Building a Real-Time Bidding Platform on AWS

February 2016

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use, global cloud computing platform. The AWS cloud delivers a comprehensive portfolio of secure and scalable cloud computing services in a self-service, pay-as-you-go model, with zero capital expense needed to manage your real-time bidding platform. This whitepaper helps architects, engineers, advertisers, and developers understand real-time bidding (RTB) and the services available in AWS that can be used for RTB. This paper will showcase the RTB platform reference architecture used by customers today, as well as provide additional resources to get started with building an RTB platform on AWS.

Introduction

Online advertising is a growing industry, and its share of total advertising spending is also increasing every year and projected to surpass TV advertising spend in 2016. A significant area of growth is real-time bidding (RTB), which is the auction-based approach for transacting digital display ads in real time, at the most granular impression level. RTB was the dominant transaction method in 2015, accounting for 74.0 percent of programmatically purchased advertising, or 11 billion dollars in the US.[1] RTB transactions are projected to grow over 30 percent in 2016, according to industry research.[2] Real-time bidding is also gaining popularity in mobile display advertising as mobile advertising spend is anticipated to grow in excess of 60 percent in 2016.[3]

As the amount of data being created and collected grows, organizations need to use it to make better decisions in determining the value of each ad impression. AWS has an ecosystem of solutions specifically designed to handle the real-time low latency analytics that allow you to make the best possible and most efficient ad impressions to drive your business.

Real-Time Bidding Explained

When you go to a website and are served an advertisement, the process to serve you that advertisement involves the website or publisher contacting an ad-exchange, which then accepts real-time bids from many different parties. The bidders use the information about the user that they know (for example, the

website, and the ad location/size, plus demographic information, such as user location, browsing history, and time of day) to determine how much they are willing to pay to deliver an advertisement to the user. The data may come directly from publishers (mobile applications or websites) or third-party data providers. Whichever bidder bids the most within a time period set by the exchange, usually under 100 milliseconds, gets to serve the ad and pay the bid price. This process at a high-level is depicted in Figure 1. RTB is the process of accepting data from Step 2 and doing the action in Step 3.

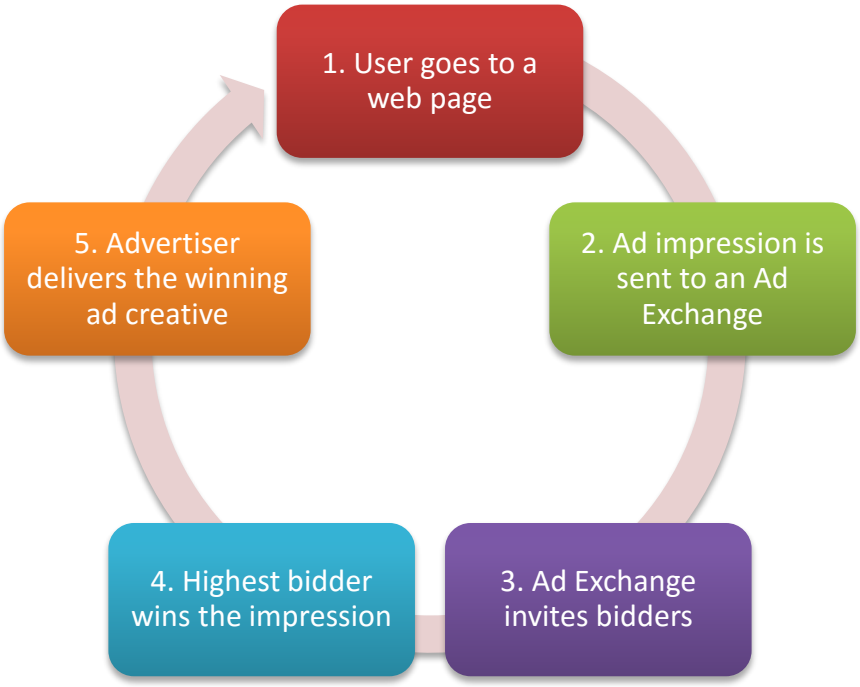

Figure 1: Real Time Bidding Process

Elastic Nature of Advertising and Ad Tech

Web traffic is the engine that drives the advertising industry. Daily web traffic volume can vary by 200 percent or more (based on time of day). In Figure 2, you can see a typical pattern of load on an RTB platform in a single day. With elasticity, you can achieve greater infrastructure savings by turning off resources as traffic decreases.

