【7】

氏名	奥山倫弘（おくやまともひろ）
博士の専攻分野の名称	博士（情報科学）
学位記番号	第 25844 号
学位授与年月日	平成25年3月25日
学位授与の要件	学位規則第4条第1項該当 情報科学研究科コンピュータサイエンス専攻
学位論文名	Accelerating GPU Programs by Reducing Irregular Control Flow and Memory Access （不規則な制御フローおよびメモリアクセスの削減によるGPUプログラムの高速化）
論文審査委員	（主査） 教授 萩原 兼一 （副査） 教授 増澤 利光　准教授 伊野 文彦

論文内容の要旨

The graphics processing unit (GPU) is recently used as a massively parallel processor to speed up general computation. However, the GPU can decrease the performance of irregular computation, because the GPU is based on the single instruction, multiple data (SIMD) architecture. The irregular computations here are conditional branches and memory accesses, which vary the behavior of threads depending on the input data. In particular, different control flow between threads causes redundant computations to follow each control flow. Moreover, uncoalesced memory accesses waste the memory bandwidth of the GPU. Therefore, there are many challenges to accelerate applications that depend on irregular computation.

This thesis presents GPU-based acceleration methods for three applications, aiming at developing techniques to efficiently process irregular computation on the GPU. We focus on irregular GPU programs that have similar threads in the entire program, although naive parallelization methods fail to exploit the similarity of threads. Our main approach is to gather similar threads for the SIMD operations before executing threads on the GPU. We achieve this preprocessing by observing the similarity of memory access pattern for the first application. For the third application, we use the similarity of operations that are executed by threads. For the second application, we evaluate another approach, which employs an algorithm that eliminates the irregularity by using a regular data structure instead of a pointer-based data structure. The details are described below.

First, we describe an acceleration method for finding the all-pairs shortest paths (APSPs) using the GPU. The APSP problem is a graph operation that finds shortest paths between all two vertices in a graph. This computation requires many uncoalesced memory accesses to refer to the graph data, while the memory bandwidth bounds the performance. Our method is based on an iterative algorithm that repeatedly solves the single-source shortest path (SSSP) problem in parallel on the GPU. We exploit the coarse-grained parallelism by using a task parallelization scheme that associates a task with an SSSP problem, in addition to the fine-grained parallelism in an SSSP problem. This scheme solves multiple SSSP problems at a time, allowing us to share the graph data on a fast on-chip memory, as well as reducing irregular memory accesses. As a result, the speedup over the existing SSSP-based implementation ranges from a factor of 2.8 to that of 13, depending on the graph topology.

We next present acceleration methods for the Floyd-Warshall (FW) algorithm using the GPU, which is another algorithm to solve the APSP problem. This algorithm uses a matrix representation of a graph, which eliminates irregular control flow and memory accesses. The proposed method contains two variations, both designed to reduce data access to off-chip memory based on an iterative blocked FW (BFW) algorithm. The first method also reduces the number of instructions using registers rather than the shared memory. The other method increases the block size because it is inversely proportional to the amount of off-chip memory access. For graphs with 256 1024 vertices, both methods are 4% faster than an existing recursive BFW method. The first method achieves approximately 70% of peak computational performance.

Finally, we demonstrate a GPU-based general biophysical simulator, called Flint. With this application, the program for threads depends on the input data, as well as the data values. Therefore, it is required to reduce the difference of control flow between threads. Flint handles heterogeneous biophysical models described by a large set of ordinary differential equations (ODEs). It uses an internal bytecode representation of simulation-related expressions to handle various biophysical models built for general purposes. The interpretation of bytecodes causes a heavy use of conditional branches. To reduce the irregular branches, we preprocess the bytecodes, which groups the similar bytecodes to assign a bytecode group to a SIMD core of the GPU. In addition, each group is unified to a unified bytecode to reduce memory accesses. We then implement two acceleration methods for Flint using a GPU. The first method interprets multiple bytecodes in parallel on the GPU. The second method translates a model into a source code through the internal bytecode, which speeds up the compilation of the generated source codes, because the code size is diminished by the bytecode unification. The first method simulates a model containing approximately 40,000 expressions 24 times faster than that on a CPU core. The second method achieves a performance of 2.4 times higher than that of the former method.

