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

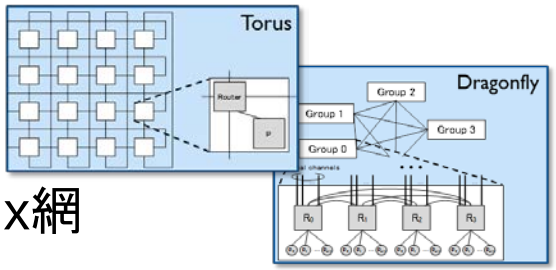
九州大学:
南里豪志、高見利也、本田宏明、薄田竜太郎、
森江善之、小林泰三
富士通株式会社:
住元真司、安島雄一郎、志田直之、佐賀一繁、野瀬貴史
公益財団法人九州先端科学技術研究所:
柴村英智、曽我武史
京都大学:
深沢圭一郎

2015年10月15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

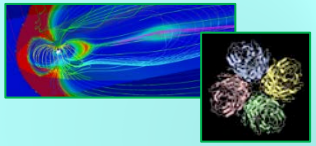


- 通信ライブラリ、およびアプリケーションへの要求：
  通信性能と省メモリの両立による高スケーラビリティの達成

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

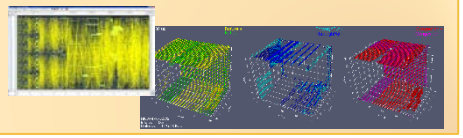
Applications

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

コミュニケーションライブラリ
通信チャネルを明示的に生成、破棄する通信関数群

データライブラリ
複数プロセスに分散配置したデータ構造を操作する関数群

Basic Layer

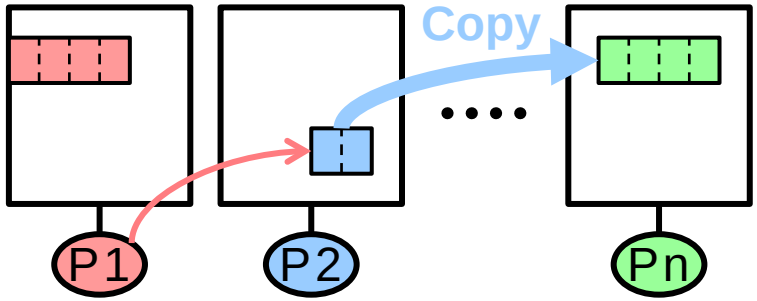
基本層
PGAS(Partitioned Global Address Space)モデルに基づく、通信抽象化層

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

Basic Layer

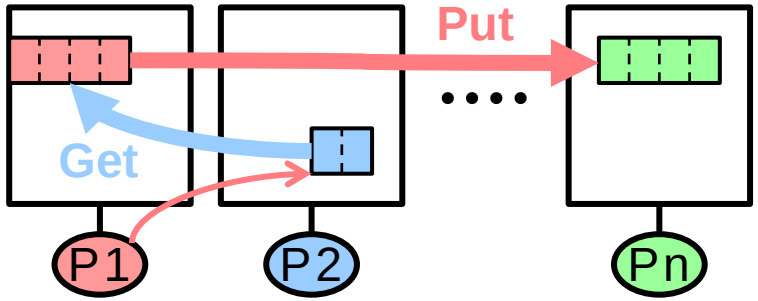
- グローバルメモリ管理
  - 各プロセスが登録したメモリ空間を 64bit グローバルアドレスで参照
  - 静的領域：初期化時に登録され、全プロセスから参照可能な領域
  - 動的領域：各プロセスで実行時に登録された領域
- グローバルメモリアクセス
  - 任意のグローバルアドレスに対するメモリアクセス（copy, atomic）
  - メモリアクセス間の順序関係を記述可能

