ハイパフォーマンス
コンピューティング 76－1
（1999．5．14）

コモディティネットワークに基づく並列入出力システム

松原正純† 沼寿隆†
板倉憲一†† 朴泰祐†

超並列計算機における大規模科学技術計算によって生じる膨大な量の計算結果を，実時間的に保存・表示するためには，現在よりも高性能かつ柔軟な入出力システムが望まれる．本稿では，その要件を満たすシステムを実現するために開発を進めている，並列入出力システムについて報告する．

ネットワーク媒体としては，既にコモディティ化されていてコストパフォーマンスに優れた100base-TX イーサネットを採用し，またこのネットワークを多重化することで，ネットワークのバンド幅を増強する．これにより，安価ではあるが全体として高速なシステムを構築可能である．実装した並列入出力システムを性能評価した結果，複数ある入出力装置を有効に利用でき，ネットワークの並列度に見合った性能向上が得られることが確認された．

Commodity Network based Parallel I/O System

MASAZUMI MATSUBARA,† HISATAKA NUMA,†
KEN'ICHI ITAKURA†† and TAISUKE BOKU†

Through and after a large scale computation on a massively parallel processor(MPP), a large quantity of data will be produced. Thus, it is desired to develop a fast and flexible Input/Output(I/O) to process such a large amount of data.

In order to improve the bandwidth of the external network of an MPP, we utilize multiple network interfaces that are based on commodity network. Building a parallel I/O system on that facility, it is possible to reduce the cost and to get better performance. As a result, the parallel I/O system could offer users fast and flexible I/O system.

1. はじめに

超並列計算機(Massively Parallel Processor : MPP)における科学技術計算の過程で生じる，大量のデータに対する入出力処理の高速化は，超並列計算機をより利用し易くするための重要な技術的要件である．例えば，超並列計算機の外部環境としてある，ビジュアリゼーション・サーバやファイル・サーバ等との間で実時間的にデータ処理することを考慮すると，現在の入出力機構では能力不足と言える．そこで我々は，超並列計算機 CP-PACS[1] のプロトタイプ機と共有メモリ型並列計算機である Origin-2000 及び Onyx2 を用いて，高性能かつ柔軟で，しかも安価な超並列計算機向け入出力システムの開発を目指している．

従来のMPP システムでは，ネットワーク・インターフェースを備える少数の入出力プロセッサに全ての入出力データが集められ，そして処理されてきた．この時点で，それまで複数あったデータ流が単一流にまとめられることになり，結果としてこの部分がシステム全体のボトルネックとなってしまう．しかし近年は，コストパフォーマンスに優れた小規模構成の並列システムを MPP システムの外部マシンとして利用可能である．この場合，ネットワークを多重化することにより，最後まで，もしくはその直前までを並列にデータ処理することが可能である．

我々は，上記のように，並列システム間を並列ネットワークで結合した環境下で，超並列計算機が備えている多数の入出力プロセッサを並列運用することにより，最大限の性能を引き出せる入出力システムの開発を行なっている．また，ユーザからはそれらの運用を特に意識せずに，外部環境との柔軟なやりとりを容易に実現できるようなアプリケーション・インタフェースを提供する．さらに，ハードウェア開発にコストと時間をかけず，近年のコモディティ化したネットワーク媒体や接続技術を積極的に利用することによる，コストパフォーマンスに優れたシステム構築も目標の一つとする．

本稿では，現在我々が開発を進めている並列入出力

† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†† 筑波大学 計算物理学計算センター
Center for Computational Physics, University of Tsukuba

システムの実装と性能評価について報告する．

2. システム概要

過去の超並列計算機プロジェクトにおいては，多くの場合，入出力系のために専用ハードウェア・ソフトウェアが導入・開発されてきた．これに対し，現在の汎用のバスあるいはネットワーク技術は，これらの研究において目標とされてきた水準に十分達しており，今後の超並列計算機用入出力系としては，むしろこういった汎用の製品 (commodity) をハードウェア面で積極的に利用していき，ソフトウェア的な手法でその利用効率を高める方向で専用化を進めるのが開発期間・コストに対する効率の点で有利であると考えられる．本研究ではこういった指針に基づき，コモディティ・ベースのハードウェアを利用していく．特に，100 Mbit/秒程度の通信性能を持つ 100base イーサネットや ATM は，その汎用性と簡便さからパーソナルユースのレベルにまで普及しているため，コストパフォーマンスの点で上記ネットワーク技術を大きく上回っているのが現状である．また，イーサネットに関しては Gigabit 化が実用化されており，そのコストも急激に低下しつつある．

