KIPS Tr. Software and Data Eng.
Vol.3, No.8 pp.293~298 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2014.3.8.293

Structural Change Detection Technique for RDF Data in MapReduce

Taewhi Lee† · Dong-Hyuk Im††

ABSTRACT

Detecting and understanding the changes between RDF data is crucial in the evolutionary process, synchronization system, and versioning system on the web of data. However, current researches on detecting changes still remain unsatisfactory in that they did neither consider the large scale of RDF data nor accurately produce the RDF deltas. In this paper, we propose a scalable and effective change detection using a MapReduce framework which has been used in many fields to process and analyze large volumes of data. In particular, we focus on the structure-based change detection that adopts a strategy for the comparison of blank nodes in RDF data. To achieve this, we employ a method which is composed of two MapReduce jobs. First job partitions the triples with blank nodes by grouping each triple with the same blank node ID and then computes the incoming path to the blank node. Second job partitions the triples with the same path and matchs blank nodes with the Hungarian method. In experiments, we show that our approach is more accurate and effective than the previous approach.

Keywords : RDF Change Detection, MapReduce Framework, Hadoop, Blank Node Matching, Large Scale Data

맵리듀스에서의 구조적 RDF 데이터 변경 탐지 기법

이 태 휘† · 임 동 혁††

요 약

RDF 데이터의 변경 내용을 탐지하고 이해하는 것은 데이터 웹의 진화 프로세스, 동기화 시스템, 버전 관리 시스템에서 매우 중요한 역할을 한다. 하지만 현재의 연구들은 대용량 데이터를 고려하지 않거나 정확하게 변경 내용을 탐지하지 못한다는 점에서 여전히 미흡하다. 본 논문에서는 대용량 데이터의 처리, 분석을 위해 여러 분야에서 사용되는 맵리듀스 프레임워크 기반의 확장가능하며 효과적인 변경 탐지 기법을 제안한다. 특히, RDF 데이터의 공노드를 비교하는 구조적인 변경 탐지에 초점을 둔다. 이를 위해, 두 개의 맵리듀스 작업으로 이루어진 방법을 사용한다. 첫 번째 작업에서는 공노드에 부여된 내부 아이디가 같은 트리플들을 그룹화하여 공노드에 연결된 경로를 계산한다. 두 번째 작업에서는 같은 경로를 가지는 트리플들을 그룹화하여 헝가리안 메소드를 이용하여 공노드 매칭을 수행한다. 실험을 통해 제안한 방법이 기존 방법보다 더 정확하고 효과적임을 보인다.

키워드 : RDF 변경 탐지, 맵리듀스 프레임워크, 하둡, 공노드 매칭, 대용량 데이터

1. 서 론

시맨틱 웹을 기술하기 위한 언어로 웹상의 정보를 표현하는 RDF(Resource Description Framework)는 기술 규격이 간단하고 웹 에이전트가 웹상의 자원을 자동화하여 처리할 수 있기 때문에 다양한 분야에서 사용되고 있다[1]. 특히 생물정보학, 의료 정보와 같은 용어체계 기술과 이미지, 동영상 등의 멀티미디어 분야에서 메타 데이터를 기술하기 위한 언어로 널리 사용된다.

RDF 데이터들은 갱신이 빈번히 일어나기 때문에 어떤 부분에 갱신이 일어났는지 탐지하는 툴은 매우 유용하다[2-5]. 특히 대용량 RDF 데이터에서 변경하는 부분이 극히 일부분일 경우 변경된 내용을 탐지하여 반영하는 것이 효율적인 방법으로 알려져 있다. 하지만 기존 연구들은 메모리 기반이거나 관계형 데이터베이스 기반의 변경 탐지들로 하나의 머신에 있는 자원만을 사용하기 때문에 대용량 데이터에 대해서는 메모리 부족 또는 수행 시간의 급격한 증가 문제를 가지고 있다. 이러한 문제점들을 해결하기 위해 분산 및 병렬 방법이 필요하며, 이 중 가장 대표적인 것으로 맵리듀스 프레임워크[6]가 많이 사용되고 있다. 하지만, 맵리듀스를 RDF

