分散メモリ向けデータ並列言語 OpenMPD の設計と実装

李 珍泌，佐藤 三久，朴 泰祐

筑波大学 大学院 システム情報工学研究科

{jinpil,msato,taisuke}@hpcs.cs.tsukuba.ac.jp

概要：分散メモリ型並列システムの標準的なプログラミングモデルである MPI は複雑な仕様を持ち，プログラマに大きな負担となる．分散メモリ並列システムに対し指示文を加えることによりデータ並列プログラムを記述するプログラミングモデルを提案し，その処理系 OpenMPD を実装した．いくつかのアプリケーションにおいては，指示文の記述によって逐次コードの変更をほとんど行わず，簡便に並列化を行うことができた．性能評価の結果，指示文によるわずかな記述により良好な性能が得られることを示した．

1 はじめに

今日では，高性能並列計算のプラットフォームとして PC クラスタを始めとする分散メモリ型のシステムが広く普及している．そのプログラミングには主に MPI（Message Passing Interface）が用いられる．MPI は並列化のための多様な機能を提供する反面，データの送受信をプログラマが明示的に指示しなければならないため，プログラミングが複雑になる．共有メモリの OpenMP のような簡単なプログラミングモデルを使うために Omni/SCASH'などが提案されているが，性能面で問題がある．

本稿では PC クラスタのような分散メモリ型並列計算機のための簡単なプログラミングモデルとして OpenMPD を提案する．OpenMPD は OpenMP のような指示文による並列化を分散メモリ上で実現したものである．データの分散や同期，ループの並列実行によるワークシェアリングなどデータ並列プログラミングで頻繁に用いられる手法を指示文の記述によって簡単に行うことができる．

CAF（Co-Array Fortran）や UPC（Unified Parallel C）のような既存の並列プログラミング言語は多様な並列化手法を提供する反面，そのモデルが抽象的で使いこなすことは容易ではない．そのため，OpenMPD の設計に際しては直観的で簡便なプログラミングモデルを提供することを心がけた．

2 OpenMPD の概要

データ並列プログラミングには多くのアプリケーションで共通して使われる典型的な手法が存在する．OpenMPD はこのような典型的な手法を指示文で記述することができる．これは逐次コードにプログラマが明示的な指示文の記述で並列化を行うというモデルで，共有メモリの OpenMP に類似する．

OpenMPD は C 言語の指示文による拡張として設計されており，実装した処理系は OpenMPD コードを MPI コードに変換するため，実行モデルは MPI 同様の SPMD(Single Program Multiple Data)である．

図 1 にその記述例を示す．#pragma で始まる行は並列化のためにプログラマが挿入したものである．#pragma ompd distvar の記述で配列のデータを各ノードに割り当てる．for ループに指示文#pragma ompd for を記述し，ループの並列実行によるワークシェアリングを記述している．共有メモリの OpenMP と同様，正しい並列化のためにはプログラマが責任を持って適切な指示文を記述しなければならない．

MPI を用いた並列プログラミングでは並列化のため，元のコードに大量の修正を施す必要がある．その結果，プログラムが複雑になり生産性が低下する．指示文による並列化は元のコードに指示文を追加するだけで済むため，MPI のように並列化によってプログラムが複雑になることを避けることができる．その結果，MPI を使うより分散メモリ上の並列プログラミングが簡単になる．

```
int array[YMAX][XMAX];
#pragma ompd distvar(var = array;dim = 2)
main(){
  int i, j, res;
  res = 0;
#pragma ompd for affinity(array) reduction(+:res)
  for(i = 0; i < 10; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      res += array[i][j];
    }
}
```

図 1. OpenMPD の記述例

OpenMPD の指示文はデータ並列プログラミングを支援する．それらは多くの科学計算アプリケーションのデータ並列化に用いられるものである．以下に OpenMPD の並列化手法とそれに対応する指示文の説明を行う．

A) 配列の割り当てと同期

指示文 distvar によって配列の割り当てを行うことができる．図 2.a のように配列 a を 4 ノードで並列に処理する時には各ノードで配列の全体領域を確保した後，異なる領域を処理するように制限することで（図 2.a の灰色の部分のように）割り当てを行う．

各ノードが自分に割り当てられていない領域にアクセスするときには他のノードとの通信による同期が必要である．OpenMPD は配列の全ての領域の同期を取る gather と一部だけの同期を取る sync_sleeve 指示文を提供する．図 2.b に sync_sleeve の動作を示す．多くのアプリケーションでは自分に割り当てられた領域以外にその領域に隣接した数個の要素を参照している（ガウス・ザイデル法など）．その場合，図 2.b のように自分に割り当てられた領域に隣接する数個分の要素の同期を取るだけで良い．sync_sleeve によってそのような同期を記述し，少ない通信で効率的な同期を行うことができる．

B) ループの並列実行によるワークシェアリング

distvar によって割り当てた配列をループで並列に処理するために指示文 for を用いることができる．図 2.a にその動作を示す．配列の割り当てに合わせて（affinity(配列名)の記述による）for 文の作業を各ノードに割り当てている．このように for 文の作業を各ノードに分散させることで実行時間を短縮させることができる．

C) 変数の同期

配列の各要素の総和を求める for 文を並列化したとする．その場合，結果として得られるのは自分に割り当てられた配列要素の総和だけになってしまう．SPMD モデルでは配列と同様，変数（配列以外のベーシックな変数）が各ノードで異なる値を持つ可能性があるため，通信による同期が必要である．OpenMPD は変数の同期に sync_var と reduction を提供する．sync_var は変数の値をあるノードの値で一致させるものである．reduction は総和のようなリダクション演算を記述するためのもので，for 文のオプションとして使うこともできる．