その一方で，分散メモリ型超並列計算機である CP-PACS[1] 及びそのプロトタイプ機には，入出力処理のためのインタフェースを持つ専用プロセッサが多数用意されており，大容量の内蔵ディスクはもちろん，多数の外部入出力チャネルをサポートすることが可能になっている．そこで，100base イーサネットのようなコストパフォーマンスに優れたコモディティ技術を，大量の並列入出力プロセッサに適用することにより，トータルとして十分な通信性能を持ち，かつ安価な並列入出力チャネルを提供することが可能であると考えられる．

また，外部環境として存在する，並列ワークステーションのような他の計算機資源においても，その計算機性能に応じた並列なネットワーク環境が提供されつつある．例えばワークステーションクラスタ等では，基本的に 1 プロセッサ当たり 1 つのネットワークインタフェースが実装されているのが普通である．従って，超並列計算機とこれらのシステムを一対一に結合する際に，並列ネットワークを用いて多数のデータ流の同時転送を行うことが可能である．さらに，ネットワークと同様に低価格化が進んでいる高速スイッチを介することにより，複数の計算資源間を同様に並列接続した，高性能かつ柔軟なネットワークシステムを提供することが可能である．

以上の環境を統合することにより，超並列計算機上で生成された複数のデータ流は，並列ネットワークを介して，途中で逐次化されることなく並列のまま外部マシンに送ることが可能となる．しかし，このような

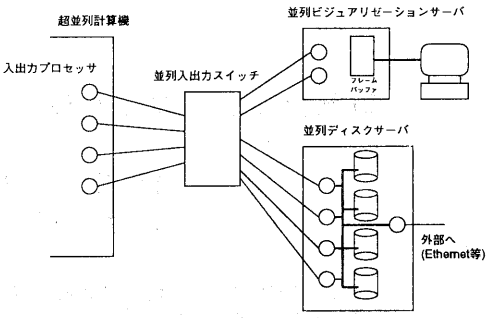

図 1　並列入出力実験システム

並列ネットワーク環境下でのアプリケーション開発は，効率良く並列ネットワークを利用するために相手側のアプリケーションやチャネル数などを考慮し，最適化されるようにプログラミングしなければならず，ユーザにとって大変な重荷となる．そこで，このような複雑なシステム情報を内部で吸収し，ユーザからは簡便なアプリケーション・インターフェースによって利用可能な入出力システム開発は必要不可欠である．

3. ハードウェア構成

上述した入出力システムを実現するために，図 1 に示すような実験環境を構築し，その上に並列入出力プログラミングを行なうための並列入出力システムを設計・実装する．

図中，データ生成システムは超並列計算機 CP-PACS[1] のプロトタイプ機 Pilot-3 である．Pilot-3 は演算プロセッサが 128 台，そしてそれらの入出力を処理するプロセッサが 8 台，計 136 プロセッサから成る．また，入出力プロセッサのうち 4 台については，100base-TX イーサネット・インタフェースを実装している．これを 100base-TX スイッチを経由し，並列ディスクサーバである Origin-2000 (8 プロセッサ) 及び並列ビジュアリゼーションサーバである Onyx2 (2 プロセッサ + 1 ラスタマネージャ) と結合している．これらのワークステーションにも，各々4 ポートの 100base-TX イーサネット・インタフェースが装備されており，各入出力チャネルは互いに 4 本ずつの 100base-TX により並列結合されている．

このようなハードウェア構成の上に，並列入出力システムのプロトタイプを構築する．以下，並列入出力システムについて詳細を述べる．

4. 並列入出力システム

現状では，データ生成システム (超並列計算機) によって生成されたデータは，一旦内蔵ディスク等に格納されるケースが多く，この点がシステムにおけるデータ逐次処理を生み出し，ボトルネックとなる可能性が

