並列プログラミング言語XcalableMP規格部会
XcalableMP V2.0に向けて

2015/6/24

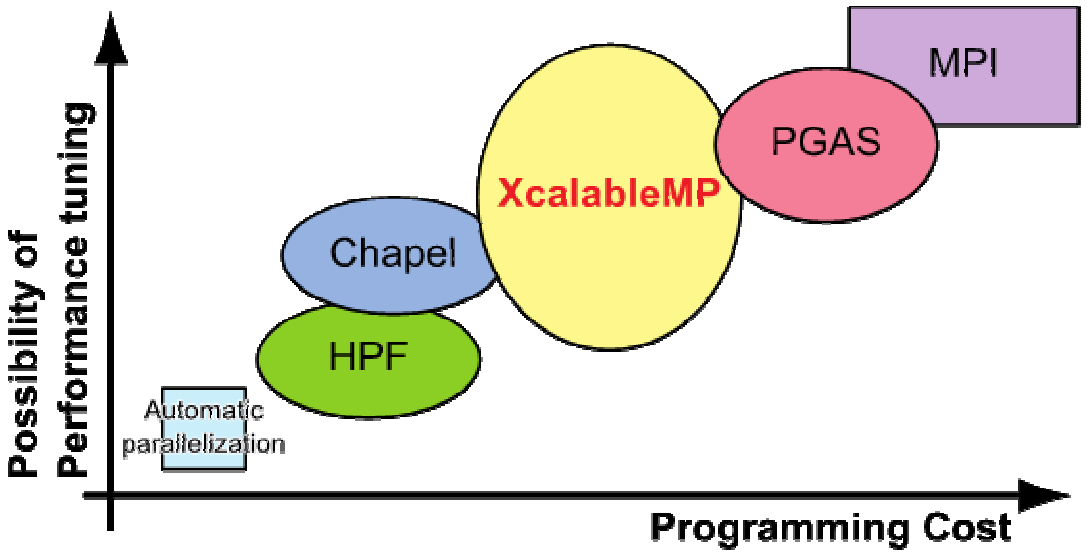
理化学研究所計算科学研究機構（富士通）
岩下 英俊

内容

- XcalableMPのご紹介
- これまでの仕様改善
- XcalableMP V2.0の目玉機能(予定)の紹介

XcalableMPの特徴

- Directive-based language extensions
  - ベース言語(CとFortran)を指示文で並列拡張
  - コードの書き換えや教育に要するコストの削減
- “Scalable” for Distributed Memory Programming
  - 実行モデルはSPMD。各ノードで独立に実行を開始する。指示されない限り冗長実行
  - グローバルビューは、データ並列プログラミングモデル
  - ローカルビューでは、片側通信を利用できる記法(Coarray)を提供
- “performance-aware” for explicit communication and synchronization
  - すべての同期・通信操作は指示文によって起きるため、HPFと異なりパフォーマンスチューニングが行い易い



XcalableMP V1.0

- 初版 2011年11月（超並列プログラミング言語検討会）
  - グローバルビューは、①データ分散、②負荷分散、③通信を、指示文で記述

XMP/F

```
module defs
  !$xmp nodes p(4)
  !$xmp template t(YMAX)
  !$xmp distribute t(block) on p
  integer a(XMAX, YMAX)
  !$xmp align a(*, j) with t(j)
end module

program main
  use defs
  integer i, j, res
  res = 0

  !$xmp loop on t(j)
  do j = 1, YMAX
    do i = 1, XMAX
      a(i, j) = foo(i, j)
      res = res + a(i, j)
    end do
  end do

  !$xmp reduction(+: res)
  ...
end program
```

①　②　③

XMP/C

```
#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
int a[YMAX][XMAX];
#pragma xmp align a[j][*] with t(j+1)


main() {

  int i, j, res;
  res = 0;

  #pragma xmp loop on t(j+1)
  for (j = 0; j < YMAX; j++) {
    for (i = 0; i < XMAX; i++) {
      a[j][i] = foo(i, j);
      res += a[j][i];
    }
  }

  #pragma  xmp reduction(+: res)
  ...
}
```

①　②　③

XMP規格部会による改善

- V1.1（2012年11月）
  - XcalableMPの文脈の中での、Coarrayの意味付けの明確化
    - Coarrayの「イメージ空間」は、XcalableMPの「タスク」にマッピングできる。これにより、複数のCoarrayプログラムによる大規模タスク並列を狙う。

```
!$xmp nodes p(8)=**
   real a(10,10)[*]

!$xmp task on p(1:4)

   call foo(a)
!$xmp end task

!$xmp task on p(5:8)

   call foo(a)
!$xmp end task
```

