mixpanel

# MIXPANEL SYSTEM ARCHITECTURE

*Vijay Jayaram, Technical Lead Manager, Mixpanel Infrastructure*

The content herein is correct as of June 2018, and represents the status quo at the time it was written. Mixpanel's systems will evolve going forward, as we continually improve functionality for our customers.

## INTRODUCTION

Mixpanel is a user analytics platform that enables everyone to learn from and act on their data. It receives data from customers' websites and mobile apps all over the world and creates reports summarizing that data in an interactive user interface (UI). To deliver powerful, real-time user analytics at scale, we've spent years building our own database infrastructure, known as Arb. In this whitepaper, we dive into our system's architecture.

We built Arb in-house to fulfill our strict requirements:

- User-Centric: Arb is custom-built to answer complex, user-centric queries
- Speed: Most queries are answered in less than 1 second
- Real-time: Ingested data is available to query in under a minute
- Scalable: We ingest TBs daily, with individual queries often scanning 100s of GB. Our use of Google Cloud enables us to effortlessly scale to meet customers' needs

By the numbers:

- Mixpanel receives over 8 trillion new data points a year.
- Mixpanel's query engine processes 20,000 TB of data per day, over millions of queries.
- Mixpanel's storage system stores >1PB of compressed data.
- Historic uptime for both ingestion and queries is 99.99%.
- Mixpanel is SOC 2, ISO27001 Data Center, GDPR and EU Privacy Shield compliant and enables companies to maintain PCI DSS and HIPAA compliance.

Mixpanel's architecture can be divided into 3 major components.

- *Ingestion*: receives data from customer products and sends it to storage.
- *Storage*: provides a distributed, replicated, column-oriented data store.
- *Query*: accepts customer queries and returns aggregated, ready-to-visualize, data.

