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

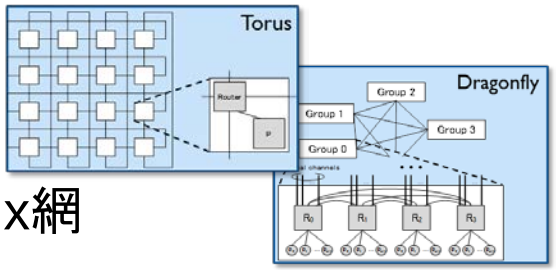
九州大学:
南里豪志、高見利也、本田宏明、薄田竜太郎、
森江善之、小林泰三
富士通株式会社:
住元真司、安島雄一郎、志田直之、佐賀一繁、野瀬貴史
公益財団法人九州先端科学技術研究所:
柴村英智、曽我武史
京都大学:
深沢圭一郎

2014年10月16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

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
  - インターコネクト：多次元トーラスもしくは high-radix網
  - 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

全体スケジュール

要素技術開発

- 省メモリ低オーバヘッド通信、およびデータ構造遠隔操作技術
- アプリケーションの通信パターン調査、および通信効率化
- 通信の自動最適化技術

ライブラリ構築

- ACP Basic Layer
- ACP Middle Layer
- Performance Estimation Tool
- 評価 / フィードバック

2014.9 ACPライブラリ公開

2011 2012 2013 2014 2015 2016 年度

全体スケジュール

要素技術開発

- 省メモリ低オーバヘッド通信、およびデータ構造遠隔操作技術
- アプリケーションの通信パターン調査、および通信効率化
- 通信の自動最適化技術

ライブラリ構築

- ACP Basic Layer
- ACP Middle Layer
- Performance Estimation Tool
- 評価 / フィードバック

2014.9 ACPライブラリ公開

2011 2012 2013 2014 2015 2016 年度

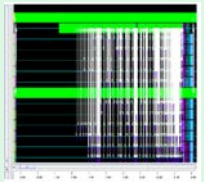
省メモリ低オーバヘッド通信、およびデータ構造遠隔操作技術

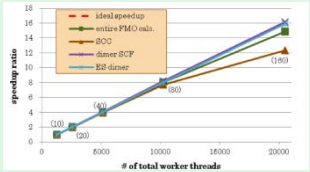
- 省メモリ低オーバヘッド通信のための既存MPI通信ライブラリの使用メモリ量解析 : [秋元他HPC138研究会、住元他,HPC138研究会]
  - Unexpected Messageによるメモリ使用問題： 明示的な資源利用と使用メモリ量を意識したプログラムインターフェイスの必要性を導出
  - MPI_Init(初期化関数)によるプロセス数に依存したメモリ使用量問題：全プロセスが持つべきデータ構造の分散配置のためのグローバルデータ構造処理の必要性を導出

- データ構造遠隔操作技術： [Y. Ajima, et al PGAS2013, 安島他 HPC140研究会]
  - 片側通信と遠隔ATOMIC操作によるグローバルデータ構造処理技術の確立（非同期グローバルヒープ,グローバルメモリ管理モデル）

アプリケーションの通信パターン解析、改良

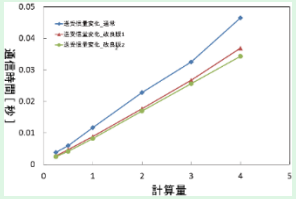
- FMO計算
  - 動的負荷分散の有無に対する通信パターンの詳細解析 [稲富、他、2012]
  - ハイブリッド並列化ならびに動的負荷分散による並列化効率向上 [稲富、他、2013]


通信パターン解析

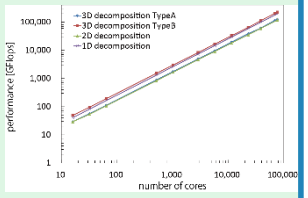


並列化効率評価

- 電磁流体計算(stencil計算)
  - 袖領域の幅調整による通信時間の隠ぺい効果を確認 [深沢、他、2012]
  - pack/unpack関わる並列性能劣化、領域分割の次元数の違いによる計算性能を確認[Fukazawa et al., 2013]

