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

1210310　仮谷 拡晃　【 高度プログラミング究室 】

1　はじめに

近年では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，XML や JSON などの木構造データ処理を並列クエリによる効率化することの重要度が高くなっている．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「クエリ分割並列化」と「データ分割並列化」を提案した．

本研究では「クエリ分割並列化」について最新の実用的な XML データベース BaseX 上で評価・考察を行う．

2　クエリ分割並列化

クエリ分割並列化とは，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し複数スレッドによって並列に実行する手法である．

例えば，「/q_1/q_2/q_3」というクエリが与えられた際，インデックスを利用したクエリ分割並列化では「/q_1/q_2[position() <= (n/2)]/q_3」と「/q_1/q_2[position() > (n/2)]/q_3」という 2 つのサブクエリへ分割する．ここで n は，事前に軽量なクエリを用いて調査するか，XML 文書の統計情報を利用して得られることを想定している．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が論理的に独立した部分に対してクエリ処理を行う (図 1)．

3　実験

BaseX 上でクエリ分割による並列 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 を持った計算機と，XML データセットに DBLP と XMark1, XMark2 を用いて行った．

実験には表 1 に示すようなクエリを使用した．クエリを分割する際に使用する n の値は事前に count によるクエリを実行し，得られた値を使用する．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．

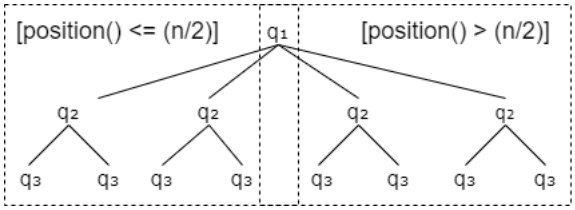

図 1　インデックスを利用したクエリ分割並列化

表 1　実験に使用したクエリ (一部抜粋)

Key	クエリ
DB1	/dblp/article/author
DB2	/dblp/inproceedings/author
DB3	/dblp/*/title
DB4	/dblp/book[count(./following-sibling ::book[1]/author) < count(./author)]

表 2　$p = 1$ に対する速度向上比

Key	$p = 2$	$p = 3$	$p = 4$	$p = 6$	$p = 8$
DB1	1.63	2.26	2.75	3.29	3.83
DB2	1.65	2.31	2.85	3.50	3.93
DB3	1.79	2.14	2.67	3.50	4.08
DB4	0.91	0.93	0.92	0.92	0.91

実行時間を測定するために各クエリに対して $p = 1, 2, 3, 4, 6, 8$ スレッドで実行した．表 2 は逐次実行に対する速度向上比を示したものである．各クエリに対して分割点のノード数と position 関数を用いてサブクエリへ分割し並列実行した結果，DB1，DB2 では適切なクエリ分割が行われ 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た．DB3 では 2 スレッド実行時に 1.79 倍，8 スレッドで 4.08 倍の高速化を達成し今回の実験でクエリの中で最も並列化の恩恵を受けていた．DB4 では逐次実行に対して 0.91〜0.93 倍と性能の向上が見られなかった．その要因として，クエリ分割並列化では following-sibling 軸にって引き起こされる冗長なアクセスが解消できていないためと考えられる．

4　まとめ

本稿では Bordawekar らが提案した「クエリ分割並列化」について BaseX 上で評価を行った．いくつかのクエリを対象に実験を行った結果，ほとんどのクエリで性能を向上させることが確認できた．

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).