データストリーム処理によるリアルタイム性を考慮した大規模グラフ処理基盤とスペクトラルクラスタリングへの応用

西井 俊介† 鈴村 豊太郎†‡

大規模かつ成長し続けるグラフ構造を持つデータを，リアルタイム性を考慮して効率よく解析するために，データストリーム処理を用いたインクリメンタルグラフ処理を提案する．そのための計算モデル Incremental GIM-V の提案，およびデータストリーム処理系 IBM System S を用いた実装と PageRank を対象とした評価を我々は行った．ここで，汎用的なグラフ処理モデルおよびシステムにおいて，実用性，応用性は重要な課題である．グラフ分割はソーシャルクラスタリング等に有用であるため，重要なグラフ処理アプリケーションの１つである．そこで本稿では，グラフ分割アルゴリズム Spectral Clustering を対象とした Incremental GIM-V システムの拡張実装および評価を行い，データストリーム処理で実用するうえで求められる性質等について議論する．

Real-time Large-scale Graph Processing System by Data Stream Processing and its Application to Spectral Clustering

SHUNSUKE NISHII† and TOYOTARO SUZUMURA†‡

We have proposed incremental graph processing using data stream processing, for effective analysis of large graph that evolves. We have proposed computation model "Incremental GIM-V" for this analysis, implement graph processing system using IBM System S, one of Data Stream Management System (DSMS), and evaluated its performance for PageRank. In a general graph processing models and systems, practicality and applicability is an important issue. Graph clustering is one of the important applications of graph processing, because it is useful for such as social clustering. In this paper, we extend and evaluate our system for graph clustering algorithm "Spectral Clustering". Then we discuss features required for data stream graph processing.

1. 導入

近年，大規模データ処理の需要は高まりつつある．その対象となるデータとして，グラフ構造を持つデータが含まれる．その代表的なものとして，ウェブグラフや，Twitter や Facebook などのソーシャルグラフが挙げられる．そのような Web 利用マイニングや Web 構造マイニング[17,18]を対象とした研究は多くなされている[2,3]が，これらはバッチ処理に基づくものである．バッチ処理では，一旦データを蓄積してから解析を始める．これにより確実な解析結果は得られるものの，源データが生起されてから解析結果を得るまでに期間があり，解析の新鮮さが失われてしまう．すなわち，ソーシャルグラフなどから得られる，瞬間的な話題性を反映した解析を行うには向いていないと言える．そこで，我々はデータストリーム処理[4,5,6]を用いたグラフ処理を提案した[16]．その中でグラフ処理モデル Incremental GIM-V を提案し，システムの実装及び PageRank[1]を対象とした評価を行った．

汎用グラフ処理モデルや処理システムでは，アプリケーションに対する適用性を明確にすることが求められる．Incremental GIM-V においては，PageRank の評価は行われている．Random Walk with Restart[9]や最短経路問題[8]を対象とした，Adaptive PageRank[14]に基づく最適化を施した実装および評価を現在我々は行っている．本稿では，それらとは異なるアプリケーションとして Spectral Clustering[20]を対象として扱う．Spectral Clustering はグラフ構造をもとに「スペクトル」を計算し，その値を元にグラフ分割を行う．K-means や Fuzzy c-means と組み合わせることによって，ハードクラスタリングにもソフトクラスタリングにも用いることが出来る．そのため，大規模かつ成長するグラフを対象とする場合，ソーシャルクラスタリングや，グラフ分割による並列グラフ処理のための前処理[19]などに適用できる．しかし，Spectral Clustering は GIM-V のような簡略化された反復計算モデルでは実装が困難である．そこで本稿では，我々の Incremental GIM-V システムを Spectral Clustering を実行できるように拡張し，評価を行う．その結果から，Spectral Clustering をデータストリーム処理で行う上での議論を行う．

以降の節では，まず第２節で関連研究について述べる．本稿で提案するデータストリームグラフ処理の背景として，第３節でデータストリーム処理，第４節でグラフ処理モデル GIM-V と Incremental PageRank の概要について述べ，第５節でインクリメンタルグラフ処理モデル Incremental GIM-V

† 東京工業大学 情報理工学研究科 計算工学専攻
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
‡ IBM 東京基礎研究所
IBM Research - Tokyo

© 2012 Information Processing Society of Japan

の概要と実装について述べる．その後，第 6 節で Spectral Clustering の概説を行い，第 7 節で Incremental GIM-V を Spectral Clustering[20] に適用できるように拡張した実装，第 8 節で評価について述べる．最後に，第 9 節で議論を行い，第 10 節で本稿のまとめを述べる．

