Journal of The Korea Society of Computer and Information
Vol. 20 No. 8, pp. 77-83, August 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.8.077

Efficient Multimedia Data File Management and Retrieval Strategy on Big Data Processing System

Jae-Kyung Lee *, Su-Mi Shin **, Kyung-Chang Kim ***

Abstract

The storage and retrieval of multimedia data is becoming increasingly important in many application areas including record management, video(CCTV) management and Internet of Things (IoT). In these applications, the files containing multimedia that need to be stored and managed is tremendous and constantly scaling. In this paper, we propose a technique to retrieve a very large number of files, in multimedia format, using the Hadoop Framework. Our strategy is based on the management of metadata that describes the characteristic of files that are stored in Hadoop Distributed File System (HDFS). The metadata schema is represented in Hbase and looked up using SQL On Hadoop (Hive, Tajo). Both the Hbase, Hive and Tajo are part of the Hadoop Ecosystem. Preliminary experiment on multimedia data files stored in HDFS shows the viability of the proposed strategy.

▸ Keyword : Hadoop Framework, Metadata, Multimedia Data, CCTV Data, Record Management, Retrieval, Hadoop Ecosystem

I. Introduction

최근 사물인터넷에 대한 산학계의 관심이 집중되고 있다. 사물인터넷은 다양한 사물들에 센싱 기술과 통신 기술을 접목하여 이용자들 간의 실제 생활과 밀접하게 관련된 정보를 수집하고, 수집한 정보를 사물 간 직접적인 정보 교환과 공유를 통해 기존의 독자적인 서비스들과 결합하여 새로운 형태의 서비스로 재창조 하는 것이다. 이때, 사물에서 나오는 데이터를 수집하여 새로운 가치 창출을 위한 분석 연구가 많이 일어나고 있으며 여기서 생성되는 데이터는 사물인터넷 시장이 커질수록 증가하고 있다. 이와 비슷하게 방송, CCTV 등의 영상 데이터를 관리하기 위한 영상 데이터 관리 시스템 및 문서, 사진 등의 기록 데이터를 관리하기 위한 기록 관리 시스템과 같은 어플리케이션들이 만들어 내는 데이터도 점점 증가하고 있다. 이렇게 증가하는 데이터는 하나의 컴퓨터에서 처리할 수 없을 정도로 커졌으며, 이를 빅데이터라 부른다. 대부분의 빅데이터 연구는 데이터를 분석하고 활용하는 연구에 비중이 있으며 많은 양의 데이터를 저장하고 관리하는 기술은 상대적으로 부족한 상황이다. 앞으로는 다양한 형식의 파일들을 효율적으로 활용하기 위해서는 우선 파일들을 기록하고 관리하는 연구가 필요하다.

현 빅데이터 처리 및 저장 분야에서 가장 많이 사용되는 프레임워크는 Apache Hadoop[1]이다. 하둡은 많은 양의 데이터를 여러 컴퓨터를 이용하여 분산 처리한다. 하둡의 구성요소는 크게 데이터를 분산하여 저장 및 관리하는 하둡 분산 파일시스템(Hadoop Distributed File System (HDFS))[2]와 데이터를 처리하는 MapReduce[3]로 구성되어 있다. 또, 하둡 분산 파일시스템은 데이터를 관리하는 Namenode와 데이터를 실제로 저장하는 Datanode로 구성된다. Namenode에서는 하둡 분산 파일시스템에 분산 저장된 파일을 유지하기 위한 기본 정보들을 메타데이터로 관리한다. 하지만 사용되는 메타데이터는 파일의 구체적인 정보를 가지지 않아 수많은 파일에서 사용자가 원하는 파일을 검색할 때 이용할 수 없다.

• First Author: Jae-Kyung Lee, Corresponding Author: Kyung-Chang Kim
*Jae-Kyung Lee(blue44rain@mail.hongik.ac.kr), Dept. of Computer Engineering, Hongik University
**Su-Mi Shin(sumi@kisti.re.kr), Dept. of Computer Engineering, Hongik University
***Kyung-Chang Kim(kckim@hongik.ac.kr), Dept. of Computer Engineering, Hongik University
• Received: 2015. 07. 14, Revised: 2015. 07. 26, Accepted: 2015. 08. 18.
• This work (Grants No. C0236971) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded by Korea Small and Medium Business Administration in 2015.

