

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

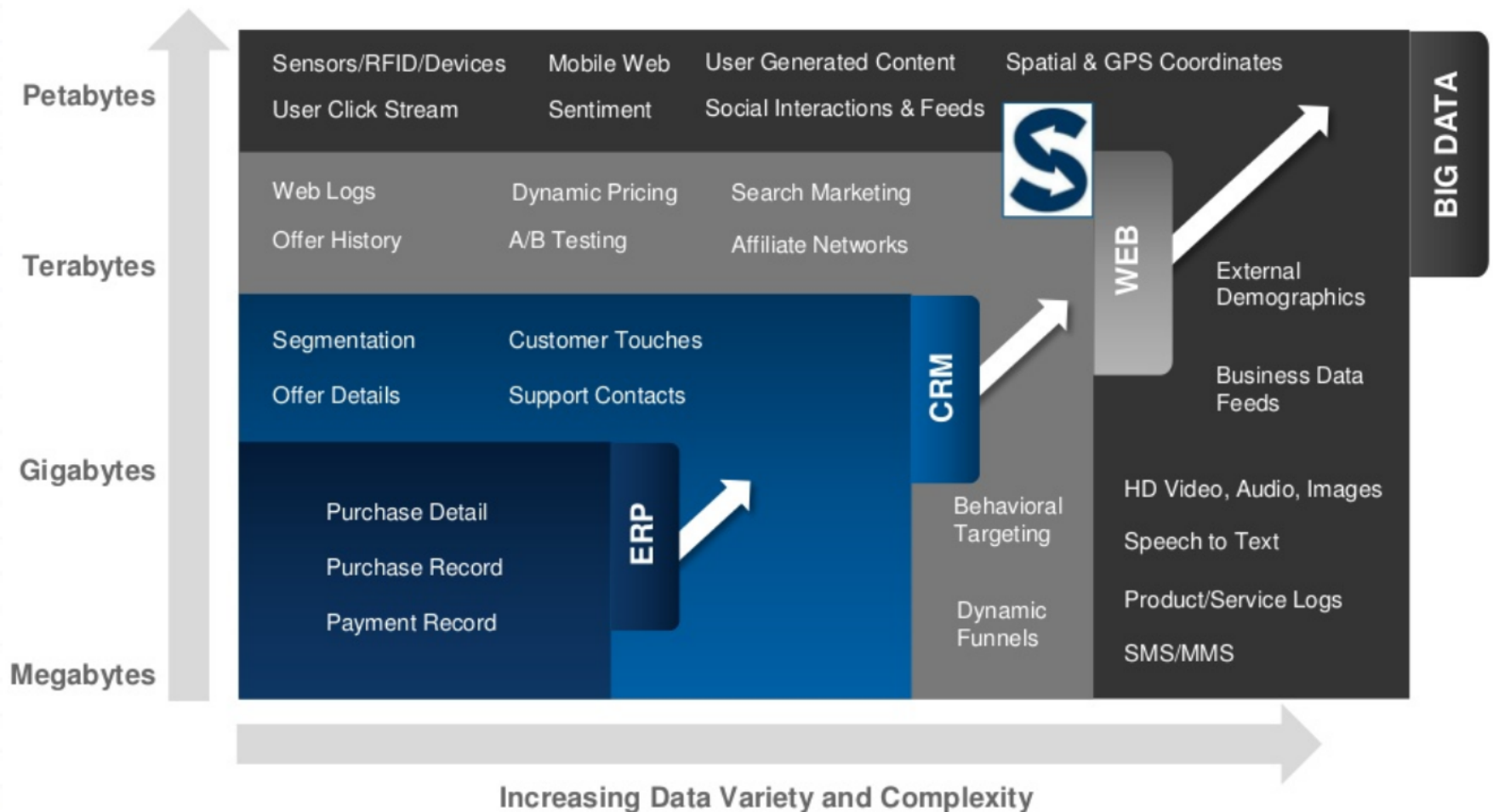
女人

使用者為中心的大數據實務

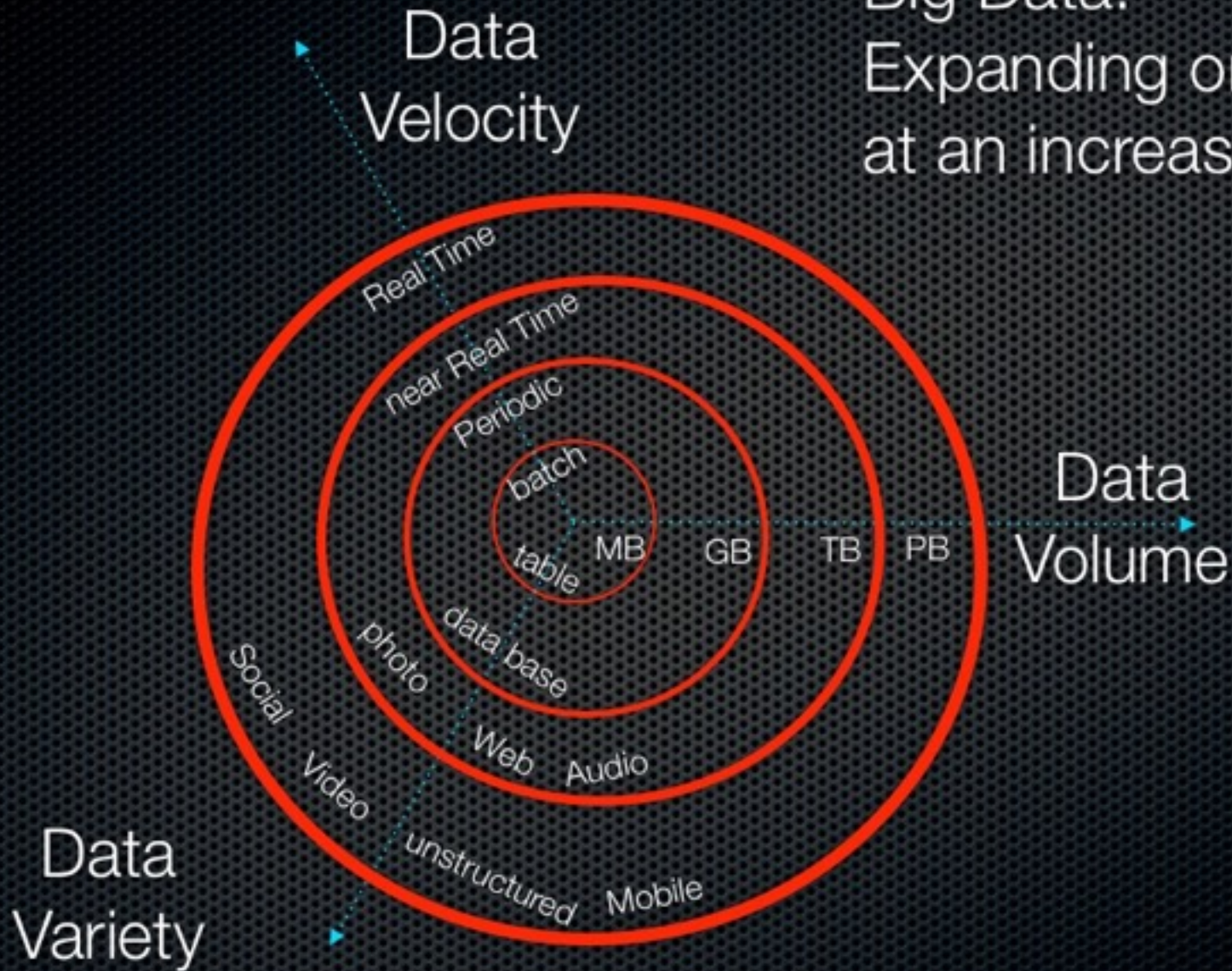


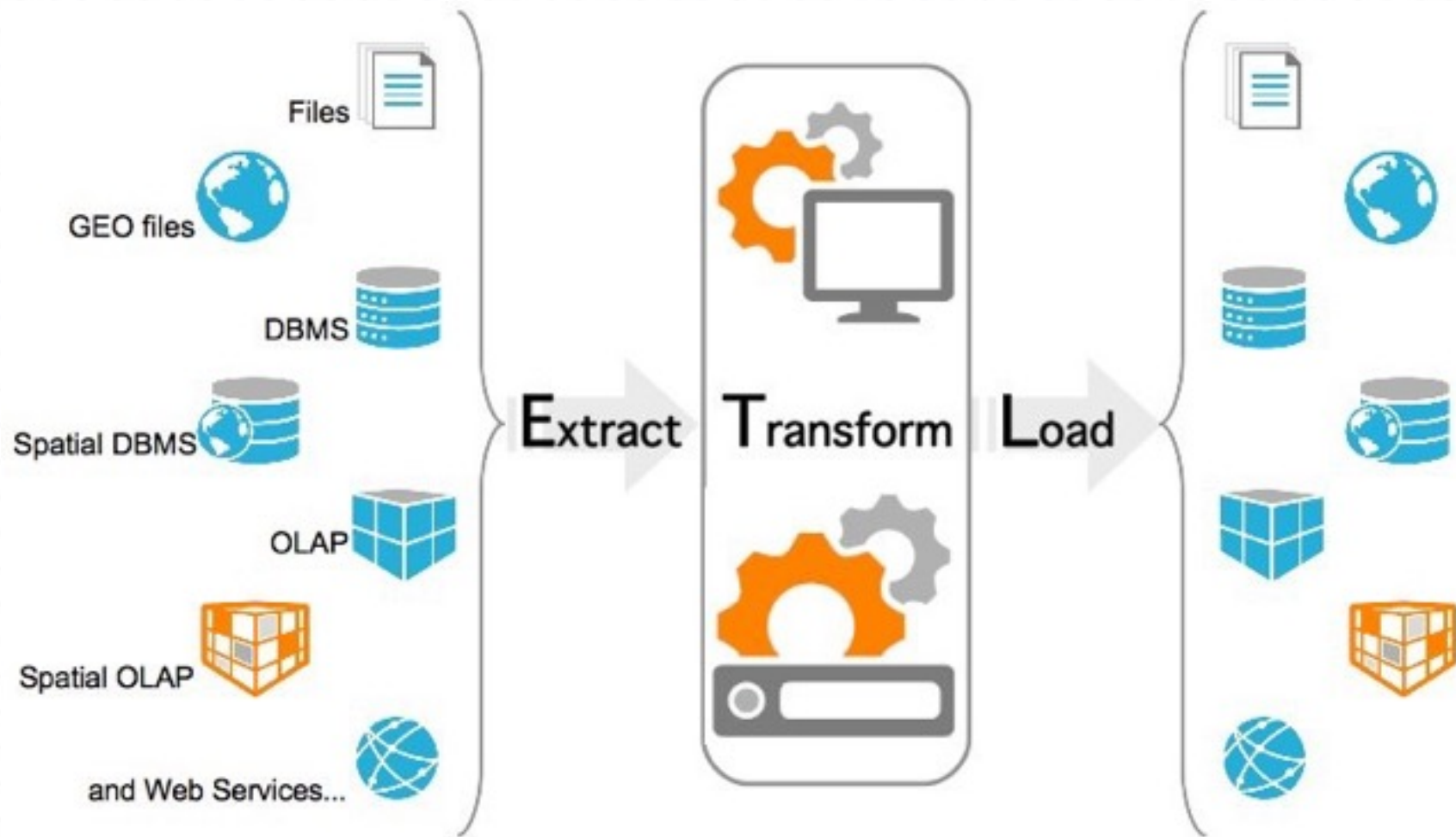
Hadoop與大數據分析

Big Data = Transactions + Interactions + Observations

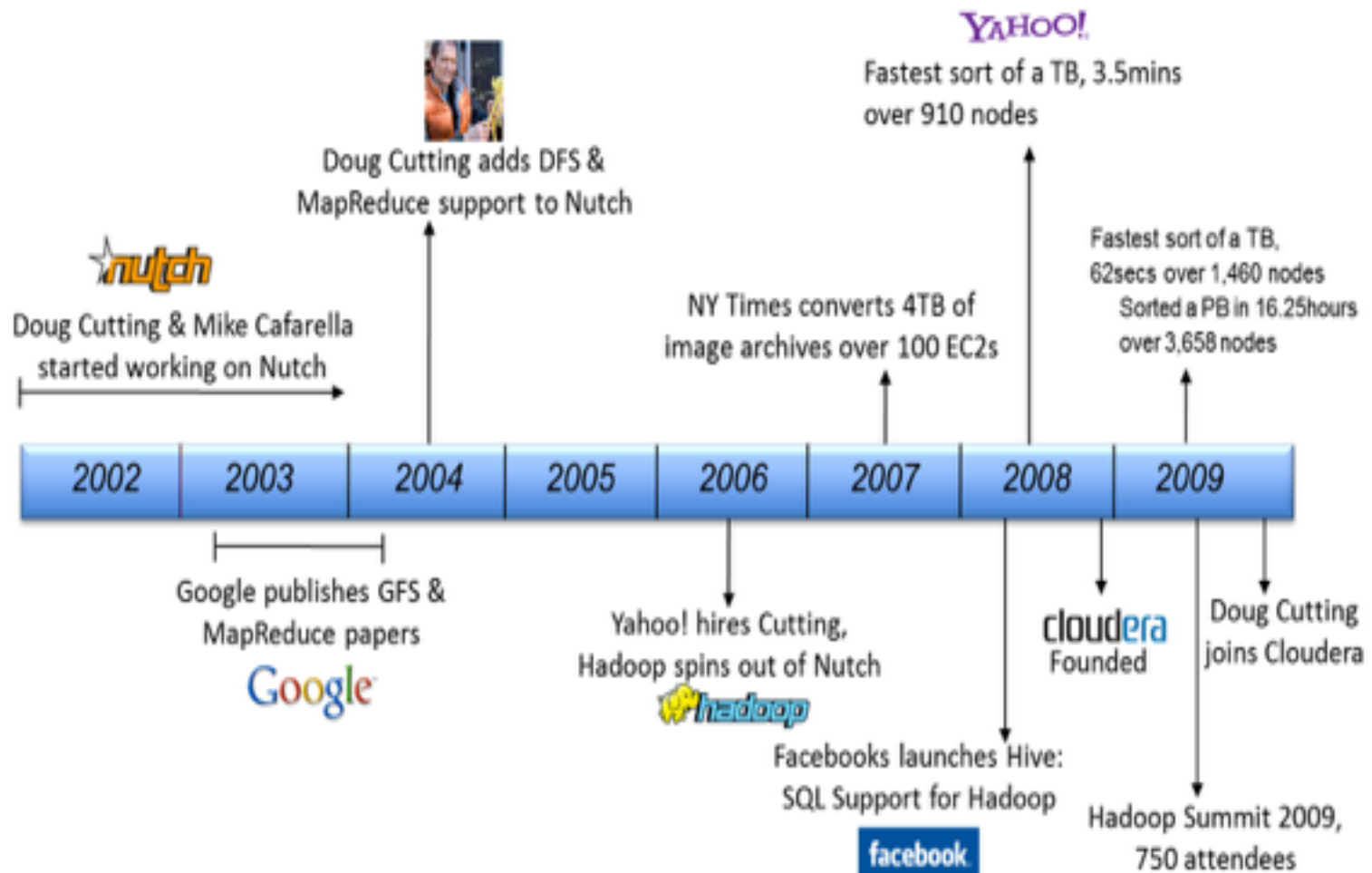


Big Data:
Expanding on 3 fronts
at an increasing rate.





Hadoop 歷史



Hadoop – 巨量資料解決方案

- Apache 計畫

- 由Doug Cutting and Mike Cafarella在Yahoo 時所發起

- 特色

- 分散式主從(Master/Slave)架構

- 高容錯性

- 標準化量產硬體

- 以Java 撰寫

- 讓開發者專注於應用開發



Hadoop 的限制

- 針對大型(Terabyte, Petabyte)及不同格式資料所設計
- 批次執行 (Batch Mode)
- 可分散的問題
 - ▣ 全域最佳化問題
 - ▣ 圖形計算問題

Hadoop 不是

- 不是資料庫的替代品
 - 無法產生快速、即時回應
 - 可以使用Hbase 等 NoSQL 解決方案
- 無法做即時性查詢 (Ad-Hoc Query)
 - 可以使用Impala取代
- 不適合用在需要大量迴圈計算的工作
 - 可以使用Spark 等技術替代

Hadoop 運作原理

Hadoop 主要功能

- HDFS (Hadoop Distributed File System)
 - ▣ 分散式檔案系統
 - ▣ 提升讀取資料的速度
 - ▣ 高容錯性
 - ▣ Master(NameNode) – Slave(DataNode) 架構
- MapReduce
 - ▣ 以Map (Divide)跟 Reduce (Conquer) 進行平行運算
 - ▣ 源自函數式(Functional)程式語言
 - ▣ Master(JobTracker) – Slave(TaskTracker) 架構 **#MRV1**

