An Implementation and Evaluation of Virtualization Environments for PC Cluster with Accelerators

Masato YOSHIMI[*], Akihiro SHITARA[**], Toshiaki KAMATA[**], Masahiro YAMADA[**], Yuri NISHIKAWA[**], Mitsunori MIKI[*], Tomoyuki HIROYASU[***] and Hideharu AMANO[**]

(Received April 20, 2011)

PC clusters including accelerators such as Cell/B.E. and GPU became popular computing system to achieve high-performance maintaining low-energy consumption. One of serious problems to use these systems is the difficult implementation of the parallel programming. This paper proposes a virtualization environment to hide data communications among computing nodes each of which has an accelerator to reduce implementation costs. Programmers using this environment can use multiple nodes by implementing a program for a single node. Virtualization environments for Cell/B.E. and GPU cluster are discussed about the influence for the performance.

Key words ： GPU, Cell/B.E., PC cluster

キーワード ： GPU, Cell/B.E., PC クラスタ

アクセラレータを搭載したPCクラスタのための仮想環境の実装と評価

吉見真聡・設楽明宏・鎌田俊昭・山田昌弘・西川由理・三木光範・廣安知之・天野英晴

1. はじめに

GPGPU，Cell などのアクセラレータの普及と共に，これを多数装備したクラスタシステムが高性能計算に利用されるようになった．これらのアクセラレータはコスト性能比に優れ，並列性の高いアプリケーションでは高い性能を発揮する．現在の Top500 ランキングの上位にはこのようなアクセラレータを多数用いたシステムが並んでいる [1)]．しかし，これらのアクセラレータを用いたシステムは並列プログラミングに関する問題を抱えている．

これらのアクセラレータは，それぞれがメニーコアシステムであるため，単一ノードで用いる場合でも並列プログラミングを行わなければならない．複数ノードで用いる場合，単一ノード内の並列処理とノード間の並列処理を階層的に記述する必要が生じ，プログラマの負担が大きい．GPGPU においてはノード内の

* Department of Intelligent Information Engineering and Sciences, Doshisha University, Kyoto
Telephone/Fax:+81-774-65-6780, E-mail:myoshimi@mail.doshisha.ac.jp
** School of Science for Open and Environmental Systems, Keio University, Yokohama
Telephone/Fax:+81-45-566-1748, E-mail:asap@am.ics.keio.ac.jp
*** Department of Life and Medical Sciences, Doshisha University, Kyoto
Telephone/Fax:+81-774-65-6932, E-mail:tomo@is.doshisha.ac.jp

記述にOpenCL[2]を使い，ノード間のデータ転送をMPIを用いて記述する必要がある．一方，Cellでは，libspe2およびpthreadライブラリを用いてノード内のプログラムを記述し，ノード間のデータ転送はやはりMPIを用いる必要がある．これら2段階の並列プログラミングには高度なスキルが要求される．

本研究では，GPGPUとCellの2種類のアクセラレータに関して，仮想的に単一ノード中に多数のコアが存在するように見せる環境を構築する．この環境を用いることで，ネットワーク上にあるアクセラレータを単一ノードのように利用できるようになるため，開発コストを低減させることができる．

本論文は以下のように構成されている．まず2. 章で関連研究について述べる．3. 章ではGPGPUの仮想環境，4. 章ではCell/B.E.を対象とする仮想環境について，設計と評価をそれぞれ述べ，5. 章でまとめる．

2. 関連研究

アクセラレータを搭載したPCが一般的になったことから，PCクラスタをはじめとするネットワーク上のアクセラレータの利用を容易にしようとする研究が多く行われている．

ネットワーク上のGPUを計算資源として活用しようとする研究[3]では，GPUタスクを投入することができるアイドル状態のPCを検出する．GPUを用いたグリッドコンピューティング環境は，既に科学計算分野で実際に利用されている[4]．

Cell/B.E.およびそのクラスタを用いた科学技術計算については多くの既存研究があり，またそのプログラム開発を支援する環境やミドルウェアも存在する．例として，Cell/B.E.搭載マシンとして複数のPlayStation3を用い，mpichやOpenMPIに代表される汎用の通信ライブラリを用いて，PPEからSPEへとジョブを自動的にオフロードして負荷分散する機構や，離れたノードのSPEを仮想的に1つのプロセッサとして見せかけ，単一のプログラムから制御するスレッド仮想化環境などが提案されている[5][6]．

