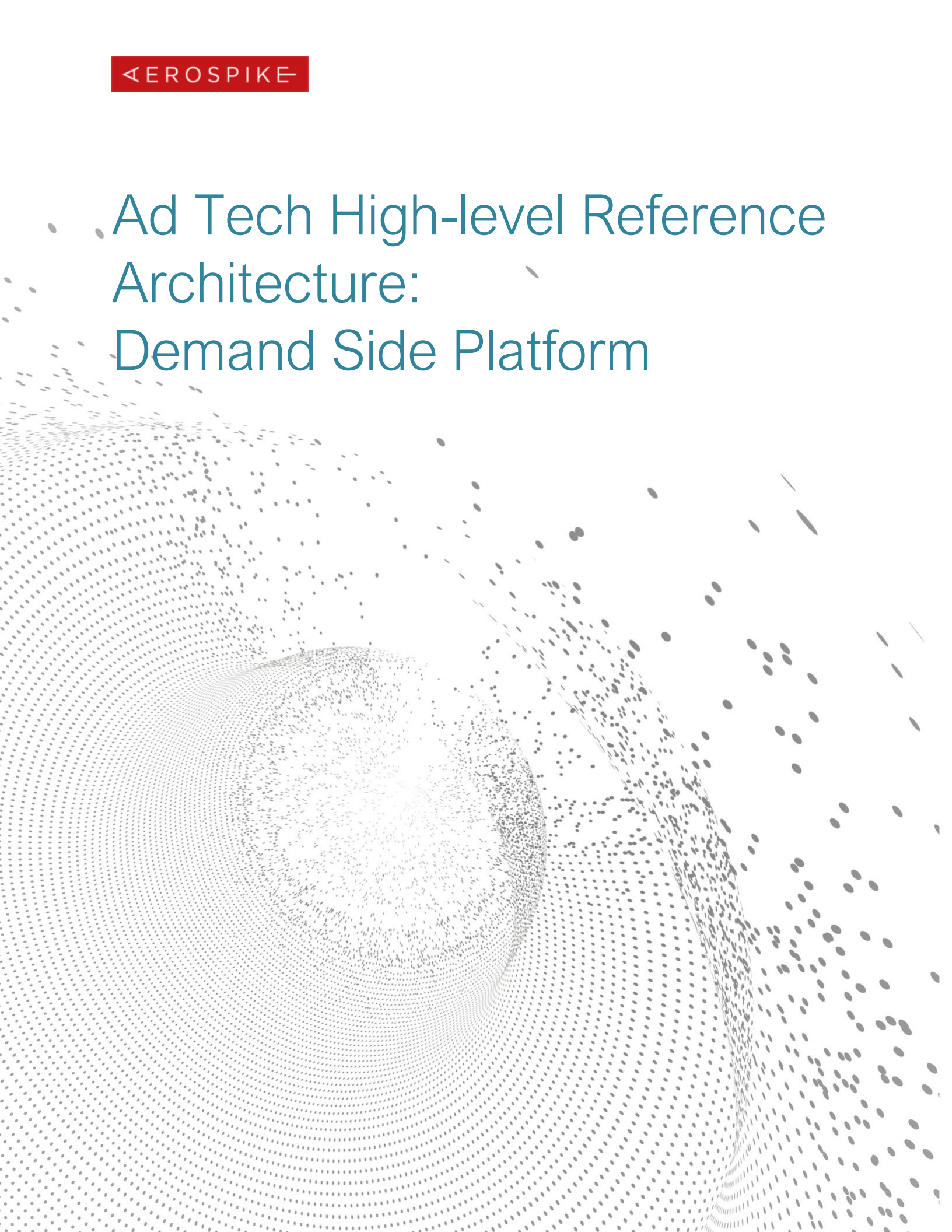
AEROSPIKE

Ad Tech High-level Reference Architecture: Demand Side Platform



Contents

Executive Summary

This document provides a simplified reference architecture for Demand Side Platform datastores for real-time bidding and campaign reporting using Aerospike as the datastore technology.

Aerospike's Hybrid memory architecture™ coupled with its built-in parallelism enables sub-millisecond latencies and millions of operations per second throughput on large data sets, even on a single server. Aerospike achieves these impressive figures through its strong consistency mode, which is required in numerous application use cases. Optimized for hardware at the storage, network and CPU layers, Aerospike can run on 25% to 35% of the server footprint of other NoSQL databases while maintaining five nines or more uptime.

Demand Side Platform (DSP)

A DSP provides Campaign Definition, Execution and Reporting on behalf of advertisers and/or their agencies.

Campaign Definition specifies the details of the campaign including duration, target audiences execution plans, budget, client, etc.

Campaign Execution is usually divided into smaller execution plans, often called line items or placements. Each execution plan is a portion of the Campaign and will have it's own target audiences (segments), budget, duration, etc. Execution plans are used by real-time bidding engines in the bidding and pricing process.

Campaign Reporting is the collation and aggregation of events produced by the user while interacting with the displayed Ad.