資料儲存分散

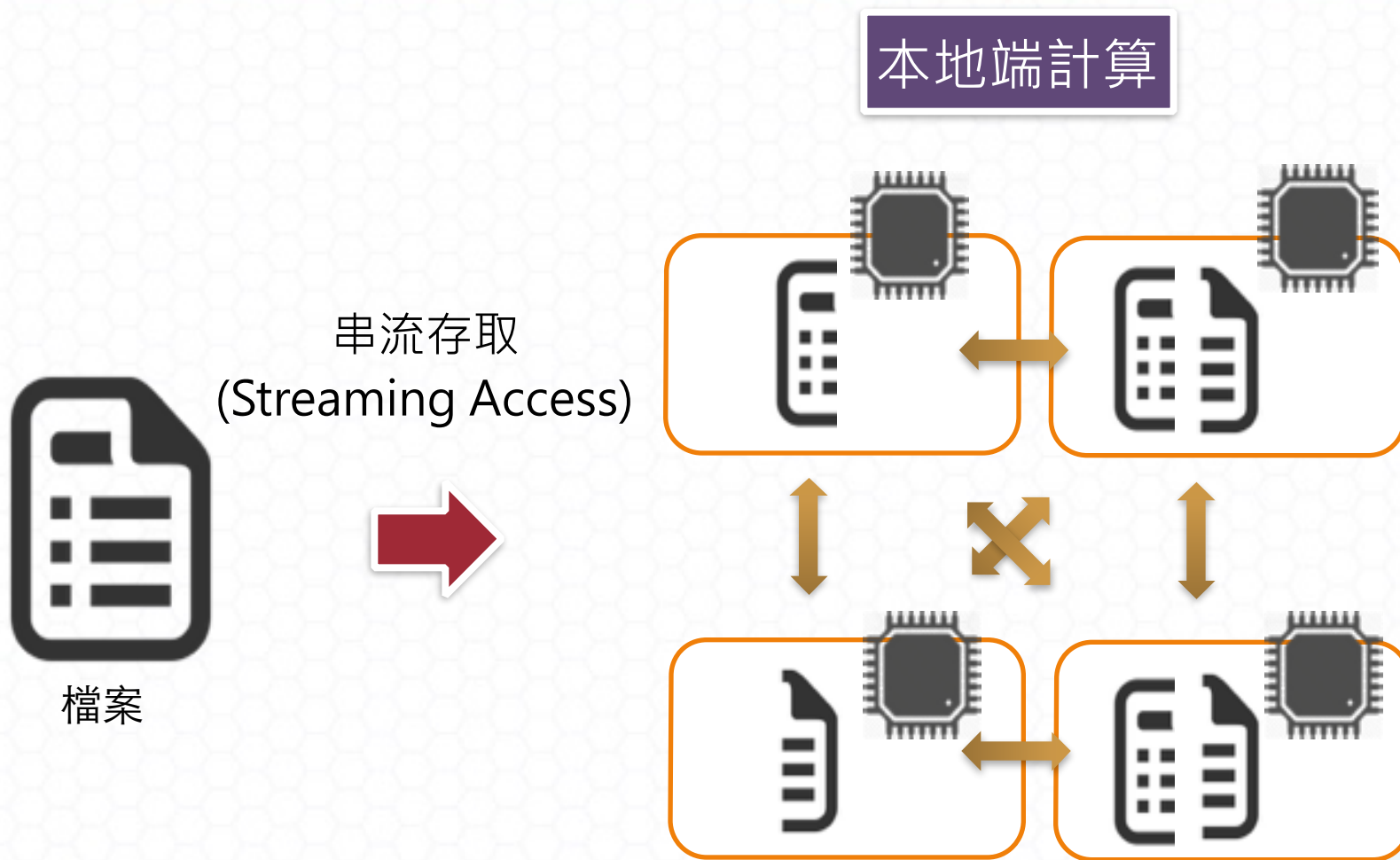
資料分割 (Striping)
提升讀取/寫入速度
RAID 0



資料鏡像 (Mirroring)
資料備援
RAID 1

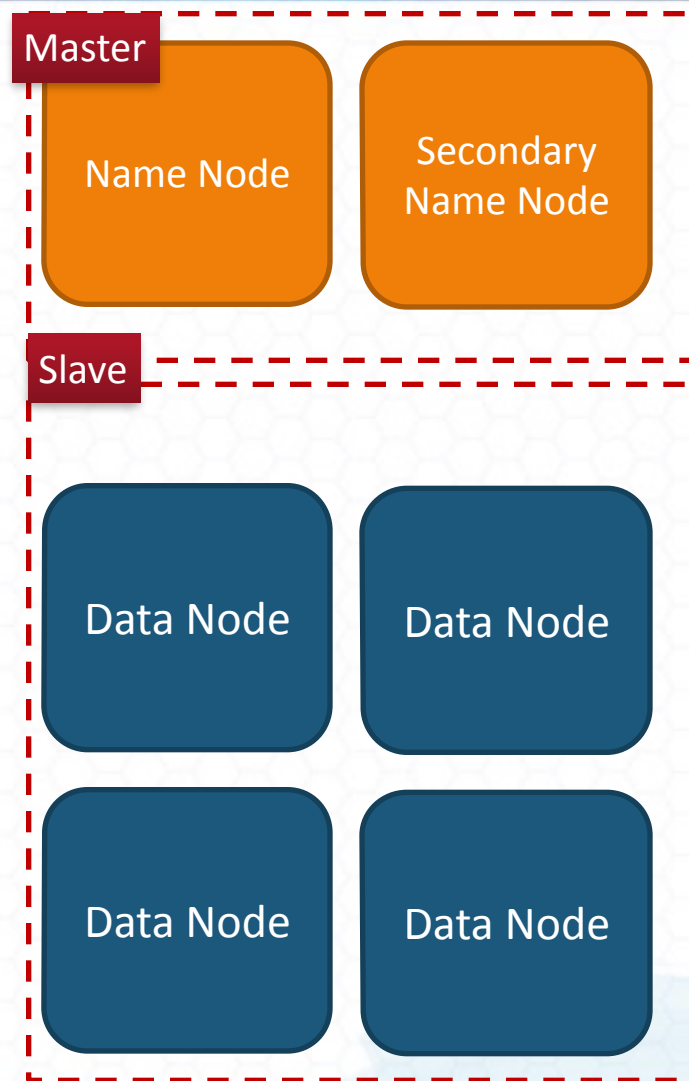


Hadoop 資料運作



名詞解釋

- Master/Slave
 - 實體機器
- Name Node/ Data Node
 - 運行的代理服務 (Daemon)
- Master Daemon
 - Name Node
 - Secondary Name Node
 - 負責工作調配
- Slave Daemon
 - Data Node
 - 負責”工作”



Name Node

- 由Name Node發起資料的寫入/讀取
- 如同Linux inode，儲存描述資料(Metadata)的資訊
 - 檔案名稱，權限，目錄
 - 哪個Data Node 包含資料塊
- Metadata 可儲存於記憶體與磁碟中
 - 必須持續將Metadata 定期做備份
 - 如果只有一個Name Node，很可能會因為單點毀損導致無法存取任何資料

Name Node

- Name Node將Meta Data 儲存於磁碟中

- 儲存於兩個檔案

- fsimage (檔案的Snapshot)
 - edits log (自snapshot 後關於HDFS 的修改紀錄)

- fsimage 跟 edits log 會定期合併

- 當Name Node 重新啟動時
 - 由Secondary Name Node 合併

需要大量的記憶體

- 儲存空間配置

- 至少兩個磁碟 (或使用RAID)

- 遠端NFS 載點

- fsimage 必須每天或每周備份

Secondary Name Node

- 負責檢核 (Checkpoint) Name Node
- 非即時備援
- 使用跟Name Node 一樣多的記憶體

Data Node

- 實際儲存資料
- 客戶端程式是向Data Node 直接存取資料

HeartBeats

- Data Node 會向Name Node 送出回應(Heartbeats)
 - 當經過30秒沒有回應(Heartbeats) , Name Node 會選擇向其他Data Node 取得資料
 - 當經過10分鐘沒有回應(Heartbeats) , Name Node 會將該Data Node 儲存的資料另外複製到其他Data Node

Block Reports

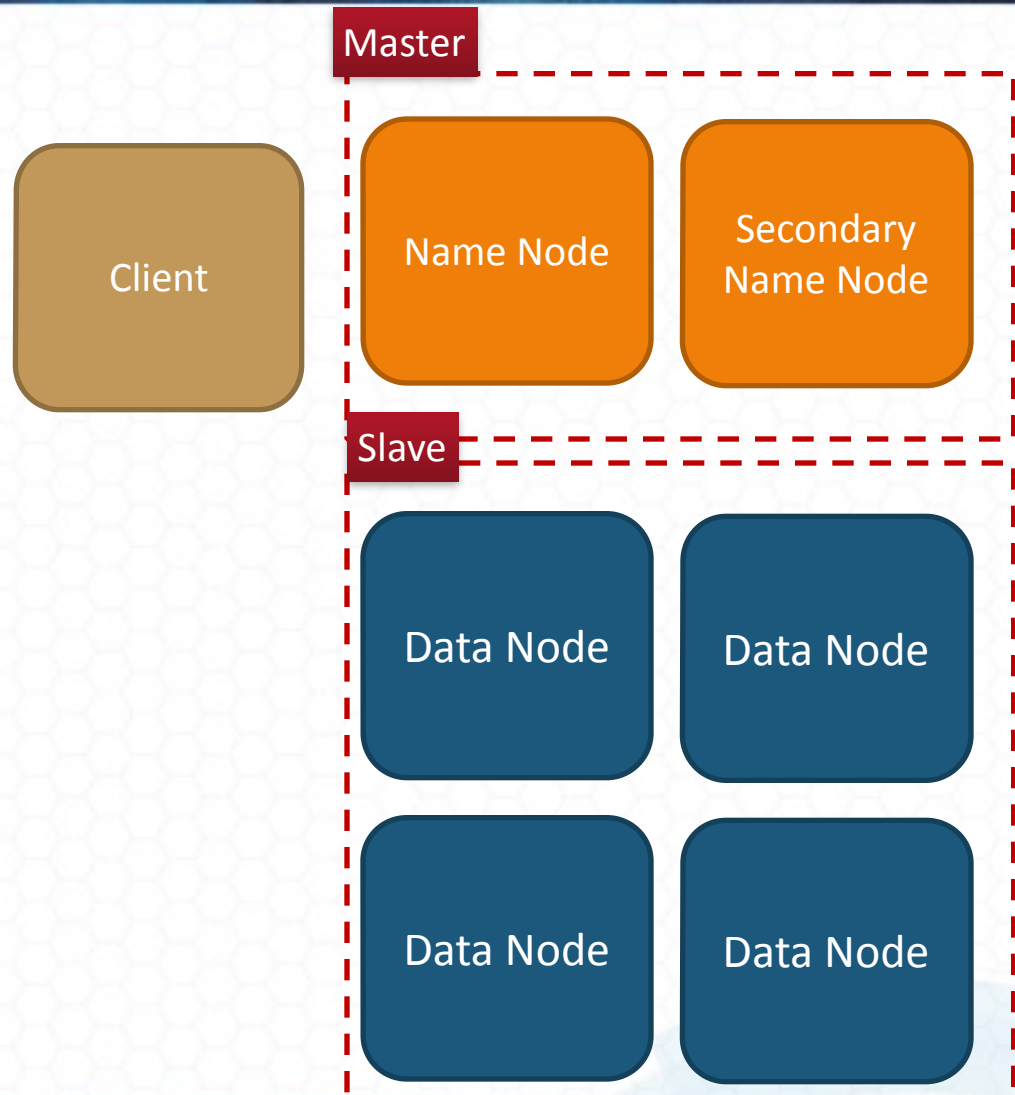
- 每隔一個小時，Data Node 會跟Name Node 同步目前內含的資料塊
- 當Name Node 發現Data Node 的資料塊比當初分配的要少，便會重新配置資料，確認資料有符合當初分散的數量設定
- 當Data Node 與Name Node 連結時，便會自動送一份報告

Checksums

- 資料在寫入時，會在寫入的資料塊旁(Chunk)增加 checksums 檔案
- 當讀取時發現資料checksums 不符
 - ▣ 會讀取儲存於另一個Data Node 的資料塊
 - ▣ Name Node 會重新複製資料塊
- Data Node 會每隔三周執行資料塊checksums確認

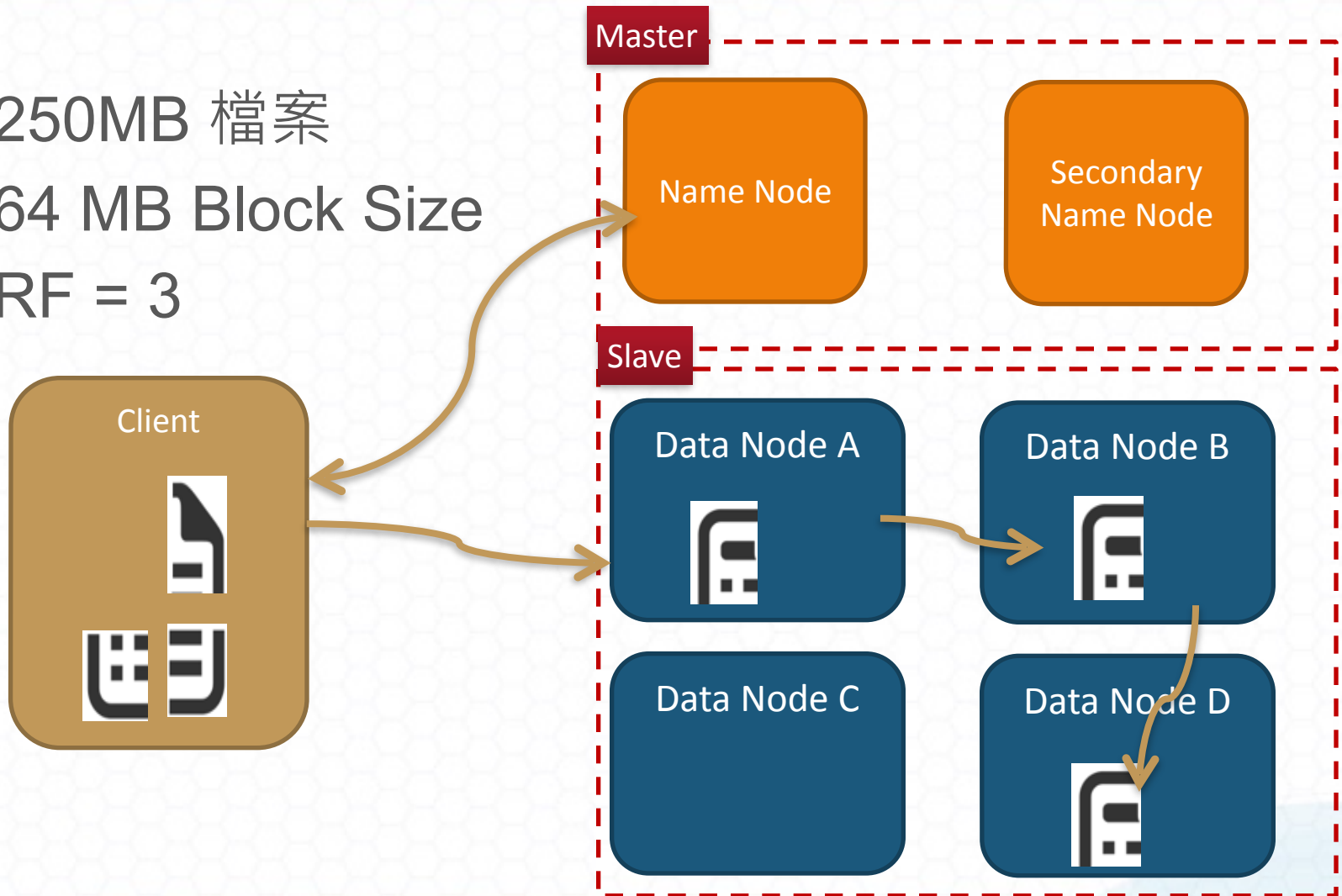
主要元件

- Client
- NameNode
 - ▣ Master Daemon
 - ▣ 工作協調與分派
- DataNode
 - ▣ Slave Daemon
 - ▣ 資料存儲



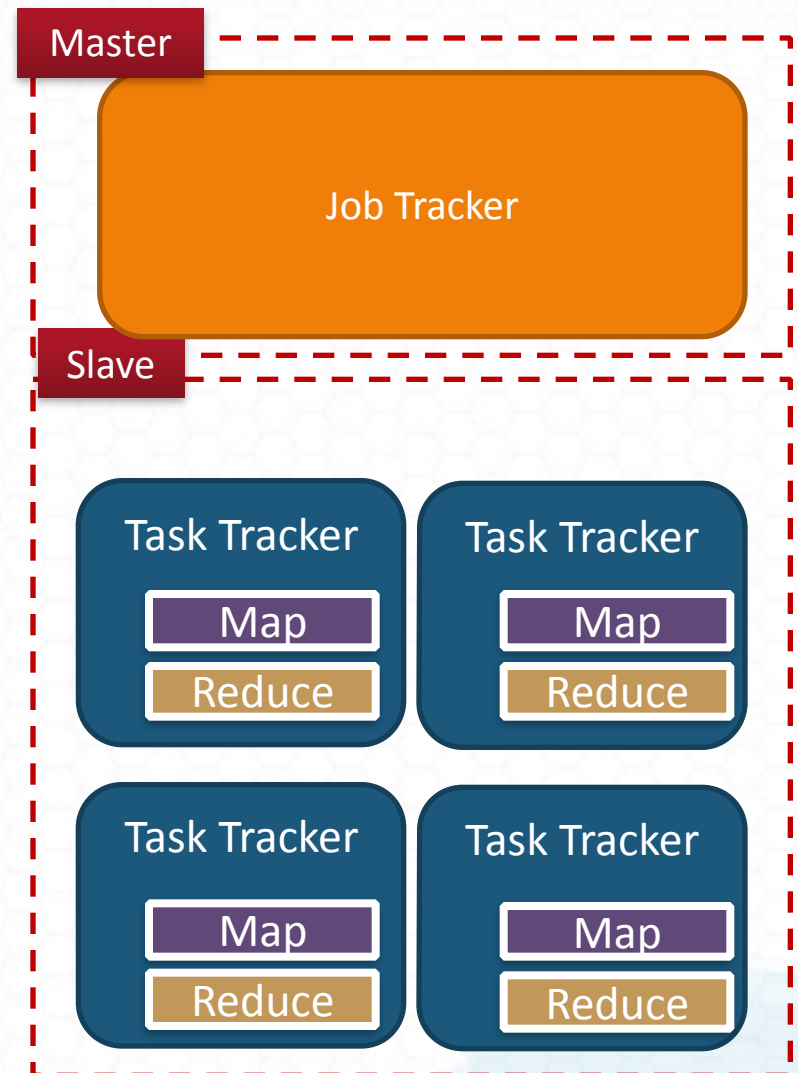
寫入範例

- 250MB 檔案
- 64 MB Block Size
- $RF = 3$



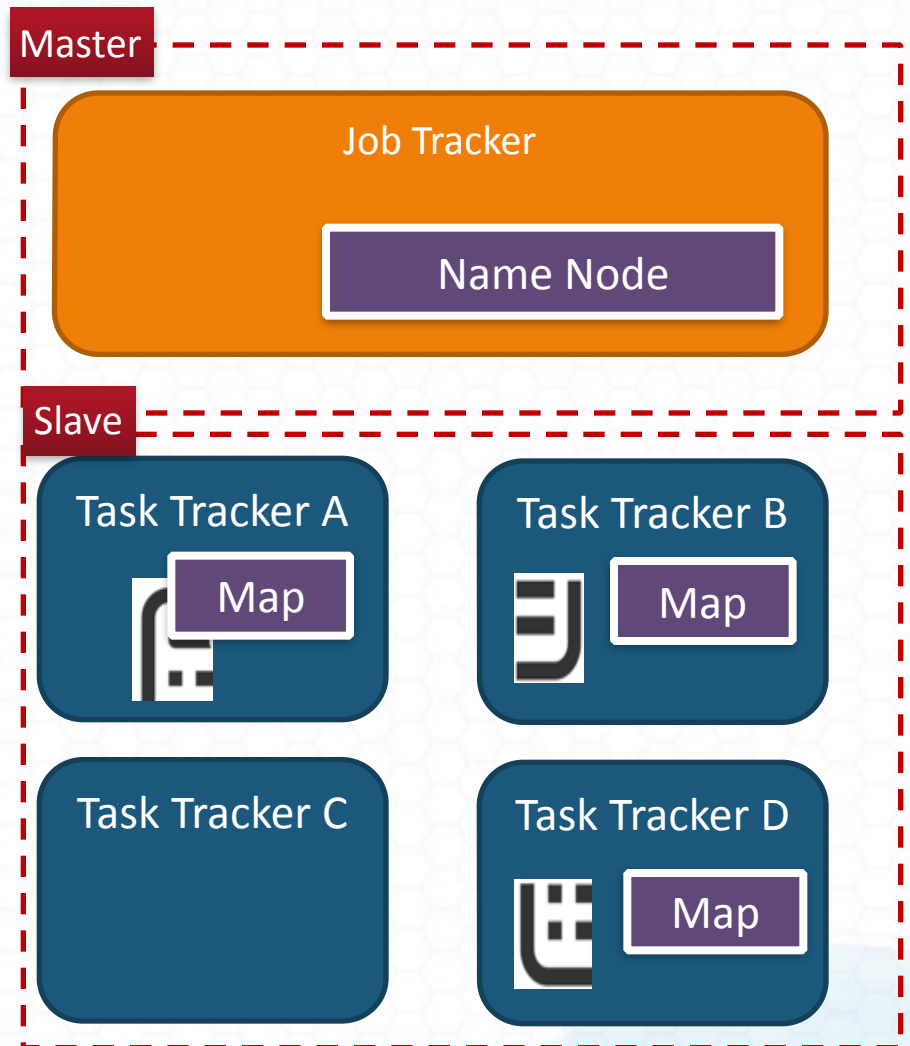
MRV1

- Master/Slave
 - 實體機器
- Job Tracker/ Task Tracker
 - 運行的代理服務 (Daemon)
- Master Daemon
 - Job Tracker
 - 負責工作調配
- Slave Daemon
 - Task Tracker
 - 負責”工作”
 - 操作Map 及 Reduce 工作



MRV1 運作模式

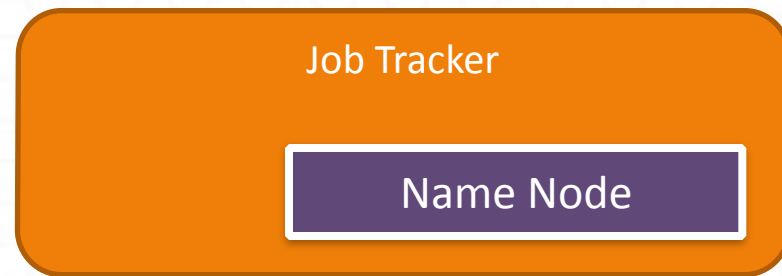
- Client 送出工作給Job Tracker
- Job Tracker 詢問Name Node 關於資料位置
- Job Tracker 根據資料 (本地性) 決定由哪幾個Task Tracker 處理資料
- 擁有資料處的Task Tracker 會開始啟用Mapper



