LEVERAGING CLOUD-BASED PREDICTIVE ANALYTICS TO STRENGTHEN AUDIENCE ENGAGEMENT

U. Shakeel and M. Limcaco

Amazon Web Services, USA

ABSTRACT

To grow their business and increase their audience, content distributors must understand the viewing habits and interests of content consumers. This typically requires solving tough computational problems, such as rapidly processing vast amounts of raw data from websites, social media, devices, catalogs, and back-channel sources. Fortunately, today's content distributors can take advantage of the scalability, cost effectiveness, and pay-as-you go model of the cloud to address these challenges.

In this paper, we show content distributors how to use cloud technologies to build predictive analytic solutions. We examine architectural patterns for optimizing media delivery, and we discuss how to assess the overall consumer experience based on representative data sources. Finally, we present concrete implementations of cloud-based machine learning services and show how to use the services to profile audience demand, to cue content recommendations, and to prioritize the delivery of related media.

INTRODUCTION

An abundance of technical advancements has expanded the range of options for media consumers. Today's consumers can choose to have 3-D, 4K, HDR, and even 8K content displayed on a variety of sophisticated devices. Given the public's appetite for these high-end devices, media creators are constantly under pressure to increase resolution and quality to compete in an ever-expanding war of content choices.

In addition to the changes in display technologies, on-demand content and streaming media delivery have changed the habits of content viewers. Gone are the days when we used to circle around the TV set at an appointed time for the airing of our favorite TV show. People now expect to watch the programming they want on their own schedule, which is driving more and more media companies to consider providing their own OTT (over the top) service. These services use the Internet for delivery, which introduces potential quality issues that are beyond the control of the media owner or distributer. To mitigate these risks, many media distributors invest heavily in solutions that detect playback issues; these solutions require large amounts of computational capacity to process massive, raw datasets to provide real-time course correction. In this way, distributors can provide more reliable content that caters to the viewing habits of their audience.

This raises the question of how to build the next generation media delivery platform that not only delivers reliable content, but also ensures that the content is experienced the way the content creator intended. One approach is to have the delivery platform predict when events such as network congestion or low-quality streams will occur in the future, and subsequently guide consumers in the right direction. Powerful tools toward this goal include using data generated by consumers from their interaction with content, social media, and multiple screens in conjunction with predictive modelling, machine learning, and real-time analytics. According to Nielsen, social media activity drives higher broadcast TV ratings for 48% of shows (1); in a similar survey by Netflix, over 75% of what people watch is based on Netflix's recommendations (2).

In terms of audience engagement, there are two basic categories:

- *Content experience* – Using predictions and analytics on viewers' viewing habits, player network logs, and datasets to quickly analyze existing issues or predict future issues. The predictions can be used to minimize or even eliminate a poor customer experience.
- *Content relevance* – Using predictions and analytics on historical and some real-time datasets to detect and recommend relevant content and personalize content and ads, thereby improving the experience for content selection.

Audience Engagement Signals

To build a next generation media delivery platform and deliver a better customer experience, content distributors can capture, fuse, and synthesize background signals (noise) to create models to analyze, for both batch and real-time data. We can leverage both transactional data (from user interactions such as searching, playing, watching/listening, and contacting sales/support) to behavioral activities (such as sharing, tagging, liking, reviewing the content, and so on) to build a prediction model. The analysis can be descriptive (aggregation, retrospective), predictive (statistical, machine learning) or prescriptive (what should we do about it?) based on technology choices. In the remainder of this paper, we focus on descriptive and predictive analytics, especially in the context of content relevance.

Machine Learning for Predictive Analytics

Machine learning (ML) is a broad area of tools and techniques that can help us use historical data to make better business decisions. ML algorithms help us discover patterns in data and construct predictive models using these patterns, allowing us to use the models to make predictions from future data. For example, we could use ML to predict whether customers will select a title to view based on data such as their viewing history, what other users in their same demographic have watched, and even who they follow on social media platforms. We then can use the predictions to identify which customers are most likely to respond to personalized, promotional marketing campaigns.

