2N-4

大規模なXML類似検索のためのMapReduceによる並列化手法の提案

VU TuanDat† 渡辺 陽介‡ 横田 治夫††

† 東京工業大学 工学部 情報工学科 ‡ 東京工業大学 学術国際情報センター
†† 東京工業大学 大学院情報理工学研究科 計算工学専攻

1 はじめに

近年、企業や組織において Office Open XML, XHTML などで記述された XML ファイルが多く利用されるようになった．そのような XML ファイル増加にと伴い、キーワード検索や XPath 検索だけでなく類似文書検索のような複数の XML 木の中から類似した XML 木を取り出すことが重要になっている．既存の精度が高く高速な類似度計算アルゴリズムとして LAX[2] があるが、それでも LAX は $O(n^2)$ で膨大なデータに対して処理時間がかかる．そこで本稿では、大量の XML 木に対応するため、MapReduce フレームワーク [1] を用いて LAX を並列化する．

MapReduce は map 処理と reduce 処理から構成される．Map 処理は key-value ペアを受け取り、新たな key-value ペアを生成する．Reduce 処理は map から取った value を key ごとにグループ化し、内部で処理した結果を新たな key-value ペアを出力する．

LAX では XML の部分木に含まれる同じ値を持つリーフノードのカウントによって文書の類似度を判定する．そのため、提案手法では XML のリーフノードのデータを key、その key を含む部分木を value とし、MapReduce によって部分木ペアの類似度を集計し、並列化を行う．また、実際の XML ファイルを利用して提案手法の処理効率を評価する．

2 LAX による XML 木間の類似度計算

LAX は与えられた 2 つの XML 木の T_b(base), T_t(target) に対して類似度 $Sim(T_b, T_t)$ を求めるアルゴリズムである．LAX の処理手順を図 1 に示し，以降では各手順の詳細を述べる．

1. **部分木の生成**：LAX では XML の木構造を部分木に切り分ける点を *SegmentationSpot* と呼んでいる． XML 木内のあるノード x が，どれだけ

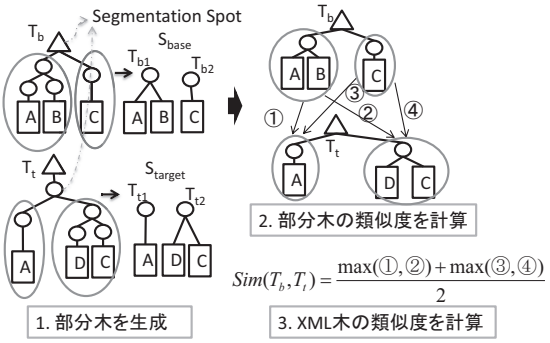

図 1: LAX

SegmentationSpot に相応しいかを表す値 W は $W = n \times d^{\phi}$ の式で表される．ただし，n はノード x の子ノードの数，d はノード x からリーフノードまでの最大距離，ϕ は重み付け調節できる定数を表わしている．base と target の XML 木それぞれに対して，全ノード中最も大きい W_x を持つノード x を見つけ，x の子ノードをルートノードとした部分木の集合に元の XML 木を分割する．

2. **部分木の類似度を計算**：T_b の部分木集合 S_{base} と T_t の部分木集合 S_{target} のそれぞれの要素同士 $(T_{bi}, T_{tj}) \in (S_{base} \times S_{target})$ で，部分木としての類似度を求める． XML 木 T_b の部分木 T_{bi}, XML 木 T_t の部分木 T_{tj} の類似度を計算する 関数 $Sim_{sub}(T_{bi}, T_{tj})$ は以下の式で定義される．

$$Sim_{sub}(T_{bi}, T_{tj}) = \frac{|MatchedLeaf(T_{bi}, T_{tj})|}{|Leaf(T_{bi})|} \times 100(\%) \quad (1)$$

ただし，$|Leaf(T_{bi})|$ は部分木 T_{bi} に属するリーフノード数、$|MatchedLeaf(T_{bi}, T_{tj})|$ は T_{bi}, T_{tj} に属するリーフノードの中で，PCDATA が同じペアを作った場合のペア数とする．

3. **XML 木間の類似度**：部分木同士の類似度から XML 木全体の類似度を求める．XML 木 T_b, T_t の類似度 $Sim(T_b, T_t)$ は，各 XML 木の部分木集合 S_{base}, S_{target} から以下の式で定義される．

$$Sim(T_b, T_t) = \frac{\sum_{T_{bi} \in S_{base}} (max_{T_{tj} \in S_{target}} (Sim_{sub}(T_{bi}, T_{tj})))}{|S_{base}|} \quad (2)$$

A parallel method for large-scale XML similarity search using mapreduce

TuanDat VU†, Yousuke WATANABE‡and Haruo YOKOTA††
†Department of Computer Science, School of Engineering, Tokyo Institute of Technology
‡Global Scientific Information and Computing Center, Tokyo Institute of Technology
††Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
{dat, watanabe}@de.cs.titech.ac.jp yokota@cs.titech.ac.jp

