RDBMS 기반 HDFS ACL 설계*

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

Design of RDBMS-based HDFS ACLs

Siwoon Son[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

본 논문에서는 하둡의 인메모리 기반 ACL(access control list)을 RDBMS 기반으로 관리하도록 기존 하둡을 재설계하였다. 기존 하둡은 ACL을 인메모리에서 관리하기 때문에 대용량 ACL 정보를 관리함에 있어 메모리 오버헤드, ACL 정보 관리의 비효율성 등 몇 가지 문제가 발생할 수 있다. 본 논문에서는 ACL 관리에 RDBMS를 사용함으로써 메모리 크기에 종속되지 않으며, 외부 응용 프로그램에서도 쉽고 일관성있게 ACL 정보를 관리할 수 있다. 이 같은 결과에 따라, 본 논문은 빅데이터를 하둡에서 안정하게 관리할 수 있는 우수한 연구 설계 결과라 생각된다.

1. 서론

최근 빅데이터[1, 2] 문제가 화두에 오르며, 이러한 빅데이터 문제를 해결하기 위해 많은 기업과 연구기관에서는 하둡 에코시스템(Hadoop ecosystem) 사용이 급증하였다. 하둡(Hadoop)[3-5]이란 대용량 데이터를 분산 환경에 저장하는 HDFS(Hadoop distributed file system)[6-8]와 HDFS에 저장된 데이터를 처리하기 위한 맵리듀스(MapReduce)[9, 10]를 지원하는 오픈 소스 프레임워크이다.

하둡은 다양한 빅데이터 문제를 처리하기 위해 새로운 버전을 꾸준히 배포하고 있다. 최근 하둡은 Hadoop-2.4.0버전을 통해 ACL(access control list)[11]을 제공하기 시작하였다. ACL은 각 파일 또는 디렉터리의 접근 인가 정보를 사용자 또는 그룹의 리스트 형태로 표현한 접근 인가 방식을 말한다. 하둡은 이러한 ACL을 네임노드(NameNode)의 인메모리(In-memory)에서 수정 또는 삭제할 수 있도록 관리하고 있다.

그러나 인메모리에서 ACL을 관리하는 것은 다음과 같은 문제점을 갖는다. 먼저, 파일 또는 디렉터리의 개수가 증가함에 따라 ACL 정보가 함께 증가할 수 있으며, 이는 곧 네임노드에서 메모리 자원의 오버헤드 문제를 야기한다. 다음으로, 파일 관리자 또는 기타 응용 프로그램이 ACL 정보에 접근하려면 반드시 HDFS를 통해야만 한다. 이러한 문제점을 해결하기 위해, 본 논문은 ACL 정보를 인메모리가 아닌 RDBMS에서 관리할 수 있도록 설계하여 네임노드의 부담을 줄이고자 한다. 또한, 이를 이용하여 외부 응용 프로그램이 쉽게 ACL 정보에 접근할 수 있으며, 파일 관리자 측면에서 효율적으로 ACL 정보를 관리할 수 있도록 한다.

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [14-000-05-001, 스마트 네트워킹 핵심기술개발].

2. 관련 연구

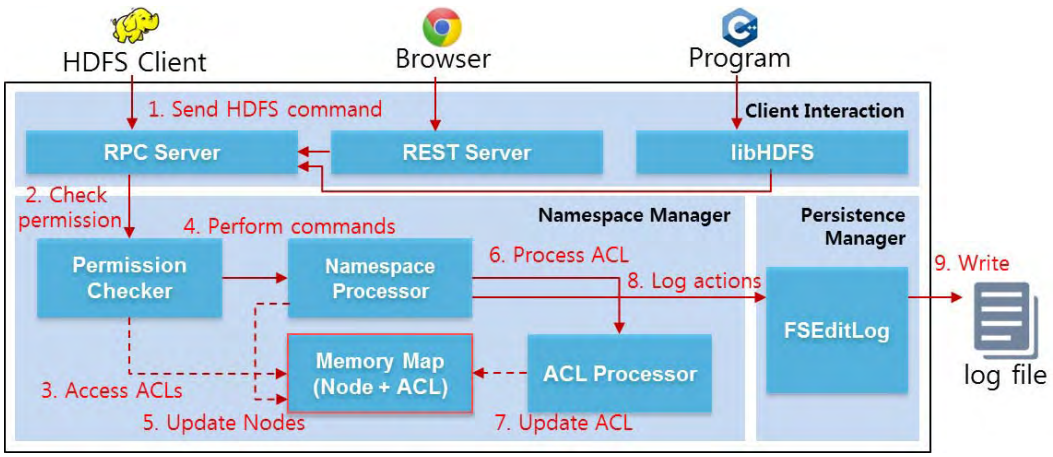
하둡은 빅데이터 문제를 해결하기 위한 방안 중 하나로, 대용량 데이터에 대한 저장 및 처리를 지원하는 오픈 소스 프레임워크이다. 하둡은 HDFS를 통해 대용량 데이터를 다수의 분산된 데이터노드(DataNode)에 저장한다. 이렇게 저장된 파일 및 디렉터리의 파일 시스템 정보는 네임노드에 의해 관리된다. 따라서 사용자는 파일 및 디렉터리의 접근이 필요할 경우, 네임노드를 통해 해당 파일 및 디렉터리의 위치를 확인한 후 데이터노드에 접근한다.

