KIPS Tr. Software and Data Eng.
Vol.4, No.2 pp.77~82 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.2.77

Analysis of the Influence Factors of Data Loading Performance Using Apache Sqoop

Liu Chen† · Junghyun Ko†† · Jeongmo Yeo†††

ABSTRACT

Big Data technology has been attracted much attention in aspect of fast data processing. Research of practicing Big Data technology is also ongoing to process large-scale structured data much faster in Relatioinal Database(RDB). Although there are lots of studies about measuring analyzing performance, studies about structured data loading performance, prior step of analyzing, is very rare. Thus, in this study, structured data in RDB is tested the performance that loads distributed processing platform Hadoop using Apache sqoop. Also in order to analyze the influence factors of data loading, it is tested repeatedly with different options of data loading and compared with data loading performance among RDB based servers. Although data loading performance of Apache Sqoop in test environment was low, but in large-scale Hadoop cluster environment we can expect much better performance because of getting more hardware resources. It is expected to be based on study improving data loading performance and whole steps of performance analyzing structured data in Hadoop Platform.

Keywords : Apache Sqoop, Hadoop, Relational Database, Data Loading Performance

아파치 스쿱을 사용한 하둡의 데이터 적재 성능 영향 요인 분석

Liu Chen† · 고 정 현†† · 여 정 모†††

요 약

빅데이터 기술은 데이터 처리 속도가 빠르다는 면에서 주목을 받고 있다. 그리고 관계형 데이터베이스(Relational Database: RDB)에 저장되어있는 대용량 정형 데이터를 더 빠르게 처리하기 위해서 빅데이터 기술을 활용하는 연구도 진행되고 있다. 다양한 분산 처리 도구들을 사용하여 분석 성능을 측정하는 연구는 많지만 분석하기 전 단계인 정형 데이터 적재의 성능에 관한 연구는 미미하다. 때문에 본 연구에서는 RDB 안에 저장되어있는 정형 데이터를 아파치 스쿱(Apache Sqoop)을 사용하여 분산 처리 플랫폼 하둡(Hadoop)으로 적재하는 성능을 측정하였다. 그리고 적재에 영향을 미치는 요인을 분석하기 위해 여러 가지 영향 요소를 변경해가면서 반복적으로 실험을 수행하였고 RDB 기반으로 구성된 서버 간의 적재 성능과 비교하였다. 실험 환경에서 아파치 스쿱의 적재 속도가 낮았지만 실제 운영하고 있는 대규모 하둡 클러스터 환경에서는 더 많은 하드웨어 자원이 확보되기 때문에 훨씬 더 좋은 성능을 기대할 수 있다. 이는 향후 진행할 적재 성능 개선 및 하둡 환경에서 정형 데이터를 분석하는 전체적인 단계의 성능을 향상시킬 수 있는 방법에 대한 연구의 기반이 될 것으로 예상한다.

키워드 : 아파치 스쿱, 하둡, 관계형 데이터베이스, 데이터 적재 성능

1. 서 론

RDB는 지난 수십 년 동안 다양한 형태의 데이터베이스를 제치고 수많은 산업 분야에서 데이터를 저장, 관리하는데 사용되어왔다. 그러나 최근 데이터의 폭증으로 인해 대용량 데이터를 저장, 분석, 처리하는 데 한계를 느끼기 시작하면서 빅데이터가 주목받기 시작했고 관련된 다양한 기술들이 등장하였다. 다양한 기술들 중 하둡은 대용량 데이터의 분석 및 처리의 한계점을 넘을 수 있는 가능성을 보였고 빅데이터 처리 표준 기술의 중심축을 담당하게 되었다. 그러나 하둡을 포함한 여러 기술을 적용하고자 할 때는 많은 어려움과 다양한 기술적 문제가 발생하고 있다. RDB상의 데이터를 하둡 플랫폼[1]으로 효율적으로 가져오기 위한 문제, 하둡 플랫폼에서 SQL을 사용하지 못하는 문제 등이 그 예이다. RDB환경에서 저장되어있는 데이터들을 하둡 환경에서 분석 작업을 수행하기 전에 먼저 데이터를 하둡 환경으로 적재하는 단계가 반드시 필요한데, 만약 데이터를 적재하는 자동화된 툴이 없다면 적재 작업은 상당히 복잡하고 어려울뿐더러 적재할 때 아주 많은 시간을 소모할 수도 있다. 이러한 문제를 해결하기 위해 효율적으로 데이터를 전송하는 툴인 스쿱(Sqoop)[2]이 개발되었다. 스쿱의 개발로 인해 RDB와 하둡과의 데이터전송을 편리하게 할 수 있게 되었고 상호 간의 데이터 전송도구로서 가장 많이 사용되고 있다. 그러나 스쿱을 사용할 때도 실제 사용환경 및 다양한 스쿱의 옵션을 적절하게 사용하여야 올바른 성능을 발휘할 수 있지만 올바른 적재 성능을 내기 위해 어떠한 요인들을 고려해야

