맵리듀스 잡을 사용한 해시 ID 매핑 테이블 기반 대량 RDF 데이터 변환 방법

김인아 · 이규철*

충남대학교

Conversion of Large RDF Data using Hash-based ID Mapping Tables with MapReduce Jobs

InA Kim · Kyu-Chul Lee*

ChungNam National University

E-mail : dodary0214@gmail.com / kclee@cnu.ac.kr

요 약

AI 기술의 성장과 함께 지식 그래프의 크기는 지속적으로 확장되고 있다. 지식 그래프는 주로 트리플이 연결된 RDF로 표현되며, 많은 RDF 저장소들이 RDF 데이터를 압축된 형태의 ID로 변환한다. 그러나 RDF 데이터의 크기가 특정 기준 이상으로 클 경우, 테이블 탐색으로 인한 높은 처리 시간과 메모리 오버헤드가 발생한다. 본 논문에서는 해시 ID 매핑 테이블 기반 RDF 변환을 분산 병렬 프레임워크인 맵리듀스에서 처리하는 방법을 제안한다. 제안한 방법은 RDF 데이터를 정수 기반 ID로 압축 변환하면서, 처리 시간을 단축하고 메모리 오버헤드를 개선한다. 본 논문의 실험 결과, 약 23GB의 LUBM 데이터에 제시한 방법을 적용했을 때, 크기는 약 3.8배 가량 줄어들었으며 약 106초의 변환 시간이 소모되었다.

ABSTRACT

With the growth of AI technology, the scale of Knowledge Graphs continues to be expanded. Knowledge Graphs are mainly expressed as RDF representations that consist of connected triples. Many RDF storages compress and transform RDF triples into the condensed IDs. However, if we try to transform a large scale of RDF triples, it occurs the high processing time and memory overhead because it needs to search the large ID mapping table. In this paper, we propose the method of converting RDF triples using Hash-based ID mapping tables with MapReduce, which is the software framework with a parallel, distributed algorithm. Our proposed method not only transforms RDF triples into Integer-based IDs, but also improves the conversion speed and memory overhead. As a result of our experiment with the proposed method for LUBM, the size of the dataset is reduced by about 3.8 times and the conversion time was spent about 106 seconds.

키워드

RDF, Knowledge Graph, MapReduce, Big Data, Parallel Processing

Ⅰ. 서 론

최근 몇 년 동안 AI(Artificial Intelligence) 기술이 발전하면서, 지식 그래프(Knowledge Graph)는 AI 모델의 성능을 향상시키기 위한 기반 데이터로 활용되고 있다. AI 기술의 성장과 함께 지식 그래프의 크기는 점차 증가하여 하나의 데이터셋이 180억개가 넘는 링크를 가지는 수준까지 확장되었다 [1]. 지식 그래프는 주로 주어(Subject), 서술어(Predicate), 목적어(Object)로 구성된 트리플(Triple)들을 가지는 RDF(Resource Description Framework)로 표현된다. RDF의 각 요소에는 URI(Uniform

* corresponding author

Resource Identifier), 리터럴(Literal), 블랭크 노드(Blank node)가 위치할 수 있으며, RDF 데이터를 저장할 때 URI나 리터럴 형태는 긴 길이를 가진 문자열 형태로 표현된다.

긴 문자열은 RDF 데이터를 조인할 때 많은 캐릭터들의 비교로 인한 오버헤드를 발생시키기 때문에, 많은 저장소들이 RDF 데이터를 더욱 압축된 형태로 변환하여 저장한다 [2, 3, 4, 5]. 그러나 제시된 저장소들은 1) 대량 RDF 데이터 변환할 때, ID 매핑 테이블(ID mapping table)의 크기로 인한 높은 처리 시간과 메모리 오버헤드가 발생하거나, 혹은 2) 써드파티(3rd party) 데이터베이스를 사용하여 변환함으로써 높은 복잡성을 가진다. 본 논문에서는 빠른 속도로 대량 RDF 데이터를 정수 기반 ID로 변환하는 방법을 제시한다. 제시한 방법은 분산 병렬 프레임워크인 맵리듀스(MapReduce)를 사용하여 대량 RDF 데이터 변환 시 처리 시간을 단축하고 메모리 오버헤드를 개선한다. 또한, 각 노드가 분산된 RDF 데이터 블록에 매핑되는 로컬 해시(Hash) 기반 ID 매핑 테이블을 만들어 사용함으로써 RDF 데이터의 변환 속도를 증가시킨다.