※ 이 논문은 2013년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임(2013-0076).
† 정 회 원 : 한국전자통신연구원 빅데이터SW플랫폼연구부 선임연구원
†† 정 회 원 : 호서대학교 컴퓨터정보공학부 조교수
Manuscript Received : March 28, 2014
First Revision : June 19, 2014
Accepted : June 19, 2014
* Corresponding Author : Dong-Hyuk Im(dhim@hoseo.edu)

데이터의 변경 탐지에 적용하는 연구는 미흡한 실정이다.

[7]에서 맵리듀스 기반의 트리플 데이터에 대한 변경 탐지를 제안하였지만 트리플에서 주어를 키로 사용하는 방법은 RDF 데이터의 구조적 특성을 반영하지 못한다. 특히 RDF 데이터에서 사용하는 공노드(Blank Node)를 고려하지 않기 때문에 최적의 변경 탐지를 할 수 없다. 공노드는 현재 존재하지 않거나 집합(Set) 등과 같은 관계를 기술할 때 사용하는 RDF 데이터 표현 중의 한 방법이다. Fig. 1의 공노드의 의미는 생년월일이 '04-21'이며 이름이 'Tom'인 사람을 표현한다. 일반적으로 공노드는 다른 공노드와 구별하기 위해 내부적인 아이디를 부여하고 사용한다. 하지만 같은 공노드라고 하더라도 변경이 발생하면 아이디가 바뀔 수 있다. Fig. 1에서와 같이 추가적인 트리플이 추가되면서 내부 아이디가 '_:1'에서 '_:2'로 바뀔 수 있다. 따라서 아이디가 다르더라도 같은 공노드를 찾는 것이 RDF 구조적 변경 탐지에 있어서 매우 중요하다.

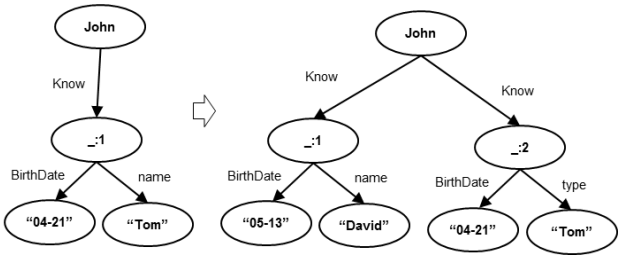

Fig. 1. RDF sample data and change

이에 본 논문에서는 공노드를 고려하는 맵리듀스 프레임워크 기반의 변경 탐지를 제안한다. [2, 8]에서 공노드를 포함하는 변경 탐지를 제안하였지만 메모리 기반으로 대용량을 다룰 수 없는 한계를 가진다. 제안하는 방법은 2개의 작업으로 이루어진 맵리듀스 프레임워크를 사용한다. 첫 번째 작업에서는 같은 공노드들을 찾기 위한 전처리 작업으로 연결된 경로를 계산하며, 두 번째 작업에서는 같은 경로를 가지는 공노드들끼리의 변경 여부를 헝가리안 메소드를 이용하여 계산한다. 본 연구의 핵심은 RDF 데이터의 구조적 변경 탐지를 위해 기존의 공노드 매칭을 맵리듀스 프레임워크에 적용하여 대용량 데이터를 다룰 수 있도록 하였다는 점이다.

본 논문의 구성은 다음과 같다. 2장에서는 RDF 데이터에 대한 기존의 변경 탐지 기법에 관한 관련 연구를 살펴보고, 본 연구에서 사용하려는 맵리듀스 기반 RDF 데이터 처리 관련 연구에 대해 자세히 알아본다. 3장에서는 배경 지식과 기존 연구들의 문제점을 설명하고, 4장에서는 본 논문에서 제안하는 RDF 데이터에 대한 맵리듀스 기반의 구조적 변경 탐지 기법을 자세히 설명한다. 5장에서는 실험 결과를 제시하며 우리가 제안한 기법의 성능을 검증한다. 마지막 6장에서는 결론을 맺으며 논문을 마무리 한다.