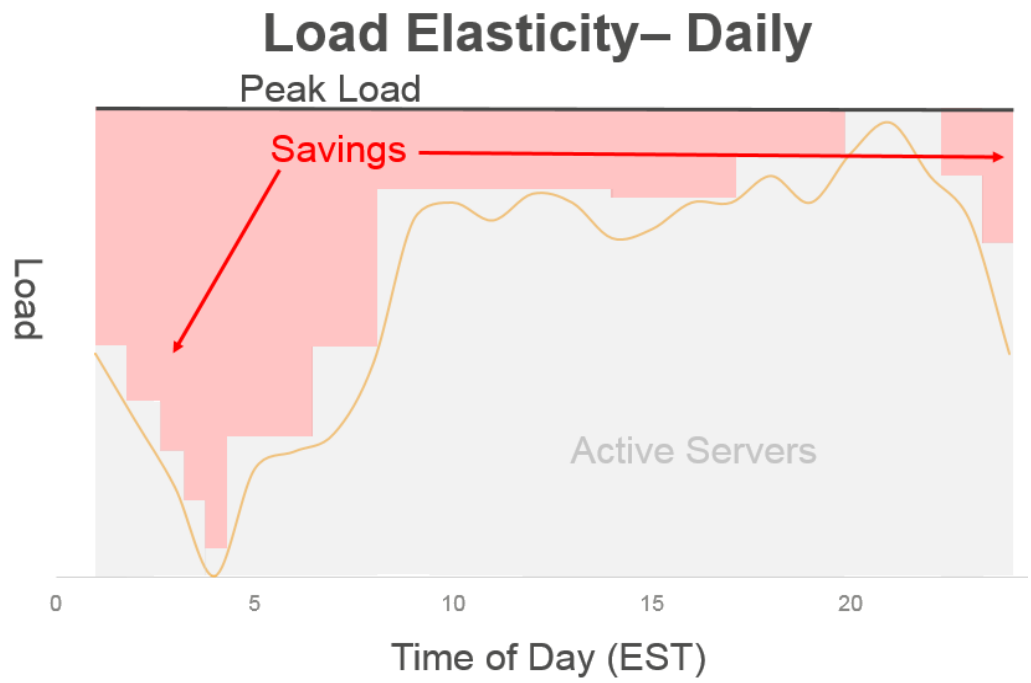

Figure 2: Daily Load Pattern for RTB

In addition, Figure 3 below illustrates the typical pattern that an RTB platform will see for seasonal events (such as the Christmas holiday in December and the spring tax season in the United States) that create very large consistent spikes that might account for more than half of all traffic for the whole year.

These peak times are the most important time to serve the right ad to the right potential customers. To accomplish this you can either build an RTB platform that always has the capacity to handle peak and spiked loads, or you can build your platform to grow and shrink based on the required need. Building elasticity into your platform can dramatically reduce your operating costs. You don't need to maintain peak capacity year-round just to avoid performance issues during important holidays and busy traffic times each day.