These results show that the GPU can be used for accelerating applications that include irregular computation. In particular, the task parallel scheme used for the APSP problem can improve the throughput of computation that includes the same type of independent subproblems. The technique used for our biophysical simulator will be applied to other ODE-based simulations. Moreover, it can be applied to an application that assigns different operations to threads. These findings will contribute to the realization of a general technique for efficient processing of irregular computation on the GPU and other accelerators.

論文審査の結果の要旨

本研究は、GPU（Graphics Processing Unit）上で不規則な計算を効率よく実行する手法に関するものである。グラフィクス処理用のアクセラレータであるGPUを、汎用計算の高速化に応用する研究が並列処理分野で注目されている。しかしながら、GPUはSIMD（Single Instruction, Multiple Data）型の並列計算器であることから、不規則な制御の流れやメモリ参照に対しては実効性能が低下する問題がある。本研究はこの問題に関して、3つの応用に対するGPUを用いた高速化手法を研究している。

第1の応用は、グラフ問題の一つである、全点対最短経路（APSP：All-pairs Shortest Path）問題を扱っている。この応用はグラフを表す情報を繰り返し参照することから、不規則なメモリ参照が多発し、メモリ帯域幅が性能ボトルネックである。そこで、メモリ参照パターンの類似性に着目し、不規則なメモリ参照を削減する。具体的には、GPU上で単一始点最短経路（SSSP：Single-source Shortest Path）問題を繰り返し解く既存手法を基に、複数のSSSP問題を同時並行に解くタスク並列処理を提案する。タスク間の共通データを高速な共有メモリに格納し、低速なグローバルメモリの参照を削減すると共に、タスク間の処理の類似性を用いて不規則なメモリ参照を削減する。その結果、既存手法に比べ、2.8倍から13倍の高速化した。

第2の応用では、規則的なデータ構造を用いて不規則性を排除するアプローチを検証している。APSP問題の解法の一つである、フロイド・ワーシャル（Floyd-Warshall）法は、グラフを表す情報を規則的な行列形式のデータとして扱うことから、不規則な処理を排除できる。提案手法は、高速な行列積実装を応用する手法および2段階のブロック化を用いる手法を含む。前者はレジスタを活用し、命令数を削減する。後者は、共有メモリ上のデータの再利用効率向上を図る。256～1024頂点のグラフに対し、両手法は既存手法よりも最大4%高速であり、1つめの手法はピーク演算性能の約70%と高い性能を実現した。

第3の応用では、スレッドが実行する操作の類似性に着目する。この研究では、Flintと呼ばれる汎用生体シミュレータを対象としている。Flintはさまざまな生体モデルを扱うために、生体モデル中の各数式をバイトコードで表現し、それらを解釈実行してシミュレーションを実行する。GPUを用いた実装では、各スレッドに1つのバイトコードを割り当てる。スレッド間での不規則な分岐を削減するために、類似したバイトコードを一連のスレッドに割り当てる。さらに、類似したバイトコードを統合し、メモリ参照量を削減する。実行方式として、バイトコードをGPU上で解釈実行するインタプリタ方式に加え、バイトコードをソースコードに変換・コンパイルするトランスレータ方式を評価した。結果、約40,000の式を含む生体モデルにおいて、インタプリタ方式はCPUに比べて24倍高速であった。トランスレータ方式はインタプリタ方式よりも最大2.4倍高速であった。

これらの研究成果は、並列処理分野において有用なものであり、汎用性をもつものである。よって、博士（情報科学）の学位論文として価値のあるものと認める。