GPU 開発環境 CUDA を用いたゲーム探索の高速化

田野 文彦* 三輪 誠† 横山 大作‡ 近山 隆*

現在，GPU の高い処理能力を画像処理以外の汎用目的にも利用する GPGPU (General Purpose GPU) が注目されている．GPU は並列処理に優れることから，多くの並列性を有するゲーム探索においても有用であると考えられる．本稿では Blokus Duo の合法手生成の並列化可能な点に着目し，GPU で処理することで計算の高速化を行う．GPU を合法手生成に用いることでランダム対戦を CPU の約 1.2 倍に高速化できたことを示す．

Speeding Up Game Search with CUDA : A GPU Development Environment

Fumihiko TANO*, Makoto MIWA†, Daisaku YOKOYAMA‡and Takashi CHIKAYAMA*

Recently, GPGPU (General Purpose GPU) has been focused. GPGPU is utilization of GPU high performance to calculate general-purpose computation not restricted to image processing. GPU is effective in game search with massive parallelism because GPU excels in parallel computation. In this paper, we focus on parallelization of move generation in Blokus Duo. We illustrate a method utilizing GPU that shows 1.2 times speed-up in Blokus Duo move search.

1 はじめに

近年の GPU（Graphics Processing Unit）の性能向上に伴い, GPU を画像処理だけでなく汎用的に使おうとする試みがなされている．このような GPU の利用を GPGPU（General Purpose GPU）と呼ぶ．GPGPU では多数のプロセッサを動作させるため並列的な計算の場合 CPU（Central Processing Unit）より高速に計算をすることができる．本研究では GPGPU を用い，CPU で計算したときよりも高速なゲームプレイヤを構築することを目的とする．

画像処理専用チップである GPU の命令体系は通常の CPU とは大きく異なり，その命令体系で直接プログラミングをするのは，プログラムを開発する側にとって負担の大きいものであり，複雑な汎用の処理を GPU で行うことは困難な作業である．NVIDIA が発表した CUDA（Compute Unified Device Architecture）[3] は，GPU 上で行うプログラミングを，画像処理を意識せずに書くことを目的とした統合開発環境である．CUDA を用いることにより，行いたい計算を画像処理用の命令に人手で変換するという困難な作業を大幅に軽減することができ，行いたい計算部分の開発に集中することができる．

ゲームプレイヤにおいても並列処理が行える部分があり，その部分に GPU を使用することにより高速化できると考えられる．例えば，手生成や多数の局面を同時に評価することが挙げられる．本稿では CUDA を用いた，GPGPU 上での Blokus Duo の合法手生成の高速化を提案する．Blokus Duo の合法手を生成する際，CPU ではこれを逐次的に処理している．この手生成は，それぞれの手に依存性はないため，並列的に生成の計算を行ったとしても計算結果に違いはない．ゆえに GPU の並列性を生かした処理を行うことができる．

2 GPU と CUDA

2.1 プログラミングモデル

まず GPGPU の概要を述べる．GPGPU とは画像処理用のプロセッサである GPU に画像処理以外の汎用的な計算を行わせる技術であり，銀河の衝突

* 東京大学大学院工学系研究科 / Graduate School of Engineering, The University of Tokyo
† 東京大学大学院情報理工学系研究科 / Graduate School of Information Science and Technology, The University of Tokyo
‡ 東京大学 IRT 研究機構 / IRT Research Initiative, The University of Tokyo

や高分子の挙動の計算などの科学技術の分野に応用されている [1][2]．GPU は多数の処理ユニットを並列に用いることで高速な処理を実現している．これは画像処理は比較的計算負荷が大きい処理であるが，画素単位など，並列的に計算しやすいという特徴があり，このような負荷に対して並列処理が有効なためである．GPU の内部では Grid，Block，Thread という要素が図 1 に示すような関係となっている．Thread は命令を実行する単位であるが，同じ Block に属する Thread，同じ Grid に属する Block ではそれぞれ異なる種類のメモリを共有できる．

