Case Study - Hiro-Media

(http://www.hiro-media.com/)

Field of occupation: Advertising

HIRO

Company concept:

HIRO is a technology company that offers innovative video based Marketing solutions and products for publishers and advertisers.

HIRO places a special focus on ensuring safety for publishers, transparency & control for advertisers and Optimized monetization –

For both Demand and Supply parties.

The situation before the project:

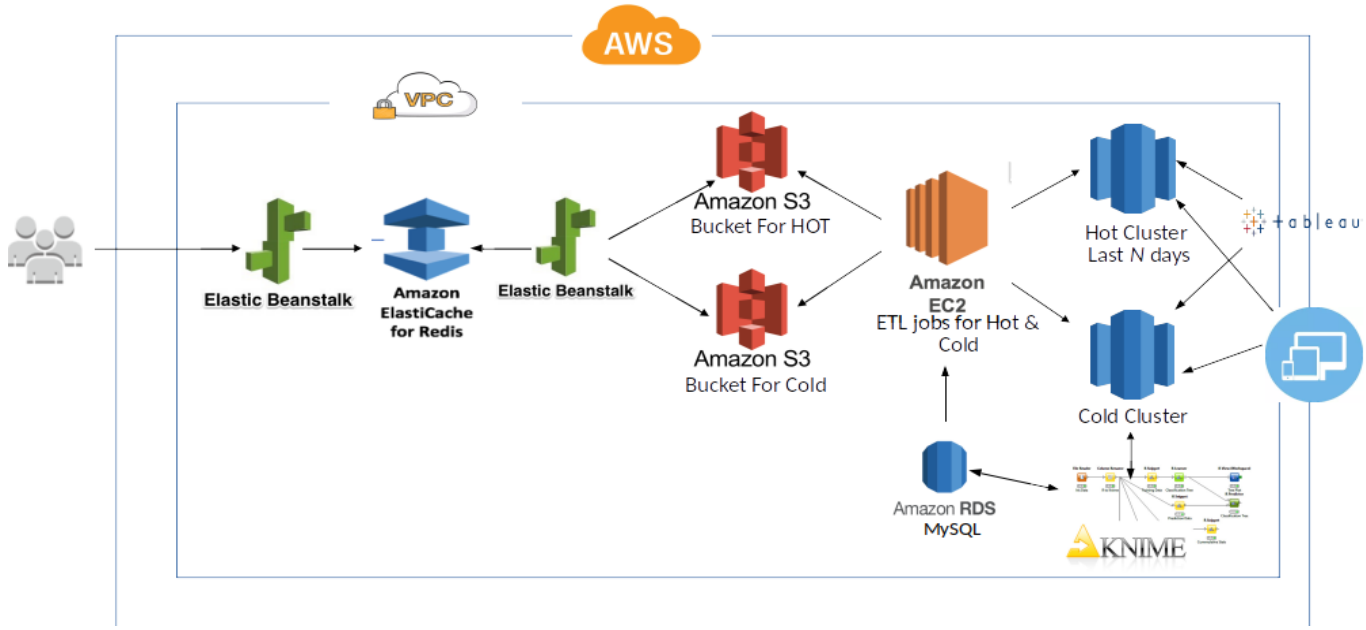
From before 2014 HIRO's collected data continued to grow rich in its level of details and in quantity and saved in the a self managed MYSQL in AWS, the need for a stronger database and analytic tools was clear and a requirement was added to search for a big-data database.