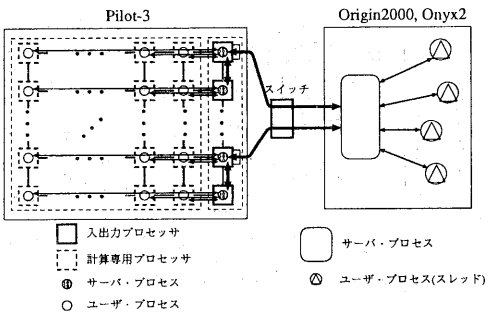

図 2　並列入出力システムの構成

高い．そこで，本システムではこのような逐次化を排除するために，アプリケーション・プログラムから直接，データを並列入出力チャネルを用いて外部環境に送出し，このような逐次化フェーズをなくすことを目的とする．

並列入出力システム ("PIO システム" と呼ぶ) の構成を，**図 2** に示す．本システムは，マシン間でのデータ入出力を行なうサーバ・デーモン (PIO サーバ) と，ユーザ・プロセスと PIO サーバ間との通信を行なうための API （Application Program Interface）から成る．

4.1 PIO サーバ

分散メモリ型並列計算機である Pilot-3 では，各入出力プロセッサ上にサーバ・デーモンを起ち上げ，サーバ間で協調動作させることでシステムとしての一貫性を保つ．また，各サーバは，3 次元格子状（Pilot-3 の場合は 2 次元）に並んだ演算プロセッサのうち，自分と y 座標の等しい演算プロセッサ群の入出力を受け持つ．ある演算プロセッサ上で走っているユーザ・プロセスからデータを受け取ったサーバは，そのデータに付加されたヘッダ情報をみて，適当な 100base-TX インターフェースを持つサーバにデータを送る．そしてデータは，その入出力プロセッサが持つ入出力チャネルから他のマシンへと送出される．反対に，他のマシンから送られてきたデータは，データ送信先の演算プロセッサを担当するサーバに一旦送られてから，最終的な送信先である演算プロセッサに送られる．

一方，共有メモリ型の並列計算機である Origin-2000, Onyx2 では，並列入出力は 1 つのサーバ・プロセスによって一元管理される．ただし，複数の 100base-TX インターフェース及びユーザ・プロセスの入出力を効率良く管理するために，サーバ・プログラムはマルチスレッド化し，処理の多重化を図っている．

PIO サーバの機能をまとめると，以下のようになる．

(1)　マシン間のデータ転送全ての管理
(2)　過剰なシステム情報の隠蔽
(3)　入出力データのバッファリング
(4)　複数チャネル間での負荷分散

図 2 のような環境の下で，並列チャネルの存在を意識しつつ通信負荷の分散を行うことは，ユーザ・プログラムを繁雑化し，またシステム構成の変更等に対する適用性を損ねる恐れがある．例えば，並列チャネル数の増減や，同時に実行されている他のユーザ・アプリケーションとの負荷バランス，あるいは接続先の相手計算機のチャネル数といった，データ転送負荷の制御に直接関係する要因をユーザ・プログラム側に意識させることは，ユーザから見て非常に使い難いシステムになってしまう．さらに，今回のケースでは高々 100 程度のユーザ・プロセスしか生じないが，より大規模な超並列計算機では 10^3 ～10^4 オーダのプロセスが通信に参加する．したがって，ファイル・ディスクリプタ等のリソース不足に陥ることが容易に推測される．そこで本システムでは，各マシン上に稼働している PIO サーバがマシン間のデータ転送を全て管理することで上記の問題を解消する．ユーザ・プロセスは，後述の簡便な API を用いて，PIO サーバとのみ交信する．そして，ユーザにとっては余分な情報をサーバ側で隠蔽・吸収することにより，プログラミングの負担を減らす．

