RDBMS 기반 하둡 메타데이터 관리의 설계 및 구현

손시운[1], 양석우[1], 길명선[1], 문양세[1], 민차우[2], 원희선[2]
[1]강원대학교 컴퓨터과학과, [2]한국전자통신연구원
e-mail: {ssw5176, seakwoo, gils, ysmoon}@kangwon.ac.kr, {chau, hswon}@etri.re.kr

Design and Implementation of RDBMS-based Management of Hadoop Metadata

Siwoon Son[1], Seokwoo Yang[1], Myeong-Seon Gil[1], Yang-Sae Moon[1],
Minh Chau Nguyen[2], and Hee-Sun Won[2]
[1]Dept. of Computer Science, Kangwon National University, [2]ETRI

요 약

최근 빅데이터 문제를 해결하기 위해 하둡의 사용이 급증하였다. 하둡은 다수의 노드에 데이터를 분산 저장 및 처리하며, 이를 위해 모든 메타데이터를 네임노드에서 관리한다. 기존 하둡은 모든 메타데이터를 메모리 상에서 관리하며, 변경 이력을 로컬 파일 시스템에서 별도의 파일로 관리한다. 이 방법에서는 데이터의 증가 및 하둡 에코시스템의 확장 등의 이유로 관리되어야 할 메타데이터가 크게 증가하며, 이는 곧 네임노드의 메모리 부하를 높이는 문제가 있다. 본 논문은 이러한 인메모리 기반의 하둡 메타데이터 관리 구조를 RDBMS 기반으로 수정하도록 설계 및 구현한다. 그리고 하둡의 모든 명령어에 대한 테스트를 작성하여 본 연구의 적정성을 검토하였다. 본 논문은 네임노드의 부하를 줄임으로써 하둡의 안정성을 높이는 좋은 연구 결과라 사료된다.

1. 서론

최근 사물 인터넷(IoT: Internet of Things) 및 소셜 네트워크 서비스(SNS: social network service) 등의 대용량 데이터를 발생시키는 서비스가 증가하였다. 이에 따라 빅데이터[1, 2] 문제가 대두되며, 많은 기업과 연구 기관에서는 빅데이터를 다루기 위해 하둡 에코시스템(Hadoop ecosystem)을 도입하였다. 하둡[3-5]은 대용량의 데이터를 분산 저장 및 관리하는 HDFS(Hadoop distributed file system)[6, 7]와 분산 처리하는 맵리듀스(MapReduce)[8]를 통해 효율적으로 빅데이터를 다룬다.

현재 하둡은 데이터노드에 저장된 파일의 네임스페이스, 접근 제어 정보, 블록의 위치 정보 등의 메타데이터를 네임노드의 인메모리를 통해 관리하고 있다. 하지만 데이터노드의 확장 및 파일이 증가함에 따라 이러한 메타데이터를 관리하는 네임노드의 오버헤드를 높이는 문제가 발생한다. 따라서 본 논문은 메타데이터를 확장성 있고 효율적으로 관리하기 위하여, 기존의 인메모리가 아닌 RDBMS(relational database management system)에서 관리되도록 저장하는 과정을 설계 및 구현하였다. 본 논문을 통해 기존의 네임노드에서 인메모리의 오버헤드를 줄이며, 하둡 관리자 측면에서 효율적으로 메타데이터를 관리할 수 있다. 또한 메타데이터를 별도로 관리함으로써 타 서비스와의 연동이 쉬워, 하둡 외의 응용 프로그램에서도 메타데이터의 일관성 있는 관리가 가능하도록 한다.

* 본 연구는 산업통상자원부와 한국산업기술진흥원의 "국제공동기술개발사업"의 지원을 받아 수행된 연구결과임.

2. 관련 연구

빅데이터 문제를 해결하는 가장 대표적인 방법인 하둡은 마스터 역할을 수행하는 네임노드와 슬레이브 역할을 수행하는 다수의 데이터노드를 사용하여 대용량 데이터를 분산 저장 및 처리한다. 하둡은 HDFS와 맵리듀스라는 대용량 파일 저장 및 처리 시스템을 지원한다. 먼저, HDFS는 분산 파일 시스템으로써, 파일을 블록(block)이라는 단위로 나누어 가용할 수 있는 데이터노드에 저장한다. 이때 블록을 복제하여 여러 데이터노드에 함께 저장하여 데이터의 유실을 쉽게 복구할 수 있다. 다음으로, 맵리듀스는 대용량 데이터를 분산 환경에서 처리하기 위한 목적으로 개발된 소프트웨어 프레임워크이다. 클라이언트가 맵리듀스 프로그램을 작성하여 하둡에게 실행을 요청했을 시, 네임노드는 필요한 데이터의 위치를 파악하고 각 데이터노드에게 프로그램만을 전달하여 맵리듀스 프로그램을 수행한다.

