



Alexandria

Analytics & Knowledge

Campaign Management : Big Data Marketing Optimization

M6. PROPOSTA TÈCNICA

Octubre de 2016

Integrants del Grup :

Francisco Gómez Martínez
Gerardo Marcos Vega
Xavier Gonzalez Arriola
Gustau Franch Espuny

INDEX

I.- Introducció

I.1.- Continguts de la proposta tècnica

I.2.- Antecedents i objectius del projecte

II.- Proposta tècnica i justificació

II.1.- Introducció de la solució global

II.2.- Detall de l'arquitectura Lambda

II.2.1 Ingestion Layer

II.2.1.1 Big Data Sources

II.2.1.2 Missatgeria distribuïda (Kafka)

II.2.1.3 Broker d'informació (Flume)

II.2.2 Batch Layer

II.2.2.1.- Distributed files storage (Data lake)

II.2.2.2.- ETL Jobs (Spark SQL)

II.2.2.2.- Maching Learning (Spark MLlib)

II.2.3.- Speed Layer

II.2.3.1.- ClickStream (Spark Streaming)

II.2.3.2.- ELT Jobs (Spark)

II.2.4.- Serving Layer

II.2.4.1.- Model dimensional OLAP (Hive, Kylin, HBase)

II.2.4.2.- Business Intelligent (Tableau)

II.3.- Conclusions Finals

III.- Recursos del Projecte

III.1.- Codi font Bomber

III.2.- Configuració de Kafka

III.3.- Configuració de Flume

III.4.- Codi ETL Job Batch layer (Users & Campaign)

III.5.- Codi ETL Job Speed layer de Hive (Visits)

III.6.- Model estrella en Hive

III.7.- Model de dades dimensional amb Kylin

III.8.- Codi Segmentació i Classificació amb Spark MLlib

III.9.- Insights i Dashboards amb Tableau

V. Bibliografia

VI. Apèndix

I. - Introducció

I.1 Continguts de la proposta tècnica

El document de proposta tècnica té dos objectius principals. L'una, la de detallar aquelles tecnologies existents en el ecosistema Hadoop i que podrien haver estat emprades per al desenvolupament del prototip, pretenent en cadascuna d'ells, argumentar i justificar la seva idoneïtat en la seva elecció respecte d'altres. Aquest exercici ha permès als integrants del projecte, identificar les principals funcionalitats i grau de madures de cadascuna d'elles, així com també reproduir el procés de selecció de tecnologies que es duria a terme en un projecte real d'aquestes característiques.

Com a segon objectiu, es pretén presentar el disseny tècnic de la solució, pel que fa a la configuració dels mòduls Hadoop seleccionats, la programació dels mateixos, el model de dades, processos d'analítics i de Machine Learning realitzats, així com també els reports de navegació i visualització extrets.

I.2- Antecedents i objectius del projecte

El projecte es durà a terme amb col·laboració amb la empresa Vir2ality, la qual ha desenvolupat una plataforma de gestió de campanyes de màrqueting anomenada Alexandria i que cedeix els Datasets de dades per dur a terme el projecte.

L'objectiu fora fer un cas d'ús de les funcionalitats actuals del Campaign Manager però en un entorn Big Data que permeti el tractament y anàlisi de les dades del ClickStream generades en la web de client i associades a les Campanyes de Marqueting generades. La idea fora realitzar un anàlisi en temps real dels events generats en la navegació web dels usuaris a l'hora de poder fer un seguiment del compliment dels objectius de les campanyes definides en el Campaign Manager .

Concretament es contempla com a nucli del projecte :

- Optimització dels processos de captació d'informació que permetin un anàlisi en Temps real dels events de campanya;
- Creació d'un repositori comú de dades principal (DataLake) que permeti avastar totes les necessitats d'anàlisi i reporting de la organització;
- Redisseny del model de dades i creació d'un entorn multi-dimensional que aporti a les eines de Business Intelligence la capacitat de visualització i descobriment del domini de informació gestionat;
- Finalment, la inserció de processos analítics de classificació i segmentació incorporats en el flux d'informació, a l'hora de poder dur a terme un profiling acurat dels usuaris.

A continuació, es presenta a l'arquitectura lògica i en alt nivell de l'actual plataforma Alexandria. Aquesta ens ajudarà a entendre millor la arquitectura proposada en el projecte més endavant en aquest mateix document.

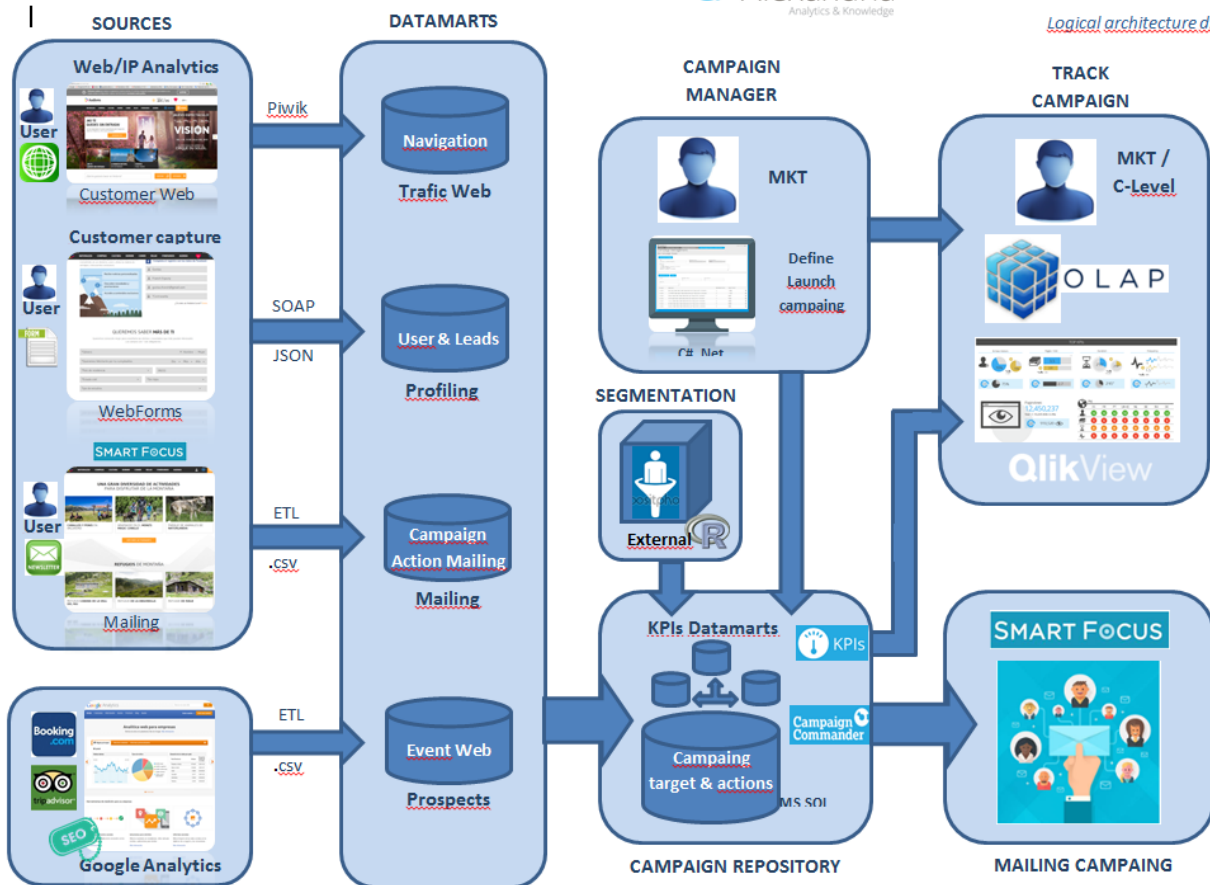


Fig.- Arquitectura lògica d'Alexandria actual

Les carències de la plataforma actual, han definit els requeriments a tenir en compte en la definició del prototip que ens ocupa en aquest projecte :

- La plataforma Alexandria actual no es distribuïda -> el manteniment la seva escalabilitat, alta-disponibilitat i rendiment son molt costosos.
- Els processos d'Ingesta de dades i d'ETLs son batch amb un decalatge de 24H -> Cost d'oportunitat al no disposar de informació actualitzada per a la presa de decisions.
- No es disposa de la capacitat de gestionar informació no estructurada -> No es pot dur a terme el seguiment de campanyes les principals xarxes socials més emprades (twitter, facebook, instagram,...).
- No existeix un repositori centralitzat d'informació -> Manteniment del model de dades és molt costos i dispers.
- Les transformacions i lògica de negoci està ubicada en tots els nivells - > Upgrades y Deploy costos.
- La segmentació és un procés R extern -> Segmentació diferida no temps real.
- La visualització resideix en un BI i model OLAP extern -> Manteniment de Datamarts intermitjos situació que provoca no disposar de Dashboards actualitzats en temps real.

II.- Proposta tècnica i justificació

II.1.- Introducció de la solució global

La solució que ha estat plantejada en el pilot, es una arquitectura que permeti, per un costat una tractament en temps real de ClickStream generat en la web de clients, així com les carregues pròpies del transaccional que composen la eina de gestió de Campaign Manager.

Aquets requeriments han segmentat l'arquitectura en diferents capes (layers) que tracten el flux d'informació a l'hora de cercar un equilibri entre latència, rendiment i tolerància a fallades. Utilitzant processos batch per aconseguir persistència en les dades, així com vistes acurades y comprensives de la informació, a l'hora que simultàniament es fan servir processos en tractaments en Real-Time que permeten apropar al procés de presa de decisió de informació vigent i valuosa.

Aquestes layers prenen el nom de Ingest layer, Speed layer, Batch layer y Serving layer i bategen l'arquitectura global amb el nom de Lambda.

En cadascuna d'aquestes layers s'empraran diferents serveis del stack hadoop i projectes Apache¹ que conferiran les capacitats anteriorment anteriorment esmentades.

A continuació es mostra la arquitectura lambda proposada en el pilot, així com les capes que el componen amb els mòduls emprats.

