클라우드 컴퓨팅을 사용한 대용량 온톨로지 저장을 위한 저장 구조

민영근 ○, 이복주

단국대학교 컴퓨터학과

minyk@dankook.ac.kr, blee@dankook.ac.kr

Data Scheme for Large size Ontology using Cloud Computing

Youngkun Min ○, Bogju Lee

Department of Computer Science and Engineering, Dankook University

요 약

많은 연구로부터 다양한 온톨로지들이 구축되었다. 온톨로지가 표현하는 영역이 점차 넓어짐에 따라 온톨로지의 크기가 증가하였으나 이를 위한 효율적인 저장방법은 연구되지 않았다. 또한 다양한 온톨로지의 사용방법 중 서술 논리를 사용한 추론은 온톨로지의 크기가 작아도 연산이 매우 많이 필요하여 실제로 사용하기가 매우 어렵다. 본 논문에서는 점차 커지는 온톨로지를 효과적으로 저장하기 위하여 온톨로지를 컴퓨터 클라우드에 저장하는 방법과 컴퓨터 클라우드에 저장된 온톨로지를 추론하기 위한 프레임워크를 제안한다. 그리고 실험을 통하여 제안한 방법이 기존의 방법에 비하여 효율적임을 보였다.

1. 서 론

온톨로지는 시맨틱 웹을 구성하는 중요한 부분이다. 온톨로지는 T. R. Gruber의 언급에 따라 "특정 관심영역에 있어서, 정형화되고 명확한 개념의 명세[1]"라고 정의할 수 있으며 인간이 가지고 있는 지식을 기계가 사용할 수 있도록 표현하는 지식 표현의 한 방법이기도 하다. 이러한 온톨로지는 그 근간으로 XML[2]을 사용하며 W3C에서 표준으로 채택한 RDF[3], OWL[4]등의 미리 정의된 어휘를 사용하여 관심영역에 대한 지식을 기술한다. 일반적인 온톨로지는 특정 영역에 대하여 만들어지므로 그 크기가 방대하지 않다. 일례로 시맨틱 웹을 이용한 지식기반 시각 미디어 검색 프레임워크 연구[5]에서 구축된 미술 작품 온톨로지는 스키마 부분과 데이터 부분을 별도의 파일로 구축되었으며, 스키마 부분에는 9개의 개념과 개념들 간의 관계 8 종류를 구축하였으며 데이터 부분은 미술 작가, 미술 작품, 화풍, 주제, 기법으로 나누어져 있으며 500명의 작가와 그들의 대표 작품 500점에 대하여 구축되었고, 작가를 설명하기 위한 관계로 이름, 대표 작품, 화풍의 3개가 관계가 구축되어 있으며 각 관계 당 평균 58바이트의 크기로 구축되었다.

표 1. 미술 작품 온톨로지의 스키마 부분

개념	관계	총 크기(바이트)
9	8	2318

표 2. 미술 작품 온톨로지의 데이터 부분

데이터	갯수	총 크기(바이트)	평균 크기(바이트)	사용한 관계	관계당 평균 크기(바이트)
작가	500 명	88166	176	이름, 대표 작품, 화풍	58
작품	500 점	89332	178	작가, 주제, 기법	59
화풍	34 종	12497	367	작가	-
주제	9 종	21387	2376	작품	-
기법	18 종	639	35	-	-

온톨로지를 서술하는 한 방법인 RDF 3항-구조(triple)의 개수로 보면 한 명의 작가는 6개의 3항-구조로 서술되며, 한 개의 작품은 4개의 3항-구조로 서술되므로 500명의 작가와 500개의 작품은 5000개의 3항-구조로 구성되어 있다. 여기에 화풍, 주제, 기법 등의 데이터와 스키마 부분을 합치면 전체 미술작품 온톨로지는 약 5100여개의 3항-구조로 이루어져 있으며, 이러한 규모는 기존의 DBMS와 온톨로지 추론엔진을 통하여 충분히 사용할 수 있는 크기이다. 하지만 분야에 따라 온톨로지의 크기는 천차만별로 바뀐다. 바이오 의학 분야의 연구자들[6]은 이미 10억 개가 넘는 3항-구조를 다루고 있다. 이러한 대규모의 온톨로지를 저장하기 위해서는 수

