ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 7, pp. 820-826, 2016. 7
http://dx.doi.org/10.5626/JOK.2016.43.7.820

매개변수 환경설정에 따른 타조의 외부합병정렬 성능 연구

(External Merge Sorting in Tajo with Variable Server Configuration)

이 종 백 † 강 운 학 †† 이 상 원 †††
(Jongbaeg Lee) (Woon-hak Kang) (Sang-won Lee)

요 약 거대한 데이터로부터 가치 있는 정보를 추출해 내는 빅데이터 기술의 필요성은 나날이 커지고 있다. 빅데이터 분석을 위해 사용되는 하둡 시스템은 맵리듀스를 통해 데이터를 처리하였으나, 맵리듀스 프레임워크는 코드 재사용성의 한계, 질의 최적화 기술의 부재 등의 단점을 보인다. 이를 극복하기 위해 SQL-on-Hadoop이라 불리는 하둡 기반의 SQL 질의 처리 기술이 주목받고 있다. SQL-on-Hadoop 기술 중 타조(Tajo)는 국내 개발진이 주축이 되어 개발되었다. 타조는 데이터 분석을 위해 외부합병정렬 알고리즘을 사용하며, 정렬 연산에 영향을 주는 매개변수로 정렬 버퍼 사이즈와 팬-아웃을 가진다. 본 논문은 타조의 정렬 연산에 영향을 미치는 매개변수인 정렬 버퍼 사이즈와 팬-아웃 값에 따른 정렬의 성능 차이를 보인다. 또한 측정한 성능에 대하여 정렬 버퍼 사이즈가 증가함에 따른 CPU 캐시 미스의 비율 증가, 팬-아웃에 따른 합병 단계 수의 변화가 성능 차이의 원인임을 보인다.

키워드: SQL-on-Hadoop, 아파치 타조, 외부합병정렬, 정렬 버퍼 사이즈, 팬-아웃

Abstract There is a growing requirement for big data processing which extracts valuable information from a large amount of data. The Hadoop system employs the MapReduce framework to process big data. However, MapReduce has limitations such as inflexible and slow data processing. To overcome these drawbacks, SQL query processing techniques known as SQL-on-Hadoop were developed. Apache Tajo, one of the SQL-on-Hadoop techniques, was developed by a Korean development group. External merge sort is one of the heavily used algorithms in Tajo for query processing. The performance of external merge sort in Tajo is influenced by two parameters, sort buffer size and fanout. In this paper, we analyzed the performance of external merge sort in Tajo with various sort buffer sizes and fanouts. In addition, we figured out that there are two major causes of differences in the performance of external merge sort: CPU cache misses which increase as the sort buffer size grows; and the number of merge passes determined by fanout.

Keywords: SQL-on-hadoop, apache tajo, external merge sort, sort buffer size, fanout

· 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(10041244, 스마트TV 2.0 소프트웨어 플랫폼)
· 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(R0126-16-1108, 비휘발성 메모리 기반 개방형 고성능 DBMS 개발)
· 이 논문은 제42회 동계학술발표회에서 '문자열의 최소 δ-근사주기와 최소 γ-근사주기 찾기'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 전자전기컴퓨터공학과
hundredbag@skku.edu
†† 학생회원 : Georgia Institute of Technology
School of Computer Science
woonhak.kang@gatech.edu
††† 종신회원 : 성균관대학교 전자전기컴퓨터공학과 교수
(Sungkyunkwan Univ.)
swlee@skku.edu
(Corresponding author임)

논문접수 : 2016년 2월 12일
(Received 12 February 2016)
논문수정 : 2016년 4월 21일
(Revised 21 April 2016)
심사완료 : 2016년 4월 22일
(Accepted 22 April 2016)

정보과학회논문지 제43권 제7호(2016. 7)

1. 서 론

빅데이터란 단순히 거대한 양의 데이터만을 의미하는 것이 아니며, 많은 양의 데이터로부터 가치 있는 정보를 추출하고 결과를 분석하는 기술을 의미한다. 대표적인 소셜 네트워크 서비스인 '페이스북'에서는 하루 500테라바이트 이상의 데이터를 처리한다. 또한, 뉴욕 증권거래소, 유로넥스트와 같은 여러 증권 거래소를 운영하는 'NYSE 유로넥스트'에서도 하루 2테라바이트 이상의 데이터가 생성된다[1]. 발생하는 데이터의 양이 증가함에 따라 거대한 양의 데이터로부터 의미 있는 정보를 추출하기 위해 빅데이터 분석의 필요성은 크게 증가하였다. 그러나 기존의 관계형 데이터베이스 시스템을 이용한 데이터 처리는 처리 시간과 비용 측면에서 한계를 나타냈고, 이를 극복하기 위하여 분산 환경에서 데이터를 처리하는 빅데이터 프레임워크 하둡(Hadoop)[2]이 등장하였다.

