Hadoop 분산 처리 환경에서 페이로드 시그니쳐에 기반한 HTTP 트래픽 분석 시스템의 처리 속도 향상

심규석, 이수강, 김명섭
고려대학교

{kusuk007, sukanglee, tmskim}@korea.ac.kr

Processing Speed Improvement of HTTP Traffic Classification Based on Payload Signature on Hadoop Distributed Computing Environment

Kyu-Seok Shim, Su-Kang Lee, Myung-Sup Kim
Dept. of Computer and Information Science, Korea Univ.

요 약

기존 페이로드 시그니쳐 기반 응용 레벨 트래픽 분석 방법은 문자열 매칭하는 과정에서 시스템에 많은 부하를 일으키며, 처리 속도가 느려지는 문제가 존재한다. 따라서 시그니쳐 기반 트래픽 분석 방법에서 처리 속도 향상에 대한 연구는 끊임없이 진행되고 있다. 본 논문에서는 처리 속도와 저장 공간의 향상을 위해 Hadoop 분산 처리 환경에서 페이로드 시그니쳐 기반 응용 분석 방법을 언급한다. 그러나 기존 시스템의 문자열 매칭 방법은 시스템의 부하를 상대적으로 많이 야기하는 regular expression 방법을 사용한다. 따라서 본 시스템의 문자열 매칭에 Aho-Corasick 알고리즘 방법을 적용한다. 기존 시스템과 Aho-Corasick 알고리즘 적용 시스템 처리속도에 대한 비교 실험을 통해 평균적으로 20 초의 처리속도가 더 효율적인 것을 증명하였다.

Ⅰ. 서 론

오늘날 네트워크 트래픽은 다양한 네트워크 트래픽을 사용하는 응용의 개발과 신속한 서비스의 요구 충족을 만족시키면서 증가하고 있다. 이러한 네트워크 트래픽 사용량의 증가로 인해 네트워크 트래픽 관리를 위한 트래픽 분석은 필수적이다. 트래픽 분석을 통해 응용 분류를 함으로써 네트워크 자원을 절약할 수 있고, 사용자에게 선택적으로 더 신속한 서비스를 제공할 수 있다. 그러나 네트워크 트래픽 사용량의 증가는 트래픽 분석의 어려움을 야기한다. 분석 대상이 많아지면서 분류 시스템의 부하로 인해 처리 속도가 느려지고, 저장 공간에 대한 부담은 증가한다.

따라서 본 논문은 기존 많은 양의 트래픽을 분석하기 위해 구현된 Hadoop 분산 처리 환경에서 시그니쳐 기반 응용 트래픽 분류 시스템[1]의 문자열 매칭 방법을 향상시키며 처리 속도 향상을 기대하는 방법을 제안한다. 기존 시스템은 JAVA 에서 기본적으로 제공하는 matches 함수를 이용하여 문자열 매칭이 이루어졌다. 그러나 본 방법은 regular expression 을 사용한다. reqular expression 은 simple string 비교 보다 더 많은 시스템 부하를 야기한다. 따라서 본 논문은 문자열 매칭 방법에 Aho-Corasick 알고리즘[3]을 적용하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 본 장의 서론에 이어, 2 장에서 Hadoop 분산 처리 환경에서 시그니쳐 기반 응용 분류 시스템에 대해 언급한다. 3 장에서 언급된 시스템에서의 문자열 매칭 방법을 Aho-Corasick 알고리즘 적용하는 것을 제안한다. 4 장에서 기존 방법과 비교 실험으로 Aho-Corasick 알고리즘 적용 방법이 더 효과적인 것을 증명하고, 마지막으로 결론 및 향후 연구에 대해 기술한다.

본 연구는 2013 년 BK21 플러스 사업 및 2012 년 정부(교육과학기술부) 의 재원으로 한국연구재단(2012R1A1A2007483)의 지원을 받아 수행된 결과임.