십 기가바이트에서 수 페타바이트에 달하는 크기가 요구되며, 이러한 대용량 온톨로지를 사용하는 방법 또한 한대의 컴퓨터에서 처리하는 방식이 아닌 다수의 컴퓨터를 이용하여 분산 컴퓨팅과 같은 병렬 처리방법을 사용하여야 한다. 기존의 많이 사용되고 있는 Jena[7] 같은 온톨로지 추론 엔진들은 확장성의 측면에서는 많은 성능 저하를 보인다. 이러한 프레임워크들은 하나의 컴퓨터상에서 동작하므로 대용량 온톨로지를 저장하고 사용하기에 적합하지 않고, 항시 주 메모리에 사용 중인 온톨로지를 저장함으로서 저장할 수 있는 3항-구조의 개수는 컴퓨터의 주 메모리의 양에 따라 달라진다. Jena의 경우 저장할 수 있는 방법으로 MySQL같은 데이터베이스도 포함되어 있지만 추론과 같은 실제의 사용을 위해서는 컴퓨터의 주 메모리로 모든 데이터들을 불러들여 메모리상에 온톨로지 구조를 만들고 사용자의 질의를 처리한다. 이러한 문제점을 해결하고자 국외에서는 RDF형식의 온톨로지를 다수의 컴퓨터로 구성된 컴퓨터 클라우드에 저장하고 SPARQL[8] 질의어 형식을 통하여 사용하고자 하는 연구[9]가 진행되고 있는 등, 대규모의 온톨로지를 저장하고 사용하려는 연구가 막 시작되고 있다. University of Texas at Dallas에서 진행하였던 위의 연구는 대규모 온톨로지를 "rdf:type"에 따라 여러 개의 분할로 나누어 분산 처리의 입력으로 사용하였으며 온톨로지 질의어인 SPARQL을 여러 개의 작업으로 나누어 Hadoop[10] 분산 처리 프레임워크을 통하여 실행시켜 결과를 얻어 오는 방법을 사용하였다.

대규모 데이터를 손쉽게 다룰 수 있는 확장성은 정보산업에 있어서 중요한 문제 중의 하나이며, 시맨틱 웹이 널리 활용되기 위해서는 대용량 온톨로지를 저장할 수 있는 프레임워크가 절실히 필요하다. 본 논문에서는 대용량 온톨로지를 클라우드 컴퓨팅 프레임워크의 한 종류인 하둡과 이를 바탕으로 구동하는 분산 데이터베이스인 Hbase[11]에 저장하기 위한 데이터 저장구조를 제안하고 제안한 저장구조를 실험을 통하여 효과적임을 입증하였다.

2. 관련연구

2.1 Jena의 저장 구조

기존의 온톨로지 추론엔진 중 대표적인 Jena는 데이터베이스를 사용하여 온톨로지를 보관할 수 있다. Jena가 온톨로지를 보관하는 구조는 다음과 같다.

표 3. Jena의 데이터베이스 저장 구조

필드명	값	의미
Subject	varchar	주어부
Predicate	varchar	서술어부
Object	varchar	목적어부

위의 저장 구조는 Jena 추론엔진에서 온톨로지를 보관만 하는 구조로 사용한다. Jena 추론엔진에서는 온톨로지를 사용하기 위해서는 온톨로지가 저장된 데이터베이스로부터 읽어들여 주 메모리상에 온톨로지 모델로 구축하여야 하므로 데이터베이스 엔진이 제공하는 다양한 검색, 인덱스 기능을 활용하지 못한다.

2.2 하둡 분산 프레임워크

Hadoop은 구글의 분산 컴퓨터 환경을 오픈소스로 구현하고자 하는 목표로 출발한 프로젝트로서 Hadoop 분산 파일 시스템(HDFS)와 이를 기반으로 데이터베이스를 제공하는 Hbase로 구성되어 있다. HDFS는 구글의 GFS[12]를 오픈 소스로 구현한 결과물이며, 대규모의 데이터를 저장하고 네트워크를 통하여 저장된 데이터를 분산되어 있는 수 많은 클라이언트들에게 제공하기 위해 만들어진 파일 시스템이다.

