PROCESAMIENTO REAL TIME EN CLARO ARGENTINA USANDO KAFKA, NIFI Y FLINK

GUSTAVO VARISCO
ARQUITECTO DE INFRAESTRUCTURA BIGDATA EN CLARO ARGENTINA
GVARISCO@CLARO.COM.AR

IVAN EZEQUIEL RODRIGUEZ
ARQUITECTO SENIOR DE SOFTWARE EN CLARO ARGENTINA
IVAN.RODRIGUEZ@CLARO.COM.AR

OBJETIVOS

En esta presentación vamos a hablar de los motivos que nos llevó a elegir en Claro Argentina Apache Kafka, Apache NiFi y Apache Flink para el procesamiento real time de datos y mediación.

Además mostraremos casos de uso y cantidades de eventos que hoy ya estamos procesando de manera productiva con dichos frameworks.

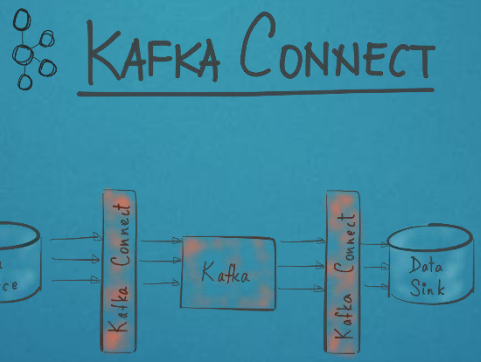
OBJETIVOS

Desde principios del 2017 en Claro Argentina estamos impulsando el uso de frameworks open sources para el procesamiento de grandes volúmenes de datos de tipo streaming.

La idea es que los mismos sean la base para el desarrollo de aplicaciones propias que sirvan tanto para nuevos casos de uso, como para el reemplazo de los que ya corren en soluciones propietarias.

TECNOLOGÍAS CORE DE LA ARQUITECTURA

SATÉLITES DEL CORE



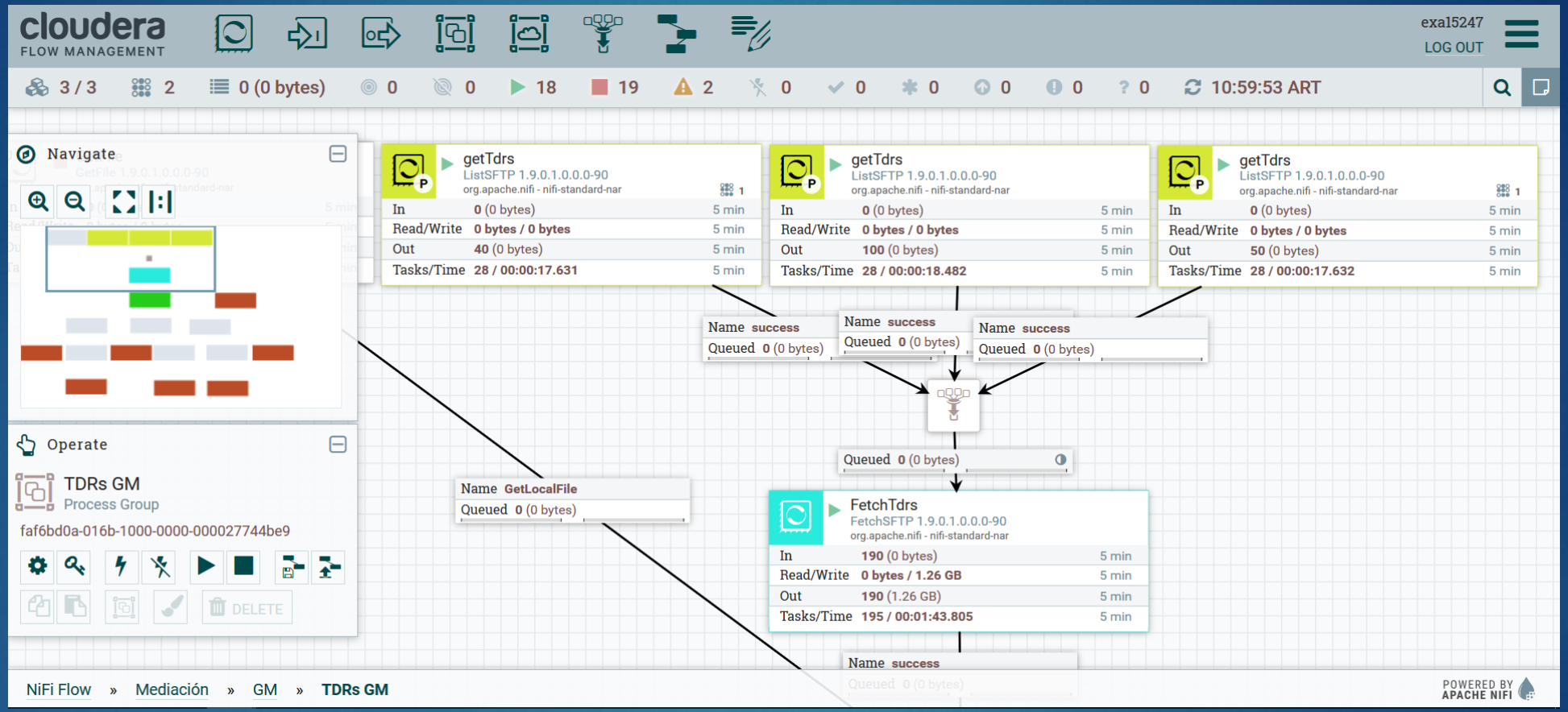
APACHE KAFKA
DISTRIBUTED STREAMING PLATFORM

