네트워크 패킷 트랜드 분석을 위한 실시간 스트림 데이터 분석 시스템 설계 및 구현

박서은*
*고려대학교 컴퓨터정보통신대학원
e-mail : seoeun25@gmail.com

Design and Implementation of a Real -Time Analytics System for Network Packet Trend Analysis

Seoeun Park*
*Graduate School of Computer Information & Communication, Korea University

요 약

스마트폰, 센서, 소셜미디어, 웹 서비스 등으로부터 발생되는 데이터의 폭증으로 인하여 빅데이터의 분석 및 활용에 대한 요구가 커져가고 있다. 특히 스마트 기기의 발달과 사용자 이용 패턴의 변화로 인하여 스트림 데이터는 끊임없이 발생되고 있지만, 기존의 하둡을 이용한 분석 시스템은 응답시간이 지연되어 빠르게 결과를 조회할 수 없는 단점으로 인하여 데이터를 실시간으로 분석하여 바로 활용할 수 있는 시스템에 대한 요구가 점점 더 증가하면서 람다 아키텍쳐가 등장하였다. 람다 아키텍쳐는 데이터 처리 과정을 배치 레이어와 스피트 레이어로 나누고, 스피드 레이어에서는 배치 결과가 나오기 전까지 스트림으로 유입되는 데이터를 실시간으로 분석하여 가장 최근의 데이터를 빠르게 조회 할 수 있도록 결과를 제공한다. 본 논문에서는 람다 아키텍쳐를 활용하여 연속적으로 유입되는 대용량의 스트림 데이터를 효과적으로 처리하여 실시간 분석과 동시에 배치 분석을 제공하는 데이터 처리 시스템을 설계하고 구현한다.

Keyword : real time analytics, big data, lambda architecture

1. 서론

최근 생산되는 디지털 데이터의 양은 해마다 그 증가폭이 커지고 있다. 조사에 따르면 2013 년 생산된 전 세계 총 데이터량은 4.4 조 기가바이트이며, 2020 년에는 약 10 배가 증가한 44 조 기가바이트에 달할 것으로 전망했다[1]. 이렇게 새로운 자산이 된 빅데이터는 가장 관심있는 이슈 중의 하나로 떠오르고 있고, 기업이나 국가에서는 대용량의 데이터를 분석하기 위해서 하둡 맵리듀스 시스템을 도입하여 데이터 분석에 활용하고 있다[2]. 맵리듀스는 페타바이트 이상의 데이터를 여러 노드로 구성된 클러스터 환경에서 병렬로 처리하는 기법으로 맵과 리듀스 방식을 사용하여 데이터를 처리한다. 따라서 대량의 데이터를 효과적으로 분산 처리 할 수 있는 장점이 있지만 배치 방식으로 데이터를 처리하기 때문에 빠른 쿼리 조회는 어려운 단점이 있다.

그러나 요즘의 데이터 분석은 과거의 분석 결과를 배치 리포트로 보는 것 만으로는 부족하다. 이것은 스마트 기기의 발달과 SNS 이용의 증가, 그리고 IOT 의 발달로 대용량의 스트림 데이터가 증가함에 따라 이러한 데이터를 실시간으로 분석하려는 요구가 증가했기 때문이다. 조사에 따르면 2013 년 60 초동안 아마존에서는 8 만 3 천 달러 규모의 판매가 이루어지고, 27 만 8 천 개의 트윗이 포스팅 되며 유투브에는 72 시간 분량의 비디오가 업로드 된다고 한다[3]. 이러한 데이터를 실시간으로 분석하면 사용자의 패턴 분석이나 실시간 트랜드 분석을 가능하게 한다. 통신 회사의 경우 네트워크에서 전송되는 패킷을 분석하면 누가 언제 어디서 무엇을 어떻게 하는 지에 대한 정보를 분석할 수 있기 때문에 네트워크의 이상 탐지나 실시간 사용량 트랜드를 분석할 수 있다. 따라서 기존 배치 방식의 맵리듀스로는 실시간 분석을 제공하기가 어렵기 때문에 이에 대한 해결책으로 스트림 프로세싱 기술이 개발되어 시계열 데이터틀 메모리에서 처리하여 응답시간의 지연 없이 빠르게 조회가 가능하게 되었다.

