JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 4: 487~494, Apr. 2022

맵리듀스 기반 대량 RDF 데이터셋 압축 변환 및 저장 방법

김인아[1] · 이경하[2,3] · 이규철[4*]

Compression Conversion and Storing of Large RDF datasets based on MapReduce

InA Kim[1] · Kyong-Ha Lee[2,3] · Kyu-Chul Lee[4*]

[1]Ph.D Candidate, Department of Computer Engineering, Chungnam National University, Daejeon, 34134 Korea
[2]Senior Researcher, Korea Institute of Science and Technology Information, Daejeon, 34141 Korea
[3]Professor, University of Science and Technology, Daejeon, 34113 Korea
[4*]Professor, Department of Computer Engineering, Chungnam National University, Daejeon, 34134 Korea

요 약

최근 데이터를 활용한 분석에 대한 수요와 함께 분석 데이터인 지식 그래프의 크기는 점차 증가하여, 웹에서 수집한 데이터를 지식 그래프로 추출하였을 때 약 820억개의 엣지(Edge)를 가지는 수준까지 도달하였다. 많은 지식 그래프들은 웹 자원에 대한 메타데이터를 표현하기 위한 W3C 표준인 RDF(Resource Description Framework) 형식으로 표현되며, RDF 특성으로 인해 기존의 RDF 저장소들은 대량 RDF 데이터를 압축하고 저장할 때 처리 시간의 오버헤드가 발생하는 문제점을 가진다. 본 논문은 이러한 문제점을 개선하기 위해, 맵리듀스를 사용하여 대량 RDF 데이터를 정수 ID로 압축 변환하고, 수직 분할하여 저장하는 방법을 제안한다. 본 논문에서 제안한 방법은 RDF-3X와 비교하였을 때 최대 25.2배, H2RDF+와 비교하였을 때 최대 3.7배까지의 높은 성능 향상을 보였다.

ABSTRACT

With the recent demand for analysis using data, the size of the knowledge graph, which is the data to be analyzed, gradually increased, reaching about 82 billion edges when extracted from the web as a knowledge graph. A lot of knowledge graphs are represented in the form of Resource Description Framework (RDF), which is a standard of W3C for representing metadata for web resources. Because of the characteristics of RDF, existing RDF storages have the limitations of processing time overhead when converting and storing large amounts of RDF data. To resolve these limitations, in this paper, we propose a method of compressing and converting large amounts of RDF data into integer IDs using MapReduce, and vertically partitioning and storing them. Our proposed method demonstrated a high performance improvement of up to 25.2 times compared to RDF-3X and up to 3.7 times compared to H2RDF+.

키워드 : 맵리듀스, RDF, 빅데이터, 압축 변환, 수직 분할

Keywords : MapReduce, RDF, Big data, Compression and conversion, Vertical partitioning

Received 7 March 2022, **Revised** 10 March 2022, **Accepted** 13 March 2022

*** Corresponding Author** Kyu-Chul Lee (E-mail:kclee@cnu.ac.kr, Tel:+82-42-821-6658)
Professor, Department of Computer Engineering, Chungnam National University, Daejeon, 34134 Korea

Open Access http://doi.org/10.6109/jkiice.2022.26.4.487 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 데이터를 활용한 분석에 대한 수요가 높아지면서 유의미한 분석 결과를 도출하기 위한 정제된 데이터에 대한 관심도 같이 높아지고 있다. 특히, 웹과 소셜 데이터와 같이 연결된 그래프 데이터를 분석하기 위한 AI 분석으로 인해 다양한 도메인에 대한 지식 그래프의 규모가 확장되어가고 있다. 지식 그래프의 크기는 점차 증가하여, 웹에서 수집한 데이터를 지식 그래프로 추출하였을 때 약 820억개의 엣지(Edge)를 가지는 수준까지 도달하였다[1].

많은 지식 그래프들은 웹 자원에 대한 메타데이터를 표현하기 위한 W3C의 표준인 RDF(Resource Description Framework) 형식으로 표현된다[2, 3]. RDF는 웹 자원을 고유 URI(Uniform Resource Identifier)로 식별하고 약속된 형식으로 데이터를 표현하여 여러 애플리케이션 간 데이터 교환을 가능하게 한다. RDF는 주어(Subject), 서술어(Predicate), 목적어(Object)를 가지는 트리플(Triple)들이 서로 연결된 형태를 가지며, 그림 1은 이러한 RDF 트리플 예시를 보여준다. RDF를 구성하는 트리플의 주어, 서술어, 목적어 각 요소에는 URI, 리터럴(Literal), 빈 노드(Blank Node)가 위치할 수 있다.