- Sirve para publicar y subscribirse a un stream de eventos/mensajes, similar al concepto de frameworks de mensajería como ActiveMQ, RabbitMQ
- Los eventos están replicados (fault tolerant) y particionados (escalables)
- Garantía de procesar los eventos en el orden en el cual los mismos ocurren.

APACHE NIFI

- Está basado en el proyecto Niagara Files desarrollado por la NSA (National Security Agency) cuyo código fue liberado en 2014
- Automatiza el flujo de datos entre diferentes sistemas (Data pipelines, ETL Jobs).
- Procesamiento distribuido real-time de datos de tipo streaming por medio de sus procesadores.
- 300+ procesadores para conectividad, transformación y enriquecimiento de los datos (no-code).
- Extensible, permitiendo desarrollar fácilmente procesadores propios.
- Excelente GUI que facilita la construcción, control y monitoreo de los flujos de datos.

INTERFASE WEB



CARACTERÍSTICAS

- Ingesta de datos sin afectar el rendimiento de los servicios de la fuente ni del destino
- Soporte nativo de integración con amplia variedad de tecnologías tales como HDFS, SFTP, BD SQL, BD No-SQL, etc.
- Soporte nativo de transformación entre numerosos formatos de datos tales como JSON, XML, Avro, CSV, etc.
- Permite la delegación de funcionalidades a otros Sistemas como Apache Kafka (mediante colas) y Apache Flink
- Permite el rastreo de datos en tiempo real

APACHE FLINK

- Procesamiento distribuido real-time de datos de tipo streaming .
- Features avanzados como manejo de ventanas, agregación y joins de flujos.
- Manejo de estado para las operaciones statefuls (ventanas).
- Diseñado para correr aplicaciones distribuidas a gran escala.

¿COMO USAMOS CADA FRAMEWORK EN CLARO?

KAFKA

- Todo evento/mensaje que pueda ser consumido por más de una aplicación se publica en tópicos de Kafka.
- Toda aplicación real-time que necesite reaccionar ante un evento, dicho evento se publica en Kafka.

NIFI

- Todo procesamiento (transformación, enriquecimiento, etc) de un evento/mensaje puntual es procesado en primera instancia por NiFi (No Flink).

- NIFI actúa como plataforma de mediación entre los elementos de red que generan información materializada como archivos (ASN1,Binary, CSV, JSON, ETC). Esta información es procesada con NIFI, enriquecida y enviada a los diferentes sistemas.

FLINK

- Todo procesamiento que implique manejo de ventanas, agregación y joins de flujos es procesado en Flink.
- En lo posible los únicos “data sources” de Flink son tópicos de Kafka (Kafka Consumer).
- En lo posible los únicos “sinks” de Flink son tópicos de Kafka (Kafka Producer).