2. 関連研究

大規模グラフ処理を行うシステムに関する研究として，Google の Pregel[2]やカーネギーメロン大学の PEGASUS[3]などが挙げられる．これらはバッチ処理に基づくグラフ処理システムであり，大規模なデータを短時間で処理するために複数の計算ノードを用いて並列分散処理を行うことができる．Pregel では，頂点ごとの処理とメッセージパッシングにより反復計算を行い，最終的な解析結果を得る．様々なグラフ処理を実装できるように，C++ API の形でインターフェースを提供している．それにより，PageRank[1]，最短路問題[8]，2 部マッチング[9]，セミクラスタリングなどのアプリケーションが実装されている．また，計算途中で故障が生じた際のために，チェックポイントを作成し，そこから復帰できるなど耐故障性を考慮した実装もなされている．計算性能としては，Google 社内の 300 コアからなるクラスタ上で，10 億頂点，1271 億辺からなるグラフに対するシングルソースの最短路問題を，700~800 秒で解くことができる．PEGASUS は MapReduce[10]の処理系である Hadoop[11]を用いたオープンソースのグラフ処理システムである．PEGASUS では，PageRank，Random Walk with Restart (RWR)[12]，直径推定[13]，連結成分推定などの処理が実装されている．また，これらの処理に共通する計算の構造を，GIM-V (Generalized Iterative Matrix-Vector multiplication) という，行列とベクトルの乗算(べき乗法)を一般化したものとして抽出し，その構造に基づいて計算の最適化を行っている．計算性能としては，Yahoo! の M45 Hadoop クラスタを含む 90 台の計算機を用いて，5.9 万頂点，2.82 億辺からなるグラフに対する PageRank の計算を 50~100 秒で解くことができる．Pregel, PEGASUS ともに，計算時間はデータの大きさ(辺の数)や計算機の台数の逆数に線形にスケールする．

Spectral Clustering の他にもグラフクラスタリングアルゴリズムの研究は多くなされている．データのクラスタリングには，全てのデータをそれぞれ丁度 1 つのみのコミュニティにクラスタする「ハードクラスタリング」と，0 個以上のいくつかのコミュニティにクラスタされうる「ソフトクラスタリング」に分かれる．グラフのクラスタリングの例として，Girvan-Newman アルゴリズム[24]や，Newman アルゴリズム[25]などがあり，これらはハードクラスタリングの手法である．Spectral Clustering もハードクラスタリングの手法であるが，Spectral Clustering ではスペクトル計算時の固有ベクトルのリストをもとに K-means[26]や Fuzzy c-means[27]を利用することで，ハードクラスタリングにもソフトクラスタリングにも応用することが出来る．

3. 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 [5,6,15]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[15]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．

4. グラフ処理モデル GIM-V と Incremental PageRank の概要

インクリメンタルグラフ処理モデル Incremental GIM-V の基となる，GIM-V(PEGASUS)[3]，および Incremental PageRank[7]について説明する．

4.1 GIM-V

GIM-V (Generalized Iterative Matrix-Vector multiplication)[3] とは，行列とベクトルの乗算を一般化したものである．n×n 行列 M と n 次元ベクトル v の乗算を行い，ベクトル v' を得る計算

$$v' = M \times v$$

を，要素ごとの式で表わすと，

$$v'_i = \Sigma_{j=1}^{n} M_{ij} \times v_j$$

となる．この計算は，「乗算」「総和」「代入」の 3 つの演算からなる．これらを一般化し，3 つの演算をそれぞれ関数 combine2, combineAll, assign とおくと，式は以下の通りとなる．

$$v'_i = \mathrm{assign}(v_i, \mathrm{combineAll}(\mathrm{combine2}(M_{i1}, v_1), \ldots, \mathrm{combine2}(M_{in}, v_n)))$$

ここで，行列の要素の型(T_M)，ベクトルの要素の型(T_v)，combine2 の戻り値の型(T_c)は，実数型(float, double)やブーリアン型(bool)などのプリミティブだけでなく，配列や構造体などを取ることも可能である．それぞれの関数の写像を書くと，

combine2 : $T_M \times T_v \rightarrow T_c$
combineAll : T_c の配列 $\rightarrow T_v$
assign : $T_v \times T_v \rightarrow T_v$

となる．

グラフ処理としては，行列 M をグラフの隣接行列(各辺の値)，ベクトル v を各頂点の値として扱い，GIM-V の計算を反復的に行うことで，様々なグラフ処理を行うことができる．なお，M_{ij} の値は，j から i への辺の重みを表わしている．

PEGASUS では，GIM-V モデルの計算を MapReduce(Hadoop)上で実行している．この MapReduce 計算は 2 ステージの MapReduce からなる．まず，ステージ 1 の MapReduce で，グラフデータの入力(辺，頂点)を Map ジ

© 2012 Information Processing Society of Japan

ョブから行い，Reduce ジョブでは j をキーとして combine2() を実行して，その出力をステージ 2 の MapReduce の入力に与える．ステージ 2 では，Map ジョブはそのまま Reduce ジョブにデータを投入し，Reduce ジョブでは i をキーとして combineAll(), assign()を実行，収束した場合はそこで全計算を終了し，再度反復する場合はステージ 2 の MapReduce の出力をステージ 1 の MapReduce の入力に与える．

なお，PEGASUS[3]では，GIM-V を用いて，PageRank, Random Walk with Restart，直径推定，連結成分推定のアルゴリズムを実装している．

4.2 Incremental PageRank

Incremental PageRank Computation[7]とは，発展するグラフに対する高速な PageRank の計算方法の 1 つである．ある時点でのグラフ(G1)に対する PageRank の計算結果が得られているとき，その計算結果を利用して，更新されたグラフ(G2)の PageRank を計算する．Incremental PageRank では，まず G1 と G2 を比較して，「新たに追加された頂点，リンク構造の変化した頂点，およびそれらからリンクを辿って到達できる頂点」(V_Q)，「それ以外で，V_Qに辺を張っている頂点」(V_b)，「その他」(V_u)の 3 つの集合に分ける．ここで，集合 V_Qは「新たに追加された頂点，リンク構造の変化した頂点」を元に幅優先探索を行うことで求めることができる．これらの集合のうち，V_bとV_uは G1 の PageRank をほぼそのまま利用できるため，後述する「スケール」のみを行えばよい．V_Qに関しては，構造の変化により新たに PageRank を計算しなおす必要があるため，V_QとV_bの構造から PageRank の計算を行う．