하둡은 분산 환경에서 빅데이터를 처리할 수 있는 프레임워크로, 높은 확장성과 신뢰성을 보인다. 하둡은 크게 데이터를 분산 저장하는 하둡 분산 파일시스템(Hadoop Distributed File System, HDFS)과 빅데이터를 분산 처리하는 맵리듀스(MapReduce)로 구성된다. 맵리듀스는 맵 함수와 리듀스 함수를 이용해 데이터를 처리하는 단순한 모델을 제공하며, 데이터 일괄 처리에 효율적인 구조를 지니고 있다.

데이터 일괄 처리에 효율적인 맵리듀스는 증가하는 실시간 질의 처리의 요구를 충족시키지 못하였다. 또한 맵리듀스는 중간 데이터의 처리에 있어서 많은 디스크 I/O와 네트워크 트래픽을 발생시키는 문제점을 보였으며, 맵과 리듀스로 이루어진 고정된 데이터의 흐름으로 인하여 복잡한 알고리즘의 구현이 어렵다는 한계를 나타냈다[3,4]. 이러한 맵리듀스의 한계를 극복하고자 HDFS에 저장된 대용량의 데이터에 대해 SQL 질의를 처리하는 SQL-on-Hadoop 기술이 주목받고 있으며, 대표적인 예로 아파치 하이브(Hive), 아파치 타조(Tajo), Cloudera의 Impala, UC Berkeley의 Spark 등이 있다. 다양한 SQL-on-Hadoop 기술 중 하나인 아파치 타조는 국내 개발진이 주축을 이뤄 개발된 하둡 기반의 대용량 웨어하우스 시스템으로, 2013년 3월 아파치 인큐베이션 프로젝트로 채택되었으며, 1년 후 2014년 4월에는 아파치 최상위 프로젝트로 선정되어 큰 관심을 받았다.

타조에서는 정렬을 수행하기 위하여 외부합병정렬 알고리즘을 사용한다. 정렬은 다양한 데이터 분석 작업에서 중요한 역할을 하는 연산으로, Order by 또는 Join 등의 SQL 질의를 처리하기 위해 사용된다. 타조에서 외부합병정렬의 수행 시 매개변수를 통한 환경설정으로 외부합병정렬의 동작에 영향을 줄 수 있다. 정렬에 사용되는 중요한 매개변수 두 가지는 각각은 정렬 버퍼 사이즈(Sort Buffer Size)와 팬-아웃(Fanout)이다. 정렬 버퍼 사이즈는 외부합병정렬의 런(Run)의 크기를 결정할 때 영향을 주는 매개변수이다. 타조에서는 이 값에 의해 정해진 크기만큼의 데이터를 청크(Chunk) 단위로 나누어 메모리 내부에서 정렬한다. 팬-아웃은 외부합병정렬의 합병 과정에 영향을 주는 매개변수이다. 팬-아웃 값은 정렬된 여러 런을 하나의 큰 런으로 합병하는 과정에서 동시에 합병되는 런의 개수를 의미한다.

본 논문에서는 타조의 정렬 연산에 영향을 주는 두 가지 매개변수인 정렬 버퍼 사이즈와 팬-아웃 값의 변화에 따른 정렬 연산의 성능을 측정한다. 본 논문의 기여사항은 다음과 같다.

첫째, 정렬 버퍼 사이즈에 따른 타조의 외부합병정렬의 성능을 측정한다. 또한 정렬 버퍼 사이즈 증가에 따른 CPU 캐시 미스 비율 증가가 정렬 성능에 영향을 주는 것을 보인다.

둘째, 팬-아웃 값에 따른 타조의 외부합병정렬의 성능을 측정한다. 측정 결과를 바탕으로 팬-아웃 값이 합병 단계 수에 주는 영향을 보인다. 그리고 타조에서 태스크당 처리하는 데이터의 양을 고정시키기 때문에 팬-아웃 값에 따른 성능 변화를 잘 활용하지 못함을 보인다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 연구의 배경이 되는 하둡에 관하여 설명하고 SQL-on-Hadoop 기술과 그 등장 배경에 관하여 설명한다. 3장에서는 타조의 구조와 타조에서 사용하는 정렬 알고리즘에 관하여 설명하며, 타조의 정렬에 영향을 주는 두 가지 매개변수와 그 역할을 소개한다. 4장에서는 매개변수의 값을 변경하며 측정한 타조의 정렬 성능 평가를 보이며, 성능 평가의 결과에 대한 분석에 관하여 논할 것이다. 마지막으로 5장에서는 결론과 향후연구로 본 논문을 마무리한다.

