한국컴퓨터정보학회지
제18권 제1호, 2010. 6.

Hadoop 설치와 애플리케이션의 구동

• 금태훈(한양대학교 컴퓨터공학과)
• 김세회(한양대학교 컴퓨터공학과)
• 이상준(평택대학교 물류정보대학원)

Ⅰ. 서론

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. 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. 한국에서도 2008년에 한국클라우드컴퓨팅협의회(CCKI)가 출범하는 등 관련 업계 중심으로 클라우드 컴퓨팅에 대한 관심을 높이고 있다. 클라우드 컴퓨팅 기술은 가상화 기술, 웹서비스, SOA (Service Oriented Architecture), 웹 2.0, 분산 파일시스템과 프로그래밍 모델 등이 있다[5].

Hadoop[8]은 이러한 클라우드 환경을 구축할 수 있는 플랫폼으로 Apache 오픈 소스 프로젝트로 개발되었다. Hadoop은 2005년 Apache의 Nutch 오픈 소스 검색엔진의 대용량의 웹 데이터를 처리하기 위해 고안되었다가 2008년 Apache Top-Level 프로젝트로 승격되었다. Hadoop은 HDFS, HBase, Hive 등의 많은 컴포넌트를 지원하며, 데이터 처리방식으로 MapReduce 소프트웨어 프레임워크를 사용한다.

본 논문에서는 Hadoop의 설치와 구동, 예제 애플리케이션 실행을 위한 과정을 리눅스 명령어와 함께 상세히 설명한다.

본 논문의 구성은 다음과 같다. 2장에서는 HDFS와 MapReduce 프로그래밍 모델에 대하여 설명하고 3장에서는 Hadoop 클러스터 구축을 위한 과정을 상세히 설명한다. 4장에서는 Hadoop의 예제 애플리케이션에 대해 설명한 후, 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

Hadoop의 설치에 앞서 분산파일시스템과 분산 처리에 대해 살펴본다. 대표적인 분산파일시스템으로는 구글의 GFS[6]와 Hadoop의 HDFS가 있다. 대표적인 분산 처리는 구글의 MapReduce[7]가 있는데 Hadoop의 MapReduce는 구글의 MapReduce 기술을 모방하여 만들어진 것이며 동작 방식도 동일하다.

1. HDFS(Hadoop Distributed File System)

HDFS는 저비용의 수백 내지 수천 노드를 가지는 클러스터를 이용하여 기가바이트 또는 테라바이트의 대용량 데이터 집합을 처리하는 응용 프로그램에 적합하도록 설계한 분산파일시스템이다. HDFS는 마스터-슬레이브 구조로 동작하며, 노드간에 TCP/IP 프로토콜을 사용하여 통신한다. 또한

노드 실패에 대비하여 데이터를 복제하여 저장한다. 이 HDFS의 구조는 그림 1과 같다.

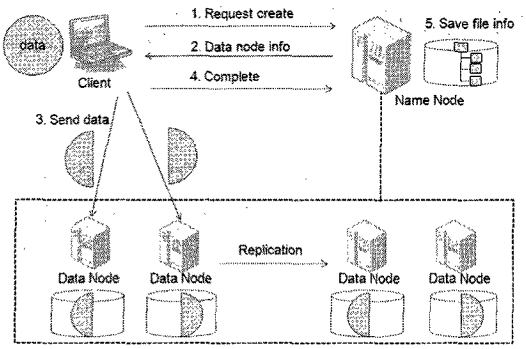

그림 1. HDFS 구조
Fig. 1. HDFS Architecture

그림 1에서 단일 네임노드(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.

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 클라이언트-네임 노드간의 통신은 TCP/IP 프로토콜을 사용한다. 클라이언트는 네임 노드에서 설정한 TCP 포트를 통하여 네임 노드와 통신한다.

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 key값을 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. 본론

Hadoop 기반의 클러스터 구축 방식에는 단일 구성 방식, 가상 분산 방식, 완전 분산 방식이 있다. 먼저 단일 구성 방식은 비분산 모드로 Hadoop을 하나의 로컬시스템에서 자바 프로세스로 실행하는 방식으로 주로 Hadoop 기반의 응용프

로그램 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 방식의 Hadoop 클러스터 구축에 대한 과정을 리눅스 명령어와 함께 상세히 설명한다. 본 논문에서 사용한 클러스터의 환경은 표 1과 같다.

