HPC 환경에서의 오픈소스 아파치 스파크 시스템 성능 측정 및 분석

전진우[0] 임성수
국민대학교 컴퓨터공학부
chichicapo00@gmail.com, sslim@kookmin.ac.kr

An Analysis on Apache Spark System for High Performance Computer Server

Jinwoo Jeon[0] Sung-Soo Lim
School of Computer Science Kookmin University

요 약

현대에는 빅데이터(BigData) 처리가 중요해지면서, 빅데이터를 처리하기 위해 하둡(Hadoop)과 스파크(Spark)같은 분산 처리 프레임워크가 분산 처리 환경(scale-out)에서 많이 사용된다. 하지만 최근 기술이 발달하여 코어수가 증가함에 따라 미래에는 고성능 컴퓨터가 보편화 될 수 있다. 고성능 컴퓨터가 보편화 된다면, 여러 대의 분산 처리 환경보다 한 대의 고성능 컴퓨터 환경이 데이터 처리에 있어 효율이 더 좋을 수 있다. 본 논문은 120코어 기반의 매니코어 서버를 기반으로 4개의 스파크 벤치마크 워크로드를 수행해 HPC 환경에서의 스파크 성능을 측정하고, 확장성(scalability) 문제를 발견하였다. 향후 연구로 HPC 환경에서도 스파크 시스템이 확장성을 갖도록 해결하는 연구를 해야 할 필요가 있다.

1. 서 론

오늘날 컴퓨터 기술과 인터넷 통신의 발달로 인해 자연스럽게 데이터량이 방대해졌다. 이러한 데이터는 얼마나 큰지, 얼마나 빨리 생성되는지, 구조화/비구조화 데이터인지 등을 고려해 빅데이터라고 한다. 현대에서는 빅데이터를 이용해 새로운 가치를 만들 수 있기 때문에 중요해지고 있다.

빅데이터를 관리하고 처리하는 대표적 오픈소스는 하둡(Hadoop)이다. 하둡은 많은 양의 데이터를 비교적 쉽게 처리할 수 있는 자바 소프트웨어 프레임워크로, 분산 파일 시스템인 HDFS와 맵리듀스(MapReduce)[1]로 구성되어 있다. 그러나 맵리듀스는 데이터를 처리할 때 디스크를 사용하여 느리다는 단점이 있다, 이 단점을 보완하기 위해 메모리를 기반으로 하여 데이터 처리를 하는 아파치 스파크(Apache Spark)가 제안되었다[2].

스파크는 많은 컴퓨팅 환경이 네트워크로 연결되어 있는 환경인 scale-out 환경에서 많이 쓰이고 있다. 이러한 scale-out 환경을 사용하는 이유는 고가의 컴퓨터 환경을 구성하지 않아도 적당한 가격의 컴퓨터 환경을 여러 대 구성하여 더 좋은 효율을 낼 수 있기 때문이다.

그러나 최근 코어수가 증가함에 따라 HPC(High-Performance Computing) 시스템 환경인 scale-up 환경에서도 스파크 시스템의 성능에 대한 연구가 필요해지고 있다. 이유는 코어 수가 많은 고성능의 컴퓨터가 보편화되기 시작한다면, 코어 수가 적은 서버 여러 대를 구성해서 데이터를 처리하는 것보다 고성능 컴퓨터 몇 대를 두고 처리하는 것이 더 효율적일 수 있다. 하지만 아직 HPC 환경에서의 스파크 시스템에 대한 연구가 부족한 것이 문제이다.

이러한 문제를 해결하기 위해, 본 논문은 분산 처리 클러스터 시

스템의 자원을 HPC 시스템에서 구성했을 때, 이에 대한 성능(Performance)과 확장성(scalability)에 대한 문제를 분석하였다. 본 연구에서는 120코어로 구성된 매니코어 시스템을 위에 분산 파일 시스템(HDFS)과 스파크 시스템(Master/Worker)[2]을 한 시스템에서 구성하였다. 성능을 측정하는 다양한 벤치마크에 대해 분석을 수행한 결과 모든 테스트 대상 워크로드에서 확장성 문제가 존재함을 확인하였고, 문제점을 분석하기 위해 CPU 사용량을 측정하고 자바 가상 머신의 Garbage Collection, NUMA 메모리 구조, 운영체제 shard memory system의 공유 문제를 살펴보았다.