Benefits of the Cloud for Predictive Analytics

Cloud Computing platforms are optimal for batch and distributed processing and can be used to analyze consumers' interactions. Cloud platforms offer high-volume data processing at scale and generally at a fraction of the cost compared to traditional data analytics infrastructure solutions. It is important to understand the scale of such a dataset. For example, as of 2011 Netflix has been managing 20 billion requests per month for millions of consumers across more than 60 geographies (3). Netflix leverages tens of thousands of Amazon Web Services (AWS) EC2 virtual instances on demand and terminating the instances when the work is complete.

Analyzing large data sets requires significant compute capacity that can vary in size based on the amount of input data and the analysis required. This characteristic of big data workloads is ideally suited to the pay-as-you-go, cloud-computing model, where applications can easily scale up and down based on demand. This elasticity means that as requirements change, we can easily resize our environment (horizontally or vertically) on the cloud to meet our needs without having to wait for additional hardware or over-invest to provision for peak capacity. This scalability is especially important for mission-critical applications. In contrast, system designers for traditional infrastructures have no choice but to over-provision because systems must be able to handle surges in data volumes due to increases in business demand. ML scenarios in particular benefit from scalable and elastic infrastructure since many ML techniques typically require abundant compute capacity with the ability for data analysts to iterate and experiment in adaptable ways.

Specifically, as an example of available public cloud computing infrastructure, Amazon Web Services (AWS) provides media customers with on-demand access to technology services available across 11 different geographic regions around the world. The overall scalability, elasticity and global accessibility of platforms such as this make it an extremely good fit for solving big data problems. Additional details on relevant case studies in this context can be found at (3).

TECHNOLOGY RECAP

The representative technology spectrum in this space may be categorized into three main areas:

1) Desktop-driven data science tooling (including Microsoft Excel, KNIME, IBM SPSS Statistics, RapidMiner, R language and environment, Weka, MATLAB, and Octave). These services are typically employed by analysts and specialists to perform detailed modelling, simulation and visualization.
2) Server-side big data platforms (several products within the Hadoop platform such as Apache Mahout, Spark MLlib, Oxdata, GraphLab, R+ Hadoop, Radoop, Apache Hama, Apache Giraph, Apache HBase Kiji, and BigML). These platform services often serve as the production-oriented backend to desktop-driven analysis. This is especially true in cases where the volume of data to be scored or assessed exceeds the capacity of a single client compute node.
3) Scoring/prediction deployment services (Zementis ADAPA, and Orynx). Using open standards to articulate prediction models (e.g., PMML) can be useful in separating out modelling and training from execution and assessment. Scoring and prediction engines can be used to implement the modelling logic expressed as output [PMML] from upstream machine learning services provided by the previous categories of technology.

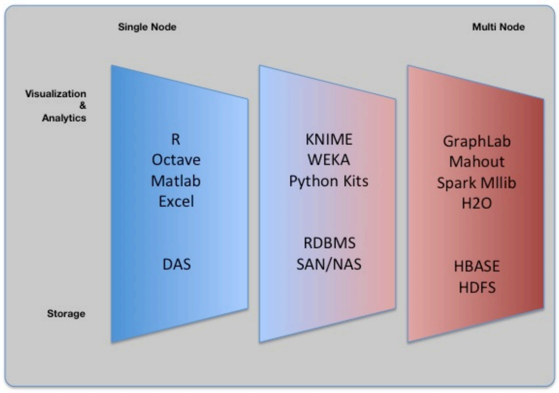


Figure 1 – Relevant analytics technologies

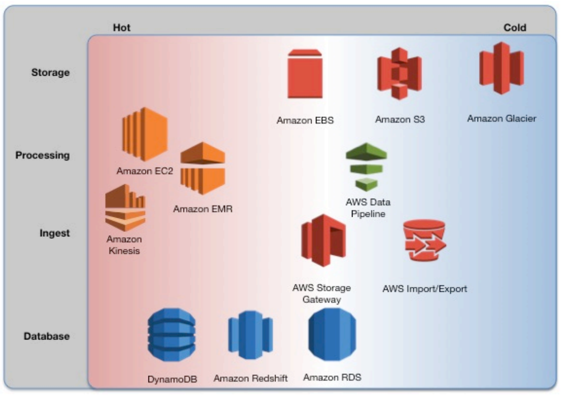


Figure 2 – Relevant Cloud services (AWS) Reference (4,5,6,7,8,9)

Many of these options (Prediction/IO, Mortar, BigML) are also available as on-demand or, alternatively, as virtual products accessed via cloud marketplace registries.