II. 맵리듀스를 사용한 ID 매핑 테이블 기반 RDF 변환

많은 저장소들은 RDF 변환을 위해 RDF 데이터의 주어, 서술어, 목적어에 위치한 내용(Content)과, 내용에 매핑되는 ID를 저장하는 ID 매핑 테이블을 생성한다. 그러나 이는 변환하고자 하는 RDF 데이터의 크기가 클 경우, 테이블 탐색으로 인한 높은 처리 시간과 메모리 오버헤드가 발생한다. 예를 들어, 해시와 같은 방법을 사용하여 ID 매핑 테이블을 생성할 경우 검색 복잡도는 *O(1)*이지만, 해시 키(Key)가 증가할 경우 해시 블록 리사이즈(Resize)로 인해 처리 시간이 증가하게 되며, 해시 키와 값(Value)을 저장하기 위한 메모리 오버헤드가 발생한다.

본 논문은 이와 같은 해시 ID 매핑 테이블 기반 RDF 데이터 변환 과정을 분산 병렬 프레임워크인 맵리듀스에서 분산 병렬로 처리함으로써, RDF 데이터를 정수 기반 ID로 압축 변환함과 동시에 처리 시간을 단축하고 메모리 오버헤드를 개선한다. 맵리듀스를 사용한 ID 매핑 테이블 기반 RDF 데이터 변환은 다음과 같이 두 개의 맵리듀스 잡(Job)으로 나누어 수행된다.

글로벌 ID 생성 잡 (1st MapReduce Job) 글로벌 ID 생성 잡은 전체 RDF 데이터에 포함된 주어, 서술어, 목적어를 중복 없는 내용으로 추출하여, 각 내용을 구별할 수 있는 정수 기반 글로벌 ID를 부여한다. 먼저 글로벌 ID 생성 잡의 맵(Map) 과정에서, 매퍼(Mapper)들은 입력 RDF 데이터들을 일정한 크기의 블록으로 나누어 입력받는다. 매퍼는 데이터를 주어, 서술어, 목적어 내용으로 분할하고, 각 내용을 중간 결과의 키로, 매퍼의 태스크 ID를 중간 결과의 값으로 작성한다. 또한, 입력받은 데이터 블록에 매퍼 태스크의 ID를 부여하여 HDFS에 블록들을 작성한다.

매퍼들이 작성한 중간 결과는 같은 키, 즉 같은 내용끼리 그룹핑되어 리듀스(Reduce) 과정의 리듀서(Reducer)들에게 전달된다. 리듀서는 같은 내용을 가진 매퍼 ID 리스트를 전달받게 되며, 매퍼 ID들은 해당 내용이 어떤 매퍼 ID의 블록에서 비롯되었는지 가리킨다. 리듀스 과정에서, 먼저 각 리듀서는 전달된 내용에 매핑되는 로컬 정수 기반 ID를 생성한다. 그러나 로컬 정수 기반 ID는 다른 리듀서에 배정된 내용과 중복될 가능성이 높기 때문에, 리듀서의 태스크 ID(1byte)와 로컬 정수 ID(7byte)를 결합함으로써 중복되지 않는 글로벌 정수 기반 ID를 생성하여 내용에 부여한다. 글로벌 ID를 생성한 후, 리듀서는 내용, 내용이 비롯된 매퍼 ID 리스트, 글로벌 ID를 잡의 최종 결과로 HDFS에 작성한다.

[그림 1]은 글로벌 ID 생성 잡의 맵과 리듀스 동작 예시를 나타낸다.

RDF 데이터 변환 잡 (2nd MapReduce Job) RDF 데이터 변환 잡은 첫 번째 맵리듀스 잡에서 생성한 결과인 글로벌 ID를 사용하여 RDF 데이터를 정수 기반 ID로 변환하는 작업이다.