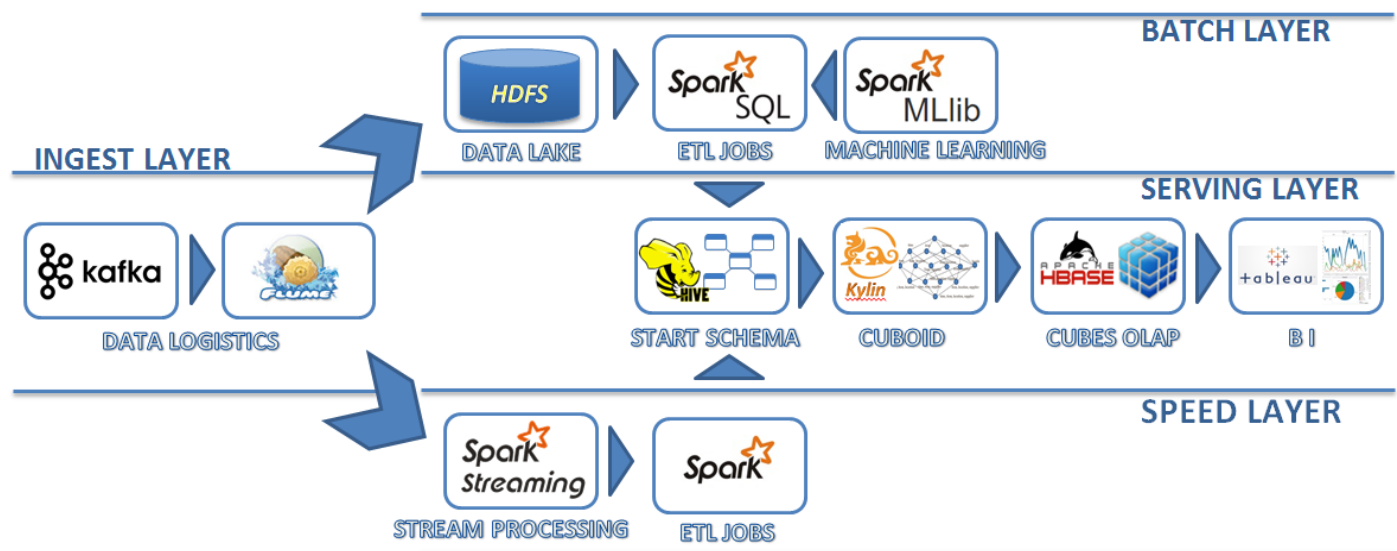


fig.- Diagrama d'arquitectura Lambda

¹ <https://projects.apache.org/projects.html>

Ingest Layer

La plataforma Alexandria actual, captura el clickstream (events) mitjançant crides webservices (SOAP) existents en la web de client. Donat que no es objectiu del pilot reproduir aquest mecanisme, s'ha desenvolupat un procediment en java capaç de bombejar la informació disposada en els Datasets big data d'entrada, generant un flux continu de dades cap a la nova plataforma. Aquests Datasets contenen tot el domini de dades necessari per dur a terme les analítiques i insights propòsit del projecte.

Al l'hora de disposar d'un ritme d'ingestió de dades que permeti el seu correcte processament i una selecció de les fonts d'informació emprarem Kafka i seguidament emprarem Flume per fer arribar als subsegüents capes de Speed i batch, la informació necessària pel seu propòsit.

Batch Layer

Un cop el flux d'informació arriba a la capa de Batch es consolida i es converteix en persistent en el sistema de fitxers distribuït HDFS, disposat a mode de data lake, i que té la principal finalitat de constituir un repositori únic i centralitzat de les dades per als diferents propòsits d'anàlisis i reporting de la solució.

Aquesta capa estarà complementada per processos de ETL en Spark i Spark SQL que tindran com a funció el filtrat, refinat i transformació de la informació que serà carregada en capes posteriors en els sistemes de explotació i visualització de la informació. Més concretament es carregarà en el model estrella en Hive, com a sistema warehouse de la plataforma.

A la seva vegada també es capacitarà amb funcions de Machine learning desenvolupats i executats amb Spark MLlib i que permetran el profiling dels clients i la segmentació de continguts para una resposta personalitzada de les accions de marketing dirigides a client.

Speed Layer

En paral·lel al procés de batch en HDFS, es dura a terme el tractament en streaming dels events generats a la web del client, a l'hora de computar i analitzar el clickstream i la navegació dels usuaris. El flux d'informació serà gestionat mitjançant Spark Streaming, a l'hora de computar les visites dels usuaris navegants i la posterior carrega d'aquest anàlisi en el model estrella en Hive.

Serving layer

La capa de consum dona accés a les dades a l'usuari que fa el seguiment i anàlisis de les dades de campanya. Aquest disposarà de la eina Tableau de Business Intelligence a l'hora de poder visualitzar les dades i navegar en ells. El model multidimensional de dades descarregarà la feina al BI en les tasques OLAP (On-Line

Analytical Processing). Aquesta tasca es durà a terme amb Kylin el qual pren el model en estralla disposat en Hive com a entrada per calcular mitjançant Map Reduce dels cuboids que donaran forma als cubs desplegats en HBase i que a la seva vegada permetrà al BI Tableau accedir a la informació ja agregada i materialitzada.

Les dades presentades prendran forma de reports analytics (insgths) i dashboards d'estat i seguiments per a la presa de decisions.

A continuació es detallarà més exhaustivament cada layer proposat, les tecnologies disponibles que podrien donar servei i la justificació dels mòduls escollits.

II.2.- Detall de l'Arquitectura Lambda

A continuació es detalla cadascuna de les capes de processament que componen la arquitectura lambda del Prototip.

II.2.1.- Ingestion Layer

II.2.1.1.- Big Data Sources

La plataforma Alexandria actual rep events produïts per la navegació dels usuaris en les webs del client on es fa el seguiment de campanya, així com també la interacció amb els banners que componen els elements visuals de les campanyes de marketing llançades.

Mitjançant protocol SOAP y la solució Piwik (<https://piwik.org>), La solució d'Alexandria actual captura els events de navegació (la adreça IP, URL, Cookie d'usuari, direcció URL de procedencia, paraules clau de cerca, informació de la campanya i més).

Donat que no era l'objectiu del projecte reproduir els mecanismes de ingesta actuals, la solució que s'ha emprat ha estat considerar els datamarts resultats d'aquesta captura com a Datasets (fitxers en format .csv) d'entrada de la nova plataforma Big Data.

En el apèndix s'han disposat els principals dasasets emprats en el projecte i que disposen com a domini de dades la navegació, la de Leads i Users, i la de campanya.

Però degut a que si es pretenia simular un clickstream real, s'ha proposat desenvolupar un mecanisme de bombeig de les dades. Això ens permet injectar les dades com un flux continuat d'informació que la plataforma caldrà gestionar mitjançant un sistema de missatgeria distribuït.

Aquest bomber, responsable de generar els events, és un projecte Maven en Eclipse, i utilitzant l'API de Kafka per fer un Producer que simplement llegeix els d'events registrats en els arxius datasets (disposats en format csv) i les va enviant al servidor Kafka.

El bomber, per simular el clickstream provocat per la navegació d'usuari, es dedica a enviar events registrats en els arxius datasets esmentats, línia per línia i per cada interval de temps escollit. Com s'ha indicat anteriorment, aquestes dades que arriben a Kafka es desaran a HDFS mitjançant Flume regularment, concretament cada 10, 100 o 1000ms.

Els Datasets indicats representen una volumetria representativa de un cas d'ús que potser tractat en tècniques de big data :

- Navegació (planes i visites): 27.000.000
- Històric de totes les segmentacions dels usuaris : 30.000.000
- Emails enviats per campanya i usuari : 21.000.000
- Esdeveniments (open. click...) produïts per emailing per usuari i campanya: 6.000.000
- Formularis omplerts de campanyes: 1.000.000
- Usuaris (profile): 600.000
- Cookies d'usuari : 300.000
- Usuaris anònims : 4.000.000

II.2.1.2.- Missatgeria Distribuïda amb Kafka

Com indica el seu nom, aquesta capa registra els processos d'ingesta de les dades i canalitzar-la posteriorment amb flume a la capa de Batch emmagatzemar-les en brut a *HDFS (data Lake)*, i en una etapa paral·lela remetreu a la Speed layer per al seu tractament streaming del clickstream.

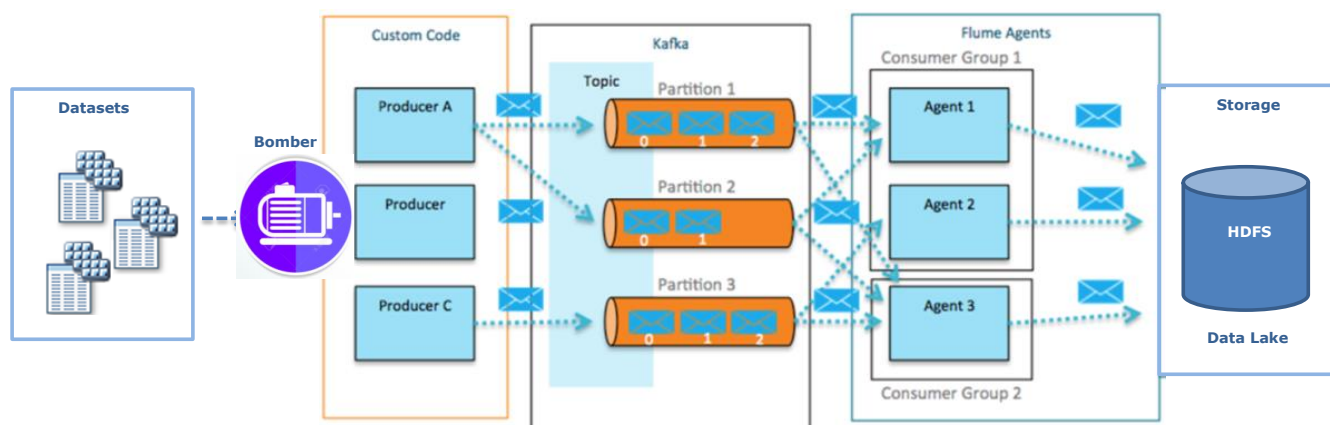


Fig.- Flux d'ingesta de dades mitjançant Apache Kafka i Flume fins Batch layer

El procés s'inicia amb la informació del visitant de les pàgines web, la qual serà registrada i enviada com a paquet de missatgeria per TCP/IP al sistema, que ingerirà aquestes dades guardant-les al Data Lake i en paral·lel a Spark Streaming.

Tecnologies existents

L'espai de processament de flux de codi obert està actualment explotant, y presenta moltes alternatives y solucions. Apache Software Foundation, en l'actualitat te més de 10 projectes de processament de fluxos, alguns en la incubació i altres amb un nivell superior de madures.

Destaquen mòduls recents com Apache Kafka, Apache Flink, Apache Storm, ... entre d'altres, així com altres solucions mes madures com RabbitMQ.

A continuació passem a detallar algunes de les característiques que els diferencien :



*Apache Storm*², és un sistema de computació en temps real distribuït per al processament de grans volums de dades a alta velocitat. Storm és extremadament ràpid, amb la capacitat de processar més d'un milió de registres per segon per node en un clúster de grandària modesta. Les empreses aprofiten aquesta velocitat i la combinen amb altres aplicacions d'accés a dades en Hadoop per evitar esdeveniments no desitjats o per optimitzar els resultats positius.

Algunes de les seves noves oportunitats de negoci inclouen: gestió en temps real de servei al client, la monetització de dades , quadres de comandament operacionals , o d'anàlisi de seguretat cibernètica i la detecció d'amenaçes.

Storm és simple i els desenvolupadors poden treballar utilitzant qualsevol llenguatge de programació.

Storm és:

- ràpid, un milió de missatges de 100 bytes per segon per node
- escalable
- tolerant a fallades
- fiable
- fàcil d'operar

Storm no es un sistema de cues exactament, és més aviat pel processat en temps real d'elements d'aquestes cues executant tot tipus de manipulacions de dades en temps real i en paral·lel.



*Apache Flink*³, consumeix i produeix *streams* de dades. *Flink* s'utilitza normalment amb *Kafka* com a capa d'emmagatzematge subjacent, però és independent d'ella.

