Linux 版の MBCF 通信機構について

松本　尚 [1]

1. はじめに

筆者のオリジナル並列分散 OS である SSS-PC[2] および SSS-CORE[1] の基盤となる通信同期機構として採用していた Memory Based Communication Facility (**MBCF**)[3][4]の Linux OS 版を開発するとともに，MBCF を 64bit アドレスに対応させた MBCF64 を新たに規定して，64bit アドレスの Linux OS にも対応可能にした．本稿では，MBCF の概要を述べ，64bit アドレス対応のための拡張方式を解説し，Linux 版の開発基本方針と実装時の注意点について述べる．

2. MBCF の概要

MBCF は Gigabit Ethernet のような高速ネットワークで接続された PC クラスタやワークステーションクラスタにおいてマシン境界を越えた共有メモリアクセス環境を実現するための通信同期機構として 1994 年に考案され，1996 年に発表された[1]ものである． MBCF を実装するに当たっては，特殊なハードウェアを必要としない．通信のための API が共有メモリ操作となっている点が，socket に代表されるこれまでの通信 API とは大幅に異なる点である．OS が提供する通信機構自体が共有メモリ操作を基本とする API を直接的に提供することにより，高機能，低遅延，低オーバヘッド，ロックフリーの通信同期機構の実現が可能になる．ロック操作が不要になる大規模分散共有メモリの実現手法である MBCF は夢の分散共有メモリ実現方法と呼べるかもしれない．

要求元タスク側でシステムコールを介して MBCF 操作要求パケットの送信を行っている．この方式が低オーバヘッドで実現できたのは，パケット送信処理に不要な処理をシステムコールから完全に取り除き，コード実装に階層構造といったオーバヘッドコストを持ち込むようなものは極力排除してコード最適化を実施した結果である．

同様に，要求先タスク側つまりパケットの受信側ノードにおいて，MBCF ではユーザプログラムが関与しなくても処理が完結する形式を採っている．つまり，OS カーネル内の受信割り込み処理で基本的な MBCF 操作要求は要求先タスクのメモリ空間に対して実行される．要求先タスクがスケジューリングされていない状況においても，メモリ空間を切り替えて操作対象アドレスのメモリに操作を加える必要がある．このアドレス空間切替えや特権モードからのユーザ権限アクセスのコストを，1994 年頃の CPU から実装された機能により，大幅に低く抑えることが可能になったため，MBCF は低オーバヘッドで実装できる．

3. MBCF の 64bit アドレス拡張

オリジナルの MBCF は，取り扱うタスク内の論理アドレスを 32bit 長に設定していた．しかし，2021 年現在では，PC の多くに 64bit CPU と 64bit OS が搭載され，実メモリも 4GByte を越えて実装されることが普通になっている．このことから，MBCF の 64bit 論理アドレス対応は不可避だと考えて，MBCF のパケットフォーマットを見直して，64bit アドレス対応を行った．以下，64bit アドレスの MBCF を MBCF64，従来の 32bit アドレスの MBCF を MBCF32 と呼称し，双方に共通な議論や特にアドレスサイズを区別する必要のない場合は，単に MBCF と呼称する．

MBCF64 のパケットフォーマット設計時に，プログラム書き換えの手間を考慮して，MBCF32 のフォーマットと極力同じ形式にするように注意した．Ethernet ヘッダの Packet Type フィールドは当然新たな MBCF64 用のプロトコル番号を割り当てた．64bit の論理アドレスを格納する Destination Logical Address は 8byte 境界であることが望ましいため，MBCF Data Length と Destination Logical Address のパケット内の順序を逆にした．あと，ノード単位の送信済みパケット番号の END_P と受信済みパケット番号の RECV_P のフィールドサイズを 8bit から 16bit に拡大した．SSS-PC の MBCF32 では使用したことがない 10GbE や 100GbE も Linux 版の MBCF64 の対象となるため，確認応答なしで発行可能なパケット数を増加させるために，フィールドサイズを 8bit から 16bit に拡大した．

4. MBCF/Linux の開発基本方針

Linux 上に実装した MBCF 通信同期機構を SSS-PC 上のオリジナル MBCF と区別するために MBCF/Linux と呼ぶことにする．MBCF/Linux の開発に当たって，以下の方針を掲げることにした．

1). API は極力オリジナルの MBCF と共通にする．
2). Linux カーネルの改変は極力回避する．
3). MBCF プロトコルスタックは Loadable Module によるデバイスドライバとして実装する．
4). MBCF プロトコルスタックをキャラクタデバイスとして実装し，システムコールは ioctl で実装する．
5). スワップアウトされたメモリは操作対象にしない．
6). sk_buff 構造体によるパケット送受信を行う．

