인메모리 관계형 데이터베이스 시스템을 이용한 대용량 텍스트 로그 데이터의 패턴 매칭 방법[1]

한혁*, 최재용*, 진성일**
*㈜리얼타임테크
**충남대학교 컴퓨터공학과
e-mail : hhan@realtimetech.co.kr

A Pattern Matching Method of Large-Size Text Log Data using In-Memory Relational Database System

Hyeok Han*, Jae-Yong Choi*, Sung-Il Jin**
*Realtimetech Inc.
**Dept. of Computer Engineering, Chungnam National University

요 약

각종 사이버 범죄가 증가함에 따라 실시간 모니터링을 통한 사전 탐지 기술뿐만 아니라, 사후 원인 분석을 통한 사고 재발 방지 기술의 중요성이 증가하고 있다. 사후 분석은 시스템에서 생산된 다양한 유형의 대용량 로그를 기반으로 분석가가 보안 위협 과정을 규명하는 것으로 이를 지원하는 다양한 상용 및 오픈 소스 SW 가 존재하나, 대부분 단일 분석가 PC 에서 운용되는 파일 기반 SW 로 대용량 데이터에 대한 분석 성능 저하, 다수 분석가 간의 데이터 공유 불가, 통계·연관 분석 한계 및 대화형·점진적 내용 분석 불가 등의 문제점을 해결하지 못하고 있다. 이러한 문제점을 해결하기 위하여 고성능 인메모리 관계형 데이터베이스 시스템을 로그 스토리지로 활용하는 대용량 로그 분석 SW 개발하였다. 특히, 기 확보된 공격자 프로파일을 활용하여 공격의 유무를 확인하는 텍스트 패턴 매칭 연산은 전통적인 관계형 데이터베이스 시스템의 FTS(Full-Text Search) 기능 활용이 가능하나, 대용량 전용 색인 생성에 따른 비현실적인 DB 구축 소요 시간과 최소 3 배 이상의 DB 용량 증가로 인한 시스템 리소스 추가 요구 등의 단점이 있다. 본 논문에서는 인메모리 관계형 데이터베이스 시스템 기반 효율적인 텍스트 패턴 매칭 연산을 위하여, 고성능의 대용량 로그 DB 적재 방법과 새로운 유형의 패턴 매칭 방법을 제안하였다

1. 서론

최근 디도스(DDOS) 공격 및 개인정보 유출 등의 보안 침해사고 피해가 급증하면서 실시간 사전 탐지 기술뿐만 아니라 사후 정확한 원인 분석을 통한 침해사고 재발 방지 목적의 사후 분석 활동 중요성이 강조되고 있다.

웹 서버, Firewall 및 시스템 등의 로그 분석 활동은 사후 분석의 대표적인 주요 분석 유형으로 이를 지원하는 Web Log Explorer[1], Forensic Explorer[2] 등의 전용 SW 가 활용되고 있다. Web Log Explorer 는 웹 로그에 대한 통계 분석이 가능한 SW 이며, Forensic Explorer 는 시스템 이미지에 대해 패턴 매칭을 비롯한 다양한 로그 분석 기능을 제공한다. 그러나 이러한 로그 분석 도구들은 단일 분석가가 소규모의 로그 파일을 분석할 수 있는 도구로, 수십 GB 이상의 대용량 로그 분석 시 성능 저하로 인한 분석 효율성 저하 및 경우에 따라서는 분석 자체가 불가능하며, 다수의 분석가가 데이터 및 중간 결과물 공유하는 효율적인 분석 활동을 수행할 수 없는 단점이 있다. 본 논문에서는 이러한 문제점들을 해결하기 위하여, 파일 시스템 기반이 아닌 고성능 인메모리 데이터베이스 시스템을 로그 스토리지 서버로 활용하여 대용량 텍스트 로그를 분석하는 방법에 대해 제안하였다.

