韓國컴퓨터情報學會 論文誌
第15卷 第5號, 2010. 5.

2010-15-5-1-6

클라우드 컴퓨팅에서 Hadoop 애플리케이션 특성에 따른 성능 분석

금 태 훈*, 이 원 주**, 전 창 호***

A Performance Analysis Based on Hadoop Application's Characteristics in Cloud Computing

Tae Hoon Keum*, Won Joo Lee**, Chang Ho Jeon***

요 약

본 논문에서는 클라우드 컴퓨팅을 위해 Hadoop 기반의 클러스터를 구축하고, RandomTextWriter, WordCount, PI 애플리케이션을 수행함으로써 애플리케이션 특성에 따른 클러스터의 성능을 평가한다. RandomTextWriter는 주어진 용량만큼 임의의 단어를 생성하여 HDFS에 저장하는 애플리케이션이고, WordCount는 입력 파일을 읽어서 블록 단위로 단어 빈도수를 계산하는 애플리케이션이다. 그리고 PI는 몬테카를로법을 사용하여 PI 값을 유도하는 애플리케이션이다. 이러한 애플리케이션을 실행시키면서 데이터 블록 크기와 데이터 복제본 수 증가에 따른 애플리케이션의 수행시간을 측정한다. 시뮬레이션을 통하여 RandomTextWriter 애플리케이션은 데이터 복제본 수 증가에 비례하여 수행시간이 증가함을 알 수 있었다. 반면에 WordCount와 PI 애플리케이션은 데이터 복제본 수에 큰 영향을 받지 않았다. 또한 WordCount 애플리케이션은 블록 크기가 64~256MB 일 때 최적의 수행시간을 얻을 수 있었다. 따라서 이러한 애플리케이션의 특성을 고려한 스케줄링 정책을 개발한다면 애플리케이션의 실행시간을 단축하여 클라우드 컴퓨팅 시스템의 성능을 향상시킬 수 있음을 보인다.

Abstract

In this paper, we implement a Hadoop based cluster for cloud computing and evaluate the performance of this cluster based on application characteristics by executing RandomTextWriter, WordCount, and PI applications. A RandomTextWriter creates given amount of random words and stores them in the HDFS(Hadoop Distributed File System). A WordCount reads an input file and determines the frequency of a given word per block unit. PI application induces PI value using the Monte Carlo law. During simulation, we investigate the effect of data block size and the number of replications on the execution time of applications. Through simulation, we have confirmed that the execution time of RandomTextWriter was proportional to the number of replications. However, the execution time of WordCount and PI were not affected by the number of replications. Moreover, the

• 제1저자 : 금태훈 교신저자 : 이원주
• 투고일 : 2010. 03. 29, 심사일 : 2010. 04. 19, 게재확정일 : 2010. 04. 29.
* 한양대 컴퓨터공학과 **인하공업전문대학 컴퓨터정보과 부교수 ***한양대 전자컴퓨터공학부 교수
※이 논문은 2009년 한국컴퓨터정보학회 제40차 하계학술대회에서 발표한 논문("클라우드 컴퓨팅을 위한 Hadoop 애플리케이션 특성 분석")을 확장한 것임

execution time of WordCount was optimum when the block size was 64~256MB. Therefore, these results show that the performance of cloud computing system can be enhanced by using a scheduling scheme that considers application's characteristics.

▸ Keyword : 클라우드 컴퓨팅(Cloud Computing), HDFS(Hadoop Distributed File System), 클러스터(Cluster), Hadoop,

Ⅰ. 서 론

클라우드 컴퓨팅이란 개인용 컴퓨터 또는 기업의 서버에 개별적으로 저장해 두었던 자료와 소프트웨어들을 클라우드 클러스터로 구축하여 필요할 때 PC나 휴대폰과 같은 각종 단말기를 이용하여 원격 작업을 수행할 수 있는 환경을 의미한다. 이러한 클라우드 컴퓨팅의 개념은 그림 1과 같다.