GPU を用いて効率的に並列化できた場合の計算能力は CPU より非常に高い．例として，単純な正弦関数の計算を 100 万回実行し，これを 1,000 回行い平均をとった．すると CPU では 72 [ms] であったのに対し，GPU では 4.4 [ms] で計算できた．したがって GPU では CPU より約 16 倍の速度が出ている．今回ゲームプレイヤに用いる計算とは性質が異なるが，この例のように GPU を有効に使うことができれば CPU より高速に計算できる．

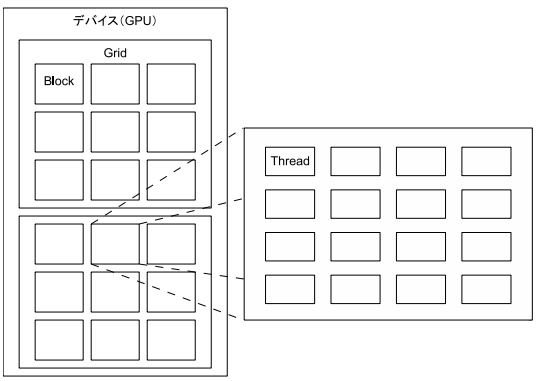

図 1　GPU の内部構成

2.2　メモリ転送の性能

GPU 上で計算をおこなう場合，CPU 側のメモリを GPU が扱うことができるメモリ領域に転送しなければならない．この点が CPU と異なり，転送にあまりに時間がかかりすぎると GPU の計算速度が活かせない．そこでどの程度転送に時間がかかるのか調べた．

メモリ転送に要する時間を各容量ごとに 100 回ずつ計測し平均をとった．CPU 側から GPU 側への転送の結果を図 2 に，GPU 側から CPU 側への転送の結果を図 3 に示す．図からわかるように CPU 側から GPU 側，GPU 側から CPU 側どちらも 100 [Bytes] 以下の少量のメモリ転送であっても 14 [μs] 程度の時間がかかる．また転送量を大きくするにつれて CPU 側から GPU 側では 9 から 10 [Gbps] 程度に，GPU 側から CPU 側では 8 [Gbps] 程度に収束する．この収束した速度は通常の CPU-主記憶間の転送速度に比べて 5 分の 1 から 4 分の 1 程度である．

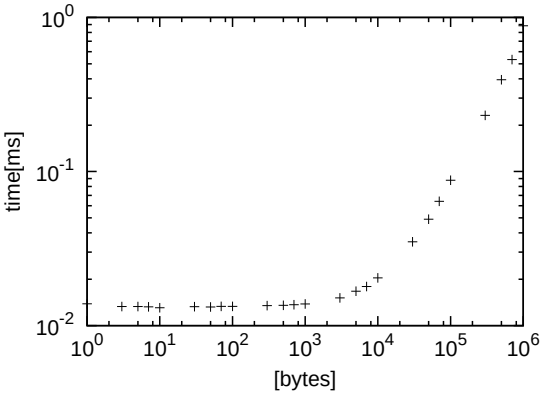

図 2　CPU 側から GPU 側へのメモリ転送時間

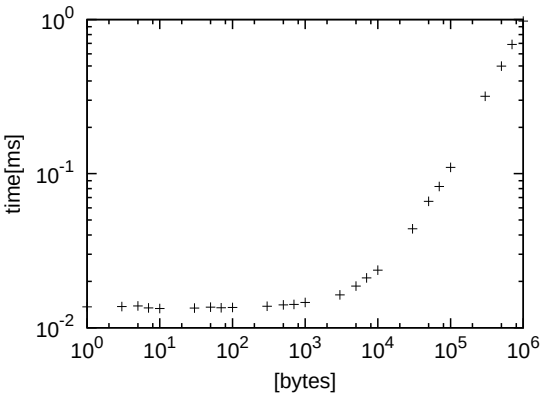

図 3　GPU 側から CPU 側へのメモリ転送時間

3　GPU による合法手生成の高速化

3.1　GPU の特徴

CPU と比較した GPU の特徴として

並列的な計算に向いている　多数のプロセッサを用いることにより一度にたくさんの計算を行うことができる．一方で逐次的な命令に対しては CPU より計算性能が低い．

メモリ転送に時間がかかる　計算を GPU で行うとき，CPU 側からメモリを転送しなければならず，時間がかかってしまうため GPGPU では CPU 側と GPU 側との間でのデータのやり取りが少ないほうがよい．