3. 인메모리 기반 HDFS 메타데이터 관리

기존의 하둡은 모든 메타데이터를 네임노드의 인메모리를 통해 관리하며, 메타데이터의 영속성을 위해 네임노드의 로컬 파일 시스템을 사용하여 메타데이터를 저장 및 유지한다. 먼저 FsImage 파일은 네임스페이스, 접근 제어 정보, 블록의 위치 정보 등을 관리하며, 두 번째 EditLog 파일은 모든 변경 사항을 기록하는 로그 파일이다. 네임노드는 이러한 메타데이터 관리 파일을 기반으로 메모리 상에 메타데이터를 구축한다.

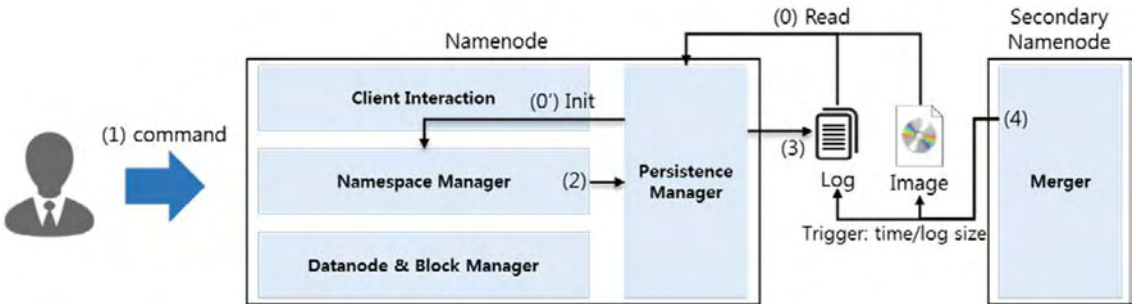

그림 1. 인메모리 기반 메타데이터 처리 과정.

그림 1은 인메모리 기반의 메타데이터를 처리하는 과정을 나타낸 것이다. 먼저 (0) 하둡이 구동될 시에 네임노드는 메타데이터 파일(FsImage, EditLog)로부터 메타데이터 정보를 불러온다. 그리고 (0′) 불러온 메타데이터 정보를 메모리에 적재함으로써 메모리 상에서 메타데이터를 유지한다. 하둡이 정상적으로 동작하는 중에 (1) 클라이언트로부터 하둡 명령어 수행이 요구될 경우, (2) 하둡은 이 명령어에 따른 메타데이터 변경 이력을 인메모리로부터 가져온다. 마지막으로 (3) 메타데이터의 변경 이력을 EditLog 파일에 저장함으로써 하둡의 비정상 종료에도 메타데이터를 유지할 수 있다. 하둡은 네임노드의 SPOF(single point of failure) 문제를 피하기 위해 보조 네임노드(Secondary NameNode)를 함께 동작시킬 수 있다. 보조 네임노드는 네임노드의 하둡의 메타데이터 정보를 네임노드와 함께 유지하는데 (4) 네임노드에서 관리하는 메타데이터 파일에 접근하여 메타데이터 정보를 획득한다.

하지만 이러한 인메모리 기반의 메타데이터 관리 방법은 메타데이터가 급증할 경우 네임노드의 메모리에 큰 오버헤드를 야기시키며, 시스템 확장에도 제한이 생긴다. 따라서 본 논문은 인메모리 기반이 아닌 RDBMS를 사용함으로써 메타데이터를 효율적이고 확장성 있게 관리할 수 있도록 설계 및 구현하였다.

4. RDBMS 기반 HDFS 메타데이터 관리 설계

본 절에서는 메타데이터를 인메모리가 아닌 RDBMS에서 저장 및 관리하는 과정을 설계한다.

4.1. RDBMS 기반 메타데이터 저장 과정

그림 2는 RDBMS 기반의 메타데이터 저장 과정을 나타낸 것으로, 기존의 인메모리 방식으로부터 일부 추가 및 수정한 것을 알 수 있다. 먼저 (0) 하둡이 구동될 시 메타데이터 파일로부터 메타데이터 정보를 읽어 (0′) 메모리에 적재하는 과정은 동일하다. 그 후 (1) 클라이언트가 하둡 명령어를 실행했을 때, (2) 네임노드는 메타데이터의 변경 이력을 인메모리로부터 읽어 (3) 메타데이터 파일에 저장한다. 하지만 단순히 파일에만 저장하지 않고 변경 이력을 보조 네임노드에 함께 전달한다. (4) 메타데이터의 변경 이력을 전달받은 보조 네임노드는 RDBMS와 연동하여 그 내용을 저장한다. 이때 네임노드에서 바로 RDBMS에 변경 이력을 저장하지 않은 이유는 네임노드의 작업량을 줄임으로써 부하를 최소화하기 위함이다.