さらに，サーバは適度なサイズのデータバッファを用意している．これにより，サーバにデータが全て送られた時点でユーザプロセスは送信処理から復帰することが可能である．

また，複数チャネル間の負荷分散を図ることも，PIO サーバの重要な役割である．個々のユーザ・プロセスが独自に入出力チャネルにアクセスしていたのでは，並列プロセスから生成されるデータ流は場合によっては空間的及び時間的に粗密が生じるため，各チャネルの負荷に不均衡が生じる可能性がある．したがって，出力データをうまく平滑化し，チャネル負荷を均等にする必要がある．そこで，入出力はこの PIO サーバを介して行なうことで，負荷分散を図る．つまり，一方のユーザ・プロセスから送出されたデータは，一旦そのマシンのサーバに渡され，そこで負荷が分散されるように適当な入出力チャネルを選んで他方のマシンのサーバ，そして送信先であるユーザ・プロセスに送られる．これにより，ユーザは並列入出力システムの存在を意識せずに，入出力システムを効率的に運用できるようにする．

現在の実装対象は，Pilot-3 システム上の OSF-1/MPP 及び Origin-2000, Onyx2 上の IRIX-6.5 である．マシン間のデータ転送は，TCP/IP を用いて行なわれる．したがって，チャネルあたりの最大性能は，100base-TX インターフェースの TCP/IP 性能に等しいか，もしくはそれ以下ということになる．しかし，TCP/IP という一般的な通信プロトコルを用いることにより，本システムに可搬性を持たせ，他プラットフォームへの移植が容易となる．

また，各マシン内部での処理に関しては，最適な通

信を行なうように実装されている．そのための最も重要な要素の一つが,「ゼロコピー通信機能」である．超並列計算機 Pilot-3 においては，ユーザ・レベルでのゼロコピー通信，すなわち，システムが提供するバッファを介することなしに，直接データの転送を行なうことが可能である[1]．Pilot-3 側の PIO システムでは，この機能を利用し，演算プロセッサと入出力プロセッサの間の通信において，余計なデータのバッファリングを排除し，処理を高速化している．また，Origin-2000 及び Onyx2 上においては，共有メモリ型ワークステーションの特性を利用し，デーモンとユーザ・プロセスの間で共有メモリ空間を用いることにより，同様に余計なデータのバッファリングを排除している．

4.2 並列入出力のための API

PIO システムを利用するための API (Application Program Interface) として，以下の機能を提供する．

- データ入出力操作
- システムで利用可能な並列入出力チャネルの構成要素情報の提供
- 接続先システムの情報の提供
- 手動または自動による通信負荷の空間的均等化 (データ生成プロセッサ毎の負荷分散)

本 API では，システム構成のうち，最低限の情報のみをユーザに提供，あるいは操作させ，大部分を自動化することを目的としている．明示的な情報操作をしない限り，ユーザプログラムからの入出力要求は，並列チャネルのいずれかに自動的に割り振られる．ユーザ・プログラム上で明示的な負荷分散を行いたい場合は，適当な情報を引き出すことにより，使用するチャネルを指定することも可能である．さらに，両者の方法を混在することも可能である．

PIO システムは，入出力双方の計算機における並列性を想定している．このため，一般的な TCP/IP 等の通信と異なり，ホストのアドレスだけでなく，そのホスト上の並列プロセッサまでを通信パートナーとして識別できるようになっている．さらに，基本的にどの計算機上でも同一な API が利用できるように設計されている．通常，PIO システムを利用する 2 台以上の計算機システム上では，双方の端点における PIO プログラミングが必要となる．API が統一化されていることにより，このプログラミングは非常に容易となり，また作成されたプログラムの可搬性も向上する．

なお，共有メモリ型並列計算機である Origin-2000, Onyx2 用のユーザ・アプリケーションは，一般的にマルチスレッド化されている可能性がある．したがって，Origin-2000, Onyx2 上の PIO システムはそのようなアプリケーションにも考慮して，スレッドセーフとなるように実装されている．