2. 하둡과 SQL-on-Hadoop

2.1 하둡

하둡(Hadoop)은 클러스터를 이용하여 커다란 데이터 집합에 대한 분산 처리를 제공하는 소프트웨어 프레임워크이다[5]. 하둡을 구성하는 중요한 두 가지 요소는 클러스터 환경에서 대용량 데이터를 분산하여 저장하는 하둡 분산 파일 시스템(Hadoop Distributed Filesystem, HDFS[6])과 HDFS에 저장된 데이터를 분산하여 병렬 처리하는 맵리듀스(MapReduce) 프레임워크[7]이다. 하둡의 코어 프로젝트인 HDFS와 맵리듀스에 추가적으로 하둡을 효율적으로 활용하거나 보완하기 위한 서브 프로젝트들이 추가로 제공되어 하둡 에코시스템을 구성하게 되며, 실시간으로 SQL 질의를 처리하는 SQL-on-Hadoop

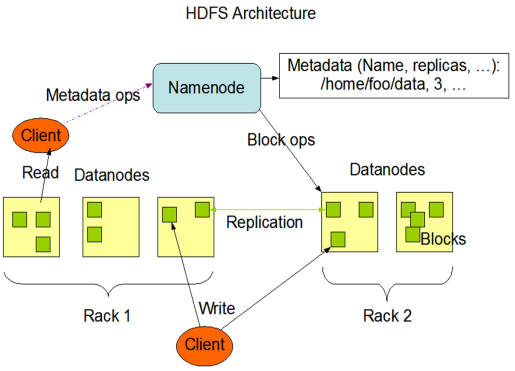

그림 1 하둡 분산 파일시스템 구조[2]
Fig. 1 Hadoop distributed file system architecture

기술들이 하둡 서브 프로젝트에 해당한다.

하둡의 코어 프로젝트 중 하나인 HDFS는 상용 하드웨어에서 실행할 수 있도록 디자인 된 분산 파일시스템이다. HDFS는 저비용의 하드웨어를 효율적으로 사용할 수 있도록 설계되었으며, 높은 내고장성을 제공하는 특징을 가진다.

그림 1에서 알 수 있듯이 HDFS는 네임노드와 데이터노드로 구성되는 마스터-슬레이브 구조를 가진다. 네임노드는 HDFS의 파일과 디렉토리의 메타데이터를 관리하며, 데이터노드에 저장된 실제 데이터 블록의 매핑 정보를 제공하는 역할을 한다. 또한 클라이언트로부터의 파일 접근을 통제하는 역할을 수행하기도 한다. 데이터노드는 블록 단위(기본 128MB)로 나누어지는 실제 데이터를 저장하는 역할을 하는 여러 노드들로 구성된다. 데이터를 읽거나 쓰고자 하는 클라이언트는 네임노드로부터 얻은 파일의 메타데이터를 이용하여 데이터노드로부터 실제 데이터를 읽거나 쓰는 작업을 수행한다. 저비용의 하드웨어를 사용하는 하둡의 특성으로 인하여 데이터노드는 같은 데이터 블록을 여러 데이터노드에 복사하여 저장함으로써 데이터 내고장성을 제공한다.

하둡의 코어 프로젝트 중 다른 하나인 맵리듀스는 HDFS에 분산 저장된 데이터에 대한 병렬 처리 기법을 제공한다. 맵리듀스는 데이터를 <Key, Value> 형태로 데이터를 분류하는 맵(Map) 함수와 맵 함수의 결과에 대하여 집계 연산을 통해 원하는 값을 얻는 리듀스(Reduce) 함수로 구성된다. 맵리듀스는 맵과 리듀스 두 개의 함수를 이용해 데이터를 처리하기 때문에 단순하고 편리하다는 특징과, 비구조적 데이터의 처리가 효율적이라는 점, 그리고 대량의 데이터를 일괄적으로 처리하는 데에 효과적이라는 특징을 가지고 있다.

2.2 SQL-on-Hadoop

하둡에서 대용량 데이터에 대한 분산 처리를 위해 사용되는 맵리듀스 기법은 데이터의 일괄 처리에 효율적으로 동작한다. 그렇지만 맵리듀스는 높은 내고장성과 확장성을 위해 중간 데이터 처리에 많은 디스크 I/O와 네트워크 트래픽을 발생시키는 문제를 보인다. 또한 고차원 언어의 부재로 인하여 코드 재사용성의 한계, 질의 최적화 기술의 부재의 문제를 가진다. 데이터의 흐름이 고정적이기 때문에 복잡한 알고리즘 구현에 한계를 나타내는 것 또한 맵리듀스의 단점 중 하나이다[3,4]. 위의 문제점과 더불어 대화형 데이터 분석에 대한 수요가 증가함에 따라 데이터 처리를 위한 맵리듀스의 대안으로 SQL-on-Hadoop 기술이 주목을 받고 있다.

