Hadoop 클러스터에서 네임 노드와 데이터 노드가 빅 데이터처리 성능에 미치는 영향에 관한 연구

(A Study on the Effect of the Name Node and Data Node on the Big Data Processing Performance in a Hadoop Cluster)

이 영 훈*, 김 용 일**

(Younghun Lee, Yongil Kim)

요 약

빅 데이터 처리는 파일이나 이미지, 동영상 등 다양한 형태의 데이터를 처리하여 문제를 해결하고 통찰력 있는 유용한 정보를 제공한다. 현재 빅 데이터 처리를 위해 다양한 플랫폼이 사용되지만, 하둡이 가지는 단순성, 생산성, 확장성, 그리고 내고장성 때문에 많은 기관, 기업에서 빅 데이터 처리에 하둡을 사용하고 있다. 또한, 하둡은 다양한 하드웨어 플랫폼으로 클러스터를 구축할 수 있으며, 네임 노드(Master)와 데이터 노드(Slave)로 구분하여 빅 데이터를 처리한다. 본 논문에서는 실제 기관과 기업에서 사용하는 완전분산모드를 사용하였으며 원활한 테스트를 위해 저전력이고 저가인 싱글 보드를 사용하여 하둡 클러스터를 구축하였다. 네임 노드의 성능 영향 분석은 싱글 보드와 랩톱을 네임 노드로 사용하여 같은 데이터 처리를 통하여 비교하였으며 데이터 노드의 개수에 따른 영향 분석은 싱글 보드를 기존 클러스터의 개수에서 2배까지 늘려가며 데이터 노드가 미치는 영향을 분석하였다.

■ 중심어 : 빅데이터 ; 하둡 ; 클러스터 ; 싱글 보드 컴퓨팅;

Abstract

Big data processing processes various types of data such as files, images, and video to solve problems and provide insightful useful information. Currently, various platforms are used for big data processing, but many organizations and enterprises are using Hadoop for big data processing due to the simplicity, productivity, scalability, and fault tolerance of Hadoop. In addition, Hadoop can build clusters on various hardware platforms and handle big data by dividing into a name node (master) and a data node (slave). In this paper, we use a fully distributed mode used by actual institutions and companies as an operation mode. We have constructed a Hadoop cluster using a low-power and low-cost single board for smooth experiment. The performance analysis of Name node is compared through the same data processing using single board and laptop as name nodes. Analysis of influence by number of data nodes increases the number of data nodes by two times from the number of existing clusters. The effect of the above experiment was analyzed.

■ keywords : BigData ; Hadoop ; Cluster ; Single Board Computing;

Ⅰ. 서 론

빅 데이터(Big Data)란 통상적으로 사용되는 데이터의 수집, 관리 또는 처리 소프트웨어의 한계를 넘어서는 크기의 데이터를 의미 하며[1] 빅 데이터의 크기는 단일 데이터 집합의 크기가 수십 테라바이트에서 수 페타바이트에 이르며, 그 크기가 끊임없이 변화하는 것이 특징이다. 2001년 가트너의 더그 레이니가 작성한 연구 보고서[2]에서는 기존의 데이터 단위를 넘어서는 크기(Volume)와 사진, 동영상 등 기존의 구조화된 데이터가 아닌 다양한(Variety) 형태의 정보, 데이터의 생성과 흐름이 매우 빠르게 진행되는 속도(Velocity)의 세 가지 특징으로 정의하였고 가트너의 3V 정의가 널리 사용되고 있는 가운데 IBM에서는 정확성(Veracity)을 브라이언 홉킨스 등은 가변성(Variability)을 추가하여 빅 데이터에 대해서 4V로 정의하고 있다. 이후 빅 데이터 2.0에서 분석을 통해 도출된 결론은 문제를 해결하고 통찰력 있

* 학생회원, 호남대학교 인터넷소프트웨어학과
** 종신회원, 호남대학교 인터넷콘텐츠학과
본 논문은 2015학년도 호남대학교 교내공모과제 지원에 의하여 연구되었음
접수일자 : 2017년 08월 14일
수정일자 : 2012년 09월 19일
게재확정일 : 2017년 09월 21일
교신저자 : 김용일 e-mail : yikim@honam.ac.kr

는 유용한 정보를 제공한다는 가치(Value)라는 특징을 추가하여 5V로 정의하고 있다. 위의 정의와 같이 빅 데이터는 정형화된 데이터뿐만 아니라 파일이나 이미지, 동영상 등 다양한 형태로 존재할 수 있으며 기존의 정형화된 방법이나 도구로는 이와 같은 비정형 데이터에 대해 수집, 저장, 검색, 분석 시각화 등을 하기 어렵게 되어 새로운 방법이나 도구가 필요하게 되었다. 또한, 2015년 가트너의 10대 전략 기술에서 첨단분석이 추가되어 빅데이터 분석을 위한 연구의 필요성이 대두 되고 있다.

