Vol.2014-DBS-160 No.20
Vol.2014-OS-131 No.20
Vol.2014-EMB-35 No.20
2014/11/18

並列分散処理による2段階ランキングアルゴリズム

武井　良太[1,a)]　新美　礼彦[1,b)]

概要：ウェブグラフを再構築することで, ランキングアルゴリズムを実行する際に必要となる解析時間を短縮することが可能である. ランキングアルゴリズムで解析する際, 並列分散化処理に適応できるようにすることで, 更なる解析時間の短縮が期待できる. 本研究は, ウェブグラフを PageRank の解析が並列分散処理可能なグラフへ再構築を行い,PageRank の解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する方法を提案する. 本提案方法は, ウェブグラフのノードを SCAN により Cluster と Hub, Outlier に分別し,Cluster と Hub で構成されるグラフと Cluster 内のノードで構成されるグラフにウェブグラフを再構築する.Cluster 内のグラフをランキングアルゴリズムで解析する際, お互いの Cluster は関係性が無いため並列分散処理が可能になり, ランキングアルゴリズムの解析時間を短縮することが出来る.

1. はじめに

ウェブグラフとはウェブをグラフ構造化したものであり, ノードをウェブページ, エッジをリンクで表現したものである. ウェブグラフを解析することで知見を導出することが盛んに行われており, 主な解析法としてランキングアルゴリズムである PageRank[1] や HITS[2], コミュニティ抽出として trawling[3] や DBG[4] がある. 近年では, ノード数とエッジ数が数億個になるグラフが存在している. 例えば Netcraft は 2014 年 10 月の時点で, 全世界で少なくとも 10 億ページ以上存在すると報告している [5]. ウェブグラフは極めて大規模になっており, 解析にかかる時間は膨大になる. 例えば, ランキングアルゴリズムである PageRank の場合,N 個ノードがある時, 計算量は $O(N^2)$ となる. そのため, 大規模なグラフに対して解析時間を短縮する方法が必要である. 解析時間を短縮する方法 [6],[7],[8],[9],[10] は複数あるが,1 つとしてウェブグラフを再構築し解析時間を短縮する方法が挙げられる. 本研究では, ランキングアルゴリズムである PageRank の解析時間の更なる短縮を目標とし,PageRank の解析を並列分散化出来るような形状にグラフを再構築, 解析に必要な時間を短縮する方法を提案する. 本方法では Xu らが提案した SCAN[11] を用いてグラフを再構築する.SCAN は構造的類似度に基づいたグラフクラスタリング方法の 1 つである. 構造類似度の下限値を示す閾値 ε と,Cluster の最小ノード数を示す閾値 μ を用いて Cluster を作成する.Cluster に属さなかったノードを 2 つ以上の Cluster とつながりがあるノードを Hub とし,1 つのみの Cluster とつながりがあるノードを Outlier と分別する. これにより, ウェブグラフを Cluster と Hub で構成されるグラフと Cluster 内のノードで構成されるグラフにウェブグラフを再構築する.Cluster 内のグラフを PageRank で解析する際, お互いの Cluster は関係性が無いため並列分散処理が可能になる. 更に, クラスタで切り分けることで 1 回あたりの PageRank の解析対象となるノードとエッジが減少する. 以上より,PageRank の解析時間を短縮することが出来る.

[1] 公立はこだて未来大学
Future University Hakodate
[a)] g2114020@fun.ac.jp
[b)] niimi@fun.ac.jp

2. 先行研究

大規模なウェブグラフを再構築することは, ウェブグラフを用いる解析方法において重要な技術である. ウェブグラフの再構築は大きく分けると, 符号化による方法と構造改良による方法の 2 種類に分かれる.

2.1 符号化

符号化による方法は既存のデータ圧縮を使用するか, ウェブグラフの特徴を用いる方法である.P. Boldi らはウェブグラフの特徴と WebGraph[12] という名の方法を提案した. ウェブグラフの特徴として, 局所性と類似性, 連続性を挙げている.

