맵리듀스에서 중복기반 조인과 비상충 조인을 이용한 효율적인 SPARQL 질의 처리

(Efficient SPARQL Query Processing using Duplication-based and Non-conflicting Joins on MapReduce)

김 태 경 † 김 기 성 † 김 형 주 ††
(Tai Kyoung Kim) (Kisung Kim) (Hyoung-Joo Kim)

요 약 최근, 분산 병렬 프레임워크인 맵리듀스를 이용한 SPARQL 질의 처리에 대한 연구가 진행되고 있다. 맵리듀스를 이용해 SPARQL 질의 처리를 하기 위해서는 여러 맵리듀스 잡이 필요하며, 이로 인해 많은 비용이 들게 된다. 최근의 연구들은 대부분 이 맵리듀스 잡의 개수를 줄이는데 초점을 맞추고 있다. 본 논문은 SPARQL 질의 처리시에 맵리듀스 잡의 개수를 줄이기 위한 두 가지 서로 다른 기법을 혼용 할 것을 제안한다. 우리가 적용한 기법은 서로 관련이 없는 조인 키들을 동시에 하나의 맵리듀스 잡에서 수행하는 비상충 조인과 중복을 이용해서 여러 개의 조인 키를 한번에 조인하는 멀티웨이 조인 기법이다. 이 두 가지 기법을 혼용함으로써, 기존에 제안된 기법보다 적은 수의 맵리듀스 잡을 이용해 질의를 처리 할 수 있다. 또한, 이로 인해 발생하는 트리플 패턴 그룹화 문제에 대한 그리디 알고리즘을 제안한다. 우리는 대용량 RDF 데이터를 이용한 실험을 통해 제안하는 알고리즘이 기존 연구보다 맵리듀스 잡의 개수를 줄일 수 있으며, 질의 처리 성능을 향상시킴을 보인다.

키워드 : 맵리듀스, SPARQL, RDF, 중복기반 조인, 멀티웨이 조인

Abstract Recently, there has been a lot of research about SPARQL query processing using MapReduce, a parallel distributed framework. To process a SPARQL query on MapReduce, in general, several MapReduce jobs are required, and these jobs cause additional costs. Therefore, most research has been focused on reducing the number of MapReduce jobs. In this paper, we propose to hybridize two different techniques to reduce the number of MapReduce jobs for processing SPARQL queries. The techniques we hybridize are the non-conflicting join and the duplication-based multi-way joins. The non-conflicting join can process independent joins in one MapReduce job, and the multi-way joins can join many join keys at once by duplicating data. Also, we present a greedy algorithm to solve the triple pattern grouping problem which occurs when hybridizing two techniques. We demonstrate that our framework can reduce MapRedcue jobs, and performs better than the previous approaches, through experiments on the large RDF data.

Key words : MapReduce, SPARQL, RDF, Duplication-based join, Multi-way join

· 연구는 BK-21 정보기술 사업단 및 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원(No. 20110017480)을 받아 수행되었음

† 비 회 원 : 서울대학교 컴퓨터공학부
kimtk@idb.snu.ac.kr
kskim@idb.snu.ac.kr
(Corresponding author임)
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr

논문접수 : 2012년 1월 4일
심사완료 : 2012년 5월 28일

정보과학회논문지: 데이타베이스 제39권 제4호(2012.8)

1. 서 론

RDF(Resource Description Framework)[1]는 시맨틱 웹을 위한 데이터 모델이며, RDF 데이터는 W3C가 권고하는 질의 언어인 SPARQL(SPARQL Protocol And RDF Query Language)[2]로 질의한다. 최근 RDF 데이터가 폭발적으로 증가하고 있으며, 이에 대용량 RDF 데이터 처리에 대한 관심이 증가하고 있다. 특히, SPARQL 질의 처리의 확장성(scalability) 문제를 해결하기 위해, 많은 연구들이 맵리듀스(MapReduce)[3]를 사용한 분산 병렬 SPARQL 질의 처리 기법을 제안하고 있다.

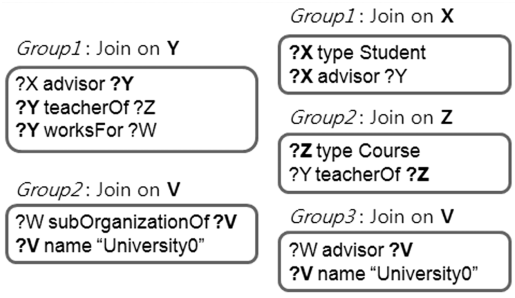
맵리듀스를 이용한 SPARQL 질의 처리시 발생하는 가장 큰 문제는 하나의 질의를 처리 하기 위해 여러 맵리듀스 잡을 수행해야 하는 것이다. 맵리듀스 잡은 맵과 리듀스 작업의 묶음을 통칭한다. SPARQL 질의에 포함된 트리플 패턴의 개수에 따라 필요한 조인 연산의 수도 증가하기 때문에 대부분의 SPARQL 질의는 많은 조인 연산을 필요로 한다. 트리플 패턴이란 변수를 포함한 RDF 트리플을 말한다. 또한 일반적으로 하나의 조인을 처리 하기 위해서는 하나의 맵리듀스 잡을 생성하게 되고 이로 인해 여러 개의 맵리듀스 잡이 생성되게 된다. 맵리듀스 잡의 개수 증가는 여러 가지 부가 비용의 증가를 의미한다. 이런 비용에는 맵리듀스 잡의 시작 비용, 중간 결과의 분산파일시스템 실체화(materialization) 비용, 중간 결과를 읽기 위한 I/O 비용 등이 있다. 특히, 매우 많은 맵리듀스 잡을 사용하고, 많은 중간 결과를 생성하는 질의의 경우에는 이런 맵리듀스 잡 자체의 비용으로 인해 질의 성능에 큰 악영향을 주게 된다.

최근 HadoopRDF[4]는 비상충 맵리듀스 조인(non-conflicting MapReduce join)을 이용해 맵리듀스 잡의 개수를 줄이는 방법을 제안했다. 서로 관련 없는 조인 연산들을 하나의 맵리듀스 잡에서 수행하는 것을 비상충 맵리듀스 조인이라 한다. 예를 들어, 그림 1의 SPARQL 질의를 보자. 이 SPARQL질의에서 1번과 4번 트리플 패턴을 X에 대해 조인하는 연산과 3번과 7번 트리플 패턴을 V에 대해 조인하는 연산은 서로 다른 변수에 대한 조인이기 때문에 상충하지 않으며 하나의 맵리듀스 잡에서 수행될 수 있다. 즉, 리듀서들을 분할해 각 변수에 대한 조인을 처리하도록 하면 한 맵리듀스 잡에서 여러 조인을 처리할 수 있다.