GMA



Copy Request

RMA



Get Request

サンプルコード

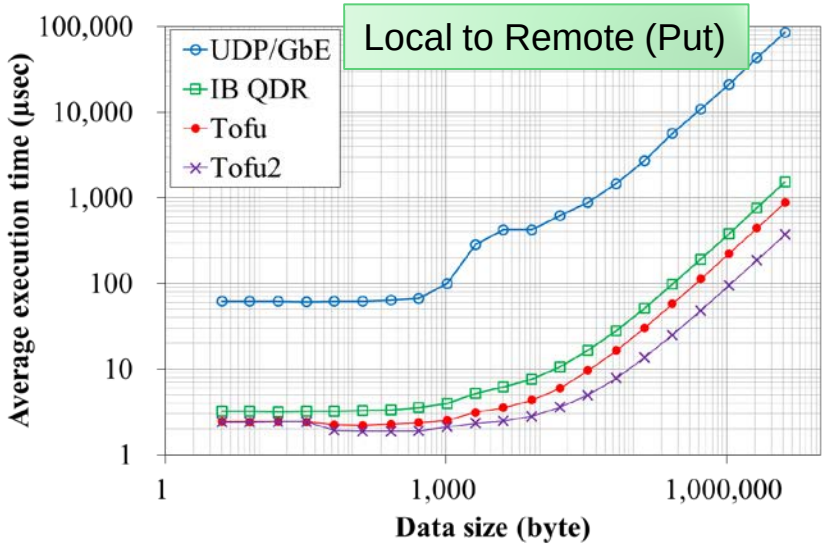
```
#include <stdlib.h>
#include <acp.h>
#define BUF_SIZE 16777216

int main(int argc, char** argv)
{
    volatile acp_ga_t* buf_ga;
    acp_init(&argc, &argv);

    acp_ga_t top = acp_query_starter_ga(acp_rank());
    buf_ga = (volatile acp_ga_t*)acp_query_address(top);

    void* buf = malloc(BUF_SIZE);
    acp_atkey_t key = acp_register_memory(buf, BUF_SIZE, 0);

    *buf_ga = acp_query_ga(key, buf);
    acp_sync();
    /* ... */
    acp_finalize();
    return 0;
}
```