그림 1. 클라우드 컴퓨팅 개요
Fig. 1. Cloud computing overview

그림 1에서 클라우드 클러스터는 서비스 제공자로서 사용자들에게 서비스 카탈로그를 제공한다. 사용자들은 네트워크 접속이 가능한 단말기만 있으면 시간과 장소 제한 없이 이러한 서비스들을 저렴한 비용으로 사용할 수 있다. 사용자는 클라우드 클러스터 내부에서 해당 작업에 필요한 컴퓨팅 자원을 할당하는 클라우드 구조 또는 동작에 대하여 전혀 알 필요 없이 단순히 작업 요청만 하면 된다.

이러한 특징을 가지는 클라우드 컴퓨팅은 2006년 구글(Google)의 엔지니어가 처음으로 제안하였으며[1] 유연한 동적 IT 인프라와 QoS가 보장되는 컴퓨팅 환경, 그리고 구성 가능한 소프트웨어 서비스를 제공하기 때문에 많은 관심을 받고 있다. 국외에서는 이미 많은 연구와 개발이 진행 중이며, 대표적인 예로 Amazon Elastic Compute Cloud[2], IBM Blue Cloud[3] 그리고 Google App Engine[4] 등이 있다. 우리나라에서도 2008년에 한국클라우드컴퓨팅협의회(CCKI)가 출범하는 등 관련 산업체 중심으로 클라우드 컴퓨팅에 대한 연구를 진행하고 있다. 클라우드 컴퓨팅 기술은 가상화 기술, 웹서비스, SOA(Service Oriented Architecture), 웹 2.0, 분산 파일 시스템과 프로그래밍 모델 등이 있다[5].

본 논문에서는 분산 파일 시스템과 프로그래밍 모델을 이용하여 클라우드 컴퓨팅을 위한 클러스터를 구축하고, 프로그래밍 모델을 이용한 애플리케이션의 특성을 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 분산 파일 시스템과 클라우드 프로그래밍 모델에 대하여 설명한다. 3장에서는 Apache 오픈 소스 프로젝트인 Hadoop을 이용한 클러스터 구축 방법에 대해서 설명한다. 4장에서는 Hadoop의 애플리케이션 의사코드와 성능 평가를 통해 애플리케이션의 특성을 분석한다. 그리고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

클라우드 컴퓨팅을 위한 대표적인 분산 파일 시스템과 프로그래밍 모델은 구글의 GFS(Google File System) 과 MapReduce 기술이 있다[6][7]. 이를 바탕으로 Apache에서는 오픈 소스 기반의 Hadoop 프로젝트를 진행 중이며 HDFS(Hadoop Distributed File System)과MapReduce 기술을 결합한 Hadoop-Core를 배포하고 있다[8][9][10].

2.1 HDFS(Hadoop Distributed File System)

HDFS는 저비용의 수백 내지 수천 노드를 가지는 클러스터를 이용하여 기가 바이트 또는 테라 바이트의 대용량 데이터 집합을 처리하는 응용 프로그램에 적합하도록 설계한 분산 파일 시스템이다. HDFS는 마스터-슬레이브 구조로 동작하며, 노드간에 TCP/IP 프로토콜을 사용하여 통신한다. 또한 노드 실패에 대비하여 데이터를 복제하여 저장한다. 이 HDFS의 구조는 그림 2와 같다.

그림 2에서 단일 네임 노드(Name Node)는 파일 시스템 네임스페이스를 관리하고, 클라이언트에 의한 파일 접근을 통제한다. 데이터 노드(Data Node)는 클러스터에서 각 노드의 스토리지를 관리하며 네임노드가 지시하는 블록 명령 등을

수행한다.

