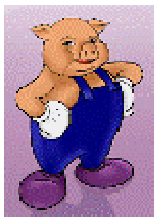
Ⅳ. 분석 프레임워크와 데이터 수집 자동화

- Author: JinKyoung Heo
- Issue Date: 2014.08.15
- Revision #: Rev 32
- Homepage: www.javaspecialist.co.kr
- Email: hjk7902@gmail.com

이 페이지는 여백 페이지입니다.

1 Pig



Objectives

- 피그는 대용량 데이터 셋을 좀 더 고차원적으로 처리할 수 있도록 해 줍니다.

이 페이지는 여백 페이지입니다.

Pig

- http://pig.apache.org/
  - Top Level Apache Project
- 피그 라틴
  - 대용량 데이터 셋을 다루기 위한 스크립트 언어
  - 데이터의 흐름을 표현하기 위해 사용하는 언어
- 실행환경
  - 피그 라틴 프로그램을 수행하는 실행 환경
  - 단일 JVM에서의 로컬 실행환경과 하둡 클러스터상의 분산 실행환경을 제공한다.
- 하나의 피그 라틴 프로그램은 입력 데이터에 적용되어 출력을 생성하는 일련의 연산 또는 변환으로 구성되어 있다.

정의
Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs.

Pig is an abstraction on top of Hadoop
데이터 처리를 위한 프로그래밍 언어를 제공합니다.
맵리듀스로 변환되고 하둡 클러스터에서 실행됩니다.

피그가 사용되는곳...
Yahoo, Twitter, Netflix, etc...

피그 왜 배우죠?
맵리듀스 프로그래밍이 어려우니까...

피그 어디에 쓰죠?
그럼 맵리듀스 프로그래밍은 어디에 사용하죠?
피그는 분석, 데이터 과학, 통계 등등에 사용합니다.

피그의 특징이라면?
Join Datasets, Sort Datasets, Filter, Data Types, Group By, User Defied Functions, Etc...

Pig and MapReduce

- MapReduce는 프로그래밍이 필요함
  - map()과 reduce()함수에 대한 개념에 대해 알아야 한다.
  - 자바 프로그래밍 언어에 대한 이해가 선수되어야 한다.
- Pig는 고수준 언어를 제공한다.
  - 분석(Analysts)
  - 데이터 과학(Data Scientists)
  - 통계(Statisticians)
  - 등등...
- Yahoo에서 데이터 분석을 위해 만들었음

Pig 특징

- Join Datasets
- Sort Datasets
- Filter
- Data Types
- Group By
- User Defined Functions
- Etc...

Pig 컴포넌트

- Pig Latin
  - Command 기반 언어
  - 데이터 처리와 흐름을 표현하기 위해 특별해 고안됨
- 실행환경
  - Pig Latin 명령이 실행되는 환경
  - Local 모드와 Hadoop 모드
- Pig 컴파일러
  - Pig Latin을 MapReduce로 변경
  - 컴파일러는 최적화된 실행계획을 수립함

설치

◈ 다운로드
- http://pig.apache.org/releases.html#Download
  - Download a release now! Pig 0.8 and later클릭
  - 미러 사이트 클릭 -> pig-0.13.0/ -> pig-0.13.0.tar.gz 다운로드

◈ 설치

[hadoop@master Desktop]$ su -
Password:
[root@master ~]# cd /usr/local/
[root@master local]# tar -xvf /home/hadoop/Downloads/pig-0.13.0.tar.gz
[root@master local]# chown -R hadoop:hadoop pig-0.13.0/

◈ 환경변수 등록
- export PIG_INSTALL=/usr/local/pig-0.13.0
- export PIG_CLASSPATH=$HADOOP_INSTALL/etc/hadoop
- export PATH=$PATH:$PIG_INSTALL/bin

다운로드 된 파일은 /home/hadoop/Download 디렉토리에 pig-0.12.0.tar.gz 파일 이름으로 되어 있다고 가정합니다.

하둡 계정 홈 디렉토리로 이동 해서 .bash_profile 파일을 수정하여 PIG_INSTALL 환경변수를 등록합니다. 그리고 PATH 환경변수에 PIG_INSTALL/bin 디렉토리를 추가합니다.

```
[hadoop@master ~]$ vi .bash_profile
export PATH=$PATH:$HOME/bin
export JAVA_HOME=/usr/local/jdk1.7.0_51
export HADOOP_INSTALL=/usr/local/hadoop-2.2.0
export PIG_INSTALL=/usr/local/pig-0.13.0
export PIG_CLASSPATH=$HADOOP_INSTALL/etc/hadoop
export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop
export ANT_HOME=/usr/local/apache-ant-1.9.3
#export CLASSPATH=$CLASSPATH:$PIG_INSTALL/build/pig-0.12.1-SNAPSHOT.jar
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_INSTALL/bin:$PIG_INSTALL/bin:$ANT_HOME/bin
[hadoop@master ~]$ source .bash_profile
[hadoop@master ~]$ pig -version
Apache Pig version 0.13.0 (r1606446)
compiled Jun 29 2014, 02:27:58
[hadoop@master ~]$ pig -help
```

* HADOOP_HOME에 설치된 하둡에 namenode와 jobtracker 정보가 설정되어 있지 않을 경우 fs.defaultFS 속성과 mapred.job.tracker 속성이 정의되어 있는 설정파일 디렉토리를 HADOOP_CONF_DIR로 지정할 수 있습니다. 또는 피그의 conf 디렉토리에 pig.properties 파일을 생성하고 다음과 같이 속성을 설정할 수도 있습니다.

```
fs.defaultFS=hdfs://master:9000/
mapred.job.tracker=master:8021
```

실행

- $ pig -x local
  - 로컬 모드, 로컬 파일시스템에서 실행, 개발/테스트/프로토타이핑 등에 사용
- $ pig -x mapreduce
  - Hadoop 모드(맵리듀스 모드), 디폴트
  - Pig Latin -> MapReduce 잡 -> 하둡 클러스터에서 실행
  - 매개변수 치환은 $, 실행은 pig -param data=test.txt myscript.pig
- -x 옵션 또는 -exectype을 사용할 수 있다.
- 종료는 quit
- pig 명령어는 탭을 누르면 자동 완성된다.
- 실행하면 다음과 같은 프롬프트를 볼 수 있다.

```
ation - mapred.used.genericoptionsparser is deprecated. Inst
lient.genericoptionsparser.used
2014-03-26 02:54:16,701 [main] INFO  org.apache.hadoop.conf.
ation - fs.default.name is deprecated. Instead, use fs.defau
grunt>
```

```
[hadoop@master ~]$ pig
2014-03-26 02:54:15,360 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
12.0 (r1529718) compiled Oct 07 2013, 12:20:14
2014-03-26 02:54:15,361 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/hadoop/pig_1395827655358.log
2014-03-26 02:54:15,385 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/hadoop/.pigbootup not found
2014-03-26 02:54:15,762 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2014-03-26 02:54:15,762 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2014-03-26 02:54:15,762 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://master:9000
2014-03-26 02:54:15,
ation - mapred.used
lient.genericoption
2014-03-26 02:54:16
ation - fs.default.
grunt>
```

```
[hadoop@master ~]$ pig -exectype local
2014-03-26 02:55:30,309 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
12.0 (r1529718) compiled Oct 07 2013, 12:20:14
2014-03-26 02:55:30,310 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/hadoop/pig_1395827730308.log
2014-03-26 02:55:30,333 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/hadoop/.pigbootup not found
2014-03-26 02:55:30,513 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2014-03-26 02:55:30,513 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2014-03-26 02:55:30,515 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: file:///
2014-03-26 02:55:30,518 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.used.genericoptionsparser is deprecated. Instead, use mapreduce.c
lient.genericoptionsparser.used
2014-03-26 02:55:30,962 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2014-03-26 02:55:30,965 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

피그 실행

◈ 스크립트 이용
- 터미널 창에서 파일을 직접 실행시킬 수 있다.
- $pig scriptFile.pig

◈ Grunt
- 그런트에는 GNU의 Readline과 같은 행 편집 기능이 있다.
- Ctrl-E는 라인의 끝으로 커서를 이동시킨다.
- Ctrl-P, Ctrl-N(또는 위/아래 키)는 이력버퍼에서 명령행을 볼 수 있다.
- Tab 키는 피그라틴 키워드의 자동 완성 기능
  - autocomplete라는 이름의 파일을 conf 디렉토리에 생성하여 자동완성 방식을 설정 할 수 있다.

◈ Embedded
- JDBC를 사용하여 자바에서 SQL을 실행하듯이 PigServer 클래스를 사용하여 자바로 피그프로그램을 실행할 수도 있다.
- PigRunner를 통해 프로그램적으로 그런트에 접근할 수 있다.

```
grunt> cat /output/delay.csv
year,month,delaycount,delaytime
2008,1,9773,429828
2008,10,2476,105092
2008,11,3662,139442
2008,12,16683,747959
2008,2,12188,544326
2008,3,10198,474687
2008,4,5491,268806
2008,5,5715,240013
2008,6,11995,571832
2008,7,10061,555794
2008,8,8630,398906
2008,9,3113,157032
```

* 예제를 테스트하기 위해서는 미국 상업용 항공기의 운항 데이터들 중에서 2007, 2008년도 데이터를 맵리듀스 어플리케이션으로 분석하여 연도,월,지연횟수,지연시간 정보를 메타정보 없는 출력파일로 생성해 놓아야 합니다.

피그 라틴 Concepts

◈ 빌딩 블록
- Fields
  - 데이터 조각
- Tuple
  - 필드들의 집합, "("와 ")"로 표현한다.
  - (10.4, 5, word, 4, field1)
- Bag
  - 튜플들의 모음, "{"와 "}"로 표현한다.
  - { (10.4, 5, word, 4, field1), (this, 1, blah) }

◈ 관계형 데이터베이스와 비교
- Bag은 데이터베이스의 테이블과 유사하다.
- Tuple은 테이블의 행(row)과 유사하다.
- Bag에 포함된 튜플들은 필드의 수가 같이 않아도 된다.
  - 관계형 데이터베이스와 다른 점

- 하둡 2.2.0이 설치되어 있다면 pig의 버전이 문제가 될 수 있습니다. 이 문서를 작성할 때에 pig의 가장 상위 버전은 0.12.0이었습니다. 이 버전은 하둡 2.2.0을 고려해서 만들어진 것이 아니므로 피그가 정상 실행되지 않습니다. Ant 도구를 이용하여 피그를 다시 컴파일 해야 합니다. 제공된 가상환경에는 재 컴파일 된 피그 파일을 포함하고 있습니다.

- 문제를 해결하기 위한 자세한 내용은 http://cafe.naver.com/javaspecialistgroup/311 문서와 그 답변 글을 확인하세요.
  문제 해결 과정은 다음과 같습니다.
  1. ant 설치
  2. ant 환경변수 설정
  3. 재 컴파일
  4. 생성된 jar 파일을 클래스패스 환경변수에 추가

- Pig 0.12.1 버전이 하둡 2.2.0에서 IncompatibleClassChangeError 가 발생된다면 다음 명령으로 빌드해야 합니다.
- `ant clean jar-all -Dhadoopversion=23`

피그 명령어

- ◈ 쉘 명령어
  - ➢ fs : 하둡 파일시스템 명령어를 사용한다.
- ◈ 파일 명령어
  - ➢ cat
  - ➢ cd
  - ➢ copyFromLocal, copyToLocal
  - ➢ cp
  - ➢ ls
  - ➢ mkdir
  - ➢ mv
  - ➢ pwd
  - ➢ rm, rmf
- ◈ 유틸리티 명령어
  - ➢ kill, exec, help, run, quit, set

아래에 피그 명령어를 간단하게 설명했습니다.

쉘 명령어

fs : 하둡 파일시스템 명령어를 사용한다.