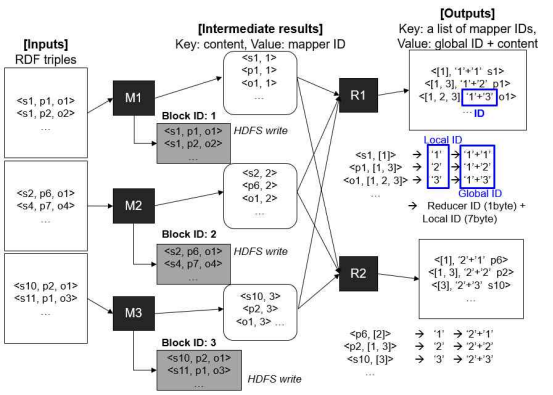

그림 1. 글로벌 ID 생성 잡의 동작 과정 예시

첫 번째 잡은 내용과, 내용이 비롯된 매퍼 ID 리스트, 매핑되는 글로벌 ID를 잡의 결과로 작성하였다. RDF 데이터 변환 잡의 맵 과정에서 매퍼들은 이러한 첫 번째 잡의 결과를 입력받으며, 각 매퍼는 입력받은 내용에 레이블되어있는 첫 번째 잡 매퍼 ID의 리스트를 분할하고, 각 ID를 중간 결과의 키로, 내용과 글로벌 ID를 중간 결과의 값으로 작성한다. 매퍼들이 작성한 중간 결과는 같은 키,

즉 같은 첫 번째 잡의 매퍼 ID끼리 그룹핑되어 RDF 데이터 변환 잡의 리듀스 과정의 리듀서들에게 전달된다. 결과적으로, 리듀서는 같은 첫 번째 잡 매퍼 ID를 가진 여러 개의 내용과 글로벌 ID를 입력받게 된다.

첫 번째 잡에서, 매퍼들은 RDF 데이터를 블록으로 나누어 각 블록에 읽은 매퍼 ID를 레이블한 후 HDFS에 작성하였다. 두 번째 잡에서 같은 첫 번째 잡 매퍼 ID를 가진 내용들은 첫 번재 잡에서 같은 매퍼가 읽은 데이터 블록에 포함되는 내용들을 의미한다. 따라서, 리듀서에게 전달된 같은 매퍼 ID로 그룹핑된 내용과 글로벌 ID를 사용하여, 해당 매퍼 ID로 레이블된 데이터 블록을 ID로 변환할 수 있다. 특정 내용에 해당하는 글로벌 ID를 보다 빠르게 검색하기 위해, 리듀서는 그룹핑된 내용과 글로벌 ID를 로컬화된(Localized) 해시 기반의 ID 매핑 테이블로 생성하며, 내용이 테이블의 해시 키로, 글로벌 ID를 테이블의 값으로 입력된다. 그 후, 리듀서는 HDFS에 저장된 데이터 블록들 중 리듀서가 읽은 첫 번째 잡 매퍼 ID와 동일한 ID를 가진 블록을 읽으며, 생성한 해시 기반 ID 매핑 테이블을 사용하여 블록에 저장된 RDF 데이터들을 글로벌 ID로 매핑하여 변환한다. 변환된 데이터들은 리듀서에 의해 HDFS에 최종 결과로 작성된다.

[그림 2]는 RDF 데이터 변환 잡의 맵과 리듀스 동작 과정을 예시를 나타낸다.

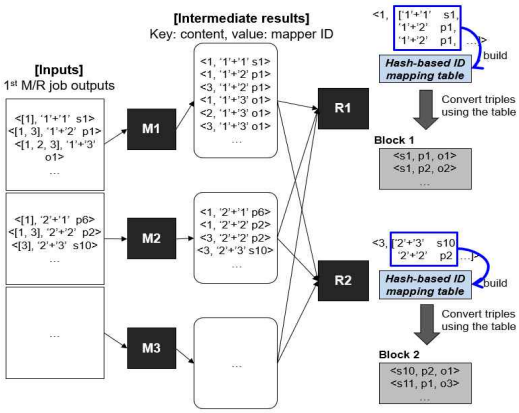

그림 2. RDF 데이터 변환 잡의 동작 과정 예시

앞의 두 개의 맵리듀스 잡을 사용하여, 본 논문에서는 모든 RDF의 내용에 대한 글로벌 ID를 메모리에 저장할 필요 없이, 나누어진 특정 데이터 블록에 사용되는 내용과 글로벌 ID만을 해시 기반 ID 매핑 테이블로 생성하여 메모리에 저장한다. 따라서 분산화된 해시 기반 ID 매핑 테이블을 사용하여 데이터를 변환함으로써 빠른 RDF 변환을 가능하게 하며, 메모리 오버헤드를 줄일 수 있다.

