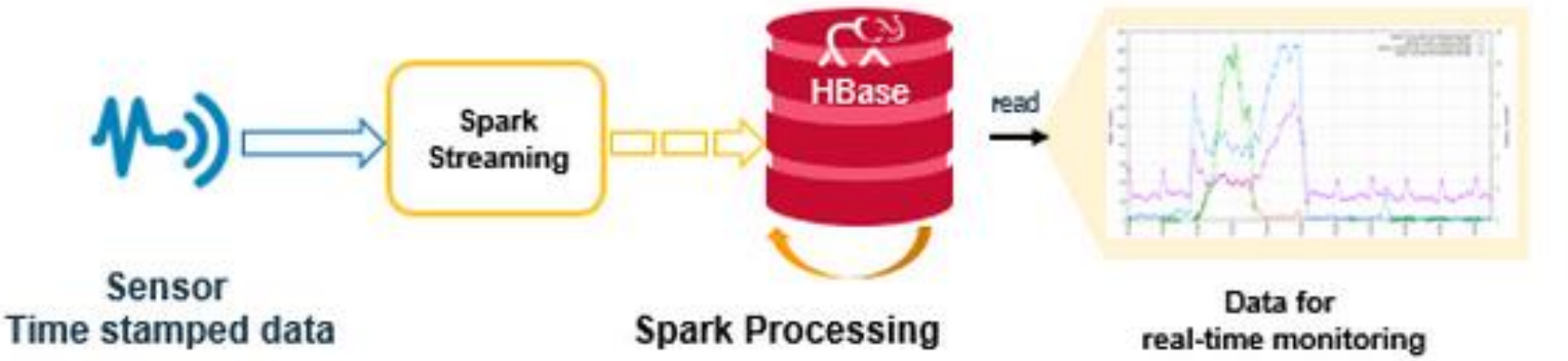
巨量資料實戰課程(II) – Kafka 與 HBase 簡介

David Chiu

2016/12/08

系統架構

如何建立一個即時監控儀表板



建構目標

- 啟動一個Kafka Producer 不斷遞送串流資料
- 啟用一個Spark Streaming 工作，(Consume)消化Producer 遞送過來的Streaming Log，並且將資料存進HBase
- 聚合(Aggregate) Hbase 的資料，並將結果呈現到儀表板上

Kafka

為何要使用Messaging 系統(I)

- Decoupling
  - 確保當資料來源與輸出源變更時，還是可以使用統一介面傳遞資料
- Redundancy
  - Message Queue可以持久化資料直到它們已經被完全處理，避免資料丟失風險
- 可擴展性
  - 增加訊息處理的頻率只要另外增加處理機器即可

為何要使用Messaging 系統(II)

- **峰值處理能力**
  - 使用訊息佇列能夠支持突發的請求，而不會因為突發的請求讓系統崩潰
- **可恢復性**
  - 當一部分元件失效，不會影響整個系統。訊息佇列降低了進程間的耦合度，所以即使一個處理訊息的進程掛掉，加入佇列中的訊息仍然可以在系統恢復後被處理
- **送達保證**
  - 保證了消息能被實際的處理。

為何要使用Messaging 系統(III)

- **順序保證**
  - 訊息佇列本來就是排序的，並且能保證資料會按照特定的順序來處理
- **緩沖**
  - 訊息佇列通過緩衝層來幫助任務的讀寫速率可以不一致，助於控制和優化資料流經過系統的速度
- **理解資料流程**
  - 訊息佇列通過消息被處理的頻率，可以輔助確定那些地方的資料流程表現不佳。
- **非同步處理**
  - 訊息佇列提供了非同步處理機制，允許你把一個消息放入佇列，但並不立即處理它

Messaging Queue 種類

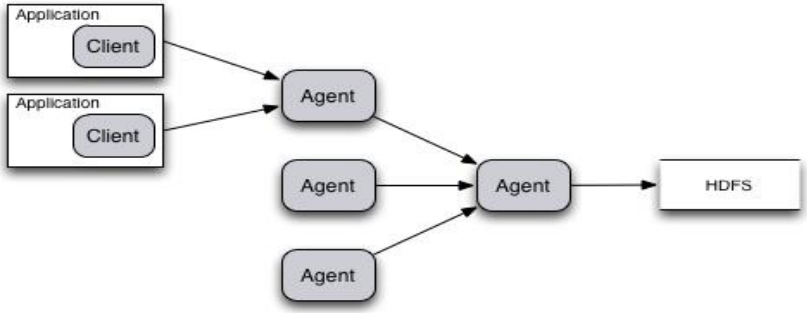
- **RabbitMQ**
  - RabbitMQ是使用Erlang編寫的一個開源的訊息佇列，本身支援很多的協定：AMQP，XMPP, SMTP, STOMP
- **Redis**
  - Key-Value對的NoSQL資料庫，支援MQ功能，可以當做一個羽量級的佇列服務來使用
- **ZeroMQ**
  - ZeroMQ號稱最快的訊息佇列系統，尤其針對大輸送量的需求場景