2).の項目は MBCF を可能な限り多くの人々に使用しても

1 奈良女子大学
Nara Women's University

らうためには, Linux カーネルへの MBCF 専用の変更はない方がよいと判断した．3).の項目は，MBCF コードの一部の権利は大学発ベンチャー企業が保有しているため, 筆者の一存ではオープンソース化ができないという理由もある．4).の項目は，MBCF の API 操作は，新規にシステムコールを追加しなければ, ユーザ空間から多くのパラメータを一度にカーネルに引き渡せる ioctl による実装にならざるを得ない．また，ブロックデバイスを選択する理由が無いため, より一般的なキャラクタデバイスとして実装する．5).の項目に関しては, MBCF は 1 台のマシンではメモリが足りないような大規模アプリケーションでこそ活かされる通信機構であるため, 個人利用を想定したスワップ機能への対応は不要と判断した. MBCF で操作するメモリに関しては，mlock によって主記憶にピンダウンしておくものとする． 6).の項目は NIC（Network Interface Card）の Linux 用デバイスドライバはすべて sk_buff と呼ばれる構造体に基づいてパケットの送受信を行っている. このため, sk_buff とその処理関数に基づいた実装を行えば, Linux 用のデバイスドライバが存在するすべての NIC が MBCF で使用可能ということになる．

5. MBCF/Linux 実装上の注意点

オリジナルの MBCF が実装された SSS-PC は完全にオリジナル OS であるため, UNIX や Linux とは相違点が多数存在する．MBCF/Linux の開発においては，これらの相違点に注意を払う必要がある．以下に，MBCF/Linux 実装上の主な注意点を列挙する．

5.1 プロセス ID と物理タスク ID の違い

ID を指定して生成できないプロセス ID では，物理タスク ID の代わりに使用することはできないため，明示的に ID を指定可能な「物理タスク ID」を MBCF/Linux において，プロセス ID の別名として，設定可能にする．

5.2 Linux プロセスの MBCF タスク化

物理タスク ID が規定されて初めて Linux プロセスは MBCF による操作対象タスク（プロセス）となる．他のタスクに対する MBCF 操作要求パケットも物理タスク ID が設定された後にしか発行できない. MBCF デバイスの close によって, それ以降の MBCF パケットの送受信はできなくなる．MBCF の操作対象領域をピンダウンしたり，初期値を代入したりする操作は，この物理タスク ID 設定前に行うことにより, 外部タスクからの MBCF 操作とプロセス自身による領域初期化の競合が回避できる．

5.3 物理ノード ID の選択

MBCF はノード（マシン）を跨いで実行される共有メモリ操作であり，要求先タスクの指定はあくまでも（物理ノード ID, 物理タスク ID）の組である．MBCF 専用の ARP 表を用意することにより，物理ノード ID としてユーザが扱いにくい IP アドレスを選択する必要性はなくなる．このため，SSS-PC 同様に 32bit の 1 から始まる物理ノード番号を物理ノード ID として各マシンに特権ユーザが付与する．

5.4 MBCF デバイス

64bit Linux に対しては MBCF デバイスとして 64bit 論理アドレスの MBCF64 デバイスを使用し，32bit Linux に対しては 32bit 論理アドレスの MBCF32 デバイスを使用する．

5.5 プロセス構造体とネットワークデバイス構造体

Linux のプロセス構造体およびネットワークデバイス構造体の情報のみでは，MBCF パケットの送受信を処理することはできない．SSS-PC では，これらの MBCF 操作に必要な情報は該当する構造体の中に含まれているか，もしくは構造体内のポインタから辿ることができる．Linux の構造体自体を拡張することは可能であるが，このことは Linux カーネルの書き換えを招き, 4 章の基本方針に矛盾する．Linux の構造体では不足する情報は，MBCF/Linux のための新たな構造体内のメンバとして定義して，この構造体から Linux の構造体へのポインタを張ることにする．

5.6 受信割り込み時のメモリ操作方法

MBCF では，MBCF パケットの受信割り込み時に，操作対象タスクのメモリ空間に切り替えを行って，ユーザ権限でメモリを読み書きする．メモリ空間の切り替えに関しては CPU 依存のコードになってしまうが，Linux においてもこれらの機能も受信割り込み時に起動される MBCF 受信のためのタスクレット内で実現できることが判った．

6. おわりに

筆者のオリジナル並列分散 OS SSS-PC の基幹となる通信同期機構 MBCF の Linux OS 版である MBCF/Linux の開発を行った. 64bit の Linux OS に対応するために, MBCF 機構も 64bit 論理アドレスへの対応を行った．本稿では, MBCF の 64bit アドレス拡張と Linux 版の開発基本方針と OS の違いから派生する実装上の注意点について述べた．高スループットと低レイテンシを両立し，高機能な共有メモリ操作とロックフリーの不可分操作が可能な MBCF 通信機構を広く使用されている Linux 上で動かすことに成功した．

参考文献

[1] 松本 尚, 他: 汎用超並列オペレーティングシステム: SSS−CORE --- ワークステーションクラスタにおける実現 ---. システムソフトウェアとオペレーティングシステム研究会報告 No.73-20, 情報処理学会, pp.115-120 (August 1996).

[2] 松本 尚: 次世代オペレーティングシステム SSS-PC の開発 --- IA32 用カーネルアーキテクチャ ---. ITX2002 Summer 論文集（CDROM），情報処理振興事業協会, (June 2002)

[3] Matsumoto, T. et al.: MBCF: A Protected and Virtualized High-Speed User-Level Memory-Based Communication Facility. Proc. of the 1998 ACM Int. Conf. on Supercomputing, pp.259-266 (July 1998).

[4] Matsumoto, T.: A Study on Memory-Based Communications and Synchronization in Distributed-Memory Systems. Dissertation Thesis, Graduate School of Science, Univ. of Tokyo (February 2001).