Transforming Batch-based Data Processing Pipeline with the Power of Aerospike

Andrius Mažeiva
Head of Solutions Architecture
Adform

My story in Adform

Andrius Mažeiva

Head of Solutions Architecture in Adform

5 years ago joined Adform as Technical Product Owner in Data Processing team. With a goal to lead refactoring of existing processing flow to support constant growth of data flows and ensure better availability.

Adform story –
Seventeen years of driving innovation in advertising technology

Gustav Mellentin
CEO & Co-founder

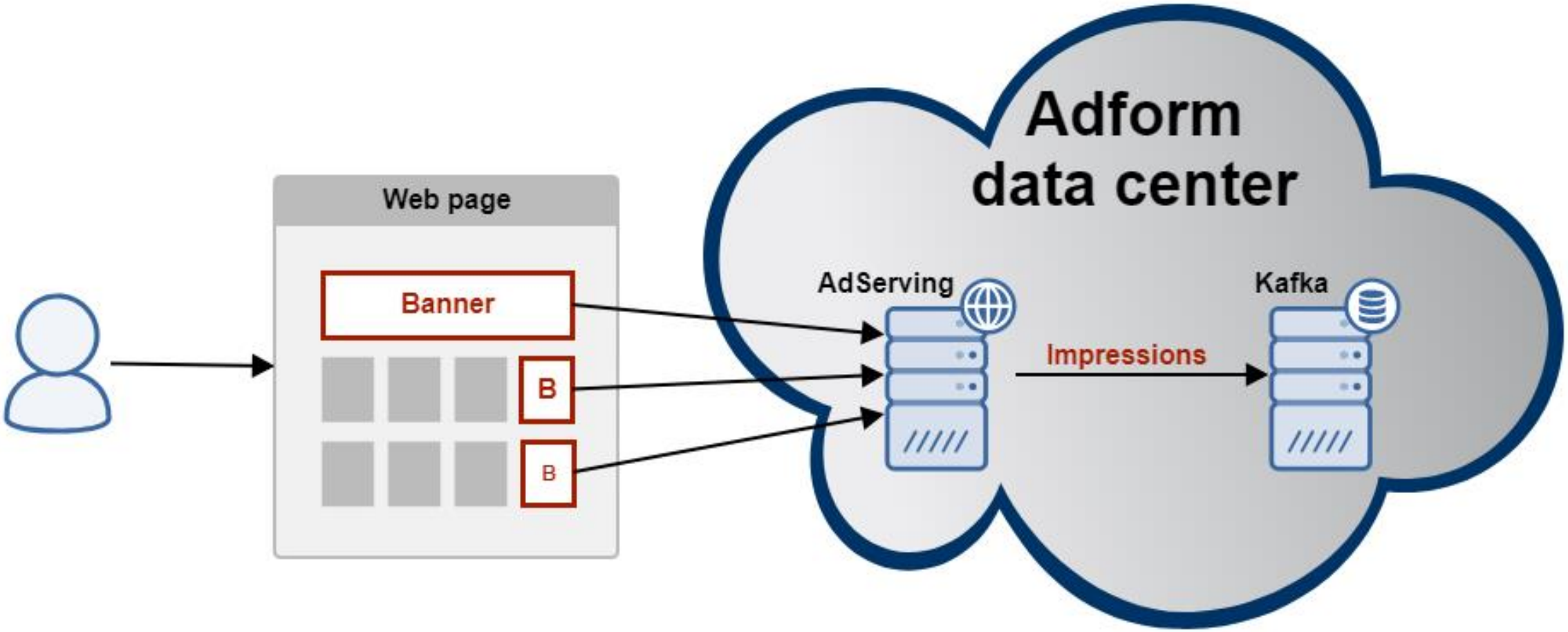
Jakob Bak
CTO & Co-founder

Stefan Juricic
CPO & Co-founder

Founded in Denmark in 2002 by 3 code-writing entrepreneurs, **Adform is** the industry's only enterprise ready **Integrated Advertising Platform** built from the ground up as one integrated advertising platform.
The company is still led by all three founders**.** Adform is one of the **leading advertising technology** companies in the world. We provide the **software** used by buyers and sellers to automate digital advertising. **850+** employees serve a high profile and loyal customer base globally.

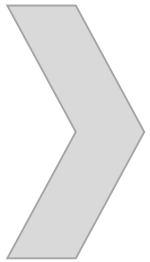
adform | AEROSPIKE SUMMIT '19

Ad Serving in Adform



So what to do with data?

Each impression comes from user identified by some ID (Cookie ID)