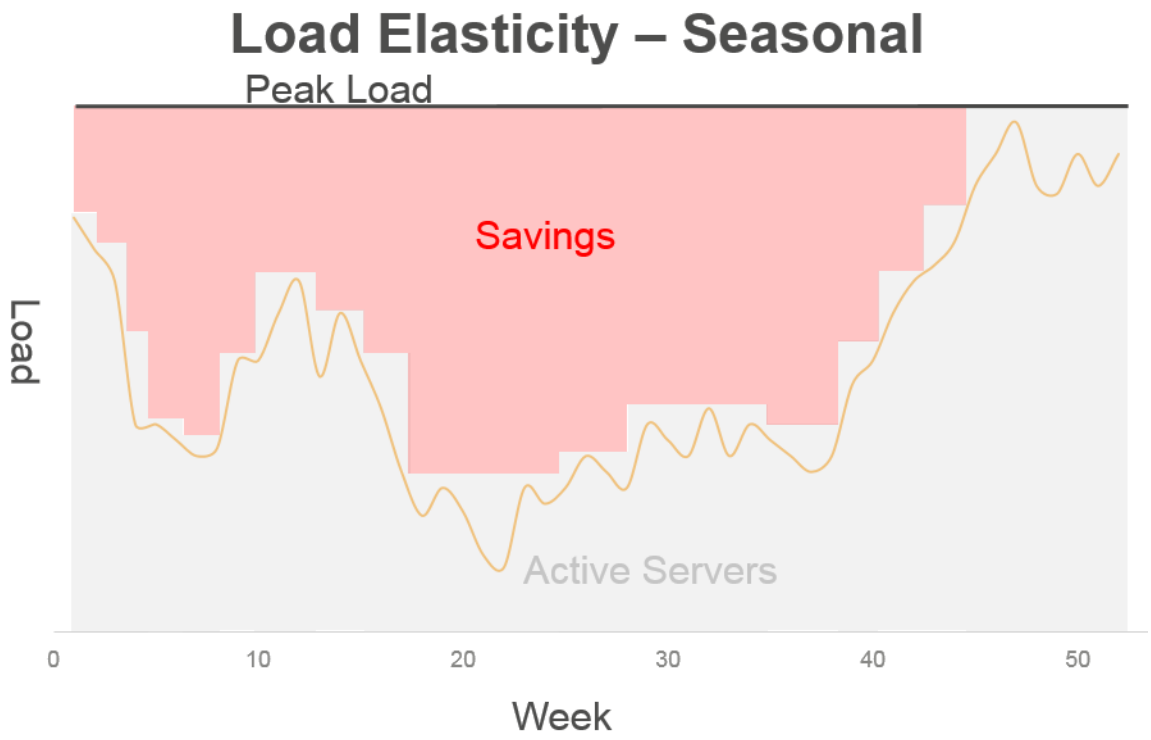

Figure 3: Yearly Load Pattern for RTB

Why Speed Matters

An ad exchange expects to hear an answer from all bidders in 100 milliseconds (ms). If your bid is even a millisecond late you will lose your opportunity to win this ad impression, and your advertisement will go unseen. Lost bids are lost opportunities to get the right advertisement to your key demographic. There are millions of bids per minute, and it's critical for advertisers to have the ability bid on all of them. Therefore, you need to make sure that the entire platform, including the network connection to the exchange, is as quick as possible. Additionally, any less time needed to transmit data is more time you can use to run analytics and make better bidding decisions. Therefore, you want to have your RTB platforms have the lowest latency connection possible to the exchange you are bidding on.

Advertising Is Global

Advertising is becoming a truly global activity. This doesn't necessarily mean that your advertising strategy isn't localized. When providing campaigns to advertisers you want to offer the freedom of reaching customers with localized messaging across the globe. To reach the largest possible audience you need to have an RTB platform physically near all the exchanges throughout the world. You cannot respond to exchanges that are physically far apart and still meet the 100 ms requirement. Therefore, when you plan for building an RTB platform you need to make sure you are able to deploy your platform throughout the world to be as effective as possible.

The Economics of RTB

The digital advertising business is extremely competitive, with ever decreasing margins. Many technological solutions might be able to deliver the required business functionality, however few can deliver it at the very low cost needed to achieve the desired profitability. Costs of RTB can be broken down into two broad categories: costs associated with listening to traffic and recording it and additional costs of executing the bidding logic and populating and maintaining the data repositories related to the bidding process. When you use AWS, these costs can be spread across AWS services with different economics and can be effectively monitored, controlled, and projected through the AWS budgets and forecasts capability. Cost optimization of RTB on AWS is a critical part of a successful solution, with numerous strategies available.

Components of a RTB Platform

This section discusses the components that make up a functioning RTB platform.

Bid Traffic Ingestion and Processing

As a user goes to a website, that website will contact an ad exchange that will then send out bid traffic to RTB platforms for bids on this impression. The bid traffic includes just the website URL that is being browsed, ad/size, and location on that website, and demographic information about the user that the publisher knows. This data must be ingested in real time and a decision must be made on whether you want to bid on this impression and the amount you're willing to bid. Each ad request comes with some form of user identification (ID) from the ad exchange. At this point, the bidder needs to be able to leverage this user ID and all available

data for that user (if this is an existing user that the system has seen previously). The bidder must map this user ID to another source of information (e.g., a cookie store) to match the user, calculate the value of the bid, and probability of winning the auction. Then, the bidder sends the bid, along with the ad link tied to that bid, so that the ad creative can be displayed to the end user in the case of an auction win. To make this decision, the solution must utilize a low-latency data store along with a campaign management system, which will be described in more detail below.

Analysis Traffic Ingestion and Processing

Analysis traffic can come from ad exchanges and directly from content publishers through tracking pixels. Analysis traffic is usually not as time-sensitive as bidding traffic, but it provides valuable information, which can be used to make the real-time bidding decision on future bid traffic. It is important to capture all or as much analysis traffic as possible and not just sample it because analysis traffic improves the system's ability to understand data patterns and learn from them. This data is critical to making an intelligent decision on how much any given impression is worth to the advertiser and how likely it is that this impression will stick with the website user or lead to a direct action like a click-through.