2. 관련 연구

2.1 RDF 데이터의 변경 탐지 기법

RDF 데이터의 변경 탐지는 크게 구조적인 부분과 의미적인 부분으로 나누어진다. 구조적인 부분은 RDF 데이터가 트리플 집합으로 이루어져 있기 때문에 두 집합의 차집합으로 계산되며 공노드를 포함하는 구조적인 변경에 초점을 둔다[2,3,8]. 의미적인 변경 탐지는 RDF 데이터에서 사용하는 추론을 이용하여 변경 탐지 결과를 줄이거나 트리플이 변경이 되어도 의미적으로 해석을 하는 변경 탐지를 주로 연구한다[4,9]. 그러나 이들 기법 모두 아직까지 대용량의 데이터를 고려하지 않고 있다.

2.2 맵리듀스 기반의 RDF 데이터 처리

RDF 데이터가 급격하게 증가함에 따라 맵리듀스를 이용하여 대용량의 RDF 데이터를 처리하는 다양한 연구가 진행되었다[10-12]. [10,11]에서는 맵리듀스를 이용하여 SPARQL 질의 처리를 할 때 맵리듀스 잡(Job)의 개수를 줄이기 위해 어떠한 조인을 먼저 할 것인지에 대한 방법을 제안하였다. [10]에서는 RDF 트리플 패턴 중에서 가장 많이 나타나는 조인키를 먼저 조인하는 방법을 사용하였으며, [11]에서는 서로 독립적으로 실행 가능한 조인을 많이 생성시키는 조인 방법을 제안하였다. [12]는 맵리듀스를 이용하여 대용량 RDF 데이터에 대한 추론을 제안하였다. 그러나 이들 연구들은 주로 질의 처리 및 추론에 초점을 두고 있으며 변경 탐지에 맵리듀스를 적용하려는 연구는 충분히 진행되고 있지 않고 있다.

3. 배경 지식

이 장에서는 배경 지식을 설명하며 본 논문에서 사용하는 용어들을 정의한다. 3.1절에서는 RDF 데이터와 변경 탐지에 대해 알아보고, 3.2절에서는 구조를 고려하지 않는 기존의 맵리듀스 변경탐지 기법을 설명한다.

3.1 RDF 데이터 모델과 변경 탐지

정의 1. RDF 트리플

RDF 데이터는 <주어(S), 술어(P), 목적어(O)>로 나타내며 다음과 같은 트리플로 정의된다.

$$T(S, P, O) \in (U \cup B) \times U \times (U \cup B \cup L)$$

이때, U는 URI, L은 Literal(문자열), B는 공노드(Blank node)를 나타낸다. 일반적으로 공노드는 실제하지 않거나 지금 당장 URI를 부여할 수 없을 때 사용하며 다른 공노드와 구별하기 위해 시스템에서 자체적으로 아이디를 부여하여 처리한다.

정의 2. RDF 그래프와 동등성(Equivalence)

RDF 트리플의 집합을 RDF 그래프라 정의하며, 두 RDF 그래프 G_1과 G_2에서 다음 조건을 만족하는 전단사(bijection) M이 존재하면 동등하다고 정의한다[1,8].

1) M은 두 그래프의 공노드와 공노드를 매핑한다.
   (G_1에 속한 모든 공노드가 G_2에 일대일 대응이 된다.)
2) M은 G_1에 속한 모든 URI U_1을 G_2에 매핑한다.
3) M은 G_1에 속한 모든 Literal L_1을 G_2에 매핑한다.

정의 2는 두 RDF 그래프의 비교를 다루고 있으며 두 그래프에서 공노드의 내부 아이디를 제외한 모든 트리플이 같이 존재하면 같은 그래프로 간주한다는 것이다. 예를 들어, Fig. 1에서 공노드 내부 아이디가 '_:1'에서 '_:2'로 바뀌었지만 공노드에 연결된 다른 트리플들이 같기 때문에 두 공노드는 같은 공노드로 간주된다.