빅 데이터 처리 과정은 데이터의 수집에서부터 수집된 자료를 처리하여 분석하고, 시각화하여 도출된 결과를 이용하거나 폐기하는 과정을 계속하여 순환하여 가치 있는 결과를 도출한다. 빅 데이터의 처리 플랫폼은 아파치 오픈소스인 하둡(Hadoop)이 가장 많이 활용되고 있으며 데이터를 유연하고 더욱 빠르게 처리하기 위해서 NoSQL 기술이 활용되기도 한다. 또한, 빅 데이터의 처리 및 분석을 통해서 도출된 데이터의 의미와 가치를 시각적으로 표현하기 위한 기술로 대표적인 것으로는 R이 있다[3].

하둡은 대량의 데이터를 처리하기 위한 병렬 분산 처리 소프트웨어로 하나의 컴포넌트로 동작하는 것이 아니며 주요 컴포넌트로는 하둡 분산 파일 시스템(HDFS), 분산 데이터베이스(HBase), 분산 데이터 처리 시스템(MapReduce) 등이 있으며 복수의 컴포넌트가 연계되어 동작한다. 또한, 빅 데이터를 처리하는 경우 복수의 노드를 이용하여 병렬로 처리해야 하지만, 하둡에서는 작은 규모에서도 적용할 수 있도록 로컬모드, 유사분산모드, 완전분산모드 등 세 가지 동작 모드를 제공하고 있다. 일반적으로 빅 데이터 처리를 할 때 완전분산모드를 사용하며, 플랫폼은 한 개의 네임 노드(Master)와 다수의 데이터 노드(Slave)로 구성한다.

본 논문에서는 완전분산모드에서 네임 노드의 성능과 데이터 노드의 개수가 빅 데이터 처리 성능에 얼마나 영향을 미치는지를 연구하였다. 데이터 노드는 모두 같은 컴퓨터를 이용했으며, 네임 노드는 두 가지 다른 사양의 컴퓨터를 이용하였다. 성능 평가를 위해서 모든 노드에 같은 운영체제와 Hadoop 2.7.3을 설치하였으며 성능평가로는 10기가바이트의 데이터 세트와 이에 맞춘 MapReduce 프로그램을 작성하여 테스트하였다.

Ⅱ. 관련 기술

1. Hadoop 동작 모드

하둡에서는 다음 세 가지 동작모드를 활용하여 빅 데이터를 처리할 수 있다. 이러한 동작 모드는 복수의 노드를 활용할 뿐만 아니라 작은 규모에서도 적용할 수 있도록 하나의 노드만 활용할 수도 있다. 각 동작 모드는 로컬모드, 의사분산모드, 완전분산모드로 나뉘며 로컬모드는 한대의 서버상에 HDFS를 사용하지 않고 MapReduce 동작 환경을 구축하는 것으로 간단한 MapReduce의 동작 원리만을 이용할 때 사용한다. 의사분산모드는 한 대의 서버 상에 HDFS를 사용한 MapReduce 동작 환경을 구축하는 것으로 하둡에서 동작하는 네임 노드 프로세스나 데이터 노드 프로세스를 모두 동작시켜 볼 수 있다. 마지막으로 완전분산모드는 여러 대의 서버를 이용하여 HDFS를 사용한 MapReduce 동작 환경을 구축하는 것으로 내부적인 동작은 의사분산모드와 같다. 또한, 의사분산모드와 완전분산모드는 사용되는 서버의 개수가 다르고 의사분산모드는 하나의 서버에서 모든 컴포넌트가 작동하기 때문에 같은 환경에서의 비교가 어려워 성능 분석을 할 수가 없으므로 완전분산모드를 사용하여야 각 노드가 데이터 처리 성능에 미치는 영향을 측정할 수 있다.

2. MapReduce 프레임워크

MapReduce 프레임워크는 구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 목적으로 제작하여 2004년 발표한 소프트웨어 프레임워크로[4] 이 프레임워크는 페타바이트 이상의 대용량 데이터를 신뢰도가 낮은 컴퓨터로 구성된 클러스터 환경에서 병렬처리를 지원하기 위해서 개발하였으며 MapReduce는 Java와 C++, 그리고 기타 언어에서 적용할 수 있도록 작성되었다. 대표적으로 아파치 하둡에서 오픈소스 소프트웨어로 적용되었다.