Low Latency Data Repository

The primary purpose of a low latency data repository is to look up and make decisions very quickly on not only if you wish to bid on an impression but also how much you are willing to pay for that impression. This decision is based on three key factors: knowledge about the user (user profile), how well the user matches a set of pre-determined advertising campaigns with specific budget objectives, and how often the user has a specific ad. The key capabilities of this data store are to provide data very fast (preferably in a single millisecond), to scale to peak traffic, and to have regional replication abilities. Regional replication is critical for targeting users who connect from different geographic locations and who can be targeted through advertising exchanges worldwide. The data that is stored in the low latency data repository is an index for fast retrieval set of aggregated data from the durable data repository.

Durable Data Repository for Long-Term Storage

The durable data repository is a storage platform built to hold large amounts of data inexpensively. It will hold all historical data for the analytical pipelines for data transformation, enrichment, and preparation for rich analytics. It's

important to have as much historical data as possible to best be able to predict user behavior and have a good impression bidding strategy. For example, shopping behavior may be very different in December around the Christmas holiday, than in April. If you have data from December of last year or Decembers over multiple years, you can make better predictions about the behavior patterns and demographics that lead to the most valuable impressions. In addition, the advertising customers may have their own "first-party" data about the customers they want to target with RTB, or they might use data from other data providers' third-party data to enhance the RTB process.

Analytics Platform

An analytics platform is used to run computation models, such as machine learning, to calculate the likelihood of specific campaigns getting the desired result from specific demographics and users. This platform will keep track of users across multiple devices, record their activities, and update user profiles and audience segments. It will run the analytics off the different data feeds and the long term durable data repository, It will take the analytical results and store them an indexed manner in the low-latency data store, so that bid processing can quickly find the data it needs to make its bidding decisions.

Campaign Management

Campaign management is typically a multi-tenant web application that manages the advertising campaigns and controls the budgets for different advertisers. This web application provides detailed statistics of the bids that have already taken place in the campaigns and the audiences that have provided the best response. In some cases, the advertising campaign can be manually or automatically adjusted "on the fly," and the information can be pumped back into the low latency data store so that new bidding traffic can incorporate new or updated campaigns.

RTB Platform Diagram

The diagram in Figure 4 displays a generic infrastructure provider independent data flow and each component involved in a generic RTB platform. This illustrates not only the components of an RTB platform but also the interactions with a website from outside sources, such as ad-exchanges, advertisers, user tracking systems, publishers, and end users.