본 논문은 다음과 같이 구성된다. 2장은 HPC 환경에서의 스파크 시스템 관련 연구를 설명하며, 3장에서는 벤치마크 소개와 측정 방법 및 성능 측정, 측정 결과에 대해서 설명한다. 4장에서는 성능 측정을 통하여 확인된 문제점에 대해 분석을 수행하고 5장에서는 결론 및 향후 연구에 대해 설명한다.

2. 관련 연구

scale-out 환경이란 서버의 수를 증가시켜 처리능력을 향상시키는 분산 처리 환경을 말하고, scale-up 환경이란 CPU, 메모리, 디스크등 하드웨어의 성능을 높여 처리능력을 향상시키

* 이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. B010-16-0644, 매니코어 기반 초고성능 스케일러블 OS 기초연구)

는 환경을 말한다. 최근 코어가 늘어남에 따라 scale-up 환경의 HPC 시스템 환경에서 스파크 시스템에 대한 확장성 연구는 기존에 몇 가지 연구가 있었다. HPC 시스템에서 스파크를 사용해 분산처리를 할 때, 스파크에서 사용하는 JVM의 GC(Garbage Collection)의 종류에 따라 성능 차이가 있음을 발견하고 성능이 제일 잘 나오는 GC를 찾아 확장성 문제를 해결하는 연구[3]가 있었고, HPC 시스템 환경에서 스파크의 확장성을 파일시스템 관점으로 한 연구[4]도 있다.

본 논문에서는 120코어로 구성된 매니코어 시스템에서 스파크의 성능 측정을 위해 기존 연구를 참고하여 GC의 종류를 두 가지로 설정하고 4개의 벤치마크 워크로드를 실험하였다.

3. 방법론

성능 측정을 위한 벤치마크로 BigDataBench의 Big Data Generator Suite(BDGS)[5]를 사용한다.

BDGS는 오픈소스 도구로써, 다양한 워크로드를 제공하고 사용자가 원하는 만큼의 데이터 크기를 생성할 수 있다. 스파크는 기본적으로 자바 가상 머신(Java Virtual Machine)에서 위에서 동작하므로, 모든 실험에서 JVM의 힙(heap) 메모리는 4GB로 설정하였다.

3.1 벤치마크

벤치마크	데이터 타입	데이터 셋	크기(GB)
Word Count	Text	Wikipedia Entries	10
Grep	Text	Wikipedia Entries	30
Naive bayes	Text	Amazon Movie Review	10
K-means	Graph	FaceBook Social NetWork	4

표 1. 벤치마크

- **Word Count**: 텍스트 파일에서 각각의 단어가 나오는 숫자를 카운트 한다.
- **Grep**: 텍스트 파일에서 키워드를 검색하고 출력 파일에 일치하는 문자열 라인을 걸러낸다. 이 실험에서는 키워드를 “The”로 실험하였다.
- **Naive Bayes**: 머신러닝 알고리즘 중에서 분류 알고리즘으로 많이 사용되는 알고리즘으로, Amazon Movie Review에서 긍정적인 또는 부정적인 리뷰를 확인한다.
- **K-means**: 주어진 데이터를 k개의 클러스터로 묶으면서 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작한다. 본 논문에서는 8개의 클러스터, 1번 반복으로 설정하였다.

3.2 시스템 설정

본 논문에서 사용된 IBM x3950 x6 시스템이다. 120코어의 Intel Xeon E7-8870 시스템을 사용하였으며, NUMA(Non-Unifrom Memory Access) 메모리 구조를 가지고 있다.

CPU	Intel Xeon E7-8870
코어 수	120코어
메모리	792GB
NUMA 노드 수	8 소켓(소켓당 15코어)
운영체제	Ubuntu 14.04.4 LTS
JVM	openjdk version 1.8.0_91
Hadoop	Version 1.2.1
Spark	Version 1.3.1