Les diferències fonamentals entre programes *Flink* i *Kafka* es troben en la forma en què aquests s'implementen i administren, i com el processament en paral·lel (incloent la tolerància a fallades) està coordinat. Aquestes són les diferències principals que estan arrelades en l'arquitectura d'aquests dos sistemes.

² <http://storm.apache.org/>

³ <https://flink.apache.org/>

Encara que certament hi ha una superposició entre *Kafka* i *Flink*, es veuen com a sistemes complementaris. *Kafka* fa fàcilment accessible els missatges a les aplicacions. Els beneficis de *Kafka* són: rendiment, escalabilitat, seguretat. De *Flink*, en canvi, podríem destacar també un alt rendiment i baixa latència.

Finalment, *Flink* i *Kafka* serien una gran combinació per a una arquitectura de *streaming*. Els equips de *Flink* i *Kafka* han funcionat molt bé junts en totes les versions.

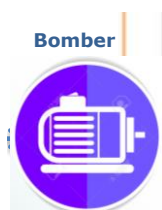
Solució escollida

En el projecte d'Alexandria les dades arriben al sistema com a paquets des de les pàgines web que reben els usuaris destí de les campanyes.

Un intermediari de missatges (*broker*) d'un servidor *Kafka*, rebrà un missatge per cada *event* d'interès registrat que seguidament *Flume* desarà en *HDFS* al *DataLake*.



Fig. – Ingesta de dades (primera etapa)



Bomber, amb l'API Producer permet mitjançant un sistema de bombeig simular els events de entrats que generarien els usuaris al navegar per la web client. Aquest envia tòpics al clúster de Kafka (tot i ser d'una sola màquina en el prototip); aleshores amb aquesta API podem fàcilment fer un Producer que vagi enviant línia per línia els més 20 GB de Dataset de navegació que registra les planes vistes pels usuaris.

Cal esmentar, que al moment de rebre els missatges Kafka auto genera els tòpics si s'ha configurat en el a l'arxiu de configuració del Broker de Kafka.

```
# KAFKA server.properties  
auto.create.topics.enable = true
```



*Apache Kafka*⁴ està dissenyat com a sistema de missatgeria altament distribuït amb lectures i escriptures ràpides i disposa d'un bon nombre d'APIs en varis llenguatges, a més d'una bona documentació disponible. *Kafka* és linealment escalable i no té un estat centralitzat, és per això que *Kafka* esdevé el candidat número u entre la resta; inicialment molts altres serien vàlids, però les prestacions de *Kafka* esdevindrien clau a mesura que el projecte es faci més gran: escalabilitat, baixa latència i tolerància a fallades.

L'aplicació original de *Kafka* era el seguiment de l'activitat de l'usuari en temps real. Això vol dir que l'activitat (pàgines vistes, recerques, o qualsevol altra acció) queda publicat com a missatge que *Kafka* rebrà i emmagatzemarà a HDFS; per posteriorment netejar i refinar, i finalment processar i presentar informes. Així doncs, és similar al ús que se li vol donar i queda definit com la primera etapa de la ingesta de dades.

Un servidor *Kafka* rebrà els paquets (del *bomber* en la demostració o diverses fonts en el cas real) i els emmagatzemarà amb el tòpic indicat; "navigation", "clients" o "campaign". El servidor *Kafka* de la nostra demostració tindrà únicament un sol *Broker*, d'una sola partició. I com podem apreciar en el fitxer de configuració `server.properties` el paràmetre `broker.id` es igual a zero. En una arquitectura real, formada per un clúster de *Kafka* aquest paràmetre prendria atendre a tòpics diferents en diferents màquines.

```
# KAFKA server.properties  
  
broker.id=0  
host.name=localhost  
Port=9092  
num.partitions=1 log.retention.hours=1 log.roll.hours=1
```

II.2.1.3.- Broker d'informació amb Flume



*Apache Flume*⁵, Informalment anomenat *Flafka*, la integració *Flume-Kafka*. *Kafka* ha estat escollit com a entrada dels missatges per monitoritzar l'activitat dels usuaris, però a l'hora de moure aquestes dades a HDFS s'ha escollit *Flume* com a *Consumer*, que suporta diferents models de flux de dades i està perfectament integrat amb *Kafka*.

Flume proporciona una gran manera de processar els *events* amb una latència molt baixa i una mínima complexitat. Per a la construcció d'una aplicació personalitzada

⁴ <http://www.slideshare.net/GeorgeLong3/architecting-big-data-ingest-manipulation>

⁵ <https://flume.apache.org/>

establint el flux de dades des de la font (*Kafka*) fins a *HDFS* (alternativament: *HBase*, *Avro*) desant els *events* rebuts al *DataLake*.

Els *events* registrats pels websites, s'envien com a paquets d'uns 700 bytes cadascun (amb informació variada: *timestamp*, IP del client, URL visitada, categoria de la plana, país, regió, ciutat...) que el *bomber* envia al servidor *Kafka*, creant un *throughput* en el nostre model/simulador d'aproximadament 7.000 paquets per segon, és a dir uns 4.8MBps, que simulen l'activitat real que es podria arribar a produir. Aquesta velocitat d'escriptura és perfectament assumible per *Kafka* en una sola màquina, de fet es podria augmentar fins a 10 vegades més sense problema. Tot i que l'activitat real dels *events* no és superior a 1 o 2 per segon, el sistema estaria preparat per una càrrega molt superior, inclús es podria escalar si fos necessari emprant diversos servidors en un clúster.

Es configura i arrenca un *Agent* de *Flume* que s'encarregarà d'emmagatzemar les dades dels paquets rebuts a *Kafka* dels tres *topics* mencionats a *HDFS* conformant el *data lake* a l'hora que seran llegits de *HDFS* per l'aplicació de *batch layer*.

```
#FLUME example.conf
....
flume1.sources.kafka-source-1.topic =navigation
flume1.channels.hdfs-channel-1.type = memory
flume1.sinks.hdfs-sink-1.type = hdfs
flume1.sinks.hdfs-sink-1.hdfs.path = /tmp/kafka/test_alexandria_navigation
```

Un altre *Agent* de *Flume* es configurarà per enviar les dades a *Spark Streaming*. En l'anterior arxiu de configuració de *Flume* hi afegirem el segon *Agent*, bàsicament haurem d'afegir les següent línies:

```
#FLUME example.conf (second Agent)
....
agent.sinks = spark
agent.sinks.spark.type = org.apache.spark.streaming.flume.sink.SparkSink
agent.sinks.spark.hostname = localhost
agent.sinks.spark.port = 9999
agent.sinks.spark.channel = memoryChannel
```



Forçant el bomber a un ritme d'un paquet per mil·lisegon començem a trobar-nos problemes de *overflow* i *out of memory* que de ben cert desapareixerien el montar el sistema en un clúster.

II.2.2 - Batch Layer

II.2.2.1.- Distributed File System (Data Lake)

Com ja s'ha indicat para la solució distribuïda de fitxers s'emprarà HDFS, encara que existeixen diverses alternatives, *MapR-FS, Ceph, BeeGFS, GlusterFS, Infini, ObjectiveFS, MooseFS, Quantcast File System,...* entre d'altres.



*Hadoop HDFS*⁶, com moltes altres, és *open source* resultant en un extremadament baix cost per byte. L'ample de banda que pot suportar és alt, fins a *2Gbps* en una xarxa de baix cost. Finalment, hem de destacar l'alta fiabilitat que *HDFS* ofereix a l'hora d'emmagatzemar i replicar la informació.

A l'hora de configurar un repositori centralitzat i unificat d'informació, a mode de *Data lake*, serà emprat *HDFS*, encara que *molt recentment, ha aparegut Apache Kudu*⁷, i on certes publicacions⁸ defensen que permet combinar lo millor el món *HDFS* i *HBase* en un sol paquet. Una escriptura a alta velocitat i exploracions i òptim per a les consultes d'accés aleatori. Això, podria portar a *Kudu* a convertir-se en un repositori de dades de propòsit general amb usos més allà de l'àmbit de l'anàlisi.

Donat que *HDFS*, representa una tecnologia molt més madura, serà emprat per conformar el *Data Lake* pretès amb les dades procedents del sistema d'ingesta. Un *Data Lake* és una ubicació central en la qual emmagatzemar totes les dades, sense importar el seu format: les dades poden ser estructurades o no estructurades. És típicament, encara que no sempre, construït utilitzant *HDFS*. Degut a que totes les dades són ben vingudes, els *Data Lakes* són una emergent aproximació als reptes de la integració de dades en un tradicional *EDW (Enterprise Data Warehouse)*⁹.

Alguns dels beneficis d'un llac de dades inclouen:

- El tipus de dades que hi podem emmagatzemar és il·limitat
- No cal tenir les respostes inicialment, les dades romandran al *Data Lake* per se refinades i tractades en qualsevol moment
- No hi ha límits sobre com consultar les dades, una varietat d'eines ens faran comprendre millor el que les dades signifiquen
- Tenim totes les dades (en brut) en un sol lloc

⁶ https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html

⁷ <https://kudu.apache.org/>

⁸ <http://www.infoworld.com/article/2986675/hadoop/cloudera-kudu-hdfs-hbase-in-one.html>

⁹ Book: Architectind Data Lakes by Alice LaPlante and Ben Sharma. [2016] O'Reilly Media.

DATA WAREHOUSE	vs.	DATA LAKE
structured, processed	DATA	structured / semi-structured / unstructured, raw
schema-on-write	PROCESSING	schema-on-read
expensive for large data volumes	STORAGE	designed for low-cost storage
less agile, fixed configuration	AGILITY	highly agile, configure and reconfigure as needed
mature	SECURITY	maturing
business professionals	USERS	data scientists et. al.

Fig. - Data Lake vs Data Warehouse

El major avantatge dels *Data Lakes* és la flexibilitat en permetre que les dades romanguin en el seu format nadiu, oferint unes oportunitats majors en disposar de les dades originals pel tractament que escaigui en cada moment, ara o més endavant, per a l'anàlisi i generació de resultats.

Un dels principals inconvenients del processat *schema-on-write* de les tradicionals *EDW* és l'enorme quantitat de temps i cost de la preparació de les dades. Per a un important projecte *EDW*, es requereix un extens modelat de dades. Moltes organitzacions inverteixen en comitès de normalització que es troben i deliberen sobre els estàndards, i poden durar mesos o fins i tot anys per completar la tasca en qüestió.

El *Data Lake* elimina tots aquests problemes. Tant les dades estructurades com les no estructurades poden ser ingerides fàcilment, sense cap tipus de modelatge de dades o normalització. A causa de que l'esquema d'emmagatzematge no precisa estar definit des del principi, el costós temps de modelatge no és necessari.

Així doncs, amb el *Data Lake* aconseguim la llibertat d'un model flexible de dades en una única ubicació (replicada segurament), capacitat per gestionar *streamings* (missatgeria en el nostre cas) amb escriptures ràpides, accessibilitat senzilla, baix cost i escalabilitat, inconvenient molt gran per les tradicionals *EDWs*.



Provant diferents valors dels paràmetres de configuració de *Flume* no s'ha aconseguit arribar a més de 10MB en els arxius de guarda a *HDFS*.