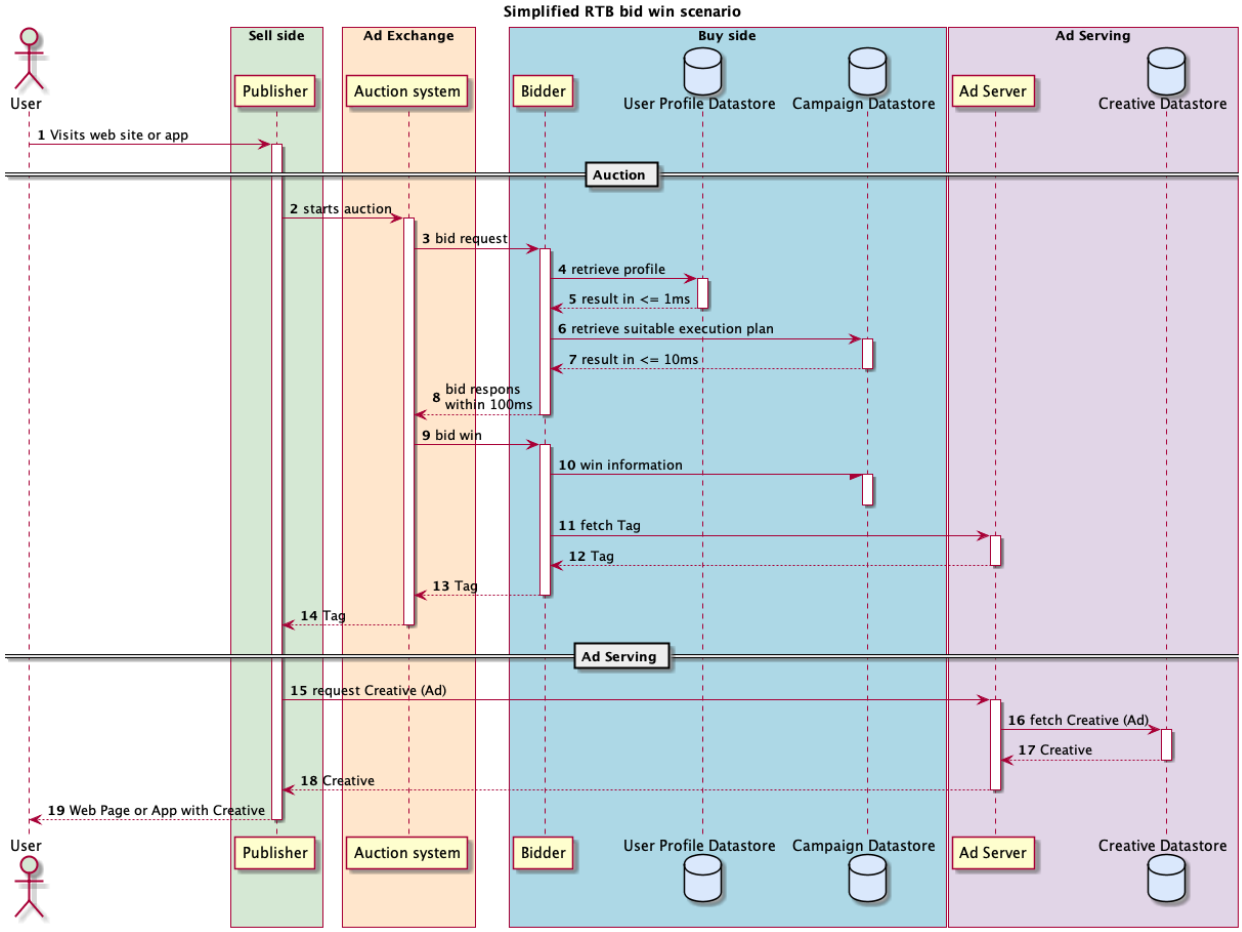
Real-time Bidding - Campaign execution

As data volumes increase at an unprecedented rate, firms face ever-greater challenges: deliver new applications faster, apply analytical technologies (such as machine learning) to hundreds of terabytes data (or more), provide a reliable and engaging user experience, drive digital transformations, and more. How do these map to the signs that you've outgrown Redis?

Real-time bidding is the process of auctioning advertising space to the highest bidder. Publishers of a web site, device application, etc. sell their advertising space, often called inventory via an auction. Advertisers, via their DSP, purchase this space by bidding in the auction. The auction is usually conducted through an Ad Exchange.

The Advertiser that wins the bid has purchased the space from the Publisher and provides a Tag to the Publisher to display the Ad rendered in the webpage or the device app. The Tag also contains information to track the user's interaction with the Ad and generate events for reporting.

The following sequence diagram shows a *simplified* publish-bid-win scenario:



A DSP will receive up to 10's of Millions of bid requests per second depending on the time of the year and their geographical market. A bid response is required in about 75 milliseconds and the auction is complete (win or lose) in about 100 milliseconds.

A DSP solution requires scalable, high throughput, low latency and high availability hardware and software, and is architected to respond to each bid request with the best price based on the anonymous profile of the user interacting with the publisher.

The Ad Server plays a significant role in the performance of the Publisher's website or App. If the Ad Server is sluggish to deliver the Creative, the Publisher appears to have poor performance.

Datastores

User Profile Store

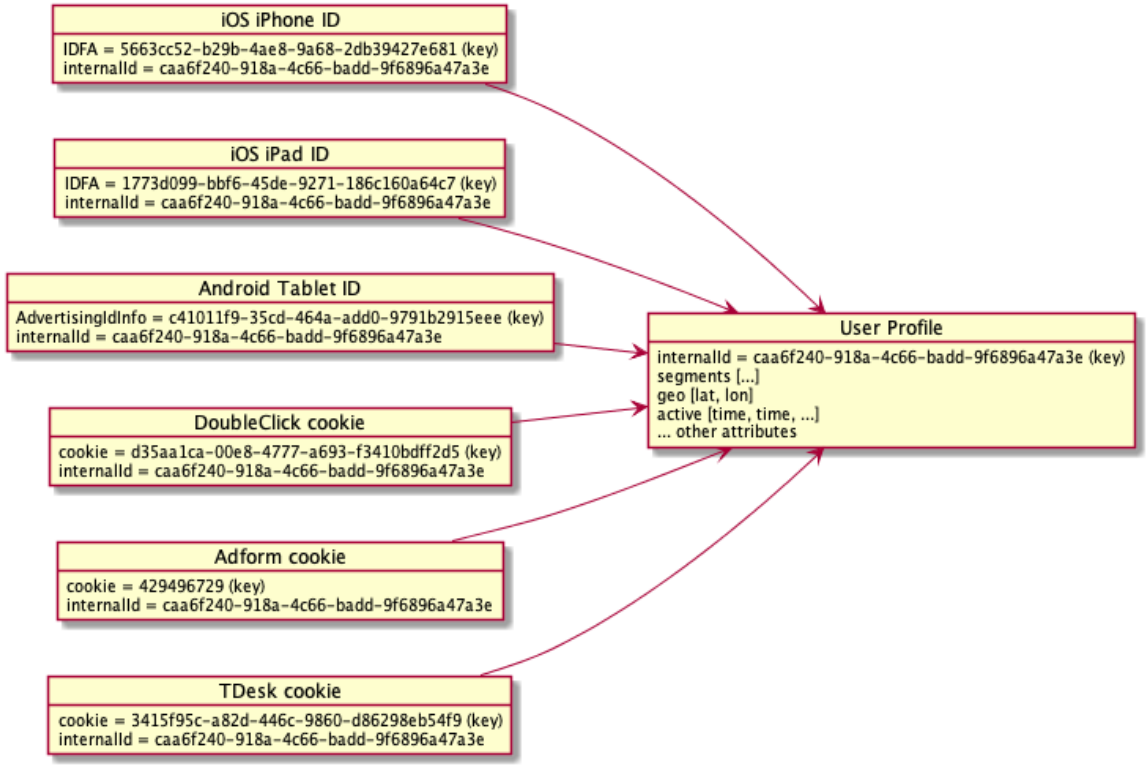
The user profile store often called the cookie store, contains anonymous user profiles detailing the profile's

- Internal ID
- Audience segments
- Geographic location
- Activity time
- …and many more details

As users usually have more than one device, any given user may have several advertising IDs, supplied by their devices and from cookies in their browser. Cross-device matching can identify several IDs as the same user, using data science techniques. This matching is not 100% accurate but is "good enough" for programmatic advertising.

