並列プログラミングモデルの海外動向 調査報告

地球シミュレータセンター

村井 均

murai@jamstec.go.jp

はじめに

- 米国における、並列プログラミングモデルの動向を報告する。
- SC2004(11/6~12)とその前後に、主要なユーザや研究者から聞き取り調査を行った。

- NASA JPL
- SC2004会場
  - LBNL, ORNL, LANL, NASA Ames研
  - IBM, Cray
  - OpenMP開発者
  - etc.
- Rice大学
- NECアメリカ

もくじ

- 現場の声
- OpenMPの動向
- 新しい言語: Co-Array Fortran (CAF)
- 新しい言語: Chapel
- 各手法の比較

現場の声

- ORNL
  - 「ほぼ全てのユーザはMPIを使っている。OpenMPやCAFは一部が試しただけ」
  - 「MPIは汚いので嫌い。OpenMPを試してみていい結果が出た」

- NASA Ames研
  - 「ほとんどMPIだけ」
  - 「Columbia(SGI Altix)ではOpenMPも使われる」
- IBM
  - 「MPIのみ。それ以外を提供する予定はない」
- Cray
  - 「CAFはX1のみ。XT3, XD1では提供しない」

OpenMP（1）

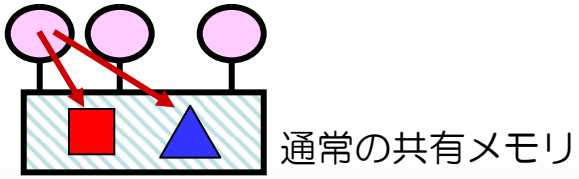
OpenMPの問題

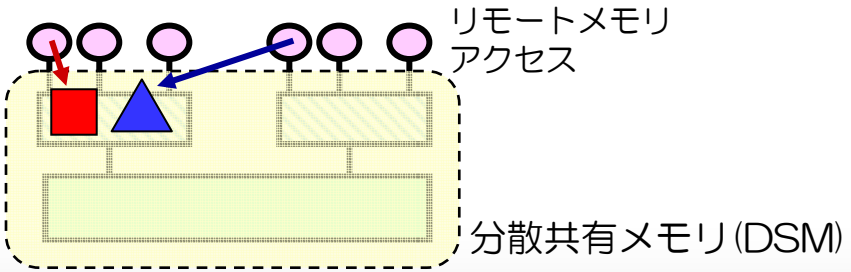
データの分散(ローカリティ)を指定できない。

➡ データと処理の分散が整合せず、

- キャッシュの効率低下
- リモートメモリアクセスの多発(分散共有メモリの場合)

が起きても、是正することができない。





OpenMP (2)

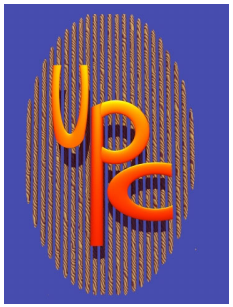
- 現状では、比較的小規模のSMPシステム以外で高性能を得るのは難しい。
- DSMへの適用についての問題は認識されているが、これを解決しようという機運はあまりない。

➡ 大規模システムでの利用は困難

Co-Array Fortran (1)

- Crayが開発したFortranの拡張言語
- 「Co-Array」と呼ばれるグローバルアドレス空間を介した通信に基づく。
- CrayがX1用の処理系を開発している他、Rice大学がフリーの処理系を開発・提供。

同じ思想の言語(Global Address Space Language):
- C版: Unified Parallel C (UPC)
- Java版: Titanium



Co-Array Fortran (2)

- プログラム例

```
      REAL a(400), b(400)
      ...
      DO i=2, 399
        b(i) = a(i-1) + a(i+1)
      END DO
```

cf. HPFによる並列化

```
      REAL a(400), b(400)
!HPF$ DISTRIBUTE (BLOCK) :: a, b
      ...
      DO i=2, 399
        b(i) = a(i-1) + a(i+1)
      END DO
```

```
      REAL a(0:101)[nprocs], b(0:101)[nprocs]
      ...
      IF (me > 1) a(0) = a(100)[me-1]
      IF (me < nprocs) a(101) = a(1)[me+1]
      ...
      DO i=max(1,istart), min(100,iend)
        b(i) = a(i-1) + a(i+1)
      END DO
```