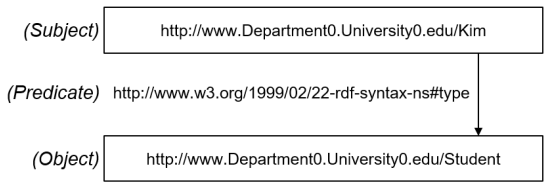

Fig. 1 RDF triple

URI, 리터럴 등은 문자열 형태로 저장되는데, URI와 같은 요소는 자원 이름 앞에 붙게 되는 Prefix로 인해 일반적인 데이터보다 많은 캐릭터(Character)를 가진 문자열의 형태를 가진다. 그리고 이러한 긴 문자열의 형태는 데이터를 서로 조인(Join)할 때 많은 캐릭터 간 비교로 인해 높은 오버헤드를 발생시키고, 분산된 노드 환경에서는 높은 전송 비용을 발생시키는 문제점을 가진다.

이러한 문제점으로 인해 많은 RDF 저장소들은 데이터를 압축하여 데이터 검색에 효율적인 레이아웃 형태로 저장한다[4-7]. 그러나 제시된 저장소들은 대량 RDF 데이터를 로딩할 때 ID 매핑 테이블(ID Mapping Table) 크기, 테이블 분할, 인덱스 생성 등으로 인한 높은 처리 시간의 오버헤드가 발생하는 한계점을 가진다. 그 예로, 가장 대표적인 RDF 저장소인 RDF-3X는 약 7억개의 LUBM 데이터를 로딩할 때 8842초의 처리 시간을 소비한다 (5장).

본 연구팀은 분산 병렬 프레임워크인 맵리듀스(MapReduce)[8]를 기반으로 대량 RDF 데이터에 대한 압축 변환에 대한 연구를 진행한 바 있다[9]. 맵리듀스는 분산된 서버를 사용하여 대용량의 데이터를 빠르게 처리할 수 있도록 하는 분산 병렬 프레임워크로, 맵(Map)과 리듀스(Reduce), 두 단계의 고정된 데이터 플로우를 사용하여 큰 데이터를 나누고 특정한 규칙에 따라 다시 조인함으로써, 대용량 데이터에 대한 작업을 간단하고 빠르게 수행할 수 있도록 한다.

본 논문에서는 선행 연구에서 제안한 압축 변환 방법에 더하여, 맵리듀스를 기반으로 N-Triple [10], Turtle [11] 형식의 대량 RDF 데이터를 압축 변환하고, 압축 변환된 대량 RDF 데이터를 서술어 기반의 수직 분할 테이블(Vertical Partitioned Table)로 저장하는 방법을 제안한다. 또한, 본 논문에서 제안한 방법의 성능을 평가하기 위해, WatDiv[12], LUBM[13], YAGO[14] 데이터셋에 대해 기존 RDF 저장소와 데이터 압축 변환 및 저장 시간을 비교한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제시하는 맵리듀스 기반 대량 RDF 압축 변환 및 저장 시스템의 구성을 설명한다. 3장에서는 맵리듀스를 사용한 RDF 압축 변환 방법에 대해 기술하고, 4장에서는 맵리듀스를 사용하여 서술어를 기반으로 RDF를 수직 분할하는 방법에 대해 기술한다. 5장에서는 제시한 방법의 성능 평가를 위해 타 RDF 저장소와 비교한 결과를 설명하며, 6장에서 결론을 맺는다.

Ⅱ. RDF 변환 및 저장 시스템 구성

그림 2와 같이, 본 논문에서 제안하는 맵리듀스 기반 RDF 압축 변환 및 저장 시스N템은 총 세 개의 단계로 구성된다. 첫 번째 단계는 '*글로벌 ID 생성 맵리듀스 잡 (MapReduce Job)*'으로, Triple과 Turtle 형식으로 저장된 RDF 데이터의 주어, 서술어, 목적어에 위치한 자원을 식별하고 고유 글로벌 ID를 생성한다. 또한, RDF 데

이터를 각 매퍼(Mapper) ID 기준으로 블록으로 나누어 저장한다. 두 번째 단계는 '*RDF 데이터 압축 변환 맵리듀스 잡*'으로, 첫 번째 단계에서 생성한 글로벌 ID를 사용하여 해시(Hash) 기반 ID 매핑 테이블을 생성하고, 이를 사용하여 매칭되는 RDF 데이터 블록을 압축 변환한다. 마지막 단계는 '*RDF 수직 분할 저장 맵리듀스 잡*'으로, 압축 변환 RDF 데이터에 대해 서술어를 기반으로 데이터를 수직 분할하여 저장한다.