A typical user profile schema has a record for each device ID that has a pointer to the user profile.

Simplified User Profile Schema



The scale of a user profile store is not trivial and is in the order of billions of records, each access to any record is expected to have a latency of under 1 millisecond. 10's of millions of reads and writes occur each second, every second of the day and every day of the year. The store must be reliable, a user profile store outage for just 1 hour can cost a DSP hundreds of thousands of dollars in lost revenue.

Latency, Throughput, Scale and Availability

(See: Speed, Scale, Durability and Affordability)

Requirements	
Number of records	10's of billions
Operations per second	10's of millions
Acceptable latency	1 millisecond or less
Read Write ratio	80% reads 20% writes
Availability	24/7/365 - never down
CAP theorem	Prioritize Availability over Consistency
Consistency	Weak

The data is denormalized for performance with the goal of the minimum number of database operations.

Example Record Structure

Cookie ID			
Bin/Key	Field	Type	Example value
key	Cookie Id	String	429496729
profile-id	User profile Id	String (uuid)	caa6f240-918a-4c66-badd-9f6896a47a3e
Advertiser ID (iOS or Android)			
Bin/Key	Field	Type	Example value
key	Advertising Id	String (uuid)	c41011f9-35cd-464a-add0-9791b2915eee9
profile-id	User profile Id	String (uuid)	caa6f240-918a-4c66-badd-9f6896a47a3e
User Profile			
Bin/Key	Field	Type	Example value
key	Internal Id	String (uuid)	caa6f240-918a-4c66-badd-9f6896a47a3e

segments	Audience segments	CDT (Map/List structure)	[{ "id": "6de5c557-73f2-48c0-84c7-be0f874c5ac8", "name": "human readable name", ... }, { "id": "b4b0dfc6-40c6-4414-a751-4f3ddb252233", "name": "human readable name", ... } ...]
geo	User geo location	GeoJson	{ "type": "Point", "coordinates": [12.483354, 55.724556] }
active	Active time periods	CDT (Map/List structure)	[{ "from": 1543569904, "to": 1543656304 }, { "from": 1543915504, "to": 1544001904 }]

Campaign Datastore

The campaign datastore is a system of record containing:

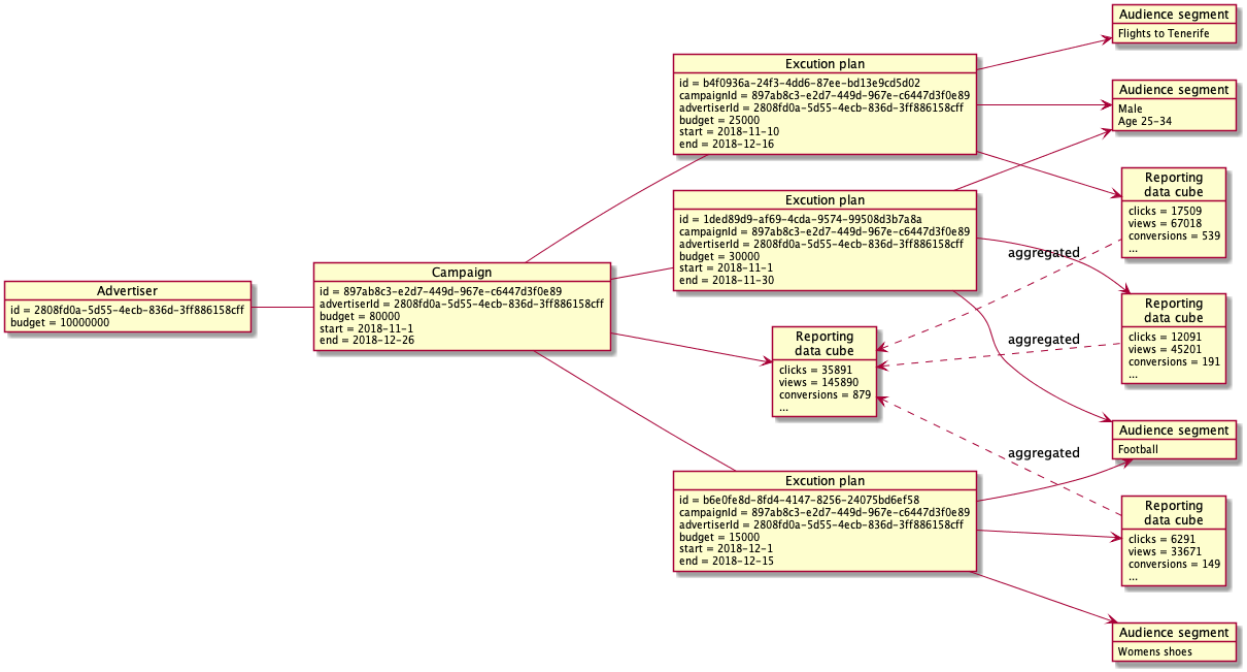
- Campaign Definitions
- Execution Plans (line items)
- Advertisers or Advertising Agencies
- Audience segments
- Data cubes
- … other information

Execution Plans are consumed by the bidding engine(s) to match the bid request to the available campaigns and determine the bid price.

The Execution Plan, and therefore Campaign, performance data is aggregated from the stream of Events and stored as a data cube. The Campaign User Interface uses this “real-time” data to show campaign performance. Billing of Advertisers and payment to Publishers is also based on the campaign reporting data and the campaign store requires strong consistency.

Simplified Campaign Schema

The Campaign schema is denormalized for performance and almost all operations use a primary key to reference data.



The Campaign datastore has requirements similar to other traditional business systems and is a System of Record. The datastore requires frequent and numerous updates to the metrics measuring the performance of a campaign. In a small scale, it is possible to use relational technology, but as scale increases, it is impossible to scale relational technology at an affordable price point to meet the demand.

