6M-6

グラフ分割を用いた 大規模2部グラフのデータストリーム処理

雁瀬　優†　　鈴村豊太郎‡

東京工業大学工学部情報工学科†　　東京工業大学/IBM 東京基礎研究所‡

1.研究背景

近年、スーパーコンピュータの評価指標として Graph500[1]が策定されるなど、大規模グラフの処理が注目を集めてきている.大規模グラフ処理が必要とされる分野の中には、株式市場やマーケットサイトのリコメンデーションシステムなど、リアルタイムな処理が要求される分野も多数存在しており、この論文では、大規模グラフに対しデータストリーム処理を適応する際の問題点と解決策を提示する.

2.問題提起

2.1 データストリーム処理と System S

データストリーム処理とは、始点･終点という概念のない情報の列をストリームと呼び、このストリームを蓄積することなくオンメモリ上で逐次的にパイプライン処理していく計算パラダイムである.データストリーム処理では様々な処理が抽象化・汎用化されており、幅広い処理に対して応用できる洗練された処理系としてまとめられている点が従来の音声や動画のストリーミングなどとは異なっている.このような処理系の一つとして開発されている System S [2]ではデータフロー図から直感的に処理を記述できる SPADE という言語が用意されており、様々なデータストリーム処理を実現している.SPADE のコード例やオペレータの詳細については論文[2]に詳細が記載されているので省略するが、SPADE は簡易な記述と高い柔軟性を兼ね備えており効率的なシステム開発が可能となっている.

2.2 データストリーム処理とグラフ処理

大規模グラフ処理の研究はこれまでも盛んに行われてきているが、これまでの大規模グラフ処理の研究は MapReduce を基盤にした PEGASUS[3]など、バッチ処理を中心に研究が行われている.データストリーム処理で解析する先行研究は[4]などがあるが、未だ発展途上の段階にある. 一方で処理すべきグラフの情報量は増加の一途を辿っており、リアルタイム処理を可能にするには抜本的な処理方法の改善が必要不可欠である.本論文では時系列上で変化が激しく、リアルタイムに逐次処理することに困難がある大規模 2 部グラフに対し、データストリーム処理を行うことを目的としている.データストリーム処理という観点からみた場合、処理するグラフ要素の増加に伴う各グラフ処理の計算量増加に加え、処理する回数の増加も考慮しなくてはならない.

2.3 既存のグラフ処理の問題点

大規模 2 部グラフの解析処理の例としては H. Tong[5]が提唱する“any-time-answer”を目的とする 2 部グラフ解析アルゴリズム Fast-Single-Update (FSU) が存在する.FSU は、グラフのランダムウォーク法 Random Walk with Restart (RWR 法) を用いた 2 部グラフ間の関連性解析アルゴリズムを、グラフの差分更新を行うことにより、毎回すべての要素を計算し直す手法と比較して計算量を減らす手法である.これは大規模 2 部グラフのバッチ的な処理を実時間内に行うことを可能にするという点では非常に大きな意味を持つが、グラフサイズが増大すると各処理の計算量も増加する欠点があり、これをデータストリーム処理という観点から見ると、ストリーム上を流れるエッジ情報の到着頻度内に処理を終えることができないとリアルタイムに処理できなくなってしまうため問題となる.

3.本研究の提案手法と実装

ソーシャルネットワークに代表される大規模グラフの持つ性質の一つとして、コミュニティ構造[6]という特徴が存在している.コミュニティ構造とはエッジが密な集合同士が疎なエッジで繋がっている構造であり、分割して計算してもある程度の処理精度を保つことが知られている.本研究ではその性質を利用した高速解析手法を提案し、実装と評価を行った. 本手法ではグラフ処理要求の頻度とグラフのサイズから、必要とされる処理速度を達成するグラフ分割数を適切に選び、各グラフに並列分散処理させることによって、リアルタイム処理を実現する.実装にはデータストリーム処理系には System S を、グラフ分割には METIS[7]を、2 部グラフの関連性を求めるアルゴリズムとして FSU を用いた.対

Data Stream Processing for Large-Scale Bipartite Graph using Graph Partition

†Masaru Ganse, Information Science and Engineering, Tokyo Institute of Technology

‡ Toyotaro Suzumura, Tokyo Institute of Technology/IBM Research-Tokyo

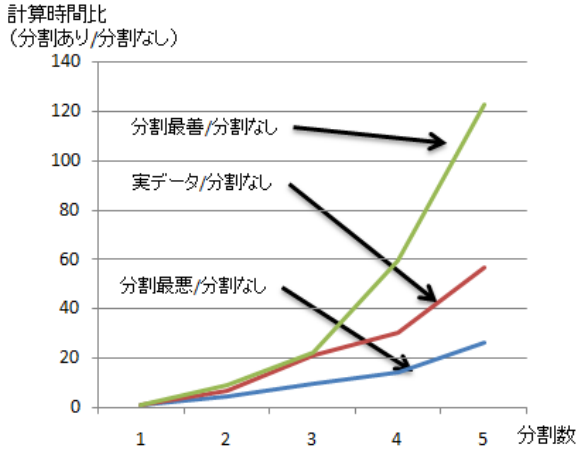

図 1：グラフ分割による計算量高速化比率

象とするグラフは 2 部グラフで、頂点の動的な変化は考えず、分割数の決定は静的に行った.

