클라우드 컴퓨팅에서의 대규모 데이터를 위한 분산 병렬 처리 기법의 성능분석[1)]

Performance Analysis of Distributed Parallel Processing Schemes for Large Data in Cloud Computing

홍승태* · 장재우
Seung-Tae Hong* · Jae-Woo Chang
전북대학교 컴퓨터공학과
sthong@dblab.chonbuk.ac.kr* · jwchang@chonbuk.ac.kr

요 약

최근 IT 분야에서 인터넷을 기반으로 IT 자원들을 서비스 형태로 제공하는 클라우드 컴퓨팅에 대한 연구가 활발히 진행되고 있다. 한편, 효율적인 클라우드 컴퓨팅을 제공하기 위해서는, 막대한 양의 데이터를 수많은 서버들에 분산 저장하고 관리하기 위한 분산 데이터 저장 기법 및 분산 병렬 처리 기법에 대한 연구가 필수적이다. 이를 위해 본 논문에서는 대표적인 분산 병렬 처리 기법에 대해 살펴보고, 이를 비교 분석한다. 마지막으로 Hadoop 기반 클러스터를 구축하고 이를 통해서 대규모 데이터를 위한 분산 병렬 처리 기법에 대한 성능평가를 수행한다.

1. 서론

최근 IT 분야에서 클라우드 컴퓨팅에 대한 연구가 활발히 진행되고 있다. 클라우드 컴퓨팅이란 인터넷을 기반으로 하여 IT 자원들을 서비스 형태로 제공하는 컴퓨팅을 의미한다[1,2]. 즉, 개인용 컴퓨터 또는 기업의 서버에 개별적으로 저장해 두었던 자료와 소프트웨어들을 클라우드 클러스터로 구축하여 필요할 때 PC나 휴대폰과 같은 각종 단말기를 이용해 원격으로 원하는 작업을 수행할 수 있는 환경을 뜻한다. 클라우드 컴퓨팅은 2006년 Google의 크리스토프 비시글리아가 CEO인 에릭 슈미츠에게 처음으로 제안하였으며[3], 이후 경제 전문지 및 대표적인 글로벌 기업의 CEO들이 잇달아 클라우드 컴퓨팅을 차기 주력 비즈니스 아이템으로 지목하면서 클라우드 컴퓨팅은 IT 시장의 주요 관심사로 떠올랐다. 국외에서는 이미 많은 연구와 개발이 진행 중이며, 대표적인 예로 Amazon Elastic Compute Cloud[4], IBM Blue Cloud[5], 그리고 Google App Engine[6] 등이 있다. 국내에서도 2009년 한국클라우드서비스협의회(CSKI)의 출범을 시작으로 산 · 학 · 연 공조 체제를 형성하는 등 클라우드 컴퓨팅 세계 시장에 합류하기 위한 준비를 서두르고 있다.

한편, 클라우드 컴퓨팅을 제공하기 위해서는 막대한 물리적인 IT 인프라 구축뿐만 아니라, 클라우드 컴퓨팅의 성장과 더불어 폭발적으로 증가하고 있는 데이터를 효율적으로 활용할 수 있는 기술들이 요구된다. 따라서 클라우드 컴퓨팅에서 다루어지는 막대한 양의 데이터를 수많은 서버들에 분산 저장하고 관리하기 위한 분산 데이터 저장 기법과 분산 병렬 처리 기법에 대한 연구가 필수적이다. 이를 위해 본 논문에서는 대표적인 분산 병렬 처리 기법인 Apache의 Hadoop MapReduce[7]

1) 본 연구는 교육과학기술부와 한국산업기술진흥원의 지역혁신인력양성사업으로 수행된 연구결과임

및 이를 개선한 MapReduce Online[8]에 대하여 분석하고 이를 비교한다. 아울러 Hadoop 분산 파일 시스템과 분산 병렬 처리 기법을 이용하여 클라우드 컴퓨팅을 위한 클러스터를 구축하고, 다양한 실험을 통하여 MapReduce와 MapReduce Online의 성능을 분석한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 Google의 파일 시스템과 Apache의 Hadoop 분산 파일 시스템에 대하여 살펴보고, 3장에서는 대표적인 분산 병렬 처리 기법인 MapReudce와 MapReduce Online에 대하여 비교 분석한다. 4장에서는 Hadoop을 이용한 클러스터 구축 방법을 설명하고, 아울러 구축한 클러스터를 통하여 MapReudce와 MapReduce Online에 대한 성능을 분석한다. 마지막으로 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 클라우드 컴퓨팅에서의 분산 데이터 저장 기법

