아파치 스파크 소개

순천향대학교 컴퓨터공학과

이 상 정

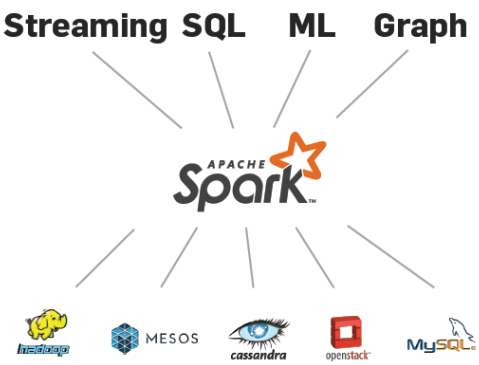
학습 내용

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

1. 스파크 소개

아파치 스파크 (Apache Apark) 개요

- 스파크는 분산 데이터 처리를 위한 통합 엔진
  - 2009년 버클리 대학에서 시작한 프로젝트
  - 맵리듀스(MapReduce)를 확장하여 각기 독립적인 엔진들이 수행하던 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



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
  - 멥리듀스 보다 풍부한 연산과 라이브러리 제공
  - 적은 코드로 알고리즘 구현
- 개발 테스트와 디버깅이 용이
  - 스칼라(Scala),파이썬 등을 사용한 상호작용 (interactive shell) 셸을 제공

스파크 주요 특징 (2)