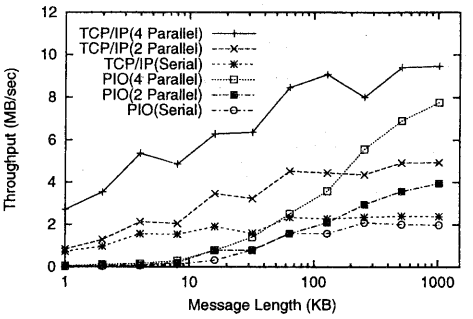

図 3　ping-pong 転送プログラムにおける PIO の性能

5. PIO システムの性能評価

5.1 ピンポン転送

PIO システムを用い，Pilot-3 と Origin-2000 間で簡単な ping-pong 転送を行なった場合の通信性能を**図 3** に示す．また，同図上にネイティブな TCP/IP 関数群をユーザアプリケーションから直接利用した場合の結果も載せる．それぞれ入出力チャネル数を 1, 2, 4 本と変えて測定を行なった．

図より，PIO システムを用いた場合も TCP/IP 関数群を直接用いた場合もどちらもチャネル数の増加に見合った性能向上が得られている．しかし，PIO システムを用いた ping-pong 転送は，TCP/IP 関数群を用いた場合より性能が低く，メッセージ長 1MB の転送の場合であっても TCP/IP 関数群を用いた場合の約 8 割の性能しか発揮していない．PIO システムではサーバを介してデータを送受信しているため，サーバで一時的にバッファリングすることによるオーバーヘッドが性能低下の主因であると考えられる．また，TCP/IP 関数群を直接用いた場合であっても，各入出力チャネル当たりのスループットは高々 2MB と，100base-TX のピーク性能の 1/6 以下であることがわかる．これは，オペレーティング・システムレベルで潜在的な性能低下となる要因があることを示唆している．したがって，100base-TX イーサネット・インターフェースの性能を極限まで引き出すためには，オペレーティング・システムに含まれる TCP/IP プロトコル自体を改良する必要がある．

5.2 一方向転送

上記の ping-pong 転送実験のように，双方がデータを送信し合うというよりも，一方（データ生成系）から他方に多量のデータが送られることのほうが本システムにおいては一般的なケースであると考えられる．この時，データ出力部分よりも演算部分がボトルネックとなるようなデータ生成系の場合，データ送信時間をできる限り短くして，早急に演算処理に復帰できる

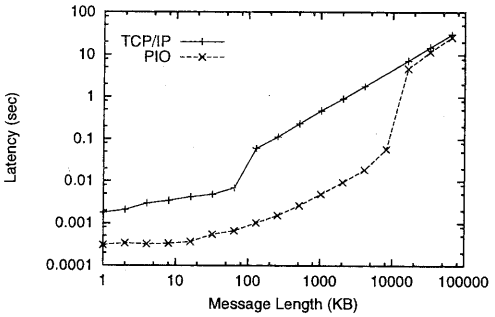

図4　一方向転送プログラムにおける PIO の性能

ようにすることが望ましい．そこで次に，データ生成系（Pilot-3）からデータ処理系（Origin-2000）に対して一方向転送を行なった場合の，送信関数を呼び出してから復帰するまでのレーテンシを求める．図4は，1チャネルだけを用いてデータ転送を行なった場合の測定結果である．

図4より，先ほどの結果とは全く異なり，どのメッセージ長においても PIO システムを用いた結果のほうが TCP/IP 関数群を用いた結果よりもレーテンシが小さい．1MB のメッセージ転送にいたっては，TCP/IP 関数群を直接用いた場合の 1/100 以下の時間で完了している．これは，さきほどの ping-pong 転送の時は性能低下の主因となった，サーバでのバッファリングが功を奏していると考えられる．つまり，PIO システムの場合は入出力プロセッサ上で走っているサーバ・プロセスにデータが送られた時点で，ユーザ・プロセスは演算処理に復帰するからである．また，ユーザ・プロセス，サーバ間のデータ転送はゼロコピー通信であるリモート DMA 転送 [1] を用いているのでピーク 300MB/sec というリンク当たりのスループットを最大限に利用できる．