분산 파일 시스템은 대용량 데이터를 저장하고 관리하기 위해 많은 서버들에 데이터를 나누어 저장하고 관리하는 파일 시스템이다. 클라우드 컴퓨팅에서의 분산 파일 시스템은 단순히 데이터를 저장하고 관리하는 것뿐만이 아니라, 데이터를 저장하고 있는 하드웨어의 장애에 유연하게 대처하고 서비스가 요구하는 충분한 성능도 보장해야 한다. 최근 클라우드 컴퓨팅이 부각되면서 기존의 분산 파일 시스템 기술들이 다양한 형태로 활용되고 있으나, 클라우드 컴퓨팅에서 대용량 데이터를 위한 분산 데이터 저장 기법으로써 활용되고 있는 분산 파일 시스템은 그리 많지 않다. 본 장에서는 분산 데이터 저장 기법 중에서 대표적으로 사용되면서 구조가 공개되어 있는 Google의 파일 시스템[9] 및 Apache의 Hadoop 분산 파일 시스템[10]에 대하여 살펴본다.

2.1 Google 파일 시스템(GFS)

GFS는 Google에서 자체 개발한 분산 파일 시스템으로써 (그림 1)에서처럼 여러 클라이언트들에 의해 접근되는 단일 마스터 서버와 여러 청크(chunk) 서버들로 구성된다. GFS에서 파일들은 청크라 불리는 64 메가바이트 크기의 단위로 나뉘어 관리되며 각 청크들은 여러 청크 서버에 분산되어 저장된다. 또한 각 청크에 대한 다수의 복제본을 여러 청크 서버에 분산하여 저장함으로써 장애로 인한 데이터의 손실을 방지한다.

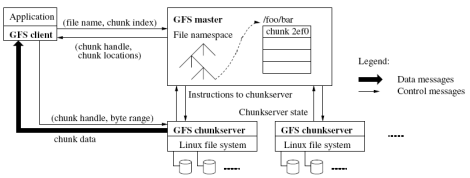


그림 1. GFS의 구조

2.2 Hadoop 파일 시스템(HDFS)

HDFS는 오픈 소스 소프트웨어 개발 프로젝트인 Hadoop에서 분산 컴퓨팅 프레임워크를 지원할 목적으로 개발된 분산 파일 시스템이다. HDFS는 현재 Amazon, IBM, Yahoo 등과 같은 글로벌 IT 기업들의 클라우드 컴퓨팅 플랫폼의 기반이 되는 분산 파일 시스템으로 가장 널리 활용되고 있다. HDFS는 GFS를 본보기로 삼아 개발된 분산 파일 시스템으로 플랫폼간의 이식성을 보장하기 위해 자바를 사용하여 구현되었다.

HDFS는 (그림 2)와 같이 GFS와 동일한 구조와 기능을 제공한다. HDFS는 네임스페이스를 관리하고 클라이언트의 파일에 대한 접근을 통제하는 단일 네임노드(namenode)와 데이터 저장소를 관리하는 수많은 데이터노드(datanode)들로 구성된다. HDFS에서 하나의 파일은 GFS의 청크와 동일한 개념의 블록들로 이루어지며 데이터노드들에 분산되어 저장된다. 마지막 블록을 제외하고는 한 파일을 구성하

는 모든 블록들은 동일한 크기를 가지며 데이터노드의 장애에 대처하기 위해 블록들의 복제본이 여러 데이터노드들에 중복되어 저장된다. 파일마다 블록의 크기와 복제본의 수를 조절할 수 있으며 복제본의 수는 파일을 생성할 때 지정할 수도 있고 나중에 변경하는 것도 가능하다.

