SSD를 가상메모리로 활용한 In-Memory System 성능 평가

권대규*, 박성민**, 강수용*
*한양대학교 컴퓨터공학부
e-mail:silence013@hanyang.ac.kr

A Performance of In-Memory Systems with SSD-based virtual memory

Dai-Kyu Kwon*, Sung-Min Park**, Soo-Yong Kang*
*Division of Computer Science Engineering, Hanyang University

요 약

최근 빅데이터의 활용에 대한 활발한 논의가 이루어지고 있는 가운데 데이터를 빠르게 처리할 수 있는 인메모리 시스템(In-memory System)에 대한 많은 연구가 진행되고 있다. 하지만 메모리만으로 인메모리 시스템을 구성할 경우 많은 비용과 전력을 필요로 한다. 이와 같은 문제점을 개선하기 위하여 SSD를 가상메모리로 활용하는 방법에 대한 연구가 진행되고 있다. 본 논문에서는 SSD를 가상 메모리로 활용하여 인메모리 어플리케이션의 성능을 측정하였다. 성능 평가 결과, SSD를 가상 메모리로 활용했을 때, RAM 사용량에 비례하여 성능 하락을 보였다. 하지만, 앞으로 SSD의 성능이 개선되고 발전할 경우 메모리를 SSD로 대체하여 사용하면 비용의 절감뿐 아니라 성능을 유지할 수 있을 것으로 기대된다.

1. 서론

최근 SNS와 Mobile-device의 발전을 통해 많은 변화가 발생했는데, 그 중 한 가지가 데이터의 급증이다. 개인뿐 아니라 기업들이 보유해야하고 처리해야 하는 데이터의 양은 엄청나게 증가하고 있다.

이러한 빅데이터를 처리하기이 위하여 각광받고 있는 분야가 인메모리 컴퓨팅(In-memory computing) 기술이다. 인메모리 컴퓨팅 기술은 모든 데이터의 처리를 메모리 안에서 처리하는 방법이다[1]. 인메모리 컴퓨팅 방식으로 인하여 이전 디스크 기반의 처리 방식에 비해 수십 배, 수백 배의 성능 향상을 이끌어 낸다. <표 1>은 HDD, SSD, DRAM 디바이스의 특징을 보여준다. DRAM은 HDD와 SSD에 비하여 접근 시간은 매우 빠르지만 가격이 비싸고 전력을 많이 사용한다. 따라서 DRAM으로 대용량 인메모리 컴퓨팅을 구성할 경우 구축 비용이 비싸고 전력을 많이 사용하는 단점이 있다.

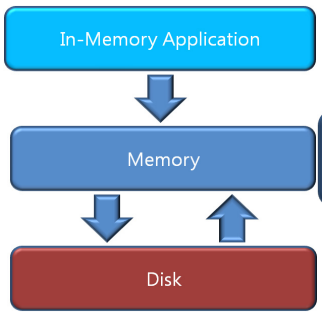
인메모리 컴퓨팅의 이러한 단점을 극복하기 위하여 SSD를 활용하는 방법에 대한 연구가 활발히 진행되고 있다[2][3][4]. SSD란 낸드 플래시 비휘발성 메모리를 병렬로 구성한 저장 장치이다. <표1>에서 알 수 있듯이, SSD는 기존의 HDD에 비해 전력 소비, 대역폭, 속도 등에서 뛰어난 성능을 가지고 있다.

본 논문에서는 SSD에 운영체제의 swap 파티션을 설정하여 인메모리로 컴퓨팅에서 SSD를 가상 메모리로 사용한다.