Kafka

- 分散式的訊息系統
- 作為多種類型的數據管道和訊息系統使用
- 目標是處理即時數據提供一個統一、高通量、低延遲的系統

Flume v.s. Kafka

- Flume
  - 將資料送至HDFS
  - 適合用作Log 分析

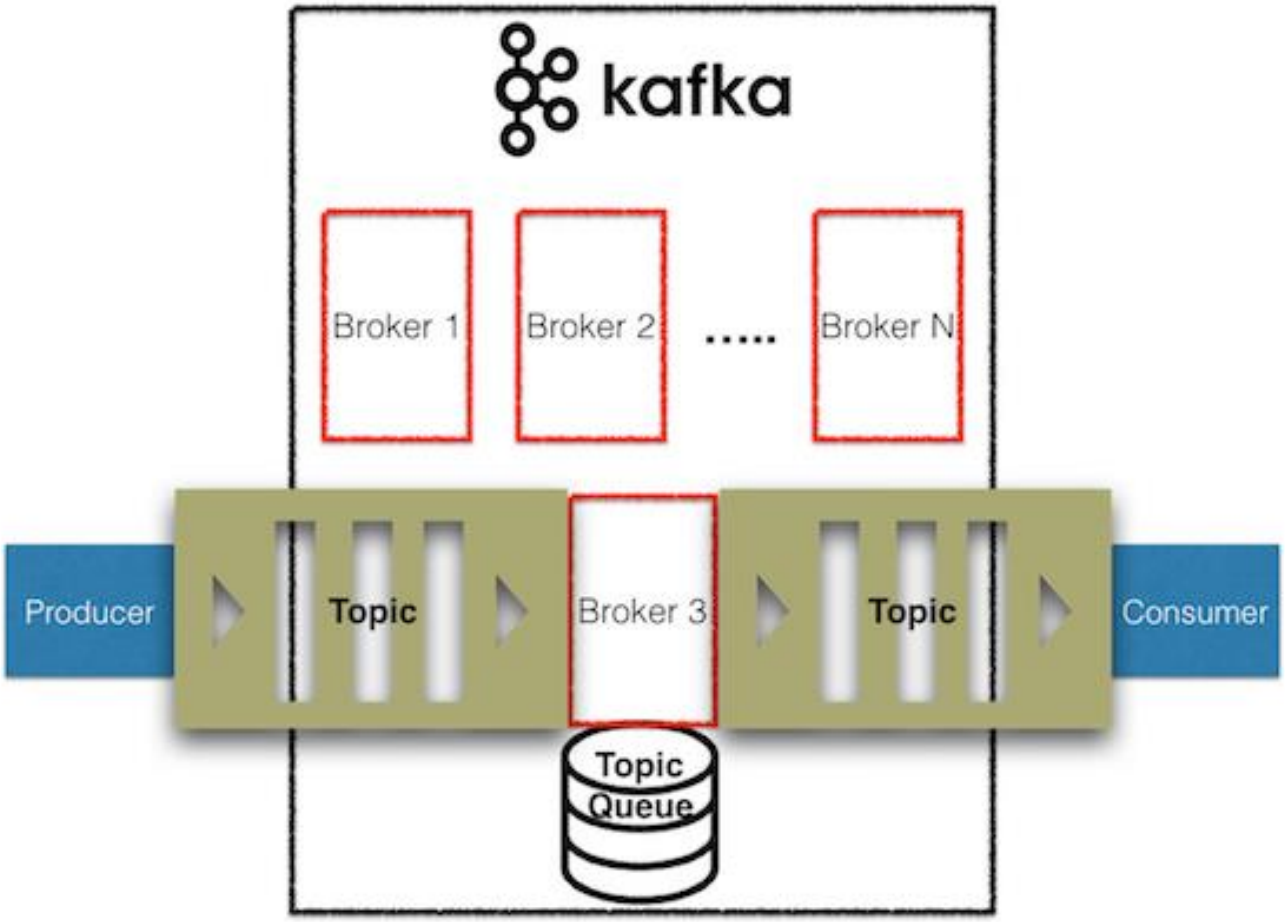


- Kafka
  - 分析串流資料
  - 可將資料留存做後續分析 (Kafka)
  - 適合用在即時分析系統

Kafka 優點

- 以時間複雜度為O(1)的方式提供訊息持久化能力，即使TB級以上資料也能保證訪問性能
- 高吞吐率。即使在非常廉價的商用機器上也能做到單機支持每秒100K訊息的傳輸
- 支持Kafka Server間的Partition，及分散式Consume，同時保證每個partition內的消息順序傳輸
- 支援離線資料處理和即時資料處理

Kafka 系統架構


kafka
Broker 1
Broker 2
.....
Broker N
Producer
Topic
Broker 3
Topic
Consumer
Topic
Queue

Push & Pull

- Kafka由producer向broker push消息並由consumer從broker pull消息
- push模式很難適應消費速率不同的消費者，因為消息發送速率是由broker決定的
  - push模式的目標是盡可能以最快速度傳遞消息，但是這樣很容易造成consumer來不及處理消息
  - pull模式則可以根據consumer的消費能力以適當的速率消費消息。

Kafka 系統元件

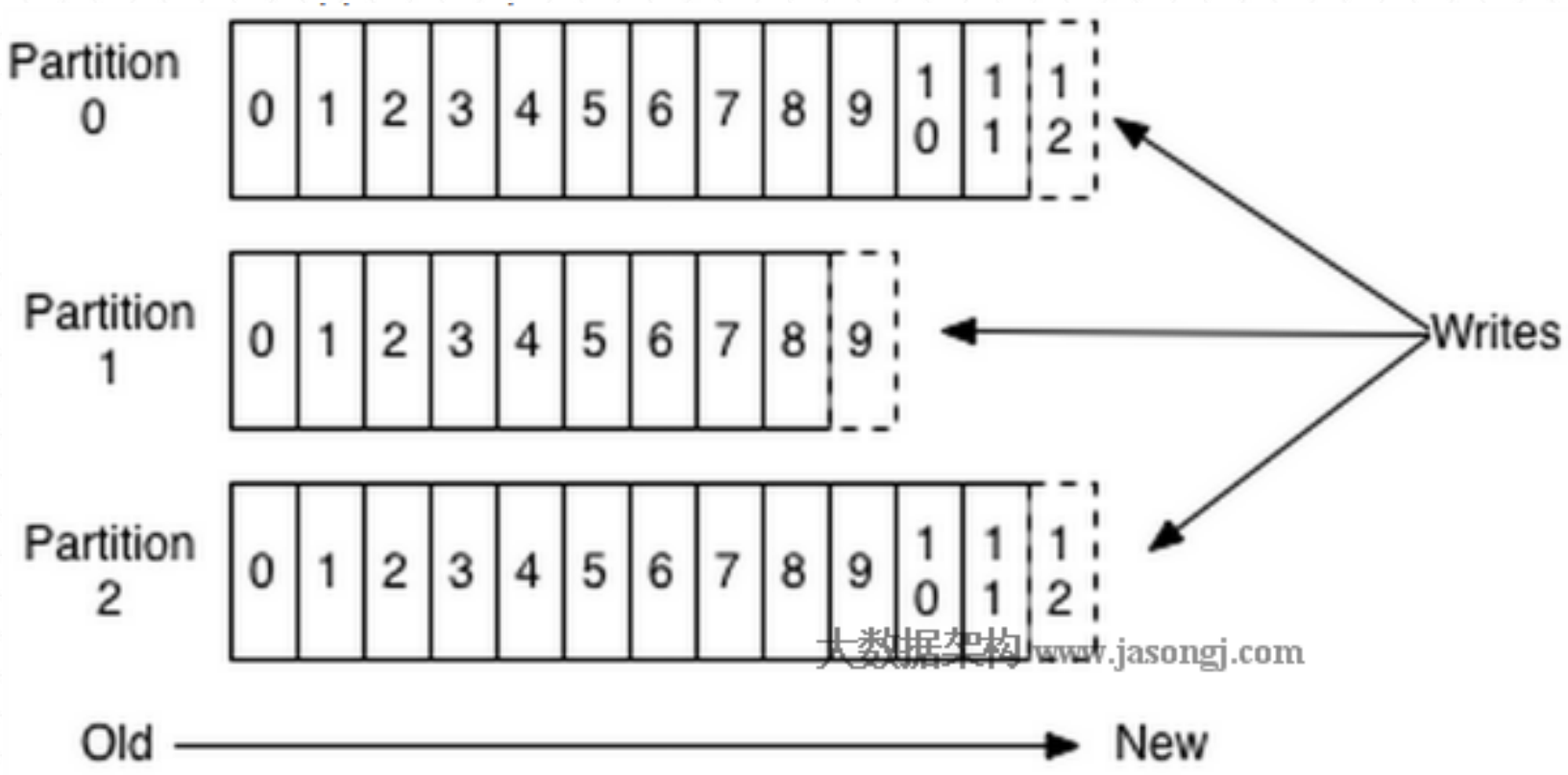
- Broker : Kafka 採用叢集的架構，由一至多台的機器所組成，而叢集內每一台機器被稱之為一個 Kafka 的 Broker
- Topic : 放訊息類別的通道(Queue)
- Partition : 每一個 Topic 內可以切分成多個 Partition
- Producer : 負責發布消息到 Kafka Topic 的角色
- Consumer : 負責接收/訂閱 Kafka Topic 內的訊息做處理
- Zookeeper：每個 Broker 中的訊息同步及叢集資源配置是透過 Zookeeper 來達成。