ということがいえる．

以上の点を踏まえると，計算結果をほかの計算で使うといった依存性がなければそれらの計算の要素を並列化でき，GPU による並列処理で高速化できる．ゲームプレイヤにおいて依存性なく並列化しやすい部分としては，合法手生成が挙げられる．Blokus Duo では各駒が盤面のどこに置けるかというのは駒を回転や移動をさせ，後に述べる方法によって判断する必要がある．これを読む局面ごとに手持ちの駒数分行うため，計算量が大きい．一方でこの処理は異なる駒同士についての計算に依存性がなく，さらに局面や駒といったデータは画像よりも容量が小さくメモリ転送の時間も大きくない．したがって合法手生成は並列化し GPU で計算する部分として適している．この部分を並列化させ，合法手生成の計算時間を GPU と CPU とで比較し評価を行う．

本稿では, Blokus Duo の合法手の生成を, CUDA を用いて GPU 上で処理する方法を示し，その性能評価結果を示す．合法手の生成は計算負荷の大きな部分を占め，GPU の並列処理性能を生かしやすい部分であるため，この処理に着目した．以降に具体的な合法手生成の方法を述べる．

3.2 合法手の判定

合法手を生成するには，各ポリオミノについて図 4 のようなデータをあらかじめ保持しておき，これを使いマスク処理を行う．駒は 21 個あるが，あらかじめ回転・反転をして対称のものを除いた 91 のパターンを用いた．これらのパターンは 1 マスずつが char 型でありその 2 次元配列で表現している．

プレイヤー 1 とプレイヤー 2 のそれぞれ 1 手目と，2 手目以降では合法手の条件が異なる．それぞれの 1 手目では盤面の座標で (5,5) と (10,10) にオレンジやパープルのポリオミノがあればよい．以下ではそれぞれの 2 手目以降の合法手生成について述べる．ポリオミノを置くには，ポリオミノの角と自分の色の角がつながりかつ，辺同士はつながらず，すでに置いてあるポリオミノに重ならないという条件があるが，その条件を角の近傍の領域に同じ色が来て，辺の近傍の領域には同じ色が来ないといった等価な条件に置き換えマスクをかけていく．これを上下左右の並行移動，回転，反転のすべての 13,729 パターンに対して試み，条件に合う合法手のポリオミノを探していく．

この処理を本稿は CUDA 上でポリオミノのパターンと並行移動において並列化を行った．

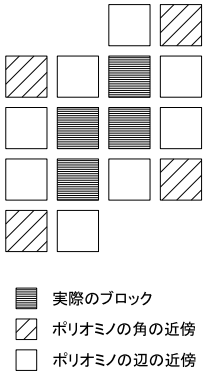

図 4　ポリオミノの例

3.3 差分処理

1 回の対戦において，局面ごとに盤面のすべての位置でポリオミノを置けるかどうかを判定する必要はない．自分が駒を置き次に自分が置くまでに盤面が変化した箇所は，前に判定してから次に置いた自分と相手の駒の図 4 に示したような実際のブロック，ポリオミノの角と辺の近傍の領域だけである．前に判定してから次に判定するまでに盤面の中で変化した領域に，駒が入った場合には判定の計算をする必要がある．しかし，前回の判定から変化していない部分は前回の判定をそのまま使うことができる．この差分処理を実装することで計算の効率が上がる．

この差分処理の効果によって計算時間が削減される合法手判定と，そうでないものがある．例えばすべての要素を同時に並列的に計算している場合には 1 番遅い要素の計算時間が全体の時間になるため，時間削減には効果がない．しかし GPU で並列化した場合，実際にはすべての駒のすべての位置を同時に計算しているわけではなく，限られた数の処理ユニットが次々に要素を計算している．ゆえに GPU であっても計算すべき要素が十分に多い場合，差分処理は効果のあるものである．

