SSD를 사용한 맵리듀스 정렬 성능개선

강석훈*, 강운학*, 이상원*
*성균관대학교 전자전기컴퓨터공학과
x860221@gmail.com, woonagi319@skku.edu, swlee@skku.edu

A Study for Improving MapReduce Performance using Solid State Drive

Seok-Hoon Kang*, Woon-Hak Kang*, Sang-Won Lee*
*Dept of Computer Engineering, SungKyunkwan University

요 약

컴퓨터 메모리의 용량이 커지고 기술이 발전하며 메모리와 저장장치의 데이터 처리속도 차이는 나날이 커지고 있다. 이를 보완하고자 데이터 처리를 가급적 메모리에서 해결하여 처리속도를 높이고자 하는 연구가 많이 있다. 그 중 MapReduce에 대한 연구는 현재 주목이 되고 있는 분야이다. MapReduce는 빅데이터를 클러스터 환경에서 처리하기에 대중적인 프로그래밍 모델이다. 본 논문은 MapReduce 기반의 Hadoop을 SSD를 적용하여 실행속도를 증진시키려 한다. 전통적인 MapReduce 모델은 데이터를 정렬하는데에 I/O가 크게 발생하는데, MapRedce가 사용하는 병합정렬의 I/O 병목현상을 개선하고자 SSD를 사용하였다.

1. 서론

저장장치의 발전과 데이터의 양이 홍수처럼 불어나는 시대에 빅데이터를 효율적으로 처리하는 MapReduce 기반의 Hadoop(Hadoop)에 대한 연구가 활발히 이루어지고 있다. Hadoop은 대량의 자료를 처리할 수 있는 컴퓨터 클러스터에서 동작하는 분산 응용 프로그램을 지원하는 자유자바 소프트웨어 프레임워크이다.

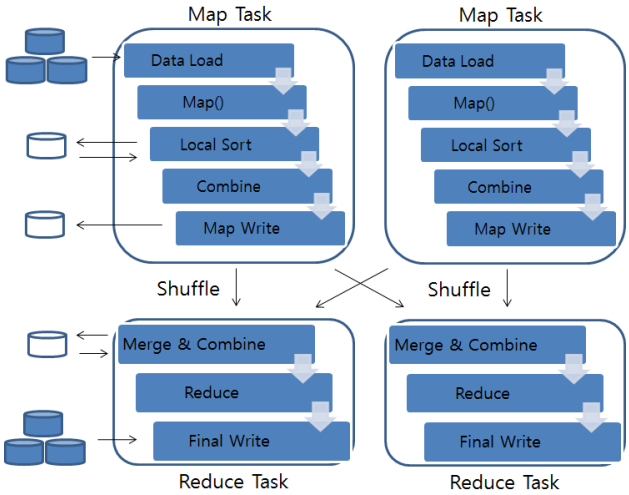
MapReduce 모델이 제안된 이후, Hadoop의 성능을 개선시키는 연구가 활발히 이루어져 왔다. 특히, Map task에서의 내부 정렬(local sort)은 많은 IO를 유발하여 전체 MapReduce 과정의 병목지점으로 나타났으며, 임의 읽기 성능이 뛰어난 SSD를 사용하여 병목현상을 제거할 수 있는 기법이 제안되었다.[1]

SSD(Solid State Disk)는 요즘 부각되고 있는 차세대 대용량 저장장치이다. 기존 저장장치인 HDD와 비교해 규칙적 읽기·쓰기 속도가 빠른 데다 전력 사용량이 적고 충격에 강하며 발열과 소음도 적다. 반면 HDD에 비해 용량당 가격이 비싸다.

본 논문에서는 이전 연구[1]와 마찬가지로 SSD를 사용하여 Hadoop 병합정렬의 성능을 개선하고자 한다. 기존 연구에서는 SSD의 특성을 고려하지 않고 단순히 정렬과정의 일부로 SSD를 사용하였으나, 본 논문에서는 [3]의 결과를 토대로 병합 정렬시 클러스터 사이즈와 버퍼 사이즈를 SSD에 최적화하여 MapReduce 성능을 개선하고자 하였다.

2장에서는 Hadoop의 전체 구조와 특징에 대해 설명하고, 3[1]장에서는 SSD의 특징을 설명하고, SSD에 최적화된 병합정렬을 위해 고려해야할 두 가지 인자에 대해 설명한다. 4장에서는 SSD를 이용해 Hadoop 성능을 테스트한 실험에 대해 설명한다. 마지막으로 5장에서는 결론과 향후연구로 논문을 마무리 한다.