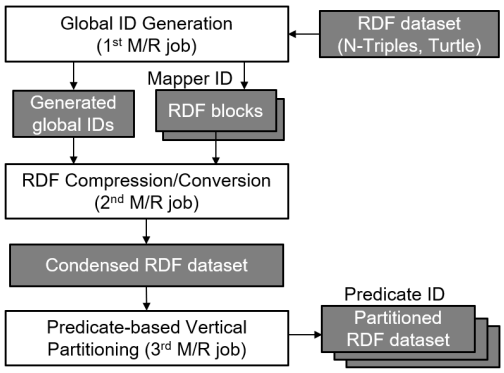

Fig. 2 RDF conversion & storage system overview

최종적으로 정수 ID로 압축되어, 서술어를 기반으로 여러 파일로 분할된 RDF 데이터가 본 논문의 제안 시스템의 결과물로 HDFS(Hadoop Distributed File System)에 저장된다.

Ⅲ. 맵리듀스 기반 RDF 압축 변환

본 장에서는 두 개의 맵리듀스 잡을 기반으로 대량 RDF 데이터를 정수 ID로 압축 변환하는 방법에 대해 기술한다.

3.1. 글로벌 ID 생성 맵리듀스 잡 (Global ID Generation MapReduce Job)

먼저, 글로벌 ID 생성 맵리듀스 잡은 압축 변환 대상 RDF 데이터를 입력받는다. 이때 RDF 데이터는 트리플 형태 그대로 저장하는 N-Triple 형식과, 데이터의 중복 부분을 제거하고 Prefix의 약어를 사용하여 압축된 형태로 저장하는 Turtle 형식이 입력될 수 있다.

N-Triple 데이터의 경우, 잡의 각 매퍼는 하나의 <주어, 서술어, 목적어> 트리플로 구성된 RDF 데이터를 입력받는다. 그림 3의 상단은 N-Triple 형식의 RDF 데이터가 입력되었을 매퍼의 동작과정을 보여준다. 매퍼는 입력받은 데이터에서 주어, 서술어, 목적어에 위치한 요소 값들을 추출하고, 각 요소 값을 잡 중간 결과의 키(Key)로, 매퍼 ID를 중간 결과의 밸류(Value)로 출력한다. 동시에 매퍼는 입력받은 트리플들을 임시로 메모리에 보유했다가, 매퍼가 종료될 때 매퍼의 ID를 이름으로 하여 해당 트리플 그룹을 하나의 파일로 저장한다(i.e., RDF 데이터 블록). 각 요소 값에 대한 매퍼 ID를 페어로 출력하였기 때문에, 이후 RDF 데이터 압축 변환 과정에서 요소 값들이 어떤 데이터 블록에 속하는지 확인할 수 있다.

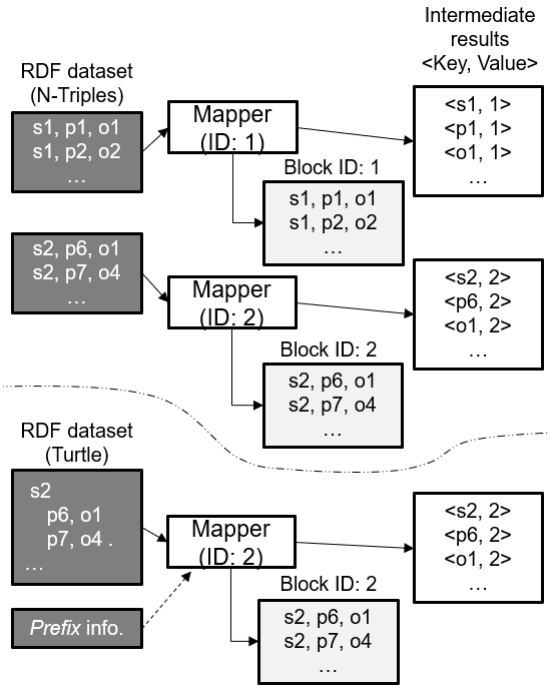

Fig. 3 The map process of the 1st MapReduce job

그림 3의 하단은 Turtle 형식의 RDF 데이터를 입력받았을 때 매퍼의 동작 과정을 보여준다. Turtle 데이터의 경우, 각 매퍼는 라인으로 구분되는 N-Triple 데이터와 다르게 마침표로 구분되는 트리플 그룹 단위를 입력받는다. 매퍼는 입력받은 트리플 그룹과 Prefix 목록을 결합하여 N-Triple 형태의 여러 트리플로 변환하며, 이후 N-Triple 데이터와 같은 방식으로 처리한다.

