ＦＭＭにおける融合型処理とソフトウェア構成

1Ｌ－4

渡壁　健　岩根雅彦　阿部　賢治
（九州工業大学）（東芝　北九州工場）

１．はじめに

FMM(Flexible Mesh-network Multi-microprocessors)では，ＰＵ(Processing Unit)を必要な数ずつ使用することで複数のグループを構成し，それぞれ独立にアプリケーションを実行させることができる．また，それぞれについて，アプリケーションに適した処理方式をMIMD型処理，ホスト制御SPMD(Single Program Multiple Data)型処理，ＰＵ制御SPMD型処理，更にMIMD型とＰＵ制御SPMD型の融合型処理から選べる．本稿では，FMMのソフトウェアの全体像を明らかにすると共に，融合型処理について述べる．

２．ソフトウェア構成

FMMにおけるソフトウェア構成を図１に示す．

ホスト側は，イベントドリブン型のマルチタスクで動作するホストＯＳと，ホスト側アプリケーションプログラム（ＡＰ）からなり，ホストＯＳは，周辺装置の制御やメモリ管理をするMS-DOSと，ＦＭＭのための拡張部に分けることができる．

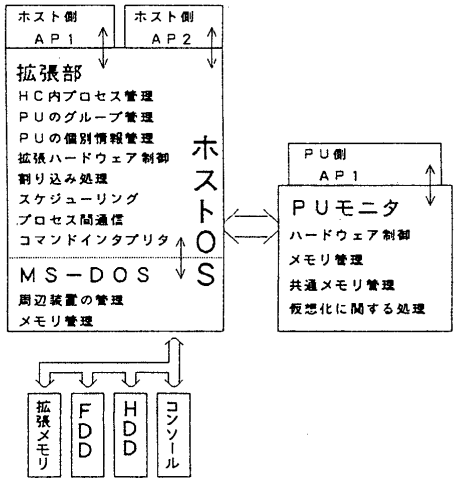

図１　FMMのソフトウェア構成

拡張部には，その機能をＡＰから利用するためのＡＰＩ(Apprication Program Interface)がある．ＰＵの個別情報はＰＵＣＢ(ＰＵ Control Block)によって管理する．ＰＵＣＢは仮想ＰＵと１対１に対応しており，仮想ＰＵと物理ＰＵの変換に必要な情報や，実行中のＡＰの環境情報等が入っている．このＰＵＣＢをリストにして，グループ情報を管理する．各グループにはグループ内共通の情報を持っているグループヘッダがあり，ＰＵの取得，開放の要求がある度に書き換える．またこのグループ情報はハードウェアで物理ＰＵを管理しているＣＡＭのデータ読みだし／書き換えが必要になったときにも参照する．例えば，ＰＵへのデータ転送要求が発生したときは，グループ情報を用いてＰＵ番号からグループ番号の変換を行いＤＴＵ(Data Transfer Unit)を制御し，ホストからＰＵへのデータ転送を行う．

プロセス間通信は各々独立に動作しているアプリケーション間で，ホストＯＳ内に作成されるメイルボックスによって行う．あらかじめ両者で決めておいたメイルボックスネームによりホストＯＳ内にメイルボックスを作成し，そこにメッセージを送るか，またはそこからメッセージを取り出すことによって通信する．

HC内プロセス管理は，HC内のＡＰやMS-DOSファンクション等が割り込みにより中断した場合の情報の保存等の管理を行う．HC内のタスクの切り替えを行うスケジューラは，ラウンドロビン法を用いた，優先順位なしのものを使用する．スケジューリングは割り込みが発生するか，現在の処理が終了した時に行う．

ユーザからの入力は，拡張部内のコマンドインタプリタを通して行う．入力された実行ファイル名は，ローダに引き渡されて，ローディングを行う．まず，プログラムをHC内に読込みヘッダ部分に格納されている，処理方式，使用するＰＵの数や形の情報，共通メモリのベースアドレス，アドレス指定変数の情報を取り出す．必要な数のＰＵを確保し，リロケート情報を元にプログラムをリロケートし，環境情報としてグループ番号等の情報やコマンドライン引き数を付加してＰＵに転送する．ＰＵの実行開始にはＰＵ Start信号を送る．

ＰＵ側は，ＰＵモニタがハードウェア制御，メモリ管理，仮想化に関する処理を行っている．これらもＰＵモニタが提供するＡＰＩによって使用する．ＰＵは，待機状態ではＰＵ Start信号を監視しループしている．ホストＯＳからのＰＵ Start信号により転送されてきた環境情報をもとにＰＵを初期化し，ＡＰの実行を開始する．

メモリ管理は，ＰＵにローディングされたＡＰの先頭／最終アドレスや実行開始アドレス等の情報を管理している．初期化時に設定された後は，ＡＰからのメモリ獲得や開放によって参照，更新する．仮想化に関する処理の一部もここで行っている．ＡＰ中で使用するアドレス指定変数は，ＡＰの先頭アドレスからのオフセットアドレスとして指定する．モニタではこれを先頭アドレスとオフセットアドレスから計算した実効アドレスに変換する．

３．融合型処理

融合型処理は，MIMD型ＰＵからＰＵ制御SPMD型サブグループを生成することで実現するもので，一つのア

Fusion mode and Software Organization on FMM
Takeshi WATAKABE[1], Masahiko, IWANE[1], Kenji ABE[2]
[1]Kyushu Institute of Technology, [2]Toshiba

