Vol.2010-ARC-190 No.21
2010/8/4

Cell Broadband Engine向けオフロード機構の提案

鎌田俊昭[†1]　西川由理[†1]
吉見真聡[†2]　天野英晴[†1]

本研究報告では，ヘテロジニアスなマルチコアプロセッサを用いたクラスタにおいて，その計算資源を効果的に利用するためのオフロード機構の提案を行う．

Cell Broadband Engine (Cell/B.E.) に代表されるアクセラレータは近年ますます注目を集めつつあるが，プログラミングの複雑さなどから，計算資源を利用するためには高度な技術が求められる．ユーザのプログラマビリティの向上を目指すため，本研究報告では，汎用プロセッサから Cell/B.E. の計算資源を利用するためのオフロード機構を提案する．

本研究報告で述べる予備評価によって，ノード間通信に要する時間などを基礎的なデータをまとめる．また，並列性の高いアプリケーションによるベンチマークの例を用いて，通常の Cell/B.E. を用いたプログラミングと同様のスケールメリットが得られることを示す．

A Proposal of Offload Structure for Cell Broadband Engine

Toshiaki Kamata,[†1] Yuri Nishikawa ,[†1]
Masato Yoshimi [†2] and Hideharu Amano [†1]

In this report, we propose an efficient offload mechanism for parallel and distributed prosessing environment for a cluster with Cell Broadband Engines(Cell/B.E.). Accelerators have become more prevalent in recent years, however, due to complexity of It's programming, it is still hard to implement program codes.

In order to reduce programming cost, we propose an offload mechanism to use computational resources of Cell/B.E. efficiently from general-porpose processors in the cluster environment. Using this mechanism, programmers can easily take advantage of the Cell processor's computational resources.

As preliminary evaluations, we show the transfer rate between node and host machines, and benchmark performance of parallel application.

1. 序　　論

科学技術計算の分野では，処理の高速化，高精度化への要求は強く，それに対応するために多くの計算資源が必要とされてきた．近年のプロセッサのマルチコア化と同様に，多数の並列に動作する演算コアを持つアクセラレータが注目されるようになってきている．例として，汎用の CPU に統合された算術演算用の計算コアを用いて処理能力を高める技術などがあげられる．PlayStation3 に搭載された Cell Broadband Engine (Cell/B.E.) もその一例といえる．

Cell Broadband Engine（Cell/B.E.）はソニー・コンピュータエンタテインメント (SCE), Sony, 東芝, IBM によって共同開発された Cell Broadband Engine Architecture（CBEA）と呼ばれるアーキテクチャ規格に準拠したマルチコアプロセッサの一種である．Cell/B.E. やその他の CBEA に準拠したプロセッサは, Intel の Core i7 や AMD の Phenom のような対称型のマルチコアプロセッサとは異なり, PowerPC Processor Element（PPE）と呼ばれる PowerPC 命令互換の汎用的な制御用のコアと 8 個の Synergistic Processor Element（SPE）と呼ばれる独自アーキテクチャの演算用のコアで構成される非対称型の構成をとっている．複数 SPE による一般的な並列処理に加えて各 SPE において 4-way の SIMD 演算が行われ，高い単精度浮動小数点演算能力を引き出すことができ[1)]，その強力な演算性能により，数値解析やマルチメディア処理などへの応用が試みられている．スーパーコンピュータの性能指標である TOP500 ランキングにおいては，米ロスアラモス国立研究所の Roadrunner がクラスタの一部に Cell/B.E. を使用しており，2008 年 6 月から 2009 年 11 月に掛けて 1 位を獲得したことが注目を集めた[2)]．Roadrunner は 2010 年 7 月現在においても，世界第 3 位に位置している．

Cell/B.E. の高い性能が注目される一方で，限界まで性能を引き出すためには独自かつ高度なプログラミング技術が要求される．これにより，Cell/B.E. の普及が阻害されているという現状がある．