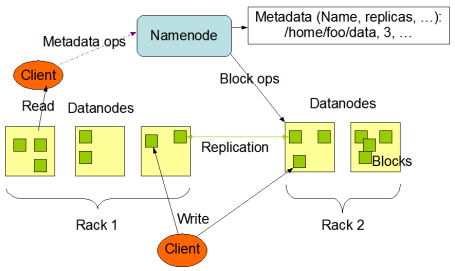


그림 2. HDFS의 구조

3. 클라우드 컴퓨팅에서의 분산 병렬 처리 기법

최근 인터넷 서비스의 발전과 더불어 온라인 커뮤니티에서의 사용자의 능동적인 참여로 인하여 인터넷 데이터가 급격히 증가하고 있다. 이에 따라 데이터의 효율적인 저장과 관리가 중요해지고 있으며, 아울러 점점 다양해지고 있는 사용자의 요구를 충족시키기 위해서는 대용량 데이터에 대한 분산 병렬 처리 기법이 요구된다. 이를 위해 각 응용에서는 독자적으로 병렬 분산 처리를 활용하여 분산 병렬 처리 기법을 개발해 오다가 이를 공통 플랫폼으로 제공하기 시작했다. Google에서 2004년 MapReduce를 발표 후, 이를 기반으로 야후, Hadoop 등에서 분산 병렬 처리 기법들이 개발되었으며, 현재는 분산 병렬 처리 기법에서는 MapReduce가 표준이 되어 가고 있다. 본 장에서는 Hadoop의 MapReduce와 Berkeley 대학의 MapReduce Online에 대하여 살펴보고, 두 기법의 차이점을 분석한다.

3.1 Hadoop의 MapReduce

MapReduce는 인터넷 서비스를 위한 배치 업무들을 빠르게 수행하기 위해서 제안된 분산 병렬 처리 기법으로써, Hadoop의 MapReudce는 Google의 MapReduce와 동일한 구조와 기능을 제공한다. 기존 분산 병렬 처리 기법은 고성능의 컴퓨팅을 요하는 분야의 응용들을 빠르게 처리하기 위한 HPC에 초점을 맞추고 있어 데이터량이 많은 경우엔 적용에 문제가 있다. 따라서 대규모 분산 병렬 처리를 위해서는 데이터 증가에 따른 확장성을 제공할 수 있어야 하며, 노드간 데이터 이동에 따른 네트워크 트래픽을 최소화 할 수 있도록 업무 분산이 필요하다. MapReduce는 이와 같은 요구사항을 고려하여 대용량 데이터 집합을 처리하기 위해 만들어진 기법이다.

MapReduce는 (key, value) 기반의 데이터를 분산 처리하는 모델로, (그림 3)과 같이 입력 데이터 소스를 기반으로 Map 태스크를 수행하여 중간 결과를 생성하고 이를 입력으로 Reduce 태스크를 수행하여 최종 결과를 산출하는 2단계로 구성된다.[11] 이때 개발자는 Map 함수와 Reduce 함수를 정의한다. Map 함수는 입력된 (key, value) 쌍들을 처리하여 중간 값 집합을 생성한다. Reduce 함수는 중간 key값을 가지는 모든 중간값들을 통합하여 최종 출력값으로 저장한다.

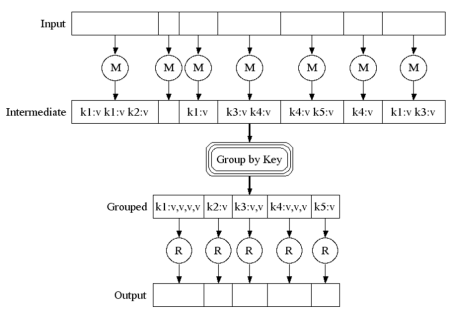


그림 3. MapReduce 실행 모델

MapReduce의 동작 과정은 다음과 같다. 첫째, 네임노드는 입력 파일들을 특정

크기로 분할하고, 분할된 M개의 조각을 클러스터의 데이터노드들에게 할당한다. 둘째, 각 데이터노드(Mapper)들은 할당 받은 Map 태스크에 필요한 조각들을 로드하여, Map 함수를 수행하고 중간 결과값을 저장한다. 셋째, Reduce 태스크를 할당 받은 데이터노드(Reducer)들은 중간 결과값을 로드하여 Reduce 함수를 수행하고, 최종 결과값을 저장한다. 마지막으로 모든 Map과 Reduce 태스크가 완료되면 네임노드는 그 결과를 사용자 프로그램에 전송한다.