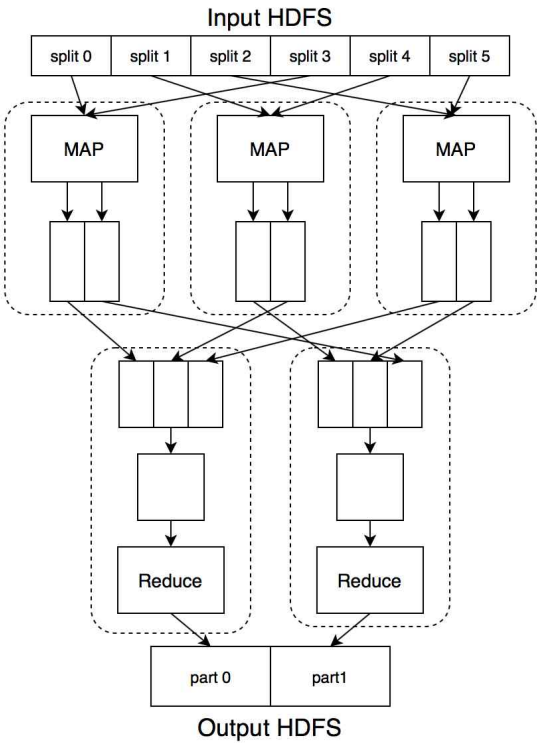

그림 1. MapReduce Framework Structure

위 그림 1과 같이 MapReduce 프레임워크는 하나의 잡(Job)을 독립된 태스크로 나누어서 병렬로 처리하며, 크게 입력 파일을 읽어 들여 나눈 후 필터링 등을 처리하는 Map 단계와 Map 단계에서 처리된 결과를 넘겨받아 키 별로 분류 및 합을 계산하는 리듀스 단계로 구분된다. 또한, Map 단계에서 처리된 결과를 리듀스 단계로 전달하기 전에 키값을 기준으로 데이터를 모으는 셔플 단계를 거치게 된다. 셔플 단계에서는 각 노드 간에 데이터 전송이 발생하므로 전송되는 데이터의 양이 전체적인 잡 처리에 많은 영향을 미칠 수 있다[5].

3. 미국 항공편 운항 통계 데이터

본 논문에서 사용한 데이터 세트는 미국 표준 협회(American National Standards Institute, ANSI)에서 2009년 공개한 미국 항공편 운항 통계 데이터로 1987년 10월부터 2008년 4월까지의 22년간의 미국 내 모든 상업 항공편에 대한 비행 도착 및 출발 세부정보를 담고 있다. 이 데이터는 총 1억 2천만 개에 달하며 12기가바이트 정도의 용량을 가지고 있으나 전체 데이터를 모두 사용하는 것은 싱글 보드를 사용하여 구성된 하둡 클러스터의 성능한계와 테스트 시간의 편의성을 고려할 때 무리라고 판단되어 1987년부터 2005년까지의 19년간의 데이터를 선택하였으며 약 10기가바이트 정도의 용량을 데이터 세트로 지정하였다. 미국 항공편 운항 통계 데이터는 연도별 CSV 파일로 구성되어 있으며 총 29개의 칼럼으로 구성되어있고 각 칼럼의 속성은[6] 아래 표 1과 같다.

표 1. 미국 항공편 운항 통계 데이터 컬럼 정보

Num	Name	Description
1	Year	1987 ~ 2008
2	Month	1월 ~ 12월
3	DayofMonth	1일 ~ 31일
4	DayofWeek	1(Monday) ~ 7(Sunday)
5	DepTime	actual departure time
6	CRSDepTime	scheduled departure time
7	ArrTime	actual arrival time
8	CRSArrTime	scheduled arrival time
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay,in minutes
17	Origin	origin IATA airport code
18	Dest(destination)	IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time, in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

칼럼 정보는 콤마를 기준으로 구분하며 1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,NA과 같은 형태로 데이터가 저장되어 있다. 이 데이터를 사용하여 지연을 최소화하기 위한 가장 좋은 시간대, 요일은 언제인가? 구형 비행기가 더 많은 지연을 겪고 있는가? 날씨가 얼마나 비행기를 많이 지연시키는가? 등의 문제를 해결하는데 필요한 정보를 도출할 수 있다.