하둡 분산 파일시스템 내에 파일 수가 적은 경우 개발자가 데이터를 디렉터리에 잘 나눠 저장하면 문제가 되지 않지만, 급격하게 증가하는 수천수만 개의 데이터와 다양한 어플리케이션에서 생성하는 멀티미디어 데이터를 저장하고 관리하기 위해서는 다양한 조건 검색이 가능한 사용자 친화적인 메타데이터가 필요하다.

본 논문에서는 이러한 멀티미디어 파일에 대한 메타 정보를 RDF 포맷을 이용한 메타데이터 모델을 제안하며, 이를 Tajo, Hive등으로 구성하여 RDF 포맷의 메타데이터로 저장하였을 때 파일의 다양한 특성을 충분히 저장 가능하며 유연한 기법이라는 것을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대한 동향을 설명하고, 3장에서는 멀티미디어 파일에 대한 메타데이터 스키마를 제안하고, HBase를 이용하여 데이터 저장과 SQL On Hadoop을 이용한 데이터 검색 기법을 제안한다. 그리고 설계한 내용을 구현하여 실험한다. 4장에서는 본 연구의 향후 연구 방향과 결론을 맺는다.

II. Preliminaries

1. Related works

앞으로 많은 양의 데이터를 저장하고 이를 통해 새로운 가치를 창출하기 위해 빅데이터 프레임워크인 하둡에 데이터를 저장하고 처리한다. 하둡의 주 연구 분야는 데이터 수집, 데이터 분석, 처리 기법에 대한 연구이며 하둡에 있는 파일을 관리하는 연구는 부족하다. 다음은 하둡에서의 파일 관리에 대한 연구 및 시스템이다.

하둡의 자체 파일 시스템 관련 명령어는 파일의 저장 및 권한 설정 등의 작업은 가능하지만 원하는 파일을 찾는 등은 현 하둡에서는 경로를 사용자가 알아야만 찾을 수 있다. 물론, Web에서 Namenode의 관리페이지를 들어가서 파일 시스템 브라우저를 이용가능 하지만 원하는 파일을 검색하는 작업은 하지 못한다. 이러한 문제점으로 하둡은 다른 에코시스템을 이용하여 사용하는데, 대표적인 예로 Lucene이 있다. 이는 인덱스와 검색 기능을 지원하는 라이브러리를 이용한다.

이와 같이 데이터 검색 기술에서 하둡은 기본적인 검색 기능을 제공하지 않아, Lucene[4]과 같은 인덱스와 검색 기능을 지원하는 라이브러리를 활용한다. Cloudera[5]는 CDH(Cloudera Hadoop)인 배포판 하둡에서 지원하나나 표준화 된 인덱스 구조를 제공하지 않아 사용자 자체적으로 인덱스를 개발해야하는 어려움이 있다.

이외에도 하둡의 메타데이터를 개선하는 연구[6][7]도 있다. 하둡 분산 파일시스템의 Namenode가 가지는 메타데이터에 관계형 모델을 적용하여 기존 메타데이터의 한계를 극복[6]하며 또 다른 메타데이터 연구는 빅데이터에서 메타데이터를 관리하기 위한 'Big Metadata'를 제안[7]한다.

본 논문의 이전 연구에서 하둡에 저장된 파일의 메타데이터를 RDBMS인 MySQL[8]저장하고 이를 다양한 색인을 통해 조건에 맞는 파일을 검색하는 연구를 통해 하둡에서의 파일 관리의 가능성을 보였다. 본 논문에서는 MySQL이 아닌 하둡의 에코시스템들로 대체하여 확장성을 높이고 파일의 다양한 특징을 메타데이터화 하고자 새로운 스키마 및 시스템을 제안한다.