G1 と G2 では，グラフに含まれる頂点の数が異なるため(n_{G1}, n_{G2})，構造が変化しない頂点(V_b, V_u)でも PageRank の変化が生じる．そのため，G1 で求まっていた PageRank の値を G2 に合う値に変換する必要がある．具体的には，G1 の PageRank の値に(n_{G1}/n_{G2})を乗算すればよい．この計算を「スケール」と呼ぶ．なお，新たに追加された頂点に対しては PageRank 計算の初期値として($1/n_{G2}$)を与えればよい．

このような計算方法を用いれば，G1 と G2 の間で V_Qに属するノードの割合(変化率)が小さければ小さいほど，PageRank の計算を高速に行うことができる．

5. インクリメンタルグラフ処理モデル Incremental GIM-V

5.1 略説

本稿で用いたデータストリーム処理によるグラフ処理のための計算モデルである，Incremental GIM-V について説明する．Incremental GIM-V の大筋の枠組みは Incremental PageRank と同様で，まず頂点を 3 つの集合(V_Q, V_b, V_u)に分ける．これらのうち，V_Qについては GIM-V の計算を再度行う必要があるとみなし，V_b, V_uについては不要だとみなす．なお，実際ここの 3 つの集合に分かれるのはある時点のグラフに対する GIM-V の計算結果が既に得られている場合のみであり，そうでない場合の最初の計算時には，すべての頂点が V_Qに属するものとする．この 3 つの集合の関係を図 1 に示す．

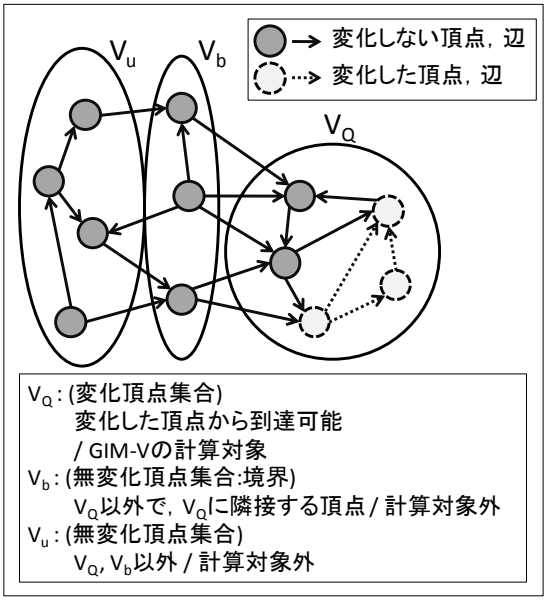

図1. グラフの計算内容による分類

その後，以前から存在した頂点の値に対して，必要ならばスケールを行い，新たに追加された頂点には計算の初期値を設定する．その後，V_QとV_bに対して GIM-V の計算を行う．この際，V_bに含まれる頂点には GIM-V の combine2 関数のみを実行し，combineAll, assign は省略する．

Incremental GIM-V で実装されるアルゴリズムは，GIM-V で実装可能であることが必要条件であるが，それに加えて以下のような性質を満たすことが望ましい．

- **変化点が生じた際，変化率(V_Qの割合)が大きくならないこと**：扱うデータが無向辺グラフの場合や，有向辺グラフではあるが変化が生じた際の変化率が大きくなる場合，計算内容に影響は生じないものの，インクリメンタルメソッドによる計算範囲の縮小による高速化が見込めない．
  この性質により，無向辺グラフを扱うアルゴリズムである Random Walk with Restart にはインクリメンタルメソッドは適さない．
- **反復計算により計算結果が徐々に収束し，アルゴリズムが反復回数を問わないこと**：例えば PEGASUS で実装される直径推定アルゴリズム[13]などは，求める直径の値=計算の反復回数であるため，反復回数を減らして計算時間を短くする手法が使えない．

5.2 実装

Incremental GIM-V の計算モデルを IBM System S 上で実行するために行った SPADE によるシステムの実装について説明する．

本システムでは，グラフの各頂点は序数(0, 1, 2, …)により ID を割り当てられる．また，グラフ処理は並列計算により行われる．各頂点の処理を担うプロセスのブロックは K 個(K は正の整数)に分けられ，図 2 中の W1~WK がそれに対応する．

ⓒ 2012 Information Processing Society of Japan

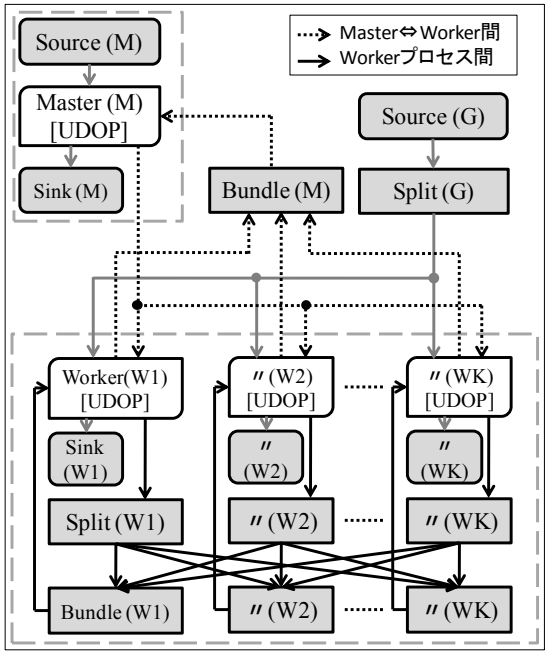

図2. グラフ処理システムのデータフロー図

頂点i (0, 1, 2, …)の処理を担うのは，(i mod K + 1)番目のブロックである．

5.2.1 データフロー図

図2にシステムのデータフロー図を示す．図中の各頂点はオペレータ(プロセス)を表し，矢印はデータの流れ(データストリーム)を表わす．実装に用いたオペレータは，Source(データの入力)，Sink(データの出力)，Split(データの分割)，Bundle(データストリームをまとめる)の4種類の組み込みオペレータと，Master(グラフ処理全体の管理)，Worker(グラフ処理の並列実行)の2種類のUDOP(ユーザ定義オペレータ)である．