III. 본 론

1. 하둡 클러스터 시스템 구성

빅 데이터 처리 성능 평가를 위하여 총 8개의 싱글 보드를 사용하여 아래 그림 2와 같은 하둡 클러스터를 구성하였다. 각 싱글 보드는 네임 노드와 보조 네임 노드, 데이터 노드를 구성하고 있으며 데이터 노드는 최대 6개까지 사용할 수 있다. 또한, 네임 노드의 성능 비교를 위해 다른 사양의 컴퓨터를 2대 준비하여 네임 노드로 사용하였다.

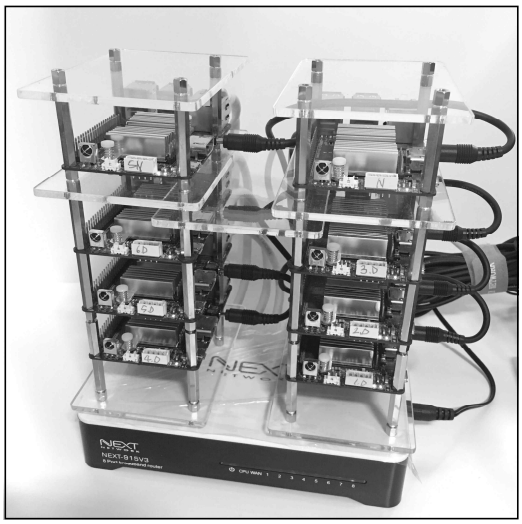

그림 2. 구성된 하둡 클러스터

위에서 구성한 하둡 클러스터의 노드별 상세 사양은[6] 아래 표 2, 표 3, 표 4와 같으며 보조 네임 노드의 경우 두 개의 타입 모두 싱글 보드를 사용하였다.

표 2. Type A 사양

종류	사양
Name Node	- Amlogic ARM Cortex-A5(ARMv7) 1.5Ghz quad core CPUs - 1GB DDR3 SDRAM - Gigabit Ethernet - 32트 MicroSD card, 32Gb emmc
Data Node	- Amlogic ARM Cortex-A5(ARMv7) 1.5Ghz quad core CPUs - 1GB DDR3 SDRAM - Gigabit Ethernet - 32GB MicroSD card, 32Gb emmc

표 3. Type B 사양

종류	사양
Name Node	- Intel Pentium 4405U 2.1Ghz dual core CPUs 4 Threads - 4GB DDR3L RAM - Gigabit Ethernet - 120GB M.2 SSD SATA3
Data Node	- Amlogic ARM Cortex-A5(ARMv7) 1.5Ghz quad core CPUs - 1GB DDR3 SDRAM - Gigabit Ethernet - 32GB MicroSD card, 32Gb emmc

표 4. Type C 사양

종류	사양
Name Node	- Intel Core i5-4590 3.3Ghz quad core CPUs 4 Threads - 4GB DDR3 RAM - Gigabit Ethernet - 470GB HDD SATA3
Data Node	- Amlogic ARM Cortex-A5(ARMv7) 1.5Ghz quad core CPUs - 1GB DDR3 SDRAM - Gigabit Ethernet - 32GB MicroSD card, 32Gb emmc

Type A, B, C의 데이터 노드의 사양은 모두 같으며 네임 노드의 사양만 차이를 두었다. 각 노드는 같은 테스트를 위해서 우분투 16.04 버전의 운영체제를 설치하였으며 소프트웨어 버전은 하둡 2.7.3버전과 Java 1.8버전을 사용하였다. 또한, 하둡 클러스터 설치 후 싱글 보드에 맞도록 가상 코어 수, 메모리 용량, 힙 사이즈 등의 설정값을 최적화 하였다[8].

2. 네임 노드 영향평가 방법

빅 데이터 처리에서 네임 노드의 성능이 처리 성능에 미치는 영향을 확인하기 위하여 미국 항공편 운항 통계 19년 치의 데이터 10기가바이트를 사용하였으며 네임 노드에 부하를 올리며 테스트하였다. 부하를 올리는 방법으로는 MapReduce 프로그램에서 Mapper와 Reducer가 처리해야 하는 조건을 하나씩 증가시켰다. MapReduce 프로그램은 총 4개를 작성하였는데 첫 번째는 정시 도착, 정시 출발, 그 외 등 세 가지를 분류하는 가장 낮은 부하의 프로그램이고, 두 번째는 조기 도착 및 출발, 지연 도착 및 출발 네 가지를 분류하는 프로그램이다. 세 번째는 두 번째의 프로그램에 정시 도착 및 출발한 경우를 합쳐서 분류하여 총 다섯 가지로 분류하였다. 그리고 마지막 프로그램은 세 번째 프로그램에서 정시 도착 및 출발을 구분하여 총 여섯 가지로 분류하였다. 또한, 각각의 부하를 테스트할 때 리듀스 태스크 개수를 2개부터 4개로 늘려가며 네임 노드에서의 I/O를 증가시켰다. 평가를 진행할 테스트 표는 아래 표 5와 같다.

