B-017

分散共有メモリシステムに適した並列言語 MpC
MpC -A Parallel Programming Language for Distributed Shared Memory Systems -

片野 真吾†　　緑川 博子†　　飯塚 肇†
Shingo Katano　Hiroko Midorikawa　Hajime Iizuka

1 はじめに

メタプロセスモデル[1]を実現するための API として，ANSI C を拡張した言語 MpC を提案する. 本報告では, MpC の特徴や UPC[2]との比較，筆者らの開発している SDSM である SMS における MpC の実装方法について述べる.

2 MpC の特徴

2.1 共有変数と分散マッピング

MpC での共有変数の宣言形式を図１に示す．MpC では，宣言の前に予約語 shared を付加することで共有変数を宣言する．また，::の後ろに分割や関連づけるプロセスの情報を付加することで分散マッピング機能を提供する．図２に示すのは，MpC の共有変数宣言例で，実際の割付イメージを図３に示す．ただし，マッピング指定は実装に対するヒントで，実際の割付は実行時の実装システムに依存する．

2.2 メモリ一貫性モデル

MpC では，共有変数の管理は緩和型メモリ一貫性モデルを前提とし，ロックやバリアなどの同期命令はユーザが明示的に記述する．これにより，計算機クラスタ等に従来の共有メモリモデル API を用いたときの，暗黙の同期などによる性能の低下を防ぐことが出来る.

2.3 MpC 定数

MpC には，コンパイル時には不定だが実行時には定数として利用可能な MpC 定数という実行時定数がある．NPROCS はメタプロセスを構成するプロセス数を表し，MYPID は各プロセスに 0～NPROCS-1 のうちの一意の整数が設定される．この２つの MpC 定数は，分散マッピングでの分割数指定やプロセス数の指定に用いることが出来る.

2.4 MpC 標準ライブラリ

synchronize，initialize／finalize，allocate 等の基本関数は，MpC 用の標準ライブラリにより提供する．ライブラリで提供される関数は，実装システムの対応する関数に置き換えられる．これにより，複数の実装システム間での互換性を実現した.

2.5 プログラミングモデル

MpC では，全てのプロセスが同一のプログラムを実行する SPMD プログラミングモデルだけでなく，各プロセスで別々のプログラムを実行する MPMD プログラミングモデルをサポートする．これにより，より柔軟なプログラムの実行が可能となる.

3 UPC との比較

MpC と同様に共有変数の概念を取り入れた言語に UPC がある．表１に MpC と UPC の主な違いを示す.

3.1 分散マッピング

UPC の分散マッピングは，共有変数が配列の場合のみ可能で, ブロックサイズを指定しプロセス 0～N にサイクリック

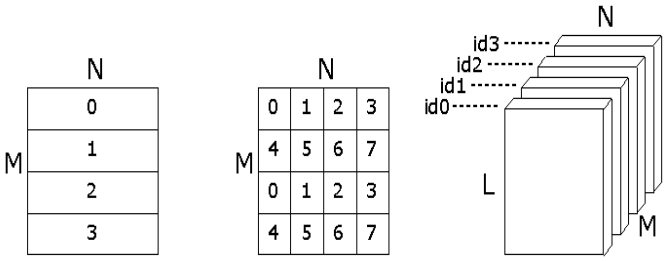
† 成蹊大学大学院 工学研究科 情報処理専攻
Department of Information Sciences , Seikei University

```
shared 型名　変数名[sm]‥[s0]::[dm]‥[d0](st, n)
        変数名[sm]‥[s0]: 変数　各次元サイズ
  [dm]‥[d0]：各次元の分割数 （省略時は分割数1）
  st　　：割付開始プロセス番号(省略時は任意プロセス)
  n　　：割付プロセス数 （省略時は全使用プロセス数）
```

図１．共有変数宣言の形式

```
(a)バンド型
  shared int band[M][N]::[4][](0,NPROCS);
(b)タイル型
  shared int tile[M][N]::[4][4](0, NPROCS);
(c)キューブ型
  shared int cube[L][M][N]::[][4][](0,NPROCS);
```

図２．共有変数宣言例



(a)バンド型　(b)タイル型　(c)キューブ型

図３．割付イメージ

表１．MpC と UPC の比較

	MPC	UPC
プロセス指定	開始プロセス ID，割付プロセス数を指定しサイクリックに割付	配列時は ID 0～N でサイクリック それ以外は ID 0
マッピングの記述性	○	△
メモリコンシステンシ	relaxed	strict/relaxed
ポインタの種類	2種類	4種類
同期関数の提供	ライブラリ	構文，ライブラリ
並列構文	なし	forall，共有変数用 sizeof threadof 等を提供
プログラミングモデル	SPMD/MPMD	SPMD

```
int *p1;
shared int *p2;
```

(a)MpC

```
int *p1;
int *shared p2;
shared int *p3;
shared int *shared p4;
```

(b)UPC

図４ 共有変数ポインタ

に割付を行う．そのため，図３(a)のようなマッピングであれば問題ないが, (c)の様なマッピングは非常に困難になり，(b)の様なマッピングの場合は指定出来ない．一方， MpC の分散マッピングは図１に示したように，配列の次元毎の分割数，割付開始プロセス番号，割付プロセス数を指定しサイクリックに割付を行うので，UPC に比べ柔軟な割付が可能となる．