II.3.3.2 – ETL jobs amb Apache Spark



Una vegada la informació estigui disponible en el *DataLake*, un procés *Apache Spark* s'ocuparà periòdicament (cada pocs segons) de comprovar l'arribada de nous arxius d'informació de *events* a *HDFS* per anar-los netejant, preparant per a ser guardats posteriorment en les taules i el metadades de *Hive* el qual es configurarà como el *warehouse* del sistema prèvia a la dimesionalitat.

Les dades hauran de ser refinades abans de desar-les a *warehouse*, i no sol seleccionant només els camps del nostre interès. Direm que les dades estan a punt per a l'anàlisi estadístic, es coneix com a "consistent Data". Per tenir consistència de dades s'ha de corregir: valors que falten, els valors especials, errors, i els valors atípics han de ser eliminats, corregits o imputats.

Els valors que falten (coneguts com NAs) són una de les inconsistències més bàsiques. Alguns mètodes d'anàlisi de dades poden suportar aquests NAs mentre que d'altres poden fallar quan es troben amb que han desaparegut els valors d'entrada, o es pot confondre un valor que falta amb una categoria per defecte. NA es confon freqüentment amb una categoria desconeguda; però aquests són dues idees diferents. Un valor NA estableix que la informació no està disponible en el conjunt de dades, mentre que un valor desconegut indica que la informació és al conjunt de dades, però es desconeix. Un enfocament simple per fer front als NAs és ignorar els registres que els contenen. Quan la relació de NAs és alta, és millor utilitzar imputacions tècniques.

Les imputacions tècniques simplement consisteixen en l'estimació dels valors NAs basats en la resta de registres. El mètode més bàsic d'imputació és determinar la mitja dels valors observats. Un model de regressió lineal també pot ser utilitzat com a imputació tècnica. En aquest cas, els valors s'estimen utilitzant una regressió lineal de variables conegudes.

Hi pot haver també algunes inconsistències evidents en les dades, com els valors negatius d'edat, fàcils de detectar i fixar mitjançant l'ús d'un conjunt de regles o restriccions definides.

Spark també s'ocuparà de carregar les dades dels datasets de les taules de "campaign" i "clients" a *Hive*, que en el present prototip seran estàtiques, tot i que en una futura aplicació seran totalment dinàmics.

Les ETL's de carrega a dur a terme son :

ETL	Domini d'informació
DT_Users_Leads	Taula de detall dels usuaris registrats, leads i usuaris anònims
DT_Action	Taula on es descriu les accions de campanya associats a les visites
DT_Campaign	Taula de campanyes que engloben a les accions
DT_GlobalCampaign	Taula de campanyes globals que inclouen les campanyes i accions

II.3.3.3 – Machine Learning amb Spark MLlib

El processos de machine learning del prototip es concreten amb un procés de classificació de qualitat de la navegació dels usuaris i de una segmentació dels continguts. Aquests processos permetran classificar al usuari segons la qualitat de la seva visita a la web, així com també la creació de regles que es desprenen del comportaments dels usuaris cosa que permetrà personalitzar les campanyes als targets de usuaris.

SEGMENTACIO DE QUALITAT

Antecedents.

En un principi l'eina aplicava un procés de segmentació propiament dit i pel qual havia sigut necessari crear una serie de variables (com per exemple el nombre de visites i el nombre de planes).

Aquesta segmentació produïa 10 segments que tractaven de classificar la qualitat de les visites que feien el usuaris i tenia sentit fer-ho perquè no es feia segmentació de contingut encara.

Al decidir fer la segmentació de contingut aquest enfoc de la segmentació de qualitat va variar: ja no calia fer tants grups i es va pensar que fent una segmentació de qualitat amb menys grups seria més senzilla i pràctica. Classificació de qualitat.

Aquest es l'objectiu de la nova 'segmentació' de Qualitat i no s'enfoca aplicant cap dels mètodes habituals d'una segmentació propiament dita sino que tractar de trobar un mètode que actuï més aviat com un 'filtre'. Per això es pot parlar de 'Classificació' més que de Segmentació.

Aquest filtre ha de servir per definir els segments primer i després ser capaç d'assignar la Qualitat del Usuaris a un d'ells. Però a més cal que aprofiti les mateixes variables que ja van ser definides en la primera segmentació original.

Aquest mètode es pot trobar analitzant estadísticament les dades:

Qualitat	Mediana	%Mediana	Mitjana	%Mitjana
Alta	135.608	5,83%	95.674	4,11%
Mitja	338.063	14,53%	255.736	10,99%
Baixa	452.742	19,46%	575.003	24,71%
No Qualitat	1.400.694	60,19%	1.400.694	60,19%
Total	2.327.107	100,00%	2.327.107	100,00%

Com es pot veure a la figura, més del 75 % de les visites tenen 1 visita i 1 plana i més del 90% 1 visita i 2 planes. Aquest doncs no sembla un bon mètode massa bó per classificar (o potser es massa simple) ja que ens classifica als usuaris en només 2 grups.

Tenim en compte la gran variabilitat com la mediana de les planes visitades, que val 4, o la mitjana de les planes visitades, que val 7, existeix a priori una alta probabilitat de que hi hagi un gran nombre d'usuaris d'alta qualitat que no es classificaran correctament i ho faran com a baixa qualitat.

Si es fa un histograma representant la distribució del nombre de planes es veu com potser podrien sortir més de dos segments.

La solució plantejada consisteix en utilitzar les dues variables conjuntament per definir quatre grups en quatre quadrants de la manera següent:

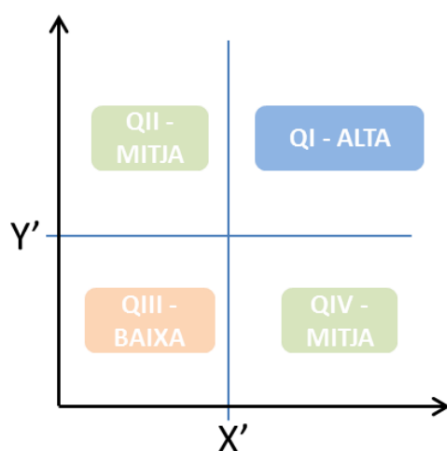


Fig. Classificació usari per qualitat de navegació

Representant el Numero de Planes en funció del Numero de Visites, s'aconsegueixen 4 quadrants a cadascú dels quals està un grup. Les fronteres rectilínees que defineixen el quadrants poden fer-se de dues maneres:

- fent servir les medianes (1 visita 4 planes)
- fent servir les mitjanes (1 visita 7 planes)

Si es representen gràficament les medianes i s'analitzen els pesos, en ambdós casos les visites sense qualitat tenen el mateix pes. En el cas de les medianes, però els altres grups estan distribuïts més uniformement, amb la qual cosa serà més encertada fer la classificació en un grup d'un nou usuari.

Es per aquesta raó, per les que s'agafa la mediana de les variables Nombre de visites i nombre de planes per fer els quadrants. És la opció més útil i recomanable, donat que es minimitza més l'error que amb l'altre opció.

La primera aplicació pràctica que es pot fer amb aquesta classificació de la Qualitat en quatre grups (alta, mitja, baixa i sense) és analitzant la qualitat de les visites dels usuaris per diferents països.

SEGMENTACIO DE CONTIGUT

Antecedents, problemàtica.

La darrera segmentació de contingut calculava els segments establint regles de associació establertes per les visites a les diferents planes de jerarquia 5 (nivell 5). Això donava un gran nombre de segments que es creu convenient revisar intentant reduir-los.

NOVA SEGMENTACIO DE CONTINGUT.

Per aconseguir-ho s'ha decidit agrupar totes les planes visitades en la jerarquia 3 (nivell 3) enlloc del nivell 5. A cada una d'aquestes categories se li ha assignat un segment d'un grup de 10 segments generats segons tres criteris:

- utilitzar alguns segments de la darrera segmentació
- incloure alguns de nous
- eliminar alguns que eren intrancendents per fer marketing (documents,...)

Els 10 segments generats amb les categories de jerarquia 3 que tenen assignades són:

- Allotjament: format principalment per categories d'allotjament i la landing, multimèdia i ofertes corresponents. S'exclouen els allotjaments rurals i albergs
- Cercadors: Cercadors de cada categoria de contingut i el cercador general
- Esdeveniments: Agenda Altres, Cultura, Esquí i Natura; Microsites (inclou Scalada)
- Gastronomia: Menjar i la landing corresponent
- Informació i Cultura: Activitats Cultura; Establiments Cultura; landing i ofertes corresponents; Informació general i Estat
- Natura: Activitats Natura; Albergs, Allotjament rural i Refugis; Establiments Natura i Esport; Itineraris; landings, multimèdia i ofertes corresponents
- Neu: Activitats Esquí; Establiments Esquí; landing, multimèdia i ofertes corresponents
- Shopping: Activitats Shopping; Agenda Shopping; Establiments Shopping; landing, multimèdia i ofertes corresponents
- Wellness: Activitats Wellness; Agenda Wellness; Establiments Wellness; landing, multimèdia i ofertes corresponents
- AltresContingutsOfertes: Resta de continguts no classificats

Cada usuari que visita la web visitarà una plana associada amb un d'aquests 10 segments de tal forma que després d'una visita quedaran enregistrats tots els segments de contingut que han sigut visitats per l'usuari.

El fitxer que recull aquestes dades serà després de ser processat convenientment utilitzat per fer la segmentació. La segmentació de contingut cercarà regles d'associació 2x2 utilitzant l'algoritme 'eclat'.

Per utilitzar aquest algoritme se li han de passar tres paràmetre a la funció eclat (el dataframe, el mínim i màxim nombre de ítems que es volen associar (min=2 max=2) en el nostre i un tercer paràmetre amb el qual ens ordena els resultats). El dataframe ha de tenir el seu contingut en forma de variables booleana TRUE/FALSE o 1/0 quan l'usuari ha visitat el segment o no.