(그림 4)는 데이터 파일로부터 각 연도별로 가장 높은 기온을 측정하는 MapReduce 수행 과정의 예를 나타낸다. (그림 4.(a))는 전체 수행 과정을 나타내며, 각 세부 수행 과정은 다음과 같다. 첫째, 하나의 레코드가 라인 단위로 저장된 입력 데이터 파일에서 (offset, record)쌍을 추출한다.(그림4.(b)) 둘째, 각 레코드에서 연도별 기온을 추출하여 (key, value)쌍을 생성한다.(그림 4.(c)) 셋째, 두 번째 Map 과정에서 결과가 많을 경우, Reduce 과정에서 병합시 처리 비용을 감소시키기 위해 각 연도별로 데이터를 그룹화한다.(그림 4.(d)) 넷째, 모든 데이터노드로부터 후보 집합을 병합하여 최종 결과를 반환한다.(그림 4.(e))

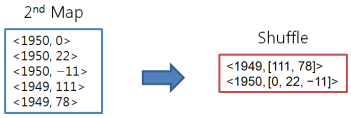
(a) 전체 수행 과정

<Key_1, Value> = <offset, record>
<0, 0067011990999991950051507004...9999999N9+00001+99999999999...>
<106, 0043011990999991950051512004...9999999N9+00221+99999999999...>
<212, 0043011990999991950051518004...9999999N9-00111+99999999999...>
<318, 0043012650999991949032412004...0500001N9+01111+99999999999...>
<424, 0043012650999991949032418004...0500001N9+00781+99999999999...>
...
연도
기온

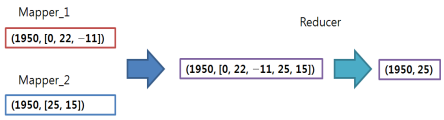
(b) 1st Map 과정

<Key_2, Value> = <year, Temp>
<1950, 0>
<1950, 22>
<1950, −11>
<1949, 111>
<1949, 78>
...

(c) 2st Map 과정



(d) Shuffle 과정



(e) Reduce 과정

그림 4. MapReduce 수행 예제

3.2 Berkeley 대학의 MapReduce Online

MapReduce Online은 실시간 분산 병렬 처리를 지원하기 위해 Berkeley 대학에서 개발한 파이프라인(pipeline) 기반 분산 병렬 처리 기법이다. MapReduce Online은 기본적으로 MapReduce와 동일한 프로그래밍 모델을 제공하며, 파이프라인을 통하여 연산 결과를 실시간적으로 전송함으로써 전체 데이터 처리 시간을 단축시킨다. 아울러 파이프라인을 통한 실시간 데이터 집계와 같은 연속 질의(continuous query)를 지원함으로써 실시간 이벤트 모니터링 및 스트림 처리를 제공한다.

MapReduce Online의 동작 과정은 다음과 같다. 첫째, 네임노드는 입력 파일들을 특정 크기로 분할하고, 분할된 M개의 조각을 클러스터의 Mapper에게 할당한다. 둘째, 각 Mapper는 할당 받은 Map 태스크에 필요한 조각들을 로드하여, Map 함수를 수행하고 중간 결과값을 저장한다. 셋째, Mapper에 저장된 중간 결과값이 일

정 시간 간격으로 파이프라인을 통하여 Reducer에 전송되며, Reducer는 전송된 중간 결과값을 로드하여 Reduce 함수를 수행하고 사용자에게 중간 결과값을 반환한다. 마지막으로 모든 Map과 Reduce 태스크가 완료되면 네임노드는 최종 결과를 사용자 프로그램에 전송한다.

(그림 5)는 데이터 파일로부터 각 월별로 가장 높은 기온을 측정하는 MapReduce Online 수행 과정의 예를 나타낸다. 그림에서 Mapper는 월별 기온을 추출하여 (key, value)쌍을 생성하고 이를 파이프라인을 통하여 Reducer에게 지속적으로 전송한다. Reducer는 전송된 (key, value)쌍에 대하여 병합을 수행하여 임시적인 중간 결과값을 사용자에게 반환한다. 최종적으로 Mapper로부터 모든 (key, value)쌍이 Reducer로 전송될 경우 최종 결과를 사용자에게 반환한다.

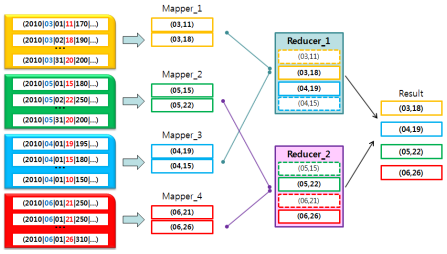


그림 5. MapReduce Online 수행 예제

3.3 MapReduce와 MapReduce Online 비교 분석

(그림 6)은 MapReduce의 데이터 연산 결과 전송 과정을 나타낸다. 우선 네임노드는 각 태스크들의 스케줄을 조정하며, 각 Mapper에 입력 파일들을 특정 크기로 분할하여 할당한다. Mapper는 Map 함수를 수행하고 중간 결과값을 저장하고, Reducer들은 Mapper에서 중간 결과값을 로드하여 Reduce 함수를 수행한다.