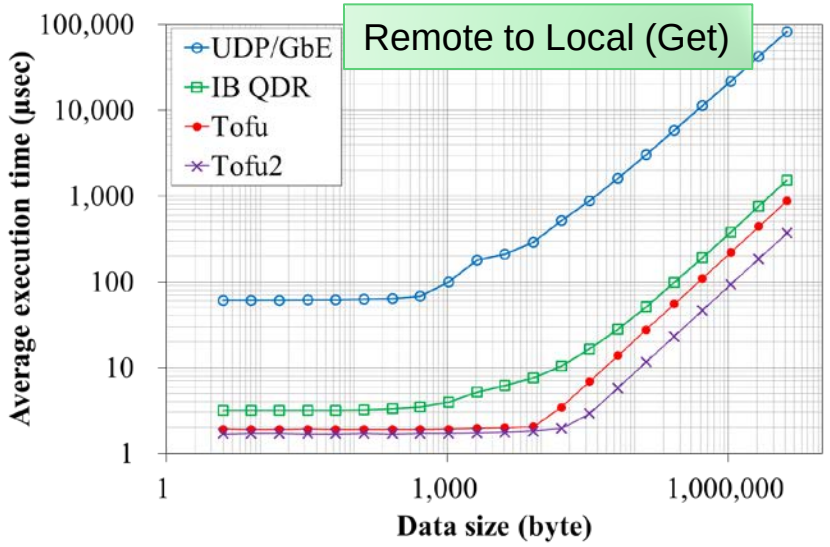
静的変数のグローバルアドレスを取得

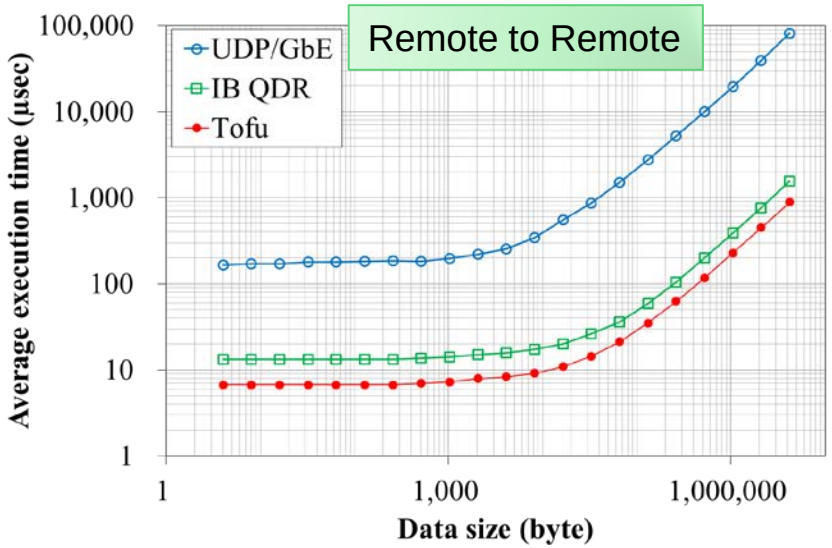
動的に確保したローカルメモリを登録

Basic Layerの基本性能

- UDP、Tofu、InfiniBandで稼働







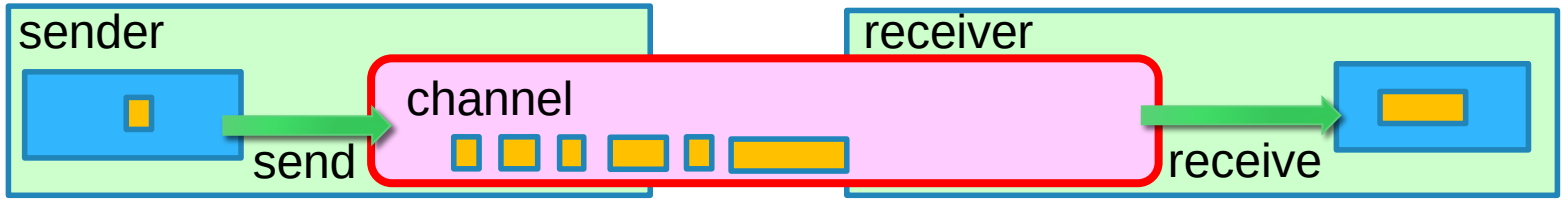
- 各インターコネクトで、ハード性能に近い性能を達成
- 小サイズでの遅延時間については、現在、チューニング中
- Tofu2 の Remote to Remoteは未計測

Basic Layer メモリ使用量見積もり（100万プロセス）

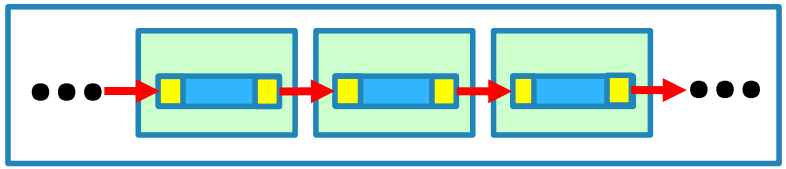
	InfiniBand	Tofu	UDP
ランク数に比例	176MB@100万プロセス • 登録メモリ領域管理 4B • QueuePair 160B • スターターメモリ情報 12B	69MB@100万プロセス • コマンド受信バッファ 64B • Tofuアドレステーブル 4B • Tofu経路テーブル 1B	18MB@100万プロセス • 待ち受けIPアドレス 4B、待ち受けポート番号 2B • 送信シーケンス番号 4B、受信シーケンス番号 4B • ランク番号 4B（初期化、リセット時使用、常時確保）
メモリ登録数に比例		9KB@128 登録 • 登録メモリ管理テーブル 40B • 論理アドレス検索テーブル 16B（比例）＋ 8 bytes × 256（固定）	
その他	2MB • コマンドキュー/バッファ 960KB • 登録メモリ領域テーブル 88B ×255 • 登録メモリキャッシュ 10KB ×1024 • CompleteQueue 128B	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178MB**	**約70MB**	**約19MB**