Kafka configuración y estadísticas

5 equipos físicos cada uno con:

- Cores: 28 (56 con Hyperthreading)
- Memoria: 512 GB
- Storage: 44 TB de disco en RAID 1 (22 TB real)
- Network: 10 Gb

Cantidad de bytes en hora pico:

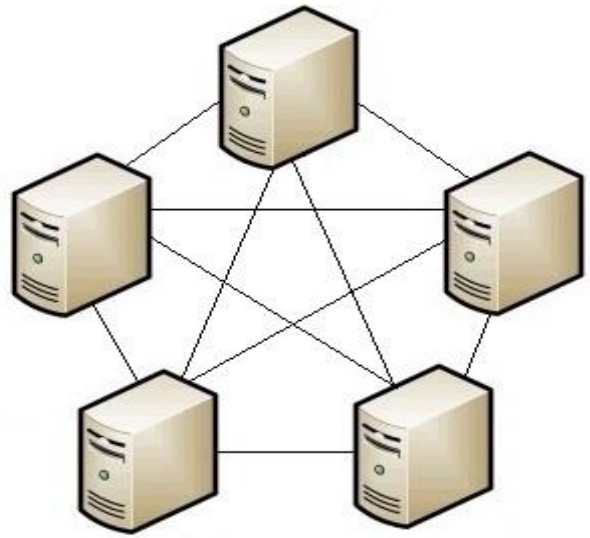
660 MB/segundo recibidos por el clúster
670 MB/segundo transmitidos por el clúster

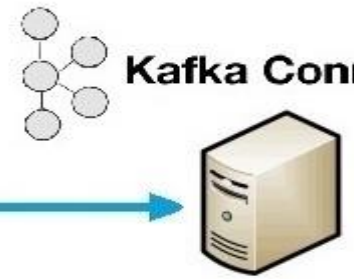
Ingesta de mensajes en hora pico:
485000 mensajes por segundo

CASO DE USO 1
FILEBEAT, KAFKA, KAFKACONNECT

1) FileBeat lee los archivos .csv en su ubicación en origen, lleva un registro del último punto de lectura de cada archivo y envía el contenido de cada archivo según lo va leyendo hacia el cluster Kafka.

2) En el Cluster Kafka se crean "topicos" que cumplen la función de colas en las cuales los mensajes son depositados para que cualquier consumidor pueda acceder a los mismos





3) Se desarrolló un conector utilizando el framework Kafka Connect, el cual lee los mensajes desde el cluster Kafka, los formatea según los requerimientos del cliente.

HDFS

4) Flume consume los mensajes desde Kafka y a traves de un Sink de HDFS deja los archivos directamente escritos en HDFS.

Filebeat, Kafka, KafkaConnect

Filebeat

- Log shipper liviano escrito en Go del Stack de Elastic.
- Usa pocos recursos, no tiene dependencias, amplia configuración según la necesidad (Por ejemplo: Multilínea)
- Lee directamente los archivos como se generan y envía cada línea a diversos tópicos de Kafka.
- Múltiples instancias de Filebeat fueron necesarias para poder paralelizar la ingesta a Filebeat.

KafkaConnect

- Conector Kafka desarrollado en Claro que procesa y transforma los mensajes de los tópicos de Kafka ingestados por Filebeat.
- Genera directamente en el destino archivos en formato CSV.

Números por día

- Hoy se está procesando un promedio de 22.000 millones de eventos.
- Filebeat lee y procesa 11 TB de archivos.
- El Conector Kafka genera 2.5 TB de archivos.
- A fin de año el número de eventos por día llegaría a 45.000 millones.