Historical information about each Cookie ID is needed to answer such questions

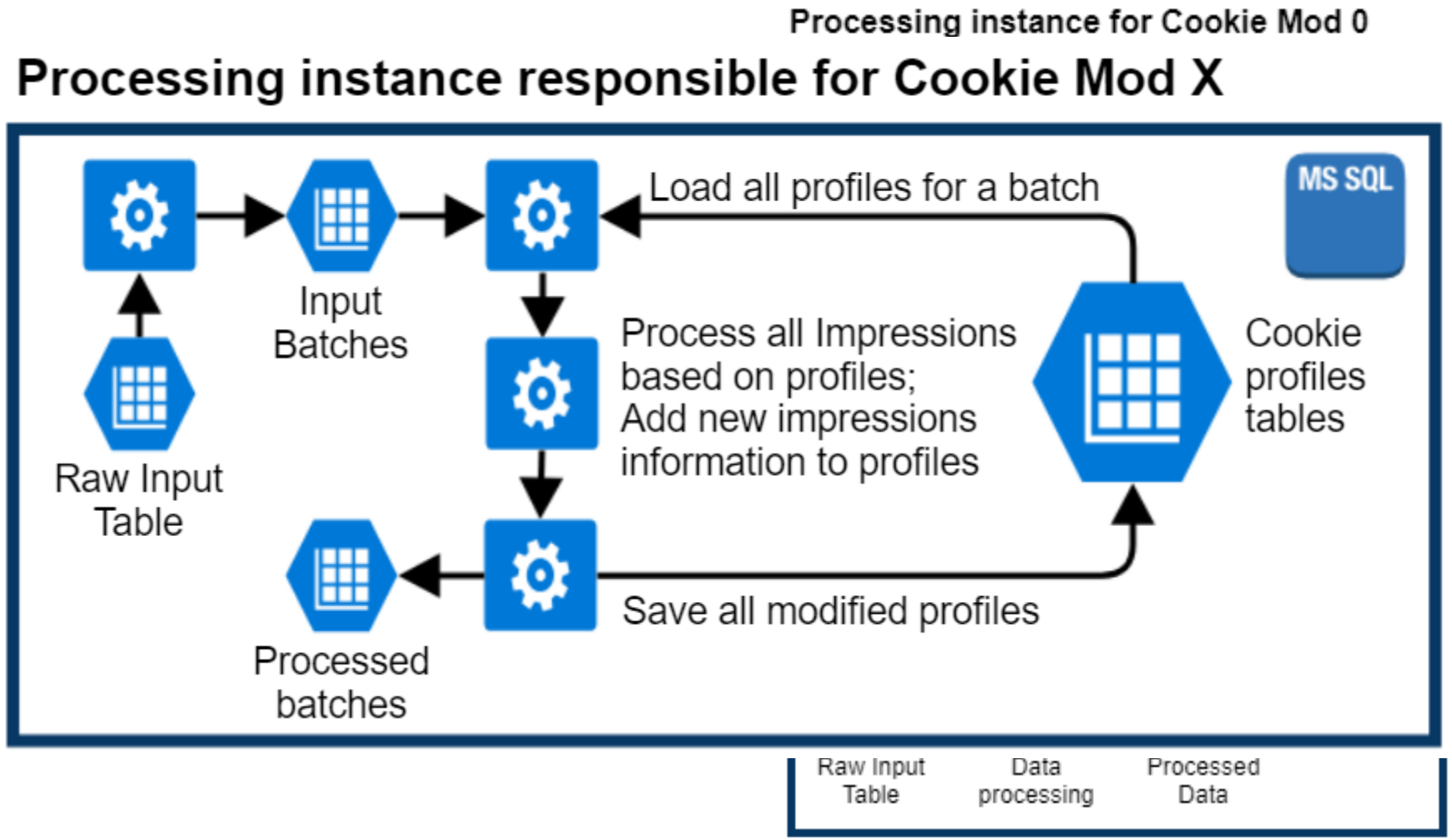
Some questions to answer:

- Was impression unique?
- Was impression unique today?
- Was impression unique per campaign?
- And etc.

How to store and access Cookie profiles?

adform | AEROSPIKE SUMMIT '19

Legacy processing data flow



adform | AEROSPIKE SUMMIT '19

How the profile were stored in MS SQL server

- **Definition of records:**
  - Cookie – record defined by **Cookie ID** to store general information about the cookie like first seen time and last seen time
  - Cookie Profile – record defined by **Cookie ID and Tracking Setup Id** to store individual information about cookies activity for each client. Record contains actual profile as binary field, first and last seen times
- **Structure of tables:**
  - One table for Cookies
  - Dynamic number of Cookie Profiles tables, were new tables are created in case amount of records in existing tables exceeds defined limit

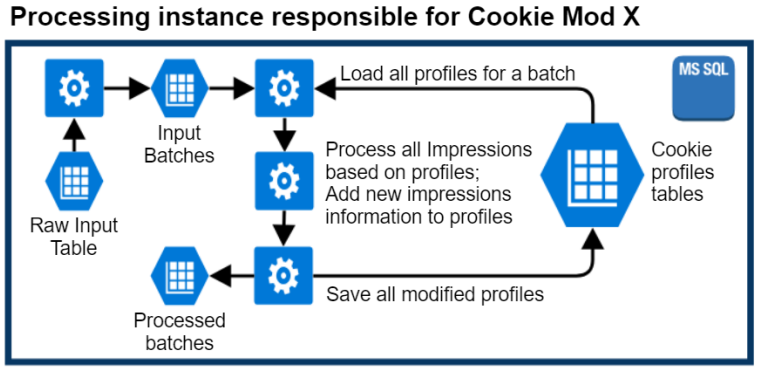
Custom cookie profiles expiration logic

- **Rules**
  - In case new cookie appears its stored for 7 days
  - If cookie is active for more than a day all cookie profiles are stored for 30 days
- **Process**
  - Once a day scan Cookies and all Cookie Profile Tables to delete records witch are not updated for 7/30 days based on the rules

Problems with legacy flow – MS SQL performance and Batch sizes

Principle of batch processing

- Scan profiles tables to find all needed profiles and delete the records (in transaction)
- Process the batch – modify the profiles
- Add modified profiles back to tables



Scanning by itself is costly operation so batches have to be big – each batch contained half an hours data to achieve best performance results

Big batches results in statistics delay, unequal usage of instance resources as if batch was processed faster server was "sleeping" and waiting until new batch will be formed.