하지만 사용자들은 대량의 데이터를 배치 프로세싱 방식과 스트림 프로세싱 방식 모두를 사용해서 분석

하기를 원한다. 하둡 배치를 실행시키고 결과를 기다리기 까지는 적어도 한시간 이상의 시간이 소요되기 때문에 실시간으로 분석 결과를 바로 볼 수 없는 단점이 있기 때문이다. 이 두가지 방식으로 동시에 데이터를 처리하고 서빙 레이어를 통해서 두 가지 분석 결과를 조회할 수 있도록 하는 데이터 처리 기법을 람다 아키텍처라고 한다[4]. 본 논문은 람다 아키텍처의 개념을 활용하여 대용량 스트림 데이터를 대상으로 배치 분석, 실시간 분석을 동시에 제공하는 데이터 분석 시스템을 제안하고, 이 시스템을 이용한 네트워크 패킷 분석을 통하여 대용량 데이터의 배치 처리 성능 및 실시간 집계 질의 속도를 검증하고 자 한다.

2. 문헌 정보

(1) 배치 프로세싱 vs 스트림 프로세싱 데이터 처리

배치 기반은 대량의 데이터를 일정한 스케쥴로 일괄 처리 하는 것을 말한다. 스트림 기반은 연속적으로 유입되는 데이터를 이벤트 단위로 처리하거나 작은 시간 단위 내에서 처리하여 결과값을 반환하는 것을 말한다. 일반적으로 수 초 내에 데이터를 처리한다.

<표 1>에서는 배치 프로세싱과 스트림 프로세싱의 특징을 비교 분석하였다.

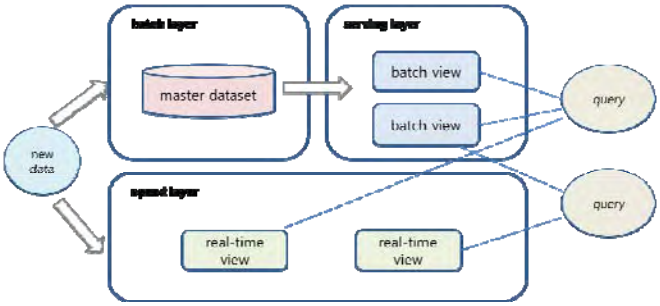
<표 1> 배치 프로세싱 vs 스트림 프로세싱 비교

배치 프로세싱	스트림 프로세싱
* 일반적인 데이터 처리의 개념 * 모든 데이터를 일괄처리 * 대량의 데이터, 다양한 데이터의 처리 * 느린 질의 응답 속도: 수 시간이 될수 있음 * 하둡 맵리듀스	* 한 개의 데이터 혹은 작은 시간 단위의 타임 윈도우 내에서 데이터 처리 * 데이터가 유입되자 마자 바로 처리 * 비교적 단순한 데이터의 처리 * 빠른 질의 응답 속도: 수 초, 수 분 내에 응답 * 스톰, 스파크 스트리밍

(2) 람다 아키텍처

람다 아키텍처는 나단 마르츠가 제안한 데이터 프로세싱 아키텍처로 대량의 데이터를 처리하는 데 있어서 배치 프로세싱과 스트림 프로세싱의 장점을 모두 활용할 수 있도록 설계하는 것이다. 람다 아키텍처는 에러수용성을 통해서 하드웨어나 사람의 실수로 인한 에러에도 복구와 정상 동작을 가능하게 하고, 지연시간과 처리량을 균형있게 분배하여 질의에 대해 빠르게 응답할 수 있도록 설계하는 것을 목표로 한다. 이렇게 설계된 시스템은 선형적 스케일 아웃이 가능해야 한다[5].

(그림 1)은 람다 아키텍처의 개념을 추상화한 도식이다.