III. The Proposed Scheme

1. Metadata Model

과거 기록 관리 시스템에서는 표준화된 데이터 형식을 따라 관리되었다. 하지만 사물인터넷 시대는 다양한 기기에서 나오는 서로 다른 데이터 형식은 물론 형식의 종류마저도 다양하다. 이러한 데이터들은 엄청난 양의 빅데이터가 되고 이러한 빅데이터를 처리하기 위해서 분석가는 모든 데이터 파일에 대해 각각의 데이터 특징을 기억하고 활용해야한다. 이는 데이터 분석 중 가장 큰 부분을 담당한다. 따라서 수많은 데이터들을 제대로 활용하기 위해 각각의 파일의 특징을 메타데이터화하여 저장한다면 사용자는 다양한 질의를 통해 필요한 파일을 검색하여 분석에 활용 가능할 것이다.

본 논문에서는 다양한 멀티미디어 파일들의 파일 특성을 메타데이터화하기 위해, E/R 다이어그램으로 나타내고, 파일 마다 다양한 특징을 나타내며 표준 형식인 시맨틱 웹 데이터 모델의 Resource Description Framework(RDF)[9] 형식으로 자세한 파일 특징을 보인다. RDF 모델은 평범하고 어떠한 데이터라도 표현 가능하기 때문에 다양한 멀티미디어 파일의 특징을 유연하게 표현 가능하다. RDF에 대한 자세한 내용은 1.2에서 다룬다.

1.1 E/R model for metadata

하둡 분산 파일시스템에는 비디오, 오디오, 이미지, 텍스트 등과 같은 다양한 멀티미디어 파일들이 포함될 수 있다. 이러한 파일들을 다양한 검색 방식을 이용하여 파일을 검색하기 위해서는 각 파일의 세부 정보를 구성해야한다. E/R 다이어그램을 이용하여 다양한 멀티미디어 파일의 특징을 볼 수 있다.

Fig 1의 예를 보면 비디오 파일의 경우 대표적인 특징은 파일의 이름, 타입, 코덱, 해상도, 재생시간, 오디오의 비트레이트, 채널, 등의 다양한 세부 정보가 있으며 텍스트, 오디오, 이미지, 기타 파일들도 이와 같은 다양한 정보들을 가질 수 있다.

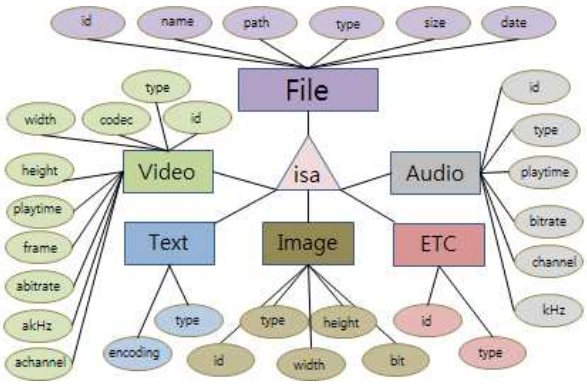

Fig. 1. Example of Characters of Multimedia File

1.2 RDF format for metadata

RDF 포맷은 Resource Description Framework[9]의 약자로 웹상에서 메타데이터를 지원하는데 있어 필요한 구조를 정의하기 위해 W3C에서 제안한 표준 기술 언어이다. 구조는 Subject, Property, Object로 구성되어 있다. Subject는 자원으로써 고유한 Property를 가진다. 하나의 Subject는 여러 Property를 가지며 해당 Property의 값은 Object가 된다. 예를 들면 Subject가 '어린왕자', Property는 '저자', Object는 '생텍쥐페리'와 같은 관계를 가진다. 이는 과거 시맨틱 웹 분야뿐 아니라 현재 사물인터넷 환경에서의 가장 문제점인 다양한 이기종 기기들 간의 데이터 공유를 위해 시맨틱 웹 기술을 활용한 표준 데이터 형식에 대한 연구가 진행되고 있다.

본 논문에서는 기존에 제시한 E/R Model을 Subject, Property, Object로 이루어진 트리플 형태의 RDF 형식으로 표현하고자 한다. 하둡 분산 파일시스템에서 파일에 대한 메타데이터를 RDF 모델로 저장하기 위해 우선, RDF 구조에 대한 재정의가 필요하다. Subject는 파일의 전체 경로, Property는 파일의 다양한 속성, Object는 속성에 대한 값으로 이루어졌으며, 아래 Table 1은 RDF 포맷에서 메타데이터의 예이다.

