Pig(피그)

Pig는 SQL과 유사한 언어인 Pig Latin을 이용하여 대규모 데이터 셋을 분석하기 위한 하둡 기반의 오픈 소스 플랫폼이다.

피그는 야후에서 처음 개발이 되었으며 지금은 아파치 오픈 소스(http://pig.apache.org/)이다.

Pig는 복잡한 데이터 병렬 연산을 위한 간단한 작업과 프로그래밍 인터페이스를 제공해준다.
야후에서 개발을 시작하여 아파치 톱 레벨 오픈 소스로 발전했다.
HDFS에 저장된 대용량 데이터 셋을 분석하고 조작 프로그램이다.
또한 부족한 기능이 있을 경우 해당 기능을 UDF(User Defined Function)라는 것을 이용하여 작성하여 보충이 가능하다.

컴포넌트	설명
Pig Latin	해당 프로그램 그 자체를 말한다. 내부 컴파일러에 의하여 MapReduce 잡들로 바뀐다.
실행 환경	로컬 모드와 맵리듀스 모드가 있다. 로컬 모드는 하나의 JVM에서 동작한다. 맵리듀스 모드는 하둡 클러스터 내에서 동작한다.

Pig Latin

Pig Latin은 데이터 셋의 연속적인 변환을 통하여 최종 데이터 셋을 생성하는 스크립트 언어이다.
최종적으로는 MapReduce 프로그램으로 컴파일이 된다.

Pig Latin의 특징

특징

1. 맵-리듀스으로 컴파일된다.
2. 데이터 흐름 기반의 프로그래밍 언어이다.
3. 분산 실행 환경
4. 사용자 정의 함수(User Defined Function)를 지원한다.
   - Pig 자체에 부족한 기능이 있는 경우 UDF를 이용하여 기능 확장을 할 수 있다.
   - 자바/파이썬/루비/자바스크립트 등의 언어로 작성이 가능하다.
5. 다양한 데이터 구조를 가지고 있다.
6. 오프라인 배치 작업에 용이하다.
7. 기타
   - Pig는 Native 언어에 비하여 속도는 느리다.
   - 하지만, 작은 양의 소스 코드가 사용되어, 높은 생산성을 자랑한다.
   - 비정형에도 잘 처리할 수 있다.

항목	설명
사용자 정의 함수 (User Defined functions)	사용자가 자신의 목적에 맞는 저장/로드/필터/조인 과정 등을 임의의 순서대로 변경이 가능하다.

	Pig 자체에 정의되어 있지 않는 함수를 구현하고자 하는 경우에 유용하다. 자바/파이썬/루비/자바스크립트 등의 언어로 작성이 가능하다.
쉬운 프로그래밍	복잡하게 구성된 데이터 분석 조작에 있어서도 데이터 흐름을 명시적으로 보여줄 수 있는 코드 작성과 관계형 데이터 처리 스타일(filter, group, join,union) 등을 지원하여 사용하기 쉽다.
비용 최적화	시스템이 코드 실행을 자동으로 최적화 해주므로 사용자는 효율성을 생각하지 않고 프로그래밍 내용에만 집중 한다.

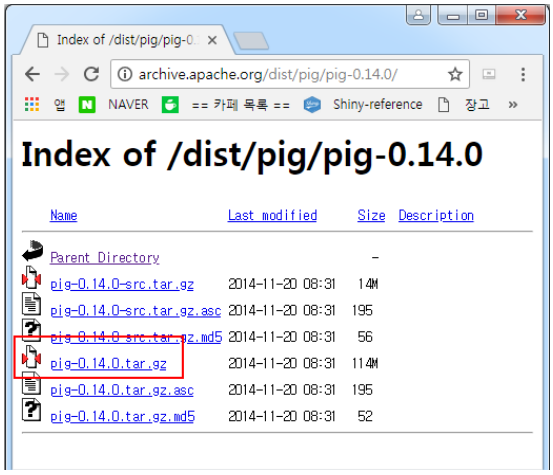
Pig 프로그램 설치

Pig와 관련된 프로그램을 설치해보자.