(그림 1) 람다 아키텍처 개념

람다 아키텍처는 일과처리, 속도, 서비스의 세 가지 요소가 서로 조화를 이루어 동작하게 하는 것이다.

배치레이어는 모든 데이터를 미리 연산하여 결과를 따로 저장해 놓고, 질의시 결과를 빠르게 반환할 수 있도록 한다. 배치레이어에서 에러가 발생하면 전체 데이터를 다시 연산해서 결과값을 업데이트 한다.

스피드레이어에서는 스트림 데이터를 실시간으로 처리한다. 스피드레이어는 가장 최근의 데이터에 대한 처리 결과를 보여준다. 따라서 처리량보다는 질의에 대해 빠르게 응답하는 것을 목적으로 한다. 또한 배치레이어의 분석 결과가 나오기 전까지 최근 데이터에 대한 분석 결과를 보여준다. 스피드레이어에서 처리한 결과는 배치레이어에서의 처리 결과만큼 완벽하지 않을 수 있지만 데이터가 유입되는 즉시 연산을 수행하기 때문에 실시간 분석을 가능하게 하고, 후에 배치레이어의 분석 결과로 대체될 수 있다.

서빙레이어는 배치레이와 스피드레이어의 처리 결과를 저장해서 애드혹 질의에 대해 빠르게 결과값을 반환하도록 한다. 두 개의 레이어에서 처리된 결과를 드루이드 같은 하나의 저장소에 저장하여 질의를 동시에 처리 할 수도 있고, 각 레이어별로 따로 저장하여 각각의 질의를 처리 할 수도 있다. 배치레이어용 저장소로는 Elephant DB, Cloudera Impala 가 있으며, 스피드레이어 용으로는 Apache HBase, Apache Cassandra 가 있다[4].

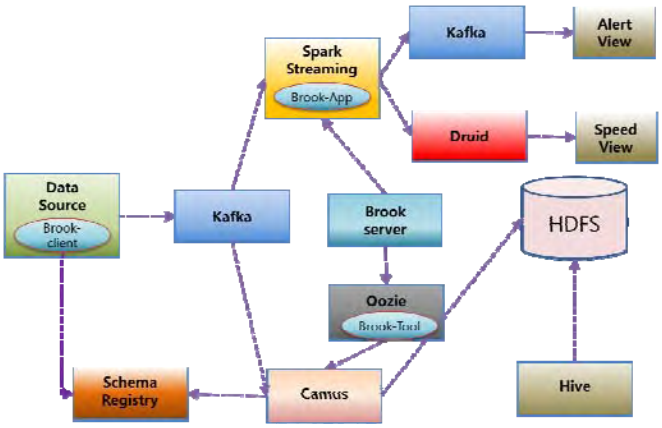
3. 제안 기술 (브룩)

(1) 연구방법

본 연구는 네트워크 사용량 분석 및 실시간 이상 탐지를 위해서 스트림으로 유입되는 웹 패킷을 분석하는 시스템을 설계하는 데서 출발하였다. 최근 한시간 단위로 웹 패킷을 파싱하여 소스 IP 별로 사용량을 집계하여 가장 많이 사용한 IP 를 TopN 처리하는 예는 스트림 프로세싱을 사용해야 한다. 그리고 일간 사용량 분석과 같이 전체 데이터를 분석 하기 위해서는 배치 프로세싱이 필요하다. 따라서 앞선 문헌정보에서 조사한 람다 아키텍처를 활용하여 두 가지 프로세싱을 동시에 처리하는 시스템을 개발하였다.

(2)브룩 시스템 설계

(그림 2)는 브룩의 시스템 설계도 이다.



(그림 1) 브룩 시스템 설계도

스트림 데이터를 위한 파이프라인을 위해서 카프카를 사용하였다. 카프카는 분산 로그 시스템으로 스케일 아웃을 지원하고, 퍼블리쉬-서브스크라이브를 사용하여 메시지를 읽고 쓸 수 있다. 복제본을 사용하기 때문에 클러스터 내의 브로커 중 하나가 중지되어도 고가용성을 제공한다[6].

