XcalableMP 2.0の概要

理研

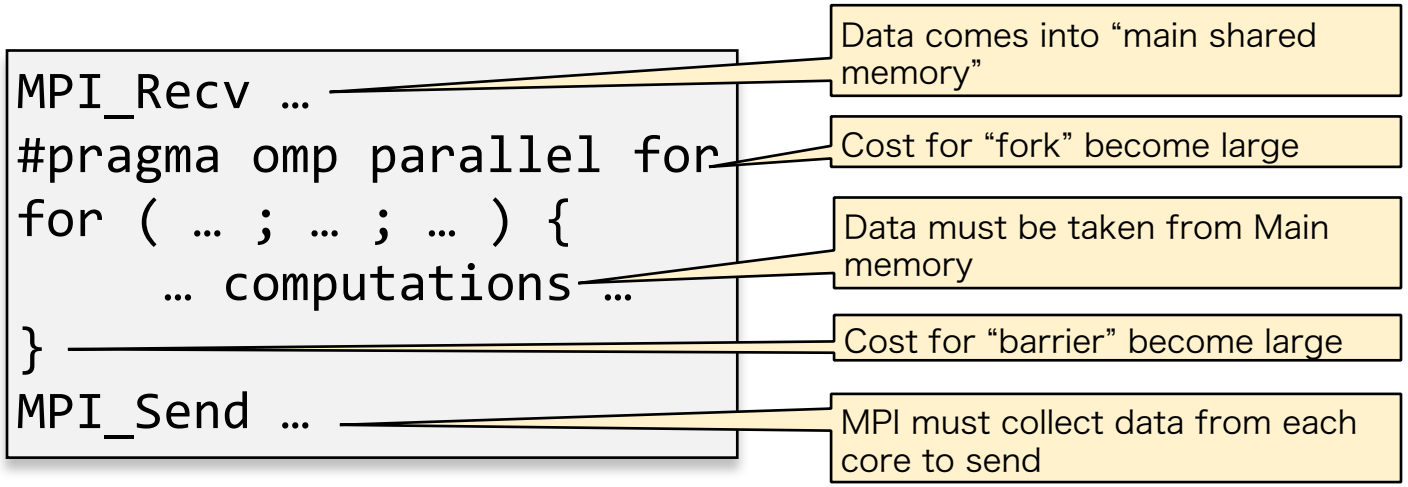
村井 均

背景 (1)

- PCクラスタコンソーシアム並列プログラミング言語XcalableMP規格部会において、XMP仕様を検討中。
- XMP1.x仕様(2016年10月現在の最新版1.2.1)は、ある程度の完成を見た。
  - グローバルビューおよびローカルビューによる分散メモリ並列処理
  - 残項目 (→ XMP1.3)
    - 少数の過誤、曖昧性の修正
    - 若干の機能拡張（主に性能向上および最適化のため）

背景 (2)

- メニーコアプロセッサの隆盛
  → 既存プログラミングモデル(MPI+OpenMP)の限界



➡ メニーコア対応を主な目標とする次期仕様XcalableMP2.0の検討を開始

もくじ

- XMP2.0
  - 基本コンセプト
  - 実行モデル
- XMP1.3
  - 修正点
  - 機能拡張
    - loop指示文
    - シャドウのリダクション
    - etc.

XMP2.0の基本コンセプト

- 互いに依存関係を持つ多数の「タスクレット」によるタスク並列処理
  - タスクレットの依存関係はノードをまたぐことができる (cf. OpenMPやOmpSs@BSCのタスク)
  - オーバヘッドの大きいバリア同期でなく、一対一同期をサポート
- ベース言語規格の更新
  - Fortran 2008
  - C99
  - C++11