CASO DE USO 2
NIFI, REDIS, ORACLE, KAFKA

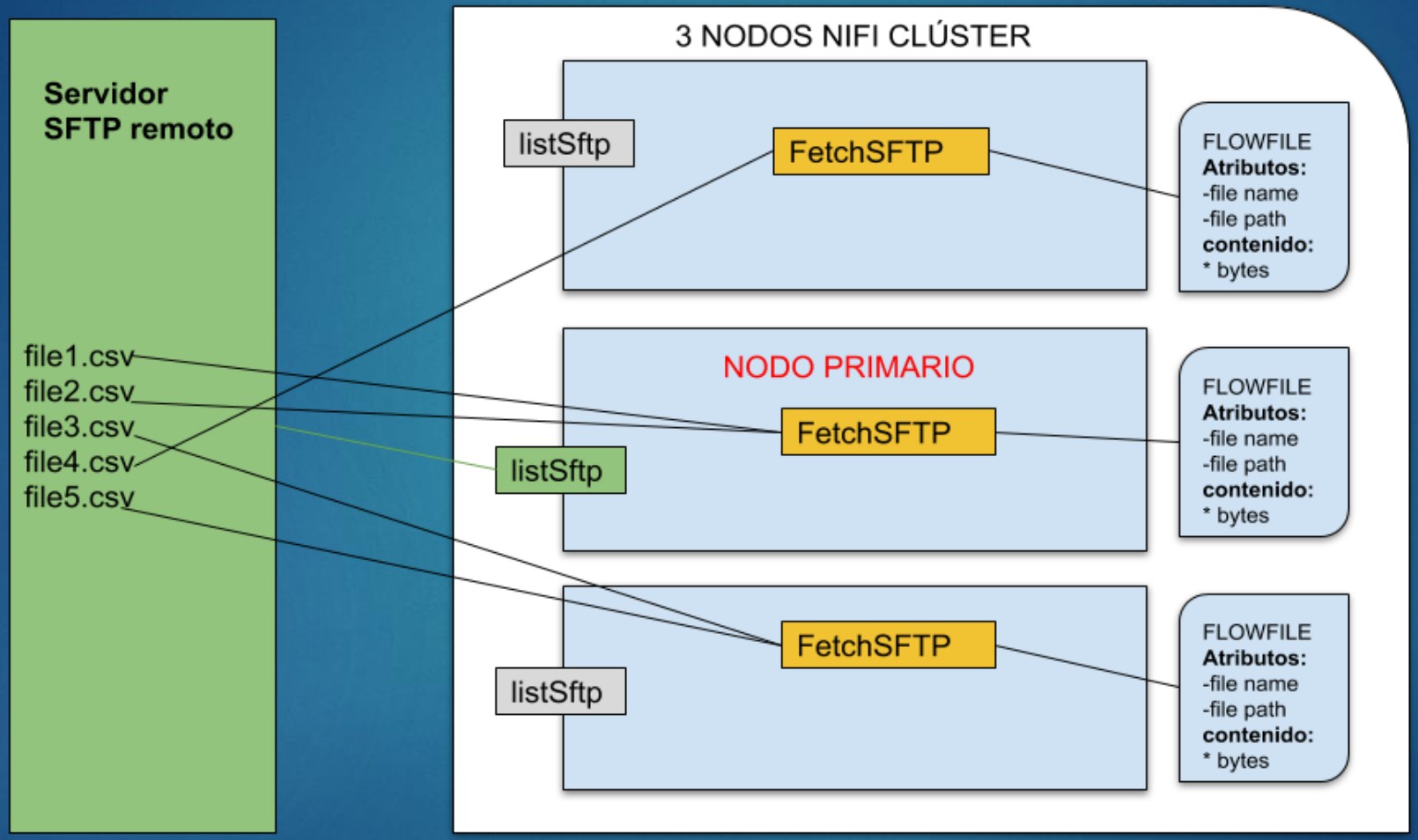
IMEI TRACKING DE EVENTOS DE RED

- Se procesan millones de eventos de RED para identificar los cambios de IMEI,IMSI o MSISDN que realizan los suscriptores con el objetivo de actualizar bases de datos locales y remotas (México) este ultimo mediante un API REST.
- Por cada evento se verifica si cambio la tripleta IMSI,IMEI,MSISDN.
- Si detectamos un cambio se genera un evento informativo que es encolado en Kafka.
- La arquitectura de la solución se basa en NIFI,REDIS,ORACLE y KAFKA.
- Se implemento un grupo de clústeres de Redis locales en los nodos de NIFI para usarlos como cache.