グリッド環境で計算資源を利用する例としてはNinfプロジェクトが挙げられる[7]．これはネットワーク上に分散配置された計算資源を効果的に利用するためのプログラミングミドルウェアである．遠隔地のノードに対してタスクをオフロードしたり，MPIでは実現の難しい耐障害性に優れたシステムの設計に用いられている．また，汎用プロセッサ向けにプログラミング言語Javaを用いてMPIを隠蔽したライブラリを実装し，既存のプログラムについて，わずかな修正のみでクラスタ環境に適応することが可能なXcalableMPが2010年11月に公開された[8]．これは，単一のノードで動作するプログラムに対して#pragmaディレクティブを挿入することで，クラスタ環境で実行可能とするものである．

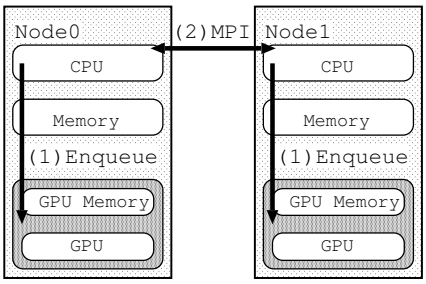

Fig. 1. The traditional concept of programming model by combination MPI and OpenCL.

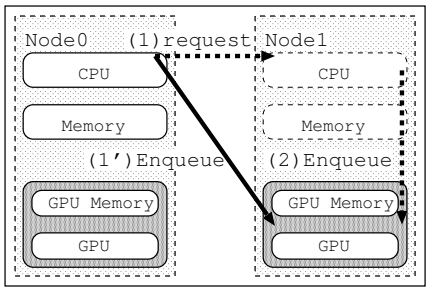

Fig. 2. The concept of virtualized OpenCL programming.

3. GPGPUの仮想化環境

OpenCL対応のGPUが搭載された計算ノードがEthernetネットワーク上に複数台接続された環境を対象に，OpenCLとMPIを組み合わせてアプリケーション開発する際のプログラミングの繁雑さを改善するミドルウェアを実装し，評価を行った．

3.1 設計

通常，OpenCLプログラミングをマルチノード環境で行う場合，まず，1台のノード上で動作するOpenCL

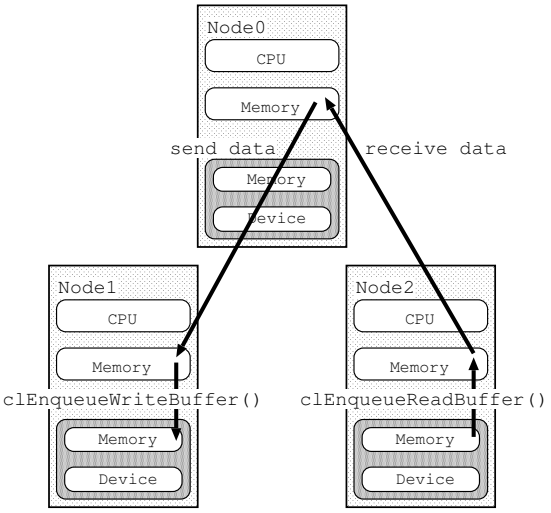

Fig. 3.　Data transfer between GPUs via an user application.

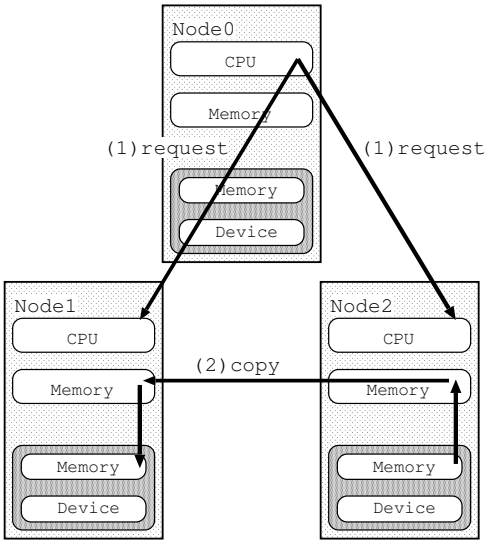

Fig. 4.　Data transfer between GPUs using VDMA functions.