Els resultats d'algoritme són un conjunt de regles d'associació amb un segon paràmetre cadascuna que és el 'support' relacionat estretament amb la probabilitat de que es produeixi la regla. Altres paràmetres importants són la 'confianza', definits com la probabilitat condicionada de que donat A es produeixi B i el i el 'lift'. D'aquesta manera les regles poden sortir ordenades de més freqüents a menys freqüents

No es considera recomenable trobar regles d'associació on intervinguin més de dues categories perquè podrien sortir molts segments. Això en principi podria ser bo per classificar be l'usuari però seria poc pràctic per crear col·lectius que fossin 'target' de campanyes i podria arribar a complicar la gestió d'aquestes.

items	support	allConfidence
{AltresContingutsOfertes,Cercadors,Natura,Shopping}	2,86%	7,53%
{Cercadors,InformacioCultura,Natura,Shopping}	1,79%	5,94%
{AltresContingutsOfertes,Cercadors,InformacioCultura,Shopping}	2,05%	5,41%
{AltresContingutsOfertes,Cercadors,InformacioCultura,Natura}	2,05%	5,41%
{Allotjament,AltresContingutsOfertes,Cercadors,Natura}	1,98%	5,21%
{Allotjament,Cercadors,Natura,Shopping}	1,48%	4,92%
{Cercadors,Gastronomia,Natura,Shopping}	1,38%	4,60%
{Cercadors,Natura,Shopping,Wellness}	1,32%	4,38%
{AltresContingutsOfertes,Cercadors,Gastronomia,Shopping}	1,64%	4,32%
{AltresContingutsOfertes,InformacioCultura,Natura,Shopping}	1,61%	4,25%
{AltresContingutsOfertes,Cercadors,Natura,Wellness}	1,58%	4,17%
{AltresContingutsOfertes,Cercadors,Gastronomia,Natura}	1,58%	4,16%
{Allotjament,AltresContingutsOfertes,Cercadors,Shopping}	1,56%	4,10%
{AltresContingutsOfertes,Cercadors,Shopping,Wellness}	1,37%	3,62%
{Cercadors,InformacioCultura,Natura,Wellness}	1,03%	3,44%
{Allotjament,Cercadors,InformacioCultura,Natura}	1,02%	3,38%
{AltresContingutsOfertes,Natura,Shopping,Wellness}	1,27%	3,36%

Fig. – Una mostra de Regles obtingudes

Profiling d'usuari :

Aquest procés permet el procés de Profiling d'usuari amb la Segmentació de contingut. D'aquesta manera quan un usuari nou es registra en base a la seva navegació es pot predir possibles productes i serveis del seu possible interès després de analitzar les correspondències de les navegacions de usuaris anteriors.



Fig. -Flux de procés de recomanació (Segmentació usuari)

La segmentació de de contingut es du a terme en una primera instància amb R i posteriorment es passa a Spark MLib.

```
#Procediment ECLAT en R

#http://www.academia.edu/9768212/Software_y_association_rules_2010_1
#https://es.wikipedia.org/wiki/Reglas_de_asociacion

setwd("C:/posgrado Big Data/proyecto/whole_navigation")
memory.limit(size=96784)

cn <- paste("v", 1:10, sep = "")
z1 <- read.table("dataeclatin1.txt", fill=TRUE,sep="|", col.names = cn,nrows= 6)
tr <- as(z1[,-1], "transactions")

tr
inspect(tr)
summary(tr)
```

```
j <- as(tr,"matrix")

#calculo Association Rules (ECLAT)
rulesEC <- eclat(tr)

#parametros ejecución y control por defecto--
rulesEC

#conjunto(set) de reglas de asociacion --
inspect(rulesEC)
```

El procediment de Eclat en R es integrat a la solució mitjançant Spark MLlib amb les llibreries y comandos següents:

```
import org.apache.spark.SparkFunSuite
import org.apache.spark.mllib.util.MLlibTestSparkContext

class FPGrowthSuite extends SparkFunSuite with MLlibTestSparkContext {

  test("FP-Growth using String type") {
    val transactions = ... Dataset_ whole_navigation ...
      .map(_.split(" "))
    val rdd = sc.parallelize(transactions, 2).cache()

    val fpg = new FPGrowth()

    val model6 = fpg
      .setMinSupport(0.9)
      .setNumPartitions(1)
      .run(rdd)

    ...
  }
}
```

Altres tecnologies existents

Es presenten a continuació les tecnologies disponibles actualment al mercat que han sigut estudiades per desenvolupar la part analítica del projecte.

Es mostra a continuació una taula amb les principals característiques de cadascuna d'elles:

Eina	Característiques
Weka	Suporta el llenguatge Java. Conté eines para preprocessat, classificació, regressió, <i>clustering</i> , visualització i <i>Machine Learning</i> .
Spark:	
SparkR	SparkR: <i>FrontEnd</i> que permet utilitzar <i>Apache Spark</i> en <i>R</i> . Permet també la utilització de <i>Machine Learning</i> amb <i>MLlib</i> . Està preparat per treballar amb <i>dataFrames</i> de gran volum de dades.
PySpark	PySpark: Permet l'adaptació del model <i>Spark</i> per treballar amb <i>Python</i> , amb les característiques pròpies d'aquest llenguatge.
ML	ML: Utilitza las API'S de SPARK. Interactua amb la Llibreria numpy Python (Spark 0.9) y llibreries R (Spark 1.5) Fàcil integració en l'entorn <i>Hadoop</i> Suporta llenguatge Java, Python, Scala y R.

Kinesis Amazon	Té un mòdul analític que permet analitzar <i>streamings</i> de dades amb sentències SQL estàndard. Utilitza els mòduls <i>Kinesis Steams</i> i <i>Kinesis Firehouse</i> per fer la ingesta de dades. Ofereix a més bones eines de visualització. Tot això la fa una eina atractiva per desenvolupar aplicacions analític-predictives que hagin de visualitzar resultats en temps real.
ML Azure	Treballa en un entorn anomenat <i>Machine Learning Studio</i> . Sobre aquest entorn les dades que volem analitzar i el recursos analítics i predictius de l'eina es troben per poder desenvolupar aplicacions. Es a dir, s'accedeix a l'entorn amb un <i>account</i> i un cop l'usuari s'ha connectat es pugen a l'entorn els <i>datasets</i> que es volen utilitzar.
KNIME	Està desenvolupat sobre <i>Eclipse</i> y programat fonamentalment e Java . Es una eina Gràfica amb nodes (que encapsulen algoritmes)y fletxes (que representen fluxos de dades) que es combinen entre elles. Permet cridar directament a <i>Weka</i> . Permet incorporar de forma senzilla codi en R o <i>Python/jpython</i> . Inclou components de <i>Machine Learning</i> i de <i>Data Mining</i> .
PENTAHO	Te una versió de pagament i un altre <i>free</i> . Es una plataforma que ofereix serveis per fer Anàlisi de Dades, Informes empresarials, <i>Data Mining</i> i Integració de dades com aspectes més rellevants.

Per obtindre informació més detallada de cadascuna d'aquestes eines s'ha de consultar els links següents:

Eina	Link
Weka	http://www.cs.waikato.ac.nz/ml/weka/
Spark:	http://spark.apache.org
SparkR	https://spark.apache.org/docs/2.0.0/sparkr.html
PySpark	https://spark.apache.org/docs/0.9.0/python-programming-guide.html
ML	http://spark.apache.org/mllib/
Kinesis Amazon	https://aws.amazon.com/es/documentation/kinesis/
ML Azure	https://azure.microsoft.com/en-us/documentation/articles/machine-learning-what-is-ml-studio/
KNIME	https://www.knime.org/
PENTAHO	http://www.pentaho.com/embedded-analytics-platform

II.4.2.-Solució escollida

Per la seva versalitat, fiabilitat i velocitat, i a més a més per ser OpenSource es tria Spark per desenvolupar la part analítica.

Spark es una eina de processament de dades a gran escala ràpida i fàcil d'usar. Es poden programar aplicacions utilitzant diferents llenguatges com Java, Scala, *Python* o R. Es molt més ràpid que Hadoop MapReduce, (depenen de les aplicacions, fins 100 vegades més ràpid en memòria). Permet consultar *queries* SQL de bases de dades com HBase i Cassandra, consultar i preparar streams de dades (*Spark Streaming*) i té llibreries analítiques avançades (*MLlib*). Pot funcionar sobre *Hadoop*, en format *standalone* o en *cloud*.



Spark MLlib, ha estat seleccionada per desenvolupar la part de Segmentació dins d'una plataforma Apache Spark, que ofereix la possibilitat de triar entre *Spark R*, *Spark Python*, *Spark Java* y *Spark Scala*.

Les raons principals per triar aquesta opció han sigut trobades tractant de respondre a las següents preguntes:

1. Quin tipus de problemes es pretén resoldre ?

La segmentació y *profiling* a realitzar son plantejades des d'un punt de vista y anàlisi estadístic de les dades. El llenguatge por excel·lència per aquest tipus de processos es R, especialment dissenyat per fer més fàcil i simples els càlculs estadístics. Encara que Java y Python son llenguatges més utilitzats a nivell general, R ha estat concebut per tractar problemes estadístics. Scala, a més de no oferir aquesta avantatge, es el menys utilitzat dels quatre.

El projecte se emmarca dintre d'una arquitectura Apache Spark i la darrera versió de Rstudio permet a més de la traducció de codi R a Spark, instal·lant els mòduls apropiats. Per aquesta primera raó es per la que es considera Spark MLlib el més adequat per realitzar la segmentació a partir del fitxer de les navegacions de la web, seguit de l'Spark Python.

2. Quin costes suposa implementar el llenguatge i la tecnologia?

Es pot desenvolupar codi R com Python amb eines Open-Source, per la qual cosa potser tingui mes pes triar tenint en compte el cost de aprenentatge que pròpiament l'econòmic.

En general, per un principiant, Python podria oferir una corba de aprenentatge més plana que R. No obstant, per la totalitat de los membres de l'equip el llenguatge R és més familiar degut als coneixements adquirits al curs de Posgrau. D'altre banda Python es conegut tant sols per algun dels membres degut a experiències prèvies.

Donat que el *background* global de l'equip en llenguatge R es major que el de *Python*, aquesta és la segona raó per seleccionar R o Spark MLlib, en detriment de *Python*.

3. Quina velocitat y visibilitat ofereixen les diferents solucions?

R podria arribar a ser més lent que *Python* a l'hora de fer càlculs, però existeixen molts paquets enfocats en optimitzar el seu rendiment: *pqR*, *renjin* and *FastR*, *Riposte* son alguns d'ells, però n'hi ha més. Per altre costat, la cerca de paquets R necessaris pot ser més ràpida utilitzant R que *Python*, degut a les raons comentades a l'apartat anterior.

Respecte a *Python*, a més, R sembla oferir millors eines de visualització encara que això te una importància relativa ja que al projecte utilitzarem *Tableau* per les visualitzacions.

4. Quines son les eines més comunes utilitzades en aquest tipus de problemàtiques?

Tant *Python* com R se utilitzen en aquest tipus de problemàtiques, ambdues opcions son cada cop més utilitzades y amb comunitats més actives. Tots els paquets de R poden trobar-se en CRAN.

Conclusions:

- Per la seva fiabilitat, versatilitat i velocitat s'ha triat la tecnologia Spark per fer també la part analítica.
- Dels quatre llenguatges que poden utilitzar amb arquitectura Spark (Python, R, Java,Scala) només es valoren les prestacions de Python i R enfront de Java i Scala, car aquests dos llenguatges son més apropiats per tractar càlculs estadístics com els que es poden trobar al procés de segmentació.
- Valorats les avantatges e inconvenients de Python i R, es decideix utilitzar R per fer el procés de Segmentació i traduir-ho a Spark MLlib per poder implementar-ho dintre de l'arquitectura del prototip.

II.2.3.1. – ClickStream amb Spark Streaming



En una speed layer *Spark* rebirà un *DStream* del que sobretot guardarà la informació rebuda per determinar *a posteriori* si les entrades de la *cookie* rebudes es consideren una nova visita o no.

Es farà ús d'unes funcions que fan una finestra del *stream* en qüestió i la va desplaçant a mesura que arriben noves dades, cosa que ens permetrà mantenir la informació per identificar noves visites.