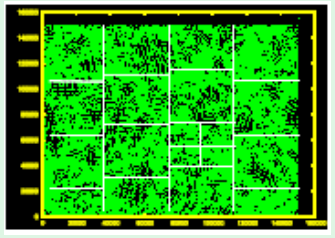


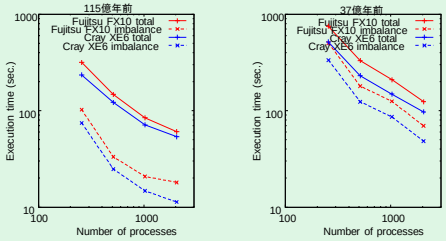
袖領域通信性能評価



領域分割性能評価

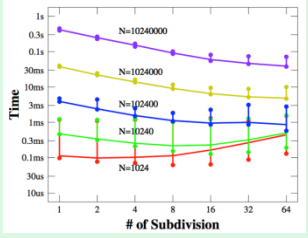
- 不規則格子による多体問題
  - 多体問題のスケーラビリティ調査 [薄田、2013]
  - 負荷不均衡による待ち時間を除いた純粋通信時間[Susukita, 2014]





スケーラビリティ調査

- 時間並列計算(1次元片方向隣接)
  - 並列パイプラインによる通信時間隠蔽[高見、他、2014]
  - パイプライン通信を有効に利用するアルゴリズムの提案 [Takami et al., 2014]

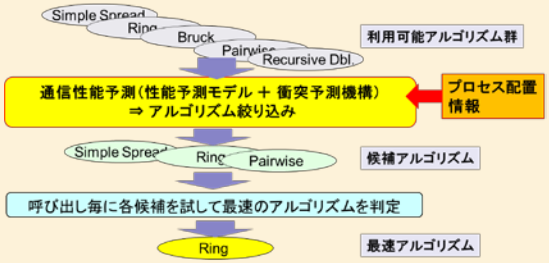


データ分割による高速化

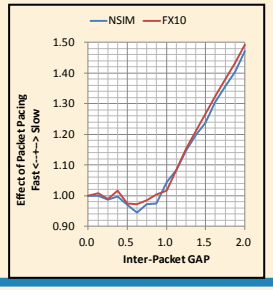
通信の自動最適化技術

- 隣接通信
  - 通信デバイス数と通信相手数に応じた隣接通信アルゴリズムの提案 [Morie, et al., 2013]

- 集団通信
  - プロセス配置に応じたアルゴリズム選択技術の効果を確認 [南里、他、2013]



- 通信間隔制御
  - トポロジと通信パターンに応じたパケット間隔制御の実機での効果を確認 [Shibamura, et al., 2014]



- ヒントインタフェース
  - プログラム情報を通信ライブラリに提供するヒントインタフェースを提案し、効果を確認 [Nanri, et al., 2014]

全体スケジュール

要素技術開発

- 省メモリ低オーバヘッド通信、およびデータ構造遠隔操作技術
- アプリケーションの通信パターン調査、および通信効率化
- 通信の自動最適化技術

ライブラリ構築

- ACP Basic Layer
- ACP Middle Layer
- Performance Estimation Tool
- 評価 / フィードバック

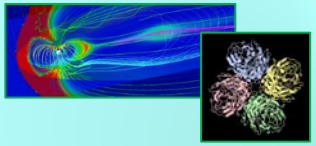
2014.9 ACPライブラリ公開

2011 2012 2013 2014 2015 2016 年度

通信ライブラリ ACP (Advanced Communication Primitives)

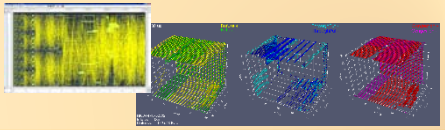
Applications

ACPライブラリを活用した
スケーラブルなアプリケーション



Performance Estimation Tool

ACP向け通信性能
予測ツール



ACP Library

Middle Layer

Channel Lib.
通信チャネルインタフェース
1D channel / collective /
neighbor ...

Data Lib.
分散データ構造インタフェース
vector / list / queue / set /
map / counter ...

Basic Layer

低レベル通信レイヤ（グローバルアドレススペースモデル).
copy / atomic / memory registration / sync / ...