(그림 7)은 MapReduce Online의 데이터 연산 결과 전송 과정을 나타낸다. MapReduce와 동일하게 네임노드는 각 태스크들의 스케줄을 조정하며, 각 Mapper에 입력 파일들을 특정 크기로 분할하여 할당한다. 아울러, Mapper와 Reducer사이에 파이프라인을 설정함으로써 연산 결과를 실시간적으로 전송함으로써 실시간 데이터 집계 처리를 지원한다. MapReduce Online에서는 Mapper에 저장된 중간 결과값이 일정 시간 간격으로 파이프라인을 통하여 Reducer에게 전송하며, Reducer는 전송된 중간 결과값을 로드하여 Reduce 함수를 수행하고 사용자에게 중간 결과값을 반환한다. 이를 통해 사용자는 MapReduce에 비해 좀 더 빠르게 결과값을 확인할 수 있을 뿐만 아니라, 최종 결과를 확인하기 이전에 일정 비율의 중간 결과를 미리 확인할 수 있다.

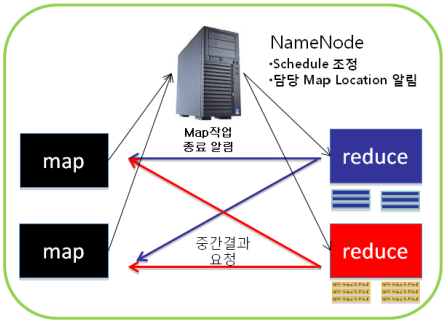


그림 6. MapReduce에서의 데이터 연산 결과 전송

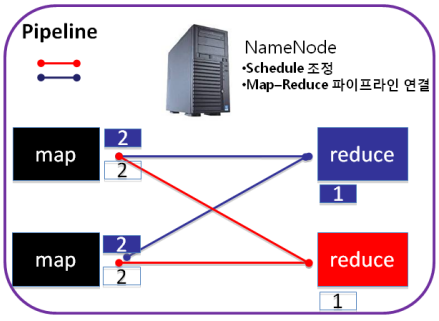


그림 7. MapReduce Online에서의 데이터 연산 결과 전송

4. 성능평가

본 장에서는 Hadoop의 분산 파일 시스템과 MapReduce 및 MapReduce Online을 이용하여 Hadoop 기반 클러스터를 구축하고 이를 통하여 MapReduce와 MapReduce Online의 성능을 비교 분석한다. 이를 위하여 각각 Hadoop-0.20.2 버전과 Hadoop-hop-0.2 버전을 설치하였다.

한편, Hadoop 기반의 클러스터 구축 방식에는 단일구성방식, 가상분산방식, 완전분산방식이 있다. 먼저 단일구성방식은 하나의 노드에서 네임 노드와 데이터 노드를 하나의 자바 프로세스로 실행하는 것이다. 이 방식은 주로 Hadoop 기반의 응용프로그램 디버깅에 유용하다. 둘째, 가상분산방식은 하나의 노드에서 네임노드와 데이터 노드 각각을 가상의 자바 프로세스로 설정하여 실행하는 것이다. 마지막으로 완전분산방식은 TCP/IP 프로토콜로 통신하는 다수의 노드로 하나의 클러스터를 구성하는 방식이다. 본 논문에서는 완전분산방식으로 Hadoop 기반 클러스터를 구축한다.

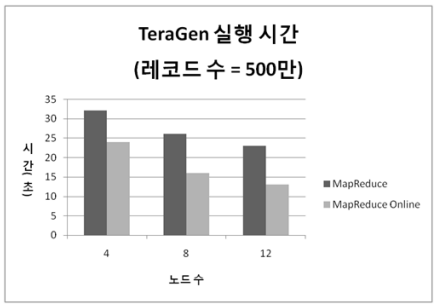
클러스터를 구성하고 있는 각 노드들의 환경은 (표 1)과 같으며, MapReduce 응용프로그램인 TeraGen을 이용하여 데이터를 생성하고 TeraSort를 이용하여 성능평가를 수행한다. TeraGen은 Mapper만을 사용하여 사용자가 입력한 레코드수만큼 임의의 문자열을 생성하여 주는 응용 프로그램이며, TeraSort는 TeraGen에서 생성된 데이터들을 설정된 Mapper와 Reducer만큼 각각 태스크를 생성하여 데이터를 정렬하여 주는 응용 프로그램이다.