本研究の目的は，Cell/B.E. の計算資源を利用するためのオフロード機構を実装し，ユーザのプログラマビリティの向上及び，柔軟性を得ることを目指すとともに，Cell/B.E. のみ

†1 慶應義塾大学 大学院理工学研究科
Graduate School of Information and Computer Science, Keio University
†2 同志社大学 理工学部 インテリジェント情報工学科
Department of Intelligent Information Engineering and Science, Doshisha University

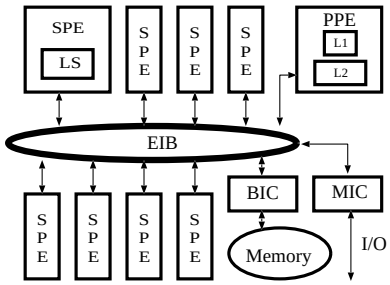

図 1 Cell/B.E. の構成
Fig. 1 Structure of Cell/B.E.

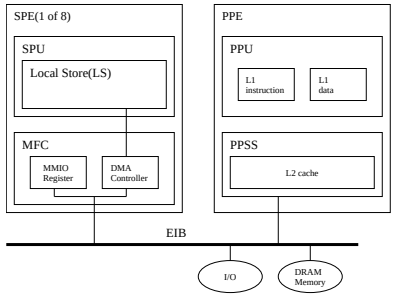

図 2 PPE 及び SPE の構成
Fig. 2 Structure of PPE and SPE

で記述したプログラムとの性能評価を行い，その効果を検証することである．

以下に本研究報告の構成を示す．まず，2 章で対象とするアーキテクチャである Cell/B.E. について述べ，3 章で，Cell/B.E. の計算資源を利用するための関連研究について述べる，4 章で実装について述べ，6 章で評価を示し，7 章で本研究報告をまとめる．

2. Cell Broadband Engine

Cell/B.E. は，IBM Power Architecture ベースの汎用コアである PPE (PowerPC Processor Element) 1 基，及びマルチメディア演算に特化した SPE (Synergistic Processor Element) 8 基からなるヘテロジニアス (Heterogeneous:非対称) マルチコアプロセッサである．各プロセッサは Element Interconnect Bus (EIB) によって接続される．本章では，Cell/B.E. のアーキテクチャ及び性能を引き出すためのプログラミングの手法について述べる．

2.1 構　　成

Cell/B.E. の構成を図 1 に，各プロセッサの詳細を図 2 に示す． Cell/B.E. は 1 基で 200GFLOPS を超える高い単精度浮動小数点演算能力をもつ．しかしながら，ヘテロジニアスマルチコアプロセッサであるという特徴から，従来とは異なるプログラミング手法が必要となる．

PPE は，メインメモリや外部デバイス等の制御をおこなう汎用プロセッサである．PPE は命令 · データ用にそれぞれ 32KB の一次キャッシュ，512KB の二次キャッシュを備えている．また PPU は PowerPC アーキテクチャをベースとした命令セットを持ち，128 ビット SIMD ユニットである VMX を搭載している．但し，PPE における VMX 命令は倍精度浮動小数点演算には対応していない．PPE は，主に計算のみを行う SPE と比べ汎用性が高い．SPE への命令実行，及び OS の管理に用いることが一般的であるとされている．

SPE は Synergistic Processor Unit (SPU)，Local Store (LS)，Memory Flow Controller (MFC) からなる 128 ビット SIMD 型のプロセッサである．SPE は libspe2 ライブラリを用いて C などの高級言語によって PPE から制御することが可能である．SPU は 128 ビット長のレジスタを搭載した SIMD 命令を持つ演算機である．1 サイクルあたり 4 並列で演算を行うことが可能であるが，SPE における単精度浮動小数点演算において丸め誤差は切り捨てられ，IEEE754 に準拠しない．SPE は専用のメモリを搭載し，容量は 256KB である．LS は 128bit/cycle アクセスが可能である．