표 5. 네임 노드 영향평가 테스트 표

Name	싱글 보드											
Load	3			4			5			6		
Reduce	2	3	4	2	3	4	2	3	4	2	3	4
Name	랩톱(Dual Core)											
Load	3			4			5			6		
Reduce	2	3	4	2	3	4	2	3	4	2	3	4
Name	데스크톱(Quad Core)											
Load	3			4			5			6		
Reduce	2	3	4	2	3	4	2	3	4	2	3	4

성능 측정 결과는 잡 태스크가 끝난 뒤 History 데몬에 모인 기록을 바탕으로 잡 경과 시간, 평균 맵 시간, 평균 셔플 시간, 평균 머지 시간, 평균 리듀스 시간 등의 5가지 항목을 가지고 비교하였으며, 각 테스트는 항목당 3회를 반복하여 평균값을 추출하고 그 평균값을 최종 결과표에 작성하였다.

3. 네임 노드 영향평가 결과

빅 데이터 처리에서 네임 노드 성능의 영향평가 테스트 결과는 아래 그림 3~6의 그래프와 같으며 왼쪽부터 싱글 보드, 랩톱, 데스크톱의 순서로 구성되어 있다.

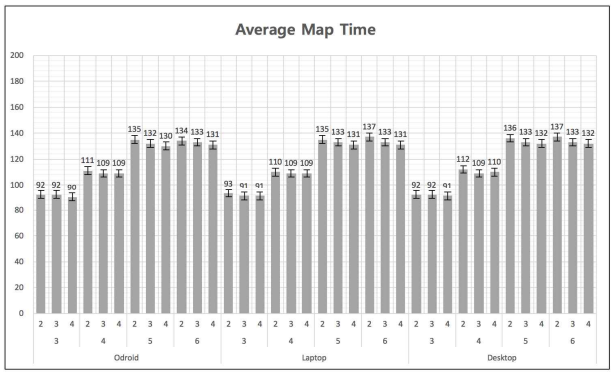

그림 3. 평균 맵 시간 결과

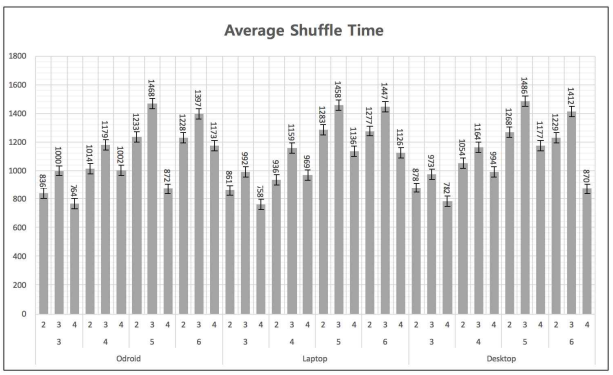

그림 4. 평균 셔플 시간 결과

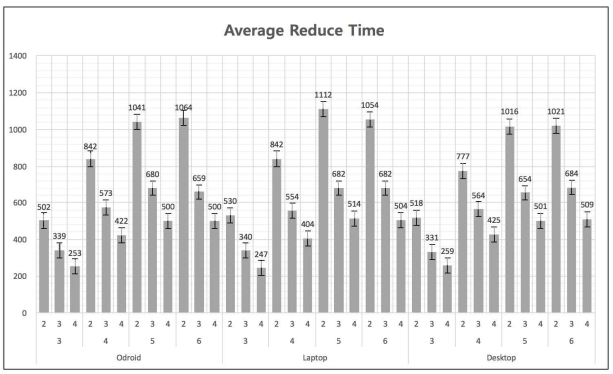

그림 5. 평균 리듀스 시간 결과

위 결과 그래프를 보면 각 네임 노드의 성능과 관계없이 평균 맵 시간, 평균, 리듀스 시간, 평균 셔플 시간 세 가지 항목 모두 MapReduce 프로그램의 부하와 리듀스 태스크의 개수가 같으면 오차범위 내에서 같은 시간이 소모되는 것을 볼 수 있다. 또한, 그래프 상에는 표현하지 않았지만 리듀스 태스크 개수가 1개인 경우에서는 MapReduce 프로그램의 부하가 3개를 초과할 경우 모두 처리에 실패하였기 때문에 따로 표시하지 않았다.