・局所性

多くのウェブページは同じホスト内のページに誘導するためのリンクを含んでいる. 例えば,“home”,“next”,“previous”,“up” などのリンクが挙げら

Vol.2014-DBS-160 No.20
Vol.2014-OS-131 No.20
Vol.2014-EMB-35 No.20
2014/11/18

れる. 例で上げたリンクは同じホスト内でリンクされることが多い. つまり, リンク元とリンク先のウェブページのURL を比較すると先頭文字列 (ホスト名) が一致しやすく, 局所性が現れる.

・類似性

同じホストのウェブページは似たようなウェブページにリンクを持つ傾向がある. 例えば, 共通のテンプレートで作成されるページなどが挙げられる. 例で上げたページは同じ構造のページがたくさんあるため同じウェブページにリンクを貼ることが多い. つまり, 同一ホスト名のウェブページのリンクの URL を比較すると URL が一致しやすく, 類似性が現れる.

・連続性

ウェブページに含まれるリンクの URL は一定の固まりで現れる傾向がある. 例えば, 大量の写真を掲載しているページが挙げられる. 例で上げたページはある一定の連続したリンクの固まりが存在する. つまり,URL を辞書順に整列し順に ID を割り当てると数字が連続し, 連続性が現れる.

P. Boldi らは上記の特徴を用いて, 連続性を利用した差分符号化と, 局所性と類似性を利用したランレングス圧縮を用いた方法を提案している. これにより,1 リンクあたり 3.08bit に抑えることに成功している.S. Tei らは WebGraph の差分符号化の後算術符号を用いることで, ウェブグラフの情報量を WebGraph よりも 2 割ほど多く削減することに成功している [13]. しかし, 符号化による方法では情報量を削減することは出来るが, ノード数とエッジ数を削減出来ない. また, 復号化する必要があり PageRank の解析時間が増加してしまう. よって本研究では, 符号化による方法は用いない.

2.2 構造改良

構造改良はウェブグラフの構造を一部組み替えることでノード数やエッジ数を削減する方法である.G. Buehrer らは Virtual Node Miner[6] と呼ばれる, 架空のノード (以下,VN) をウェブグラフに挿入することでエッジ数を削減する方法を提案した.VN は図 1 のような完全 2 部グラフの中間に図 2 のように挿入する. これによって, リンクを集約しノードを削減している.Virtual Node Miner は完全 2 部グラフの探査に最小値独立置換族を利用している. これにより, ノード数約 7800 万, エッジ数約 3 億のウェブグラフに対して, 約 1700 万の VN を挿入しエッジ数を 1/7 に削減出来たとしている. 片瀬らは LittleWeb[7] と呼ばれる, 類似ノード集約によるウェブグラフ圧縮手法を提案した.LittleWeb は集約対象となるノードを探査するためのクラスタリング処理と, クラスタリング結果を用いた構造改良の 2 ステップでウェブグラフを再構築している. これにより, ノードを 67.3 %, エッジを 56.3 %削減している. 更に,PageRank の解析時間を 67 %短縮することに成功している. 構造改良による方法は, ノード数とエッジ数を削減することが出来かつ,PageRank の解析時間を短縮することが可能である. よって本研究では,PageRank の解析時間の短縮を目標としているため構造改良による方法を用いる. 従来方法はエッジ数やノード数を削減し,PageRank の解析時間を短縮することには成功している. 本研究では, ウェブグラフを PageRank の解析が並列分散処理可能なグラフへ再構築を行い,PageRank の解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する方法を提案する.

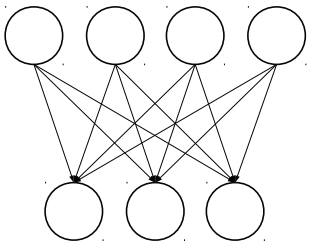

図 1 圧縮前

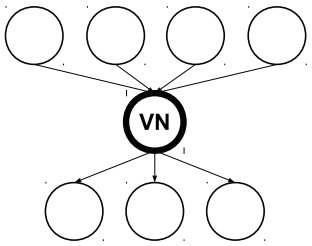

図 2 圧縮後

3. SCAN

提案方法で用いるクラスタリング方法の SCAN[11] について説明する.SCAN はクラスタリング対象とするグラフ $G = (V, E)$ を解析し, 構造類似度の下限値を示す閾値 ε と,Cluster の最小ノード数を示す閾値 μ に応じた structure-connected cluster を作成する.Cluster に属さなかったノードは Hub か Outlier の分別が行われる.