2.2 libspe2 を用いたプログラミング

本節では，Cell/B.E. における一般的なプログラミング手法について述べる．

我々が数値計算などで用いるプログラムを Cell/B.E. 上で動作させても，SPE を利用することはできない．SPE の計算資源を利用するためには，libspe2 と呼ばれるライブラリを使用する必要がある．ユーザは PPE と SPE で動作するプログラムをそれぞれ記述し，SPE を扱うために抽象化されたコンテキストに対し処理を行うことによって PPE から制御を行う．また SPE は LS にあるデータに対してのみ処理を行うことが可能であるため，対象のデータを DMA 転送によってメインメモリから取得する必要がある．LS のデータはメインメモリとは完全に独立しており，プログラマによってメモリへのアクセスパターンなどを詳細に記述できる反面，高度なプログラミング技術が求められる．

3. 関 連 研 究

本章では，Cell/B.E. を計算資源として利用した例とその課題について，さらに本研究報告で述べるオフロード機構との差異について述べる．

Cell/B.E. は PlayStation3 に搭載されていることでも知られており，その費用対性能の高さから，科学技術計算への利用を試みる例も多数報告されている[3]．汎用の通信ライブラリである mpich や OpenMPI を用いて計算資源を利用した例や，スレッド仮想化環境を用いて Cell/B.E. の計算資源を活用する例が報告されている[4),5]．また，Cell/B.E. 上でタスクをオフロードする機構としてジョブキューを用いることによってタスクを管理し，負荷分散を考慮して SPE にジョブを割り当てるライブラリの開発も行われている[6]．

本研究報告での提案は，Intel x86 等の汎用プロセッサを含む環境での利用を想定し，ク

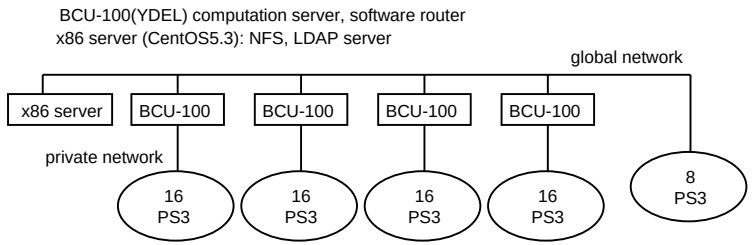

図 3 Cell/B.E. 及び x86 の 2 種類のプロセッサからなるクラスタの概要
Fig. 3 Outline of cluster contains Cell/B.E. and x86 processor

ラスタに拡張していった場合でも，柔軟性に優れる点で優位であると考える．プログラマビリティと柔軟性を確保しつつ，異種のアーキテクチャが混在する環境において Cell/B.E. の計算資源を活用することを考え，提案をおこなう．

4. 設　　計

本節では，提案するオフロード機構について，想定する環境，及び機構の利用方法についてまとめる．

4.1 想定する環境

現在，我々は SONY-BCU100 4 基, PlayStation3 72 基, 及び Intel Xeon サーバの 2 種類のプロセッサからなるクラスタを所持している．概要を図 3 に示す． 同図に示すような，汎用プロセッサと Cell/B.E. が混在した環境を想定している．上記のような環境で Cell/B.E. の計算資源を利用する場合，汎用プロセッサと Cell/B.E. の間は mpich や OpenMPI などの通信ライブラリを用いて各ノードの制御を行う必要がある．加えて，各 Cell/B.E. は SPE を pthread ライブラリなどを用いて制御する必要があり，プログラマは 2 種類の異なるプログラミングを習得する必要がある．また利用する Cell/B.E. の数を，要求される処理に応じて変更可能とするため，オフロード機構に拡張性を持たせる必要がある．