D) タスク並列化

プログラムを幾つかの独立した機能に分けられる場合，指示文 single によって並列化することができる．single は指定された機能（C 言語でいうとブロック文）を一つのノードだけで実行するようにする．各機能を異なるノードで実行させ，処理の高速化を図ることができる．但し，処理の結果は実行したノードのみで有効なので並列処理の後に変数の同期を行う必要がある．

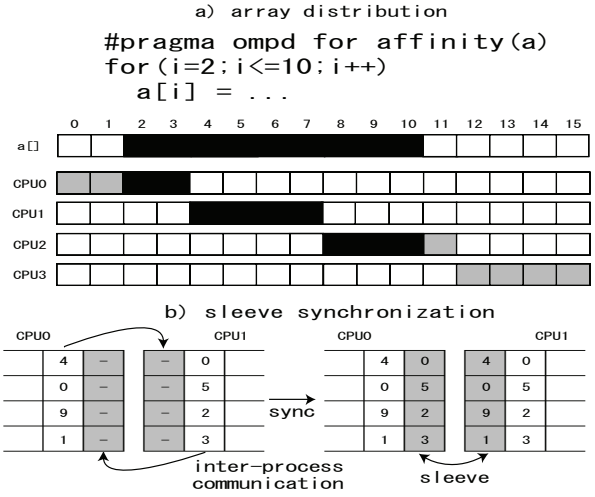

図 2. OpenMPD のデータ並列化手法

3 処理系の実装と性能評価

OpenMPD の処理系は C 言語の拡張として実装されている．Omni OpenMP Compiler[2]をベースにして指示文の実装を行った．処理系は指示文によって得られた並列化に関する情報を用いて OpenMPD ソースを並列コードに変換する．並列コードは通信ライブラリを用いてノード間通信による並列処理を行う．現在の実装では通信に MPI を用いるため，MPI コードが生成される．従ってユーザは性能の最適化のため，OpenMPD のコードの中で MPI 関数を使うことができる．

図 3 に Nas Parallel Benchmark の EP と CG，姫野ベンチマークを用いた性能評価の結果を示す．1GB のメモリと 1000 base-T の Ethernet ネットワークを持つ Intel Xeon プロセッサ搭載のクラスタを最大 8 ノード使用した．

図 3.b にコードのサイズを示す．並列化に大量のコード修正が必要な MPI と比べ，OpenMPD はごく僅かな変更だけで並列化することができる．これは並列化に必要なプログラマの作業量が減り，分散メモリの並列プログラミングを簡単に行えることを証明する．

図 3.a に MPI 版と比べた OpenMPD のパフォーマンスを示す．EP は通信がほとんど発生しないため，MPI 版と OpenMPD 版両方が理想的な性能を示す．姫野ベンチマークの OpenMPD 版は sync_sleeve による同期がアプリケーションと上手くマッチして効率的に通信を行うため，MPI 版とほぼ同じ性能を示す．CG の OpenMPD 版の性能が MPI 版と比べ著しく悪いのは現在の OpenMPD の実装が配列の多次元分割をサポートしていないため，並列化できないループが存在するからである．そのため，今後実装を見直す必要がある．

実行結果は OpenMPD が多くのアプリケーションの並列化を行うに十分な機能を提供することを実証している．このように，OpenMPD はデータ並列に関する典型的な並列化を支援するため，限られた機能で多くのアプリケーションの並列化を記述することができる．

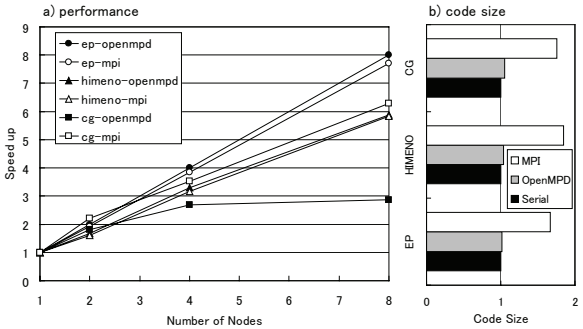

図 3. OpenMPD の性能評価

4 まとめと今後の課題

OpenMPD は MPI プログラミングを簡単に行う手段として，分散メモリ型並列システムに対する簡単なプログラミングモデルを提供する．多くの科学技術計算アプリケーションで用いられるデータ並列の典型的な手法を指示文を用いて記述することができる．

今後の課題として多次元配列分割のサポートや通信ライブラリの最適化など OpenMPD の改良を行う．また，現在の実装では配列の全体領域が各ノードで重複して宣言されるため，メモリを無駄に消費する．各ノードで自分に割り当てられた領域だけを宣言するように OpenMPD の実装を直す必要がある．

最後にノード内並列化を OpenMP で記述した場合のスレッドセーフティを確保して共有メモリと分散メモリのハイブリッドなプログラミングモデルを提供することを目指す．

参考文献

[1] SCASH
http://www.pccluster.org/score/dist/score/html/en/reference/scash/index.html

[2] Omni OpenMP Compiler
http://phase.hpcc.jp/Omni/home.ja.html

[3] Mitsuhisa Sato, Shigehisa Satoh, Kazuhiro Kusano and Yohio Tanaka, "Design of OpenMP Compiler for an SMP Cluster", Proc. of 1st European Workshop on OpenMP EWOMP'99, pp. 32-39, 1999.

[4] Taisuke Boku, Mitsuhisa Sato, Masazumi Matsubara, Daisuke Takahashi, "OpenMPI - OpenMP like tool for easy programming in MPI", Proc. of 6th European Workshop on OpenMP (EWOMP'04), pp. 83-88, 2004.