HPFを取り巻く現状

坂上@核融合研
村井@NEC

●大きな期待に応えられず，その反動で不評を買った.
●マーフィーの法則により複数の不幸が重なった.

コードの並列化

- 実験結果と直接比較できる項目のある大規模のシミュレーションがしたいので,分散メモリ型並列計算機が前提になる.
  - ようやく,それが可能な並列計算機が出現した.
  - しかし,コードを並列化しなければならない.
- コードを並列化するためには,本来の物理モデル／アルゴリズムのプログラミング以外に,並列化のためのプログラミングが必要である.
  - 並列プログラミングにおけるデファクトスタンダードは,MPIである.

MPI並列プログラミング

- 開発途上のコードをMPIでプログラミングすることは,大きなストレスとなる.
  - 新しい物理モデルの導入,新しいアルゴリズムの採用によるコードの改変が頻繁にある.
  - 単純なデバック出力でさえも,プログラミングに大きな労力が必要である.
  - バグの原因が,そもそものプログラム／アルゴリズムにあったのか,MPI化によって混入したのかの判断が難しい.
  - 唯でさえ,大規模な分散メモリ型並列計算機の性能を引き出すことは難しい.

本当にMPIで十分か?

- 計算科学ユーザ(≠計算機科学ユーザ)にとって,MPIによる並列プログラミングはあまりにも煩雑である.
  - 並列プログラミングは,本業ではない.
  - 可能なら完全自動並列化が望ましいが…
- もしMPIしかなければ,多くの計算科学ユーザにとって,分散メモリ型並列計算機を本来のHPCのために使いこなすのは,難しい.
  - 複数のパラメータランを同時に実行する環境として並列計算機を見る!!

なぜHPFか?

- 比較的簡単にプログラミングできて,そこそこの並列性能が得られればよい.
  - プロダクションラン用コードなら,がんばってMPI化を一回すればよいのだが…
- OpenMPは,プログラミングは簡単だが共有メモリ型並列計算機でしか使えない.
  - 分散メモリ型並列計算機では使えない.
  - PCクラスタでも使えない.
- 他に選択肢がなかった.
  - DARPAはHigh Productivity Computing Systemプロジェクトを進めているが,まだ顕著な成果なし.

他の並列プログラミング手法

- OpenMP系列
  - OpenMP+ccNUMA
  - OpenMP+Software Distributed Shared Memory
  - Cluster OpenMP
- 並列言語
  - Co-Array Fortran
  - Unified Parallel C
  - Chapel, X10, ~~fortress~~

OpenMP系列

- 分散メモリを共有メモリのように見せるので,プログラミングは楽だが…
- 超高速なネットワークが必須である.
  - ハードウェアが非常に高価になる.
- リモートメモリへのアクセスが発生すると,性能が急激に低下する.
  - アクセスのローカリティを保証できない.
  - first touchだけでは,無理がある.
- 大規模並列での性能は,期待できない.

Co-Array Fortran

- 今までのFortranコンパイラでは,エラーになる.
- 通信はGET/PUTを原則としており,高い並列性能はアーキテクチャに大きく依存している.
- プロセッサ間のデータ転送を明示的に記述しなければならない.
  - MPIプログラミングと手間は同じ?????
- 2005年にFortran2008の標準として一旦採用されたが,まだ最終仕様や標準実装について,決着していない.

サンプルプログラム

```
real :: r[*]     ! Scalar co-array
real :: x(n)[*] ! Array  co-array
! Co-arrays always have assumed co-size

real :: t        ! Local scalar
integer :: p     ! Local scalar

! Remote array references
! MPI_GET communication
t = r[p]
x(:) = x(:)[p]

! Reference without [] is to local part
! MPI_PUT communication
x(:)[p] = r
```

Chapel

- Cascade High-Productivity Language
- アドレス空間はグローバルであり,通信はコンパイラが自動的に生成する.
- データ並列,タスク並列を抽象化できる記述方法を提供する.
  - localeとdomain
- 考え方は,ほとんどHPFと同じ???
  - ただし, HPFとは逆に, OpenMPと同様にデータ分散より処理分担を優先している.

サンプルプログラム

```
var N: integer = 1000;
var A, B: [1..N] float;

// forall specify parallel execution
forall i in 2..n-1 do
   A(i) = B(i-1) + B(i+1);
```

```
var N: integer = 1000;
var CompGrid: [1..N] locale;
var D: domain(2) distributed(Block(2), CompGrid);
var A, B: [1..N] float;

forall i in D on B(i) do
   A(i) = B(i);
```

並列プログラミング手法の比較

- 並列プログラミングで大事な三つのポイントについて,それぞれの手法を比較する.