プログラムを，ホスト側とアクセラレータ側のそれぞれに実装する．GPU 上で動作するソフトウェアは，各アクセラレータの持つキューにタスク投入することで実行される (Fig.1 (1))．続いて，プログラマは MPI ライブラリを用いてホスト側プログラムを拡張する形でノード間の通信制御を実装する (Fig.1 (2))．この通常の手法において，プログラマは OpenCL と MPI を組み合わせたソースコードを記述する必要があるため，高いプログラミング技術を要し，デバッグも困難である．そこで本研究では，ホストとなる 1 台のノード上の CPU プロセスから，他ノード上の GPU の持つキューに対してタスクを投入できるミドルウェアを開発した．実装には UNIX/Linux の TCP ソケットプログラミングを用いた．プログラマが記述した OpenCL アプリケーションが Fig. 2 の Node0 上で実行されるときの，仮想化環境のイメージを Fig.2 に示す．また，GPU 間でデータを転送するときの動作を Fig.3 に示す．このような環境における性能低下の要因として，複数の計算ノード間てデータ転送を行う際に，ミドルウェアを実行しているホスト計算ノードのメモリを介してデータを転送し，転送時間が増大する問題が挙げられる．

これを解消するために，本研究では，Fig.4 に示す Virtual Direct Memory Access(VDMA) 転送の機能を新たに実装した．

3.2　評価

直交格子法による移流項の計算 [9] を Cubic ラグランジュ補間を用いた CUDA 実装 [10] を OpenCL に移植し，ミドルウェアおよび，VDMA 機能の性能を評価した．直交格子法による移流項の計算を Fig.5 に示す．

性能評価では，Ethernet で接続された NVIDIA

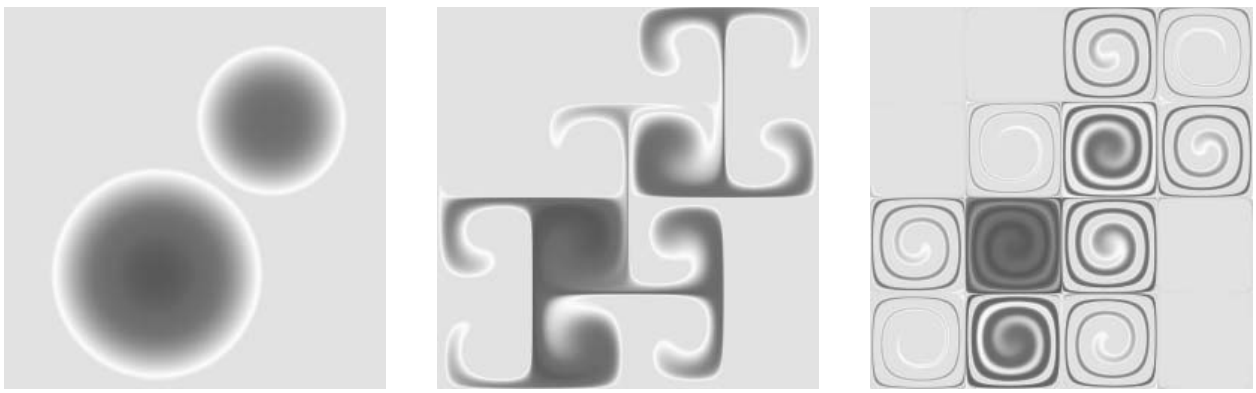

Fig. 5.　Ink diffusion.

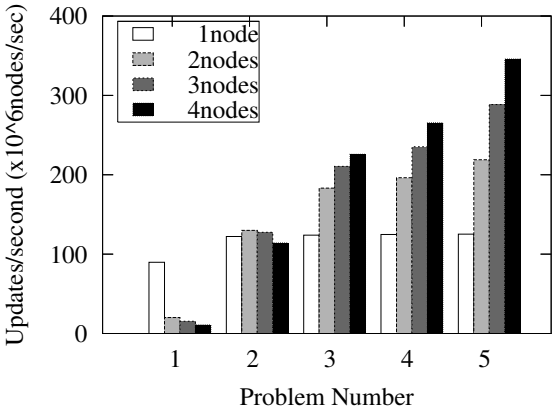

Fig. 6. Performance versus number of node using VDMA functions.