DEEP DIVE: USE CASE SOLUTIONS

In this section, we focus on the use case for content relevance, and we examine the cloud features and their value proposition in the light of architectural design patterns.

Content Relevance

Let's start by looking at how we can generate predictions based on an analysis of historical and real-time datasets to detect and recommend relevant content as well as to personalize content.

Data sources

One of the key questions is "What data is meaningful for determining content relevance?" The answer typically is data about user likes/dislikes, user interactions on social media channels, and users' viewing habits correlated to user profiling (because they like *x*, they may like *y*, and so on). However, this can result in a very large dataset, both historical as well as real-time, when we gather data across all consumers of our content. Certain native cloud services, such as Amazon Kinesis and the highly scalable fleet of virtual machines in the cloud can provide us the means to ingest very large quantities of data and scale on demand in a highly available and durable fashion.

Additionally, Kinesis provides scalable connectivity to its cloud storage services like the object storage and archive, which we can leverage for historical data. We can also use these services to transition real-time data to archives to make the data available for historical analysis.

Social media

Today's gadget-savvy customers are connected to their peers, family, and friends seemingly all the time, even while they are consuming content. Social media sentiment is an important source of audience interest. Social media channels like Facebook, Twitter, Instagram, Pinterest, Foursquare, Tumblr, Google Plus, IMDB, and Flicker are common examples. Most of these social media channels provide API access (RESTful in some cases) that we can use to get data about near real-time user behavior. We can then filter the resulting dataset to gather signatures for a specific piece of content.

Media player logs

Most video players provide the capability to capture and stream a real-time dataset to a backend ingestion point. The dataset can contain anything from seek, play, pause, and other player controls to even the specific bitrate delivered to the customers (in the case of adaptive bitrate). Most off-the-shelf media players provide this capability; however, many distributors also build their custom players to include additional data collection and reporting capabilities for interactive experiences like search, on-the-fly recommendations, camera angle selection, or even shopping experiences, effectively creating an interactive knowledge dump about the content. Data describing how users interface with these controls and engagement points can be streamed back to the backend analytics engines and environments.

Viewing history

Viewing history can be captured by the front-end application or the medium that the viewer uses to navigate and select the content. This data is secured, logged, cleansed, aggregated and archived. We can then leverage this historical event trail and build recommendations based on user interactions resulting into both positive (views, downloads) or negative preferences (unsubscribe, delete).

Approaches

Once the data has been captured (both historically or streamed in real-time), the challenge is to make sense of the raw data (from the data sources discussed above). Depending on the turn-around time required for the underlying application, cloud scalability can be of great advantage to spin up clusters of hundreds of compute nodes running the analytics engine of our choice in a matter of minutes.

Sentiment analysis

From a technical standpoint, the concept behind sentiment analysis is to process largely unstructured natural language sources and to extract subjective meaning behind the words being expressed. It is often used as a means of automatically gauging user interest and community trending on topics related to events, products, and personalities in the social media space. We can then apply this understanding in near real-time towards use cases ranging from content recommendations to digital media ad campaign efficacy.

Analysis in this context involves the application of text processing and machine learning techniques to classify audience commentary by using a generalized understanding of what is deemed positive vs. negative sentiment. Natural language processing itself is a challenging subject with numerous studies dedicated to obtaining high precision in the extraction and understanding of meaning behind the spoken and written word. This is especially complicated in social media space with format-limiting colloquial expressions (tweets). The other challenge comes from the large volume of this content, often streaming from a variety of outlets: tweets, posts, blogs, and movie reviews.

One general technique to deal with this is the application of conditional probabilities as a "bag of words" concept. That is, a system is initially trained with examples of both positive and negative sentiment, and these text examples are tokenized and treated as simple collections of words. The system uses this baseline collection of words and word frequencies to determine whether new word collections are likely examples of either category. We can use numerous programming kits and libraries that implement the "Naïve Bayes" classifier to this effect, such as the Python-based Natural Language Toolkit (11); however, the goal of performing this operation in near real-time at high volumes remains a challenge. The proposed approach to this is to perform the learning process offline with a smaller subset of the overall corpus, and then to apply the trained model inline in a high-volume stream ingest processing pipeline.