4.2 提 案 手 法

前述の通り，Cell/B.E. プログラミングでは，8 個の独立して動作する SPE 用プログラムを記述し，かつ PPE-SPE 間の DMA 転送によるデータ送受信を明示的に記述する必要がある．ここでは，それらの複雑なプログラミングを隠蔽し，かつ SPE の計算資源を利用するための機構の設計手法を述べる．はじめに，本機構の概要を図 4 に示す． 今回は汎用プロセッサと Cell/B.E. が接続された環境を想定し，ユーザは Cell/B.E. に対し演算対象

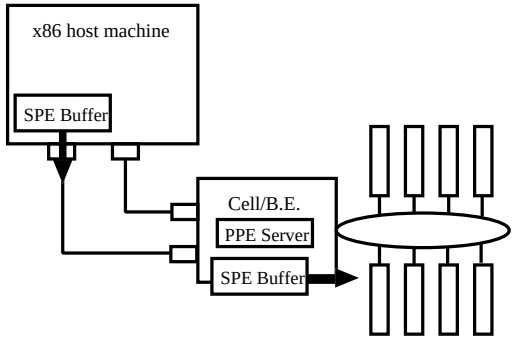

図 4 オフロード機構の概要
Fig. 4 Outline of offload structure

となるデータを転送し，処理を行う．その際には，あらかじめ SPE の LS に送信するためのバッファを確保し先頭アドレスとサイズを指定することによって SPE への転送を行うこととした．

Cell/B.E. 側には，SPE にデータを送信するためのプログラムをあらかじめ転送しておき，ノード側のプロセッサにおいてプログラムを実行する，現状では，ホスト - ノード間のデータ送受信の手段としてソケット通信を用いる．

ここまでに述べた機能を実現するために，以下の機能を実現する必要がある．

- ソケット通信により，ホスト - ノード間の通信を行う
- SPE に対して，任意のサイズのデータを送信する
- 指定した SPE からデータを受信する

本機構を利用するためには，ユーザは汎用プロセッサを対象としたプログラムに対して変更を加える必要が生じる．具体的には，SPE に対して送るデータのアドレスなどを指定し，関数を呼ぶことによりデータの送信，受信を行う．また，Cell/B.E. の性能を発揮するためには SPE 上で動作するプログラムについて，SIMD 命令等を用いたチューニングが不可欠であるため，SPE プログラム作成の余地は残す必要がある．上記をまとめると以下のようになる．

- ユーザはホスト - ノード間の通信を意識することなく，プログラムを記述することが可能となる
- ユーザは既存のプログラムについて，SPE で処理をするデータのサイズ，及びそのア

表 1 関数名と機能の概要
Table 1 List of functions

関数名	機能
API_Initialize	ノード間通信の確立
API_Finalize	通信の切断
SPESend	SPE に対してデータの転送を行う
SPERecv	SPE からデータを受信する
SPERun	SPE による計算を開始する

ドレスをホスト側のプログラムに通知するのみで，データを転送することが可能となる

- ホストマシンから受信したデータに対して，どのような処理を施すかを記述した SPE プログラムの作成はユーザが行う

5. 実　　装

本章では，前章で述べた設計に基づき，実装を行う．今回は計算資源の利用に際して以下の関数を実装する．関数の一覧を表 1 にまとめる． また，同表の関数を用いたホスト側 · ノード側のプログラムの一例を図 5 及び図 6 にそれぞれ示す． ユーザが SPE に対してデータを送信 · 受信したい場合には SPESend，及び SPERecv 関数を用い，データを送信した後の SPE による処理は SPERun 関数により実現できる．

5.1 PPE 側に配置するサーバの実装

通常，ネットワークによって汎用プロセッサと Cell/B.E. を搭載するプロセッサを接続してもデータを直接 SPE に転送することは出来ない．この問題を解決するため，ホストマシンと SPE の間でデータ送受信の仲介を行う簡易的なサーバプログラムを作成し，PPE に常駐させる．これは汎用プロセッサからネットワーク越しに信号を受信し，その種類に応じて SPE に対して命令を発行する．今回作成したサーバの機能を以下にまとめる．