특히, 기 확보된 보안 공격자 프로파일을 활용하여 공격의 유무를 확인하는 텍스트 패턴 매칭 연산은 관계형 데이터베이스 시스템을 로그 스토리지로 활용하는 경우, 데이터베이스 시스템이 제공하는 LIKE 연산자를 활용하거나 FTS(Full-Text Search) 기능을 활용하는 방법이 있다. LIKE 연산의 경우 전문 검색 결과와 동일한 결과를 얻기 위해서는 데이터베이스 구축 시 원본 로그를 레코드 단위로 분해하여 테이블을 생성해야 하므로 데이터베이스 구축 시간이 길어지고, 검

[1] 본 연구는 2016 년도 우수기술연구센터(ATC)사업 (과제명: 익스트림 트랜잭션 및 분산 확장성을 제공하는 대용량 비휘발성 메모리(SCM) 기반의 차세대 인메모리 빅 데이터베이스 시스템 상용 기술 개발)의 지원을 받았음.

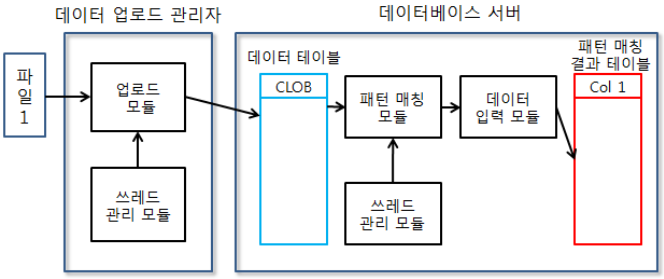
색 성능이 느린 단점이 있으며, FTS 의 경우에는 검색 속도는 빠르나, 대용량 로그 분석의 경우 전용 색인 생성에 따른 장시간의 DB 적재 시간 및 원본 대비 3 배 이상의 추가적인 저장소 용량이 요구되어 사후 분석에 활용하기에는 적합하지 않다. 본 논문에서는 이러한 문제점들을 해결하기 위하여 대용량 로그에 대한 고성능 DB 적재 및 패턴 매칭 연산을 제안하였으며, 이러한 방법은 전체적인 사후 분석 수행 주기를 단축하는 효과를 기대할 수 있다.

2. 관련 연구

데이터베이스 시스템 또는 유사 기술을 활용한 침입 탐지 및 로그 분석에 관한 연구 사례로는 데이터 마이닝 기법을 활용한 침입 탐지 연구[4], CEP 엔진을 사용하여 온라인 게임의 악용 패턴을 모니터링한 연구[5], 웹 환경에서 발생하는 모든 데이터에 대해 실시간으로 패턴 매칭을 수행하여 데이터를 분류하고 유용한 데이터만 데이터베이스에 저장하는 연구[6]등이 있다. 본 논문과 가장 유사한 연구 사례로는 데이터베이스 시스템 기반으로 로그 데이터를 저장 후 패턴 매칭을 수행하는 방법으로 침입 탐지를 자동화한 연구[3]가 있으나, 원본 로그를 레코드 단위로 분해하여 테이블을 구성하고 LIKE 연산 기반 패턴 매칭을 수행한 연구 사례로 대용량 로그 분석 시의 성능 문제 해결에는 한계가 있다.

3. 시스템 설계 및 구현

그림 1 은 본 논문에서 제안한 방법을 구현하기 위해 설계한 시스템의 구성도이다. 시스템은 크게 데이터 업로드 관리자와 데이터베이스 서버로 나뉜다. 데이터 업로드 관리자와 데이터베이스 서버 모두 쓰레드 관리 모듈이 존재하며 멀티 쓰레드 기반으로 데이터 업로드와 패턴 매칭을 진행한다. 본 논문에서는 국산 인메모리 관계형 DBMS 인 Kairos 기반으로 개발하였다.



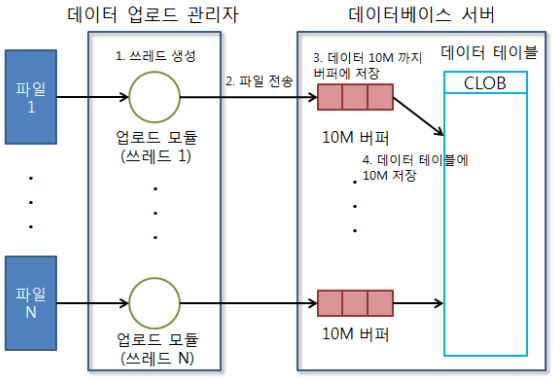
(그림 1) 전체 시스템 구성도

먼저 1 절에서는 데이터 입력 모듈의 동작 방식에 대해 설명하고, 2 절에서는 대용량 텍스트 데이터에 대해 패턴 매칭을 진행하는 방법에 대하여 설명한다. 3 절에서는 패턴 매칭이 제공하는 두 가지의 OR, AND 연산에 대하여 설명한다.