```
fs -mkdir /tmp
fs -copyFromLocal file-x file-y
fs -ls file-
```

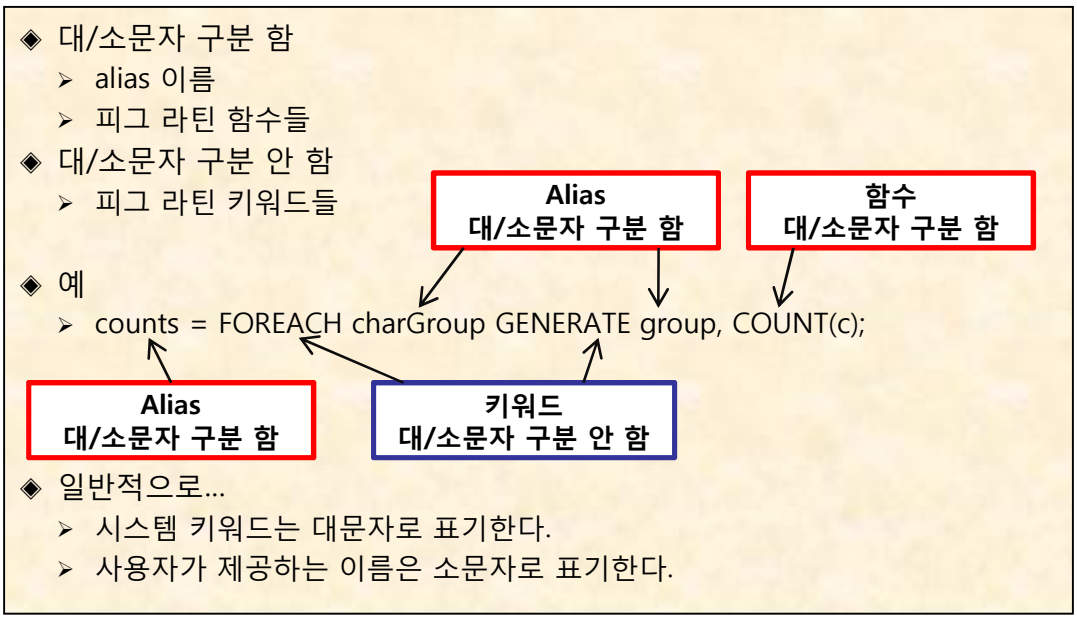
파일 명령어

cat : 한 개 이상의 파일 내용을 출력한다.
cd : 현재의 경로를 변경한다.
copyFromLocal : 로컬 파일을 하둡으로 복사한다.
copyToLocal : 하둡 상의 파일을 로컬로 복사한다.
cp : 파일 또는 디렉토리를 다른 경로에 복사한다.
ls : 파일을 조회한다.
mkdir : 새로운 디렉토리를 생성한다.
mv : 파일 또는 디렉토리를 다른 경로로 이동시킨다.
pwd : 현재의 실행 디렉토리의 경로를 출력한다.
rm : 파일 또는 디렉토리를 삭제한다.(디렉토리가 비어있지 않아도 삭제한다.)

유틸리티 명령어

kill : 맵리듀스 작업을 제거한다.
exec : 백그라운드로 스크립트를 실행
help : 사용 가능한 명령어와 옵션을 보여준다.
run : 그런트 쉘에 있는 스크립트를 실행한다.
quit : 명령처리기를 종료한다.
set : 피그의 옵션을 설정한다.
참고 URL : http://pig.apache.org/docs/r0.7.0/piglatin_ref2.html#Shell+Commands

Coding Conventions

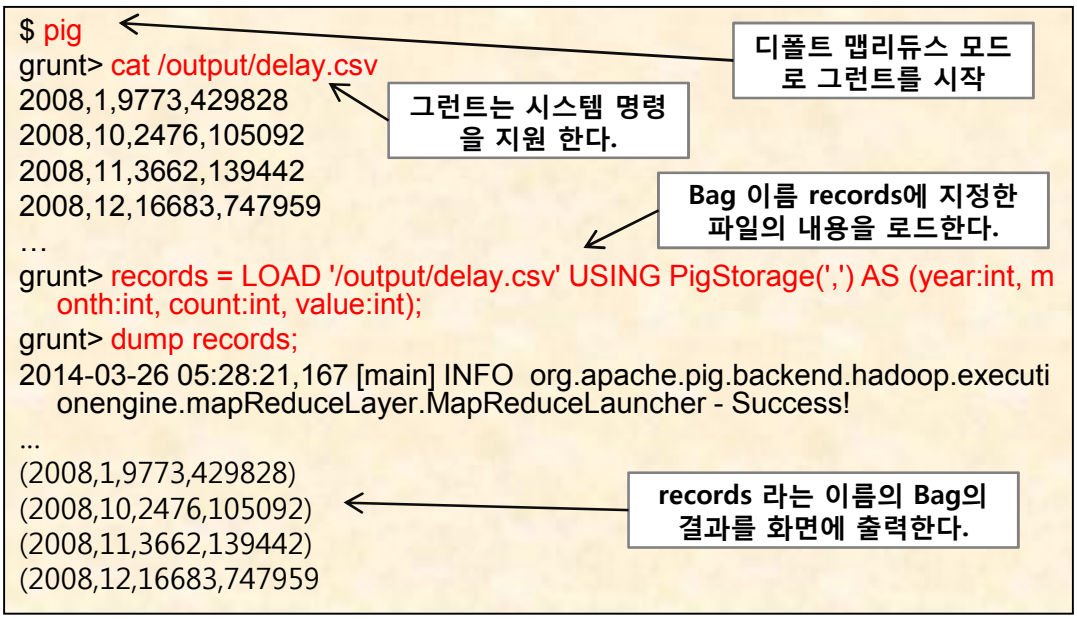


Pig 컴파일러는 명령문에서 키워드는 대/소문자를 구분하지 않지만 함수이름, alias, 사용자가 선언한 bag, field 이름 등 alias는 대/소문자를 구분합니다.

명령문을 작성할 때 시스템에서 제공하는 키워드는 대문자로 표기하며, 사용자가 제공하는 이름은 소문자로 표기합니다. 사용자가 제공하는 이름이 두 단어로 이루어 질 때에는 두 번째 단어의 첫 문자를 대문자로 표기합니다.

Pig 스크립트에서 사용하는 주석은
/* 구간 주석 */ 과
-- 한줄 주석입니다.

Pig Latin 예제



…

```
2014-03-26 05:28:21,167 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2014-03-26 05:28:21,171 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2014-03-26 05:28:21,186 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2014-03-26 05:28:21,187 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(2008,1,9773,429828)
(2008,10,2476,105092)
(2008,11,3662,139442)
(2008,12,16683,747959)
(2008,2,12188,544326)
(2008,3,10198,474687)
(2008,4,5491,268806)
(2008,5,5715,240013)
(2008,6,11995,571832)
(2008,7,10061,555794)
(2008,8,8630,398906)
(2008,9,3113,157032)
```

Schema Data Types

Type	Description	Example
	Simple	
int	32비트 부호화 정수	10
long	64비트 부호화 정수	10L 또는 10l
float	32비트 부동소수점 수	10.5F 또는 10.5f
double	64비트 부동소수점 수	10.5 또는 0.5e2 또는 0.5E2
	Arrays	
chararray	UTF-16 형식 문자 배열	hello world
bytearray	바이트 배열(blob)	
	Complex Data Types	
tuple	field의 집합	(19, 2)
bag	tuple 들의 모음	{(19,2), (18,1)}
map	tuple들의 모음	[open#apache]

```
A = LOAD 'data' USING MyStorage() AS (T: tuple(name:chararray, age: int));
B = FILTER A BY T == ('john', 25);
D = FOREACH B GENERATE T.name, [25#5.6], {(1, 5, 18)};
```

필드의 데이터타입/이름/값

	First Field	Second Field	Third Field
데이터 타입	chararray	int	float
위치 표기법(시스템에 의해 생성됨)	$0	$1	$2
가능한 이름 예	name	age	gpa
필드의 값 예	John	18	4.0

. 연산자를 이용하면 bag 내의 특정 필드를 참조할 수 있습니다.

null과 연산자

Operator	Interaction
비교 연산자: ==, != >, <, >=, <=	피 연산자가 null 이면 결과도 null
비교 연산자: matches	매치되는 문자열 중 하나가 null 이면 결과도 null
산술 연산자: + , -, *, / % modulo ? bincond	피 연산자가 null 이면 결과도 null
Null 연산자: is null	테스트 값이 null이면 true리턴, 그렇지 않으면 false 리턴.
Null 연산자: is not null	테스트 값이 not null이면 true리턴, 그렇지 않으면 false 리턴.
역참조 연산자: tuple (.) or map (#)	역 참조되는 튜플이나 맵이 null이면 null을 리턴함
Cast 연산자	Casting a null from one type to another type results in a null.
함수: AVG, MIN, MAX, SUM	함수들은 null을 무시함
함수: COUNT	null을 포함하는 모든 값의 수를 샘
함수: DIFF	튜플에 있는 두 필드를 비교한다. 만약 비교하는 두 필드들이 bag 이라면 한쪽 bag에는 있지만,다른 bag에는 없는 튜플을 반환한다.
함수: CONCAT	두 문자열(chararray)또는 두 bytearray를 합친다. 서브 표현식이 null이면 결과도 null.
함수: SIZE	테스트 객체가 null이면 null.

Diagnostic Tools

◈ DESCRIBE
- grunt> DESCRIBE *<bag_name>*
- Bag 구조를 나타냄

◈ EXPLAIN
- grunt> EXPLAIN *<bag_name>*
- 여러가지 보고서를 출력한다.
  - Logical Plan
  - MapReduce Plan

◈ INNUSTRATE
- grunt> ILLUSTRATE *<bag_name>*
- 피그 엔진이 데이터를 어떻게 변환하는지 보여준다.

```
grunt> cat /output/delay.csv
2007,1,12684,505125
2007,10,6655,287745
…
2008,9,3113,157032
grunt> records = LOAD '/output/delay.csv'
>>               USING PigStorage(',')
>>               AS (year:int, month:int, count:int, value:int);
2014-04-06 05:29:28,845 [main] INFO
    org.apache.hadoop.conf.Configuration.deprecation -
    mapred.jobtracker.maxtasks.per.job is deprecated. Instead, use
    mapreduce.jobtracker.maxtasks.perjob
…
2014-04-06 05:29:28,862 [main] INFO
    org.apache.hadoop.conf.Configuration.deprecation -
    mapred.jobtracker.instrumentation is deprecated. Instead, use
    mapreduce.jobtracker.instrumentation
grunt> DESCRIBE records
records: {year: int,month: int,count: int,value: int}
grunt> EXPLAIN records
2014-04-06 05:31:09,100 [main] INFO
    org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer -
    {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite,
    GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer,
    LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer,
    PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter,
    StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier]}
```

Diagnostic Tools

```
#-----------------------------------------------
# New Logical Plan:
#-----------------------------------------------
records: (Name: LOStore Schema: year#9:int,month#10:int,count#11:int,value#12:int)
|
|---records: (Name: LOForEach Schema:
    year#9:int,month#10:int,count#11:int,value#12:int)
    |   |
    |   (Name: LOGenerate[false,false,false,false] Schema:
    year#9:int,month#10:int,count#11:int,value#12:int) ColumnPrune:InputUids=[9, 10, 11,
    12]ColumnPrune:OutputUids=[9, 10, 11, 12]
    |   |   |
    |   |   (Name: Cast Type: int Uid: 9)
    |   |   |
    |   |   |---year:(Name: Project Type: bytearray Uid: 9 Input: 0 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: int Uid: 10)
    |   |   |
    |   |   |---month:(Name: Project Type: bytearray Uid: 10 Input: 1 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: int Uid: 11)
    |   |   |
    |   |   |---count:(Name: Project Type: bytearray Uid: 11 Input: 2 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: int Uid: 12)
    |   |   |
    |   |   |---value:(Name: Project Type: bytearray Uid: 12 Input: 3 Column: (*))
    |   |
    |   |---(Name: LOInnerLoad[0] Schema: year#9:bytearray)
    |   |
    |   |---(Name: LOInnerLoad[1] Schema: month#10:bytearray)
    |   |
    |   |---(Name: LOInnerLoad[2] Schema: count#11:bytearray)
    |   |
    |   |---(Name: LOInnerLoad[3] Schema: value#12:bytearray)
    |
    |---records: (Name: LOLoad Schema:
    year#9:bytearray,month#10:bytearray,count#11:bytearray,value#12:bytearray) Required
    Fields:null
```

Diagnostic Tools

```
#-----------------------------------------------
# Physical Plan:
#-----------------------------------------------
records: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-14
|
|---records: New For Each(false,false,false,false)[bag] - scope-13
    |   |
    |   Cast[int] - scope-2
    |   |
    |   |---Project[bytearray][0] - scope-1
    |   |
    |   Cast[int] - scope-5
    |   |
    |   |---Project[bytearray][1] - scope-4
    |   |
    |   Cast[int] - scope-8
    |   |
    |   |---Project[bytearray][2] - scope-7
    |   |
    |   Cast[int] - scope-11
    |   |
    |   |---Project[bytearray][3] - scope-10
    |
    |---records: Load(/output/delay.csv:PigStorage(',')) - scope-0

2014-04-06 05:31:09,248 [main] INFO
    org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler  - File
    concatenation threshold: 100 optimistic? false
2014-04-06 05:31:09,277 [main] INFO
    org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer  -
    MR plan size before optimization: 1
2014-04-06 05:31:09,277 [main] INFO
    org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer  -
    MR plan size after optimization: 1
```

Diagnostic Tools

```
#-----------------------------------------------
# Map Reduce Plan
#-----------------------------------------------
MapReduce node scope-15
Map Plan
records: Store( fakefile:org.apache.pig.builtin.PigStorage ) - scope-14
|
|---records: New For Each( false,false,false,false)[bag] - scope-13
    |   |
    |   Cast[int] - scope-2
    |   |
    |   |---Project[bytearray][0] - scope-1
    |   |
    |   Cast[int] - scope-5
    |   |
    |   |---Project[bytearray][1] - scope-4
    |   |
    |   Cast[int] - scope-8
    |   |
    |   |---Project[bytearray][2] - scope-7
    |   |
    |   Cast[int] - scope-11
    |   |
    |   |---Project[bytearray][3] - scope-10
    |
    |---records: Load(/output/delay.csv:PigStorage(',')) - scope-0--------
Global sort: false
----------------

grunt> ILLUSTRATE records
2014-04-06 05:38:21,451 [main] INFO
    org.apache.pig.backend.hadoop.executionengine.HExecutionEngine  - Connecting to
    hadoop file system at: hdfs://master:9000
...
2014-04-06 05:38:22,511 [main] INFO
    org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map  -
    Aliases being processed per job phase ( AliasName[line,offset]): M: records[1,10] C:
    R:
-------------------------------------------------------------------------------
| records     | year:int     | month:int     | count:int     | value:int     |
-------------------------------------------------------------------------------
|             | 2008         | 4             | 5491          | 268806        |
-------------------------------------------------------------------------------

grunt>
```

LOAD

◈ LOAD 'data' [USING function] [AS schema];

◈ data
- 디렉토리 또는 파일의 이름
- '(single quotes)로 묶여진다.

◈ USING
- 사용할 load 함수를 지정한다.
- 디폴트 함수는 PigStorage 이며 각 라인은 탭(₩t)로 구분한다.
- 구분자는 정규표현식을 이용하여 커스터마이징 할 수 있다.
- ,로 필드를 구분하려면 USING PigStorage(',') 절을 포함한다.
- 필드가 탭키로 구분되어 있으면 USING PigStorage('₩t')

◈ AS
- 입력 데이터의 스키마를 지정한다.
- 필드의 이름과 타입을 지정한다.