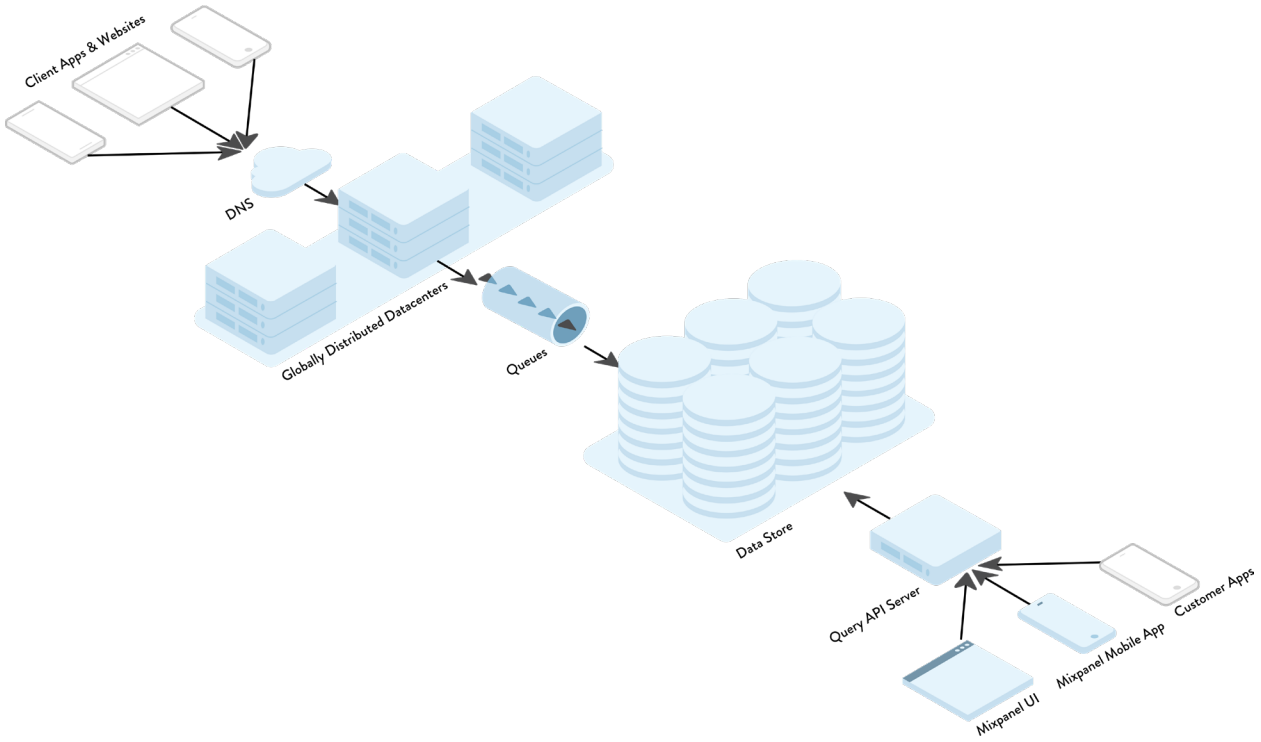


*Figure 1: Mixpanel architecture overview*

We describe each of these components in turn. There are other specialized components to handle messages, A/B testing, and our machine learning products, like Predict. We will describe those components in future whitepapers.

## USER-CENTRIC DATA MODEL

Mixpanel's data model is designed around understanding user behavior on our customers' products. A customer sends data points to their own Mixpanel project. Each data point contains information about an event or a user. Customers send an event whenever a user takes an action on their product. Customers send or update people records whenever they get new information about a user, such as at signup. Figure 2 shows a few data points for an example project.

Events

| TIMESTAMP | USERID | EVENT | OS | NAME | ITEM |
|---|---|---|---|---|---|
| 2016-12-15T08:02 | qrt-345 | Browse | iPhone | | raincoat |
| 2016-12-15T08:05 | qrt-345 | Signup | iPhone | Janet | |
| 2016-12-15T08:07 | pnm-321 | AddToCart | Android | | popcorn |
| 2016-12-15T08:10 | Janet | AddToCart | iPhone | | raincoat |
| 2016-12-15T08:10 | Vijay | AddToCart | iPhone | | chocolate |
| 2016-12-15T08:15 | Janet | Checkout | iPhone | | |

People

| USERID | EMAIL | COUNTRY | PHONE | FIRST_PURCHASE_DATE | NUMBER_OF_PURCHASES |
|---|---|---|---|---|---|
| Janet | janet@mixpanel.com | US | 650-555-1234 | 2016-12-15 | 1 |
| Vijay | vijay@dogsrus.nl | NL | | 2016-02-19 | 7 |
| Jessica | jw@gogogo.com | US | 914-555-5678 | 2015-07-29 | 12 |

Aliases

| NAMED_USERID (ALIAS) | ORIGINAL_USERID |
|---|---|
| Jessica | abc-123 |
| Jessica | azy-456 |
| Janet | qrt-345 |
| Vijay | xyz-789 |

*Figure 2: Events, people, and aliases*

## MIXPANEL SCHEMA

Mixpanel's schema consists of event and people "tables".

- **Event** records describe user actions. They contain:
  - *a timestamp*: when the user took that action.
  - *an identifier for the user*: who took the action.
  - *an event name*: what action was it.
  - *properties*: more information describing the event, such as the item name for a "purchase item" event.
- **People** records describe users. They contain:
  - *an identifier for the user*: who is it, also used for all events from that user.

***"I'm looking at 2.5m users and over 100M events, and in seconds I have the real-time answer to my question."***

Adam Dobrin, Senior Data Analyst, 2K

- *properties*: more information about the user. Some are information that a user typically provides once, when signing up or making their first purchase, such as their email or mailing address. Others are updatable information about a user, such as the cumulative number of purchases or their more recent purchase date.

The event and people tables themselves, however, have a "flexible" schema. All event records have a timestamp, user identifier, and event name. All people records also have a user identifier, which allows joining data from a people record with each event for that user. The rest of the schema is defined by the properties that are sent with each data point. It may be a different set of properties for each data point. In most cases, it is a fixed or highly overlapping set of properties for each event type (records with the same event name). However, there may be little overlap between event types.

Event records are immutable: they are sent once and do not change. The only exception to this is to meet GDPR requirements; if a user requests that their data be deleted, we delete all events associated with that user. People records are mutable: they contain the most recent information about a user and they can be updated anytime.