- ホストマシンとの接続をソケット通信によって確保し，データの転送を行う
- ユーザが指定した任意の数の SPE を制御するためのスレッドを作成し，管理を行う
- ホストマシンから受け取ったデータを DMA 転送によって SPE へ転送する
- ホストマシンからのメッセージを受信し，DMA 転送によって SPE から PPE のメインメモリへデータを転送する

上記の機能をもつサーバをノードマシン上で実行した後，ホストマシンで実際に処理を行うプログラムを実行する．

```
#include <stdio.h>
#include <stdlib.h>
#include "vcell_runtime.h"
#define NUMSPE 8 // Number of SPE

int main(int argc, char *argv[]) {
    // buffer for SPE
    unsigned char buffer[1024]
    API_Initialize();

    for (size_t i = 0; i < NUMSPE; i++) {
        SPESend(i, (void *)&buf[0],
                sizeof(buf));
        SPERun(i);
    }

    for (size_t i = 0; i < NUMSPE; i++) {
        SPERecv(i, (void *)&buf[0],
                sizeof(buf));
    }
    API_Finalize();

    return EXIT_SUCCESS;
}
```

図 5 プログラムの一例 (ホスト側)
Fig. 5 Example of program(Host machine)

```
#include <stdio.h>
#include <stdlib.h>
#include "vcell_runtime.h"
extern unsigned char buf[BUFNUM][BUFSIZE];

int main(unsigned long long spe, ...)  {
    API_Main(spe, my_func);

    return EXIT_SUCCESS;
}

void my_func(void) {
     // Do something.
}
```

図 6 プログラムの一例 (ノード側)
Fig. 6 Example of program(Node machine)

5.2 関数の詳細

ここでは表 1 に示した関数について詳説する．各関数に用いる引数及び機能は以下の通りである．

- API_Initialize
  ソケット通信を用いて Cell/B.E. を搭載したマシンと接続を行う．
- API_Finalize
  通信を終了し，ソケットの破棄を行う．
- SPESend
  使用する SPE の番号を第 1 引数として指定し，対象とする SPE に対し，第 2 引数のアドレスより指定したバイト分書き込む．ソケット通信と DMA 転送の仲介は PPE 側のプログラムが自動的に行う．
- SPERun
  対象とする SPE 番号を指定し， 後述する API_Main 関数によりユーザが指定した関数の実行を開始する．各 SPE はデータを通信した段階では処理は開始せず，SPERun

が実行されて初めて処理を開始する．処理の終了後は再び待機状態に戻る．

- SPERecv
  使用する SPE の番号を第 1 引数として指定し，対象とする SPE に対し，第 2 引数のアドレスに対し指定したバイト分データを読み込む．ソケット通信と DMA 転送の仲介は PPE 側のプログラムが自動的に行う．
- API_Main
  SPE コンテキストの実効アドレス (通常の SPE プログラムの第 1 引数) に対し，第 2 引数で指定した関数を処理させる．ユーザが SPE プログラムを記述する際には，API_Main 及びデータに対する処理を施す関数のみを記述すればよい．

従来の Cell/B.E. プログラミングにおいて煩雑とされているデータの通信部分を関数によって隠蔽しているため，ユーザはどのような処理を施したいかのみを記述すればよい．データを格納するためのバッファは，高速化のために必要なダブルバッファリングをサポートするよう，初期状態で 2 つの配列を確保している．この配列長はユーザ側で，SPE の LS 容量を超えない範囲で任意に拡張可能である．ユーザはアプリケーションに合わせて適切なバッファの容量，数を選択することが可能である．例として 2 つのベクトルの内積演算を行う場合には，2 つのバッファにそれぞれベクトルのデータを転送した後，SPERun 関数によって処理を実行すればよい．

6. 予 備 評 価

前章までに，オフロード機構の設計，及び実装について述べた．本章では，実装したオフロード機構に対して，ノード間の通信性能と，Cell/B.E. が本来用いる DMA 転送とのスループットの比較，及びオフロード機構によって得られる性能，並列効果について述べる．

