

# 巨量資料分析應用與實作班

Brian Luan  
2015/07/01

# 關於我

- 現職：
  - 大數軟體有限公司 顧問
  - 網鈺數位科技 技術經理
- 曾任職
  - 新創 - MIGO entertainment
  - IBM/ORACLE 資料解決方案
- 大數學堂 <http://course.largitdata.com/>
- 粉絲頁 <https://www.facebook.com/largitdata>





# 輿情分析 & 輿論監控





卡提諾論壇  
CK101.com

阿宅



iBeauty  
愛漂亮

女人



DONG  
動網

男人



媽媽經

媽媽



NIUS  
妞新聞

女人

# 使用者為中心的大數據實務

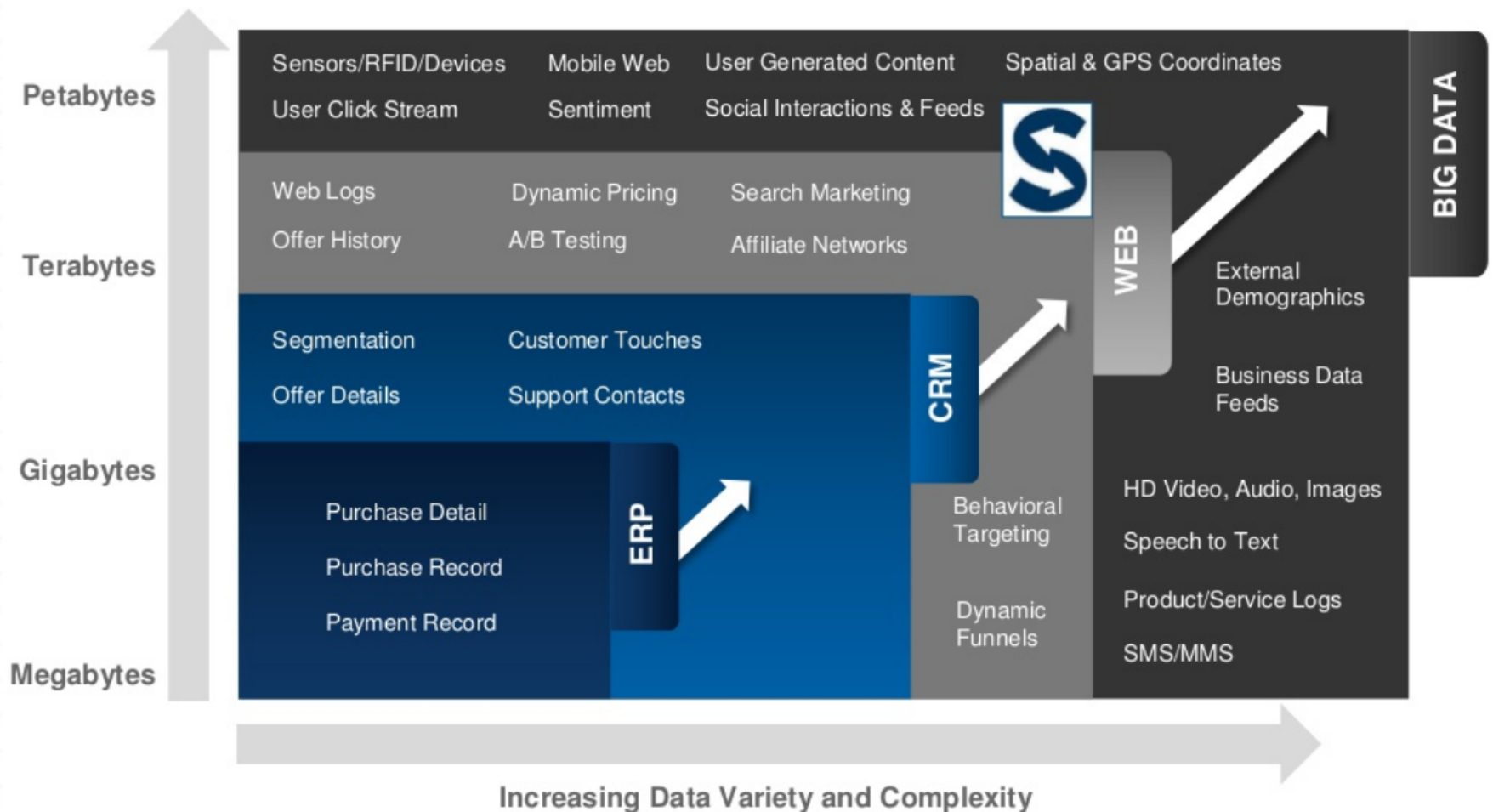




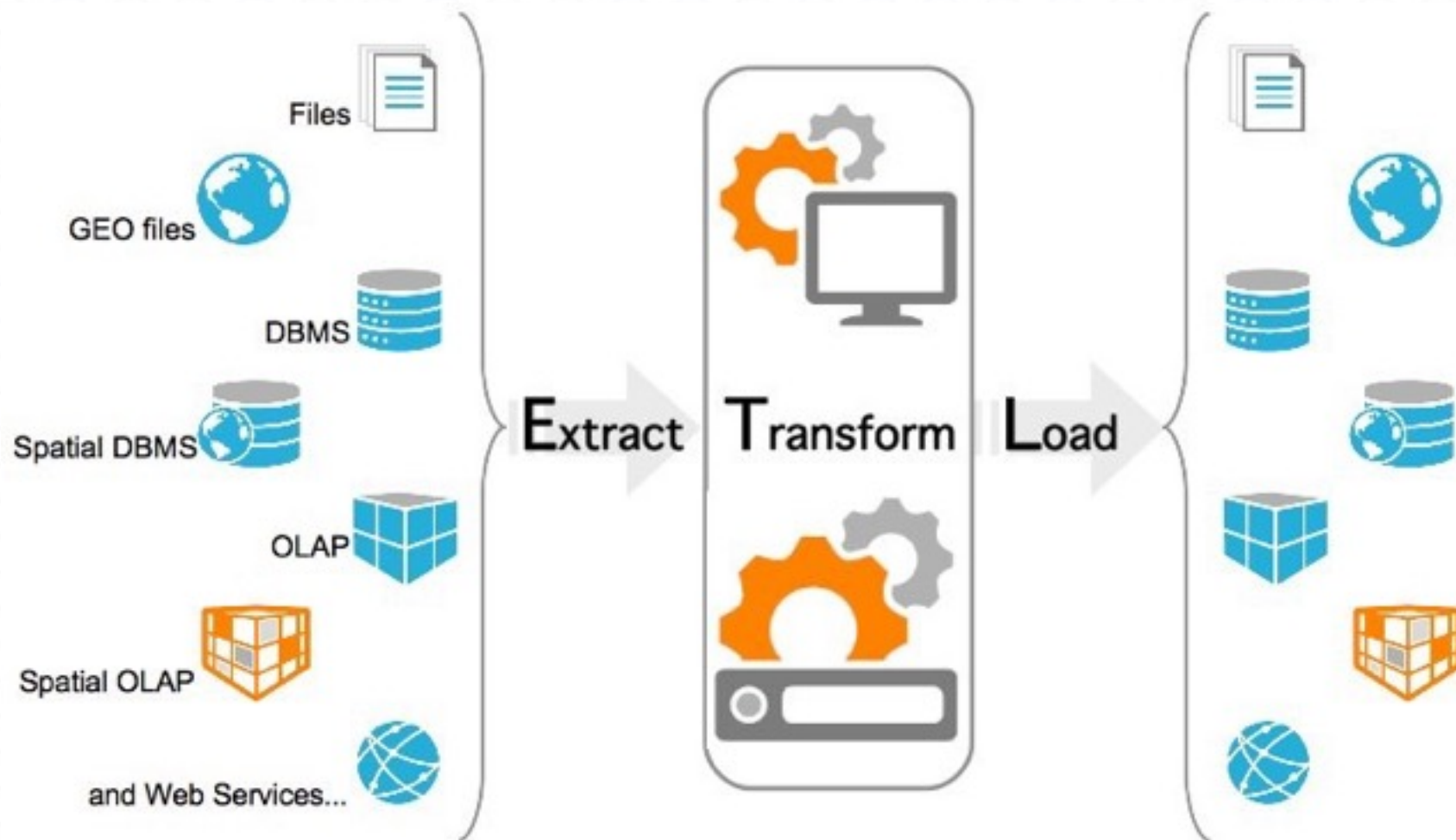
Big Data:  
Expanding on 3 fronts  
at an increasing rate.



# Big Data = Transactions + Interactions + Observations





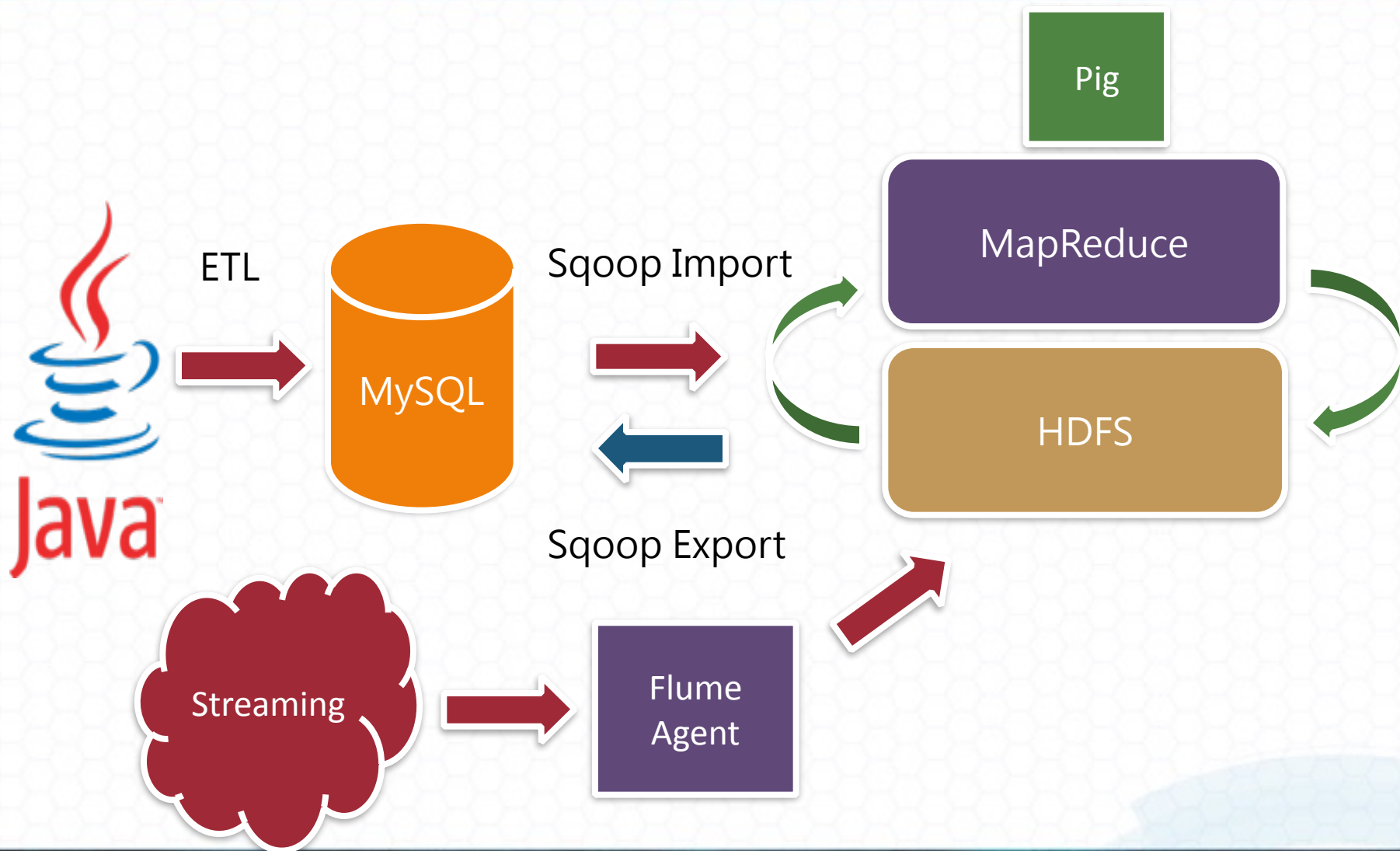






# 資料分析模組 **PIG**介紹

# 分析情境



# PIG

- 由Yahoo 發明
- 類腳本語言 (Scripting Like – Data Flow Language)
- 將MapReduce 工作抽象化
  - ▣ 實際上是轉化成MapReduce 工作後在Cluster 上執行
- 功能
  - ▣ 合併資料
  - ▣ 聚合資料
  - ▣ 流程控制
  - ▣ 客製化載入



