6J-09

クエリ分割並列化による並列 XPath クエリの XML データベース BaseX における評価

仮谷 拡晃† 松崎 公紀†

高知工科大学情報学群†

1. はじめに

近年では多くのデータが，XML や JSON などの木構造データ（半構造化データ）として管理されており，それらのデータは増大している．また，マルチコアプロセッサが広く利用できるようになったため，並列クエリによる木構造データの効率化の重要度が高まっている．

このような背景のもと，Bordawekar ら[1]は XML データ処理の中心である XPath クエリの並列化手法として，「データ分割並列化」と「クエリ分割並列化」の 2 種類の並列化手法を提案した．そのうち，データ分割並列化については，2018 年に Sato ら[2]によって最新の実用的な XML データベースである BaseX 上で評価・考察が行われ，その有効性や課題が示された．本研究では，もう一つの並列化手法であるクエリ分割並列化について，同じく BaseX 上で評価・考察を行う．

2. クエリ分割並列化

クエリ分割による並列化とは，与えられたクエリを述語・インデックスを利用してサブクエリへ分割する手法である．分割されたサブクエリは，複数のスレッドで並列に実行される．このとき，サブクエリはデータ全体に対して適用され，データを事前に分割する前処理を必要としない点が重要である．

例えば，述語を利用したクエリ分割では，$/q_1[q_2 \text{ or } q_3]$ というクエリが与えられた際に $/q_1[q_2]$ と $/q_1[q_3]$ という 2 つのクエリへ分割する．インデックスを利用したクエリ分割では，$/q_1/q_2/q_3$ が与えられた際に，$/q_1/q_2[\text{position}() <= (n/2)]/q_3$ と $/q_1/q_2[\text{position}() > (n/2)]/q_3$ のようなサブクエリに分割する．ここで n は，事前に軽量なクエリを実行して調査するか，BaseX がもつ統計情報などを利用して得られると想定している．後者のインデックスを利用したクエリ分割並列化では，サブクエリを割り当てられた複数のスレッドは，論理的に独立な部分に対してクエリ処理を行う（図 1）．

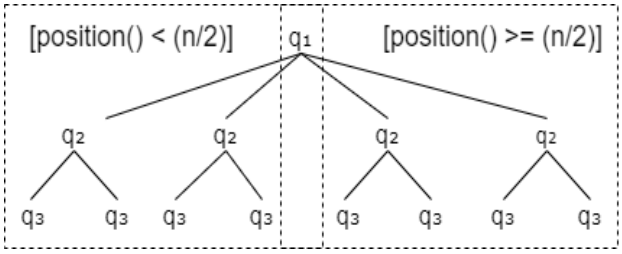

図 1. クエリ分割並列化の実行例

3. 実験

3.1. 実験設定

XML データベース BaseX 上でのクエリ分割並列化による並列 XPath クエリの性能を評価する実験を行った．実験は CPU に AMD Ryzen 7 3800X（8 コア 16 スレッド）を，メモリを 128 GB もち，ubuntu 20.04.1 LTS，OpenJDK(version 1.8.0 275)，BaseX 9.0.1 をインストールした計算機を用いて行った．

XML データセットには DBLP と XMark を使用した．DBLP データセットはウェブサイト [1] より 2020 年 10 月 2 日にダウンロードしたものであり，ファイルサイズは 2.79GB，ノード数は約 6880 万ノードである．XMark はウェブサイト [2] よりダウンロードしたものであり，ファイルの日付は 2008 年 1 月 7 日であり，ファイルサイズは 113MB，ノード数は約 322 万ノードである．

実験には表 1 に示した XPath クエリを使用した．クエリを分割する際に使用する n の値は，事前に count()によるクエリを行って表 1 に示す値を得た．Java プログラムから各クエリを発行した結果を BaseX サーバからバイトストリーム形式で受け取るまでの時間を測定した．実行時間には，ウォームアップ実行後の 25 回の実行時間の中央値を選択した．