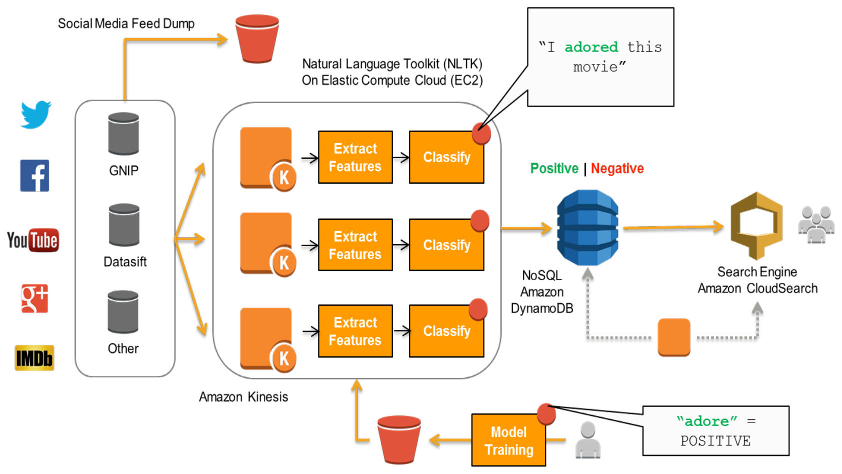


Figure 3: Sentiment analysis on real-time sources

Training the system with different types of sentiment is performed out-of-band from a few thousand to ideally several hundred thousand samples. Several predefined training sets are generally available from generic and academic repositories; however, to train using very specific industries it is desirable to use more specific sources. We may, for example, elect to *crowdsource* the process of inspecting a phrase and classifying the intent. One example technology service enabling this is Amazon Mechanical Turk (14). Herein we find a cloud-based human workflow system that allows customers to specify tasks up for bid, which then can be executed by a vast global workforce connected through this service. Customers can programmatically allocate specific classification tasks with required parameters, process the community feedback, and automate the creation of the training model data structure using this information. Once the initial model is created (assessed and fine-tuned), it can then be deployed into a real-time flow. Data sources such as Twitter, Facebook, and managed data brokers such as Datasift provide interfaces for ingesting large volumes of social media commentary. Data may be ingested and filtered for specific attributes, phrases, and sources. The key is to receive and process the data and apply the trained model on the inbound stream at high velocity. Herein, cloud computing platforms provide very effective options for collecting and processing large streams of data records in real-time.

In this logical architecture, we receive data through established APIs from social media sources. We then apply the pre-trained Naïve Bayes Classifier model against the continuously updated stream of sentiment. The training logic is embedded in simple call-back handlers registered with the Amazon Kinesis streaming service SDKs such as those found in Apache Kafka, Flume, Storm, Spark Streaming or the Amazon Kinesis Client Library (KCL) framework. Classifications (positive or negative) are updated in near real-time against a high-throughput data store such as a NoSQL system. This data can then be used in several ways: for example, analysts can make timely queries to inform digital marketing strategists on the efficacy of media efforts to date. Alternatively media delivery platforms can leverage this data to filter or affect content/video recommendations as part of an OTT experience. (We discuss recommendations in greater detail in a later section.)

Segment audience

It is often desirable to automatically categorize subgroups contained within a larger audience population to more readily engage these members with personalized content and targeted initiatives (i.e., digital marketing campaigns). Machine learning provides several approaches. We can, for example, use a form of unsupervised learning known as "K-means clustering" to automatically and recursively identify audience member affinities across an n-dimensional space. K-means clustering seeks to categorize entities based on a defined similarity measure; the "k" represents the seeded number of target groups that the system then iterates on and converges until k-number of segments are discovered. However, when the size of the overall population and the breadth of user attributes (geo, demographics, viewing behavior, and so on) is large, this can exceed the capabilities of a desktop toolset such as R or Weka. Here, we can burst into the cloud and extend the reach of the client toolset by delegating the processing to a scalable backend machine learning cluster readily capable of processing millions of records. One such example of this configuration is in the combined R + 0xdata H20 platform. 0xdata provides distributed scale-out k-means clustering on virtual compute resources and can seamlessly present results back to an analyst using the R desktop client for visualization and final disposition.