図中の各オペレータの動作について説明する．なお，図中のKの値，Workerオペレータおよびそれに付随するオペレータ(Sink, Split, Bundle)の並列数を表わす．また，各オペレータの処理はそれに対応するプロセスが1つずつ立ち上げられ，並列実行される．これらのプロセスは複数の計算機間に渡って配置することが可能である．

- **Source(G)**：システムへグラフデータの入力を行う．(辺の値M_{ij}を設定する)
- **Split(G)**：グラフデータを担当するWorkerに渡す．(各M_{ij}の値をjで分類する)
- **Source(M)**：グラフ処理の開始命令を受け付ける．
- **Master(M)**：グラフ処理全体の管理，反復計算の同期の管理を行う．
- **Sink(M)**：グラフ処理に要した時間などのログ情報を出力する．
- **Worker(W1~WK)**：グラフ処理のメインとなる部分を実行する．a(=1, …, K) 番のオペレータは，a = (j mod K + 1)を満たす頂点jの情報(v_j, M_{ij} for ∀i)を保持する．
- **Sink(W1~WK)**：各Workerが担当する頂点に対するグラフ処理の計算結果を出力する．
- **Split(W1~WK), Bundle(W1~WK)**：Worker間のデータの流れを制御する．Worker(Wa)からWorker(Wb)へのデータの流れは，Split(Wa)とBundle(Wb)を介する．
- **Bundle(M)**：WorkerからMasterへのデータの流れを束ねる．

5.2.2 グラフ処理の流れ

グラフ処理の流れについて説明する．まず，グラフ処理の開始はSource(M)からMasterへのデータの入力により行われる．処理全体の流れとしては「待機」「V_Q導出」「V_b導出」「GIM-V計算」「結果出力」の5つのフェイズからなる．このうち「V_Q導出」と「GIM-V計算」の2つは反復計算を行うため，複数のステップからなる．Masterは，各フェイズ各ステップの開始を各Workerに伝え，Worker間の通信によりステップを実行し，実行が完了したことを(反復計算が収束したか否かも含めて)Masterに伝える．全てのWorkerからのステップの実行完了通知をMasterが受け取ったら，次のステップに進む．このような流れで計算を進めていく．グラフ処理の各フェイズの詳細は割愛する．

6. Spectral Clusteringへの拡張に向けて

本節では，Spectral Clustering, Incremental Spectral Clusteringの略説，およびSpectral ClusteringをIncremental GIM-Vシステム上で実行する上での問題点の説明を行う．

6.1 Spectral Clustering略説

Spectral Clustering[20] (以下，SC)とは，無向辺グラフ上の頂点の集合を，グラフ構造に基づいていくつかの集合に分類する手法(グラフ分割)の一つである．SCでは，まず入力として，グラフ構造を表わす類似度行列(Similarity Matrix; 以降，Wで表わす)が与えられる．Wはn×n行列(nは頂点数)であり，W_{ij}は頂点iから頂点jの辺の重みを表わす．W_{ij}は以下の要件を満たす．

- $0 \leqq W_{ij} \leqq 1$
- $W_{ii} = 1$
- $W_{ij} = W_{ji}$ (無向辺グラフなので)

この条件より，Wは対称行列である．なお，この行列は隣接行列を拡張したものと言える．そして，類似度行列を元に度数行列(Degree Matrix; 以降，Dで表わす)が定まる．Dはn×n行列であり，以下の要件を満たす．

- $D_{ii} = \Sigma_{j=1}^{n} W_{ij}$
- $D_{ij} = 0$ if $i \neq j$

この条件より，Dは対角行列である．

これらの行列をもとに，一般化固有値問題を解く．SCには，解くべき一般化固有値問題にいくつかの流儀が存在する．本研究ではShi Malikの流儀[21]を採用している．まず，ラプラシアン行列(Laplacian Matrix; 以降，Lで表わす)L = D - Wを求め，

ⓒ 2012 Information Processing Society of Japan

式 $Lq = \lambda Dq$ で表わされる一般化固有値問題(λ: 固有値, q: n 次元固有ベクトル)を解く．本研究では 2 部分割を対象としており，0 でない最小固有値 λ と，それに対応する固有ベクトル q を求める．頂点 i のクラスタリングは，q_i の正負で分割される．

6.2 Incremental Spectral Clustering 略説

Incremental Spectral Clustering[22](以下，ISC)とは，発展するグラフに対する SC の高速計算手法であり，近似解を求める．

SC の計算は大部分が行列計算であり，その計算時間は $O(|V|^3)$ ともなる．ISC では，λ, q を直接求めるのではなく，$\Delta\lambda$，Δq を，値の変化が大きい頂点の集合に限定した行列計算により，近似的に求める．近似手法であるため，ISC の実行を繰り返すにつれて誤差が蓄積されていく．それゆえに，定期的に完全な SC を再実行する必要がある．

