並列マシン Cenju-3 上でのユーザレベル IPC に関する考察
— Mach Microkernel をベースとする並列 OS DenEn での実現 —

3H－9

荒木 宏之　高野 陽介　小長谷 明彦
日本電気(株)C&C 研究所

1　はじめに

並列マシン Cenju-3 をターゲットとしたユーザレベル通信機構 NODO について述べる。NODO は、Mach Microkernel をベースとする並列 OS、DenEn[1] 上で提供される。本論文では、セキュリティ、API、キャッシュ・コントロール、および、マルチユーザ化等の観点から、ユーザレベル IPC 実現上の課題を整理し、それらに対する NODO での解決策を考察する。

2　ユーザレベル IPC

Cenju-3 は、R4400 をベースとする並列マシンであり、各プロセッサ・エレメント (PE) は、高速なネットワークで接続されている。並列マシンの有効利用のためには、高速な IPC が不可欠であるが、汎用の OS が提供する IPC は性能的に不充分であることが多い。例えば、Mach MK が提供する通信のメカニズムは、

- システムコールによるコンテキスト・スイッチ、
- 通信バッファ管理、
- 通信データのコピー、

等、多くのオーバヘッドを伴い、通信速度を著しく低下させる。この様なオーバヘッドを軽減するために、ユーザレベル IPC を実現する試みがなされている [2][3][4]。

ユーザレベル IPC は、通信バッファの管理やデータの保証などをユーザが行ない、或は、ユーザプロセスが直接ネットワークデバイスを利用するなどして、通信のオーバヘッドを減らすものである。

DenEn においても、これらのオーバヘッドを軽減した、ユーザレベル IPC 機構 NODO の実現を進めている。

3　設計思想

ユーザレベル IPC の実現には、様々な課題が存在する。以下、それぞれの課題について、NODO の設計思想を示す。

(1) マルチユーザ化

ネットワークデバイスを、マルチユーザで利用するためには、デバイスへのアクセスの排他制御などを行なう必要がある。このため、送信処理をサーバコールとして実現し、NODO サーバが一括管理する。

(2) 高速性

NODO では送信処理が、サーバコールとなるので、コンテキストスイッチのオーバヘッドが発生するが、通信バッファの管理やデータのコピーなどが軽減されるので、性能の向上が見込める。また、ハードウェアの通信機構が複数ある場合、NODO は自動的に適切なものを選択し利用する。Cenju-3 では、ネットワークデバイスがメモリの保護を実現しているバッファ (優先バッファ) が、各 PE に一つずつ存在するので、これを利用する際には NODO サーバを介さずに送信を行なうことが出来、更に高速な通信ができる。

(3) セキュリティ

ユーザレベル IPC では、ユーザが OS をバイパスし、直接ネットワークデバイスにアクセスすることになる。ネットワークデバイスが他の PE のメモリを自由にアクセスできる場合には、メモリの保護が必要である。Cenju-3 のネットワークデバイスは、他 の PE 上のメモリに対する DMA 機能を持っているので、メモリの保護機構の実現が必要である。NODO では、データの送信を、NODO サーバで一括処理ため、ユーザによる不当な書き込みを防ぐことが出来るので、メモリの保護が実現される。

(4) キャッシュ・コントロール

ネットワークデバイスが CPU とメモリを共有しながらデータの並列転送が出来る場合、ネットワークデバイスと CPU との間でデータの一貫性を保つ必要がある。Cenju-3 では、キャッシュはネットワークデバイスからの DMA により自動的には更新されないので、ソフトウェアでキャッシュの一貫性を保証する。

"User-level IPC on Cenju-3 — The implementation on Mach MK based parallel OS DenEn —"
Hiroyuki ARAKI,Yousuke TAKANO,Akihiko KONAGAYA
NEC Corporation. C&C research laboratories.
4-1-1 Miyazaki,Miyamae,Kawasaki 216,Japan

