Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

メニーコア混在型並列計算機における MPI 通信基盤の提案

吉永一美[†1,†5] 六車英峰[†1,†5] 辻田祐一[†1,†5]
堀敦史[†2,†5] 並木美太郎[†3,†5] 佐藤未来子[†3,†5]
下沢拓[†4] 澤田武男[†4] 石川裕[†4,†2]

本稿では，エクサスケールを実現する次世代の並列計算機環境として，汎用マルチコアプロセッサとメニーコアプロセッサが混在するメニーコア混在型並列計算機を想定し，その上で動作する MPI 基盤システムソフトウェアの提案を行う．メニーコアプロセッサは高い並列演算性能を持つ一方，コアあたりの演算性能，メモリ容量，キャッシュ容量はマルチコアプロセッサと比較して低いという特徴があるため，現在の一般的な MPI 処理基盤をそのまま利用するだけでは，効率的な並列分散処理を行うことができない．そこで本稿では，マルチコアプロセッサに MPI 処理を委託する機構を用いた MPI 基盤システムソフトウェアを提案し，その設計と検討課題について述べる．

MPI Communication Infrastructure on a Heterogeneous Platform with Multi-core and Many-core CPUs

Kazumi Yoshinaga,[†1,†5] Hidetaka Muguruma,[†1,†5]
Yuichi Tsujita,[†1,†5] Atsushi Hori,[†2,†5]
Mitaro Namiki,[†3,†5] Mikiko Sato,[†3,†5]
Taku Shimosawa,[†4] Takeo Sawada[†4]
and Yutaka Ishikawa [†4,†2]

We propose a scalable MPI communication infrastructure towards realization of a hybrid parallel computer with many-core and multi-core CPUs in the forthcoming Exascale era. In general, many-core CPUs have totally high computing power, however each CPU core lacks for computing power and main memory and cache resources compared with multi-core CPUs. Therefore enhanced approach based on existing MPI implementations is not favorable for Exascale systems. In order to have good scalability, we propose a delegation mechanism so that many-core CPUs can send delegation requests to multi-core CPUs in the proposed MPI implementations. In this paper, we explain our preliminary design and discuss some issues to be realized in our future investigation.

1. はじめに

現在スーパーコンピュータの性能はペタフロップスを達成し[1)]，2018 年にはエクサフロップスに達することが予想されている[2)]．エクサフロップスの実現に向け，メニーコアプロセッサが注目されており，今後の HPC の一つの主流になると予想される．メニーコアプロセッサの利用形態としては，Intel 社の Single-chip Cloud Computer（SCC）[3)] などのメニーコアプロセッサ単独で動作するホモジニアスな形態と，汎用マルチコアプロセッサをホスト CPU として別に持ち，Intel 社の Many Integrated Core（MIC）[4)] などのメニーコアプロセッサを搭載したアクセラレータボードを組み合わせた，ヘテロジニアスな形態が存在する．

我々は後者のメニーコアプロセッサをアクセラレータとして利用するメニーコア混在型並列計算機に着目し，エクサスケールを実現するための MPI 基盤システムソフトウェアの研究を行っている．メニーコアプロセッサは従来のマルチコアプロセッサと比較するとコア数が多く，並列演算に特化した性能を有している一方で，コアあたりのメモリやコアが有するキャッシュメモリの容量が少なく，コア単体の性能も低いという特徴がある．そのため，現在の MPI 実装を単純に流用しただけでは，バッファを十分に確保できず通信や I/O などの処理の高性能化が困難であるだけでなく，メモリ・キャッシュ領域の逼迫によりアプリケーションプログラムの性能低下を招いてしまう．

そこで本研究ではこの問題を解決するために，通信や I/O などの MPI 処理をマルチコアプロセッサに委託し，メニーコアプロセッサを並列演算処理に専念させる機構を提案する．

†1 近畿大学
Kinki University
†2 理化学研究所 計算科学研究機構
RIKEN AICS
†3 東京農工大学
Tokyo University of Agriculture and Technology
†4 東京大学
The University of Tokyo
†5 独立行政法人科学技術振興機構 CREST
JST CREST

Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

また，ノード内の通信処理・I/O 処理をマルチコアプロセッサに集約し，高速化する手法を取り入れることにより，MPI 処理全体の高速化も目指す．

本稿では，2 章において提案する MPI 基盤システムソフトウェアの設計について述べ，3 章では，模擬環境上での試験実装を用いた実験について述べる．4 章では検討課題を述べ，5 章において関連研究について述べる．そして 6 章で本稿のまとめを行う．

2. MPI 基盤システムソフトウェアの設計

2.1 対象とする環境

本研究が対象とする環境は，ノード内にマルチコアプロセッサとメニーコアプロセッサが混在した，メニーコア混在型並列計算機アーキテクチャである．**図 1** にシステムの概略図を示す．この環境の利用形態として，以下の 3 種を想定している．

(1) メニーコアプロセッサのみ利用
数値計算など高速な並列演算が必要なアプリケーションは，並列演算性能の高いメニーコアプロセッサで実行する．

(2) マルチコアプロセッサのみ利用
ゲノム解析等，大量のデータを扱うアプリケーションでは，マルチコアプロセッサの高い I/O 性能と潤沢なメモリを用いることで効率的に実行する．

(3) メニーコア・マルチコアプロセッサ双方の利用
計算処理と並列ファイルシステム等への I/O 処理が混在するアプリケーションでは，両方の処理を高速に行うためにメニーコア・マルチコア双方を利用する．

本稿で提案する基盤システムは，上記 3 種のうち（3）で述べた利用形態を支援するものである．

2.1.1 ハードウェア環境

ノード内には，ホストとなるマルチコアプロセッサと，メニーコアプロセッサを搭載したアクセラレータボードが存在する．アクセラレータボードは PCI Express バスを経由し，一枚あるいは複数枚がマルチコアプロセッサと接続される．

また，ノードには InfiniBand などの高速ネットワークインターフェースが存在し，各ノード間を高速ネットワークを介して接続することで，高性能な並列計算機クラスタとしての動作を実現する．

2.1.2 ソフトウェア環境

ソフトウェア環境としては，汎用マルチコアプロセッサ上では Linux 等の一般的な汎用

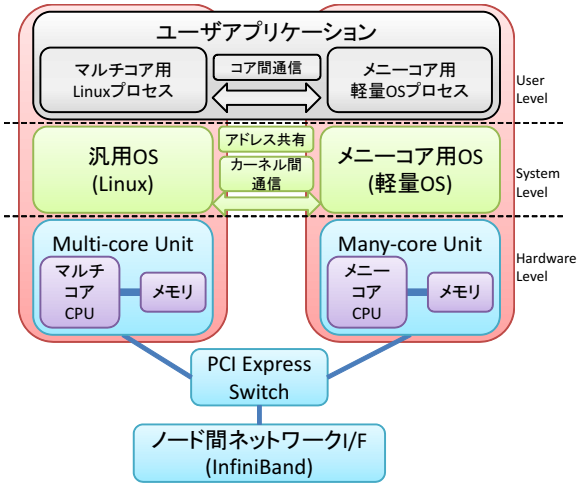

図 1 対象環境の概略図

OS が，メニーコアプロセッサ上ではメニーコアユニット上の CPU とメモリを管理するための軽量 OS が動作していることを想定する[5]．

両 OS カーネルには，通信手段としてカーネル間通信機構が備わっている．また，PCI Express バスを通じて，マルチコアプロセッサとメニーコアプロセッサのメモリ領域が互いにマッピング可能であり，バスを介した共有メモリ空間が作成できる．これらを用いることで，マルチコアプロセッサとメニーコアプロセッサ間において，コア間通信が実装されていると想定する．

2.2 設 計 方 針

本研究では，マルチコアプロセッサ上に MPI 代行処理サーバプロセスを起動し，メニーコアプロセッサで発行された MPI 命令を代行して処理する機構を提案する．この機構を用いて，メモリやキャッシュが潤沢であるマルチコアプロセッサに通信や I/O などの MPI 処理を，コア数が多く高い並列演算性能を持つメニーコアプロセッサに並列度の高い演算処理を割り当てることにより，両者の特長を活かした高効率な MPI システムソフトウェアを実現する．