※ 이 논문은 부경대학교 자율창의학술연구비(2014년)에 의하여 연구되었음.
† 준 회 원 : 부경대학교 컴퓨터공학과 박사과정
†† 준 회 원 : 부경대학교 컴퓨터공학과 석사과정
††† 정 회 원 : 부경대학교 컴퓨터공학과 교수
Manuscript Received : September 16, 2014
First Revision : October 30, 2014
Accepted : November 26, 2014
* Corresponding Author : Jeongmo Yeo(yeo@pknu.ac.kr)

하는지에 대한 연구와, 스쿱을 사용하여 실제 전송 및 적재 성능을 측정한 연구는 매우 드물다. 스쿱에서도 단지 간단한 사용법만을 매뉴얼로 제시하고 있을 뿐이다. 그렇기 때문에 본 연구에서 스쿱을 사용하여 RDB에 저장된 정형 데이터를 하둡으로 적재하는 성능을 측정하고 적재 성능에 영향을 미치는 요인에 대해 연구하고자 한다.

2. 관련 연구

2.1 하둡 플랫폼

하둡은 수많은 컴퓨터들을 네트워크로 연결한 후 데이터를 분산시켜 처리하는 오픈 소스 프레임워크이다. 단일 서버를 사용하여 대용량 데이터를 처리할 때의 성능 한계를 넘어서고자 최근에 큰 주목을 받고 있다. 하둡은 주로 검색엔진 색인저장소(Indexing), 데이터 분석 또는 통계분석(Data Analytics/ Statistical Analytics), 데이터의 전처리(Table Precomputaion and Rollup), 또는 정형 데이터의 저장소(Structured Data Storage)의 4가지 분야에서 많이 활용되고 있다. 하둡이 이와 같이 다양한 분야에 활용되고 빅데이터 처리에 적합하다고 하는 이유는 다음과 같다. 첫째, 하둡은 대용량 비정형 데이터를 효율적으로 저장할 수 있는 HDFS(Hadoop Distributed File System)를 제공한다. 둘째, 기존 단일 서버의 성능 부하 집중 문제를 해결하기 위해 여러 개 서버로 구성된 클러스터를 통해 부하를 분산시킨다. 또한 맵리듀스(MapReduce)를 통해서 개별적인 서버에서 진행되는 병렬처리 결과를 하나로 묶어 시스템의 과부하나 병목 현상을 줄였다. 셋째, 하둡은 여러 개 서버로 구성된 클러스터이기 때문에 하드웨어 장비를 추가시킴으로써 구성된 서버의 자원을 최대한 활용하여 성능이 선형적으로 향상된다. 넷째, 값이 비싼 서버를 요구하지 않고 다양한 오픈 소스 소프트웨어를 무료로 사용할 수 있기 때문에 누구나 쉽게 하둡 클러스터를 구성하여 정형이나 비정형 데이터를 처리할 수 있다[3].

하둡의 가장 큰 장점은 대용량 데이터의 분산 저장과 다수의 서버 클러스터에서 진행하는 병렬처리 기능이다. Fig. 1[4]과 같이 하둡은 메타정보를 저장하고 있는 NN(Name Node) 한 대 및 데이터를 저장하고 처리하는 SN(Slave Node) 여러 대로 구성되어있다. 이는 NN와 SN로 분리되어 동작한다. 여러 요소로 인해 SN에 장애가 발생하였으면 새로운 SN를 추가하기만 하면 기존의 시스템을 유지시켜주는 안정적 설계로 되어있다.

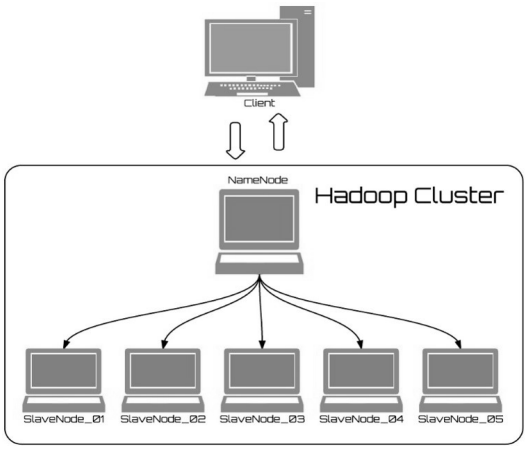

Fig. 1. Conceptual Structure of Hadoop Cluster

2.2 분산 파일 시스템의 복제정책(Replication Policy)

분산 시스템은 서로 분산된 여러 서버들을 가지고 파일 시스템을 구성함으로써 높은 확장성과 고가용성을 지원한다. 원본 데이터를 여러 개 복제본으로 서로 다른 스토리지에 저장함으로써 시스템 장애가 발생하더라도 지속적으로 서비스를 제공할 수 있다[5-6].