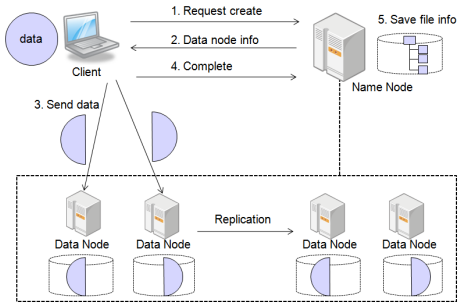

그림 2. HDFS 구조
Fig. 2. HDFS Architecture

클라이언트의 대용량 파일 원본을 HDFS에 저장할 때 블록 단위로 나누어서 저장한다. 기본적인 블록 크기는 64MB 또는 128MB이며 사용자가 임의로 설정 가능하다. 각 파일의 블록들은 데이터 노드 실패 시에 자동 복구를 위해 복제본을 생성한다. 이때 복제 설정값은 기본적으로 3 이지만 사용자가 임의로 설정 가능하다. 이러한 복제 블록을 어떻게 배치하느냐에 따라 HDFS의 신뢰성과 성능에 중요한 영향을 미치는데 현재 HDFS는 Rack-aware 복제 배치 정책을 사용하고 있다. 이것은 다른 랙(rack)에 위치한 두 노드 사이의 통신은 스위치를 통하기 때문에 대역폭이 작은 반면 동일한 랙에 위치한 두 노드 간의 대역폭은 훨씬 크다는 점을 고려한 것이다. 예를 들면 복제 설정값이 3일 경우, 동일 랙에 존재하는 두 노드에 각각 복제본을 저장하고, 다른 랙의 노드에 동일한 복제본을 저장한다. 따라서 노드 실패가 발생하면 동일한 복제본 2개를 이용하여 데이터를 복구함으로써 데이터 손실을 방지할 수 있다. 또한, HDFS의 통신은 TCP/IP 프로토콜을 사용하며, 네임 노드와 데이터 노드간의 TCP 포트는 클러스터 구축 시 설정한다. TCP 포트 설정은 3.2절에서 자세히 설명한다.

2.2 MapReduce

MapReduce는 대용량 데이터 집합을 처리하는 기법이다. 또한 MapReduce는 일정한 데이터 포맷을 생성하여 분산처리 하는 기능을 제공한다. 이때 개발자는 Map 함수와 Reduce 함수를 정의한다. Map 함수는 입력된 〈key, value〉쌍들을 처리하여 〈key, value〉쌍의 중간값 집합을 생성한다. Reduce 함수는 중간 key 값을 가지는 모든 중간값들을 통합하여 최종 출력값으로 저장한다. MapReduce의 동작 과정은 다음과 같다.

① 마스터 노드는 입력 파일들을 특정 크기로 분할하고, 그 분할된 M개의 조각을 클러스터의 작업 노드들에게 할당한다.
② 각 작업 노드들은 할당 받은 Map 작업에 필요한 조각들을 로드하여, Map 함수를 수행하고 중간 결과값을 저장한다.
③ Reduce 작업을 할당 받은 작업 노드들은 중간 결과값을 로드하여 Reduce 함수를 수행하고, 최종 결과값을 저장한다.
④ 모든 Map과 Reduce 작업이 완료되면 마스터 노드는 그 결과를 사용자 프로그램에 전송한다.

마스터 노드는 주기적으로 모든 작업 노드들이 동작하는지 체크한다. 만일 특정 시간 동안 작업 노드의 응답이 없으면 마스터 노드는 해당 작업 노드를 실패로 처리한다. 이때 더 이상 작업 실패 노드 접근이 불가능하기 때문에 로컬 디스크에서 작업 중이던 결과물에 접근할 수 없다. 따라서 수행 중이던 작업을 모두 초기 상태로 되돌리고, 다른 작업 노드에 할당하여 재시작 한다.

III. Hadoop 기반의 클러스터 구축

Hadoop 기반의 클러스터 구축 방식에는 단일 구성 방식, 가상 분산 방식, 완전 분산 방식이 있다. 먼저 단일 구성 방식은 비분산모드로 Hadoop을 하나의 로컬 시스템에서 자바 프로세스로 실행하는 방식으로 주로 Hadoop 기반의 응용프로그램 디버깅에 유용하다. 가상 분산 방식은 하나의 노드에서 네임노드와 데이터 노드 각각을 가상의 자바 프로세스로 설정하여 실행하는 것이다. 마지막으로 완전 분산 방식은 TCP/IP 프로토콜로 통신하는 다수의 노드로 하나의 클러스터를 구성하는 방식이다. 본 논문에서는 완전 분산 방식으로 Hadoop 기반의 클러스터를 구축한다.