# 分析顧客對商品評價

- 找出顧客平均評價高於三的產品

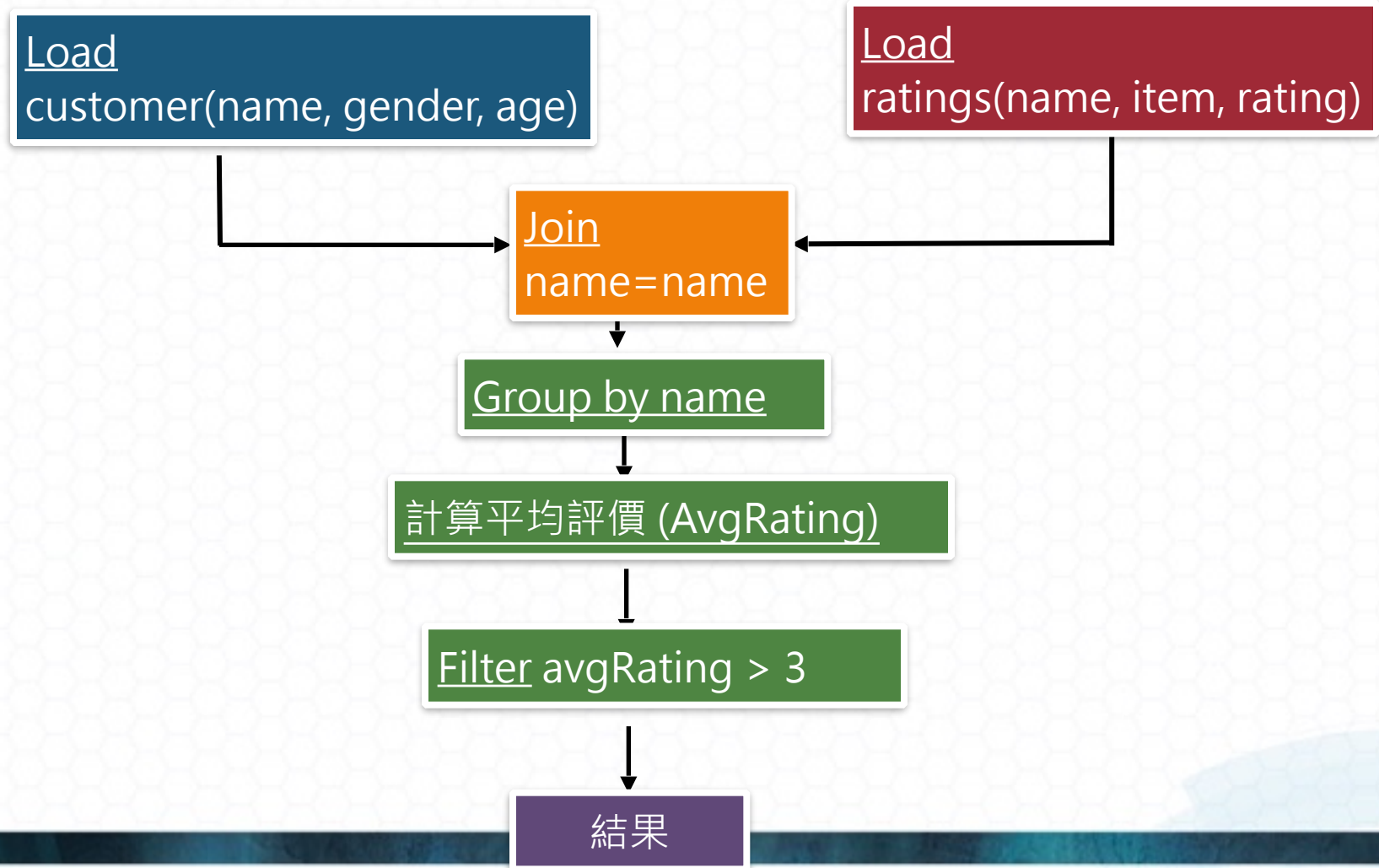
customer

name	gender	age
John	M	30
Amy	F	23
Jane	F	47
James	M	25

rating

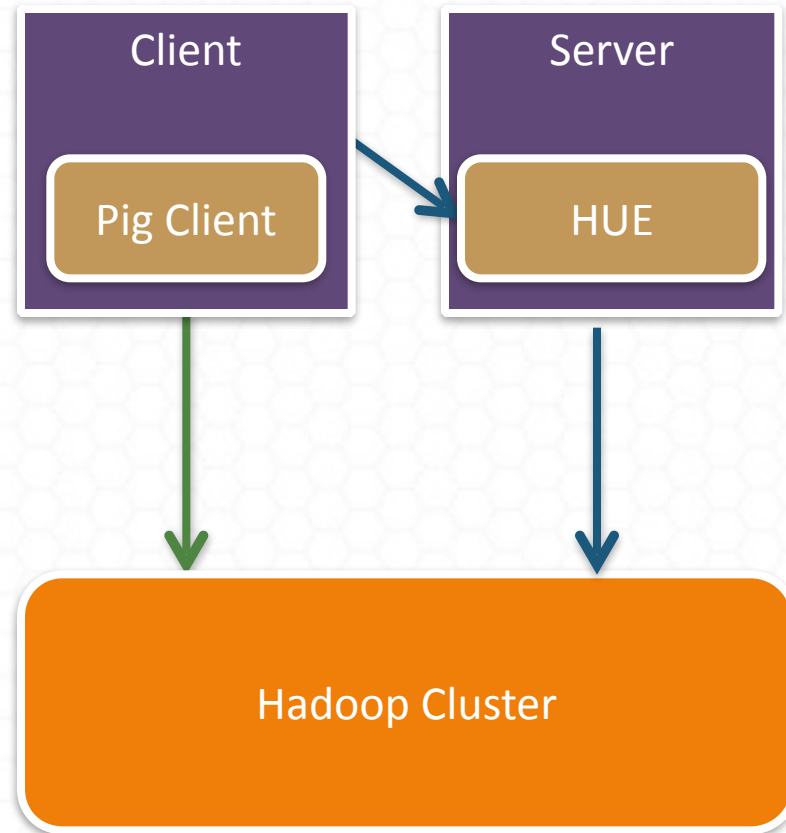
name	item	rating
John	PS4	5
John	Iphone	4
Amy	PS4	3
James	PS4	5

# 資料分析流程



# PIG架構

- 使用Client 存取
  - 可以單獨安裝在Client 機器上
  - 透過Pig Client 送出MR工作
- 安裝成Central Service
  - 使用Hue
  - Talend, Qubole
- 不需要Metastore





## Pig 如何工作

- Pig 會跟Pre-built 的Binaries 會轉換成Jar檔，並透過Map Reduce 的工作執行Query
- 不會產生Java 原始碼

# Pig 的優點

- 簡潔方便 (相較於Java 而言)
- 擅長處理非結構化資料
  - ▣ Map, Tuple, Databags
- 系統管理員跟開發人員容易上手 (類似Shell)
- 提供流程控制(Data Flow Control)
- 擅長進行資料ETL (Transformation)

# Pig 的缺點

- 無法即時性分析(使用Impala)
- 非高效能 (使用Impala)
- 如需要共用的架構 (View, Table) (使用Hive)
- 如需要使用資料庫已有的SQL Script (使用Hive)
- 資料有明確的Schema (使用Hive)



# 執行模式

- Interactive mode

- ▣ 會直接進入grunt shell互動模式，console前面會多帶一個grunt >

- Batch mode

- ▣ 直接執行 pig script檔案內容

# 範例檔案

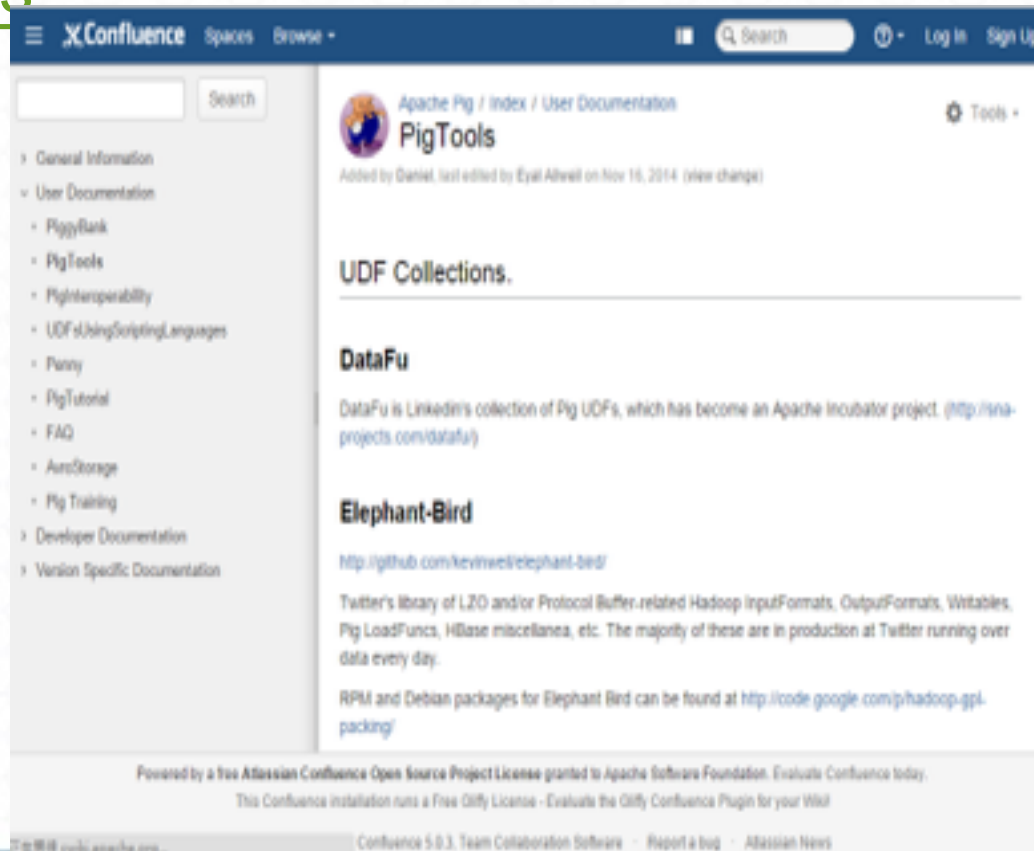
name	item	rating
<i>Amy</i>	<i>Iphone</i>	<i>4</i>
<i>Amy</i>	<i>PS4</i>	<i>4</i>
<i>John</i>	<i>XBOX</i>	<i>5</i>
<i>John</i>	<i>PS4</i>	<i>4</i>
<i>Tom</i>	<i>PS4</i>	<i>3</i>

sample.pig

```
records = load '/data/ratings.tsv'  
  as (name: chararray, item:chararray, rating: int);  
dump records;  
describe records;
```

# 編輯pig 檔案

- <https://cwiki.apache.org/confluence/display/PIG/PigTools>



The screenshot shows the Apache PigTools page on the Confluence wiki. The page has a blue header with the Confluence logo and navigation links. A left sidebar contains a table of contents with links to General Information, User Documentation (PiggyBank, PigTools, PigInteroperability, UDF Using Scripting Languages, Penny, PigTutorial, FAQ, AutoStorage, Pig Training), Developer Documentation, and Version Specific Documentation. The main content area features the Apache Pig logo, the title 'PigTools', and a note that it was added by Daniel and last edited by Eyal Alkell on Nov 16, 2014. Below this, there is a section for 'UDF Collections' with two entries: 'DataFu' and 'Elephant-Bird'. 'DataFu' is described as LinkedIn's collection of Pig UDFs, which has become an Apache Incubator project, with a link to its website. 'Elephant-Bird' is described as Twitter's library of LZO and/or Protocol Buffer-related Hadoop InputFormats, OutputFormats, Writables, Pig LoadFuncs, HBase miscellanea, etc. The majority of these are in production at Twitter running over data every day. A link to its GitHub repository is provided. Below the descriptions, there is a link to find RPM and Debian packages for Elephant Bird. At the bottom of the page, there is a footer with a license notice and version information.