Table 1. Example of RDF Triple Model

S	P	O
/test/video1	type	video
/test/video1	extension	avi
/test/video1	playtime	30
/test/video1	size	25
/test/video2	type	video
/test/video2	extension	wmv
/test/video2	width	640
/test/video2	height	360
/test/text1	type	text
/test/text1	encoding	UTF8
/test/image1	type	image

video1 파일의 경로는 /test/video1이며 type은 video, 확장자는 avi, playtime은 30초, 파일 크기는 25mb 등의 내용이 저장되어 있다. video2는 text1은 video1과 다른 속성인 인코딩을 저장할 수 있다. 이와 같이 RDF형식을 이용하면 파일의 다양한 속성을 유연하게 저장 및 관리가 가능하다.

1.3 Metadata schema in HBase

HBase[10]는 Rowkey와 Column Family들로 구성되어 있다. 또, Column Family는 여러 Column Qualifier를 포함한다. HBase에 멀티미디어 파일에 대한 메타데이터를 저장하기 위해 Fig 2와 같이 HBase의 Rowkey와 Column Family, Column Qualifier 형식으로 나타낸다. 본 논문에서는 RDF의 Subject인 실제 파일 경로를 Rowkey 값으로 하고 각 Column Family는 멀티미디어 파일의 1차 분류인 데이터 타입을 구별하며 각 Column Family안의 Column Qualifier는 RDF의 Property와 Object를 저장한다.

Fig 2는 메타데이터에 대한 HBase 스키마를 표현한 것이다. 예를 들면 Rowkey는 '/hdfs/test/test.avi'이면 이때 Column Family는 Video이며 Column Qualifier는 'Video:Extension' 그리고 값은 'avi'가 된다. 이렇게 저장된 내용은 'test.avi'라는 파일의 비디오 확장자는 avi라는 내용을 저장한 것이다.

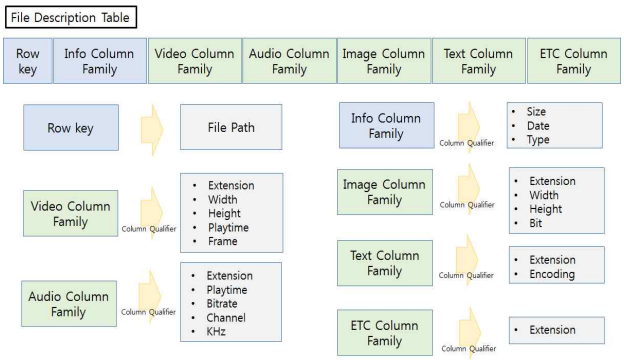

Fig. 2. Example of Schema of Multimedia File on HBase

2 Retrieval Algorithm

앞의 RDF와 HBase 스키마를 이용해 어떻게 파일의 메타데이터를 저장하고 저장한 메타데이터를 이용하여 검색하는 내용을 다룬다.

본 논문의 알고리즘은 저장과 검색으로 나뉜다. 우선 저장은 멀티미디어 파일에 대한 다양한 정보를 RDF 형식으로 정형화하여 HBase에 저장하는 것이다. 그리고 검색은 HBase에 저장된 정보를 SQL On Hadoop 에코시스템을 이용하여 표준SQL을 통해 검색하는 부분이다.

2.1 Store

저장 부분의 알고리즘은 Fig 3과 같다. 저장은 파일에 대한 메타데이터를 하둡의 에코시스템 중 NoSQL DB인 HBase에 저장하는 것이다.

세부 알고리즘은 다음과 같다. 우선 사용자가 파일을 저장하고자 할 때, 해당 파일에 대한 정보를 추출 한다. 이렇게 추출한 데이터는 파일마다 모두 파일에 대한 속성이 다르기 때문에 이를 RDF 형식으로 1차 정형화 시킨다. 이는 HBase에 효율적으

로 저장하기 위해 변환하는 것이다[11]. 이때 변환하는 RDF 형식은 앞에서 언급했듯이 <Subject, Property, Object>로 구성되어 있으며 Subject는 저장할 파일의 경로, Property는 속성, Object는 해당 속성에 대한 값으로 이루어진다. 이렇게 정형화된 데이터를 HBase에 저장하고 동시에 하둡 분산 파일시스템에 해당 파일을 저장한다.