3.1 클러스터 구축

본 논문에서는 클라우드 컴퓨팅을 위해 소규모의 동일한 노드로 구성된 클러스터를 구축한다. 클라우드 클러스터 환경은 표 1과 같다[11].

표 1. 클라우드 컴퓨팅을 위한 클러스터 환경
Table 1. Cluster environment for cloud computing

노드 수	네임 노드	1개
	데이터 노드	4개
노드 성능	CPU 성능	듀얼 코어 2 Ghz
	RAM 용량	2 GB
운영체제	Linux(Ubuntu)	
Hadoop	Hadoop-0.19.1	

표 1의 클라우드 클러스터 환경에서 각 노드의 운영체제는 Linux(Ubuntu)를 포팅하고, Hadoop-0.19.1을 설치한다. 또한, 네임 노드와 데이터 노드들은 외부 트래픽이 없고 대역폭이 100Mbps인 독립적인 네트워크상의 동일한 스위치에 연결된다.

3.2 Hadoop 환경 설정

Hadoop은 hadoop-site.xml 파일을 통해 네임 노드와 통신할 포트, HDFS 데이터 복제 수, 데이터 블록 크기 등의 속성을 설정할 수 있다. hadoop-site.xml 파일의 내용은 그림 3과 같다.

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://192.168.0.1:9000</value>
    </property>

    <property>
        <name>mapred.jop.tracker</name>
        <value>192.168.0.1:9001</value>
    </property>

    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>

    <property>
        <name>dfs.block.size</name>
        <value>67108864</value>
    </property>
</configuration>
```

그림 3. hadoop-site.xml 설정
Fig. 3. Configuration of hadoop-site.xml

그림 3에서 fs.default.name 은 파일시스템의 마스터 데이터를 관리하는 네임 노드 서버의 URI(Uniform Resource Identifier)를 지정하는 속성이다. fs.default.name의 속성값인 hdfs://192.168.0.1:9000는 네임 노드 서버 IP와 포트 번호를 각각 192.168.0.1과 9000으로 설정하고 hdfs 프로토콜을 사용하여 통신한다는 의미이다. mapred.job.tracker는 MapReduce 분산 처리 작업을 관리하는 Job Tracker 서버를 설정하는 속성이다. mapred.job.tracker의 속성값인 192.168.0.1:9001은 Job Tracker 서버 IP와 포트 번호를 각각 192.168.0.1과 9001로 설정하여 사용한다는 의미이다.

일반적으로 네임 노드가 Job Tracker의 기능을 수행하기 때문에 그림 3에서는 서버 IP를 192.168.0.1로 동일하게 설정한다. hadoop-site.xml 파일에서 fs.default.name 속성과 mapred.job.tracker 속성은 반드시 설정하고, 다른 속성은 Hadoop에 설정한 기본적인 속성값을 지정한다. HDFS에 데이터를 저장할 때 생성하는 복제본의 수는 dfs.replication 속성에 지정한다. 기본적인 복제본의 수는 3 이다. dfs.block.size는 데이터 블록의 크기이며 기본적으로 64MB이다. 하지만 본 논문에서는 dfs.replication과 dfs.block.size의 속성을 다양하게 변경하면서 애플리케이션 특성을 분석하고 성능을 평가한다.

IV. 애플리케이션 분석 및 성능 평가

본 논문에서는 Hadoop의 기본적인 애플리케이션인 RandomTextWriter, WordCount, PI의 특성을 분석한다. 각 애플리케이션의 특성은 표 2와 같다.

표 2. 애플리케이션 특성
Table 2. Application's characteristics

애플리케이션	특성
RandomTextWriter	I/O 중심
WordCount	계산(CPU) 및 I/O 중심
PI	계산(CPU) 중심

RandomTextWriter는 주어진 용량만큼 임의의 단어를 생성하여 HDFS에 저장하는 I/O 중심의 애플리케이션이고, WordCount는 입력 파일을 읽어서 블록 단위로 단어 빈도수를 계산하는 I/O 및 계산 중심의 애플리케이션이다. 본 논문에서는 RandomTextWriter에서 생성한 파일을 WordCount의 입력 파일로 사용하여 단어 빈도수를 계산한다. PI는 몬테카를로법을 사

용하여 PI 값을 유도하는 계산 중심의 애플리케이션이다. RandomTextWriter, WordCount, PI 애플리케이션에 대하여 좀 더 상세히 분석해 본다.

4.1 애플리케이션 의사코드 분석

RandomTextWriter, WordCount, PI 애플리케이션을 분석하기 위해 각 애플리케이션의 동작을 의사코드로 표현한다. RandomTextWriter, WordCount, PI 애플리케이션은 100~500 라인 정도로 간단하게 MapReduce 프로그래밍 모델로 작성할 수 있다. 그림 4, 5, 6은 각 애플리케이션들의 주요 동작에 관련된 부분인 map 함수와 reduce 함수를 의사코드로 표현한 것이다. 이러한 애플리케이션은 map 함수와 reduce 함수를 오버라이딩(overriding)하여 사용함으로써 단순한 코드만으로 분산 처리를 구현할 수 있다.