Applications

Based on these approaches, let's analyze how we can use this normalized data across audience segments to deliver some real-world applications in the content delivery space.

Recommendations and interactive experiences

Content recommendations are an important part of overall audience experience. Building this as a part of any media platform provides a great opportunity to optimize the overall user experience and to strengthen overall engagement. In addition to leveraging previously learned audience segment characteristics, a good recommendation system leverages other audience signals on an on-going basis to optimize and personalize the user experience. Such signals include explicit user interactions (the user purchases a movie) as well as implicit user interactions (the user searches for a genre). We can collect and analyze these signals on an on-going basis by generating history and observation matrices, and then we can find similar items and users based on these observations to recommend new items of interest. Of course, the growing volume of subscribers and content in current and emergent OTT/broadcast scenarios drives us to consider techniques designed to scale to the large quantity of engagement signals.

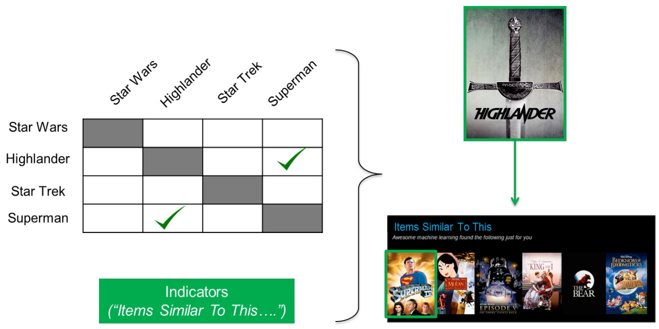


Figure 4: Finding interesting pairs of items ("co-occurrences")

One technique uses a distributed cluster of compute nodes to process high-volume log file inputs (sourced from players and web service infrastructure) to create "other people also liked these things" types of recommendations (12). In this example, we use the open source, distributed machine learning project Apache Mahout to transform historical observations (user-A watched video-stream-A) to create a co-occurrence matrix that describes what items have been observed with others (video-stream-A co-occurred with video-stream-B in user-B's experience). We further refine this matrix to surface the most interesting co-occurrences using the log-likelihood ratio LLR measure. This effectively gives us the baseline data structure to calculate item similarity results – for example, the "items similar to this" in a typical online media or e-Commerce experience (Figure-4). Very specifically, we can utilize Apache Mahout's *spark-itemsimilarity* function to process source log data stored in a web store like Amazon S3. Using a backend Hadoop cluster to host the infrastructure, we can process large amounts of historical logs, cleanse and normalize the data, and then pass it into the machine learning pipeline. The Mahout process converts the log data and presents output results in the form of *<item> <appeared with item-1> <item-2> <item-3>* and so on.

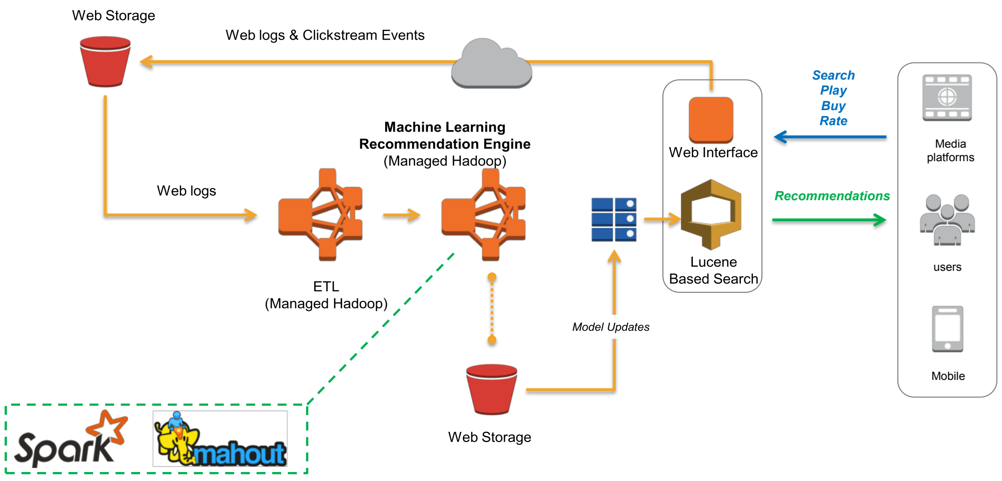
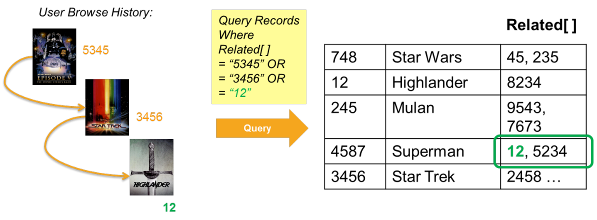