항목	설명
Pig	pig-0.14.0

Pig 다운로드하기

http://archive.apache.org/dist/pig/으로 접속하여 하위 pig-0.14.0/ 을 다운로드 한다.
pig-0.14.0.tar.gz 파일을 다운로드한다.



Pig 설치 및 소유권 이전

다음 명령어를 사용하여 설치하도록 한다.

설치하기 :

```
# 관리자로 로그인한다.
# 현재 경로 : /usr/local
cd /usr/local
tar -xvf /home/hadoop/Downloads/pig-0.14.0.tar.gz

chown -R hadoop:hadoop /usr/local/pig-0.14.0/
```

실습 : bash 파일 설정하기

pig 관련 명령어들을 bash 프로파일에 등록해두면 긴 문장을 타이핑 하지 않아도 된다.
다음 명령어를 이용하여 설정한다.

실습 하기 :

```
cd
pwd
# /home/hadoop

# 각 프로그램이 설치된 경로를 미리 확인해 둔다.
vi .bash_profile
PATH=$PATH:$HOME/bin
export PATH=$PATH:$HOME/bin
export PIG_HOME=/usr/local/pig-0.14.0
export
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$PIG_HOME/bin:$HIVE_HOME/bin:/usr/local/eclipse/

... 중략
!wq

source .bash_profile # 배시 프로파일을 다시 실행한다.
```

실습 : 설치된 버전 확인

해당 프로그램들이 잘 설치가 잘 되었는지 다음과 같이 확인하도록 한다.

실습 :

$ **pig -help**를 치면 여러 가지 명령어들이 나온다.

Pig 스크립트 실행 방법

Pig 스크립트는 다음과 같이 3가지로 실행할 수 있다.

1. hadoop jar 커맨드를 통하여 실행하는 방법
   - 맵리듀스 모드에서만 의미가 있다.

2. 자바 프로그램 내에 임베드해서 실행하는 방법
   - 맵리듀스 모드에서만 의미가 있다.

3. Grunt라는 이름의 Pig shell을 통하여 실행하는 방법
   - 로컬 모드와 맵리듀스 모드 둘 다 사용 가능하다.
   - Pig 셸을 실행할때 – x 옵션을 사용하여 지정할 수 있다.
   - -x local은 로컬 모드, -x mapreduce은 맵 리듀스 모드이다.
   - 프롬프트를 빠져 나가려면 quit; 을 입력하면 된다.

실행 모드	설명
배치 모드	셸 실행시 인자로 실행하려는 스크립트의 위치를 입력한다.
대화 모드	Pig 명령문을 하나씩 실행하고 결과를 확인할 수 있다.

Pig의 데이터 타입

Pig의 타입은 단순 타입과 복합 타입 두 가지가 존재한다.

단순 타입

Pig에서 처리할 수 있는 단순 타입은 다음과 같은 항목들이 있다.

타입	설명
int	32 비트의 정수형을 의미한다.
long	64 비트의 정수형을 의미한다. int 와 구분하기 위하여 숫자 뒤에 l 또는 L 을 붙인다.
float	32 비트 부동 소수점을 의미한다. double 와 구분하기 위하여 숫자 뒤에 f 또는 F 를 붙인다. 실수형 타입은 기수 표기를 위하여 E 나 e 를 사용할 수 있다. 예를 들어서, 23400 은 23.4E3F 가 사용 가능하다.
double	64 비트 부동 소수점을 의미한다.
bytearray	바이트 스트림
chararray	자바의 String 형으로 표현이 되는 데이터를 처리한다. 문자 코드는 UTF-8 만 가능하다.

복합 타입

Pig에서 처리할 수 있는 복합 타입은 다음과 같은 항목들이 있다.

