



아파치 스파크 소개

순천향대학교 컴퓨터공학과

이 상 정



순천향대학교 컴퓨터공학과

1

아파치 스파크 소개

학습 내용

1. 스파크 소개
2. 스파크 컴포넌트
3. 하둡 YARN에서 스파크 설치

순천향대학교 컴퓨터공학과

2

1. 스파크 소개

아파치 스파크 소개

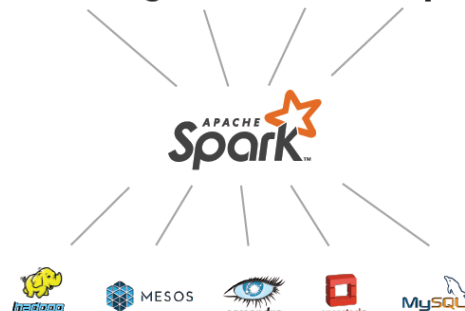


아파치 스파크 (Apache Spark) 개요

□ 스파크는 분산 데이터 처리를 위한 통합 엔진

- 2009년 버클리 대학에서 시작한 프로젝트
- 맵리듀스(MapReduce)를 확장하여 각기 독립적인 엔진들이 수행 하던 SQL, 스트리밍, 기계학습, 그래프 처리 등의 컴포넌트들을 통합
- 메모리 상에서 실행
 - 한 컴포넌트들의 결과를 HDFS와 같은 저장소에 출력하고, 다른 컴포넌트가 이를 읽어 들여 처리
 - 스파크는 메모리에 상주하는 동일한 데이터 상에서 컴포넌트들이 다양한 함수를 수행

Streaming SQL ML Graph



스파크 주요 특징 (1)

□ 빠른 성능

- 반복 알고리즘(iterative algorithm) 수행 시 디스크를 경유하여 데이터를 전달하지 않고 **메모리 상에 데이터를 상주**
- 기존 맵리듀스와 비교해서, Spark는 디스크 상에서 10배, 메모리 상에서는 100배 이상 빠른 성능을 달성

□ 개발의 편의성

- 고난도 데이터 처리 알고리즘의 신속한 구축이 가능
 - 맵리듀스 보다 **풍부한 연산과 라이브러리** 제공
 - 적은 코드로 알고리즘 구현
- 개발 테스트와 디버깅이 용이
 - 스칼라(Scala),파이썬 등을 사용한 **상호작용 (interactive shell) 셸**을 제공

스파크 주요 특징 (2)

□ 유연한 실행 환경

- **하둡 빅데이터 환경**에서 실행
 - YARN 환경에서 HDFS, MapR-FS, HBase, HIVE 등에 저장된 데이터 처리
- 다양한 실행 환경 지원
 - 아파치 Mesos(오픈소스 클러스터 관리자), 아마존 S3(스토리지 서비스)
- 단일 컴퓨터에서 로컬로도 실행