표 1. 클러스터 노드 환경

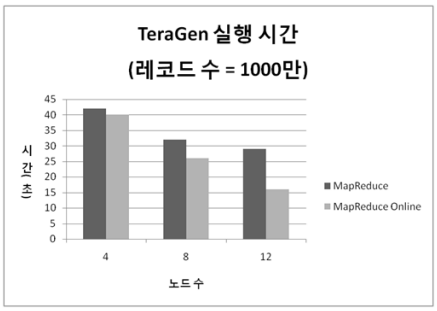
항목	성능
CPU	3400 MHz X 2 CPU
Memory	4GB
OS	Redhat Linux 2.6.9-67.0.7.ELsmp

4.1 노드 수에 따른 데이터 TeraGen 실행 시간

본 절에서는 MapReduce와 MapReduce Online의 노드 수에 따른 TeraGen 실행 시간에 대한 성능평가를 수행한다. 성능평가를 위하여 클러스터의 노드 수를 4, 8, 12로 증가하면서 TeraGen을 사용하여 데이터 생성 시간을 측정하였다. 아울러, 각 클러스터의 노드 수에 따라 Mapper의 수는 3, 7, 11개로 설정하였다.



(a) TeraGen 실행 시간
(레코드 수 = 500만)



(b) TeraGen 실행 시간
(레코드 수 = 1000만)

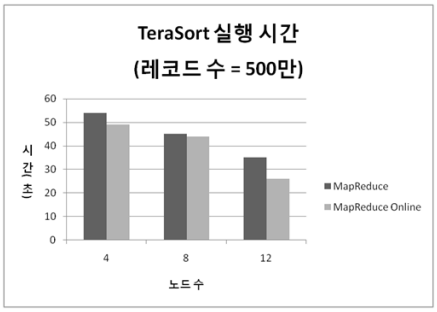
그림 8. 노드 수에 따른 TeraGen 실행 시간

(그림 8)은 레코드 수가 각각 500만개와 1000만개 일 때 노드 수에 따른 TeraGen 실행 시간을 비교한 것이다. 이를 통하여 MapReduce Online은 MapReduce보다 데이터 생성 시간이 최대 45% 감소함을 알 수 있다. 또한, 데이터의 크기가 증가할수록 노드 수 증가에 따른 실행 시간이 크

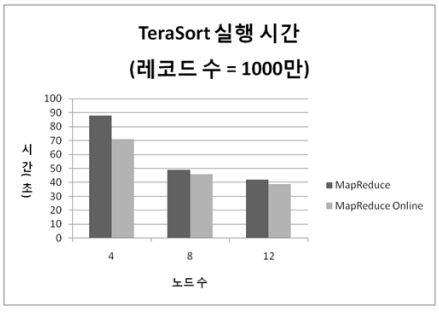
게 감소하는 것을 알 수 있다. 이는 생성할 데이터의 크기가 커짐에 따라 분산 병렬 처리 기법을 통하여 대용량 데이터에 대해 좀 더 효율적으로 관리할 수 있음을 나타낸다.

4.2 노드 수에 따른 데이터 TeraSort 실행 시간

본 절에서는 MapReduce와 MapReduce Online의 노드 수에 따른 TeraSort 실행 시간에 대한 성능평가를 수행한다. 성능평가를 위하여 클러스터의 노드 수를 4, 8, 12로 증가하면서 TeraSort를 사용하여 데이터 정렬 시간을 측정하였다. 아울러, 각 클러스터의 노드 수에 따라 Mapper와 Reducer의 수는 3, 7, 11개로 설정하였다.