Apache Pig / Index / User Documentation  
**PigTools**  
Added by Daniel, last edited by Eyal Alkell on Nov 16, 2014 (view change)

**UDF Collections.**

**DataFu**  
DataFu is LinkedIn's collection of Pig UDFs, which has become an Apache Incubator project. (<http://sina-projects.com/datafu/>)

**Elephant-Bird**  
<http://github.com/kevinweir/elephant-bird/>  
Twitter's library of LZO and/or Protocol Buffer-related Hadoop InputFormats, OutputFormats, Writables, Pig LoadFuncs, HBase miscellanea, etc. The majority of these are in production at Twitter running over data every day.  
RPM and Debian packages for Elephant Bird can be found at <http://code.google.com/p/hadoop-gpl-packing/>

Powered by a free Atlassian Confluence Open Source Project License granted to Apache Software Foundation. Evaluate Confluence today.  
This Confluence installation runs a Free Giffy License - Evaluate the Giffy Confluence Plugin for your Wiki!

Confluence 5.0.3, Team Collaboration Software - Report a bug - Atlassian News



# 執行模式

- 本地模式 v.s. MapReduce 模式
- 互動模式 v.s. 批次模式

# MapReduce 模式

- MapReduce mode(Hadoop mode)
  - 需要完整的Hadoop cluster
  - 存取的檔案需要先上傳至HDFS
  - Pig將自動地對個集群進行MapReduce
  - Pig會自動對MapReduce程式進行優化
  - 叢集可以是真實的分散式(fully distributed)，也可以是偽分散式(pseudo distributed)
- 執行方式
  - `pig -x mapreduce`

# 本地模式

- Local mode

- Pig運行在一個JVM裡面

- 存取的是本地的文件系統

- 沒有用到Hadoop的Local runner

- Pig把查詢轉換為物理的Plan然後執行

- 可以在沒有Hadoop cluster的環境下執行在本機執行Pig  
程式常用在 Prototyping, debugging, small data

- 執行方式

- `pig -x local`



# Pig 基本指令

- -e or execute
  - 在pig 執行單一指令
  - 列出目錄清單. e.g. `pig -e fs -ls /`
- -h or -help
  - 列出所有有用的指令選項
- -x [mapreduce | local]
  - 使用特定模式 (local 或 mapreduce)

## Pig 基本指令 (續)

- `-h properties`
  - ▣ 列出pig 使用到的相關屬性
- `-P`
  - ▣ 使用已設置的屬性檔案
- `-version`
  - ▣ 列出pig 的版本

## Pig 基本指令 (續)

- **exec** : 執行 Pig 腳本

**exec**在一個新的Grunt shell中執行script file，所以執行完畢後，任何script file中定義的aliases或者命令都不能存取

```
grunt> cat sample2.pig
records = load '$in'
  as (name: chararray, item:chararray, rating: int);
describe records;
grunt> exec -param in=ratings.tsv sample2.pig;
```

## Pig 基本指令 (續)

- **run** : 執行 Pig 腳本

相當於將script file中的語句逐一複製到目前的Grunt shell中執行，所以跟正常模式一樣

```
grunt> records = load 'ratings.tsv'  
>>   as (name: chararray, item:chararray, rating:  
int);  
grunt> cat sample3.pig  
describe records;  
grunt> run sample3.pig
```



## 其他指令

- `kill jobid`
  - ▣ 指定ID砍掉MapReduce job
- `cat`
  - ▣ 輸出檔案內容到stdout。也可以輸出一個目錄的內容
- `copyFromLocal`
  - ▣ 複製一個本機的檔案到HDFS
- `rmr`
  - ▣ 遞迴地移除檔案，同Linux下的`rm -r`

# 純量型態

型態	敘述	範例
int	Signed 32-bit integer	10
long	Signed 64-bit integer	10L
float	32-bit floating point	10.5F, 6.022e23f
double	64-bit floating point	10.5
chararray	Character array (string) in Unicode UTF-8 format	Hello world
bytearray	Byte array (blob)	
boolean	boolean	true/false (case insensitive)

# 複合型態

## ■ Tuples (<>)

- ▣ 類似資料庫中的列 (row)
- ▣ 可以存放任意類型資料 (Atom, Tuple, Databag)
- ▣ 範例: `t = <5, {<1,2>, <3,4>}, ['name': 'qoo']>`

## ■ Databags

- ▣ Tuple 的組合
- ▣ 可以想像是資料庫中的表格
- ▣ 範例: `{<'localhost', '127.0.0.1'>, <'google', '8.8.8.8'> }`

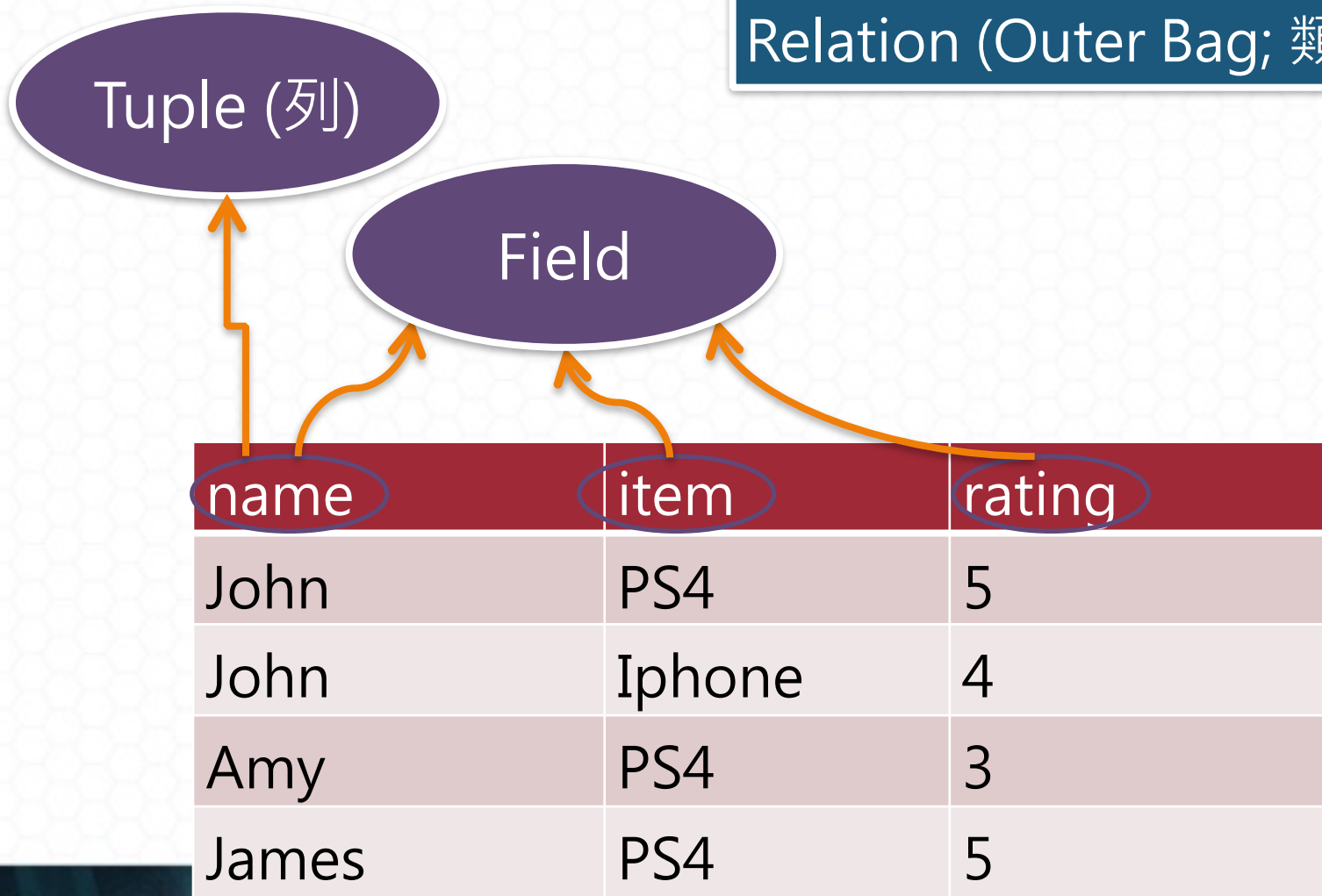
## 複合型態 (續)

- Maps ({})
  - ▣ Key-Value Pair
  - ▣ Value 可以是任一資料類型 (Atom, Tuple, Databag, Map)
  - ▣ 範例 ['name': 'qoo']



# 資料範例

Relation (Outer Bag; 類似表格)



# 資料關聯性

## 資料對應關聯

- **Field** 代表資料; 相當於**SQL**的欄位
- **tuple** 為**Field**的有序集合; 相當於**SQL**的列
- **bag** 為**tuple**的集合
- **Relation** 相當於**bag (Outer Bag)**; 相當於**SQL**的表格

所有**Pig**語句都是作用於**Relation** 之上

# 關於Pig 語法

- Pig Latin 是Data Flow Language
  - 每次操作都會產生一個relation
- Relation
  - 讀進來的資料集
  - 並非變數
  - 避免重複命名
- 大小寫有區別
- Schema-on-read
  - 資料的綱要是讀的時候才決定的，適合非結構化資料

# 標準流程

讀取資料

處理資料

• 存儲中介結果

存儲最終結果

測試與除錯



# 語法範例

- 找出年紀大於等於30的男性顧客姓名

customer

name	gender	age
John	M	30
Amy	F	23
Jane	F	47
James	M	25

## 完整範例

John	M	30
Amy	F	23
Jane	F	47
James	M	25



找出年紀大於等於30  
的男性顧客姓名

```
a = LOAD 'customer.tsv'  
  AS (name:chararray, gender:chararray, age:int);  
b = FILTER a BY (gender == 'M');  
c = FOREACH b GENERATE name;  
-- This is a comment line  
STORE c INTO 'targetlist';
```

## 讀取資料 (LOAD)

- 從檔案系統中讀取資料
- PigStorage 是預設讀取函式, 可以使用自訂函式
- 預設使用 \t 做資料分隔

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

```
a = LOAD 'myfile.txt' USING PigStorage('\t');  
a = LOAD 'myfile.txt' AS (f1:int, f2:int, f3:int);
```

## 存儲資料 (STORE)

- 將資料存儲於檔案系統中
- PigStorage 是預設存儲函式, 可以使用自訂函式
- 假使輸出目錄已存在, 則會輸出失敗
- 輸出檔案名以 part- 做開頭 (預設以\t做資料分隔)

```
STORE 'alias' INTO 'directory' [USING function]
```

```
STORE a INTO 'myOutput' USING PigStorage ('*');
```



# 輸出資料 (DUMP)

- 接結果輸出至畫面上
- 適用於debugging, prototyping或ad-hoc query
- 輸出的欄位會以逗號分隔，Null值會為空(missing value)，
- 當資料列是tuple，輸出會以()表示資料

```
DUMP alias;
```

```
dump a;
```

## 資料過濾 (FILTER)

- 根據條件從relation 中挑選 tuples
- 如果predicate為真，該筆資料就會傳到pipeline
- 運算子
  - ==, !=, >=, <=, >, < 等運算子適用各種資料類型
  - ==和!=則可以用於maps和tuples
- Pig 使用Java 的正規表達法
- NULL 可以用來尋找 null值

```
alias = FILTER alias BY expression;
```

```
x = FILTER a BY (f1 == 8) OR (NOT (f2+f3 > f1));
```

# 資料去重複(DISTINCT)

- 去除relation 中重複的 tuples
- DISTINCT作用於整個records，而無法只針對指定fields，如果需要針對特定欄位，可以先把那些欄位輸出到另外一個relation

```
alias = DISTINCT alias;
```

```
(8,3,4)  
(1,2,3)  
(4,3,3)  
(4,3,3)  
(1,2,3)
```

```
x = DISTINCT a;
```

```
(1,2,3)  
(4,3,3)  
(8,3,4)
```