❑ 유연한 실행 환경

- 하둡 빅데이터 환경에서 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

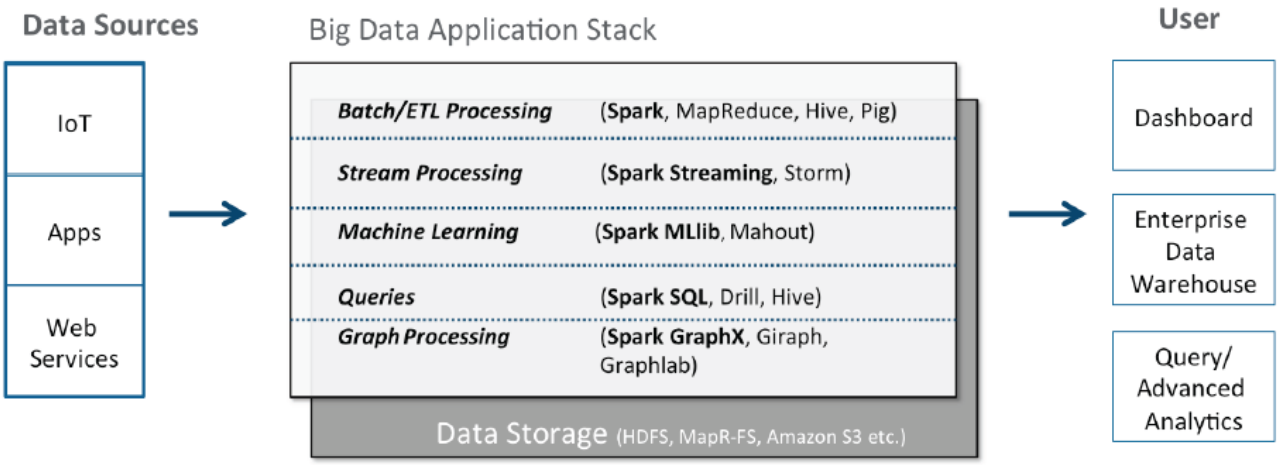
❑ 통합 개발 환경

- 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

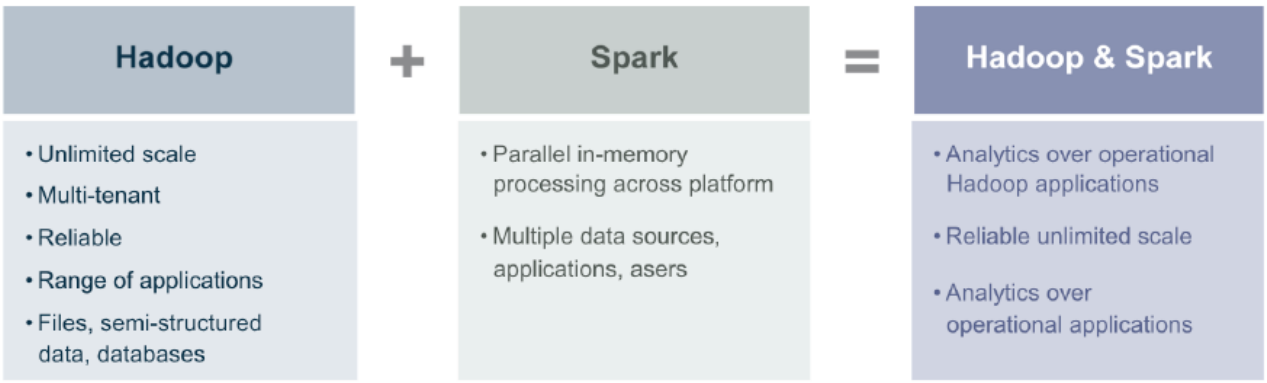
❑ 다양한 프로그래밍 언어 지원

- 스칼라, 파이썬, 자바, SparkR

스파크와 빅 데이터



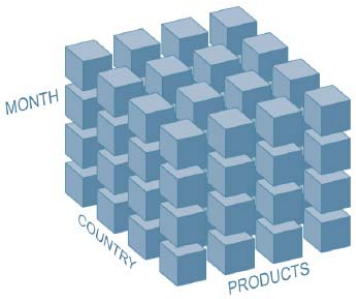
하둡 상의 스파크 (Spark on Hadoop)



스파크 사용 사례 (1)

❑ OLAP (OnLine Analytical Processing) 분석

- 서비스 제공자가 스파크를 사용하여 실시간 다차원 OLAP 분석
  - OLAP는 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)

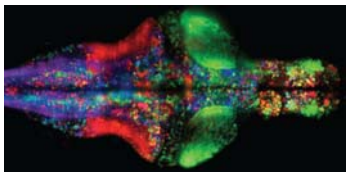
❑ 스트림 처리 (Stream Processing)