정의 3. RDF 그래프에서 변경(Delta)

두 트리플 집합(RDF 그래프)에서 하나의 RDF 그래프로부터 다른 RDF를 동등하게 만들기 위한 삽입 혹은 삭제되는 트리플들을 RDF의 변경 집합으로 정의한다.

정의 3은 RDF 그래프에서의 변경 연산을 설명한다. 일반적으로 트리플의 삭제 연산과 삽입 연산을 고려하며 갱신은 삭제 및 삽입 연산으로 표현된다[2,4]. Fig. 1의 두 RDF 그래프의 변경은 D={(John know _:1), (_:1 BirthDate "05-13"), (_:1 name "David")}의 3개의 트리플 삽입으로 표현할 수 있다.

정의 4. 최소 변경

RDF 변경 중에서 변경 연산의 개수가 최소가 되는 변경을 최소 변경이라고 정의한다.

두 그래프에서 가능한 변경은 매우 많다. 최악의 경우는 한 그래프의 모든 트리플을 삭제하고 다른 그래프의 모든 트리플을 삽입하는 것이다. 이러한 변경 중에 최소의 연산을 가진 변경은 동등한 그래프를 만들 때 최소의 시간만이 필요하기 때문에 최소 변경을 탐지하는 것은 빠르게 변경 탐지를 하는 것만큼 중요하다. 공노드를 포함하고 있는 두 RDF 그래프에서 최소 변경을 찾는 것은 [8]에서 NP-HARD 문제로 증명되었다. 따라서 많은 변경 탐지 기법이 최소 변경에 근접한 변경을 찾는 방법을 사용하고 있다. 본 논문에서도 이와 같은 근접한 최소 변경을 찾는 것에 초점을 둔다.

3.2 기존 맵리듀스 기반 변경 탐지의 한계

[7]에서는 맵리듀스 프레임워크를 이용하여 RDF 데이터의 변경 탐지를 제안하였다. Fig. 2와 같이 트리플의 주어(S)를 맵 함수의 킷값으로 사용하여 같은 킷값을 가지는 트리플을 같은 리듀서로 보낸다. 각 리듀서에서는 원본(S)과 타깃(T) 트리플을 비교하여 변경 내용을 계산한다. 하지만 공노드가 있는 트리플에 대해서 임시 내부 아이디를 가지고 계산을 하므로 같은 공노드라고 하더라도 다른 임시 아이디를 가지면 다른 공노드로 간주하여 변경 탐지 결과가 너무 커진다. 예를 들어, Fig. 2의 예제 결과를 보면 실제 추가되는 트리플은 2개이지만 탐지된 결과는 3개의 트리플이 삭제되고 5개의 트리플이 삽입되는 결과를 얻게 된다. 비록 정확한 변경 내용을 다 포함하고 있더라도 불필요한 변경 내용을 포함하고 있는 문제점을 가지게 된다.

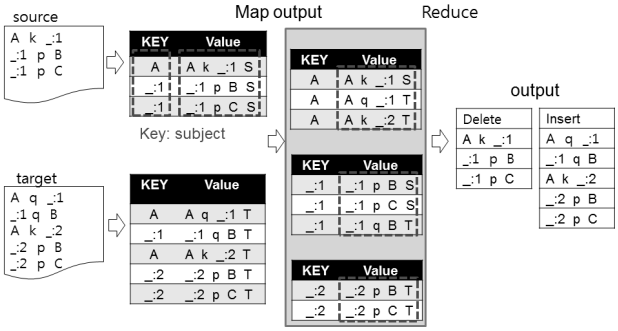

Fig. 2. Existing approach for blank node

4. 공노드를 고려하는 맵리듀스 기반 변경 탐지

본 논문에서는 기존 연구들이 가진 문제점을 해결하기 위해 공노드를 고려하는 맵리듀스 변경 탐지 기법을 제안한다. 공노드를 포함하는 트리플만 논문에서 다루며 공노드가 없는 트리플들은 [7]의 방법을 적용하도록 한다. 제안하는 기법에서는 다음과 같이 2번의 맵리듀스 잡을 수행하며 변경 탐지 결과를 생성한다.