# 資料遍歷 (FOREACH)

- Foreach能從複合型別中抽取資料
  - Map是#
  - tuple是. (dot)
  - 參考到不存在欄位位置時，會回傳null值
  - 參考到不存在欄位名時，會出現錯誤訊息。
- Foreach中可以定義schema，但foreach可以只針對特定欄位（ Load是針對整個relation ）。

# 資料遍歷 (FOREACH)

- Foreach支援各種表達式，最基本就是常數和欄位（根據名稱或欄位位置）
- 當schema未知(unknown)或未定義(undeclared)時，使用欄位位置是很有用的(適合用在複雜或非結構資料)
- \* (asterisk)可以用來表示所有欄位。
- Foreach也支援binary 條件運算子( ?: )



# 資料遍歷 (FOREACH)

```
alias = FOREACH alias GENERATE expression [AS schema]  
[expression [AS schema]....];
```

```
1 2 3  
4 5 6  
7 8 9
```

```
a = LOAD 'data' AS  
    (a1:int,a2:int,a3:int);  
x = FOREACH a GENERATE a1+a2 AS f1:int,  
    (a3==3? TOTUPLE(1,1):TOTUPLE(0,0));
```

```
DUMP x;  
  
(3,(1,1))  
(9,(0,0))  
(15,(0,0))
```

# 資料合併 (JOIN)

- Join欄位的值如果是空值，inner join會過濾空值
- 可以多個欄位做JOIN，只要在將欄位用逗號分隔
- Pig也支援Self join
  - ▣但是資料就需要被載入兩次並指定給兩個變數
  - ▣如果寫JOIN a by a1, a by a1會出現錯誤
- 在SQL中，JOIN可以用任何條件結合好幾個資料來源，但是Pig只支援equi-joins
- JOIN會保留relation中原本欄位名稱

# 資料合併 (JOIN)

- 以DESCRIBE查看JOIN所產生的relation，可以看到schema

□ 如：relation: {

```
daily::exchange:bytearray, daily::symbol:bytearray,  
dives::exchange:bytearray, dives::date:bytearray, dives::dividends:bytearray}
```

daily

dives

- Pig在實作JOIN時，會逐筆掃左邊relation，再以join key去對應記憶體中右邊relation，最後再輸出
- Pig也有OUTER JOIN, 包括LEFT, RIGHT, 或者FULL

# 資料合併實例

alias = JOIN alias BY field (, alias BY field);

2	4
8	9
1	3
2	7
2	9
4	6
4	9

1	2	3
4	2	1
8	3	4
4	3	3
7	2	5
8	4	3

```
a = LOAD 'data1' AS  
    (a1:int,a2:int);  
b = LOAD 'data2' AS  
    (b1:int,b2:int,b3:int);  
x = JOIN b BY b1, a BY a1;
```

```
DUMP x;  
(1,2,3,1,3)  
(4,2,1,4,6)  
(4,3,3,4,6)  
(4,2,1,4,9)  
(4,3,3,4,9)  
(8,3,4,8,9)  
(8,4,3,8,9)
```

Describe x;

x: {a::a1: int,a::a2: int,a::a3: int,b::b1: int,b::b2: int}

# Group

- SQL的Group需要搭配aggregate function，Pig Latin的Group和aggregation function沒有直接關連
- GROUP需要指定一個key作為分組欄位，依據多個欄位分組時，GROUP BY需要以逗號分隔欄位

```
b = GROUP a BY (age, sex);
```

- 第一個輸出欄位是Group Key 名稱，第二個輸出的欄位名稱會就是relation的名稱，型態是bag



# Group(續)

```
alias = GROUP alias {ALL | BY expression} [,alias ALL | BY expression ...]
```

```
John 18 4.0F  
Mary 19 3.8F  
Bill 20 3.9F  
Joe 18 3.8F
```

```
a = LOAD 'data' AS (name:chararray  
    , age:int, gpa:float);  
b = GROUP a BY age;  
c = FOREACH b GENERATE group, COUNT(a);
```

Describe b;

```
b: {group: int,a: {(name: chararray,age: int,gpa: float)}}
```

Dump b;

```
(18, {(John,18,4.0F), (Joe,18,3.8F)})  
(19, {(Mary,19,3.8F)})  
(20, {(Bill,20,3.9F)})
```

Dump c;

```
(18,2L)  
(19,1L)  
(20,1L)
```

# ORDER BY

- ORDER需要指定一個key作為排序的欄位
- 除了ORDER，Pig其他的運算子不保證回傳的records的順序
- 數字欄位都會依據數字大小排序
  - ▣ chararray和bytearray則依據字元排序
  - ▣ null永遠被當成最小的值

```
alias = ORDER alias BY {*[ASC|DESC]} | field[ASC|DESC]  
[,field[ASC|DESC] ...]
```

```
x = ORDER a BY a3 DESC;
```

# LIMIT

- LIMIT需要先把所有records集合在一起，才在reducer過程把所有資料讀完後，再取得指定的筆數
- 不論LIMIT 指定10筆或10000筆，Pig都會把整個records讀完

alias = LIMIT alias n

```
1 2 3  
4 5 6  
7 8 9
```

```
a = LOAD 'data1' AS (a1:int,a2:int,a3:int);  
x = LIMIT a 2;
```

```
DUMP x;  
(1,2,3)  
(7,8,9)
```

# UNION

- 不同於SQL的UNION而像是UNION ALL，Pig的UNION並不會把重複的資料過濾掉。

```
UNION alias1, alias2 [, alias...];
```

```
a = LOAD 'data1' AS (f1:int, f2:int  
    , f3:int);  
b = LOAD 'data1' AS (f1:int, f2:int  
    , f3:int);  
c = UNION a, b;
```

```
1 2 3  
4 5 6  
7 8 9
```

```
DUMP a;  
(1,2,3)  
(4,5,6)  
(7,8,9)
```

```
DUMP b;  
(1,2,3)  
(4,5,6)  
(7,8,9)
```

```
DUMP c;  
(1,2,3)  
(4,5,6)  
(7,8,9)  
(1,2,3)  
(4,5,6)  
(7,8,9)
```

# UNION (續)

- 如果union欄位的資料類型不同，有時Pig可以自行做隱含轉換(implicit conversion)
  - ▣ 若Pig無法轉換，則schema會變成UNKNOWN
- 可以加上關鍵字onschema強制執行轉換
  - ▣ 所有的input都必須有schema
  - ▣ 所有欄位會根據名稱而不是次序做對應



# SPLIT

- 將單一Relation 分割成多個Relations
  - 一個Tuple 可能會被分派到多個Relations
  - 一個Tuple 可能不會被分派到任一Relation

SPLIT alias INTO alias IF expression, alias IF expression [, alias IF expression ...] [, alias OTHERWISE];

1	2	3
4	5	6
7	8	9

```
a = LOAD 'data1' AS (f1:int, f2:int, f3:int);  
SPLIT a INTO x IF f1==7, y IF (f1 < 5  
    AND f2 < 6), z OTHERWISE;
```

DUMP x;  
(7,8,9)

DUMP y;  
(1,2,3)  
(4,5,6)

DUMP z;

# COUNT

- 計算bag中有多少元素
  - ▣ 需要搭配GROUP
  - ▣ 如果要考慮NULL，需使用 COUNT\_STAR

```
1 2 3
8 3 4
7 2 5
8 4 3
1 1
1 1
```

```
a = LOAD 'data' AS (f1:int, f2:int, f3:int);
b = GROUP a BY f1;
x1 = FOREACH b GENERATE COUNT(a);
x2 = FOREACH b GENERATE COUNT_STAR(a);
```

```
DUMP x1;
(1,1)
(7,1)
(8,2)
(,0)
```

```
DUMP x2;
(1,1)
(7,1)
(8,2)
(,2)
```

# SUM

- 計算bag中值的總和
  - ▣ 需要搭配GROUP

```
Alice,turtle,1  
Alice,goldfish,5  
Alice,cat,2  
Bob,dog,2  
Bob,cat,2
```

```
a = LOAD 'data' USING PigStorage( ',' ) AS  
  (owner:chararray,pet_type:chararray  
   ,pet_count:int);  
b = GROUP a BY owner;  
x = FOREACH b GENERATE group,SUM(a.pet_count);
```

```
DUMP x;  
(Alice,8)  
(Bob,4)
```

# 使用者自定函數

## ■ 演進

- ▣ 0.7版：增加UDF只要利用Java class開發JAR檔指定給Pig
- ▣ 0.8版起：Python也可以開發UDF。Pig會使用Jython來執行Python所開發的UDF
- ▣ 0.8版後，有些新功能也利用UDF來增加到Pig中
- PiggyBank是一個collection，把很多user-contributed UDF和Pig一起包裝並釋出
- PiggyBank UDF並不包含在Pig JAR中，因此你必須手動註冊它

## 非內建於Pig的UDF

- 必須先經過註冊，使用register指令，讓Pig知道從哪裡可以找到這個UDF
- Pig會開啟所有的註冊JAR檔，取出其中所有檔案，並且把他們上傳到Hadoop
- 0.8.0之後的版本，REGISTER指令也可以註冊HDFS上的JAR檔
- DEFINE提供了alias的作用，而可以不用每次使用完整package名稱



## Lab: 分析NCDC 的GSOD 資料

- NCDC是世界最大的動態天氣資料中心，它可以向全世界提供各種氣候公報及相關資料
- GSOD (Global Summary of the Day)
  - ▣ 超過9,000天氣觀測站回報的天氣資料
- 步驟請參閱pig-exercise.txt

# 資料說明

2014.zip 中的\*.op

欄位	說明
station	觀測站ID
wban	海軍觀測站ID
ymd	年月日
temp_avg	平均氣溫
temp_max	最高氣溫
temp_min	最低氣溫
windspeed_avg	平均風速
windspeed_max	最高風速

ISH-HISTORY.tsv

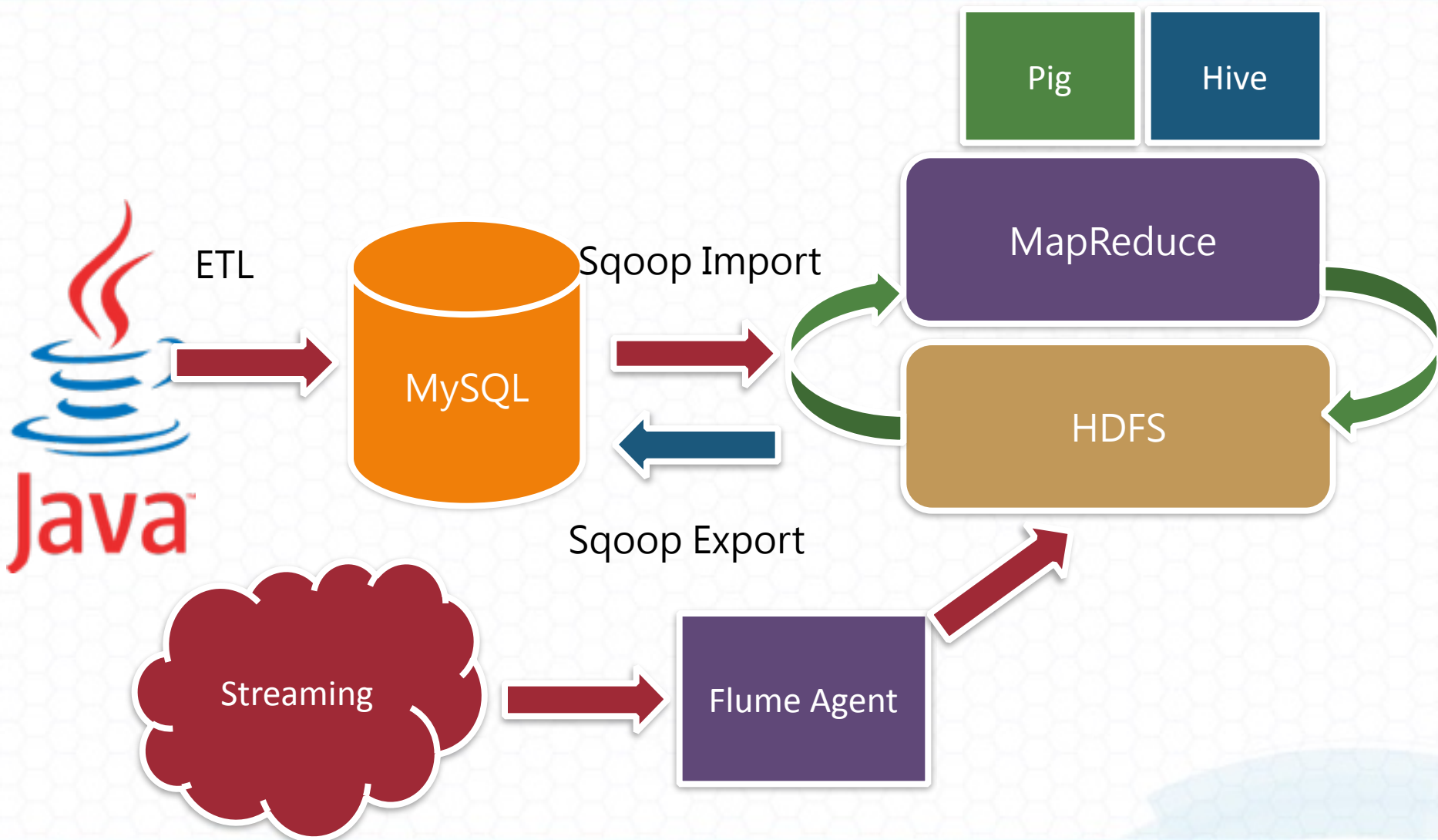
欄位	說明
station	觀測站ID
wban	海軍觀測站ID
station_name	觀測站名稱
country	國家
state	州