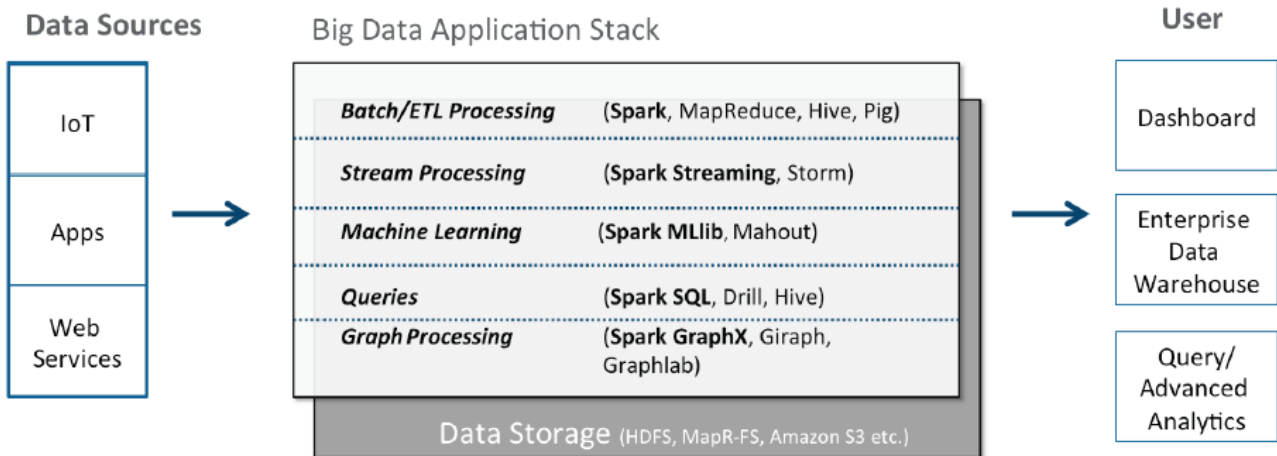
□ 통합 개발 환경

- 그래프 처리, 고급 질의, 스트림 처리, 기계학습 등과 같은 고수준 분석 도구를 위한 **통합 프레임워크**
- 전체 작업 과정에서 단일 프로그래밍 언어를 사용하여 하나의 응용으로 이들 라이브러리들의 결합이 가능

□ 다양한 프로그래밍 언어 지원

- 스칼라, 파이썬, 자바, SparkR

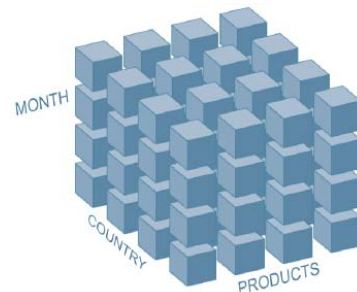
스파크와 빅 데이터



스파크 사용 사례 (1)

OLAP (OnLine Analytical Processing) 분석

- 서비스 제공자가 스파크를 사용하여 실시간 다차원 OLAP 분석
 - OLAP은 의사결정 지원 시스템으로 동일한 데이터를 여러 각도로 분석하는 다차원 데이터 분석을 지원
- 어떤 형식의 데이터도 수집하여 처리
- 1-2 TB 까지의 대규모 데이터 분석



운영 분석 (Operational Analytics)

- 보험회사는 의료 기록과 함께 환자 정보를 저장
- 스파크가 환자의 재입원 확률을 계산
- NoSQL을 사용하여 실시간 분석
 - 재입원 확률이 높은 경우 재택 간호 등과 같은 부가 서비스 제공

스파크 사용 사례 (2)

❑ 복잡한 데이터 파이프라이닝

- 제약회사에서 유전자 염기서열 분석에 스파크를 사용
- 스파크 상에서 수행되는 ADAM 툴을 사용하여 염기서열 일치여부 분석 처리에 몇 주 걸리는 작업을 수 시간으로 단축
- 맵리듀스를 사용하지 않고 복잡한 기계학습



스파크 사용 사례 (3)

❑ 배치 처리 (Batch Processing)

- (로그 파일과 같은) 원시 데이터 형식을 좀 더 구조화된 데이터 형식으로 변환하는 ETL(Extract-Transform-Load) 워크로드 등과 같은 대용량의 데이터 세트를 사용하는 배치 처리
 - Yahoo의 개인화된 페이지 및 추천
 - Goldman Sachs에서의 데이터 레이크(data lake) 관리
 - Alibaba의 그래프 마이닝(graph mining)
 - Riusk Calculation의 Financial Value
 - Toyota의 고객의 피드백 텍스트 마이닝
- 가장 큰 대규모 활용 사례
 - Chinese social network Tencent의 8000개 노드의 클러스트 사용 사례
 - 매일 1PB의 데이터를 수집

스파크 사용 사례 (4)

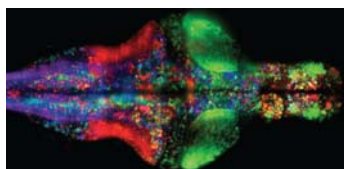
□ 스트림 처리 (Stream Processing)

- 실시간 분석 및 의사 결정 응용 등에서 요구되는 **실시간 처리**
 - Cisco의 네트워크 보안 모니터링
 - 삼성 SDS의 처방전 분석
 - Netflix의 로그 마이닝
- 많은 응용들이 **스트리밍을 배치와 상호 작용 질의와 결합**하여 사용
 - Conviva 비디오 회사는 콘텐츠 분배 서버의 성능 모델을 계속 유지보수 하면서, 클라이언트들이 서버들 간에 이동하는지 여부의 질의 처리를 병렬로 수행하는 응용에 스파크 사용