Algorithm 1 : Map (1st Job)

Input: input triple x ∈S ∪ T
/* S: 원본, T: 타깃 */
1: **if** x∈S **then**
2: ID = triple's blank node ID + "#S"
3: **else**
4: ID = triple's blank node ID + "#T"
5: Output(ID, triple)

Algorithm 2 : Reduce (1st Job)

Input: (ID, [t_1, t_2, ... ti])
/* ti는 subject, property, object로 구성된 트리플 */
/* ID 는 공노드 ID 와 tag 정보(S or T)로 구성 */
1: Result = ∅
2: **for all** ti in input list **do**
3: **if** t_i.object == blank node **then**
4: Path = ti.subject + "." + ti.property
5: **else**
6: Result = Result ∪ t_i
7: Output(Path, Result+ID.tag)

첫 번째 작업의 맵리듀스는 공노드 매칭을 위한 전처리 작업을 진행한다. [7]과는 다르게 공노드의 내부 아이디를 킷값으로 맵 함수를 수행하여 같은 내부 아이디를 가지는 트리플들을 같은 리듀서로 모이게 한다. 리듀서 함수에서는 공노드에 들어오는 경로(incoming path)를 계산하여 다음 맵리듀스 단계에서 사용할 수 있게 맨 앞에 저장한다. 알고리즘 1과 2는 첫 번째 작업의 맵함수와 리듀스 함수를 보여주며 Fig. 3은 알고리즘의 예를 보여준다.

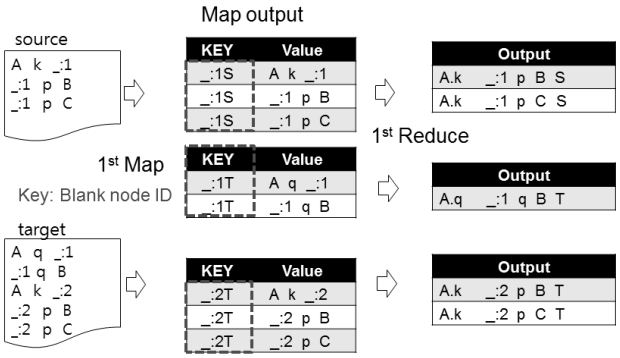

Fig. 3. MapReduce in first job

Algorithm 3 : Map (2nd Job)

Input: input ($Path_i$, Result R_i) ∈1st Output
1: Key = $Path_i$
2: Value = R_i /* R_i = (ti, tag) */
3: Output(Key, Value)

Algorithm 4 : Reduce (2nd Job)

Input: (Path, [(t_1, tag), (t_2, tag),...(ti, tag)])
1: SList = ∅; TList = ∅ ; Delta = ∅
2: **for all** (ti, tag) in input list **do**
3: **if** tag = "S" **then**
4: SList = SList ∪ ti
5: **else**
6: TList = TList ∪ ti
7: Delta = BlankNodeMatching(SList, TList)
8: Output(Delta)

두 번째 작업의 맵리듀스에서는 실제 공노드 매칭을 진행한다. 알고리즘 3과 4는 두 번째 작업의 맵 함수와 리듀스 함수를 보여준다. 작업 1에서 공노드에 들어오는 경로가 이미 계산되었으므로 맵 함수에서는 공노드에 들어오는 경로를 킷값으로 트리플을 분배한다. 리듀스 함수에서는 같은 경로를 가진 공노드들이 모이게 되며 이 공노드들끼리 원본과 타깃으로 분류하여 매칭을 수행한다. 공노드 매칭은 [2]에서 사용한 헝가리안 메소드를 사용한다. 헝가리안 메소드는 최적 할당 문제에 사용되는 알고리즘으로 공노드 매칭에 있어서 최적의 매칭을 찾아주는 역할을 한다. 예를 들면, Fig. 5에서 두 RDF 데이터는 경로가 모두 'John.know'로 같은 공노드들을 가지고 있다. 따라서 여러 경우의 매칭이 나오게 되는데 이중에서 최적의 매칭은 공노드 _:1과 _:a, _:2와 _:c 그리고 _:3과 _:b를 매칭하는 것이다. 본 작업에서 맵리듀스의 결과는 최소 변경의 공노드 매칭을 찾아준다.

