CUDA を用いたパラメタスイープアプリケーションの並列化手法の評価

重岡 謙太朗 [1]　奥山 倫弘 [2]　伊野 文彦 [2]　萩原 兼一 [2]
[1] 大阪大学基礎工学部情報科学科　[2] 大阪大学大学院情報科学研究科

1　はじめに

パラメタスイープアプリケーション（PSA）を，GPU（Graphics Processing Unit）向けの汎用計算環境 CUDA（Compute Unified Device Architecture）により高速化する研究がある [1]．PSA は，異なる入力（パラメタ）に対して同一プログラムを繰り返し実行する．以降，1 つのパラメタに対する処理をタスクと呼ぶ．

PSA を高速化するためには，2 種類の並列性を利用できる．タスク並列性を利用する場合，複数のタスクを各コアに割り当て並列処理する．一方，データ並列性を利用する場合，単一タスク内のデータを各コアに割り当て並列処理する．既存の並列化手法 [1] はタスク並列性に着目している．しかし，どのような性質をもつ PSA がタスク並列性により高速化できるのかは明らかでない．

そこで本研究では，メモリ参照パターンに着目し，タスク並列性と親和性の高い PSA を明らかにすることを目指す．そのために，メモリ参照のランダム性，再現性，およびパラメタ並び替えのコストに関して，2 種類の PSA を評価する．

2　既存の並列化手法

k $(1 \leq k \leq n)$ 番目のパラメタを処理するために必要な入力を I_k，出力を O_k とする．PSA では，入力 I_1〜I_n の各々に対して同一プログラムを実行し，出力データ O_1〜O_n を得る．既存手法 [1] では，グローバルメモリへの参照を削減するために，以下の 2 つの工夫を用いる．

1. b 個のパラメタをインタリーブ状に並び替え，それらを並列処理する．例えば，k 番目の入力データの集合を $I_k = \{i_{k,1}, i_{k,2}, \cdots, i_{k,m}\}$ とすると，$i_{k,1}, i_{k+1,1}, \cdots, i_{k+b-1,1}$ が連続アドレスに格納されるように並び替える．k〜$k+b-1$ 番目のパラメタを処理するスレッドが上記アドレスを参照するとき，coalesced 参照を実現できる．

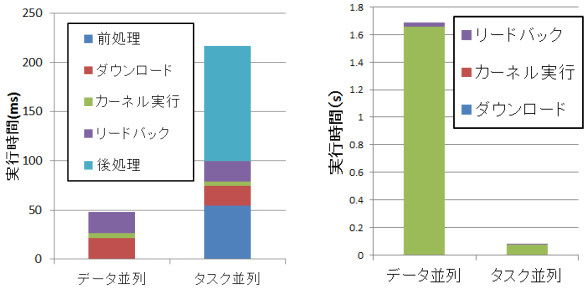
2. 複数のパラメタにおいて，共通して参照するデータ集合 D を共有メモリに格納し再利用する．データ集合 D を 1 度のみ共有メモリにコピーし，グローバルメモリではなく共有メモリを用いて参照する．b 個のパラメタを並列処理するとき，データ集合 D のグローバルメモリ参照を $1/b$ に削減できる．

3　タスク並列性に向く PS アプリケーション

上記の工夫は，PSA のメモリ参照パターンの違いにより，高速化の効果に違いが生じると考えた．そこで，メモリ参照パターンにおける 3 つの特徴に着目する．

(a)　ランダム性：単一パラメタにおいて，データ参照が不規則であること
(b)　再現性：各パラメタ処理のデータ参照パターンが入力に依存せず同一であること
(c)　パラメタ並び替えのオーバヘッドが小さいこと

ランダム性をもつ PSA は，データ並列性において coalesced 参照には不向きである．一方，工夫 1. を用いたように，タスク並列性は coalesced 参照に向いている．よって，タスク並列性に向いている．再現性をもつアプリケーションは，入力の値に依存せず各タスクのメモリ参照パターンが同一となる．



(a) 畳み込み処理　(b) 全点最短経路探索

図 1: PSA の実行時間

それによりメモリ参照が不規則とならず，メモリ参照のシリアライズやダイバージェントブランチを発生させずに，工夫 1.，2. により高速化できる． さらに，工夫 1. はパラメタの並び替えが必要となる．したがって，並び替えるデータ量が多いものはタスク並列性に適さない．

4　実験

実験を用いて，着目した 3 つの特徴と，PSA のタスク並列性による高速化の関係を確認する．畳み込み処理と全点最短経路探索（APSP）[2] について，データ並列性とタスク並列性を利用した実行時間を比較した．

畳み込み処理は，再現性はあるがランダム性はない．このため，工夫 1. の coalesced 参照の効果は小さい．さらに，このプログラムではデータ集合 D はない．これにより，工夫 2. の共有メモリの効果も小さい．データ並列とタスク並列のカーネル実行時間の変化はなかった．さらに，各入力が画像ファイルであり，データ集合 D は存在しないため，並び替えのコストは大きい．

入力は画像ファイルであり，各タスクで独立した情報となっている．したがって，タスク間で共有できないがゆえ，並び替えのコストが大きい．

並び替え実行時間はカーネル実行時間の 37 倍である．したがって，畳み込み処理はタスク並列に適さない（図 1(a)）．

APSP は，ランダム性はあるが再現性がない．このため，ダイバージェントブランチが発生してしまうが，工夫 1.,2. を用いて高速化できる．さらに，各入力は同一グラフデータを参照する．このため，データ集合 D が多い．したがって，パラメタ並び替えのコストは発生しない（図 1(b)）．以上の特徴から，APSP はタスク並列に適している．

実験結果より，タスク並列に適した PSA には上記 3 つの性質が必要であることを確認した．今後の課題は，3 つの全ての特徴をもつアプリケーションについて評価し，この特徴がタスク並列に適するための条件となっているかを検討したい．

参考文献

[1] Masaya Motokubota, Fumihiko Ino, and Kenichi Hagihara. Accelerating parameter sweep applications using CUDA. In *Proc. PDP'11*, pp. 111–118, February 2011.
[2] Tomohiro Okuyama, Fumihiko Ino, and Kenichi Hagihara. A task parallel algorithm for computing the costs of all-pairs shortest paths on the CUDA-compatible GPU. In *Proc. ISPA'08*, pp. 284–291, December 2008.