4.2. RDBMS 기반 메타데이터 테이블 설계

RDBMS를 통해 메타데이터를 관리할 경우, 관리될 메타데이터의 종류와 RDBMS의 스키마 설계가 중요하다. 본 논문에서는 하둡에서 사용되는 각 명령어에 해당하는 클래스를 분석하고, 각 클래스에서 관리하는 변수들을 통해 RDBMS에 저장될 메타데이터를 확인한다. 그림 3은 스키마를 설계하여 ER 다이어그램으로 표현한 것으로, 총 12개의 테이블을 설계하였다. 이들 테이블에 대한 자세한 설명과 관리 절차는 지면 제약상 생략한다.

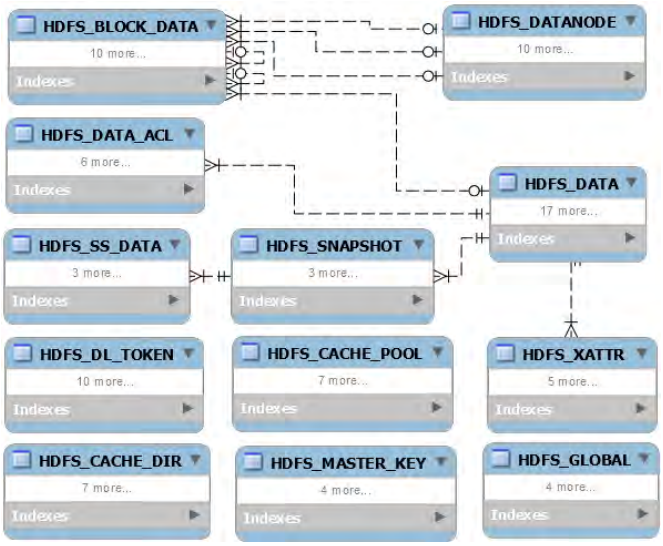

그림 3. 메타데이터 테이블 ER 다이어그램.

5. 구현 및 테스트

본 절에서는 제4절에서 설계한 RDBMS 기반의 메타데이터 관리 기술을 구현 및 테스트한다. 먼저, 본 논문에서 사용한 하둡은 HDFS-6994 버전을 수정하였으며, 메타데이터를 저장할 RDBMS로는 SkySQL MariaDB[9]를 사용하였다. 그림 4는 RDBMS 기반 메타데이터 관리 기술을 구현한 소스 코드 트리이다. 그림에서 보듯이, 소스 코드는 크게 세 개의 추가된 컴포넌트와 다수의 수정된 컴포넌트로 구성된다. 이들 컴포넌트에 대한 자세한 설명은 생략한다.

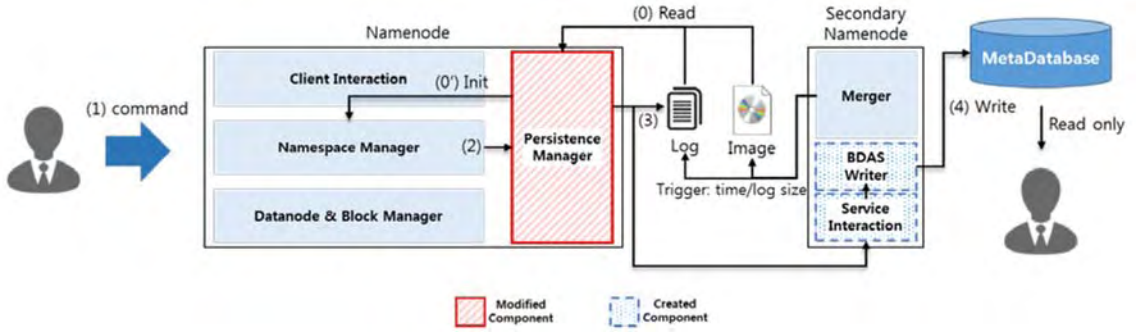

그림 2. RDBMS 기반 메타데이터 처리 과정.