通信チャネルインタフェース

- 2プロセス間のSend/Recv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

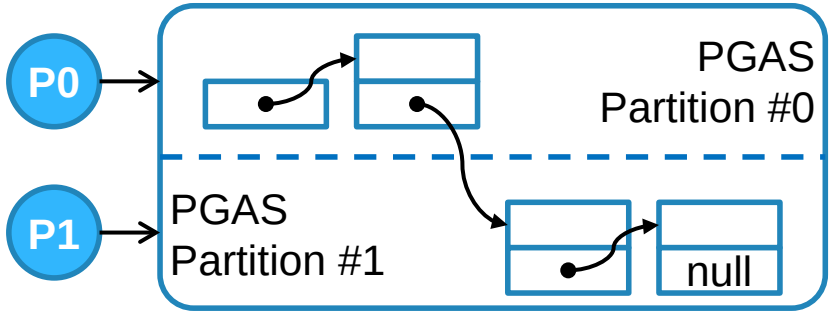
計算

チャネル解放

データライブラリ

- データ構造を複数プロセスに分散
  - 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

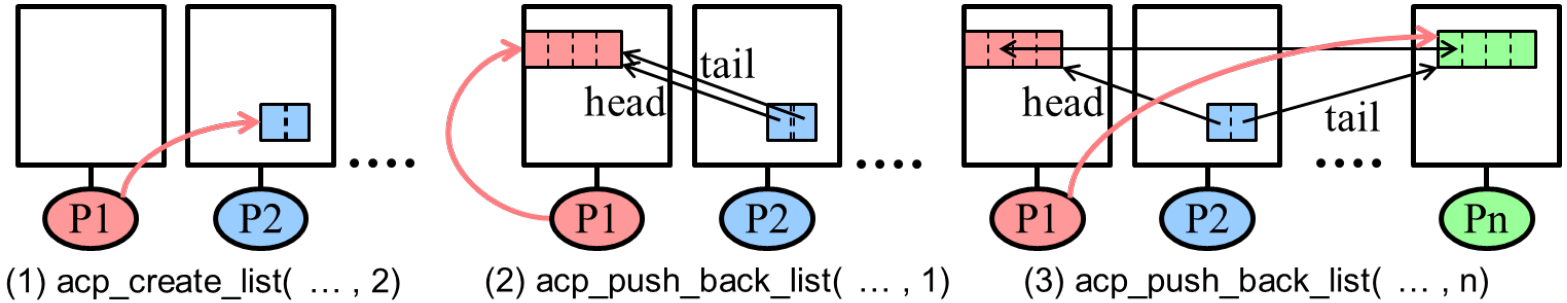
  C++言語の標準テンプレートライブラリ(STL)を参考



- グローバルメモリアロケータ
  - データライブラリの基盤技術
  - 指定したプロセスに動的にメモリ割り当て

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)

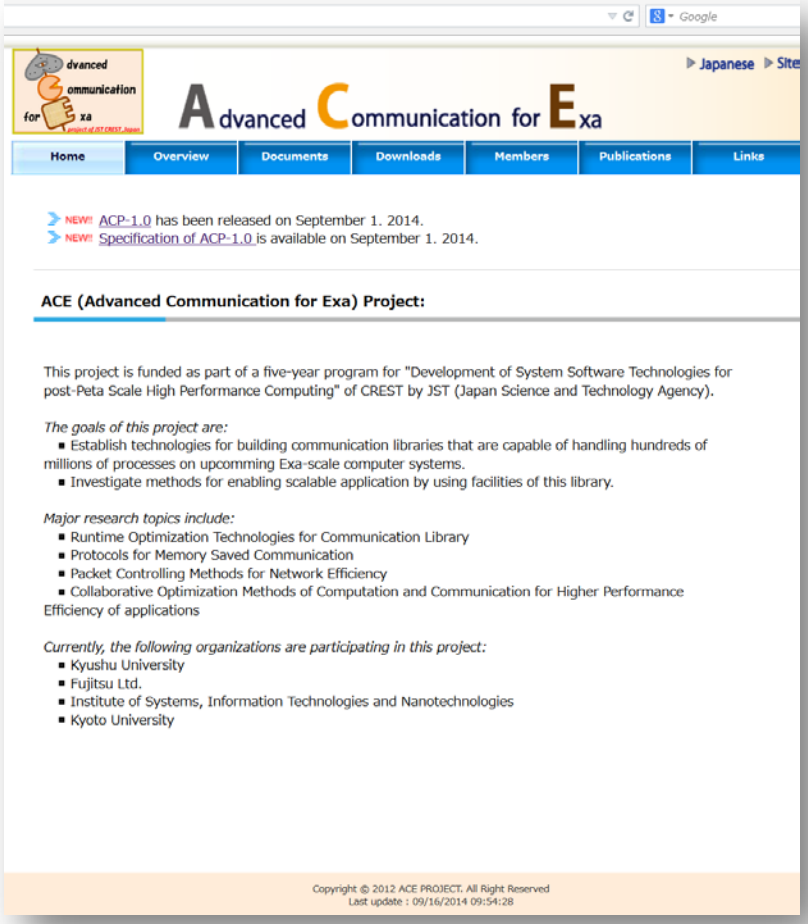
名称	定義
メモリ割当	acp_ga_t **acp_malloc**(size_t *size*, int *rank*);
メモリ解放	void **acp_free**(acp_ga_t *ga*);
ベクタ生成	acp_vector_t **acp_create_vector**(size_t *nelem*, size_t *elsize*, int *rank*);
ベクタ末尾要素追加	void **acp_push_back_vector**(acp_vector_t *vector*, acp_ga_t *ga*);
リスト生成	acp_list_t **acp_create_list**(size_t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