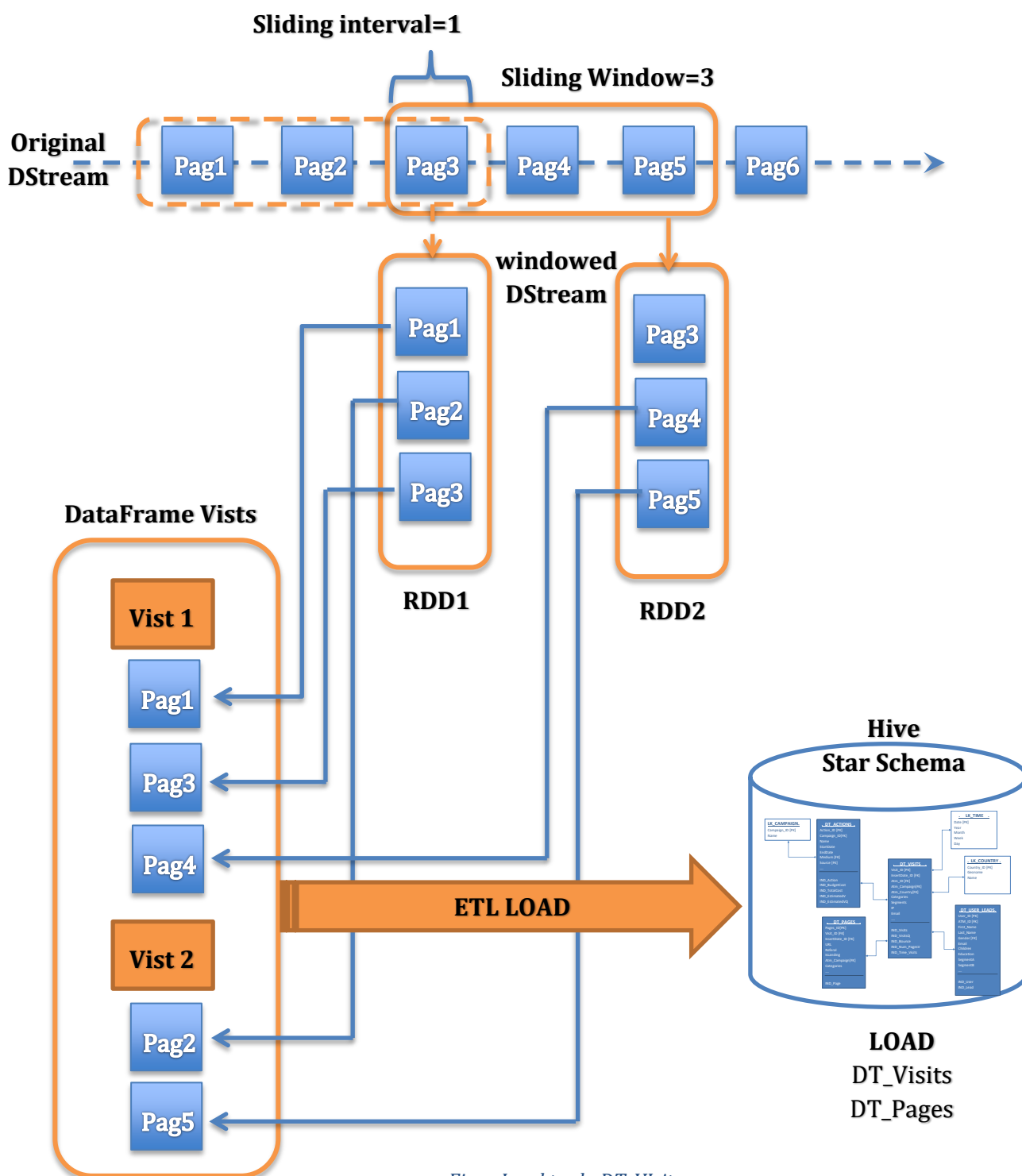


Fig. – Load taula DT_Visites

Transformació	Propòsit
<i>ReduceByKeyAndWindow</i>	Retorna un nou <i>DStream</i> de parelles (<i>K</i> , <i>V</i>) on els valors per a cada clau s'agreguen utilitzant una funció donada sobre els <i>batches</i> en un <i>sliding window</i> .
<i>UpdateStateByKey</i>	Actualitza l'estat de cada <i>key</i> aplicant una funció al seu estat previ.

Així al llarg del temps els nous missatges van sent avaluats per *Spark Streaming* que mantindrà i actualitzarà en una taula temporal (on hi tenim les visites no tancades), tancant (eliminant de la taula temporal i afegint a la taula de visites de *Hive*) les visites de les *cookies* que correspongui o mantenint la resta.

II.2.3.2. – ETL Job amb Spark

Un cop es disposa dels Dataframe temporals carregats amb les visites i les planes de client es procedirà mitjançant una ETL amb Spark a carregar la DT_Visita i la DT_Pagines de les visites en el model estrella en HIVE.

```
# Amb rescritura de la taula externa de Hive
df.write.mode(SaveMode.Overwrite).parquet("/UPC_Pilot/etl/destination")

# Sense rescritura de la taula externa de Hive
df.write.mode(SaveMode.Append).parquet("/UPC_Pilot/etl/destination")
```

II.2.4.1.- Model Dimensional OLAP

Actualment Alexandria empra una eina de Business Intelligence per realitzar la visualització i navegació en les dades. S'importa tota la informació d'uns datamarts disposats per tal efecte. Això implica que l'eina de BI ha de carregar amb les tasques pròpies de modelat de cubs i funcions OLAP. Part d'aquets datamarts presenta un volum de milions de registres cosa que provoca que el BI tingui un nivell de resposta lent, essent en alguns moments inoperatiu pels usuaris.

Altres limitacions son l'impossibilitat de creuar certs dominis de informació donat el gran volum d'informació a creuar. Per aquest motiu, s'ha plantejat en la nova arquitectura incloure una nivell multidimensional distribuït que permeti treure carrega de computació al BI, i aporti una experiència de resposta en la visualització i navegació satisfactoris pel usuari.

Tecnologies existents

Hi ha certes solucions que permete crear una capa de dimensionalitat en Hadoop, i que passarem a referència com a fonts bibliogràfic, ja que no es objectiu d'aquest document. Aquestes son :

- Apache Phoenix¹⁰ : OLAP with Apache Phoenix, HBase and Saiku
- Apache Shark¹¹: Shark, Spark SQL, Hive on Spark, and the future of SQL on Apache Spark
- FiloDB¹²: Open-source distributed analytical OLAP database

Solució escollida



*Apache Kylin*¹³, el qual presenta un motor d'anàlisi de codi obert i distribuït dissenyat per proporcionar una interfície l'anàlisi i creació de models de dades multidimensionals (OLAP) en Hadoop, suportant un gran volum de dades. Original van contribuir el projecte la companyia eBay Inc.

La arquitectura de kylin es basa en la pre-existència de un esquema en estrella en el warehouse *Hive*. Aquest en el projecte Alexandria caldrà que sigui pre-carregat mitjançant una ETL amb Spark extraent les dades per un costat del Data Lake en HDFS en el que fa referència al Campaign manager i de les ETLs provinents de la Speed Layer pel que fa al ClickStream.

Recentment la comunitat Kylin ha anunciat la integració amb Spark SQL¹⁴ com a procés de càlcul dels cuboids que actualment es duen a terme internament amb Map Reduce, circumstancia que permetria optimitzar x10 el temps de resposta, però caldrà valorar la idoneïtat de la madures d'aquest canvi seu us en el moment del dur a terme el desenvolupament del prototip.

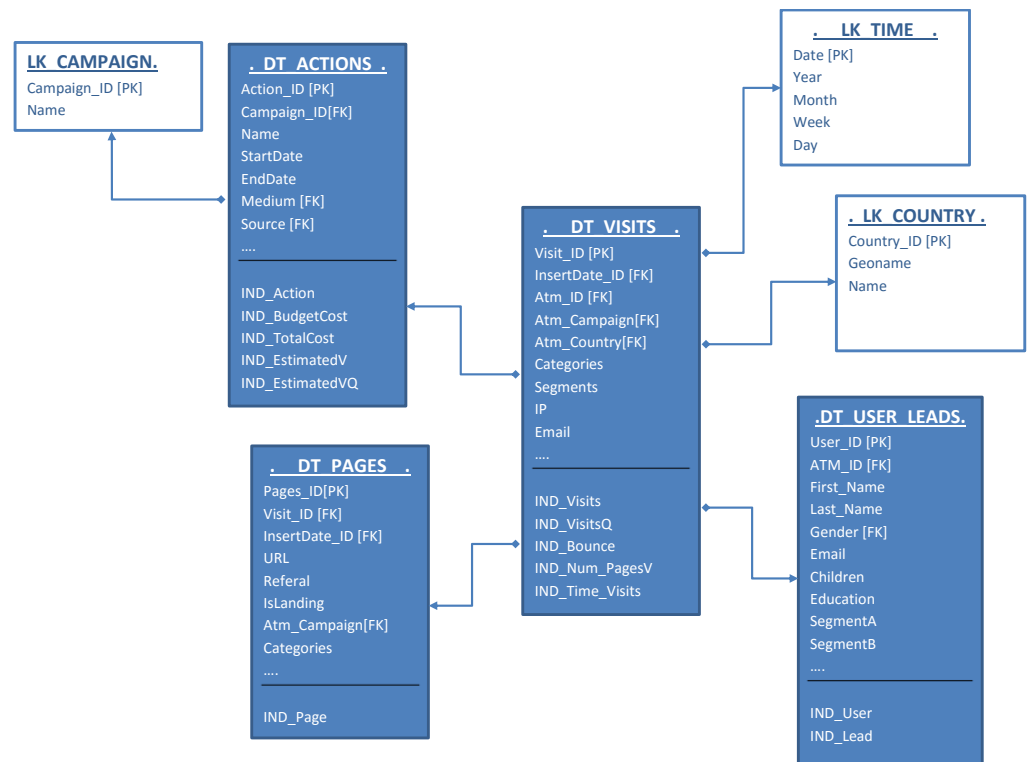


Fig. – Star Schema model Hive

¹⁰ https://blogs.apache.org/phoenix/entry/olap_with_apache_phoenix_and
¹¹ <https://databricks.com/blog/2014/07/01/shark-spark-sql-hive-on-spark-and-the-future-of-sql-on-spark.html>
¹² <http://es.slideshare.net/EvanChan2/filodb-breakthrough-olap-performance-with-cassandra-and-spark>
¹³ <http://kylin.apache.org/>
¹⁴ <http://kylin.apache.org/blog/2015/09/09/fast-cubing-on-spark/>

El model en estrella situat a Hive i que serveix a Kylin com a case dels cubos a construir disposar de les taules DT (de fets) i LK (de detall) així com les dimensions i les mesures que cal sumaritzar que seran emprats a posteriori per constituir els reports i dashboards.

Prosseguint en el procés de construcció dels cubos, les mètriques que com s'indicava son sumaritzades mitjançant processos Map Reduce es guardaran en la BBDD columnar NoSql HBase.