분산 파일 시스템 안에서 데이터들을 어떻게 저장하였는지, 복제정책을 어떻게 설정하였는지에 따라서 분산 처리성능의 차이가 발생한다. 예를 들어, 처리하고자 하는 데이터들이 모두 하나의 서버에 집중되어 저장되어있으면 분산 처리의 효과를 전혀 얻을 수 없다. 이러한 측면을 방지하기 위하여 일반적으로 분산 처리 플랫폼 내부적으로 분산 알고리즘에 따라서 데이터를 분산시킨다. 사용자들이 복제정책 설정 옵션을 통해서 데이터의 복제본 수량을 제어할 수도 있다. 복제본 수량이 많아지고 데이터를 여러 서버에 복제하여 분산 저장하고 있다면 분산 처리할 때도 좋은 성능을 얻을 수 있다.

분산 시스템에서 복제정책을 적용하여 많은 이점을 가져올 수 있는 반면에 모든 데이터에 대하여 블록 단위로 여러 개 복제본을 유지하기 때문에 저장 공간이 추가적으로 필요하다. 예를 들어, 복제 인수를 3으로 설정한다면 원본 데이터에 추가적으로 복제본 2부를 저장해야 한다. 앞서 언급한 바와 같이 여러 복제본을 만들어 분산하여 저장한다면 분산처리의 성능상 이점을 얻을 수 있고 가용성을 높일 수 있지만 반면 저장 공간이 늘어난다. 그렇기 때문에 사용자가 하둡 클러스터 환경을 고려하여 적절한 복제정책을 세워 운영할 필요가 있다.

2.3 아파치 스쿱

아파치 스쿱은 맵리듀스[7]를 기반으로 구현된 데이터 적재 프로그램이다. 특히 RDB 및 HDFS 사이에 데이터 적재가 가능하기 때문에 많은 프로젝트에서 널리 사용하고 있다. 스쿱은 모든 적재 과정을 자동화하며 병렬처리 방식으로 작업하며, 좋은 내고장성(Fault tolerance)을 지원한다[8-9].

스쿱은 Row-by-row 방식으로 RDB에 저장되어있는 테이블을 읽고 HDFS에 저장한다. RDB에서 읽어온 테이블 하나를 HDFS에서 파일셋으로 저장한다. 병렬처리 방식으로 적재하기 때문에 적재한 후에 HDFS에서 여러 개 파일으로 저장하게 된다. 스쿱을 사용하여 HDFS에 저장되어있는 파일셋을 읽고 RDB로 적재하는 Export과정도 가능하다. 파일셋을 병렬처리 방식으로 읽고 레코드 형태로 변환하여 RDB의 해당 테이블에 삽입한다[10-11].

스쿱이 RDB의 데이터를 HDFS로 적재하는 절차는 Fig. 2[12]와 같다.

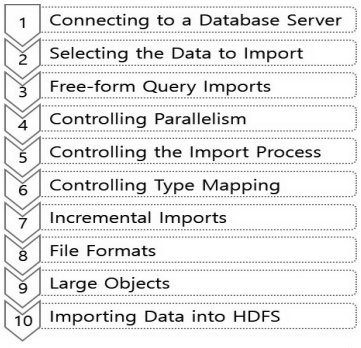

Fig. 2. Data loading process of Apache Sqoop

3. 아파치 스쿱을 사용한 정형 데이터 적재 성능 테스트

RDB 기반 업무시스템에서 수집된 데이터들을 분석하기 위하여 일반적으로 DW(Data Warehouse)로 적재하여 진행한다. 기존에 사용하는 DW도 RDB로 구축된 경우가 대부분이다. 그렇기 때문에 본 연구에서는 스쿱의 적재 성능을 비교하기 위하여 RDB에서 RDB로 적재하는 성능과 RDB에서 하둡으로 적재하는 성능을 대비하여 진행하였다. 테스트 절차는 다음과 같다.