매퍼가 출력한 중간 결과들은 같은 요소 값을 기준으로 여러 매퍼 ID가 그룹핑되어 각 리듀서(Reducer)에 전달된다. 각 리듀서는 전달받은 요소 값에 대해 카운트를 증가하여(i.e., Increased Count) 로컬 정수 ID를 생성한다. 그러나 로컬 ID는 다른 리듀서와 중복될 가능성이 있으므로, 리듀서의 태스크 ID 1 Byte와 로컬 ID 7 Byte를 결합하여 8 Byte로 글로벌 ID를 생성하면 중복 없이 요소 값에 대한 식별이 가능하다. 리듀서는 이를 기반으로 <요소 값, 글로벌 ID, 매퍼 ID 그룹>을 최종 결과로 출력한다. 그림 4는 이러한 리듀서의 동작 과정을 나타내며, 리듀서 1이 요소 값 's1'에 대해 리듀서의 태스크 ID '1'과 로컬 ID '1'을 결합하여 글로벌 ID를 생성한다. (본 예시에서는 간단한 예시 표현을 위해 단순하게 두 정수를 문자열로 결합한 결과인 '11'로 글로벌 ID를 표현하였으나, 실제로는 두 정수 값을 byte 기준으로 결합하여 8byte의 정수를 도출한다.)

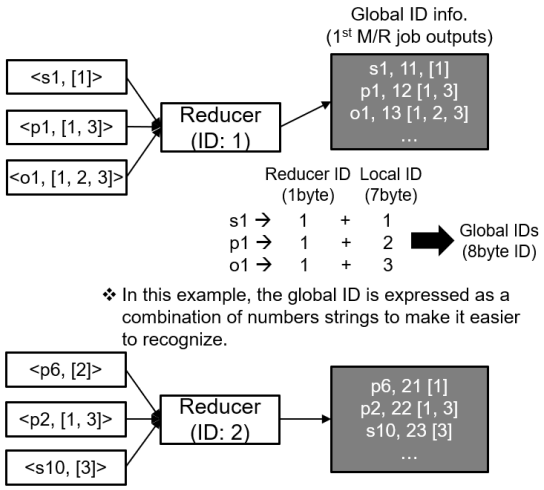

Fig. 4 The reduce process of the 1st MapReduce job

3.2. RDF 압축 변환 맵리듀스 잡 (RDF Compression and Conversion MapReduce Job)

첫 번째 맵리듀스 잡에서 생성된 결과를 사용하여, 두 번째 맵리듀스 잡은 대상 RDF 데이터를 압축 변환하는 과정을 수행한다.

잡의 각 매퍼는 첫 번째 잡에서 생성한 글로벌 ID 정보를 입력받는다. 입력 데이터는 요소 값에 대한 고유 식별자인 글로벌 ID와, 첫 번째 잡의 어떤 매퍼들이 해당 요소 값을 전송했는지 알 수 있는 매퍼 ID 그룹으로 구성된다. 두 번째 잡의 매퍼는 이러한 ID 그룹을 매퍼 ID 단위로 나누어서 <매퍼 ID, 요소 값, 글로벌 ID>를 중간 결과로 출력한다. 그림 5에서 매퍼는 요소 값 'o1'에 대한 첫 번째 잡의 매퍼 ID 그룹 '[1, 2, 3]'을 읽고 각 매퍼 ID를 기준으로 'o1'에 대한 정보를 중간 결과로 출력한다.