타입	설명
튜플(tuple)	필드들의 집합으로 구성이 되며, ()안에 표시한다. 필드를 해당 이름이나, 인덱스로 접근할 수 있다. 예시 ("hong", 43)은 두 개의 필드로 구성된 튜플이다. 필드는 다른 튜플이나 bag 이 될 수 있다.
백(bag)	순서 없는 집합으로 관계형 DB 의 Table 과 유사하다. {}안에 표시한다. 예시 {("hong", 43), ("park", 32), ("kim", 10)}은 두 개의 필드를 갖는 세 개의 튜플로 구성된 bag 이다.

	bag 은 리스트 등의 컬렉션 데이터를 처리하는 데 사용된다.
맵(map)	키(key)는 무조건 chararray 형 이어야 한다. 밸류(value)는 어느 Pig 데이터의 타입도 될 수 있다. Pig 는 기본적으로 밸류의 타입을 bytearray 으로 간주한다. []안에 표시하며, 키와 밸류 사이에 #을 사용한다. 예시 ['name'#'soo', 'age'#30]이라고 하면, 두 개의 키가 name 과 age 를 갖는 맵이다.

Pig의 데이터 입출력 방법

Pig에서 데이터를 읽어 들이기 위하여 LOAD 구문을 사용한다.

LOAD 구문

항목	설명
사용 형식	릴레이션 = LOAD '경로' [using LOAD 함수] [AS 스키마]
경로	입력 데이터가 있는 경로를 지정한다. * 등의 glob로 정의할 수 있다.
LOAD 함수	Pig에서 데이터를 처리하기 위한 방법을 정의해 놓은 항목들이다. LOAD 함수의 종류를 참고하길 바란다.
AS 스키마	사용할 데이터의 접근 이름이과 데이터의 유형을 지정할 수 있다.

LOAD 함수의 종류

항목	설명
PigStorage('delimiter')	파일 내에 행 단위로 기록되어 있는 UTF-8 방식의 데이터를 읽을 때 사용한다. 'delimiter'는 구분자를 의미하는 데, 기본 값은 TAB이다.
BinStorage()	바이너리 데이터를 읽을 때 사용하는 데, 매개 변수는 없다.
TextLoader()	파일 내에 행 단위로 기록되어 있는 UTF-8 방식의 데이터를 읽을 때 사용한다. PigStorage와는 달리 구조화되지 않은 데이터를 처리하고, 매개 변수는 없다.
JsonLoader('schema')	JSON 형식의 데이터를 읽을 때 사용한다. 'schema'를 이용하여 JSON의 내부 구조에 맞추어 데이터를 읽을 수 있다.

Pig의 내장 함수

Pig에서는 릴레이션 데이터를 처리하기 위하여 여러 가지 함수를 제공한다.
함수는 Pig 내장형 함수와 PiggyBank가 제공하는 것, 사용자 정의 함수(UDF) 등이 있다.
함수는 FOREACH-GENERATE 구문이나 FILTER 구문 등으로 데이터 처리, 평가 등에 사용된다.

Pig 내장 함수

Pig 내장 함수로 평가 함수, 수학 함수, 문자열 처리 함수, 날짜/시간 처리 함수, bag이나 tuple 처리 함수, 데이터 입출력 함수 등이 있다.

함수	설명
평가 함수	AVG(평균), COUNT(합계), IsEmpty(요소가 비어 있는가), MAX(최대), MIN(최소), TOKENIZE(데이터 분할), SUM(합계) 등등
수학 함수	SIN/COS/TAN 등(삼각 함수), EXP(지수 함수), LOG/LOG10(로그 함수), FLOOR(버림), CEIL(올림), ROUND(반올림), ABS(절대값), RANDOM(난수), SQRT(루트), CBRT(입방값) 등등
문자열 처리 함수	INDEXOF(문자열 위치 검색), REGEX(정규 표현식), REPLACE(치환), SUBSTRING(문자열 추출), TRIM(전후 공백 제거), STRSPLIT(문자열 분할) 등등
날짜/시간 처리 함수	GETDay(일/시 등 특정 요소 추출), HoursBetween(기간 연산), ToDate(날짜 변환), CurrentTime(현재 시각 추출) 등등
bag/tuple 처리 함수	TOBAG(bag 로 변환), TOP(bag 내의 topN 구문), TOTUPLE(tuple 으로 변환하기)
데이터 입출력 함수	PinStorage(UTF-8 형으로 데이터 입출력) BinStorage(바이너리 형으로 데이터 입출력)

PiggyBank가 제공하는 함수

Pig는 내장 함수 이외에 UDF를 모아서 PiggyBank라는 라이브러리로 제공하고 있다.