Evaluation of Parallel XPath Queries by Query-partitioning Technique on XML Database BaseX
Hiroaki Kariya, Kochi University of Technology †
Kiminori Matsuzaki, Kochi University of Technology†

[1] https://dblp.uni-trier.de/xml/

[2] http://xmlcompbench.sourceforge.net/Dataset.html

表 1. 実験に使用したクエリ

Key	クエリ	分割点の n の値
DB1	/dblp/article/author	2380490
DB2	/dblp/inproceedings/author	2680964
DB3	/dblp/*/title	7881980
XM1	/site/*	6
XM2	/site/open_auctions/open_auction	12000

表 2. 実行結果（単位はミリ秒）括弧内は $p = 1$ に対する速度向上比

Key	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 6$	$p = 8$
DB1	4229	2602 (1.63)	1875 (2.26)	1536 (2.75)	1285 (3.29)	1103 (3.83)
DB2	5072	3070 (1.65)	2197 (2.31)	1780 (2.85)	1450 (3.50)	1289 (3.93)
DB3	8232	4588 (1.79)	3839 (2.14)	3088 (2.67)	2353 (3.50)	2016 (4.08)
XM1	1382	718 (1.92)	695 (1.99)	708 (1.95)	715 (1.93)	
XM2	345	182 (1.90)	136 (2.54)	130 (2.65)	124 (2.78)	108 (3.19)

3.2 実験結果

実行時間を測定するために DB1, DB2, DB3, XM2 に対して $p = 1, 2, 3, 4, 6, 8$ スレッド, XM1 に対して $p = 1, 2, 3, 4, 6$ スレッドで実行した. 表 2 は, 全てのサブクエリの結果が得られるまでに要した時間を示したものである.

各クエリについて, 予備クエリにより得た n の値と position()関数を利用してサブクエリへ分割し並列実行した結果, DB1, DB2, DB3, XM2 ではスレッド数の増加に伴い性能が向上した. DBLP データセットを対象とした DB1, DB2 では 8 スレッド実行時に 3.8 倍以上の高速化という結果が得られ, DB3 では 4.08 倍の高速化を達成した.

XMark データセットを対象とした XM1 では 2 スレッドでの実行時に逐次実行時に比べ 1.92 倍の高速化を達成しているが, 3 スレッド以上の実行ではスレッド数を増加させても実行時間にほとんど変化はなかった. 実行時間に変化がない要因として, XM1 では分割点におけるノード数が少なく, スレッド間での処理に不均衡が生じているためと考えられる. XM2 ではスレッド数の増加に伴い性能が向上しており, 最大で 3.19 倍の高速化を達成している.

ここで得られた速度向上は, Sato らが行ったデータ分割並列化による並列化と比べても大きく劣ることはなかった. しかし実行した計算機環境が異なるため, 単純な比較ができないことには注意が必要である. 事前のデータ分割を必要としないことから, クエリ並列化は十分に有効性があるのではないかと考える.

4. まとめ

本論文では, Bordawekar らが提案した XPath クエリ並列化手法のうち,クエリ分割並列化について最新の XML データベース BaseX を用いて評価を行った. いくつかのクエリに対して実行時間を測定した結果, ほとんどのクエリにおいて性能を向上させることが確認できた. 特に DBLP データセットを対象としたクエリでは, 8 スレッドを用いて 3.8 倍以上の速度向上を達成できることが分かった.

現時点では XPath クエリの サブクエリへの分割は手作業で行っており, データベースの統計情報を用いた自動クエリ分割は重要な今後の課題である. また, 単純にノード数を均等に分割すればよいかは今後より深く検討する必要がある.

参考文献

[1] R. Bordawekar., L. Lipyeow, O. Shmueli, Prallelization of XPath Queries using Multi-core Processors: Challenges and Experiences, Proceedings for the 12th International Confernce on Extending Database Technology (EDBT'09), pp. 180-191 (2009).

[2] S. Sato, W. Hao, K. Matsuzaki, Parallelization of XPath Queries using Modern XQuery Processors, New Trends in Databases and Information Systems, pp. 54-62 (2018).