비상충 맵리듀스 조인을 사용하면 트리플 패턴 그룹화 방법에 따라 필요한 맵리듀스 잡의 개수가 달라진다. 트리플 패턴 그룹화란 같은 조인키를 갖는 트리플 패턴을 묶는 것을 말한다. 위의 예의 경우 그림 2와 같은 두 가지 트리플 패턴 그룹화 방법이 존재하며, 각각의 경우

```
SELECT ?X ?Y ?Z WHERE {
1    ?X type Student .
2    ?Z type Course .
3    ?V name "University0" .
4    ?X advisor ?Y .
5    ?Y teacherOf ?Z .
6    ?Y worksFor ?W .
7    ?W subOrganizationOf ?V
}
```

그림 1 예제 SPARQL 질의



(a) 케이스 1 (b) 케이스 2

그림 2 그룹화 예

는 총 4개, 3개의 맵리듀스 잡을 통해 수행할 수 있다(4절에서 자세히 설명). 이 경우 3개의 맵리듀스 잡을 사용하는 경우가 4개의 잡을 사용하는 경우보다 효율적으로 질의를 처리할 수 있게 된다. 이와 같이, HadoopRDF에서는 질의 처리 성능을 개선하기 위해 최소의 맵리듀스 잡을 필요로 하는 트리플 패턴 그룹을 찾는 방법을 제안하였다.

그러나 HadoopRDF는 한 트리플 패턴 그룹에 조인키 한 개만을 고려하는 한계를 가진다. 즉, 각 트리플 패턴 그룹은 하나의 조인키 만을 조인할 수 있다. 하지만, 최근에 제안된 중복 조인 기법[5]을 이용하면, 데이터를 중복 전송함으로써 두 개의 조인키도 한 잡에서 조인할 수 있다. 따라서 중복 조인을 이용하면, 기존에 한 개의 조인키 만을 고려했을 때보다 잡의 개수를 줄일 수 있게 된다.

이에 본 논문에서는 맵리듀스를 이용한 SPARQL 질의 처리에 중복 조인을 사용할 것을 제안한다. 그러나 중복 조인을 적용하면, 기존의 트리플 그룹화 방법을 그대로 이용할 수 없다. HadoopRDF의 트리플 그룹화 기법은 조인키 한 개만을 고려했기 때문에, 트리플 그룹화 기법을 확장해 조인키 두 개를 다룰 수 있도록 해야 한다. 이를 해결하기 위해, 우리는 중복 조인을 고려한 트리플 패턴 그룹화 방법을 제안한다. 이는 기존의 HadoopRDF

의 트리플 패턴 그룹화 기법을 조인키 두 개를 고려하도록 확장한 것으로 이해할 수 있다. 이와 같이 중복 조인을 SPARQL 질의 처리에 적용하고, 트리플 패턴 그룹화 문제를 해결함으로써 우리는 HadoopRDF 보다 더 적은 수의 맵리듀스 잡으로 질의를 처리할 수 있다.

본 논문의 구성은 다음과 같다. 2장과 3장에서는 맵리듀스를 이용해 RDF 데이터를 다루는 관련 연구들과 SPARQL 질의를 맵리듀스로 처리하기 위한 배경지식을 살펴본다. 4장에서는 본 논문의 타겟 시스템인 Hadoop-RDF와 그 한계를 알아본다. 5장에서는 우리가 제안하는 중복 조인을 이용한 트리플 패턴 그룹화 방법을 설명하고, 6장에서는 LUBM[6] 데이터와 LUBM 질의를 가지고 우리가 제안하는 알고리즘이 HadoopRDF보다 성능향상을 가져왔음을 보인다. 마지막 7장에서는 결론과 향후 연구과제를 논한다.

2. 관련연구

RDF 데이터가 급격하게 증가함에 따라 이를 분산 및 병렬로 처리하려는 연구들이 활발히 이루어지고 있다. RDF 데이터를 하나의 머신에서 저장 및 검색하는 기존 시스템들(RDF-3X[7], Jena[8], BigOWLIM[9], Sesame[10])은 확장성의 한계가 있다[1][4]. 이들 시스템들은 하나의 머신에 있는 자원만을 사용하기 때문에 대용량 데이터에 대해서는 메모리 부족 또는 수행 시간의 급격한 증가 문제가 따른다. 따라서 많은 연구들이 확장성의 문제를 분산 및 병렬로 처리함으로써 해결하고 있으며, 그 중 가장 대표적인 프레임워크인 맵리듀스가 많이 이용되고 있다. [11]은 맵리듀스 잡을 자동으로 생성해주는 상위 언어인 Pig Latin[12]을 이용하며 이를 위해 SPARQL을 Pig Latin으로 변환하는 방법을 제안하였고, [13]은 Pig Latin과 UDF를 이용하여 RDF 그래프에 대한 분석적 질의를 처리하는데 I/O 비용을 줄인 방법을 제안하였다. [14]는 대용량 RDF 데이터에 대한 추론을 맵리듀스를 이용하여 처리하는 방법을 제시하였다. SPIDER[15]는 Hadoop[2]과 HBase[3]를 이용하여 RDF 데이터를 분산 시키고 질의를 분해해 각 노드로 전송한 후 처리된 결과를 다시 모으는 시스템을 제안하였다.

[16,4]는 맵리듀스를 이용하여 SPARQL 질의 처리를 할 때 맵리듀스 잡의 개수를 줄이기 위해 어떠한 조인키를 먼저 조인할 것인지에 대한 방법을 제시하였다. [16]은 SPARQL 질의의 트리플 패턴들에 가장 많이 나타나는 조인키를 먼저 조인하는 그리디 방법을 제시한 반면, [4]는 서로 독립적인 조인이 많이 이루어지도록 하는 조인키를 먼저 조인하는 휴리스틱 방법을 제안하였다. [4]에서 제안한 방법은 [16]에서 제안한 그리디 방법을 개선시키는 방법으로 우리는 [4]에 대해서만 4절부터 비교를 통해 좀더 자세히 살펴보도록 한다.