ACPライブラリ公開

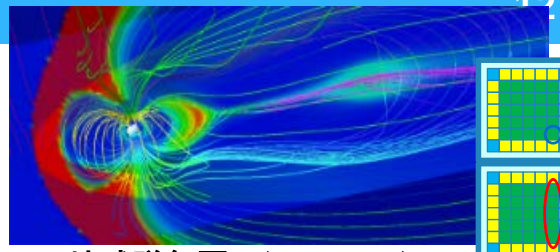
- Webサイト

  http://ace-project.kyushu-u.ac.jp

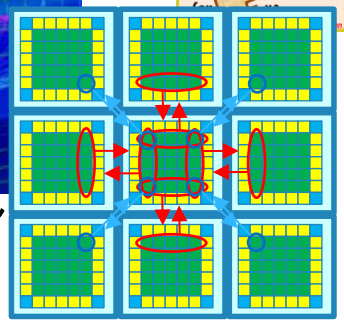
  - V1.1.1 公開中



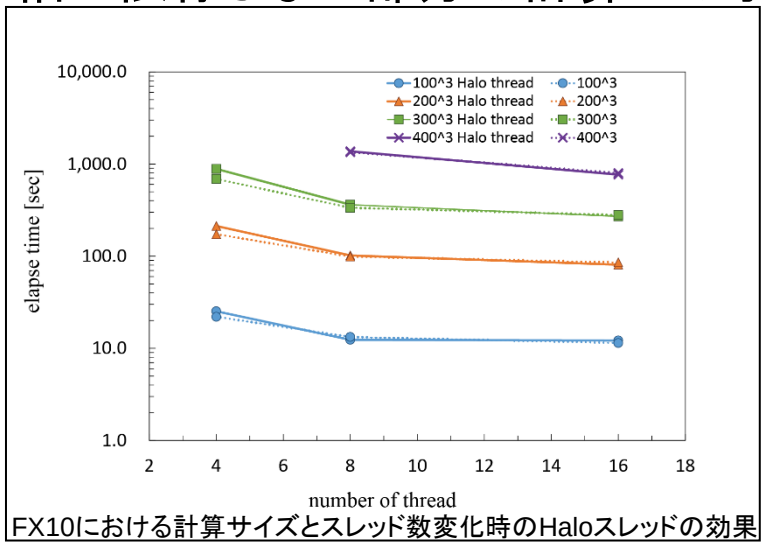
電磁流体シミュレーションのStencil計算



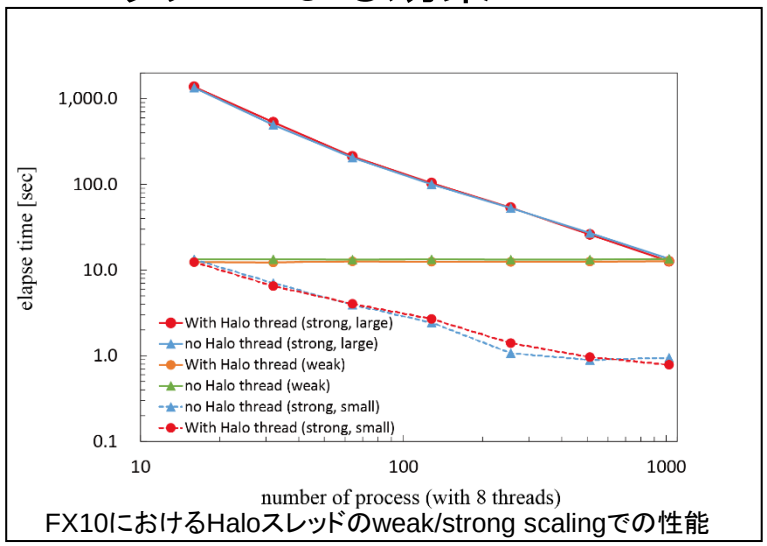
地球磁気圏のシミュレーション



- Haloスレッド
  - Stencil計算における袖通信、およびその通信に依存する計算を担当するスレッド
  - 袖に依存しない部分の計算とのオーバラップによる効果



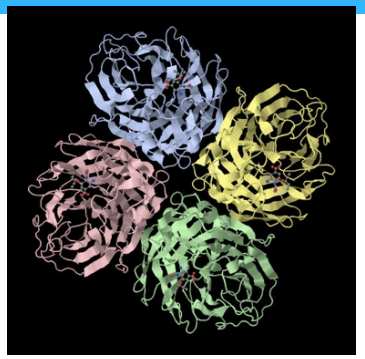
FX10における計算サイズとスレッド数変化時のHaloスレッドの効果



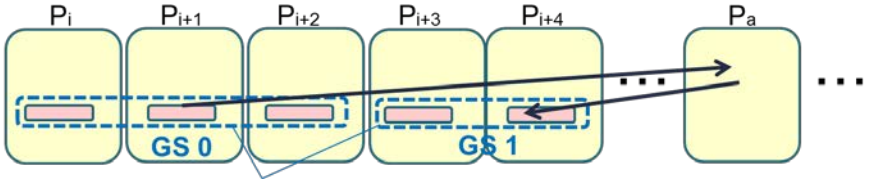
FX10におけるHaloスレッドのweak/strong scalingでの性能

