B-42

分散共有メモリ SMS における共有変数の分散割付ＡＰＩ

Improvement of the API for a shared data mapping on SMS

成蹊大学大学院工学研究科情報処理専攻

緑川 博子 片野真吾 飯塚 肇

Hiroko Midorikawa　Shingo Katano　Hajime Iizuka

1. はじめに

筆者らは，クラスタ型並列コンピュータシステム上に共有メモリプログラミング環境を実現する，ソフトウェア分散共有メモリ SMS の設計，実装を行ってきた[1]．このようなシステムでは，クラスタ全体で共有データ変数を通常の逐次プログラミングと同等の記述で，アクセスすることができる．しかし，処理性能の効率化を考えると，データ変数の割付場所は，クラスタを構成する計算機ノードになるべく分散して割り付けたほうがよいことが，通信の集中を避ける上でも，処理の局所性を生かす上でも重要であることがわかっている．ソフトウェア分散共有メモリシステムでは，共有データ分散割付のためのインターフェースがないもの[2]や，簡単な関数を提供するもの[3]などがあるが，いずれも，ユーザに提供する API は，malloc 類似形式のみをとっており，共有データそのものの構造を生かした並列処理アルゴリズムに適した分散割付を行おうとすると，ユーザプログラムの記述は煩雑になり，データ分割方式によっては，記述が不可能になる場合もある．

そこで，筆者らは，共有データの構造情報も用いることにより，並列処理で頻繁に用いられるバンド割付，タイル割付などを，従来の逐次プログラムの大域多次元配列宣言のような形式で，ユーザにわかりすい形で，分散割付を行うＡＰＩを構築した．

2. 従来の共有データの割付

従来，専用コンパイラを用いない汎用ソフトウェア分散共有メモリシステムの多くは，図１に示すように，malloc 形式で共有データを割り付けする．しかし，malloc の返したポインタを使った記述は煩雑で，わかりにくい．さらに，図２に示すように並列処理で多用されるデータ領域の分散割付（バンド割付，タイル割付など）を行う場合には，記述が難しい．従来，SMS では，図１に示す２つの割付方法を提供していた．①は，集中割付した後に，共有データの指定アドレスから指定サイズ分のデータを指定プロセス（ページマネージャー）に再割り付けする方法である．これは，先頭アドレスとサイズを自由に指定して割付を行えるので，柔軟性は高いが，図２のような割付をするには，ループによりパラメタを変えて複数回実行する必要がある．また実行時に動的に割り付けを変更する必要がある場に有効であるが、静的割付の場合には，初期割付時にプロセスを割り付けるほうがより効率的である．②は，初期割付時に，全体サイズをブロックサイズ毎に指定プロセスからサイクリックに割り付ける方法である．しかし，これを用いても図２(b),(c)のような縦方向割付やタイル割付は，記述することが不可能であった．

3. 汎用割付関数と共有データの分散割付

今回，図２のような不連続アドレス方向のデータ割付をも可能にする，自由度の高い割付関数 sms_mapalloc を用意した．さらにユーザの使いやすさを向上させるために，分割方向，分割数，割付プロセスの指定などを，わかりやすく記述できる，共有データの分散割付宣言をプログラムに記述できるようにした．

4. 共有データ変数の宣言と割付

宣言の一般形は図３に示す．予約語 shared を従来の変数宣言の前におき，データ変数名，次元の後に：：に続けて，分散割付の次元と分割数を指定する．さらに＜＞で，割付先頭プロセスと割付プロセス範囲を指定できる．：：以下は，省略可能で，この場合には，集中割付することになる．割付プロセスを省略することもできる．省略時は，それまでに割り付けられたプロセスとは異なるプロセスになるべく割り付けられ，割付が分散される．指定プロセスｓｔから割付範囲ｓｔ+ｎ－１の範囲でサイクリックに割り付けされる．ｎが省略されると，使用プロセス全体となる．図４に図２の記述例、省略形の記述例などを示す。

ユーザプログラムは，専用プリプロセッサを通すことで，適切な引数をもった汎用割付関数 sms_mapalloc に変換され，sms_startup というSMSシステム起動関数の直後に展開される．

SMS はページ単位の割付であるため，分散割付宣言によっては１ページ内のデータに複数の異なる割付指定プロセスが指定される場合もありうるが，この場合には，そのページ内で一番大きいサイズを持つデータの指定ページマネージャーに割り付けられる．したがって，ユーザはページ境界を意識せずに記述することで，最善の割り付けがされる．このため，ページ境界割付を前提にした関数[3]などでは問題であった，ユーザの想像とは違う全く関連性のないプロセスに割り付けられるという問題も生じない．

† 成蹊大学　大学院工学研究科，情報処理専攻

5. おわりに

新しい分散割付つき共有データ宣言を用いることで，従来にも増して，記述性が良くなり、効率がよく自由度の高い並列プログラム作成が容易になった.