[5]는 SPARQL이 아닌 다른 도메인에서 맵리듀스 잡의 개수를 줄이기 위해 다수의 조인 연산을 데이터를 중복시킴으로써 한번에 처리하는 방법을 제안하였다.

3. 배경 지식

이 절에서는 배경지식을 설명하며 본 논문에서 사용하는 용어들을 정의한다. 3.1절에서는 RDF와 SPARQL에 대해 알아보고, 3.2절에서는 맵리듀스에서 2-웨이 조인을 이용한 나이브 RDF 질의처리를 설명한다.

3.1 RDF와 SPARQL

RDF 데이터는 <주어, 술어, 목적어>로 구성되는 트리플의 집합이다. RDF 데이터는 레이블이 있는 방향성 그래프로 표현 할 수 있으며, 이 그래프를 RDF 그래프라 한다. RDF 그래프의 정점은 RDF 트리플의 주어, 목적어에 해당된다. 각 트리플은 RDF 그래프의 간선이 되는데, 주어 정점에서 목적어 정점을 연결하며, 트리플의 술어는 간선의 레이블이 된다.

SPARQL 질의는 그래프 패턴을 가지며, 그래프 패턴은 트리플 패턴의 집합으로 표현할 수 있다. 트리플 패턴은 RDF 트리플과 형태가 비슷하지만 검색을 원하는 개체가 변수(예: ?X)라는 점이 다르다. 변수가 될 수 있는 부분은 RDF 트리플의 주어, 술어, 목적어 모두 다 가능하며, 동시에 두 개의 변수도 지정이 가능하다. SPARQL의 where절이 결합하는(conjunctive) 트리플 패턴만으로 구성되면 기본 그래프 패턴(basic graph pattern)이라 한다. 우리는 이 기본 그래프 패턴만을 포함한 SPARQL 질의만을 고려하며, 각 트리플 패턴들은 술어를 제외한 최대 두 개의 변수를 가질 수 있다고 가정한다.

트리플 패턴 중 같은 변수를 포함한 트리플 패턴은 조인되어야 함을 의미하며, 조인에 참여하는 변수를 조인키라고 한다. 여기서 조인키와 SPARQL 질의는 다음과 같이 정의할 수 있다.

정의 1. 조인키(join key)

SPARQL 질의에서 조인키는 두 개 이상의 트리플 패턴에 공통으로 나타나는 변수이며, SPARQL 질의 Q에서 한 트리플 패턴 tp에 대한 조인키 집합을 JoinKey (tp)로 나타낸다. 또한 트리플 패턴 tp에 대해, tp가 조인키 K를 포함하고 있음을 다음과 같이 나타낸다. $K \in tp$.

그림 1의 SPARQL 질의에서 1번과 5번 트리플 패턴의 조인키 집합은 각각 다음과 같다. tp_n은 n번 트리플 패턴을 나타낸다.

1) http://www.w3.org/wiki/LargeTripleStores
2) http://hadoop.apache.org
3) http://hbase.apache.org

JoinKey(tp_1) = {X}, JoinKey(tp_5) = {Y,Z}.

본 논문에서는 SPARQL 질의를 조인키의 관점에서 다음과 같이 정의한다.

정의 2. SPARQL 질의

한 SPARQL 질의 Q는 각 트리플 패턴들의 조인키 집합으로 나타낸다. Q={JoinKey(tp_1), …, JoinKey(tp_n)}.

예를 들어, 그림 1의 SPARQL 질의는 다음과 같이 나타낸다.

Q = {{X}, {Z}, {V}, {X,Y}, {Y,Z}, {Y,W}, {W,V}}

이와 같은 정의는 어떤 형태의 조인키를 가진 트리플 패턴이 있는지에 대한 정보를 제공해준다. 만일 같은 조인키 집합을 가진 여러 트리플 패턴이 있어도 한번만 나타나게 된다. 우리가 제안하는 기법에서는 같은 조인키를 가진 트리플 패턴은 한번에 처리하기 때문에 이와 같이 표현할 수 있다.

3.2 맵리듀스에서의 SPARQL 질의 처리

SPARQL 질의를 맵리듀스에서 처리하는 가장 기본적인 방법은 2-웨이 조인을 이용하는 것이다. 2-웨이 조인을 이용하는 조인을 맵리듀스의 잡으로 구현하면 하나의 조인 연산이 하나의 맵리듀스 잡이 된다.

그림 1의 예제 질의를 2-웨이 조인을 이용한 실행 계획이 그림 3이며, 이 질의 처리에서 각각의 조인을 처리하기 위해서는 6개의 맵리듀스 잡이 필요하다. 주목할 것은 이렇게 구성된 맵리듀스 잡은 한 개의 잡에서 한 개의 조인키 만을 처리한다는 것이다. 그림 3에서 각 조인을 한 개의 맵리듀스 잡으로 만들었기 때문에, 결과적으로 각 맵리듀스 잡은 한 개의 조인키 만을 처리하게 된다. 그림의 왼쪽 상자는 SPARQL질의 Q의 조인키들이 하나의 맵리듀스 잡이 완료될 때마다 남는 조인키들을 나타낸다.

4. HadoopRDF 개괄

이번 절에서는 본 논문의 타겟 시스템인 HadoopRDF [4]에 대해 설명한다. HadoopRDF는 맵리듀스 잡의 개수를 줄이기 위해 하나의 맵리듀스 잡에서 최대한 많은 조인키를 조인함으로써 필요한 잡의 개수를 줄이고자 한다. HadoopRDF에서는 여러 비상충 조인을 한 개의 맵리듀스 잡에서 동시에 처리할 수 있다는 사실을 이용한다. 이를 위해 한 SPARQL 질의를 여러 비상충 조인으로 분할하기 위해 트리플 패턴 그룹화를 수행한다. 트리플 패턴 그룹은 다음과 같이 정의된다.