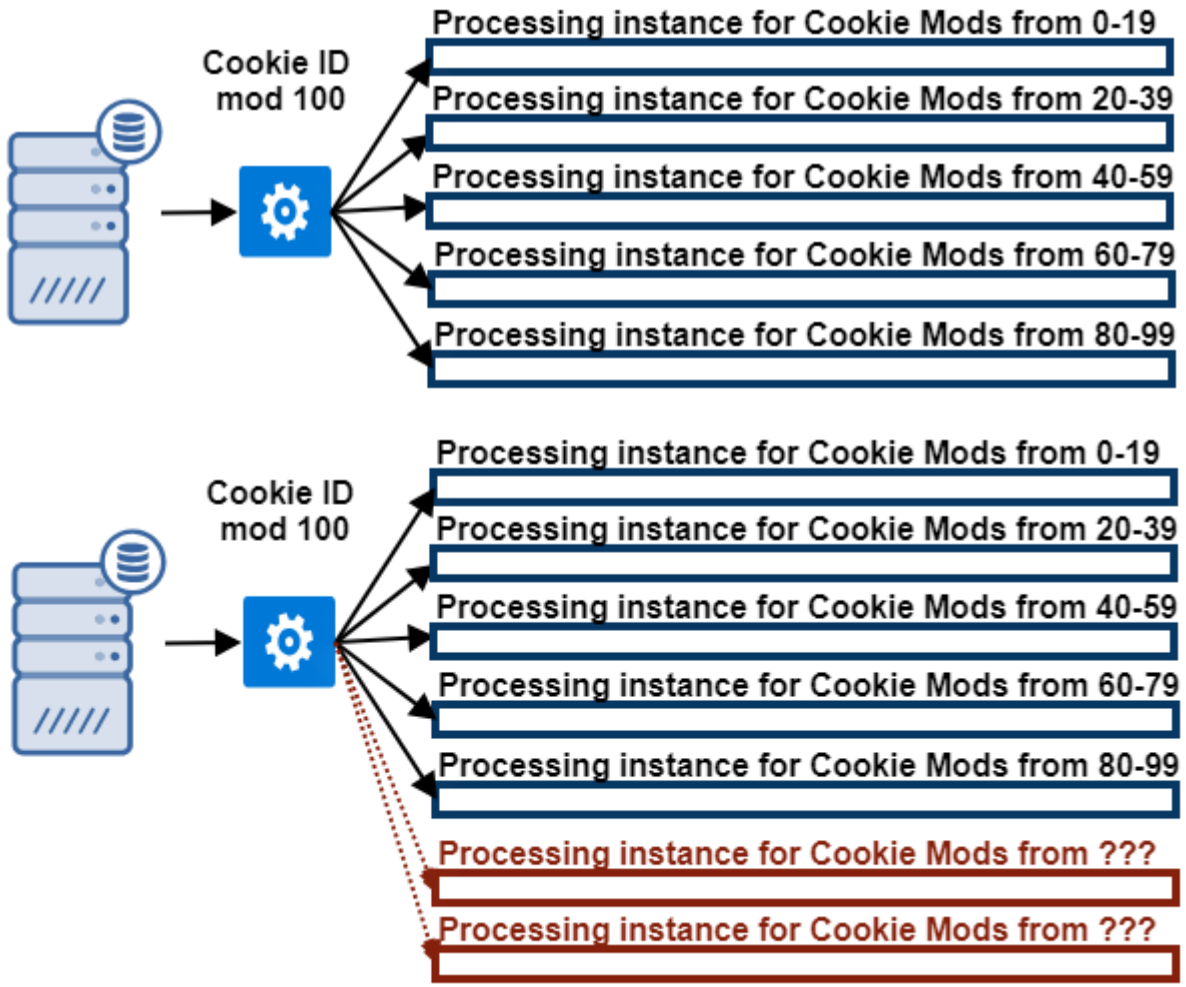
Maintenance windows were needed to perform various MS SQL maintenance tasks (as Index rebuilds) and to maintain custom profile expiration

Problems with legacy flow – Scaling

Initial setup – 5 servers, each taking care of 20 CMOD's. Load grows with time, amount of impressions, number of profiles increases. Need for scale arises.

So lets add 2 additional servers:

- **Full reshuffling of profiles between instances is needed**
- **All data witch is already sent to processing instances also needs to be reshuffled**
- **With growing amount of servers its hard to achieve equal load distribution between servers without changing mod function (impossible to evenly distribute 100 CMODs to 7 servers)**