하둡을 이용하는 사용자의 증가에 따라, 하둡 내부파일에 대한 각 사용자의 접근 제어가 필요하다. 하둡은 이러한 접근 제어를 위해 전통적인 POSIX 권한 체계와 유사한 파일 접근 인가 방식을 사용하였다. 즉, 각 파일의 접근 인가를 `owner`, `group`, `others`로 나누어 관리하였다. 하지만 이러한 방식의 접근 인가 체계는 다수의 사용자에게 서로 다른 권한 정보를 부여하기에는 어려움이 있다. 예를 들어, (그림 1)과 같은 접근 권한을 갖는 파일 '`/file01`'의 권한 정보를 유지하며, `READ` 권한만을 갖는 사용자를 표현하기 어렵다.

```
-rwxrw----   3 user01 group01    0 2015-01-01 /file01
```

(그림 1) POSIX 권한 체계의 예시.

하둡은 이러한 문제를 해결하기 위하여, Hadoop-2.4.0(HDFS-4685)[12] 버전을 통해 네임노드의 인메모리에서 ACL 관리 기능을 제공하기 시작하였다. ACL은 파일 또는 디렉터리의 접근 인가 정보를 사용자 또는 그룹의 리스트 형태로 표현함으로써 보다 구체적으로 권한 정보를 표현할 수 있다.



(그림 2) 인메모리 기반의 ACL 작업 흐름도.

3. 인메모리 기반의 접근 제어 리스트

(그림 2)는 기존 인메모리 기반의 ACL을 사용하는 하둡에서, ACL 관련 작업을 수행할 경우의 처리 과정을 추상적으로 표현한 흐름도이다. ACL 관련 작업의 처리 과정은 다음과 같다.

(1) **Send HDFS command**: 외부 클라이언트로부터 하둡에 접근하는 방법은 크게 세 가지가 있으나, 웹에서 접근하는 REST Server와 외부 응용 프로그램에서 접근하는 라이브러리(libHDFS)는 모두 RPC Server를 사용하게 된다.

(2) **Check permission**: RPC Server는 전달받은 명령어를 Permission Checker 컴포넌트를 통해 명령어를 요청한 사용자의 접근 인가를 확인한다. 이를 위해, 현재 사용자의 정보와 요청한 명령어가 요구하는 권한 정보를 전달한다.

(3) **Access ACLs**: Permission Checker는 사용자가 요청한 명령어의 인가를 확인하기 위해 메모리에 저장되어있는 해당 사용자의 권한 정보를 요청한다. 이때, 접근 가능 여부를 판별하여 불가능할 경우에는 작업을 중단한다.

(4) **Perform commands**: 앞서 Permission Checker를 통해 권한이 확인되었다면 Namespace Processor에게 작업을 수행하도록 요청한다.

(5) **Update Nodes**: 전달받은 명령어가 파일 정보를 수정하는 작업일 경우, 이 과정을 통해 메모리에 저장되어있는 파일 정보를 수정할 수 있다.

(6) **Process ACL**: 전달받은 명령어가 ACL 정보를 수정하는 작업일 경우, ACL Processor에게 ACL 정보 수정을 요청한다.

(7) **Update ACL**: ACL Processor는 새로운 ACL 정보를 메모리에 저장한다.

(8) **Log actions**: 모든 HDFS 작업이 끝날 경우, 작업 내용을 로그 파일에 저장한다.

상기 처리 과정을 통해 알 수 있듯이, ACL 정보는 인메모리에서 관리되며 접근 인가를 확인할 경우에도 인메모리에 저장되어 있는 ACL 정보에 접근한다. 이러한 인메모리 기반의 ACL은 접근 인가를 관리함에 있어서 빠르다는 장점을 갖지만 여러 문제가 나타날 수 있다. 첫째, 빅데이터를 다루는 하둡에서 다수의 파일을 저장할 경우에는 대용량의 ACL 정보를 관리해야 할 수 있다. 이는 네임노드에서 메모리 자원의 오버헤드 문제를 발생한다. 둘째, 외부 응용 프로그램이 하둡을 통하지 않으면 ACL 정보를 별도로 관리하기 어렵다. 셋째, 파일 관리자가 ACL 정보를 관리하기 위해서는 반드시 하둡이 실행 중이어야만 한다. 또한, ACL 정보는 사용자 정보와 파일 정보에 종속되지 않는 특성을 갖기 때문에, 실제로 존재하지 않는 사용자 또는 파일에 대한 ACL 정보도 관리할 수 있어야 한다. 본 논문은 이러한 인메모리 기반 ACL 관리의 문제들을 해결하기 위해 RDBMS 기반의 ACL 관리 기능을 설계한다.