① RDB서버 및 RDB서버 간의 적재 성능 테스트
② 아파치 스쿱을 사용한 RDB서버에서 하둡클러스터로의 적재 성능 테스트
③ 하둡 SN수량 증가에 따른 스쿱 적재 테스트 반복 수행

3.1 테스트 방법 및 환경

앞서 언급한 데이터 적재 테스트를 진행하기 위해 RDB서버 두 대와 하둡 클러스터를 구축하였다. 하둡 클러스터는 1대의 NN와 3대의 SN로 구성하였고, 점차 SN를 1대씩 추가해가면서 5대가 될 때까지 반복적으로 구성한 뒤 테스트하였다. 각 서버의 사양은 Table 1과 같다.

Table 1. Configuration of server computer

	RDB	Hadoop Node
CPU	(Intel i5) 4 Core	
RAM	4GB	2GB
HDD	2TB	500G
OS	Windows 7 64bit	CentOS 6.4
SW	ORACLE 11g r2	Hadoop 2.0.5-alpha Tajo 0.2.0 Sqoop 1.4.4

먼저 적재 테스트를 진행하기 전에 RDB서버에서 각각 다른 유형의 4개의 테이블을 이용하여 1TB의 테스트 데이터를 확보한 뒤, 각 테이블을 대상으로 테스트를 진행한다. 테이블의 상세정보는 Table 2와 같다.

Table 2. Information of tables

Table Name	Size(GB)	Records
Big_orders	103.5668	1,677,721,600
Big_employees	146.359	1,677,721,600
Big_order_items	256.5738	11,156,848,640
Big_customers	569.6716	5,033,164,800

RDB서버 간의 CTAS(Create Table As Select) 방식으로 적재 시 사용하는 SQL문장의 유형은 Table 3과 같다.

Table 3. SQL query for data loading by CTAS

```
CREATE TABLE [Target Table Name] AS
SELECT [Column Name List]
  FROM [Source Table Name];
```

3.2 RDB서버 간의 정형 데이터 적재 성능 테스트

RDB서버에 저장되어있는 테이블들을 오라클이 제공해주는 CTAS, DATAPUMP 방식으로 대상 RDB서버에 각각 한 번씩 적재하였고, 적재 시 Parallel Degree 옵션은 4로 지정하였다.

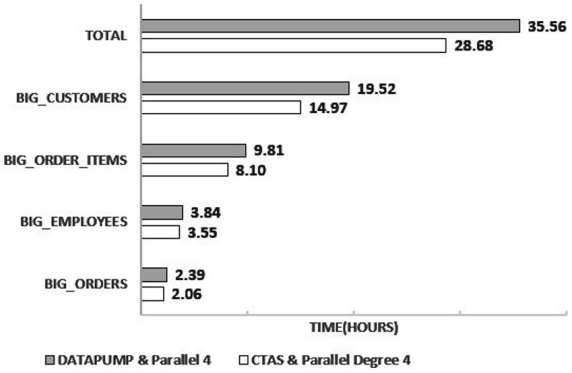

Fig. 3. Result of data loading among RDB servers

Fig. 3은 각 테이블의 CTAS, DATAPUMP를 사용한 적재 시간 결과이다. CTAS 방식으로 적재할 때 DATAPUMP 방식보다 약 19% 정도 성능 향상을 보였다.

3.3 적재 성능 테스트 및 성능 영향 요인 분석

하둡 환경 구성은 RDB보다 복잡도가 높고 아파치 스쿱을 사용하여 적재할 때 하둡 클러스터의 서버 사양 및 다양한 옵션들을 고려하여 적절하게 설정하여야 성능을 제대로 발휘할 수 있다. 스쿱에서 지원하는 설정 중 적재 방법과 환경 구성 두 가지 측면에서 성능 영향 요인을 분석하였다.

1) 직접 경로 쓰기(Direct Path Write)

RDB 환경에서 데이터를 적재할 때 흔히 사용하는 CTAS 방법은 메모리를 경유하지 않고 디스크에 바로 쓰는 DMA(Direct Memory Access) 방식이다. 스쿱에서도 Direct 옵션을 제공하고 있다. Direct옵션의 효과를 확인하기 위해 Direct 옵션을 적용하여 적재 테스트를 수행하였다.

Fig. 4. Result of data loading with direct path write option

Fig. 4에서 나타난 것과 같이 본 실험에서는 Direct 옵션을 적용하기 전과 적용한 후 성능 차이가 없었다. Direct 옵션이 적재 테스트를 진행하는 과정에서 효과를 발휘하지 못한 이유는 스쿱은 MySQL, PostgreSQL DBMS에서만 Direct 옵션을 사용할 수 있도록 하였기 때문에 실험에서 사용한 오라클에서는 효과를 볼 수 없었다.