最初にすべてのノードに対して *unclassfied* のラベルを与える.SCAN は *unclassfied* のラベルが付いているノードに対して, そのノードが core であるかどうかの判定を行う.core ノードは Cluster の中心となるノードである.core であるか判断するためには, 対象となる *unclassfied* なノードの構造的隣接ノード集合を定め, 構造的類似度を計算, 閾値 ε を用いて ε-neighborhood を算出することで core かどうか判断することが出来る. 以下に構造的隣接ノード集合と構造的類似度,ε-neighborhood の定義を記述する.

・構造的隣接ノード集合

$v \in V$ であるとき, 構造的隣接ノード集合はノード v にエッジで接続するノードと, ノード v 自らで構成される集合 $\Gamma(v)$ で表される.

$$\Gamma(v) = \{v \in V \mid \{v, w\} \in E\} \cup \{v\}$$

・構造的類似度

$|\Gamma(v)|$ が隣接ノード集合に含まれるノード数とすると, ノード v, w 間の構造的類似度は $\sigma(v, w)$ で表される.

$$\sigma(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

・ε-neighborhood

構造類似度の下限値を示す閾値 ε を用いることで,ε-neighborhood と表される.

Vol.2014-DBS-160 No.20
Vol.2014-OS-131 No.20
Vol.2014-EMB-35 No.20
2014/11/18

$$N_\varepsilon(v) = \{w \in \Gamma(v) | \sigma(v,w) \geqq \varepsilon\}$$

・Core

$\varepsilon \in \mathbb{R}$,$\mu \in \mathbb{N}$,$v \in V$,$|N_\varepsilon(v)|$ をノード v の ε-neighborhood のノード数とすると,core は $CORE_{\varepsilon,\mu}(v)$ で表される.

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geqq \mu$$

SCAN の判定によりノードが core であった時, この core ノードにユニーク ID である *clusterID* のラベルを与え,core ノードを中心に structure-connected cluster を作成する. ノードが core でなかった時, そのノードに *non-menber* のラベルを与える.

SCAN は core ノードと判定されたノードから,structure reachability で到達可能なノードが所属する Cluster に *clusterID* のラベルを与える. 最初に, キュー Q へ core ノードの ε-neighborhood に含まれるすべてのノードを挿入する. 次に, キュー Q へ挿入されたノードに対して,direct structure reachability となるノードの集合である R を求める. 以下に direct structure reachability と structure reachability の定義を記述する.

・Direct structure reachability

ノード v とノード w における direct structure reachability は $DirREACH_{\varepsilon,\mu}(v,w)$ で表される.

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge N_\varepsilon(v)$$

・Structure reachability

$\varepsilon \in \mathbb{R}$,$\mu \in \mathbb{N}$,$v \in V$ とすると, ノード v とノード w における Structure reachability は $REACH_{\varepsilon,\mu}(v,w)$ で表される.

$$\begin{aligned} &REACH_{\varepsilon,\mu}(v,w) \Leftrightarrow \\ &\quad \exists_1, ...v_n \in V \ : \ v_1 = v \wedge v_n = w \wedge \\ &\forall i \in \{1, ..., n-1\} \ : \ DirREACH_{\varepsilon,\mu}(v_i, v_{i+1}) \end{aligned}$$

ノード集合 R を求めるには, キュー Q に含まれる全てのノードの構造的隣接ノード集合に対して, 構造的類似度を計算する. 最後にノード集合 R に含まれるノードの所属する Cluster が *unclassfied* の時, ノードをキュー Q へ挿入する. ノード集合 R に含まれるノードが *unclassfied*, または *non-menber* の時, 作成した *clusterID* のラベルを与える. 一連の処理をキュー Q がなくなるまで行い,structure-connected cluster を作成し続ける.

structure-connected cluster の作成が終了したら,*non-menber* のラベルが付いたノードに対して Hub と Outlier の分別を行う.*non-menber* のラベルが付いたノードが 2 つ以上の異なる Cluster と接続している場合, そのノードに対して Hub のラベルを与える.1 つのみの Cluster と接続している場合, そのノードに対して Outlier のラベルを与える.