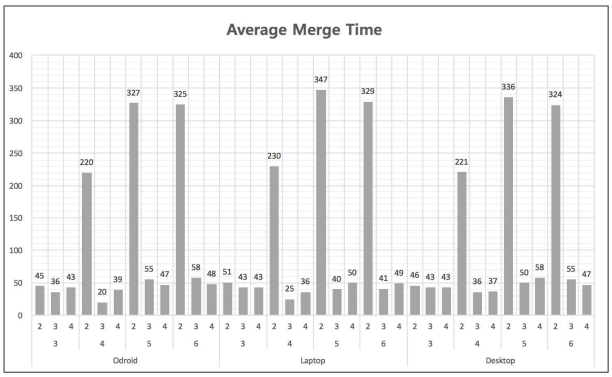

그림 6. 평균 머지 시간 결과

그림 6의 평균 머지 시간을 보면 MapReduce 프로그램의 부하가 4개 이상이고 리듀스 태스크 개수가 2개인 경우 머지 시간이 급격하게 증가하는 것을 볼 수 있다. 이는 충분한 리듀스 개수가 충족되지 못하여 머지 처리가 지연되면서 생긴 현상으로 보인다.

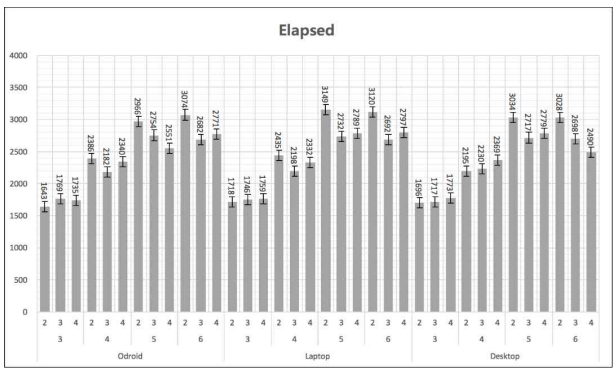

그림 7. 잡 경과 시간 결과

잡 전체 경과 시간은 그림 7과 같은 그래프와 같으며, 평균 머지 시간으로 인하여 모든 네임 노드에서 부하 4부터 6까지의 리듀스 태스크 개수가 2개인 항목의 경과 시간이 증가한 것을 제외하면 모두 유사한 증감 폭을 가지고 있음을 알 수 있다.

4. 데이터 노드 영향평가 방법

빅 데이터 처리에서 데이터 노드의 개수 차이가 성능에 미치는 영향을 확인하기 위하여 네임 노드 성능평가와 같은 데이터 세트에서 연도별로 얼마나 많은 항공기가 정해진 시간에 출발, 도착하

였는지 계산하는 MapReduce 프로그램을 작성하여 데이터를 처리하였다.

클러스터 서버의 구성은 Type A와 Type B 두 가지를 사용하여 테스트하였으며 보조 네임 노드를 사용한 경우와 사용하지 않은 경우, 마지막으로 데이터 노드의 개수를 3개에서 6개까지 늘려가며 테스트를 진행하였으며, 테스트 결과를 작성할 실험 표는 아래 표 6과 같다.

표 6. 데이터 노드 영향평가 테스트 표

Name	싱글 보드								랩톱(Dual Core)							
S.N*	O				X				O				X			
Data	3	4	5	6	3	4	5	6	3	4	5	6	3	4	5	6

* : 보조네임노드(Secondary Namenode)

성능 측정 결과는 네임 노드 성능평가와 같이 잡 태스크가 끝난 뒤 History 데몬에 모인 기록을 바탕으로 하였으며, 비교 항목 역시 같다. 각 테스트 항목에 대해서 3회를 반복하여 평균값을 추출하였으며 그 평균값을 최종 결과표에 작성하였다.

5. 데이터 노드 영향평가 결과

빅 데이터 처리에서 데이터 노드 개수의 영향 평가 테스트 결과는 아래 그림 8~12와 같으며 그래프는 왼쪽부터 싱글 보드, 랩톱의 순서이고 다시 4개 단위로 보조 네임 노드 사용, 미사용으로 구성되어있다.

그림 8, 9, 10을 보았을 때 데이터 노드의 개수 증가와 관계없이 평균 맵 시간, 평균 리듀스 시간, 평균 머지 시간은 오차범위 내에서 같은 시간이 소모되는 것을 볼 수 있다. 이는 데이터 노드의 개수가 맵, 머지, 리듀스 단계에 영향을 미치지 않는 것을 뜻하며, 맵, 머지, 리듀스 단계에 영향을 미치려면 리듀스 태스크 개수를 변경해야 할 것이다.

