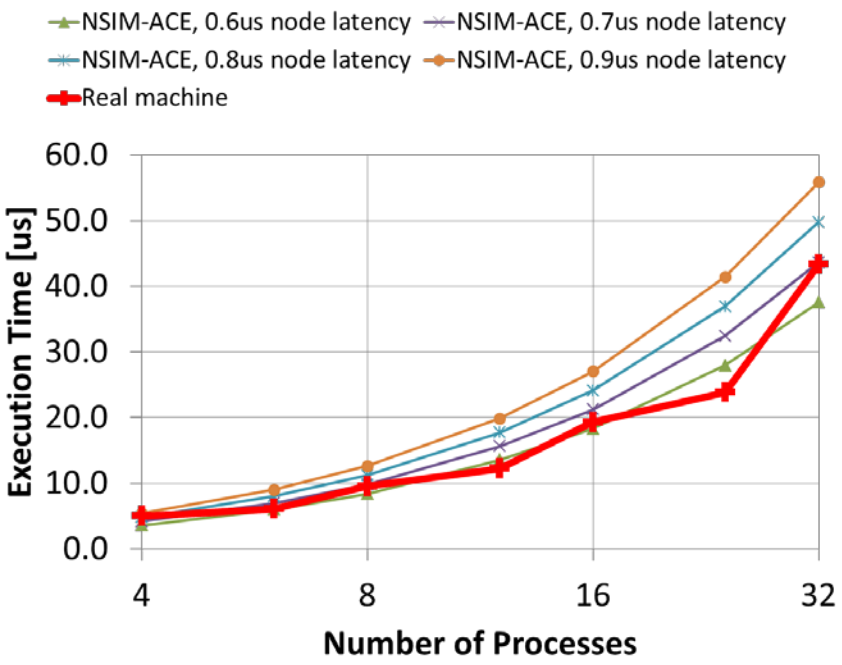


RMAシミュレーション機能の実装

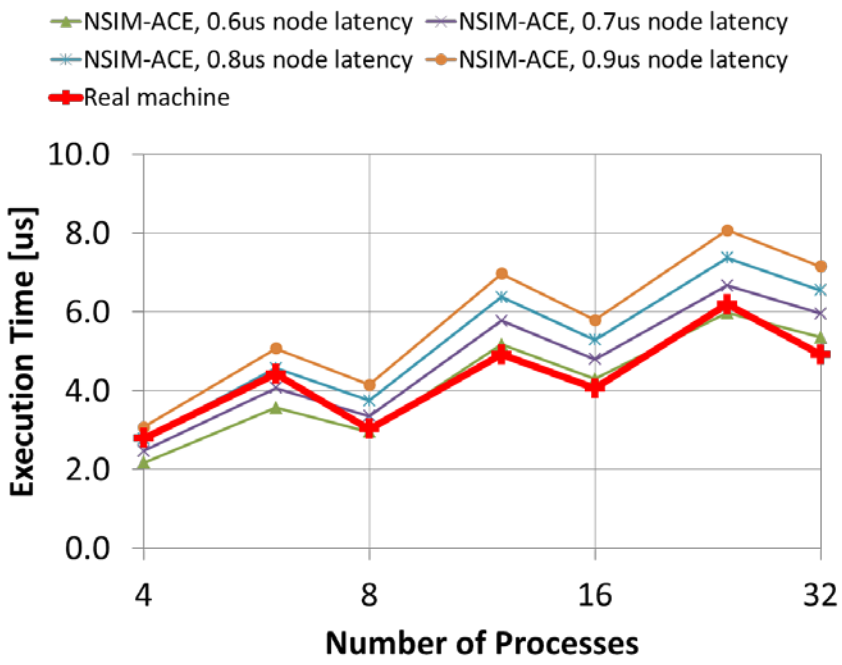
- 実装したMGEN関数
  - **MGEN_rdma_put()**
    - データ転送の発行
  - **MGEN_rdma_poll()**
    - 受信側でデータ転送完了を待つ
    - 実機で受信領域をポーリングする処理に対応する
  - **MGEN_acp_complete()**
    - 受信側からのACKパケットを待つ

NSIM-ACEのシミュレーション精度

- RDMA通信による同期
  - シミュレーションパラメータ：実機計測値を使用
  - 実機で観測不可能なパラメータは較正値を使用
- アルゴリズムの挙動を表す十分な精度



Ring algorithm



Recursive doubling algorithm

講演内容

- ACE（Advanced Communication for Exa）プロジェクト概要
- ACP（Advanced Communication Primitives）ライブラリ紹介
- アプリケーション、ミドルウェアへの応用
  - 通信最適化、ACP + MPI、Ruby / Pythonインタフェース、タスク並列言語処理系
- 性能解析ツール
  - メモリ量計測、通信時間予測
- イベント・デモ

イベント

- 2014
  - SC14 チュートリアル
    - 九州大学ブースで開催
- 2015
  - ISC2015 BoF
  - SC15 チュートリアル
- 2016
  - HPCS2016オーガナイズドセッション
    「PGASプログラミング環境」共同開催（2016年6月）
    - Xcalable MP、ACPを中心に PGAS環境を紹介
  - ICDCS2016デモ（2016年6月）
    - Raspberry Pi2クラスタ上の ACP実行
  - SC16 チュートリアル ＆ デモ

ACP Raspberry Pi 2でのデモ

SC16九大ブースでも開催予定
（ACP on Rasp. Pi 3）

- 実験目的
  - ACP の省メモリ性
    - 使用メモリ量の評価
  - ACP のポータビリティ
    - ARM での動作
  - 省電力な環境での動作
    - 消費電力の評価

図. Raspberry Pi 2 Model B

表. Raspberry Pi クラスタの仕様

	Raspberry Pi 2 Model B	
計算ノード	CPU	ARM Cortex-A7　900 MHz
	メモリ	1GB
	ネットワーク	10/100 Mbps イーサネット
	電源	4.5 – 5.5 W
コア数　（ノード数）		16コア　(4ノード)

- 実験結果
  - Raspberry Pi 2 Model B上でACP動作確認。
    - ARMプロセッサにおいても問題なく動作。
  - 分子軌道法プログラムを実行。
    - 16プロセスでメモリ使用量、消費電力の確認。

表. メモリ使用量と消費電力（16プロセス時）

	メモリ使用量	消費電力
ACP	5788KB	11-13w
Open MPI	10488KB	11-13w

まとめ

- ACEプロジェクト:
  数百万プロセスでの利用に耐えるスケーラブルな
  通信ライブラリ、およびアプリケーション開発

- 開発ソフトウェア:
  - 通信ライブラリ: ACP
  - アプリケーション: Halo通信関数、粒子系シミュレーション、量子力学、ゲノムアセンブリ
  - 各種言語向けインタフェース: Ruby、Python
  - 各種ツール: DMATP-MPI、NSIM-ACE

今後の展望

- タスク並列言語 Tascellへの適用
  - 分散メモリ環境向け言語処理系の通信層として適用を検討中
  - 開発元の八杉研究室（九州工業大学）、平石研究室（京都大学）と協議開始
- 開発した各ソフトウェアの GitHub化
  - 開発に関わったメンバーを中心に管理を継続