2) Mapper옵션

스쿱을 사용하여 RDB에 있는 데이터를 하둡으로 적재할 때 HDFS의 모든 저장 현황을 파악하고 있는 하둡 NN이 내부적 알고리즘에 의해서 SN에 있는 Mapper에게 적재 작업을 할당해준다. Mapper가 NN의 지시에 따라서 데이터를 지정된 위치로 적재하는 작업을 수행한다. 하나의 SN에서 여러 개의 Mapper를 생성하여 동시에 작업할 수 있기 때문에 RDB의 Parallel Degree 옵션과 비슷하다고 생각할 수 있다. Parallel Degree는 CPU의 개수 및 데이터 양에 따라서 적절하게 지정해주어야 되는 것이고 Mapper수량은 Node수량 및 각 서버의 하드웨어의 성능에 따라서 적절하게 지정해주어야 한다. 즉 Mapper옵션은 SN수량과 밀접한 관계가 있다.

그렇기 때문에 Mapper수량에 따른 성능 변화를 측정하기 위해 다음과 같은 두 가지 테스트를 진행하였다. 먼저 SN수량을 동일하게 한 뒤 Mapper수량을 늘려가면서 적재 시간을 측정하고, 그다음 Mapper수량을 동일하게 한 뒤 SN수량을 늘려가면서 적재 시간을 측정하였다.

● Node수량이 동일한 경우에 Mapper수량에 따른 성능 차이 : 5개 SN로 구성된 환경에서 Mapper를 추가하여 테스트를 진행하였다.

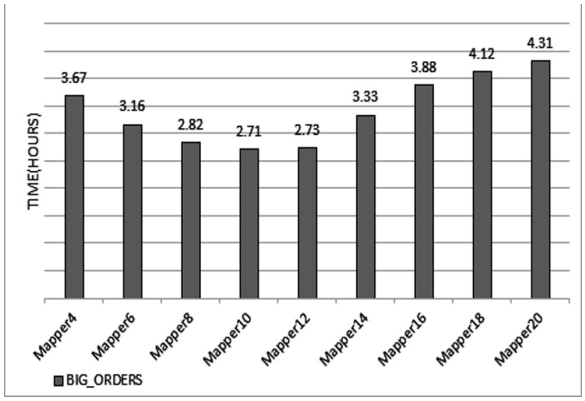

Fig. 5. Result of data loading with 5 SN and different mapper

Fig. 5에서와 같이 Mapper수량을 증가시키면서 테스트한 결과 적재 성능이 점차 향상되었다. Fig. 5에서 Mapper수량이 4에서 10인 사이의 결과에서 볼 수 있듯이 Mapper수량을 늘린다면 적재 성능을 향상시킬 수 있다는 결론을 얻을 수 있다. 그렇지만 Mapper는 병렬처리와 유사하게 동작하기 때문에 하드웨어 사양을 고려해야 한다. 그렇지 않다면 Mapper수량이 12 이상인 결과에서와 같이 오히려 더 낮은 성능이 나타날 수 있다. Mapper수량을 한계치 이상으로 설정한다면 작업이 먼저 끝난 Mapper가 더 이상 작업을 하지 않고 하드웨어 자원만 차지하고 있기 때문이다. Mapper수량이 10으로 설정되었을 때 4일 때보다 약 26% 향상되었고 12부터 점차 감소하기 시작하여 20으로 변경한 후에는 적재 시간이 약 2배 정도 많은 시간이 걸렸다. 때문에 Mapper수량을 설정할 때 하드웨어의 성능을 고려하여 적절하게 설정해주어야만 좋은 성능을 얻을 수 있다는 결론을 도출할 수 있다.

● Mapper수량이 동일한 경우 Node수량에 따른 성능 차이 : 하둡은 분산 처리 환경에서 SN를 적은 비용으로 비교적 자유롭게 하드웨어 자원을 추가할 수 있다는 장점이 있다. 그렇기 때문에 Node수량은 적재의 성능을 향상시킬 수 있는 요소가 될 수 있다.

이 단계는 Mapper수량을 동일하게 4로 설정해둔 상태에서 SN수량을 3대, 4대, 5대로 늘려가며 테스트를 수행하여 적재 시간을 측정하였다.

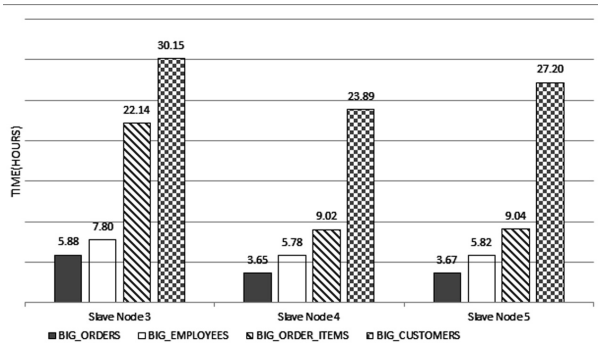

Fig. 6. Result of data loading with 4 mapper and different SN

Fig. 6에서 SN수량이 3대 및 4대일 경우에 적재 시간을 비교해보면 성능이 향상되었다는 것을 확인할 수 있다. 그렇지만 SN수량이 4대 및 5대일 경우에는 적재 성능에 거의 변화가 없었다.