Coarray変数aはノード1～8に配置

サブルーチンfooをノードセット1～4で実行

サブルーチンfooをノードセット5～8で実行

```
subroutine foo(a)
   !$xmp nodes p(8)=**
   real a(10,10)[*]
    !$xmp coarray a on p

   real b(10,10)[*]

end subroutine
```

2つのノードセットでそれぞれ実行。
Coarray変数aは2つのノードセットで共有。

  - 分散問合せ組込み手続の追加
    - 分散配列のローカルイメージでのサイズの問合せなど

XMP規格部会による改善(つづき)

- V1.2(2013年11月)
  - OpenMPとのハイブリッドプログラミング作法の明確化
    - 外側ループをXcalableMP、内側ループをOpenMPで並列化する書き方
    - 一つのループをXcalableMPとOpenMPの両方で並列化する書き方
  - XMP/Cに配列関数を導入(ベース言語拡張)
    - Fortran90の配列代入文に似た記述
    - C言語版Coarray

```
double a[10][20], b[10][20]
double c[10][20]:[*];

a[:][:] = sqrt( b[:][:] )
me = this_image()
if (me > 1)  c[2:9][2:19]:[me-1] = a[2:9][2:19]
```

Coarray配列Cの宣言

sqrtは要素別処理組込み関数
自イメージ(ノード)番号を得る
左隣りのノードへ配列データをput通信

- V1.2.1(2014年11月)
  - マイナーチェンジ
- V1.3(2015年11月予定)向け検討中の項目
  - 袖領域を含むループ実行(HPF/JAのext_home相当)
  - task構文、宣言指示文の高速化を目的とする指示
  - Coarray/Cの仕様明確化

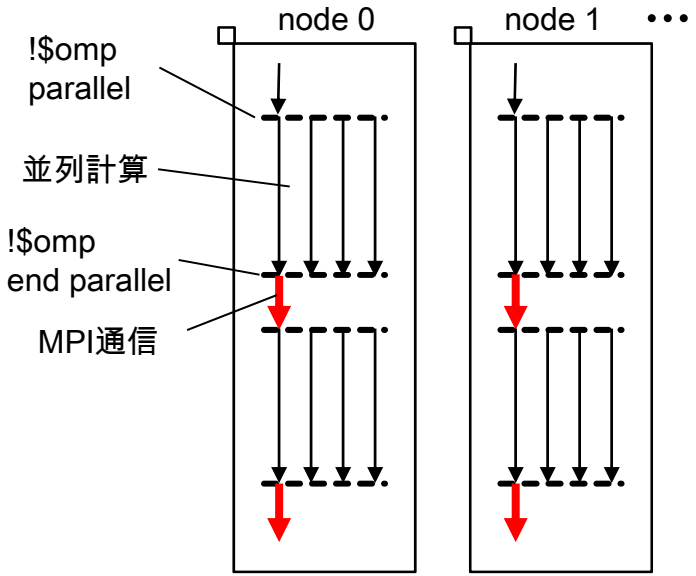
XcalableMP V2.0に向けて

- XcalableMPに今後必要なもの
  - 階層化の進むハードウェアへの対応
  - ベース言語の近代化
- V2.0では
  - メニーコアへの対応
  - ベース言語として、C++に対応
- 2016年11月公開を目指す。

メニーコアへの対応

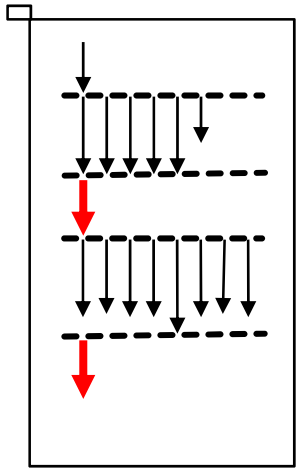
- ノード内の並列化の再考
  - 現在は、ノード内はループ並列（スレッド並列）
  - メニーコア×超並列になると、ループ並列だけでは性能が出ないのではないか。

ノード内ループ並列（現在）



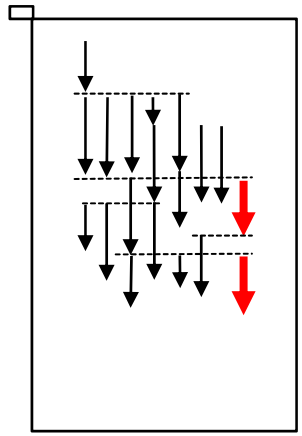
- ループ並列化によって、静的で均等な負荷分散を狙う。
- 計算性能を優先し、通信を計算区間の外に出す。