以上より, 全てのノードに対してラベルが付け終わり SCAN によるクラスタリングは完了する.

4. 提案方法

本研究では, ウェブグラフを PageRank の解析が並列分散処理可能なグラフへ再構築を行い,PageRank の解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する方法を提案する. 提案方法によるウェブグラフ解析方法は,3 つのステップで行われる.

Step 1

SCAN を用いて Web graph をクラスタリングし Cluster と Hub,Outlier にノード分別する.

Step 2

PageRank の解析が並列分散処理可能にするため,Cluster と Hub を用いた Compression graph と Cluster 内のウェブグラフである Cluster graph の 2 種類のグラフを作成し,Web graph を再構築する.

Step 3

Compression graph の PageRank 値を算出し,Cluster と Hub によるランキングを作成する. その後,Cluster graph の PageRank 値を算出し,Cluster 内のノードによるランキングを作成し, それぞれのランキングをマージする.

図 3 は提案方法の流れを図示したものである. 以降, それぞれのステップを詳しく説明する.

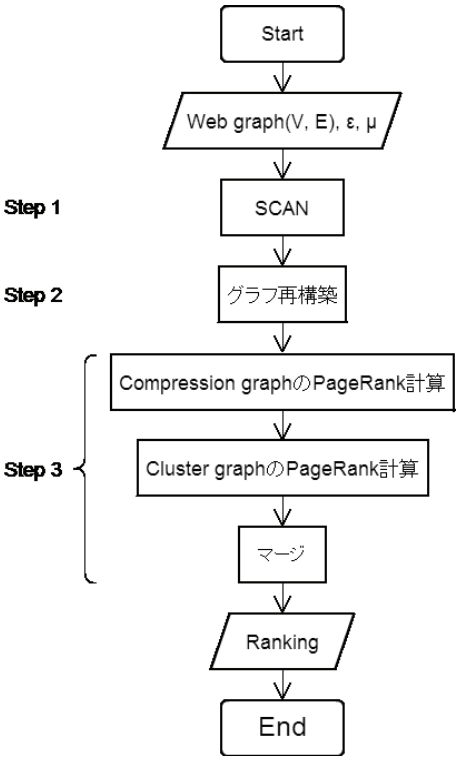

図 3 提案方法の流れ

4.1 クラスタリング

提案方法では,SCAN を用いてクラスタリングを行

Vol.2014-DBS-160 No.20
Vol.2014-OS-131 No.20
Vol.2014-EMB-35 No.20
2014/11/18

う.SCAN はグラフ内のすべてのエッジを解析し, 構造的類似度の下限値を表す閾値 ε と Cluster の最小ノード数を表す閾値 μ に応じた Cluster を作成する.Cluster に属さなかったノードに対して 2 つ以上の Cluster とエッジで接続されているノードは Hub とし,1 つのみの Cluster とエッジで接続されているノードは Outlier と分別する. クラスタリング処理の例である図 4 は上段の Web graph に対して SCAN を実行し, 下段はノードを Cluster と Hub,Outlier に分別したグラフである.