SN수량이 3대 및 4대일 경우에 적재 성능이 향상된 이유는 SN수량이 늘어나면서 Mapper가 활용할 수 있는 자원이 늘어났기 때문이다. SN수량이 3대일 경우 Mapper가 사용할 수 있는 모든 자원을 활용하였음에도 불구하고 Mapper가 활용할 수 있는 자원의 최대치에 도달하지 못했다. 즉 4개의 Mapper의 성능을 모두 발휘하기에는 3대의 SN으로는 부족했다는 뜻이다. SN수량을 4로 변경하였을 때 활용 자원이 늘어남으로써 4개의 Mapper가 더 많은 자원을 사용하였기 때문에 처리 성능이 향상되었다. 반대로 SN의 수량이 4에서 5로 변경되었을 때는 각 Mapper가 활용할 수 있는 자원의 최대치보다 더 많은 시스템 자원이 남아있기 때문에 4개의 Mapper가 남은 자원을 활용하지 못해 성능이 향상되지 않았다.

이 테스트 결과의 요지는 Mapper가 활용할 수 있는 자원의 최대치를 넘는 자원이 있을 때, 즉 지정한 Mapper의 수

량이 처리할 수 있는 것보다 많은 노드에서 가용자원이 남아있을 때, 더 이상의 Mapper를 늘리지 않고 단지 Node수량을 증가시키는 것으로는 더 나은 성능을 얻을 수 없다는 것이다. 반드시 SN수량에 맞추어서 적절하게 Mapper수량을 설정해야 좋은 성능을 얻을 수 있다.

3) HDFS의 복제정책

하둡은 가용성(Availability)을 보증하기 위해 동일한 데이터를 정해진 분산 알고리즘에 따라서 여러 부 복제본을 생성하여 저장한다. 또한 데이터를 분석할 때 동일한 데이터가 분산되어 저장되어있다면 하드웨어의 성능을 최대한 충분히 활용해서 분산되어있는 데이터를 처리할 수 있기 때문에 분석 성능을 향상시킬 수 있다. HDFS에서 복제본의 개수를 설정하는 옵션을 Replication라고 한다. 하둡 클러스터에서 이 옵션의 기본값은 3으로 설정되어있다. 즉, Replication 값을 3으로 설정하였을 때 1TB의 데이터가 실제 디스크에 저장된다면 데이터의 용량이 3TB가 된다.

지금까지 적재 성능 테스트를 수행하였을 때 RDB에 저장되어있는 데이터를 동일하게 옮기기 위해서 복제본의 수를 1로 변경하였는데 실제 운영환경에서는 하둡의 복제본의 수를 1로 설정해 사용하는 경우는 테스트 환경을 제외하곤 극히 드물다. 하둡 클러스터의 Node수량 및 데이터를 분석할 때의 올바른 성능을 고려하면 최소한 기본값 이상으로 설정하여야 한다. 그렇기 때문에 본 논문에서 데이터를 분석 처리하는 성능을 고려하여 복제본의 수를 디폴트값으로 설정해 여러 번의 테스트를 추가로 수행하였다. 즉, 복제본의 수를 3으로 설정하고 RDB에서 가져온 1TB 데이터를 복제정책을 적용하여 HDFS에서 3배까지 복제해서 저장하는 테스트를 수행하였다. 그리고 수행할 때 앞서 언급한 Mapper 수량도 적절하게 설정하였다.

Table 4에서는 Node수량 및 Mapper수량의 변화에 따라서 Replication값이 1 또는 3으로 설정된 경우의 데이터 적재 시간을 보여준다. 전체적으로 비교해봤을 때 복제정책을 적용한 후에 데이터 양이 3배 정도 늘어났기 때문에 각 테이블의 적재 시간도 30~50% 정도 늘어날 수밖에 없다. 그러나 적재 시간이 조금 더 소요되더라도 실제 운영환경에서는 가용성 및 적재 데이터에 대해 수행할 분석 처리 작업의 성능을 고려해서 일반적으로 복제본의 수량을 3 이상으로 설정한다. 특히 복제본의 수량이 많으면 데이터 분석의 효율성을 향상시킬 수 있기 때문에 RDB에서 저장된 대용량 정형 데이터에 대해서 하둡 플랫폼으로 한 번 적재하여 여러 번 분석 작업을 수행하는 경우에 적재 성능이 조금 낮더라도 복제정책을 필히 적절하게 적용하여 사용해야 한다.