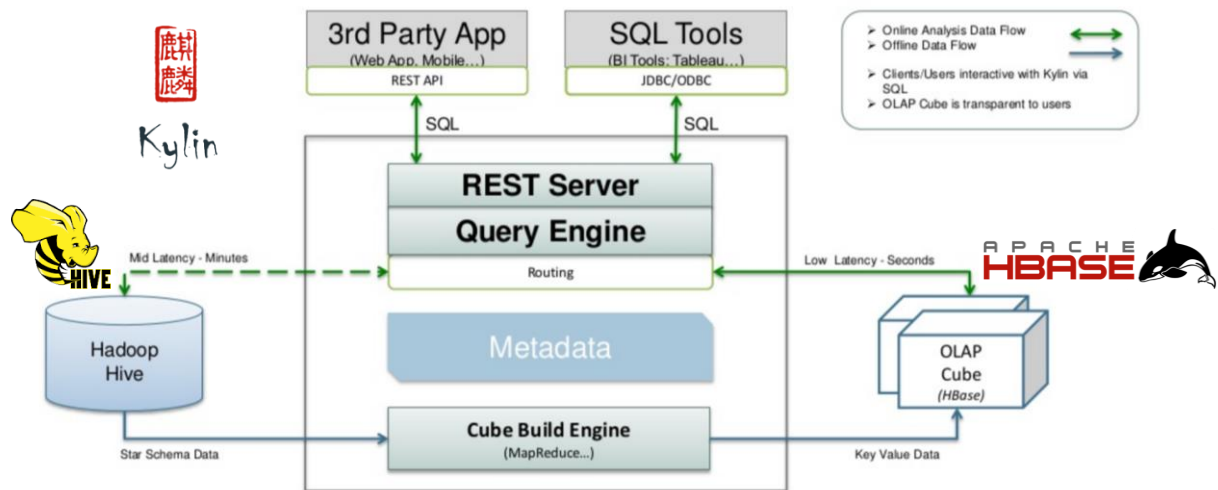


Fig. – Arquitectura interna Apache Kylin

Els sistemes encarregats de crear cubos solen crear a la seva vegada infinitats de cuboides que representen totes les combinacions que apareixen al creuar totes les dimensions. Cadascun d'aquesta combinació requereix de rondes de Mapreduce que realitzen els creuaments.

Kylin per optimitzar aquest procés a anat evolucionant⁹⁹ els diferents mètodes de creació dels cuboides¹⁵: Layered Cubing, Fast Cubing, in-memory cubing. Cadascun d'ells requerirà d'un tractament i manteniment diferenciat.

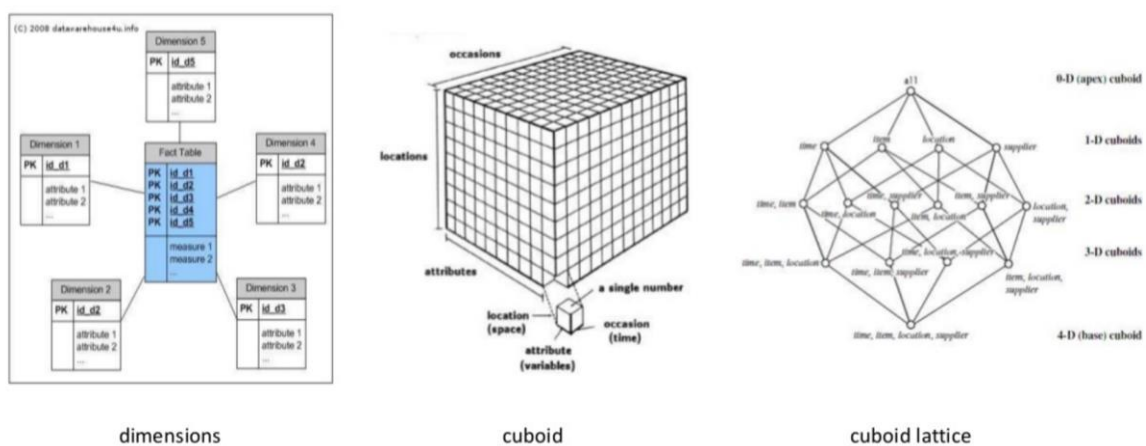


Fig. – Estratègia de creació dels cubos

Finalment els cubs obtinguts s'emmagatzemaran en un esquema HBase, valgui com exemple la següent figura respecte el model anterior :

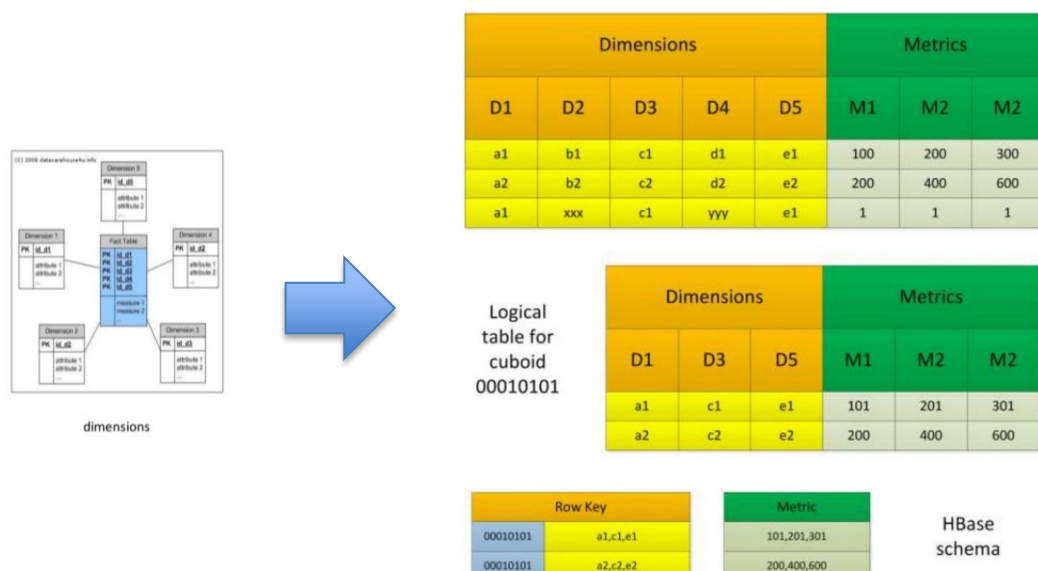


fig. – Esquema HBase resultant de la construccions dels cubs

Aquest procés requerirà d'un data modeling que caldrà desenvolupar i mantenir :

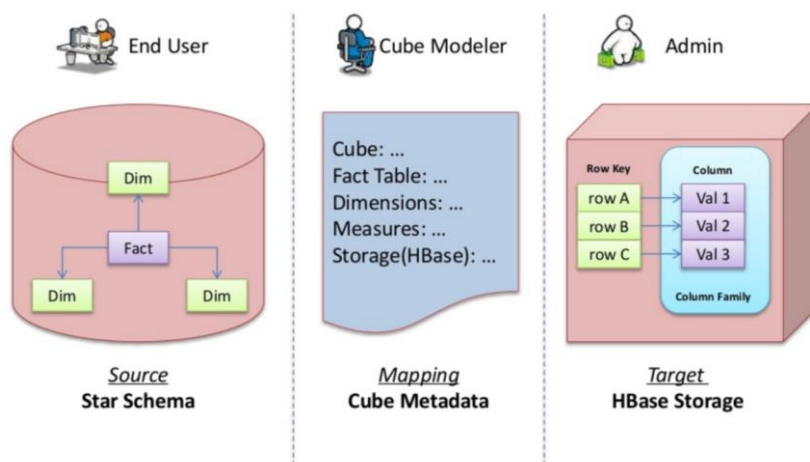


Fig. – Procés de modelat de dades origen i cubs destí

Funcionalitats resum que ens aporta Kylin:

- Motor OLAP extremadament ràpid: Kylin està dissenyat per reduir la latència de consulta en Hadoop per a més de 10 mil milions de files de dades

- *Interfície d'ANSI SQL en Hadoop: Kylin ofereix ANSI SQL en Hadoop i és compatible amb la majoria de les funcions de consulta ANSI SQL*
- *La capacitat interactiva de consulta: Els usuaris poden interactuar amb les dades de Hadoop via Kylin en menys d'un segon de latència, millor que les consultes del Hive pel mateix conjunt de dades.*
- *MOLAP Cub: L'usuari pot definir un model de dades i pre-construcció en Kylin amb més de 10 milers de milions de registres de dades en brut*
- *Perfecta integració amb eines de BI: Kylin actualment ofereix una capacitat d'integració amb eines de BI com Tableau, PowerBI i Excel.*
- *Altres destacats:*
 - *Gestió del cicle de vida dels treballs en curs, ETLs i monitorització*
 - *Suport de compressió i codificació*
 - *Actualització incremental dels cubs*
 - *Aprofitament del coprocessador de Hbase per a consultes de latency*
 - *Capacitat de consulta aproximat de recompte diferent (HyperLogLog)*
 - *Fàcil interfície de Web per administrar, construir, controlar i cubs de consulta*
 - *Capacitat de Seguretat per configurar l'ACL en el Cub i de Projecte*
 - *Suport d'integració a LDAP*

Els nivells de rendiment respecte una solució de consulta directa de un BI sobre Hive son substancialment millors, com es pot apreciar en aquesta comparativa de temps respecte Hive :

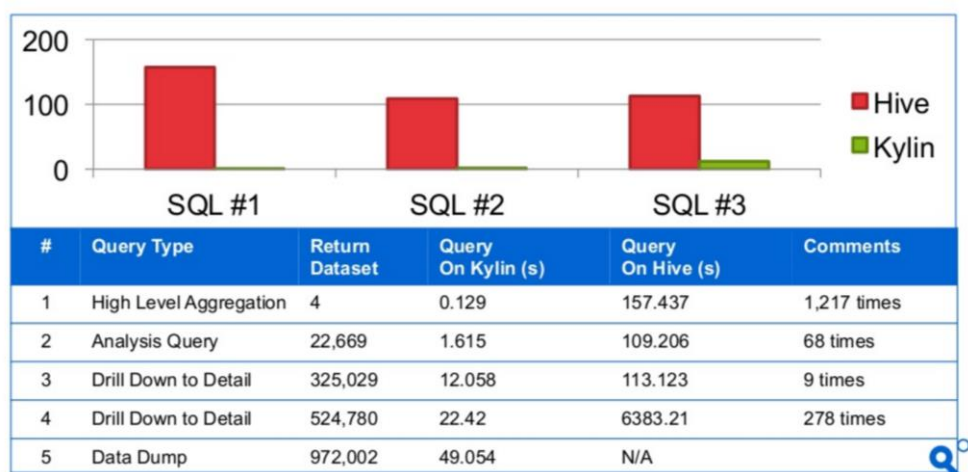


Fig. – Taula comparativa de rendiment en consultes Kylin vs Hive

II.2.4.2.- Business Intelligence amb Tableau

Un cop disposat la capa de dimensionalitat creada amb els cubs creats per cadascun dels insights i dashboardas a crear, podem emprar la eina de Business Intelligence per visualitzar i explorar les dades. El model de dimensionalitat esmentat en el punt II.2.4.1, permetrà emprendre en majors garanties i millors


temps de resposta les tasques pròpies del descobriment y navegació entre dades (Drill-up/Down).

