빅데이터 로그파일 처리 분석을 통한 성능 개선 방안

이재한, 유헌창
고려대학교 컴퓨터정보통신대학원
e-mail : {tkstka22, yuhc}@korea.ac.kr

Improving Performance based on Processing Analysis of Big data log file

Jaehan Lee, Heonchang Yu
Graduate School of Computer & Information Technology, Korea University

요 약

최근 빅데이터 분석을 위해 아파치 하둡(Apache Hadoop) 기반 에코시스템(Ecosystem)이 다양하게 활용되고 있다. 본 논문에서는 수집된 로그 데이터를 가공하여 데이터베이스에 로드하는 과정을 효율적으로 처리하기 위한 성능 평가를 수행한다. 이를 기반으로 텍스트 파일의 로그 데이터를 자바 코드로 개발된 프로그램에서 JDBC 를 이용하여 오라클(Oracle) 데이터베이스에 삽입(Insert) 하는 과정의 성능을 개선하기 위한 방안을 제안한다. 대용량 로그 파일의 효율적인 처리를 위해 하둡 에코시스템을 이용하여 처리 속도를 개선하고, 최근 인메모리(In-Memory) 처리 방식으로 빠른 처리 속도로 인해 각광받고 있는 아파치 스파크(Apache Spark)를 이용한 처리와의 성능 평가를 수행한다. 이 연구를 통해 최적의 로그데이터 처리 시스템의 구축 방안을 제안한다.

1. 서론

최근 수년 동안 스마트 기기 및 무선 인터넷의 발전으로 인해 미디어 이용 행태가 다양해지고 로그 데이터가 급격히 증가하여 빅데이터 분석을 위한 다양한 솔루션이 사용되고 있다. 하둡의 등장으로 기존의 Data Warehouse 시스템에 비해 적은 비용으로 스케일 아웃을 통한 효율적인 데이터 처리가 가능하게 되었다. 아파치 하둡은 다양한 서드파티 도구들이 출시되어 있으며 본 논문에서는 아파치 하이브(Apache Hive)와 아파치 스쿱(Apache Sqoop)을 사용하였다.

본 논문에서는 아파치 하둡 기반 시스템과 아파치 스파크 기반 시스템의 데이터 처리 성능을 비교평가하여 보다 효율적인 데이터 처리를 위한 시스템 구성 방안을 제안한다. 개선하고자 하는 처리 과정은 JDBC 를 사용하여 개발된 자바 프로그램을 사용하여 텍스트 형식의 로그 데이터를 읽어 오라클 데이터베이스에 저장하는 과정이다. 데이터 처리를 효율적으로 개선하기 위해 하둡 에코시스템을 사용하여 시스템을 구성하고 성능 분석을 진행하였다.

수집된 로그데이터를 HDFS 에 하이브 테이블 형식으로 저장하기 위해 스쿱을 사용하였으며, 저장된 테이블을 오라클 데이터베이스에 로드하기 위한 과정 또한 아파치 스쿱을 사용하여 구성하였다. 하둡 기반 빅데이터 처리 시스템 구축을 통해 기존 자바 코드로 만들어진 프로그램에 비해 상당한 성능 개선 효과를 확인할 수 있었다. 추가로 복잡한 하둡의 에코시스템의 구성없이 보다 간편하고 빠른 데이터 처리 시스템을 구축하기 위해 스파크를 사용하여 시스템 성능 평가를 진행하였다. 이를 통해 데이터 크기에 따른 시스템 간의 성능 차이를 확인할 수 있었다.

2. 관련연구

하둡(Hadoop)은 대용량 데이터를 처리할 수 있는 소프트웨어 프레임워크이다. 하둡은 병렬 프로세싱 처리를 위한 맵리듀스(MapReduce)와 분산파일시스템(HDFS)등으로 구성되어 있다. 하둡은 라이브러리 형태의 소프트웨어로 데이터 분석을 구조화된 RDBMS 형태로 가공하기 위해서는 직접 코드로 구현해야 하는 불편함이 있다. 이런 단점을 해결하기 위해 하둡을 기반으로 하는 다양한 서드파티 소프트웨어가 등장하였다.