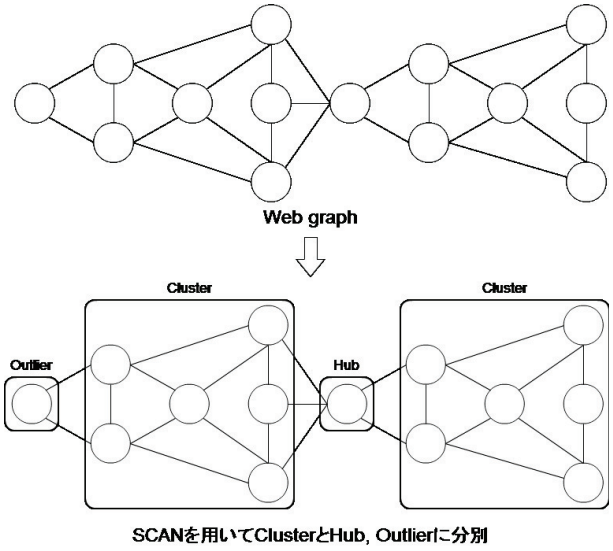

図 4 クラスタリングの流れ

4.2 再構築

SCAN の結果を用いて Web graph を再構築する.Cluster と Hub で構成されたウェブグラフである Compression graph と Cluster 内のウェブグラフである Cluster graph の 2 種類を作成する. 再構築の例である図 5 は 4.1 で SCAN を用いてノードを分別した上段のグラフより,Cluster と Hub のみで構成される Compression graph と,Cluster 内のノードで構成される Cluster graph に再構築したものを下段に示す.

4.3 PageRank

PageRank での解析ステップは 2 つのステップに分かれている.1 つ目は Compression graph の解析,2 つ目は Cluster graph の解析である.2 つ目の Cluster graph の解析は並列分散処理を用いる. 図 6 は解析のステップを図示したものである. 最初に Cluster と Hub で構成されたウェブグラフである Compression graph の PageRank 値を算出し,Cluster と Hub に対するランキングを求める. この時, 異なる 2 つの Cluster に属するノード間のエッジのみを考慮し,Cluster 内のノード間のエッジは無視する. また, 同じ Cluster 間のエッジが 2 つ以上存在しても 1 つのみのエッジとして扱う.

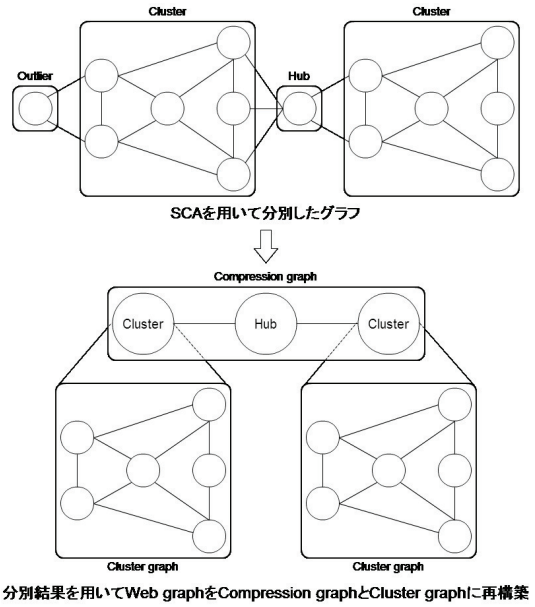

図 5 再構築の流れ

次に Cluster 内のウェブグラフである Cluster graph の PageRank 値を算出し,Cluster 内のノードに対するランキングを求める. この時,Cluster 外のノード間のエッジは無視する.Cluster graph のランキングが求まり次第,Compression graph の対応する Cluster 部分に Cluster graph のランキングを挿入し, ウェブグラフのランキングを作成する. 解析のステップの例である図 6 は, 初めに Compression graph を PageRank を用いてランキングを作成. 次に,Cluster graph を PageRank を用いてランキングを作成. この時,Hub に関しては特に操作は行わない. 最後に, それぞれの解析結果であるランキングをマージしている.

4.4 特徴

本研究では, ウェブグラフを PageRank の解析が並列分散処理可能なグラフへ再構築を行い,PageRank の解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する方法を提案しているが, 並列分散処理は Cluster 内のノードに対するランキングを求める時に行われる.Cluster 内のノードに対するランキングを求める際,Cluster 外のノード間のエッジは無視するため, お互いの Cluster は関係性が無くなり, 並列分散処理が可能となる. 更に,Cluster ごとに PageRank 値を算出するため,1 回あたりの PageRank 値の算出時に計算対象となるノードとエッジが減少する. 以上より, ウェブグラフを PageRank で解析する際の解析時間を短縮することが可能となる.