```
records = LOAD '/sample/info/member.log'
          USING PigStorage('.')
          AS (userId:chararray, age:int, address:chararray);
```

피그의 빌트인 함수에는 BinStorage() , PigStorage(field_delimiter), PigDump(), TextLoader() 등이 있습니다, PigStorage 함수는 gzip, bzip 압축을 지원합니다. BinStorage는 압축을 지원하지 않습니다.

BinStorage() : 바이너리 포맷 데이터를 로드하거나 저장합니다.
```
A = LOAD 'data' USING BinStorage();
STORE X into 'output' USING BinStorage();
```

PigStorage(field_delimiter) : UTF-8 포맷으로 데이터를 로드하거나 저장합니다.
```
A = LOAD 'student' USING PigStorage('₩t') AS (name: chararray, age:int, gpa: float);
A = LOAD 'student' AS (name: chararray, age:int, gpa: float);
STORE X INTO  'output' USING PigStorage('*');
```

PigDump() : UTF-8 포맷을 데이터를 저장합니다.
```
STORE X INTO 'output' USING PigDump();
```

TextLoader() : UTF8 포맷 비 구조적 데이터를 로드합니다.
```
A = LOAD 'data' USING TextLoader();
```

자동 형변환이 되지 않는 필드는 null이 됩니다.

DUMP와 STORE

◈ DUMP 또는 STORE 명령이 실행되기 전에는 어떤 액션도 실행되지 않는다.
- ➢ 피그는 DUMP 또는 STORE 명령이 실행되기 전에 실행한 명령은 유효성 검증과 분석을 위한 파싱을 하지만 실제 실행시키지는 않는다.

◈ DUMP
- ➢ 결과를 화면에 디스플레이 한다.

◈ STORE
- ➢ STORE alias INTO 'directory' [USING function];
- ➢ 결과를 저장한다.(일반적으로 파일에 저장한다.)

```
grunt> records = LOAD '/output/delay.csv' USING PigStorage(',') AS (year:int, m
    onth:int, count:int, value:int);
…
…
…
grunt> DUMP records;
```

dump 또는 store 를 실행하기 전에 피그는 전체 스크립트 청크를 최적화 한다.

함수가 실행되는 시점

출력해야 할 데이터가 많을 경우에는 다음과 같이 일부분만 출력할 수 있습니다. 다음은 10개 레코드를 출력합니다.

```
grunt> toPrint = LIMIT records 10;
grunt> DUMP toPrint;
```

다음은 하둡 파일 시스템에 저장합니다.

```
grunt> STORE toPrint INTO '/output/toprint/' USING PigStorage(',');
```

toPrint의 각 필드들의 구분자를 콤마(,)로 하여 /outptu/toprint 디렉토리에 part-r-0000 파일로 저장합니다.

Grouping

```
grunt> records = LOAD '/output/delay.csv'  USING PigStorage(',')  AS (year:int, month:int, count:int, value:int);
grunt> dump records;
(2007,1,12684,505125)
...
(2008,9,3113,157032)
grunt> recordsGroup = GROUP records BY year;
grunt> dump recordsGroup;
(2007,{(2007,1,12684,505125),(2007,9,5214,246889),(2007,8,11453,547181),(2007,7,15311,698240),(2007,6,17042,803209),(2007,5,8499,382267),(2007,4,7623,317424),(2007,3,9091,404860),(2007,2,13554,646296),(2007,12,15711,697218),(2007,11,5012,203195),(2007,10,6655,287745)})
(2008,{(2008,9,3113,157032),(2008,8,8630,398906),(2008,7,10061,555794),(2008,6,11995,571832),(2008,5,5715,240013),(2008,4,5491,268806),(2008,3,10198,474687),(2008,2,12188,544326),(2008,12,16683,747959),(2008,11,3662,139442),(2008,10,2476,105092),(2008,1,9773,429828)})
```

records 백을 year 필드로 그룹화 한다.

```
grunt> DESCRIBE records;
records: {year: int,month: int,count: int,value: int}

grunt> DESCRIBE recordsGroup;
recordsGroup: {group: int,records: {(year: int,month: int,count: int,value: int)}}

grunt> ILLUSTRATE records;
...
-------------------------------------------------------------------
| records     | year:int    | month:int    | count:int    | value:int    |
-------------------------------------------------------------------
|           | 2007        | 8            | 11453        | 547181       |
-------------------------------------------------------------------

grunt> ILLUSTRATE recordsGroup;
...
-------------------------------------------------------------------
| records     | year:int    | month:int    | count:int    | value:int    |
-------------------------------------------------------------------
|           | 2007        | 9            | 5214         | 246889       |
|           | 2007        | 3            | 9091         | 404860       |
-------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
| recordsGroup     | group:int    | records:bag{:tuple(year:int,month:int,count:int,value:int)}              |
-------------------------------------------------------------------------------------------------------------
|                | 2007         | {(2007, ..., 246889), (2007, ..., 404860)}                                |
-------------------------------------------------------------------------------------------------------------
```

Inner Bag

Outer Bag

FOREACH

- FOREACH <bag> GENERATE <data> AS <schema>
  - bag 안에 있는 모든 엘리먼트를 조사하여 결과를 생성한다.
- grunt> result = FOREACH bag GENERATE f1;
- grunt> result = FOREACH bag GENERATE group, FUNCTION(f1);
  - FOREACHE 문에 COUNT, FLATTEN, CONCAT 등의 함수를 사용할 수 있다.
  - 커스텀 함수를 포함할 수 있다.
- 함수
  - AVG, MIN, MAX, SUM
  - COUNT
  - CONCAT
  - SIZE

- 원본 데이터에서 연도와 지연횟수 정보만 출력하라.
- 그룹화 된 데이터를 분석해서 연도별 평균 지연횟수를 출력하라.

```
grunt> DESCRIBE records;
records: {year: int,month: int,count: int,value: int}

grunt> yearRecords = FOREACH records GENERATE year, count;
grunt> dump yearRecords;
(2007,12684)
(2007,6655)
(2007,5012)
(2007,15711)
(2007,13554)
...
(2008,5491)
(2008,5715)
(2008,11995)
(2008,10061)
(2008,8630)
(2008,3113)

grunt> DESCRIBE recordsGroup;
recordsGroup: {group: int,records: {(year: int,month: int,count: int,value: int)}}

grunt> yearAverage = FOREACH recordsGroup GENERATE group, AVG(records.count);

grunt> dump yearAverage
(2007,10654.083333333334)
(2008,8332.083333333334)
```

TOKENIZE 함수

◈ 문자열을 분리하여 토큰으로 만든 다음 bag에 출력 한다.
◈ 토큰으로 분리하는데 사용하는 문자
  - 공백, 쌍 따옴표("), 콤마(,), 괄호(()), 별표(*)
◈ 예
  - grunt> tokenBag = FOREACH linesOfText GENERATE TOKENIZE(line);

◈ FLATTEN 연산자
  - 중첩된 bag과 데이터 타입들을 단조롭게 한다.
  - grunt> flatBag = FOREACH tokenBag GENERATE flatten($0);

```
grunt> linesOfText = LOAD '/output/delay.csv' AS ( line:chararray);
grunt> dump linesOfText;
(2007,1,12684,505125)
(2007,10,6655,287745)
...
(2008,8,8630,398906)
(2008,9,3113,157032)

grunt> DESCRIBE linesOfText;
linesOfText: {line: chararray}
grunt> tokenBag = FOREACH linesOfText GENERATE TOKENIZE(line);
grunt> dump tokenBag
({(2007),(1),(12684),(505125)})
({(2007),(10),(6655),(287745)})
...
({(2008),(8),(8630),(398906)})
({(2008),(9),(3113),(157032)})

grunt> DESCRIBE tokenBag;
tokenBag: {bag_of_tokenTuples_from_line: {tuple_of_tokens: (token: chararray)}}
grunt>
grunt> flatBag = FOREACH tokenBag GENERATE flatten($0);
grunt> dump flatBag;
```

연산자

연산자	설명
SPLIT	SPLIT alias INTO alias IF expression, alias IF expression [, alias IF expression …]; 하나의 오브젝트를 두 개 이상의 오브젝트로 나눈다. 하나의 튜플이 한 개 이상의 오브젝트에 할당되거나, 전혀 할당되지 않을 수 있다.
UNION	alias = UNION alias, alias,[, alias …] 두 개 이상의 오브젝트 들을 합친다. 다음 내용을 확인하자 - 어떤 오브젝트 이 오더라도, 튜플의 순서를 보장하지 않는다. - 오브젝트 들이 동일한 스키마일 필요가 없다.심지어 필드 개수가 서로 달라도 된다. - 중복된 튜플을 제거하지 않는다.
FILTER	alias = FILTER alias BY expression; 표현(expression)의 참, 거짓에 따라서 어떤 튜플을 가져올지 결정한다. 필요한 튜플을 가져오거나,필요없는 튜플을 제거할 때 사용된다.
DISTINCT	alias = DISTINCT alias [PARALLEL n]; 중복된 튜플을 없애고, 하나만 나타낸다.
SAMPLE	alias = SAMPLE alias factor; 샘플 오브젝트를 임의로 생성한다. 샘플링 비율은 설정 가능하다. 예를 들어 small_data = SAMPLE LARGE_DATA 0.01;는 오브젝트 large_data의 1% 정도의 임의의 샘플 데이터를 small_data에 가져온 것이다. 이 연산은 확률적이라, small_data의 크기는 정확히 large_data의 1%가 아니고, 매번 같은 샘플 데이터를 가져온다는 보장은 없다.
FOREACH	alias = FOREACH alias GENERATE expression [,expression …] [AS schema]; 루프를 돌면서 각 튜플을 읽어서 새로운 튜플을 생성하는데, 필드 추가나 삭제 같은 데이터 칼럼을 변경하는 데 주로 사용된다. AS 옵션으로 생성된 결과의 스키마를 지정할 수 있다. 예를 들면, 새로운 필드에 이름을 지정한다.
FOREACH (nested)	alias = FOREACH nested_alias { alias = nested_op; [alias = nested_op; …] GENERATE expression [, expression …] ; }; 루프를 돌면서 튜플을 생성하며, nested_alias의 필드는 bag이어야 한다. 해당 bag 안에서의 연산을 위해 DISTINCT, FILTER ,LIMIT, ORDER, SAMPLE 등을 사용할 수 있다.
JOIN	alias = JOIN alias BY field_ alias, alias BY field_alias [,alias BY field_alias …] [USING “replicated”] [PARALLEL n]; 두 개 이상의 오브젝트간의 공통 필드값으로 inner join 을 계산한다. Pig는 빠른 처리를 위해 복제 옵션을 사용하여 첫 번째 연산 후 모든 오브젝트들을 메모리에 저장한다.물론 메모리 상황을 고려해 적당한 크기의 오브젝트 들인지 확인할 필요가 있다. JOIN에서 입력 오브젝트는 bag 같은 중첩된 필드가 없는, 일련의 기본 데이터 타입 필드만을 가지고 있다. 즉 입력 오브젝트는 flat하다고 표현한다. 또한 JOIN 결과 역시 flat 오브젝트를 가진다. 결과 relation의 필드 개수는 입력 relation들의 필드 개수의 합과 동일하다. 그리고 결과 relation의 스키마는 입력 relation의 스키마를 모두 합친 것이다.

연산자

연산자	설명
GROUP	alias = GROUP alias { [ALL] | [BY { [field_ alias [, field_ alias]] | * | [expression]] } [PARALLEL n] ; 하나의 오브젝트에서, 동일한 그룹 키를 가지고 튜플들을 그룹화한다. 보통 그룹 키는 하나 또는 그 이상의 필드로 지정된다. 심지어 전체 튜플(*) 또는 표현(expression)을 그룹 키로 사용할 수 있다. 모든 튜플들을 하나의 그룹으로 하기 위해 ALL 을 사용하는데, 해당 결과 오브젝트는 두 개의 필드를 갖게 되는데, 필드명은 Pig에서 자동 생성한다. 첫 번째 필드명은 항상 “group” 이고, 그룹 키와 동일한 타입이다. 두 번째 필드명은 입력 relation 과 동일한 이름을 가지고, 그 타입은 bag이다. 이 bag의 스키마는 입력 relation의 스키마와 같다.
COGROUP	alias = COGROUP alias BY field_ alias [INNER | OUTER], alias BY field_ alias [INNER | OUTER] [PARALLEL n]; 다수의 오브젝트의 튜플을 그룹화 한다. 그 기준으로, 공통 그룹 값(특정 필드가 될 수 있고, 위에서는 field_ alias 로 볼 수 있다) 을 사용한다. 연산 결과 생기는 오브젝트를 살펴보면, 공통 그룹 값에 해당(첫 번째 칠드) 하는 튜플은 하나만 존재할 것이다. 각 튜플은 첫 번째 필드에 그 그룹 값을 가지고 있고, 두 번째 필드는 입력 오브젝트에 존재하는 튜플 중에서 지정된 필드(field_ alias)값이 해당 그룹 값에 일치하는 튜플들을 모아 놓은 bag이다. 기본적으로 COGROUP은 OUTER join 처럼 동작한다. 즉 입력 오브젝트에 해당 그룹 값이 존재하는 모든 튜플이 결과 오브젝트에 나타나고, 만약 특정 그룹 값이 입력 오브젝트에 존재하지 않을 경우 빈 bag을 가진다. 만약 INNER 옵션이 지정되어 있으면, 입력 오브젝트에 해당 그룹 값이 일치하는 튜플만이 결과 오브젝트에 나타나게 된다. 즉 결과 오브젝트에는 빈 bag이 있을 수 없다. 다수의 필드들을 가지고 그룹화할 수 있다. 그렇게 하기위해 field_ alias에 콤마로 분리된 필드 목록을 지정하고, 괄호로 둘러싼다. COGROUP (INNER 옵션)은 중첩된 튜플을 결과로 갖는 것만 빼고는 JOIN과 비슷하다.
CROSS	alias= CROSS alias, alias [,alias ….] [PARALLEL n]; 두 개 이상의 오브젝트들을 가지고 교차 계산을 수행하기 때문에, 처리시간이 다른 연산에 비해 늦다. 가능한 한 이 연산을 피하도록한다.
ORDER	alias= ORFER alias BY { * [ASC | DESC] | field_alias [ASC | DESC],[, field_alias [ASC|DESC] …] } [PARALLEL n]; 하나 이상의 필드를 기준으로 오브젝트를 정렬한다. ORDER 바로 다음에 DUMP나 STORE를 수행하면, 정렬된 오브젝트가 보장되지만, FILTER, DINTINCT 같은 연산을 수행하면, 정렬이 제대로 되지 않을 수 있다.
STREAM	alias = STREAM alias [,alias …] THROUGH { ‘command’ | cmd_ alias} [AS schema]; 외부 스크립트로 오브젝트를 연산하고 처리한다.

http://pig.apache.org/docs/r0.13.0/basic.html 에서 Pig의 관계형 연산자들에 대한 설명서를 참고하시기 바랍니다.

SET

- 피그의 매개변수를 지정하는 것으로 프로그램의 성능에 미치는 영향이 큼
- 피그 스크립트 파일의 첫 번째 줄과, Grunt 쉘을 시작할 때 SET 명령어를 이용해서 리듀서의 개수를 지정하는 습관을 들이는 것을 권장
- grunt> set key value

Key	Value	설 명
default_parallel	숫자	피그에 의해 실행되는 맵리듀스 잡에서 리듀서의 개수를 지정한다.
debug	on 또는 off	디버깅 로그를 On 또는 Off 한다.
job.name	'job name'	Job에 대한 사용자 정의 이름을 지정한다. (앞뒤에 작은 따옴표를 사용)
job.priority	very_low, low, normal, high, very_high	피그 잡의 우선순위를 설정한다.
stream.skippath	문자열	스트리밍 작업시, 데이터를 제거하기 위한 경로를 설정한다.

다음은 SET 명령어를 사용한 예제입니다.