Latency, Throughput, Scale and Availability

Requirements	
Number of records	
Operations per second	100's of thousands
Acceptable latency	5-15 milliseconds
Read Write ratio	50% reads 50% writes
Availability	High availability
CAP theorem	Prioritize Consistency over Availability
Consistency	Strong

The record structure is denormalized for performance and also has a "document" like data cube to hold the reporting metrics.

Example Record Structure

Advertiser			
Bin/Key	Field	Type	Example value
key	Advertiser Id	String (uuid)	2808fd0a-5d55-4ecb-836d-3ff886158cff
budget	Advertiser Budget	Long	120000
Campaign			
Bin/Key	Field	Type	Example value
key	Campaign Id	String (uuid)	897ab8c3-e2d7-449d-967e-c6447d3f0e89
advertiser-id	Advertiser Id	String (uuid)	2808fd0a-5d55-4ecb-836d-3ff886158cff
start	Start Date	Long (time stamp)	1541063106
end	End Date	Long (time stamp)	1545815106
budget	Campaign budget (portion of advertiser budget)	Long	80000

data-cube	Campaign Reporting
	(aggregation of execution plan data)
		CDT
		(Map/List structure)
			<pre>{
			"clicks": 35891,
			"views": 145890,
			"conversions": 879
			}</pre>

Execution Plan			
Bin/Key	Field	Type	Example value
key	Execution plan Id	String (uuid)	1ded89d9-af69-4cda-9574-99508d3b7a8a
campaign-id	Campaign Id	String (uuid)	897ab8c3-e2d7-449d-967e-c6447d3f0e89
advertiser-id	Advertiser Id	String (uuid)	2808fd0a-5d55-4ecb-836d-3ff886158cff
start	Start Date	Long (time stamp)	1541063106

end	End Date	Long (time stamp)	1543569904
budget	Execution plan budget (portion of campaign budget)	Long	30000

data-cube	Execution plan reporting	CDT
		(Map/List structure)
			<pre>{
			"clicks": 1209,
			"views": 45201,
			"conversions": 191
			}</pre>

Tag to Execution Plan			
key	Tag ID	String (uuid)	d114969b-19ac-474b-bdd0-cdcd9d5e2fbb
ex-plan-id	Execution plan ID	String (uuid)	1ded89d9-af69-4cda-9574-99508d3b7a8a

Document vs Relational

The Campaign datastore can be implemented using any database technology and as a system of record relational databases come first to mind, so why use Aerospike?

Database operations on the Campaign Datastore are simple, single record transactions where Aerospike excels, and require no multi-record atomic transactions where relational technology excels.

High throughput, high availability and low latency are essential in providing real-time campaign reporting and an affordable cost.These requirements match the capabilities of Aerospike with the added advantage of a low cost because of Aerospike's unique use of Flash storage.

Aerospike Complex Data Types (CDT) are used as the Data cubes for each execution plan. CDTs are the document store functionality of Aerospike. Updates to elements of a CDT are granular and atomic. Data cube values (counters, averages, histograms) can be precisely and atomically updated with low latency.

Here is an example of a very simple Execution Plan data cube using a CDT (expressed in JSON)

```
{
    "datacube": {
        "clicks": 35891,
        "impressions": 145890,
        "visits": 2103,
        "conversions": 879
    }
}
```

Real-time Campaign Reporting

Each Advertiser/Agency wants to measure the performance of their Campaign. Did the user see the Ad, did they click on it, play it (video), visit the vendor's site?

When a user interacts with an Ad rendered by a Publisher a number of events are generated. There are many kinds of events, some examples are:

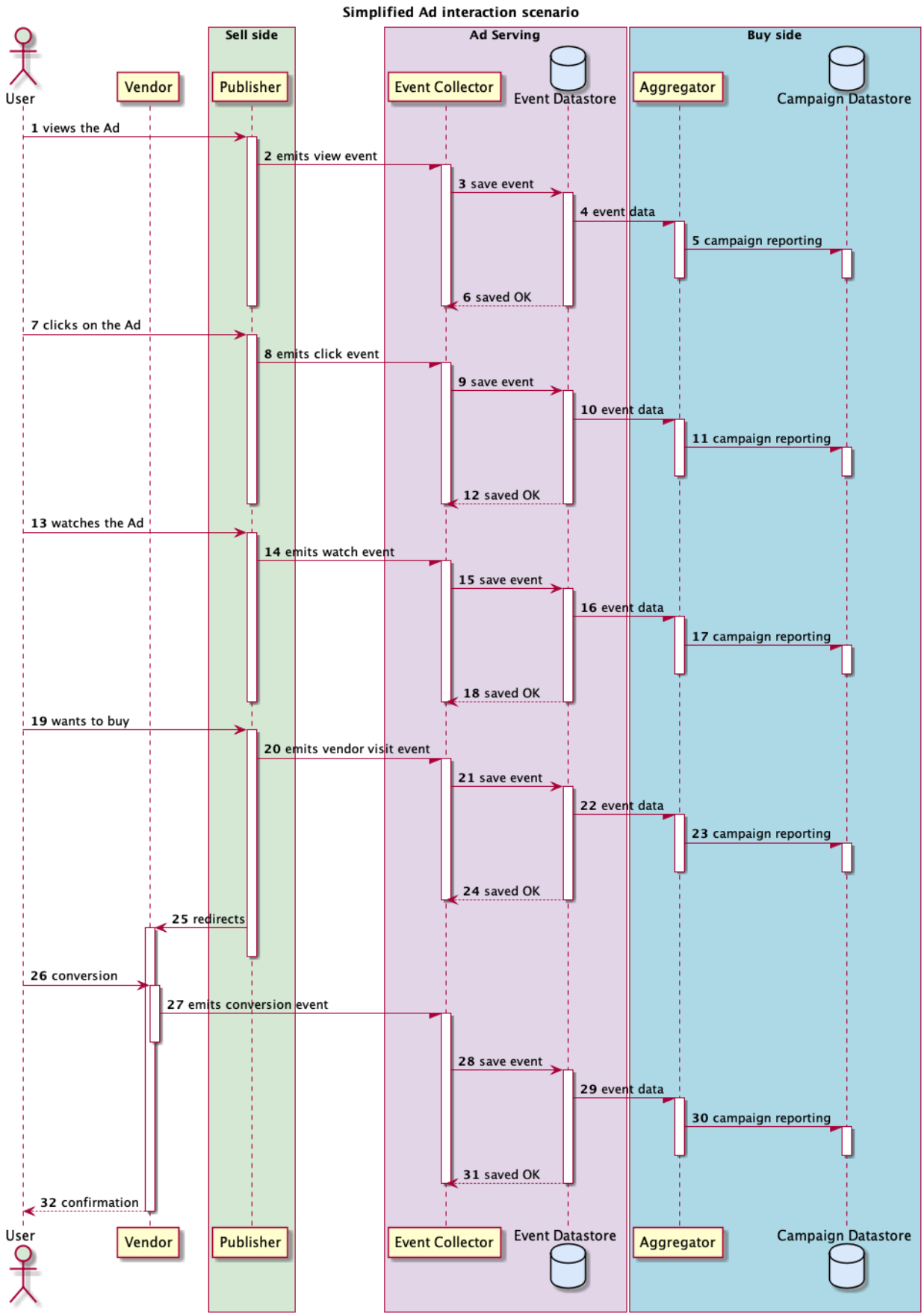
- Ad impressions
- Ad clicked
- Ad watched (video)
- Ad watched Duration (Video)
- Visiting the vendor site
- Conversion to a sale
- ...and there are many more