4. 実験と評価

4.1 評価対象のアプリケーション

評価対象として匿名掲示板のスレッド間の関係性解析を選択した.匿名掲示板には一つのテーマの集合体“板”（例:ニュース板）とテーマの中のトピック“スレッド”(例:事件 A について)、書き込みである“レスポンス”が存在し、ユーザは ID によってある程度の期間（通常一日）自己の同一性が保証される.アプリケーション選択の基準として、匿名掲示板は時系列データ特有のプライバシー侵害の問題を考慮する必要がなく、また、一つのスレッドが終了した場合次のスレッド（例:事件 A について 2）に移る必要があるため時系列上のスレッド間の関係が明確であるという 2 点があげられる.2 部グラフの頂点群としてユーザ ID とスレッドを、エッジとして“レスポンス”を用いた.グラフとしての性質は“板”の性質に依存するが、今回用いたグラフは 2 日間でユーザ数 25532、スレッド数 492、エッジ数 85656 で、グラフの更新要求は平均すると約 2 秒に一回ある計算となる.

4.2 実験環境

測定にはエッジ情報を送信するサーバとして1ノード（Dual Socket Opteron 242 1.6GHz,Memory 8GB ）、計算サーバとして4ノード（Phenom 9850 2.5GHz, Memory 8GB）を用い、各ノードは1GbE のネットワークで接続されている. ソフトウェアは全ノード共通で、OS は CentOS 5.2、gcc 4.1.2、InfoSphere Streams 1.2.0（System S）を用いた.

4.3 評価

グラフ分割により、計算時間は図 1 のように変化し、グラフ分割数 5 においての計算時間は 1 処理 0.007 秒となり、分割数なしの場合の計算

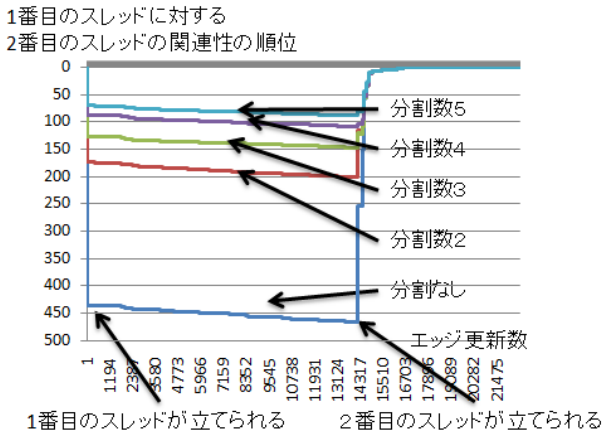

図 2：スレッド間の関連性のグラフ分割数における変化

時間 0.40 秒と比較して 57 倍の高速化を実現した.高速化比率はグラフ構造によって一意ではないが、“グラフ処理の要求がひとつのグラフに集中したため、計算量は減少したが処理頻度は減少しなかった場合（分割最悪）”、“グラフ処理の要求が均一に振り分けられ、計算量と処理頻度の両者が最も減少した場合（分割最善）”の計算量を考えることにより、ある程度の予測が立てられる.一方で、図 2 では一つのスレッドの書き込みが限界に達し次のスレッドに移る際の関係性推定にかかる処理回数をグラフ分割数ごとに測定しているが、グラフ分散による処理回数には変化がなく、2 部グラフの関連性推定にはグラフの特定部分のみが関係していることが明らかになった.

5. まとめと今後の展望

グラフ分割を行い、推定までの処理回数を増やすことなく、分割前と比較して処理時間を 57 倍減らすことに成功した.今後の展望としては、動的な分散数の決定や、動的な頂点の追加に対応できるシステムの構築が求められる.

参考文献

[1] http://www.graph500.org/
[2] 松浦紘也, 雁瀬優, 鈴村豊太郎. データストリーム処理系 System S と Hadoop の統合実行環境. ComSys 2010
[3] U.Kang, et.al. PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations. ICDM 2009
[4] Aggarwal C.C., Online Analysis of Community Evolution in Data Streams, SDM 2005
[5] H. Tong, et.al. Proximity Tracking on Time-Evolving Bipartite Graphs. SDM 2008.
[6] M.E.J Newman. Fast Algorithm for Detecting Community Structure in Networks. Phys. Rev 2004.
[7] G. Karypis, et.al. Multilevel k-way partitioning scheme for irregular graphs. JPDC 1998.