第3章 ラズパイやA64FXマシンでの実装実験も

Linuxのプロセス間通信を高速に実現するxpmem

張 雷，森住 能久

並列処理においては，プログラム実行単位ごとに計算結果や経過を共有するためにデータを相互にやり取りする必要があります．

プログラム実行単位ごとに通信する仕組みはいろいろありますが，本稿ではその中でもLinux上でアプリケーション間のデータ交換を行う仕組みである，プロセス間通信について解説します．

さらにプロセス間通信を高速化するためにコンピュータでも利用されている，Linuxカーネルの拡張モジュールxpmemをラズベリー・パイ上で動かしてその性能を測定します．

プログラム間で通信するしくみ「プロセス間通信」とは

■ プロセス間通信の基礎知識

プロセス間通信とは，プログラムの実行単位の1つであるプロセスと別のプロセスとの間でデータ送受信を行うことです．本章で紹介するxpmemはLinuxのプロセス間通信を高速に実現するための拡張モジュールの1つです．

● Linuxでのプログラムの実行単位

Linuxにはプログラムの実行単位にプロセスとスレッドがあります．

▶スレッドの場合

アドレス空間を共有しているので，スレッド間でのデータのやり取りが比較的容易です．実際には，データのやり取りに同期が必要になるため，それほど単純ではありませんが，データのやり取り自体に特別な仕組みを用いる必要はありません．

▶プロセスの場合

それぞれのプロセスが独立したアドレス空間を持つので，データのやり取りにはOSの提供する特別な仕組みが必要です．Linuxでも幾つかの仕組みが提供されています．

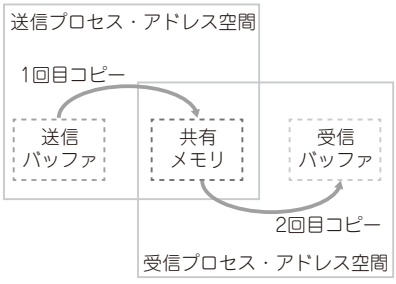

図1 共有メモリ方式プロセス間通信

■ Linuxで使える代表的なプロセス間通信

● その1：共有メモリ

共有メモリは，複数のプロセスからアクセスできるメモリ領域です．共有メモリを介して複数のプロセス間でデータをやり取りできます(**図1**)．

データを送る側のプロセスが共有メモリにデータをコピーした後，データを受け取る側のプロセスが共有メモリから自プロセスが管理する領域にデータをコピーする必要があるため，メモリ・コピーが2回以上発生します．

Linuxでは，POSIX共有メモリ(shm_open, shm_unlinkなど)やSystem V 共有メモリ(shmget, shmopなど)と呼ばれる仕組みでこの機能を提供しています．

共有メモリ自体はアクセスを排他制御する仕組みを持たないため，複数のプロセスから同じメモリ領域に同時にアクセスするとデータの一貫性が損なわれます．従って，アクセスの排他制御は別の方法を用いて行う必要があります．

● その2：パイプ，FIFO(名前付きパイプ)

パイプやFIFOは単方向のプロセス間通信を提供します．

▶pipeシステム・コールを使う

パイプはpipeシステム・コールにより作成されます．このシステム・コールは読み出し側と書き込み側の2つのファイル・ディスクリプタを返します．このファイル・ディスクリプタ間でデータのやり取りを行