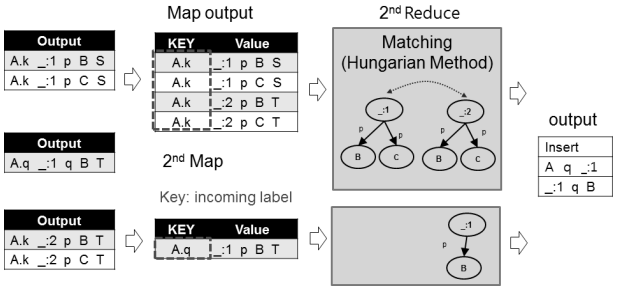

Fig. 4. MapReduce in second job

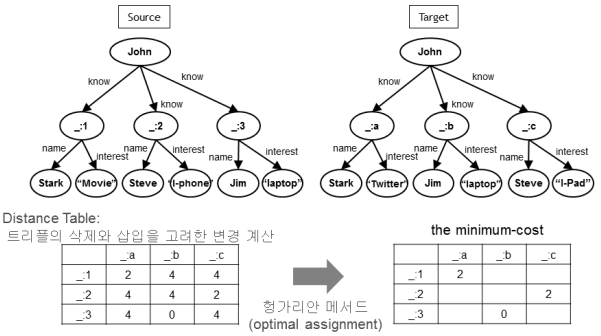

Fig. 5. Blank node matching using Hungarian method

5. 실 험

5.1 실험 환경 및 실험 데이터

맵리듀스를 이용하기 위한 클러스터는 11개로 구성되며 각 노드에 해당하는 컴퓨터는 3.1GHz 쿼드코어 CPU, 4GB RAM, 2TB의 하드디스크의 사양을 가지며, 운영 체제는 32비트 Ubuntu 10.10, 자바 버전은 1.6.0_26, 제안한 기법은 하둡(Hadoop) 1.2.1 버전에서 구현하였다.

실험 데이터로는 SP2B[13]의 데이터 생성기를 이용하여 5, 10, 15, 20GB의 데이터를 생성하여 사용하였다. SP2B는 출판물과 저장 정보 등을 포함하는 DBLP[14]와 유사한 스키마의 데이터베이스를 생성하여 N3(주어, 술어, 목적어) 형식의 RDF 문서로 출력한다. 생성한 DBLP 데이터는 저자와 문헌 참조 정보가 공노드 형태로 표현하는 구조이며, SP2Bench는 저자를 생성할 때 사용하기 위해 기본으로 각각 88799개의 성과 이름을 텍스트 파일로 가지고 있다. 본 논문에서는 실험을 위해 공노드인 저자 정보가 변경되는 데이터를 생성하였으며, 생성에 사용하는 텍스트 파일을 수정하여 성은 전부 'node'로 하고, 이름의 경우 두 파일에 차이를 두도록 생성하였다. 또한 크기가 증가하여도 변경되는 공노드의 수를 똑같이 유지하여 변경 탐지의 정확도를 계산할 수 있도록 하였다. 비교 대상은 [7]에서 사용한 공노드를 고려하지 않는 맵리듀스 방법(MRNaiveDiff)이며 본 논문에서 제안한

MRTripleDiff와 변경 탐지 결과된 트리플의 수와 실행 시간을 비교하였다. 비록 [2, 3, 8]이 본 논문과 같이 RDF 데이터의 변경 탐지에서 공노드를 고려하고 있지만 메모리 기반의 방법이라 대용량에 적용할 수 없는 문제점을 가지기 때문에 비교에서 제외하였다.