These events indicate the engagement of the user with the Ad and when collated and aggregated represent the performance of the Execution Plan and the overall Campaign. Events are **emitted** from the Publisher and are **consumed** by the event collector associated with the Ad Server.

For every bid win and Ad served, then event collector will receive 1 or more events, and thus it also requires high throughput, latency and availability.

Consider the scenario where the user interacts with the Ad and wants to buy. Several events are emitted and collected to form Campaign reporting.

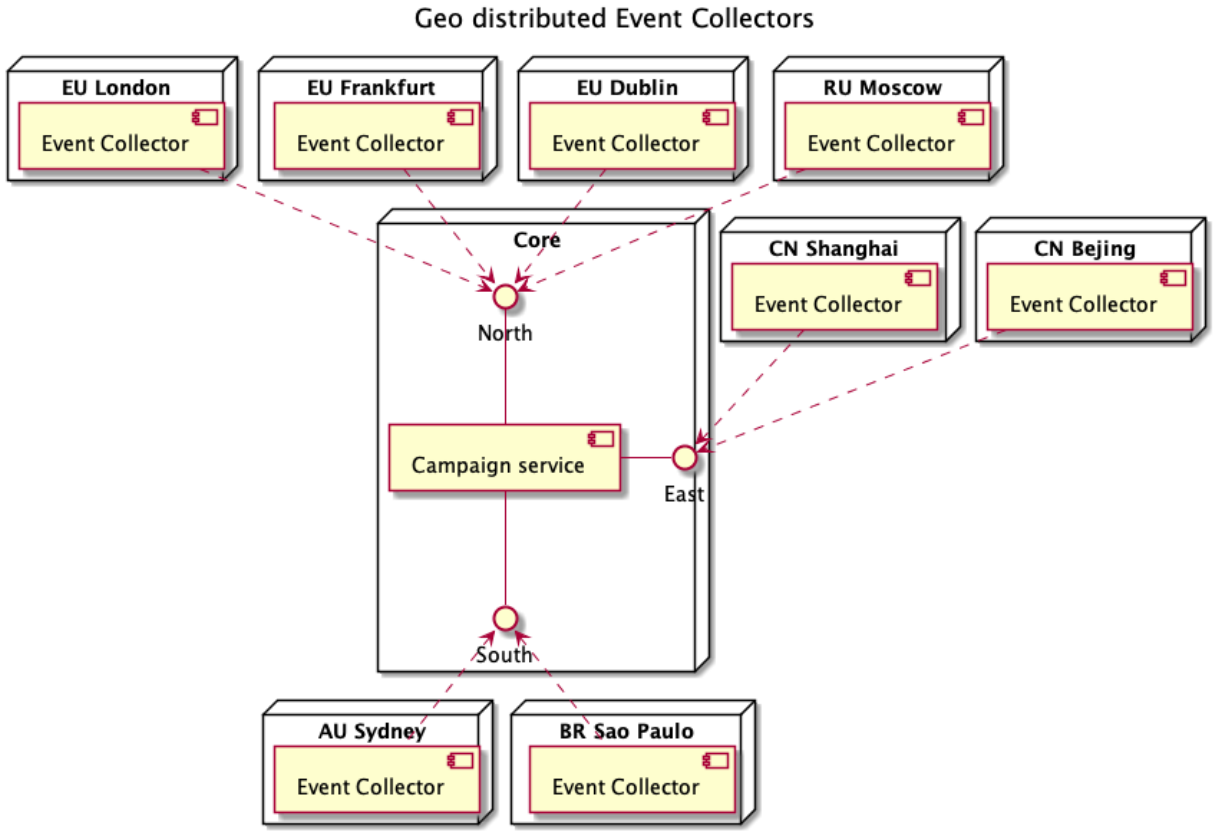
As bidding and serving are done in real-time, it is desirable for Campaign reporting to be done in near real-time.


Simplified Ad interaction scenario
Sell side
Ad Serving
Buy side
User
Vendor
Publisher
Event Collector
Event Datastore
Aggregator
Campaign Datastore
1 views the Ad
2 emits view event
3 save event
4 event data
5 campaign reporting
6 saved OK
7 clicks on the Ad
8 emits click event
9 save event
10 event data
11 campaign reporting
12 saved OK
13 watches the Ad
14 emits watch event
15 save event
16 event data
17 campaign reporting
18 saved OK
19 wants to buy
20 emits vendor visit event
21 save event
22 event data
23 campaign reporting
24 saved OK
25 redirects
26 conversion
27 emits conversion event
28 save event
29 event data
30 campaign reporting
31 saved OK
32 confirmation
User
Vendor
Publisher
Event Collector
Event Datastore
Aggregator
Campaign Datastore