SQL-on-Hadoop이란 HDFS에 저장된 데이터에 대하여 SQL 스타일의 질의처리를 제공하는 시스템을 지칭한다. 다양한 SQL-on-Hadoop 기술들이 존재하며, 이들은 데이터 파일을 HDFS에 저장하고 SQL문을 통하여 데이터를 분산 환경에서 처리하는 특징을 지닌다.

대표적인 SQL-on-Hadoop 기술들은 아파치 하이브(Hive)[8], Cloudera의 Impala[9], UC Berkeley의 Spark[10,11], 아파치 타조(Tajo)[12] 등이 있다. 하이브는 사용자가 SQL과 유사한 HiveQL을 통해 하둡 상의 데이터에 대한 질의 처리와 관리 기능을 제공한다. 맵리듀스의 맵, 리듀스 기능이 HiveQL에 플러그-인 되어 질의를 처리하는 특징을 가진다. Impala와 타조는 HDFS에 저장된 파일에 대하여 맵리듀스 모델을 사용하지 않고 독자적인 데이터 처리 프레임워크를 사용하는 특징을 가진다. Spark는 RDD(Resilient Distributed Datasets)라는 메모리에 캐싱되어있는 분산 객체를 이용하며 빠른 데이터 처리 기능을 제공하는 특징이 있다. 각각의 기술은 처리하고자 하는 질의 수행의 시간, 스케줄링 기법 등 다양한 특성을 지니고 있기 때문에 시스템의 특성을 이해하고, 워크로드에 알맞은 시스템을 선택하는 것이 중요하다[13].

3. 아파치 타조

본 장에서는 타조의 구조와 타조에서 사용하는 외부 합병정렬 알고리즘의 동작 과정, 그리고 타조의 외부합병정렬에 영향을 주는 두 매개변수인 정렬 버퍼 사이즈, 팬-아웃 값의 역할에 관하여 논한다.

3.1 타조의 구조

SQL-on-Hadoop 기술 중 하나인 타조는 하둡 기반의 대용량 분산 데이터 웨어하우스 시스템이다[12]. 타조는 HDFS에 저장된 데이터에 대하여 짧은 지연시간을 요구하는 질의 처리를 위해 설계되었으며, 실시간 질의 처리 뿐 아니라 긴 시간동안 수행되는 대용량 데이터에 대한 ETL 작업도 지원한다. 또한 SQL 표준을 지원하며, 성능을 위하여 질의 전체를 분산 처리하는 특징을 지닌다.

타조는 타조 마스터와 타조 워커로 이루어지는 마스터-슬레이브 구조로 구성된다. 타조 마스터는 타조 클러스터의 마스터 역할을 담당하는 노드로, 클라이언트로부터 요청된 질의를 파싱하며 질의 수행 계획을 세우고 계획에 대한 최적화를 수행하는 노드이다. 그 밖에도 타조 마스터는 각종 통계 정보를 관리하고, 클러스터 전체의 자원을 관리하는 역할을 한다. 타조 워커는 마스터가 요청한 질의를 실제로 실행하는 역할을 하며 이 과정에서 필요한 저장장치에 접근한다. 여러 타조 워커 중 하나는 쿼리마스터로써 동작하도록 선택된다. 쿼리마스터는 마스터-슬레이브 구조적인 특성으로 인해 마스터 서버에 문제가 생길 경우 질의 처리가 정상적으로 수행되지 못하는 경우를 극복하기 위한 노드로, 질의 별로 질의 실행에 관한 분산 실행 계획을 제어하는 역할을 한다.

3.2 타조의 외부합병정렬

타조에서는 Order by, Join 등의 정렬이 필요한 SQL 질의를 처리하기 위하여 외부합병정렬 알고리즘을 사용한다. HDFS에 분산되어 저장되어있는 데이터에 대하여 SQL 질의를 처리하는 SQL-on-Hadoop 기술의 특징으로 인하여 타조에서 사용하는 외부합병정렬 알고리즘은 기존의 상용 DBMS에서 사용하는 외부합병정렬 알고리즘과 구분되는 특징을 가진다.

타조는 하나의 SQL 질의를 여러 노드로 구성되는 타조 워커에 분산하여 실행한다. 각각의 타조 워커는 분산된 SQL 질의 처리 작업을 여러 Task 단위로 나누어 각 Task를 쓰레드(Thread) 별로 수행한다. 이 때 하나의 워커 노드에서 동작하는 Task의 수는 해당 노드의 CPU, 메모리 등의 시스템 환경을 바탕으로 결정된다.