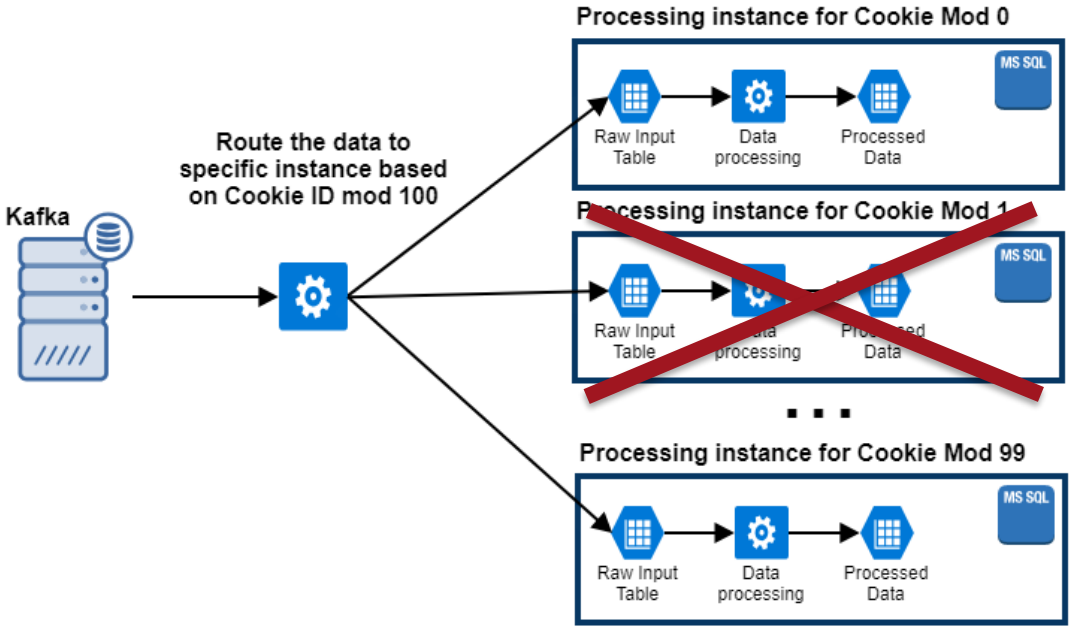
adform | AEROSPIKE SUMMIT '19

Problems with legacy flow – Availability and single point of failure

What if some instance goes down?

- All flow stops until server is restored ☹
- Its ok if it's a temporary issue or short term server maintenance is needed, but what if serious problems with server and it can't be restored?

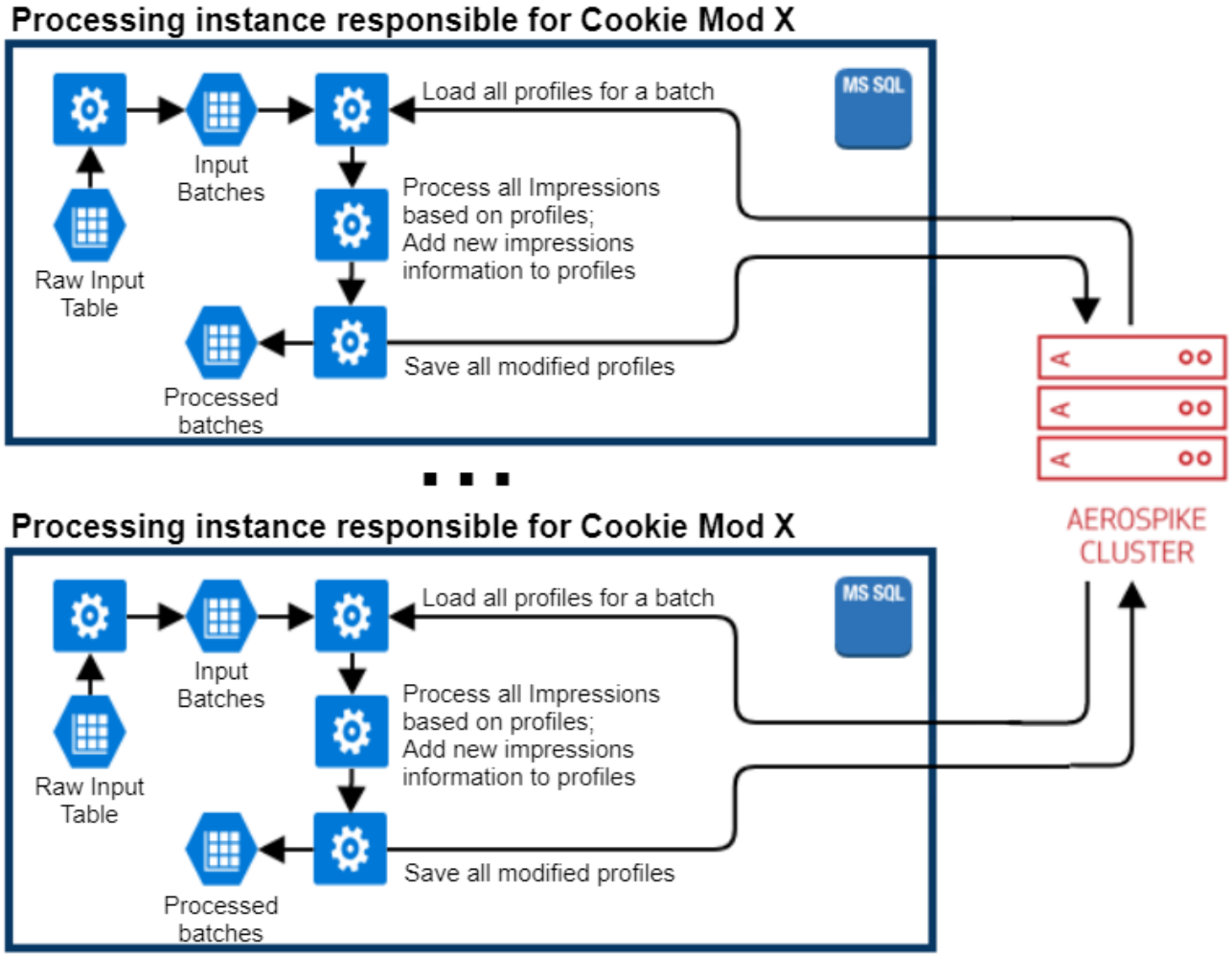
Some portion of profiles can be lost in worst case scenario ☹



adform | AEROSPIKE SUMMIT '19

Solution – instead of local storage in each instance lets use Aerospike cluster for storing Cookie profiles

Good news - minimal changes to old solution needed !!!



adform | AEROSPIKE SUMMIT '19

Why Aerospike?