4. RDBMS 기반의 접근 제어 리스트 설계

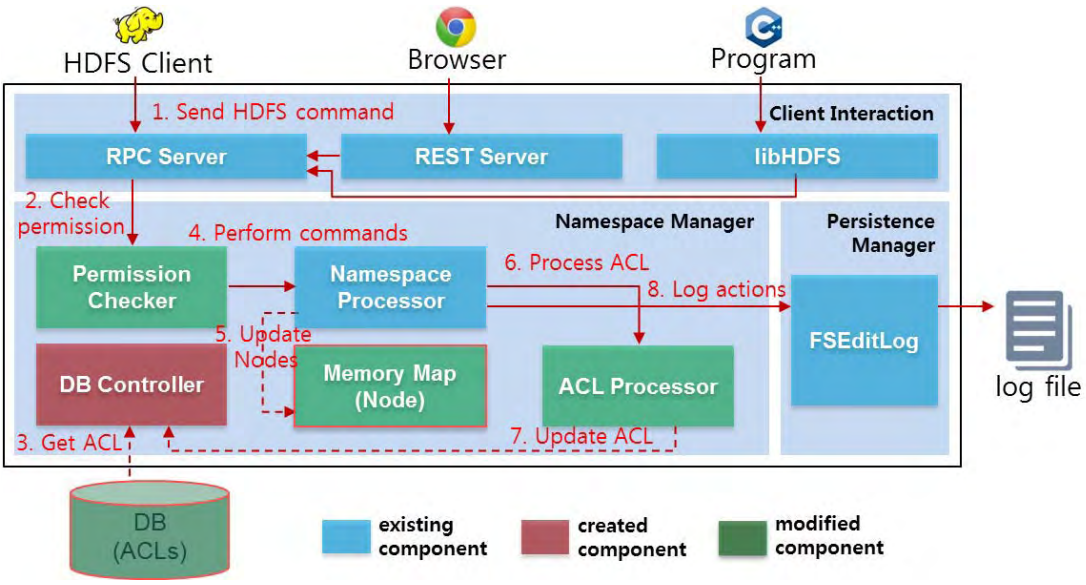
(그림 3)은 본 논문에서 제안하는 RDBMS 기반의 ACL 작업에 대한 흐름도이다. 외부적으로 기존 인메모리 방식과 같은 인터페이스를 유지하기 위해 기존 컴포넌트 중 ACL과 관련된 내용만을 수정하였다. 추가 및 수정한 컴포넌트는 다음과 같다.

4.1. 신규 추가한 컴포넌트

- DB Controller: DB Controller는 DB와의 접속 정보를 관리하며, DB에 ACL 관리 테이블이 존재하지 않을 경우 테이블을 생성하는 작업을 수행한다. 또한, ACL 정보의 수정/삭제/확인 작업 등 다른 컴포넌트와의 인터페이스를 지원한다.

4.2. 수정한 컴포넌트

- Permission Checker: 인메모리로부터 ACL 정보를 가져와 접근 가능 여부를 확인하는 컴포넌트로서, DB Controller를 통해 DB로부터 ACL 정보를 가져오도록 수정한다.
- Memory Map: ACL 정보를 저장하고 있는 Memory Map 컴포넌트의 내용 중, ACL과 관련된 내용은 불필요하므로 삭제한다.
- ACL Processor: ACL 정보의 수정/삭제를 처리하는 컴포넌트로서, Memory Map과의 연결을 DB Controller로 수정하여 ACL 정보를 DB에 수정/삭제하도록 한다.



(그림 3) RDBMS 기반의 ACL 작업 흐름도.

5. 결론 및 향후 연구

본 논문은 기존 하둡에서 인메모리 기반 ACL의 한계점을 설명하고, 이를 해결하기 위해 RDBMS에서 ACL을 관리하도록 설계하였다. 하지만 ACL 정보를 대용량으로 저장할 경우 인메모리 기반 ACL에 비해 속도가 크게 떨어지게 된다. 따라서, 향후 연구로는 설계한 RDBMS 기반 ACL을 구현하고, 이를 바탕으로 인덱스를 사용하여 대용량 ACL 관리에서의 속도 저하 문제를 해결하고자 한다. 또한 인메모리에 저장된 메타데이터를 RDBMS에서 관리함으로써 네임노드의 부담을 최소화하고자 한다.

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] Dhruba Borthakur, “The Hadoop Distributed File System: Architecture and Design,” Technical Report, http://hadoop.apache.org/core, pp. 1-14, 2007.

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

[10] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, “Building a High-Level Dataflow System on Top of MapReduce: The Pig Experience,” In *Proc. of Very Large Data Bases*, Vol. 2, No. 2, pp. 1414-1425, Aug. 2009.

[11] R. S. Sandhu, and P. Samarati, “Access Control: Principles and Practice,” *IEEE Communications Magazine*, Vol. 32, Issue 9, pp. 40–48, Sept. 1994.

[12] Implementation of ACLs in HDFS, https://issues.apache.org/jira/browse/HDFS-4685.