```
Text text_array[] = {word1, word2, word3 ... }
Function RandomText()
    r = RandomInt()
    return text_arrya[r]
End RandomText

Function map(output)
    while (size of output < desired size)
        Text key = RandomText()
        Text value = RandomText()
        output.collect(key, value)
    End while
End map
```

그림 4. RandomTextWriter 애플리케이션 의사코드
Fig. 4. RandomTextWriter application pseudocode

그림 4의 RandomTextWriter 애플리케이션의 의사코드를 살펴보면 reduce 함수는 사용하지 않고, map 함수만 사용한다. 즉, RandomTextWriter 애플리케이션은 단순히 임의의 단어들을 저장하여 주어진 크기의 파일을 생성해야 하기 때문에 reduce 함수를 사용하지 않는다. 그리고 map 함수에서 주어진 파일의 크기까지만 임의의 단어들을 output에 저장한다. output에 저장된 데이터는 key, value 구분 없이 key와 value 모두 텍스트 형 데이터이다.

그림 5의 WordCount 애플리케이션은 map 함수에서 파일의 내용을 단어 단위로 분리하여 단어 수를 카운트 한다. 각각의 단어를 key로 하고, value는 1로 지정하여 output에 저장한다. reduce 함수에서는 map 함수의 수행 결과를 key, values로 전달 받는다. 그리고 key에 해당되는 특정 단어에 대한 value를 합산함으로써 단어 수를 계산한다. 그리고 그 결과를 output에 저장하게 된다.

```
Function map(key, value, output)
    Text word = first word in value
    while (until last word in value)
        output.collect(word, 1)
        word = next word in value
    End while
End map

Function reduce(key, values, output)
    integer sum = 0
    while (until last value in values)
        sum += values.next()
    End while
    output.collect(key, sum)
End reduce
```

그림 5. WordCount 애플리케이션 의사코드
Fig. 5. WordCount application pseudocode

```
Function map(key, value, output)
    integer i = numOutside
                = numInside = 0
    double x,y,d
    while ( i <= key, i += 1 )
        x = randomDouble()
        y = randomDouble()
        d = x * x + y * y
        if ( d > 1 )
            numOutside += 1
        else
            numInside += 1
        End if
    End while
    output.collect(0, numOutside)
    output.collect(1, numInside)
End map