- Adform already had experience in AE usage - only positive impressions from existing use cases
- Aerospike feature set meets the main needs for Profile storage:
  - Really fast access of the data by record key
  - Dealing with profile expiration comes out of the box
  - Has needed scalability and availability features

Main benefits of new approach

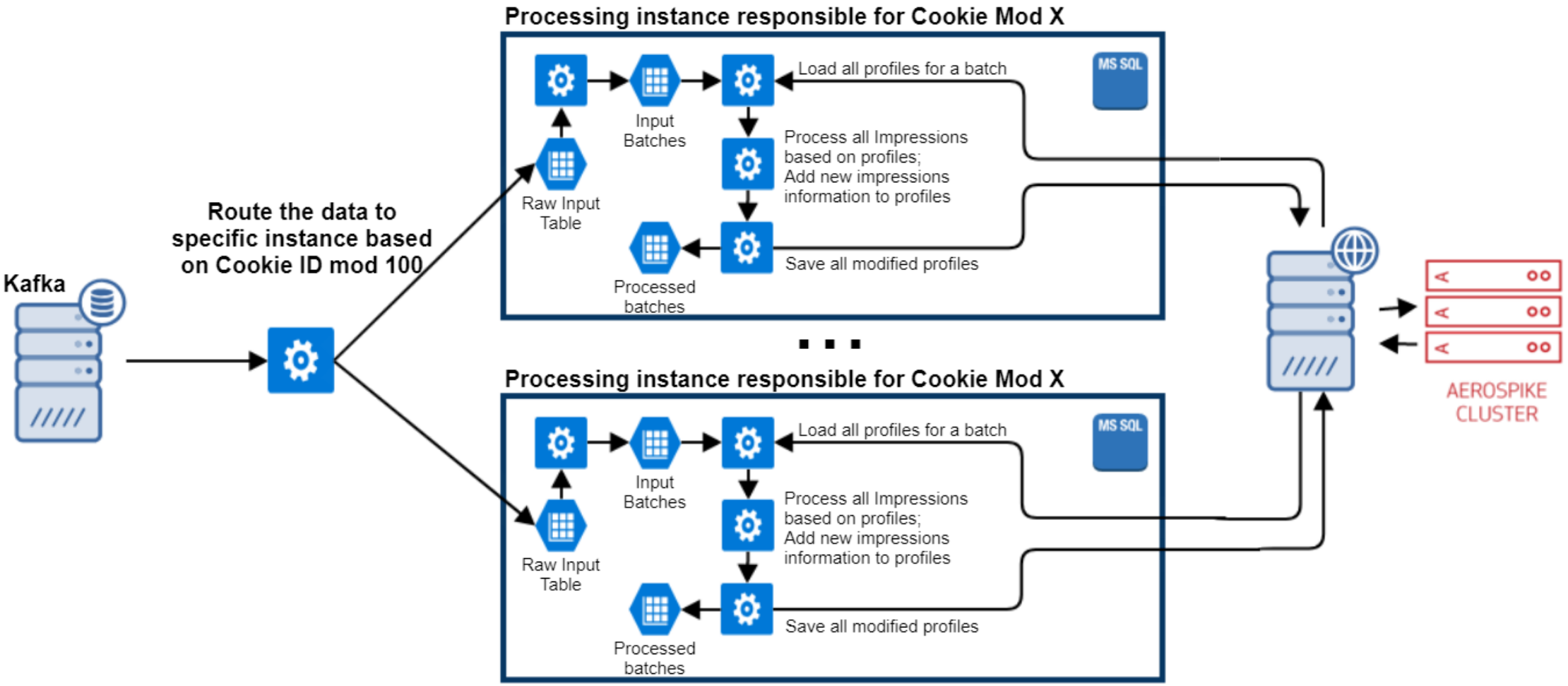
- Possibility to change storage without big refactoring of overall flow
- Changing approach from scanning to lookup by key should allow to reduce batch sizes drastically
- Scaling of storage and processing instances becomes independent processes
- No loss of the profiles data in case one server outage

Design of the solution for Cookie Profiles Storage

- **4 nodes, each with 7 TB of SSDs, 756 GB of RAM, AE version 3.5.15**
- **One namespace with replication factor 2, two sets:**
  - One – for Cookie Profiles, where Key is defined as *Cookie ID + ID of Tracking Setup* (individual profile for each client's context). Profiles data stored in binary type bin
  - Second – for Cookies, where Key is defined as *Cookies ID*, for storing Cookie's first seen date, last seen date and references to all existing Cookie Profiles. References to all existing Profiles were needed for calculating correct TTL's for similar rules we had in previous solution.