3 提案並列化手法

節2では2つのXML木の類似度計算を説明した．本節は複数のXML木に対する類似度計算を述べる．

XML木の類似度を計算するとき、LAXのステップ1でXML木を解析（部分木生成）しなければならない．複数ファイルに対してはXMLファイルを各Mapタスクに分割し、分散処理を行う．

また、ステップ2で部分木の類似度を計算するとき、同じ値を持つリーフノードをカウントする必要がある．そして、LAXの処理（ステップ2）ではS_{base}とS_{target}の部分木同士のすべての類似度を計算しなければならないので、カウント時間がかかる．そこで、リーフノードのデータをkey、そのkeyを含む部分木をvalueとし、MapReduceによってリーフノードをカウントするときに並列化を行う．

同じ値を含むリーフノードの部分木リストがあっても、部分木ペアごとにグループ化しなければならないのでMapReduce2段で行う必要がある．まず、1段目では木集合同じ値を持つリーフノードを含む部分木をグループ化して、部分木ペアとして出力する．次に、2段目では部分木のペアを集計して、部分木ごとの類似度を計算して、XML木の類似度を計算する．ただし、本手法内で用いられる部分木オブジェクトは（XML木Id、XML木の部分木数、部分木Id、部分木のリーフノード数）のみを持つものとする．Base, Targetの二つのXML木集合に対して提案手法のMapReduceの手順は以下のように行われる．

1. 一段目のMap
   入力：key:XML木Id, value:XML木
   出力：key:リーフノードのデータ, value:部分木
   処理：XML木を解析し（LAXのステップ1)、各リーフノードに対して、データとリーフノードを含む部分木を出力する．

2. 一段目のReduce
   入力：key:リーフノードのデータ, value:部分木のリスト
   出力：key:Baseの部分木, value:Targetの部分木
   処理：部分木のリストから同じリーフノードの値をkeyとするbase, target　の部分木ペアを生成する．これはLAXのステップ2において、同じ値を含むリーフノードを持つ部分木のペア(T_{bi}, T_{tj})のMatchedLeaf集合の要素を求める．

3. 二段目のMap
   入力：key:Baseの部分木, value:Targetの部分木
   出力：key:XML木Idペア, value:部分木ペア
   処理：Baseの部分木とTargetの部分木からそれぞれの部分木を含むXML木のIdを取り、ファイルIdペアをkey、部分木ペアをvalueとして出力する．

4. 二段目のReduce
   入力：key:XML木Idペア, value:部分木ペアのリスト
   出力：key:XML木Idペア, value:類似度
   処理：部分木ペアのリストからLAXのステップ2において、各部分木ペアの同じ値を含むリーフノードのMatchedLeaf集合の要素数をカウントし、部分木の類似度を計算する．そして、その結果からXML木の類似度を計算する（LAXのステップ3)．

4 実験

提案手法は大規模なXML木に対して効果があるかを調べるための実験を行った．実験の環境として、OS Ubuntu 11.10, CPU Intel Xeon E5620OS 2.40GHz/8コア, メモリ24GBのマシン4台を利用してHadoop version 1.0.4でMapReduceを実行した．XMLデータはDBLPのレコード[3]から取り、小さなXML木に分けたものを用いた．一つのXML木は、部分木数が100、リーフノード数は約900とした．

Base集合のXML木数は100とし、Target集合のXML木数を増やした場合の実行時間を図2に表す．

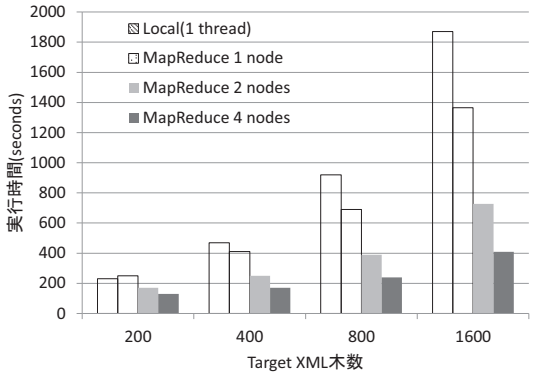

図 2: 実験結果

図より、XML木が少ない場合 (木数200)、実行時間にはあまり差が見られないが、XML木数が増加した場合にはマシンの台数に応じて実行時間が短縮できてる．

5 まとめ

本研究では、MapReduceを用いてリーフノードのデータをkey, 部分木をvalueとしてLAXを並列化し、大量なXML木から類似したXML木を検索手法を示した．実験によりXML木が多い場合MapReduceによる並列化の効果が見え、提案手法が大規模なXML木集合に対して適用可能であることが確認できた．今後は、他の手順による並列化も検討する．

参考文献

[1] Apache Hadoop. http://hadoop.apache.org

[2] W.Liang & H.Yokota. LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration. In Proc. of BNCOD, pp.82-97, 2005

[3] DBLP in XML. http://dblp.uni-trier.de/xml/