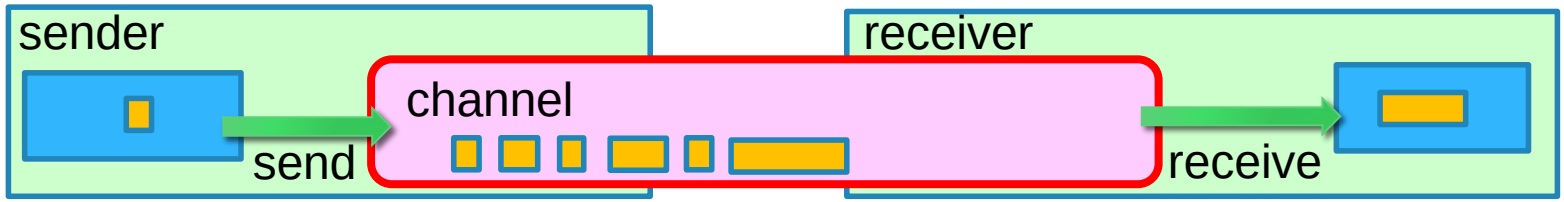
インターコネクトネットワーク （InfiniBand, Tofu, Ethernet)

通信チャネルインタフェース

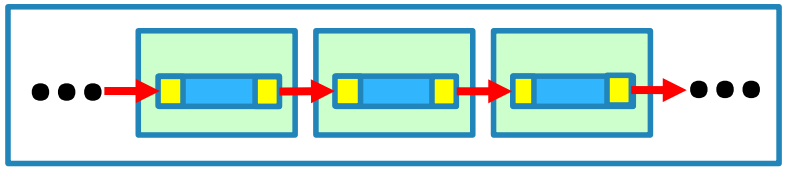
- プロセス間の通信路（チャネル）の確立と破棄を明確化
  - 不要となった通信用メモリ領域の解放による省メモリ化
  - 通信路確立操作（バッファ確保、メモリ登録、アドレス交換、等）を通信自体から分離することによるオーバヘッド隠蔽
- 通信パターン毎に独立したインタフェース設計
  - 個々の通信インタフェース実装の省メモリ化、低オーバヘッド化
- 通信パターン
  - 片方向チャネル
  - 集団通信
  - 隣接通信

片方向チャネル

- 2プロセス間の片方向メッセージパッシング通信



- 特徴:
  - 片方向 ⇒ 役割(sender or receiver)に応じて必要最小限のバッファ確保が可能
  - in order ⇒ MPIのタグのような out-of-orderのメッセージ管理が不要
- プログラム例
  - 片方向一次元シフト通信



```
ch0 = acp_create_ch(left, myrank);
ch1 = acp_create_ch(myrank, right);

for (...){
  req0 = acp_nbsend(ch0, addr0, size);
  req1 = acp_nbrecv(ch1, addr1, size);
  acp_wait_ch(req0);
  acp_wait_ch(req1);
  calc();
}

req0 = acp_nbfree_ch(ch0);
req1 = acp_nbfree_ch(ch1);
acp_wait_ch(req0);
acp_wait_ch(req1);
```

チャネル生成

通信

計算

チャネル解放

MPIに対するチャネルインタフェース提案

- MPI Forum（9月）でチャネルインタフェースを紹介
  - 主なコメント:

    「まだ、単に省メモリの問題をプログラマに押し付けるだけに見える。
    実アプリケーションでの効果を示すことが重要。」

    「効果が期待できるアプリケーションとしては、Adaptive Mesh Refinementの他、
    Data Analysis関連も考えられる。」

    「Collective通信や隣接通信等への応用も検討すべきである。」

    「他のインタフェースとの関係を考慮しながら、議論を進めるべきである。」

- インタフェース提案の狙い:
  - 本プロジェクトにおける成果の社会への貢献の一環
  - より広い利用者からのフィードバック

- MPI vs ACPライブラリ
  - ACPは既存のインタフェースにとらわれず、省メモリ低オーバヘッドを追求した通信ライブラリとして設計

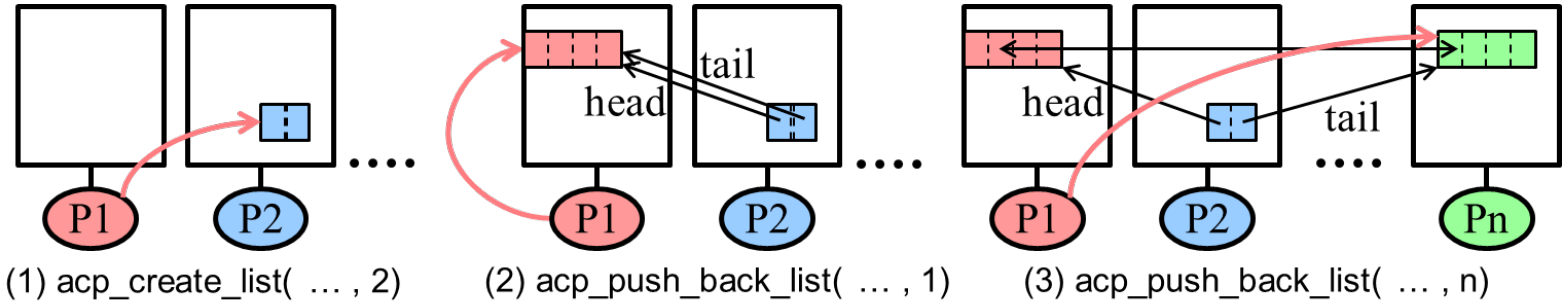
分散データ構造

- データ構造を複数プロセスに分散
  - 不要なコピーを減らし、グローバルなデータ配置を最適化
  - データ生成時に配置を明示的に指定
- データの生成，操作，破棄は非同期
  - 計算と通信のオーバーラップを促進する
- データ構造の型
  - vector　　可変長一次元配列
  - list　　双方向リンクトリスト
  - deque　　双方向キュー
  - map　　連想配列
  - set　　集合

C++言語の標準テンプレートライブラリ(STL)を参考にした

API例

- メモリアロケータ関数
  - データの生成、破棄で内部的に使用、ユーザーも使用可能
- ベクタ関数
  - 単一プロセスに配置される、可変長配列
- リスト関数
  - 要素毎に異なるプロセスに配置できる、双方向リスト

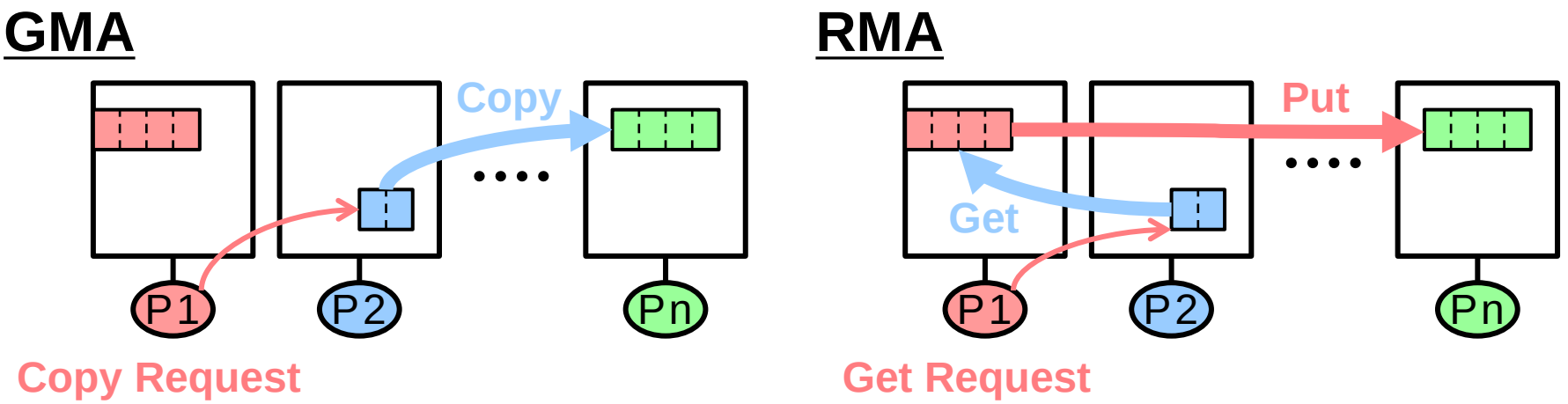


(1) acp_create_list(… , 2)　(2) acp_push_back_list(… , 1)　(3) acp_push_back_list(… , n)

名称	定義
メモリ割当	acp_ga_t **acp_malloc**(size_t *size*, int *rank*);
メモリ解放	void **acp_free**(acp_ga_t *ga*);
ベクタ生成	acp_vector_t **acp_create_vector**(size_t *nelem*, size_t *elsize*, int *rank*);
ベクタ末尾要素追加	void **acp_push_back_vector**(acp_vector_t *vector*, acp_ga_t *ga*);
リスト生成	acp_list_t **acp_create_list**(size_t *elsize*, int *rank*);
リスト先頭要素追加	void **acp_push_front_list**(acp_list_t *list*, acp_ga_t *ga*, int *rank*);