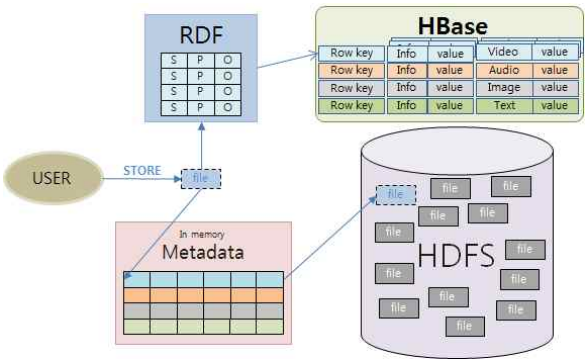

Fig. 3. Store Algorithm

2.2 SQL On Hadoop

SQL On Hadoop은 Interactive Analysis라고도 불린다. 이는 하둡 분산 파일시스템이나 HBase에 저장된 데이터를 표준 SQL언어를 이용하여 다룰 수 있게 해준다. 이러한 기술은 배치 시스템의 단점인 실시간 데이터 처리에 대한 약점을 극복하고, 사용자로 하여금 의미 있는 데이터를 뽑아낸다.

가장 대표적인 SQL On Hadoop의 에코시스템은 HIVE[12]이다. HIVE는 많은 양의 데이터를 RDB 스키마로 정형화하여 표준 SQL을 통해 정형화된 데이터에 대한 질의를 가능하게 한다. HIVE의 경우 질의를 MapReduce 알고리즘으로 바꿔 수행하기 때문에 많은 양의 데이터를 처리하는데 좋다. 하지만 Mapper와 Reducer의 준비 시간 등으로 인하여 질의를 처리하는데 많은 시간이 걸린다. 이러한 단점을 해결하기 위해 다양한 SQL On Hadoop 에코시스템이 개발되고 있다.

HIVE후 새롭게 나온 SQL On Hadoop 기술은 구글의 드레멜, 클라우데라의 임팔라, 호튼웍스의 스팅거, 맵알의 드릴, 그루터의 타조[13] 등이 있다. 이들은 HIVE 보다 빠른 처리 속도의 장점이 있다. 하지만, 많은 양의 데이터를 처리하는데 있어서 안정성 부분은 Hive가 더 뛰어나다.

각각의 SQL On Hadoop 에코시스템들의 장단점이 있기 때문에 어느 하나의 에코시스템을 고르는 것 보다, 상황에 맞게 선택해야 한다.

2.3 Retrieval

HBase에 저장된 파일 정보를 가진 메타데이터를 SQL On Hadoop을 이용하여 다양한 검색 질의를 통해 파일 검색이 가능하다.

다음 Fig 4는 파일의 검색 과정을 보인다. 사용자는 HBase의 내용을 효율적으로 질의하기 위해서 SQL On Hadoop인 Hive, Tajo 등을 통해 질의를 한다. Hive를 통한 검색 과정은 다음과 같다. 사용자가 원하는 검색에 따라 쿼리를 보내면 Hive를 통해 HBase에서 검색에 만족하는 파일에 대한 정보들을 가지고와 사용자에게 뿌려주면 사용자는 그 중 원하는 파일을 선택하여 Namenode의 메타데이터를 통해 파일을 Open한다. 이는 Hive를 통해 가지고 오는 HBase의 Rowkey가 하둡 분산 파일시스템의 경로를 저장하고 있기 때문에 가능하다. 즉, Rowkey의 값을 통해 하둡 분산 파일시스템로부터 파일을 읽어온다. 이러한 전략을 이용하면 다양한 검색을 이용하여 하둡 분산 파일시스템 내의 다양한 파일에 대해 효율적인 검색이 가능하다.