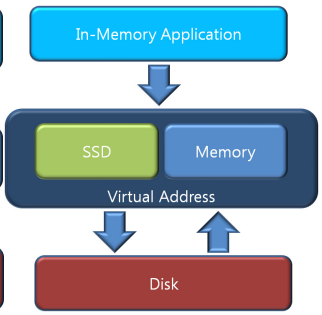
장치	지연시간		대역폭 (GB/s)		파워 (W)		용량 (GB)	가격 (₩/GB)
	읽기	쓰기	읽기	쓰기	유휴	활성		
Samsung HDD	9ms	9ms	0.25	0.3	6.4	7.4	1,024	122
Samsung DDR3 DRAM	14ns	14ns	13	13	-	3.7	4	12,625
Samsung SSD	-	-	0.52	0.4	0.078	0.127	512	1,914

<표 1> 디바이스의 특징

2. 본론



<그림1> RAM만을 활용한 In-Memory ComPuting



<그림2> RAM과 SSD를 활용한 In-Memory Computing

<그림 1> <그림2> 는 인메모리 컴퓨팅을 활용하여 데이터를 처리하는 형태를 나타내고 있다. <그림1>은 온전히 RAM만을 사용하여 데이터를 처리하는 방식으로서 메모리에 굉장히 많은 비용을 필요로 한다. <그림2>는

RAM뿐 아니라 SSD를 메모리처럼 사용하여 사용하는 경우를 나타낸다.

SSD를 메모리로 사용하는 방법은 여러 가지가 있다. 그 중에서도 스왑(Swap)을 통해 쉽게 스토리지(Storage)를 메모리로 사용할 수 있다. 스왑은 SSD를 메모리로 활용하는 방법 중 하나로, 리눅스 환경에서 간단히 설정할 수 있다. 사용할 메모리의 양을 필요에 맞게 설정하여 SSD를 쉽게 메모리로 스왑하여 사용할 수 있다.

이 외에도 SSDalloc[3], NVMalloc[4] 등의 SSD를 메모리화 하여 사용할 수 있는 많은 방법들이 있으나, 대다수가 공개가 되어있지 않아 쉽게 환경을 구축하기 어려운 상황이다. 이러한 상황에 따라, 이 성능평가에서는 리눅스를 활용하여 SSD를 메모리로 스왑하여 사용하는 형태로 성능을 평가하고자 한다.

3. 성능 평가

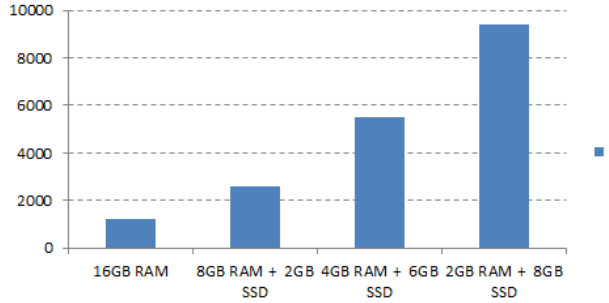
이번 성능평가는 인텔 i3, 삼성 DDR3 RAM 16GB, 삼성 128GB SSD를 사용하였고, 운영체제는 리눅스 2.6.18 버전을 사용하여 성능평가를 진행하였다. 실험은 리눅스에서 SSD를 메모리로 스왑하여 진행하였다. 사용하는 툴은 스파크(Spark)라는 하둡(Hadoop)을 활용한 프레임워크를 사용하였다. 스파크는 오픈 소스로서, 클러스터 컴퓨팅 시스템으로 데이터의 분석을 빠르게 하는데 목적을 가진 툴이다[5]. 실험은 두 가지의 알고리즘으로 진행되었는데, 로지스틱회귀분석(Logistic Regression) 알고리즘과 워드카운트(Word Count)알고리즘으로 실험을 진행하였다. 또한 실험 데이터의 크기는 10GB로 설정하여 진행하였다. 워드카운트 알고리즘은 <그림 3>과 같이 임의의 텍스트 파일을 받아서 단어의 개수를 세는 알고리즘이다.

```
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

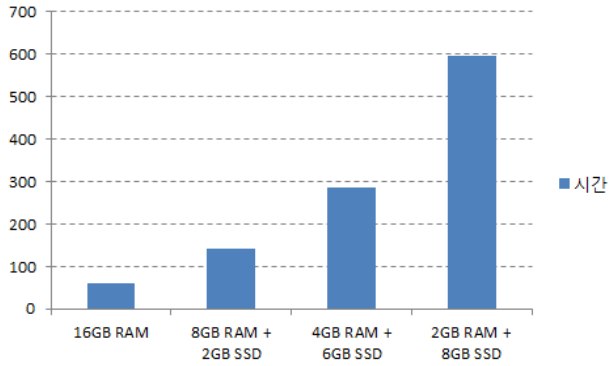
<그림 3> 워드 카운트 알고리즘