Tecnologies existents

Existenixen un gran nombre de eines de BI existents en el mercat, encara que son poques encara les que s'integren plenament en entorns Big Data. Les principals son:

- Tableau¹⁶
- Qlick Sense¹⁷
- Pentaho¹⁸
- Microstrategy¹⁹
- MS PowerBI²⁰
- Sisense²¹

Solució escollida

 La eina de BI escollida ha estat Tableau, donat que disposa de totes les capacitats OLAP requerides para el projecte, així com una integració nativa amb Kylin mitjançant un ODBC de connexió propi.

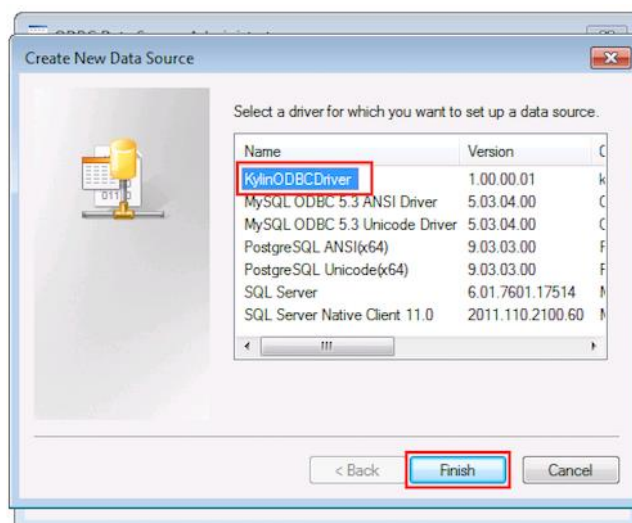


Fig. – Connector ODBC natiu entre Kylin i Tableau

Un cop connectada les dues plataformes podrem accedir a tots els cubs generats i realitzar aquells insights i Dashboards requerits. Per fer-ho, podrem accedir al metadades existent en Kylin on podrem disposar de les dimensions i mesures.

¹⁶ <http://www.tableau.com/es-es>

¹⁷ <http://global.qlik.com/es/>

¹⁸ <http://www.pentaho.com/>

¹⁹ <https://www.microstrategy.com/es>

²⁰ <https://powerbi.microsoft.com/es-es/>

²¹ <https://www.sisense.com>

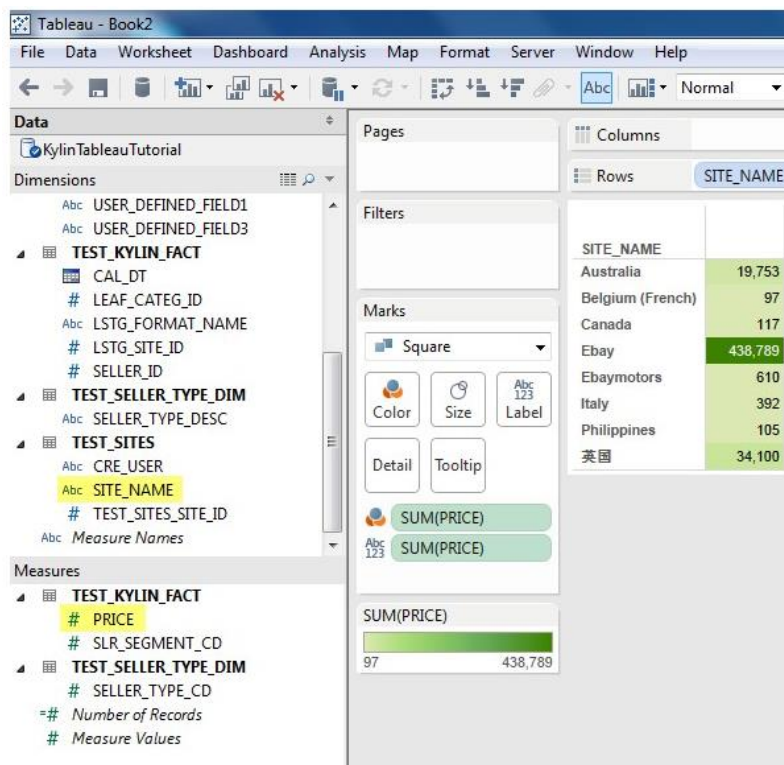


Fig. –Accés a les dimensions i mesures del model dimensional

Un cop dispossem a tot el metadades i cubs de Kylin podem crear els reports, infografies i Dashboards requerits.

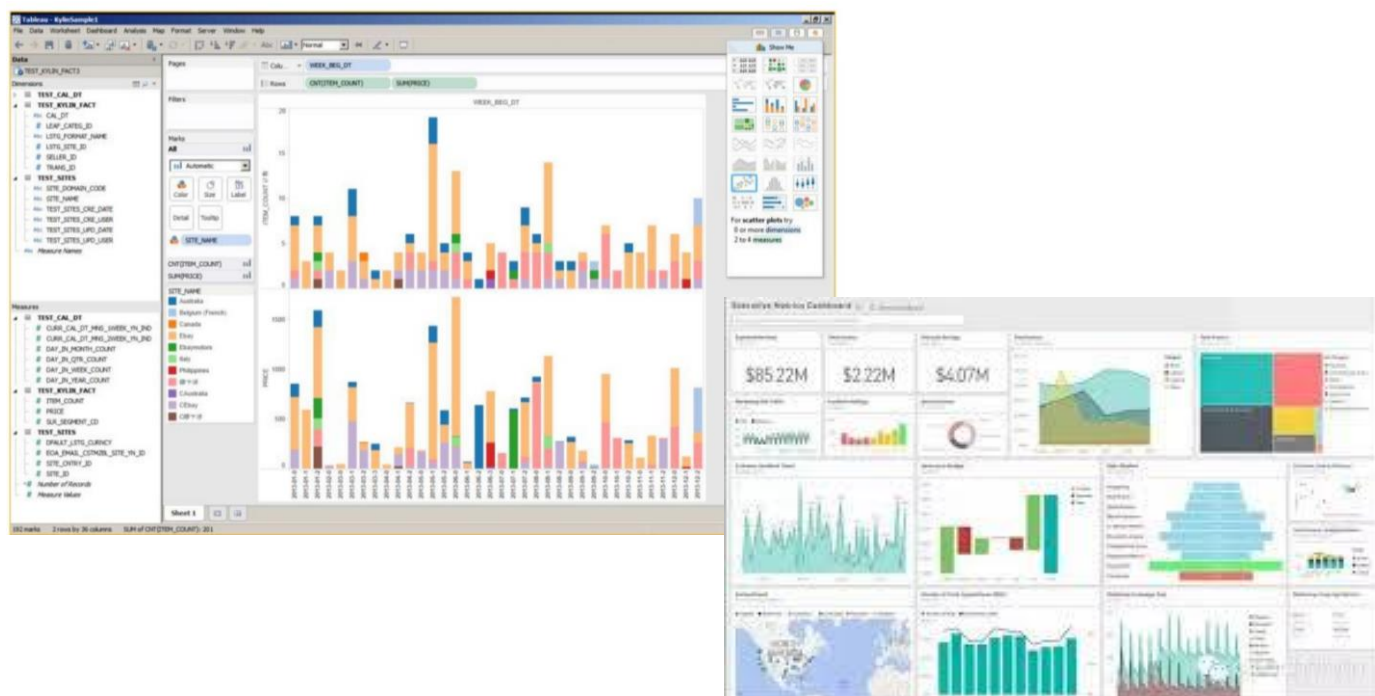


Fig. –Insights i Dashboards enel BI Tableau

II.3.- Conclusions Finals

L'objectiu principal que ens em establert els membres de l'equip ha estat poder conèixer de primera mà aquelles tecnologies i arquitectures existents en el entorn Hadoop. Es partia amb el objectiu molt ambiciós de poder reproduir part de la capacitat analítica de la Solució Alexandria i per tant els reptes es elevats i complexos, tenint en compte el temps i el nombre integrants.

Dit això, i ja reconeixent que en cara ens trobem desenvolupant part dels processos de carrega dels diferents models de dades, podem dir que creiem hem assolit un coneixement per poder valora amb un cert criteri aquelles solucions i alternatives existents per resoldre una problemàtica que es pugues plantejar en el àmbit professional en el entorn del Big Data.

Els reptes i dificultats han estat en certs moments grans donat que el stack de hadoop esta canviant contínuament i hi ha realment poca bibliografia i exemples quan fas referencia a certs mòduls. En especial, hem i seguim tenint dificultats amb la darrera versió de Spark SQL i Streaming. També ha estat una constant els problemes de configuració dels mòduls del stack de Hadoop per que treballessin junts, així com també encertar amb les dependències entre mòduls en els projectes Eclipse-Maven.

Estem satisfets del que hem après, tot reconeixent que encara hem de seguir profunditzant més encara amb tecnologies com Spark i Streaming, així com també analitzar altres aspectes que un projecte real requereixen com són el Data governance, security i infraestructura dels sistemes.

Esmentar com a millora de cara a propers post-graus, la necessitat de plantejar l'aprenentatge de Spark en el llenguatge de programació Scala, per la seva senzillesa, potencia i extensió.

III.- Recursos del Projecte

- III.1.- Codi font Bomber
- III.2.- Configuració de Kafka
- III.3.- Configuració de Flume
- III.4.- Codi ETL Job Batch layer (Users & Campaign)
- III.5.- Codi ETL Job Speed layer de Hive (Visits)
- III.6.- Model estrella en Hive
- III.7.- Model de dades dimensional amb Kylin
- III.8.- Codi Segmentació i Classificació amb Spark MLlib
- III.9.- Insights i Dashboards amb Tableau

NOTA: Els següents apartats seran lliurats a mesura que es va desenvolupant i finalitzant el prototip.

V. Bibliografia

Raman Jhajjk, Apache Hadoop Cookbook, Java Code Geeks, 2016

Tom White, Hadoop The Definitive Guide, O'Reilly, 2009

Judith Bishop, Java Gently (3rd Edition), Addison-Wesley, 2015

Jason Rutherglen, Programming Hive, O'Reilly, 2012

Bahaaldine Azarmi, Scalable Big Data Architecture, 2016

Nick Dimiduk, HBase in action, Manning, 2013

Apache Flume <http://hortonworks.com/apache/flume/>

Spark Streaming + Flume Integration Guide

<http://spark.apache.org/docs/latest/streaming-flume-integration.html>

Spark Streaming Programming Guide

<http://spark.apache.org/docs/latest/streaming-programming-guide.html>

Hortonworks Data Platform Kafka Guide, 2015

HDP Non-Ambari Cluster Installation Guide, 2015

KAP User Manual

<https://kyligence-git.gitbooks.io/kap-user-manual/content/en/>

Sandy Ryza, Advanced Analytics with Spark, 2015

Pat O'Neil, Betty O'Neil, Xuedong Chen, Star Schema Benchmark, June 5, 2009

Yanchang Zhao (<http://www.rdatamining.com/>)

R and Data Mining: Examples and Case Studies , April 26, 2013

Wangechi Doble

Comparing the Use of Amazon DynamoDB and Apache HBase for NoSQL, Sept 2014

Blagoy Kaloferov

Building a Data Warehouse for Business Analytics using Spark SQL

<https://www.youtube.com/watch?v=gsR1ljgZLq0>