```
grunt> SET debug 'off';
grunt> SET job.name 'my pig job';
grunt> SET default_parallel 20;
```

다음은 5개의 리듀서가 실행되는 피그 스크립트 예제입니다. 맨 앞에 default_parallel 속성을 SET 명령어를 사용하여 지정한 후에 피그 문장을 작성했습니다. 이럴 때에는 Grunt 쉘을 종료할 때까지 아래에 작성된 모든 문장은 해당 속성을 따르게 됩니다.

```
SET default_parallel 5;
A = LOAD 'excite.txt' USING PigStorage() AS (t, u, v);
B = GROUP A BY t;
C = FOREACH B GENERATE GROUP, COUNT(A.t) AS mycnt;
D = ORDER C BY mycnt;
STORE D INTO 'cntoutput' USING PigStorage();
```

피그의 SET 명령어를 이용하여 여러 개의 key-value를 설정하는 것은 맵리듀스 프로그램에서 Job-Conf를 설정하는 것과 동일하다고 보면 됩니다. 다음은 지속성을 가지는 피그의 Job-Conf 설정의 예제입니다. SET 명령어를 이용하여 key-value를 설정하면 맨 마지막 값이 이후에 실행되는 모든 작업에 적용됩니다.

```
...
SET mapred.map.tasks.speculative.execution false;
SET pig.logfile mylogfile.log;
SET my key my value;
```

Lab

1. /output/delay.csv 파일을 분석하여 2007년 이후 데이터에 대하여 월별 지연횟수평균과 지연 시간 평균을 /pig_output3 디렉토리에 생성하세요.
위 스크립트를 스크립트 파일로 작성해서 실행시키세요.

```
$ vi monthlyDelayAverage.pig
records = LOAD '/output' USING PigStorage(',') AS (year:int,month:int,count:int,value:int);
records = FILTER records BY year >= 2007;
recordsGroup = GROUP records BY month;
result = FOREACH recordsGroup GENERATE group, AVG(records.count) AS avgcount, AVG(records.value) AS avgvalue;
STORE result INTO '/pig_output3' USING PigStorage(',');

$ pig monthlyDelayAverage.pig
```

2. 항공운항 데이터를 분석하여 날씨관계로 지연된 항공편의 지연 횟수와 평균 시간을 연도별 월별로 출력하는 스크립트를 작성하세요.

```
$ vi weatherDelay.pig
airline = LOAD '/airline' USING PigStorage(',') AS (Year:int, Month:int,
DayofMonth:int, DayOfWeek:int,
DepTime:int, CRSDepTime:int, ArrTime:int, CRSArrTime:int,
UniqueCarrier:chararray, FlightNum:int, TailNum:chararray,
ActualElapsedTime:int, CRSElapsedTime:int, AirTime:int,
ArrDelay:int, DepDelay:int,
Origin:chararray, Dest:chararray,
Distance:int, TaxiIn:int, TaxiOut:int,
Cancelled:int, CancellationCode:chararray, Diverted:chararray,
CarrierDelay:int, WeatherDelay:int, NASDelay:int, SecurityDelay:int, LateAircraftDelay:int);

airlineNN = FILTER airline BY (WeatherDelay is not null);

airlineFiltered = FILTER airlineNN BY (WeatherDelay > 0);

airlineGroup = GROUP airlineFiltered BY (Year, Month);

airlineRecords = FOREACH airlineGroup GENERATE (group.Year, group.Month, COUNT(airlineFiltered.Month),

AVG(airlineFiltered.WeatherDelay));

$ pig weatherDelay.pig
```

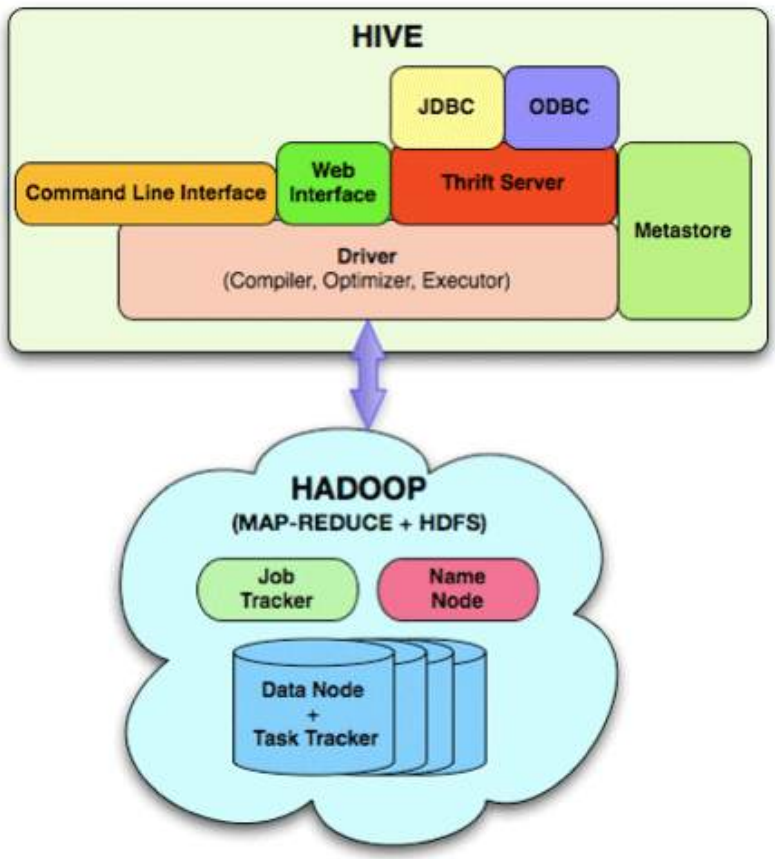
2 Hive

Objectives

이 페이지는 여백 페이지입니다.

Hive

- http://hive.apache.org/
- 하둡 기반의 데이터 웨어하우징 프레임워크
  - 페이스북의 제프 해머바처(Jeff Hammerbacher) 팀이 개발
- 하이브는 사용자의 워크스테이션에서 수행되고 SQL 쿼리가 하둡 클러스터에서 구동되도록 일련의 맵리듀스 작업으로 변경됨
- 데이터를 테이블로 표현하여 구조체와 HDFS에 저장된 데이터가 연결되는 수단을 제공
- 테이블 스키마 같은 메타데이터는 메타스토어(metastore)라는 데이터베이스에 저장



HIVE 설치

◈ 설치 환경
- jdk 1.7 u51(필수)
- hadoop 2.2.0(필수)
- hive 0.12.0
  - Hadoop 2.2.0에서 Hive 0.13이 정상 실행되지 않습니다.

◈ # tar -xvf hive-0.12.0-tar.gz

◈ $ vi .bash_profile
- export HIVE_HOME=/usr/local/hive-0.12.0
- export PATH=$PATH:$HIVE_HOME/bin


```
[root@master ~]# cd /usr/local/
[root@master local]# tar -xvf /home/hadoop/Downloads/hive-0.12.0.tar.gz
[root@master local]# chown -R hadoop:hadoop hive-0.12.0/
[root@master local]# ls -l
total 64
drwxr-xr-x.  6 hadoop hadoop 4096 Dec 23 06:52 apache-ant-1.9.3
drwxr-xr-x.  8 hadoop hadoop 4096 May  6 22:19 hive-0.12.0
drwxr-xr-x.  9 hadoop hadoop 4096 Mar  6 07:44 apache-tomcat-7.0.52
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 bin


[hadoop@master ~]$ $HIVE_HOME/bin/hive
...
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
hive> show tables;
OK
Time taken: 4.287 seconds
```

HIVE 쉘

```
[hadoop@master ~]$ $HIVE_HOME/bin/hive
hive> SHOW TABLES;
OK
Time taken: 4.287 seconds
hive>
```

◈ 비 대화식 모드 실행
- $ hive -f script.hql
- hive> source /path/to/file/withqueries.hql

◈ 명령어를 인라인으로 지정
- $ hive -e 'SELECT * FROM dummy'
- $ hive -S -e "select * FROM mytable LIMIT 3" > /tmp/myquery

◈ !을 붙이면 운영체제 명령 실행
- $ hive> !hadoop fs -ls /usr/hive/;

HiveQL은 대소분자를 구분하지 않습니다. 그리고 탭 키를 사용하면 하이브가 제공하는 예약어와 함수를 자동 완성 시킬 수 있습니다.

SHOW TABLE; 명령어를 수행하면 몇 초가 걸릴 수 있습니다. 그 이유는 메타스토어 데이터베이스가 필요한 시점에 생성되기 때문입니다. 이 데이터베이스는 hive 명령어를 실행한 위치에 metastore_db라는 이름의 디렉토리를 만들어 필요한 파일을 저장합니다.