6.1 評 価 環 境

今回の評価は汎用プロセッサをもつマシンとして Intel Quad-Core Xeon 2.0GHz を搭載したサーバ，Cell/B.E. を搭載したマシンとしては SONY BCU-100 を用いた．2 台のマシンは Gigabit Ethernet により接続された環境で評価を行った．評価環境の詳細を表 2 に示す．

表 2 評価環境
Table 2 Environment of evaluation

	Intel Xeon	SONY BCU-100
CPU	Intel Xeon 2GHz	Cell Broadband Engine 3.2GHz
Memory	2GB	1GB
OS	CentOS 5.4	Yellow Dog Linux 6.0
Compiler	gcc 4.1.1	ppu-gcc 4.1.1, spu-gcc 4.1.1

6.2 通信速度の評価

ここではオフロード機構を用いた場合の通信遅延に関して評価を行う．ホスト・ノードから大きさを変えたデータをそれぞれ送信した場合のスループットと Cell/B.E. における一般的な DMA 転送との速度比較を表 3 に示す． 同表よりわかるように，Cell/B.E. において通常用いる DMA 転送とホスト・ノード間の通信速度を比較した場合，GET の場合で約 0.011 倍，PUT では約 0.007 倍の速度となった．Gigabit Ethernet で接続された環境においてソケット通信でのデータ送受信を行っているため，この結果は妥当であると考えられる．なお，DMA 転送では，理論上の最大性能が出るとされている 128 バイトアラインメントにより測定を行った[7]．

表 3 Cell/B.E. の DMA 転送との速度比較
Table 3 Compared with DMA transfer on Cell/B.E.

	Throughput
DMA Transfer (PUT)	14.5 GByte/sec
DMA Transfer (GET)	23.3 GByte/sec
Gigabit Ethernet	165 MByte/sec

6.3 モンテカルロ法による評価

モンテカルロ法は乱数によって数値計算やシミュレーションを行う手法である．今回の評価はモンテカルロ法を用いて円周率の計算を行った．モンテカルロ法は並列度が高く，データを分割しても依存関係が生じない．

今回はテストケースとして，汎用プロセッサと提案手法で試行回数を 1000×1000 回とし，実行時間を測定した．結果を図 7 に示す．また，SPE1 基で実行した時間を 1 に正規化し，比較した性能向上のグラフを図 8 に示す． 同図よりわかるように，並列度の高いアプリケーションに対しては，通常の Cell/B.E. を用いた場合と同様，高い並列効果を得ることができた．

7. 結論と今後の課題

本研究報告では，Cell/B.E. をアクセラレータとして使用するためのオフロード機構について提案し，その評価を示した．結果を以下にまとめる．

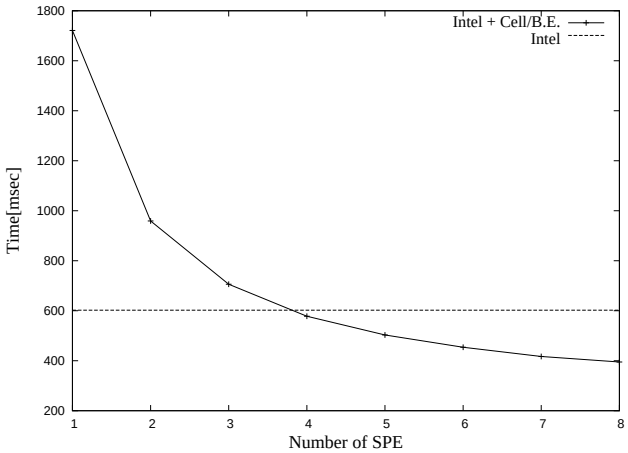

図 7 使用した SPE 数と実行時間の関係, 水平方向の波線は汎用プロセッサでの実行時間を示す.
Fig. 7 Relation between number of SPE and execution time. (horizontal line shows execution time of general-porpose processor.)