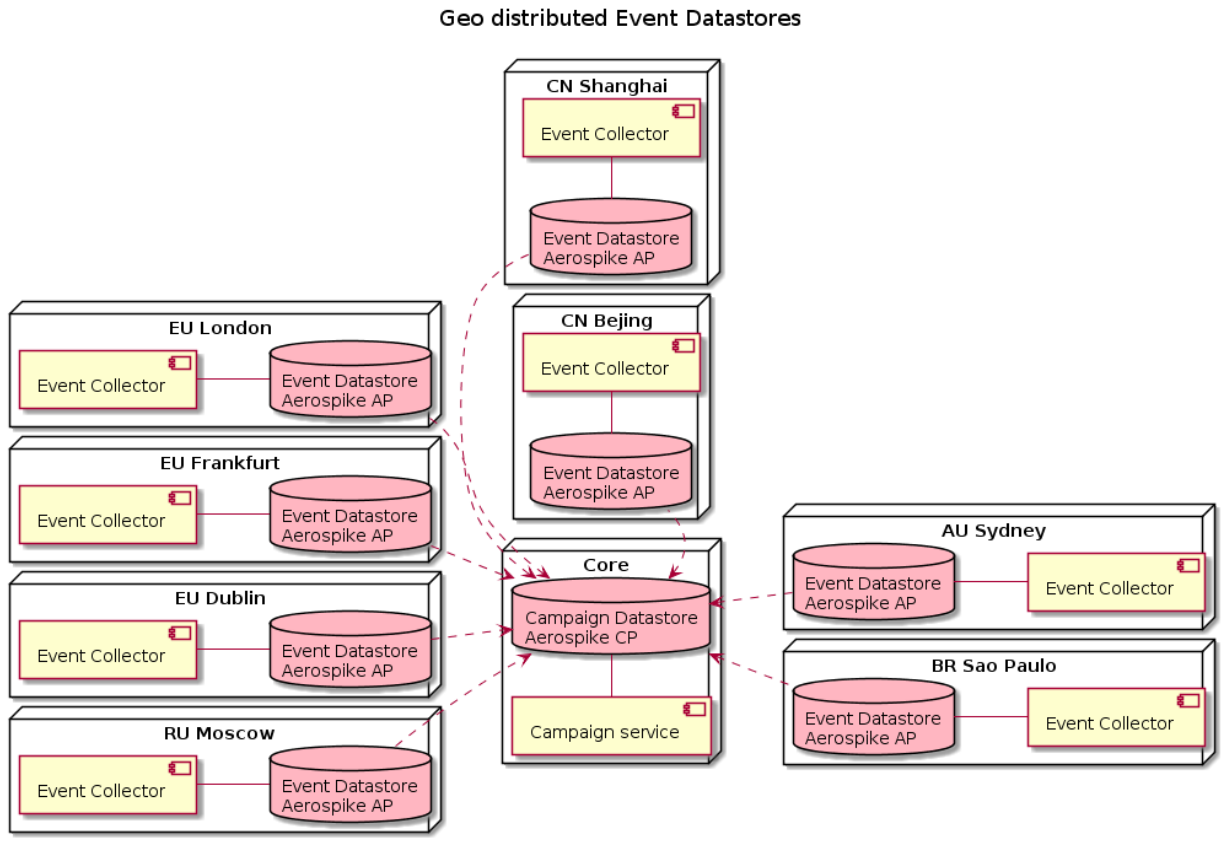
Event Collector

The Event Collector is a high availability web API that receives events sent from the Publisher site. The Tag encapsulating the Creative (Ad) calls the event collector API. Each event is unique and contains data that relates the event to an Execution Plan related to a Campaign. A way to think of event data is as raw data or log level data.

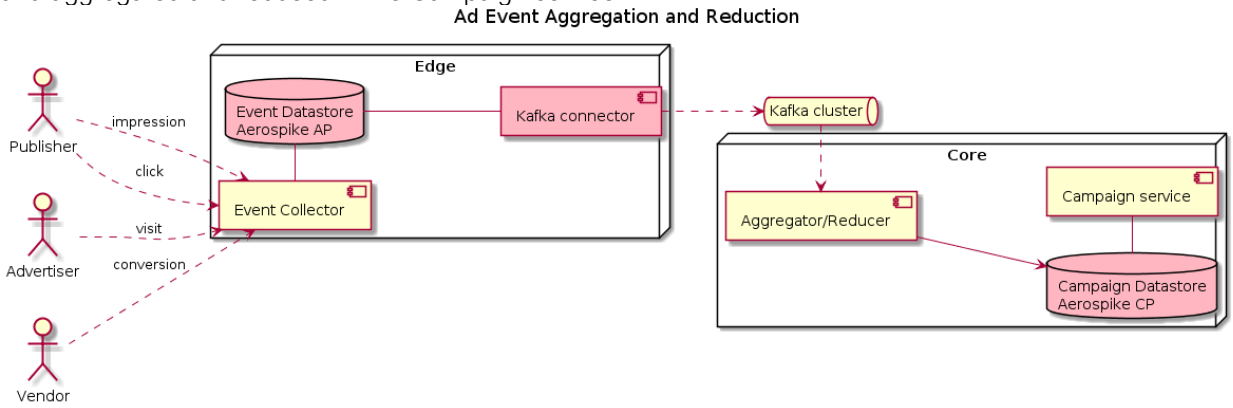
The Event Collector is located geographically and highly parallelized for high throughput, high availability and low latency, typically using containers (Docker) and orchestration (Kubernetes).



Events are received and recorded in an event datastore (do not confuse this with 'Event Store' technology) for later transmission the Campaign datastore for Campaign and Execution Plan reporting. This is an edge datastore geographically located with the Event Collector.



Data is propagated from the Event datastores to the Campaign datastore using Aerospike Kafka Connector and aggregated and reduced in the Campaign service.

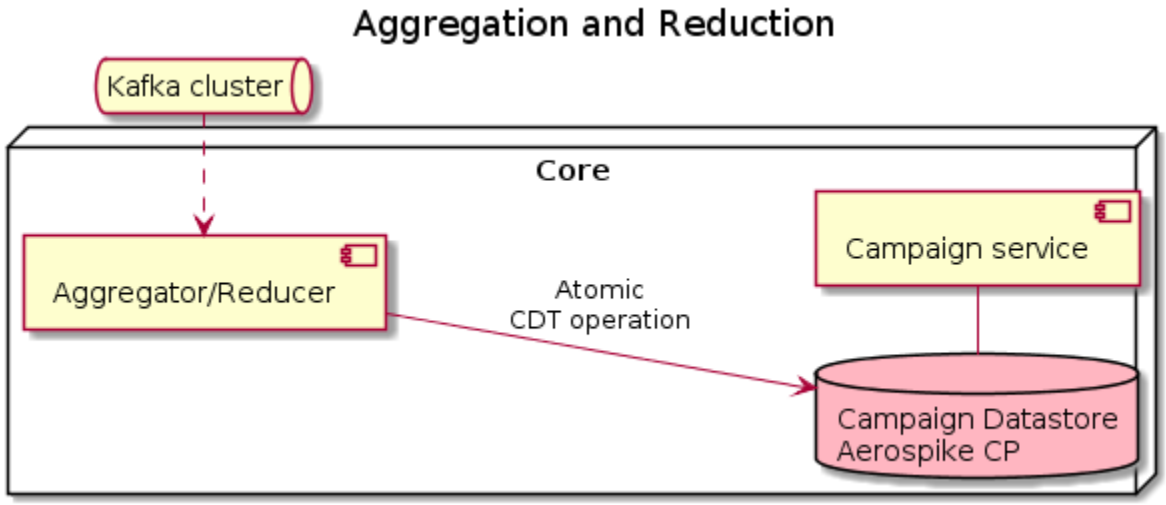


It is possible to build this architecture without geolocated Event collectors or datastores when the throughput is small but on a global scale high throughput and low latency are the basic requirements.