Decision was made to introduce additional REST API layer between processing instances and AE to isolate processing from actual storage technology

Redesigned processing data flow



adform | AEROSPIKE SUMMIT '19

Plan for data migration

- **Migration of data was implemented as a step in processing flow:**
  - Before starting to process a batch, collect all Cookie Profiles from local MS SQL database (delete data from MS SQL with transaction)
  - Save all the profiles to AE via REST API
  - Commit deletion transaction
  - Perform batch processing by getting and storing Cookie Profiles to AE via REST API

Migration

- **Migration started, amount of data in AE started to grow, size of local MS SQL databases started to reduce. After some time performance stabilized, even with additional migration steps.**
- **All migration was intended to finish in 30 days as longest time for storing Cookie profiles was 30 days.**
- **After ~2 weeks, suddenly all processing flow stopped… Aerospike stopped accepting new records.**
- **After two hours investigation we found out a limit of max 256 GB of RAM for primary index for one namespace was reached…**

Resolution

- **Complication was ability to keep the flow stopped – the maximum amount of time the flow can be stopped is 24 hours**
- **Rollback was not an option as putting data back to MS SQL server would take too much time, not event taking into account all the preparations needed**
- **The only option we found out was to extend AE cluster to keep memory used by primary index bellow 256 GBs**
- **Good news – we managed to find and prepare additional AE nodes in just few hours. As soon as first node was added to the cluster, AE started to accept new records and all processing flow resumed.**
- **We kept adding new nodes to ensure memory used is not exceeding 256 GBs. And ended with 12 nodes in a cluster at the end of migration**

So, what's next?

To enable efficient hardware utilization we decided to introduce sharding inside AE:

- To use **four** namespaces instead of **one** to bypass record amount limitations for namespace

- **In addition to existing AE cluster, new one with 4 nodes, each with 7 TB of SSDs, 756 GB of RAM, AE version 3.5.15 was created with modified REST API layer witch was performing data sharding to 4 namespaces based on Cookie ID**
- **Existing migration logic from MS SQL to AE was adjusted to perform migration from one Aerospike to another.**
- **After one additional month of migrations we ended with perfectly working solution on initially planned hardware**

Impact on performance

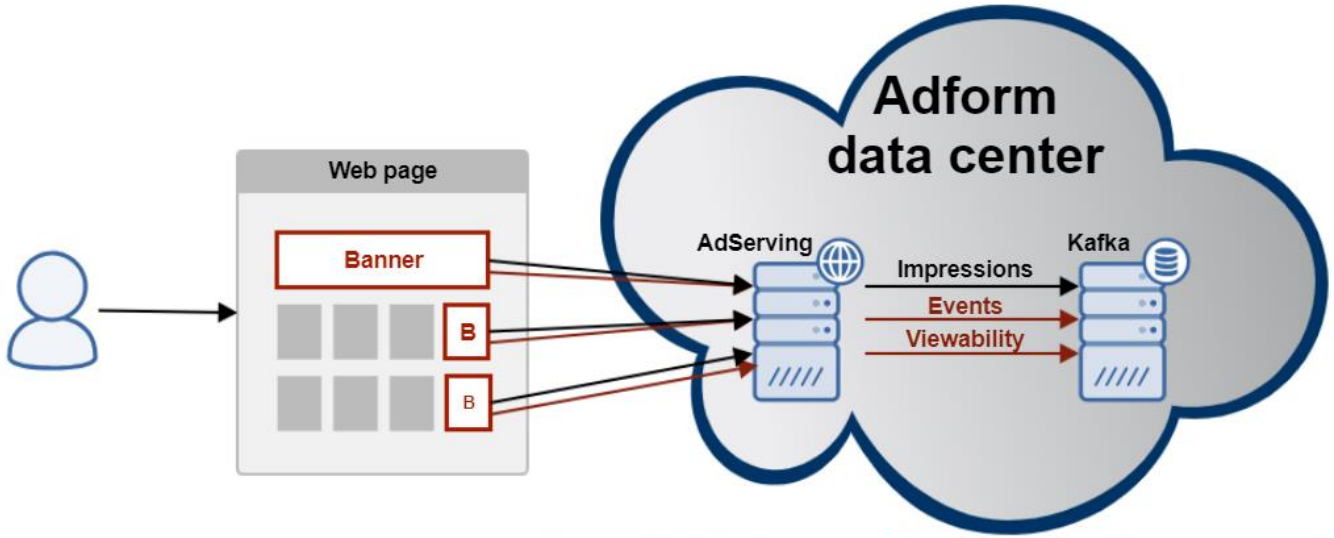
- **After migration was finished batch size was changed to 30 seconds instead of 30 minutes**
- **Removal of huge Cookies and Cookie profile tables from local database had significant impact on MS SQL Server performance. As a result maximum throughput of individual processing instance increased by ~3-4x.**

Lessons learned

Consult Aerospike support while planning your new Aerospike based solutions for correct cluster sizing!!!

Ad Serving in Adform

- **User visits web page**
- **Each placement in Web Page performs a request to Adform Ad Server**
- **Ad Server decides witch banner to show and:**
  - Returns needed banner scripts to web page
  - Logs **Impression** information to Kafka
- **Scripts renders a banner in a placement**



- **Scripts are continuously measuring if banner is In Screen and sends viewability information based on rules defined (up to 5 request to Ad Serving can happen)**
- **On different user activities requests are sent to Ad Serving to log events like Stop/Play video, mouse over, extend banner and etc.**