하이브는 구조화된 데이터를 HDFS 에 저장하고 SQL 문법을 사용하여 데이터를 쉽게 조작할 수 있도록 하는 소프트웨어이다. HiveQL 이라고 불리는 SQL 언어를 지원하며 실행된 쿼리는 맵리듀스로 변환되어 처리된다.

스쿱은 대량의 데이터와 하둡과 효과적인 데이터 전송을 위해 만들어졌으며, 구조화된 데이터를 RDBMS 에 전송하기 위해서 사용되기도 한다.

스파크는 인메모리 처리 방식으로 하둡에 비해 빠른 속도로 최근 데이터 처리 시스템으로 각광받고 있다.

3. 실험 환경 및 성능평가

실험은 Amazon EC2 서비스를 이용하여 클라우드 환경에서 진행하였으며, OS는 Amazon Linux, 하드웨어는 CPU 1Core, Memory 2Gb, HDD 8Gb 환경에서 진행하였다. 실험에 사용된 소프트웨어 버전은 JDK 1.8.0, Hadoop 2.6.4, Spark 1.6.1, Hive 1.2.1, Sqoop 1.4.6 버전이 사용되었다.

진행된 실험은 두 가지 이며, 첫 번째 실험에서는 I/O 가 크게 발생하는 작업을 처리하는 경우에 대한 성능을 비교하였다. 두 번째 실험에서는 연산이 많은 작업 성능을 비교하기 위하여 SQL 을 사용하여 테이블을 조회하였다. 쿼리 조건은 where 절에서 서브쿼리를 사용하여 Like 처리하여 Join 결과를 비교하였다.

첫 번째 실험에 사용된 데이터의 크기는 일간 약 1.6 MB, 3 만 Rows 이며 일간, 주간, 월간 단위로 비교하였다. 실험에 사용된 데이터는 10 자 내외의 문자로 구성된 컬럼 7 개, 3 자리 이하 숫자로 구성된 컬럼 1 개, 총 8 개의 컬럼으로 구성된 테이블 형식의 데이터이며, 동일한 형식의 데이터 크기를 조절하여 자바, 하둡, 스파크 각 시스템에서 오라클 데이터베이스에 삽입 되는 시간을 측정한 결과를 비교하였다. 데이터 열의 수와 크기는 <표 1>과 같다. 데이터 수의 증가에 비례하여 데이터의 용량이 증가한다. 데이터 처리 시스템에 따른 처리 시간을 비교한 결과는 <표 2>와 같다.

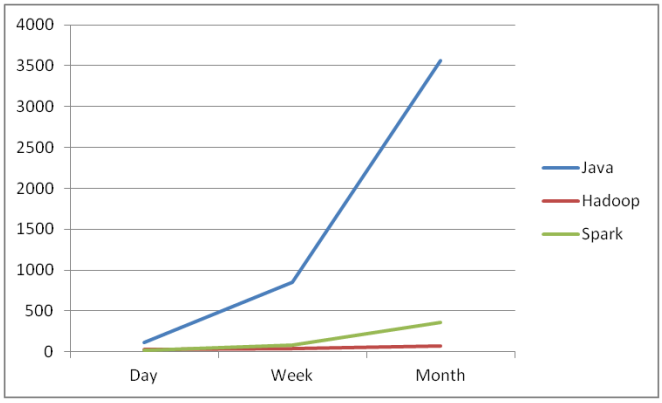
<표 1> 데이터 양에 따른 크기 변화

Type	Day	Week	Month
Volume(MB)	1.6802	9.9266	43.8682
Rows	34,875 개	205,133 개	906,473 개

<표 2> 처리 시스템에 따른 처리 시간 비교 단위(초)

System	Day	Week	Month
Java	117	857	3562
Hadoop	27.5526	36.4626	69.6552
Spark	23.9687	87.25357	362.0395