정의 3. 트리플 패턴 그룹(TPG)

질의 Q에 대해, 트리플 패턴 그룹 $TPG_K(Q)$는 조인키 K를 포함하는 모든 트리플 패턴들의 집합이며, 간단히 조인키 집합들만 나타내어 표현한다.

$$TPG_K(Q) = \{JoinKey(tp_i)\} \mid tp_i \in Q \wedge K \in tp_i\}$$

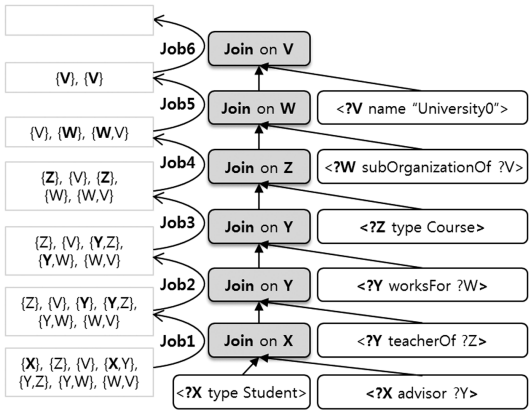


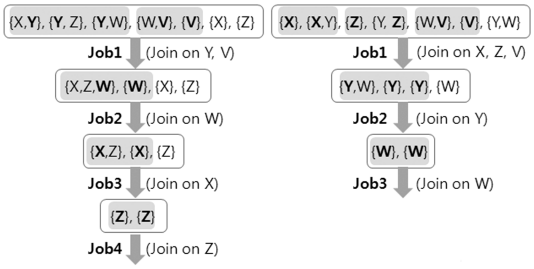
그림 3 2-웨이 조인 실행 계획 예제

트리플 패턴 그룹은 특정 조인키 하나를 포함하는 모든 트리플 패턴의 조인을 의미한다고 볼 수 있다. 이때, 서로 다른 조인키를 갖는 트리플 패턴 그룹은 비상충이다. SPARQL 질의 Q의 각 트리플 패턴을 하나의 트리플 패턴 그룹에만 포함되도록 트리플 패턴 그룹으로 만들면 모든 트리플 패턴 그룹은 비상충이다. 이렇게 비상충 트리플 패턴 그룹으로 나누면 각 트리플 패턴 그룹이 의미하는 조인을 한 맵리듀스 잡에서 수행할 수 있다.

각 트리플 패턴 그룹에서 조인 하려는 조인키의 변수 이름과 값을 맵의 출력 키로 넣으면(예: X#student1) 같은 조인키이면서 같은 값을 가진 데이터는 같은 리듀서로 모이게 되어 조인을 할 수 있게 된다. 비상충 맵리듀스 조인도 맵의 출력 키에 조인키 변수를 넣어주면 여러 개의 조인키도 리듀서가 이를 구분할 수 있기 때문에 조인이 가능하다.

이와 같은 비상충 맵리듀스 조인을 사용하는 경우 트리플 패턴 그룹화 방식에 따라 맵리듀스 잡의 개수가 결정된다. 예를 들면, 그림 1의 질의는 그림 2와 같이 두 가지 트리플 패턴 그룹화 경우가 있다. 각 트리플 패턴 그룹화에 대해 조인을 수행하는 과정은 다음과 같다. 케이스 1은 첫 번째 잡에서 Group1과 Group2를 조인하고, 조인 후 {{X, Z, W}, {W}, {X}, {Z}}의 조인키가 남는다. 이들을 조인하기 위해서는 X, Z, W 각각의 조인키를 조인하는 3번의 잡이 더 필요하며, 총 4번의 잡이 필요하다. 케이스 2는 첫 번째 잡에서 Group1~3의 조인 결과로 {{Y}, {W}, {Y, W}}가 남게 되고, Y, W 각각을 조인하는 두 번의 잡을 더하면 총 3번의 잡이 필요하다. 즉, 케이스 2가 더 적은 수의 잡을 필요로 한다. 그림 4는 이 과정을 보여준다.

이와 같이 한 SPARQL 질의에 대해 최적의 트리플 패턴 그룹을 찾는 문제를 트리플 패턴 그룹화 문제라 한다.



(a) 케이스 1 (b) 케이스 2

그림 4 두 가지 트리플 패턴 그룹의 조인과정

정의 4. 트리플 패턴 그룹화 문제

트리플 패턴 그룹화 문제란 SPARQL 질의 Q에 대해, 맵리듀스 잡의 개수를 최소로 만드는 트리플 패턴 그룹을 정하는 문제다.

HadoopRDF에는 트리플 패턴 그룹화 문제를 풀기 위해 휴리스틱을 사용한 그리디 기법을 제안하였다. HadoopRDF에서는 트리플 패턴 그룹이 조인 후에 남는 조인키를 최소로 만드는 트리플 패턴 그룹을 우선적으로 선택하는 휴리스틱을 사용한다. 이 때, 조인 후에 남는 조인키의 개수를 나타내는 e-count를 사용하며, e-count는 다음과 같이 정의한다.

정의 5. e-count

e-count(K)는 $TPG_K(Q)$가 K에 대해 조인 후 남게 되는 조인키의 개수를 말한다. 예를 들어, 그림 2는 특정 조인키 하나씩을 선택하여 만든 그룹들인데, 각 그룹들은 선택된 조인키에 대해 e-count를 구할 수 있다. 그림 2의 (a)에서 Group1에 대한 e-count(Y)는 Y를 조인 후 {X, Z, W} 세 개의 조인키가 남게 되어 3이다.

HadoopRDF에서는 e-count(K)가 최소인 K를 조인키로 선택하는 그룹을 먼저 생성하도록 한다. 이렇게 e-count가 더 적은 트리플 패턴 그룹을 선택하게 되면 하나의 잡에서 더 많은 조인키를 제거할 수 있다. 이것은 하나의 그룹에 포함되는 조인키들이 적을수록 남은 조인키를 가지고 더 많은 그룹을 만들 수 있게 되고, 더 많은 그룹들이 만들어지면 더 많은 조인키가 제거될 수 있기 때문이다.

HadoopRDF는 이러한 휴리스틱을 이용하여 어떠한 질의가 N개의 트리플 패턴과 K개의 조인키를 가지고 있을 때 다음과 같은 잡의 개수 J에 대한 상계(upper bound)를 보장함을 증명하였다.

$$J = \begin{cases} 0 & N = 0 \\ 1 & N = 1 \; or \; K = 1 \\ \min\left(\lceil 1.71 \log_2 N \rceil , K \right) & N, K > 1 \end{cases} \quad [4]$$

그러나 HadoopRDF에서는 트리플 패턴 그룹을 만들기 위해 조인키를 한 개만 고려하는 한계가 있다. 즉, 하나의 트리플 패턴 그룹은 하나의 조인키 밖에 조인하지 못한다.