표 2. 시스템 사양 및 소프트웨어 버전

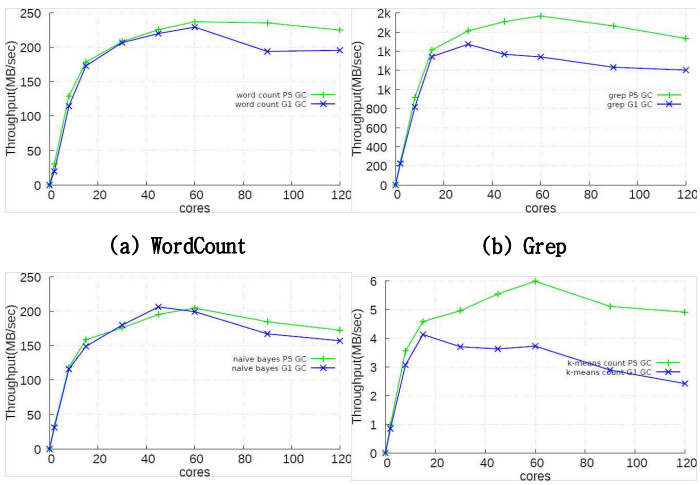
3.3 성능 측정

실험환경으로는 하둡의 분산 파일 시스템(HDFS)을 저장소로 스파크를 분산 처리 시스템으로 사용하였다. 이 논문에서는 벤치마크에서 지원하는 하둡 1.2.1 버전, 스파크 1.3.1 버전을 사용한다.

하둡과 스파크는 마스터/슬레이브 구조로써 마스터 노드와 슬레이브 노드가 있어 분산처리를 한다. 그러나 성능 측정 실험에서는 HPC 시스템의 확장성을 실험하기 위해 하둡/스파크의 모든 데몬(java process)들을 한 시스템에 구성한다.

스파크[2]는 하나의 노드에서 할당받은 작업(Task)을 처리하는 Worker의 수를 설정할 수 있다. 실험하기 전에 미리 HPC 환경에서 여러 개의 Worker를 설정하여 성능을 측정 해본 결과, 하나의 Worker일 때와 성능차이가 눈에 띄지 않았다. 따라서 본 논문에서는 하나의 Worker를 설정하고 실험하였다.

또한 자바 가상 머신 GC(Garbage Collection)의 종류에 따른 성능도 확인하기 위해 Parallel GC, G1 GC 두 종류의 GC를 설정하여 실험하였다. 하둡의 저장소는 디스크 입출력 병목 현상을 줄이기 위해 메모리를 사용하는 임시 파일 시스템(tmpfs)을 사용하였다.



(a) WordCount (b) Grep

(c) Naive Bayes (d) K-means

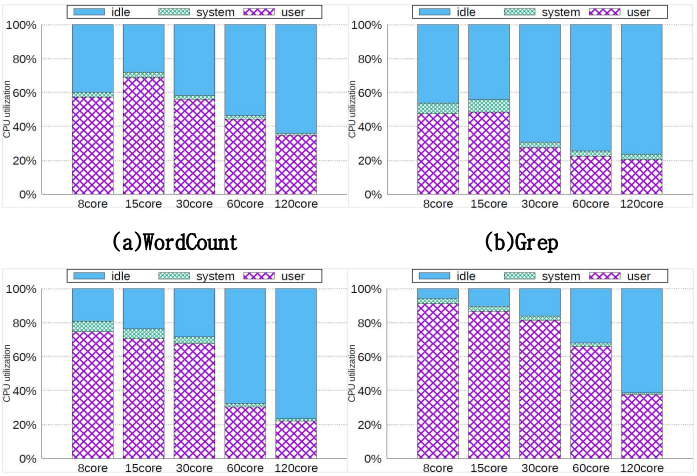
그림 1. MB/s 당 처리량

그림 1.은 워크로드 각각의 처리량을 그래프로 표시한 그림이다. 코어의 수를 2, 8, 15, 30, 45, 60, 120으로 껐다, 켜면서 실험을 하였다. 실험한 4가지 워크로드 모두 60코어에서 가장 높은 처리량(MB/s)을 보였다. 또한 GC 종류에 따른 워크로드의 처리량을 보았을 때, Parrallel GC의 처리량이 더 높게 나왔다. 이를 통해 GC에 따른 성능차이가 있음을 확인하였다.

코어 수가 늘어남에 따라 처리량도 올라가면서 그래프가 일정하게 비례하여 증가하는 것을 기대했지만, 60코어에서 코어수가 더 늘어나자 처리량이 떨어지는 것을 볼 수 있다. 따라서 GC 종류에 따른 성능 차이는 있지만, 모든 벤치마크 워크로드가 확장성 문제를 가진다.