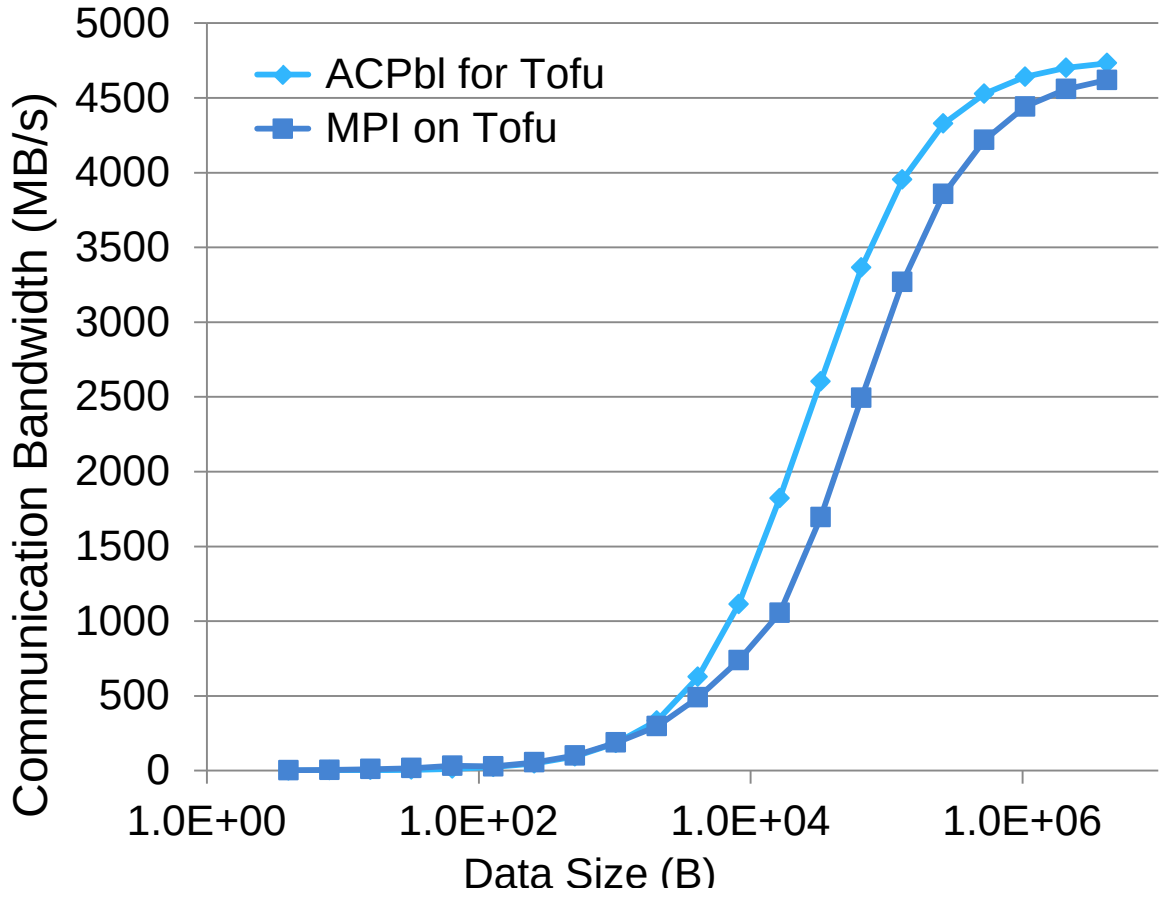
Basic Layer

- Middle Layer実装に使用する、低レベル通信インタフェース
  - 先進的ユーザーによる直接利用も可能
  - 現在の通信デバイスでも効率よく実装可能
  - 将来の通信デバイスでのハードウェア化を考慮
- グローバルメモリ参照(Global Memory Access , GMA)
  - 分散データ構造の実装に適したインタフェース
  - 64ビット・グローバルアドレス ＋ プロセス間のメモリツーメモリコピー