표 1. 클러스터 환경

노드 수	네임 노드	1개
	데이터 노드	10개
노드 성능	CPU 성능	2.4 Ghz
	RAM 용량	768 MB
운영체제	Linux(Ubuntu 9.04 Server)	
Hadoop	Hadoop-0.19.1	

1. 소프트웨어 다운로드 및 설치

Hadoop을 설치하기에 앞서 모든 노드에 동일한 사용자를 등록한다. 본 논문에서는 'user'라는 사용자를 등록하였다. 그리고 모든 노드에 http://hadoop.apache.org 에서 hadoop을 다운로드한다. 다운로드한 압축파일을 'user' 폴더 아래에 압축을 푼다. 그리고 편의상, hadoop의 폴더명을 'hadoop'으로 변경하였다. 따라서 모든 노드에서 hadoop의 위치는 '/home/user/hadoop/'이 된다.

Hadoop은 Java로 개발되었기 때문에 모든 노드에 Java를 설치한다.

```
$ sudo apt-get install sun-java6-jdk
```

그리고 Hadoop은 ssh를 통해 수행 명령을 송수신하기 때문에 ssh를 설치한다.

```
$ sudo apt-get install ssh
```

2. Hadoop 환경 설정

Hadoop의 환경설정을 위해 hadoop-env.sh을 수정한다. conf/hadoop-env.sh 파일에 JAVA_HOME의 주석을 제거하고 Java가 설치된 폴더로 설정한다.

```
$ vi /home/user/hadoop/conf/hadoop-env.sh
export
JAVA_HOME=/usr/lib/jvm/java-6-sun-1.6.0.10
```

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 2와 같다.

그림 2에서 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 2에서는 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 3이다. dfs.block.size는 데이터 블록의 크기이며 기본적으로 64MB이다. 그리고 그 하위의 속성은 HDFS와 MapReduce가 생성하는 데이터가 저장될 폴더를 정의하는데, Hadoop을 테스트하는 도중, 이상 발생시 dfs 폴더를 삭제하고, Name node의 정보를 포맷하고 다시 실행하면 된다.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://192.168.0.1:9000</value>
  </property>
  <property>
    <name>mapred.job.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>
  <property>
    <name>dfs.name.dir</name>
    <value>/home/user/hadoop/dfs/name</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/user/hadoop/dfs/data</value>
  </property>
  <property>
    <name>mapred.system.dir</name>
    <value>/home/user/hadoop/dfs/mapreduce/system
    </value>
  </property>
  <property>
    <name>mapred.local.dir</name>
    <value>/home/user/hadoop/dfs/mapreduce/local
    </value>
  bb</property>
</configuration>
```

그림 2. hadoop-site.xml 설정

다음으로 Name node와 Data node의 IP를 설정한다.

```
$ cd /home/user/hadoop/conf
$ vi masters
192.168.0.1
```

```
$ cd /home/user/hadoop/conf
$ vi slaves
192.168.0.2
192.168.0.3
192.168.0.4
...
```

이로써 Hadoop의 설정은 완료되었지만, 이를 모든 노드에 똑같이 적용하는 것은 매우 번거로운 일이다. 따라서 한 노드에서 위와 같이 설정을 끝낸 후, hadoop 폴더를 압축하여 각 노드로 전송한 후, 각 노드에서 압축을 해제하면 된다.

```
$ cd /home/user
$ tar -cvf hadoop.tar hadoop/
$ gzip -9 hadoop.tar
$ scp /home/user/hadoop.tar.gz
hadoop@IP_ADD:/home/user/hadoop.tar.gz
```

위의 명령어 중 IP_ADD는 전송할 노드의 IP 주소이다.

3. System 설정

Hadoop은 시스템의 이름을 통해 통신을 하기 때문에 hosts 파일에 클러스터에 참여하는 모든 노드의 시스템 이름과 IP가 나열되어야 한다.

```
$ sudo vi /etc/hosts
192.168.0.1 hadoop-N01
192.168.0.2 hadoop-N02
...
```

그리고 Hadoop에서 ssh를 이용하여 각 노드의 실행 메시지를 보내는데, 인증키가 없다면 매번 연결할 때마다 패스워드를 묻게 된다. 이를 간편하게 하기 위해 다음과 같이 Name node에서 ssh- keygen을 실행한다.

```
$ ssh-keygen -t rsa (이후 계속 엔터)
$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
$ scp /home/user/.ssh/authorized_keys
hadoop@IP_ADD /home/user/ .ssh/authorized_keys
```

위와 같이 Name node에서 RSA키를 생성한 후, 다른 모든 노드에 authorized_keys를 보내준다.

4. Hadoop 실행

모든 설정이 완료 되었다면, Hadoop을 실행하기 전 Name node가 가진 정보를 초기화한다.

```
$ cd /home/user/hadoop
$ bin/hadoop namenode -format
```

만약, dfs에 접근을 못하거나 에러가 발생한다면 Hadoop을 중단하고, hadoop/dfs 폴더(hadoop-site.xml에 정의한 속성)를 삭제한 후, 위와 같이 초기화를 하면 해결할 수 있다. 단, HDFS에 저장된 파일은 모두 삭제된다.

아래와 같이 start-all.sh를 실행하면 Hadoop이 실행된다. start-mapred.sh 와 start-dfs.sh 는 각각 MapReduce, HDFS 를 실행하는데, start-all.sh 를 실행하면 모두 함께 실행된다. Hadoop의 중단도 동일한 형태이며, 명령어는 아래와 같다. 그리고 Hadoop의 주요 명령어는 표 2와 같다.