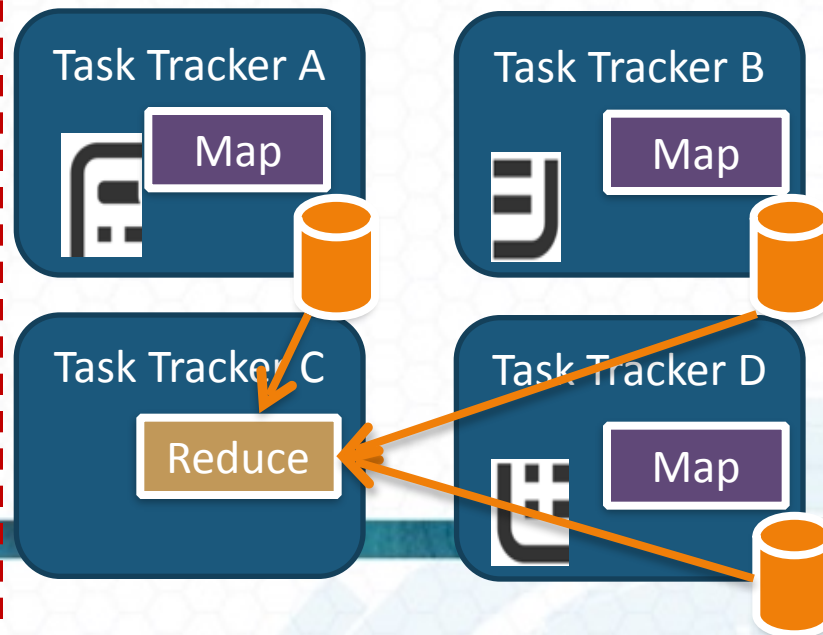
MRV1 運作模式

- 將中介結果寫到本地端的磁碟
- 空閒的Task Tracker (沒有資料本地性)會開始啟用Reducer
- 將Mapper 的中介結果複製到Reducer 處進行Reduce
- 將處理結果寫回HDFS

Master



Slave



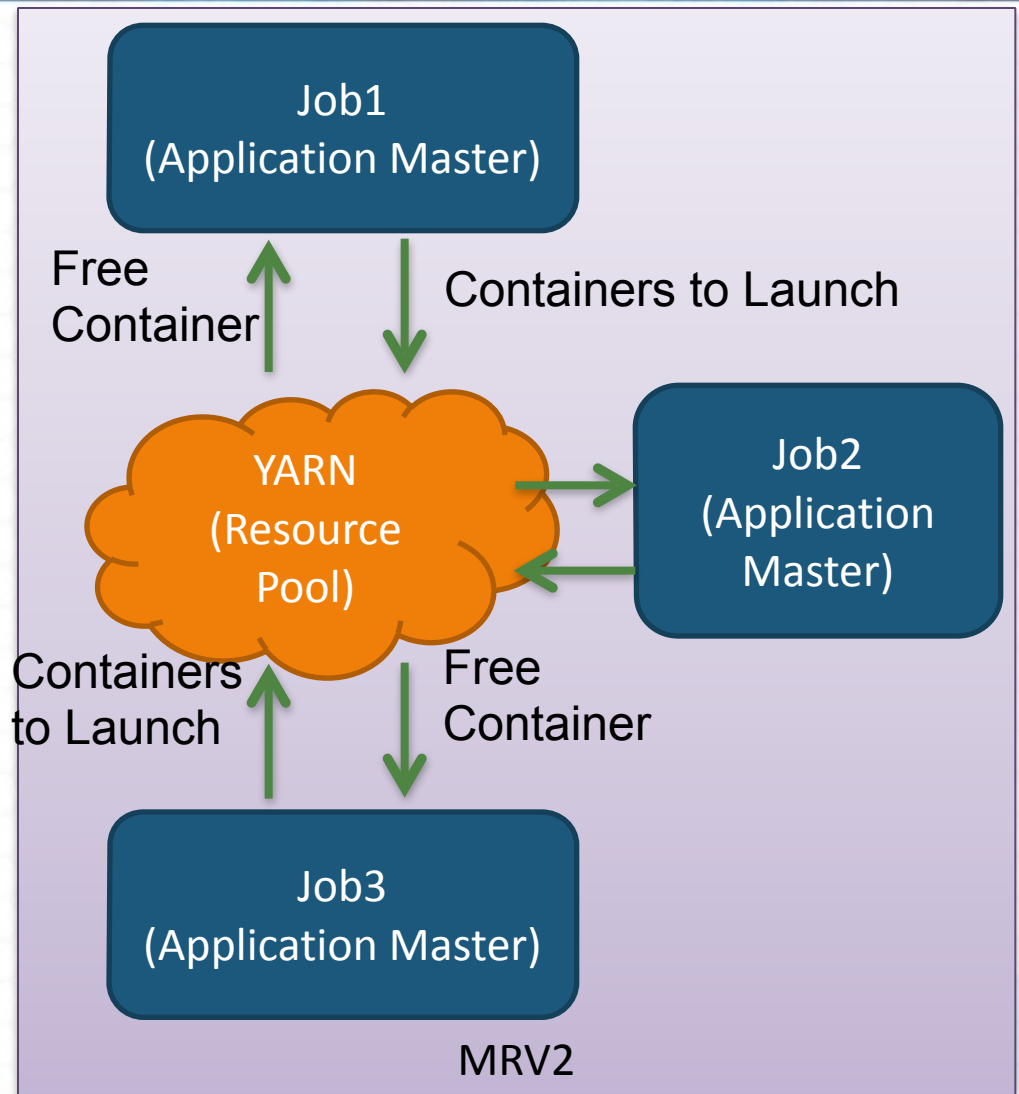
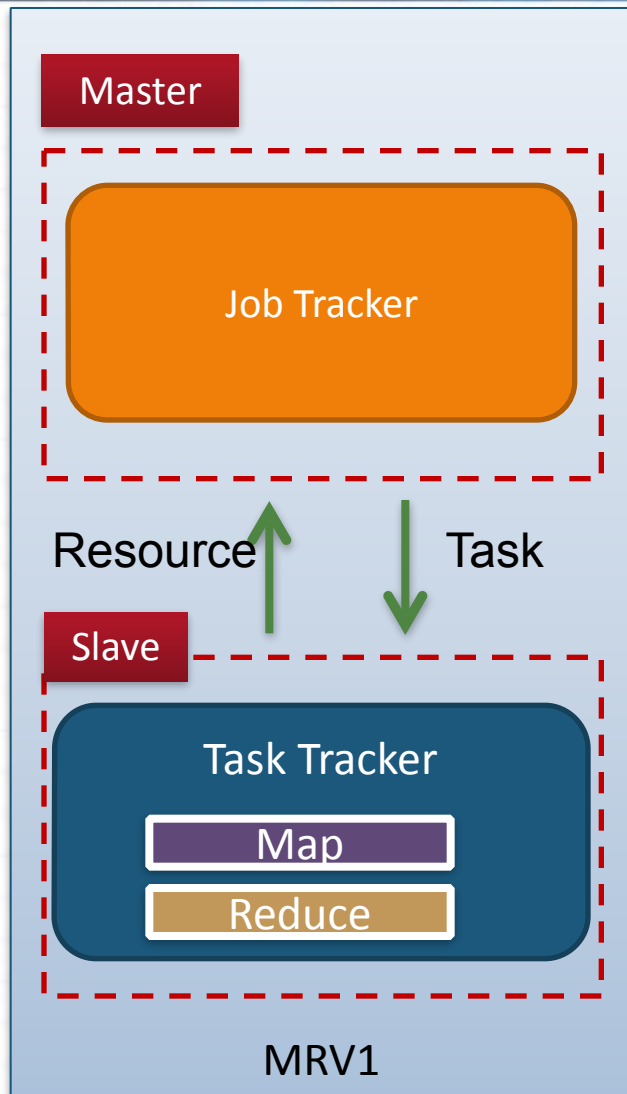
MRV2

- 改進MRV1 的延展性跟異質框架相容性
- 延展性
 - ▣ 將Job Tracker 中的作業控制及資源管理分開
 - ▣ Resource Manager: 資源管理與調度
 - ▣ ApplicationMaster 應用程式任務執行與作業控制
 - ▣ 可動態分配Map Slot 與 Reduce Slot
- 異質框架
 - ▣ 可以替換上面的計算框架
 - 使用MPI, Spark

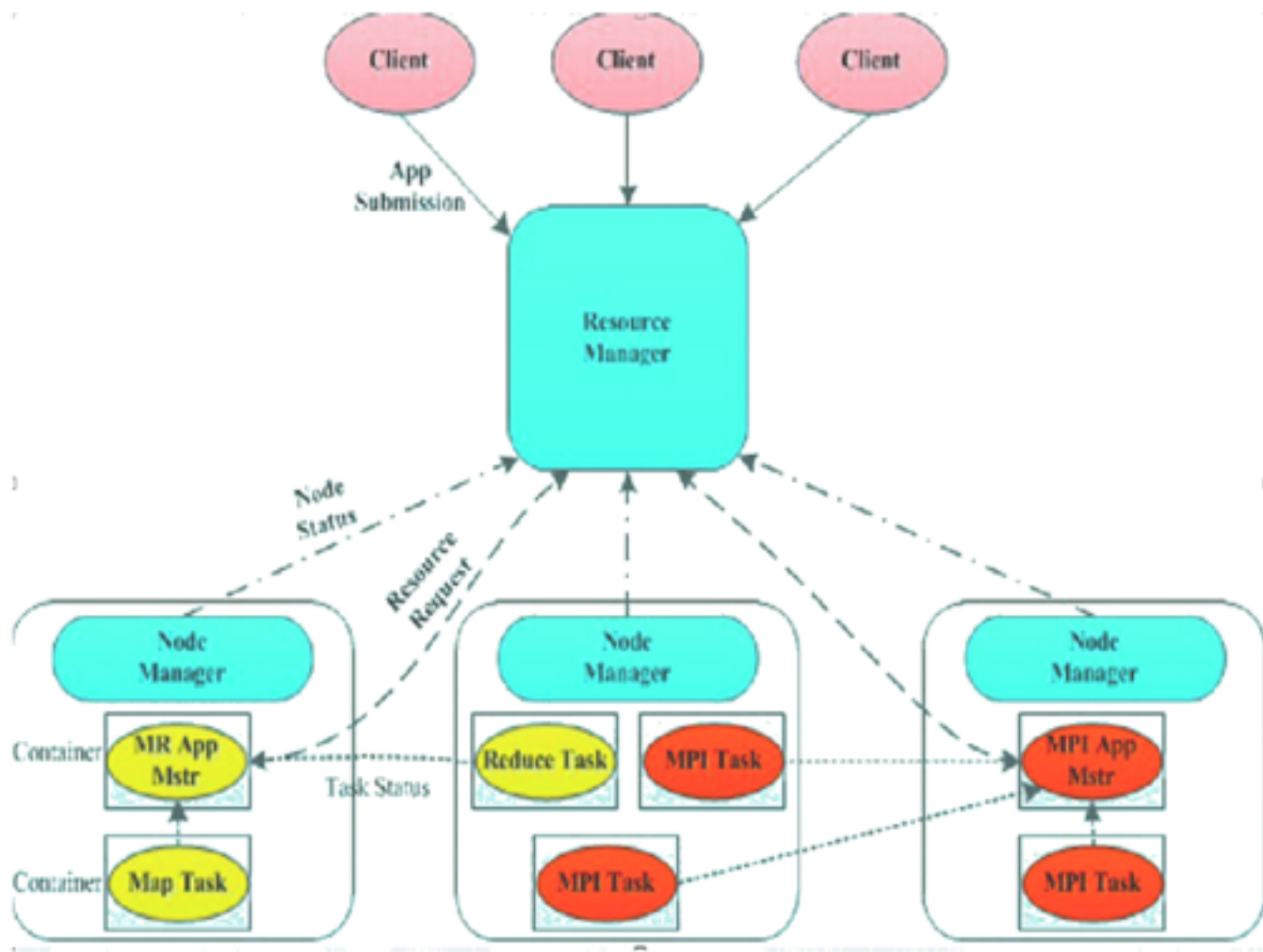
YARN 元件 - Yet Another Resource Negotiator

- Resource Manager
 - 負責資源管理與分配
 - Applications Manager: 管理系統中的應用程式
 - Scheduler: 檢查資源，分配資源給運行中的程式
- ApplicationMaster
 - 與RM 溝通
 - 分配工作
 - 告知Node Manager啟動或停止
 - 監控所有運行程式
- NodeManager
 - 每個Slave 上的資源與任務管理器，定時向RM報告資源與Container 運行
- Container
 - 根據需求所動態配置的資源 (不同於MRV1)

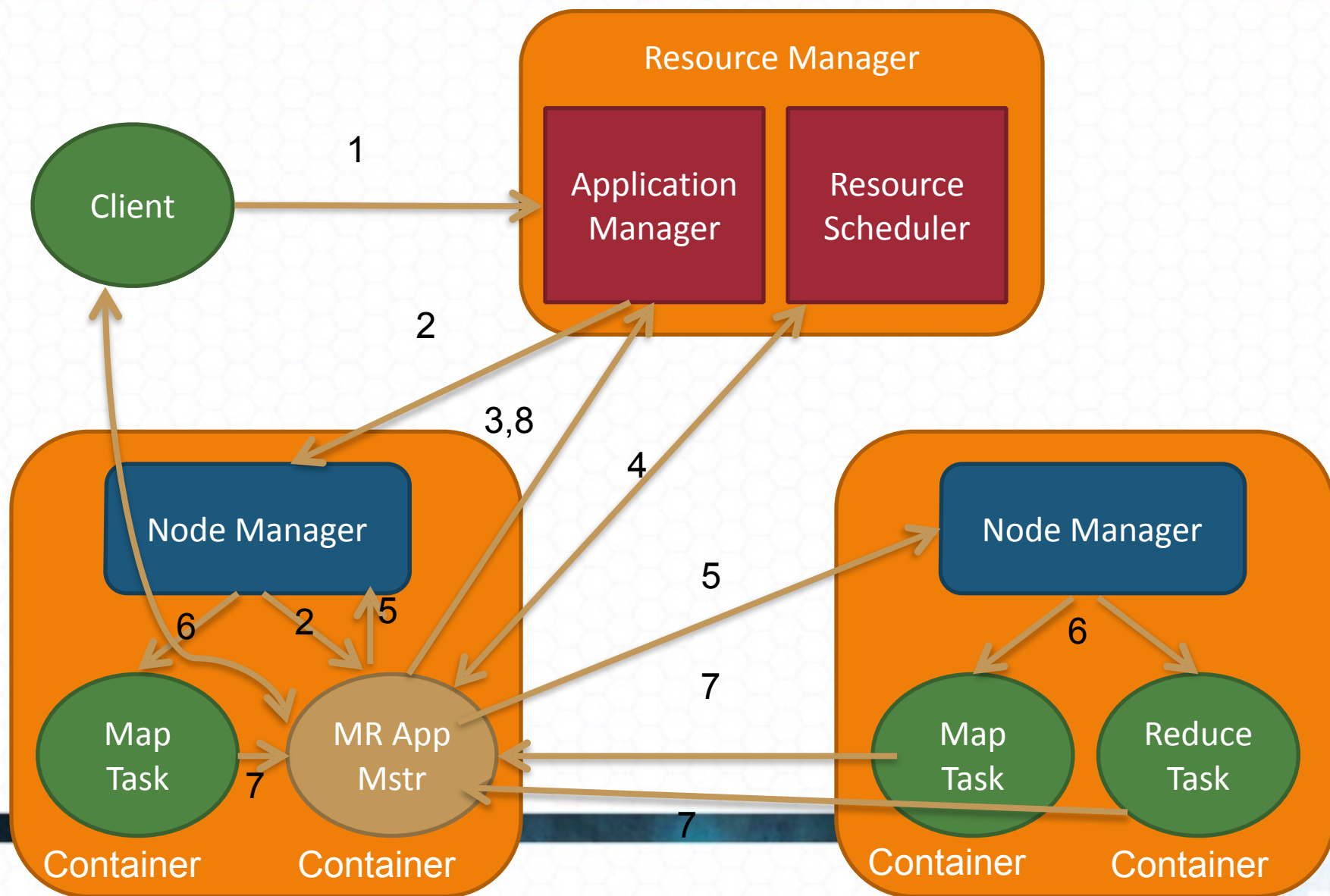
MRV1 v.s. MRV2



YARN 架構



YARN 工作流程



HDFS 簡介

HDFS 解決的問題

- 硬體毀損
 - 單一機器毀損的機率低，但是管理多台機器時有任一台毀損的機率就相當高
- 串流(Streaming)資料存取
 - Hadoop 是以批次方式取得大量資料
 - 希望以最高效率取得資料
- 巨量資料儲存
 - 可以透由多個機器分散式存儲資料

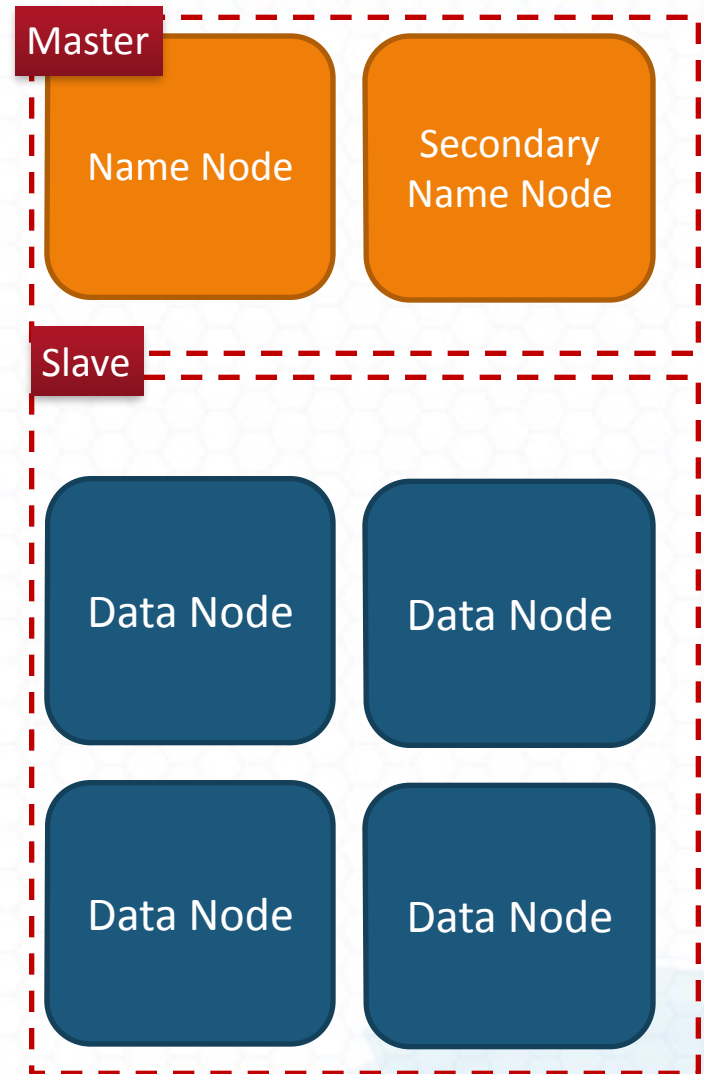
HDFS 解決的問題(二)

- 簡單一致的存儲模型
 - ▣ 寫一次可以多次讀取
- 搬運電腦比搬運資料的成本要小
 - ▣ 資料跟運算資源盡可能放在一起
- 可以在異質環境與平台下建立或轉移HDFS

HDFS

- Hadoop Distributed File System
 - 概念源自於 Google GFS
- 檔案會以資料塊(Chunk)的方式分散式儲存於Slave的Data Node中
 - Block Size:64MB
- 預設將資料塊複製成三份儲存於不同的Data Node 中 (hdfs-site.xml)

```
<property>  
  <name>dfs.replication</name>  
  <value>3</value>  
</property>
```



HDFS

- 可以承受Slave Node 毀損，或資料塊毀損
- 一次寫入，多次讀取 (Write Once, Read Many Times)
- 通常會將資料切成64MB ~ 128 MB大小的資料塊 (Chunk)

優點

- HDFS 可以建立於量產化標準硬體 (Commodity Hardware)
- HDFS 可以分散式存儲大量資料
- HDFS 適合串流資料寫入
 - ▣ Write once, read many times
 - ▣ 比如說Log 檔
- HDFS 會將運算資源跟運算資料配置於一起

缺點

- HDFS 不適合存放小檔案
 - 會以Block Size 做存儲
- HDFS不適合隨機存取 (Random Access)
 - 必須從開頭或結尾循序讀取
 - 不允許多個代理服務存取

HDFS v.s. 關聯式資料庫

- HDFS 不是關聯式資料庫
 - 將資料分散儲存在不同節點
 - 使用MapReduce 存取資料
 - 無法快速存取資料
 - 沒有資料庫ACID 的特色
 - 不會將資料Cache 住
- 可以在上面建立 NoSQL (例如HBase)

HDFS v.s. 分散式存儲系統

- Hadoop 進行平行運算時，強調資料本地化 (Locality)
 - ▣ 資料與運算資源盡量配置於同一台
- 不建議以 SAN 或 NAS 作背後儲存機制
 - ▣ 因為巨量資料在節點的傳輸過程中會造成瓶頸

其他運作細節

- HDFS 是虛擬檔案系統
 - ▣ 預設Block Size: 64MB (所有Block 大小都相同)
 - ▣ 預設RF: 3 (節點數必須 \geq RF)
- Data 只在Data Node 之間傳輸
 - ▣ 資料寫入時會同時寫入Checksums
 - ▣ 採取循序寫入的動作
- Name Node只有Meta Data
 - ▣ Name Node 會單點毀損，導致工作失敗
 - ▣ 導致只看到一堆Block，看不到檔案

操作HDFS

將page view 統計放置於HDFS

■ 使用Wikipedia Page View 統計資訊

▣ <https://dumps.wikimedia.org/other/pagecounts-raw/>

Page view statistics for Wikimedia projects

Pagecount files per year

- [2007](#)
- [2008](#)
- [2009](#)
- [2010](#)
- [2011](#)
- [2012](#)
- [2013](#)
- [2014](#)

What are the page view statistics files and what do they contain?