一方，TCP/IP 関数群を用いた場合は，オペレーティング・システムが用意した通信バッファが溢れるとそこで通信バッファが空くまでブロックされる．実験した結果，Pilot-3 のオペレーティング・システムが各ソケット用に確保できるバッファサイズは，送信用，受信用共最大で 63550(Bytes) であった．したがって，このサイズ以上のデータを送る場合は，100base-TX ネットワークのスループットがボトルネックとなる．さらに，TCP/IP 用のパケット作成にも多少の時間を要するため，メッセージ長が短い場合でもオペレーティング・システムの関与を極力排除したリモート DMA 転送を用いた PIO システムよりもレーテンシが大きい．

なお，PIO を利用した場合でも，サーバのバッファが一杯になるとユーザ・プロセスはバッファが空くまでブロックしてしまう．それは，図中の 16MB 以上のデータを転送した場合に現れてきている．しかしながら，それでもレーテンシは “TCP/IP” とほぼ等しく，限界まで性能を引き出せていることがわかる．

表1　リモート・ファイル転送に要する実行時間
(NFS, rcp では1本，PIO では4本のチャネルを使用)

(1) Output data from an application program	
via NFS	69.39 sec (0.60 MB/sec)
via PIO	9.10 sec (4.61 MB/sec)
(2) Copy a file from Pilot-3 to Origin-2000	
using rcp	22.43 sec (1.87 MB/sec)
using PIO	10.01 sec (4.19 MB/sec)

5.3　リモート・ファイル転送

PIO システムを利用したより現実的なアプリケーションとして，リモート・ファイル転送の測定を行った．Pilot-3 側では，4プロセスによって生成もしくは読み込まれたデータを Origin-2000 上で稼働しているファイル・サーバに転送する．Origin-2000 側のファイル・サーバは，Pilot-3 から送られてきた4つのデータ流を単一流にまとめ，そしてローカルのファイルシステムに一つのファイルとして書き込む処理を行なう．

今回は2種類のファイル転送実験を行なう．まず最初に，ファイル・サーバに送るデータを Pilot-3 側のプログラム内で直接生成する場合である．つまり，データを読み込むためのディスクアクセスは一切発生しない．また，PIO を用いたファイル転送性能と比較するために，NFS を介したファイル転送の性能も測定する．

結果は表1 (1) のようになった．これによると，PIO を用いたデータ転送は，NFS を利用した場合の約13%の実行時間で済んでいる．これは，PIO が4本の 100Base-TX イーサネットを効率良く利用できているのに対し，NFS はそのうちの1本しか用いていないことが主因である．さらに，NFS の場合はリモートのファイルシステムをあたかもローカルのファイルシステムであるかのように見せるために，様々なソフトウェア・オーバヘッドが生じていることが本測定結果に影響しているものと考えられる．

そこで次の実験として，NFS よりもソフトウェア・オーバヘッドの小さい UNIX の “rcp” コマンドと PIO を利用したファイルコピー・プログラムの性能を比較する．対等な比較評価を行なうために，PIO を用いたプログラムもファイルからデータを読み込むようにする．PIO を利用したプログラムでは，4つのプロセスが同時に一つのファイルからデータを読み込み，そしてある程度のサイズに分割して4つのチャネルからデータを送るようにする．一方，“rcp” コマンドの場合は1プロセスがファイルからのデータ読み込み，チャネルへのデータ出力を行なう．また，使用するチャネル数も一本のみである．

結果は 表1 (2) のようになり，PIO を利用したファイルコピー・プログラムは "rcp" コマンドの約45%の転送時間を要した．今回の実験では，4 倍のチャネル数を利用しているにもかかわらず，それほど効果がでていないという結果になった．これは Piot3 (Origin-2000) におけるファイル読み込み (書き込み) のところで逐次化されてしまうところがボトルネックとなり，それほど差が生じなかったのではないかと推測される．

6. 今後の展望

現在のところ，超並列計算機 CP-PACS のプロトタイプ機 Pilot-3 と Origin-2000 もしくは Onyx2 との間で並列入出力システムのプロトタイプが動作している．各マシンは，それぞれ 4 本の 100base-TX インターフェースを備えている． 将来的には，ネットワークを増強し，そしてより大規模システムである CP-PACS との結合が考えられる．