```
$ bin/start-all.sh
```

```
$ bin/stop-all.sh
```

표 2. Hadoop의 주요 명령어

DFS 파일보기	bin/hadoop dfs -ls
DFS 파일삭제	bin/hadoop dfs -rmr FILENAME
Job 리스트	bin/hadoop job -list
Job 중지	bin/hadoop job -kill JOB_ID

웹브라우저에서 http://Name node_IP:50070 으로 접속하면 HDFS의 정보를 볼 수 있다. 또한, MapReduce 작업에 대한 정보는 http://Name node_IP:50030/ 으로 접속하여 확인할 수 있다.

Ⅳ. 애플리케이션 구동

본 논문에서는 Hadoop의 기본적인 예제 애플리케이션인 WordCount를 통해 MapReduce의 동작에 대해 설명한 후, Hadoop의 수정 및 리빌드에 대해 추가적으로 설명한다. WordCount는 입력 파일을 읽어서 블록 단위로 단어 빈도수를 계산하는 애플리케이션이다.

1. WordCount의 동작

WordCount의 실행에 앞서 WordCount의 동작을 살펴본다. 그림 4는 WordCount의 주요 동작에 관련된 부분인 map 함수와 reduce 함수를 의사코드로 표현한 것이다. Hadoop의 애플리케이션을 개발할 때, map 함수와 reduce 함수를 오버라이딩(Overriding)하여 사용하기 때문에 단순한 코드만으로 분산 처리를 구현할 수 있다.

그림 3의 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 합산하여 단어 수를 계산한다. 그리고 그 결과를 output에 저장한다.

```
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 3. WordCount 애플리케이션 의사코드

2. WordCount의 실행

3장에서 설명한 것과 같이 Hadoop이 실행된 상태에서 WordCount를 실행한다. WordCount의 입력으로 사용할 텍스트는 폴더형태나 파일형태에 관계없이 HDFS내에 존재하기만 하면 된다. 먼저, 아래의 명령을 사용하여 입력으로 사용할 폴더나 텍스트를 HDFS에 저장한다. 아래의 예에서는 Hadoop의 'conf' 폴더 및 하위 파일들을 HDFS에 'input'으로 저장하고 있다.

```
$ bin/hadoop dfs -put conf input
```

이제 WordCount를 실행한다

WordCount는 hadoop-*-examples.jar에 포함되어 있으며, WordCount의 첫 번째 파라메터는 단어를 카운트할 HDFS 내의 파일, 두 번째 파라메터는 결과를 저장할 파일이다.

```
$ bin/hadoop jar hadoop-*-examples.jar
wordcount input output
```

위와 같이 WordCount가 실행이 완료되면 실행결과는 HDFS의 'output'으로 저장되는데, HDFS에 저장된 데이터를 로컬로 복사하는 명령어는 아래와 같다.

```
$ bin/hadoop dfs -get output localout
```

그림 4, 5, 6은 3장에서 언급한 바 있는 웹브라우저에서 Hadoop의 정보를 나타낸 것이다. 그림 4는 HDFS의 정보를 확인한 화면이고, 그림 5는 Hadoop에서 수행중이거나 수행된 작업을 모두 보여주고, 그림 6은 특정 작업에 대한 정보를 나타낸다.