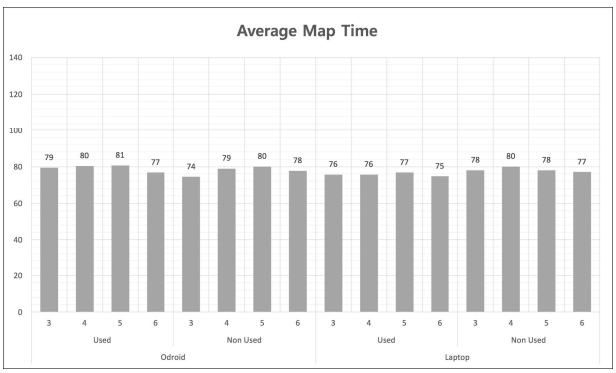

그림 8. 평균 맵 시간 결과

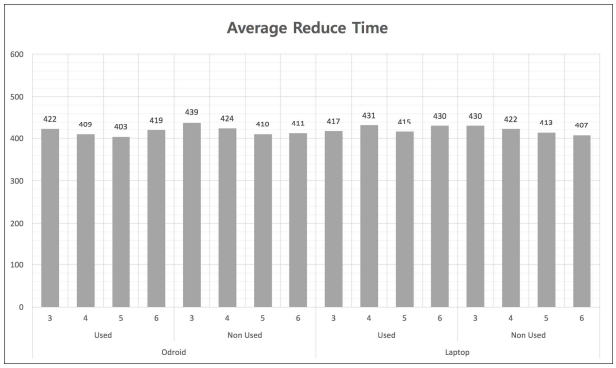

그림 9. 평균 리듀스 시간 결과

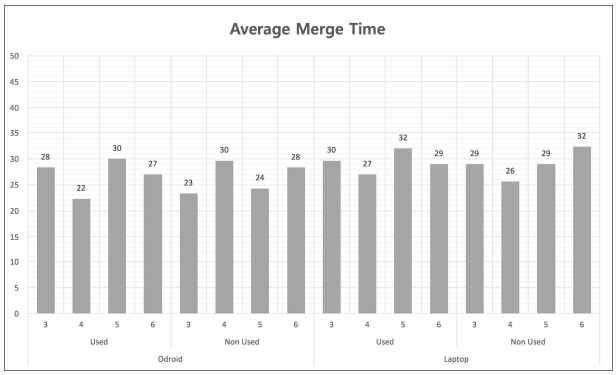

그림 10. 평균 머지 시간 결과

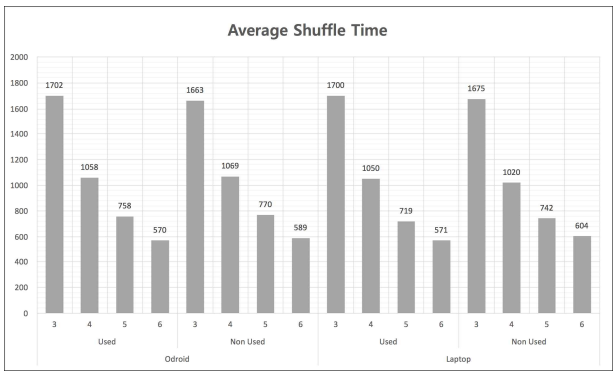

그림 11. 평균 셔플 시간 결과

하지만, 그림 11의 평균 셔플 시간 결과 그래프에서 데이터 노드의 개수가 증가하면 셔플 시간이 줄어들고 있으나 처리성능이 점점 줄어드는 것을 볼 수 있다. 이는 데이터 노드를 구동하고 있는 싱글 보드의 한계로 인해서 처리성능이 줄어드는 것으로 보이며 추후 데스크탑을 데이터 노드로 사용하면 감소 폭이 일정하게 줄 것으로 예상한다. 또한, 네임 노드를 싱글 보드와 랩톱 두 가지를 사용하였으나 처리 성능에 영향을 미치지 못함을 볼 수 있다. 이를 통해서 잡 전체 경과 시간은 아래 그림 12와 같은 그래프로 나타낼 수 있으며 평균 셔플 시간 그래프와 유사하게 표현되는 것을 볼 수 있다.