스파크 사용 사례 (5)

□ 과학 응용 (Scientific applications)

- 대규모 스팸 검출, 이미지 프로세싱, 게놈 데이터 처리 등의 **과학 영역**에 스파크 활용
- 배치와 상호 작용 및 스트림 처리를 결합하여 사용하는 응용 예로는 Howard Hughes Medical Institute의 신경 과학용 **Thunder 플랫폼**이 있음
 - 실시간으로 뇌-이미지 데이터를 처리하도록 설계되었으며, 제브라피시 (줄무늬가 있는 열대어), 생쥐 등의 전체 뇌 이미지 데이터 처리에 최대 1 TB/hour 속도로 처리
 - 과학자들은 Thunder를 사용해서 특정 동작에 관여하는 뉴런의 구분하는 기계학습 - 클러스터링과 주성분 분석(Principal Component Analysis, PCA) - 을 적용



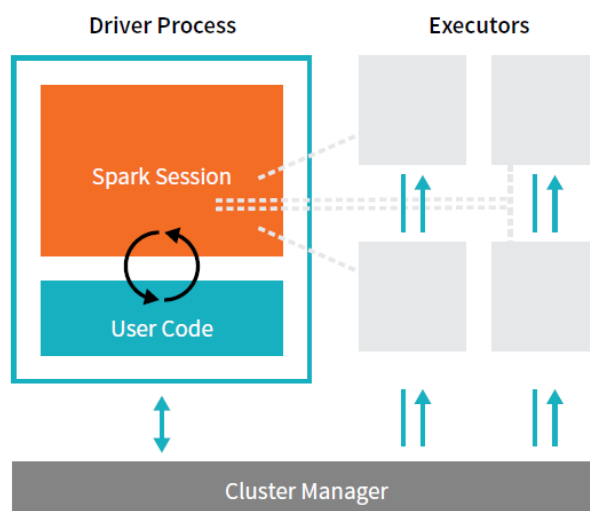
2. 스파크 컴포넌트

아파치 스파크 소개

스파크 응용 구조 (1)

□ 스파크 응용 주요 구성 요소

- 클러스터 관리자(cluster manager)
- 드라이버(driver) 프로세스
- 실행자(executor) 프로세스



스파크 응용 구조 (2)

❑ 클러스터 관리자

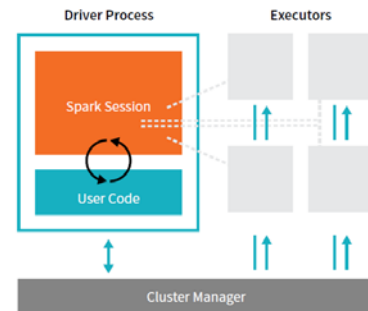
- 스파크 응용의 **자원을 할당**하고 **머신을 관리**

❑ 드라이버 프로세스

- SparkSession** 인스턴스를 생성하여 관리
- 스파크 응용에 **관한 정보** 관리
- 사용자 프로그램 응답
- 작업을 분석**하고, 실행자 프로세스들에 분산하고 **스케줄링**

❑ 실행자 프로세스(executor process)

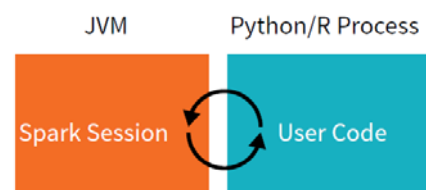
- 드라이버가 할당한 **코드를 실행**
- 계산의 상태를 드라이버에 보고



프로그래밍 언어 API

❑ 스파크는 다양한 프로그래밍 언어의 API를 제공

- 스칼라 (Scala)**
 - 스파크는 **스칼라 언어로 구현**되었고, 스파크의 디폴트 프로그래밍 언어
- 파이썬 (Python)**
 - 스칼라와 거의 유사할 정도로 파이썬을 지원
- SQL**
 - 데이터프레임(dataframe) 자료에 대해 SQL 사용
- 자바**
- R**
 - 통계 계산 프로그래밍 언어 R 지원