Topics v.s. Partition

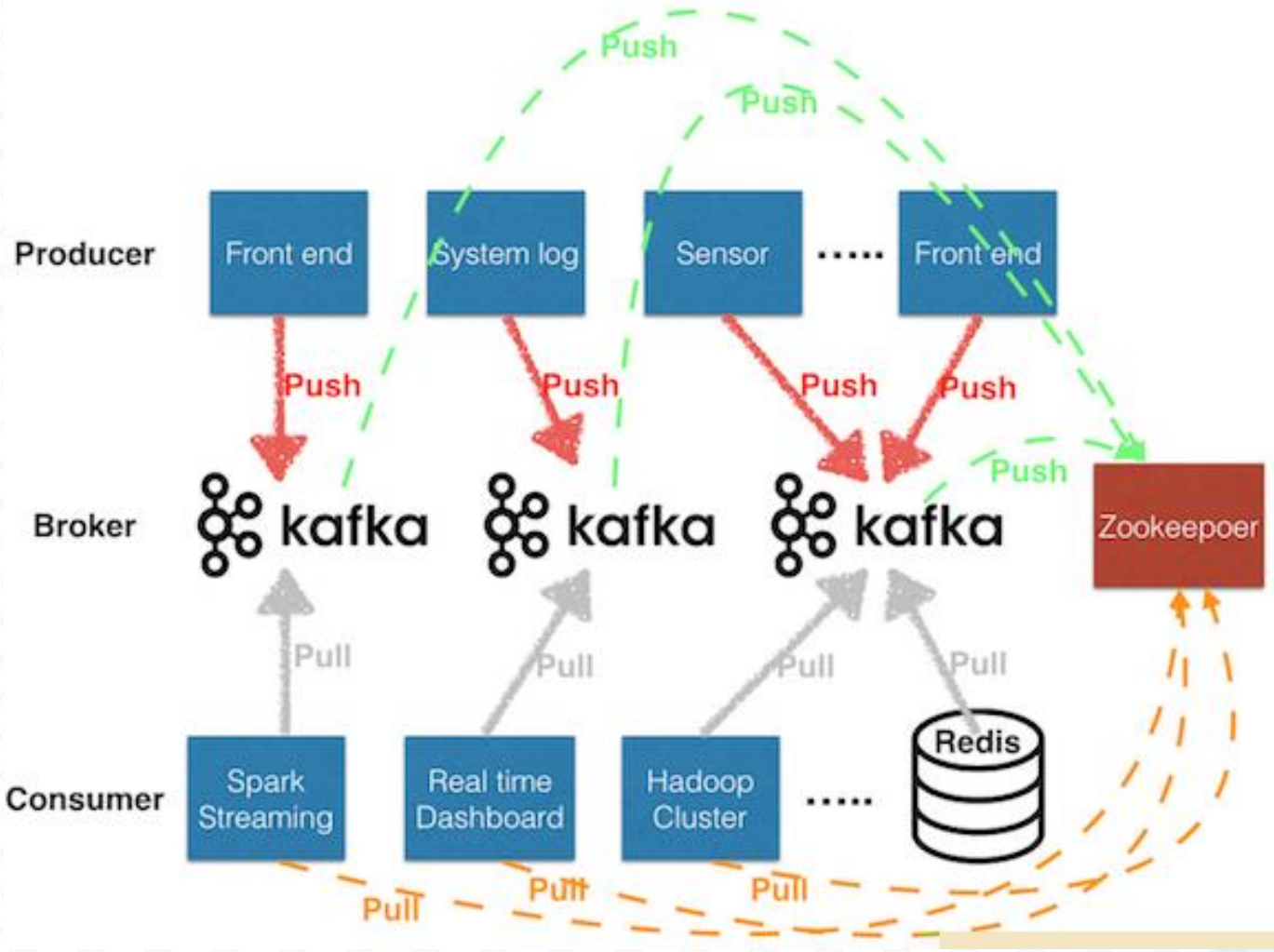
- Topic在邏輯上可以被認為是一個queue
- 每條消費都必須指定它的topic ,可以理解為必須指明把這條消息放進哪個queue裡
- 為了使得Kafka可以水平擴展，Kafka把topic分成一個或多個partition，每個partition在物理上對應一個資料夾，該資料夾下存儲這個partition的所有消息和索引檔