# 資料分析模組

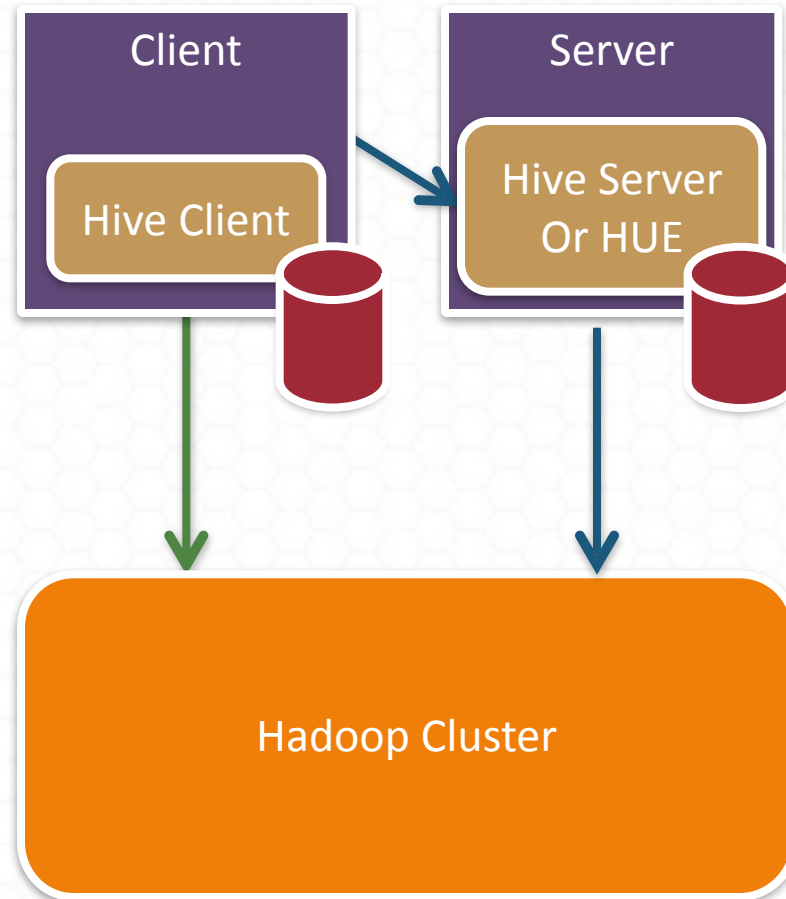
## **HIVE**介紹

# 分析情境



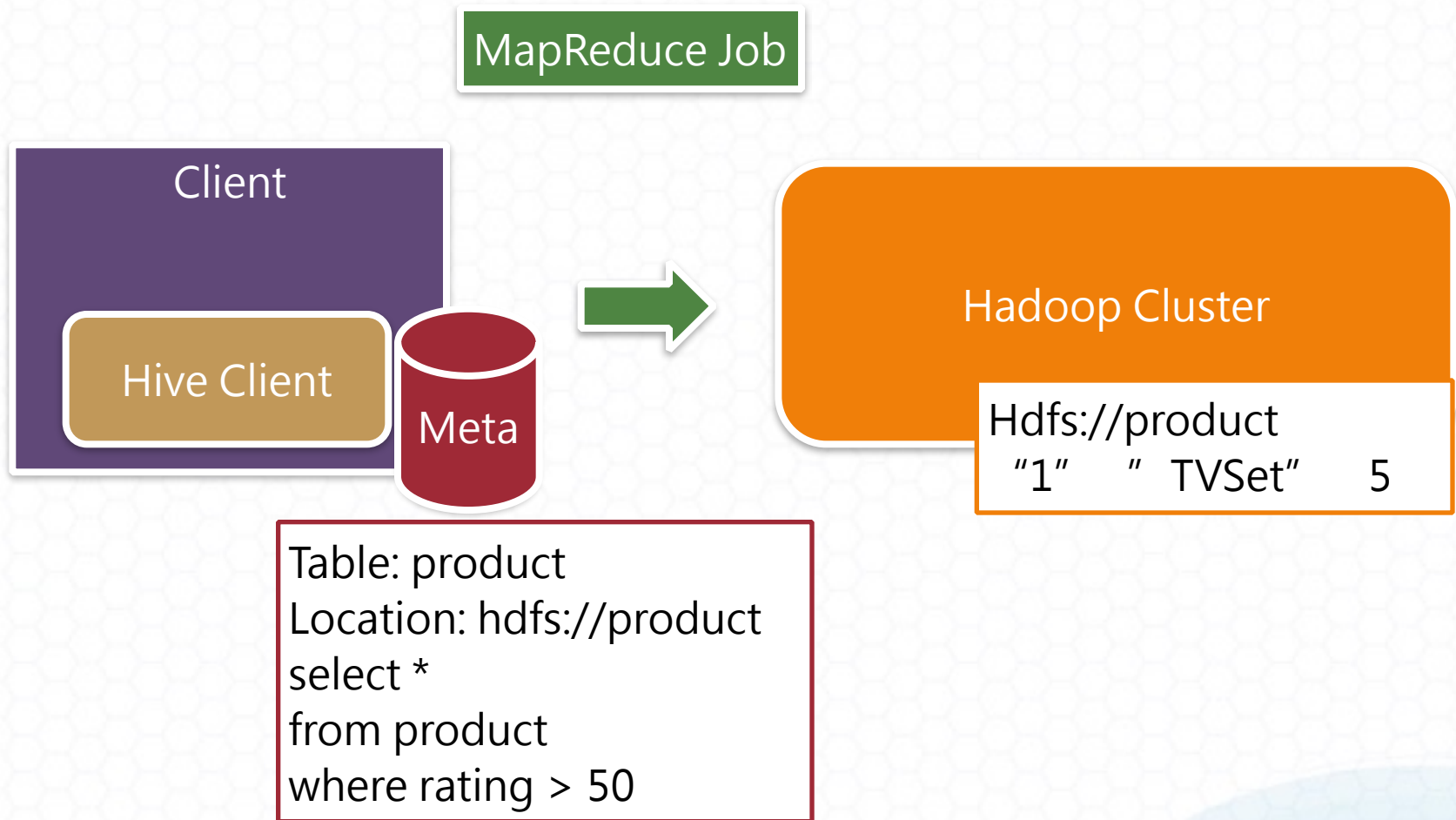
# Hive 架構

- Facebook 所開發
- SQL Like 語言
- Hive 可以以Client 的形式安裝
- 或是可以使用網路服務存取
  - ▣ Hue (beeswax, web interface)
  - ▣ HiveServer2 (ODBC, JDBC)
- 需要使用metastore
  - ▣ 提供Hive 對HDFS 資料的映對





# Metastore



# Hive 如何工作

- Hive會跟Pre-built 的Binaries 會轉換成Jar檔，並透過Map Reduce 的工作執行Query
- 不會產生Java 原始碼

# Hive 的優點

- 簡潔方便 (相較於Java 而言)
- Ad-Hoc 分析 (比Impala 慢)
- 可以依時間分割資料 (Partition)
- DBA 可以重複使用部分Query (同MySQL 語法)
- 使用共用的View節省表格建置的時間成本
  - ▣ Hive 專注在資料表的建立與寫入
  - ▣ 共用的View 可以加速分析

# Hive 的缺點

- 無法即時性分析(使用Impala)
- 非高效能 (使用Impala)
- 如需要使用資料ETL (使用PIG)
- 如需要精細控制流程(IF ...ELSE) (使用 PIG)
- 難以處理非結構化(沒有明確的Schema)資料 (使用PIG)

# HiveQL

- Hive 是建立於Hadoop 之上的資料倉儲套件
- 提供SQL Like 的語法 Hive Query Language (HiveQL)，使用者可以使用HiveQL 以MapReduce存取HDFS內的資料
- 語法接近MySQL



# 與傳統資料庫比較

- Hive 不提供
  - row-level insert
  - updates
  - deletes
  - Transactions
- 並非資料庫
- Hive 並非用作線上交易
- Hive 並不提供即時查詢 (可使用Impala 替代)

# Hive & Pig

- MapReduce 的介面
  - ▣ 讓非開發者也能輕鬆操作Hadoop
  - ▣ Hive – SQL Like Language (HiveQL)
  - ▣ Pig – Data Flow Language
- 比Native Java Code 運行緩慢
  - ▣ 比原生Java平均大約慢10~15%
  - ▣ 讓開發者比較有生產力

# 使用情境

## ■ Hive

- 當有明確的Schema
- 當資料適合使用SQL執行時

## ■ Pig

- 可以執行複雜的ETL
- Pig 提供較Hive 更多控制

## ■ Impala

- 有明確的Schema
- 即時性互動查詢
- 低容錯性的即時分析

# 資料分析時該怎麼選擇？

- Hive, Pig, Impala 三種工具可以搭配使用
- Pig 跟Hive 可以透過Oozie管理，執行需可容錯性高的MapReduce 工作

# 使用Hive

- 使用Hive

- \$ hive

- 於Hive 指令列中建立測試表格

- hive> create table test(id string)

- 切換到MySQL Shell，檢視MySQL 內的表格

- Use hive\_metadata;

- show tables;



# HiveQL v.s. SQL

功能	SQL	HiveQL
更新資料 (Update)	<ul style="list-style-type: none"><li>• UPDATE, INSERT,</li></ul>	<ul style="list-style-type: none"><li>• INSERT</li></ul>
資料查詢 (Select)	<ul style="list-style-type: none"><li>• 支援SQL92語 法</li></ul>	<ul style="list-style-type: none"><li>• 可以針對單一表格或視 界查詢</li><li>• SORT BY 可對資料做部 分排序</li><li>• LIMIT 可限制回傳筆數</li><li>• 不支援Having</li></ul>
資料合併(Join)	<ul style="list-style-type: none"><li>• 支援</li></ul>	<ul style="list-style-type: none"><li>• 支援 inner, outer, map, semi join</li></ul>

# HiveQL v.s. SQL

功能	SQL	HiveQL
子查詢 (Subquery)	<ul style="list-style-type: none"><li>可出現在SQL任意處</li></ul>	<ul style="list-style-type: none"><li>只能出現在FROM處</li></ul>
視界 (View)	<ul style="list-style-type: none"><li>可以修改</li></ul>	<ul style="list-style-type: none"><li>唯讀，不能修改</li></ul>
擴充元件 (Extension)	<ul style="list-style-type: none"><li>使用者自訂函式</li></ul>	<ul style="list-style-type: none"><li>使用者自訂函式</li><li>或使用 MapReduce 腳本</li></ul>

## 建立資料庫

```
CREATE DATABASE test_db      # 建立資料庫
COMMENT "This is a test db" # 增加註解
LOCATION "/hivetest/"         # 設定HDFS路徑
WITH DBPROPERTIES ("creator" = "hadoop", "date" =
"2014-12-28");              # 設定屬性
```