3.1 데이터 입력 모듈

본 논문에서는 대용량 데이터를 효과적으로 데이터베이스 시스템에 업로드(적재)하기 위해 기존 로그 데이터의 레코드 단위 대신 데이터베이스의 CLOB 컬럼을 이용하여 데이터를 입력한다. CLOB 컬럼의 경우 시스템에 따라 GB 단위까지 데이터를 한 컬럼에 저장이 가능하다. 하지만 파일을 GB 단위로 저장하게 되면 패턴 매칭 시 쓰레드를 효과적으로 사용할 수 없기 때문에 구현 시스템에서는 CLOB 컬럼에 10Mbytes 단위로 데이터를 입력하였다.

그림 2 는 원본 로그 파일을 데이터베이스 테이블에 입력하는 과정을 나타낸다. 업로드 관리자는 적재할 대상 로그 파일 마다 하나의 전담 쓰레드를 생성한다. 각 쓰레드는 업로드 모듈을 실행하여 원본 로그 파일을 읽고 이를 데이터베이스 시스템에 전송한다. 데이터베이스 시스템은 각 클라이언트 세션을 담당하는 서버 쓰레드가 받은 데이터를 10MBytes 버퍼에 저장하여 테이블의 CLOB 컬럼에 입력한다.



(그림 2) 로그 데이터를 입력하는 과정

데이터를 대용량 테이블에 특정 크기(10M)로 입력할 경우 전체 레코드 수가 줄어들기 때문에 전체적으로 데이터 입력시간이 줄어들게 된다. 예를 들어 약 1GB 로그 파일의 경우, 한 레코드의 크기를 약 100 bytes 로 가정하면 전체 레코드의 수는 약 1 천만개이다. 하지만 10M 단위로 데이터를 저장하게 되면 전체 레코드의 수는 100 개로 줄어들기 때문에 데이터를 입력하는 시간이 줄어들어 되며, 원본 로그 파일 단위의 쓰레드가 처리하므로 전체 데이터 입력 시간이 줄어든다.

3.2 패턴 매칭 모듈

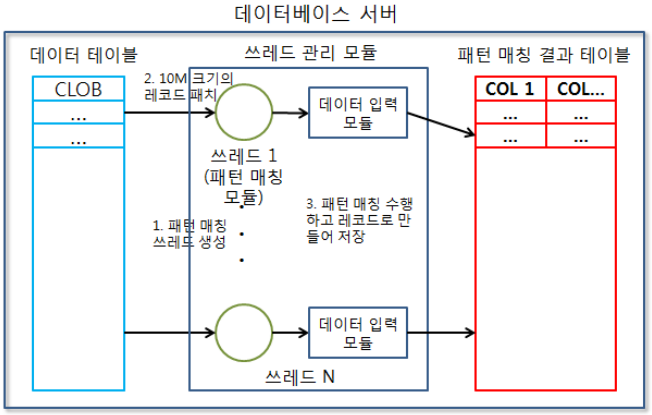
본 논문에서는 대용량 자료형(CLOB)에 저장된 원본 데이터 기반 패턴 매칭 연산을 수행하기 위하여 그림 3 과 같이 새로운 유형의 SQL 문을 설계하였다.

```
INSERT INTO result_table
        MATCHING_TABLE(source_table,source_column)
        MATCH IN ('key_word-1' AND 'key_word-2');
```

(그림 3) INSERT..MATCH 질의

MATCHING_TABLE 절은 패턴 매칭 대상 테이블의 명세를, MATCH IN 절은 패턴 매칭 검색 키워드를 나타내며, 검색의 결과는 *result_table* 에 레코드로 저장된다.