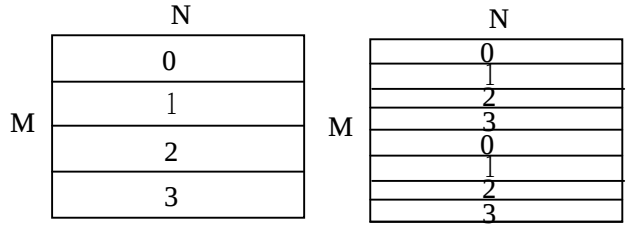
参考文献

[1]緑川, 飯塚:"ユーザレベル・ソフトウェア分散共有メモリ SMS の設計と実装",情報処理学会論文誌ハイパフォーマンスコンピューティングシステム Vol.42,No.SIG9(HPS 3),pp.170-190(2001,8)

[2] P. Keleher, et.al., ：TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, Procs.of the Winter USENIX Conf., pp.115-132, 1994

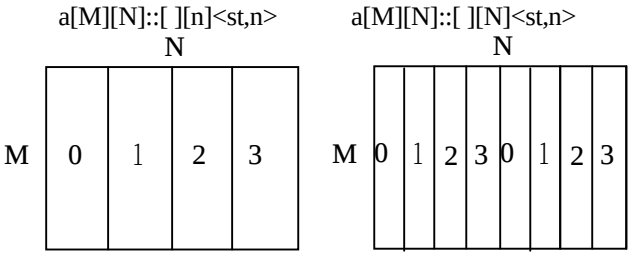
[3] M.R.Eskicioglu et.al., ：Evaluation of the JIAJIA Software DSM System on High Performance Computer Architectures, Procs.of the 32nd Hawaii International Conf. on System Sciences,1999

プロセス数ｎで分割　１行毎にサイクリック分割

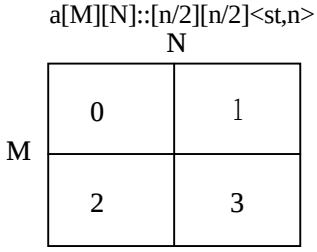


(a)**横バンド割付**

プロセス数ｎで分割　行毎にサイクリック分割



(b)**縦バンド割付**



(c)**タイル割付**

図２ データ分散割付例 (st=0, n=4)

横バンド割付例

```
int (*a) [N];    /* MxN ２次元配列のためのポインタ */
size=M*N*sizeof(int);
①一括割付後の、割付場所移動指定の場合
a=(int (*)[N]) sms_alloc( size,st);  /*  st に集中割付 */
for(i=0;i＜P; i++) /* P 個のプロセスに分散割付　0～P-1*/
   sms_change_page_manager(a[M/P*i], size/P, i );
②初期サイクリック割付
a=(int (*)[N])sms_alloc2(size,size/P,st);
```

図１ 従来の SMS での分散割付

```
shared 型名　変数名[sn]・・[s0]::[dn]・・[d0]<st, n>
 変数名[sn]・・[s0]:  変数　各次元サイズ
[dn]・・[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３　多次元配列データの分散割付の宣言形式

```
２次元配列の分散割付例
プロセス数 P で分割
横バンド割付
  shared  int a [M][N]::[P][ ]<st,n>;  X 方向分割
縦バンド割付
  shared  int a [M][N]::[ ][P]<st,n>;  Y 方向分割
タイル割付
 shared  int a [M][N]::[P/2][2]<st>;  X,Y 両方向に分割
３次元配列の分散割付例
バンド割付（プレート割付）
shared  int a [L][M][N]::[P][ ][ ]<st,n>;  Z 方向分割
shared  int a [L][M][N]::[ ][P][ ]<st,n>;  Y 方向分割
shared  int a [L][M][N]::[ ][ ][P]<st,n>;  X 方向分割
タイル割付（直方体割付）
shared  int a [L][M][N]::[P][Q][R]<st,n>;  XYZ 方向に分割
その他の分散割付、省略例
shared float b[M]::[N]<st>;  st から使用プロセス全体に N 分割
typedef struct data_type Stype;
shared Stype c::<st>;  指定プロセス st に割付
shared double d[L];  任意プロセス割付
```

図４ 分散割付データ宣言例