	データの分散	処理の分担	通信の生成
MPI			
OpenMP+DSM	※分散共有メモリ		※分散共有メモリ
Cluster OpenMP			※分散共有メモリ
HPF			
CAF(UPC)	※Co-Array		※代入文
Chapel			※グローバルメモリ

自動(不要)
自動+手動の最適化
手動(指示行)
手動

JAHPF

- HPF合同検討会(Japan Association for HPF)
  - http://www.hpfpc.org/jahpf/
  - 1997年1月29日に発足し,活動を開始した.
- HPFをベースとし,以下の3条件を満たす並列言語仕様を確立する.
  - 標準性:多用なプラットフォーム上で動作
  - 適用性:科学技術計算スキームの実装
  - 高速性:ハードウェア性能の活用
- HPF/JA仕様を策定した.

HPFPC

- HPF推進評議会(HPF Promoting Consortium)
  - http://www.hpfpc.org/
  - JAHPFを発展解消し,2001年7月9日に設立総会を開催して,任意団体としての活動を開始した.
- 活動の中心を拡張言語仕様の開発から,実用アプリケーションのHPF化促進,実環境でのHPFの評価にフォーカスする.
  - HPF利用を支援するために,会員所有のコードをHPF化するための個別相談や講習会を行う.

日本のコンパイラー

- HPF/ES - 地球シミュレータ用
  - HPF2.0+HPF/JA+独自拡張
- HPF/SX - NEC SXシリーズ用
  - HPF/ESとほぼ互換(除く:並列I/O)
- HPF/ES for PC cluster
  - HPF/ESとほぼ互換のNEC製PCクラスタ用
- HPF/VPP - 富士通VPP用
- fhpf - 富士通製
  - フリーのHPFコンパイラ(HPF推進協議会で配布)

海外のコンパイラー

- ADAPTOR
  - ドイツのGMDで開発されたパブリックドメインのHPFコンパイラ
  - HPF2.0+公認拡張機能の多く
- SHPF
  - オーストリアのVCPCで開発されたパブリックドメインのHPF2.0コンパイラ
- PGHPF Compiler
  - 米国PGI社で開発されたHPFコンパイラ
  - HPF2.0+公認拡張機能の一部

HPFの国際会議

- HPF User Group meeting
  - 1997 HUG #1, Santa Fe, US
  - 1998 HUG #2, Porto, Portugal
  - 1999 HUG #3, Redondo Beach, US
  - 2000 HUG2000 (#4), Tokyo, Japan
- HPF international Workshop: Experiences and Progress
  - 2002 HiWEP2002, Nara, Japan
  - 2005 HiWEP2005, Nara, Japan

HPFの現状

- 現在活動しているのは,ほぼ日本だけ.
- 残念ながら,この段階から今後広く普及する可能性は低いと言わざるを得ない.
  - 富士通のVPP Fortranと同様に,NECの方言として利用される可能性が高い.
- 期待が大きかったのに,なぜ失敗したのか?
  - 失敗から教訓を学ぶべきである.
  - その上で今後を考える.

HPFの光と影

- The Rise and Fall of High Performance Fortran
  - Proc. of the third ACM SIGPLAN conference on History of programming languages, San Diego, California, June 9 - 10, 2007.
    - Ken Kennedy, Rice University, Houston, TX.
    - Charles Koelbel, Rice University, Houston, TX.
    - Hans Zima, University of Vienna, Austria.
- この内容も踏まえて,HPF擁護の立場で振り返ってみると...

アンチMPIに対する大きな光

- ユーザが最小限の指示文によってデータの分割配置の方法さえ指定すれば,コンパイラが残りの作業(処理の分担と通信の生成)をすべて自動的に行う.
- 指示文を無視すれば,並列化されていない正しいFortranコードと見なせる.

(教訓) 過度な期待をユーザに抱かせないようにしよう!

f77 vs. f90問題の f77 vs. HPF問題への転嫁#1

- HPFはf90をベースにしている.
- HPFの評価を既存のプログラムで行った.
- でも,当時f90 cleanな既存のプログラムはほとんどなかった.
- このため,膨大な手間をかけてf77をf90 cleanなプログラムにする必要があった.
  - call by addressを逆手にとったプログラミング
    - ユーザ独自のメモリ管理
    - 仮引数と実引数での配列次元数の違い
  - HUG #2では,その苦労話が多かった.

f77 vs. f90問題の f77 vs. HPF問題への転嫁#2

- 既存プログラムをHPF化するときに大幅なソース修正が必要になったが,それをHPFのせいにされた.
- せっかく苦労してHPFに書き換えたのに,性能が出ない…
  - 性能が出ないのは別問題だが…

(教訓) 状況をよく見よう.
ユーザはコンパイラ開発者よりコンサバ!