Topics v.s. Partition

- 採取循序寫入

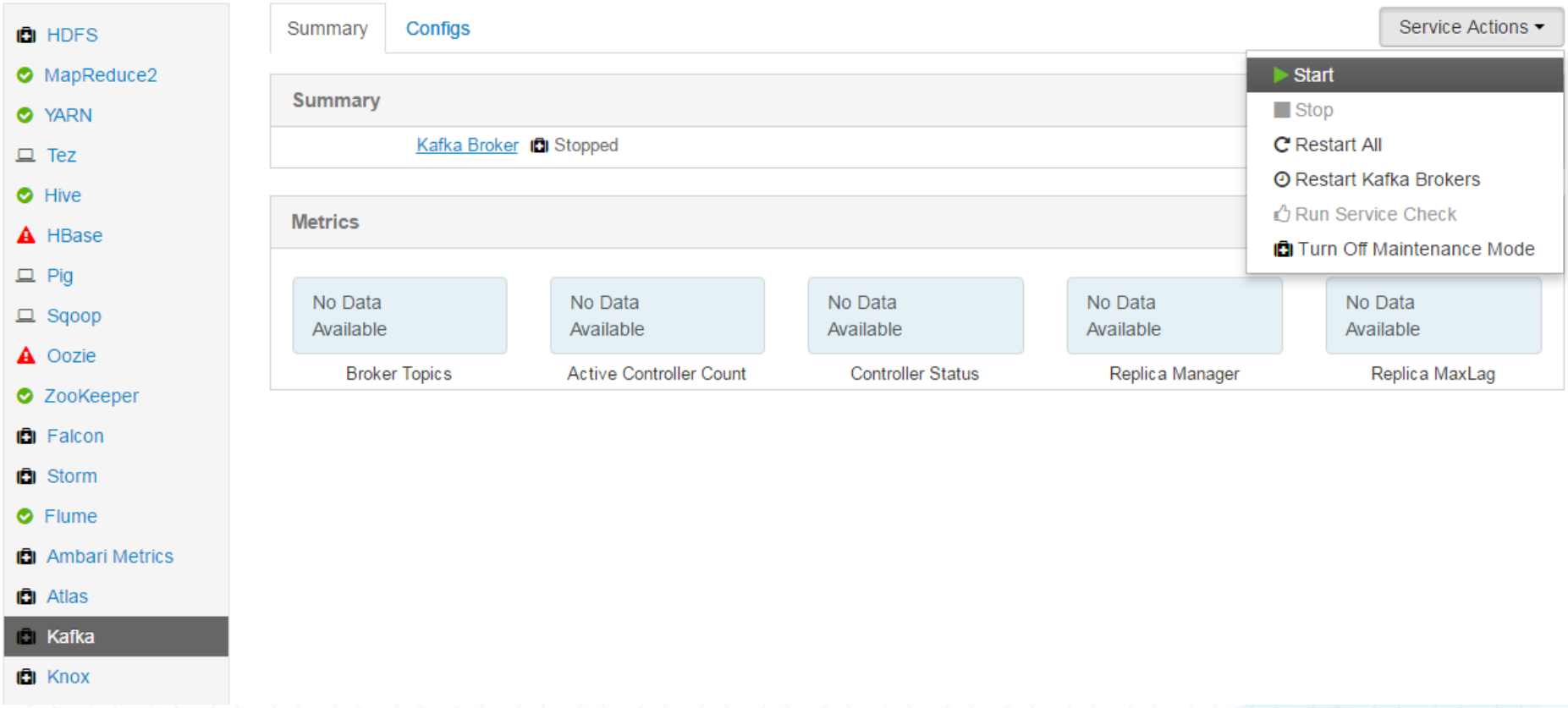


Kafka 叢集運作


Push
Push
Producer
Front end
System log
Sensor
.....
Front end
Push
Push
Push
Push
Push
Broker
kafka
kafka
kafka
Zookeepoer
Pull
Pull
Pull
Pull
Redis
Consumer
Spark Streaming
Real time Dashboard
Hadoop Cluster
.....
Pull
Pull
Pull

啟用Kafka

- 在Ambari 啟用Kafka



手動啟用Kafka

- 連線到Putty
- 指定server.properties
- 啟用Kafka Server
  - sh /usr/hdp/2.4.0.0-169/kafka/bin/kafka-server-start.sh /usr/hdp/2.4.0.0-169/etc/kafka/conf.default/server.properties

建立與觀看Topic

- Create Topic
  - /usr/hdp/2.4.0.0-169/kafka/bin/kafka-topics.sh --create --zookeeper 127.0.0.1:2181 --replication-factor 1 --partitions 1 --topic queue_test
- List Topic
  - /usr/hdp/2.4.0.0-169/kafka/bin/kafka-topics.sh --list --zookeeper 127.0.0.1:2181

撰寫Producer

```
# -*- coding: utf-8 -*-
import time, json
from kafka import KafkaConsumer, KafkaProducer

producer =
KafkaProducer(bootstrap_servers='localhost:9092')

while True:
    producer.send('queue_test','test')
    time.sleep(1)
    print "awake"
```

啟用Producer 及Consumer

- Start Producer
  - python producer.py
- Start Consumer
  - sh /usr/hdp/2.4.0.0-169/kafka/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic queue_test

(選擇性)撰寫Consumer

```
# -*- coding: utf-8 -*-
from kafka import KafkaConsumer
import io

# To consume messages
consumer = KafkaConsumer('queue_test',
                         group_id='my_group',
                         bootstrap_servers=['localhost:9092'])

for msg in consumer:
    print msg
```

使用Spark Streaming 接受Streaming Log

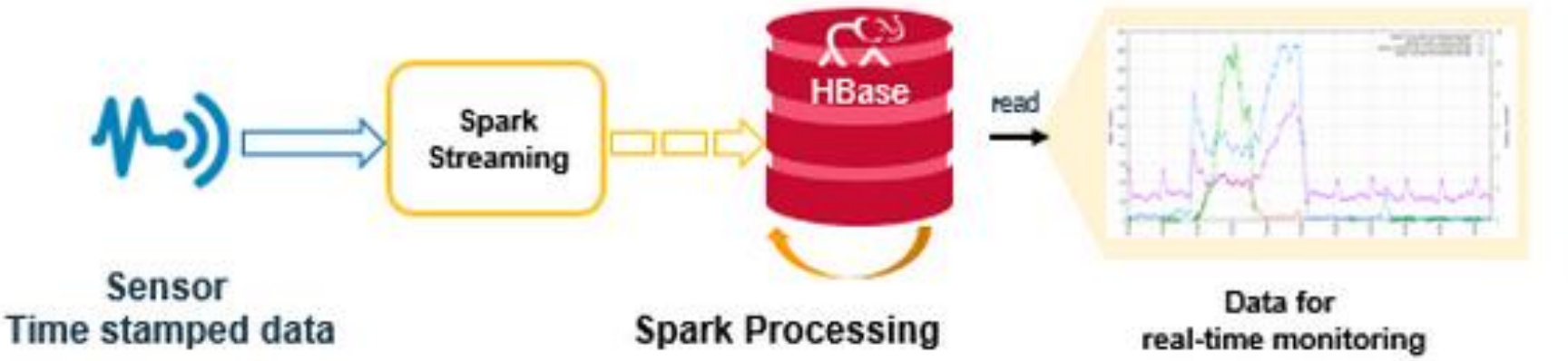
```
sc = SparkContext(appName="qoo")
sc.setLogLevel("ERROR")
ssc = StreamingContext(sc, 1)

kafkaStream = KafkaUtils.createStream(ssc, "localhost:2181",
"GroupNameDoesntMatter", {"queue_test": 1})


messages = kafkaStream.map(lambda xs:xs)
messages.pprint()
```