함수	설명
평가 함수	IsNumeric 함수등 Tuple 요소가 수치형인지 확인하는 함수 ISOToUnix 함수와 같이 날짜 시간 정보를 다루는 함수 DoubleMax 함수와 같이 Pig 표준에는 없는 수학 함수 등이 제공된다.
데이터 로드 함수	CSV 파일을 읽어 들이는 CSVLoader 함수 Xml 을 읽어 들이는 XMLLoader 함수 Hadoop 의 잡 실행 이력을 읽는 HadoopJobHistoryLoader 함수 등이 제공되고 있다. 참조 주소 : http://pig.apache.org/docs/latest/api/

사용자 정의 함수

Pig에서 표준 내장 되지 않는 함수는 사용자가 독자적으로 개발할 수 있다.
작성 방법은 다음과 같다.

함수	설명
평가 함수 작성	EvalFunc 를 상속하여 exec 메소드를 구현하면 된다.
필터 함수 작성	FilterFunc 를 상속하여 exec 메소드를 구현하면 된다.

Pig의 연산자

연산자

Pig에서 사용 가능한 연산자는 다음과 같다.

연산자	설명
산술 연산	+, -, *, /을 사용할 수 있다.
비교 연산	반환 값으로 논리형을 반환한다. ==, !=, <, >, <=, >=, matches(자바의 정규 표현식을 사용한 문자열 일치)
논리 연산자	비교 연산자를 다수 조합하여 평가하기 위해 사용한다. AND, OR, NOT
NULL 체크	필터 규칙이나 필드 값이 NULL 인지 확인할 때 사용한다. is null, is not null 이 있다.

Pig의 명령어

Pig의 명령어는 다음과 같은 것들이 있다.

명령어	설명
LOAD	파일 시스템에서 데이터 셋을 로딩한다. 디폴트로 필드들이 TAB 문자로 나누어진 텍스트 파일을 로딩해준다. AS 구문 다음은 파일 내 필드들의 이름과 타입을 지정해준다. AS 를 사용하지 않은 경우 #0, #1 과 같이 포지션 번호로 접근할 수 있다.
GROUP	relation 을 By 다음의 필드를 이용하여 그룹핑하는 구문이다.
FOREACH	relation 의 그룹 리스트를 하나씩 스캔하면서 Generate 다음에 명시된 필드들로 새로운 relation 을 만들어 낸다.
FILTER	주어진 relation 에서 BY 다음에 주어진 조건을 만족하는 튜플들만 필터링한다.
ORDER	주어진 relation 에서 BY 다음에 주어진 필드(하나 이상)으로 정렬한다.
LIMIT	주어진 relation 에서 주어진 수만큼 튜플을 추출하여 새로운 relation 을 만들어 낸다.

	즉, 레코드의 수한 제한하여 주어진 수 만큼 추출한다.
STORE	주어진 relation 에 대한 처리 결과를 파일 시스템에 저장하는 역할을 한다.
JOIN	연관된 키에 따라 둘 이상의 데이터 셋을 조인한다. 내부 조인과 외부 조인을 모두 지원한다.
DISTINCT	동일한 값의 행에 대한 정보는 1 개 값으로 중복되는 레코드를 제거해준다.
SPLIT	Filter 의 조건에 따라 2 개 이상의 데이터로 분할하여 기록한다.
UNION	둘 이상의 데이터 셋을 하나의 데이터 셋으로 병합하여 기록한다.

실습 : Pig 실습해보기

간단한 데이터 파일을 이용하여 pig를 테스트 해보기로 한다.