  - ただし,f77のままじゃダメでf90 cleanは必須だろう. タイミングが早過ぎた?

ユーザ手間削減の罠#1

- できるだけ自動でやり,できないときはユーザに補ってもらうという基本方針である.
  - どこまで書けばいいのか不明瞭
    - PROCESSORS+DISTRIBUTE
    - INDEPENDENT
    - SHADOW+REFLECT
    - ON HOME LOCAL+ENDON
  - コンパイラによる性能の大きな非互換性が発生してしまう.
    - NEC SX vs. 富士通VPP
  - コンパイラ実装者にとっても曖昧になる.

ユーザ手間削減の罠#2

- 原則として,ユーザの明示的なコントロールを排除していた.
  - 指示文を書いても,最終的に何が起こるかはコンパイラ次第である.
  - ただし,HPF/JAでは一部許した.

(教訓) 多少はユーザにじゃまくさくても,
仕様は明確な方がいい!
「小さな親切大きなお世話」は止めよう!

言語としての完成度を求めた罪

- ユーザの希望や要望より,言語としての完成度や美しさを重視し,使いもしない実装が難し過ぎる仕様を入れたしまった.
  - 多様な分散指定
    - 仕様があるだけで性能発揮が難しくなる.
  - 自動マッピング
    - あらゆる場合を考慮しなければならず,難しい.
    - 通信が非常に重い処理が裏で動いてしまう.

(教訓) 絵に描いた餅は止めよう!

初期コンパイラの未熟性

- 完成度が低いまま,節操なく商業主義に走ってしまった.
  - 単純なcyclic分散ですら,まともな性能で動かなかった!
- 仕様の曖昧さと実装の未熟さの相乗効果でプログラムの移植性が悪かった.
- 既存のFortranライブラリをうまく呼べない場合が多かった.
  - 分散情報の引き渡しと実装方法

(教訓) 完成度が低いままで公開するな!

リファレンス実装の不在

- MPIにおけるMPICHのようなフリーのリファレンス実装がなく,普及が進まなかった.
  - 今なら,fhpfがある!
- 仕様の曖昧さに対するベンダ毎の異なった解釈を生みやすかった.

(教訓) フリー/オープンソースの影響力は侮れない!

チューニングの困難性

- 全自動化を意識していたため,性能が出ない場合にどうしたらいいか,分かりにくかった.
  - 並列化されているのに性能が出ない.
- 熟練者がいなかった.
  - 今ならHPFPCがある!
- 解説本がなかった.
  - 今なら各種ドキュメントがある!
  - プラ・核学会誌でも特集した.

(教訓) ユーザ教育を最初から考えよう!

その他

- 仕様策定段階におけるコンパイラ実装者とシミュレーション研究者の交流不足?
  - JAHPFでは,それを埋めたのだが...
- ユーザに忍耐がなさ過ぎた!
- 欧米では,シミュレーション研究者とテクニシャンが別々のことが多かった.
  - プログラムのMPI化を任せられたので,日本の研究者ほどの切実さがなかった?
  - むしろ,仕事がなくなる??

(教訓) ユーザ啓蒙をやっておこう!

教訓のまとめ

- 過度な期待をユーザに抱かせないようにしよう!
- 状況をよく見よう.ユーザはコンパイラ開発者よりコンサバ!
- 多少はユーザにじゃまくさくても,仕様は明確な方がいい!
- 絵に描いた餅は止めよう!
- 完成度が低いままで公開するな!
- フリー/オープンソースの影響力は侮れない!
- ユーザ教育を最初から考えよう!
- ユーザ啓蒙をやっておこう!

再び,本当にMPIで十分か?

- ソフトウェアの継承と開発の持続性を考えるとMPIプログラムでは,どう考えても厳しい.
- 少なくともEUには,MPIだけではダメだという空気があった.
  - EPCC, Dresden Univ., INRIA (2009/02)
- Life is too short for MPI
  - T-shirts message@WOMPAT2001

XcalableMP(XMP)

- 二つの並列プログラミングスタイルをサポートする新しい並列言語仕様である.
- グローバルビュー
  - データ並列プログラミングモデルとワークシェアにより,典型的な並列化をサポートする.
  - HPFと同等な機能である.ただし,HPFの失敗を教訓としている.
- ローカルビュー
  - 個別のノードを意識してプログラミングできるようPGAS機能を提供する.