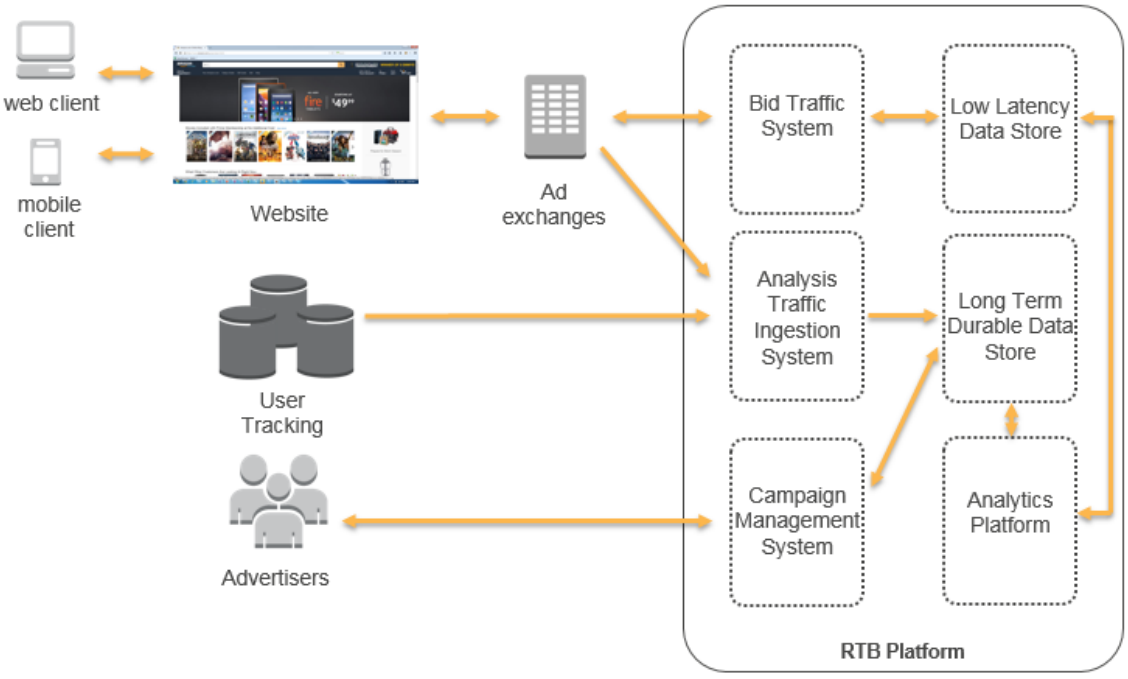

Figure 3: RTB Platform Components

Real Time Bidding on AWS

We will now explore the specific advantages that AWS offers to RTB systems. We'll show how AWS helps RTB providers implement all of the components discussed earlier for their platforms. AWS provides many services and features, so customers can focus on analytics, models, and your own customers instead of spending a significant amount of time on infrastructure, networking, availability, and the platform.

Elasticity on AWS

The AWS platform is built with elasticity in mind; at any time, you can utilize compute, databases, and storage. You only pay for what you use. For example, Amazon Elastic Compute Cloud (EC2) reduces the time required to obtain and boot new server instances to minutes. This allows you to quickly scale capacity, both up and down, as your computing requirements change. You can build your RTB platform to scale up and down in size as more traffic comes in. You also can do computational analytics on your data set in batches and then release the resource back when the analytics are done, so you're not continuing to pay for it. This elasticity not only gives you the assurance that you can handle very large unpredictable spikes in traffic that may occur, but also that you are not tied to architectural or software choices. You can freely change because there is no long-term commitment or investment to your existing infrastructure.

Low Latency Networking on AWS

In the simplest case, both the RTB solution and the exchange are located in the same AWS Region. This is an increasingly popular scenario among the rapidly growing mobile and video exchanges. In some cases, however, the exchange is not located on AWS so the traffic between the RTB solution and the ad exchange goes over the public Internet. To reduce the latency and jitter caused by the Internet, a private connectivity path via AWS Direct Connect can be established between your Amazon Virtual Private Cloud (VPC) that hosts the RTB solution and the provider that hosts the exchange. Some hosting providers may require a public Autonomous System Number (ASN) in order to connect to the exchanges in the most efficient way. If a company does not own a public ASN, this can be accomplished by leasing an ASN from AWS Direct Connect Partners. Additionally when choosing the EC2 instance type you want to make sure to pick instances with enhanced networking with SR-IOV to get the best possible network performance. In some cases, customers may take advantage of Placement Groups that ensure non-blocking low latency connections between instances. In addition, different networking stacks can be deployed to further reduce latency for connections inside the VPC and outside of the VPC.

AWS Global Footprint

AWS offers many different regions around the world where you can deploy your RTB platform to be as close as possible to the different exchanges around the

world. To see a full list of current locations click here. One of the big advantages of the AWS platform is that you can use deployment services like AWS CloudFormation, AWS OpsWorks, and AWS Elastic Beanstalk to easily deploy the exact same architecture to any region you want with a simple click in the AWS Management Console or a service API call. This allows you to easily meet the demands of new campaigns. If you no longer have a campaign tied to a specific geographic location, you can shut down operations at that location until there is demand. Due to the AWS pay-as-you-go model, you will pay nothing once operations cease. When a new campaign starts that requires this geographic location again, just spin it up in minutes with your deployment tool of choice.

The Economics of RTB on AWS

There are several ways of improving the economics of RTB on AWS. Some of the common methods include the following:

1. Elastically scale your compute and memory resources using Auto Scaling to maximize your resources and to ensure that you are paying for peak load when only when you need the resources.
2. Use Spot Instances, especially with latest EC2 Spot Fleet API and Spot Bid Advisor.
3. Use Reserved Instances.
4. Reduce the costs of outbound network traffic with Direct Connect to exchanges outside of the AWS network.
5. Dynamically scale Amazon DynamoDB.

These methods will typically lead to significant savings over building it yourself or using other providers without sacrificing performance or availability.