스파크 데이터셋

□ 데이터셋(Dataset)은 스파크의 기본 추상화로 클러스터의 노드에 분산된 객체들의 컬렉션

- 데이터셋 상에서 연산 수행
- 데이터셋이 생성된 후에는 변경 불가능 (immutable)
- 데이터셋은 디스크 또는 메모리 상에서 저장 및 캐싱
- 한 노드의 태스크가 실패하면 데이터셋은 나머지 노드에서 자동으로 재구축되어 작업을 완료



데이터셋 연산

□ 데이터셋 상에서 변환(transformation)과 액션(action)을 수행

- 기존의 데이터셋을 변환하여 새로운 데이터셋을 생성



- 액션은 수행된 결과 값을 리턴



- 변환과 액션을 임의의 순서로 결합하여 데이터를 처리하고 분석

스파크와 빅데이터

Data Sources	Big Data Application Stack	Output
IoT	Batch/ETL Processing: Spark , MapReduce, Hive, Pig	Dashboard
	Stream Processing: Spark Streaming , MapR-ES, Storm	
Apps	Machine Learning: Spark MLlib , Mahout	Query/ Advanced Analytics
Web Services	Queries: Spark SQL , Drill, Hive	Enterprise Data Warehouse
	Graph Processing: Spark GraphFrames , Giraph, Graphlab	

스파크 라이브러리 (1)

Component	Function
Spark SQL	<ul style="list-style-type: none"> Structure Data Querying with SQL/HQL
Spark Streaming	<ul style="list-style-type: none"> Processing of live streams Micro-batching
MLlib	<ul style="list-style-type: none"> Machine Learning Multiple types of ML algorithms
GraphFrames	<ul style="list-style-type: none"> Graph processing Graph parallel computations
Spark Core <ul style="list-style-type: none"> Task scheduling Memory management Fault recovery Interacting with storage systems 	

스파크 라이브러리 (2)

- 스파크 코어는 태스크 스케줄링, 메모리 관리, 고장 복구, 저장 시스템 접근 등을 수행하는 계산 엔진
 - 데이터세트를 정의하고 관리하는 API도 포함
- 스파크 SQL은 구조화된 데이터 상의 작업을 수행
 - 구조화된 Hive 테이블, 복잡한 JSON 데이터 등과 같은 데이터 소스의 유형 지원
- 스파크 스트리밍은 데이터 스트림들을 처리하고 실시간 분석
- MLib는 분류(classification), 회귀분석(regression), 클러스터링(clustering) 등의 다양한 머신러닝 알고리즘을 구현한 라이브러리
- 그래프프레임은 그래프 관련 병렬 계산을 수행하는 라이브러리

3. 하둡 YARN에서 스파크 설치

스파크의 실행 동작 모드

□ 스파크 실행 모드

- 로컬 모드 (local mode)
 - 로컬 머신의 하나의 JVM에서 실행
- 독립형 모드 (standalone mode)
 - 독립적인 클러스터 상에서 실행
- 하둡 YARN
 - 하둡 YARN 상에서 실행
- 아파치 Mesos
 - 아파치 Mesos 자원 관리자 상에서 실행

□ 여기서는 하둡 YARN 상의 설치 소개

스파크 다운로드 및 압축해제

□ 마스터 노드에서 설치한 후에 슬레이브 노드에 복사

- 스파크 2.4.5 설치
 - 제플린 노트북과의 연동 호환성을 고려
 - 최신 버전은 2.4.5 (2020년 3월 기준)
- 설치 디렉토리: `~/spark-2.4.5-bin-hadoop2.7`

□ 스파크 2.4.5 다운로드

- <http://spark.apache.org/downloads.html>

```
$ wget http://mirror.apache-kr.org/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz
```

□ 압축 해제 (tar.gz, tgz)

```
$ tar -xvzf spark-2.4.5-bin-hadoop2.7.tgz
```

환경 변수 설정

□ 하둡 환경 변수 설정

- ~/.bashrc에 추가

```
#Spark Path
export SPARK_HOME=$HOME/spark-2.4.5-bin-hadoop2.7
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native

export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib
/py4j-0.10.7-src.zip:$PYTHONPATH
export PYSPARK_PYTHON=python3
```

- 추가 후 아래 명령을 통해 적용
\$ source ~/.bashrc

스파크 설정 - 설정 파일 복사

□ 기존 스파크 설정 파일들 복사 & 변경

- 경로 : ~/spark-2.4.5-bin-hadoop2.7/conf/
- slaves.template
- spark-defaults.conf.template
- spark-env.sh.template

```
$ cp slaves.template slaves
$ cp spark-defaults.conf.template spark-defaults.conf
$ cp spark-env.sh.template spark-env.sh
```