また，通信と I/O 処理が MPI 代行処理サーバに集約されるため，集団型通信・集団型 I/O の効率化手法を MPI 代行処理サーバに実装し適用することで，更なる性能の向上を図

Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

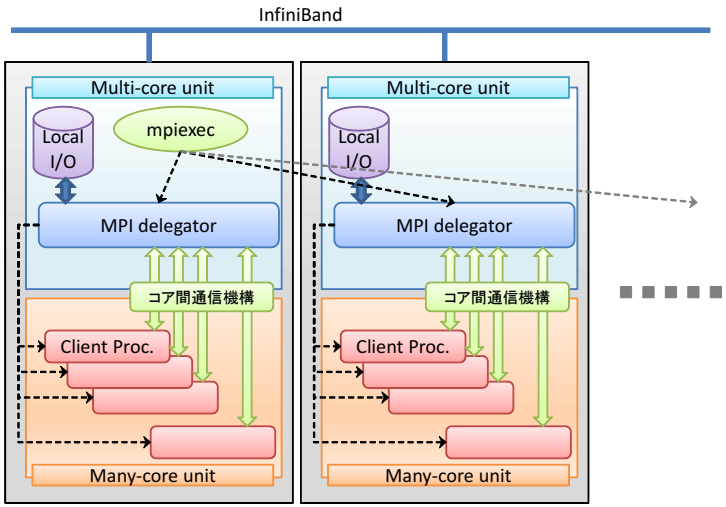

図 2　提案する MPI 基盤システムの概略図

ることが可能である．

提案する MPI 基盤システムソフトウェアの概略図を図 2 に示す．

実際の MPI プログラムの実行の流れは以下のとおりである．

(1) MPI プログラムが実行されるとまず，利用するノードのマルチコアプロセッサ上に，MPI 代行処理サーバプロセスを起動する．

(2) MPI 代行処理サーバプロセスは，並列演算や実際に MPI 処理を行うクライアントプロセスを，メニーコアプロセッサ上に起動する．

(3) クライアントプロセス内で MPI 命令が発行されると，コア間通信機構を用いて，MPI 処理依頼を MPI 代行処理サーバプロセスへと通知する．処理依頼に必要な情報として，MPI 命令の ID，その MPI 命令に必要な引数，データ転送が生じる場合はそのデータのアドレスなどを同時に送信する．

(4) MPI 処理依頼を受け取った MPI 代行処理サーバプロセスは，受け取った命令 ID から実行する MPI 命令を決定し，受け取った引数を用いて実際に MPI 処理を実行する．ここで，データ通信や I/O 処理が生じる MPI 命令の場合は，クライアントプロセスのメモリ上のデータ領域をマッピングして直接利用する．実行結果はコア間通信機構を用いてクライアントへと送信される．

3. 模擬環境を用いた実装と予備評価

3.1 模擬環境構築のねらい

現在，実装対象であるハードウェアは存在しないため，システムのプロトタイプ開発の方向性を確認する目的で，マルチコア-メニーコア間の委託システムを模擬する環境を作成した．この環境を用いて，想定されるハードウェア構成で起こりうる問題の洗い出しを行った．今回は，汎用 Linux クラスタ内で，マルチコア上で実行される MPI 代行処理サーバプロセスを模擬した MPI プロセス（以降、擬似サーバプロセス）と，メニーコアプロセッサ側で実行されるクライアントプロセスを模擬したプロセス（以降、擬似クライアントプロセス）を実装し，処理時間の比較を行った．

3.2 実 装 詳 細

擬似サーバプロセス

擬似サーバプロセスは，提案手法における MPI 代行処理サーバプロセスに相当するもので，MPI プロセスとして実装した．このプロセスは実行時に利用するノード上で起動され，後述する擬似クライアントプロセスからの MPI 代行処理依頼を受け取り，実際の MPI 処理を行う．