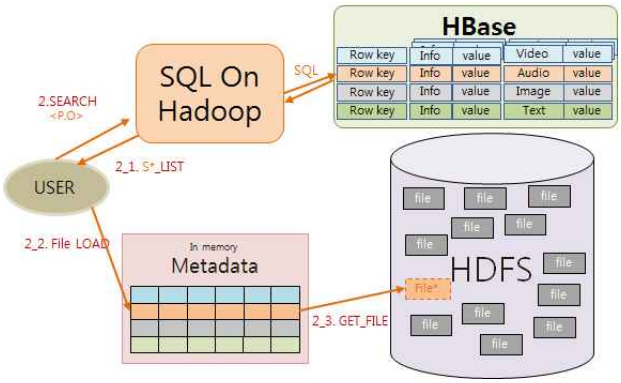

Fig. 4. Retrieval Algorithm

위의 두 가지 알고리즘을 통해 다양한 멀티미디어 파일에 대한 메타데이터 관리와 메타데이터를 이용한 검색이 가능하다. 이를 IV. Experimental에서 두 가지 알고리즘을 구현하여 실험한다.

IV. Experiments

본 논문에서 제안한 알고리즘을 실제 하둡 클러스터에 구축하여 실험한다.

1 실험 환경 구축

본 실험에서 사용할 하둡 클러스터는 Namenode 1대와 Datanode 3대 총 4대로 구성하였으며, 호턴웍스의 하둡 배포판인 HDP-2.2 버전을 이용한다.

하둡 및 하둡 에코시스템의 버전은 다음 Table 2와 같다. Hadoop은 2014년 11월에 나온 2.6.0 버전을 이용하며, HBase는 0.98,4 버전을 사용한다. SQL On Hadoop 에코시스템인 Hive는 0.14.0 버전. Tajo는 0.10.0 버전으로 구축한다.

Table 2. Hadoop, Hadoop Ecosystem's versions

	Hadoop	2.6.0
NoSQL	HBase	0.98.4
SQL On Hadoop	Hive	0.14.0
	Tajo	0.10.0

2 데이터 저장 실험

다양한 멀티미디어 파일에 대한 데이터 저장 실험을 위해 비디오 파일, 문서 파일, 이미지 파일, 오디오 파일, 기타 파일들을 각각 100개씩 랜덤한 특성을 가지도록 생성하였다.

생성한 파일들에 대한 속성을 3개의 컬럼 형태인 RDF 형식으로 정형화한다. 각각 3개의 컬럼<Subject, Property, Object>에서 Subject는 파일의 전체 경로, Property는 속성, Object는 해당 속성에 대한 값으로 저장한다. 즉, 1개의 파일에 대해 파일 타입, 확장자 등의 속성이 존재하면 각각의 속성별로 하나의 record가 생성된다.

위에서 생성한 RDF 데이터를 데이터베이스에 넣기 위해 NoSQL인 HBase를 이용한다. HBase를 본 논문에서 제시한 스키마로 구축하여 테이블 이름을 'file'이라 하였으며 생성한 RDF 데이터를 저장하였다. HBase의 count를 통해 아래 Table 3과 같이 500개의 데이터가 모두 들어간 것을 확인했다.

Table 3. Result of Store Algorithm Experiment

Hbase의 count 명령어(Record수 확인)
hbase(main):3708:0> count 'file' 500 row(s) in 0.6820 seconds => 500

3 데이터 검색 실험

사용자가 원하는 조건에 따른 다양한 검색을 위해 표준SQL을 이용하여 검색이 가능해야 한다. 따라서 앞에 구성한 HBase에 저장된 데이터를 SQL On Hadoop의 Hive나 Tajo 등을 이용해 표준SQL 질의를 가능하게 한다.

본 논문의 실험에서는 SQL On Hadoop의 Hive와 Tajo에 각각 'file'이라는 외부 테이블을 생성하여 HBase 테이블을 참조한다. 이는 SQL On Hadoop 에코시스템들이 가지는 기술로 HBase의 실시간으로 들어오는 데이터에 대해 매번 재정의 할 필요 없이 사용가능하다. 이렇게 Hive와 Tajo에 구성한 'file' 테이블을 HBase의 테이블과 비교하기 위해 레코드 수를 비교해본 결과, Table 4와 5를 통해 Hive와 Tajo에서 부른 레코드의 수가 HBase와 같은 상태인 레코드 수 500임을 알 수 있다.