타조에서 사용하는 외부합병정렬은 크게 두 가지 부분으로 나눌 수 있다. 첫 번째 단계에서 타조의 각 Task는 외부합병정렬을 수행하기 위한 런(Run)을 생성한다. 이 때 생성되는 런은 청크(Chunk) 단위로 나누어 생성되며, 메모리 내부에서 정렬되어 타조의 각 노드의 로컬 Temp 영역에 저장된다. 두 번째 단계에서 타조 워커는 정렬된 각 런을 합병하는 작업을 수행한다. 이 때 각각의 Task는 Temp 영역에 저장된 런을 정렬된 하나의 큰 런으로 합병한다. 다른 타조 워커 노드에서 작업된 내용의 경우 셔플 작업을 통해 정렬된 런을 가져와 합병하는 작업을 수행한다.

타조의 외부합병정렬에 영향을 주는 두 가지 환경설정 매개변수는 정렬 버퍼 사이즈와 팬-아웃이다. 타조의 외부합병정렬은 초기 런의 크기를 정렬 버퍼 사이즈 매개변수 값에 맞추어 생성하는 특징을 가진다. 팬-아웃 매개변수는 합병 작업에서 하나의 큰 런을 생성하기 위해 동시에 합병되는 런의 수를 의미하며, 이는 기존의 외부합병정렬 알고리즘에서의 팬-아웃과 같은 역할을 한다.

4. 성능 평가 및 분석

본 장에서는 타조의 외부합병정렬의 성능에 영향을 주는 매개변수인 정렬 버퍼 사이즈와 팬-아웃 값에 따른 타조의 성능을 평가하고 그 결과에 관하여 분석한다.

4.1 실험 환경

실험에 사용한 HDFS 클러스터는 1개의 네임노드와 3개의 데이터노드로 구성되며 각 노드는 10Gbit/s 네트워크로 연결된다. 각 데이터노드는 2개의 소켓에 각각 12코어 CPU를 장착한 24 코어 Intel Xeon E5-2670V3 2.3GHz 환경에서 실시하였으며, 논리적 코어는 48개로 동작한다. 메인메모리는 DDR3 64GB로 구성되며, 리눅스(커널 3.17.2) 운영체제에서 실험하였다. HDFS는 데이터 블록의 크기를 128MB로 사용하도록 설정되었으며, 각 데이터 블록의 복제본의 수는 기본 값인 3으로 설정되었다. 각 데이터노드에서 데이터를 저장하기 위해 사용하는 저장장치는 Samsung SSD 850 Pro 256GB로 총 3개의 장치가 사용되었다.

실험을 위해 사용된 타조의 버전은 0.10.1 이며, 타조 클러스터의 구성은 표 1과 같다. 타조 마스터는 HDFS 클러스터의 네임노드와 같은 노드에서 동작한다. 타조 워커는 HDFS 클러스터의 데이터노드를 구성하는 3개의 노드 중 하나이다. 타조 워커에서 사용하는 Temp 저장장치는 Samsung SSD 840 Pro 256GB를 사용하였다.

정렬 연산의 대상으로 TPC-H 벤치마크에서 사용되는 테이블 중 'LINEITEM' 테이블을 이용하였다. 실험에 사용한 전체 TPC-H 벤치마크 데이터의 양은 약 31GB로 HDFS 상에 저장하였다. 'LINEITEM' 테이블은 전체 데이터의 약 72%인 23GB를 차지한다. 'LINEITEM' 테이블의 여러 컬럼 중 'L_SUPPKEY' 컬럼을 정렬에 사용되는 키로 사용하였다. 정렬에 사용한 쿼리는 order by 문을 포함하는 단순 SQL 질의로 실험하였다.

표 1 타조 클러스터 환경설정
Table 1 Tajo cluster configuration

Tajo Master	Description
CPU	AMD Opteron(tm) Processor 6128 16 Core
RAM	40GB

Tajo Worker	Description
CPU	Intel Xeon E5-2670V3 2.3GHz 24 Core (48 Thread)
RAM	64GB
Temp Storage	Samsung SSD 840 Pro 256GB
Sort Buffer Size	4MB, 64MB, 200MB(default)
Fanout	2, 8(default), 16

4.2. 정렬 버퍼 크기에 따른 실험 결과 및 분석

정렬 버퍼 사이즈는 기본적으로 200MB로 설정되어 있으며, 본 논문에서는 해당 값을 4MB, 64MB로 변경하며 실험하였다. 정렬 버퍼 사이즈를 증가시켜 실험할 경우 자바의 가비지 컬렉션의 발생 빈도가 증가하여 정렬 연산의 성능 측정에 영향을 주어 기본 값보다 큰 경우에 대해서는 실험하지 않았다.