# 檢視資料庫

- 表列所有資料庫

- > SHOW DATABASES;

- 描述資料庫

- > DESCRIBE DATABASE test\_db;

- > DESCRIBE DATABASE EXTENDED test\_db;

# 修改與捨棄資料庫

- 修改資料庫

- > ALTER DATABASE test\_db SET DBPROPERTIES ('edited-by' = 'hadoop');

- 捨棄資料庫

- ❑ DROP DATABASE test\_db;



# 建立內部資料表

## 建立內部(Internal) 表格: ratings

```
create table ratings(userid INT,itemid INT, rating INT)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

## 將資料從本地端ratings.tsv 讀入ratings 表格

```
LOAD DATA LOCAL INPATH 'ratings.tsv' OVERWRITE INTO TABLE  
ratings;
```

# 建立外部資料表

## 建立外部(External)表格: ratings

使用external 代表外部資料表

```
create external table items(itemid INT, category STRING)ROW  
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/  
yelp'
```

## 將資料從本地端items.tsv 放置於HDFS

```
hadoop fs -mkdir /yelp  
hadoop fs -put items.tsv /yelp
```

# 資料查詢

- 查詢items

```
DESC items;  
DESC EXTENDED items;  
select * from items;
```

- 合併資料

```
select * from ratings s join items t  
on s.itemid=t.itemid limit 30;
```

運行MapReduce  
要花上40 秒左右運行

- 本地端設置

```
SET hive.exec.mode.local.auto=true;
```

# 捨棄資料表

- 捨棄外部資料表items

```
drop table items;  
hadoop fs -ls /yelp
```

- 捨棄內部資料表ratings

```
drop table ratings;  
hadoop fs -ls /user/hive/warehouse/
```

# Hive 內部資料表與外部資料表

- Hive 內部資料表 (Internal)

- ▣ 存在於HDFS 中的/user/hive/warehouse
- ▣ 預設只有Hive 可以存取該部分資料
- ▣ Drop Table 指令可以移除metastore 跟 HDFS 資料

- 外部資料表 (External)

- ▣ 可以使用/user/hive/warehouse 以外的HDFS 資料
- ▣ Drop Table 只有移除metastore



# Partition & Bucket

- Hive 可以使用分區(partitions) 加速查詢
  - ▣ Partition 是以目錄的方式存在於主表格中
  - ▣ 可以使用日期做分割(table/2014-10, table/2014-11)
  - ▣ 每次只會存取符合查詢條件的分割, 可加速資料查詢
- Bucket 可以做資料取樣
  - ▣ 建立 Bucket 會將資料做依鍵值切割
  - ▣ 每個Bucket 包含隨機取樣資料
  - ▣ 可以使用Bucket 去取樣部分資料

# 分區表格(Partition)

## ■ 建立分區表格及讀取資料

```
create table top_ratings (userid INT, itemid INT) partitioned by(rating INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' ;
LOAD DATA LOCAL INPATH 'top_ratings.tsv' overwrite INTO TABLE
top_ratings partition (rating = 5) ;
LOAD DATA LOCAL INPATH 'second_ratings.tsv' overwrite INTO TABLE
top_ratings partition (rating = 4) ;
```

## ■ 分區查詢資料

```
select * from top_ratings where rating = 5;
```

# 分桶表格(Bucket)

- 建立分桶表格及讀取資料

```
CREATE TABLE bucket_ratings (userid int, itemid int, rating int)  
CLUSTERED BY (rating) sorted by (rating asc) into 5 buckets  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

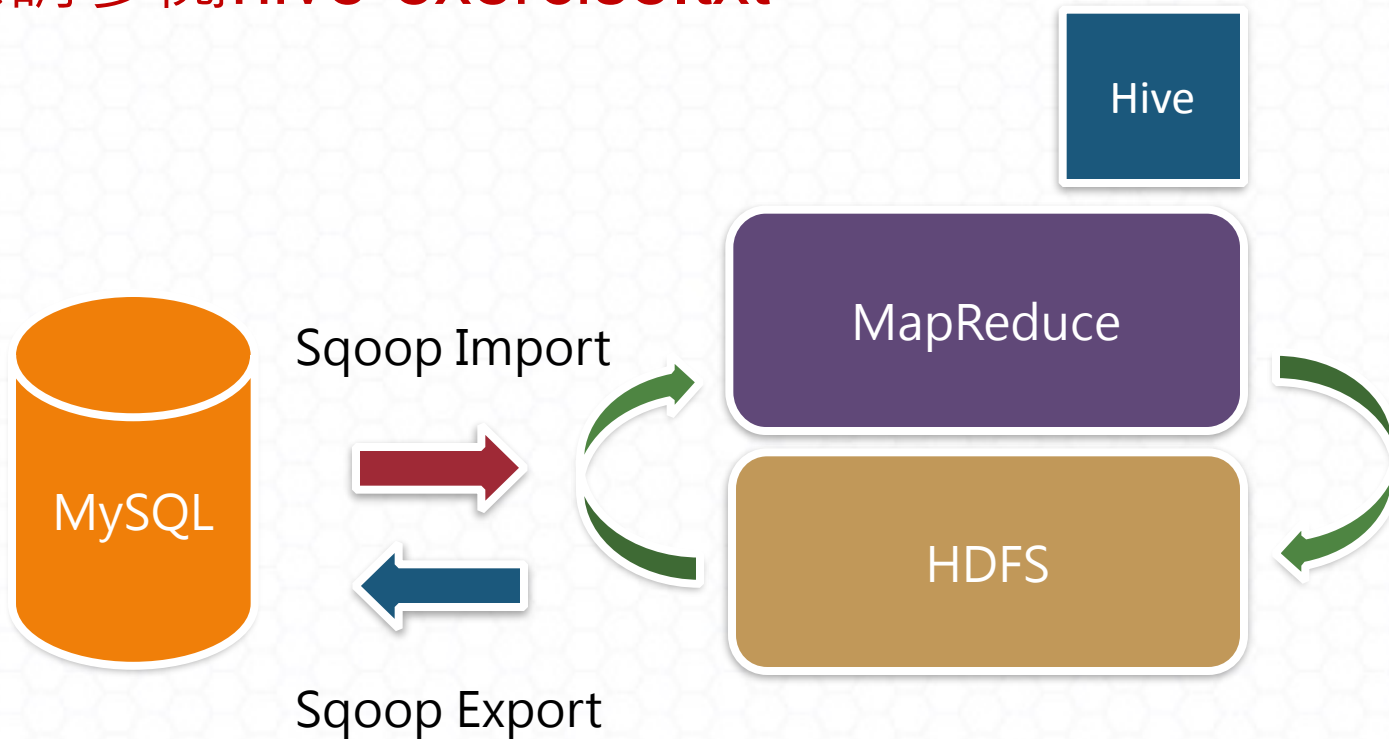
```
LOAD DATA LOCAL INPATH 'ratings.tsv' INTO TABLE  
bucket_ratings;
```

- 分桶查詢資料

```
select * from bucket_ratings tablesample(bucket 1 out of 5);
```

# Lab: Hive 實戰

- 步驟請參閱hive-exercise.txt





# 資料分析模組 **Impala**介紹



# Pig & Hive & Impala



# Hive & Pig & Impala

- MapReduce 的介面
  - 讓非開發者也能輕鬆操作Hadoop
  - Hive – SQL Like Language (HiveQL)
  - Pig – Data Flow Language
- 比Native Java Code 運行緩慢
  - 比原生Java平均大約慢10~15%
  - 讓開發者比較有生產力

# Impala

- 沿用HiveQL 的介面
- Low-Latency 互動式查詢
- 支援於CDH 4.X 以上 (為Cloudera 所開發)



# Impala 架構

- 完全不使用MapReduce
  - ▣ 非Fault-Tolerant – 只要一個節點產生錯誤，所有的工作會失敗
- 使用HDFS資料
- 需要使用Hive 的Metastore
- Impalad在Slave Node 上運行
- Impala statestore 會在Cluster 中其中一台運行 (通常跟Metastore 放在一起)
  - ▣ 將metadata, 及最佳化路徑(哪邊可能有資料)送給Impalad, 加速運行
- Impala 服務接受Impala 查詢指令，並會偕同其他Impala 執行工作
  - ▣ 並不需要statestore, 但沒有statestore 速度會較慢

# Impala 使用情境

- 把Impala 運用在
  - 即時性分析 (ad-hoc query)
  - 資料探索
  - 當MR 執行太慢
- 不要在Impala 運用在
  - 需要容錯性高的工作
  - 資料 ETL (尤其是使用到自訂函式者)
  - 讓使用者 (End-User) 即時透過Impala 查詢資料
    - 應該使用Sqoop 將HDFS資料回傳回資料庫，透過資料庫做即時性服務



# Impala

- 在不同機器上有運行不同服務
  - ▣ Impalad: 在每個Slave Node 上面運行
  - ▣ Statestored: 在單台機器上運行, 每個Cluster 只會有一個
- Cloudera 所開發
- Open Source
- **Non-Fault Tolerant**

## 與Pig / Hive 比較

- Hive 與 Pig 使用 MapReduce
- Impala 使用自己的運行服務 (daemon)
- Impala 使用Hive 的metastore
- Impala 沿用HiveQL 的查詢語句
  - Imapala 不支援自訂部分函式
- 使用Impala 時，Slave Node 需要有足夠的記憶體
- Imapala 大量使用記憶體處理資料

# Hive v.s. Pig v.s. Impala

Tool	Run Model	Fault Tolerant	Run Speed	Ease of use	Robustness	Learning Curve
Hive	MapReduce	Y	Slow	Easy	SQL-like (limited)	minimal
Pig	MapReduce	Y	Slow	Harder	PigLatin (very)	weeks
Impala	Real time	N	Fast	Easy	SQL-like (most limited)	minimal

# 使用情境

## ■ Hive

- 當有明確的Schema
- 當資料適合使用SQL執行時

## ■ Pig

- 可以執行複雜的ETL
- 可以執行複雜的資料操作(Pig 提供較Hive 更多控制)

## ■ Impala

- 有明確的Schema
- 即時性互動查詢
- 低容錯性的即時分析

# 資料分析時該怎麼選擇？

- Hive, Pig, Impala 三種工具可以搭配使用
- Pig 跟Hive 可以透過Oozie管理，執行需可容錯性高的ETL 工作



# 推薦系統

# 推薦結果

- 看此商品的人也看了？
- 買此商品的人也買了？
- 通常一起購買的商品？

The screenshot displays an Amazon product page for a car electric blanket. The main product is the "RoadPro RP-208EC 12V 12' Extension Cord with Cigarette Lighter Plug". The page features two recommendation sections:

- Frequently Bought Together:** This section shows the main product bundle for \$36.49, along with a "Heated Fleece Travel Electric Blanket" for \$24.99 and a "RoadPro RP-208EC 12V 12' Extension Cord with Cigarette Lighter Plug" for \$8.82.
- Customers Who Bought This Item Also Bought:** This section displays a carousel of related products, including various car chargers and blankets, with prices ranging from \$19.90 to \$28.90.

On the left side of the image, there is a vertical list of product recommendations in Chinese, including items like "eForCity 221132 AC/DC Power Converter Charger" and "Heated Fleece Travel Electric Blanket".

# 利用使用者偏好產生推薦結果？

- 傳統使用Content-based, 沒法提供互補或是偏好推薦結果
- 是否能藉由其他和你相似的使用者的偏好，去預測你的個人偏好，進而達到個人化的推薦效果？

# 協同過濾法(Collaborative Filtering)

- Minnesota大學的GroupLens研究團隊於SIGIR 發表論文:An Algorithm Framework for Performing Collaborative Filtering

## An Algorithmic Framework for Performing Collaborative Filtering

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl  
(herlocker, konstan, borchers, riedl)@cs.umn.edu  
Dept. of Computer Science and Engineering  
University of Minnesota  
www.cs.umn.edu/research/groupLens/

**Abstract** Automated collaborative filtering is quickly becoming a popular technique for reducing information overload, often as a technique to complement content-based information filtering systems. In this paper we present an algorithmic framework for performing collaborative filtering and new algorithmic designs that increase the accuracy of collaborative prediction algorithms. We then present a set of recommendations on selection of the right collaborative filtering algorithmic components.