- 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 결합하여 사용
  - Conviva 비디오 회사는 콘텐츠 분배 서버의 성능 모델을 계속 유지보수하면서, 클라이언트들이 서버들 간에 이동하는지 여부의 질의 처리를 병렬로 수행하는 응용에 스파크 사용

스파크 사용 사례 (5)

❑ 과학 응용 (Scientific applications)

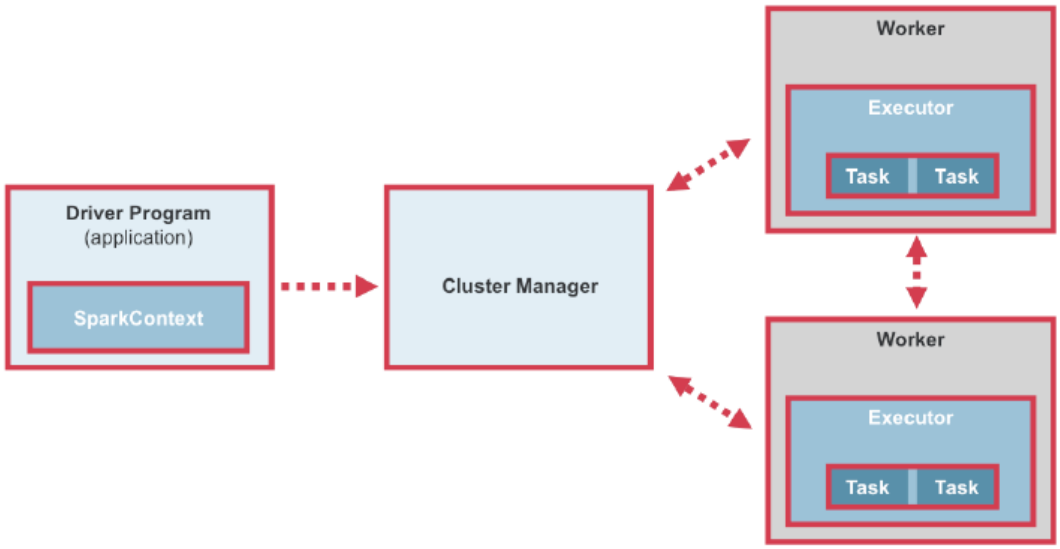
- 대규모 스팸 검출, 이미지 프로세싱, 게놈 데이터 처리 등의 과학 영역에 스파크 활용
- 배치와 상호 작용 및 스트림 처리를 결합하여 사용하는 응용 예로는 Howard Hughes Medical Institute의 신경 과학용 Thunder 플랫폼이 있음
  - 실시간으로 뇌-이미지 데이터를 처리하도록 설계되었으며, 제브라피시(줄무늬가 잇는 열대어), 생쥐 등의 전체 뇌 이미지 데이터 처리에 최대 1 TB/hour 속도로 처리
  - 과학자들은 Thunder를 사용해서 특정 동작에 관여하는 뉴런의 구분하는 기계학습 – 클러스터링과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