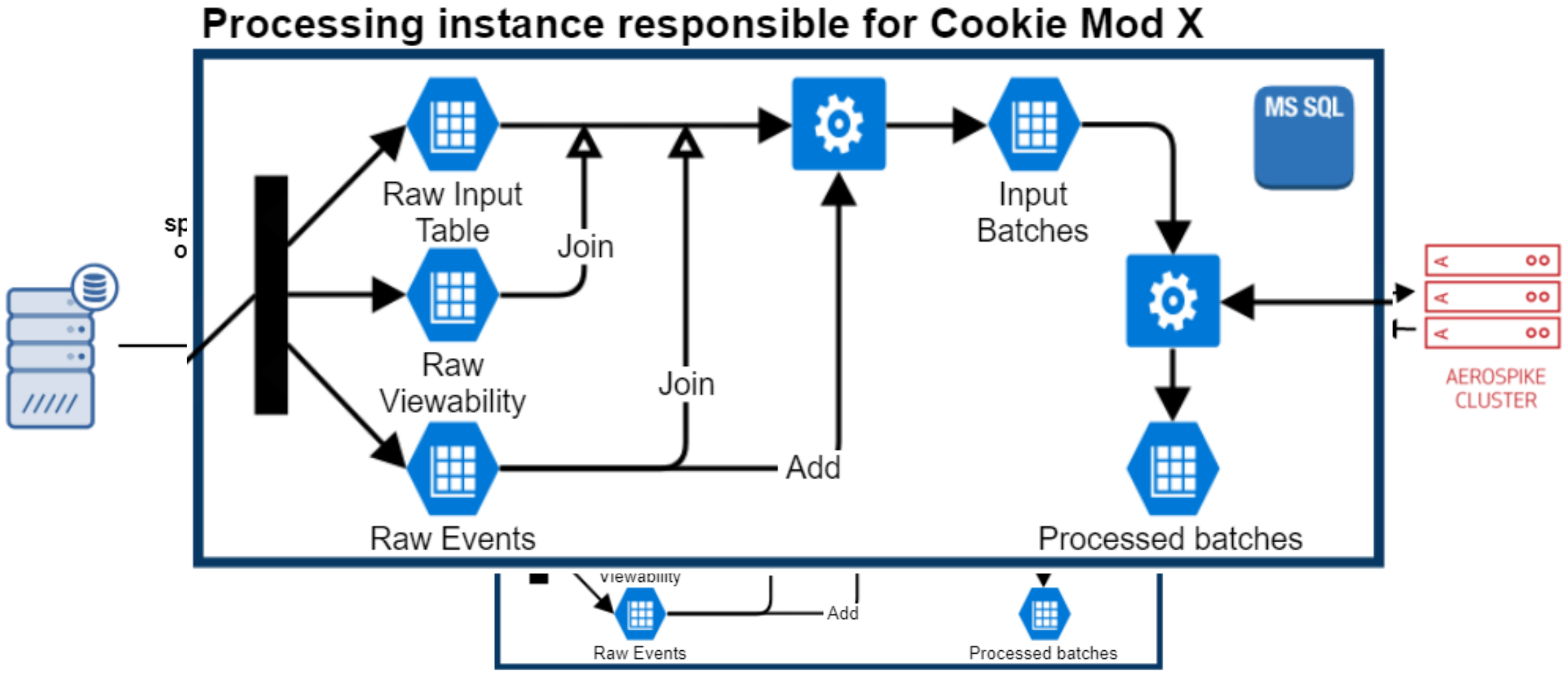
adform | AEROSPIKE SUMMIT '19

Additional impression enrichment needed

- **After processing each impression should have additional information:**
  - Was it viewable during next 10 minutes? For how long?
  - Did any engaging event happened after impression was served?

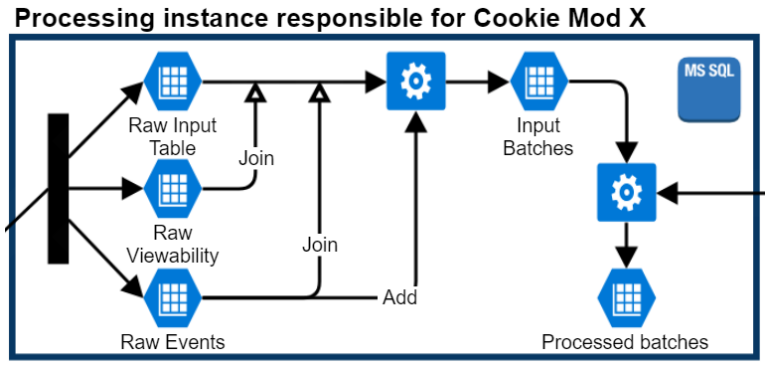
Different streams needs to be joined with 10 minutes window by applying custom join logic

Processing data flow



Problems with joining on SQL side

- **Joining for viewability requires only last viewability information to be taken – makes SQL query inefficient**
- **Raw events table is used for checking if impression had at least one engaging event and for moving events further for processing**



- **Separate processes for deleting data are needed – adding, joining data and deleting conflicts between each other and can't happen at same time**
- **Constant adding and deleting data makes native SQL indexes very inefficient – rebuild of indexes is needed at least once an hour**
- **If tables grows, everything slows down – hard to ensure stability of the process**