Ⅱ. Hadoop 환경에서 시그니쳐 기반 응용 분석 방법

본 장에서는 Hadoop 환경에서의 시그니쳐 기반 응용 분석 시스템의 구조와 과정에 대해 기술한다. Hadoop 분산 처리 환경에서 페이로드 시그니쳐 기반 응용 분석 방법은 기존 하나의 머신에서 실행되는 페이로드 시그니쳐 기반 응용 분석 방법을 Hadoop 환경에서 구현한 것이다. Hadoop 은 분산 처리 기능과 분산 저장소 기능을 모두 지원하는 플랫폼이다.

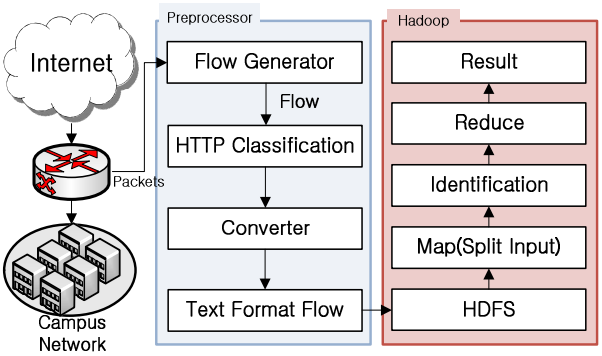

그림 1. Hadoop 환경에서의 시그니쳐 기반 트래픽 분석 과정

먼저 네트워크 트래픽이 수집되고, 분석되는 과정은 그림 1 과 같다. 본 시스템은 두 가지 과정으로 나누어진다. 첫 번째 과정은 Flow 정보가 HDFS(Hadoop Distributed File System)로 저장되기 전까지의 과정이다. HDFS 는 Hadoop 에서 지원하는 분산 저장소이다. 학내에서 발생하는 트래픽을 packet 단위로 수집해서 flow 생성기를 통해 binary 형태의 flow 파일을 생성한다. 본 시스템에서 flow 는 5-tuple(source IP, source Port, Protocol, destination IP, destination Port)가 같은 packet 의 집합으로 정의하고, flow 파일은 5-tuple, flow 에 포함된 packet 의 개수, flow 가 전송한 byte 양, 첫 번째 request packet 의 페이로드 등 8 가지 정보를 포함한다. 생성된 flow 에서 HTTP 트래픽만을 분류한다. HTTP 트래픽이 아닌 암호화 된 트래픽은 페이로드를 분석하는데 한계가 있기 때문이다. 마지막 단계로 분류된 binary 형태의 flow 파일을 Hadoop 시스템에서 분석하기 위해서 text 형태의 flow 파일로 변환시켜야 한다.

첫 번째 과정이 네트워크 트래픽 분석을 하기 위한 전처리 단계였다면, 두 번째 과정은 Hadoop 환경에서 페이로드 시그니처 기반 응용 분석 과정이다. 첫 번째 과정에서 마지막 단계에서 변환시켰던 text 형태의 flow 파일을 HDFS 로 저장한다.

다음 단계는 MapReduce 단계이다. MapReduce 는 Hadoop 에서 지원하는 분산 연산 기능이다. Map 은 데이터를 분할하여 처리하고, Reduce 는 분할된 데이터를 다시 합치는 역할을 한다. 본 시스템에서 Map 은 HDFS 에 저장된 text 형태의 Flow 파일에서 하나의 Flow 씩 입력 받는다. 하나의 Flow 를 입력 받은 Map 은 정의된 시그니처와 flow 정보를 비교 분석하여 응용을 분류하고, 분류된 결과를 Reduce 로 보내게 된다. Reduce 는 분산되어 있는 결과를 다시 합쳐서 사용자에게 결과를 보여주는 역할을 한다.

그러나 Map 에서 정의된 시그니처와 flow 정보를 문자열 매칭할 때, JAVA 에서 제공되는 라이브러리인 matches 함수를 사용한다. 그러나 matches 함수는 regular expression 방법이기 때문에 시스템에 많은 부하를 야기할 수 있다. 따라서 본 논문에서는 문자열 매칭할 때, Aho-Corasick 알고리즘을 적용하는 방법을 제안한다.