擬似クライアントプロセス

擬似クライアントプロセスは，提案手法におけるメニーコアプロセッサ上のクライアントプロセスに相当する．擬似サーバプロセスによって起動され，MPI 代行処理依頼を発行する．

コア間通信の模擬

メニーコア混在型システム内のコア間通信を模擬するために，通信データの送受信に用いる共有メモリ領域と，送受信の通知に用いるキャラクタデバイスを用意し，擬似的なコア間通信機構を実装した．MPI 代行処理依頼にはこのコア間通信機構を用いる．

送受信バッファ領域

実際のシステムでは，メニーコアプロセッサ側メモリが PCI Express バスを通じてマルチコアプロセッサからマッピング可能となる予定である．そのため，クライアントプロセスがメニーコアプロセッサ側に用意した送受信バッファを，MPI 代行処理サーバがマッピングしアクセスすることで，マルチコアプロセッサとメニーコアプロセッサ間で送受信バッファのコピーを伴わずに MPI 通信の代行処理を行うことができると考えている．

Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

表 1　実験環境の仕様

PC ノード	CPU	Intel Pentium 4 660 (3.6GHz FSB 800MHz)
	Memory	DDR2-533 SDRAM 1GB （512MB × 2，Dual Channel）
	NIC	Broadcom BCM 5721
	OS	Cent OS 5.5
	MPI	MPICH2-1.3.2p1
ネットワークスイッチ		3Com SuperStack 3 Switch 4900

模擬環境の実装においては，擬似サーバプロセスと擬似クライアントプロセスの双方からアクセス可能な共有メモリ領域を作成し，擬似クライアントプロセスは送受信バッファをこの領域上に確保するようにした．この実装により，MPI 代行処理時に送受信バッファのメモリコピーを伴わない状況を模擬している．

3.3 評 価 実 験

実験には，4 台のノードが 1000BASE-T のネットワークで接続されている Linux クラスタ環境を用いた．**表 1** に各ノードとネットワークスイッチの仕様を示す．

模擬環境を用いた MPI 代行処理サーバ利用時（Delegator モード）と，通常の MPI プロセスによる直接通信時（Normal モード）の性能を比較する．

3.3.1 一対一通信

まず，クラスタ内の 2 ノード間で PingPong 通信を行い，その実行時間を比較した．実行結果を**図 3** に示す．横軸は送信データサイズ，縦軸はスループットである．

MPI 処理の依頼時にキャラクタデバイスを用いた通知処理と，共有メモリを通じたプロセス間通信が生じるため，Delegator モードのスループットは Normal モードと比較して低下している．転送データサイズが 8KB 以下の場合は 20〜30%の性能低下が見られるが，データサイズが増加するとデータ転送時間が増加し全体の処理時間に占める割合が大きくなるため，次第に性能差は縮まり，1MB 以上では 1%未満の性能低下となっている．

3.3.2 Alltoall 通信

次に，4 ノードを用いた Alltoall 通信の実行時間を比較した．その実験結果を**図 4** に示す．なお，1 つのプロセスからの全送信データ数をデータ転送時間で割ったものをスループットとしている．

全体の傾向として，一対一通信の場合と同様にデータサイズが増加するにつれてオーバーヘッドの影響は少なくなり，4KB 以下のデータサイズでは 20〜30%の性能低下が見られるが，64KB 以上では 1〜5%となっている．両モード共に 64KB〜128KB 付近で性能低下が