Table 4. Load Data On Hbase Using Hive

Hive	Time taken: 0.292 seconds, Fetched: 500 row(s) hive> select * from file;

Table 5. Load Data On Hbase Using Tajo

Tajo	default> select count(*) from file; Progress: 0%, response time: 1.357 sec Progress: 0%, response time: 1.359 sec Progress: 0%, response time: 1.76 sec Progress: 0%, response time: 2.562 sec Progress: 0%, response time: 3.563 sec Progress: 0%, response time: 4.565 sec Progress: 50%, response time: 5.567 sec Progress: 100%, response time: 6.025 sec ?count ------------------------------ 500 (1 rows, 6.025 sec, 4 B selected)

Hive와 Tajo에 각각 구성된 'file' 테이블을 이용해 간단한 검색질의를 만들어 테스트를 한다. 검색 질의는 간단하게는 정의한 속성들마다 범위질의, 선택 질의, 정렬 등이 가능하다. Hive는 단순 선택 질의 또는 대용량 데이터 처리에 적합하기 때문에 본 논문에서는 검색 질의로 선택 질의에 대한 실험을 진행하며 Tajo를 이용하여 선택 질의, 범위 질의, 정렬 등의 실험을 진행한다.

HBase에 저장한 데이터에 대해 선택 질의는 video 파일 중 확장자가 'avi'이면서 2014년에 저장된 파일에 대해 Hive와 Tajo를 통해 검색하였다. 다음 Table 6은 각각의 Query와 그 결과이다.

Table 6. Example of Select Query

선택 질의	select rowkey from file where vextension='avi' and date like '%2014';
Hive 결과	/hdfs/test/test26.avi /hdfs/test/test87.avi Time taken: 0.177 seconds, Fetched: 2 row(s)
Tajo 결과	/hdfs/test/test26.avi /hdfs/test/test87.avi (2 rows, 3.002 sec, 44 B selected)

Tajo를 이용하여 범위 질의와 데이터 정렬을 실험한다. 범위 질의는 모든 오디오 파일에 대해 용량이 6mb에서 8mb 사이의 파일에 대해 검색하고 정렬은 범위 질의의 결과를 재생 시간에 따라 정렬한다. 다음 Table 7은 범위질의에 대한 Query와 그 결과이고 Table 8은 정렬에 대한 Query와 그 결과이다.

Table 7. Example of Range Query

범위질의	select rowkey from file where size>=6 and size<=8 and aextension!='NULL';
범위질의 결과	/hdfs/test/test11.mid /hdfs/test/test17.mid /hdfs/test/test27.mid /hdfs/test/test42.mid /hdfs/test/test61.wav /hdfs/test/test78.mp3 /hdfs/test/test80.wma /hdfs/test/test82.mp3 /hdfs/test/test95.wma (9 rows, 0.763 sec, 198 B selected)

Table 8. Example of Sort Query

정렬	select rowkey, aplaytime from file where size>=6 and size<=8 and aextension!='NULL' order by aplaytime desc;
정렬 결과	/hdfs/test/test27.mid, 650 /hdfs/test/test11.mid, 597 /hdfs/test/test17.mid, 575 /hdfs/test/test95.wma, 565 /hdfs/test/test42.mid, 554 /hdfs/test/test78.mp3, 466 /hdfs/test/test82.mp3, 420 /hdfs/test/test80.wma, 415 /hdfs/test/test61.wav, 310 (9 rows, 1.767 sec, 234 B selected)

4 실험 결과

본 논문에서는 빅데이터 환경에서의 다양한 멀티미디어 파일을 효율적으로 관리 및 활용하기 위해 파일의 특성을 메타데이터로 저장하여 활용하는 실험을 보였다. 위의 실험으로 파일의 여러 속성에 대해 메타데이터를 저장하고 다양한 검색 질의를 통해 원하는 자료를 찾는 것을 보였다. 실험 결과를 이용하여 Hive와 Tajo와 같은 SQL On Hadoop은 모두 JDBC를 제공하기 때문에 다양한 어플리케이션에서 연동하여 원하는 파일을 검색하고 검색 결과인 파일의 세부 경로를 통해 파일을 쉽게 가지고 오거나 R과 같은 분석 프로그램에서도 rhive 라이브러리를 이용하여 원하는 질의를 통해 하둡 분산 파일 시스템 내 파일을 바로 사용 가능하다. 이러한 분석 환경에서의 다양한 활용이 가능하다.