5. 중복 조인을 이용한 SPARQL 질의 처리

이 절에서는 우리가 제안하는 중복 조인을 이용한 SPARQL 질의 처리 기법에 대해 설명한다. 이전 절에서 봤듯이, HadoopRDF는 한 트리플 패턴 그룹에서 한 개의 조인키만을 고려하고 있다. 그러나 우리는 다수의 조인키를 한 맵리듀스 잡에서 조인 할 수 있는 중복 조인을 SPARQL 질의 처리에 적용해 필요한 잡의 개수를 더 줄일 수 있다.

5.1 SPARQL 질의 처리에의 중복 조인의 적용

우리는 SPARQL 질의 처리에 중복 조인 기법을 적용하여 한 트리플 패턴 그룹에서 두 개의 조인키를 조인하도록 한다. 이를 위해서 [5]가 제안한 맵리듀스에서 멀티웨이 조인 시 데이터를 중복 전송하여 다수의 조인키를 조인하는 기법을 이용한다.

예를 들어, 다음 3개의 트리플 패턴에서 W와 V에 대한 조인을 하나의 맵리듀스 잡으로 처리하는 과정을 보자.

1. <?Y worksFor ?W>,
2. <?W subOrganizationOf ?V>,
3. <?V name “University0”>

W와 V에 대에 조인 하려면 1번과 2번 트리플 패턴의 W와 2번과 3번 트리플 패턴의 V를 같은 리듀서에 모아야 한다. 그림 5를 보면, 2번 트리플 패턴의 결과인 $<w_0,v_0>$, $<w_0,v_1>$, $<w_1,v_1>$은 각각 하나의 리듀서로 전송된다. 하지만, 1번 트리플 패턴의 결과인 $<y_0,w_0>$는 조인에 참가하기 위해 ‘w_0’가 있는 2, 3번째 리듀서에 중복해서 전송해야 하고, 3번 트리플 패턴의 결과인 $<v_1,"">$은 ‘v_1’이 있는 3, 4번째 리듀서에 중복해서 전송해야 한다. 이렇게 전송하게 되면 그림 5의 예제에서는 3번째 리듀서가 조인에 참가할 모든 트리플들을 가지고 있으므로 조인을 수행한다.

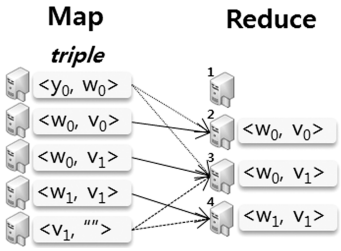


그림 5 중복기반 조인 과정

5.2 중복 조인을 고려한 트리플 패턴 그룹화

우리가 제안하는 트리플 패턴 그룹화 방법은 트리플 패턴 그룹을 만들 때, 두 개의 조인키에 대한 조인을 고려하며 이 두 개의 조인키는 두 개의 조인키를 가진 트리플 패턴의 것으로 선택한다. 두 개의 조인키를 고려한

트리플 패턴 그룹은 다음과 같이 정의한다.

정의 6. 확장된 트리플 패턴 그룹(TPG)

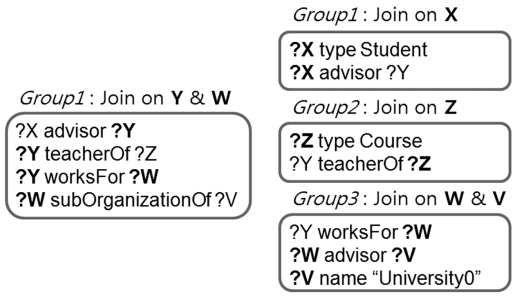
질의 Q에 대해, 트리플 패턴 그룹 $TPG_{K,L}(Q)$는 조인키 K 또는 L를 포함하는 모든 트리플 패턴들의 집합이며, 간단히 조인키 집합들만 나타내어 표현한다.

$$TPG_{K,L}(Q) = \{JoinKey(tp_i) \mid tp_i \in Q \wedge (K \in tp_i \text{ or } L \in tp_i)\}$$

예를 들어, 그림 1의 질의에서 7번 트리플 패턴으로 만든 그룹은 $TPG_{W,V}(Q)$={{W,V}, {Y,W}, {V}}이다.

우리가 제안하는 두 개의 조인키를 고려한 트리플 패턴 그룹을 만드는 데에도 트리플 패턴 그룹화 문제는 존재한다. 예를 들어, 그림 1의 질의에 대해 그림 6과 같은 그룹화 방법이 존재한다. 그림 6에서 (a)의 Group1은 {Y, W}를 조인한 후 남는 {X, V, Z}를 그 다음 잡에서 두 개를 조인하고 다시 그 다음 잡에서 나머지 한 개를 조인하게 되며 총 3번의 잡이 필요하다. (b)는 첫 번째 잡에서 {X}, {Z}, {W, V}를 각각 조인하고 남는 {Y}를 다음 잡에서 조인하게 되며 총 2번의 잡이 필요하다.

우리는 트리플 패턴 그룹화 문제를 HadoopRDF의 e-count 를 확장하여 해결한다. 두 개의 조인키를 조인할 때의 e-count와 한 개의 조인키를 조인할 때의 e-count모두 비교해 보다 더 적은 e-count를 가진 조인키들을 먼저 그룹으로 만드는 것이다. 만약, 하나의 조인키와 두 개의 조인키에 대한 e-count가 같은 경우는 두 개의 조인키를 먼저 그룹으로 만든다. 이것은 두 개의 조인키를 선택하는 것이 한번에 더 많은 조인키를 조인할 수 있기 때문이다. 두 개의 조인키에 대한 확장된 e-count는 다음과 같이 정의한다.