Basic Layerの開発状況

- UDP、Tofu、InfiniBandでリファレンス実装が稼働
- 現在試験、性能最適化中



ACP基本層(Tofu)の通信性能

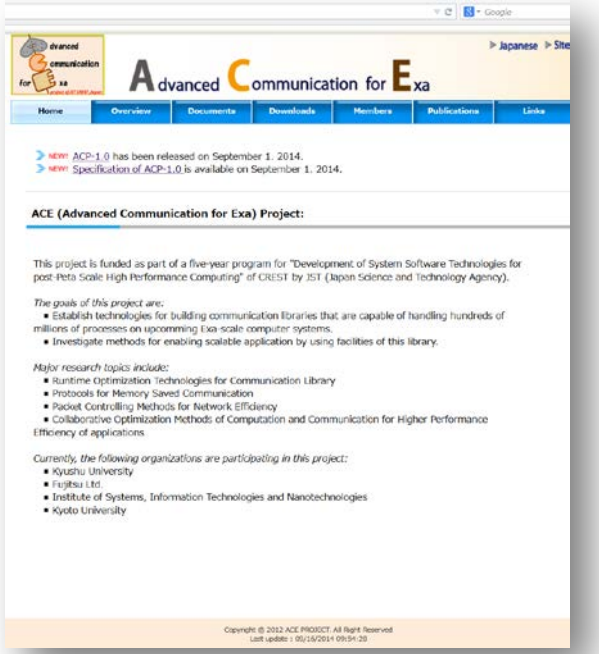
測定環境:
システム: 富士通FX10
CPU: SPARC64™ IXfx
メモリ: 32GB/node

Basic Layer メモリ使用量見積もり（100万プロセス）

	Tofu	UDP
ランク数に比例	69MB@100万プロセス • コマンド受信バッファ 64B • Tofuアドレステーブル 4B • Tofu経路テーブル 1B	18MB@100万プロセス • 待ち受けIPアドレス 4B、待ち受けポート番号 2B • 送信シーケンス番号 4B、受信シーケンス番号 4B • ランク番号 4B（初期化、リセット時使用、常時確保）
メモリ登録数に比例	9KB@128 登録 • 登録メモリ管理テーブル 40B • 論理アドレス検索テーブル 16B（比例）＋ 8 bytes × 256（固定）	
その他	262KB • コマンドキュー兼コマンド送信バッファ 64B × 4,096	647KB • コマンドキュー 80B × 4,096 • コマンドステーション 1,504B × 64 • デリゲートステーション 3,480 × 64
合計	**約70MB**	**約19MB**

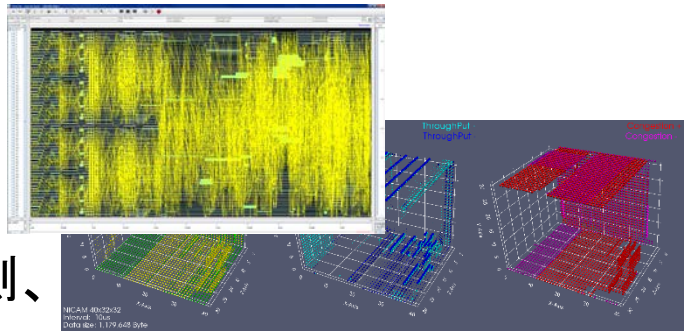
ACPライブラリ公開

- Webサイト
  http://ace-project.kyushu-u.ac.jp
- GNU/Autotools, Doxygen 採用
  - 実現した機能
    - 一般的な configure, make, make install
    - ネットワークデヴァイス自動判別
    - rpath と共有ライブラリ
  - 効果
    - OpenSource の体裁（ひいては信頼性）を確保
    - 開発サイクル（開発→テスト→更新）の効率向上
- コンパイル・実行ラッパ
  - acpcc, acprun を同梱
    - 利便性向上と人為ミスの低減
- V1.0.0-rc 公開：2014年9月開始
  - 開発したインタフェースに対する評価、フィードバック
    - BTSを設置
  - 投稿論文で提案した技術についての proof of concept