啟動Spark Streaming 工作

- receive work
  - spark-submit --packages org.apache.spark:spark-streaming-kafka_2.10:1.5.0 consumeFromKafka,py

HBase

如何建立一個即時監控儀表板



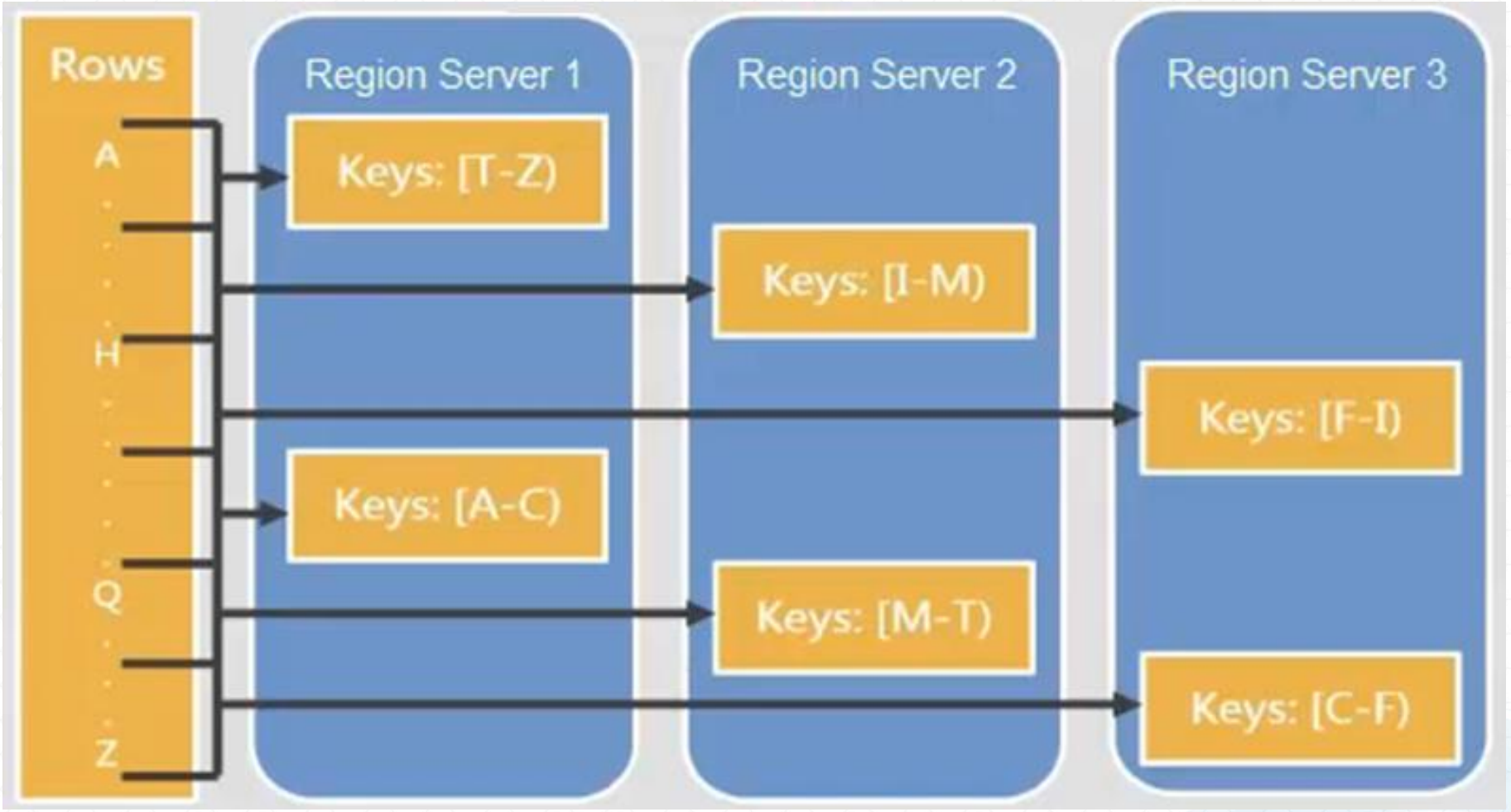
Hbase

- 根據Google 2006年 BigTable 論文實作
- 分散式、多維度、高效能的存儲系統
- 存數十億列、數百萬欄
- High- Availability
- Column- Oriented