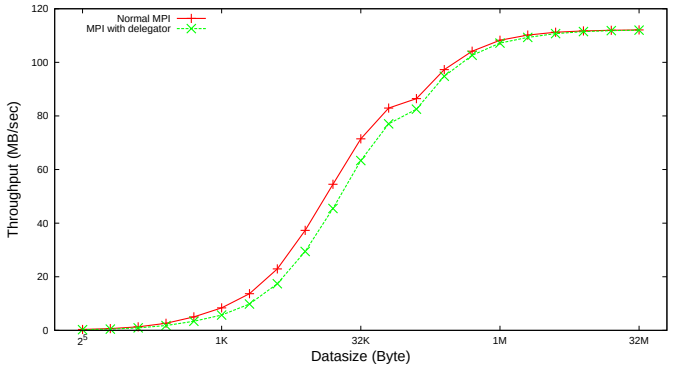

図 3　一対一通信の結果

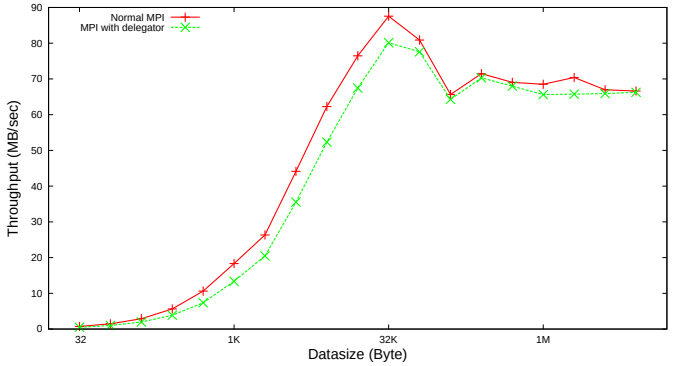

図 4　Alltoall 通信の結果

見られるが，この要因は通信プロトコルの切り替えではないかと考えられる．

3.4 考　　察

MPI 代行処理サーバを利用することで，処理依頼分のオーバーヘッドが生じるが，通信データサイズが大きい場合は全体の実行時間の多くをノード間での通信時間が占めるため，性能への影響は少なくなる．しかし，実際のエクサスケールのスーパーコンピュータを考慮した場合，今回の模擬実験環境と比較しネットワーク速度が大きく向上するため，オーバーヘッドの影響が大きくなる可能性がある．

Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

例えば，Intel MIC アーキテクチャ製品の Knights Corner[6] を搭載する予定のメニーコア混在型並列計算機 Stampede[7] では，InfiniBand FDR 4X を採用した最大 56Gbps のネットワークを用いる．模擬実験環境のネットワークは 1Gbps であったため，単純計算で 56 倍の性能となる．一方，マルチコアプロセッサと Knights Corner 間は PCI Express 3.0 スロットで接続され，x16 であればマルチコアプロセッサ側でマッピングしたメニーコアボード上のメモリの共有領域との最大帯域幅は 16GB/s（128Gbps）である．模擬実験環境のプロセッサ-メモリ間の帯域幅は 6.4GB/s（51.2Gbps）であり，比較すると 2.5 倍の性能である．このことから，ノード間の通信時間の短縮率に対し，ノード内での共有メモリ領域へのアクセス時間の短縮率が少ないことが予測され，全体の処理時間に対する処理依頼のオーバーヘッドの割合がより高まる可能性が考えられる．

今後，メモリアクセス時間だけでなく，カーネル間通信の影響も含めたオーバーヘッドについて，短いデータ長のケースも含め検討を進めると共に，積極的な非同期通信の活用を促すなどオーバーヘッドの隠蔽に関しても検討が必要である．

4. 検 討 課 題

4.1 同一プロセッサ内通信の高速化

ノード間，ノード内に関わらず，全ての通信が MPI 代行処理サーバを介して行われる実装では，マルチコアプロセッサ側で全てのデータ転送処理が行われる．MPI 代行処理サーバが行う通信処理では，メニーコアプロセッサのメモリを PCI Express バスを介してマルチコアプロセッサ上でマッピングし，その領域を用いて送受信処理を行う．そのため，送信元プロセスと送信先プロセスが同一メニーコアプロセッサ内であっても，PCI Express バスを介したデータ移動となる．

このデータ移動処理をメニーコアプロセッサ側で行うことで，PCI Express を介するよりも高速な内部でのメモリコピーとして通信処理を実装可能となり，高速化できる．ただしそのためには，メニーコアプロセッサのプロセスを管理する軽量 OS において，他プロセスからのメモリ領域へのアクセスを実現する機構が必要である．

