GPU において動的グラフを高速処理するためのフレームワークの検討

三谷康晃 [1]　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

Pregel[1] は，大規模グラフを高速処理するための並列分散フレームワークである．Pregel では頂点ごとの独立な処理を記述し，グラフ上のすべての頂点を並列処理できるため，クラスタ環境や GPU などの並列実行環境において高い性能を期待できる．そのため，近年では，同様の処理体系を持つフレームワークの実装が幅広く登場しており，GPU 上の実装にとして Medusa[2] が存在する．

しかし，我々の知る限り，計算時における頂点や辺の追加や削除が可能な GPU 実装は存在しない．この操作はクラスタリングアルゴリズムや時間的な要素により動的に変化するグラフ（道路ネットワークやソーシャルグラフ）上の問題を処理するために必要である．Medusa では，グラフのトポロジをもとにメッセージを管理するバッファを構築しているため，辺や頂点が追加されるたびにデータ構造の再構築が必要となり，計算効率が低下する可能性がある．

本研究では，計算時における頂点や辺の動的な変更機能を持つ Pregel の GPU 上での実装を実現することにより，上の問題に対して GPU による高速化を行う手段を提供することを目的としている．現段階ではその準備として，単一の GPU 上において，辺や頂点が変化してもデータ構造の再構築を必要とせず，かつ，効率的にメッセージを管理できるデータ構造と，その実現のために，今回の問題に特化したメモリアロケータを提案する．

2　Pregel のモデル

Pregel の入力は有向グラフであり，各頂点と辺にはユーザが定義した値が関連付けられている．Pregel ではスーパーステップと呼ばれる処理単位で，すべての頂点に対して 1 つ処理を並列実行した後，同期するという操作を繰り返すことによって処理を実行する．各スーパーステップにおける頂点の処理は，関数としてユーザが定義する．ユーザが定義する関数は，他の頂点に対してのメッセージの送信（送信したメッセージは直後のスーパーステップでのみ参照される），直前のスーパーステップで自身宛に送信されたメッセージの参照，自身や自身から出ている辺に関連付けられた値の変更および新たな頂点や辺の追加リクエストの送信などの操作を行える．

3　提案手法

提案手法は，各頂点が受信したメッセージのリスト構造を保持することによりメッセージを管理する．メッセージ送信時には，各頂点が送信先の頂点のメッセージのリストに対して直接メッセージを挿入する．挿入時には，アトミック演算を用いることにより，競合状態を回避できる．図 1 の例では，スーパーステップ n において，頂点 1 および 3 が頂点 2 に対して，それぞれメッセージ X および Y を送信している．頂点 1 および 3 は各々の送信メッセージを，スーパーステップ $(n+1)$ において頂点 2 が参照するリストに対して挿入している．

次に，このリスト構造の実現には，動的なメモリ確保が必要になる．提案手法では，Hong ら [3] のメモリアロケータを 2 つ用意して交互に利用することで高速なメモリ確保を実現している．まず，複数のメモリアロケータを用いる理由としては，リスト構造は一度確保したメモリの再利用が難しいという問題がある．リスト構造を再利用する場合，挿入時には使用されていない領域を先頭から順番に探索しなければならず，リストのデータ数が増加した場合，挿入する箇所を探す計算量が大きくなる．更に，メッセージとしては扱われるデータはユーザが定義するため，常にアトミック演算が使えるとは限らず，ロック無しで競合せずにデータを書き込むのは難しい．しかし，領域の再利用を行わない場合，計算全体を通して送信されるメッセージの領域をすべて確保しておかなければならず現実的でない．今回の場合，2 つ前のスーパーステップに送信されたメッセージ領域は不要になるため，Hong らのメモリアロケータを 2 つ用いて，偶数番目と奇数番目のスーパーステップ，それぞれのメッセージ領域のメモリ確保に用いることでメモリ領域を再利用することが可能になる．

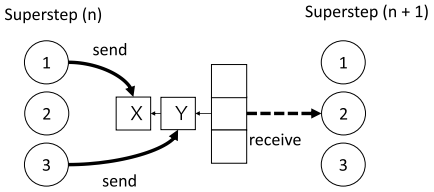

図 1: メッセージ管理の仕組み

4　評価実験

提案手法の性能を評価するために，提案手法を含むフレームワークを用いて，PageRank を実装した際の実行時間に関して単一 CPU における PageRank の実装と比較した．なお，PageRank の反復回数は 30 回である．実験に使用した GPU は NVIDIA GeForce GTX 680 であり，CUDA のバージョンは 5.5 である．グラフデータとして roadNet-CA および WikiTalk(`http://snap.stanford.edu/data/`) を使用した．提案手法の roadNet-CA に対する実行時間は 496 ミリ秒となり，CPU 実装の 1616 ミリ秒の約 0.3 倍となった．一方で，WikiTalk に対する実行時間は 4457 ミリ秒となり，CPU 実装の 1889 ミリ秒の約 2.4 倍となった．WikiTalk に対する結果が CPU よりも低速だった理由としては WikiTalk のトポロジが 1 つの頂点に辺が集中した形状をしており，頂点ごとに並列化を行っている提案手法では低い並列度が低いためであると思われる．

今後の課題は，偏ったトポロジに関しても高い並列度を得られるような工夫をすることと，動的なグラフ処理に必要な頂点や辺の追加や削除等の機能を実現していくことである．

参考文献

[1] Grzegorz Malewicz, at el., Pregel: a system for large-scale graph processing, SIGMOD'10, pp.135–146, 2010.
[2] Jianlong Zhong and Bingsheng He, Medusa: Simplified graph processing on GPUs, Trans. Parallel and Distributed System, to appear.
[3] Chuntao Hong, at el., MapCG: Writing parallel program portable between CPU and GPU, PACT '10, pp.217–226, 2010.