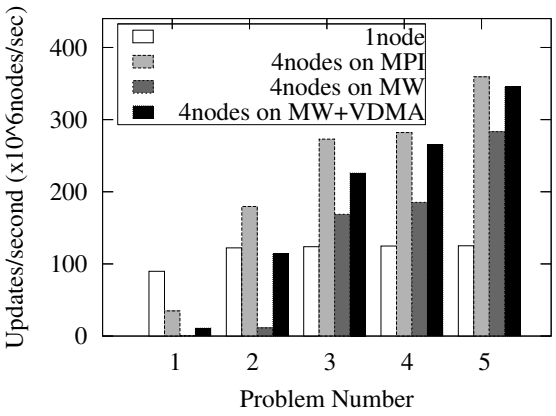

Fig. 7. Performance conparison between MPI and VDMA.

GeForce 9500GT GPU を 1 枚搭載したを 4 台の PC(Intel Core2Quad 2.83GHz) 環境においてミドルウェアを用いた場合の台数効果を調べた．ベンチマークには行列積の計算と, 直交格子法の移流項の計算を用いた. 各問題の問題サイズを Table1 に示す．結果, VDMA 機能を用いない場合でも 1.7 倍，2.0 倍，2.4 倍，VDMA 機能使用時では 1.7 倍，2.3 倍，2.7 倍にそれぞれ性能が向上した．ここで，OpenCL と MPI 記述を併用してチューニングを Fig.った場合の性能を 100%としたとき，4 ノードで VDMA 機能を使用しない場合は 78%程度であった性能が， VDMA 機能を用いることで 96%を達成し，VDMA 機能による性能の向上が確認された.

Table 1. Size of each question.

Q. ID	X	Y
1	256	256
2	1024	1024
3	2048	2048
4	4096	2048
5	4096	4096

4. Cell/B.E. の仮想化

4.1 設計

Cell/B.E. の仮想化環境では，Cell/B.E. 搭載マシンがネットワーク接続された状況を対象とする．この場合，サーバ · クライアント型のプログラミングモデルを使用し，クライアントとなる Cell/B.E. ノードがプログラムを実行し，負荷に応じてネットワークで接続された複数のサーバノードに処理をオフロードする．提案するミドルウェアの目的は，開発者に対して SPE プログラムの最適化のみに注力することが可能な開発環境を提供することである．通常，OpenMPI やソケット通信などを用いて Cell/B.E. 間で通信を行った場合, SPE に対して直接データを送信することはできない. この問題を解決するため，ホスト · ノード間及び PPE ·SPE 間の通信を仲介するサーバを設計し，ノードマシン上で動作させることを考える．サーバプログラムが持つべき機能を以下にまとめる．

- ソケット通信を用いてホスト · ノード間の通信を確立する
- ホストマシンと指定した SPE 上の LS 間で，任意のサイズのデータを送受信する

4.2 実装

前節で述べたミドルウェアの機能を実現するため, 以下に示す機構を実装した．

- ノードマシンとの通信ををおこなう API
- 転送を仲介するサーバプログラム
- SPE を仮想化するための Virtual SPE クラス

本章では，上記の各実装について，その詳細を述べる．

Table 2.　Correspondance table for functions.

Name	Function
API_Initialize	The function is called automatically to connect the server program when the program is started.
API_Finalize	The function is called automatically to disconnect the session right before the program is started.

Table 3.　Correspondance table between function and method.