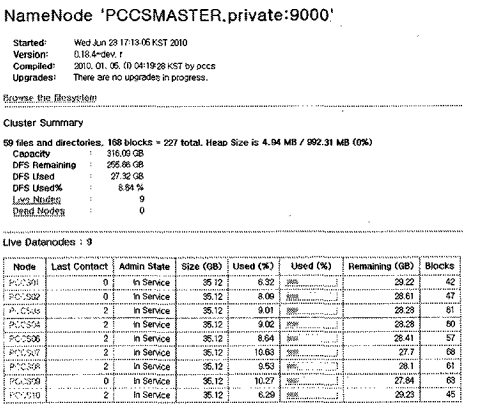

그림 4. HDFS 정보

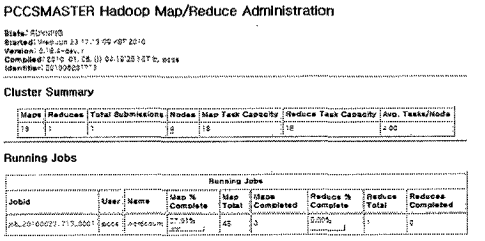

그림 5. Hadoop 작업 리스트

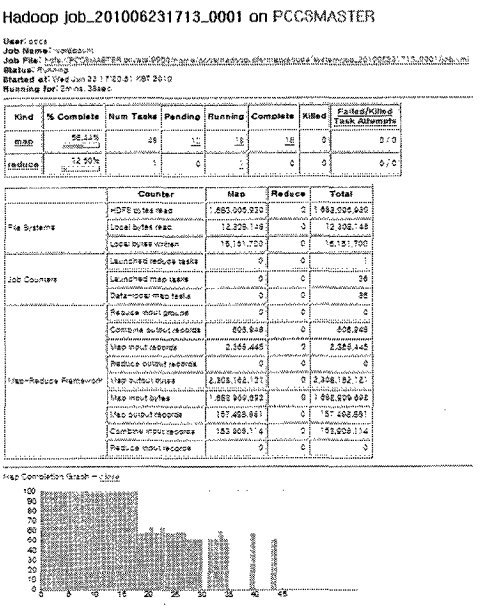

그림 6. 작업에 대한 세부 정보

3. Hadoop의 수정 및 리빌드

Hadoop은 오픈 소스 프로젝트로 코드가 공개되어 있다. 따라서 사용자는 자신의 클러스터 환경에 맞게 코드를 수정할 수 있는데, 본 논문에서는 간단한 예를 통해 Hadoop을 수정하고 리빌드한다. 먼저, 아래와 같이 Apache의 자바 기반 빌드 도구인 Ant를 설치한다.

```
$ sudo apt-get install ant
```

다음으로 /home/user//bashrc 파일을 수정한다.

```
$ vi /home/user/.bashrc
export ANT_HOME=Ant 설치디렉토리
export PATH=$ANT_HOME/bin:$PATH
```

만약, 애플리케이션을 실행 중에 Reduce가 실행이 되지 않는다면 Map과의 연결성을 검사해야 할 것이다. 먼저, Reduce수행시 Map의 Output을 가져오는 루틴에서 연결성을 검사한다고 가정한다. 따라서 /home/user/hadoop/src/mapred/org/apache/hadoop/mapred/ReduceTask.java 파일을 수정한다. 1225라인 근처의 getMapOutput 함수로 이동하여 아래와 같이 로그 출력 코드를 추가한다.

```
private MapOutput
getMapOutput(MapOutputLocation
             mapOutputLoc, Path filename)
    throws IOException, InterruptedException {
// Connect
URLConnection connection =
mapOutputLoc.getOutputLocation().openConnection
();

LOG.warn("TEST LOG: " +

connection.getURL().toString());

InputStream input =
getInputStream(connection,
               DEFAULT_READ_TIMEOUT,
               STALLED_COPY_TIMEOUT);
```

수정을 완료하고, 아래와 같이 빌드를 한다.

```
$ cd /home/user/hadoop
$ ant package
```

빌드가 완료되면 ./build 폴더에 현재 버전보다 높은 버전이 만들어져 있다. 이것을 아래와 같이 복사한다.

```
$ cp ./build/hadoop-core-버전.jar ./현재파일명.jar
```

그리고 다시 Hadoop을 시작하고, 애플리케이션을 실행하면 Reduce 함수에서 Map 데이터를 가져올 때마다 해당 Map의 URL을 출력한다.

V. 결론

본 논문에서는 클라우드 컴퓨팅을 위한 플랫폼인 Hadoop의 설치와 실행에 대해 상세히 설명했다, 또한 Hadoop의 예제 애플리케이션인 WordCount를 통해 MapReduce의 동작을 살펴보았다.

본 논문은 http://hadoop.apache.org의 Hadoop Guide[9]에서 설명한 기본적인 사항들을 바탕으로 하였다. 하지만, Hadoop Guide에 세부적인 사항들이 기술되지 않아, Hadoop 설치 및 실행에 다소 어려운 점이 존재한다. 본 논문에서는 예제를 이용한 구축 단계를 상세히 기술하였기 때문에 실제 클라우드 환경을 구축하는데 많은 도움이 될 것이다.

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/pressrel

ease/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] Hadoop, http://hadoop.apache.org/common/docs/r0.19.2/, 2009

저 자 소 개

금 태 훈

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

이 상 준

1989: 한양대학교 전자계산학과 공학사.
1991: Univ. of Utah Mechanical Eng. 공학석사.
1991: Arizona State Univ. Mechanical & Aerospace Eng. 공학박사.
현 재: 평택대학교 물류정보대학원 전임강사.
관심분야: 물류정보시스템, 유비쿼터스, RFID/USN, 클라우드컴퓨팅.

김 세 회

2009: 한양대학교 전자컴퓨터공학부 학사.
현 재: 한양대학교 컴퓨터공학과 석사과정.
관심분야: 클라우드컴퓨팅, 분산파일시스템.