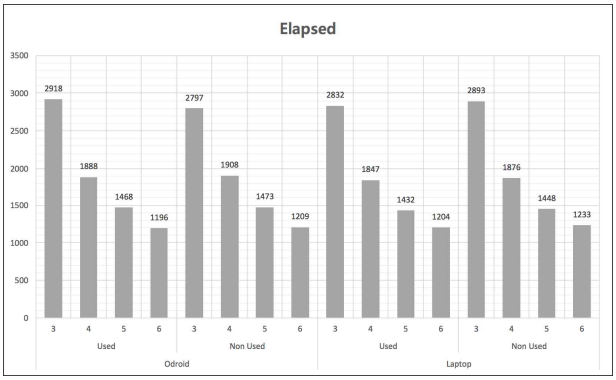

그림 12. 잡 경과 시간 결과

Ⅳ. 결 론

많은 기관, 기업에서 빅 데이터에 관심이 집중되고 있으며, IoT의 등장으로 빅데이터 처리에 다양한 연구들이 진행되고 있다[9]. 일반적으로 기관과 기업에서는 빅 데이터 처리를 위해 클러스터 구성을 통한 완전분산모드를 사용하게 되는데 이때 클러스터의 각 노드가 처리 성능에 미치는 영향에 대해 알고 있으면 효율적인 클러스터 구성이 가능하다.

본 연구에서는 하둡 클러스터 환경에서 네임 노드와 데이터 노드가 빅 데이터 처리에 미치는 영향을 분석하였으며, 분석 결과 네임 노드의 성능은 빅 데이터 처리에 영향을 미치지 않는 것을 볼 수 있었다. 네임 노드의 성능을 올리는 것보다 맵 태스크, 리듀스 태스크의 개수를 적정 수준으로 조정하는 것이 더 효율적일 것이다. 두 번째로 데이터 노드의 개수 차이는 데이터 처리에 상당한 영향을 미쳤다. 이는 셔플 단계의 시간이 큰 폭으로 줄어드는 것을 통해 알 수 있다. 데이터 노드의 개수는 위와 같이 맵과 리듀스 사이의 이동 시간을 줄여주게 되어 결과적으로 잡 태스크의 전체 경과 시간을 줄여주는 효과를 가져왔다.

본 연구의 결과를 통하여 기관과 기업에서 빅 데이터 처리를 위한 플랫폼을 구축할 때 네임 노드의 성능을 우선 하기보다 충분한 양의 데이터 노드를 구성하는 것이 비용을 절감하고 효율적인 분석을 할 수 있게 됨을 알 수 있으며, 앞으로는 직접 클러스터 서버를 구동하는 환경 이외에 아마존 웹 서비스와 같은 클라우드 컴퓨팅 환경에서의 빅 데이터 처리 성능에 관한 연구가 필요할 것으로 생각된다.

REFERENCES

[1] Chris Snijders, Uwe Matzat, Ulf-Dietrich Reips, "Big Data: Big gaps of knowledge in the field of Internet Science". *International Journal of Internet Science*, vol.7, no.1, pp. 1-5, 2012.
[2] Laney, Douglas, "3D Data Management: Controlling Data Volume, Velocity and Variety", Gartner
[3] 조성우, "빅데이터 시대의 기술", *KT종합기술원*, 5-7쪽, 2011년 10월
[4] CNET, "Google spotlights data center inner workings", Tech news blog
[5] 정재화, "시작하세요! 하둡 프로그래밍", *wikibooks*, 98-101쪽, 2014년 12월
[6] ANSI, "Airline on-time performance", Data expo, 2009.
[7] Hardkernel, "Odroid C1 Product Introduce", http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141578608433, 2014.
[8] Khaled Tannir, "Optimizing Hadoop for MapReduce", Packt Publishing, 2014.
[9] 이영훈, 김용일, "Mi Band와 MongoDB를 사용한 생체정보 빅데이터 시스템의 설계", *스마트미디어저널*, vol.5, no.4, 124-130쪽, 2016년 12월

저 자 소 개

이영훈(학생회원)

2017년 호남대학교 인터넷소프트웨어학과 학사 졸업

<주관심분야 : IoT, Bigdata, Android, Hybird App>

김용일(정회원)

1984년 3월 전남대학교 계산통계학과 (이학사)
1986년 2월 한국과학기술원 전산학과 (공학석사)
1986년 3월~1994년 2월 한국원자력연구소 선임연구원
1994년 3월~2000년 2월 초당대학교 컴퓨터학과 조교수
2002년 3월~현재 호남대학교인터넷콘텐츠학과 부교수

<주관심분야 : 빅데이터, 클라우드 컴퓨팅, 지능형 에이전트>