Function reduce(key, values)
    if(key == 1)
        while (until last value in values)
            numInside += values.next()
        End while
    else
        while (until last value in values)
            numOutside += values.next()
        End while
    End if
End reduce
```

그림 6. PI 애플리케이션 의사코드
Fig. 6. PI application pseudocode

그림 6의 PI 애플리케이션은 map 함수 내에서 key 값만큼 반복하여 임의의 소수를 생성하고 계산한다. 이는 몬테카를로법을 통한 PI 유도식을 map 함수로 구현한 것이다. 계

산된 결과는 numOutside와 numInside로 분류되는데, output에 저장할 때 key를 0과 1로 구분하여 저장한다. 그리고 reduce 함수에서는 모든 map 함수에서 계산한 numOutside 값과 numInside 값을 각각 더하게 된다. 그리고 최종적으로 reduce 함수의 결과를 이용하여 PI값을 유도하게 된다.

4.2 성능 평가

본 논문에서는 애플리케이션의 성능 평가를 위해 3장에서 구축한 클라우드 클러스터를 이용한다. 클라우드 클러스터 환경에서 수행할 애플리케이션 환경은 표 3과 같다.

표 3. 애플리케이션 환경
Table 3. Application's environment

RandomTextWriter	파일 생성 크기 : 3 GB Task 당 1GB 생성
WordCount	입력 파일 크기 : 3 GB
PI	샘플 수 : 1000 만개

표 3의 RandomTextWriter 애플리케이션은 3GB 크기의 텍스트 파일을 생성한다. 그리고 이 파일을 WordCount 애플리케이션의 입력으로 사용한다. WordCount 애플리케이션을 수행할 때는 매우 큰 용량의 텍스트 파일이 필요하기 때문에 RandomTextWriter 애플리케이션을 먼저 수행하여 생성된 3GB 크기의 텍스트 파일을 입력으로 사용한다.

본 논문에서는 애플리케이션의 특성에 따른 성능을 분석하기 위해 블록 크기와 데이터 복제본 수 증가에 따른 수행시간을 측정하여 성능 평가의 척도로 사용한다. 블록 크기와 데이터 복제본 수는 hadoop-site.xml 파일의 dfs.replication 속성과 dfs.block.size 속성에 해당 값을 설정한다. HDFS의 입·출력에 대한 기본 단위는 블록이며, 데이터 복제본 수는 노드 실패를 대비해 블록을 다른 노드에 저장하는 수를 의미한다.

블록 크기 증가에 따른 각 애플리케이션의 수행시간을 측정한 결과는 그림 7과 같다.

그림 7을 살펴보면 RandomTextWriter 애플리케이션은 블록 크기에 영향을 크게 받지 않음을 알 수 있다. 이것은 각 태스크에서 블록 단위로 데이터를 생성하는 것이 아니라 주어진 크기만큼 데이터를 생성하고, 로컬 디스크에 저장하기 때문이다. 하지만 WordCount 애플리케이션은 블록 크기에 크게 영향을 받는 다는 것을 알 수 있다. WordCount는 map 함수를 수행할 때, 데이터를 블록 단위로 계산하기 때문에 블록 크기가 작다면 그만큼 map 함수의 수행 빈도 수가 증가한다. 따라서 통신비용과 메모리 할당 등 계산 준비에 따른 시간이 증가한다. 특히, 블록 크기가 256MB 이상일 때는 수행시간이 급격히 증가하는 것을 볼 수 있다. 이것은 클라우드 클러스터를 구성하는 노드들의 성능이 하드웨어적으로 동일하다 하더라도 미비한 전체적인 성능 차이 때문에 발생한 것이다. 즉, 64MB의 블록을 계산할 때 가장 빠른 노드와 가장 느린 노드의 실행 시간 차이가 10초라면 1024MB의 블록을 계산할 때는 약 160초 정도의 실행시간 차가 발생한다. 따라서 블록 크기가 64MB일 때는 빠른 노드에서 더 많은 작업을 실행할 수 있지만, 블록 크기가 클수록 빠른 노드와 느린 노드는 동일한 작업 수를 할당 받게 되어 최종 작업 완료시간은 가장 느린 노드의 완료시간이 되는 것이다. 그리고 WordCount는 중복된 단어들이 모두 제거 되고 그 단어의 빈도 수만 output에 작성되기 때문에 output의 크기가 매우 작다.