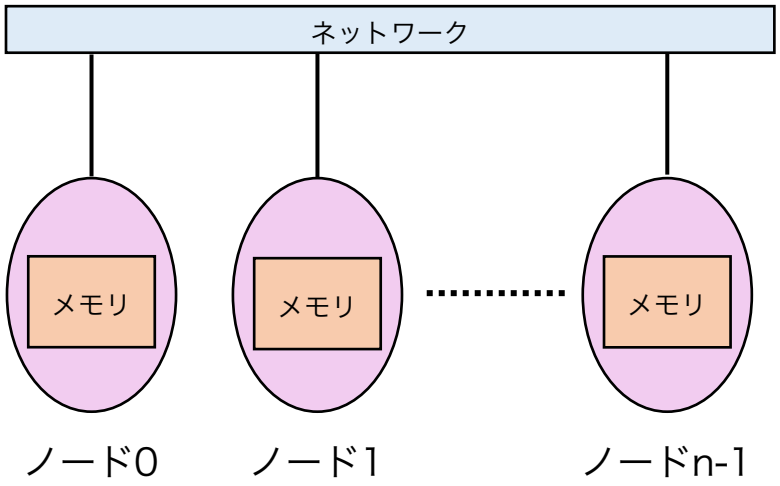
用語

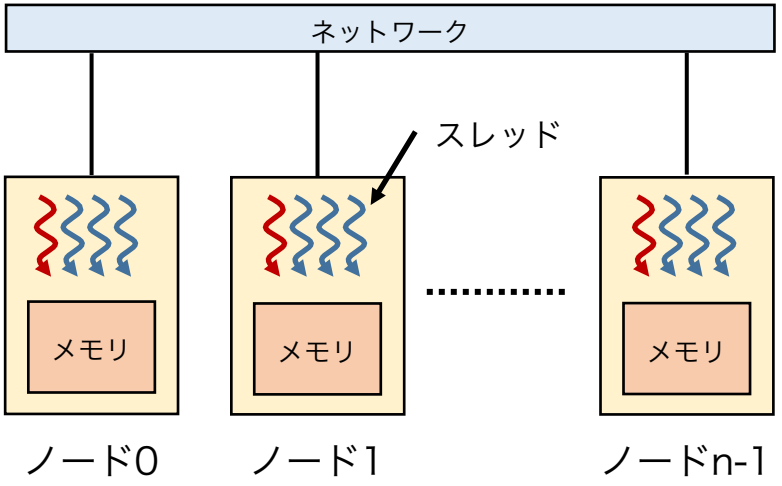
	XMP1.x	XMP2.0
ノード	XMPプログラムの実行者（局所性の単位）	XMPスレッドが動作する場所（局所性の単位）
スレッド	-	XMPプログラムの実行者
プログラム	実行コードとデータ	1つ以上のタスクレットの集合
タスクレット	-	互いに依存関係を持つ、実行コードとデータのインスタンス。互いにネストしていてもよい。

- XMP1.x実行モデル
  - 各ノードは、SPMDモデルに従って、XMPプログラムを実行する。

XMP1.0とXMP2.0の比較

- XMP1.x
- XMP2.0





XMP2.0実行モデル (1)

1. XMPプログラム全体を、一つの暗黙のタスクレットが囲んでいると見なす。
2. [実行開始時] XMP実行時システムは、
   - 各ノードに1つ以上のスレッドを生成する。
   - マスタスレッドに暗黙のタスクレットを割り当てる → マスタスレッドは実行を開始。
   - 他のワーカスレッドを休止状態にする。

XMP2.0実行モデル (2)

3. [実行中] 各スレッドは、
   - `tasklet`構文に遭遇したら、タスクレットを生成する。
   - `taskletwait`構文(仮)に遭遇したら、自ノード上の全てのタスクレットの実行が完了するまで待つ。
   - XMPプログラムの末尾に暗黙の`taskletwait`を配置。
4. [タスクレットスケジューリング]
   - タスクレットは、依存関係が満たされるまで実行されない。
   - これ以外の一切については実装依存。

XMP2.0実行モデル (3)