3.4 적재 성능 영향 요인을 고려하여 적절하게 설정한 후의 적재 성능 비교

현재 환경에서 앞서 분석된 적재 성능 영향 요인을 적합하게 설정한 후에 적재 테스트를 수행한 결과를 RDB서버간의 적재 시간과 비교해보았다.

Fig. 7에서와 같이 CTAS 방식에서 가장 좋은 적재 성능이 나타났다. 또한 이는 5대 SN으로 구성된 하둡 환경에서 Mapper수량을 10개로 설정하였을 때의 적재 성능보다 평균

Table 4. Result of data loading and the effect of replication policy

	Node	Mapper	Replication	Time(H)	Comparision
BIG_ORDERS	3	6	1	4.15	44.96%
	3	6	3	6.02	
	4	8	1	3.50	88.35%
	4	8	3	6.59	
	5	10	1	2.71	30.33%
	5	10	3	3.53	
BIG_EMPLOYEES	3	6	1	7.21	46.05%
	3	6	3	10.53	
	4	8	1	4.90	48.16%
	4	8	3	7.26	
	5	10	1	4.80	37.29%
	5	10	3	6.59	
BIG_ORDER_ITEMS	3	6	1	11.13	49.43%
	3	6	3	16.63	
	4	8	1	7.11	49.57%
	4	8	3	10.63	
	5	10	1	9.34	8.01%
	5	10	3	10.08	
BIG_CUSTOMERS	3	6	1	22.00	49.39%
	3	6	3	32.87	
	4	8	1	20.11	45.80%
	4	8	3	29.32	
	5	10	1	18.77	35.32%
	5	10	3	25.4	

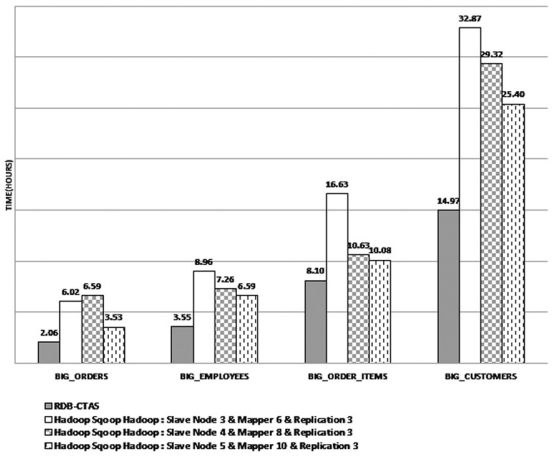

Fig. 7. Result of data loading after improved negative factors

30% 정도 빠른 것도 확인할 수 있다. 그리고 하둡 클러스터의 SN수량 및 Mapper수량이 늘어나면서 성능이 점차 향상되는 결과를 볼 수 있다.

앞서 분석한 요인들을 고려하고 복제정책을 사용한 후 종합적인 성능을 비교한 결과를 볼 때 적재할 데이터의 양이 적은 RDB의 적재 성능이 더 빠르다고 볼 수 있다. 그렇지만 SN수량을 증가시키고 그에 맞게 Mapper수량을 늘림에 따라서 적재 성능이 향상되는 것을 보였기 때문에 실제 운영환경에서 추가적으로 더 많은 SN을 사용한다면 스쿱의

적재 성능이 많이 향상되어 복제정책으로 인해 더 많은 데이터를 적재함에도 불구하고 RDB와 비슷한 성능을 보이거나 그보다 나은 성능을 보일 것으로 예상한다.

4. 결 론

수십 년 동안 사용해온 RDB 기반의 데이터베이스 안에는 방대한 양의 데이터가 쌓여있다. 이제 그 방대한 데이터들에 대해서 분석하여 그 가치를 최대한 활용하려고 하는데 기존에 있는 RDB 기반 기술들로는 처리하지 못하는 한계점을 만나게 되었다. 빅데이터 기술은 이러한 정형 데이터를 더 빨리 처리할 수 있다는 점에서 많은 기대를 받고 있다. 그리고 분석하기 전 단계에서 기존에 RDB에 저장되어있는 정형 데이터들을 빅데이터 플랫폼으로 이전하여야 하기 때문에 효율적으로 이전하기 위한 도구인 아파치 스쿱이라는 툴도 개발되었다.

스쿱이 출시된 후에 다양한 분야에서 많이 활용되고 있는데 아직 이와 관련된 연구는 미미하기 때문에 본 연구에서 여러 가지 테스트를 진행함으로써 스쿱을 사용하여 RDB에 있는 데이터를 HDFS로 적재하는 성능을 측정하였고, 성능에 영향을 미치는 요인에 대해 분석하였다.