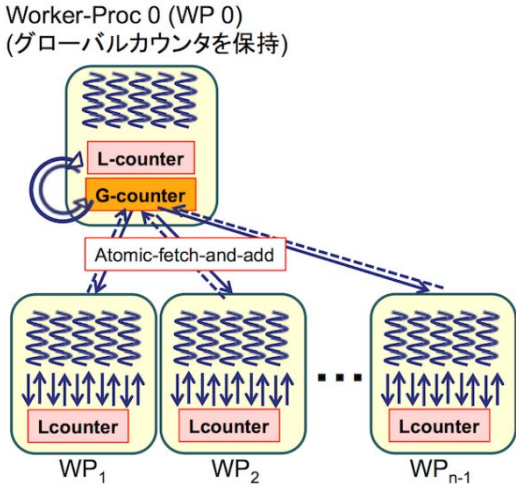
- Halo通信のフレームワーク化
  - 通信と計算のオーバラップを簡単に記述
  - ACPによる省メモリ実装

FMO計算



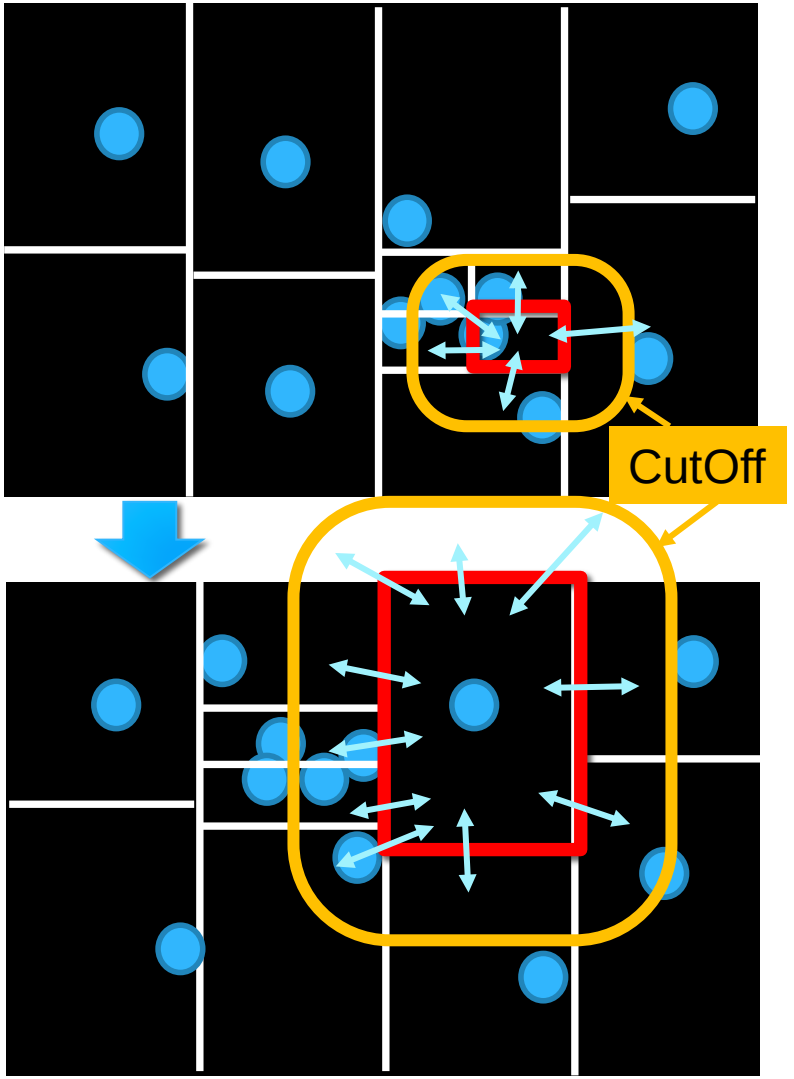
- FMO計算： master / worker モデル
  - 大規模分子の量子科学計算をタスク並列処理
- ACPによる実装
  - 共有ワークスペース
    - 全プロセスが非同期アクセス可能な連続メモリ領域
    - 1ノードに納まらない大規模連続領域を用いた計算が可能
  - グローバルカウンタ
    - 動的負荷分散に使用
    - Remote Atomic Fetch and Add による実装