표 2는 정렬 버퍼 사이즈를 변경하며 수행한 정렬 연산의 총 수행 시간을 나타낸다. 정렬 버퍼 사이즈가 기본 값인 200MB일 때 타조의 외부합병정렬의 수행 시간은 577.9초로 가장 느린 성능을 보인다. 정렬 버퍼 사이즈를 64MB로 설정하였을 때 외부합병정렬의 수행 시간은 497.4초, 정렬 버퍼 사이즈를 4MB로 설정하였을 때의 수행시간은 429.5초로 측정되었다.

정렬 버퍼 사이즈에 따른 성능 차이의 원인 분석을 위하여 타조에서 외부합병정렬을 수행하는 과정에서 정렬에 사용하는 시간과 임시 데이터의 쓰기를 위해 사용되는 시간을 측정하였다. 정렬 버퍼 사이즈가 200MB인 경우 정렬에 사용된 시간은 614.4초, 데이터 쓰기에 사용된 시간은 439.4초로 측정되었다. 정렬 버퍼 사이즈가 64MB인 경우 정렬에 사용된 시간은 392초, 데이터 쓰기에 사용된 시간은 373.2초로 측정되었으며, 정렬 버퍼 사이즈가 4MB인 경우 정렬에 사용된 시간은 139.7초, 데이터 쓰기에 사용된 시간은 355.8초로 측정되었다. 임시 데이터 쓰기에 비해 정렬에 사용되는 시간이 정렬 버퍼 사이즈가 커짐에 따라 크게 증가하는 것을 확인할 수 있다.

그림 2는 측정된 정렬 시간과 임시 데이터 쓰기 시간을 튜플의 수로 나누어 튜플 당 정렬 시간, 튜플 당 쓰기 시간을 측정한 결과를 나타낸다. 측정 결과에 따르면 정렬 버퍼 사이즈가 증가함에 상관없이 튜플 당 쓰기 시간은 약 0.002ms로 비슷한 값을 보이는 반면, 튜플 당 정렬 시간은 정렬 버퍼 사이즈가 200MB일 때 0.0038ms로 해당 값이 64MB일 때의 튜플 당 정렬 시간인 0.002초, 4MB일 때의 튜플 당 정렬 시간인 0.001초에 비해 크게 증가한 것을 확인할 수 있다.

튜플 당 정렬 시간과 정렬 버퍼 사이즈의 관계를 분석하기 위해 리눅스 시스템 성능 측정 도구인 Perf[14]를 이용하여 CPU의 Last Level 캐시의 캐시 미스 비율,

표 2 정렬 버퍼 크기에 따른 수행 시간
Table 2 Execution time with variable sort buffer size

Sort Buffer Size	Execution Time(sec)
4MB	429.5
64MB	497.4
200MB(default)	577.9

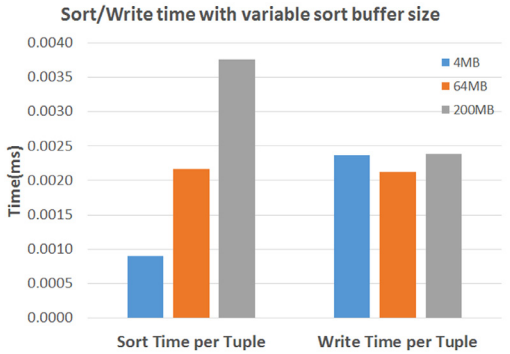

그림 2 정렬 버퍼 사이즈에 따른 정렬/쓰기 시간
Fig. 2 Sort/Write time with variable sort buffer size

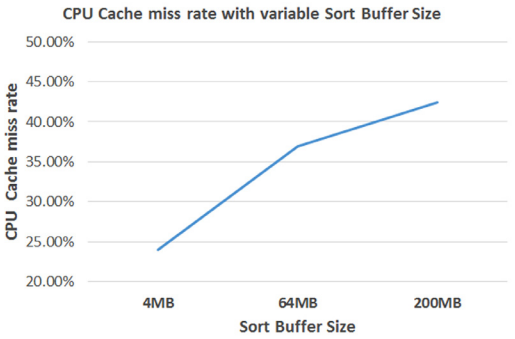

그림 3 정렬 버퍼 사이즈에 따른 CPU 캐시 미스 비율
Fig. 3 CPU Cache miss rate with variable sort buffer size

Branch miss의 비율, TLB-load miss의 비율을 측정하였다. 측정 결과 TLB-load miss의 비율은 약 0.05%, Branch miss의 비율은 약 0.7%로 정렬 버퍼 사이즈의 크기에 따른 차이가 크게 나타나지 않았다. 그림 3은 정렬 버퍼 크기에 따른 CPU 캐시 미스 비율을 나타낸다. 측정 결과 정렬 버퍼 사이즈가 4MB, 64MB, 200MB일 때 CPU 캐시 미스 비율은 24.05%, 36.94%, 42.47%로 측정되었다.