4. 문제점 분석

4.1 CPU 사용량 측정



(a)WordCount (b)Grep

(c) NaiveBayes (d) K-means

그림 2. CPU 사용량 분석

그림 2. 는 8, 15, 30, 60, 120 코어 마다 워크로드를 수행했을 때 CPU 사용량의 비율을 그래프로 나타낸 것이다.

user mode, system mode, idle의 실행시간을 측정한 결과, 워크로드 전체적으로 코어수가 늘어날 때 마다 CPU 대기시간인 idle이 높은 비율을 보였다. 스파크는 단일 노드(single node)로 동작하는 시스템의 확장성 특성을 고려하지 않았기 때문에 코어 수가 많아져도 CPU를 다 사용하지 못하는 확장성 문제가 있는 것을 볼 수 있다.

4.2 확장성(Scalability) 저해 요소

HPC 환경에서 확장성을 저해하는 요소로는 먼저 JVM(Java Virtual Machine)의 GC(Garbage Collection)에 의한 문제가 있다. JVM에서 GC가 호출되면, stop-the-world 현상이 발생한다. 자바에서 stop-the-world란 GC가 수행이 되면 JVM의 모든 쓰레드들이 멈췄다가 GC의 수행 완료 후 다시 시작되는 것을 말한다. 따라서 코어가 많아질수록 GC가 수행되는 동안 영향을 받는 코어들이 늘어나 오버헤드가 발생하여 처리량이 떨어진다. 두 번째는 NUMA(Non-Unifrom Memory Access) 메모리 구조에 의한 오버헤드이다. 본 논문에서 사용된 시스템은 NUMA 메모리 구조로 코어마다 접근하는 메모리 지역이 다르다. 따라서 메모리 접근 지연 시간(memory access latency)에 의해 확장성 문제가 나타난다고 볼 수 있다. 마지막으로 운영체제 자체의 저해요소를 생각할 수 있다. 분산 처리 시스템인 스파크를 하나의 단일 노드(single)로 구성하면 공유 메모리 시스템(shared memory system)에 의해 메모리 접근에 대한 락(lock)문제가 있을 수 있다.

5. 결론 및 향후 연구

본 논문은 최근 코어 수가 증가함에 따라 scale-up 환경도 중요해지고 있다고 보고, 분산 처리 환경인 스파크를 HPC 환경에서 구성했을 때 확장성을 가지는지 측정하였다.

120코어의 매니코어 시스템을 대상으로 4개의 스파크 벤치마크 워크로드를 실험한 결과 코어수가 증가함에 따라, 처리량이 올라가는 것을 기대했지만 반대로 일정 코어 수가 늘어나자 처리량이 떨어지는 확장성 문제를 발견하였다.

확장성 문제에 대해 먼저 CPU사용량을 측정하여 코어가 많아질수록 CPU 대기시간이 길어지는 것을 확인하였고, 확장성 저해 요소를 자바 가상 머신의 Garbage Collection, NUMA 메모리 구조에 의한 메모리 접근 지연 시간, 공유 메모리 시스템에 의한 락(lock)문제로 분석하였다. 향 후 연구로는 HPC 환경에서 확장성 문제에 대해 분산 처리 환경에 맞게 구현된 스파크 시스템을 어떻게 확장성이 있게 만들 것인가에 대해 연구할 필요가 있다.

참고 문헌

[1] J. Dean and S. Ghemawa. “MapReduce: Simplified Data Processing on Large Clusters”. In OSDI, 2004.

[2] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica. “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing”. In NSDI, 2012.

[3] Ahsan Javed Awan, Mats Brorsson, Vladimir Vlassov, and Eduard Ayguade. “How Data Volume Affects Spark Based Data Analytics on a Scale-up Server”. arXiv :1507.08340 2015.

[4] Nicholas Chaimov, Allen Malony, Shane Canon, Costin Iancu. “Scaling Spark on HPC Systems”. In HPDC, 2016.

[5] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, and J. Zhan, “BDGS: A scalable big data generator suite in big data benchmarking,” Lecture Note in Computer Sciences, Extended Version for the Fourth Workshop on Big Data Benchmarking, 2014.