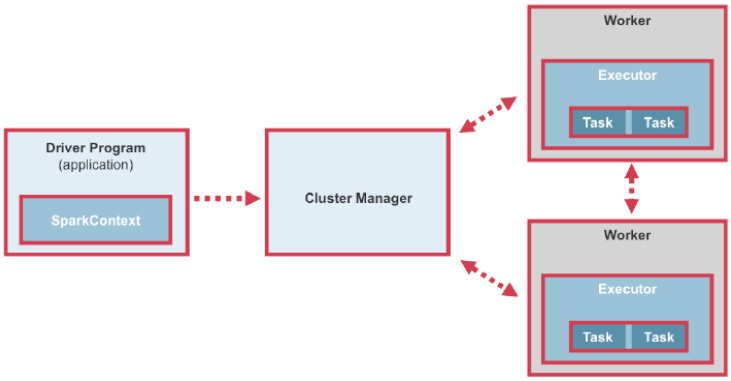


2. 스파크 컴포넌트

스파크 클러스터 구성

- 스파크 클러스터는 두 종류의 노드들로 구성
  - 하나의 드라이버 프로그램 노드
  - 다수의 작업자 노드 (worker nodes)
    - 클러스터의 각 노드는 실행자 프로세스 (executor process)를 실행



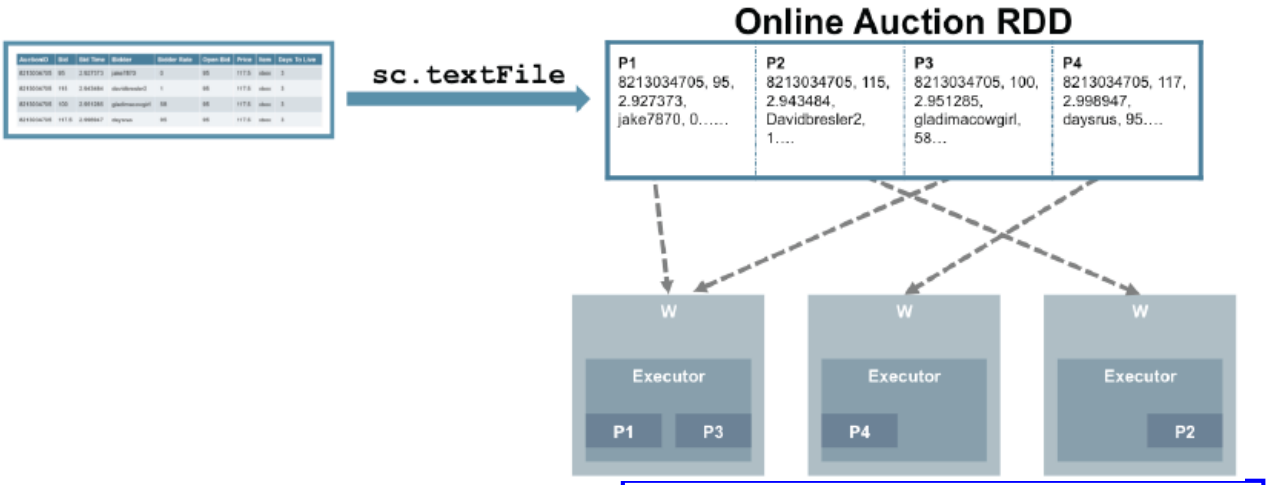


스파크 실행 순서

1. 드라이버 프로그램이 SparkContext 객체를 생성
   - SparkContext는 한 응용의 클러스터 연결 접근 방식을 관리
2. SparkContext가 클러스터 관리자(cluster manager)에 연결
   - 클러스터 관리자는 각 응용들에 대한 자원을 할당
3. 클러스터 관리자에 연결되면 스파크는 작업자(worker) 노드들의 실행자(executor)들을 획득
   - 실행자(executor)는 각 응용에 대한 계산과 데이터를 저장하는 프로세스
4. SparkContext에 전달된 Jar 또는 파이썬 파일들이 실행자에게 송신
5. SparkContext가 실행자가 수행할 태스크들을 송신
6. 작업자 노드들은 데이터 저장소를 접근하여 데이터를 수집하고 출력

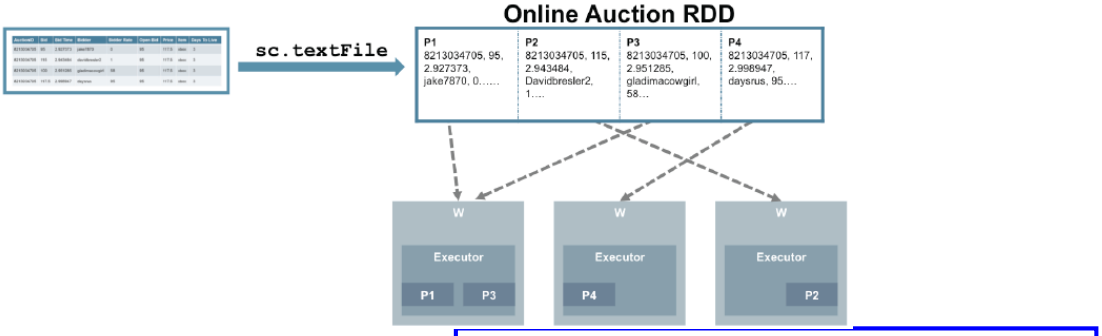
RDD (Resilient Distributed Dataset) (1)

- RDD는 스파크의 핵심적인 프로그래밍 추상화로 병렬로 처리될 수 있도록 클러스터 상에서 분할된 고장-감내형 객체들의 컬렉션 (fault-tolerant collections of objects partitioned across a cluster)



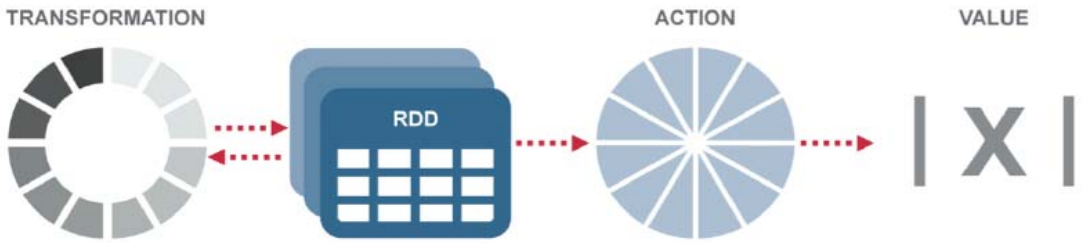
RDD (Resilient Distributed Dataset) (2)

- 일단 생성된 RDD는 수정 불가능 (immutable)
- RDD는 디스크에 지속 저장(persist)되거나 메모리에 캐시 될 수 있음
- 스파크 RDD는 고장 감내(fault-tolerant)를 지원하여, 한 노드나 태스크가 실패하면 자동으로 나머지 노드들에 RDD가 재구축되어, 수행 중인 작업을 완료



RDD 연산

- RDD 상에서는 변환(transformation)가 액션(action)의 두 가지 연산이 수행
  - 변환의 결과로 RDD를 리턴. RDD는 수정 불가능이므로 새로운 RDD를 리턴
  - 액션은 수행된 결과 값을 리턴