Fig. 6은 두 방법의 변경된 트리플 수(삭제와 삽입된 트리플의 수)를 로그스케일로 보여준다. 둘 다 동등한 RDF 그래프를 만들 수 있는 트리플들을 포함하고 있지만 결과에서 알 수 있듯이 MRTripleDiff가 MRNaiveDiff보다 훨씬 적은 트리플을 가지는 것을 알 수 있다. 특히 데이터의 크기가 커짐에 따라 그 차이가 커지는 것을 확인할 수 있다(20G의 경우 약 100배의 차이). 따라서 대용량 RDF 데이터에서 변경 탐지할 경우 제안한 기법이 효과적으로 사용 가능함을 알 수 있다.

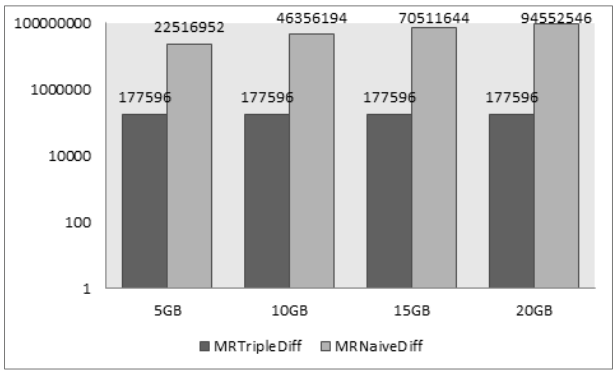

Fig. 6. Result of Diff quality (# of triples)

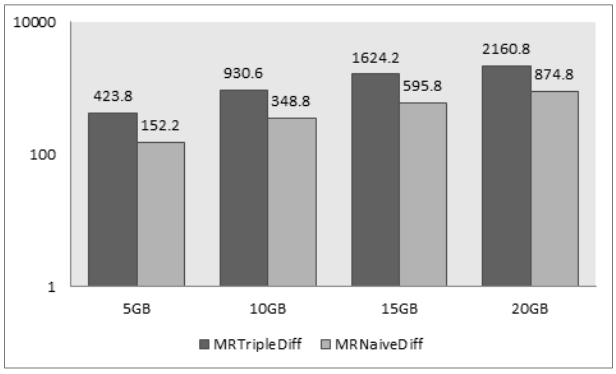

Fig. 7. Execution time (second)

Fig. 7은 두 방법의 실행 시간을 보여준다. 결과에서 알 수 있듯이 공노드를 고려하지 않는 기존 방법(MRNaiveDiff)이 성능이 좋게 나온다. 본 논문에서 제안하는 방법(MRTripleDiff)은 2개의 작업으로 맵리듀스가 이루어졌고 헝가리안 메소드를 적용하는 시간이 추가되었기 때문이다. 하지만 Fig. 6의 결과와 비교를 하면 변경 탐지 결과에서의 이득이 훨씬 크다고 볼 수 있으며 훨씬 적은 트리플이 향후 업데이트 같은 작업이나 버전 관리를 수행할 때 상당히 효율적이므로 제안한 기법 역시 사용 가능하다고 볼 수 있다.

6. 결 론

본 연구에서는 맵리듀스를 이용한 RDF 데이터의 구조적 변경 탐지 기법을 제안하였다. 공노드를 고려하지 않는 기존의 문제점을 해결하기 위해 2개의 작업으로 구성된 맵리듀스를 이용하여 공노드 매칭을 사용하였다. 첫 번째 작업에서는 같은 경로를 가지는 공노드들을 계산하고 다음 작업에서는 기존의 헝가리안 메소드를 적용하여 공노드 매칭을 수행하였다. 실험을 통하여 기존의 공노드를 고려하지 않는 맵리듀스 기반 변경 탐지보다 최소의 변경을 탐지하는 것을 보였다.