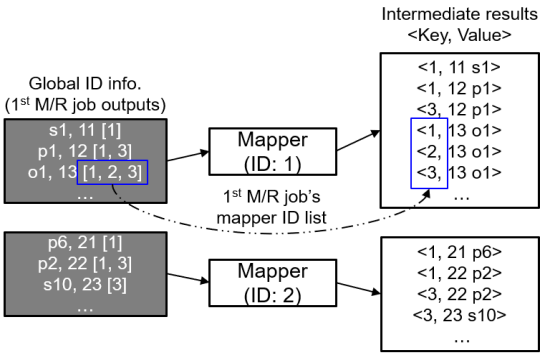

Fig. 5 The map process of the 2nd MapReduce job

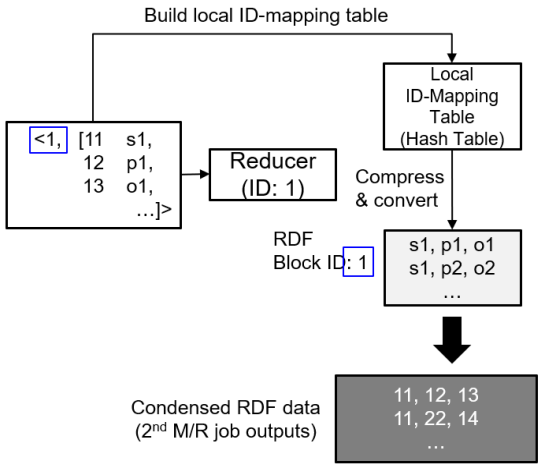

Fig. 6 The reduce process of the 2nd MapReduce job

중간 결과는 키로 출력된 매퍼 ID를 기준으로 그룹핑되어 리듀서에게 전달된다. 리듀서는 전달받은 매퍼 ID를 확인하고 첫 번째 맵리듀스 잡에서 생성한 RDF 데이터 블록에서 해당 매퍼 ID와 일치하는 블록을 읽는다. 매퍼 ID가 같다는 것은, 전달받은 요소 값들이 블록의 트리플에 포함되었다는 것을 의미한다.

따라서 그림 6과 같이 리듀서는 전달받은 요소 값을 해시 테이블의 해시 키로, 매핑되는 글로벌 ID를 밸류로

설정하여 해시 기반 로컬 ID 매핑 테이블을 생성한다. 해시 테이블을 사용하여 매칭되는 문자열에 대한 글로벌 ID를 O(1)의 복잡도로 추출할 수 있으므로, 리듀서가 전달받은 매퍼 ID에 해당되는 RDF 데이터 블록의 트리플들을 정수 ID로 빠르게 압축 변환한다.

RDF 데이터를 압축하기 위해 많은 저장소들이 <키, 밸류>로 구성된 딕셔너리 형태의 ID 매핑 테이블을 사용하여 데이터를 변환한다. 이러한 테이블 구조는 빠른 RDF 데이터 변환에 효율적이지만, 대량의 RDF 데이터의 경우 너무 많은 <키, 밸류> 페어가 입력되어 있을 때 해시 테이블과 같이 지속적인 테이블의 리사이즈(Resize)로 인해 느린 처리 속도를 보인다. 본 논문에서 제안한 방법은 분산 환경에서 RDF 데이터를 분할하고, 분할한 데이터 변환에 필요한 ID 정보만으로 해시 기반 ID 매핑 테이블을 생성함으로써, 대량 RDF 데이터 압축 변환의 처리 속도 오버헤드를 개선한다.

Ⅳ. 맵리듀스 기반 RDF 수직 분할 저장

본 장에서는 압축 변환된 RDF 데이터를 맵리듀스를 사용하여 서술어 기반으로 수직 분할하는 방법에 대해 기술한다.

많은 RDF 저장소들이 단일 트리플 저장소(Single Triple Store)[4], 술어 기반 테이블(Property Table), 수직 분할 테이블(Vertical Partitioned Table), 비트맵 기반 저장소(Bitmat Store)[15, 16]와 같이 적은 저장공간을 소비하고, 빠르게 데이터를 검색하기 위한 효율적인 저장 방식을 제안하였다. 본 논문에서는 맵리듀스를 사용하여 빠르게 데이터를 저장하고, 쿼리 요청 시 보통 서술어는 쿼리에서 변수를 가지지 않으므로, 서술어를 사용하여 빠르게 데이터를 검색할 수 있는 서술어 기반의 수직 분할 저장을 수행한다.

잡의 각 매퍼는 앞의 단계에서 정수 ID로 압축 변환된 RDF 데이터를 트리플 단위로 입력받는다. 매퍼는 입력받은 트리플에서 서술어 값을 추출하여 중간 결과의 키로 설정하고, 해당 트리플의 주어, 목적어 값을 밸류로 설정하여 중간 결과를 출력한다. 이 과정을 수행하여 동일한 서술어를 가진 트리플을 그룹핑할 수 있다.

그러나 실세계의 RDF 데이터는 각 서술어를 기준으로 했을 때 트리플 수가 유사하지 않은, 높은 데이터 불균형(i.e., High Data Skew)의 형태를 가진다. 예를 들어, 대학에 관련된 RDF 데이터의 경우 교수의 강의를 나타내는 서술어인 'lectureOf'에 비해, 학생, 교수, 학교, 건물, 강의 등의 이름을 나타내는 서술어인 'name'의 트리플 수가 압도적으로 많을 수밖에 없다. 이러한 데이터의 높은 불균형은 가장 마지막으로 실행이 끝나는 태스크 실행 시간에 의해 잡의 전체 실행 시간이 결정되는 맵리듀스 특성으로 인해[17], 결과적으로 높은 실행 시간을 발생시킨다.