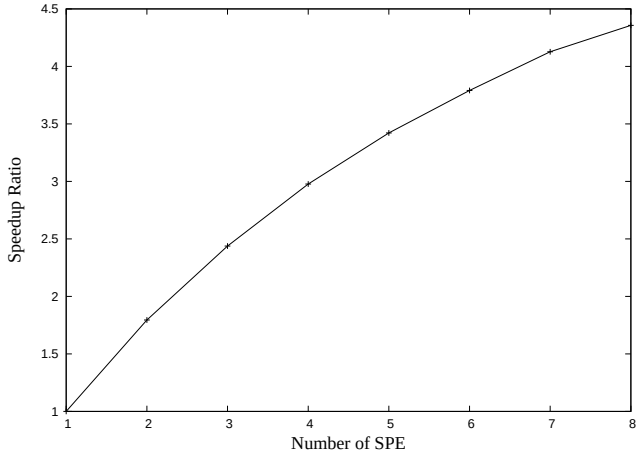

図 8 SPE1 基を用いて計算した時間を 1 として正規化した場合の倍率
Fig. 8 Speed up ratio (compared with 1 SPE.)

- 提案したオフロード機構を用いることで，ユーザは汎用プロセッサ用のプログラムに対し，5 種類の記述を追加するのみで SPE を制御し，その計算資源を利用できることを確認した
- 並列性の高いモンテカルロ法を例にとり，通常の Cell/B.E. のみで行う計算と同様，並列計算に利用できる可能性を示した
- 現状では，並列度の低いアプリケーションに対しては SPE の計算資源を効果的に利用することは難しい．これは Cell/B.E. の計算資源を通常のプログラミング手法により使用した場合と同様の課題である

本研究報告による評価は，Gigabit Ethernet で接続されたクラスタを用いて評価を行った，このためノード間の通信遅延が大きく，Cell/B.E. を用いることによる性能向上を得るためには通信がボトルネックとなることが明白である．通信遅延の問題に関しては，今後は Infiniband 等の機構をもつクラスタでの性能評価を行う予定である．また，今回の予備評価では各ノード間のデータ共有に関しては言及していない．データ共有の問題を解決するために MPI などを用いてノード間通信の機能拡張を行うこと，加えて mpich や OpenMPI に代表される通信ライブラリを用いた場合との性能比較を今後の課題とする．

参 考 文 献

1) 近藤伸宏：“CELL プロセッサに見るアーキテクチャ - 次世代デジタルホームに向けて”，東芝レビュー， Vol.60, No.7, pp.48–51 (2005).
2) K.J.Barker, K.Davis, A.Hoisie, D.J.Kerbyson, M.Lang, S.Pakin and J.C.Sancho: Entering the Petaflop Era: The Architecture and Performance of Roadrunner, *SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pp.1 – 11 (2008).
3) Kistler, M., Gunnels, J., Brokenshire, D. and Benton, B.: Programming the Linpack benchmark for the IBM PowerXCell 8i processor, *Scientific Programming*, Vol.17, No.1-2, pp.43–57 (2009).
4) A.Buttari, J.Kurzak and J.J.Dongarra: Limitation of the PlayStation3 for High Performance Cluster Computing, *Technical Report CS-07- 597* (2007).
5) 山田昌弘，西川由理，吉見真聡，天野英晴：Cell Broadband Engine を用いたスレッド仮想化環境の提案，電子情報通信学会コンピュータシステム研究会 (CPSY) (2010).
6) Kunzman, D.M. and Kale, L.V.: Towards a Framework for Abstracting Accelerators in Parallel Applicatoins: Experience with Cell, *SC'09: Proceedings of the 2009 ACM/IEEE conference on Supercomputig*, pp.1– 2 (2009).
7) System and TechnologyGroup, I.: Developing Code for Cell DMA (2005).