Ⅲ. Aho-Corasick 알고리즘

페이로드 시그니처 기반 트래픽 분석 툴인 Snort[2]에서 사용되는 Aho-Corasick 알고리즘은 멀티플 패턴 매칭에 기반한 알고리즘이다. 복수 개의 시그니처를 하나의 트리로 구성하여 한번의 트리 탐색으로 분류가 가능하기 때문에 시그니처에 대한 탐색 공간을 줄일 수 있다. 따라서 본 장에서는 Hadoop 분산 처리 환경에서 시그니처 기반 응용 트래픽 분석 방법에 Aho-Corasick 알고리즘을 적용하는 방법을 제안한다.

그림 2 는 Aho-Corasick 알고리즘의 문자열 매칭 방법의 간단한 예이다. 예를 들어, 대형 포탈 사이트인 naver, nate 의 시그니처가 각각 naver, nate 라면 Aho-Corasick 알고리즘은 다음과 같은 트리를 만든다.

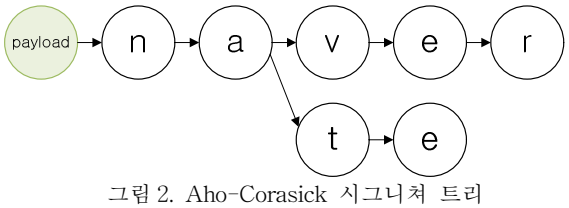

그림 2. Aho-Corasick 시그니처 트리

Aho-Corasick 알고리즘을 적용하면 다음과 같이 정의된 모든 시그니처에 대한 트리를 구성하여, 문자열 매칭이 이루어진다. 페이로드가 입력될 때마다 모든 시그니처를 하나씩 비교하는 방법보다 한번에 모든 시그니처를 비교하는 것은 시간복잡도를 상당히 감소시킨다.

Ⅳ. 실험 및 결과

본 장에서는 기존 regular expression 방법으로 문자열 매칭할 때의 처리속도와 Aho-Corasick 알고리즘 을 적용하여 매칭할 때의 처리속도를 비교한 결과를 기술한다. 실험은 하루에서 5 일동안의 트래픽 데이터를 분석한다.

그림 3. Aho-Corasick 과 matches 함수 처리속도 비교

실험 결과 Aho-Corasick 알고리즘을 적용한 시스템이 보다 더 효율적인 것을 확인하였다. 평균적으로 20 초의 차이를 보인다.

Ⅴ. 결론 및 향후 연구

본 논문에서는 HTTP 트래픽을 Hadoop 분산 처리 환경에서 페이로드 시그니처 기반 응용 분석 시스템에서 문자열 매칭 방법을 Aho-Corasick 알고리즘으로 적용하였다. 기존 regular expression 방법을 사용할 때와 비교하였을 때, 시스템 부하로 인한 처리 속도가 감소되었음을 확인하였다. 본 방법은 시그니처의 종류가 다양해 질수록 더 많은 효과를 기대할 수 있다.

향후 연구로는 본 시스템의 처리속도를 줄이기 위해 flow 파일을 text 형태로 변환하는 과정 없이 HDFS 상에 binary 형태의 flow 파일을 저장하여 분석하는 시스템 구현이 필요하다.

참 고 문 헌

[1] 심규석, 이수강, 김성민, 김명섭, "하둡 분산 컴퓨팅 환경에서 페이로드 시그니처 기반 응용 트래픽 분류 ", 2014 년 통신망운용관리 학술대회 (KNOM 2014), 충남대학교, 대전, May. 15-16, 2014, pp.77-81.

[2] M Roesch "Snort: Lightweight Intrusion Detection for Networks." LISA. Vol. 99. 1999 pp 229-238. 1999.

[3] Aho, A.V., Corasick, M.J., " Efficient string matching: an aid to bibliographic search." Commun. ACM 18 (6), pp 333- 340. 1975.