Figure 5 - Integrated recommendation architecture

In processing source data to compute recommendations, it's also important to recognize the distinction between explicit user interactions vs. implicit interactions. In the former case, users may make explicit choices regarding content they wish to engage (e.g., positively rate a video); in the latter, users may express less formal preferences through browsing, searching, scanning or other. The special role of implicit user feedback in calculating content recommendations, and specifically in the case of TV broadcasts, is described in (15). The required ML techniques to address these cases is articulated in the Apache Spark MLlib recommendation component based on Alternating Least Squares Matrix Factorization (16). This capability attempts to estimate a composite product rating matrix (users x content) by iteratively solving for the two "lower-rank" matrices of users and products. Given the potential number of captured audience ratings for an online media service, where millions of ratings on tens of thousands of titles may be the norm, this may prove to be computationally expensive. Platforms such as Spark MLlib however are designed to operate in a parallel manner when deployed to distributed compute platforms such as those readily provisioned on the cloud.


User Browse History:
5345
3456
12
Query Records Where Related[ ] = "5345" OR = "3456" OR = "12"
Query

		Related[ ]
748	Star Wars	45, 235
12	Highlander	8234
245	Mulan	9543, 7673
4587	Superman	12, 5234
3456	Star Trek	2458 …

Figure 6 - Using user online clicks to search an item similarity matrix

When the resulting matrix predictions are coupled with a search engine, long with a record of realtime user interactions, we also can use this data to deliver personalized results (Figure-5). That is, by caching recent online user activity (clicks, downloads, wish lists, plays) we can generate a growing list of implicit search criteria that can be supplied to a backend search engine capable of finding similar items of interest. In this case, the backend search index is supplied with data from our item similarity data structure from the offline process that we described earlier. We can load this data structure into an online search engine such as a high-scale Lucene-based platform. By applying concepts such as TF-IDF (term frequency inverse document frequency), we can execute continuous searches against our index. This search provides a form of distance measure between clickstream history vs. a precomputed base of related content. This answers the following question: "*Given the input set of movies I'm interested in, what other movies have been observed in interesting quantities that match?*" Served out of a search engine, the result is a dynamic user experience that more or less reflects recent and on-going user interaction by providing lists of targeted content and then, from an end user perspective, adapts content recommendations accordingly.

Personalized ads

We also can use the preceding approaches of user sentiment analysis and segmentation to deliver personalized ads. For example, a player requests an ad based on the user profile signature, and the ad-server returns a personalized ad in the context of a play-back experience to a live scenario, where the ad-markers can be replaced with a personalized ad on the fly.

Media monetization is often ad-driven, and a personalized, targeted ad has better chances of eventual conversion. Although the content player can enforce the serving of ads before the actual content starts playing or in between during the ad-breaks/ad-markers in the stream, it does not always mean the audience is engaged with the content of the ads. Advertising companies spend a lot of money coming up with creative, catchy ways of attracting their audiences, but more than that the ad content has to appeal to, and be consumed by, the right audience. We can use content recommendations across the player logs to mine ad content behavior and create signatures based on different demographics. These and other ad tech-related processing demand requires expansive compute and storage resources. Video ad platforms such as Brightroll achieve this kind of scale by using cloud services such as Amazon EMR and Amazon S3 to manage over 25 billion video ad inventory requests per month (13). Figure 7 shows a reference architecture for dynamic ad-serving.