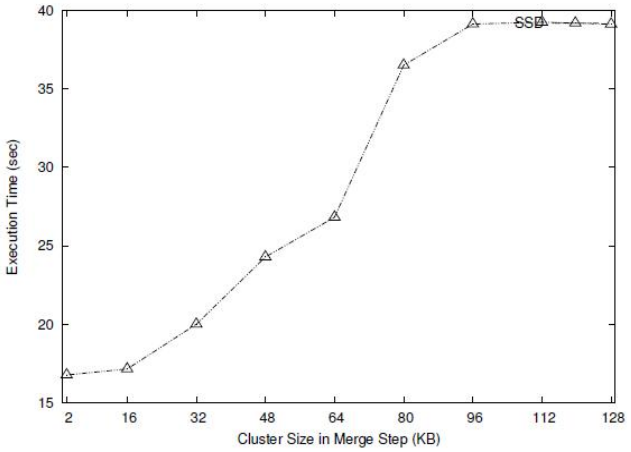
2. Hadoop의 구조



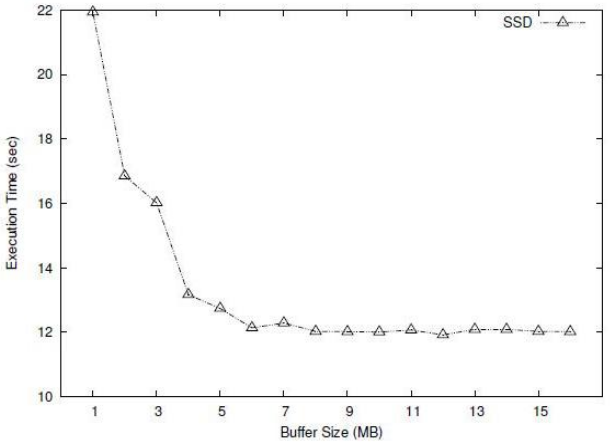
(그림 1) Hadoop의 구조

Hadoop은 대용량 데이터 분석을 위한 큰 규모의 분산 컴퓨팅 환경을 지원하는 프레임 워크이다. 원래 너치의 분

1) 국문 : 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10041244, 스마트TV 2.0 소프트웨어 플랫폼]



(그림 2) 클러스터 사이즈 조정에 따른 수행시간 변화



(그림 3) 버퍼 크기 변화에 따른 수행시간 변화

산처리를 지원하기 위해 개발된 것으로, 아파치 루씬의 하부 프로젝트이다[2]. 분산처리 시스템인 구글 파일 시스템을 대체할 수 있는 Hadoop 분산 파일 시스템(HDFS: Hadoop Distributed File System)과 맵리듀스를 구현한 것이다. MapReduce는 인덱스(Index)역할을 하는 Key와 그에 대한 값(Value)으로 나누는 Map task와 데이터를 사용자의 목적에 맞게 처리하는 Reduce task로 나뉜다. 그리고 Map task가 Reduce task로 넘어가는 과정에서 반드시 병합 정렬 과정을 거친다.

그림 1은 Hadoop의 Map과 Reduce작업이 진행되는 과정을 표현한 것이다. Hadoop은 병렬처리를 지원하는 시스템이기 때문에 먼저 데이터크기를 적당한 크기로 나누어 각 노드에게 처리될 데이터를 나누어준다. 그 후 Hadoop의 데이터 처리순서는 데이터가 로드가 되고 Map함수에서 <키, 값>의 쌍으로 분류하는 작업을 한다. 그 후에 데이터가 처리되기 쉽게 중간에 정렬작업이 진행된다. 각 정렬된 데이터 집합들은 병합(combine)이 되고 메모리에 쓰여지면서 Map 작업이 종료가 된다. Map의 결과물은 Reduce작업으로 넘어가기 전에 로컬 디스크에 저장이 된다.

Reduce작업은 Reduce 함수가 진행되기 전에 각 노드들의 map 결과를 받아서 병합하는 전처리과정을 거친다. 그리고 Reduce 함수에서 사용자 목적에 따른 연산을 마치고 최종적 결과를 디스크에 저장한다.