4.2 非同期通信・非同期 I/O

非同期通信及び非同期 I/O は，MPI 代行処理サーバにリクエストキューを実装することで実現可能である．クライアントプロセスからの非同期処理の依頼をリクエストキューに追加していき，MPI 代行処理サーバはリクエストを取り出して処理を行う．現在策定中の MPI-3[8] では，集団型非同期通信が制定される予定であり，仮にマルチコア側のネイティブな MPI 実装がこの機能をサポートしていなくても，MPI 代行処理サーバを用いた擬似的な集団型非同期通信機能がメニーコア上に提供できると考えている．

4.3 効率的な集団型通信と I/O

エクサスケールを視野に入れると，プロセス数の増大から単純な集団型通信では通信量が莫大なものとなることが予想される．本研究が提案する機構は 2.2 節で述べたように，MPI 代行処理サーバプロセスにノード間の通信やディスク I/O が集約されるため，通信や I/O を効率化する処理を実装することで性能の向上を図ることができる．例えば，ブロードキャストなどの同一データを多数のクライアントプロセスへと転送する通信は，MPI 代行処理サーバプロセスのバッファへと送信した後に，そのプロセスの管理するクライアントプロセスのバッファにコピーすることで，ネットワーク通信量を減らすことが可能である．

同様に I/O を利用するクライアントプロセスも増大することが考えられるが，その処理は MPI 代行処理サーバに委託され集約されるため，連続した小規模データをまとめることによる I/O アクセス回数の削減など，I/O 性能を向上させる手法が容易に適用できる．また，実際に I/O を行うプロセスは MPI 代行処理サーバのみであり，I/O 要求回数の削減による I/O 競合の低減も期待できる．

4.4 コミュニケータ情報の効率的な管理

エクサスケールマシンにおいては，実際の計算プロセスであるクライアントプロセスの数が膨大になる．メニーコアプロセッサのメモリ領域が小さいことを考えると，既存の MPI 実装の延長でのコミュニケータ情報の管理は実現不可能なため，最低限必要な情報として通信相手のランクなど，自分の属するコミュニケータに関する情報の一部のみを保管するようにする．一方，マルチコアプロセッサ側では実際に MPI 処理を行うため，通信相手を決定するために必要となるアドレス等の情報を含め，全てのコミュニケータ情報を持たなければならず，データ量の増加，情報の検索時間の増大などの問題が予測される．

そのため，MPI のコミュニケータ情報などの管理方法に関して，今後スケーラビリティを考慮した設計をしていく必要がある．

5. 関 連 研 究

MPI の処理の一部を委託する研究として，Arifa Nisar らによる IODC（I/O Delegate Cache System）[9] がある．この研究では，MPI アプリケーションの実行時に，計算を担当する Application Node と呼ばれる MPI プロセス群と，MPI-IO 処理を担当する IOD（I/O Delegate）Node と呼ばれる MPI プロセス群を準備し，Application Node が発行した I/O

Vol.2011-ARC-197 No.5
Vol.2011-HPC-132 No.5
2011/11/28

リクエストを IOD Node に転送し実際の I/O を行う．I/O の発行を IOD Node に集約することで，ファイルシステムへの同時アクセス数を減少させると共に，キャッシュ機構の実装や，複数の小さな I/O リクエストをまとめることにより，並列 I/O の性能を向上している．MPI 処理の一部を委託するという点は本研究と共通しているが，IODS は MPI-IO 処理のみを委託するものであり，ホモジニアスな環境内において MPI プロセスの役割を分担し，MPI の inter-communicator を利用してやり取りしている点が異なる．

集団型通信の高速化の研究として，Teng Ma らの研究[10] がある．この研究は NUMA システム上において集団型通信の高速化を図るものである．一般的な MPI 実装において，同一ノード内の MPI プロセス間の通信は共有メモリを介して行うためメモリコピーが 2 回必要であるが，KNEM[11] というカーネルモジュールを用いてプロセスのメモリ領域に直接アクセスし，メモリコピー回数を 1 回にすることで速度を向上している．また，ブロードキャスト通信の高速化手法として，NUMA ノードの中の一つのプロセスを仲介プロセスとして扱う手法が提案されている．送信元プロセスから，別の NUMA ノード内の複数のプロセスへとブロードキャストを行う際に，代表である仲介プロセスへのみデータをパイプライン処理で転送し，その他のプロセスは仲介プロセスが受け取ったデータをメモリ内でコピーすることで，通信量を減らして高速化している．この文献ではプロセッサ性能が一様な NUMA 環境についてのみ言及されているが，我々はヘテロジニアスなメニーコア混在型計算機環境を対象にしており，ノード間だけでなくノード内での性能差も考慮した設計を行う点が大きく異なる．