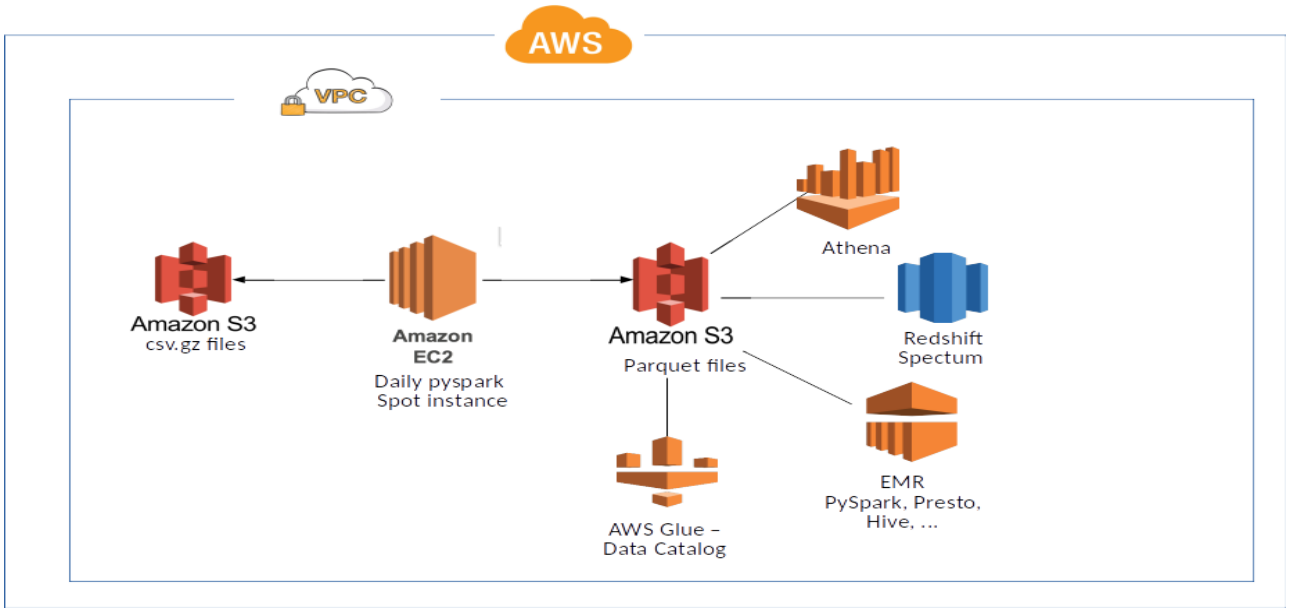
Major challenges:

1. The original architecture was not build for rapid growth that occur in the last 10 years at the Ad industry. Originally events arrives from the player to Tomcat/Jboss app-servers (with load-balancer) and stored at self-managed MySQL DB (before 2014)
2. Request for near real time reports that are based on 350-400GB of raw data per quarter.
   Data can be grow or shrink based on the % of events we save. The percentage can be change in real-time and depends on the dynamic nature of the industry.
   The main data come from 3 types of events
   a. network events
   b. clips viewing events
   c. publishers events
   Hiro users needs the data to be refreshed and update near real-time as well as recalculate and add new calculation.
   The data is aggregate to 2 main objects : campaign counters & publisher counters.
3. React to fast changes in schema with minimal deployment downtime of production environment (needs to run 24*7*365).
4. Reduce the price of system components and adopt new services and make sure that time to market is not effected.

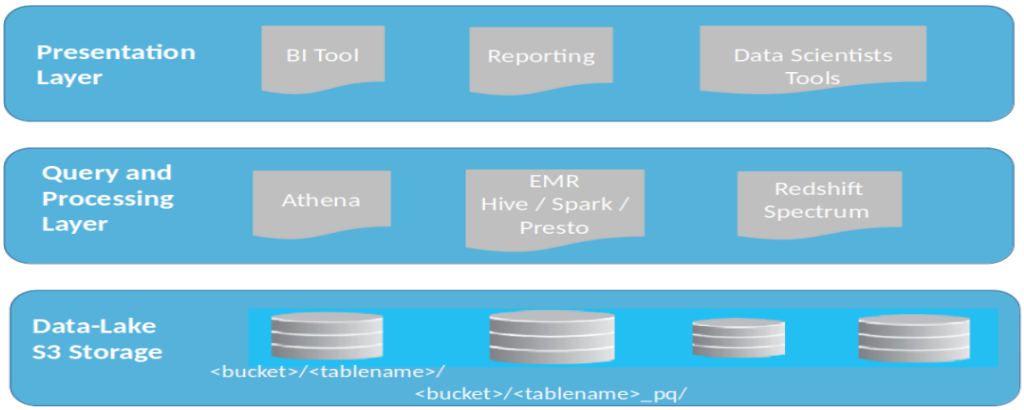
Solutions:

HIRO selected to implement Amazon Redshift DB and BI tool-Tableau in 2014. In 2015 a decision that the data-lake will be in S3 (still data saved as csv.gz files). To reduce the cost of using the s3 data-lake in 2016 we started to convert the csv.gz to Parquet via EMR. The data scientist and heavy monthly/quarterly/yearly management reports moved to run on EMR cluster (which grow and shrink on request) with different tools (R-Studio, Knime, Zeppelin, SparkSQL, Hive, etc) .





The Latest data-lake solution



1. New architecture consists of:
    - Split the data to 4 types of databases/storage:
        - RDS – To hold transnational data (basic OLTP).
        - Redshift – raw events data and aggregation.
        - S3 – All raw data include data we discard from the Redshift clusters.
        - Redis (Elasticache) – use Redis as queue storage and application metadata caching.
    - Move handling of incoming events from Tomcat/Jboss and MySQL RDS to AWS Beanstalk (PHP). The Tomcat/Jboss application handle the OLTP application side.
        - Events arrive to Beanstalk validate and written to a queue table in Redis. Every N minute a job is start, consume all messages and write them to bucket in S3.
        - Implement distributed caching in Redis to reference data from RDS that is frequently used.
    - In production two Redshift cluster
        - Hot Cluster: Hold raw and aggregation data for the last N days (we started with N=2 and currently N=7 after reduce undeeded long string columns) for near realtime users and reports one the last N dates.
        - Cold Cluster: Hold raw data for 6-12 month and aggregation historical data for 1-2 years.
    - Moving data in the system
        - ETL (python) for Redshift
            - Extract updated metadata from RDS
            - Consuming data into Redshift from RDS and S3 and move it to all clusters. Very large strings are limited in length. We save longer stings for N days (for example 8000 chars) and after it we reduce for longer storage (we have types of tables: table T_newevents where the columns can reach up to varchar(8000) and T_final where the max length of varchar that we allowed is 2000. A new event enter to T_newevents table and after N days (parameter) will move to T_final (and will be trimmed to 2000).
              The full sting (have no limits) is saved in the Data-Lake (s3) and converted to parquet format.
        - ETL (python) for that can
            - Create the Redis caching layer from scratch
            - Update the Redis caching with changed metadata from Redshift
            - Move calculated aggregated data from Redhsift back to RDS.
        - Convert raw events data from csv.gz to parquet and build S3 as the data-lake via
            - PySpark job that convert the raw data to parquet (EMR or standalone EC2 machine depends on the volume)

2. The ETL sends data to both of the cluster, if one cluster is unavailable then all users are routed to the available cluster so they can see that incoming data.
   In this way when running an upgrade, maintenance or any other reason where the original cluster is unavailable you always have an available cluster.
   Hiro must have the latest data available to all users. Historical (cold) data is not as important in day-to-day operation as the latest data.
3. Build infrastructure for fast changes in table schema and table “partitions”.
    - Build scripts for adding, remove, alter columns in existing tables include view changes.
    - Auto build new table partitions (big tables are partitioned to quarters).
4. Integrate Redshift and RDS with Knime (data analytic open-source) to run some propriety machine learning algorithms (like the bid price for campaign). The output is pushed to the BI tool (pop up in Tableau).
5. Hiro decide to use Tableau as their BI tool, we needed to make sure that the BI tool generate good queries because the required response time was 3-5 second (avg) end-to-end in some of the reports.
6. In Hiro some of the technologies are early adopted and needed to be re-evaluate and check if the early implementation is still cost-effective. We need to provide the ability to change the infrastructure. For file format conversion we started with EMR and hive thought pyspark and ended with the ability to convert on a spot stanalone EC2 (pyspark) on-demand.
7. Users can use the data directly from S3 using (metadata definition is managed in AWS Glue)
    - Notebook / RStudio (standalone or under EMR)
    - AWS Athena

- Redshift Spectrum