```
[hadoop@master ~]$ ls -l
total 48
-rw-rw-r--. 1 hadoop hadoop  641 May  6 22:38 derby.log
drwxr-xr-x. 2 hadoop hadoop 4096 May  4 07:12 Desktop
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Documents
drwxr-xr-x. 2 hadoop hadoop 4096 May  6 18:13 Downloads
drwxrwxr-x. 3 hadoop hadoop 4096 Apr 13 07:29 Lab
drwxrwxr-x. 5 hadoop hadoop 4096 May  6 22:38 metastore_db
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Music
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Pictures
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Public
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Templates
drwxr-xr-x. 2 hadoop hadoop 4096 Feb 18 04:18 Videos
drwxrwxr-x. 5 hadoop hadoop 4096 May  4 07:11 workspace
[hadoop@master ~]$
```

HIVE 서비스

- --service 옵션으로 서비스를 지정할 수 있다.
- cli
  - 하이브 쉘에 대한 명령 행 인터페이스
- hiveserver
  - 여러 언어로 개발한 클라이언트가 연동하도록 쓰리프트(Thrift) 서비스를 제공하는 서버로 실행하라
  - 서버가 수신할 포트(기본값 10,000번)를 지정하려면 HIVE_PORT 환경변수 수정
- hwi
  - 하이브 웹 인터페이스
- jar
  - 자바 응용프로그램을 실행하기 위한 방법이다. hive jar는 hadoop jar에 상응
- metastore
  - 메타스토어를 독립형(원격) 프로세스로 실행시킬 수 있다.

하이브 웹 인터페이스

다음 명령으로 하이브 웹 인터페이스를 시작할 수 있습니다.

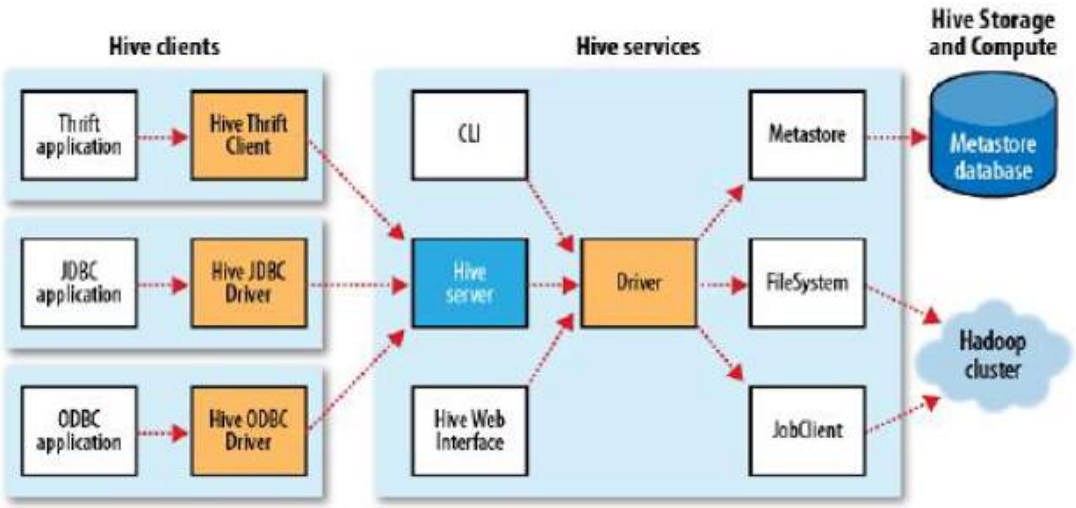
```
[hadoop@master ~]$ export ANT_LIB=/usr/local/apache-ant-1.9.3/lib
[hadoop@master ~]$ hive --service hwi
```

하이브 웹 인터페이스를 사용하려면 설정파일에 몇 가지 속성이 추가되어야 합니다. 하이브 웹 인터페이스에 관한 자세한 내용은 설정파일에 대한 소개 후 설명됩니다.

브라우저로 http://localhsot:9999/hwi에 접속하면 하이브 웹 인터페이스를 사용할 수 있습니다.

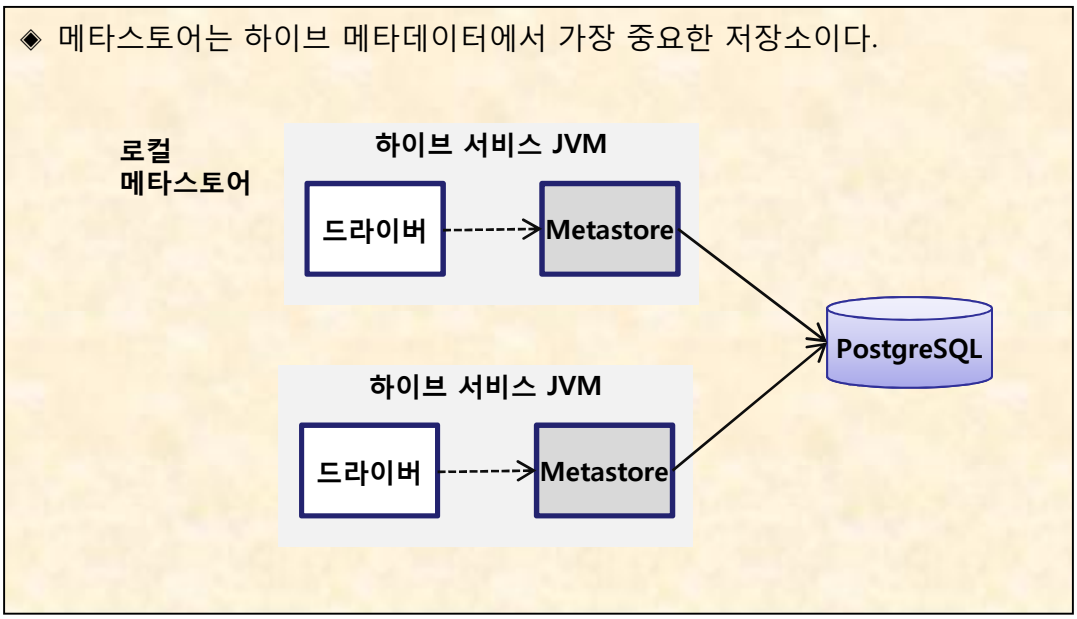
하이브 클라이언트

사용자가 hive --service hiveserver 명령으로 하이브 서버를 실행하고 나면 응용프로그램은 다양한 방법으로 하이브에 연결할 수 있습니다.

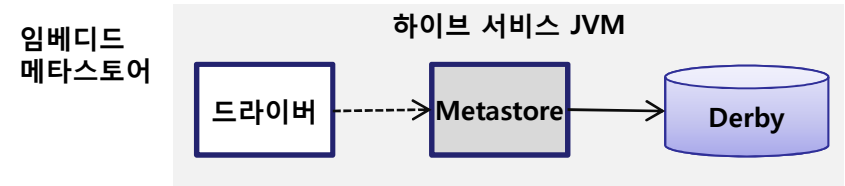


HIVE 메타스토어

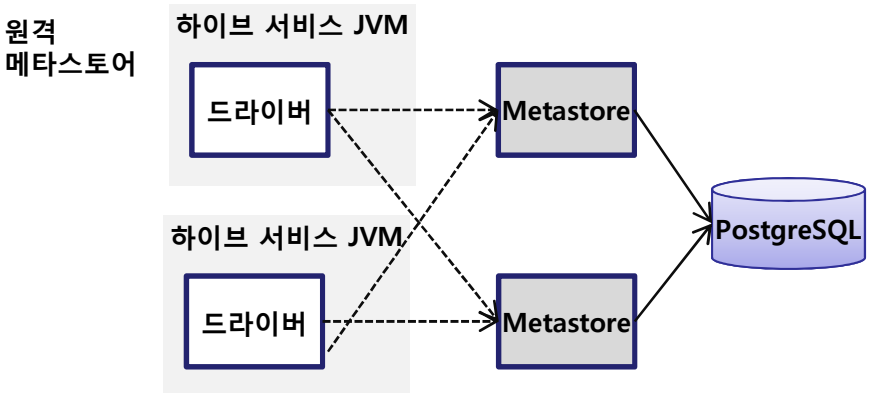
◈ 메타스토어는 하이브 메타데이터에서 가장 중요한 저장소이다.



메타스토어는 서비스와 데이터 백업 저장소로 나뉩니다. 기본적으로 메타데이터 서비스는 하이브 서비스와 동일한 JVM에서 실행되고 로컬 디스크에 백업되는 임베디드 더비 데이터베이스 인스턴스를 포함합니다. 이를 임베디드 메타스토어 설정이라고 합니다.



원격 메타스토어는 하나 이상의 메타스토어 서버가 하이브 서비스와는 별도의 프로세스로 실행됩니다. 데이터 베이스 계층이 방화벽의 역할을 대신하고, 클라이언트는 더 이상 데이터베이스 자격을 얻을 필요가 없기 대문에 원격 메타스토어 설정은 좋은 관리성과 보안을 제공합니다.



HIVE 메타스토어 환경설정 속성

- hive.metastore.warehouse.dir
  - 관리 테이블에 저장되는 fs.default.name에 상대적인 디렉토리
  - 기본값은 /user/hive/warehouse
- hive.metastore.local
  - 임베디드 메타스토어 서버를 사용할 지(true), 원격 인스턴스로 연결할지 (false) 여부. false라면 hive.metastore.uris가 설정되어야 한다.
- hive.metastore.uris
  - ,로 구분 된 값을 사용하며 연결할 원격 메타스토어 서버를 지정하는 URI
- javax.jdo.option.ConnectionURL
  - JDBC 커넥션 URL
- javax.jdo.option.ConnectionDriverName
  - JDBC 드라이버 클래스 명
- javax.jdo.option.ConnectionUserName
  - JDBC 사용자 명
- javax.jdo.option.ConnectionPassword
  - JDBC 암호

하이브는 기본값으로 derby를 사용하여 메타스토어를 저장합니다. 더비 메타스토어는 싱글로만 동작하며 한 세션이 점유하고 있으면 다른 하이브 세션이 정상적으로 구동되지 않습니다. 이런 문제를 해결하기 위해 다른 RDM를 사용하는데 여기에서는 PostgreSQL 데이터베이스를 사용합니다.

제공되는 환경에는 이미 PostgreSQL 데이터베이스가 설치되어 있으며 이미 데이터베이스와 계정이 생성되어 있습니다.

PostgreSQL 과 관련된 설정 내용은 다음과 같습니다.
드라이버 클래스명 : org.postgresql.Driver
커넥션 URL : jdbc:postgresql://localhost:5432/hadooptestdb
사용자명 : scott
사용자 암호 : tiger

루트로 접속해서 PostgreSQL 데이터베이스를 실행해야 합니다.
service postgresql-9.2 start

PostgreSQL 설치에 관한 내용은 아래 문서를 참고하세요.
http://cafe.naver.com/javaspecialistgroup/154

$ hdfs dfs –mkdir –p /uesr/hive/warehouse
$ hdfs dfs –mkdir –p /tmp/hive
$ mkdir ~/tmp

HIVE 메타스토어 환경설정 속성

다음은 conf/hive-site.xml 파일의 내용입니다.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<!-- 하둡상의 tmp 경로 -->
<property>
  <name>hive.exec.scratchdir</name>
  <value>/tmp/hive</value>
  <description>Scratch space for Hive jobs</description>
</property>
<!-- 시스템상의 tmp 경로 -->
<property>
  <name>hive.exec.local.scratchdir</name>
  <value>/home/hadoop/tmp</value>
  <description>Local scratch space for Hive jobs</description>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://master:9000</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://localhost:5432/hadooptestdb</value>
</property>
<!-- metastore db 유저 -->
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>scott</value>
</property>
<!-- metastore db 유저의 비번 -->
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>tiger</value>
</property>
</configuration>
```

HIVE 웹 인터페이스

하이브 웹 인터페이스를 사용하기 위해서는 conf/hive-site.xml 파일에 아래 내용을 추가해야 합니다.

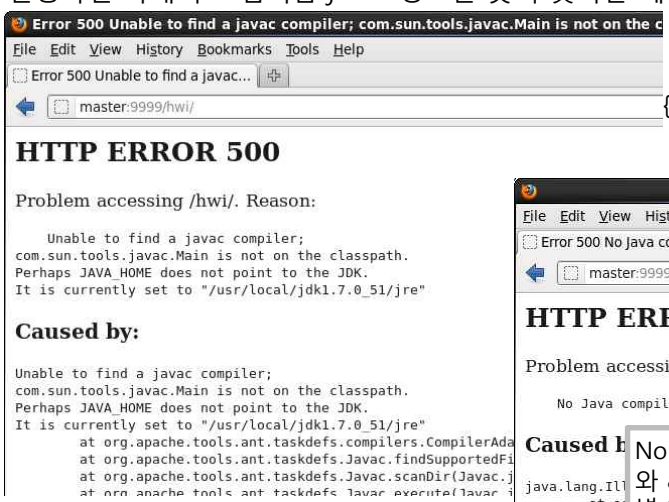
```
<property>
  <name>hive.hwi.listen.host</name>
  <value>master</value>
</property>
<property>
  <name>hive.hwi.listen.port</name>
  <value>9999</value>
</property>
<property>
  <name>hive.hwi.war.file</name>
  <value>lib/hive-hwi-0.12.0.war</value>
</property>
```

위 내용 중에서 master는 하이브 웹 인터페이스를 실행시키는 호스트의 이름입니다. 아이피 또는 도메인을 기록해도 됩니다.