(그림 1)에 보이는 것과 같이 자바 시스템의 처리 속도는 데이터 크기에 비례하여 처리 속도가 증가되는 것이 확인되었다. 하둡 시스템의 일간 데이터 처리 속도는 자바 시스템 처리 속도에 비해 약 4.3배 빠르게 처리되었다. 월간 데이터 크기는 일간 데이터 크기에 비해 30배 정도 증가되었으나 월간 데이터 처리 속도는 일간 데이터 처리 속도에 비해 약 2배 가량만 증가되어 대용량 처리에 높은 성능이 확인되었다. 스파크 시스템의 일간 데이터 처리 속도는 자바 시스템 처리 속도에 비해 약 4.9배 빠르게 처리되었다. 월간 데이터 처리 속도 또한 자바 시스템에 비해 약 9.8배 빠른 속도로 처리되었다. 하지만 스파크 시스템은 데이터 크기가 증가될수록 하둡 시스템의 처리 속도에 비해 처리 속도가 느려지는 것으로 나타났다.



(그림 1) I/O 처리 시간 비교

다음으로 하둡 시스템에서 노드 수 증가에 따른 처리 성능 변화를 측정하였다. 하둡 노드 수 증가에 따른 처리 시간 측정 결과는 <표 3>과 같다.

<표 3> 노드 개수 별 데이터 처리 시간 단위(초)

	1-node	2-node	3-node
Hadoop	27.5526	23.9687	22.7489

분산처리 실험 결과 2-node 로 구성된 시스템의 데이터 처리 속도는 1-node 로 구성된 시스템의 데이터 처리 속도보다 약 3.6 초 단축된 것으로 측정되었다. 또한 3-node 로 구성된 시스템의 데이터 처리 속도는 2-node 로 구성된 시스템의 데이터 처리 속도보다 약 1.2 초 단축되는 것으로 측정되었다. 노드 수가 증가함에 따라 데이터 처리 속도가 개선되었지만 속도 차이가 크지 않고 노드 수가 증가될수록 속도 개선 효과의 폭이 줄어드는 것으로 나타났다. 노드 수 증가에 따른 속도 개선 효과가 크지 않은 원인은 데이터베이스 서버의 병목현상에 의한 것으로 예상된다.

두 번째 실험에 사용된 데이터는 URL 데이터이며 두 개의 테이블이 사용되었다. 두 개의 테이블을 Like 처리하여 Join 한 결과를 측정하였다. 데이터 처리에 사용된 쿼리는 <표 4>와 같다. 첫 번째 테이블에는 6,811 개의 URL 데이터가 있고 참조되는 데이터의 수는 <표 5>와 같다.

Spark SQL CLI 는 하이브의 Metastore 데이터를 조회할 수 있도록 해준다. 데이터 SQL 처리 성능 실험에서는 하둡에 하이브 테이블 형태로 저장된 동일한 데이터를 HiveQL 을 이용하여 조회한 결과와 Spark SQL 을 사용하여 조회한 결과를 비교하였다.

SQL 시스템에 따른 데이터 처리 시간을 비교한 결과는 <표 6>과 같다.

<표 4> 데이터 처리에 사용된 쿼리

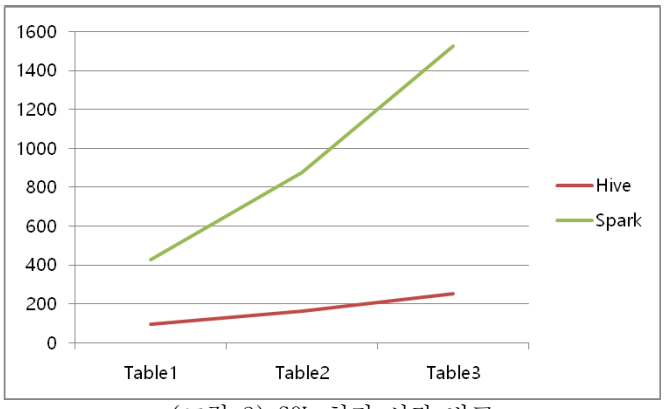
```
select count(*) cnt
from   table_0 a, table1_1 b
where  access_day = '20160905'
and    a.url like b.url
```