Each request of a page, whether for editing or reading, whether a "special page" such as a log of actions generated on the fly, or an article from Wikipedia or one of the other projects, reaches one of our squid caching hosts and the request is sent via tcp to a filter which routes requests from our internal hosts, as well as requests for wikis that aren't among our general projects. This filter writes out the project name, the size of the page requested, and the role of the page requested.

Here are a few sample lines from our file:

```
fr:h Special:Research/Archiva_Burgundy_00000000000000000000 1 614  
fr:h Special:Research/Archiva_00000000000000000000 1 100  
fr:h Special:Research/Archiva_00000000000000000000 1 743  
fr:h Special:Research/Archiva_00000000000000000000 1 798  
fr:h Special:Research/Archiva_00000000000000000000 1 737
```

In the above, the first column "fr:h" is the project name. The following abbreviations are used:

取得pagecount資料

- 取得 2007年12月的pagecounts
 - ▣ wget <https://dumps.wikimedia.org/other/pagecounts-raw/2007/2007-12/pagecounts-20071209-180000.gz>
 - ▣ gunzip pagecounts-20071209-180000.gz

使用hadoop fs CLI (I)

- 建立目錄

- ▣ `hadoop fs -mkdir /data`

- 檢視目錄

- ▣ `hadoop fs -ls /data`

- 將資料上傳

- ▣ `hadoop fs -put pagecounts-20071209-180000 /data`

使用hadoop fs CLI (II)

- 檢視檔案內容

- ▣ `hadoop fs -cat /data/ pagecounts-20071209-180000 | head`
- ▣ `hadoop fs -tail /data/ pagecounts-20071209-180000`

- 檢視存儲空間大小

- ▣ `hadoop fs -df`

- 檢視特定目錄大小

- ▣ `hadoop fs -du /data`

使用hadoop fs CLI (III)

- 下載檔案

- ▣ `hadoop fs -get /data/ pagecounts-20071209-180000 test.txt`

- 刪除檔案內容

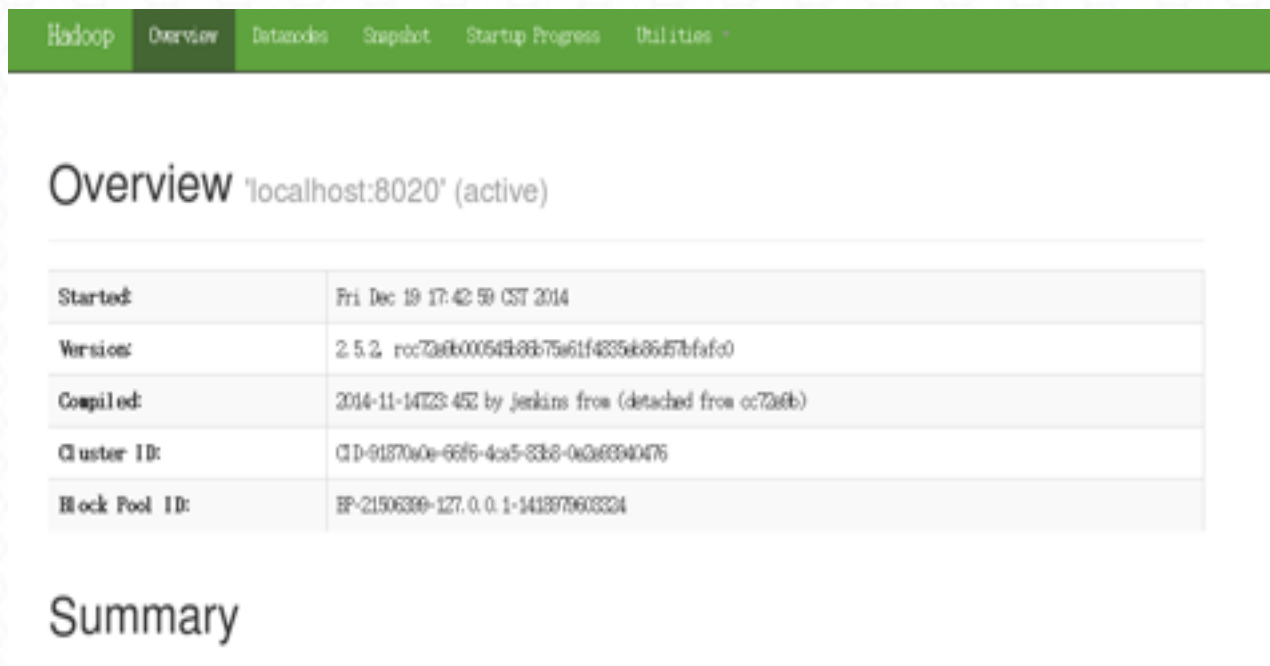
- ▣ `hadoop fs -rm /data/ pagecounts-20071209-180000`

- 刪除目錄

- ▣ `hadoop fs -rm -r /data`

檢視Name Node

■ localhost: 50070



Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Overview 'localhost:8020' (active)

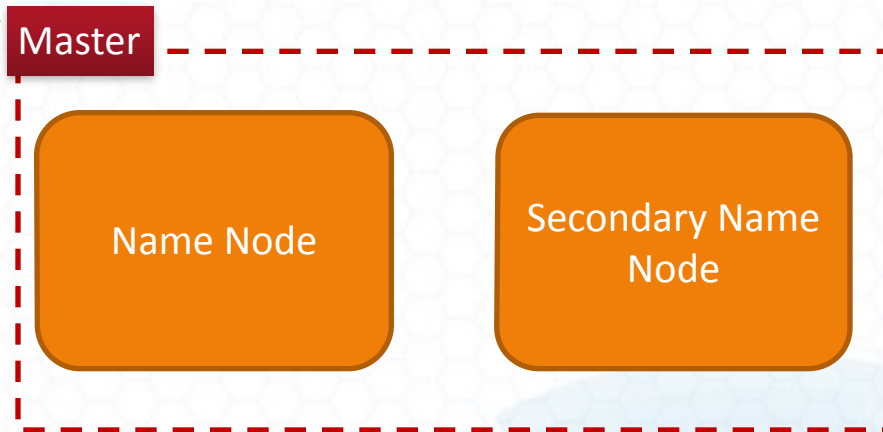
Started:	Fri Dec 19 17:42:59 CST 2014
Version:	2.5.2-rc72a6b000549b8b75e61f4335e886d57bfafcd
Compiled:	2014-11-14T23:45Z by jenkins from (detached from oc72a6b)
Cluster ID:	CID-91870e0e-66f6-4ca5-83b8-0a3a63940476
Block Pool ID:	BP-21506390-127.0.0.1-1413979603324

Summary

HDFS進階議題

Name Node組成

- fsimage
 - HDFS 當時的Snapshot
 - 檔案較為龐大 (讀取/寫入很慢)
- edits log
 - HDFS自Snapshot後的變更
 - edits log 檔案比較小



Secondary Name Node

- 非即時備援 (Non-Hot Failover)
- 監控Name Node 運行狀態
- 排程檢核 (每個小時)
 - ▣ 從 Name Node 處備份 fsimage 跟 edits log
 - ▣ 將 edits log 與fsimage 合併為大的fsimage

Secondary Name Node 備援

- Secondary Name Node 需要跟Name Node 一樣大的記憶體
 - ▣ 必須將fsimage 跟 edits log 讀取到記憶體中
 - ▣ 最好是將Secondary Name Node 放置於第二台機器
- 如果Name Node 毀損，可以使用Secondary Name Node 取代
 - ▣ 大約晚一小時
 - ▣ 不能即時備援

HDFS 權限控管

- HDFS 授權

- 透過類似Linux 檔案權限做權限控管
- 沒有驗證機制

- Kerberos 驗證

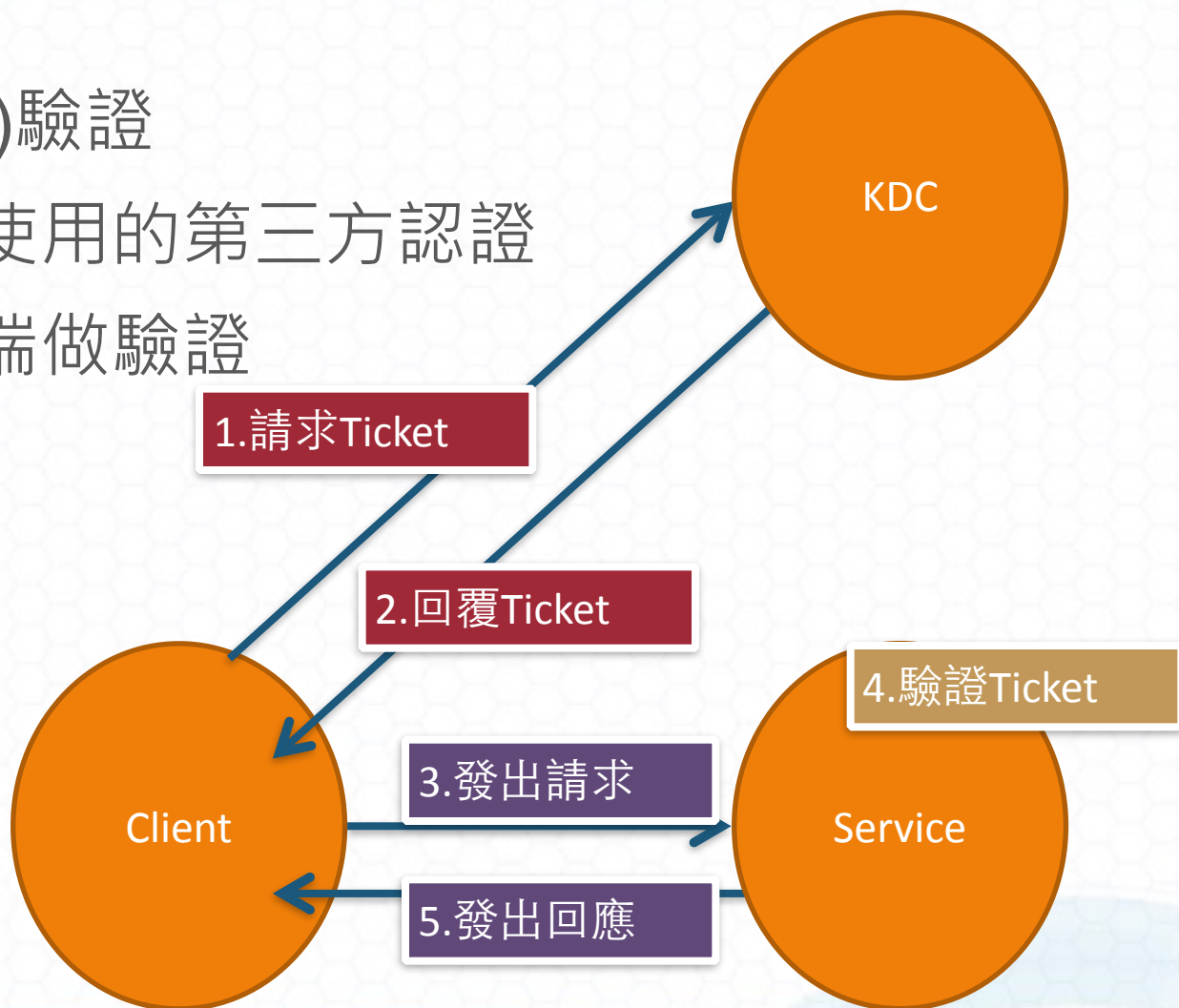
- 提供驗證機制

HDFS 權限

- 使用類Linux 的 `rw-rw-rw` 做 User, Group, Others 的權限控管
 - ▣ 644 (`rw-r--r--`)
 - ▣ 755 (`rw-r-xr-x`)
 - ▣ 600 (`rw-----`)
- `x` 代表可以瀏覽目錄
- `w` 代表可以刪除檔案
- `hadoop fs -chmod 755 /data/test2.txt`

Kerberos 驗證

- 票券(Ticketing)驗證
- 最廣為測試跟使用的第三方認證
- 可以由接受終端做驗證



其他資安防護

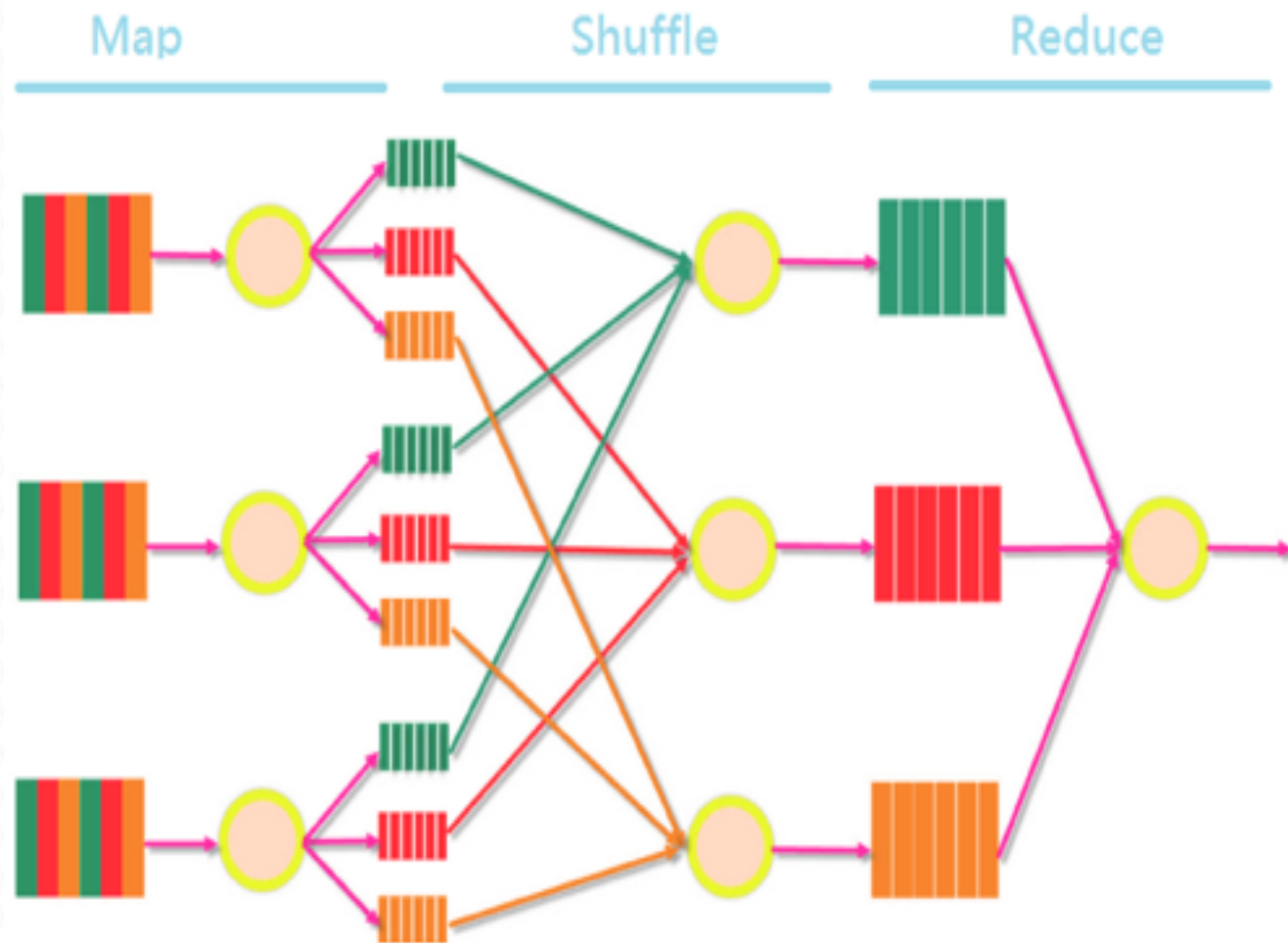
- 將Hadoop Cluster 放到獨立網段
- 允許登入方法
 - 使用ACL
 - 或使用跳板

MapReduce 簡介

MapReduce 與分散式處理

- Mapper (Divide)
 - 將工作量分散給數個機器處理
- Reducer (Conquer)
 - 整併(join) Map 所產生的結果
 - 如同資料的聚合 (Aggregation)
- MapReduce 中一定會有一個以上的Mapper 但不一定有Reducer

MapReduce 示意圖



以簡單的程式做理解

- 該如何統計這段文字中出現的字詞頻率

Hello my name is David , Tibame is my favorite



```
, 1  
David 1  
favorite 1  
Hello 1  
is 2  
my 2  
name 1  
Tibame 1
```


撰寫一個Mapper

```
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

針對所有字詞產生 (word, 1)

進行排序

echo "Hello my name is David , Tibame is my favorite"

| python mapper.py

Map

| sort -k 1

Shuffle

撰寫一個Reducer

```
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

合併(Join)Map得到的結果

呼叫MapReduce

echo "Hello my name is David , Tibame is my favorite"

| python mapper.py

Map

| sort -k 1

Shuffle

| python reducer.py

Reduce

Hadoop Ecosystem 簡介

Hadoop 與夥伴



什麼是Ecosystem?