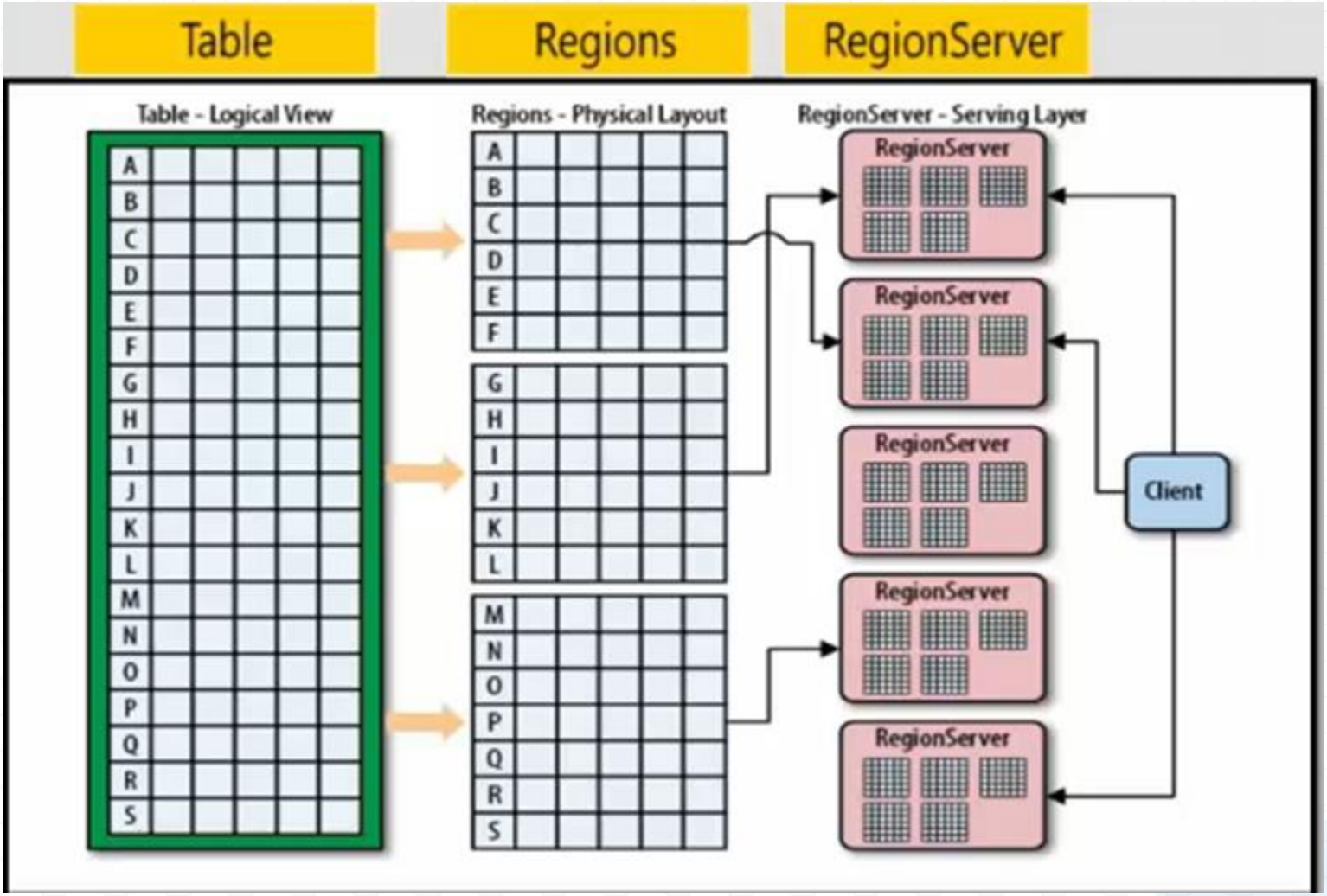
存儲大量的行、列及多版本資料，並能分散到數以萬計的機器中

資料存放模式

資料是根據字典順序擺放

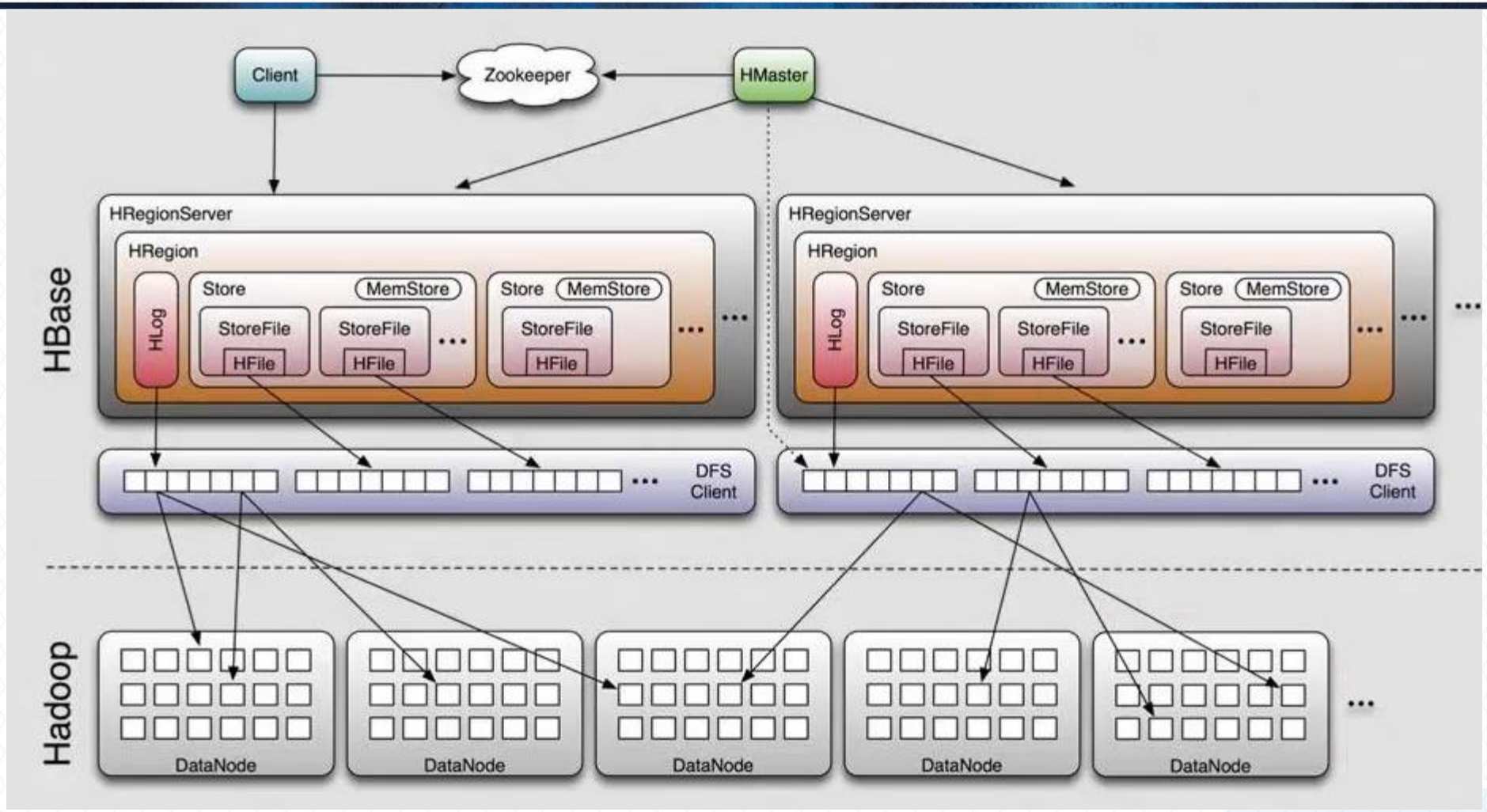


Table, Region, Region Server



一個Region Server 所存放的Region 可能來自不同的Table
由Master 決定哪台Region Server 擺放哪些Regions

完整HBase 架構圖


Client
Zookeeper
HMaster
HBase
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
DFS Client
Hadoop
DataNode

資料存放架構

- Table (HBase Table)
  - Region (Regions for the Table)
    - Store = Column Family
    - MemStore
    - StoreFile
      - ✓ Block