(5)API

ユーザレベル通信の高速化のために、出来るだけ簡素なAPIを利用する。MPIの様な高度なAPIは、NODOの上位のレベルでライブラリとして提供する。

4 実現

PE#1上のprocess#1から、PE#2上のprocess#2へ、通信する場合を示す(図1)。

process#1がNODOサーバにprocess#2へのバッファのマッピングを要求する(nodo_map())。NODOサーバは、優先バッファが空いていれば、それを、さもなければ、物理メモリから必要な分のページを確保し、process#2の仮想アドレス空間にマッピングする。獲得した物理アドレスは、ページテーブルに格納し、送信バッファに関連付ける。

process#1は、送信データを作成し、nodo_send()を発行する。優先バッファからの送信であれば、NIFがアドレス変換を行ない(図中破線)、そうでない場合には、NODOサーバ内のページテーブルを用いて、NODOサーバが変換を行なう(図中実線)。NODOサーバ、或はNIFがアドレスを変換するため、不正なアドレスへのDMAを防ぐことが出来る。優先バッファを用いた送信の場合はNODOサーバを経由すること無く、process#1から直接送信が出来る。

process#2は、nodo_recv()を行なう。nodo_recv()は、内部でキャッシュの更新を行なう。

ページングは、NODOサーバを利用して行なう。ページャは、NODOサーバに受信バッファをページアウトすることを通知する。NODOサーバは、受信バッファへの送信が終了するのを待って、対応する送信バッファへのnodo_send()要求を禁止し、ページャにページアウトの許可を出す。受信バッファの無いnodo_send()を受けると、NODOサーバは、要求したプロセスを受信バッファがページインされるまで、サスペンドする。ページャは、受信バッファがページインされたら、それをNODOサーバに通知する。

5 利用方法

上記メカニズムを利用するAPIとして、psh(Parallel SHell)を実現した。pshはDenEnのIPCのコネクションジェネレータであり、ライブラリとして提供される。

```
buffer=[buf 0x1000];
task[1]=["process1" 1 [buf s]];
task[2]=["process2" 2 [buf r]];
```

の様な簡単なスクリプトファイルを解釈し、IPCのためのコネクションを生成する。

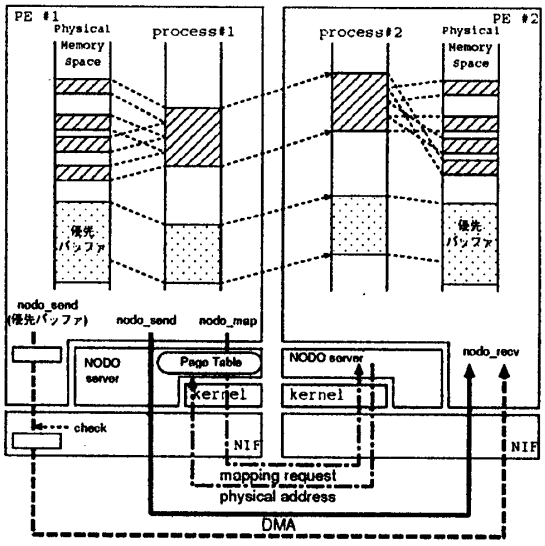

図 1: ユーザレベルIPCのモデル

6 終わりに

現在、NODOのための基本的部分の試作を完了している。MachのIPCでは1wordの通信が、およそ300μsを要するのに対して、NODOのユーザレベルIPCでは、およそ25μsで実現できることが判った。

今後は、NODOサーバをバイパスする優先バッファの利用などを含む、NODOの全ての機能について実装を行ない、評価を行なう予定である。

参考文献

[1] 高野他、"Machマイクロカーネルをベースとした並列OS DenEnの実現"、情報処理学会第50回全国大会論文集.

[2] Matthias A.Blumrich, Kai Li, Richard Alpert, Cezary Dubnicki, and Edward W.Felten, "Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer", Proceedings of the 21st Annual Symposium on Computer Architecture, pp 142-153, April 1994.

[3] Matthias A.Blumrich, Cezary Dubnicki, Edward W.Felten, Kai Li, R.Mesarina, "Two Virtual Memory mapped Network Interface Designs", Hot Interconnects II Symposium Record, pp 134-142, August 1994

[4] Jon Beecroft, Mark Homewood, Moray McLaren, "Meiko CS-2 interconnect Elan-Elite design", Parallel Computing, pp 1627-1638, November 1994.