ISC は，グラフ情報の更新(辺の追加・更新・削除)が 1 回発生するごとに実行される．更新された辺の頂点を i, j, 辺の重みを W_{ij} とする．更新された W_{ij} と元の W_{ij} の差分 ΔW_{ij} が求まる．そこから，行列 L と D の差分 ΔL, ΔD が求まる．$\Delta\lambda$ を

$$\Delta\lambda = \Delta W_{ij}((q_i - q_j)^2 - \lambda(q_i^2 + q_j^2))$$

から求める．更新の対象とする集合 δ を

$$\delta = \{k \mid W_{ik} > \tau \text{ or } W_{jk} > \tau\}$$

から求める．τ は閾値となる定数で，疎なグラフであれば 0 でよい．行列 $K = L - \lambda D$ を，δ に属する頂点のみ残して求める．これを K' と置く．K' の擬似逆行列 $Z' = (K'^T K')^{-1} K'^T$ を求める．なお，K および K' は対称行列であるため，$Z' = K'^{-1}$ となる．Z' を n×n 行列に戻し，Z とする．ベクトル $h = (\Delta\lambda D + \lambda\Delta D - \Delta L)q$ を求める．Δq を

$$\Delta q = Zh$$

から求める．以上の ΔL, ΔD, $\Delta\lambda$, Δq を L, D, λ, q に反映して ISC の計算を完了する．

6.3 問題点

Spectral Clustering では，一般化固有値問題を解く必要がある．これは，GIM-V のような簡略化された反復計算モデルでは実装が困難である．また，Spectral Clustering のインクリメンタルな高速化手法は存在するが[22]，これは Incremental GIM-V のように『ある程度のグラフ情報の更新間隔ごとに一回のシーケンスを実行し，そのシーケンスの中で一定の反復回数を実行する』というものではなく，『グラフ上の辺が 1 つ更新されるごとに一回のシーケンスを実行し，そのシーケンスの中で，一回ずつの各処理を実行する』という方法である．また，Incremental Spectral Clustering は近似手法であるため，計算を重ねるごとに誤差が蓄積していくため，一定の更新ごとに完全な Spectral Clustering を実行する必要がある．

そこで本節では，5 節の Incremental GIM-V システムをもとに，Spectral Clustering / Incremental Spectral Clustering を実行できるように拡張を行う．以降の節で実装および評価について述べる．

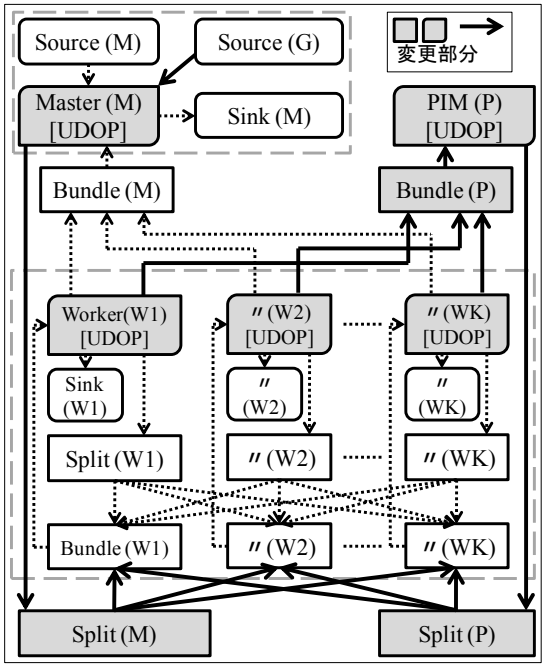

図3. Spectral Clustering向けに拡張した
システムのデータフロー図

7. Spectral Clustering への拡張実装

7.1 データフロー図

図 3 に，システム拡張後のデータフロー図を示す．図 2 から変更のある部分を強調表示している．図 2 との大きな差異を以下に説明する．

- **Source(G)**：グラフデータの入力先を Worker ではなく Master としている．
- **Source(M)**：グラフ情報が更新された際にただちに SC/ISC を行うか否かの設定を行う．また，Worker に固有ベクトルの出力を行わせる命令を受理する．
- **Split(M, P, W1~WK), Bundle(W1~WK)**：Master, Worker, PIM から Worker へのデータの流れを制御する．
- **PIM(P)**：与えられた部分行列の逆行列を計算する．

7.2 処理の流れ

Source(M)から，グラフ情報が更新された際にただちに SC/ISC を行うように設定されているものとする．グラフ情報の更新(辺の追加・重みの更新・削除)は Source(G)から入力される．更新があった時，通常は ISC を開始するが，この更新が最後に SC を行ってから一定回数(=T)後の更新であれば，ISC ではなく SC を開始する．

ISC を実行する場合，まずグラフの更新情報(辺の両端の頂点 ID i, j, 更新後の辺の重み W_{ij})を Master から Worker に伝える．それを受け取ったら，各 Worker が集合 N_{ij} を求め，その要素を PIM に伝える．PIM に N_{ij} の要素を伝え終えたら，PIM