RDD 연산 예

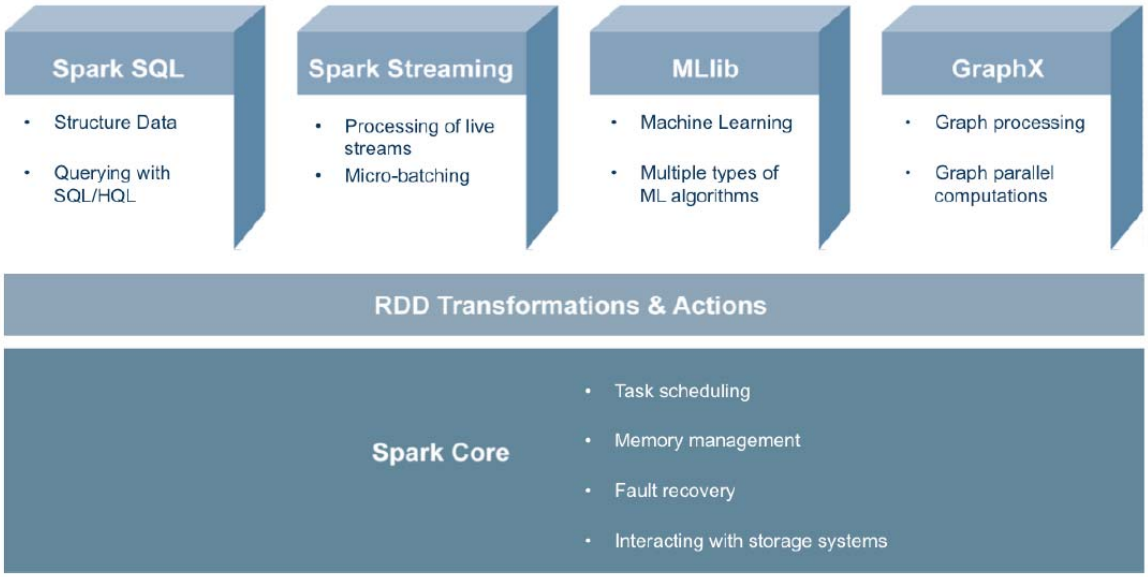
- 다음은 로그 파일에서 ERROR로 시작하는 라인들을 검색하여 에러 메시지를 나타내는 RDD를 생성하고, 전체 에러의 수를 출력하는 스칼라 코드 예

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(
  s => s.startsWith("ERROR"))
println("Total errors: "+errors.count())
```

  - 첫 줄은 HDFS (Hadoop Distributed File System)에 저장된 파일을 가져와서 RDD를 생성
  - 두 번째 줄은 lines RDD의 필터 변환을 통해 새로운 RDD를 생성
  - 세 번째 줄은 RDD 액션 count를 호출하여 프로그램의 결과 (RDD의 요소 수)를 리턴

스파크 시스템 구성 (1)

- 스파크 시스템은 스파크 코어(Spark Core), RDD 관리 API, 고수준 라이브러리(High-Level Library) 로 구성



스파크 시스템 구성 (2)

- 스파크 코어는 태스크 스케줄링, 메모리 관리, 고장 복구, 저장 시스템 접근 등을 수행하는 계산 엔진
- RDD 관리 API는 RDD를 정의하고 관리하는 API
- 고수준 라이브러리
  - 스파크 SQL은 구조화된 데이터 상의 작업을 수행
    - 기본적인 데이터 변환을 수행하는 데이터프레임(DataFrame)이라는 고수준 추상화
    - 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 기계학습 알고리즘을 구현한 라이브러리
  - GraphX는 그래프 관련 병렬 계산을 수행하는 라이브러리

3. 하둡 YARN에서 스파크 설치

스파크의 실행 동작 모드

- 스파크 실행 모드: 7장에서 자세히 소개
  - 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 설치

- 마스터 노드에서 설치한 후에 슬레이브 노드에 복사
  - 설치 디렉토리: ~/spark-2.1.0-bin-hadoop 2.7
- 스파크 2.1.0 다운로드
  - http://spark.apache.org/downloads.html