그림 4 은 저장된 로그 데이터에 대해 패턴 매칭을 수행하는 과정을 나타낸다. 쓰레드 관리 모듈은 특정 수만큼 패턴 매칭 쓰레드를 생성한다. 해당 쓰레드들은 원본 로그 테이블에 저장된 10M 단위의 레코드를 가져와 요청된 키워드에 대한 패턴 매칭을 수행한다. 패턴 매칭 시, 찾은 키워드를 기준으로 해당 키워드가 속한 로그 레코드의 시작 위치를 데이터 입력 모듈로 전달하게 되며, 데이터 입력 모듈은 레코드 시작 위치를 기준으로 로그 레코드를 생성하여 패턴 매칭 결과 테이블에 해당 레코드를 입력한다.



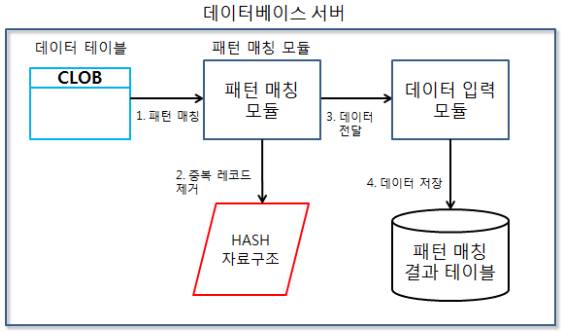
(그림 4) 로그 데이터에 대해 패턴 매칭하는 과정

그림 5 는 패턴 매칭 후 해당 키워드가 속한 레코드를 저장하는 과정을 나타낸다.

(그림 5) 패턴 매칭 후 결과 레코드 입력 과정

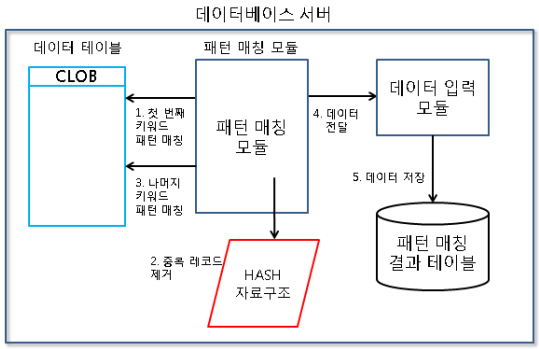
3.3 패턴 매칭 연산

본 논문에서 개발한 패턴 매칭 모듈은 OR, AND 연산 두 가지를 제공한다. OR 와 AND 를 혼합한 연산은 지원하지 않으며, 패턴 매칭 시에는 하나의 연산만 적용할 수 있다.



(그림 6) 패턴 매칭 모듈의 OR 연산 흐름도

그림 6 는 패턴 매칭 모듈의 OR 연산의 흐름을 나타낸다. 패턴 매칭 모듈은 데이터로부터 패턴 매칭을 진행하여 각 키워드가 나타난 레코드의 시작 위치를 계산한다. 계산된 시작 위치를 key 로 HASH 자료구조를 사용하여 로그 레코드의 중복을 제거한다. 그 후 중복 제거된 레코드 시작 위치들을 데이터 입력 모듈로 전달 후 *result_table* 에 저장한다.



(그림 7) 패턴 매칭 모듈의 AND 연산 흐름도

그림 7 은 AND 연산 과정을 나타낸다. AND 연산에서는 우선 첫 번째 키워드로 패턴 매칭을 진행한다. 패턴 매칭을 진행한 후, HASH 자료 구조를 이용하여 매칭된 레코드 시작 위치들의 중복을 제거한다. 제거 후 계산된 레코드의 시작 위치를 기준으로 나머지 키워드에 대해 패턴 매칭을 진행한다. 이 부분에서 모든 키워드에 대해 패턴 매칭이 성공하면 AND 연산이 성공한 것으로 판단한다. 모든 키워드에 대해 매칭된 레코드 시작 위치를 데이터 입력 모듈로 전달 후 해당 레코드를 *result_table* 에 저장한다.