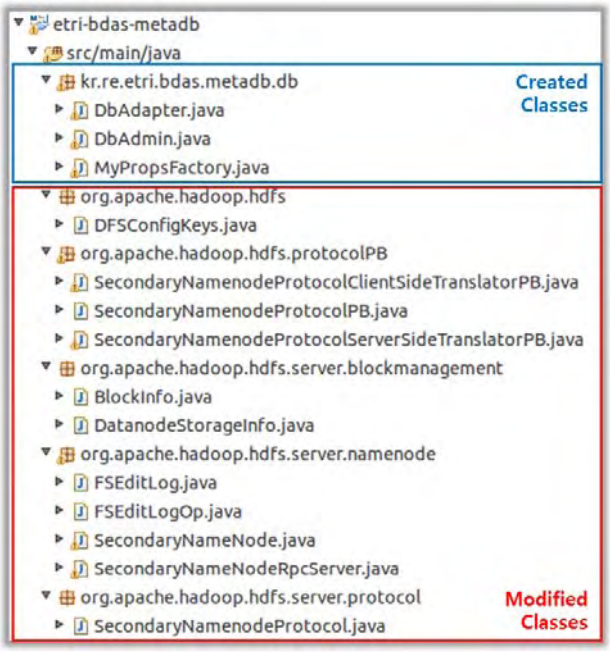

그림 4. 소스 코드 트리.

다음으로, 테스트는 하둡에서 지원하는 모든 명령어들 간에 연관성이 높은 명령어를 5개의 카테고리로 분류하여 구현하였다. 각 카테고리 별 명령어는 파일 관리 19개, Delegation token 관리 3개, Quota 관리 3개, Snapshot 관리 5개, Cache 관리 6개 등 총 36개로 구성되어 있다. 테스트는 그림 5와 같이 카테고리 별로 테스트 클래스를 만들었으며, 각 클래스에는 해당 카테고리에 포함된 모든 명령어에 대한 메타데이터의 변경 이력이 올바로 저장되었음을 확인하였다.

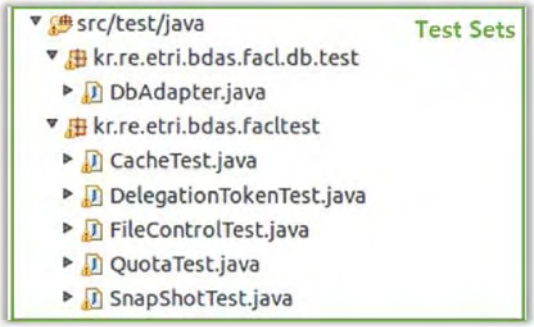

그림 5. 테스트 코드 트리.

6. 결론 및 향후 연구

본 논문은 기존의 하둡 메타데이터 관리 방법이 인메모리에 부하를 높이는 문제를 제기하고, 이러한 문제를 해결하기 위해 RDBMS 기반의 메타데이터 관리 기술을 설계 및 구현하였다. 그리고 구현 결과에 대한 적정성을 검토하기 위해, 하둡에서 지원하는 모든 명령어에 대한 메타데이터 변경 이력이 RDBMS에 올바로 저장됨을 테스트하였다. 그 결과 메타데이터 정보가 RDBMS에서 저장됨을 확인하였다. 이를 통해 하둡 관리자는 메타데이터를 효율적으로 관리하고, 외부의 응용프로그램 또한 메타데이터에 쉽게 접근할 수 있다.

하지만 본 논문의 결과는 메타데이터 정보를 인메모리에서 RDBMS로 저장하는 과정만을 다루었다. 따라서 향후 연구를 통해 인메모리에 저장된 메타데이터를 모두 제거하고 하둡 내에서 메타데이터를 요구할 경우 RDBMS를 사용하도록 함으로써 부담을 최소화할 예정이다.

참고문헌

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers, “Big Data: The Next Frontier for Innovation, Competition, and Productivity,” Technical Report, McKinsey Global Institute, 2011.

[2] M. Saecker and V. Markl, “Big Data Analytics on Modern Hardware Architectures: A Technology Survey,“ *Springer Lecture Notes in Business Information Processing*, Vol. 138, pp. 125-149, 2013.

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

[4] C. Lam and J. warren, “Hadoop in Action,” Manning Publications, 2010.

[5] T. White, “Hadoop: The Definitive Guide,” O'Reilly Media, Yahoo! Press, June 2009.

[6] HDFS, http://hadoop.apache.org/hdfs/.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” In *Proc. of the IEEE 26th Symp. on Mass Storage Systems and Technologies(MSST)*, pp. 1-10, May 2010.

[8] J. Dean and S. Ghemawat, “MapReduce: a Flexible Data Processing Tool,” *Communications of the ACM*, Vol. 54, No. 1, pp. 72-77, Jan. 2010.

[9] MariaDB, https://mariadb.org/.