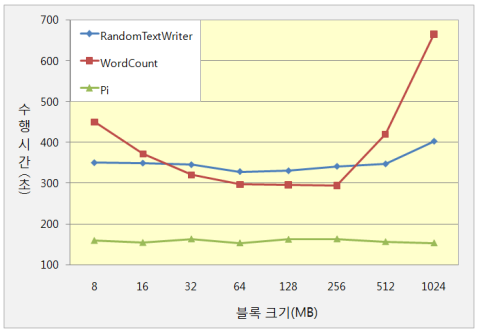

그림 7. 블록 크기 증가에 따른 애플리케이션 수행시간
Fig. 7. Block size vs. Execution time

데이터 복제 수에 따른 각 애플리케이션의 수행시간을 측정한 결과는 그림 8과 같다.

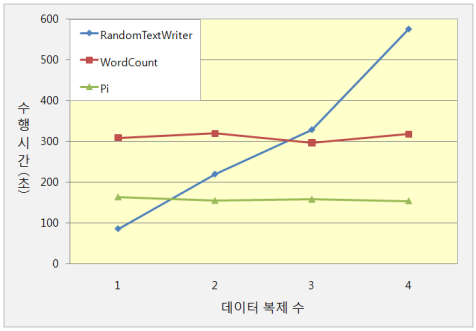

그림 8. 데이터 복제본 수 증가에 따른 애플리케이션 수행시간 (단어 중복 사용)
Fig. 8. The numbers of data replication vs. Execution time (using overlaped words)

그림 8을 살펴보면 RandomTextWriter 애플리케이션은 데이터 복제본 수 증가에 비례하여 수행시간이 증가함을 볼 수 있다. 이것은 생성 데이터를 다른 노드로 전송하기 때문에 통신비용이 증가하고, HDFS에 저장해야 하는 데이터 크기는 배수로 증가하기 때문에 수행시간이 증가한다. 반면에 WordCount와 PI 애플리케이션은 데이터 복제본 수에 큰 영향을 받지 않는다. PI 애플리케이션은 각 노드에서 map 함수로 계산하고, 2개의 실수형 데이터만 파일로 저장한다. 따라서 PI가 생성하는 output은 실험에서 가장 작은 블록 크기인 8MB에도 미치지 못하기 때문에 블록 크기에 영향을 받지 않으며, 데이터 복제수가 증가해도 큰 영향을 받지 않음을 알 수 있다.

Input	Output
word1	
word2	word1, 2
word3	word2, 1
word1	word3, 3
word3	
word3	

(a) 단어 중복 허용
(a)overlaped words

Input	Output
word1	word1, 1
word2	word2, 1
word3	word3, 1
word4	word4, 1
word5	word5, 1
word6	word6, 1

(b) 단어 중복 허용 않됨
(b) non-overlaped words

그림 9. WordCount 출력
Fig. 9. Output of WordCount

WordCount의 출력은 그림 9의 (a)와 같이 중복된 단어들이 모두 제거 되고 그 단어의 빈도수만 output에 저장되기 때문에 output의 크기가 매우 작아진다. 하지만 그림 9의 (b)와 같이 중복되지 않는 단어로 구성되었을 경우에는 WordCount의 output은 입력 파일 크기인 3GB보다 커진다.

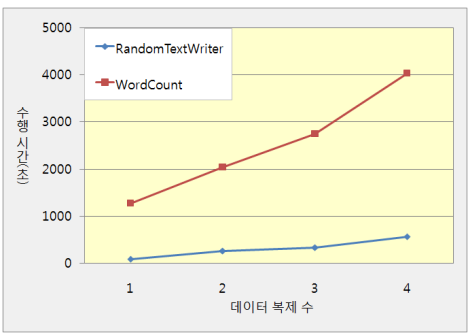

그림 10. 데이터 복제본 수 증가에 따른 애플리케이션 수행시간(단어 중복 허용 않된 경우)
Fig. 10. The numbers of data replication vs. Execution time (using non-overlaped words)

단어 중복을 허용하지 않는 경우, 데이터 복제 수에 따른 각 애플리케이션의 수행시간을 측정한 결과는 그림 10과 같다.

그림 10을 살펴 보면 에서 WordCount의 수행시간이 RandomTextWriter에 비해 크게 증가함을 볼 수 있다. 중복되지 않는 단어로 구성된 입력 파일을 처리하는 경우 WordCount의 map 함수 결과로 발생한 output이 입력 파일 크기인 3GB보다 커지기 때문에 계산량이 증가하고, reduce 함수에서 대부분의 수행시간을 차지하게 된다.

V. 결론

본 논문에서는 클라우드 컴퓨팅을 위해 Hadoop 기반의 클러스터를 구축한다. 그리고 이 클러스터 환경에서 Hadoop 애플리케이션인 RandomTextWriter, WordCount, PI 를 수행하여 애플리케이션 특성에 따른 클러스터의 성능을 평가한다.

RandomTextWriter는 주어진 용량만큼 임의의 단어를 생성하여 HDFS에 저장하는 애플리케이션이고, WordCount는 입력 파일을 읽어서 블록 단위로 단어 빈도수를 계산하는 애플리케이션이다. 그리고 PI는 몬테카를로법을 사용하여 PI 값을 유도하는 애플리케이션이다. 이러한 애플리케이션을 실행시키면서 블록 크기와 데이터 복제본 수 증가에 따른 수행시간을 측정하여 클러스터의 성능을 평가한다. RandomTextWriter 애플리케이션은 데이터 복제 수 증가에 비례하여 수행시간이 증가함을 알 수 있었다. 반면에 WordCount와 PI 애플리케이션은 데이터 복제 수에 큰 영향을 받지 않았다. 또한 WordCount 애플리케이션은 블록 크기가 32MB이하 또는 256MB 이상에서 수행시간이 증가한다. 따라서 WordCount 애플리케이션은 블록 크기가 64~256MB 일 때 최적의 수행시간을 얻을 수 있었다.