(a) 케이스 1 (b) 케이스 2

그림 6 중복 조인을 고려한 그룹화 예

정의 7. 확장된 e-count

e-count(K,L)는 $TPG_{K,L}(Q)$가 K와 L에 대해 조인 후 남게 되는 조인키의 개수를 말한다. 예를 들어, 그림 6의 (a) 에서 Group1에 대한 e-count(Y,W)는 Y와 W에 대해 조인 후 {X, Z, V} 세 개의 조인키가 남게 되어 3이다.

```
    Input: SPARQL query Q
    Output: A list of MapReduce jobs
1:  U := getSortedJoinKey(Q) // A list of join key
2:   set in each triple pattern sorted by e-count
3:  j:=0
4:  Job[j] := ∅          // A set of TPG
5:  while Q ≠ ∅ do
6:  O := ∅
7:  j++
8:     for i=1 to K do  // K is the size of U
9:        // ui is the i-th element of U
10:       if Can-Eliminate(Q, ui)=true then
11:        // Can-Eliminate return true when Q
12:   has at least two triple patterns each has ui
13:           Group := TPGui(Q)
14:           Q := Q − Group
15:           O := O ∪ Join-result(Group)
16:          Job[j] := Job[j] ∪ Group
17:       end if
18:    end for
19: Q := Q ∪ O
20: U := getSortedJoinKey(Q)
21: end while
22: return Job
```

그림 7 중복기반 조인을 이용한 플랜 생성 알고리즘

그림 6에서 (a)의 Group1과 (b)의 Group3은 두 개의 조인키를 고려한 경우인데 (a)는 e-count(Y,W)가 3이고, (b)는 e-count(W,V)가 1이 된다. 이와 같은 경우 우리는 e-count가 1이 되는 {W,V}를 먼저 선택하여 트리플 패턴 그룹을 만든다. 이 방법을 적용하여 맵리듀스 잡을 구성하는 플랜을 생성하는 알고리즘은 그림 7과 같다.

그림 1의 질의를 그림 7의 알고리즘을 이용하여 생성되는 플랜을 예를 들어보자. 입력은 SPARQL질의 Q(={{X}, {Z}, {V}, {X,Y}, {Y,Z}, {Y,W}, {W,V}})를 받는다. 1줄의 U는 Q의 모든 원소들을 정렬되어 저장되는데, 이때 사용되는 함수는 먼저 Q에 있는 조인키들에 대해 e-count의 오름차순으로 정렬하며, 같은 e-count를 가진 원소는 조인키의 개수에 대해 내림차순으로 정렬한다. 만약, 어떤 원소가 3개 이상의 조인키 집합을 가지고 있다면, 이들을 2개의 모든 조합으로 나열한 다음 e-count를 구한다. 예제에서 {X}, {Z}, {V}, {W,V}는 e-count가 1이며, {X,Y}, {Y,Z}는 2이고, {Y,W}는 3이므로 U=<{W,V}, {X}, {Z}, {V}, {X,Y}, {Y,Z}, {Y,W}>가 된다. 3줄의 j는 현재 잡의 번호를 나타내며 4줄의 job에는 각 잡의 번호에 따라 트리플 패턴 그룹들이 저장된다. 8줄의 for 루프에서는 U의 모든 조인키들을 순회하며, 10줄에서 조인 가능 여부를 검사한다. 조인 가

능 여부는 현재 그룹으로 생성하려는 조인키 ui (U의 i 번째 원소)를 포함하는 트리플 패턴들이 Q에 두 개 이상 남아있는지를 검사한다. 두 개 이상 있다면, 현재 조인키 ui를 그룹의 조인키로 선택하고 이 조인키를 하나라도 포함하는 모든 트리플 패턴들을 Q에서 뽑아 13줄의 Group에 저장한다. 14줄에서는 이 그룹에 포함되는 트리플 패턴들을 Q에서 제거하고, 15줄에서는 이번 그룹의 조인 결과에 남는 조인키들을 다음 잡에서 조인하기 위해 O에 저장한다. 현재 생성된 그룹을 16줄에서 job에 저장하고, 다시 for루프에서 다른 조인키로 새로운 트리플 패턴 그룹을 생성한다. 이렇게 한 잡에서 생성 가능한 트리플 패턴 그룹이 모두 생성되면 19줄에서 어떤 그룹에도 포함되지 않은 트리플 패턴의 조인키와 이번 잡의 결과로 남게 되는 조인키들을 다시 Q에 넣고 새로운 잡에서 그룹을 만들 수 있도록 준비한다. 20줄에서 Q를 1줄과 마찬가지로 정렬하여 U에 저장하며 다시 while문을 수행한다. 더 이상 Q에 남는 조인키가 없을 때 현재까지 생성된 잡의 리스트를 결과로 반환한다.

5.3 HadoopRDF와의 그룹화 방법 비교

어떠한 질의 Q가 N개의 트리플 패턴과 K개의 조인키를 가지고 있을 때, 본 논문에서 제안하는 방법의 잡의 개수 J의 계산식은 다음과 같다.

$$J = \begin{cases} 0 & N = 0 \\ 1 & N \le 3 \ \ or \ \ K = 2 \\ \min(\lceil 1.71 \log_2 N \rceil, K) & N > 3 \ and \ K > 2 \end{cases}$$

우리가 제안하는 방법은 두 개의 조인키를 한번에 조인시킬 수 있으므로 K≤2인 경우는 조인키가 최대 두 개인 경우이며 하나의 잡으로 처리된다. N≤3인 경우는 조인키가 1~3개가 존재할 수 있는데, 조인키가 2개 이하인 경우는 앞에서 설명했듯이 K≤2이므로 하나의 잡이 필요하다. 또한 N≤3이며, 조인키가 3개인 경우는 {X, Y}, {Y, Z}, {Z, X}와 같은 형태 밖에 존재하지 않는다. 이 3개의 트리플 패턴은 이 중 두 개의 조인키에 대한 조인을 수행하면 조인 결과에 나머지 하나의 조인키가 모두 모여있게 되므로 역시 한번의 잡으로 처리가 가능하다. 예를 들어, <x0,y1>, <y1,z2>, <z2,xi> 트리플이 있을 때, Y와 Z에 대해 조인하면 조인 결과는 <x0, y1, z2, xi>가 되며 여기서 x0과 xi가 동일한지 조인을 수행한 리듀서가 다시 비교해보면 된다.

그 밖의 경우에 대해서는 최악의 경우, 어떤 그룹도 두 개의 조인키를 조인하지 않는 그룹화를 가진 SPARQL질의가 존재하므로 HadoopRDF에서의 상계인 min(⌈1.71 $\log_2 N$⌉, K)가 J의 상계가 된다.