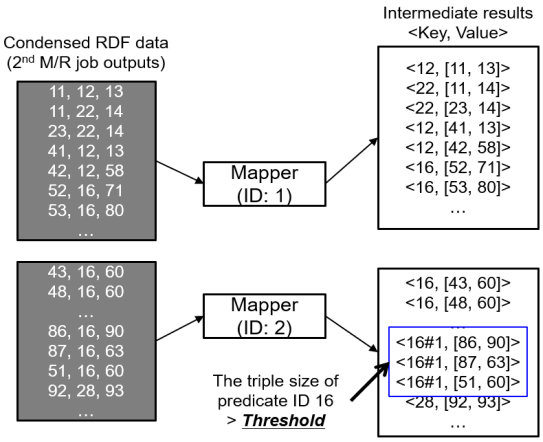

Fig. 7 The map process of the 3rd MapReduce job

본 논문에서는 이러한 한계점을 개선하기 위해, 일정 Threshold 이상으로 하나의 서술어에 해당되는 트리플 수가 포함될 경우, 해당 서술어의 트리플들을 여러 리듀서에 분할하여 전송하는 방법을 사용한다. 그러나 매퍼들은 서로의 데이터에 대한 정보를 알 수 없으므로, 최종적으로 리듀서에 전달되는 한 서술어의 트리플 행 수를 정확하게 알기 어려운 문제점을 가진다. 따라서 매퍼는 동일 서술어의 트리플들이 균등하게 매퍼에게 입력되는 것을 가정하여, $|M|$이 전체 매퍼 M의 수를, $|R|$이 전체 리듀서 R의 수를, $ratio$가 트리플 단위 파티셔닝을 위한 scale factor를 의미할 때, $1/(|M|\cdot|R|\cdot ratio)$로 계산되는 Threshold를 도출한다. 매퍼는 입력된 서술어의 트리플 행 수가 Threshold를 초과할 경우, 중간 결과의 키를 다르게 하여 다른 리듀서에 분할 전송되도록 한다.

이러한 과정은 그림 7에 자세히 나타나며, 예시에서 매퍼 2는 서술어 '16'에 대한 트리플 수가 Threshold를 초과하여, 이후 입력된 '16'의 트리플에 대해서는 다르게 키를 부여함으로써 다른 리듀서에 전송되도록 한다.

Table. 1 The statistics of RDF datasets

Dataset	# of triples	# of distinct subjects	# of distinct objects	# of distinct predicates	Size (GB)
WatDiv-100M	109,904,729	5,212,385	17,922,685	86	14.64
WatDiv-500M	549,630,643	26,060,385	89,563,145	86	74.02
LUBM-700M	690,895,862	108,498,628	246,519,412	18	115.85
YAGO-250M	244,800,060	72,048,816	84,731,249	104	37.02

Table. 2 The comparison of the approaches of loading RDF datasets

Comparative systems	Approaches
RDF-3X[4]	Load RDF datasets using ID mapping tables based on a single server
SHAPE[5]	Load RDF datasets using ID mapping tables based on Hadoop (Base system: RDF-3X)
H2RDF+[7]	Load RDF datasets using the compression method of HBase based on Hadoop
Our system	Load RDF datasets using partitioned hash-based ID mapping tables based on Hadoop

Table. 3 Performance comparison (The total loading time of an RDF dataset, sec)

Dataset	RDF-3X[4]	SHAPE[5]	H2RDF+[7]	Our system
WatDiv-100M	1594	3658	337	135
WatDiv-500M	9233	59110	651	365
LUBM-700M	8842	12248	649	429
YAGO-250M	4012	2625	853	230

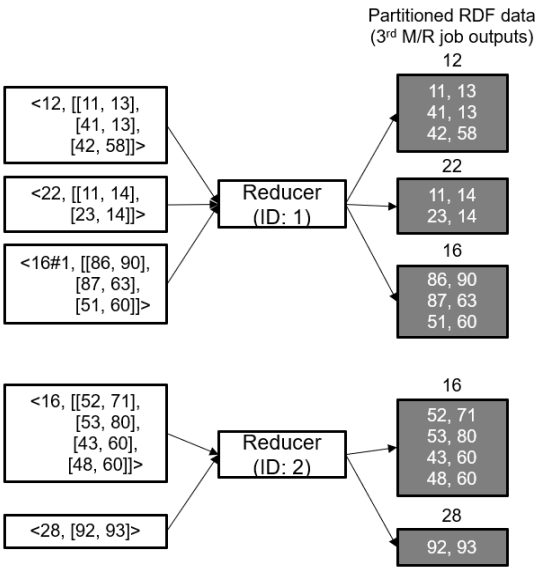

Fig. 8 The reduce process of the 3rd MapReduce job

그 결과로 그림 8과 같이 리듀서는 같은 서술어로 그룹핑된 주어, 목적어 리스트를 전달받으며, 각 서술어 기준으로 결과 파일을 HDFS에 출력한다.