III. 실험 결과

본 논문에서 제시한 방법을 사용하여 RDF 데이터 변환 시간과 변환 후 크기 감소에 대한 실험을 진행하였다. 실험을 진행한 서버는 Xeon E5-2620 2,1GHz CPU, 64GB RAM, 1TB 7200RPM HDD 스펙을 가지고 있으며, 같은 스펙을 가진 15대의 서버에 Hadoop 1.2.1 버전을 설치하였다. 또한, 실험을 위해 RDF 데이터 실험에 널리 활용되는 WatDiv [6], LUBM [7] 데이터셋을 [표 1]과 같이 생성하였다.

표 1. 실험 데이터셋 (WD: WatDiv, LB: LUBM, sf: scale factor)

데이터	원본 크기(GB)	트리플 수
WD (sf:10)	0.14	1,096,060
WD (sf:100)	1.44	10,969,058
WD (sf:1000)	14.64	109,904,729
LB	23.02	138,280,374

[표 2]는 본 논문에서 제시한 방법을 사용하여 RDF 데이터셋을 변환한 후 각 데이터셋의 크기와 변환 시간을 나타낸 결과이다. 실험 결과, WatDiv (sf: 1000)의 경우 원본에 비해 약 3배 가량 크기가 감소되었으며, 약 109초의 변환 시간이 소모되었다. 또한, LBUM의 경우 원본에 비해 약 3.8배 가량 크기가 감소되었고, 약 106초의 시간이 소모되는 것을 알 수 있다.

표 2. 변환 후 크기와 시간 (WD: WatDiv, LB: LUBM, sf: scale factor)

데이터	변환 크기(GB)	변환 시간(ms)
WD (sf:10)	0.05	58,858
WD (sf:100)	0.47	62,385
WD (sf:1000)	4.68	109,644
LB	5.98	106,457

IV. 결 론

본 논문에서는 대량의 RDF 데이터를 압축된 ID로 변환할 때 발생하는 속도와 메모리 오버헤드를 개선하기 위해, 맵리듀스를 사용하여 해시 ID 매핑 테이블을 기반으로 한 RDF 데이터의 변환 방법을 제안하였다. 제안한 방법은 맵리듀스를 사용하여 각 노드에 분산된 데이터 블록에 매핑되는 해시 기반 ID 매핑 테이블만을 메모리에 저장하여 데이터를 변환함으로써, 메모리 오버헤드 없이 빠르게 RDF를 정수 기반 ID로 압축 변환을 가능하게 한다. 본 논문의 실험에서는, LUBM 데이터를 제시한 방법을 이용하여 변환했을 때, 크기는 약 3.8배

가량 줄어들었으며 약 106초의 변환 시간이 소모되었다.

Acknowledgement

본 연구는 국가과학기술연구회에서 시행한 개방형데이터솔루션(DDS) 융합연구단사업 “AI기술을 활용한 공공데이터 기반 지역현안 솔루션 개발 및 실용화-안전안심사회 실현을 위한 실증연구중심으로-”의 지원을 받아 수행된 연구임.

References

[1] Jin, Jiahui, et al. “GStar: an efficient framework for answering top-k star queries on billion-node knowledge graphs.” *World Wide Web,* Vol. 22, No. 4, pp. 1611-1638, 2019.

[2] Neumann, Thomas, and Gerhard Weikum. “RDF-3X: a RISC-style engine for RDF.” *Proceedings of the VLDB Endowment*, Vol. 1, No. 1, pp. 647-659, 2008.

[3] Lee, Kisung, and Ling Liu. “Scaling queries over big RDF graphs with semantic hash partitioning.” *Proceedings of the VLDB Endowment*, Vol. 6, No. 14, pp. 1894-1905, 2013.

[4] Yuan, Pingpeng, et al. “TripleBit: a fast and compact system for large scale RDF data.” *Proceedings of the VLDB Endowment*, Vol. 6, No. 7, pp. 517-528, 2013.

[5] Papailiou, Nikolaos, et al. “H2RDF+ an efficient data management system for big RDF graphs.” *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, Snowbird Utah USA, 2014.

[6] Waterloo SPARQL Diversity Test Suite (WatDiv) [Internet]. Available: https://dsg.uwaterloo.ca/watdiv/.

[7] Guo, Yuanbo, Zhengxiang Pan, and Jeff Heflin. “LUBM: A benchmark for OWL knowledge base systems.” Journal of Web Semantics, Vol. 3, No. 2-3, pp. 158-182, 2005.