そのために，プロトタイプにはない以下の要件を満たすよう機能拡張を施していく予定である．

- 並列入出力システム PIO の実行時サポートシステムに，データ流の動的負荷分散機能を実装し，より柔軟で効率的な並列入出力環境を提供する．また，PIO のプラットフォームの範囲を広げ (例えば，Windows 環境を持つパーソナルコンピュータ)，より多くのアプリケーションから PIO システムを簡単に利用できるように整備する．これらにより，よりインタラクティブ性の強いアプリケーションにおいて，超並列計算機を一層使い易くすることが可能となる．
- PIO システムに基づく各種アプリケーション・プログラムを構築する．現在，PIO システムと並行して，PIO を利用した並列可視化システムの実装が進んでいる [4]．しかし，それ以外に，フロントエンド並列ワークステーション上に，並列データ流を高速でディスクに格納するファイルシステムの構築や，超並列計算機から生成されたデータに対する後処理を機能分散的に同時並行処理するプログラムの開発等が考えられる．また，これらの運用を通じ，PIO システムの問題点の抽出や性能チューニングも併わせて行なう．
- CP-PACS 上の入出力機構をチューニングする．現在の CP-PACS 入出力プロセッサでは，デーモン上のデータを最終的なネットワーク・デバイスに出力する段階で，オペレーティング・システムの性質に依存する性能低下要因が存在している．この点を，PIO システムの実装に適するように最適化し，単体チャネル性能をより向上させる．さらに，並列チャネル数を増加させ，総合的なスループットをより向上させていく．

7. おわりに

本稿では，現在我々が開発を進めている超並列計算機向けの並列入出力システムについて報告した．

ネットワークを並列化することにより，両端のマシンに複数ある入出力プロセッサを効率良く利用でき，その結果，複数データ流を一旦逐次化することなく，並列のまま相手側マシンに送ることが可能となった．また，近年高性能化が進んでいるコモディティ化されたネットワークを用いることで，安価ではあるが高速な環境を実現できる．我々は，このようなハードウェア環境下で，並列ネットワークを効率良く利用するための並列入出力システムを開発した．ユーザからは，簡便なインターフェースを利用することで，並列ネットワークの恩恵を享受できる．本システムの性能を評価した結果，入出力を並列化することにより，容易に線形な性能向上が見込めることが確認された．

本システムには，さらなる性能改善が望まれる．これには，OS レベルでのチューニング及びより高位レベルでのチューニングである複数チャネルに対する動的負荷分散などが含まれる．また，動的負荷分散のサポート，アプリケーションの開発なども今後の課題として挙げられる．

謝　辞

本研究を行なうにあたり，御意見・御協力頂いた筑波大学計算物理学研究センター未来開拓プロジェクト関係各位に感謝致します．なお，本研究は日本学術振興会未来開拓学術研究推進事業「計算科学」のプロジェクトの一つとして行なわれている．

参考文献

1) 岩崎 洋一, 中澤 喜三郎 ほか : 計算物理学と超並列計算機 – CP-PACS 計画 –, 情報処理, Vol.37, No.1, pp.10-42 (1996).
2) 北井 克佳, 吉澤 聡, マシエル フレデリコ, 鍵政 豊彦, 稲上 泰弘 : TCP/IP 通信を用いた並列ネットワーキング方式の検討, 情報処理学会論文誌, Vol.39, No.11, pp.3044-3053 (1998).
3) Berdahl, L. : *Parallel Transport Protocol Proposal*, Lawrence Livermore National Laboratory, (1995).
4) 松原 正純, 沼 寿隆, 朴 泰祐, 中本 泰史, 梅村 雅之, 白川 友紀, 宇川 彰 : 超並列計算機のための Commodity Network に基づく並列入出力・可視化システム, 電子情報通信学会技術研究報告, CPSY98-161, pp.81-88 (1999).
5) Stevens, W. R 著, 篠田 陽一 訳 : *UNIX Network Programming*, トッパン (1992).
6) S. Kleiman, D. Shah, B. Smaalders : *Programming with THREADS*, SunSoft Press/Prentice Hall PTR (1996).