4 評価

GPU 上と CPU 上で Blokus Duo の 1 局面あたりの判定時間とランダム対戦の計算時間とを評価する．実験に用いた環境は

CPU　Xeon 2.80 GHz（NetBurst マイクロアーキテクチャ世代の Nocona）
メモリ　2GB
GPU　GeForce 9800 GTX
OS　Linux （Ubuntu 8.04）

開発環境・コンパイラ　CUDA 2.0 Beta，nvcc release 1.1，V0.2.1221

である．また GeForce 9800 GTX の主な仕様は

プロセッサーコア　128
グラフィッククロック　675 MHz
プロセッサークロック　1688 MHz
メモリクロック　1100 MHz
メモリ容量　512MB

である．

ゲームプレイヤの構成を述べる．差分処理を実装した GPGPU でのランダム対戦の流れは図 5 のようになる．はじめに CPU 側から GPU 側へ駒のデータを送る．次に GPU 側では手生成を行い，CPU 側では生成された合法手の中からランダムに手を選択する．手の生成と選択とをプレイヤー 1 とプレイヤー 2 が交互に行い，どちらも合法手がなくなったところでゲームを終了する．

1 局面あたりの実行時間では，何も駒の置いていない初期局面を用意し，差分処理は使わずに仮想的に全ての手生成を判定させた．この実験は実際の Blokus Duo の対局における初手を求めているわけではない．GPU 側ではメモリ転送の時間も含んでいる．表 1 にこの計算の 1,000 回の平均値を求めた結果を示す．ランダム対戦では合法手は差分のみマスク処理をそれぞれの手について調べ，これを 1,000 回実行し平均値を求めた．表 2 に結果を示す．

表 1　局面の判定

	消費時間 [ms]	[局面/s]
CPU	37.9	26.4
GPU	31.7	31.5

表 2　ランダム対戦

	消費時間 [ms]	[対戦/s]
CPU	661	1.51
GPU	565	1.77

表 3　GPU での計算時間の詳細

処理内容	消費時間
CPU から GPU への転送	46.3 [μs]
GPU での計算	31.6 [ms]
GPU から CPU への転送	38.6 [μs]

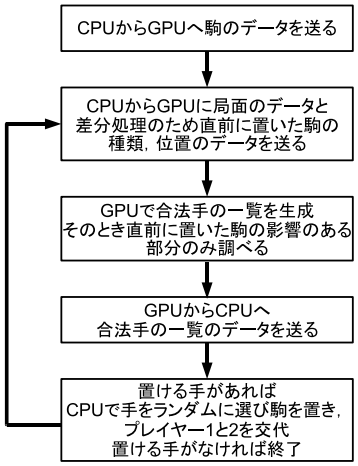

図 5　GPU を使用したランダム対戦の流れ

GPU では CPU に比べ 1.2 倍高速に実行でき，合法手生成における GPU の有効性が確認された．また，GPU での計算時間の 1,000 回平均とメモリ転送の時間の 100 万回平均をそれぞれ調べた結果を表 3 に示す．今回の測定では計算時間に比べメモリ転送の時間は少ないという結果となった．

5　結論と今後の課題

Blokus Duo の合法手生成を GPGPU を用いることで CPU の 1.2 倍程度高速化でき，ゲームプレイヤにおいて GPGPU の有効性を示すことができた．

今回は Blokus Duo の合法手生成という限られた部分を並列化しただけであったが，今後はそのほかの各要素が依存性が少ない分野の処理も高速化できると考えている．まず考えられるのがモンテカルロ法である．これはランダム対戦を多数行う必要があり，それぞれが独立しているため並列化できる．評価関数も独立した評価要素の重ね合わせである場合，GPU により高速化できると考えられる．

参考文献

[1] 新庄貞昭，高橋誠史，木村秀敬，白井暁彦，宮田一乗，『GPU の先駆的利用の研究動向と将来像』, 芸術科学会論文誌 Vol.6 No.3 pp.167-178，2007

[2] Daniel Cederman, Philippas Tsigas, “A Practical Quicksort Algorithm for Graphics Processors”, 2008

[3] Cuda Zone, http://www.nvidia.co.jp/object/cuda_home_jp.html