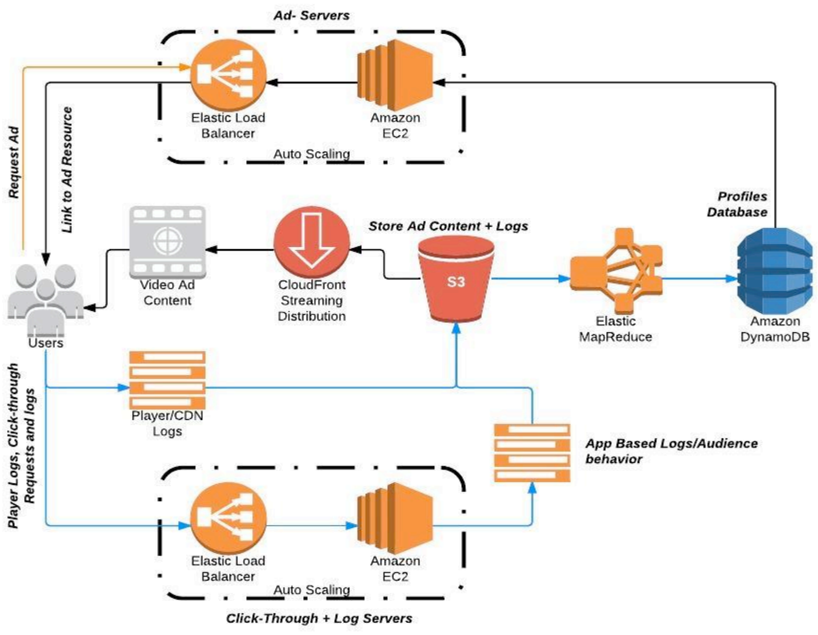


Figure 7 – Reference architecture for ad-serving on AWS

Content Experience

When we use predictions and analytics on datasets from users' viewing habits and player or network logs, we quickly can analyze existing issues or predict future issues. We then can take actions to minimize or even eliminate a poor customer experience. Distributors like Netflix use CDN logs, media player logs, and bounce-rate data to build a model of churn detection and network performance for a particular geography and audience segment. The analysis can be done in real-time, which leads to real-time actions around CDN switching to provide a better customer experience (for example, quality in the case of ABR, or less buffering). Additionally, based on the user viewing habits or recommendations across a segment of users, the content can be pushed to the appropriate CDN pops to provide a faster streaming experience. In the race to engage customers with our content, we can win or lose a customer based on the time it takes a player to start streaming our content after a user selects it. If it takes too long, we risk losing the user to a competitive channel.

CONCLUSION

Machine learning (ML) tools and techniques can help us strengthen our audience engagement. The ever-expanding volume, variety, and veracity of audience signal data, however, forces us to re-evaluate and expand our current approaches. Readily available compute and storage resources in the cloud allow these ML technologies to perform with unprecedented scale and, in many cases, improved accuracy. Tooling such as Apache Mahout, SparkMLlib, and Oxdata H20 provide a broad spectrum of machine learning techniques (such as k-means clustering, collaborative filtering-based recommendations, and logistic regression), but gain true scale and throughput improvements when deployed on cloud platforms like AWS. Moreover, cloud-native fully managed services such as Amazon EMR and Amazon Machine Learning provide even greater ease and convenience for data science and digital marketing specialists, freeing them from the drudgery of managing infrastructure. Instead, they can focus on discovering new audience insights and delivering solutions that strengthen overall engagement.

REFERENCES

1. http://venturebeat.com/2013/08/06/nielsen-tweets-drive-higher-broadcast-tv-ratings-for-48-of-shows/
2. http://www.pcmag.com/article2/0,2817,2402739,00.asp
3. http://www.slideshare.net/AmazonWebServices/maximizing-audience-engagement-in-media-delivery-med303-aws-reinvent-2013-28622676
4. http://aws.amazon.com/kinesis/
5. http://aws.amazon.com/s3/
6. http://aws.amazon.com/dynamodb/
7. http://aws.amazon.com/redshift/
8. http://aws.amazon.com/emr/
9. http://aws.amazon.com/ec2/purchasing-options/spot-instances/
10. http://aws.amazon.com/glacier/
11. http://www.nltk.org/_modules/nltk/classify/naivebayes.html
12. https://mahout.apache.org/users/algorithms/intro-cooccurrence-spark.html
13. http://aws.amazon.com/solutions/case-studies/brightroll/
14. http://www.mturk.com
15. "Collaborative Filtering for Implicit Feedback Datasets", available at http://dx.doi.org/10.1109/ICDM.2008.22
16. https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html