```
val points = spark.textFile(...).map(parsePoint) .cache()
var w = Vector.random(D) // current separating plane
for (i <- 1 to ITERATIONS) {
  val gradient = points.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}
println("Final separating plane: " + w)
```

<그림 4> 로지스틱 회귀분석 알고리즘



<그림5> 워드카운트 알고리즘 평가 결과



<그림6> 로지스틱 회귀분석 알고리즘 평가 결과

<그림 4>에서 알 수 있듯이, 로지스틱 회귀분석 알고리즘은 데이터를 받아 이 데이터를 두 종류로 분류하는 알고리즘으로 반복의 횟수가 진행될수록 데이터를 필터링하여 메모리에 부하를 줄이는 알고리즘이다.

실험 조건은 4가지의 방법으로 구성하였는데, 16GB 메모리만을 활용하는 방법, 8GB 메모리와 8GB SSD를 이용한 방법, 4GB 메모리와 12GB SSD를 이용한 방법, 2GB 메모리와 14GB SSD를 이용한 방법의 총 네 가지 조건으로 실험을 진행하였다.

워드 카운트 알고리즘으로 실험을 진행할 경우 <그림 5> 와 같은 RAM의 사용량에 따라 비례하여 감소하는 결과를 얻을 수 있었다. 처음 실험은 오로지 RAM만을 메모리로 사용하여 데이터를 처리하는 실험의 시간 결과이다. 사용하는 RAM을 반으로 줄이고 SSD를 추가하여 메모리로 사용하는 경우 시간은 2배가 더 걸린다. 또한 RAM을 1/4로 줄이면 시간이 4배 정도 더 걸린다. 마찬가지로 RAM을 1/8로 줄이면 시간이 8배 정도 더 걸린다. 로지스틱 회귀분석 알고리즘으로 실험을 진행할 경우도 <그림 6>에서 알 수 있듯이, 첫 번째 실험과 비슷한 결과를 얻었음을 알 수 있다. 위의 실험 결과와 <표 1>의 내용을 가지고 살펴보면, SSD를 사용하는 경우 시간은 2배, 4배, 8배가 걸리는 반면에 비용 문제에서는 10배의 차이가 나기 때문에 시간은 더 걸리지만 비용은 실험의 방법에 따라 1/10, 1/20, 1/40을 절약하는 효과를 보이는 것을 알 수 있다.

4. 결론

위의 두 가지 방법으로 실험을 진행한 결과, 아직까지 메모리만을 활용하는 방법에 비해 성능적인 측면으로는 부족한 면이 있었다. 그럼에도 불구하고, 성능의 하락에 비해 많은 비용을 절감할 수 있을 것이라고 사료된다. 이 결과를 통해, 인메모리 컴퓨팅 시스템이 어떠한 방향으로 나아가야할지 확인할 수 있는 성능평가임을 알 수 있었다. 최근, 새로 개발된 PCI SSD의 경우 위의 성능평가에서 활용한 SSD SATA3 보다 4배 우수한 성능을 보인다. 앞으로 발전된 SSD를 활용할 경우 비용이 많이 들게 되는 메모리만으로 데이터를 처리하는 것이 아닌 SSD를 메모리처럼 활용한 메모리 처리 방식도 활용될 수 있을 것이다.

참고문헌

[1] Massini Pezzini, The Next Generation Architecture: In-Memory Computing, SAP Innovation Forum 2012

[2] M. Saxena and M. M. Swift. Flashvm: Virtual memory management on flash. In Proc. USENIX Annual Technical Conference, Boston, MA, June 2010.

[3] A. Badam and V. S. Pai. SSDAlloc: Hybrid SSD/RAM Memory Management Made Easy. NSDI, 2011.

[4] Chao Wang, Sudharshan S. Vazhkudai, Xiaosong. NVMalloc: Exposing an Aggregate SSD Store as a Memory Partition in Extreme-Scale Machines. IPDPS'12, Shanghai, China, May 2012.

[5] Spark: Cluster Computing with Working Sets. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. HotCloud 2010. June 2010.