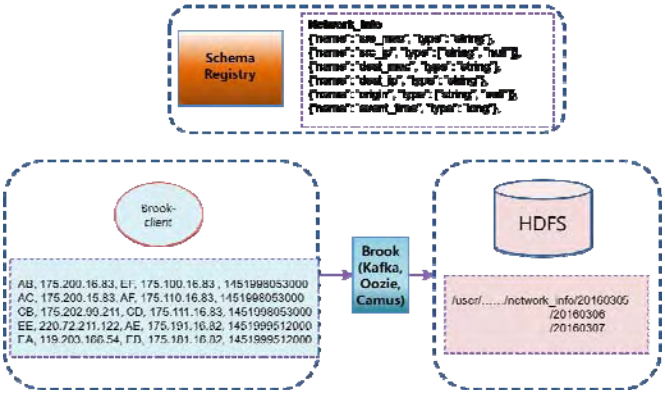
스트림 프로세싱을 위해서는 스파크 스트리밍을 사용하였다. 대안으로 스톰, 쌈자와 같은 컴포넌트가 있지만 스파크 스트리밍을 사용하면 스파크 SQL 과 연결하여 스파크에서도 데이터 조회가 가능하기 때문에 확장성을 위해서 선택하였다[7]. 스파크 스트리밍에서 처리된 데이터는 드루이드에 저장된다. 드루이드는 시계열 데이터에 특화된 OLAP 용 데이터 저장소로 시간단위로 집계하는 데 빠른 성능을 보인다. 대부분의 분석 질의가 시간 단위의 집계 함수, 예를 들면 10 분 동안 소스 IP 에서 사용한 총 패킷량을 계산하는 유형이기 때문에 드루이드를 선택하였다[8].

배치 프로세싱을 위해서는 먼저 카프카에 있는 데이터를 HDFS 에 저장해야 한다. 케머스는 맵리듀스를 사용해서 데이터를 업로드하기 때문에 한꺼번에 대용량 데이터를 업로드 할 수 있는 장점이 있다[9]. 이렇게 HDFS 에 저장된 데이터는 하이브를 통해서 배치 프로세싱을 하게 된다. 하이브는 HQL 이라는 SQL 과 유사한 질의 언어를 제공하기 때문에 SQL 에 익숙한 사용자들이 쉽게 사용할 수 있다[10].

(3)브룩 시스템 동작과정

첫번째 단계는 브룩 클라이언트를 이용하여 데이터를 에이브로 형식으로 카프카에 전송한다. 이렇게 카프카에 저장된 데이터는 배치 프로세싱과 스트림 프로세싱의 원천 데이터가 된다.

배치 프로세싱 과정에서 브룩은 한 시간에 한 번의 주기로 케머스를 사용하여 카프카의 데이터를 HDFS 에 저장한다. 케머스를 실행할 때 맵리듀스 갯수를 조절하는 것이 가능하기 때문에 대용량일 경우 더 많은 맵퍼를 실행하여 분산 처리 할 수 있다. 이 과정에서 브룩 내부의 하이브 링커 모듈은 이벤트의 발생 시간을 기준으로 파티셔닝하여 데이터를 저장하기 때문에 하이브에서 데이터를 조회할 때 날짜를 기준으로 빠르게 접근할 수 있다. (그림 3)은 하이브 링커를 통하여 데이터가 하이브에 저장되기 까지의 흐름을 설명하고 있다.



(그림 3) 배치 프로세싱에서 하이브 링커

스트림 프로세싱 과정에서 카프카의 데이터는 스파크 스트리밍 어플리케이션에 연결된다. 이 데이터는 배치 프로세싱에서 사용한 데이터와 동일한 데이터이다. 스파크 스트리밍 어플리케이션에서는 필터, 조인 등의 작업을 통해 오류 데이터를 수정하고 비정규화된 형식으로 만들어서 드루이드로 전송한다. 드루이드에서는 패킷량, 이벤트 발생 횟수를 소스 IP 를 기준으로 집계하여 한 시간 단위의 파티션에 저장한다.