3.2 ポインタ

MpC では，ポインタ変数が共有されるかどうかでポインタ変数を分類し，ポインタの指す先が共有かどうかの区別はない．例えば，図４のように int 型へのポインタは MpC では２種類，UPC では４種類ある．UPC では，ポインタ変数が共有されるか，ポインタの指す先が共有かどうかの２点でポインタの分類を行っている．MpC では，このような区別はなくても実際上の問題はないと考え，よりシンプルで扱いやすいように２種類のポインタのみとしている．

４ MpC の実装

MpC のコンパイルは図５に示す２段階のパスで行われる．図中には, 筆者らが開発した SDSM である SMS で実際に実行する際のコマンドやファイル名の例も示した．

4.1 第１フェーズ

コンパイルの第１フェーズでは，プログラムモジュールの作成と**共有変数情報ファイル**の作成が行われる．MpC プログラムを，コンパイル時に指定した SDSM の C プログラムに変換し，次にターゲットマシンの C コンパイラによりプログラムモジュールが作られる．この時，共有変数情報ファイルも同時に出力される．共有変数情報ファイルには，MpC プログラムで宣言された共有変数の内容が記述されており，第２フェーズで使用される．

4.2 第２フェーズ

第２フェーズでは，**メタプロセス記述ファイル**と第１フェーズで作成した共有変数情報ファイルを使用して，**メタプロセス実行ファイル**を作成する．メタプロセス記述ファイルには，メタプロセスを構成するために必要なモジュールが記述されており，このモジュール間の共有変数宣言の整合性を共有変数情報ファイルを使ってチェックする．整合性に問題がなければメタプロセス実行ファイルを作成する．メタプロセス実行ファイルは，実際はシェルプログラムで，実装システムに応じた実行コマンドが記述されている．

4.3 実行フェーズ

プログラムの実行フェーズを図６に示す．プログラムの実行には，プログラムモジュール，メタプロセス実行ファイル，**プロセッサファイル**を用いる．プロセッサファイルには，プロセスを起動するマシン名が記述される．

５ おわりに

メタプロセスモデルを実現するための API として MpC を設計した．MpC は，共有変数の明確なスコープ概念，柔軟な分散マッピング機能，明示的な並列処理記述などをユーザに提供する．これにより, 優れた可読性が保たれるうえ，計算機クラスタなどにおいて従来の共有メモリモデル API を用いた場合に比べて性能低下を防ぐことが出来る．現在，SMS へのトランスレータを開発し SMS での実行が可能になっている．さらに， TreadMarks や JIAJIA などの他の SDSM へのトランスレータも作成中である．また，分散共有メモリだけでなく共有メモリ並列システム用に pthread プログラムへのトランスレータも開発中である．

６ 参考文献

[1] メタプロセスモデル－明示的並列処理記述のための分散共有メモリプログラミングモデル－，FIT2003 情報技術レターズ

[2] UPC，http://upc.gwu.edu/

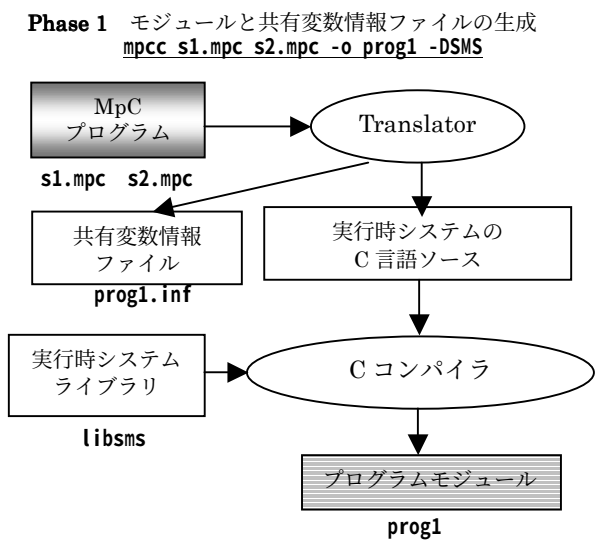


図５ SDSM における MpC のコンパイル

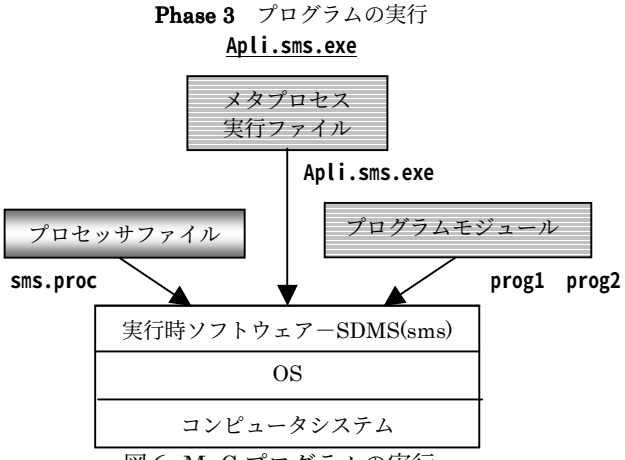


図６ MpC プログラムの実行