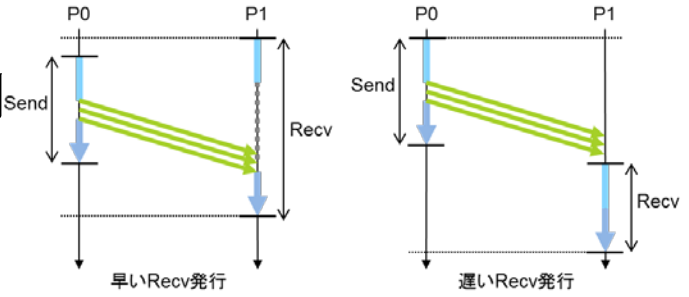


性能予測・解析環境 NSIM-ACE

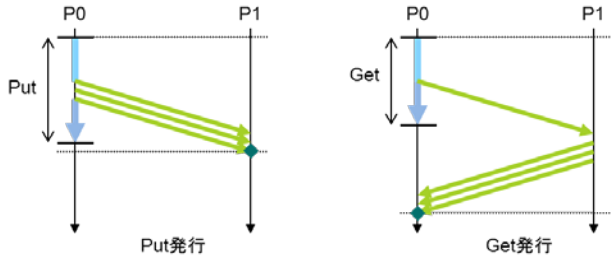
- NSIM-ACE
  - メッセージパッシング／RDMAモデルをサポートするインターコネクトシミュレータ
  - トポロジに応じた輻輳予測に基づく通信時間予測、解析



- 従来のNSIM
  - メッセージパッシングモデルに基づいた性能予測
  - 評価対象アプリケーションは、MPI準拠のAPI関数で記述（MGENプログラム）



- NSIM-ACEへの拡充
  - RDMAモデルに基づいた性能予測機構の実装
  - ACP準拠のAPI関数の実装



関連プロジェクト・組織との連携

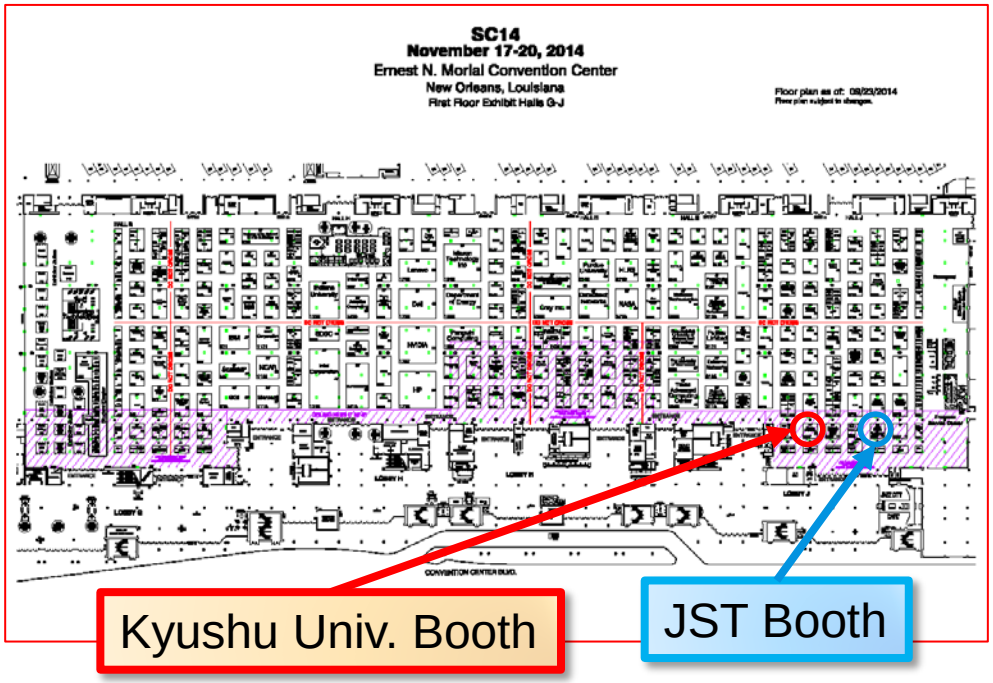
- MPI Forum
  - MPI Forum（9月）においてチャネルインタフェースを紹介。
    新規インタフェース案として議論を継続。

- XcalableMP
  - XcalableMPワークショップ（10月）で講演予定。
    高級並列言語のバックエンドとしての ACPへのフィードバックを期待。

- Jülich Supercomputing Centre
  - 日独共同研究プロジェクトへの申請に向けて協議中。
    - 共同研究案: JSCのアプリケーションへの ACP適用、およびフィードバック。

ACPチュートリアル @ SC14

- 内容
  - ACPライブラリのインタフェース紹介
  - 実機を使ったプログラミング実習
- 場所:
  九州大学ブース （#3507)
- 時間:
  14:30〜15:30
  (Tue, Wed and Thu)
- 定員:
  4 persons / day
- 景品:
  Sony Dual Port USB Memory Stick
  - 32GB, USB + microUSB