メニーコアでは



- ループ並列では、コアを均等に使い切れない。
- 静的な負荷分散＋バリア同期では、計算時間にぶれがあると、最も遅いスレッドに律速
- 計算時間が減ると、相対的に通信時間が顕在化

ノード内タスク並列（今後有望）



- 小さなタスクをコアに動的に割り当てる。
- タスク間のデータ依存と制御依存に沿ったスケジューリング。バリア同期は不要。
- ノード内とノード間の通信コストを考慮する。

メニーコアへの対応(つづき)

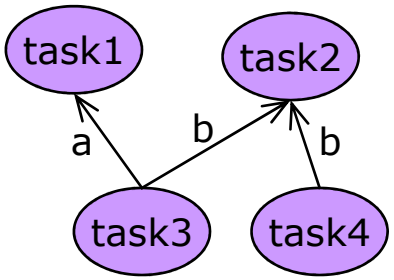
- XcalableMPの書式(検討中)
  - OpenMP4.0のtask構文の拡張、または、これに似た新しい構文となる見通し
  - 欧米を含む研究者とのコラボレーションにより、実装と併行して検討する。

プログラム例(書式検討中)

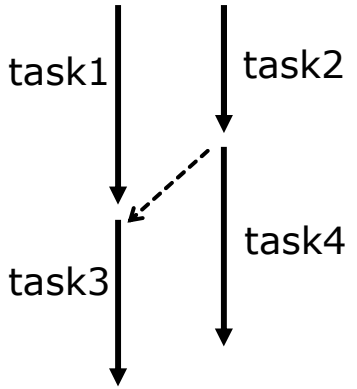
```
#pragma xmp tasklet out(a)
{
   task1();
}
#pragma xmp tasklet out(b)
{
   task2();
}
#pragma xmp tasklet in(a, b) out(c)
{
   task3();
}
#pragma xmp tasklet inout(b)
{
   task4();
}
```

分散配列？
Coarray配列？
同期用タグ？

記述から得られるデータ依存関係



プログラム実行例(2コア)



- a,bがリモートデータを含むなら、タスクの前後にノード間通信が必要。

C++のサポート

- C++のユーザは増加傾向。避けて通れない。
  - 国内スパコンユーザは現状 Fortran > C > C++
  - 新しいプログラムは C/C++ で書かれる傾向
- C++サポートとは、何をすることか？　他の言語／モデルに学ぶ
  - MPIでは
    - インタフェースをC++らしい表現に変更。機能的にはC版とほとんど同じ。
      - コミュニケータ、グループ、終了状態などを、C++のオブジェクトとして定義
      - C版の関数は、オブジェクトのメソッドとして再定義
      - MPI独自のname spaceの提供
      - MPIが検出した例外を、C++のマナーで処理できる
  - OpenMPでは
    - 利用者定義（constructor/destructor、利用者定義代入、例外処理）を生かすことで、機能的拡張
    - 新しい型（Vectorなど）への対応はない。（reduction max/minの組込み型が増えたくらい）
      - parallel regionの入口・出口、reduction演算などで生じる変数の生成・消滅・コピーに対し、constructor, destructorなどを使用
      - copyin節などに、利用者定義代入演算を使用
      - 並列化されるforループにiteratorを許す
      - throw/catchペアの出現場所に一定の制限を持たせて、例外処理をサポート
      - 実行時ライブラリインタフェースは、C版から変更なし
- XcalableMPで検討すべきは？
  - 通信の指示文にcopy constructorや利用者定義代入を利用
  - クラス、テンプレート対応
  - 例外処理。throw/catchの利用
  - 他、まだ全体が見えず。

まとめ

- XcalableMP仕様の概要と、これまでのエンハンスをご紹介
  - FortranとCがベース
  - グローバルビュー（指示文拡張）とローカルビュー（Coarray）で、適材適所の利用を推奨
- プラットフォームへの対応と、ベース言語の近代化を進め、2016年11月 XcalableMP V2.0公開を目指す。
  - メニーコア対応
    - 実装などの研究者とのコラボがキー
  - C++サポート
    - C++仕様を深く理解するところから
- XcalableMP規格部会

  http://www.xcalablemp.org/ja/index.html