資料結構

- Table Terms
  - Table
  - Column Family
  - Row
  - Qualifier(Column)
  - Cell

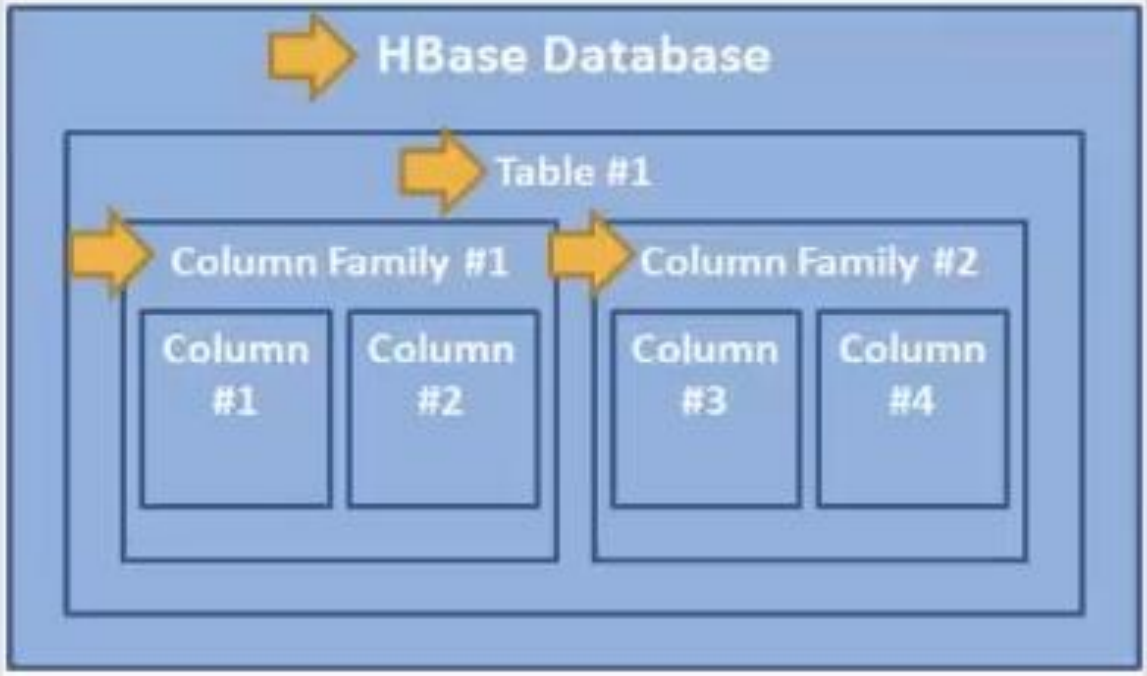
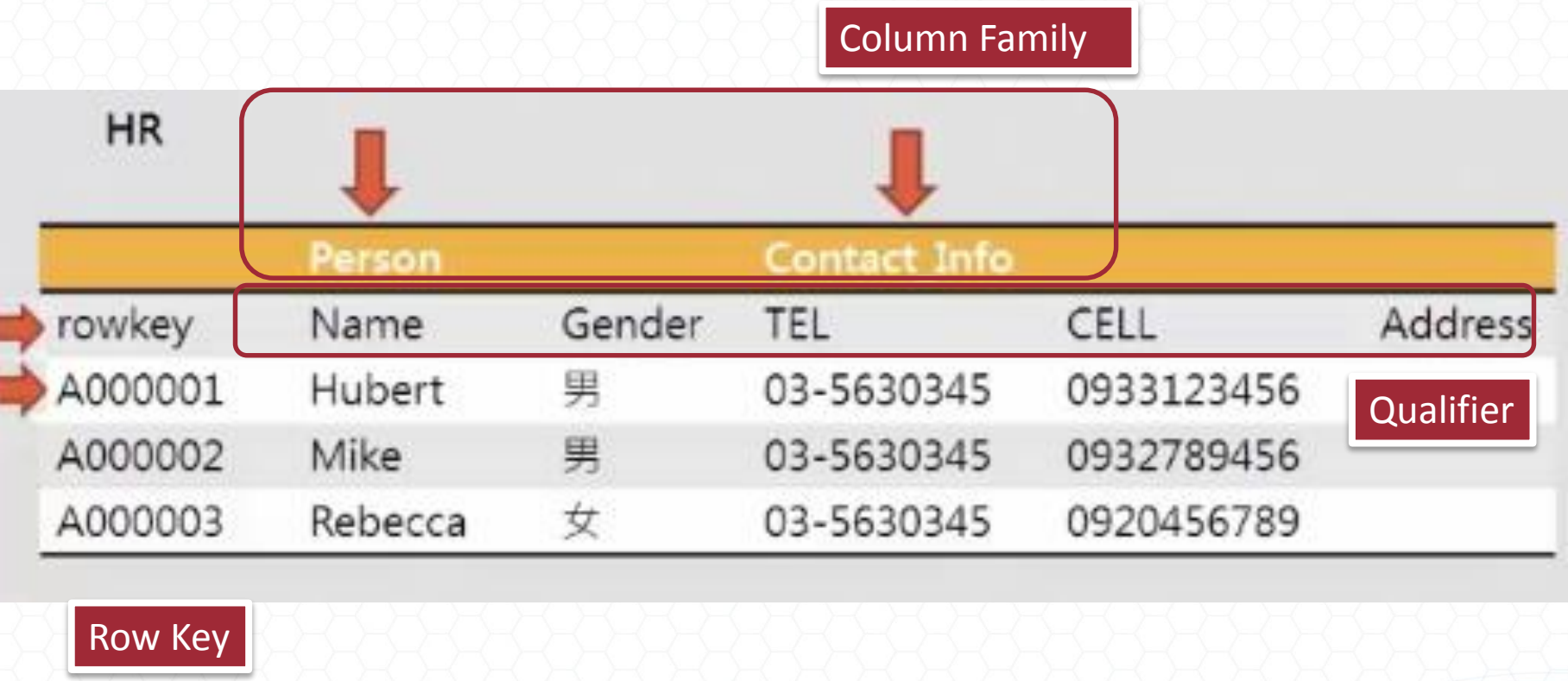


Figure 1 - HBase Data Organization

資料結構


Column Family
HR
Person
Contact Info
rowkey
Name
Gender
TEL
CELL
Address
A000001
Hubert
男
03-5630345
0933123456
A000002
Mike
男
03-5630345
0932789456
A000003
Rebecca
女
03-5630345
0920456789
Qualifier
Row Key