今後の取り組み

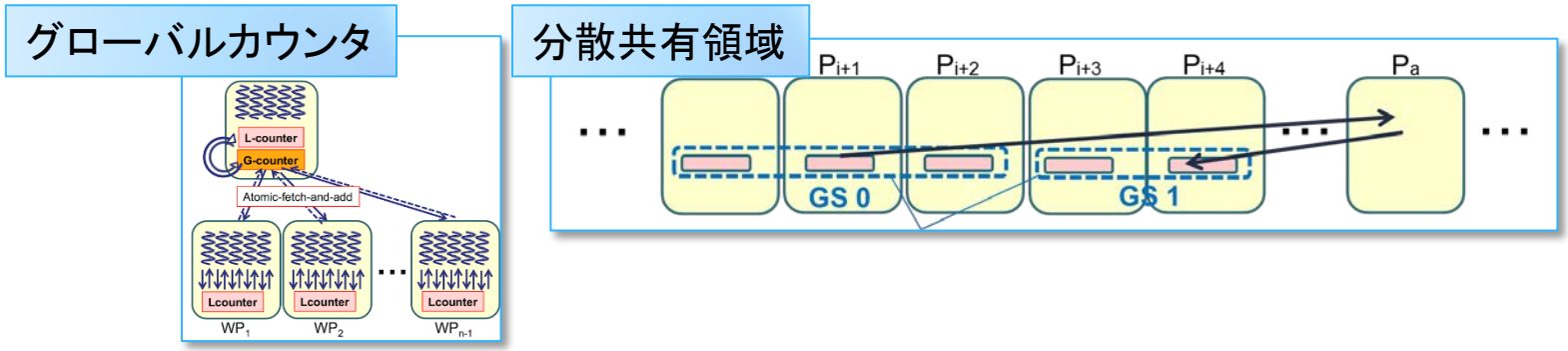
- ACPライブラリ開発
  - インタフェースの追加 （～2016.3）
    - collective, neighbor, deque, map, counter, etc.

- 評価、フィードバック
  - ACP上のアプリケーション開発、性能解析
    - NSIM-ACEによるスケーラビリティ予測
  - 各インタフェースの実装に対する自動最適化技術の適用
    - アルゴリズム選択、メモリ配分、etc.

- 外部組織との連携
  - MPI Forum、XcalableMP、Jülich Supercomputing Centre、etc.
  - チュートリアル、ワークショップ開催

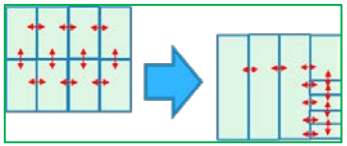
ACPのアプリケーションへの応用

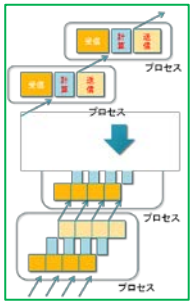
- FMO計算
  - 内部通信パターンの ACPによる実装
    - ACPを利用したFMO実装の通信における必要メモリ量解析[本田、他、2014]

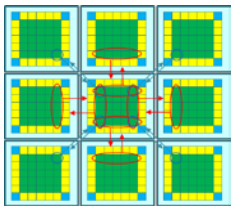


  - Middle Layerインタフェースの検討
    - FMO 実装向け通信インタフェースの提案[本田、他、2014]

- 多体問題、時間並列計算
  - チャネルインタフェースによる実装







- 電磁流体計算
  - 隣接通信インタフェースの検討

まとめ

- プロジェクトの目的:
  省メモリと通信性能を両立するスケーラブルな通信ライブラリ開発

- 進捗状況:
  - 要素技術の研究開発
    - 通信ライブラリのメモリ使用量解析
    - アプリケーションの通信パターン解析、効率化
    - 通信の自動最適化技術
  - 通信ライブラリ ACP (Advanced Communication Primitives) 公開開始
    - Basic Layer
    - Middle Layer
  - 性能予測・解析環境 NSIM-ACEの開発
  - 外部との連携
    - MPI Forum, XcalableMP, Jülich Supercomputing Centre

- 今後
  - インタフェースの充実化
  - 実装の最適化
  - アプリケーションへの応用、評価、およびフィードバック
  - 外部組織との連携