따라서 이러한 애플리케이션의 특성을 고려한 클라우드 컴퓨팅 환경의 스케줄링 정책을 개발한다면 애플리케이션의 실행시간을 단축하여 시스템의 성능을 향상시킬 수 있다.

참고문헌

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

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

〔3〕 IBM Blue Cloud project, *http://www04.ibm.com/jct03001c/press/us/en/pressrelease/22613.wss*,2009.
〔4〕 Google App Engine, *http://code.google.com/appengine*, 2009.
〔5〕 L. Wang and G. Von Laszewski, "Cloud Computing: A Perspective Study," In Proceedings of the Grid Computing Environments (GCE) workshop, Nov. 2008.
〔6〕 S. Ghemawat, H. Gobioff, S.T. Leung, "The Google file system," ACM SIGOPS Operating Systems Review, Vol. 37, No. 5, pp. 29-43, Dec. 2003.
〔7〕 J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters," Communications of the ACM, Vol. 51, No. 1, pp. 107-113, Jan. 2008.
〔8〕 Hadoop, *http://hadoop.apache.org*, 2009.
〔9〕 J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang and M. Yang, "Chukwa: A large-scale monitoring system," Proceeding of international conference on Cloud Computing and Its Applications, pp. 1-5, Oct. 2008.
〔10〕 J. Tan, X. Pan, S. Kavulya, R. Gandhi and P. Narasimhan, "Mochi: Visualizing Log-Anlaysis Based Tools for Debugging Hadoop," In USENIX Workshop on Hot Topics in Cloud Computing(HotCloud), SanDiego, CA, June 2009.
〔11〕 금태훈, 김세회, 김건우, 이원주, 전창호, "클라우드 컴퓨팅을 위한 Hadoop 애플리케이션 특성 분석(An Analysis of Hadoop Application's Characteristics for Cloud Computing)," 한국컴퓨터정보학회 2009 하계학술발표논문집, 제17권, 제1호, 11-12쪽, 2009년 7월.

저 자 소 개

금 태 훈

2009: 경일대학교 컴퓨터공학과 학사.
현 재: 한양대학교 컴퓨터공학과 석사과정.
관심분야 : 클라우드컴퓨팅, Grid컴퓨팅

이 원 주

1989: 한양대학교 전자계산학과 공학사.
1991: 한양대학교 컴퓨터공학과 공학석사.
2004: 한양대학교 컴퓨터공학과 공학박사
현 재: 인하공업전문대학 컴퓨터정보과 부교수
관심분야: 병렬처리시스템, 성능분석, Grid컴퓨팅, 클라우드컴퓨팅

전 창 호

1977: 한양대학교 전자공학과 학사.
1982: Cornell University 컴퓨터공학과 석사.
1986: Cornell University 컴퓨터공학과 박사.
1977-1979: 전자통신연구소 연구원.
현 재 : 한양대학교 전자컴퓨터공학부 교수.
관심분야: 병렬처리시스템, 성능분석, Grid컴퓨팅, 클라우드컴퓨팅