□ 스파크 환경 변수 설정

- 경로 : ~/spark-2.4.5-bin-hadoop2.7/conf/spark-env.sh

```
export HADOOP_CONF_DIR=${HADOOP_DIR}/etc/hadoop
export SPARK_WORKER_MEMORY=2g
```

- \$ source ~/spark-2.4.5-bin-hadoop2.7/conf/spark-env.sh

스파크 설정 - 설정 파일 변경

스파크 설정 파일 변경

- 경로 : `~/spark-2.4.5-bin-hadoop2.7/conf/spark-defaults.conf`

```
spark.master      spark://master:7077
spark.yarn.jars    hdfs://master:9000/jar/spark-jars/*.jar
```

스파크 슬레이브 설정

- 경로 : `~/spark-2.4.5-bin-hadoop2.7/conf/slaves`

```
slave1
```

스파크 설정 - jar 파일 적재

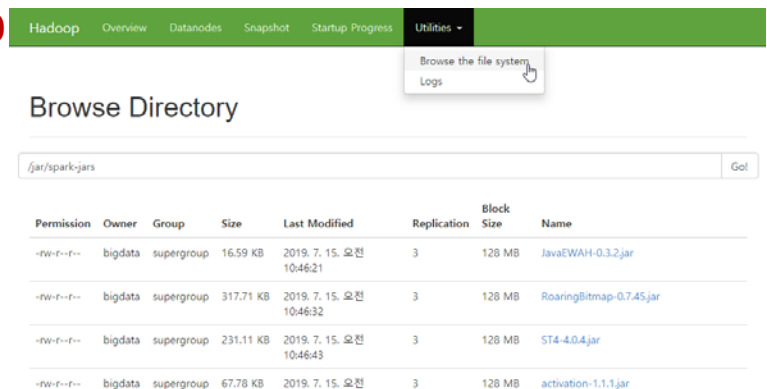
하둡을 실행하고 모든 스파크의 jar 파일을 하둡의 파일 시스템으로 적재

- `--master yarn` 모드 실행을 위한 작업

```
$ hadoop fs -mkdir /jar
$ hadoop fs -mkdir /jar/spark-jars
$ hadoop fs -put $SPARK_HOME/jars/* /jar/spark-jars/
```

네임노드 웹 접속하여 파일 복사 확인

- `http://192.168.0.1:50070`



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	bigdata	supergroup	16.59 KB	2019. 7. 15. 오전 10:46:21	3	128 MB	JavaEWAH-0.3.2.jar
-rw-r--r--	bigdata	supergroup	317.71 KB	2019. 7. 15. 오전 10:46:32	3	128 MB	RoaringBitmap-0.7.4.jar
-rw-r--r--	bigdata	supergroup	231.11 KB	2019. 7. 15. 오전 10:46:43	3	128 MB	ST4-4.0.4.jar
-rw-r--r--	bigdata	supergroup	67.78 KB	2019. 7. 15. 오전 10:46:43	3	128 MB	activation-1.1.1.jar

스파크 설정 - 스파크 배포

□ 스파크 설치 디렉토리 배포

- 각 서버로 스파크 설치 디렉토리를 배포

```
$ scp .bashrc slave1:~/
$ scp -r ~/spark-2.4.5-bin-hadoop2.7 slave1:~/
```

스파크 실행 및 동작 확인 - jps

□ 스파크 실행

- `$SPARK_HOME/sbin/start-all.sh`

```
bigdata@master:~$ $SPARK_HOME/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/bigdata/spark-2.3.3-bin-hadoop2.7/logs/spark-bigdata-org.apache.spark.deploy.master.Master-1-master.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /home/bigdata/spark-2.3.3-bin-hadoop2.7/logs/spark-bigdata-org.apache.spark.deploy.worker.Worker-1-slave1.out
bigdata@master:~$
```