4. 실험

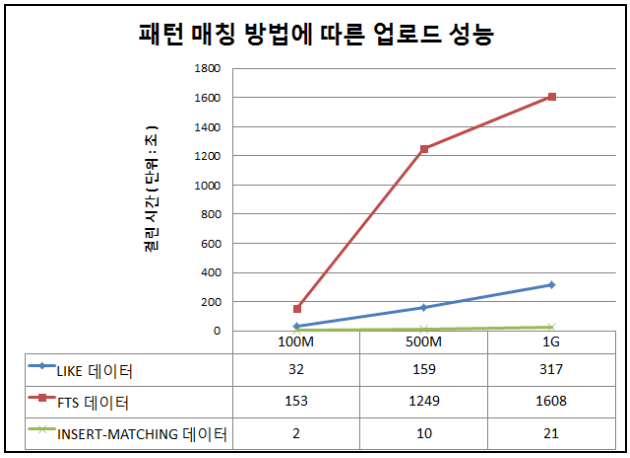
본 논문에서는 제안한 업로드와 패턴 매칭 방법에 대하여 LIKE 를 이용한 패턴 매칭 방법과 FTS 를 이용한 패턴 매칭 방법을 비교 평가하였다. 실험은 클라이언트 SW 와 데이터베이스 시스템 모두 동일한 장비에 설치하여 실험하였으며, LIKE 와 FTS 를 이용한 패턴 매칭의 경우는 기반 데이터베이스 시스템(Kairos)에서 제공하는 업로드 유틸리티(Kloader)를 사용하여 데이터를 적재하였다. 데이터는 apache 웹 서버의 샘플 로그를 사용하였다. 실험 환경 및 SQL 은 표 1 과 같다.

<표 1> 실험 환경

구분	설 명
CPU	Intel Xeon E5-2637 v3
No. of Processors	16
RAM	256
OS	Windows Server 2012 R2

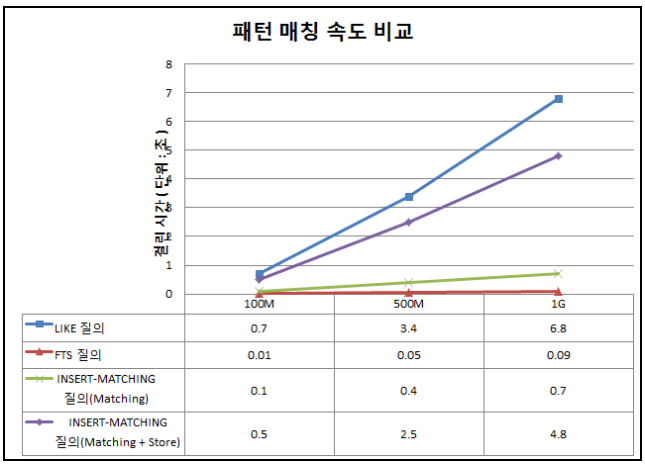
구분	SQL
LIKE	SELECT COUNT(*) FROM *source_table* WHERE *text_field* LIKE '%login%';
FTS	SELECT COUNT(*) FROM *source_table* WHERE CONTAINS(*text_field*, 'login') > 0;
INSERT.. MATCH	INSERT INTO *result_table* MATCHING_TABLE(*source_table, text_field*) MATCH IN('login');

그림 8 은 패턴 매칭 방법에 따른 업로드 성능을 나타낸다. LIKE 의 경우 인덱스가 없기 때문에 단순히 데이터를 업로드 하는 성능만을 나타낸다. FTS 의 경우 데이터를 입력하면서 인덱스를 생성하기 때문에 데이터 용량이 클수록 시간이 더 오래 걸리는 단점이 있다. 본 논문에서 제안한 방법은 CLOB 컬럼에 10Mbytes 단위로 데이터를 넣기 때문에 업로드 성능은 다른 방법과 비교하여 매우 빠른 것을 볼 수 있다.