5. 考察

提案方法に対して 4 つの考察すべき点がある.

1 つ目に本提案方法の妥当性を考える. 並列分散処理を行いたい場合, 考慮しなければならない問題として, 分割

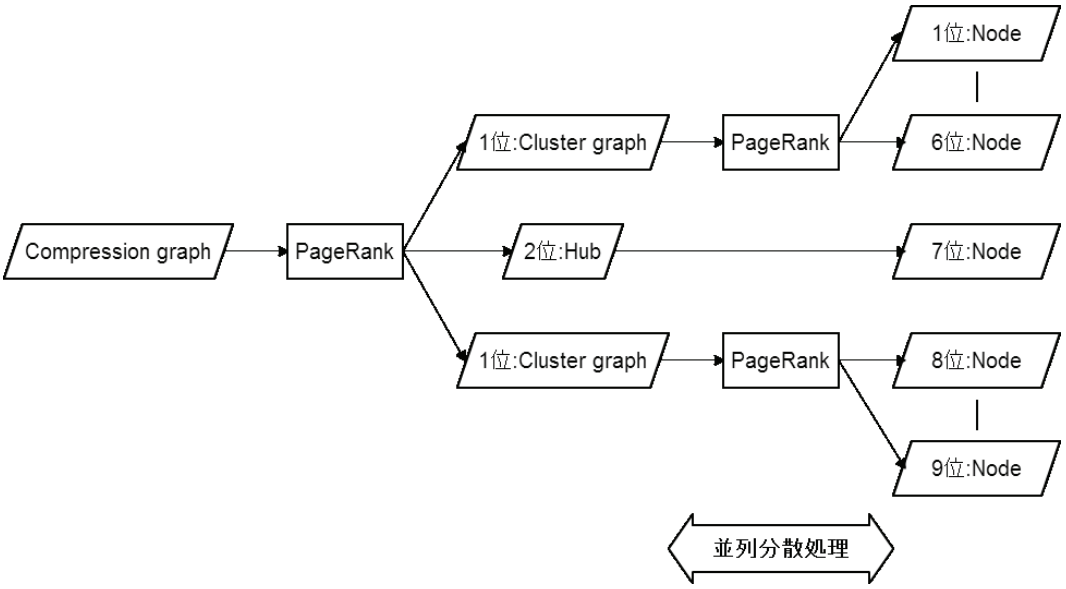

図 6 解析の流れ

した処理どうしに関係性が無いかどうか考える必要がある. 本提案方法は Cluster 内のノードを PageRank にて解析する際に並列分散処理を行うが,Cluster 内のノードは他の Cluster や Hub,Outlier とはエッジにて接続されていない. すなわち,Cluster 内のノードは Cluster 内のノード間のみにてエッジがあり, 関係性がある. よって,Cluster 内の PageRank 解析は独立した処理となり並列分散化が可能となる.

2 つ目に解析対象となりうるウェブグラフを考える.SCAN は相互に密な関係性, すなわち相互に密なエッジがある部分を Cluster として抜き出す. よって, 解析対象とするウェブグラフには全体に粗密が存在するグラフである必要性があると考える. 具体的な解析対象となるウェブグラフは, ウェブを普遍的にクローリングしたグラフであることが望ましい. 本提案方法で PageRank 解析の高速化が望めないウェブグラフは, 単一ドメインを対象としたグラフやブログサイトをクローリングしたグラフは解析時間が短縮しづらいと思われる.

3 つ目にエッジの扱いに関する検討である. 現在は SCAN の方法を使用し,Cluster とのエッジに注目して Hub と Outlier の分別を行っており, エッジがあるかないかのみを分別条件としている. ここにエッジの本数, すなわちエッジに重みの情報を追加する. この情報の追加により, 重みの下限値の閾値を設定し, 閾値に満たない重み付きエッジを所有しているノードを切り落とすことで PageRank の解析が更に高速化出来ると考える. この他にエッジの方向による分別手法に検討の余地があると考える. 対象とするノードが両方向のエッジ, すなわちインとアウト両方のエッジを所有しているノードと, 片方向のみエッジ, すなわちインまたはアウト片方のエッジを所有しているノードとに分別する. この分別方法により, インのみのエッジを所有しているノードを切り落とすことで PageRank の解析時間が更に短縮出来ると考える.