© 2012 Information Processing Society of Japan

からWorkerを経てMasterに処理を返す．次に，Masterから，頂点jを管理するWorkerに対してメッセージを送る．頂点jを管理するWorkerは，頂点iを管理するWorkerに対してq_jを伝える．頂点iを管理するWorkerは，$\Delta\lambda$を求め，頂点jを管理するWorkerにq_jを伝え，全てのWorkerに$\Delta\lambda$を伝える．各頂点xを管理するWorkerは，自身が管理する頂点の中でN_{ij}に所属するものに対して，h_x, $K_{xy}(x \in N_{ij})$を求め，K_{xy}をPIMに伝える．K_{xy}の送信を終えた後，PIMはZ_{xy}を求め，それを各Workerに返す．各頂点xを管理するWorkerは$Zxy \times hy$を頂点yを管理するWorkerに伝え，その値の合計をΔq_yとして頂点yごとにまとめる．最後に，各WorkerがD，λ，qを更新して，Masterに返して，計算を完了する．

SCを実行する場合，まず，Masterから各Workerに，Masterがストックしている全てのグラフ更新情報(i, j, W_{ij}のペア)を全てWorkerに伝え，各WorkerにDの値を更新，LとDをファイルに保存させる．その後，Masterに戻し，保存されたファイルを一般化固有値を求める外部プログラムを呼び出させる．結果のλとqがファイルに出力されるので，Masterから各Workerに，そのファイルを読み込んでλとqを更新させる．最後に処理をMasterに戻して計算を完了する．

8. 評価

本拡張システムを用いたSC/ISCの性能評価を行う．

8.1 実験データ

実験時の計算機環境として，AMD Phenom X4 2.5GHz L2 512KB (4 cores), Memory 8GBからなる計算機を5台用いた．このうち1台はMasterプロセスやグラフ入力の実行(図2中のM, Gに対応)に用いて，残りの4台はWorkerプロセスブロックの実行(図2中のW1~WKに対応)に用いた．これらの計算機は1Gbpsのイーサネットで接続されている．また，OSやミドルウェアとしては全計算機共通でCentOS 5.22.6.18-92.el5 AMD64, gcc4.1.2, InfoSphere Streams 1.2 (System S)を用いた．Workerプロセスブロックの並列数Kは64であり，4台の計算機に対してそれぞれ16プロセスブロックずつ配置した．なお，スケールアウト性の評価のみ，コア数Kに16より小さい値も用いた．

8.2 実験データ

人工データの生成には，Pregel[2]の評価実験と同様に，対数正規分布に基づく無作為生成を行った．ただし，これは有向辺グラフを生成する手法であるため，ここでは有向辺グラフを無向辺グラフに変換した．すなわち，iからjまたはjからiの有向辺が生成されれば，iとjの間に無向辺が生成されたものとして扱った．なお，辺の重みは1とする．以下の2種類の人工データを生成した．

- **LN5000**：|V|=5000, μ=3.0, σ=1.1, |E|=180808
- **LN1000**：|V|=1000, μ=3.0, σ=1.1, |E|=32925

また，実データとしてSNAP(Stanford Network Analysis Platform)[23]にて提供されているStanford Large Network Dataset Collectionから，以下のデータを選択した．なお，グラフ処理の都合上，最大弱連結成分のみを残し，その他の頂点と辺を取り除いた．なお，これらのデータは本来有向辺グラフだが，上記の人工データと同様に無向辺グラフに変換した．辺の重みは1とする．

- **ca-GrQc**：|V|=4158, |E|=13422, arXivの一般相対性理論と量子宇宙論部門の論文の，著者と論文のコラボレーションネットワーク．
- **p2p-Gnutella08**：|V|=6299, |E|=20776, Gnutellaのp2pネットワークのスナップショット．2002年8月の9階のスナップショットの合計．
- **wiki-Vote**：|V|=7066, |E|=100736, Wikipediaの管理者投票のユーザー間のネットワーク．

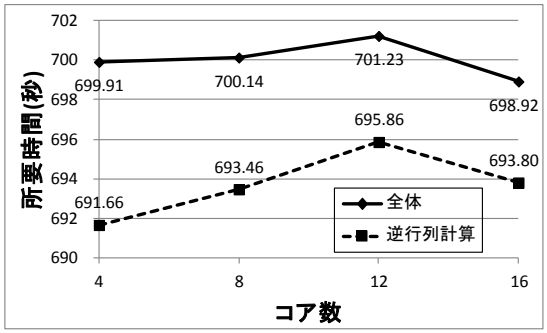

図4. LN5000に対する並列コア数ごとのISC100回の所要時間

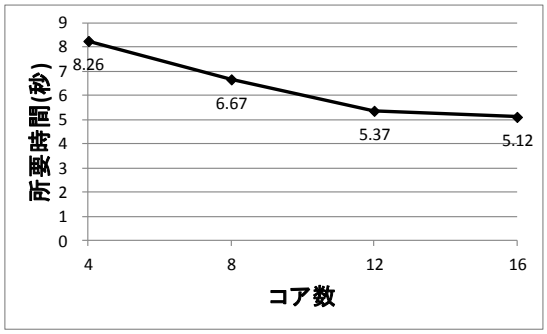

図5. LN5000に対する並列コア数ごとのISC100回の所要時間(逆行列計算を除く)

8.3 実験結果

人工データLN5000を用いて，並列計算の並列数(Workerインスタンス数K)に対するスケールアウト性を評価した．LN5000に対するSCの結果が得られている場合の，更新時のISC実行時間を，並列数Kを変更して評価した．更新の内容は，無作為に選んだ辺の重みを2に変更するというものを100回行った．結果を図4, 5に示す．図4はISC100回の所要時間『全体』とそのうち『逆行列計算(非並列処理)』に要した時間を，図5は『全体』から『逆行列計算』に要した時間を減じたものを示している．図4より，ISC計算の所要時間の大部分は逆行列計算に依存していることが分かる．図5より，逆行列計算以外の並列処理は，非線形だがスケールしていることが分かる．本稿における実装では，処理時間の大部分を占める逆