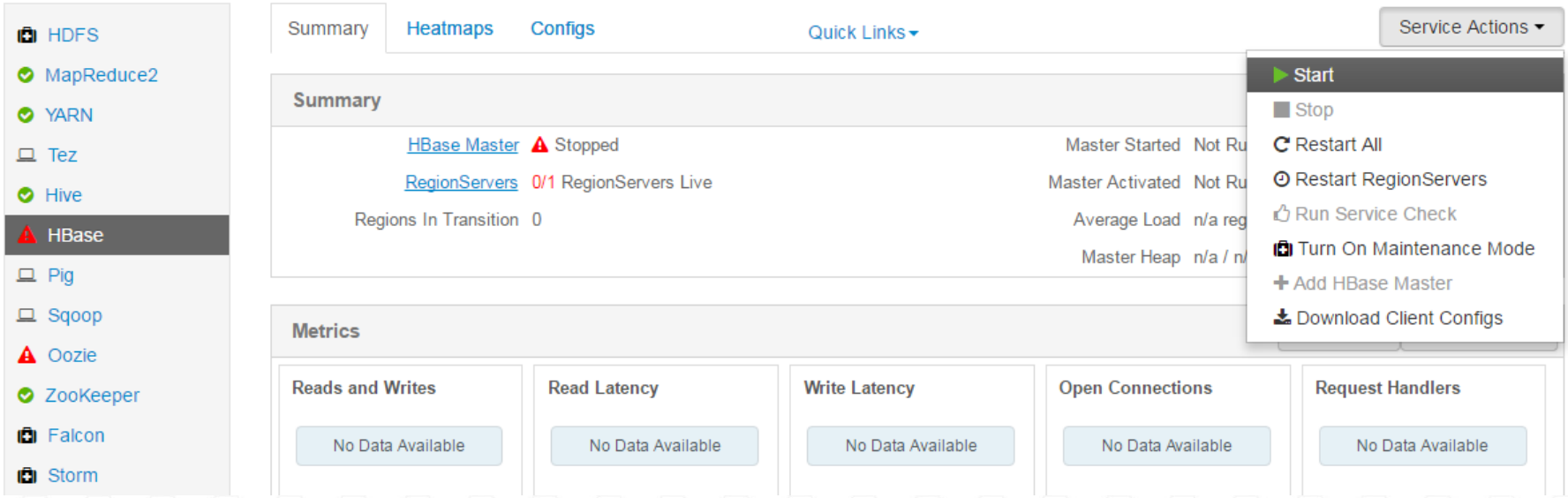
資料結構


HBase Database
Table #N
Table #2
Table #1
Column Family #2
Column Family #1
Column #1
Column #2
Column #3
Column #4
Column #5
Row #1
Value001
Value029
Row #1
Value005
Row #1
Value009
Value015
Row #1
Value023

資料儲存範例

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

在Ambari 中啟用HBase



啟動Thrift

- 啟動Thrift Server
  - /usr/hdp/current/hbase-master/bin/hbase-daemon.sh start thrift -p 9090

Hbase Service Port

- https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.2/bk_HDP_Reference_Guide/content/hbase-ports.html

HBase REST Server Web UI (optional)	All REST Servers	8085	http	The port used by HBase Rest Servers web UI. REST servers are optional, and not installed by default	Yes (Typically admins, dev/support teams)	`hbase.rest.info.port`
HBase Thrift Server (optional)	All Thrift Servers	9090		The port used by HBase Thrift Servers. Thrift servers are optional, and not installed by default	Yes	
HBase Thrift Server Web UI (optional)	All Thrift Servers	9095		The port used by HBase Thrift Servers web UI. Thrift servers are optional, and not installed by default	Yes (Typically admins, dev/support teams)	`hbase.thrift.info.port`

Hbase Shell

檢查狀態	• hbase> status
檢查版本	• hbase> version
檢查跟表格相關說明	• hbase> table_help
使用者資訊	• hbase> whoami

Hbase DDL 指令

- 建立表格
  - Syntax :
  - create ‘table-name’ , { NAME=>‘column-family-name’ } , { NAME=>‘column-family-name’ }
  - Example :
  - create ‘easylearning’ , {NAME=> ’Student’} , {NAME => ’Teacher’}
- 檢視表格
  - Syntax :
  - describe ‘table-name’
  - Example :
  - describe ‘easylearning’
- 表列所有表格
  - Syntax :
  - list

Hbase DML 指令

- 新增資料
  - Syntax :
  - put ‘table-name’ , ‘Row-Key’ , ‘Colum-family-name : column-name’ , ’value to be inserted’
  - Example :
  - put ‘easylearning’ , ‘stud-101’ , ‘Student : stud_name’ , ‘Alice’
- 取得資料內容
  - Syntax :
  - get ’<table name>’,’row1’
  - Example :
  - get ‘easylearning’, ‘stud-101’

Hbase DML 指令

- 刪除Cell 資料
  - Syntax :
  - delete ‘<table name>’, ‘<row>’, ‘< Colum-family-name : column name >’, ‘<timestamp>’
  - Example :
  - delete ‘easylearning', ‘stud-101', ‘Student : stud_name’, 1417521848375
- 刪除全部資料
  - Syntax :
  - deleteall ‘<table name>’, ‘<row>’
  - Example :
  - deleteall ‘easylearning’, ‘stud-101’

Hbase DML 指令

- 取得特定表格資料
  - Syntax :
  - scan ‘table-name’
  - Example :
  - scan ‘easylearning’
- 計算列數
  - Syntax :
  - count ‘table-name’
  - Example :
  - count ‘easylearning’

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,

透過Python 操作Hbase (範例)

```
import happybase
connection = happybase.Connection('localhost')
connection.open()
print connection.tables()

connection.create_table(
    'recommendation',
    {'cf1': dict() }
)
table = connection.table(' recommendation')
table.put('userid', {'cf1:rec1': 'value1'})
row = table.row('userid')
print row['cf1:rec1']

for key, data in table.scan():
  print key, data

connection.close()
```

使用Spark Streaming 將資料寫入HBase

```
def SaveRecord(rdd):
    connection.open()
    table = connection.table('mytable')
    datamap = rdd.collect()
    for k, v in datamap:
        v2 = json.loads(v)
        #print v2
        table.put(str(uuid.uuid4()), {'cf1:col1': v2['name']})
    connection.close()


messages = kafkaStream.map(lambda xs:xs)
messages.foreachRDD(SaveRecord)
```

啟動Spark Streaming 工作

- receive work
  - spark-submit --packages org.apache.spark:spark-streaming-kafka_2.10:1.5.0 SaveToHBase.py

THANK YOU