6. 실 험

이 절에서는 우리가 제안하는 방법에 대한 성능 평가 실험을 소개한다. 6.1절에서는 실험 환경을 설명하고 6.2절에서는 실험 결과를 분석한다.

6.1 실험 환경 및 실험 데이터

맵리듀스를 이용하기 위한 클러스터는 아마존 EC2 서비스를 이용하여 17개의 노드를 구성하였다. 각 노드는 1개의 가상 코어와 1.7GB의 메인 메모리, 그리고 160GB의 디스크를 갖는다. 맵리듀스는 오픈 소스인 Hadoop의 0.20.2버전을 이용하였다.

실험에 사용된 데이터는 LUBM데이터로 LUBM에서 제공하는 데이터 생성기로 1000개의 대학교 데이터 셋을 시드값 0으로 생성하였다. 생성된 데이터에는 약 1억 천만개 정도의 RDF 트리플이 존재한다. LUBM은 가장 널리 알려져 있는 RDF 벤치마킹 데이터 셋이며 우리가 비교하려는 HadoopRDF를 비롯한 많은 연구에 이용되고 있다.

SPARQL질의는 LUBM에서 제공하는 14개의 질의 중 조인키가 한 개인 질의와 비슷한 형태의 질의를 제외한 2, 9, 12번 질의에 대해 실험을 하였다. 실험에 사용된 질의는 표 1과 같은 특징을 가진다.

표 1 질의 정보

	질의 2	질의 9	질의 12
조인키 개수	3	3	2
트리플 패턴 개수	6	6	4

6.2 실험 결과

본 논문에서 제안한 방법의 성능을 평가하기 위해 HadoopRDF와 동일한 조건으로 2가지 측면에서 실험을 수행하였다. 첫 번째는 리듀서를 4개로 했을 때, 각 질의 별 응답속도에 대한 측정이고, 두 번째는 질의 2에 대해 다양한 리듀서 개수에 따른 응답속도 측정이다. 우리가 제안한 방법은 DuplicateRDF라 한다.

먼저 각 질의 별 맵리듀스 잡의 개수는 표 2와 같이 우리가 제안한 방법이 모두 하나씩 더 적었다. 여기서 질의 2, 9번이 조인키가 3개임에도 하나의 잡으로 처리할 수 있는 것은 5.3절에서 설명한 패턴이기 때문이다. 또한, 양쪽 모두 잡의 개수가 비교적 작은 이유는 2-웨이 조인을 방법에서는 3개 이상의 잡이 필요하지만 양쪽 모두 제안한 방법으로 인해 잡의 개수를 줄일 수 있었기 때문이다.

표 2과 같은 개수의 잡을 실행한 결과는 그림 8과 같다. 이 실험에서 DuplicateRDF가 HadoopRDF보다 뛰어

표 2 각 질의 별 맵리듀스 잡의 개수

	질의 2	질의 9	질의 12
DuplicateRDF	1	1	1
HadoopRDF	2	2	2
2-Way Join	5	5	3

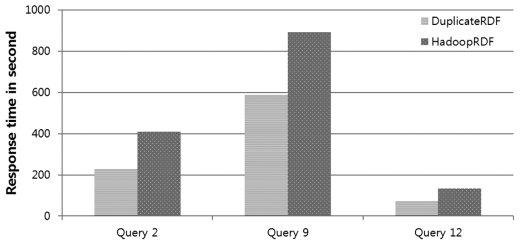


그림 8 각 질의 별 응답 시간

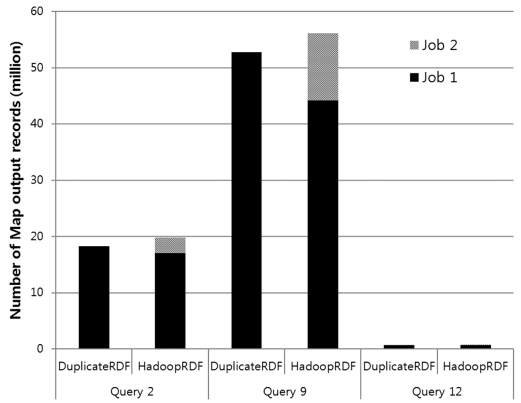


그림 9 각 질의 별 리듀서로 전송된 트리플 수

난 성능을 보였다. 이는 잡의 개수의 차이가 성능에 큰 영향을 미치는 것으로 볼 수 있으며, 중복으로 인한 부하가 그렇게 크지 않음을 나타낸다. 그림 9는 두 시스템의 각 질의 별 맵의 출력 레코드 수에 대한 비교이다. 맵의 출력 레코드 수는 리듀서로 전송되는 트리플 개수와 같다. DuplicateRDF는 첫 번째 잡에서 리듀서로 전송될 맵의 출력인 트리플들의 수가 HadoopRDF보다 조금 많았지만 하나의 잡 만으로 완료하였기 때문에 전체 잡을 비교해보면 오히려 HadoopRDF가 더 많은 트리플들을 전송하는 결과를 보였다.

두 번째 실험은 우리가 제안하는 방법이 리듀서의 개수에 따라 얼마만큼의 부하를 보이는지에 대한 실험이다. 중복 조인을 고려하게 되면 트리플들의 중복이 발생하는데, 중복이 얼마나 발생하는지는 리듀서의 개수에 따른다. 따라서, HadoopRDF와의 비교를 통해 우리의 방법이 어느 정도의 효과를 가져오는지 알아보았다. 그림 10은 리듀서의 개수 변화에 따른 두 시스템에 대한 응답속도를 비교한 결과이다. 그림 10에서 HadoopRDF는 리듀서의 개수가 증가할수록 꾸준히 응답시간이 감소되는 결과를 보였지만, DuplicateRDF는 리듀서의 개수가 12개까지는 어느 정도 감소하다가 16개일 때 다시 증가하는 현상을 보인다. 이는 리듀서가 증가할수록 중복기반 조인을 위해 중복시키는 데이터도 증가하기 때

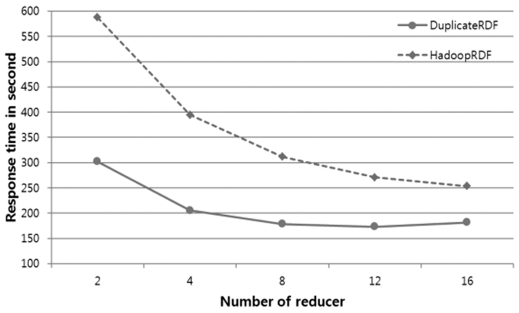


그림 10 리듀서 개수에 따른 응답시간

문이다. 중복으로 인한 부하는 맵의 출력 데이터를 디스크에 중복해서 쓰는데 발생하며, 이러한 비용을 줄이는 위한 연구가 [17]에서 소개 되었다. 이 방법을 적용하면 중복으로 인한 추가 비용을 줄일 수 있게 되어 성능 향상은 물론 더 뛰어난 확장성을 가져올 수 있을 것이다.