4 つ目にウェブグラフのクラスタリングに掛かる時間に関する検討である. 提案方法は,PageRank によるウェブグラフの解析に前処理となるウェブグラフのクラスタリング作業があるため, 単純に PageRank による解析よりも処理数が多くなっている. そのため, クラスタリング処理は短時間で終わることが望ましい. 実装する際,SCAN の処理を MapReduce に対応させた PSCAN[14] や, クラスタ係数が大きい現実のグラフ構造に対してクラスタリング処理時間を短縮できる改良型 SCAN[15] があり, これらを用いることでより処理時間を短縮することが可能になるのではないかと思われる.

6. MapReduce による提案方法の並列分散化

本提案方法はノード数が億単位以上のウェブグラフが解析対象となるため, クラスタリング処理と PageRank による解析は MapReduce を用いた並列分散処理を行う方針を取る.MapReduce は大規模データを並列分散処理するためのフレームワークである.Map 処理でデータを定義した方法で分割を行い,Reduce 処理で定義した方法で分割したデータを収集し, 処理を行う.SCAN に関しては PSCAN[14] で,PageRank に関しては Data-intensive text processing with MapReduce[16] で既に MapReduce で定義されているため,Web graph の再構築である Cluster graph と Compression graph に関して MapReduce の定義を述べる.

まず,Cluster graph における MapReduce について述べる.Key/Value は表 1 に示すものを用いる. 表中の label

はノードに付いている Cluster や Hub,Outlier ことであり,node はノードに固有で与えられている ID,outdegree はノードから出ているリンク先の node のことを示す. また,Cluster と Hub にはそれぞれ固有の ID が与えられているものとする.

表 1 Cluster graph における Key/Value

		Key	Value
Map	input	label	node
	output	label	node, outdegree
Reduce	input	Cluster ID	node, outdegree
	output	node	outdegree

Cluster graph における MapReduce は, まず Map 処理として,input では,Cluster や Hub,Outlier のラベルを Key とし, それぞれに対応する node を Value に挿入する.output では,Value の node に対応する outdegree を追加する. 次に Reduce 処理として,input では,Key から Cluster のラベルが付いたものを収集する.output では,Value に格納されている node を Key に挿入する.Reduce 処理を Cluster ID ごとに繰り返し, 全ての Cluster graph を作成する.

次に,Compression graph における MapReduce について述べる.Key/Value は表 2 に示すものを用いる.

表 2 Compression graph における Key/Value

		Key	Value
Map	input	label	node
	output	label	node, outdegree
Reduce	input	Hub ID	node, outdegree
	output	Hub ID	Cluster ID

Map 処理に関しては Cluster graph と同じ処理を行う.Reduce 処理として,input では,Key から Hub のラベルが付いたものを収集する. 収集後,Value の outdegree から, どの Cluster と接続しているのかを調べ,Cluster ID を output の Value に追加する.Reduce 処理を Hub ID ごとに繰り返し,Compression graph を作成する.