V. Conclusions

본 논문은 앞으로 빅데이터 시장의 범주가 점차 커져감에 따라 저장하고 관리하는 데이터도 다양해 질 것이며, 그러한 파일들을 쉽게 검색하고 관리하기 위해서는 파일의 속성들을 메타데이터로 잘 정리하여 관리해야한다는 것에서 시작되었다. 그러한 점을 중심으로 시맨틱 웹의 데이터 저장 형식의 RDF 데이터 저장 기술 및 NoSQL DB인 HBase를 이용한 데이터 저장 그리고 Hive와 Tajo 같은 SQL On Hadoop 에코시스템을 이용한 검색 기능에 대해 서술하고 실험하였다.

현재 하둡을 이용한 빅데이터 플랫폼은 하둡의 다양한 에코시스템들도 같이 운행되며 이러한 환경에서 생성되는 파일들이 생성될 수 있으며, 에코시스템에서 생성되는 파일들 역시 관리가 필요하다. 본 논문의 확장으로 하둡 분산 파일시스템에 생성되는 모든 파일들을 자동으로 감지하고 관리하며 이에 맞춰 제안한 메타데이터를 더 세분화할 필요가 있다.

REFERENCE

[1] Hadoop. *https://hadoop.apache.org/*

[2] Shvachko, Konstantin, et al. "*The hadoop distributed file system.*" Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010.

[3] J. Dean, S. Ghemawat, "*MapReduce: Simplified Data Processing on Large Clusters,*" In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, Dec. 2004.

[4] Lucene. https://lucene.apache.org/

[5] Cloudera. https://www.cloudera.com/

[6] Hakimzadeh, Kamal, Hooman Peiro Sajjad, and Jim Dowling. "*Scaling HDFS with a Strongly Consistent Relational Model for Metadata.*" Distributed Applications and Interoperable Systems. Springer Berlin Heidelberg, 2014.

[7] Smith, Ken, et al. "*Big Metadata: The Need for Principled Metadata Management in Big Data Ecosystems.*" Proceedings of Workshop on Data analytics in the Cloud. ACM, 2014.

[8] MySQL. https://www.mysql.com/

[9] Schreiber, G., Raimond, Y. "*RDF 1.1 primer.*" W3C Note, 2014.

[10] HBase. http://hbase.apache.org/

[11] Craig Franke, Samuel Morin, Artem Chebotko, John Abraham, and Pearl Brazier. "*Efficient Processing of Semantic Web Queries in HBase and MySQL Cluster.*" IEEE Computer Society, Vol. 15, No. 03, pp. 36-43, May-June. 2013.

[12] Hive. https://hive.apache.org/

[13] Tajo. http://tajo.apache.org/

[14] JunSang Kim, ChangHyeon Kim, WonJoo Lee, ChangHo Jeon "*A Block Relocation Algorithm for Reducing Network Consumption in Hadoop Cluster*", The Korea Society of Computer and Information, Vol. 19, No. 11, pp. 9-15, Nov. 2014.

[15] TaeHoon Keum, WonJoo Lee, ChangHo Jeon, "*A Performance Analysis Based on Hadoop Application's Characteristics in Cloud Computing*", The Korea Society of Computer and Information, Vol. 15, No. 5, pp.4 9-56, May. 2010.

Authors

Jae-Kyung Lee received the B.S. degree in Computer Engineering from Hongik University, Korea, in 2013. He is currently a M.S. student in Dept. of Computer Engineering, Hongik University, Korea.

He is interested in big data analysis, semantic web, Internet of Things and cloud computing.

Su-Mi Shin received the B.S. and M.S. degrees in Computer Engineering from Hongik University, Korea, in 1997 and 2005, respectively. She is currently a Ph.D. student in Dept. of Computer Engineering, Hongik University, Korea.

She is interested in big data processing, web databases, data retrieval and Internet of Things.

Kyung-Chang Kim received the B.S. in Computer Science from Hongik University in 1978, M.S. in Computer Science from KAIST in 1980 and Ph.D in Computer Science from University of Texas at Austin in 1990.

Dr. Kim joined the faculty of the Department of Computer Engineering at Hongik University, Seoul, Korea, in 1991. He is currently a Professor in the Department of Computer Engineering, Hongik University. He is interested in main memory databases, sensor databases, web databases, Internet of Things and big data processing.