- マスタスレッドのみ、tasklet構文なしの場合、XMP1.x実行モデルに一致。
- スレッドのforkは実行開始時のみなので、OMPのparallel構文に相当するものはない。
- XMPグローバル構文は、tasklet構文の中に書けない (ローカル構文(coarray)は可)。
- XMPスレッドとOMPスレッドの関係 → 実装依存

XMP1.3における変更点

- 過誤、曖昧性の修正
  - wait_async構文の動作
  - xmp_pack組込み関数のmask引数の型
  - xmp_scatter組込み関数の動作
  - リダクション種別「-」の意味
- 機能拡張
  - xmp_exit組込み関数/ビルトイン関数の追加
  - [XMP/C] 角括弧によるノードおよびテンプレートの参照、角括弧内の3つ組
  - [XMP/C] 多次元分散配列の動的割付け (xmp_mallocビルトイン関数の拡張)

```
n1 = ...; n2 = ...; n3 = ...;
float (*a)[n2][n3];
#pragma xmp align a[i][j][k] with t(i,j,k)


a = (float (*)[n2][n3])xmp_malloc(xmp_desc_of(a), n1, n2, n3);
a[i][j][k] = ...;
```

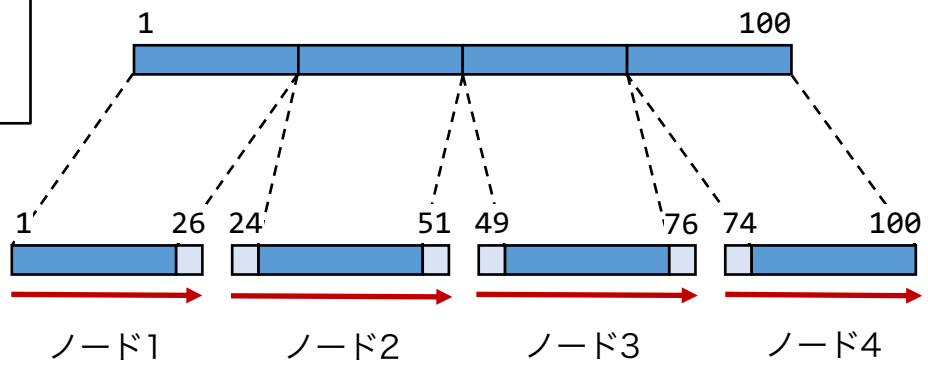
機能拡張: loop指示文の拡張

- 以下の4つの節を新たに追加
  - expand → 分散後のループ範囲を拡大
  - margin → 分散後のループ範囲のフチだけを実行
  - peel_and_wait → 通信と計算のオーバラップのため、ループを分割
  - pipeline → ウェーブフロント並列化

expand節

- 分散後のループ範囲を拡大または縮小
- 用途:
  - シャドウの初期化
  - テンポラルブロッキング

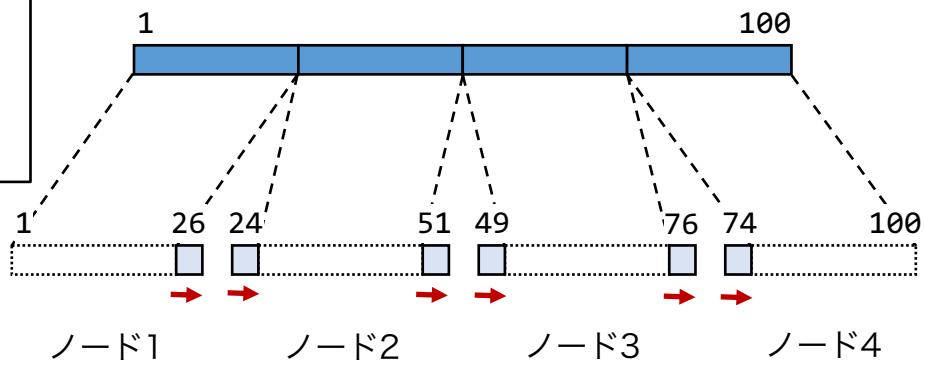
```
!$xmp loop on t(i) expand(1:1)
do i = 1, 100
  a(i) = ...
end do
```