- Hadoop 的子計畫或元件
- 除了下列元件以外的模組
 - ▣ MapReduce
 - ▣ HDFS
- 來源
 - ▣ Apache Foundation,
 - ▣ Cloudera, MapR, Hortonworks

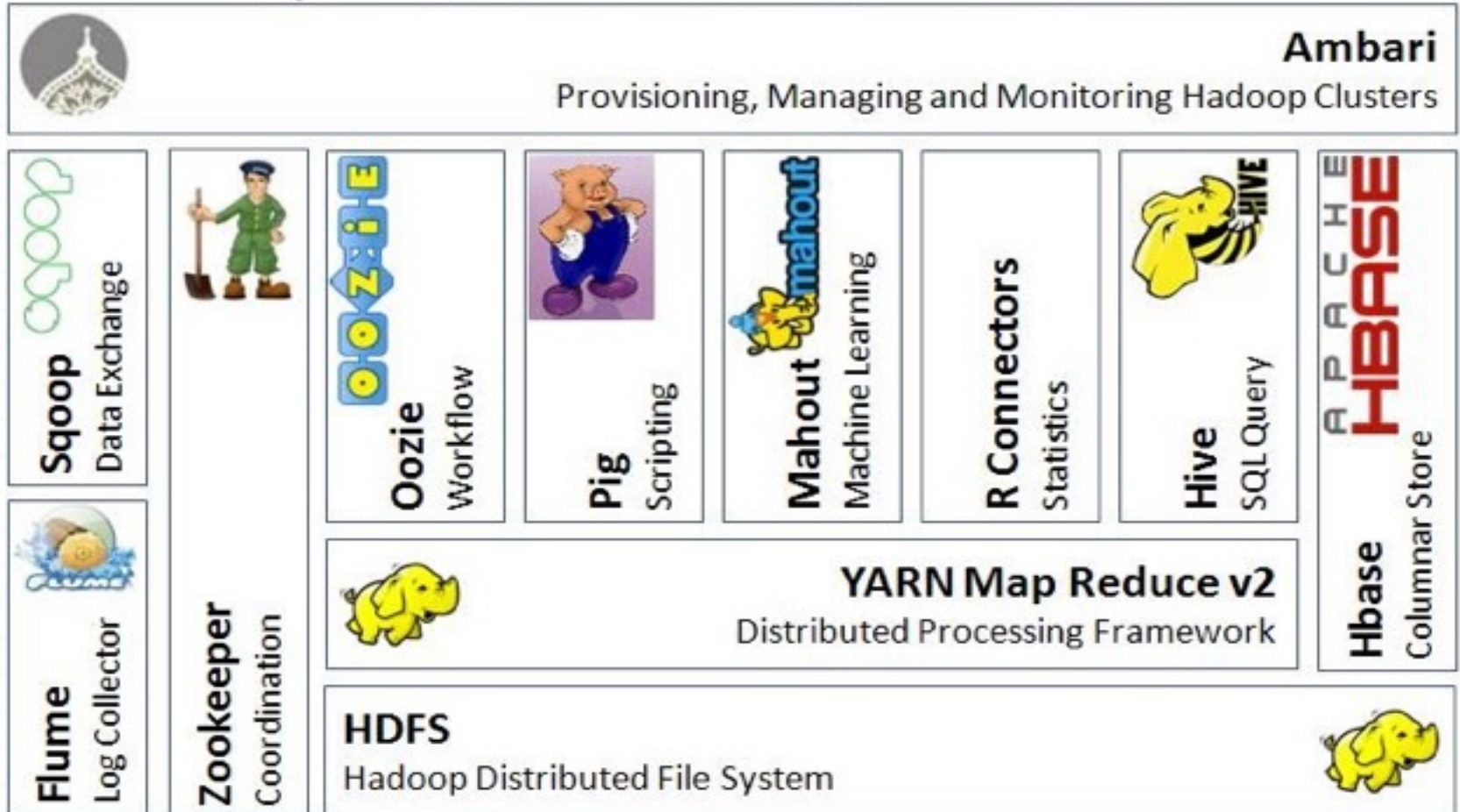
Ecosystem的目的

- 建立於Hadoop 的平行運算及分散式儲存
- 依照目的可以分為:
 - 資料轉移: Sqoop, Flume
 - 讓MapReduce 更簡易使用: Pig, Hive
 - 依附於Hadoop Core 上: Impala, Hbase

Hadoop Ecosystem



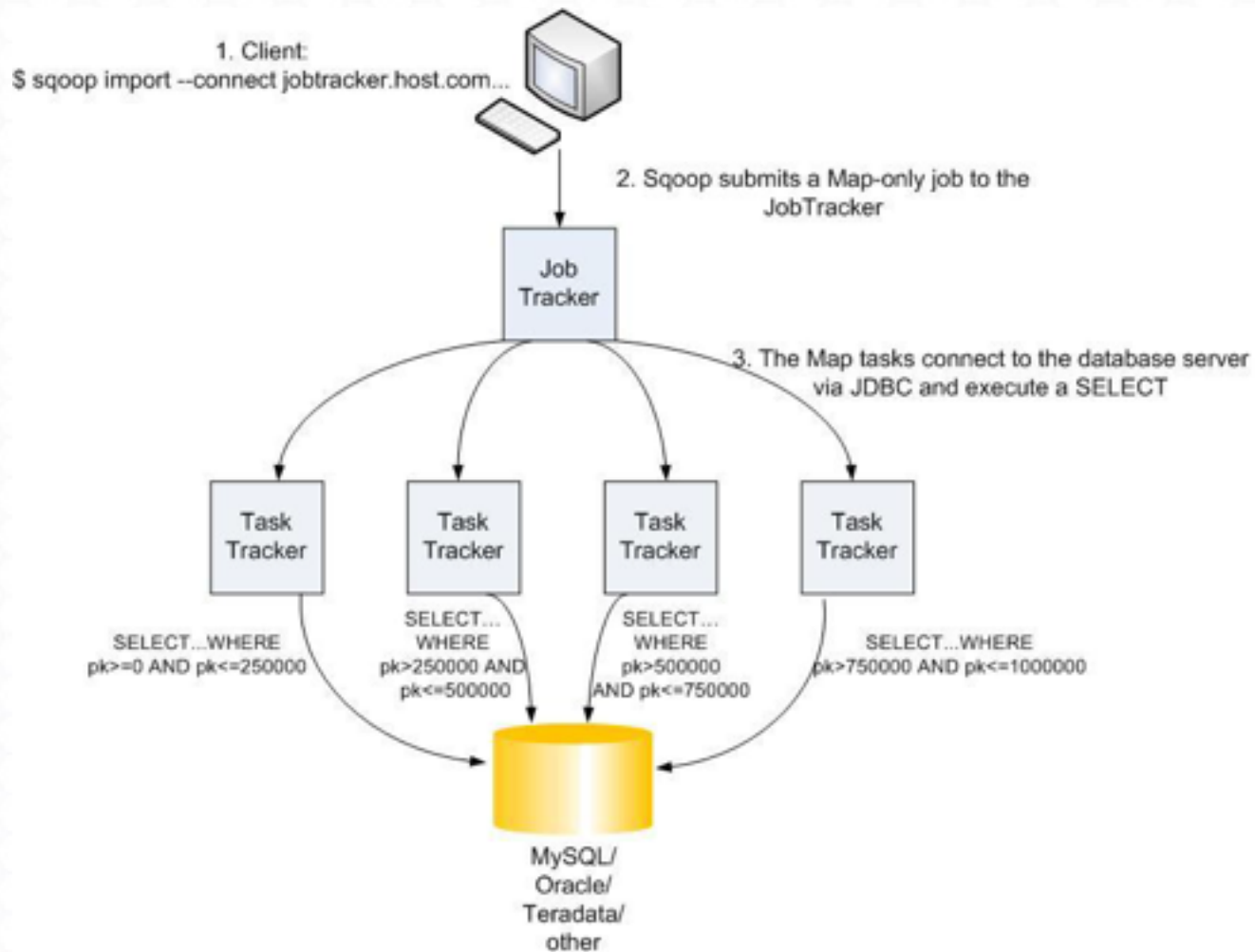
Apache Hadoop Ecosystem



Ecosystem 列表

專案	類型	用途
Hive	資料處理	讓使用者可以用SQL進行MR
Pig	資料處理	讓使用者可以寫腳本語言進行MR
Hbase	NoSQL	生成可延展的NoSQL
Hue	Web 介面	可以讓使用者透過Web 介面操作Hadoop
Sqoop	資料蒐集	讓Hadoop 與資料庫進行資料交換
Flume	資料蒐集	蒐集串流資料
Oozie	工作流程管理	管理 MR 工作與資料轉移腳本
Impala	即時SQL查詢	即時查詢資料

Sqoop 示意圖



轉移結構化資料



ETL



Sqoop Import



Sqoop Export

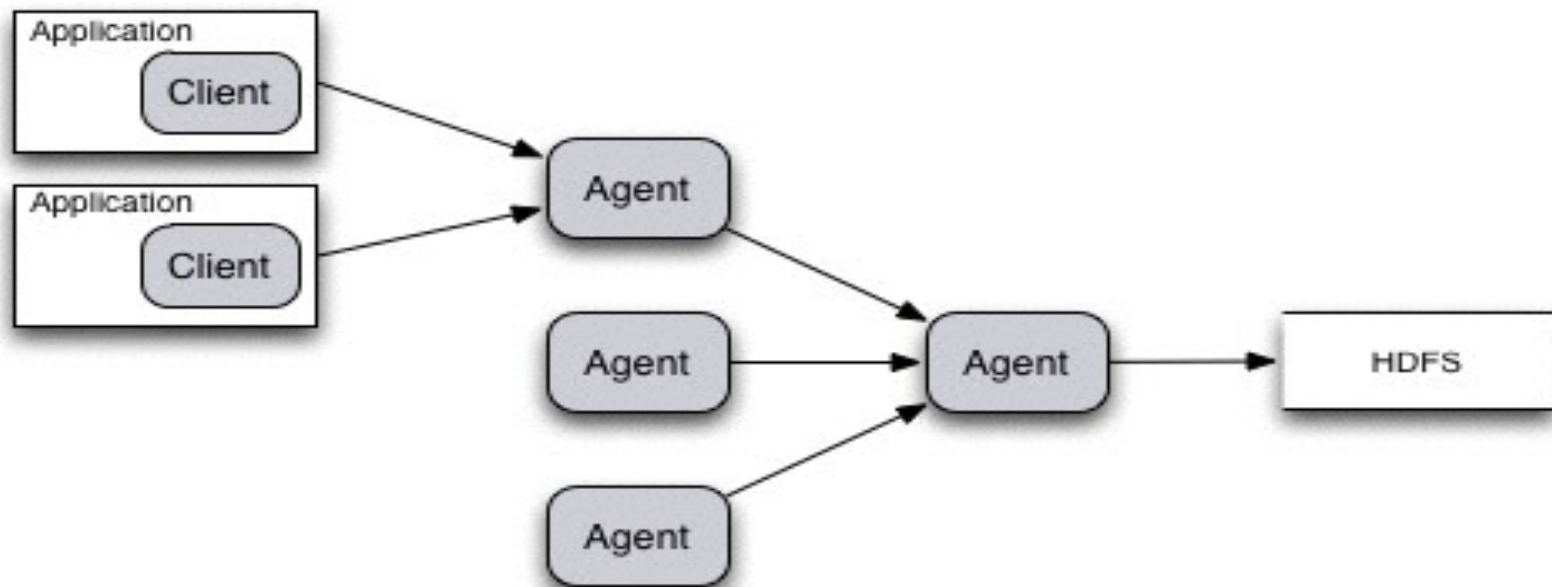
MapReduce

HDFS

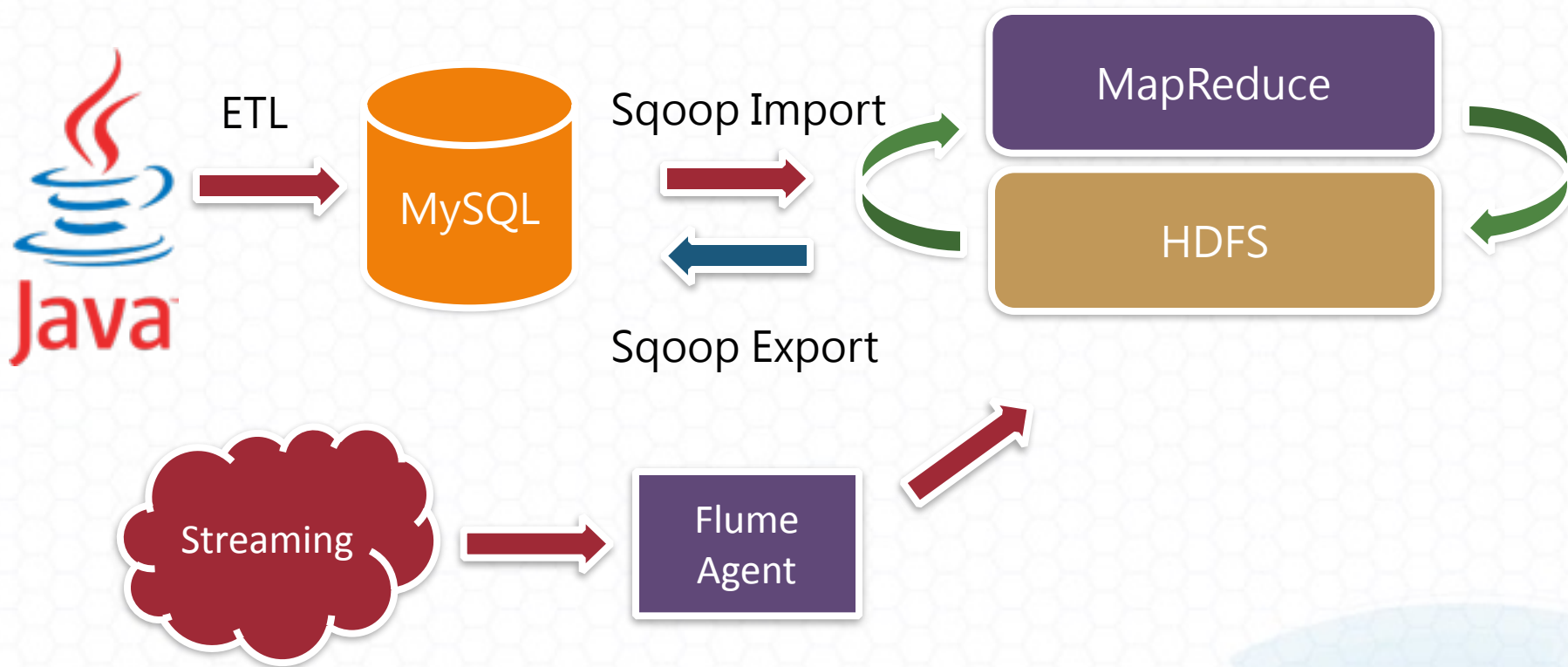
Flume

- 透過代理服務(Agent) 傳輸即時串流資料
- 每台機器通常只有一個代理服務(Agent)
- 可以增添使用者自訂函式做資料去重複或過濾
- 可以水平延展 (Horizontal Scalable)

Flume



轉移非結構化資料



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



Hive

- 可以透過客戶端程式或網頁存取(Hive2)
- Facebook 所開發
- SQL Like 語言
- 需要使用Metastore (例如使用MySQL) 存儲跟HDFS 的對應關係

Pig

- 可以透過客戶端程式或網頁存取
- Yahoo所開發
- Data Flow 語言 – 適合用作ETL
- 使用Pig Interpreter

使用情境

■ Hive

- 當有明確的Schema
- 當資料適合使用SQL執行時

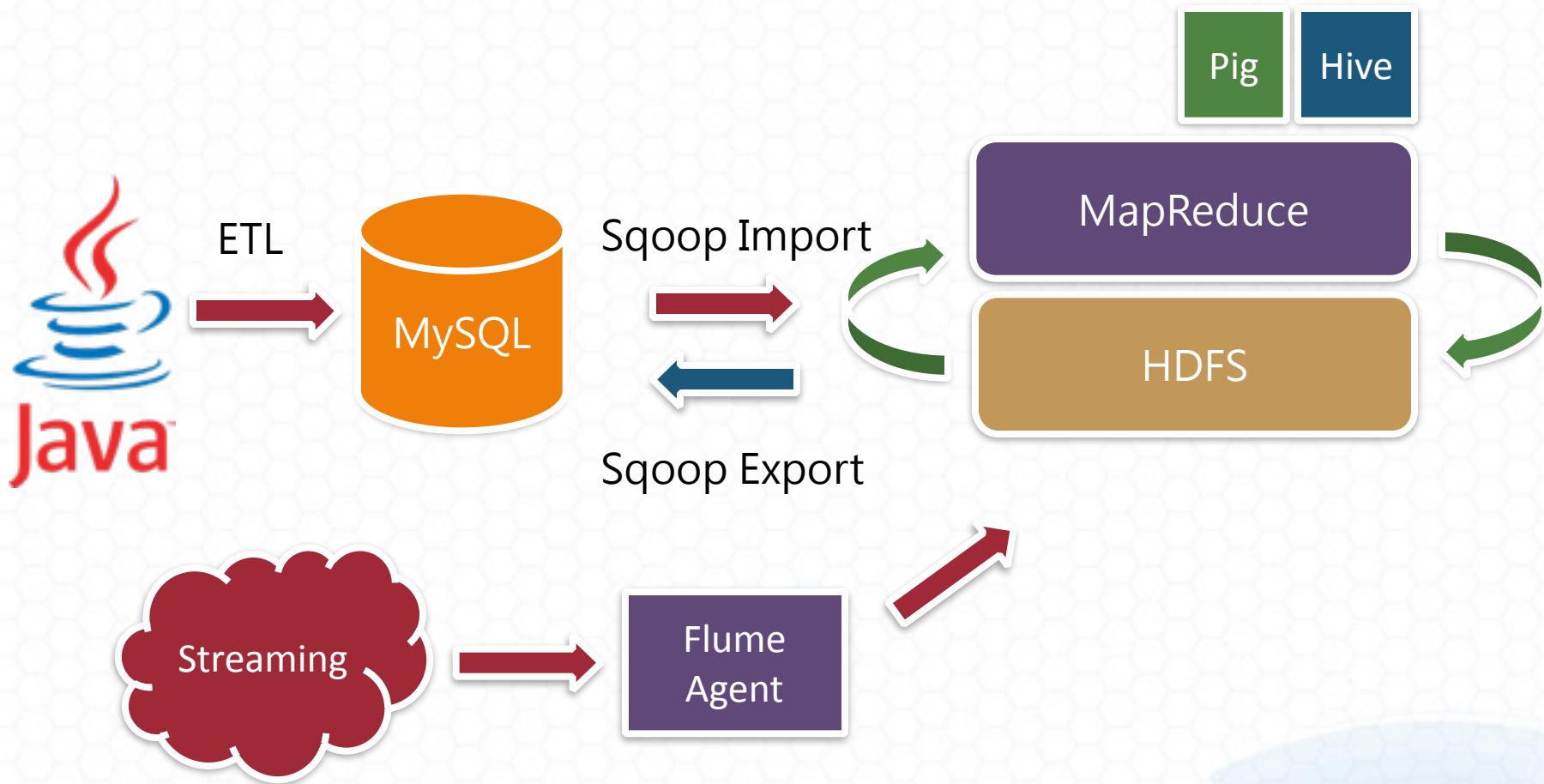
■ Pig

- 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)

Hive v.s. Pig

工具	Hive	Pig
執行模式	<i>MapReduce</i>	<i>MapReduce</i>
可容錯	是	是
執行速度	慢	慢
使用難度	易	高
彈性	SQL 語法	彈性高
學習難度	簡單	數週

處理資料



MapReduce 相當緩慢

- 動輒需要數秒到數分鐘才能產生結果
- OLAP 其他分析系統需要在幾個millisecond 就能得到回應
- Impala 實現了即時查詢

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

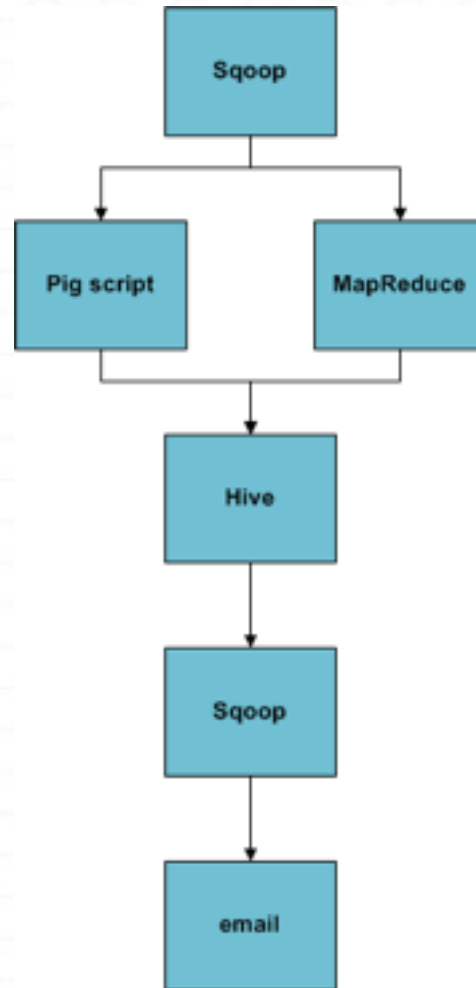
- Cloudera 所開發
- Open Source
- 非可容錯
- 使用情境
 - 有明確的Schema
 - 即時性互動查詢
 - 低容錯性的即時分析

Oozie

- 獨立的Java Web App,可以接受請求而執行流程
- Workflow 是以XML 編寫 (或可透過Hue 編輯)
 - 動作是以Sqoop, Pig, Hive, Java 等程式組成
- Workflow
 - 按需求
 - 定期執行
 - 透過Java API 呼叫
 - 可以監控資料夾是否有資料而執行動作
- 有權限控管(Kerberos及HDFS)