```
$ wget http://d3kbcqa49mib13.cloudfront.net/spark-2.1.0-bin-hadoop 2.7. tgz
```

- 압축 해제 (tar.gz, tgz)

```
$ tar -xvzf spark-2.1.0-bin-hadoop 2.7. tgz
```


스파크 설정 (1)

- 스파크의 YARN 클러스터 설정
  - 템플릿 설정 파일 복사

```
$ cd  ~/spark-2.1.0-bin-hadoop2.7
$ cp  conf/spark-env.sh.template  conf/spark-env.sh
$ cp  conf/spark-defaults.template  conf/spark-defaults.sh
$ cp conf/log4j.properties.template conf/log4j.properties
```

  - 설정파일 수정
    - conf/spark-defaults.conf에 스파크 라이브러리 위치 기술

```
spark.yarn.archive        hdfs://master:9000/user/hadoop/spark-jars.zip
```

    - conf/log4j.properties에 출력 옵션 수정: 정보 출력을 줄임

```
log4j.rootCategory=WARN, console
```

  - *conf/slaves에 데이터 노드의 호스트 이름 등록 (독립형 모드)*
    - *$ vi conf/slaves*

```
master
slave1
```

스파크 설정 (2)

- ~/.bashrc 에 하둡, 스파크 위치 환경변수 삽입

```
export HADOOP_HOME=/home/hadoop/hadoop-2.7.3
export HADOOP_CONF_DIR=/home/hadoop/hadoop-2.7.3/etc/hadoop
export YARN_HOME=/home/hadoop/hadoop-2.7.3
export YARN_CONF_DIR=/home/hadoop/hadoop-2.7.3/etc/hadoop
export SPARK_HOME=~/spark-2.1.0-bin-hadoop2.7
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native
```

- $SPARK_HOME/jars의 모든 jar 파일을 spark-jars.zip 압축하여 아카이브 형식으로 HDFS의 /user/hadoop/spark-jars.zip 으로 복사

```
$ hadoop fs -put spark-jars.zip /user/hadoop/spark-jars.zip
```


하둡 설정 수정

- 스파크 셸 실행 에러 시 다음 하둡 설정 수정 추가
  - 하둡 마스터, 슬레이브 모든 노드의 설정 수정 후에 재시작
  - https://issues.apache.org/jira/browse/YARN-4714
  - ~/hadoop-2.7.3/etc/hadoop/yarn-site.xml

```
<property>
   <name>yarn.nodemanager.pmem-check-enabled</name>
   <value>false</value>
</property>
<property>
   <name>yarn.nodemanager.vmem-check-enabled</name>
   <value>false</value>
</property>
```

스파크 배포 및 실행 – 독립형 모드

- 하둡 YARN이 아닌 독립형 모드로 실행하는 경우
- 마스터 노드의 스파크 설치 디렉토리를 배포