(4)연구 분석 방법

웹 패킷 로그를 초당 200,000 건의 이벤트로 생성하여 스트림으로 카프카로 전송한 후, 소스 IP 별 총 패킷량과 이벤트 발생 건수를 집계하여 시계열로 보여주는 질의를 배치 프로세싱과 실시간 프로세싱에 동시에 적용하여 응답시간을 측정한다.

4. 연구 분석 결과

(1) 웹 패킷 스트림

130 바이트 내외의 이벤트를 200,000/sec 의 스트림으로 생성한다. <표 2>는 데이터 유입량에 대한 설명이다.

<표 2> 데이터 유입량

	1 second	1 hour	1 day
event count	200,000	7,200,000,000	17,280,000,000
event size	25 (MB)	90 (GB)	2.160(TB)

(2) 배치 vs 스트림 프로세싱의 응답 시간 비교

1 일 데이터를 기준으로 배치 프로세싱과 실시간 프로세싱에서 다음과 같이 한 시간 단위의 시계열 집계 질의를 실행했을 때 걸리는 시간을 측정하여 비교하였다. <표 3>은 분석에 사용한 질의에 대한 설명이다.

<표 3> 배치 vs 스트림 프로세싱 질의 응답시간 비교

질의	설명	배치 응답시간	스트림 응답시간
count(*)	시간별로 웹 트래픽의 이벤트 발생 건수를 합한다.	1 시간 2 분	0.1 초
group by … having	소스 IP 별로 패킷량 합을 구한 후 그 합이 1500000 이상인 소스 IP 를 시간별로 나타낸다.	1 시간 7 분 54 초	31 초
top10	소스 IP 별로 트래픽 이벤트의 발생 건수를 기준으로 top10 을 구한고 시간별로 나타낸다	1 시간 10 분	1 초

(3) 실시간 분석

스트림 프로세싱은 배치 결과가 나오기 전까지 최근의 데이터에 대한 분석을 시스템을 제공한다. 최근 한 시간 동안의 데이터에 대한 groupby 질의는 25 초 정도 시간이 걸렸고, count 와 topN 의 경우 1 초의 시간의 걸렸다. 동일한 질의를 여러번 사용할 경우 캐시되어 3 초 내로 결과값이 리턴된다. 사용한 질의는 <표 3>와 동일하다.

5. 결론과 향후 연구 방향

스마트 기기의 발달과 소셜미디어의 발달로 인하여 데이터량이 증가 하였고, 이를 분석하기 위해서 배치 분석과 실시간 분석을 함께 제공하는 분석시스템에 대한 요구가 증가하는 추세이다. 이 두가지 방식을 함께 제공하기 위해 람다 아키텍쳐를 사용하여 시스템을 설계하고 구현한 결과 하이블를 통한 배치 분석과 스파크 스트리밍과 드루이드를 통한 실시간 분석이 가능한 것을 증명하였다. 하지만 두개의 프로세싱을 분리함으로 인하여 같은 분석 로직을 적용하는 데 있어서 데이터 파이프라인과 프로세싱용 어플리케이션을 따로 구현해야 하는 문제가 발생하였다. 앞으로 본 시스템의 복잡도를 줄이고 사용자 질의를 처리하는 서빙 레이어를 단순화 하는 설계를 도입하여 좀더 효율적으로 시스템을 관리할 수 있는 방향으로 연구 범위를 넓힐 계획이다.

참고문헌

[1] IDC, “Digital Universe of Opportunities”, 2014.04.11
[2] IMcKinsey, “Game changer”, 2013.07
[3] GIZMODO, “How much happens on the Internet Every 60 seconds”, 2013.07.29
[4] Lambda Architecture, https://en.wikipedia.org/wiki/Lambda_architecture
[5] Nathan Marz, ”What is the Lambda Architecture” http://lambda-architecture.net/
[6] Kafka, http://kafka.apache.org/
[7] Spark, https://spark.apache.org/
[8] Druid, http://druid.io/
[9] Camus, https://github.com/linkedin/camus
[10] Hive, http://hive.apache.org/