소스 코드	pig_test_01.txt(코드 파일), pig_sample_01.txt(데이터 파일)	카페

pig_sample_01.txt

```
100,1
100,2
100,3
100,4
100,5
200,6
200,7
200,8
```

pig_test_01.txt

```
cd
mkdir pigtest
cd pigtest/

pig -x local

# PigStorage : 디폴트 값으로 필드들이 tab으로 나누어진 행을 로딩하는 데 사용한다.
# 여기서는 콤마를 이용하여 로딩하고 있다.
mydata = load 'pig_sample_01.txt' using PigStorage(',') as (mynum:int, myvalue:int) ;

# by 키워드 다음의 항목인 mynum 컬럼을 이용하여 그룹핑한다.
mygrp = group mydata by mynum;

describe mygrp ;
# mygrp: {group: int,mydata: {(mynum: int,myvalue: int)}}

feach = foreach mygrp generate group, COUNT($1) as cnt ;

mylimit = limit feach 10 ;

dump mylimit ;
# ... 코드 생략
# (200,3)
# (100,5)
```

실습 : Pig 실습해보기

첨부 파일을 이용하여 pig를 테스트 해보기로 한다.
이 문제는 컬럼 2개를 이용한 그룹핑 문제이다.

소스 코드	pig_test_02.txt(코드 파일), pig_sample_02.txt(데이터 파일)	카페

출력 결과는 다음과 같다.

```
# (200,30,4)
# (100,10,2)
# (100,20,3)
```

실습 : Pig 실습해보기

실습에 사용할 데이터는 가장 많이 인용된 문헌 찾기에 사용된 파일 2M.SRCID.DSTID이다.

소스 코드	2M.SRCID.DSTID 실습하기.txt	카페

2M.SRCID.DSTID 실습하기.txt

```
cd
mkdir mypig
cd mypig/
cp /mnt/hgfs/shared/2M.SRCID.DSTID ./
pig -x local

# 로드된 데이터 셋의 이름을 records라고 명시했는 데 pig에서는 이것을 relation 또는 alias라고 한다.
# 해당 파일을 Tab으로 구분하고, 컬럼 이름은 srcid와 dstid으로 읽어 들이세요.
records = load '2M.SRCID.DSTID' as (srcid:long, dstid:long) ;

# describe 명령어를 사용하면 주어진 relation의 스키마를 볼 수 있다.
describe records
# records: {srcid: long,dstid: long}

# records : 그룹핑을 수행할 relation
# dstid 컬럼(by 뒤에 있는 컬럼)을 기준으로 그룹핑해주세요.
grouped_records = group records by dstid ;
describe grouped_records
# grouped_records: {group: long,records: {(srcid: long,dstid: long)}}

# GENERATE 다음의 group은 앞서 사용된 group의 키 필드를 말한다.
# COUNT는 같은 dstid를 갖는 튜플들의 수를 카운트한다.
# EVAL 함수의 종류 : COUNT, AVG, MAX 등등
```

```
# 대소문자 주의
count_records = FOREACH grouped_records GENERATE group, COUNT(records) as sum ;
describe count_records
# count_records: {group: long,sum: long}

filtered_records = FILTER count_records BY sum > (long)1 ;

ordered_filtered_records = ORDER filtered_records BY sum DESC ;

temp = LIMIT ordered_filtered_records 10 ;

# 다음 문장은 시간이 약간 걸린다.
DUMP temp ;

# 출력 결과는 다음과 같다.
# 해당 튜플의 1번째 항목은 인용이 많이 된 문서의 id이고,
# 2번째 항목은 그 문서들 각각의 인용 횟수를 의미한다.
(9316,34884)
(14533,24610)
(14532,23075)
(15573,21453)
(5407,15945)
(3383,13632)
(9239,12545)
(5405,12187)
(38523,10900)
(18951490,9536)

# STORE는 주어진 relation을 저장하는 역할이다.
# 하위 폴더 pig_citation가 자동으로 생성이 된다.
STORE ordered_filtered_records INTO 'pig_citation' ;

# pig 프롬프트를 빠져 나가려면 다음 명령어를 사용하면 된다.
quit

cd pig_citation/
ls
# 해당 디렉토리에 part-r-00000이라는 결과 파일이 생성된다.
part-r-00000  _SUCCESS

head part-r-00000
```