그 아래 부분의 hive.hwi.war.file의 경로는 절대경로가 아닌 상대경로만 지원됩니다. {HIVE_HOME}이 기준입니다.

```
<property>
    <name>hive.hwi.war.file</name>
    <value>lib/hive-hwi-0.12.0.war</value>
</property>
```

실행하면 아래의 그림처럼 javac 경로를 찾지 못하는 에러가 발생할 수 있습니다.



{JAVA_HOME}/lib/tools.jar 파일을 {HIVE_HOME}/lib에 복사하면 해결됩니다.

No Java compiler available 에러는 ant-launcher.jar와 ant.jar를 찾지 못했기 때문입니다. ANT_LIB 환경변수를 설정해 줘야 합니다.
export ANT_LIB=$ANT_HOME/lib

실행은 다음 명령으로 합니다.

```
[hadoop@master ~]$ hive --service hwi
```

만약 이렇게 하면 백그라운드 실행은 아니므로 다음과 같이 별도 쉘을 만들어 백그라운드로 실행할 것을 추천합니다.

```
[hadoop@master ~]$ hive --service hwi > hwi.log &
```

HIVE 테이블 생성

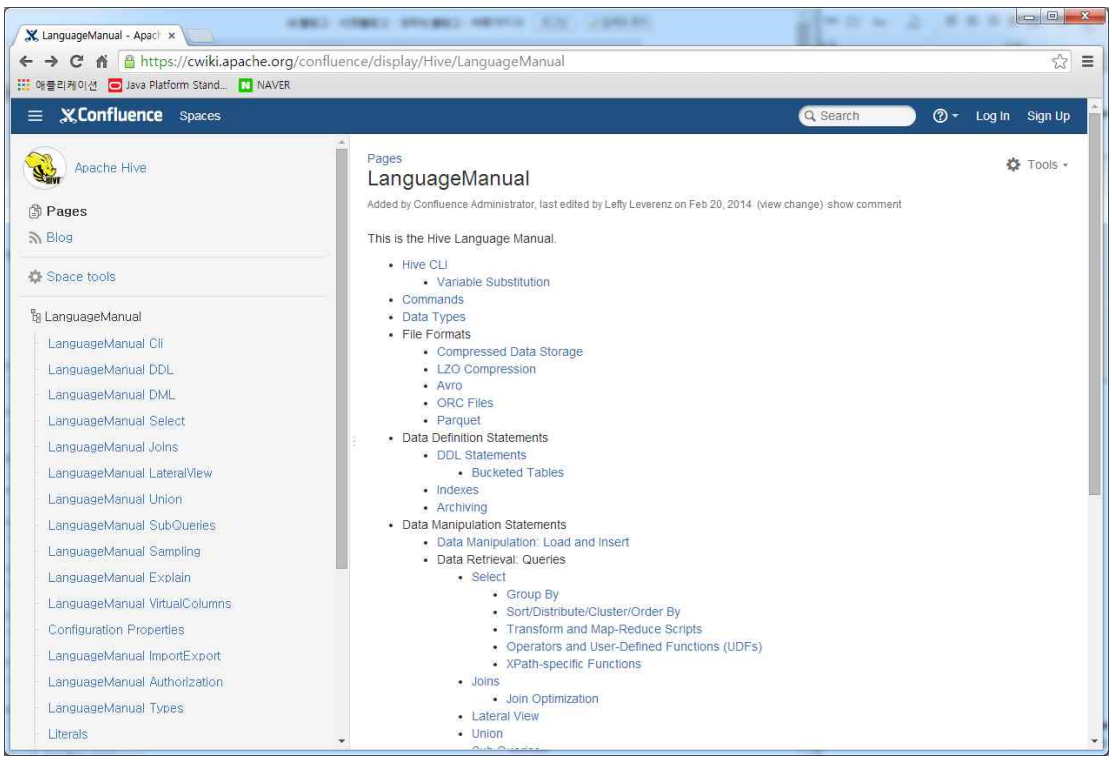
```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)]
INTO num_buckets BUCKETS]
  [SKEWED BY (col_name, col_name, ...) ON ([(col_value, col_value, ...), ...|col_value,
col_value, ...]) [STORED AS DIRECTORIES]
  [
   [ROW FORMAT row_format] [STORED AS file_format]
   | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION hdfs_path]
  [TBLPROPERTIES (property_name=property_value, ...)]
  [AS select_statement]
```

```
hive> create external table if not exists delay(
    > year int, month int, delaycount int, delayvalue int)
    > row format delimited fields terminated by ',' lines terminated by ' ₩n'
    > location '/output';
OK
Time taken: 0.122 seconds
hive> select * from delay;
OK
2008            1               9773            429828
2008            10              2476            105092
2008            11              3662            139442
2008            12              16683           747959
2008            2               12188           544326
2008            3               10198           474687
2008            4               5491            268806
2008            5               5715            240013
2008            6               11995           571832
2008            7               10061           555794
2008            8               8630            398906
2008            9               3113            157032
Time taken: 0.533 seconds, Fetched: 12 row(s)
```

HIVE QL

- MySQL의 SQL 표현 방식에 가장 닮았다.
- SQL-92 표준 명세서를 완벽하게 지원하지는 않는다.
- 데이터형
  - 기본형
    - TINYINT, SMALLINT, INT, BIGINT
    - FLOAT, DOUBLE, BOOLEAN, STRING, BINARY, TIMESTAMP
  - 복합형
    - ARRAY, MAP, STRUCT
- 연산자와 함수
  - 관계 연산자(=, IS NULL, LIKE), 산술 연산자, 논리연산자(||는 논리연산자, 문자열 연결은 concat함수)
  - SHOW FUNCTIONS
- 테이블
  - 관리 테이블(테이블 생성 후)
    - LOAD DATA INPATH '/xxx' INTO table xxx_managed;
  - 외부 테이블

HiveQL의 완벽한 레퍼런스를 원한다면 아래 사이트를 방문하세요.
https://cwiki.apache.org/confluence/display/Hive/LanguageManual



WordCount

```
CREATE TABLE docs (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE docs;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\s')) AS word FROM docs) w GROUP BY word
ORDER BY word;
```

Pig와 Hive 분석 비교 실습

Q. 1차 분석된 항공운항 데이터를 이용해서 2차 분석하세요. 분석해야 할 데이터는 년도 별 지연 횟수 평균입니다.
[hadoop@master Lab]$ hadoop jar WeatherDelay.jar /airline명령으로 생성된 /output/delay.csv 파일을 분석하면 됩니다.

1. Pig를 이용해서 분석하세요.

```
grunt> records = LOAD '/output/delay.csv' USING PigStorage(',')
>>     AS (year:int, month:int, count:int, value:int);
grunt> recordsGroup = GROUP records BY year;
grunt> yearAverage = FOREACH recordsGroup GENERATE group, AVG(records.count);
grunt> dump yearAverage
```

Pig를 이용해서 분석한 결과는 다음과 같습니다.
(2006,9478.583333333334)
(2007,10654.083333333334)
(2008,8332.083333333334)

2. Hive를 이용해서 분석하세요.

```
hive> create external table if not exists delay (
    > year int, month int, delaycount int, delayvalue int)
    > row format delimited fields terminated by ','
    > lines terminated by '₩n'
    > location '/output';

hive> select year, avg(delaycount) from delay group by year;
```

Hive를 이용해서 분석한 결과는 다음과 같습니다.

2006	9478.583333333334
2007	10654.083333333334
2008	8332.083333333334

3 Sqoop

Objectives

- Sqoop 을 이용하면 정형 데이터를 하둡 파일 시스템으로 쉽게 옮길 수 있습니다.

이 페이지는 여백 페이지입니다.

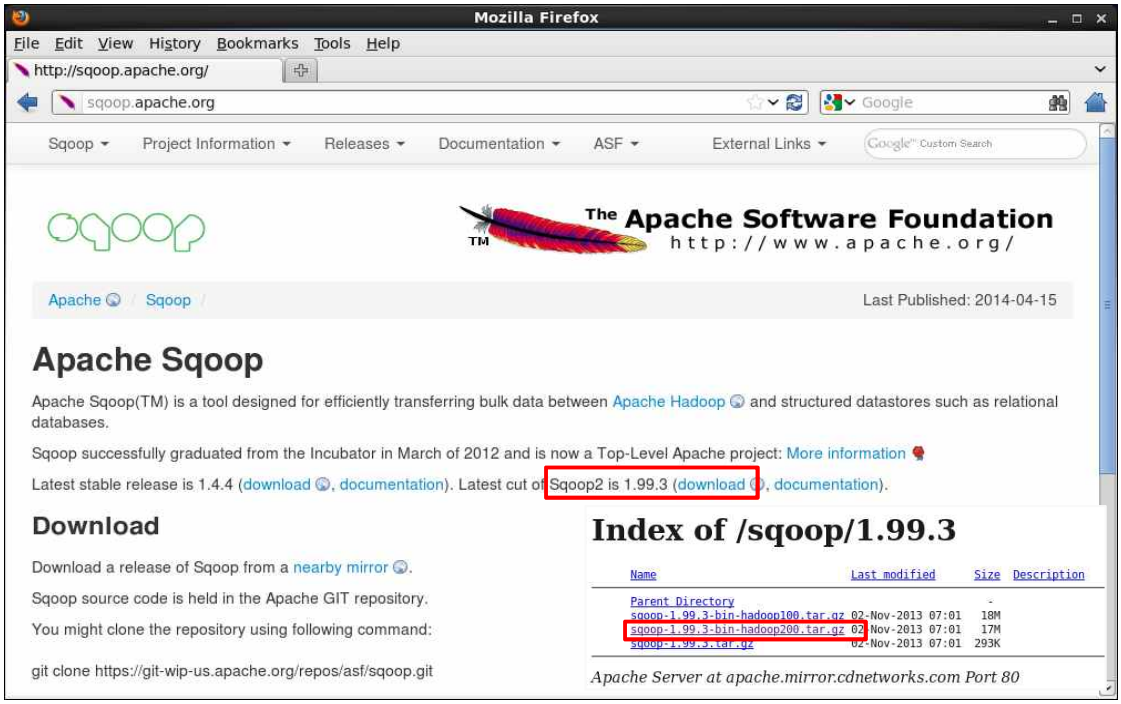
Sqoop

- http://sqoop.apache.org
- Bulk data transfer tool
  - 관계형 데이터베이스, 엔터프라이즈 데이터 웨어하우스, NoSQL 시스템 등으로부터 Import/Export
  - Oozie 스케줄러를 이용한 import/export task 자동화
  - 커넥터 기반 아키텍처를 통한 플러그인 지원
- 설치 환경
  - jdk 1.7 u51(필수)
  - hadoop 2.2.0(필수)
  - Sqoop2 1.99.3

Sqoop 설치

1. Sqoop 다운로드

http://sqoop.apache.org/ 에 접속해서 다운로드(Sqoop2 1.99.3 Download)



미러사이트 클릭한 다음 sqoop-1.99.3-bin-hadoop200.tar.gz 다운로드

2. /usr/local/ 디렉토리에 루트 권한으로 압축 해제

```
# cd /usr/local/
[root@master local]# tar -xvf /home/hadoop/Downloads/sqoop-1.99.3-bin-hadoop200.tar.gz
```

3. 사용자 변경

```
[root@master local]# chown -R hadoop:hadoop sqoop-1.99.3-bin-hadoop200/
[root@master local]# mv sqoop-1.99.3-bin-hadoop200/ sqoop-1.99.3/
[root@master local]# ls -l
total 64
drwxr-xr-x.  6 hadoop hadoop 4096 Dec 23 06:52 apache-ant-1.9.3
drwxr-xr-x.  9 hadoop hadoop 4096 Mar  6 07:44 apache-tomcat-7.0.52
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 bin
drwxrwsr-x.  9 hadoop hadoop 4096 Apr 26 06:33 eclipse
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 etc
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 games
drwxr-xr-x. 11 hadoop hadoop 4096 Feb 18 04:55 hadoop-2.2.0
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 include
```

다음페이지에 이어집니다.