(a) 질의처리 시간 (레코드 수 = 500만)



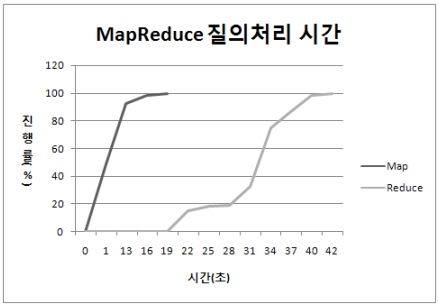
(b) 질의처리 시간 (레코드 수 = 1000만)

그림 9. 노드 수에 따른 TeraSort 실행 시간

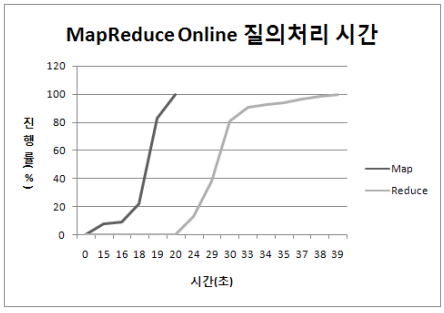
(그림 9)는 레코드 수가 각각 500만개와 1000만개 일 때 노드 수에 따른 TeraSort 실행 시간을 비교한 것이다. 이를 통하여 전반적으로 MapReduce Online의 질의처리 시간이 MapReduce 보다 적게 나타남을 알 수 있으며 최대 26% 감소함을 알 수 있다.

4.3 시간에 따른 질의처리 진행률

본 절에서는 MapReduce와 MapReduce Online의 시간에 따른 질의처리 진행률에 대한 성능평가를 수행한다. 성능평가를 위하여 클러스터의 노드 수와 레코드 수를 각각 12개와 1000만개로 고정하고 TeraSort 질의처리 수행 시간을 측정하였다.



(a) MapReduce 질의처리 진행률



(b) MapReduce Online 질의처리 진행률

그림 10. 시간에 따른 질의처리 진행률

(그림 10)은 시간에 따른 질의처리 진행률을 나타낸다. 그림에서와 같이 MapReduce의 경우 질의처리가 시작된 후 40초 정도 후에야 Reduce 태스크가 완료되어 최종 결과를 확인할 수 있으며, MapReduce Online의 경우 질의처리가 시작된 후 30초 정도 후에 Reduce 태스크

의 80%가 처리됨으로써 보다 빠르게 데이터 집계가 수행되고 있음을 확인할 수 있다. 이는 MapReduce Online의 경우 파이프라인을 통하여 Mapper의 중간 결과를 지속적으로 전송하기 때문이다.

5. 결론 및 향후연구

본 논문에서는 분산 병렬 처리 기법인 Hadoop MapReduce와 MapReduce Online을 비교 분석하였다. 아울러 Hadoop 기반 클러스터를 구축하고, 이를 통하여 MapReduce와 MapReduce Online의 노드 수에 따른 질의처리 시간 및 시간에 따른 질의처리 진행률에 대한 성능평가를 수행하였다. 성능평가 결과 MapReduce Online은 MapReduce에 비해 데이터 생성 질의처리 측면에서 최대 45% 단축되었으며, 데이터 정렬 질의처리 측면에서 최대 26% 단축되었다. 아울러 중간 집계 질의 결과를 지속적으로 전송함으로써 보다 빠르게 데이터 집계를 수행함을 확인할 수 있었다.

향후 연구는 대규모 클러스터를 구축하여 대용량 스트림 데이터에 대한 성능평가를 수행하는 것이다.

참고문헌

[1] 한국소프트웨어진흥원, “클라우드 컴퓨팅의 현재와 미래, 그리고 시장전략“, 2008.

[2] Cloud computing, http://en.wikipedia.org/wiki/Cloudcomputing

[3] Wikipedia, http://en.wikipedia.org/wiki/Christophe_Bisciglia, 2009.

[4] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2, 2007.

[5] IBM Blue Cloud project, http://www04.ibm.com/jct03001c/press/us/en/pressrelease/22613.wss, 2009.

[6] Google App Engine, http://code.google.com/appengine, 2009.

[7] Hadoop MapReduce, http://hadoop.apache.org/mapreduce

[8] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, ““MapReduce Online,” UCB/EECS-2009-136, 2009.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System,“ In Proc. of ACM Symp. on Operating Systems Principles, 2003, pp.20-43.

[10] HDFS, http://hadoop.apache.org/core/docs

[11] Jeffrey Dean and Sanjay Ghemawat, “MapReduce : Simplified Data Processing on Large Clusters,“ Sixth Symp. on Operating System Design and implementation, San Francisco, USA, 2004.