향후 연구로는 헝가리안 메소드를 사용하지 않는 맵리듀스 기반 변경 탐지 방법과 리듀스 작업의 균등을 통한 성능 개선이 필요하다. 헝가리안 메소드는 정확한 매칭에 사용되는 방법이므로 유사 매칭(Similarity matching)을 통한 맵리듀스 기반 변경 탐지 방법이 가능하다. 또한 특정 리듀서에 의해 지연되는 맵리듀스 작업을 균등하게 분배함으로써 맵리듀스의 성능을 개선할 수 있다. 이러한 부분들이 본 논문의 변경 탐지에 고려된다면 성능이 더욱 향상 될 것이다.

Reference

[1] G. Klyne and J. J. Carroll, “Resource Description Framework (RDF): Concepts and Abstract”, http://www.w3.org/TR/rdf11-concepts/, 2004.

[2] D. H. Lee, D. H. Im, H. J. Kim, “A Change Detection Technique Supporting Nested Blank Nodes of RDF Document,” Journal of KIISE : Database, Vol.34, No.6, pp. 518-527, 2007.

[3] T. Berners-Lee and D. Connolly, “Delta: An Ontology for the Distribution of Differences between RDF Graphs,” http://w3.org/DesignIssues/Diff

[4] D. Zeginis Y. Tzitzikas, and V. Christophides, “On Computing Deltas of RDF/S Knowledge Bases,” ACM Transactions on the Web(TWEB), 2011.

[5] D. H. Im, S. W. Lee, and H. J. Kim, “A version management framework for RDF triple stores,” International Journal of Software Engineering and Knowledge Engineering, Vol.22, No.1, pp.85-106, 2012.

[6] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation, pp.137-150, 2004.

[7] J. H. Ahn, D. H. Im, J. W. Jung, N. Zong, K. S. Ha, H. G. Kim, “Design and implementation of change detection for Linked Data using mapreduce framework,” HCI KOREA, 2013.

[8] Y. Tzitzikas, C. Lantzaki, and D. Zeginis, “Blank Node Matching and RDF/S Comparison Functions,” in Proceedings of the 11th International Semantic Web Conference(ISWC’12), 2012.

[9] D. H. Im, S. W. Lee, and H. J. Kim, “Backward inference and pruning for RDF change detection using RDBMS,” Journal of Information Science, Vol.39, No.2, pp.238-255, 2013.

[10] J. Myung, J. Yeon, and S. Lee, “SPARQL basic graph pattern processing with iterative MapReduce,” in Proceedings of MDAC, pp.6:1-6:6, 2010.

[11] M. Husain et al., “Heuristics based Query Processing for Large RDF Graphs using Cloud Computing,” IEEE TKDE, Vol.23, No.9, pp.1322-1327, 2011.

[12] J. Urbani et al, “Webpie: A web-scale parallen inference engine using mapreduce”, Journal of Web Semantics, Vol. 10, pp.59-75, 2012.

[13] M. Schmidt et al, “SP²Bench: A SPARQL Performance Benchmark,” in Proceedings of ICDE, pp.222-233, 2009.

[14] DBLP computer science bibliography. http://www.informatik.uni-trier.de/~ley/db/

이 태 휘

e-mail : taewhi@etri.re.kr

2004년 서울대학교 컴퓨터공학부(학사)

2007년~2010년 티맥스소프트 선임연구원

2014년 서울대학교 컴퓨터공학부(석·박사통합)

2014년~현 재 한국전자통신연구원 빅데이터SW플랫폼연구부 선임연구원

관심분야 : 빅데이터 처리 기술, 시맨틱 웹, 온톨로지

임 동 혁

e-mail : dhim@hoseo.edu

2003년 고려대학교 컴퓨터교육과(학사)

2005년 서울대학교 컴퓨터공학부(석사)

2011년 서울대학교 컴퓨터공학부(박사)

2011년~2012년 서울대학교 치의학연구소 선임연구원

2013년~현 재 호서대학교 컴퓨터정보공학부 조교수

관심분야 : 빅데이터 처리 기술, 시맨틱 웹, 온톨로지