Hbase는 Hadoop 분산 파일 시스템을 사용하는 데이터베이스로서 구글의 Bigtable[13]로 부터 시작된 오픈소스 프로젝트이다. 는 기존의 RDBMS와는 전혀 다른 성격을 가지고 있는 열-기반(Column-based)의 데이터베이스이며 다음과 같은 특징을 가지고 있다. 첫 째, 하나의 행에 해당하는 일련의 값들은 "열 집합(column family)"에 의하여 분류되어 있으며 동일한 열 집합은 같은 prefix를 가진다. 둘 째, 열 집합은 테이블의 최초 생성 시 만들어져야 하지만, 각각의 열들은 필요에 따라 응용프로그램에 의하여 생성될 수 있고 같은 열 집합은 파일 시스템에 물리적으로 같이 저장된다. 셋 째, Hbase는 인덱스가 없다. 테이블의 값은 row key에 의하여 정렬되어 있으므로, 인덱스가 비대해지는 경우가 없고 따라서 삽입 연산의 성능이 테이블의 크기와 무관하다. 넷 째, Hbase는 데이터를 자동적으로 분산한다. 테이블이 커짐에 따라, 몇 개의 영역으로 나뉘어 사용가능한 노드에 자동적으로 분산 저장된다.

본 논문에서는 분산 열 기반 데이터베이스인 Hbase의 특징을 온톨로지의 특징과 잘 연결시킬 수 있는 저장 구조를 제안하였다.

3. 제안하는 저장 구조

본 논문에서 제안하는 저장 구조는 하둡의 분산 데이터베이스인 Hbase에서 적용가능하며, 추론엔진이 효과적으로 사용해야 한다는 점에 초점을 맞추어 설계하였다. 일반 관계형 데이터베이스와는 다른 성격인 Hbase에 데이터를 저장하고 사용하기 위하여 다음과 같은 목표를 수립하였다.

목표 1: 온톨로지의 TBox 부분과 ABox 부분을 나누어 저장한다. TBox는 온톨로지의 구조를 나타내는 부분으로서 ABox에 비하여 사용해야 하는 빈도가 월등히 높으므로 별도로 저장하여야 한다.

목표 2: 주어부가 같은 트리플을 같은 노드에 저장한다. 하나의 주어가 사용되면 그 주어와 연관되어 있는 다른 데이터들도 사용될 확률이 높다.

목표 3: 3가지 주요 질의 형태 3가지(유형 1: [주어, 서술어, ?x], 유형 2: [주어, ?x, 목적어], 유형 3: [?x, 서술어, 목적어])를 효과적으로 검색할 수 있어야 한다.

목표 4: 저장 구조의 크기를 제약하지 않는다. 본 저장구조는 분산 파일시스템과 이를 기반으로 구동하는 분산 데이터베이스를 가정한 것이므로 데이터의 중복을 피하지 않는다. 또한 온톨로지는 한번만 쓰여지고 많이 읽어 사용하는 특성이 있기 때문에 데이터의 중복이 데이터의 일관성에 영향을 미치지 않는다.

목표 5: 온톨로지가 각각의 분산 데이터 노드에 고르게 분산되어 저장 되어야 한다.

이상의 조건을 만족시키기 위하여 다음 표 4와 같은 구조를 제안한다.

표 4. 제안된 온톨로지 저장 구조

행 값(row key)	타임스템프(Timestamp)	열 집합 "predicate"			열 집합 "object"		
		NS#last	NS#work	...	NS#Painting_1	NS#Painting_2	...
NS#Painter_1	5		NS#Painting_5				
	4		NS#Painting_4				
	3		NS#Painting_3				
	2		NS#Painting_2				
	1	LastName1	NS#Painting_1		NS#work	NS#work	
NS#Painter_2	5		NS#Painting_10				
	4		NS#Painting_9				
	3		NS#Painting_8				
	2		NS#Painting_7				
	1	LastName2	NS#Painting_6				