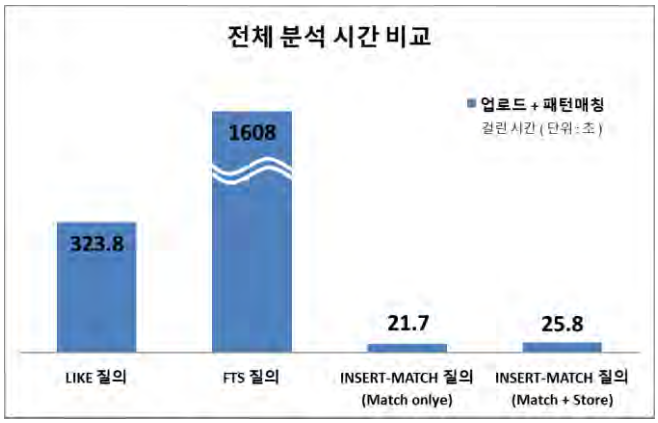
(그림 8) 패턴 매칭 방법에 따른 업로드 성능

그림 9 에서는 각 방법의 패턴 매칭 속도를 비교해 놓았다. LIKE 의 경우 인덱스가 없이 순수 모든 레코드를 탐색하는 시간을 나타낸다. FTS 의 경우 데이터베이스 적재 성능은 다른 방법보다 많이 느렸지만, 인덱스가 있는 상태이므로 패턴 매칭 속도에서 월등한 성능을 보여준다. 본 논문에서 제안한 방법은 LIKE 의 성능과 비슷하거나 약간 더 좋은 성능을 보여주고 있다.



(그림 9) 패턴 매칭 속도 비교

실험 결과, LIKE 를 이용한 패턴 매칭 방법은 패턴 매칭 성능에 있어선 본 논문에서 제안한 방법과 비슷하였지만 업로드 성능에 있어 속도가 느린 단점이 있었다. FTS 의 경우 업로드 성능이 매우 느리지만 검색 성능이 매우 빨랐다. 하지만 데이터를 입력하고 분석하는 전체적인 과정을 보면 그림 10 에서와 같이 본 논문에서 제안한 방법이 침해 발생 후 데이터베이스 적재(업로드)와 패턴 매칭을 포함하는 전체적인 시간에 있어 차별화 성능을 보이는 것을 확인할 수 있다.



(그림 10) 전체 분석 시간 비교

5. 결론

본 논문에서는 관계형 데이터베이스를 이용하여 대용량 텍스트 로그 데이터를 입력하고 분석하는 방법을 제안하였다. 기존의 로그 분석도구를 사용한 로그 분석은 작은 용량의 로그에 대하여 효율적이었지만 대용량 및 여러 분석가가 동시에 한 데이터에 대해 분석이 불가능한 단점이 있었다. 또한 데이터베이스 시스템의 FTS 를 사용하여 로그를 분석하는 방법은 데이터의 업로드 성능이 좋지 않아 침해 사고 후 분석 시 시간이 오래 걸리는 단점이 있었다.

본 논문에서는 이러한 문제들을 해결하고자 데이터베이스의 CLOB 컬럼을 이용하여 데이터를 입력하고 패턴 매칭을 하는 방법에 대하여 제안하였다. 또한 실험을 통해 기존 FTS 를 사용한 로그 분석보다 전체적인 로그 분석 시간을 단축할 수 있는 것을 확인하였다.

본 논문의 향후 연구로는 데이터베이스를 이용하여 패턴 매칭 뿐만 아니라 통계 분석을 하는 방법을 제안할 예정이다.

참고문헌

[1] http://www.exacttrend.com/WebLogExplorer/
[2] http://www.forensicexplorer.com/
[3]신상윤, 장원태, 연규철, 김영철, “자동화 침입탐지 데이터베이스 시스템의 개발”, 보안공학연구논문지 Vol.9 No.6, 2012.12
[4]편석범, 정종근, 이윤배, “새로운 침입 패턴을 위한 데이터 마이닝 침입 탐지 시스템 설계”, 전자공학회논문지 TE 편 제 39 권 제 1 호, 2002.3
[5]노창현, “CEP 기반 온라인 게임 악용 패턴 모니터링 방법”, 한국콘텐츠학회논문지 제 10 권 제 1 호, 2010.1
[6]강만모, 구자록, 이동형, “차세대 웹 환경에서 Complex Event Processing 엔진을 이용한 대용량데이터 처리”, 정보과학회논문지 : 데이타베이스 제 37 권 제 6 호, 2010.12