Sqoop 환경변수 설정

```
drwxr-xr-x.  8 hadoop hadoop 4096 Dec 18 19:25 jdk1.7.0_51
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 lib
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 libexec
drwxr-xr-x. 17 hadoop hadoop 4096 Mar 26 05:23 pig-0.12.0
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 sbin
drwxr-xr-x.  5 root   root   4096 Feb 18 13:06 share
drwxr-xr-x. 17 hadoop hadoop 4096 Apr 26 07:57 sqoop-1.99.3
drwxr-xr-x.  2 root   root   4096 Sep 23  2011 src
[root@master local]#
```

4. 환경변수 설정

```
[hadoop@master Desktop]$ cd
[hadoop@master ~]$ vi .bash_profile
export SQOOP_HOME=/usr/local/sqoop-1.99.3
...
export PATH=$PATH:$SQOOP_HOME/bin
```

5. 하둡 설정 파일 경로 지정

```
[hadoop@master conf]$ vi $SQOOP_HOME/server/conf/sqoop.properties
# Hadoop configuration directory
org.apache.sqoop.submission.engine.mapreduce.configuration.directory=/usr/local/hadoop
-2.2.0/etc/hadoop/
```

6. 필요한 하둡 라이브러리 설정

[hadoop@master conf]$ vi $SQOOP_HOME/server/conf/catalina.properties

```
common.loader=${catalina.base}/lib,${catalina.base}/lib/*.jar,${catalina.home}/l
ib,${catalina.home}/lib/*.jar,${catalina.home}/../lib/*.jar,/usr/lib/hadoop/*.ja
r,/usr/local/hadoop-2.2.0/share/hadoop/common/*.jar,/usr/local/hadoop-2.2.0/shar
e/hadoop/common/lib/*.jar,/usr/local/hadoop-2.2.0/share/hadoop/yarn/*.jar,/usr/l
ocal/hadoop-2.2.0/share/hadoop/yarn/lib/*.jar,/usr/local/hadoop-2.2.0/share/hado
op/hdfs/*.jar,/usr/local/hadoop-2.2.0/share/hadoop/hdfs/lib/*.jar,/usr/local/had
oop-2.2.0/share/hadoop/mapreduce/*.jar,/usr/local/hadoop-2.2.0/share/hadoop/mapr
educe/lib/*.jar
```

다음 jar 파일들이 추가된 것임

/usr/local/hadoop-2.2.0/share/hadoop/common/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/common/lib/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/yarn/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/yarn/lib/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/hdfs/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/hdfs/lib/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/mapreduce/*.jar
/usr/local/hadoop-2.2.0/share/hadoop/mapreduce/lib/*.jar

Sqoop 시작

7. 하둡 클러스터 시작

```
hadoop@master ~]$ $HADOOP_INSTLL/sbin/start-all.sh
```

8. PostgreSQL JDBC Driver 복사

PostgreSQL JDBC 드라이버 파일을 SQOOP_HOME/server/lib 폴더에 복사해 주세요.

```
[hadoop@master~]$ cp ~/Downloads/postgresql-9.2-1004.jdbc41.jar $SQOOP_HOME/server/lib
```

9. PostgreSQL 서버 시작

```
[hadoop@master Desktop]$ su -
Password:
[root@master ~]# service postgresql-9.2 start
Starting postgresql-9.2 service:                           [  OK  ]
```

스쿱 서버를 실행 하기 전에 JDBC 드라이버 파일이 복사되어 있어야 합니다.

10. 스쿱 서버 시작

```
[hadoop@master ~]$ sqoop.sh server start
Sqoop home directory: /usr/local/sqoop-1.99.3
Setting SQOOP_HTTP_PORT:     12000
Setting SQOOP_ADMIN_PORT:     12001
Using   CATALINA_OPTS:
Adding to CATALINA_OPTS:    -Dsqoop.http.port=12000 -Dsqoop.admin.port=12001
Using CATALINA_BASE:   /usr/local/sqoop-1.99.3/server
Using CATALINA_HOME:   /usr/local/sqoop-1.99.3/server
Using CATALINA_TMPDIR: /usr/local/sqoop-1.99.3/server/temp
Using JRE_HOME:        /usr/local/jdk1.7.0_51
Using CLASSPATH:       /usr/local/sqoop-1.99.3/server/bin/bootstrap.jar
```

11. 스쿱 클라이언트 시작

```
[hadoop@master ~]$ sqoop.sh client
Sqoop home directory: /usr/local/sqoop-1.99.3
Apr 26, 2014 8:25:38 AM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Sqoop Shell: Type 'help' or '₩h' for help.

sqoop:000>
```

스쿱 서버를 실행시킨 후 jps 로 프로세스 확인했을 때 Bootstrap 프로세스가 보여야 합니다.

```
[hadoop@master ~]$ jps
4588 ResourceManager
4161 NameNode
4285 DataNode
4432 SecondaryNameNode
4696 NodeManager
18567 Bootstrap
8777 JobHistoryServer
18609 Jps
```

12. Connection 생성

```
sqoop:000> create connection --cid 1
Creating connection for connector with id 1
Please fill following values to create new connection object
Name: postgresql
```

Sqoop Job 생성

```
Connection configuration

JDBC Driver Class: org.postgresql.Driver
JDBC Connection String: jdbc:postgresql://localhost:5432/hadooptestdb
Username: scott
Password: tiger
JDBC Connection Properties:
There are currently 0 values in the map:
entry#

Security related configuration options

Max connections: 10
New connection was successfully created with validation status FINE and persistent id
1
sqoop:000> show connection
+----+------------+-----------+---------+
| Id |    Name    | Connector | Enabled |
+----+------------+-----------+---------+
| 1  | postgresql | 1         | true    |
+----+------------+-----------+---------+
```

13. Job 생성

```
sqoop:000> create job --xid 1 --type import
Creating job for connection with id 1
Please fill following values to create new job object
Name: Emp

Database configuration

Schema name: public
Table name: emp
Table SQL statement:
Table column names: empno,ename,job,mgr,hiredate,sal,comm,deptno
Partition column name:
Nulls in partition column:
Boundary query:

Output configuration
```

컬럼의 길이는 50자 이하

Sqoop Job 실행

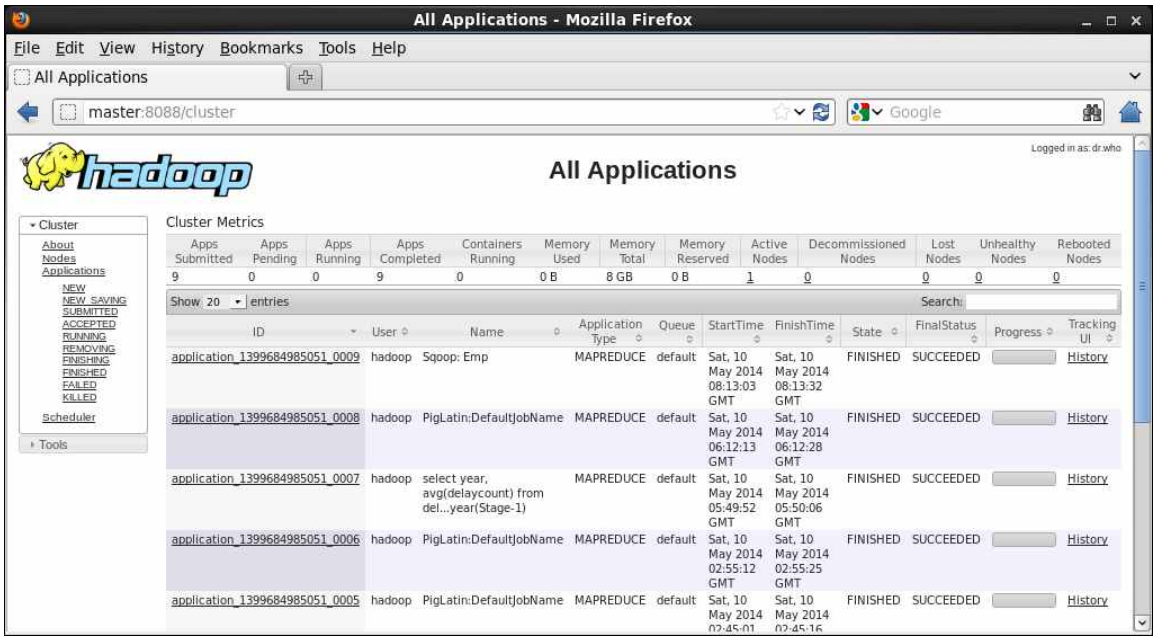
```
Storage type:
  0 : HDFS
Choose: 0
Output format:
  0 : TEXT_FILE
  1 : SEQUENCE_FILE
Choose: 0
Compression format:
  0 : NONE
  1 : DEFAULT
  2 : DEFLATE
  3 : GZIP
  4 : BZIP2
  5 : LZO
  6 : LZ4
  7 : SNAPPY
Choose: 0
Output directory: /output/emp

Throttling resources

Extractors:
Loaders:
New job was successfully created with validation status FINE  and persistent id 1
```

14. Job 실행

```
sqoop:000> start job --jid 1
Submission details
Job ID: 1
Server URL: http://localhost:12000/sqoop/
Created by: hadoop
Creation date: 2014-05-10 01:13:02 PDT
Lastly updated by: hadoop
External ID: job_1399684985051_0009
             http://master:8088/proxy/application_1399684985051_0009/
2014-05-10 01:13:02 PDT: BOOTING  - Progress is not available
```

Sqoop Job 실행

15. http://master:8088으로 잡 실행 확인



16. HDFS에 생성된 파일 확인

```
[hadoop@master ~]$ hdfs dfs -ls /output/emp
Found 11 items
-rw-r--r--   1 hadoop supergroup          0 2014-04-30 09:25 /output/emp/_SUCCESS
-rw-r--r--   1 hadoop supergroup         53 2014-04-30 09:25 /output/emp/part-m-00000
-rw-r--r--   1 hadoop supergroup          0 2014-04-30 09:25 /output/emp/part-m-00001
-rw-r--r--   1 hadoop supergroup        111 2014-04-30 09:25 /output/emp/part-m-00002
-rw-r--r--   1 hadoop supergroup         56 2014-04-30 09:25 /output/emp/part-m-00003
-rw-r--r--   1 hadoop supergroup          0 2014-04-30 09:25 /output/emp/part-m-00004
-rw-r--r--   1 hadoop supergroup        114 2014-04-30 09:25 /output/emp/part-m-00005
-rw-r--r--   1 hadoop supergroup          0 2014-04-30 09:25 /output/emp/part-m-00006
-rw-r--r--   1 hadoop supergroup        112 2014-04-30 09:25 /output/emp/part-m-00007
-rw-r--r--   1 hadoop supergroup        168 2014-04-30 09:25 /output/emp/part-m-00008
-rw-r--r--   1 hadoop supergroup        163 2014-04-30 09:25 /output/emp/part-m-00009
[hadoop@master ~]$ hdfs dfs -cat /output/emp/part-m-00000
7369,'SMITH','CLERK',7902,1980-12-17,800.0,'null',20
```

Sqoop Job 실행

17. Job 상태 확인

history 서버가 실행되어 있어야 합니다.

```
sqoop:000> status job --jid 1
Submission details
Job ID: 1
Server URL: http://localhost:12000/sqoop/
Created by: hadoop
Creation date: 2014-05-10 01:13:02 PDT
Lastly updated by: hadoop
External ID: job_1399684985051_0009
             http://master:8088/proxy/application_1399684985051_0009/
2014-05-10 01:13:33 PDT: SUCCEEDED
Counters:
             org.apache.hadoop.mapreduce.JobCounter
                          SLOTS_MILLIS_MAPS: 95543
                          TOTAL_LAUNCHED_MAPS: 10
                          OTHER_LOCAL_MAPS: 10
             org.apache.hadoop.mapreduce.lib.output.FileOutputFormatCounter
                          BYTES_WRITTEN: 777
             org.apache.hadoop.mapreduce.lib.input.FileInputFormatCounter
                          BYTES_READ: 0
             org.apache.hadoop.mapreduce.TaskCounter
                          MAP_INPUT_RECORDS: 0
                          MERGED_MAP_OUTPUTS: 0
                          PHYSICAL_MEMORY_BYTES: 1061195776
                          SPILLED_RECORDS: 0
                          COMMITTED_HEAP_BYTES: 833880064
                          CPU_MILLISECONDS: 4750
                          FAILED_SHUFFLE: 0
                          VIRTUAL_MEMORY_BYTES: 13022064640
                          SPLIT_RAW_BYTES: 1271
                          MAP_OUTPUT_RECORDS: 14
                          GC_TIME_MILLIS: 581
             org.apache.hadoop.mapreduce.FileSystemCounter
                          FILE_WRITE_OPS: 0
                          FILE_READ_OPS: 0
                          FILE_LARGE_READ_OPS: 0
                          FILE_BYTES_READ: 0
                          HDFS_BYTES_READ: 1271
                          FILE_BYTES_WRITTEN: 889520
                          HDFS_LARGE_READ_OPS: 0
                          HDFS_WRITE_OPS: 20
                          HDFS_READ_OPS: 40
                          HDFS_BYTES_WRITTEN: 777
             org.apache.sqoop.submission.counter.SqoopCounters
                          ROWS_READ: 14
Job executed successfully
```

에러…

Caused by: Exception: java.net.ConnectException Message: Call From master/192.168.224.135 to 0.0.0.0:10020 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused

$HADOOP_INSTALL/etc/hadoop/mapred-site.xml에 아래 내용 추가

```
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>master:10020</value>
</property>
```

*** 에러 발생했을 경우 에러 메시지를 자세하게 보고 싶다면 다음과 같이 verbose 옵션을 true로 설정합니다.**

sqoop:000>set option --name verbose --value true

*** Job Submission status**

Status	Description
BOOTING	In the middle of submitting the job
FAILURE_ON_SUBMIT	Unable to submit this job to remote cluster
RUNNING	The job is running now
SUCCEEDED	Job finished successfully
FAILED	Job failed
NEVER_EXECUTED	The job has never been executed since created
UNKNOWN	The status is unknown

참고

```
show connection
update connection -xid 1
show job
update job --jid 1
start job --jid 1
status job --jid 1
stop job --jid 1
delete job -jid 1
```

이 페이지는 여백 페이지입니다.