7. おわりに

本研究では, ウェブグラフを PageRank の解析が並列分散処理可能なグラフへ再構築を行い,PageRank の解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する方法を提案する. 提案方法によるウェブグラフ解析方法は,3 つのステップで行われる.1 つ目は,SCAN を用いて Web graph をクラスタリングし Cluster と Hub,Outlier にノード分別する.2 つ目に,PageRank の解析が並列分散処理可能にするため,Cluster と Hub を用いた Compression graph と Cluster 内のウェブグラフである Cluster graph の 2 種類のグラフを作成,Web graph を再構築する.3 つ目は,Compression graph の PageRank 値を算出し,Cluster と Hub によるランキングを作成する. その後,Cluster graph の PageRank 値を算出し,Cluster 内のノードによるランキングを作成, マージする.Cluster 内のノードに対するランキングを求める時,Cluster 外のノード間のエッジは無視するため, お互いの Cluster は関係性が無くなり, 並列分散処理が可能となる. 更に, クラスタで切り分けることで 1 回あたりの PageRank の解析対象となるノードとエッジが減少する. 以上より,PageRank の解析時間を短縮することが出来る. これにより,PageRank を用いている検索エンジンなどで, より高度な検索結果を返すことが可能になり, 情報社会における品質の高いサービスを提供することに貢献できる. 今後は実験を行い, 本提案方法が可能か検証を行う予定である. 可能であれば, 現在は SCAN を利用しているが, 他のクラスタリング方法でも提案方法が実現できる可能性があると思われるため, 追加で実験を行う予定である.

参考文献

[1] Brin, S. and Page, L.: The anatomy of a large-scale hypertextual Web search engine, *Computer networks and ISDN systems*, Vol. 30, No. 1, pp. 107–117 (1998).
[2] Kleinberg, J. M.: Authoritative sources in a hyperlinked environment, *Journal of the ACM (JACM)*, Vol. 46, No. 5, pp. 604–632 (1999).
[3] Kumar, R., Raghavan, P., Rajagopalan, S. and Tomkins, A.: Trawling the Web for emerging cyber-communities, *Computer networks*, Vol. 31, No. 11, pp. 1481–1493 (1999).
[4] Reddy, P. K. and Kitsuregawa, M.: An approach to relate the web communities through bipartite graphs, *Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on*, Vol. 1, IEEE, pp. 301–310 (2001).
[5] Netcraft: October 2014 Web Server Survey.
[6] Buehrer, G. and Chellapilla, K.: A scalable pattern mining approach to web graph compression with communities, *Proceedings of the 2008 International Conference on Web Search and Data Mining*, ACM, pp. 95–106 (2008).
[7] 片瀬弘晶, 上田高徳, 山名早人 : LittleWeb: 類似ノード集約による Web グラフ圧縮手法, *The 2nd Forum on Data Engineering and Information Management*, DBSJ, pp. E1–4 (2010).
[8] Kohlschütter, C., Chirita, P.-A. and Nejdl, W.: Efficient parallel computation of pagerank, *Advances in information retrieval*, Springer, pp. 241–252 (2006).
[9] Wicks, J. and Greenwald, A.: Parallelizing the computation of pagerank, *Algorithms and Models for the Web-Graph*, Springer, pp. 202–208 (2007).
[10] Kamvar, S., Haveliwala, T., Manning, C. and Golub, G.: Exploiting the block structure of the web for computing pagerank, *Stanford University Technical Report* (2003).
[11] Xu, X., Yuruk, N., Feng, Z. and Schweiger, T. A.: Scan: a structural clustering algorithm for networks, *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 824–833 (2007).
[12] Boldi, P. and Vigna, S.: The webgraph framework I: compression techniques, *Proceedings of the 13th international conference on World Wide Web*, ACM, pp.

Vol.2014-DBS-160 No.20
Vol.2014-OS-131 No.20
Vol.2014-EMB-35 No.20
2014/11/18

595–602 (2004).
[13] シュウテイ，　片山薫：算術符号を用いたウェブグラフ表現のための圧縮方法，*The 6nd Forum on Data Engineering and Information Management*, DBSJ, pp. D6–1 (2014).
[14] Zhao, W., Martha, V. and Xu, X.: PSCAN: A Parallel Structural Clustering Algorithm for Big Networks in MapReduce, *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, IEEE, pp. 862–869 (2013).
[15] 塩川浩昭，藤原靖宏，鬼塚　真：構造的類似度に基づくグラフクラスタリングの高速化，*The 6th Forum on Data Engineering and Information Management*, DBSJ, pp. D6–2 (2014).
[16] Lin, J. and Dyer, C.: Data-intensive text processing with MapReduce, *Synthesis Lectures on Human Language Technologies*, Vol. 3, No. 1, pp. 1–177 (2010).