Components of an RTB Platform on AWS

Now that you have a solid understanding of what RTB platforms are and what their generic components are, let's look at how customers have implemented this successfully on the AWS Platform. The AWS platform offers a rich ecosystem of self-managed servers via Amazon EC2, third party products via the AWS Marketplace, and managed services offerings such as Amazon DynamoDB and Amazon ElastiCache, so there are multiple ways to architect your platform on

AWS. We will explore how each component of an RTB platform could be deployed on AWS.

Bid Traffic Ingestion and Processing on AWS

To build an elastic bid traffic ingestion and processing platform you need to front all traffic into a load-balancing tier. The load balancing can be done in AWS using an Elastic Load Balancing (ELB) load balancer, which is a fully managed software load balancer that will scale with traffic at a very attractive price point. You can also run your own load balancing software such as HAProxy, Netscaler, or F5 on Amazon EC2 instances in a self-managed implementation. However running your own load balancer requires you to ensure scalability and availability across Availability Zones. Typically, DNS with a health check is used to monitor your load balancers and move new traffic around if any of the instances running your load balancer has an issue or is overloaded.

You will also want to scale your web and application tier up and down independently as traffic fluctuates to not only ensure that you can handle traffic demand but also reduce your infrastructure cost when you do not need max capacity of servers to handle the current traffic. You scale your servers yourself using the AWS API or Command Line Interface (CLI), or you can use Auto Scaling to automatically manage your fleet. A best practice is to use the smallest possible instance type that can manage your web and application tier without sacrificing network throughput. This will lead to the lowest possible price when running at your minimum capacity. It will also reduce cost by allowing you to scale up and down in small increments that best match your compute and memory resources to your actual needs as the bid traffic varies throughout the day. For more details on best practices for building and managing scalable architectures see the AWS whitepaper, Managing Your Infrastructure at Scale. An example of launching an open source bidder (RTBkit) on AWS can be found in the RTBkit GitHub repository.

Analysis Traffic Ingestion and Processing on AWS

Analysis traffic can flow into Amazon Kinesis directly from users, or it might require some pre-processing. In the second scenario, it will go through a load balancer to a fleet of scalable EC2 instances that pre-process the data. After data arrives to EC2 instances (Kinesis Producers) and then is forwarded to Amazon Kinesis (likely with some batching to reduce the costs), it can be picked up by a

number of applications directly from the Amazon Kinesis stream using the Kinesis Client Library (KCL). Kinesis Producer Library (KPL) can be used to simplify the process of putting records into an Amazon Kinesis stream. Kinesis is a convenient data store for the multiplexed stream data from several EC2 instances. This data can be used to compute some metrics and do some time window calculations to understand the patterns in the web traffic. In order to optimize the costs for this additional processing step, the data can be flushed in small batches by concatenating the logs to Amazon Kinesis 1 MB record size to minimize the costs associated with the put record requests. From Amazon Kinesis data is typically moved into a durable repository like Amazon S3 and processed with frameworks like Apache Spark (using Spark Streaming and Kinesis integration). In addition, the Amazon Kinesis Firehose service significantly simplifies the process of large volume data capture.

Low Latency Data Repository on AWS

To have a low latency data repository on AWS you need AWS managed services like Amazon DynamoDB and Amazon ElastiCache or a multitude of do-it-yourself options that you would run on Amazon EC2, such as Aerospike, Cassandra, and Couchbase. Amazon DynamoDB offers the simplicity of managing very large tables with low administrative overhead and human intervention, while providing single-digit millisecond latency and utilizing multiple data centers for high durability and availability.

Amazon DynamoDB can be combined with DynamoDB Streams, which captures all activity that happens on a table. This simplifies development and administration of cross-region multi-master replication scenarios. Amazon DynamoDB is a convenient repository for user profile, audience, and cookie data as well as for keeping track of advertising served (frequency capping) and advertising budgets.

Amazon DynamoDB also allows for easily scaling up and down the amount of transaction requests the system can handle on a per-table basis. This allows you to scale your data tier up and down as your transaction load changes throughout the year.

Each table in Amazon DynamoDB has its own provisioned amount of throughput that can be scaled. This makes administration of your database easy; you don't need to turn a clustered set of servers into a set of tables with different

performance characteristics, and you avoid poorly written scales or an unexpected spike in traffic occurring on one table affecting your other tables. This allows you to deploy the concept of hot and cold tables very easily. For example, there is the typical pattern of time-series data where new data is examined often, and older data is rarely needed. In this case, you can create a unique table for each day, week, or month and have the new tables have very high throughput. You can also programmatically dial back the throughput on your older tables over time to further save money since older data accessed less often. This simple per-table throughput administration reduces performance variation and uncertainty found in clusters trying to manage many tables with varying in unpredictable loads.

