木構造データの縮約アルゴリズムの Hadoop 上での実現とその評価

1130320　尾形 勇磨　【 松崎研究室 】

1　はじめに

木構造は，XML 文書やファイルシステム等で用いられているデータ構造である．近年取り扱われるデータの容量は肥大化しており，それは木構造を持った XML 文章等でも例外ではない．しかし，現状の CPU やメモリでは取り扱えるデータ量や処理速度に限界があり、1 台のコンピュータ機器を用いての処理では高速化は困難である．そこで大規模クラスタ向け並列計算フレームワークである MapReduce [1] などの並列処理を行う技術が求められているが，木構造処理のようなデータ同士の依存性を利用する処理には向いていないという問題がある．そこで本稿では，木構造を分割しやすい形にすることで，MapReduce の実装である Hadoop による並列な木構造処理を実現する．そして，どの程度効果があるのかを検証する．

2　Hadoop

Hadoop は Google によって提案された MapReduce のプログラミングモデルとフレームワークの実装である．MapReduce では map と reduce という 2 つの関数を用いて MapReduce プログラミングを行う．MapReduce フレームワークによって入出力データの受け渡しは自動的に行われるため，プログラマは map と reduce の処理を記述することにより並列プログラムを実現できる．

3　木構造データのリスト表現と縮約

木の縮約 [2] は，葉ノードを親ノードにマージしていき，根ノードが残るまで繰り返すことである．この実験では木の縮約の一つである maxPathSum と呼ばれるものを使用する．これは任意の葉ノードから根ノードまでのパスに現れる値をすべて足し合わせた値のうちの最大値を求める計算である．例えば図 1 の木の maxPathSum は 58 となる．

その計算のために木の深さを表す整数とそのノードの値 (open 要素) もしくは値がないことを示すタグ (close 要素) のペアによるリスト構造で木構造データを表現する．図 1 で示す木をリスト表現にすると

$$[(0,21),(1,17),(2,8),(2,/),(1,/),(1,13),$$

$$(2,24),(2,/),(2,16),(2,/),(1,/),(0,/)]$$

のようになる．木のリスト表現では，部分木が親の区間に含まれているという特徴がある．この事から対応する open 要素と close 要素を持った部分リストが与えられた場合，対応する部分木の縮約が行えることが分かる．

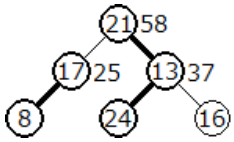

図 1　ノード 6 の木構造データの例

4　縮約アルゴリズムの MapReduce 上での実装

リスト表現された木構造データを MapReduce を用いた並列処理によって縮約する．MapReduce にはその実装である Hadoop を用いた．木構造データの縮約アルゴリズムを MapReduce に実装する際に，map 処理ではデータ間の依存性のない部分リスト毎の縮約を行い，reduce 処理では縮約された部分リストをマージすることによって木全体の縮約を行った．

このプログラムを用いて 16 台のコンピュータ機器で構成されたクラスタで台数効果を調べる評価実験を行った．また，map 処理に渡す際の入力データの分割数を変えることによる実効速度の変化を検証した．

5　実験結果と考察

表 1 で示すように使用するコンピュータ機器を増やす事によって，処理を高速化することができた．また，表 2 で示すように並列化するために入力データを分割する際には，分割数を減らし過ぎると map タスク一つ当たりの処理が大きくなり，処理時間が遅くなるという結果を得た．この実験の結果から分割されたリストのノード数が 10^3 前後となるように分割することが効率的であることが分かった．

台数	1	2	4	8	16
実行時間 (秒)	134	82	59	46	36

表 1　ノード数 10^8 分割数 10^3 の入力データに対する台数毎の実行時間

分割数	25	50	10^2	10^3	10^4	10^5
実行時間 (秒)	240	95	48	38	35	34

表 2　8 台でノード数 8×10^7 の入力データを処理した時の分割数毎の実行時間

参考文献

[1] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In OSDI, pp. 137–150, 2004.

[2] 井町 宏人．準構造化データ処理の効率的 MapReduce 実装に関する研究．東京大学大学院情報理工学系研究科数理情報学専攻．2012 年．