제안하는 저장구조는 Hbase의 행 값에 온톨로지의 주어부를 사용한다. Hbase의 특성상 같은 행 값을 가진 행은 한 곳에 모여 저장된다.(목표 2) 검색의 속도를 위하여 하나의 주어를 기준으로 서술어를 열 이름으로 목적어를 열 값으로 사용하는 열과 목적어를 열 이름으로 서술어를 열 값으로 사용하는 열을 생성한다. 이러한 구조는 데이터의 중복과 저장 공간의 낭비를 가져오지만 온톨로지의 사용상의 특성과 분산 파일 시스템의 사용으로 단점을 보완하면서 더 빠른 검색 속도를 얻을 수 있다.(목표 3, 4) 그리고 동일한 저장 구조를 사용하여 온톨로지의 TBox 부분과 ABox 부분을 나누어 저장(목표 1)하며, 하둡 분산 파일 시스템과 Hbase를 사용함으로서 저장하는 온톨로지가 각각의 분산 노드에 고르게 분산됨을 보증할 수 있다.(목표 5)

또한 같은 주어와 같은 서술어를 사용하여 여러 개의 목적어를 지시할 수 있으므로 Hbase의 타임스템프 기능을 사용하여 같은 주어(행), 같은 서술어(열)에 여러 개의 값을 저장하도록 하였다. 위의 표 에서 주어 “Painter_1”은 서술어 “work”로 모두 다섯 가지의 데이터를 가지고 있음을 의미한다.

제안하는 저장방법을 사용하여 온톨로지를 질의하는 방법은 다음과 같다.

1. 주어부와 서술어부가 주어지며 목적어부를 찾는 경우: 이 경우는 먼저 **row-key** 를 이용하여 주어부와 일치하는 행을 찾고 행의 **column-family** 중 서술어부를 기록한 **"Predicate"** 열에서 일치하는 서술어부를 찾아서 저장되어 있는 목적어부를 반환한다.

2. 주어부와 목적어부가 주어지며 서술어부를 찾는 경우: 이 경우는 **row-key** 를 이용하여 주어부와 일치하는 행을 찾고 행의 **column-family** 중 목적어부를 기록한 **"Object"** 열에서 일치하는 목적어부를 찾아 저장되어 있는 서술어부를 반환한다.

3. 서술어부와 목적어부가 주어지며 주어부를 찾는 경우: 이 경우에는 주어가 주어져 있지 않으므로 바로 검색 할 수 없으며 두 가지 경우로 나누어 검색한다. 첫째, 온톨로지의 구조를 나타내는 **TBox** 에서 주어진 서술어부의 역 관계에 해당하는 서술어가 있는가를 검색하여 존재하면 주어진 목적어를 주어부에 사용하고, 찾아낸 역 관계 서술어를 서술어부에 사용하는 새로운 질의로 변환시킨 후 **1** 번의 과정을 수행한다. 둘 째, 역 관계 서술어를 찾을 수 없는 경우에는 주어진 서술어를

가질 수 있는 주어의 개념(클래스)을 찾고, 모든 행에 대하여 서술어를 가질 수 있는 개념의 인스턴스인가를 먼저 확인하고 확인된 경우는 목적어부를 기록한 "Object" 열에서 일치하는 목적어부가 존재하나 확인하고 존재한다면 결과로서 주어부를 반환한다.

이상의 제안한 저장구조와 제안한 질의 알고리즘은 모두 하둡 분산 프레임워크에서의 구현 가능에 초점을 맞추어 설계하였다.

4. 구현 및 실험

실험을 위하여 코어 2 듀오 2.0 Ghz, 주 메모리 2GB, 하드디스크 100GB 컴퓨터 10 대에 하둡 분산 파일시스템과 Hbase 를 설치하여 컴퓨터 클라우드를 구축하였다. 구축된 컴퓨터 클라우드는 하둡에서 제공하는 TestDFSIO 클래스를 사용한 성능평가에서 다음의 표 5 와 같은 성능을 보였다.