One of the popular use cases for Amazon DynamoDB is a distributed low-latency user profile store. The user store contains the categories (or segments) a specific user belongs to as well as the times that user was assigned a given segment. This user-segment information can be used as inputs for bidding decision logic. Amazon DynamoDB can be very flexible in terms of schema design and there are several best practices for data modeling. One example of a best practice is to use hash and range keys for data retrieval and modification of multiple items (segments) belonging to the same or different hash keys. In this scenario, the hash key is the user ID, and the range key is the segment the user belongs to.

User ID (Hash Key)	User Segment (Range Key)	Timestamp (Attribute)
1234	Segment1	1448895406
1234	Segment2	1448895322
1235	Segment1	1448895201

Durable Data Repository for Long-Term Storage on AWS

Amazon Simple Storage Service (S3) provides a scalable, secure, highly available, and durable repository for analytical data. Amazon S3 runs a pay-as-you-go model, so that you are only charged for what you used. Amazon S3 also has

different storage classes, S3 standard for general-purpose storage, S3 Infrequent Access (S3 IA) for data that is long-lived but infrequently accessed, and Amazon Glacier for a long-term archive. You can also set up Object Lifecycle Management policies, which will move your data between these different storage options based on a schedule at no additional cost. For example, a policy might move data older than a year to S3 IA, then after three years to Amazon Glacier, and then after seven years the data is deleted. Amazon S3 is a durable, scalable, and inexpensive option for RTB long-term storage that can then be used as a data source for the analytical pipelines for data transformation, enrichment, and preparation for rich analytics.

AWS has several technologies you can use for distributed data transformation. Amazon Elastic MapReduce (EMR) is a managed cluster compute framework that can natively read directly from Amazon S3 utilizing open source tools such as Apache Spark. In addition, AWS Data Pipeline is a highly available managed service that allows easy data movement. Processing jobs can be implemented for managing workflows including those done by Amazon EMR and other processing and database technologies. Finally, you can take advantage of event-driven processing when objects are written to Amazon S3. Event-driven processing can automatically trigger an event handled by an AWS Lambda function to simplify processing at scale and not require batch-based architectures.

RTB Analytics Platform on AWS

AWS has a wide variety of analytics platforms that can be utilized by RTB platforms so that bidding decisions can be as effective as possible. In the machine learning space for very large data sets, a common pattern is to use the machine learning library that comes with Spark MLlib on EMR. You can also utilize other tools that run on Amazon EMR, or you can use a managed service such as Amazon Machine Learning (Amazon ML). All of these options have full integration with Amazon S3 storage for your long-term data set. This allows the data to be analyzed so you can utilize many different tools to achieve your predictive analytics goals. You can also read about different options and benefits AWS provides for large-scale analytics in the Big Data Analytics Options on AWS whitepaper. Typically, an analytical workload requires a workflow component and can be implemented using Amazon Simple Workflow Service (SWF), AWS Data Pipeline, or AWS Lambda.

Campaign Management on AWS

Campaign management systems architectures on AWS look like typical well-architected web applications, similar to those described in our bid-processing system, but this time with a full-scale persistent data tier. Campaign management should exist in Auto Scaling groups, sit behind ELB load balancers and security groups, and deploy in multiple Availability Zones for high availability. You can use Amazon Relational Database Service (RDS) for your campaign management. Amazon RDS is a managed RDBMS service that supports Oracle, SQL Server, Aurora, MySQL, PostgreSQL, and MariaDB engines. Amazon RDS will install, patch, maintain, perform multi-AZ synchronous replication, and back up your database. You could also run your own database technology on Amazon EC2, but you would need to take ownership of managing and maintaining that database yourself. Your application will typically tie into your low-latency data tier to provide real-time information on the success of your campaigns back to your customers. We recommend using a content delivery network such as Amazon CloudFront, which is a managed content delivery network that helps speed up and securely deliver dynamic and static data (e.g., JavaScript, ad images) as close to your users as possible.

Reference Architecture Example

Figure 5 is an example of a reference architecture that customers have successfully deployed. It has Auto Scaling groups to allow for scalability, and it spans multiple Availability Zones, so that any localized failure would not stop its ability to responds to bids.