Why use Aerospike as the edge Event datastore? Aerospike allows the event to be “stored and forgotten” by the event collector. Aerospike’s low latency ensures that the service time for each event is small and this results in a smaller number of Event Collectors, ultimately saving money.

Aggregator/Reducer

The Aggregator/Reducer receives messages/events from Kafka, aggregates/reduces the raw event data and updates the data cubes in the Campaign Datastore.



In its simplest form, this process increments counters, more complex scenarios update histograms, matrices, etc.

Why not simply use Kafka? Events directly into Kafka is certainly possible, but the throughput of each Event Collector instance will be dependent on Kafka throughput and ultimately the size of the Kafka cluster. The combination Aerospike, as the Event datastore, and a smaller Kafka cluster allow peek throughput at a lower cost than using a large scale Kafka cluster alone.

Datastores

Campaign Datastore

The Campaign datastore is described above and includes data cubes for each Execution Plan and optionally for the Campaign as a whole.

Event Datastore

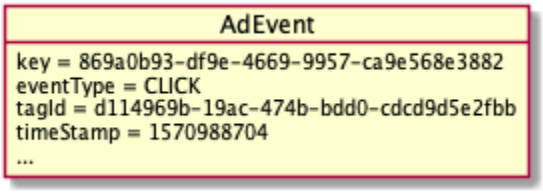
An edge data store, the Event datastore captures raw events for later processing. Edge data stores are located geographically and therefore capture the “local traffic” events, for later aggregation and reduction in the core Campaign datastore. They act as a low latency buffer between buffer between events and aggregation process, they also are the source data for Data Science.

Latency, Throughput, Scale and Availability

(See: Speed, Scale, Durability and Affordability)

Requirements	
Number of Records	100's of billions
Operations Per Second	100's of thousands
Acceptable Latency	5-10 milliseconds
Read Write Ratio	50% reads 50% writes
Availability	24/7/365 - never down
CAP Theorem	Prioritize Availability over Consistency
Consistency	Weak

Simplified event schema



Example Record Structure

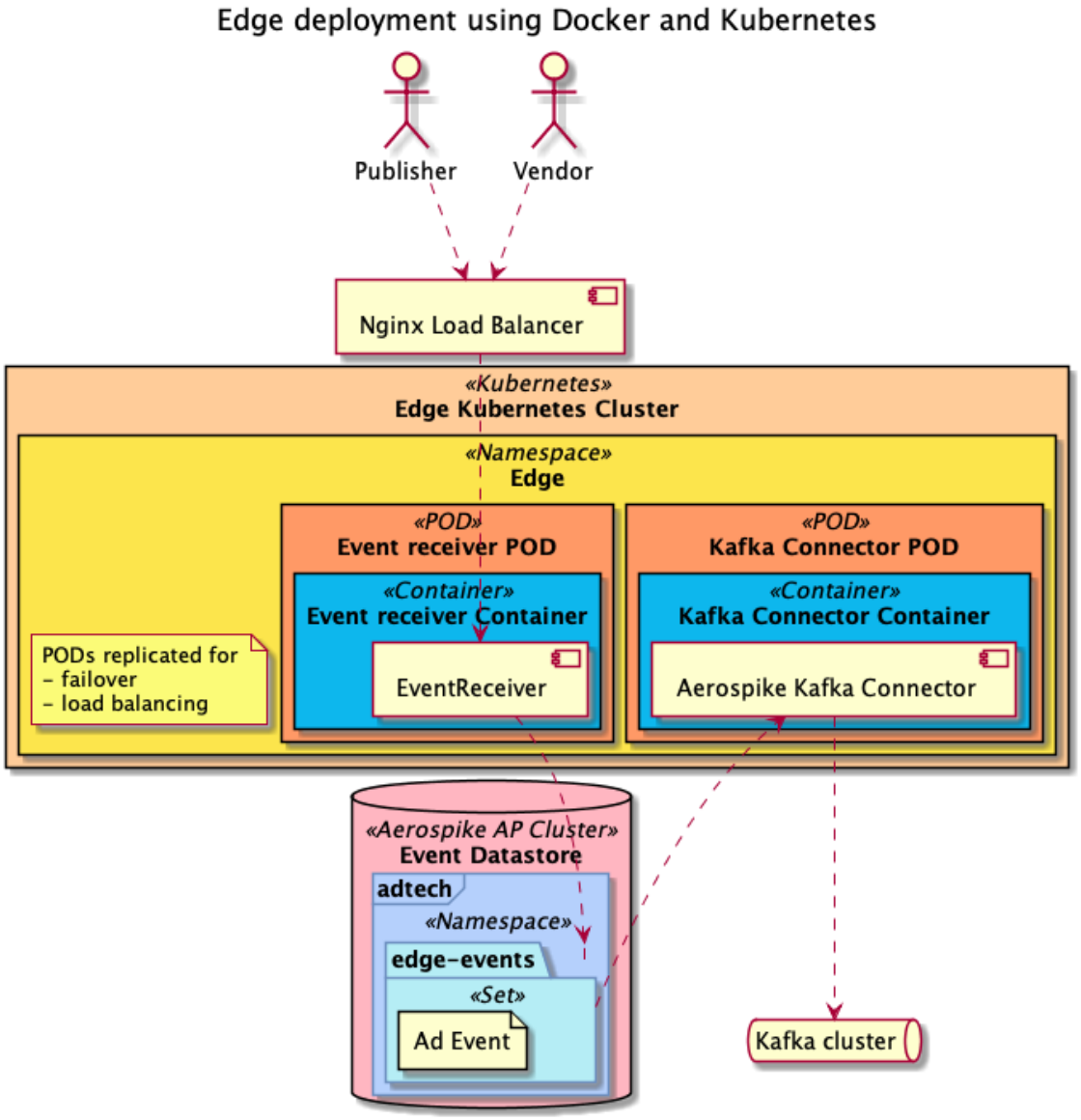
Event			
Bin/Key	Field	Type	Example value
key		String (uuid)	869a0b93-df9e-4669-9957-ca9e568e3882
event-type	Type of event	String	"CLICK"
tag-id	ID of the Tag	String (uuid)	d114969b-19ac-474b-bdd0-cdcd9d5e2fbb
time-stamp	Time stamp of the event	Long (time stamp)	1570988704deployment