「Co-Array」の宣言

「Co-Array」を介したリモートデータアクセス(通信)

※ 片側通信として実装

Co-Array Fortran (3)

- MPIと同レベル・同モデルの並列化を簡便に記述するためのインタフェース。

データの分散・処理の分散・通信の生成は全てユーザが陽に指定する。

➡ 生産性はMPIとあまり変わらない。

- 開発者・推進者の声は大きいが、(現在のところ)実際に使われているわけではない。

Chapel（1

性能(=実行時のproductivity)に加えて、開発時のproductivityにも重点を置く。

- Cascade High Productivit
- DARPAが進める「High Productivity Computing System (HPCS)」計画に含まれるCascadeプロジェクトで開発中の新し 並列言語
- Cr h, NASA JPL, Stanford 大, e大)
- NASA JPLのHans Zima教授が主導

・高マルチスレッド
・NUMA
・分散共有メモリ

Chapel (2)

Chapelの特徴

- マルチスレッド並列プログラミング
- 局所性を考慮したプログラミング

実行時の Productivity (従来の「HPC」)

- オブジェクト指向プログラミング
- 総称的プログラミングと型推論

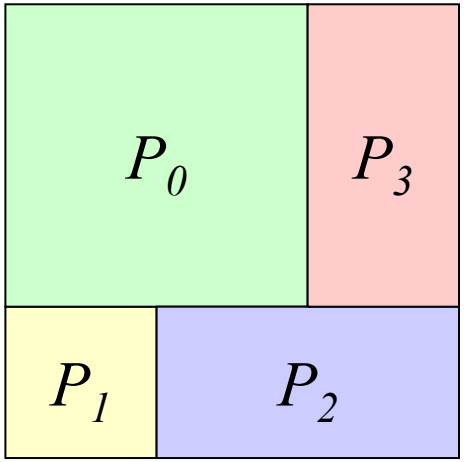
開発時の Productivity

Chapel (3)

局所性を考慮したプログラミング

= 「(分散共有メモリマシンを仮定しているので)データの局所性を陽に制御する必要はないが(e.g. 通信)、考慮した方が性能は出る」

- ドメイン：第一種の言語要素。インデックス空間と分散を融合した概念。
  - カルテシアン積ドメイン
  - 不透明ドメイン(暗黙に分散される)
- ドメインの分散
  - HPFの分散
  - ユーザ定義分散
  - タイル分散



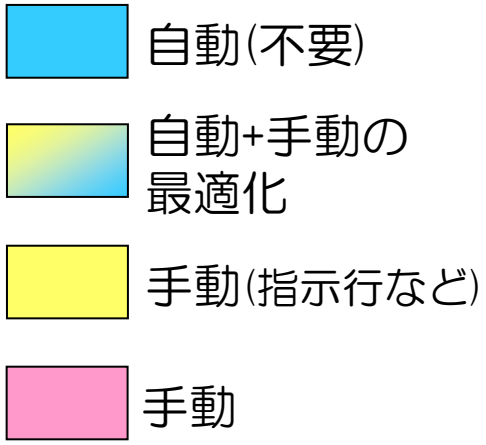
タイル分散の例

Chapel (4)

- 立ち上がったばかりのプロジェクトであり、詳細は明らかではない。
- Zima教授曰く「ものになるには10～15年掛かる」
- 現在、最も進んだ仕様を掲げる並列言語だが、実際に広まるか(使われるか)どうかは未知数。

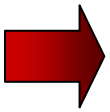
各手法の比較

	データの分散	処理の分散	通信の生成
HPF			
HPF/JA			
OpenMP	※共有メモリ		※共有メモリ
Chapel New!			※共有メモリ
XPF			
CAF New!			
MPI			



まとめ

- MPIに代わる並列化手段を望む声は依然としてある。
- 今後数年のうちに広く普及しそうな並列言語・プログラミングモデルは見当たらない。
- CAFやChapelから有用な機能を取り入れていくことも必要。



HPFPCは、海外の動向(特にChapel)に注視し、今後もHPFの普及・推進を続ける。