Oozie 使用情境



Mahout

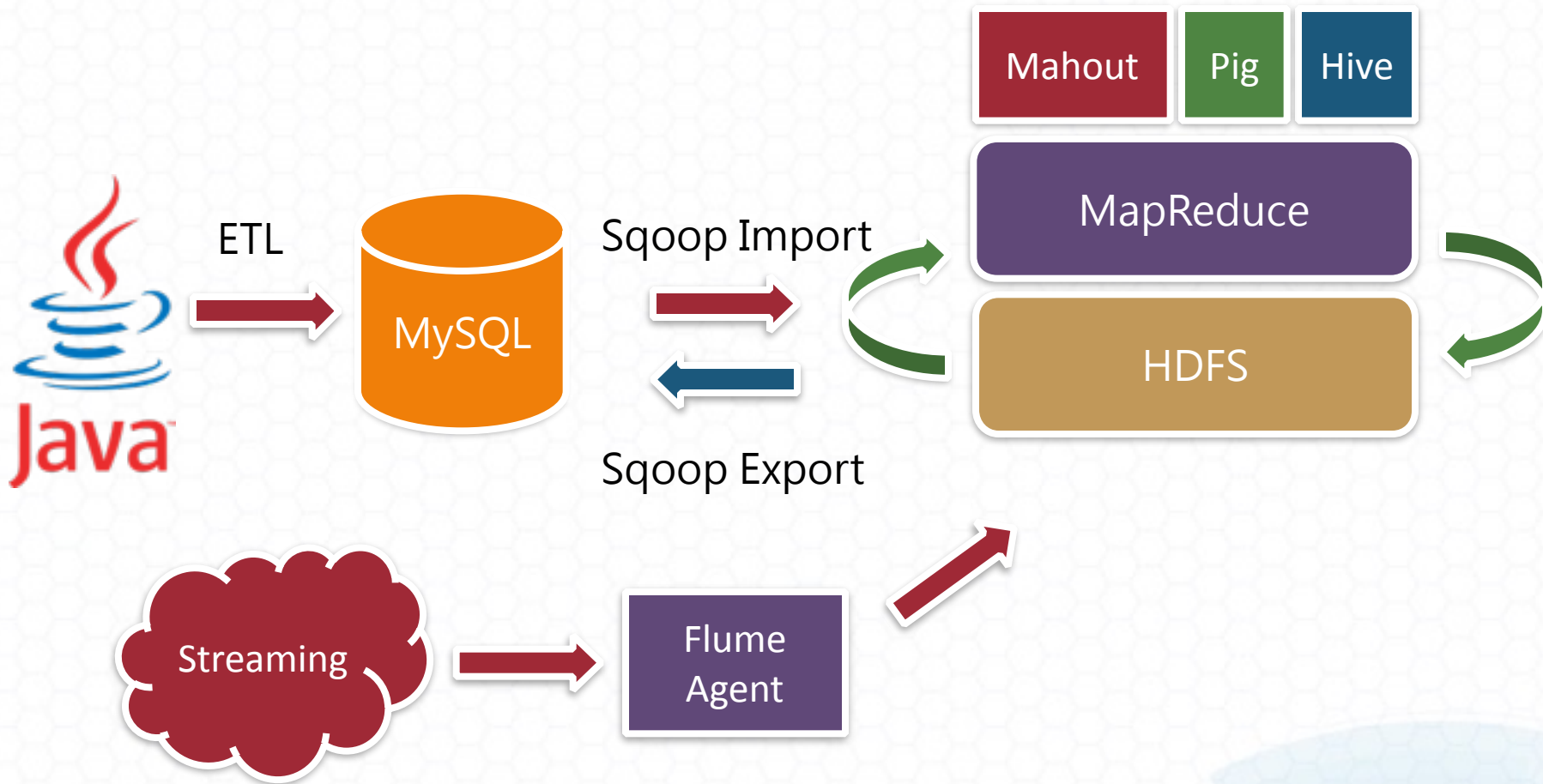
- 可以使用MapReduce 進行機器學習
 - ▣ 但也可以不透過MapReduce 便可執行
- 使用Java 撰寫
- 現行版本 0.9
 - ▣ <http://mahout.apache.org/>



Mahout

- 可以引用Mahout 提供的類別實現機器學習
- 或可使用內建指令進行機器學習
- 提供多個機器學習演算法
 - ▣ 分類演算法 (Classification)
 - ▣ 分群演算法 (Clustering)
 - ▣ 降低維度 (Dimension Reduction)
 - ▣ 迴歸分析 (Regression)
 - ▣ 推薦演算法 (Collaborative Filtering)

產生推薦結果



什麼是Hbase

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

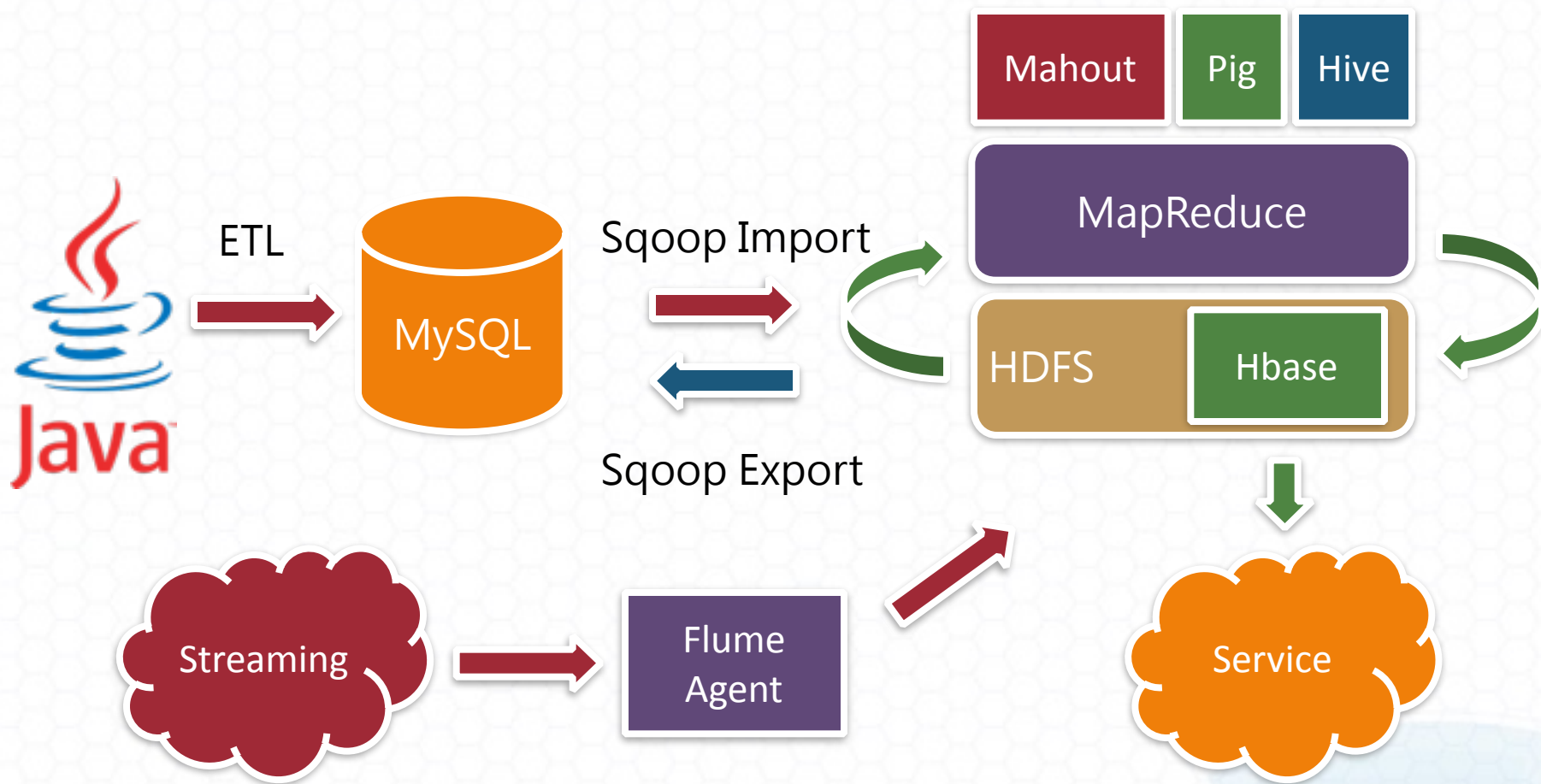


存儲大量的行、列及多版本資料，並能分散到數以萬計的機器中

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

建立推薦系統



操作環境簡介

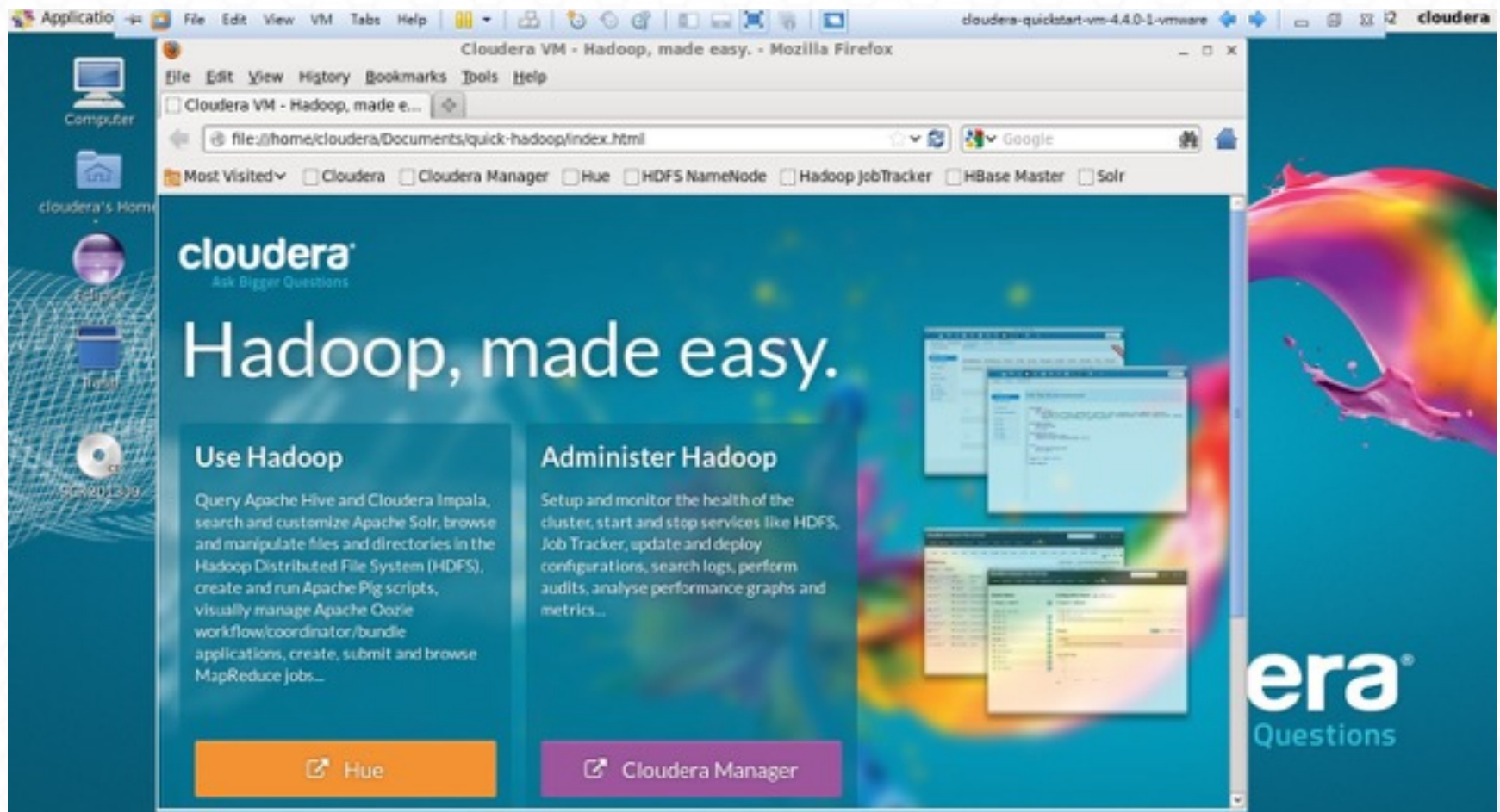
Cloudera QuickStart VM

■ 下載Cloudera QuickStart VM

https://downloads.cloudera.com/demo_vm/vmware/cloudera-quickstart-vm-5.2.0-0-vmware.7z



CDH 5.2



HUE

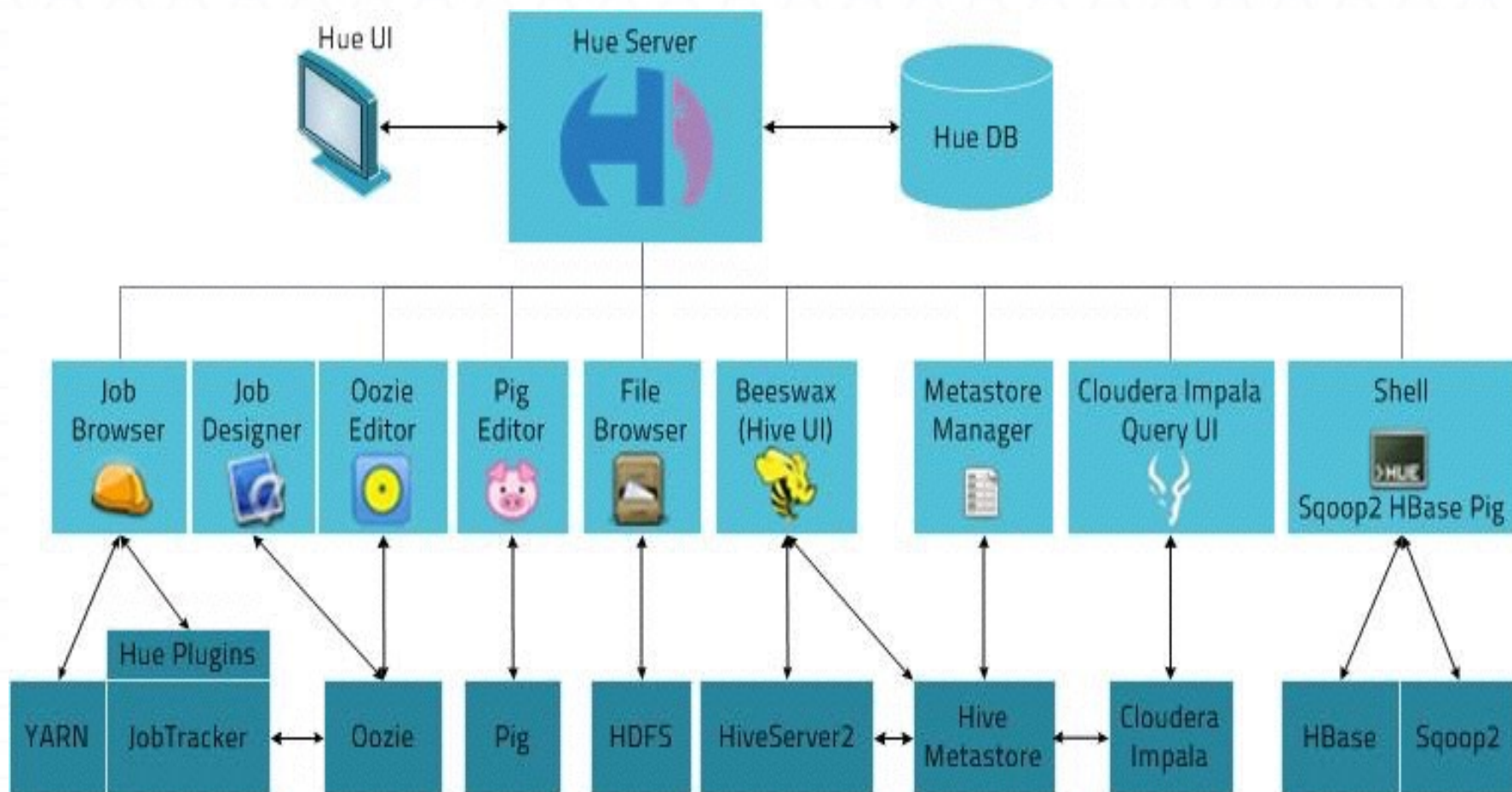
- Hadoop User Experience
- 以單一UI 介面整合所有Hadoop Eco System 不同模組
 - Oozie
 - Hive
 - HDFS



HUE

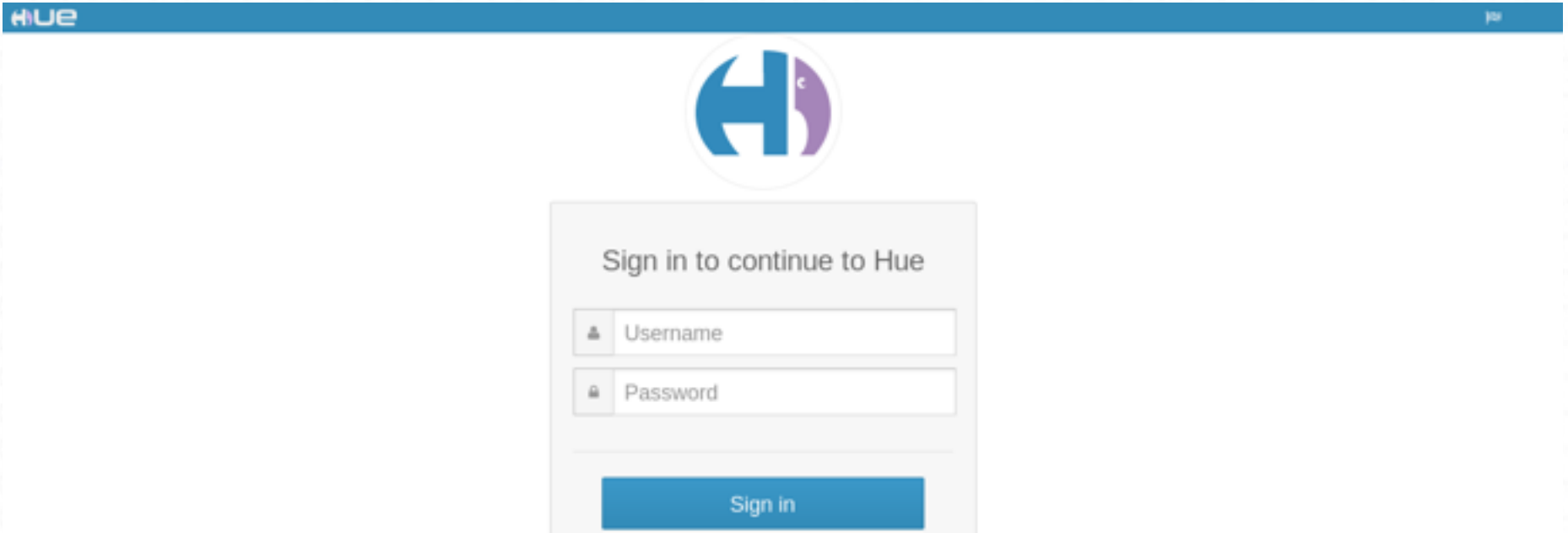
- 由Cloudera 所開發
- 100% 免費跟開源
- 可支援於大部分已知Hadoop 版本
- 提供API

Hue 架構



登入HUE

- Username/Password: cloudera/cloudera
- 步驟請參閱analysis-exercise.txt

The image shows the Hue web interface for login. At the top, there is a blue header bar with the 'HUE' logo on the left and a small 'v3.10' version indicator on the right. Below the header, the main content area is white. In the center, there is a circular logo featuring a stylized 'H' in blue and purple. Below the logo is a light gray rectangular box containing the text 'Sign in to continue to Hue'. Underneath this text are two input fields: the first is labeled 'Username' with a small user icon to its left, and the second is labeled 'Password' with a small lock icon to its left. At the bottom of the box is a blue button with the text 'Sign in' in white.