Ⅴ. 실 험

본 장에서는 본 논문에서 제안한 압축 변환 및 저장 방법의 성능을 평가하기 위해, 기존 RDF 저장소와 RDF 데이터의 로딩 시간을 비교한 실험 결과를 기술한다.

본 실험에서는 Xeon E5-2620 2,1GHz CPU, 64GB RAM, 1TB 7200RPM HDD의 스펙을 가지는 15대를 사용하여 실험을 진행하였으며, 이 중 1대를 마스터 서버로, 14대를 슬레이브 서버로 사용하였다. 또한, OS는 우분투(Ubuntu) 18.04, 하둡(Hadoop)은 1.2.1 버전을 사용하였다. 실험을 위해 대표적인 RDF 벤치마크 데이터인 WatDiv, LUBM 데이터를 사용하였으며, 추가로 실세계 데이터로 구성된 RDF 데이터인 YAGO 데이터를 사용하였다. 표 1은 실험 데이터에 대한 구체적인 통계 정보를 나타내며, 표 2는 비교 대상 기존 RDF 저장소의 RDF 데이터 로드 방법에 대해 기술한 것이다.

표 3은 본 논문의 제안 방법을 사용했을 때 로딩 시간(i.e., 압축 변환 및 저장)과 다른 RDF 저장소의 로딩 시간을 비교한 결과이다. 실험 결과에서 단일 노드 환경 RDF 저장소인 RDF-3X는 세 개의 데이터셋에 대해 최소 1594, 최대 9233초의 로딩 시간을 가지며, 단일 노드

환경으로 인해 데이터의 크기가 증가할 경우 처리 시간이 급격하게 증가한다. RDF-3X를 맵리듀스 기반 분산 저장소로 변환한 SHAPE의 경우 로딩 시 파티셔닝으로 인해 RDF-3X에 비해 더 많은 시간을 소비한다. 맵리듀스를 사용한 다른 RDF 저장소인 H2RDF+는 최소 337초, 최대 853초로 다른 저장소들에 비해 가장 빠른 로딩 시간을 보인다. H2RDF+는 써드파티(3rd party) 저장소인 HBase에서 제공하는 압축 방식을 사용하여 RDF 데이터를 압축 변환하여 저장한다.

본 논문에서 제안한 방법은 세 개의 데이터 모두에 대해 최소 135초, 최대 429초로, 실험한 저장소 중 가장 빠른 로딩 시간을 보이며, 단일 환경 저장소인 RDF-3X와 비교하였을 때 최대 25.2배, 분산 저장소인 H2RDF+와 비교하였을 때도 최대 3.7배 빠른 성능을 가진다. 본 논문의 제안 방법은 데이터 변환 과정을 분산 환경에서 수행함으로써 단일 환경 저장소가 가지는 확장성 문제를 개선하고, 해시 기반 ID 매핑 과정을 분산 환경에서 수행함으로써 대용량 RDF 데이터를 빠르게 변환할 수 있었다.

Ⅵ. 결 론

본 논문에서는 기존의 RDF 저장소들이 대량 RDF 데이터를 압축 변환하고 저장할 때 발생하는 로딩 시간의 문제점을 개선하기 위해 분산 병렬 프레임워크인 맵리듀스를 기반으로 대량 RDF 데이터를 압축 변환하는 방법을 제시하였다. 또한, 데이터를 빠르게 압축하여 저장하고 쿼리 요청 시 서술어 기반으로 빠르게 데이터를 검색하기 위해 맵리듀스 기반의 RDF 데이터 수직 분할 방법을 제안하였다. 본 논문의 실험 결과, 본 논문에서 제안한 방법으로 RDF 데이터를 압축 변환 및 저장하였을 때, H2RDF+ 대비 최대 3.7배, RDF-3X 대비 최대 25.2배 로딩 시간 성능을 향상한 것을 확인할 수 있었다.

향후 연구에서는 압축 변환 및 저장한 대량 RDF 데이터에 대해, 맵리듀스를 기반으로 SPARQL 쿼리 처리 속도를 향상하는 방안을 연구할 계획이다.

ACKNOWLEDGEMENT

This work was partly supported by KISTI (K-21-L04-C03-S04) and the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (1711101951).