이 때 Hadoop 사용의 목적상 큰 데이터를 병합 정렬하는 부분에서 정렬되는 데이터들이 메모리의 용량을 초과하면 외부정렬을 하게 되면서 저장장치로 I/O가 발생한다. I/O 시간은 메모리에서의 데이터 처리 시간보다 매우 느리기 때문에 Hadoop의 전체 수행시간은 병합 정렬의 성능에 많은 영향을 받는다.

3. 관련연구 : SSD를 사용한 해시조인 성능개선

Sigmod'08 [3]에 발표된 HDD와 SSD의 성능비교 논문에서 기존 DB들의 병합정렬 알고리즘은 디스크를 기반으로 하여 클러스터 사이즈가 일반적으로 크게 설정되어 있고, 클러스터 사이즈를 SSD에 최적화하여 외부 병합 정렬 성능을 크게 개선 할 수 있었다. 아래 그림 2는 SSD에서 외부정렬을 하는 쿼리의 실행시간을 측정한 그림이다. 오라클의 Sqlplus 툴을 이용해 임의로 생산한 테이블을 조인하고 count하는 쿼리를 만들었다. 테이블에는 외부정렬이 일어날 수 있도록 충분히 큰 데이터를 입력했고 각 테이블을 조인하고 count하면서 발생되는 IO를 고려하였다. 발생하는 IO와 관련된 클러스터와 버퍼 사이즈를 단계별로 조정하며 각 크기마다 총 수행시간을 측정하여 그래프로 비교하였다.

쿼리 실행 시 I/O하는 클러스터의 크기별로 실행시간을 측정한 결과 2KB에서 가장 좋은 성능을 보였다. 이 때 버퍼 크기는 2MB로 고정하였다.

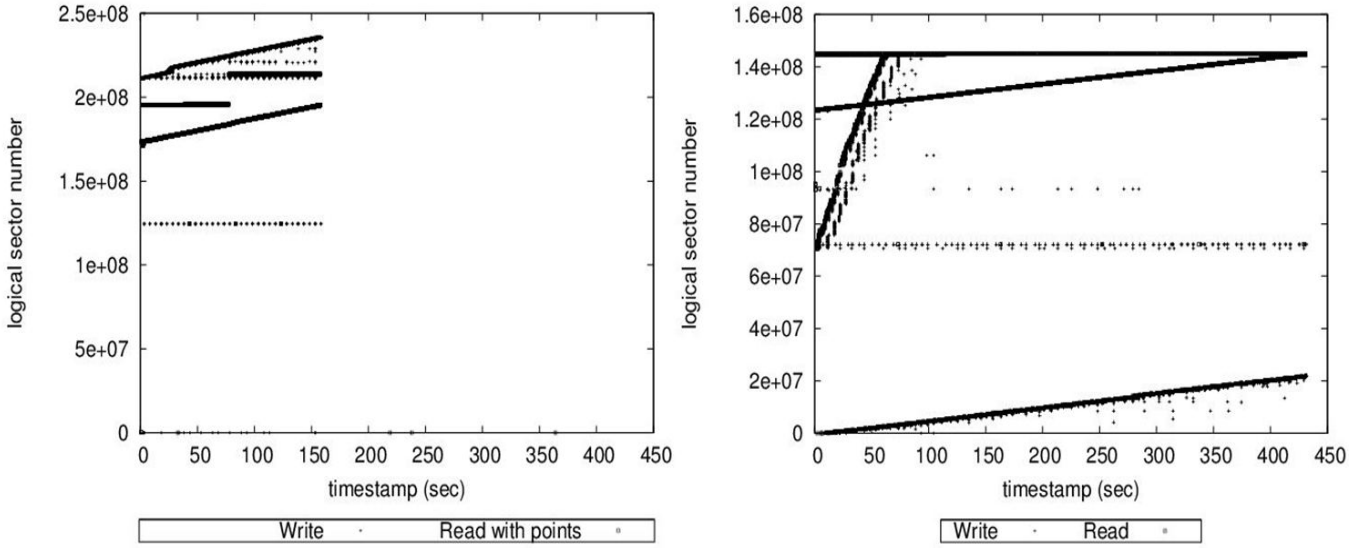
SSD의 경우 큰 클러스터 값으로 인한 대역폭(bandwidth) 증가는 병합(merge) 시간과 비교하여 성능에 효율적이지 못했다. 대역폭이 클수록 그에 따른 병합하는 시간도 상대적으로 증가하게 된다. 그래서 결과적으로 대역폭증가로 줄어드는 실행시간보다 병합시간의 증가로 늘어나는 실행시간이 더 크기 때문에 작은 클러스터를 사용하는 것이 시간을 크게 단축시켰다.