표 5. 구축된 클라우드의 성능

	쓰기	읽기
파일 개수	10	10
전체 크기(MB)	100000	100000
스루풋(MB/s)	15.64	21.70
평균속도(MB/s)	16.34	24.57
IO 표준편차	3.48	10.18
총 수행 시간	104.635	84.66

실험에 사용한 온톨로지는 미술작품 온톨로지를 다음과 같이 확장하여 사용하였다. "이름"을 "성씨"와 "이름"으로 세분하고, "작품"도 한 개 이상의 작품을 서술 할 수 있도록 수정하였으며 작품에서도 대표작을 별로도 서술하기 위하여 "대표작품" 관계를 추가하였다. 작가의 개인 정보로서 "생일", "사망일", "태어난 지역", "사망 지역", "거주 지역"을 추가하였다. 또한 작가와 다른 작가 사이의 연관성을 서술하기 위하여 "스승", "제자" 관계를 추가하였고, 작가의 작품에 자주 사용되는 주제를 서술하는 "관심주제" 관계도 추가하여 작가를 서술하는 관계의 수를 총 11 개로 증가하였다. 다음 그림은 실험에 사용한 온톨로지의 개념간의 관계이다.

그림 1. 실험에 사용한 온톨로지의 개념간 관계

실험에 사용할 온톨로지는 실제 데이터가 아닌, 임의로 데이터를 생성하였으며 작가 1 만명, 작품 5 만점과 위치, 등을 포함하여 약 1GB 크기의 온톨로지를 구축하여 사용하였다. 실험은 3 장의 목표 3 에 기술한 세 종류의 질의 유형을 임의로 반복하여 수행하였다.

표 6. 성능 측정 결과

구분	평균 연결 속도 (ms)	평균 처리 속도 (ms)
질의 유형 1	64	78
질의 유형 2	65	77
질의 유형 3	63	150

Jena 혹은 기존의 추론엔진과의 비교실험은 1GB 크기의 온톨로지 파일을 메모리에 적재하는 시간이 많이 걸려, 기술적인 어려움으로 인하여 진행하지 못하였다.

5. 결론

본 논문에서 제안한 대용량 온톨로지 저장구조는 하둡 분산 컴퓨팅 프레임워크에서 사용 가능한 저장구조이며, 대용량 온톨로지의 단순한 저장만이 아니라 대용량 온톨로지를 사용하는 추론엔진에서 고려해야 할 사항을 목표로 선정하여 저장 구조를 설계하였다. 본 저장구조는 기존의 추론엔진과 달리 저장하는 목적만이 아니라 추론에 바로 적용할 수 있다는 것이 특징이다. 향후 연구로는 제안한 저장구조의 성능을 좀 더 정확히 측정하기 위하여 LUBM[13]과 같은 공개되어 있는 성능 측정 도구를 사용하여 다른 방법과의 구체적인 비교가 필요하며,

또한 제안한 저장구조와 MapReduce[14] 분산 프로그래밍 모델을 사용하여 대용량 온톨로지를 추론할 수 있는 방법에 대한 연구가 진행되어야 할 것이다.

6. 참고문헌

[1] 조성정, 김진형, "시맨틱 외부 응용사례 연구", 특집: 시맨틱 웹, 한국 정보과학회지 제21권 제3호 pp11-17, 2003

[2] Extensible Markup Language (XML), http://www.w3.org/XML/

[3] Resource Description Framework (RDF), http://www.w3.org/RDF/

[4] OWL Web Ontology Language Guide (OWL), http://www.w3.org/TR/owl-guide/

[5] 시맨틱 웹을 이용한 지식기반 분산 시각미디어 검색 프레임워크, 한국과학재단 특정기초연구, R01-2003-000-10133-0, 2006.

[6] Newman A., Hunter J., Li Y.F., Bouton C. and Davis M., "A Scale-out RDF Molecule Store for Distributed Processing of Biomedical Data", Semantic Web for Health Care and Life Sciences. In: Workshop WWW 2008, Beijing, China, 2008

[7] Jena-A Semantic Web Framework for Java, http://jena.sourceforge.net/

[8] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

[9] Mohammad F. H., Pankil D,m Latifur K. and Bhavani T., "Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce", CloudCom 2009, LNCS 5931, pp.680-686, 2009

[10] Hadoop, http://hadoop.apache.org/

[11] Hbase, http://hadoop.apache.org/hbase

[11] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung , “The Google File System”, Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003, pp. 20-43

[12] Fay Chang, et al, “Bigtable: A Distributed Storage System for Structured Data”, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2006, pp. 205-218.

[13] Y. Guo, Z. Pan. and J. Heflin., “LUBM: A Benchmark for OWL Knowledge Base Systems.”, Journal of Web Semantics, 2005, pp158-182.

[14] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, 2004. 12.