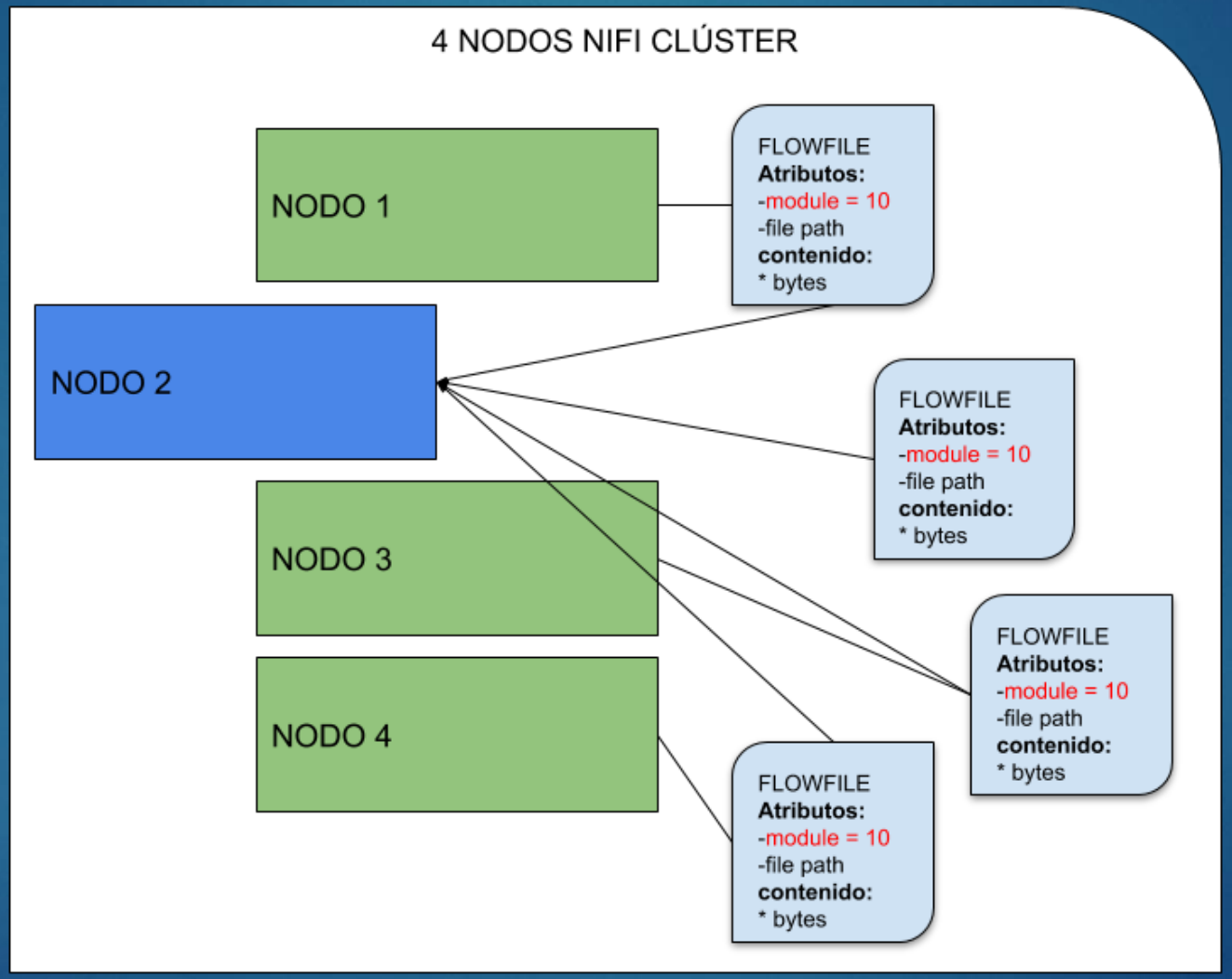
IMEI TRACKING DE EVENTOS DE RED

- Se diseño un balanceo en el flujo para que todos los eventos de un determinado IMSI vayan al mismo nodo dentro del clúster NIFI.
- Se aplica balanceo en el origen de datos usando procesador listSFTP y distribución con algoritmo ROUND_ROBIN a los diferentes nodos del clúster.
- Se calcula el modulo del campo IMSI por cada evento.
- Se particiona los flowfiles usando campo modulo (procesador partition record).
- Se aplica balanceo a través del clúster usando atributo “modulo”.