資料擷取模組 **SQOOP**介紹

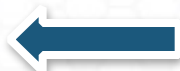
轉移結構化資料



ETL



Sqoop Import



Sqoop Export

MapReduce

HDFS

Sqoop 1

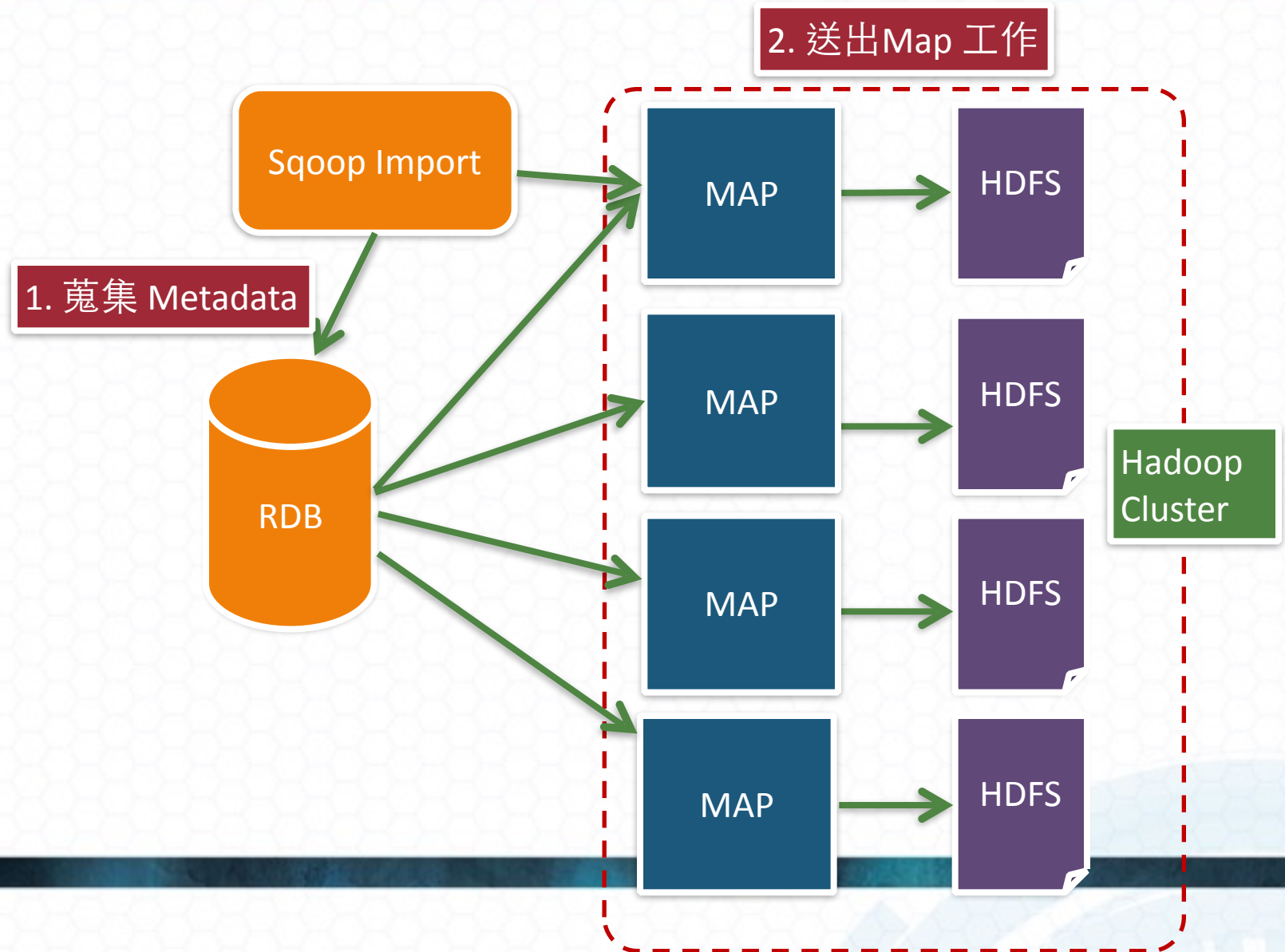
- Sqoop (**SQL** to Had**oop**)
- 可以交換資料庫 與 HDFS 內的資料
 - ▣ 使用JDBC做連結
- 使用MR 的Map 去查詢或新增資料
 - ▣ 預設使用4個Mapper
 - ▣ 可以限制占用頻寬



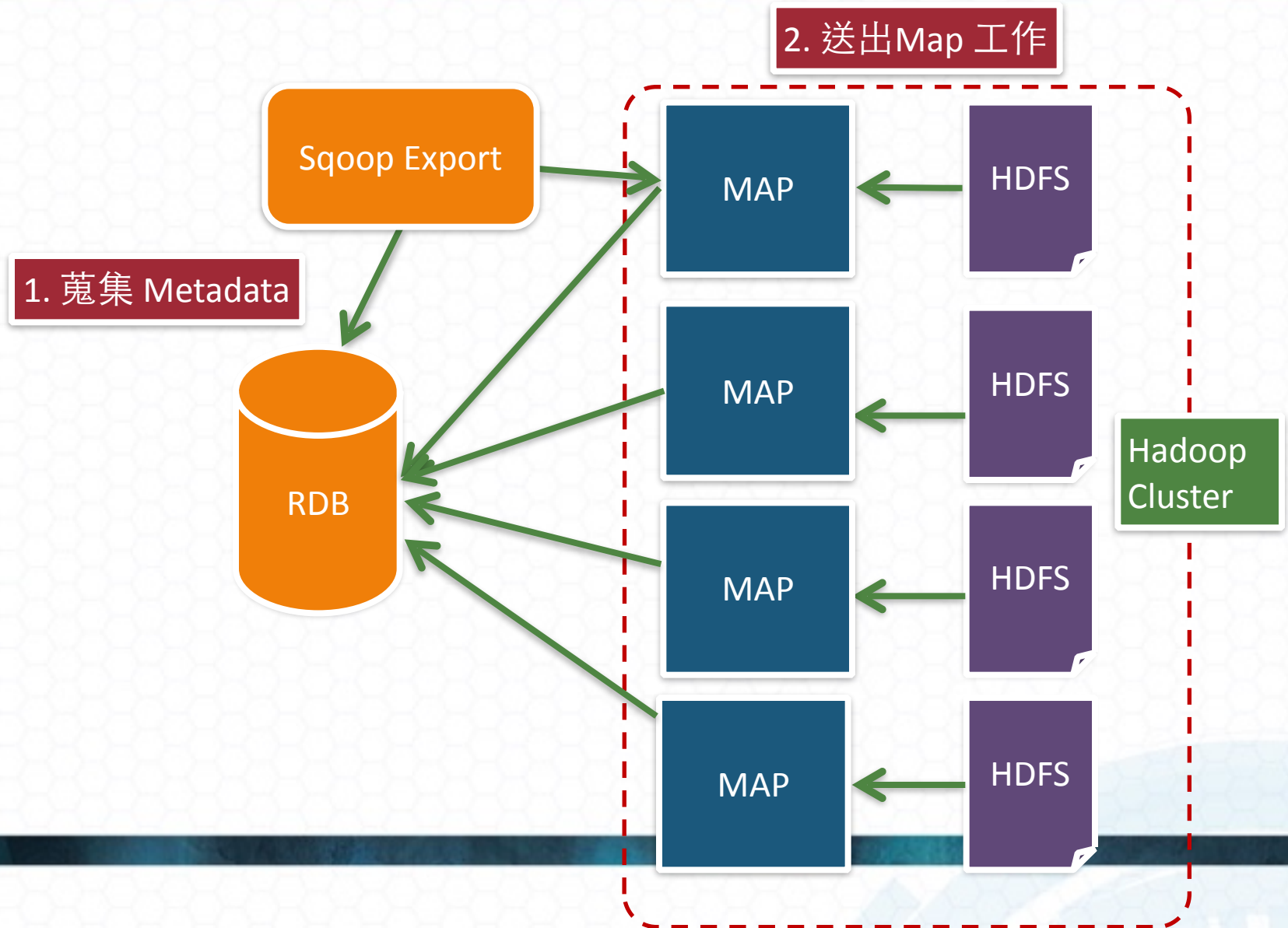
資料庫匯入與匯出

- 不能進行過濾或去重複等動作
- 通常是定期執行匯入與匯出工作
 - ▣ 使用Oozie 做資料同步
 - ▣ 使用情境
 - 定期於MySQL 取得使用者訂單資料
 - 進行MR Job 運行推薦系統
 - 將推薦系統結果匯回MySQL

Sqoop Import



Sqoop Export



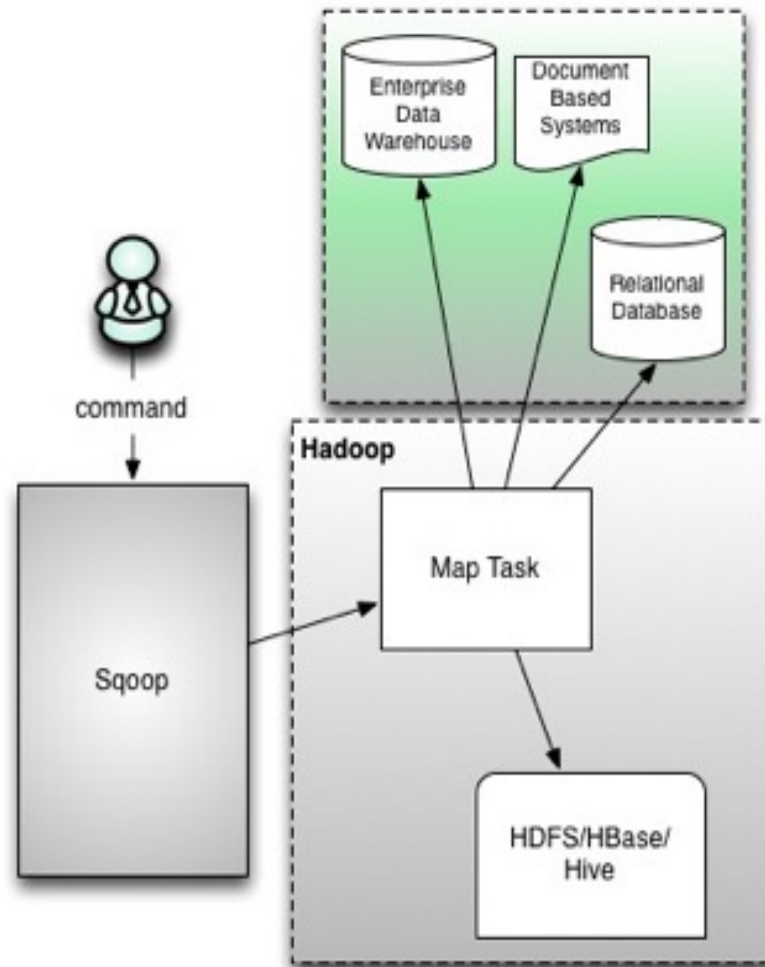
Sqoop 進階功能

- 支援Incremental Updates (整數, Timestamp)
- 可以使用SQL 限制匯入的欄與列
- 可以自動產生Hive Metastore Table或 Hbase 資料列
- 可以匯入 CSV 或 Avro 格式檔案
- 支援大型物件 (Blob, Clob)
- 支援部分資料庫檢視語句 (Show, List)

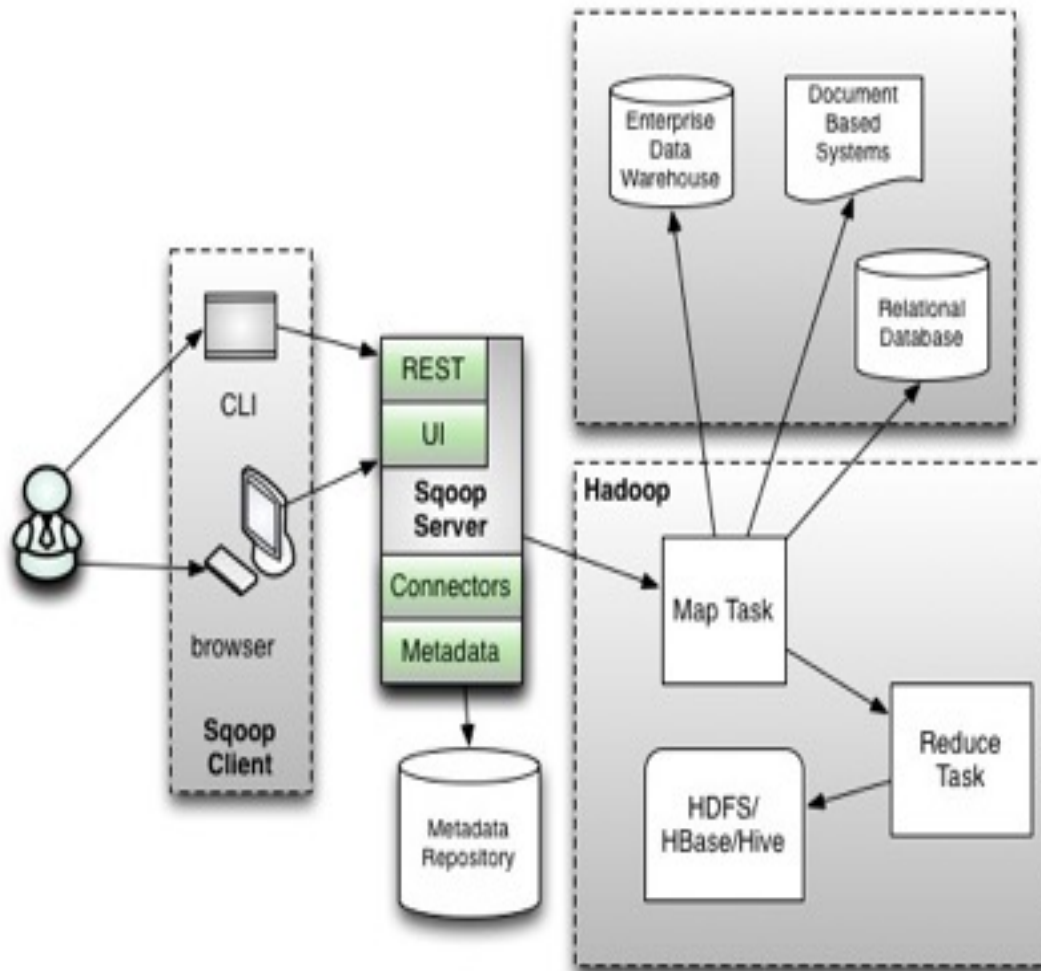
Sqoop v.s. Sqoop2

- Sqoop1
 - Client Model (Connector 跟 Driver 都裝在 Client 上)
 - 只送出"Map"工作
 - 1.4.5版
- Sqoop2
 - Service Model (Connector跟Driver都裝在Server 上)
 - Mapper 轉移資料，Reducer 轉換資料
 - 提供更佳的安全性
 - 1.99.X 版

Sqoop1 架構

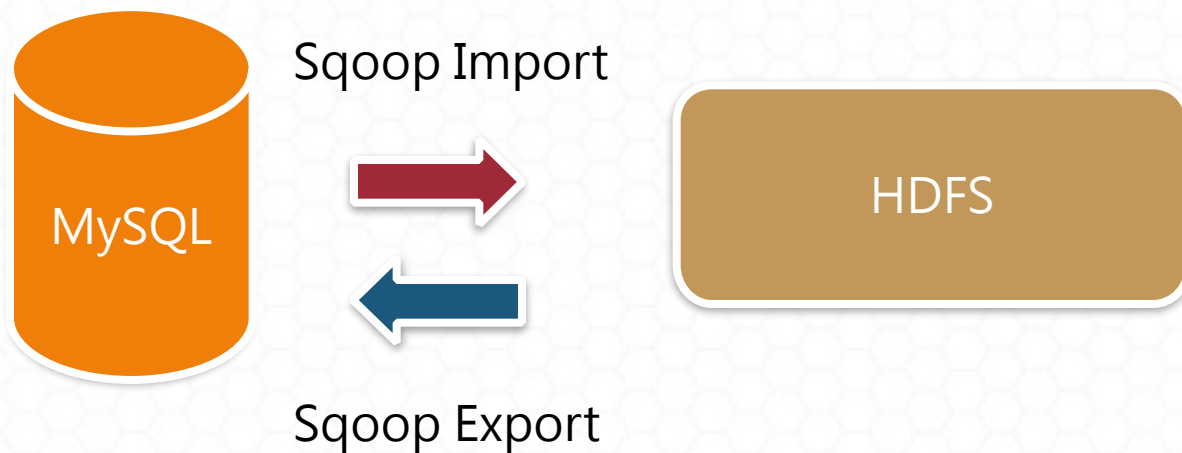


Sqoop2 架構



Lab: SQOOP應用情境

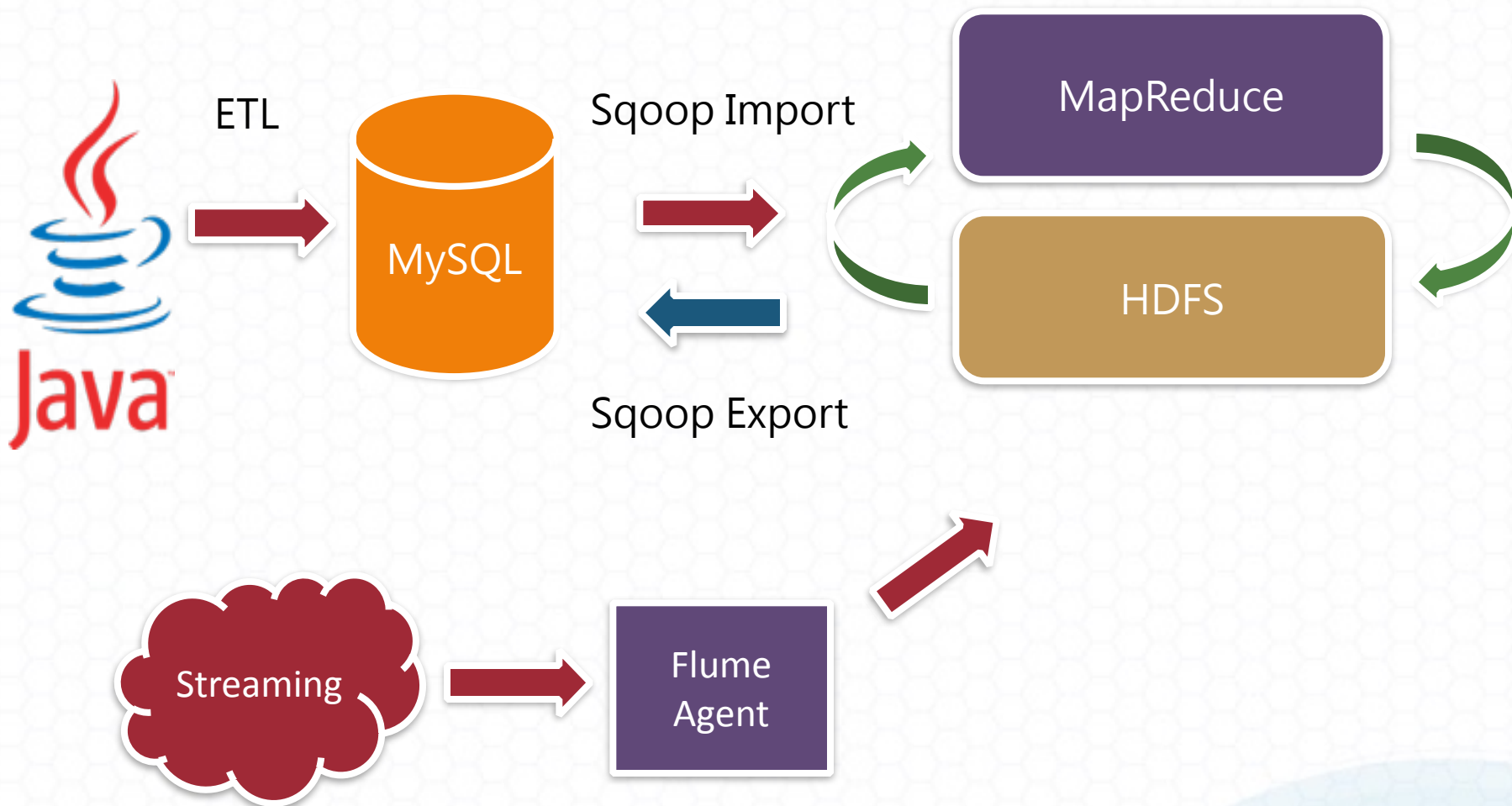
- 將使用者評分(Ratings)匯入HDFS之中
- 步驟請參閱[sqoop-exercise.txt](#)