margin節

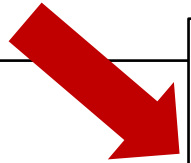
- 分散後のループ範囲のフチ(外側または内側)だけを実行
- 用途:
  - シャドウの初期化
  - テンポラルブロッキング

```
!$xmp loop on t(i) margin(1:1)
do i = 1, 100
  a(i) = ...
end do
```



peel_and_wait節

```
!$xmp reflect (a) async(10)

!$xmp loop on t(i) peel_and_wait(-1:-1, 10)
do i = 2, 99
  b(i) = (a(i-1) + a(i) + a(i+1)) / 3
end do
```

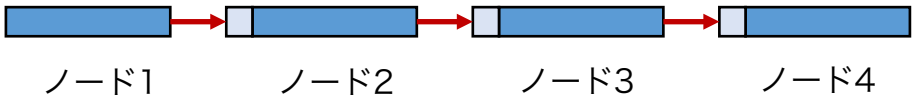


```
!$xmp reflect (a) async(10)

!$xmp loop on t(i) expand(-1:-1)
do i = 2, 99
  a(i) = (a(i-1) + a(i) + a(i+1)) / 3
end do

!$xmp wait_async(10)

!$xmp loop on t(i) margin(-1:-1)
do i = 2, 99
  a(i) = (a(i-1) + a(i) + a(i+1)) / 3
end do
```

- expand、wait_async、marginに展開される(syntactic sugar)。
- 用途: テンポラルブロッキング

pipeline節

```
!$xmp loop on t(i) pipeline(a/1/)
do i = 2, 100
  a(i) = a(i) + a(i-1)
end do
```

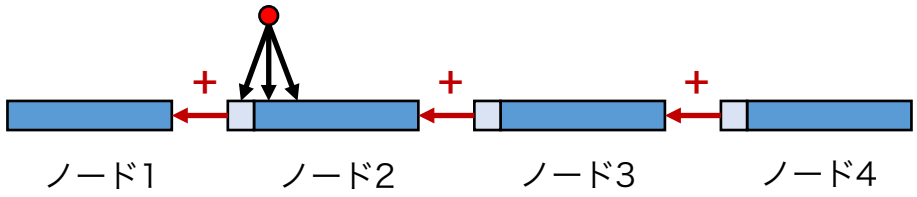



```
下限側ノードからrecv
do i' = lb, ub
  a(i') = a(i') + a(i'-1)
end do
上限側ノードへsend
```

- ループの直前で、指定した配列のシャドウに対するrecv通信、直後でsend通信を行う。
- 結果として、ループはパイプライン的に実行される。
- 用途: ウェーブフロント並列化

機能拡張: シャドウのリダクション

```
!$xmp reduce_shadow (a) width(1:0)
```



- シャドウの値を、対応するデータ実体へ足し込む。
- 用途: Particle-in-Cell法における場の更新

今後の予定

- XMP1.3とXMP2.0ドラフトを今月にリリース予定
  - XMP1.3
    - 少数の過誤、曖昧性の修正
    - 若干の機能拡張
  - XMP2.0ドラフト
    - スレッドに基づく実行モデル
    - 「タスクレット」機能
    - スレッド間の同期
    - etc.

まとめ

- XMP1.xの検討は、ほぼ収束
  - XMP1.3として、少数の修正と拡張をリリース予定
- XMP2.0を検討中
  - メニーコアに向けスレッドの概念とタスクレット機能を導入
  - ベース言語規格の更新 (Fortran 2008、C99、C++11)
- 興味を持たれた方、ぜひXMP規格部会へご参加ください！

www.xcalablemp.org