Balanceo con listSftp y Round Robin



Balanceo por atributo del flowfile



Logros caso de uso 2

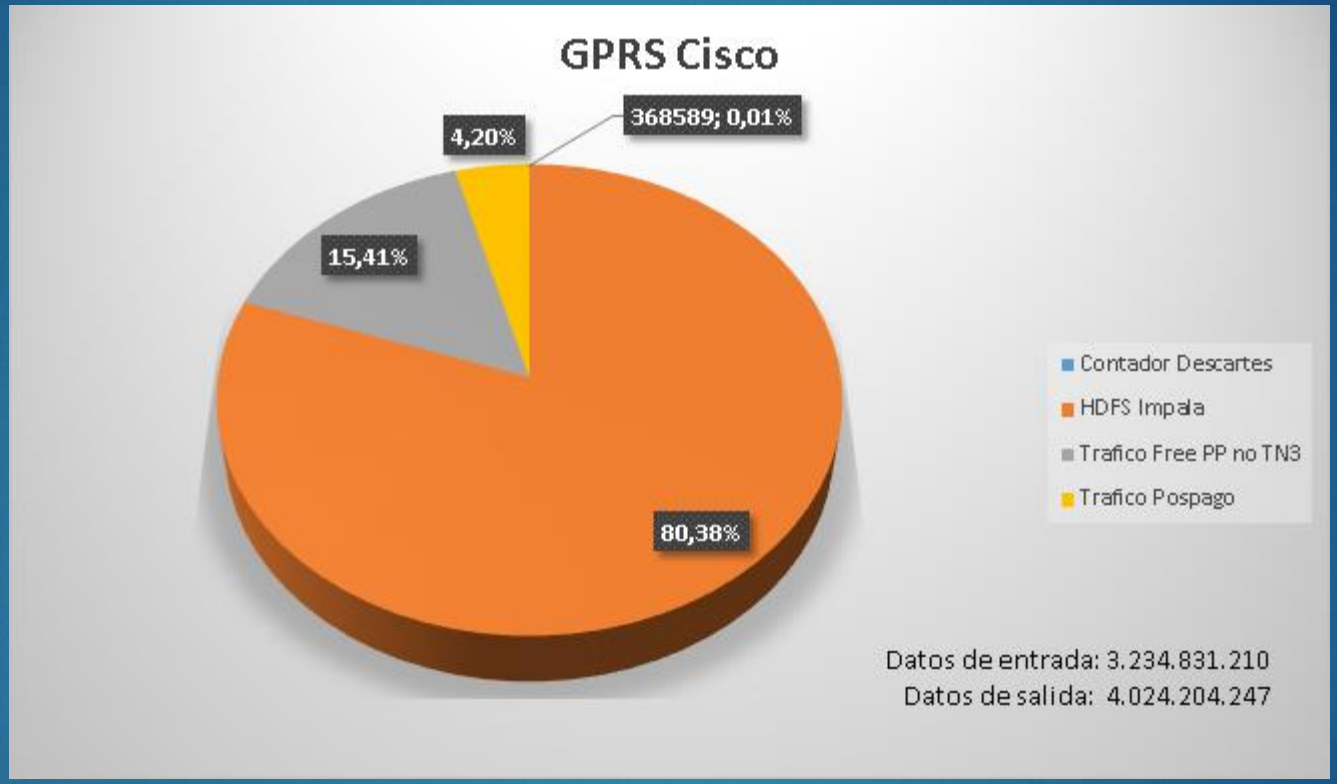
- Reducción de la cantidad de request a la base y el uso de red.
- 150.000 request por segundo.
- Con el algoritmo de distribución de IMSI evitamos tener un cache Redis replicado por la cantidad total de suscriptores, lo que reduce la cantidad de recursos de hardware para cachear los 16 millones de clientes activos en la red y permite aprovechar la memoria disponible en los nodos donde corre NIFI.
- 8000 millones de eventos al día.

CASO DE USO 3
NIFI ASN1

Decoder ASN1

- Procesador custom Nifi para decodificar ASN1.
- 4.000 millones de registro al día.
- Distribución a Kafka y hdfs.

PROCESAMIENTO ASN1 PGW CISCO



Open Source contribuciones

- https://issues.apache.org/jira/browse/NIFI-6800
  - Error detectado en el core del framework en relación a manejo de contadores en sesiones de alta performance. Se acepto el pull request y se realizo merge a la rama master.
  - Git commit: https://github.com/apache/nifi/commit/67f167721236829022 8b373fdbeb73f0265e2211
  - Junit test de la contribución al código: https://github.com/apache/nifi/commit/ce5eae5b2cd5cf49a90d2980c58cd6647f4af292

- https://issues.apache.org/jira/browse/NIFI-6395
  - Pull request: https://github.com/apache/nifi/pull/3552

Lo que viene…

Flink y Machine Learning

Python

- Uso de Python sobre la API Table a partir de Flink 1.9 (FLIP-38).
  A partir de 1.9 el soporte de Python (PyFlink) está basado en Py4j.
  Similar a PySpark.

- Investigación y pruebas de ejecutar Python por medio del el uso
  de Flink PortableRunner del proyecto Apache Beam (Portability Framework).

FlinkML

- Seguir la evolución de algoritmos java Flink ML sobre la API Table (FLIP-39).

Link útiles

- https://github.com/cloudera/cm_ext/wiki
- https://github.com/apache/nifi
- https://github.com/apache/flink
- https://nifi.apache.org/docs/nifi-docs/html/developer-guide.html
- https://nifi.apache.org/docs/nifi-docs/
- https://cwiki.apache.org/confluence/display/NIFI/For+Developers

FIN