앞의 두 실험 결과를 통해 우리가 제안한 중복기반 조인을 고려한 그룹화를 이용하면, 기존 시스템보다 성능 향상을 가져올 수 있음을 알 수 있다.

7. 결론 및 향후 연구

본 논문에서는 맵리듀스를 이용한 SPARQL질의 처리에서의 성능 향상을 위한 기존의 방법을 알아보고, 성능 향상을 위한 새로운 방법을 소개하였다. 우리가 제안한 방법은 서로 다른 도메인에 있는 두 가지 기법인 중복기반 조인 기법과 트리플 패턴 그룹화를 혼용해 적용한 것이며, 이 때 발생하는 문제들에 대한 해법을 제시하였다. 또한 6장의 실험 결과에서처럼 우리가 제안한 방법은 기존 시스템에 비해 월등한 성능을 보였다.

향후 연구로는 보다 정확한 트리플 패턴 그룹화를 위해 통계 정보를 이용하는 방법과 트리플 패턴 그룹에 더 많은 조인키를 조인하는 방법이 필요하다. 동일한 개수의 맵리듀스 잡을 생성하는 트리플 패턴 그룹화 방법이 여러 개가 존재할 경우에는 통계 정보를 이용하면 중간 결과의 크기를 예측할 수 있어, 보다 정확한 트리플 패턴 그룹을 만들 수 있다. 또한 트리플 패턴 그룹을 생성할 때, 더 많은 조인키를 고려하기 위해서는 중복으로 인한 부담을 고려할 수 있는 비용 모델이 필요하며, 이러한 부분들이 고려된다면 질의 처리 성능이 더욱 향상될 것이다.

참 고 문 헌

[1] G. Klyne et al., Resource Description Framework (RDF): Concepts and Abstract Syntax, http://www.w3.org/TR/rdf-concepts, 2004.

[2] SPARQL Query Language for RDF, http://www.

w3.org/TR/rdf-sparql-query, 2006.
[3] J. Dean and S. Ghemaawat, MapReduce: Simplified Data Processing on Large Clusters, *Proc. of the 6th conference on Symposium on Operation System Design & Implementation*, pp.137-150, 2004.
[4] M. Husain et al., Heuristics Based Query Processing for Large RDF Graphs Using Cloud Computing, *IEEE TKDE*, vol.23, pp.1312-1327, 2011.
[5] F. N. Afrati and J. D. Ullman, Optimizing Multiway Joins in a Map-Reduce Environment, *IEEE TKDE*, vol.23, pp.1282-1298, 2011.
[6] Y. Guo, Z. Pan and J. Heflin, LUBM: A Benchmark for OWL Knowledge Base System, *Journal of Web Semantics*, vol.3, no.2-3, pp.158-182, 2005.
[7] Thomas Neumann, Gerhard Weikum, RDF-3X: a RISC-style engine for RDF, *Proc. VLDB Endow.*, vol.1, no.1, pp.647-659, 2008.
[8] Jeremy J. Carroll et al., Jena: implementing the Semantic Web Recommendations, *Proc. of the 13th international World Wide Web conference on Alternate track papers & posters*, pp.74-83, 2004.
[9] A. Kiryakov, D. Ognyanov and D. Manov, OWLIM-A Pragmatic Semantic Repository for OWL, *Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems*, 2005.
[10] J. Broekstra, A. Kampman and F. van Harmelen, Sesame: An Architecture for Storing and Querying RDF Data and Schema Information, Semantics for the WWW, MIT Press, 2001.
[11] A. Schatzle, M. Przyjaciel-Zablocki and G. Lausen, PigSPARQL: Mapping SPARQL to Pig Latin, *Proc. of the International Workshop on Semantic Web Information Management*, 2011.
[12] C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data, *Proc. of SIGMOD*, pp.1099-1110, 2008.
[13] P. Ravindra, V. V. Deshpande, and K. Anyanwu, Towards Scalable RDF Graph Analytics on MapReduce, *Proc. of MDAC*, pp.1-6, 2010.
[14] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen, Scalable distributed reasoning using mapreduce, *Proc. of ISWC*, vol.5823, pp.634-649, 2009.
[15] H. Choi, J. Son, Y. Cho, M. K. Sung, and Y. D. Chung, SPIDER: a system for scalable, parallel/distributed evaluation of large-scale RDF data, *Proc. of CIKM*, pp.2087-2088, 2009.
[16] J. Myung, J. Yeon, and S. Lee, SPARQL basic graph pattern processing with iterative MapReduce, *Proc. of MDAC*, pp.6:1 - 6:6, 2010.
[17] D. Jiang, A. K. H. Tung, and G. Chen. Map-join-reduce: Towards Scalable and Efficient Data Analysis on Large Clusters, *IEEE TKDE*, vol.23, pp.1299-1311, 2010.

김 태 경
2010년 충남대학교 컴퓨터공학부 학사 2012년 서울대학교 컴퓨터공학부 석사 2012년~현재 티베로 연구원. 관심분야는 맵리듀스, 빅데이터 처리

김 기 성
2003년 서울대학교 응용화학부 학사. 2006년~2009년 티맥스소프트. 2003년~현재 서울대학교 컴퓨터공학부 박사과정 재학 중. 관심분야는 데이터베이스, 시맨틱 웹, 분산 병렬 데이터 처리

김 형 주
1982년 서울대학교 전산학과(학사). 1985년 Univ. of Texas at Austin(석사) 1988년 Univ. of Texas at Austin(박사) 1988년~1990년 Georgia Institute of Technology(부교수). 1991년~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이타베이스, XML, 시맨틱 웹, 온톨로지