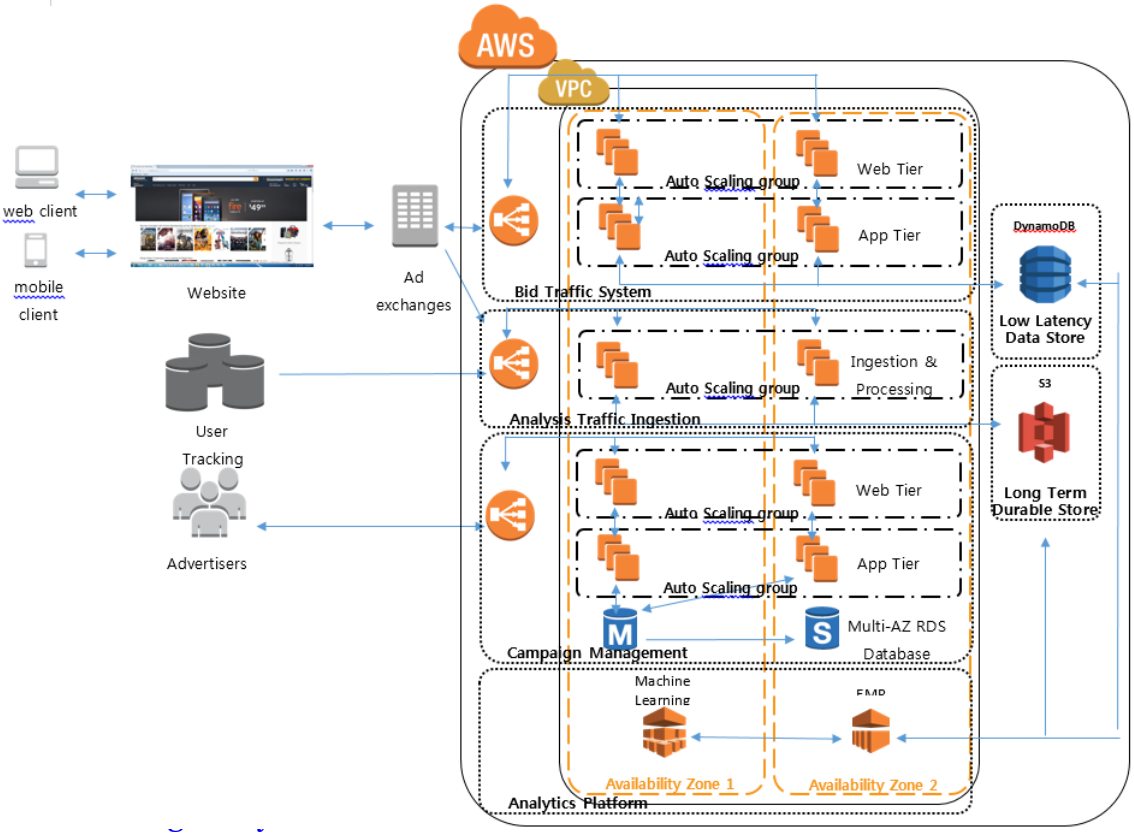

Figure 5: Example Reference Architecture

Conclusion

Real-time bidding is a growing trend that has many different components required to effectively deliver intelligent real-time purchasing of media. The AWS platform is a perfect fit for each component of the RTB platform due to the global reach and breadth of services. An RTB architecture on AWS allows you to get the real-time performance necessary for RTB as well as reduce the overall cost and complexity involved in running an RTB platform. The result is a flexible big data

architecture that is able to scale along with your business on the AWS global infrastructure. Deploying on AWS offloads a significant amount of the complexity of operating a scalable real-time infrastructure, so that you can focus on what differentiates you from your competitors and focus on making the best possible bidding strategies for your customers.

Contributors

The following individuals and organizations contributed to this document:

- Steve Boltuch, solutions architect, Amazon Web Services
- Chris Marshall, solutions architect, Amazon Web Services
- Marco Pedroso, software engineer, A9
- Erik Swensson, solutions architect manager, Amazon Web Services
- Dmitri Tchikatilov, business development manager, Amazon Web Services
- Vlad Vlasceanu, solutions architect, Amazon Web Services

Further Reading

For additional help, please consult the following sources:

- IAB Real Time Bidding Project
- Beating the Speed of Light with Your Infrastructure on AWS
- Deploying an RTBkit on AWS with a CloudFormation Template

Notes

[1] US Programmatic ad spend to double by 2016 eMarketer analysis

[2] US Programmatic digital display ad spending 2014-2017 eMarketer analysis2014-2017

[3] US Programmatic ad spend to double by 2016 eMarketer analysis