- 스파크 중지: `$ $SPARK_HOME/sbin/stop-all.sh`

□ jps 확인

```
bigdata@master:~$ jps
3904 Jps
2897 ResourceManager
3079 NodeManager
2441 NameNode
2617 DataNode
3854 Master
bigdata@master:~$
```

```
bigdata@slave1:~$ jps
5632 Worker
5426 NodeManager
5684 Jps
5098 DataNode
5276 SecondaryNameNode
bigdata@slave1:~$
```

스파크 실행 및 동작 확인 - 마스터 웹 접속

□ 스파크 마스터 포트 웹 접속 확인

- 192.168.0.1:8080



Spark Master at spark://master:7077

URL: spark://master:7077

REST URL: spark://master:6066 (cluster mode)

Alive Workers: 1

Cores in use: 1 Total, 0 Used

Memory in use: 2.0 GB Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190731013622-192.168.0.201-36703	192.168.0.201:36703	ALIVE	1 (0 Used)	2.0 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

순천향대학교 컴퓨터공학과

31

스파크 셸 (Spark Shell)

□ 스파크 셸은 사용자와 상호작용하며 프로그램 작성 가능

- 스칼라, 파이썬 셸
 - 마스터, 슬레이브 노드 어디서나 실행 가능
- 셸이 시작하면 **스파크세션(SparkSession)** 객체가 초기화되고, 변수 **spark**가 이를 가리킴
 - 스파크 세션은 한 응용의 클러스터 연결 접근 방식을 관리
- 실행 환경 옵션: **--master**
 - 하둡 클러스터: **--master yarn**
 - 로컬 PC: **--master local[N]**
 - N은 실행할 스레드 수
 - * 로 지정하면 가용한 CPU 코어 개수로 자동 지정
- 클러스터 실행 예

\$ \$SPARK_HOME/bin/spark-shell --master yarn

스파크 셀 실행

33

```
bigdata@slave1:~$
```

스파크 실행 및 동작 확인 - YARN 웹

- <http://192.168.0.1:8088>

가상 머신 실습 시작 및 종료 시 주의 사항 (1)

□ 실습 시작 시

- 마스터, 슬레이브 가상 머신 시작
- 가상 머신 마스터 로그인
- 마스터에서 하둡 시작: `$ start-all.sh`
- 마스터에서 스파크 시작: `$SPARK_HOME/sbin/start-all.sh`
- Putty로 마스터 연결
 - Putty 콘솔에서 실습

□ 실습 종료 시

- 마스터에서 스파크 종료: `$SPARK_HOME/sbin/stop-all.sh`
- 마스터에서 하둡 종료: `$ stop-all.sh`
- 마스터/슬레이브 가상 머신 종료: 저장 또는 전원 끄기

가상 머신 실습 시작 및 종료 시 주의 사항 (2)

□ 하둡 정상 종료하지 않고 마스터/슬레이브 종료하면 하둡 오류 발생

□ 하둡 오류 시 아래의 절차로 복구

- 스파크 종료: `$SPARK_HOME/sbin/stop-all.sh`
- 하둡 종료: `$ stop-all.sh`
- 하둡 데이터 디렉토리 삭제: `$ rm -r $HADOOP_HOME/hdfs/data`
- 하둡 재시작: `$ start-all.sh`
- 하둡 포맷: `$ hadoop namenode -format`
 - 기존의 하둡 데이터는 모두 삭제됨
- jar 파일 복사
 - `$ hadoop fs -mkdir /jar`
 - `$ hadoop fs -mkdir /jar/spark-jars`
 - `$ hadoop fs -put $SPARK_HOME/jars/* /jar/spark-jars/`
- 스파크 시작: `$SPARK_HOME/sbin/start-all.sh`

- 스파크를 사용하여 구현된 응용 사례 조사
- 강의 시간의 실습 내용을 정리하여 제출
 - 스파크 실행 및 동작 확인
 - 스파크 셀 실행

- MapR Academy, <http://learn.mapr.com/>
 - Introduction to Apache Spark
 - <https://learn.mapr.com/series/sparkv2/dev-360-introduction-to-apache-spark-spark-v21>
 - Lesson 1: Introduction to Apache Spark
- 2016년 Spark 소개 논문
 - Apache Spark: A Unified Engine for Big Data Processing, *Communications of the ACM*, 59(11):56–65, November 2016.
 - M. Zaharia, R. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica.
 - <http://cacm.acm.org/magazines/2016/11/209116-apache-spark/fulltext>