### 1 Introduction

Automated collaborative filtering is quickly becoming a popular technique for reducing information overload, often as a technique to complement content-based information filtering systems. Automated collaborative filtering has seen considerable success on the Internet, being used at sites like Amazon.com—the largest book store on the Internet, CDNow.com—the largest CD store on the Web, and MovieFinder.com—one of the most visited movie sites on the Internet.

Content-based and collaborative filtering use different types of data to arrive at a filtering decision. Content-filtering tools select the right information for the right people by comparing representations of content contained in the docu-

ment's personalized recommendations for interesting items. Collaborative filtering provides three key additional advantages to information filtering that are not provided by content-based filtering: (i) support for filtering items whose content is not easily analyzed by automated processes; (ii) the ability to filter items based on quality and taste; and (iii) the ability to provide serendipitous recommendations.

First of all, in collaborative filtering, humans determine the relevance, quality, and interest of an item in the information stream. As a result, filtering can be performed on items that are hard to analyze with computers, such as movies, ideas, fads, people, and politicians.

Second, collaborative filtering systems can enhance information filtering systems by measuring, in dimensions beyond that of simple content, how well an item meets a user's need or interests. Humans are capable of analyzing on dimensions such as quality or taste, which are very hard for computer processes. A content-based search of the Associated Press could retrieve all articles related to Minnesota Governor Jesse Ventura, but by combining content filtering with collaborative filtering a search could return only those relevant articles that are well-written.

Finally, a collaborative filtering system will sometimes make serendipitous recommendations—recommending items that



# 推薦演算法 (使用者為基礎)





# 使用者為基礎的協同過濾演算法

1. 對所有使用者產生相似度權重 ( similarity weighting )
2. 取出相似的子集合(subset)當做用於預測的鄰居 ( neighborhoods )
3. 計算預測值和正規化
  - 產生Top-K 推薦結果
  - 效率較差

## 何謂使用者偏好？

- 「Customers Who Bought This Item Also Bought (買了這項商品的顧客，也會買這些商品)」
- Joe和James在電影的評分類似（可預測的鄰居），可嘗試藉由和James相似的Joe推算出問號的評分）

	驚奇四超人	驚聲尖笑	星際大戰	復仇者聯盟
<b>Joe</b>	<b>5</b>	<b>2</b>	<b>5</b>	<b>4</b>
<b>Amy</b>	<b>2</b>	<b>5</b>		<b>3</b>
<b>Annie</b>	<b>2</b>	<b>2</b>	<b>4</b>	<b>2</b>
<b>James</b>	<b>5</b>	<b>1</b>	<b>5</b>	<b>?</b>


























# 商品為基礎的協同過濾演算法

1. 對所有商品產生相似度權重 ( similarity weighting )
2. 取出相似的子集合(subset)當做用於預測的鄰居 ( neighborhoods )
3. 計算預測值和正規化
  - 可以離線計算 (商品已知)
  - 準度比較低，但運算速度比較快

# ALS 演算法

- Alternating Least Squares with Weighted-Lambda-Regularization (ALS-WR).
  - ▣ 為矩陣分解演算法(Matrix factorization algorithm)
  - ▣ Large-scale Parallel Collaborative Filtering for the Netflix Prize
  - ▣ Collaborative Filtering for Implicit Feedback Datasets
- 尋找隱含因子以解釋使用者對商品的評價，希望可以找到一個最佳化的因子可以降低實際與預測評價的差異

# 利用使用者相似度預測?內容

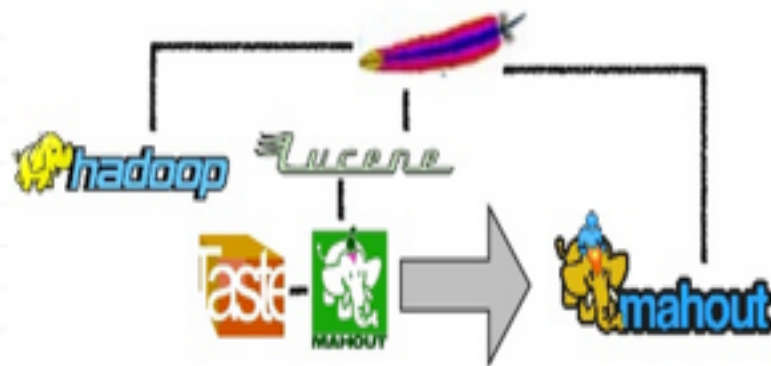
				
				
				
				
				
				



# Mahout

# Mahout 簡介

- 原為Apache Lucene 的子計畫
  - ▣ 在2010年成為Apache top-level的專案
  - ▣ 同時納入協同過濾式推薦演算法專案，Taste中
- 可以使用MapReduce 進行機器學習
- 使用Java 撰寫
- 現行版本 0.9
  - ▣ <http://mahout.apache.org/>



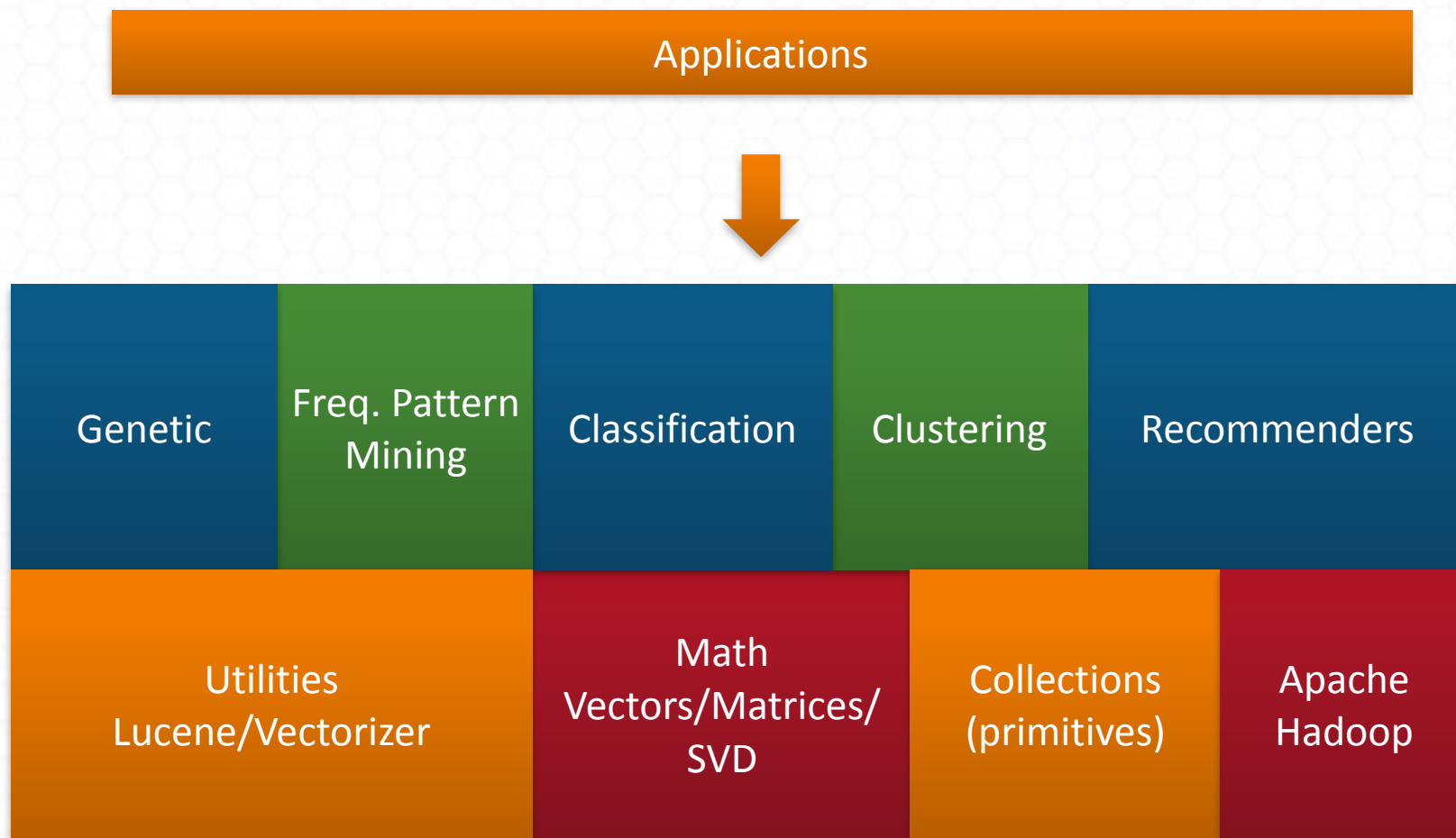
# Mahout 簡介

- 注重在可延展性
  - 內建的演算法都可以透過MapReduce 進行分散運算
- 簡單易用
  - 可透過指令或寫成Java 程式進行機器學習
- 提供多種常見機器學習演算法
  - 推薦系統
  - 資料分類
  - 資料分群

# 不同資料級的處理方式

量級	分析種類	工具
數行；資料樣本	分析與視覺化	Whiteboard, Bash, ...
KB級~ 低 MB級	分析與視覺化	Matlab, Octave, R, Processing, Bash, ...
MB級 ~ 低 GB級	存儲	MySQL (DBs), ...
	分析	NumPy, SciPy, Pandas, Weka..
	視覺化	PyGal, AmCharts, HiChart, d3.js
GB級 ~ TB級 或 PB級	存儲	HDFS, Hbase, Cassandra,...
	分析	RHadoop, Pig, Hive, Mahout, storm, spark
大數據		

# 支援演算法





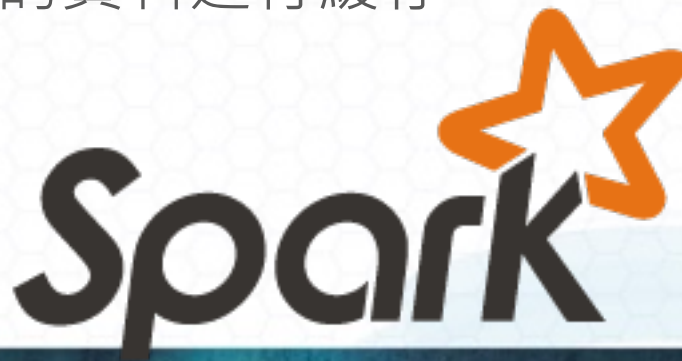
# Spark 與 MLlib

# Mahout 與MapReduce

- 機器學習需要大量迭代(Iteration)運算
  - 每次進行MapReduce 都需要數分鐘時間
  - 無法進行大量迭代(時間考量)，導致精準度下降
  - MR不會記錄之前的中介結果(資料無法重新使用)
- Mahout 已停止支援 MapReduce
  - 請見**25 April 2014 - Goodbye MapReduce**
  - 更改支援Spark

# Spark

- 由DataBricks (前身為AMLab )建造
- 可以在YARN 的架構下運行
- 特色
  - ▣ 適合(Iterative)計算
  - ▣ 透過RDD 讓使用者對訪問頻繁的資料進行緩存
  - ▣ 適合用作Machine Learning



# MLLib

- Spark 的機器學習套件
  - ▣ 內建於Spark
- 比Mahout 在MR 上快上40 ~ 200 倍
  - ▣ Mahout 也開始支援Spark
- 支援多數Machine Learning演算法
  - ▣ <https://spark.apache.org/mllib/>

# Mahout 與推薦系統



# 實作步驟

1. 根據csv 檔案建立推薦模型(Model)
2. 建立簡單的推薦系統
3. 為每位使用者產生推薦結果

步驟請參閱mahout-exercise.txt

# Grouplens

grouplens [about](#) [datasets](#) [publications](#) [blog](#)

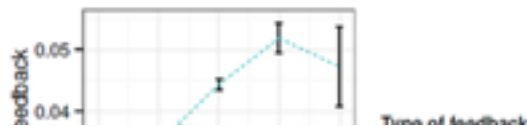
## Social Computing Research at the University of Minnesota

Recommender systems, online communities, human-computer interaction

### Featured Research

We publish research articles in conferences and journals primarily in the field of computer science, but also in other fields including psychology, sociology, and medicine. See our [publications](#) page for a full list; see below for excerpts from recent articles.