측정 결과를 통해 정렬 버퍼 사이즈가 증가함에 따라 CPU 캐시 미스 비율이 증가하며, 이것이 타조의 외부합병정렬의 수행 시간에 영향을 준 것을 확인할 수 있다. 하나의 타조 워커에서는 정렬을 수행하는 여러 태스크가 동시에 동작하게 된다. 정렬 버퍼 사이즈의 값이 증가함에 따라 하나의 런에 대한 인-메모리 정렬에 필요한 시간이 증가하게 된다. 동시에 여러 태스크가 동작하는 환경에서 하나의 런에 대한 인-메모리 정렬의 시간이 증가하여 정렬 대상이 되는 데이터가 CPU 캐시로부터 쫓겨날 가능성이 증가한 것이 CPU 캐시 미스 비율이 증가한 원인으로 볼 수 있다.

4.3 팬-아웃 크기에 따른 실험 결과 및 분석

타조에서 한 번에 합병되는 런의 개수를 의미하는 팬-아웃은 기본적으로 8로 설정되어있다. 본 논문에서는 정렬 버퍼 사이즈를 4MB로 고정시킨 뒤 팬-아웃의 크기를 2, 8, 16으로 변경함에 따른 외부합병정렬의 성능을 측정하였다. 표 3은 팬-아웃 값에 따른 타조의 외부합병정렬의 성능을 나타낸다. 팬-아웃을 기본 값인 8로 설정하여 정렬 연산을 수행할 경우 총 수행 시간은 429.5초로 측정되었으며, 팬-아웃을 16으로 증가시킨 뒤 수행한 연산의 수행 시간은 407.2초로 측정되었다. 마지막으로 팬-아웃의 크기를 2로 설정하여 수행한 실험에서 정렬 연산은 1021.8초의 수행시간을 나타냈다.

팬-아웃 값이 증가함에 따른 성능 향상의 원인은 외부합병정렬 연산에서의 합병 단계의 수가 감소했기 때문이다. 본 논문의 실험 환경에서 정렬 버퍼 사이즈가 4MB일 때, 정렬을 위해 생성되는 초기 런의 개수는 각 Task마다 약 185개이다. 따라서 팬-아웃 값이 2로 설정된 경우 하나의 태스크가 생성한 런을 합병하는 단계는 log2185로 총 8단계가 필요하다. 같은 방법으로 계산했을 때, 팬-아웃이 기본 값인 8로 설정된 경우 필요한 합병 단계는 3번, 팬-아웃이 기본 값의 두 배인 16으로 설정된 경우 필요한 합병 단계는 2번이 필요하며, 합병 단계 수의 감소가 성능 향상에 영향을 준 것을 알 수 있다.

데이터를 합병하는 단계마다 데이터에 대한 읽기, 쓰기가 발생하기 때문에 합병 단계가 많을수록 Temp 저장장치에 쓰는 데이터의 양은 증가하는 것을 측정을 통해 확인할 수 있었다. 팬-아웃을 기본 값인 8로 설정했을 때 Temp 저장장치에 쓰인 임시 데이터의 양은 총 60.8GB이다. 팬-아웃을 기본 값보다 2배 큰 16으로 설정하여 실험한 결과 Temp 저장장치에 쓰인 임시 데이터의 양은 총 52.8GB로 팬-아웃이 8일 때보다 감소하는 것을 확인할 수 있다. 팬-아웃이 2로 설정된 경우 Temp 저장장치에 쓰는 데이터의 양은 총 134.2GB로 합병 단계의 수가 크게 증가함에 따라 Temp 저장장치에 쓰는 임시 데이터의 양이 증가하는 것 또한 확인할 수 있다.

본 실험을 수행하던 중 확인한 타조의 동작 방식을 통해 타조의 외부합병정렬이 팬-아웃 매개변수의 값을 잘 활용하지 못하는 것을 발견하였다. 타조에서는 기본적으로 하나의 Task 당 정렬할 데이터의 양을 매개변수를 통해 변경할 수 있는 것이 아닌 고정된 값으로 설정하고 있다. 따라서 정렬 버퍼 사이즈를 기본 값인 200MB로 설정하고 외부합병정렬 연산을 수행할 경우, 하나의 Task에서 처리해야 할 총 런은 4개가 생성된다. 따라서 정렬 연산에 영향을 주는 정렬 버퍼 사이즈, 팬-아웃 값을 기본 값으로 설정하여 수행하는 경우 팬-아웃의 기본 값인 8을 잘 활용하지 못하는 문제가 있다.