重力N体シミュレーション

- Adaptive P^3M法
  - 粒子分布に応じた領域分割
    ⇒ 非構造格子
  - 不規則な通信
    - 粒子の移動
    - 領域分割の変更
- ACPによる実装
  - 通信用領域の動的確保、解放
  - グローバルメモリアクセス（copy, atomic)による効率的な通信

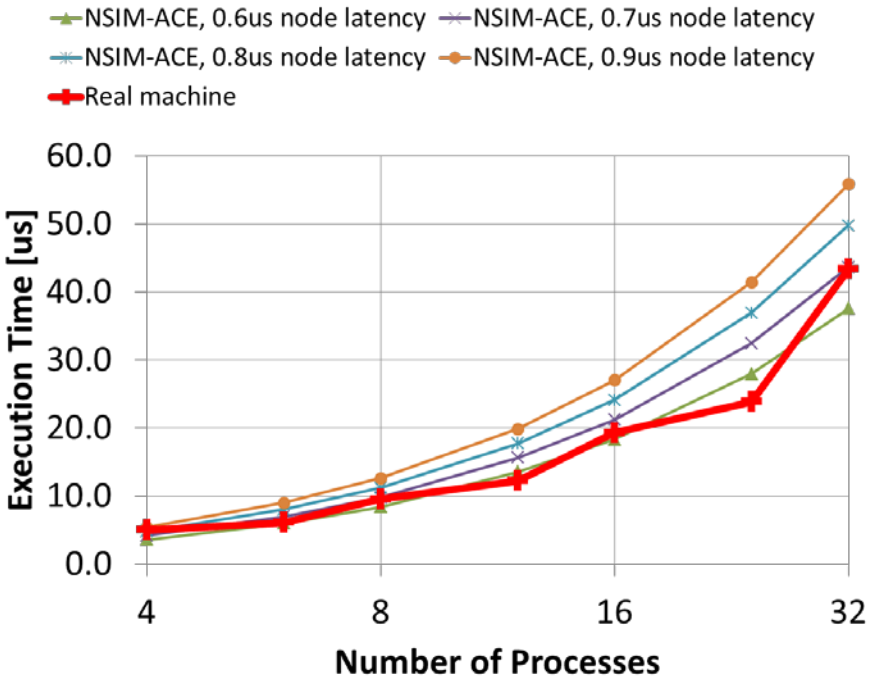


性能予測・解析環境 NSIM-ACE

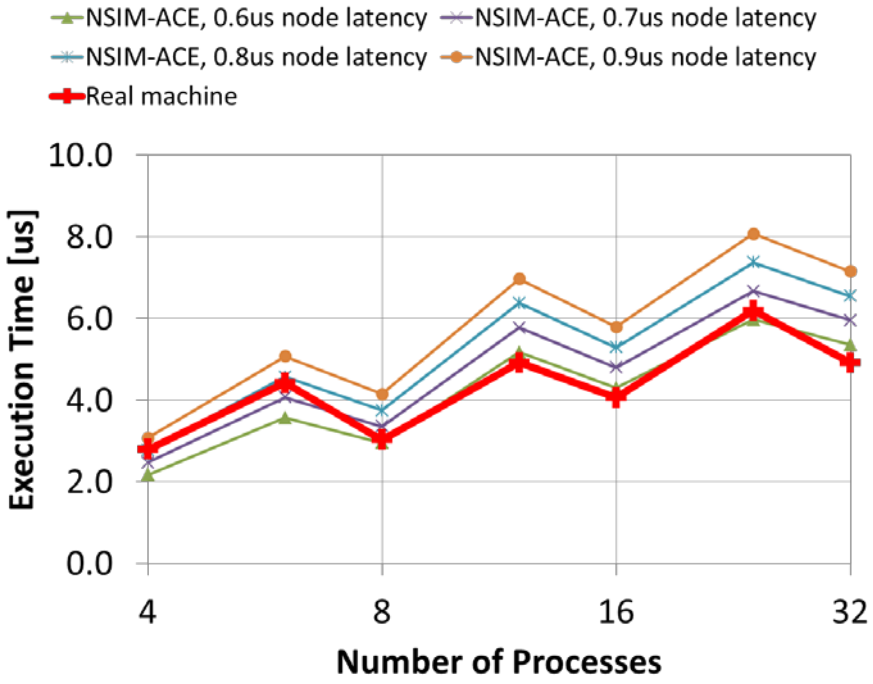
- NSIM-ACE = NSIM + RDMAシミュレーション機能
  - メッセージパッシング／RDMAの両通信モデルをサポート
  - トポロジや通信パターンによる通信衝突を踏まえた通信時間推定
- エクサスケール計算環境をターゲットとした性能予測
  - ACPアプリケーションやACPライブラリの開発・評価に有効
    - 通信モデル比較（MPI vs RDMA）
    - ライブラリ内で用いるアルゴリズム選定
    - スケーラビリティ評価
- NSIM-ACEの開発
  - RDMAモデルに基づいた性能予測機構の実装（完了）
  - ACP準拠のAPI関数の実装（一部完了、進行中）
  - シミュレーション精度の検証（実施中）