プリケーション内でMIMD型に適した部分，SPMD型に適した部分を適切な処理方式で実行することができる．融合型処理を実現するためには，サブグループからの割り込みと，MIMD型ＰＵからの割り込みを区別する必要がある．ホストＯＳは，サブグループ作成時にＣＡＭＵ(CAM Unit)のＰＵマスクレジスタを操作することで，サブグループからのBarrier割り込みやグループ転送の対象からMIMD型ＰＵを外すことができる．

融合型のアプリケーション例を図３～５に示す．これらのプログラムは図３のMIMD型ＡＰ１と，図４のＰＵ制御SPMD型ＡＰ２を２×２個使って1000個の整数データの最大値を求め，図５のMIMD型ＡＰ３で同じデータの合計を計算する（図２）．

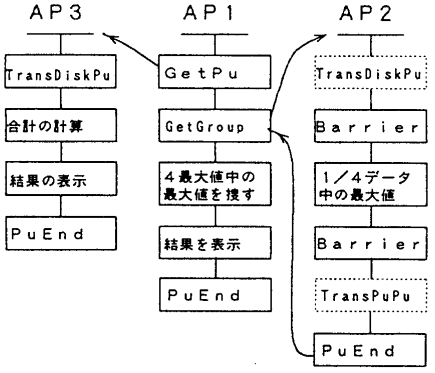

図２　融合型アプリケーション例の動作

まず，ＡＰ１からＡＰ３を起動する．このSVCを受け取ったＰＵモニタはホストＯＳに対し，MIMD型のＰＵ起動のSVCを発行する．また，ＡＰ１は，ＡＰ３の実行終了を待たないので直ちに次の命令を実行する．SVCを受けたホストＯＳは未使用のＰＵリストから一つＰＵを確保し，MIMD3.EXEを実行させる．

ＡＰ１は，次にサブ・グループの起動を行う．２×２のSPMD型ＰＵを確保し，ＰＵSPMD2.EXEを実行させる．マスタＰＵ用とスレーブＰＵ用のプログラムは同じものを使用する．if (IsMaster())の部分によってＰＵモニタが持っている．マスタかどうかの情報を参照して制御を行う．プログラムを共通にすることによって開発コストは下がり，実行時のローディングも一度のブロードキャストですむので，SVCの発行回数，実行時間が少なくなる．

ＡＰからのSVCは，ＰＵモニタが受けサービスを行うがデータ読込み等，必要があればホストＯＳにSVCを発行する．その際，仮想ＰＵから物理ＰＵへのアドレス変換や，データの一時転送等は前もってＰＵモニタが行う．

ＡＰが終了するときは，PuEndを実行する．引数によって，同一グループの他のＰＵと同期を取って終了するか，直ちに終了するかの２種類の方法がある．MIMD型のＰＵの場合，常に同期を取らずに終了する．ＰＵ制御SPMD型の場合，全てのＰＵが同期を取って終了する指定なら，PuEndを受け付けたＰＵモニタ内部でPuEndのためのHsync同期が実行され終了する．直ちに終了する場合は，マスタＰＵからのみホストＯＳへPuEnd SVCを発行する．スレーブＰＵは何もせずに次の実行待機状態となる．これによりSVC割り込みの回数を少なくしている．ホストＯＳはサブグループの終了を検出すると，それを起動したＰＵへStartGroup SVCが終了したことを知らせる．このとき起動したＰＵが終了を待っていれば次の処理を実行することになる．

```
/* MIMD1.C */
void main(int argc, char *argv[])
{
    ABS int     *iPuMax = (int*)0x02000000UL;
    char        szMsg[100];
    char        szArgs[3][20];

        strcpy(szArgs[0], "");/*--引数なし--*/
        GetPu("MIMD2.EXE", szArgs
                        , NO_WAIT, WAIT);
        GetGroup("SUSPMD.EXE", szArgs
                        , 2, 2, WAIT);
        StartGroup(WAIT);
        {
        （４つのＰＵの最大値中の最大値を選ぶ）
        }
        PuEnd();
}
```

図３　MIMD型処理用プログラムＡＰ１

```
/* ＰＵSPMD2.C */
void main()
{
    COMMON      0x20000;/*共通変数領域の指定*/
    ABS int     *iPuMax = (int*)0x02000000UL;
    int         iBuff[1000];
    int         *Shared = (int*)0x20000000;

    if (IsMaster())  {
        TransDiskPu(iBuff, "DATAFILE"
                        , sizeof(int), 1000);
        for (i = 0; i < 1000; i++)  {
            Shared[i] = iBuff[i];
        }
    }
    Barrier();
    {
        (1000/4 個のデータの最大値を探す)
    }
    if (IsMaster())  {
        TransPuPu(Info.PaPuNo, iPuMax
            , &Shared[1000], sizeof(int), 4);
    }
    PuEnd(SYNC);
}
```

図４　ＰＵ制御SPMD処理用プログラムＡＰ２

```
/* MIMD3.C */
void main()
{
    int     iData[1000];

    TransDiskPu(iData, "DATAFILE"
                    , sizeof(*iData), 1000);
    {
        （合計を計算する）
    }
        PuEnd();
}
```

図５　MIMD型処理用プログラムＡＰ３

４．むすび

本稿では，FMMにおける基本ソフトウェアとアプリケーションの全体像を述べた上でアプリケーションの具体例を示しながら，融合型処理の動作について説明した．FMMは，現在ホストＯＳおよびＰＵモニタの設計を終え，コーディングに入っている．

今後はユーザインターフェイス等の部分について，より検討していく必要がある．

参考文献

［１］高橋義造編，並列処理機構，丸善，１９８９