Name	Function
Vspe.Send	Sending data to a SPE
Vspe.Recv	Receiving data from a SPE
Vspe.Run	Starting computation

4.2.1　ノードマシンとの通信 API の実装

ノードマシンとの通信及び制御のため，3 つの関数を実装した．プログラムの開発者はあらかじめ使用するマシンの IP アドレスを列挙した設定ファイルを用意した上で，本ミドルウェアを使用する．各関数の詳細を Table2 にまとめる．

4.2.2　サーバプログラムの実装

通常，ネットワークで接続した Cell/B.E. を使用する際に，OpenMPI などの通信ライブラリを用いた場合，データを直接 SPE に転送することはできない．この問題を解決するため，ホスト · ノード間及び PPE-SPE 間の通信を仲介するサーバプログラムを設計し，ノードマシン上で常駐させるものとした．このサーバプログラムは，ネットワークで接続されたホストマシンから動作の種類を示すサーバプログラムは Socket 通信によってホストマシンとのデータ通信を行う．また，自身の SPE について制御を行う． これにより，開発者は SPE 用プログラムのみ記述すれば良い．

4.2.3　Virtual SPE の実装

ネットワークで接続された Cell/B.E. の SPE を操作するため，開発者は Virtual SPE を用いる．Virtual SPE は物理的に存在する SPE と一対一で対応する．開発者はホストマシン上で Virtual SPE のオブジェクトを使用する SPE の数だけ宣言し，プログラム中から操作する．Virtual SPE がもつメソッドの機能を Table 3 に示す．

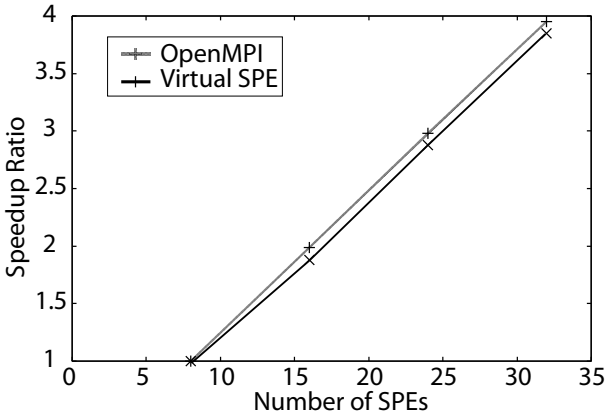

Fig. 8.　Performance versus number of SPEs.

4.3　評価

複数の SONY BCU-100[11] で構成したクラスタで実装したミドルウェアの評価を行った．実行環境を Table4 に示す．まず，モンテカルロ法をアプリケーションとして用いて並列効果を確認する．モンテカルロ法は確率を利用して近似解を求める手法である．モンテカルロ法を用いた円周率の計算 [12] を行った結果を Fig.8 に示す．

Fig.8 に示すように，Virtual SPE を用いた場合，OpenMPI を用いた場合と同様の並列効果が得られている．

次に行列積の計算を行う．おこなう．行列のサイズと使用する SPE の数をそれぞれ変化させ，測定した結果を Fig.9 および Fig.10 に示す．

Fig.9 および Fig.10 に示すように，行列のサイズが大きくなるに従って通信のオーバーヘッドが無視できなくなるものの，4096 次元までの行列演算においては 80%近い性能を得られている．その一方で，行列の次元数を大きくした場合には通信遅延がより顕著に現れ，OpenMPI と比較した場合に期待した性能が得られなかった．これはソケット通信によるものと，DMA 転送を複数回に渡って実行したことが原因だと考えられる．

本ミドルウェアの開発により得られた結果は以下の

Table 4.　Evaluation environment.

Hardware	SONY BCU-100
CPU	Cell/B.E. 3.2GHz
Compiler	{ppu, spu}-gcc 4.1.1
MPI	OpenMPI 1.3.3

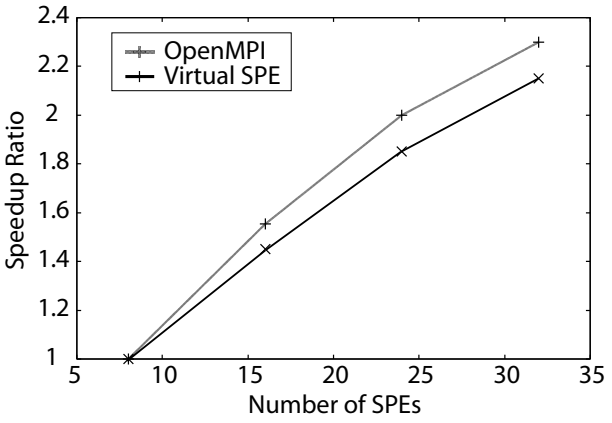

Fig. 9. Performance versus number of SPE in 4096 dimension.