資料擷取模組 **FLUME**介紹

分析情境

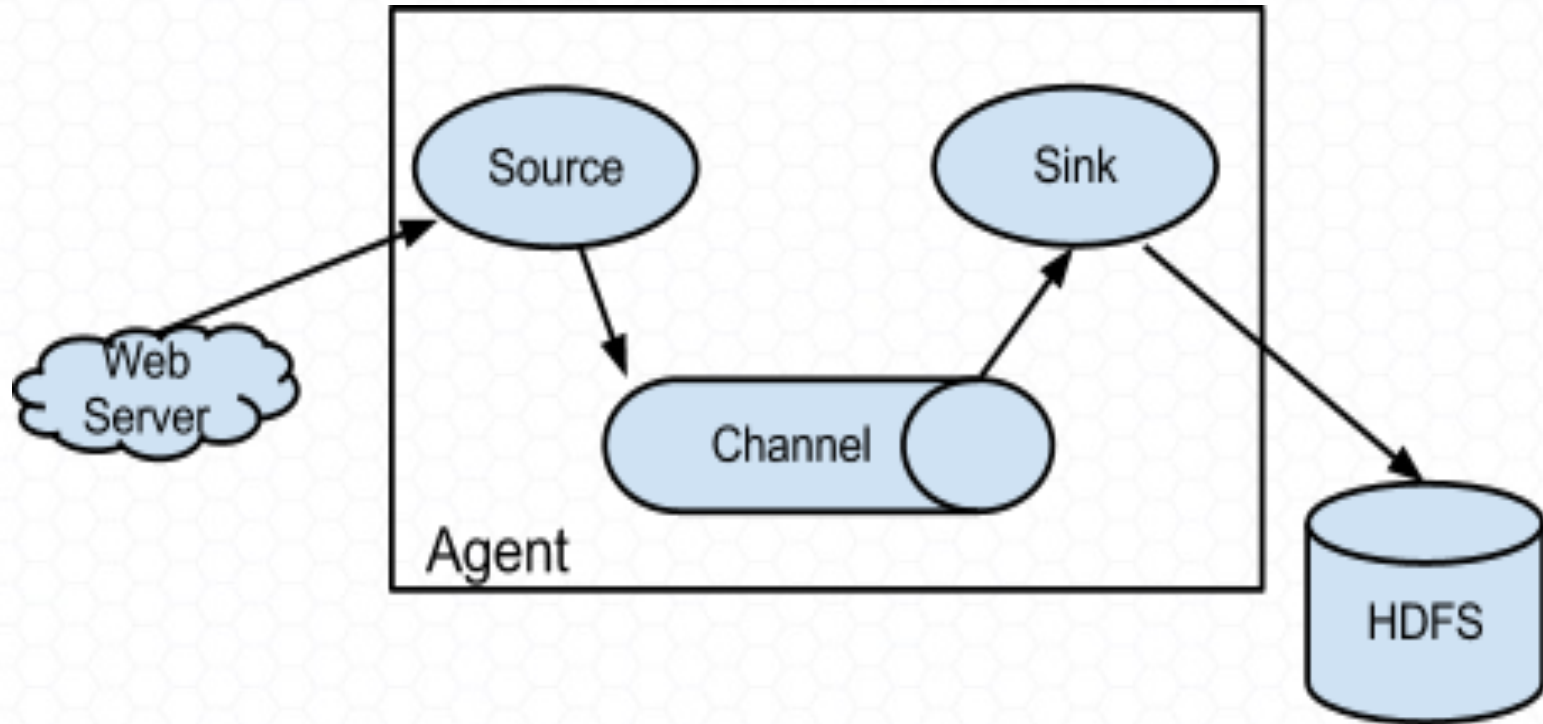


FLUME

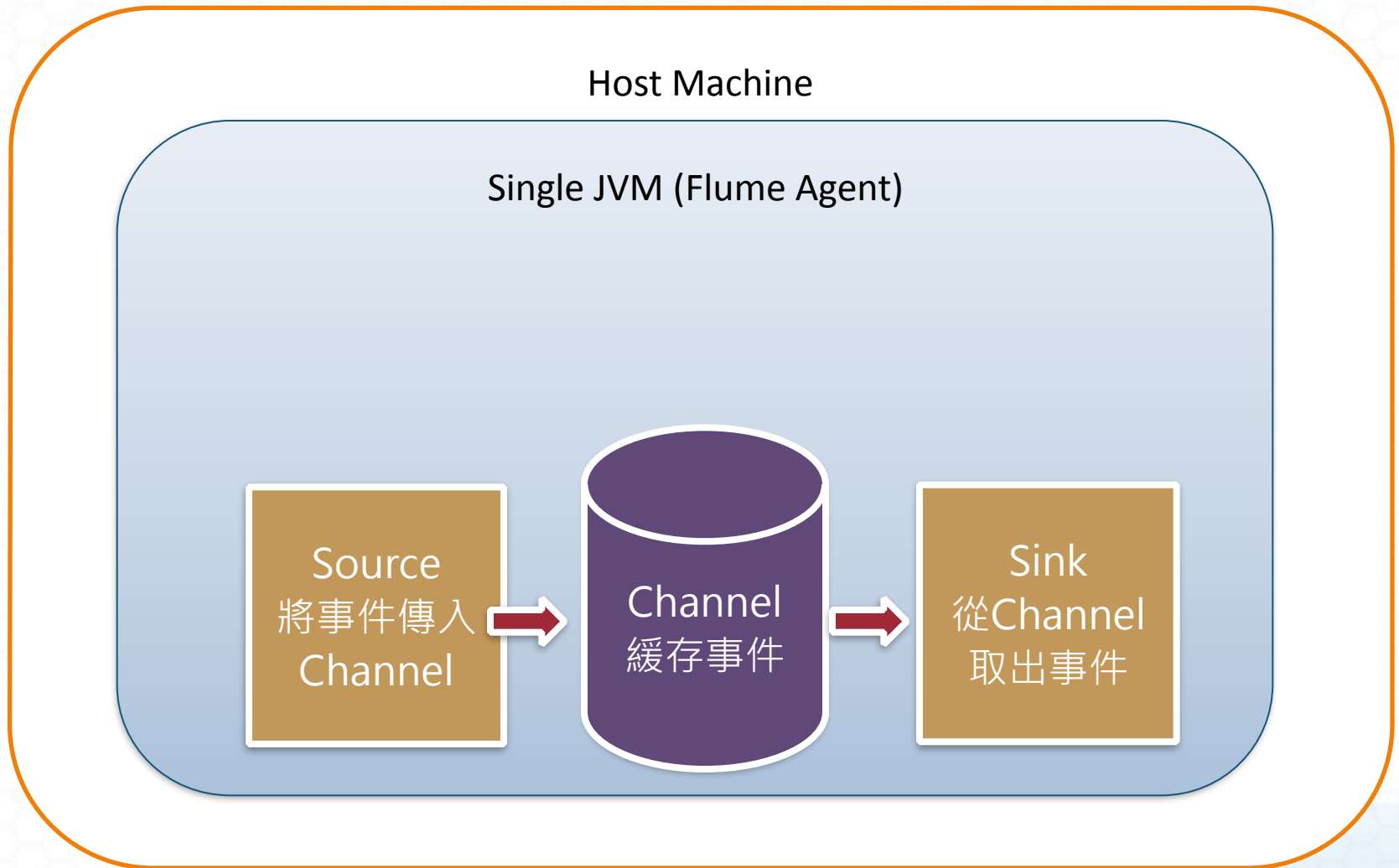
- 從不同來源蒐集(Collect)、聚合(Aggregate)資料串流(Streaming)
 - ▣ 以Log 資料為主
- 每個Flume 的執行程式皆命名為Agent
- Agent 皆包含
 - ▣ Source (讀取Log File 來源)
 - ▣ Sink (將資料放置於HDFS)
 - ▣ Channel (Source 跟 Sink 間的緩衝)



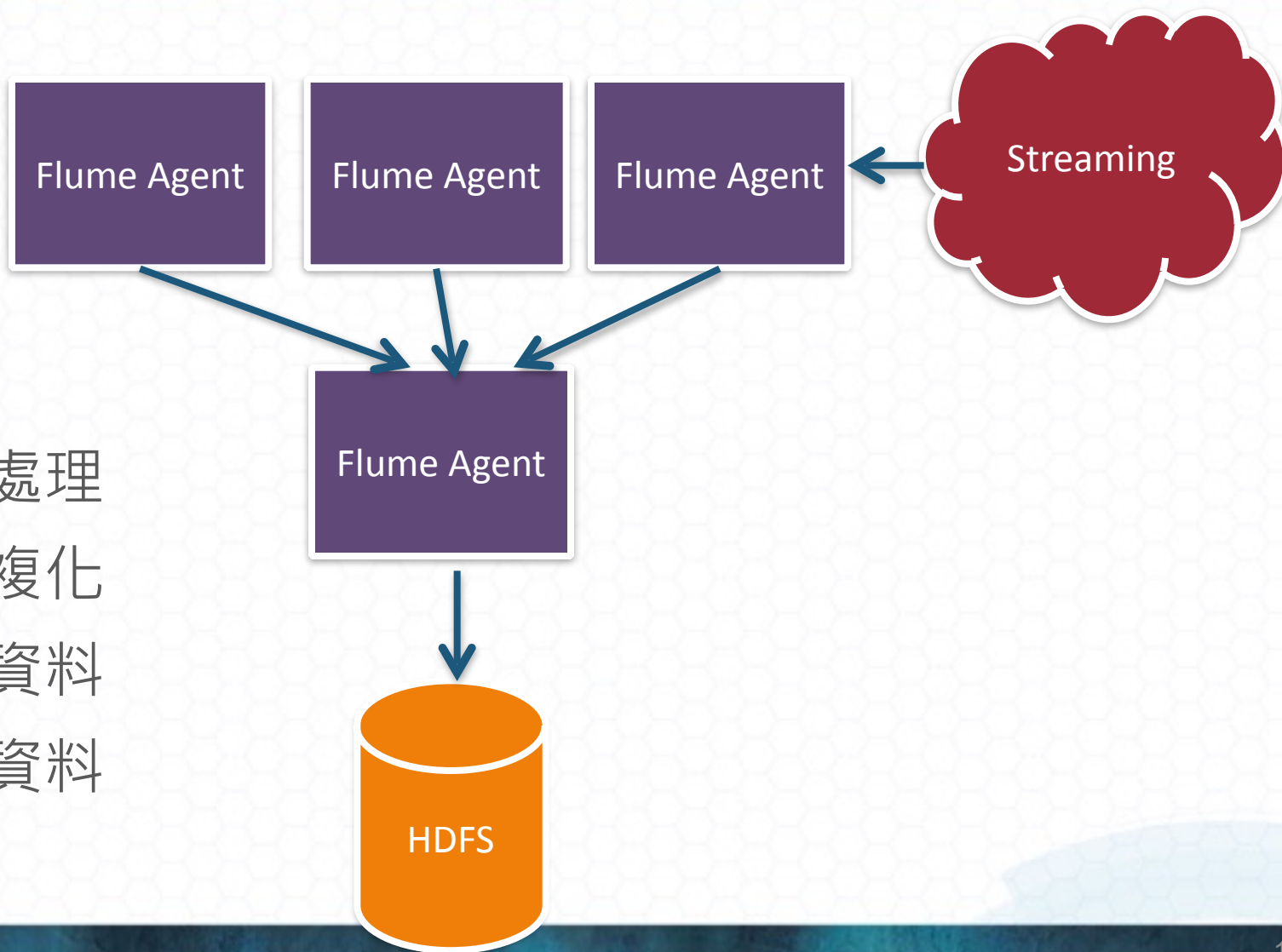
Flume 簡易範例



單一Flume 內容



Flume 架構



- 加值處理
- 去重複化
- 清理資料
- 過濾資料
- ...

Source, Sink, Channel

- Source: 取得資料
 - Event Driven 或 Polling based
 - 從特定檔案取得資料 (tail -f file)
 - 從特定連結取得資料 (192.1.1.10:8888)
- Sink: 存儲資料
 - Polling
 - Localhost:9988
 - hdfs://
- Channel: Source 與 Sink 之間的橋樑
 - Memory, Disc, JDBC

Source 設定

名稱	類型	簡介
avro source	avro	<ul style="list-style-type: none">• 提供一個以avro為協議的服務，並bind到某個port上，等待avro協定用戶端發過來的訊息• 一般在agent之間傳輸資料時，可以配置
thrift	thrift	<ul style="list-style-type: none">• 使用Thrift做為傳輸協議
exec source	exec	<ul style="list-style-type: none">• 執行一個unix 指令，以stdout為資料來源 命令可以通過 <agent>.<source>.command配置， 如：
netcat source	netcat	<ul style="list-style-type: none">• 監控特定Port，每一行作為一個事件的傳輸；• 假使輸入的資料為text，每一行資料最大長度（max-line-length）預設為512

Source 設定

名稱	類型	簡介
http source	http	<ul style="list-style-type: none">支援http的post和get
scribe source	org.apache.flume. source.scribe.Scrib	<ul style="list-style-type: none">支援 scribe
syslog source	syslogtcp syslogudp	<ul style="list-style-type: none">監聽syslog，支援tcp以及udp
sequence source	seq	<ul style="list-style-type: none">自動產生編號
spooling directory source		<ul style="list-style-type: none">監控某個目錄下的所有檔，將其新加入的檔作為資料來源傳輸每傳輸一個檔後，會被rename成其他名字（表示已經傳輸過）或者刪掉預設監控目錄下的檔具有： immutable, uniquely named
jms source	jms	<ul style="list-style-type: none">從訊息佇列獲取資料。active message queue

Channel 設定

名稱	類型	簡介
memory channel	memory	<ul style="list-style-type: none">訊息放在記憶體中，提供高吞吐量(High Throughput)，但容錯性低(Not Robust)
file channel	file	<ul style="list-style-type: none">對資料容錯性高(Highly Robust)；但是配置較為複雜，需要配置資料目錄和checkpoint目錄不同的file channel均需要配置一個checkpoint 目錄
jdbc channel	jdbc	<ul style="list-style-type: none">內置的derby資料庫，對event進行了高容錯性(Highly Robust)可用來取代file channel

Sink 設定

名稱	類型	簡介
hdfs sink	hdfs	<ul style="list-style-type: none">• 將數據寫到hdfs上• 可以配置目錄（支援日期格式）
logger sink	logger	<ul style="list-style-type: none">• 採用logger，logger可以配置輸出到控制台，也可輸出到檔案• logger對訊息長度有限制，超過限制會截
avro sink	avro	<ul style="list-style-type: none">• 發送給另外一個avro的source
thrift sink	thrift	<ul style="list-style-type: none">• 發送給另外一個thrift的source
IRC sink	irc	<ul style="list-style-type: none">• Internet Relay Chat
file roll sink	file_roll	<ul style="list-style-type: none">• 本地檔案，支援檔案輪替(rotate)• 可設定大小、時間、事件計數來進行輪替
null sink	null	<ul style="list-style-type: none">• 丟棄事件

Sink 設定

名稱	類型	簡介
hbase sink	Hbase asynchbase	<ul style="list-style-type: none">寫到Hbase中；需要配Hbase的table，columnFamily
morphline solr sink	org.apache.flume.sink.solr.morphline.MorphlineSolrSink	<ul style="list-style-type: none">將資料傳遞至Solr
elastic search sink	org.apache.flume.sink.elasticsearch.ElasticSearchSink	<ul style="list-style-type: none">將資料傳遞至Elastic Search
custom sink	\$FQCN	<ul style="list-style-type: none">寫到特定FQCN

Flume Interceptor

- Flume 的擴充元件
- 可以在資料即時傳輸的過程中更改資料
- 實作 `org.apache.flume.interceptor.Interceptor` 介面

現有的**Flume Interceptor**

- **Timestamp Interceptor**
 - 於事件標頭上寫入時間標籤
- **Host Interceptor**
 - 於事件標頭上寫入Agent 來源
- **Static Interceptor**
 - 使用者可以在標頭上增添任意值
- **UUID Interceptor**
 - 使用者可以為每個事件加註UUID

現有的**Flume Interceptor**

- Morphline Interceptor
 - ▣ 可以透過正規表達法過濾或修改事件
- Regex Filtering Interceptor
 - ▣ 以正規表達式過濾事件
- Regex Extractor Interceptor
 - ▣ 用正規表達式抽取符合樣式的資訊

HA & 可擴充性

- 具交易性(Transactional)
 - ▣ 使用ack 確定下游(downstream)有收到資料，否則上游(upstream)會重送資料
- HA 及平行擴充
 - ▣ Hot / Standby Mode
 - ▣ 可以平行擴充成多台上游(Upstream)蒐集資料

資料加值功能

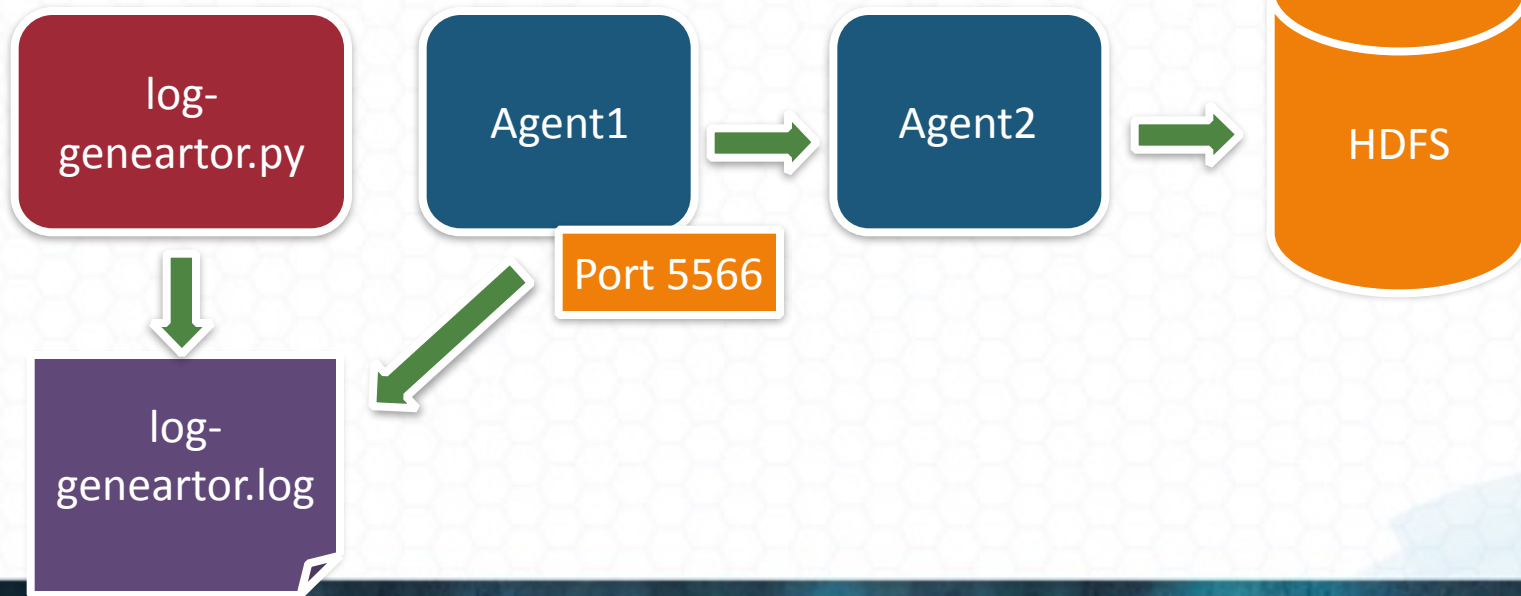
- 可以擴充新的Source, Sinks, Channel
- 可以在Source, Sinks 之間增加資料處理功能
 - ▣ Interceptor
 - ▣ 正規表達式
 - ▣ 客製的擴充腳本 (必須是Java)

維護Flume

- 試想有上千個Agent 同時啟動？該如何管理？
- 使用Deployment 或Config 管理工具
 - Puppet
 - Chef
- Config 會隨著property 的變更即時更改
 - 不用重啟Agent

Lab: Flume 實戰

建立兩支Flume Agent (agent1, agent2) , agent1 蒐集由log-generator.py 產生出的/tmp/log-generator.log , 並透由agent2 將/tmp/log-generator.log的內容放置於HDFS (/flume) 目錄中



將網站流量導入HDFS

- 啟動log-generator.py
 - `python log-generator.py`
- 啟動Flume 1 & 2
 - `flume-ng agent --conf-file agent1.properties --name agent1`
 - `flume-ng agent --conf-file agent2.properties --name agent2`
- 步驟請參閱flume-exercise.txt

HANDS ON

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

聯絡方式

- Website:

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

- Email:

- 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