Deployment in Production

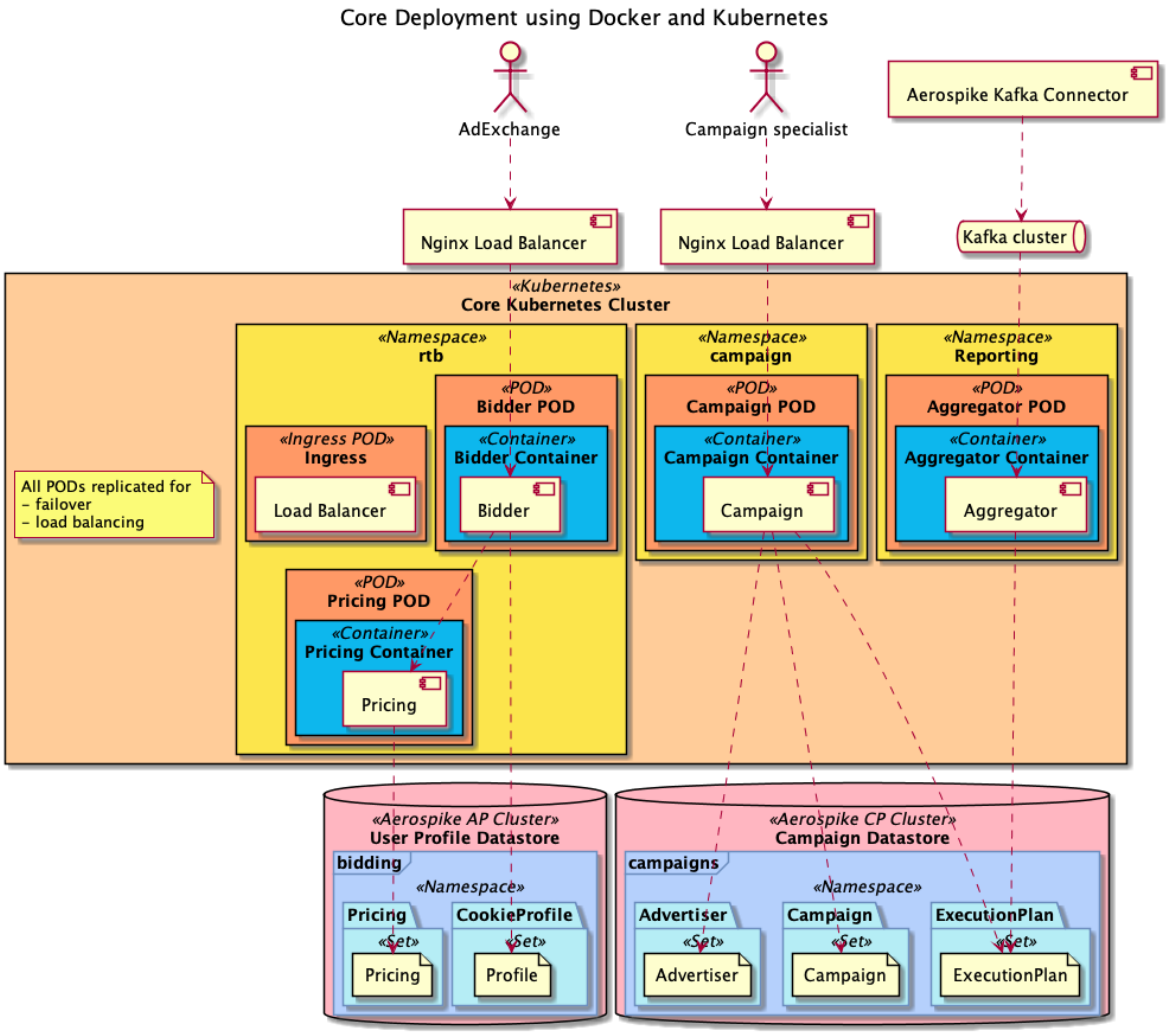
High availability, low latency, high throughput and scale are the common theme. Aerospike is all of these, and the services to capture events, aggregate them and interact with the Campaign also require similar characteristics.

A successful approach is to use containers with orchestration, specifically Docker and Kubernetes.

Edge Deployment
Deployments located in each geographical datacenter are minimal. Pods are replicated for high availability and throughput.



Core Deployment
The Core deployment is centrally located and is more sophisticated. Pods are replicated for high availability, low latency and throughput.



Technology References

Aerospike	https://www.aerospike.com/
Kafka	https://kafka.apache.org/
Nginx	https://www.nginx.com/
Docker	https://www.docker.com/
Kubernetes	https://kubernetes.io/

Conclusions

Aerospike is datastore technology of choice for the AdTech industry because of its low latency, high throughput, large scale, reliability and low cost of ownership.

We have covered how a DSP uses Aerospike as a User Profile (cookie) store for real-time bidding and how to use edge-to-core Aerospike technology for fast campaign reporting.

About Aerospike

Aerospike enterprises overcome seemingly impossible data bottlenecks to compete and win with a fraction of the infrastructure complexity and cost of legacy NoSQL databases. Aerospike's patented Hybrid Memory Architecture™ delivers an unbreakable competitive advantage by unlocking the full potential of modern hardware, delivering previously unimaginable value from vast amounts of data at the edge, to the core and in the cloud. Aerospike empowers customers to instantly fight fraud; dramatically increase shopping cart size; deploy global digital payment networks; and deliver instant, one-to-one personalization for millions of customers. Aerospike customers include Airtel, Baidu, Banca d'Italia, Nielsen, PayPal, Snap, Verizon Media and Wayfair. The company is headquartered in Mountain View, Calif., with additional locations in London; Bengaluru, India; and Or Yehuda, Israel.

2525 E. Charleston Road, Suite 201
Mountain View, CA 94043
Tel: 408 462 2376
www.aerospike.com