© 2012 Information Processing Society of Japan

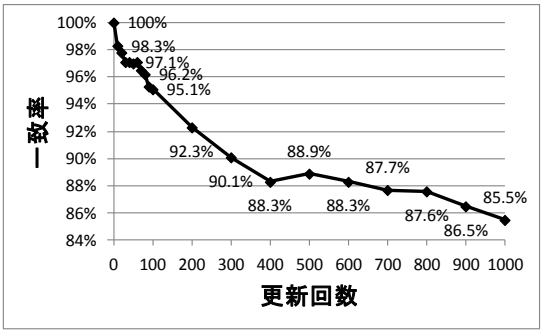

図6. LN1000に対するISCの近似性能
（更新回数～1000）

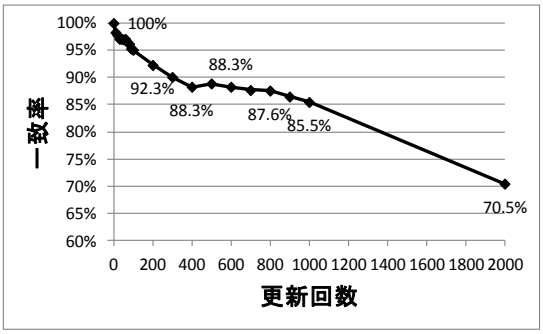

図7. LN1000に対するISCの近似性能
（更新回数～2000）

行列計算をシングルコア実行しているが，これを並列分散化できれば，処理のスケールアウトが見込めることがわかった．また，スケールアップの面では，SC1 回の所要時間がコア数に依らず約 7988 秒，ISC1 回の所要時間がコア数に依らず約 7 秒であり，1141 倍の高速化が出来ていることがわかる．逆行列計算を GPGPU 利用などにより高速化できれば，さらなるスケールアップが見込める．

人工データ LN1000 を用いて，ISC の近似性能を評価した．LN1000 の SC の実行結果から，1000 回の更新に対し，ISC を繰り返し行っていった場合のクラスタリング結果と，SC で正確に計算した場合のクラスタリング結果を，結果の一致率で評価した．更新の内容は，無作為に辺を追加するものとする．なお，クラスタリングの正負は，正と負を入れ替えても本質的に差異はない．それゆえに，SC と ISC の比較を行う際，正負の一致率と正負を反転させた場合の一致率のうち良い方を採用した．一致率の最低値は 50%となる．結果を図 6, 7 に示す．結果より，更新を繰り返すことにより誤差が蓄積していくことが分かる．なお，更新前の SC の実行時間は 64.95 秒であり，ISC1 回当たりの平均時間は 0.10 秒であった．これらの結果に関して，9 節の議論へ続く．

SNAP による実データを用いて評価を行った．評価の内容は，SC，ISC の実行時間，および ISC の近似性能である．更新の内容は，各グラフに対して無作為に選んだ辺の重みを 2 にするというものを 1000 回行った．実行結果を表 1 に示す．

	ca-GrQc	p2p-Gnutella08	wiki-Vote
SC 所要時間 (更新前)(秒)	4291	14986	20754
ISC1000 回 所要時間(秒)	22.14	27.66	2601.40
ISC 所要時間 1 回当たり(秒)	0.022	0.028	2.601
SC 所要時間 (更新後)(秒)	4378	12517	17495
ISC1000 回 近似性能(一致率)	99.86%	99.63%	99.86%

表 1. SNAP に対する SC/ISC 実行結果

9. 議論

Incremental Spectral Clustering に対する最適な Spectral Clustering の再実行タイミングについて議論する．

ISC において，ISC1 回の平均所要時間を t_{isc}，SC1 回の平均所要時間を t_{sc}，更新の起こる周期を t_{update} とする．T 回の更新ごとに SC を実行し，それ以外の更新では ISC を実行するものとする．更新 T 回の SC/ISC 所要時間は，

$t_{isc} \times (T-1) + t_{sc}$

となり，1 回当たりだと

$t_{isc} \times (T-1)/T + t_{sc}/T$

となる．ここで，t_{isc} と t_{sc} が定数であるとして，さらに近似性能に対して許容できる ISC での更新回数が既知であるならば (T が定数)，システムが処理できる更新周期の上限(頻度，スループット)は

$t_{update} = t_{isc} \times (T-1)/T + t_{sc}/T$

と求まる．例えば，8 節の条件下における LN1000 が対象であれば，t_{isc} が 0.10 秒，t_{sc} が 64.95 秒であり，T=1000 程度まで近似計算を許すならば，システムが処理できる更新周期の上限は上式より 0.16 秒，スループットとしては 1 秒間に 6.07 回までの更新を許容できることになる．

逆に，時系列データの更新周期が既知であるならば(t_{update} が定数)，T の下限が

$(t_{sc} - t_{isc})/(t_{update} - t_{isc})$

と求まる．上記の条件で，仮に 1 秒間に 8 回の更新がある(t_{update} が 0.125 秒)とするならば，T の下限は 2594 となる．

このように，t_{isc} や t_{sc} の条件が定まっているならば，システム利用者の要求に応じて，可能な t_{update} や T の値が推測できる．ただし，実際にはグラフが大きくなるにつれて t_{isc} や t_{sc} も大きくなっていき，また，t_{update} もシステムが扱う時系列データによって変化しうるため，上記の方法による T の最適化は利用できない．そこで，これらの条件の変化に応じて T を動的に変化させる機構などを実装すれば有用になるかもしれない．