4 Flume

Objectives

- Flume 설치와 사용법에 대해서 알아보겠습니다.

이 페이지는 여백 페이지입니다.

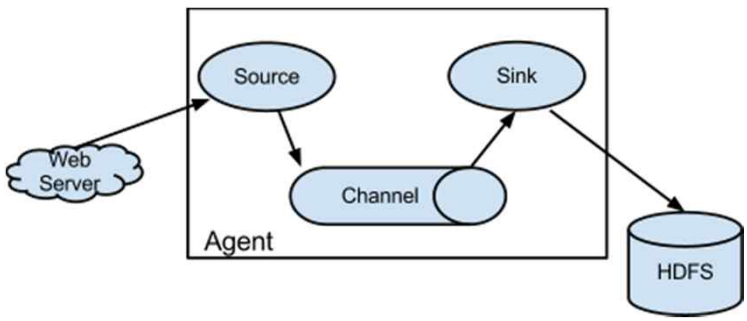
Flume

- 플룸(Flume)는 분산 환경에서 대량의 로그 데이터를 효과적으로 수집하여, 합친 후 다른 곳으로 전송할 수 있는 신뢰할 수 있는 서비스이다.
- 플룸은 단순하며 유연한 스트리밍 데이터 플로우(streaming data flow) 아키텍처를 기반으로 한다.
- 또한 플룸은 장애에 쉽게 대처 가능하며, 로그 유실에 대한 신뢰 수준을 상황에 맞게 변경할 수 있을 뿐 만 아니라, 장애 발생시 다양한 복구 메커니즘을 제공한다.
- 실시간으로 로그를 분석하는 어플리케이션을 개발할 수 있도록, 간단하며 확장 가능한 데이터 모델을 사용한다.

클러스터에 있는 모든 장치로부터 로그 파일들을 수집한 후, 하둡 분산 파일 시스템(HDFS)과 같은 중앙 저장소에 저장하는 로깅 시스템을 구축해야 할 때 플룸은 가장 제격입니다..

플룸은 아래의 핵심 목표를 만족시키도록 만들어졌습니다.
- 신뢰성(Reliability)
- 확장성(Scalability)
- 운영가능성(Manageability)
- 확장성(Extensibility)

flume 아파치 프로젝트 홈 :http://flume.apache.org
Getting Started 페이지: https://cwiki.apache.org/confluence/display/FLUME/Getting+Started
도큐멘트 페이지 :http://flume.apache.org/documentation.html
다운로드 페이지 :http://flume.apache.org/download.html



Flume 아키텍처

- 플룸 아키텍처는 스트림 지향의 데이터 플로우를 기반으로 한다.
- 데이터 플로우(data flow)란 하나의 데이터 스트림이 생성지에서 목표지로 전달되어 처리되는 방식을 뜻한다.
- 데이터 플로우는 이벤트를 전송하고 수집하는 일련의 논리 노드(logical node)들로 구성된다.
- 여러 논리 노드들을 서로 순차적으로 연결하여 데이터 플로우를 구성한다.
- 논리 노드가 서로 연결되는 방식을 논리 노드 설정(logical node's configuration)이라고 부른다.
- 이러한 논리 노드들은 모두 플룸 마스터(Flume Master)에서 관리한다.
- 플룸 마스터란 플룸이 설치된 물리 노드(physical node)와 논리 노드에 대한 정보를 유지하고 있는 별도의 서비스다.
- 플룸 마스터가 설정을 논리 노드에 할당하며, 사용자가 설정을 바꾼 경우 변경된 사항을 모든 논리 노드에 알려주는 역할을 한다.
- 각 논리노드는 차례대로 플룸 마스터와 주기적으로 통신하여(heartbeat), 모니터링 관련 정보를 서로 공유하며 자신과 관련된 설정정보가 변경되었는지 확인한다.

플룸의 아키텍처는 단순하며, 튼튼하고, 유연합니다. 플룸의 아키텍처는 플룸의 핵심 목표인 신뢰성(Reliability), 확장성(Scalability), 운영가능성(Manageability), 확장성(Extensibility)을 어떻게 만족시켰는지 설명하고 있습니다.

Flume 설치

- http://flume.apache.org/
- 설치 환경
  - jdk 1.7 u51(필수)
  - hadoop 2.2.0(필수)
  - flume-1.4.0
- 루트 계정으로 /usr/local에 설치 후 소유권한 변경 및 디렉토리 이름 변경
  - # tar -xvf /home/hadoop/Downloads/apache-flume-1.4.0-bin.tar.gz
  - # chown -R hadoop:hadoop /usr/local/apache-flume-1.4.0-bin/
  - # mv apache-flume-1.4.0-bin/ apache-flume-1.4.0/
- SQOOP_HOME 환경변수 추가, PATH에 $SQOOP_HOME/bin 추가

hadoop 2.2 에서는 아래와 같은 에러가 발생합니다.

```
class
    org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$DeleteSnapshot
    ResponseProto overrides final method
    getUnknownFields.()Lcom/google/protobuf/UnknownFieldSet;

Stacktrace

java.lang.VerifyError: class
    org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$DeleteSnapshot
    ResponseProto overrides final method
    getUnknownFields.()Lcom/google/protobuf/UnknownFieldSet;

at java.lang.ClassLoader.defineClass1(Native Method)
```

이럴 경우에는 {flume-home}/lib 에서 아래 두 jar 파일을 삭제 또는 백업 해 놓고 실행하면 됩니다.

```
[root@master ~]# cd /usr/local/apache-flume-1.4.0/lib/
[root@master lib]# mv protobuf-java-2.4.1.jar protobuf-java-2.4.1.jar.bak
[root@master lib]# mv guava-10.0.1.jar guava-10.0.1.jar.bak
```

```
14/05/07 03:42:08 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
14/05/07 03:42:08 INFO hdfs.BucketWriter: Creating hdfs://master:9000/tmp/flume/140507/03/api.1399459328056.log
14/05/07 03:42:08 ERROR hdfs.HDFSEventSink: process failed
java.lang.VerifyError: class org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$SetOwnerRequestProto override
s final method getUnknownFields.()Lcom/google/protobuf/UnknownFieldSet;
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(ClassLoader.java:800)
```

Flume 에이전트 설정파일

- [hadoop@master Desktop]$ cd /usr/local/apache-flume-1.4.0/conf/
- [hadoop@master conf]$ cp flume-conf.properties.template flume.conf
- [hadoop@master conf]$ vi flume.conf

- source : local file
  - agent.sources.logSource.spoolDir=/home/hadoop/temp
  - /home/hadooop/temp 디렉토리에 저장되는 파일(디렉토리 생성해야 함)
- channel : memory
- sink : hdfs
  - agent.sinks.hdfsSink.hdfs.path=hdfs://master:9000/tmp/flume/%y%m%d/%H
  - /tmp/flume/년월일/시간 디렉토리에 파일 생성되도록 함
  - HDFS에 /tmp/flume 디렉토리 생성 후 쓰기 권한 부여해야 함

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#  http://www.apache.org/licenses/LICENSE-2.0
#
# 생략…
# The configuration file needs to define the sources,
# the channels and the sinks.
# Sources, channels and sinks are defined per agent,
# in this case called 'agent'

logAgent.sources = logSource
logAgent.channels = memoryChannel
logAgent.sinks = hdfsSink
```

Flume 에이전트 설정파일

```
# For each one of the sources, the type is defined
logAgent.sources.logSource.type = spooldir

# The channel can be defined as follows.
logAgent.sources.logSource.channels = memoryChannel
logAgent.sources.logSource.spoolDir=/home/hadoop/temp
logAgent.sources.logSource.fileHeader=true

# Each sink's type must be defined
logAgent.sinks.hdfsSink.type = hdfs

#Specify the channel the sink should use
logAgent.sinks.hdfsSink.channel = memoryChannel
logAgent.sinks.hdfsSink.hdfs.path=hdfs://master:9000/tmp/flume/%y%m%d/%H
logAgent.sinks.hdfsSink.hdfs.round=true
logAgent.sinks.hdfsSink.hdfs.roundUnit=minute
logAgent.sinks.hdfsSink.hdfs.fileType=DataStream
logAgent.sinks.hdfsSink.hdfs.filePrefix=api
logAgent.sinks.hdfsSink.hdfs.inUseSuffix=.log
logAgent.sinks.hdfsSink.hdfs.useLocalTimeStamp=true

# Each channel's type is defined.
logAgent.channels.memoryChannel.type = memory

# Other config values specific to each type of channel(sink or source)
# can be defined as well
# In this case, it specifies the capacity of the memory channel
logAgent.channels.memoryChannel.capacity = 100
```

Flume을 이용한 로그 데이터 수집

◈ 실행

- # ./bin/flume-ng agent -c ./conf -f ./conf/*${설정파일이름}.conf* -n *${agent명}*
- -Dflume.monitoring.type=http -Dflume.monitoring.port=5555 를 붙이면 5555 포트로 모니터링이 가능하다.

◈ ./flume-ng agent -c conf -f ../conf/flume.conf -n logAgent -Dflume.root.logger=INFO,console

console 은 소문자입니다.

◈ 실행 후 agent.sources.logSource.spoolDir 디렉토리에 파일 생성 후 HDFS의 agent.sinks.hdfsSink.hdfs.path 디렉토리에 파일 저장되는지 확인

```
[hadoop@master temp]$ hadoop fs -ls /tmp/flume
Found 1 items
drwxrwxrwx   - hadoop supergroup          0 2014-05-07 03:46 /tmp/flume/140507
[hadoop@master temp]$ hadoop fs -ls /tmp/flume/140507/
Found 1 items
drwxrwxrwx   - hadoop supergroup          0 2014-05-07 03:47 /tmp/flume/140507/0
3
[hadoop@master temp]$ hadoop fs -ls /tmp/flume/140507/03
Found 1 items
-rw-r--r--   1 hadoop supergroup         26 2014-05-07 03:47 /tmp/flume/140507/0
3/api.1399459596880
[hadoop@master temp]$ hadoop fs -cat /tmp/flume/140507/03/api.1399459596880
```

위와 같이 스크립트를 실행 한 다음 설정파일에서 지정한 디렉토리(/home/hadoop/temp)에 파일을 생성해 보세요. 자동으로 하둡 파일시스템에 파일이 저장되는 것을 확인할 수 있습니다.

```
hadoop@master:/usr/local/apache-flume-1.4.0/bin
File  Edit  View  Search  Terminal  Help
14/05/07 03:46:21 INFO conf.FlumeConfiguration: Processing:hdfsSink
14/05/07 03:46:21 INFO conf.FlumeConfiguration: Processing:hdfsSink
14/05/07 03:46:21 INFO conf.FlumeConfiguration: Post-validation flume configuration contains configuration for agents: [agent
]
14/05/07 03:46:21 INFO node.AbstractConfigurationProvider: Creating channels
14/05/07 03:46:21 INFO channel.DefaultChannelFactory: Creating instance of channel memoryChannel type memory
14/05/07 03:46:21 INFO node.AbstractConfigurationProvider: Created channel memoryChannel
14/05/07 03:46:21 INFO source.DefaultSourceFactory: Creating instance of source logSource, type spooldir
14/05/07 03:46:21 INFO sink.DefaultSinkFactory: Creating instance of sink: hdfsSink, type: hdfs
14/05/07 03:46:21 INFO hdfs.HDFSEventSink: Hadoop Security enabled: false
14/05/07 03:46:21 INFO node.AbstractConfigurationProvider: Channel memoryChannel connected to [logSource, hdfsSink]
14/05/07 03:46:21 INFO node.Application: Starting new configuration:{ sourceRunners:{logSource=EventDrivenSourceRunner: { sou
rce:org.apache.flume.source.SpoolDirectorySource{name:logSource,state:IDLE} }} sinkRunners:{hdfsSink=SinkRunner: { policy:org
.apache.flume.sink.DefaultSinkProcessor@c04835 counterGroup:{ name:null counters:{} } }} channels:{memoryChannel=org.apache.f
lume.channel.MemoryChannel{name: memoryChannel}} }
14/05/07 03:46:21 INFO node.Application: Starting Channel memoryChannel
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Monitoried counter group for type: CHANNEL, name: memoryChannel
, registered successfully.
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: memoryChannel started
14/05/07 03:46:21 INFO node.Application: Starting Sink hdfsSink
14/05/07 03:46:21 INFO node.Application: Starting Source logSource
14/05/07 03:46:21 INFO source.SpoolDirectorySource: SpoolDirectorySource source starting with directory: /home/hadoop/temp
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Monitoried counter group for type: SINK, name: hdfsSink, regist
ered successfully.
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: hdfsSink started
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Monitoried counter group for type: SOURCE, name: logSource, reg
istered successfully.
14/05/07 03:46:21 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: logSource started
14/05/07 03:46:35 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /home/hadoop/temp/a.log to /home/hadoop/t
emp/a.log.COMPLETED
14/05/07 03:46:36 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
14/05/07 03:46:36 INFO hdfs.BucketWriter: Creating hdfs://master:9000/tmp/flume/140507/03/api.1399459596880.log
```