그림 3은 Sort에 사용되는 버퍼 크기를 조정한 것이다. 그림과 같이 9MB이상부터는 결과에 큰 차이가 없었다. 이 때 클러스터 크기는 2KB로 고정하였다.

버퍼 사이즈가 클수록 버퍼안에서 처리되는 데이터 처리량이 늘어나게 되고 그로 인해 IO 횟수가 줄어 IO에 소모되는 시간과 함께 총 실행시간이 짧아진다. 그러나 9MB 이후부터는 버퍼안의 처리되는 데이터 크기에 비해 버퍼의 크기가 충분히 커서 시간에 영향을 크게 미치지 않다는 것을 알 수 있다.

4. SSD를 이용한 Hadoop 성능 테스트

Hadoop 시스템에서 HDD와 SSD의 성능을 비교하기 위해 기존 Hadoop에 있는 예제인 *Sort*를 benchmark 하여 실험하였다. Intel(R) Core CPU 3.30GHz, 리눅스 Fedora14,



(그림 4) HDD(좌) vs SSD(우) : 10GB 정렬 (WRITE-점, READ-선)

Hadoop 0.20.2, Sun Java 1.6u31에 Western Digital HDD 74GB 와 Samsung SSD 830 Series 128GB를 설치하였고 테스트와 디버그가 편한 단일노드 환경에서 실험을 하였다.
*Sort*예제는 *Randomwriter*라는 Hadoop의 다른 예제의 결과물을 정렬하는 예제이다. 먼저 *Randomwriter*가 사용자가 지정한 크기만큼 이진파일을 무작위로 만들어낸다. 본 논문은 1, 10, 30GB를 만들어냈다. 그리고 *Sort*예제가 이 이진파일을 <키,값>으로 나눈 후 이름순으로 정렬을 한다. 결과적으로 MapReduce로 binary파일을 정렬하는 *Sort*예제로 각 1GB, 10GB, 30GB의 파일을 정렬하였다.
위 그림 4는 10GB를 정렬한 것의 I/O를 나타낸 결과이다. 그림의 왼쪽이 HDD로 *Sort*예제를 돌렸을 경우, 오른쪽이 SSD로 돌렸을 경우의 결과이다. 각 장치로 실행시간동안 Read와 Write한 것을 그래프로 표현하여 IO발생을 한눈에 알 수 있었다.
나머지 1GB와 30GB 파일을 *Sort*예제로 실험했을 때 결과시간을 표 1에 나타내었다.
표 1을 봤을 때 HDD와 SSD의 정렬완료 시간은 3배 가량 차이나는 것을 알 수 있다.

	HDD	SSD
1GB_SORT	40s	15s
10GB_SORT	424s	136s
30GB_SORT	1195s	403s

(표 1) HDD vs SSD : 실행완료시간

5. 결론 및 향후 연구

Hadoop을 실행하면서 Hadoop의 정렬부분의 I/O문제는 SSD를 사용함으로 더 나은 성능을 보여준 것을 확인하였고 기존 HDD를 사용했을때보다 3배정도 빠른 것을 알 수 있었다.
본 논문에서는 위 실험을 바탕으로 Sort-merge에 사용되는 클러스터와 버퍼 크기값을 변화시켜 Hadoop에 SSD를 최적화하는 것이 최종목표이다.
3장의 실험처럼[3] Hadoop내의 Parameter조정으로 SSD의 클러스터, 버퍼값을 단계별로 조정하여 I/O 실행시간이 가장 단축되고 성능이 가장 빠른 단계를 검색하는 것이 향후 연구방향이다.

참고문헌

[1] B. Li, E. Mazur, et al. A Platform for Scalable One-Pass Analytics using
MapReduce. In SIGMOD'11, June12-16, Athens, Greece, 2011.

[2] Hadoop Guide.
http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/ .

[3] Sang-Won.L, Bongki.M, et al. A Case for Flash Memory SSD in Enter-
prise Database Applications. In SIGMOD'08, June9-12, Vancouver, BC, Canada, 2008.

[4] Tom white, Hadoop Perfect Guide.

[5] Sayantan.S, Hao.W, Jian.H, Xiangyong .O and Dhabaleswar K. Panda. Can High-Performance Interconnects Benefit Hadoop Distributed File System?

[6] Jonas Schmid. Implementation and Evaluation of a Key-Value Store for Flash-based Storage