NSIM-ACEのシミュレーション精度検証

- 同期通信の精度検証
  - 実機のネットワーク／ノード仕様をNSIM-ACEに設定
  - 実機で観測困難なパラメータは較正により補正
  - 実用的な精度でのシミュレーション結果（予測時間、振る舞い）を確認



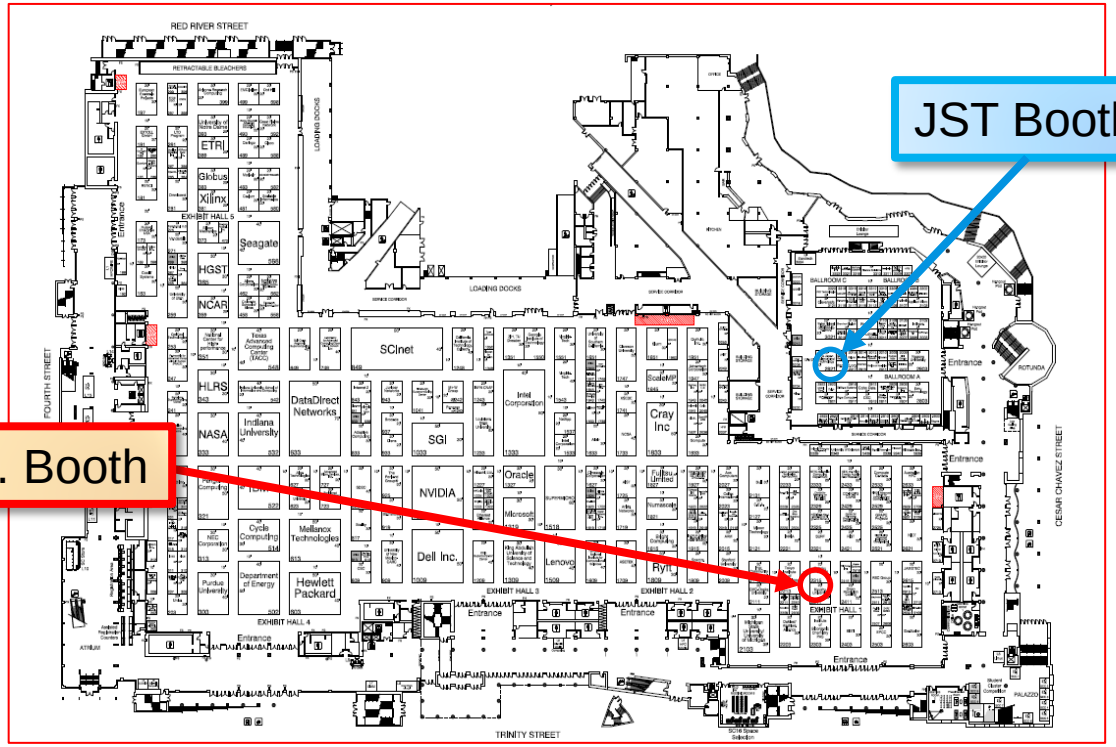
Ringによる同期時間



Recursive Doublingによる同期時間

ACPチュートリアル @ SC15

- 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
  (#2311)
- **参加者に景品あり**



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:
  - ACPライブラリ開発、公開
  - アプリケーション開発
  - シミュレータ NSIM-ACE開発

- 今後
  - インタフェースの充実化
  - アプリケーションへの応用、評価、およびフィードバック
  - ACPを用いた上位レイヤの構築
    - DSL、MPI