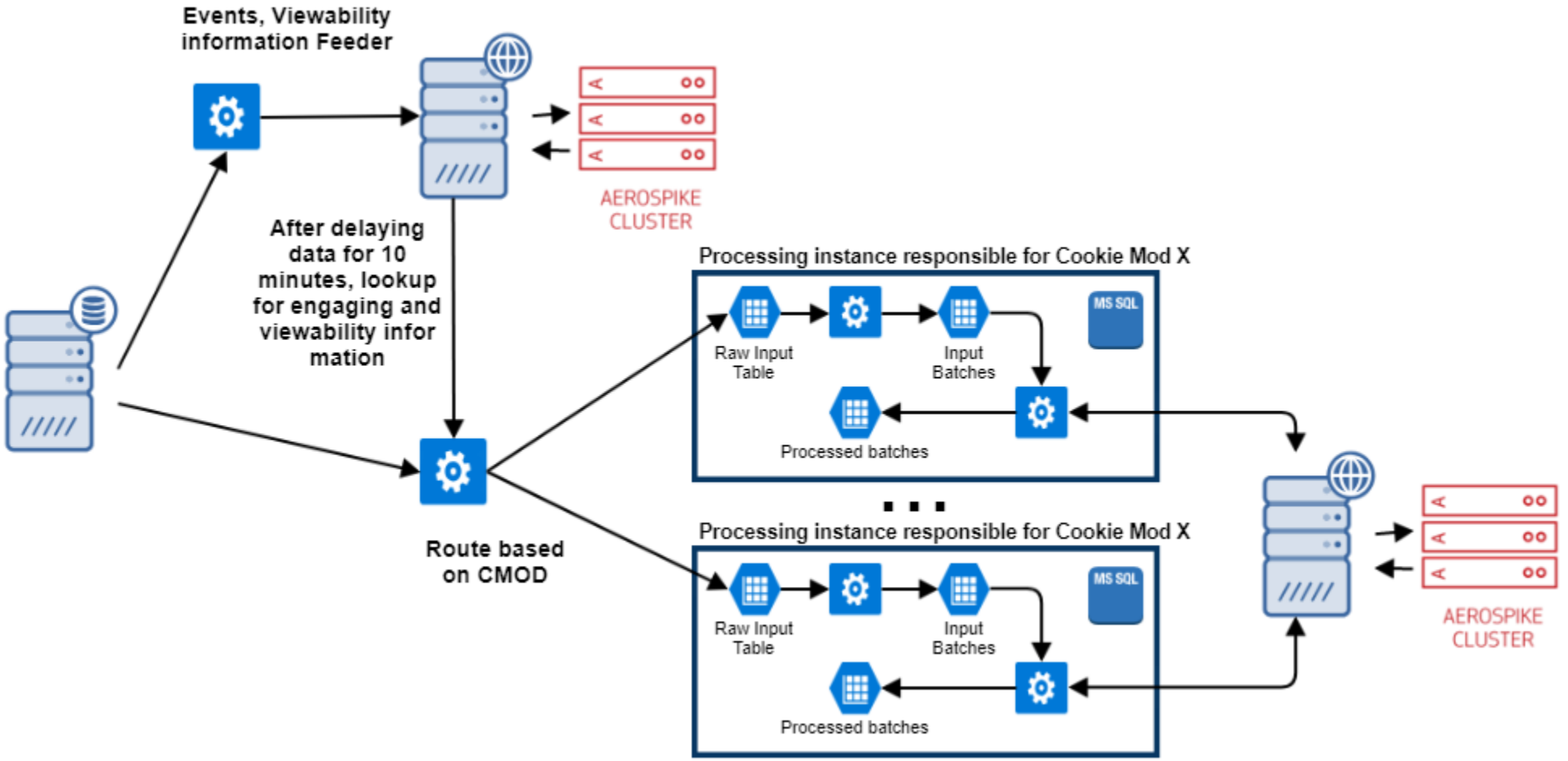
adform | AEROSPIKE SUMMIT '19

Possible solution

Goal: Instead of joining data inside individual processing instances to join the streams before routing

- **As far as joining window is quite big (10 minutes) we need some temporary storage to store viewability and events information – In Memory Aerospike cluster perfectly fits here**
- **Separate processes for writing Events and Viewability to AE needs to be introduced**
- **Current process responsible for routing data to individual processing instances needs to be extended to lookup for Event and Viewability information before routing**

Redesigned processing data flow



adform | AEROSPIKE SUMMIT '19

Design of the solution for stream joining

- **4 nodes, each with 512 GB of RAM for In Memory AE storage**
- **One In memory namespace, replication factor 2**
- **Flexible API layer**
  - enables to define new sets for separate use cases
  - supports different update strategies on records and bins level: always update, keep last by date, create if not exists
  - Enables to define TTL for each set (Use Case)
- **No limitation on record size as all data is kept In Memory**

Potential problems

- **How to know if lookup from router can be performed?**
  - Are all viewably and events information already in AE?
- **What if In Memory Aerospike cluster will loose more than 2 nodes at the same time?**
  - How to avoid possible data losses?

Synchronization between processes

- **Introduce Separate independent process (Data Delays collector) witch is:**
  - continuously checking the committed Kafka offsets
  - extracts actual time of last already processed record
  - Updates the time (data delay) in same In Memory Aerospike via API (Data Delays API)
- **As a result, router can check data delays of Events and Viewability Information Feeder via API**
- **In case more then 1 AE node goes down, some of Data Delays are also lost. Stream joining API was extended to continuously perform checks of potential data losses. In case of missing data, API stops responding to data requests to avoid data losses**

What's next?

- **Stream joining solution with small extensions is currently used for 5 different Use Cases including complex join for Sessions calculation**
- **Data delays API currently is used as a main source for stream processing service health monitoring**

Conclusion

- **Adding Aerospike to processing flow allowed to speed up all the flow – average delay for data processing reduced from 4 hours to 15 minutes (including 10 minutes delay required by business rules)**
- **Aerospike enabled scalability for most critical paths**
- **By utilizing Aerospike overall costs of solution reduced both from hardware perspective and operational perspective**


AEROSPIKE
SUMMIT '19
adform
Questions?