6. おわりに

本稿では，エクサスケールを実現する次世代の並列計算環境として，メニーコア混在型並列計算機を想定し，ノード間通信や I/O 処理を含む MPI アプリケーションを効率的に実行するために必要な，MPI 基盤システムソフトウェアを提案した．MPI 代行処理サーバを用いることで，通信や I/O 処理をマルチコアプロセッサに委託しメニーコアプロセッサを並列演算処理に専念させると共に，通信と I/O 処理を集約することによる全体の処理の高速化についても期待できる．今後は 4 章で述べた課題や，MPI 処理依頼のオーバーヘッドの影響などについて更に検討し，実装と評価を進める．

謝辞 本研究は，科学技術振興機構（JST）戦略的創造研究推進事業（CREST）における研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」研究課題「メニーコア混在型並列計算機用基盤ソフトウェア」によるものである．

参考文献

1) TOP500.org: June 2011 — TOP500 Supercomputing Sites, TOP500.Org (online), available from ⟨http://www.top500.org/lists/2011/06⟩ (accessed 2011-10-26).
2) Intel: Intel Equipped to Lead Industry to Era of Exascale Computing, Intel Corp. (online), available from ⟨http://newsroom.intel.com/community/intel_newsroom/blog/2011/06/20/intel-equipped-to-lead-industry-to-era-of-exascale-computing⟩ (accessed 2011-10-26).
3) Howard, J., Dighe, S., Hoskote, Y., Vangal, S., Finan, D., Ruhl, G., Jenkins, D., Wilson, H., Borkar, N., Schrom, G. and et al.: A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS, *2010 IEEE International SolidState Circuits Conference ISSCC*, pp.108–109 (2010).
4) Intel: Intel®Many Integrated Core Architecture, Intel Corp. (online), available from ⟨http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html⟩ (accessed 2011-10-26).
5) 佐藤未来子，辻田祐一，堀　敦史，並木美太郎：マルチコア・メニーコア混在型並列計算機向け OS の構想，情報処理学会研究報告 [システムソフトウェアとオペレーティング・システム]， Vol.2011-OS-118, No.6, pp.1–6 (2011).
6) Intel: Intel Unveils New Product Plans for High-Performance Computing, Intel Corp. (online), available from ⟨http://www.intel.com/pressroom/archive/releases/2010/20100531comp.htm⟩ (accessed 2011-10-26).
7) TACC: “Stampede’s” Comprehensive Capabilities to Bolster U.S. Open Science Computational Resources, Texas Advanced Computing Center (online), available from ⟨http://www.tacc.utexas.edu/news/press-releases/2011/stampede⟩ (accessed 2011-10-26).
8) Graham, R.: Preview of the MPI 3 Standard, *SC09, BOF* (2009).
9) Nisar, A., keng Liao, W. and Choudhary, A.: Scaling parallel I/O performance through I/O delegate and caching system, *SC Conference*, Vol.0, pp.1–12 (online), DOI:http://doi.ieeecomputersociety.org/10.1145/1413370.1413380 (2008).
10) Ma, T., Bosilca, G., Bouteiller, A., Goglin, B., Squyres, J.M. and Dongarra, J.J.: Kernel Assisted Collective Intra-node MPI Communication Among Multi-core and Many-core CPUs, *Proceedings of the 40th International Conference on Parallel Processing (ICPP-2011)*, Taipei, Taiwan (2011).
11) KNEM: High-Performance Intra-Node MPI Communication, Inria Runtime Team-Project (online), available from ⟨http://runtime.bordeaux.inria.fr/knem/⟩ (accessed 2011-10-26).