```
$ scp  -r  ~/spark-2.1.0-bin-hadoop2.7  hadoop@slave1:~
```

- 스파크 실행

  $ sbin/start-all.sh

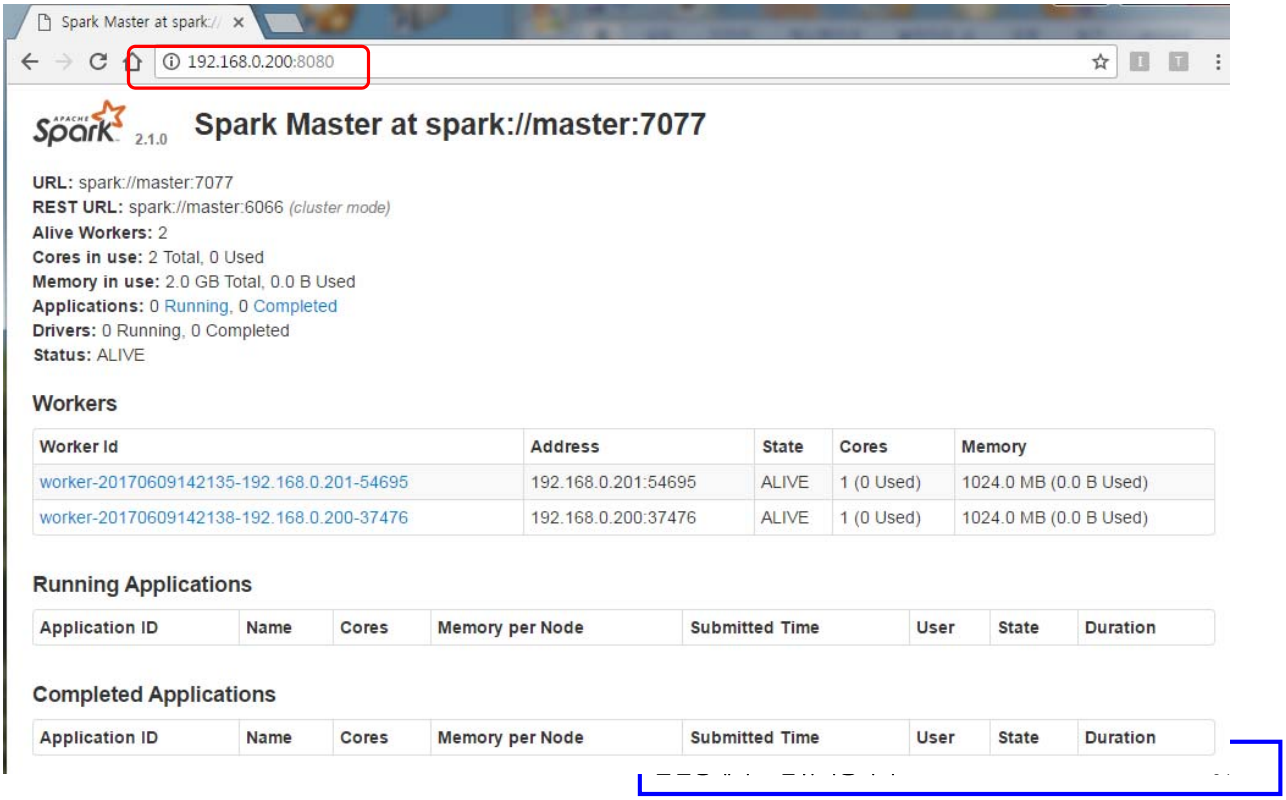
  - 종료시에는 $ sbin/stop-all.sh

스파크 동작 확인 (1)

- 스파크 마스터 실행 후 마스터 8088 포트 웹 접속
  - $SPARTK_HOME/sbin/start-master.sh

```
hadoop@master:~$ jps
8929 Jps
8290 ResourceManager
7878 NameNode
8873 Master
8013 DataNode
hadoop@master:~$
```

스파크 동작 확인 (2)



스파크 셸 (Spark Shell)

- 스파크 셸은 사용자와 상호작용하며 프로그램 작성 가능
  - 스칼라, 파이썬 셸
  - 셸이 시작하면 SparkContext 객체가 초기화되고, 변수 sc가 이를 가리킴
    - SparkContext는 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 셸 실행

```
hadoop@master:~$ $SPARK_HOME/bin/spark-shell --master yarn
17/07/03 14:00:54 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObject
Exception
Spark context Web UI available at http://192.168.0.200:4040
Spark context available as 'sc' (master = yarn, app id = application_1499057418344_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show)
:h? <string>             search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the interpreter
:replay [options]        reset the repl and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the repl to its initial state, forgetting all session entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:settings <options>      update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <expr>        display the kind of expression's type
:warnings                show the suppressed warnings from the most recent line which had any

scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@11c40a65

scala> :quit
hadoop@master:~$
```


과 제

- 스칼라 학교의 아래 주제에 대해 각자 준비하고 실행하여 발표
  - http://twiter.github.io/scala_school/ko/index.html
  - 개인별 발표 내용 및 순서는 과제 게시판 공지 참조

참고 자료

- MapR Academy DEV 360 Apache Spark Essentials
  - http://learn.mapr.com/dev-360-apache-spark-essentials
  - 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