표 3 팬-아웃에 따른 수행 시간
Table 3 Execution time with variable fanout

Fanout	Execution Time(sec)
2	1021.8
8 (default)	429.5
16	407.2

5. 결론 및 향후 연구

본 논문에서는 SQL-on-Hadoop 기술 중 하나인 타조의 외부합병정렬 연산에 사용되는 정렬 버퍼 크기, 팬-아웃 매개변수를 변경함에 따른 정렬 성능을 평가하였고, 각 매개변수의 값에 따른 성능 차이가 발생하는 원인에 관하여 분석하였다.

성능 평가를 통해 정렬 버퍼 크기를 200MB에서 4MB로 줄일 경우 정렬 연산의 총 수행 시간은 577초에서 429.5초로 감소하는 것을 확인할 수 있었다. 정렬 버퍼 크기가 외부병합정렬 연산에 영향을 주는 원인을 분석한 결과 튜플 당 정렬 시간이 정렬 버퍼 크기가 클수록 증가하는 것을 확인하였으며, 정렬 버퍼 크기가 증가함에 따라 CPU 캐시 미스 비율의 증가하는 것이 이러한 현상의 원인임을 확인하였다.

또한 팬-아웃 값에 따른 성능 평가를 통해서 팬-아웃 값이 클수록 외부합병정렬 알고리즘의 합병 단계의 수가 감소하여 외부합병정렬의 성능이 증가하는 것을 보였다. 그리고 하나의 태스크에서 정렬할 데이터의 양을 고정시키는 타조의 동작으로 인하여 정렬 버퍼 사이즈를 기본 값인 200MB로 설정하였을 때 팬-아웃 값의 증가에 따른 성능 향상을 기대할 수 없는 문제를 보였다.

향후 연구에서는 서버의 시스템 환경에 따른 타조에서의 최적의 환경설정에 관하여 연구를 진행할 것이다. 또한 또 다른 대표적인 연산인 해시 연산에 관해서도 같은 방향의 연구를 진행할 것이다.

References

[1] Cisco, "Data Virtualization Redefines the Stock Exchange," Cisco, 2013.

[2] Apache Hadoop. [Online]. Available: http://hadoop.apache.org/

[3] K.-H. Lee, W.J. Park, K.S. Cho, W.Ryu, "The MapReduce framework for Large-scale Data Analysis: Overview and Research Trends," Electronics and Telecommunications Trends, Vol. 28, No. 6, pp. 156-166, Dec. 2013. (in Korean)

[4] Ma, Zhiqiang, and Lin Gu, "The limitation of MapReduce: A probing case and a lightweight solution," *Proc. of the 1st Intl. Conf. on Cloud Computing, GRIDs, and Virtualization*, pp. 68-73, 2010.

[5] White, Tom, Hadoop: The definitive guide, O'Reilly Media, Inc, 2012.
[6] Shvachko, Konstantin, et al., "The hadoop distributed file system," *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE*, pp. 1-10, 2010.
[7] Dean, Jeffrey, and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, Vol. 51, No. 1, pp. 107-113, 2008.
[8] Apache Hive. [Online]. Available: http://hadoop.apache.org/hive
[9] Kornacker, Marcel, et al., "Impala: A Modern, Open-Source SQL Engine for Hadoop," *CIDR*, 2015.
[10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, "Spark: cluster computing with working sets," *Proc. of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 10-10, Jun. 2010
[11] Zaharia, Matei, et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*, USENIX Association, pp. 2-2, 2012.
[12] Apache Tajo: A big data warehouse system on Hadoop, [Online]. Available: http://tajo.apache.org/
[13] Chen, Yueguo, et al., "A study of sql-on-hadoop systems," *Big Data Benchmarks, Performance Optimization, and Emerging Hardware. Springer International Publishing*, pp. 154-166, 2014.
[14] Arnaldo Carvalho de Melo, "The New Linux perf tools," presentation from Linux Kongress, 2010.

이 종 백

2013년 성균관대학교 컴퓨터공학과 학사 2013년~현재 성균관대학교 전자전기컴퓨터공학과 석박통합과정. 관심분야는 플래시 메모리 DBMS, 분산 데이터베이스 시스템

강 운 학

2006년 성균관대학교 컴퓨터공학과 학사 2010년 성균관대학교 전자전기컴퓨터 공학과 석사. 2010년~2016년 성균관대학교 전자전기컴퓨터공학과 박사. 2016년~현재 Georgia Institute of Technology School of Computer Science 박사후 과정. 관심분야는 데이터베이스 시스템 구조 및 개발, 플래시 메모리 DBMS

이 상 원

1991년 서울대학교 컴퓨터학과 학사. 1994년 서울대학교 컴퓨터학과(석사). 1999년 서울대학교 컴퓨터학과(박사). 1999년~2001년 한국오라클. 2001년~2002년 이화여대 BK21 계약교수. 2002년~현재 성균관대학교 컴퓨터공학과 교수. 관심분야는 플래시 메모리 DBMS