10. まとめと今後の課題

本稿では，Incremental GIM-V グラフ処理システムの拡張

ⓒ 2012 Information Processing Society of Japan

によりSpectral Clusteringへの応用，評価を行った．計算の枠組みとしては大きく変わるものの，既存のデータフローを有効に使って実装することが出来ることを示した．また，Incremental Spectral Clusteringのスケールアウトの発展性やIncremental Spectral Clusteringをデータストリーム処理として用いる際の近似性能，スループットなどについて議論した．

現状の実装では，完全なSpectral Clusteringの計算を行う部分(外部プログラム)，およびIncremental Spectral Clusteringのうち逆行列計算を行う部分を，並列計算ではなくシングルコア実行するように実装されている．これらを並列計算できるようにし，スケールアウト性を強化し，評価する必要がある．また，本稿において採用したIncremental Spectral Clusteringは2部クラスタリングしか行えず，より多いコミュニティへのクラスタリングやソフトクラスタリングに適用できない．これらを行えるクラスタリングアルゴリズムが求められる．

謝辞

本研究は，科学技術振興機構(JST)の戦略的創造研究推進事業「CREST」における研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」によるものである．

参 考 文 献

[1] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Library Technologies, January 1998.

[2] G Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser and G Czajkowski, "Pregel: A System for Large-Scale Graph Processing!" ACM 2010.

[3] U Kang, C. E. Tsourakakis and C, Faloutsos, "PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations", ICDM 2009.

[4] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing and S. Zdonik, "The Design of the Borealis Stream Processing Engine", In Proc. CIDR, pages 277.289, 2005.

[5] J. Wolf, N. Bansal, K. Hildrum, S. Parekh, D. Rajan, R. Wagle, K.-L. Wu and L. Fleischer, "SODA: An Optimizing Scheduler for Large-Scale Stream-Based Distributed Computer Systems", Middleware, 2008.

[6] B. Gedik, H. Andrade and K.-L. Wu, "A Code Generation Approach to Optimizing High-Performance Distributed Data Stream Processing", In Proc. USENIX, pages 847-856, 2009.

[7] P. Desikan, N. Pathak, J. Srivastava and V. Kumar, "Incremental Page Rank Computation on Evolving Graphs", ACM 2005.

[8] B. V. Cherkassky, A. V. Goldberg and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation", Mathematical Programming 73, 1996, 129-174.

[9] T. Anderson, S. Owicki, J. Saxe and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks", ACM Trans. Comp. Syst. 11(4), 1993, 319-352.

[10] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", OSDI, 2004.

[11] "Hadoop information", http://hadoop.apache.org/ .

[12] J.-Y. Pan, H.-J. Yang, C. Faloutsos and P. Duygulu, "Automatic multimedia cross-modal correlation discovery", ACM SIGKDD, Aug. 2004.

[13] U. Kang, C. Tsourakakis, A. Appel, C. Faloutsos and J. Leskovec, "Hadi: Fast diameter estimation and mining in massive graphs with hadoop", CMU-ML-08-117, 2008.

[14] S. Kamvar, T. Haveliwala and G. Golub, "Adaptive methods for the computation of PageRank", Linear Algebra and its Applications 386 (2004) 51-65.

[15] 松浦紘也，雁瀬優，鈴木豊太郎，"データストリーム処理系 System S と Hadoop の統合実行環境", ComSys 2010.

[16] 西井俊介，鈴木豊太郎，"データストリーム処理によるインクリメンタルグラフ処理に向けて"，電子情報通信学会データ工学研究会, 2011, pp.43-48.

[17] G Chang, M. J. Healey, J. A. M. McHugh and J. T. L. Wang, "Mining the World Wide Web: An Information Search Approach (The Kluwer International Series on Information Retrieval, 10)", Kluwer Academic Pub 2001/6/1.

[18] G Chang, 梅村 恭司, M. J. Healey, 藤井 敦, J. A.M. McHugh, J. T.L. Wang, 武田 善行, "Web マイニング"(翻訳), 共立出版 2004/01.

[19] 雁瀬優，上野晃司，鈴木豊太郎，"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"，情報処理学会論文誌(ACS 論文誌第35 号), 2011.

[20] U. V. Luxburg, M. Belkin, O. Bousquet and Pertinence, "A tutorial on spectral clustering", Stat. Comput, 2007.

[21] J. Shi and J. Malik. "Normalized cuts and image segmentation". IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2000.

[22] H. Ning, W. Xu, Y. Chi, Y. Gong and T. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities", In SIAM Int. Conf. on Data Mining, 2007.

[23] http://snap.stanford.edu/ "SNAP: Stanford Network Analysis Project".

[24] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", PHYS.REV.E, vol. 69, 2004.

[25] Clauset, Aaron and Newman, M. E. J. and and Cristopher Moore, "Finding community structure in very large networks", Physical Review E, 2004, pp.1-6.

[26] Andrew Y. Ng, Michael I. Jordan and Yair Weiss, "On Spectral Clustering: Analysis and an algorithm", NIPS, 2002.

[27] Bezdek, James C. and Ehrlich, Robert and Full, William, "FCM: The fuzzy c-means clustering algorithm", Computers & Geosciences, 1984, vol. 10, pp.191-203.

ⓒ 2012 Information Processing Society of Japan