References

[1] Web Data Commons, Microdata, RDFa, JSON-LD, and Microformat Data Set [Internet], Available: http://webdatacommons.org/structureddata/index.html#results-2021-1

[2] W. Ali, M. Saleem, B. Yao, A. Hogan, and A. -C. N. Ngomo, “A survey of RDF stores & SPARQL engines for querying knowledge graphs,” *The VLDB Journal*, pp. 1-26, Nov. 2021.

[3] W3C, Resource description framf theework (rdf) model and syntax specification [Internet], Available: https://www.w3.org/TR/1998/WD-rdf-syntax-19980819/

[4] T. Keumann and G. Weikum, “RDF-3X: a RISC-style engine for RDF,” in *Proceedings of VLDB Endowment*, Auckland, New Zealand, vol. 1, iss, 1, pp. 647-659, Aug. 2008.

[5] K. Lee, L. and Liu, “Scaling queries over big RDF graphs with semantic hash partitioning,” in *Proceedings of the VLDB Endowment*, Trento, Italy, vol. 6, no. 14, pp. 1894-1905, 2013.

[6] F. Goasdoué, Z. Kaoudi, I. Manolescu, J. -A. Quiané-Ruiz, and S. Zampetakis, “CliqueSquare: Flat plans for massively parallel RDF queries,” in *2015 IEEE 31st International Conference on Data Engineering*, Seoul, South Korea, pp. 771-782, 2015.

[7] N. Papailiou, D. Tsoumakos, I. Konstantinou, P. Karras, and N. Koziris, “H2rdf+ an efficient data management system for big rdf graphs,” in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, Utah, USA, pp. 909-912, Jun. 2014.

[8] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, Jan. 2008.

[9] I. A. Kim and K. -C. Lee, “Conversion of Large RDF Data using Hash-based ID Mapping Tables,” in *Proceedings of the Korean Institute of Information and Commucation Sciences Conference*, Gunsan South Korea, pp. 236-239,

2021.
[10] W3C, RDF 1.1 N-Triples [Internet], Available: https://www.w3.org/TR/n-triples/
[11] W3C, RDF 1.1 Turtle [Internet], Available: https://www.w3.org/TR/turtle/
[12] University of Waterloo, Waterloo SPARQL Diversity Test Suite (WatDiv) v0.6 [Internet], Available: https://dsg.uwaterloo.ca/watdiv/
[13] SWAT, The Lehigh University Benchmark (LUBM) [Internet], Available: http://swat.cse.lehigh.edu/projects/lubm/
[14] Max-Planck-Institute Saarbrücken, YAGO: A High-Quality Knowledge Base [Internet], Available: https://yago-knowledge.org/
[15] M. Wylot, M. Hauswkrth, P. Cudré-Mauroux, and S. Sakr, "RDF data storage and query processing schemes: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1-36, 2018.
[16] B. B. Mahria, I. Chaker, and , A. Zahi, "An empirical study on the evaluation of the RDF storage systems," *Journal of Big Data*, vol. 8, no. 1, pp. 1-20, 2021.
[17] K. L. Bawankule, Q. K. Dewang, and A. K. Singh, "Historical data based approach to mitigate stragglers from the Reduce phase of MapReduce in a heterogeneous Hadoop cluster," *Cluster Computing*, pp. 1-19, Feb. 2022.

김인아 (InA Kim)

충남대학교 컴퓨터공학과 박사과정 재학중
충남대학교 컴퓨터공학과 공학석사
충남대학교 컴퓨터공학과 공학학사
※관심분야 : 데이터베이스, 분산 병렬 처리, 빅데이터, 데이터 분석

이경하 (Kyong-Ha Lee)

정보과학회 이사 및 KDBC2022 조직위원장
UST연합대학원 부교수
정보과학회지 편집위원
정보과학회 DB소사이어티 상임이사
한국과학기술정보연구원 국가과학기술데이터본부전략팀장
ETRI 지능형미디어융합연구부 선임연구원
KAIST 전산학과 연구조교수
미국 애리조나대학교 포스닥
충남대학교 SW연구소 전임연구원
정보과학회 종신회원
충남대학교 컴퓨터공학과 공학박사
※관심분야: 기계학습 시스템, 분산/병렬 처리, 데이터베이스 시스템

이규철 (Kyu-Chul Lee)

충남대학교 컴퓨터공학과 교수
서울대학교 전산학과 공학박사
서울대학교 전산학과 공학석사
서울대학교 전산학과 공학학사
※관심분야 : 데이터베이스, 인공지능, 빅데이터, 데이터 분석