#### Deviant Behavior in League of Legends



#### Social Curation in Pinterest



# 如何判斷推薦效果好壞？

1. **Coverage** 係指可以提供預測的項目比率，值越高越好，但通常必須跟**MAE**做取捨
2. **MAE ( Mean Absolute Error )** 預測值和實際值的差異絕對值平均，值越低越好
3. **ROC**曲線 越往左上方是越好的情況，值越高越好



# Hbase

# 什麼是Hbase

- 分散式
- 多維度
- 高效能
- 存儲系統
- High- Availability
- Column- Oriented

存儲大量的行、列及多版本資料，並能分散到數以萬計的機器中





# Hbase 與RDBMS 的差異

- 並非關聯式資料庫
  - 沒有Join, Query engine, 類型, SQL
  - 有支援Transaction 跟 Secondary Index – 發展中
- 並非傳統資料庫的替代品
- 沒有Schema (Anti-Schema)的概念
  - 資料是非正規化的
  - 超大型Excel

# Hbase 架構

- 表格是由Regions 所組成
- Region 是由startKey 與endKey 所定義的
  - 空白表格:
    - Table, NULL, NULL
  - Two-region table
    - (Table, NULL, “streamy”) and (Table, “Streamy”, NULL)
- 每個Region是由多個HDFS 檔案與Block所組成，同時存在於多台node 上

# Hbase 架構 (續)

- 兩種Hbase Node
  - Master & RegionServer
- Meta資訊存儲表格 -ROOT- 和 .META.
  - 存儲schema 資訊以及Regions 的位置
- Master 扮演監控(Monitor) Region Server 的角色  
並會把工作量分配到regions 上

# Hbase 架構

- HMaster**
- Responsible for Admin Operations
  - Manages and Monitors the Cluster
  - Assigns Regions to Region Servers
  - Controls Load Balancing and Failover

## HRegionServer

### HRegion

HLog

Store  
MemStore  
StoreFile

## HRegionServer

### HRegion

HLog

Store  
MemStore  
StoreFile

## HRegionServer

### HRegion

HLog

Store  
MemStore  
StoreFile

## HRegionServer

### HRegion

HLog

Store  
MemStore  
StoreFile

# Hbase 表格


- 所以表格是依據Row 做排序
- 表格的Schema 會定義Column Family
  - 每個Family 都會有任意數量的欄位
  - 每個欄位都會有不同版本
  - 欄位只有在塞資料的時候才存在
  - 同個Family 的欄位會同時被排序以及存儲
  - 所有的欄位與定義都以byte[]存儲
- (Table, Row, Family:Column, Timestamp) -> value



# Hbase 範例

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	\$400.00
102	Jane Brown	Atlanta, GA	Lamps	\$200.00
103	Bill Green	Pittsburgh, PA	Desk	\$500.00
104	Jack Black	St. Louis, MO	Bed	\$1600.00

Column Families

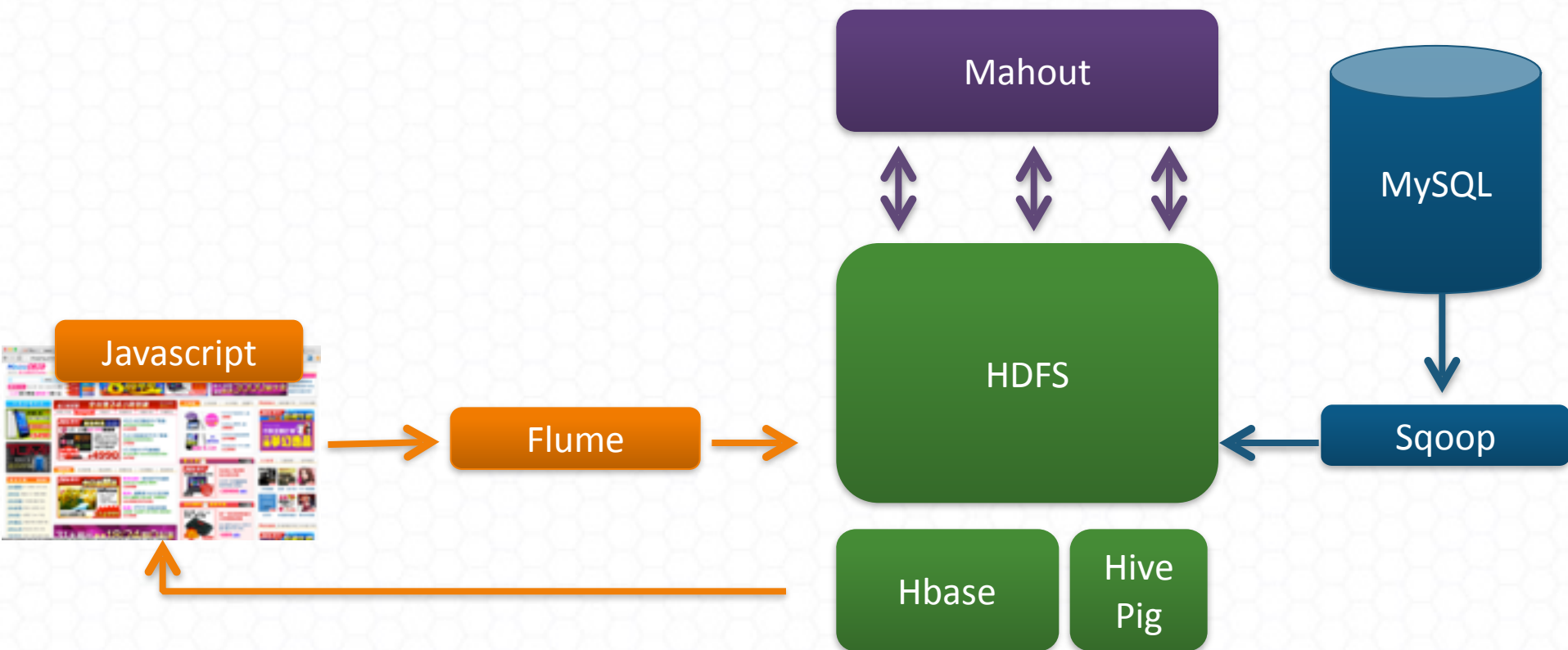


# 透過Python 操作Hbase

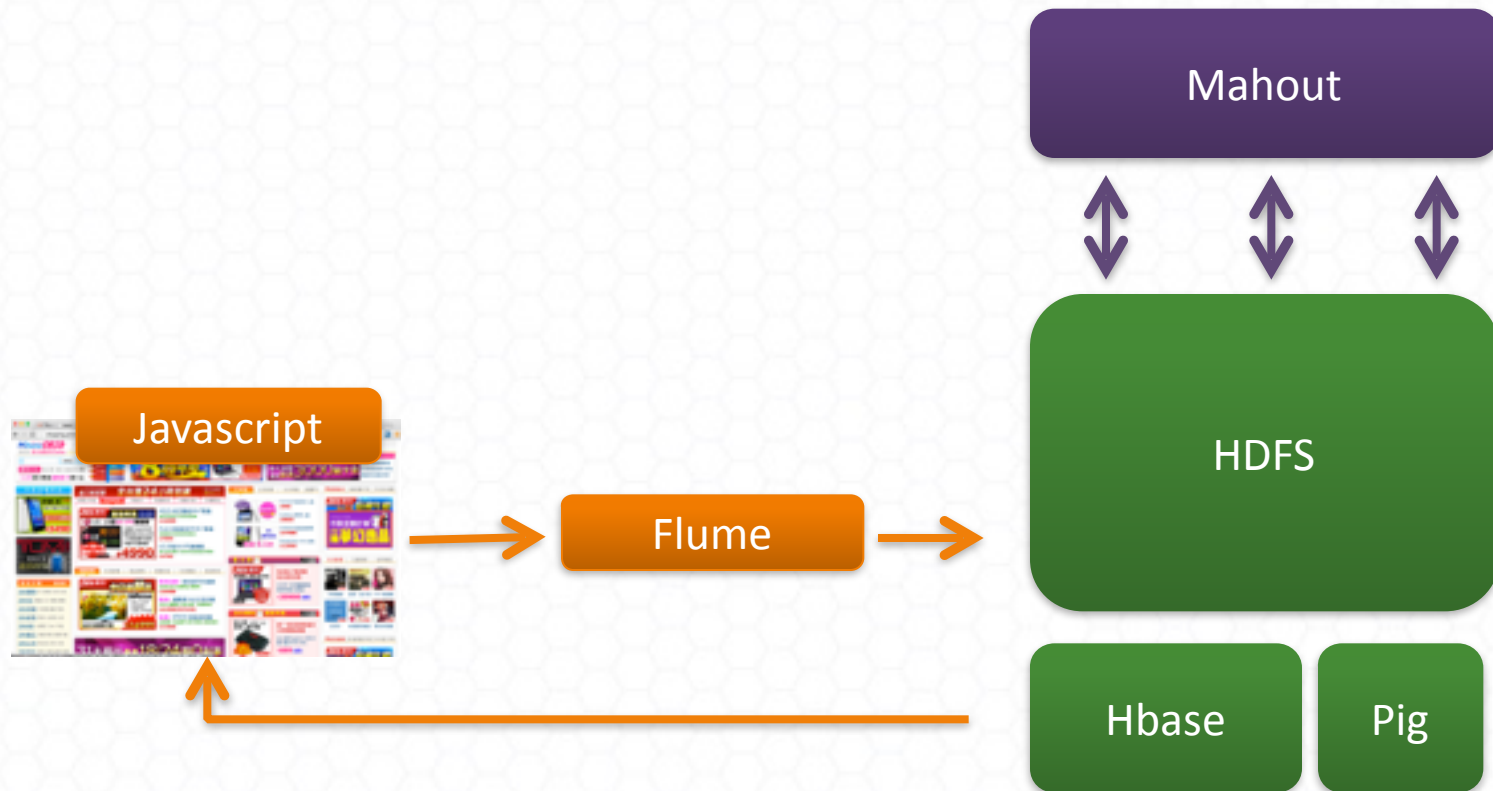
- 安裝HappyBase
  - `pip install happybase`
- HappyBase 可以讓使用者透過Python 操作 Apache Hbase.
- HappyBase 使用 Thrift連結Hbase,
- Hbase 0.9x 被正式納入
- 步驟請參閱[hbase-exercise.txt](#)

# 推薦系統實作

# 完整版推薦系統架構



# 簡易版推薦系統架構





## 簡易版推薦系統步驟

1. 在欲監控網頁埋Javascript
2. 使用Flume 蒐集使用者行為
3. 使用Pig 整理使用者行為資料
4. 使用Mahout 產生推薦結果
5. 使用將推薦結果放入Hbase
6. 使用Javascript取得推薦結果

使用Oozie  
或crontab 管理工作

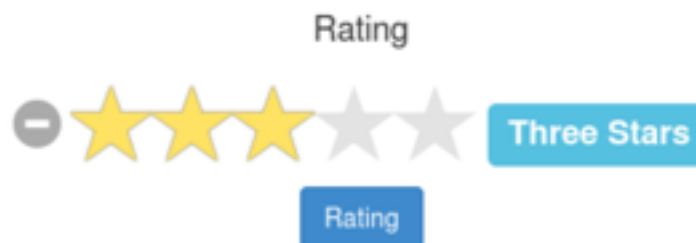
- 步驟請參閱[recommend-exercise.txt](#)

# 監控使用者所下的評等

## Shopping Recommendation Example



ASUS X550JD

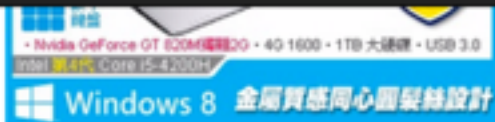


## Recommendation List



# 取出推薦結果

## Shopping Recommendation Example



ASUS X550JD

## Recommendation List



ASUS X550JD



ASUS X550JD



ASUS X550JD

# HANDS ON

<https://github.com/Jazznight/hadoopsience2>



# 聯絡方式

- Website:

- <http://www.largitdata.com/>

- Email:

- [brian@largidata.com](mailto:brian@largidata.com)



The background features a light blue hexagonal grid pattern. Overlaid on this is a large, stylized graphic of concentric circles and arcs in shades of blue and white, resembling a target or a complex geometric design. The text "THANK YOU" is centered in a bold, dark blue font.

**THANK YOU**