본 연구에서 여러 절차의 테스트를 통해서 직접 경로 쓰기, Mapper수량, Node수량, 복제정책 등 요인들이 적재 성능에 미치는 영향을 보여주었다. 그리고 이런 성능 영향 요인들을 적절하게 조절하면 성능 향상을 보일 수 있다. 최종적인 테스트 결과로 RDB 서버 사이에서 데이터를 적재하는 성능보다 더 나은 결과는 나타나지 않았다. 그 이유는 테스트를 진행할 때 실제 운영환경과 분석단계의 성능을 고려해서 복제정책을 적용하였기 때문이다. 그러나 실제 운영환경에서는 하둡을 구성하는 서버 수량이 훨씬 많고 하드웨어 사양도 뛰어나기 때문에 보다 훨씬 좋은 효율을 기대할 수 있다. 또한 본 논문에서 제시한 요인들을 고려해서 적절하게 설정하면 적재 시간을 단축함으로써 RDB에 저장되어있는 데이터에 대해 빅데이터 기술을 활용해 분석하여 결과 도출하는 전체 과정의 시간도 단축할 수 있다.

향후에 본 연구를 토대로 다양한 빅데이터 분석 프로그램과 결합해서 전체 단계의 RDB 정형 데이터를 빅데이터 플랫폼에서 처리하는 연구를 진행할 예정이다.

References

[1] Apache Hadoop [Internet], http://hadoop.apache.org

[2] Apache Sqoop [Internet], http://sqoop.apache.org

[3] Lee Hyunjong, “Use of Big Data Hadoop Platform,” *in Journal of Communications and Networks,* Vol.29, No.11, 2012.

[4] Shvachko, K., Hairong Kuang, Radia, S., and Chansler, R., “The Hadoop Distributed File System,” in Mass Storage Systems and Technologies (MSST), *2010 IEEE 26th Symposium on,* Mar., 2010.

[5] HooYoung Ahn, KyongHa Lee, SooHo Lee, YoonJoon Lee, SangMin Lee, and YoungKyun Kim, “An Efficient Method for Enhancing the Storage Efficiency in Hadoop DFS,” *in Journal of KISS : computing practices,* Vol.19, No.3, 2013.

[6] Dae Soon Choi, Jeehong Kim, and Young Ik Eom, “Analyses of Replica Placement Schemes in Distributed File Systems,” *in Journal of Computing Science and Engineering,* Vol.39, No.1A, 2012.

[7] Tom White, “Hadoop: The Definitive Guide, Third Edition,” *O'Reilly/Yahoo Press,* 2012.

[8] Kathleen Ting, Jarek Jarcec Cecho, “Apache Sqoop Cookbook,” *O'Reilly,* 2013.

[9] Rinusha Irudeen, Sanjeeva Samaraweera, “Big data solution for Sri Lankan development: A case study from travel and tourism,” in Advances in ICT for Emerging Regions, 2013 International Conference on.

[10] Nodar Momtselidze, Alex Kuksin “Hadoop Integrating with Oracle Data Warehouse and Data Mining,” *in Journal of Technical Science and Technologies,* Vol.2, No.1, 2013.

[11] Ankit Jain, “Instant Apache Sqoop,” *Packt Publishing Ltd,* 2013.

[12] Ognjen V. Joldzic, Dijana R. Vukovic, “The Impact of Cluster Characteristics on HiveQL Query Optimization,” *in Telecommunications Forum (TELFOR),* 2013 21st.

Liu Chen

e-mail : 6chen.cn@gmail.com

2011년 부산외국어대학교 E-Business학과(학사)

2013년 부경대학교 컴퓨터공학과(석사)

2013년~현 재 부경대학교 컴퓨터공학과 박사과정

관심분야 : 데이터 아키텍처, 데이터 모델링, 빅데이터 분석

고 정 현

e-mail : jhko9120@pknu.ac.kr

2013년 부경대학교 컴퓨터멀티미디어공학과(학사)

2013년~현 재 부경대학교 컴퓨터공학과 석사과정

관심분야 : 데이터 아키텍처, 데이터 모델링, 빅데이터 분석

여 정 모

e-mail : yeo@pknu.ac.kr

1980년 동아대학교 전자공학과(학사)

1982년 부산대학교 전자공학과(공학석사)

1993년 울산대학교 대학원 전자 및 전산기공학과(공학박사)

1986년~현 재 부경대학교 컴퓨터공학과 교수, (주)엔코아 사외이사

관심분야 : 데이터 아키텍처, ITA/EA