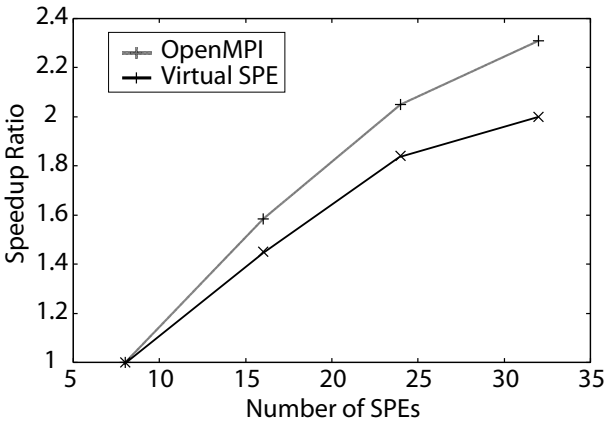

Fig. 10. Performance versus number of SPE in 8192 dimension.

とおりである．

- 本ミドルウェアを用いることで，通信を意識することなくSPEを利用することが可能となった
- モンテカルロ法など，並列度の高いアプリケーションを用いた場合については，OpenMPIを用いた通常の並列プログラミングと同様の並列効果を確認した．
- 行列演算についても，OpenMPIと同様の並列効果を確認することができた．但しデータサイズの増加に伴い，転送のオーバヘッドが顕著に現れた．
- 通信の遅延に関しては，インターコネクトの改善による性能向上の余地がある．

5. まとめ

本研究を通して，GPUおよびCell/B.E.が組み込まれたPCクラスタを対象に，プログラム開発を容易にするための仮想環境の実装と評価を行った．仮想環境はGPUやCell/B.E.が必要とする2段階の並列プログラミングのうち，ネットワーク上のノードに対するデータ通信部分を隠蔽する機能を持っており，単一ノードの場合とほぼ同じプログラムで複数ノードでの並列計算が可能になる．並列性の高いアプリケーションを使用した評価の結果，仮想環境の利用による性能の大幅な下落は確認されず，仮想環境の有効性が確認された．

今後の展開として，ノード間通信が多いアプリケーションの性能下落を抑えるために，仮想環境にキャッシュ機構を組み込む検討が挙げられる．

本研究の一部は，同志社大学理工学研究所研究助成金の助成を受けて行われた．

参 考 文 献

1) “Top500 Supercomputing Sites”, http://www.top500.org/.

2) NVIDIA, “The OpenCL Specification Version: 1.1”, (2009).

3) Y. Kotani, F. Ino, and K. Hagihara, “A Resource Selection System for Cycle Stealing in GPU Grids”, Journal of Grid Computing, **6(4)** 399–416 (2008).

4) A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, “Folding@home: Lessons From Eight Years of Volunteer Distributed Computing”, Proc. IEEE International Symposium on Parallel and Distributed Processing, 1–8 (2009).

5) D. M. Kunzman and L. V. Kale, “Towards a framework for abstracting accelerators in parallel applicatoins: Experience with cell”, Proc. the 2009 ACM/IEEE conference on Supercomputig, 1–2 (2009).

6) 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”, 信学技報, **110(3)** 27–32 (2010).

7) Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka, "Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing", Journal of Grid Computing, **1(1)** 41–51 (2003).

8) J. Lee and M. Sato, "Implementation and Performance Evaluation of XcalableMP: A Parallel Programming Language for Distributed Memory Systems", Proc. the 2010 39th International Conference on Parallel Processing Workshops, 413–420 (2010).

9) 情報処理学会主催 GPU チャレンジ 2010 実行委員会, "GPU Challenge 2010 規定課題マニュアル (ツールキット ver.0.60 対応版)", http://www.hpcc.jp/sacsis/2010/gpu/.

10) 須藤郁弥, 坂内恒介, 本田耕一, 松田健護, 篠原歩, "2GPU による Cubic セミ・ラグランジュ法の高速化", SACSIS2010 GPU Challenge 2010, (2010).

11) SONY, http://pro.sony.com/bbsccms/ext/ZEGO/files/BCU-100_Whitepaper.pdf.

12) "モンテカルロ法による円周率の計算", http://hp.vector.co.jp/authors/VA014765/pi/montecalro.html.