<표 5> Like 처리 시 참조되는 데이터 수

	Table1	Table2	Table3
Rows	1,600 개	3,291 개	5,781 개

<표 6> SQL 시스템에 따른 처리 시간 비교 단위(초)

	Table1	Table2	Table3
Hive	95.188	161.739	251.964
Spark	426.829	873.91	1523.568



(그림 2) SQL 처리 시간 비교

(그림 2)에서 보이는 것과 같이 동일한 데이터를 하이브 테이블에 저장하고 조회하는 경우 HiveQL을 이용하여 조회하는 방식에서 Spark SQL을 사용하여 조회하는 방식보다 빠른 속도로 처리되는 것을 확인할 수 있었다. 하이브 시스템에서 데이터를 조회하는 경우 참조되는 URL 수가 증가함에 따라 처리 속도가 완만하게 증가하는 반면, 스파크 시스템에서 데이터를 조회하는 경우 참조되는 URL 수가 커질수록 속도가 저하되는 결과가 나타났다.

4. 결론 및 향후 연구 과제

데이터 처리 시스템에 따른 성능 평가를 통해 기존의 자바 프로그램으로 데이터를 처리하는 방식에 비해 하둡, 스파크와 같은 빅데이터 처리 시스템을 사용하는 경우 약 4.3배 이상 빠른 성능 개선 효과를 확인하였다.

실험 전 스파크 시스템은 인메모리 처리 방식으로 빠른 처리 성능을 예상하였다. 데이터 처리 성능 평가 실험 결과 크기가 작은 데이터에서는 하둡 시스템과 비교하여 빠른 성능을 보였지만 데이터 크기가 늘어날수록 처리 속도가 저하되는 현상이 확인되었다. 하둡 시스템에서 노드를 추가하여 로그 데이터를 오라클 데이터베이스에 삽입하는 실험 결과에서는 노드 수 증가에 따른 속도 개선 효과가 크지 않았다. 따라서 I/O 가 크게 발생하는 작업의 처리 시스템은 노드 수를 무리하게 늘리지 않도록 시스템을 구축하는 것이 비용적인 측면에서 효율적일 것이다. 하이브 테이블에 저장된 데이터를 조회하는 경우 HiveQL을 사용하는 방법이 Spark SQL을 사용하여 조회하는 방법 보다 빠른 속도로 동작하는 것이 확인되었다.

본 논문에서 제안하는 데이터 처리 시스템 구축 방안은 처리되는 데이터의 크기를 고려하여 배치 처리와 같이 데이터 양이 많은 작업은 하둡 기반 시스템에서 처리하고 데이터 양이 적지만 빠르게 처리되어야 하는 실시간 데이터 분석 처리는 스파크 시스템을 구축하여 처리하는 방식의 시스템 구성 방안을 제안한다. 또한 하둡 시스템에 하이브 테이블 형태로 저장되어 있는 데이터는 별도로 스파크에서 처리하지 않고 하이브에서 처리하는 방식으로 시스템을 구축하여 데이터 처리에 소요되는 시간을 단축할 수 있을 것으로 예상한다.

향후 연구 과제는 스파크 시스템에서 노드 수 증가에 따른 처리 속도에 변화를 평가하고 데이터 크기 및 유형별 시스템 구축 방안을 연구하는 것이다. 또한 스파크 시스템에서 대용량 파일을 처리하는 경우 속도 저하 원인 분석 및 처리 속도 향상을 위한 튜닝 기법을 연구하는 것이다.

참고문헌

[1] Apache Hadoop. http://hadoop.apache.org/
[2] Tom White. “Hadoop: The Definitive Guide, Oreilly”, 2012
[3] Apache Hive. https://hive.apache.org/
[4] Edward Capriolo, Dean Wampler, Jason Rutherglen. “Programming Hive, Oreilly”, 2012
[5] Apache Sqoop. http://sqoop.apache.org/
[6] Apache Spark. http://spark.apache.org/
[7] Mohammed Guller. “Big Data Analytics with Spark, Apress”, 2016