### USER-CENTRIC PRODUCTS

Mixpanel features analyze user behavior: what are users most likely to do and not do when they are using a product? The user-centric data model enables many of these features to be simple for customers to use and efficient for us to implement.

- *Insights:* The insights report provides a flexible slice-and-dice of all event and people data, by any property. Having a user identifier in both event and people records allows implicit joins between them; in the report UI, people properties are displayed and used as if they were properties of each event. Furthermore, the user identifier allows computing the common metric daily active users (DAU) by counting the number of unique user identifiers.
- *Funnels*: The funnel report determines conversion rates for users through a custom sequence of events. In an E-Commerce example, we may explore a funnel between the "Viewed Item", "Add to Cart" and "Purchased Item" events, to see which fraction of users make it through each step of this process. Like insights, funnels supports filters and segmentation on all event and user properties. Funnels need not be created ahead of time; they are fully retroactive.

- *Retention*: The retention report analyzes sequences of user events to determine the stickiness of the product amongst users. The report groups users by the time of their first event and aggregates the number of users who return to do another event within specific time windows. Retention can be filtered and segmented on any property and is fully retroactive.
- *Messages*: Messages are email or mobile app messages that are sent to a set of users. While this set may include all users, more commonly, the set is identified by either their people record properties (e.g., lives in "CA" or has device "iphone6") or by their recent events (e.g. purchased a book last week). Once again, the user identifier links the events to the people record, which contains the email address.

Note that the user identifier is crucial for all of the above features, as it enables Mixpanel to group events by the user who performed them.

### USER IDENTITY MANAGEMENT

User identity management ensures that all event and people records created for a single user can be identified as belonging to the same user. In most cases, the user identifier is sufficient to identify all events by a user. Correct identification of each event's user is a crucial part of user-centric data analysis. However, anonymous users and people who use multiple devices create complications. Most notably, they can lead to multiple user identifiers for the same person.

- *Anonymous users*: a single user may have more than one identifier if, for example, they perform some actions anonymously before signing in and then some actions after signing in. Before signing in, each user will receive a random hash value for their user identifier. After they have signed in, the customer may choose a different user identifier, such as an account number or email address. For example, in Figure 2, userid "qrt-123" is later identified as "Janet." We need to recognize that the event at 8:02 by "qrt-123" and the event at 8:10 by "Janet" belong to the same user.
- *Multiple devices*: a user who is first identified as "qrt-123" and later "Janet" on her laptop may also have been identified as "sdr-234" (and later "Janet") on her mobile phone. We now need to recognize that "qrt-123", "sdr-234", and "Janet" are all ways to identify the same user.

The different user identifiers for the same user must be consolidated to compute Mixpanel's features correctly. For example, to compute DAU, each person must have only one user identifier. If a person has multiple user identifiers, then DAU will be inflated. To compute

a funnel, it is necessary to follow a user's event flow for its entire duration, even if, for example, a purchase begins with browsing on a mobile phone and ends with a credit card payment on a laptop.

Mixpanel solves the user identity management problem with user aliases. A user alias is a user identifier that is sent by customers in event and people records, such as "Janet", but is replaced by a different user identifier during ingestion and storage. In the above example, both "Janet" and "sdr-234" become aliases for "qrt-123". All event and people records for the same person therefore contain the same user identifier in the Mixpanel data store.

## INGESTION

Mixpanel receives over 8 trillion new data points each year from our customers. Let's follow the lifecycle of an event into our system:

1. The user's device sends the event to https://api.mixpanel.com/track
2. This request is routed to our globally distributed edge machines by our DNS provider
3. The event is validated and pushed onto queues at the edge
4. An ETL processor running in a central Google Cloud region pulls events off these queues, performs data validation, resolves aliases, and routes the records to the storage system
5. Our storage system ingests the event and quickly persists it to a file, replicated and encrypted by Google's Persistent Disk infrastructure -- at this point it is available to query
6. After some time, a batch of events gets indexed into a more query-efficient, column-oriented format and uploaded to Google Cloud Storage

99% of records take less than a minute from reaching our edge machines to being query-able; this is what makes Mixpanel real-time.

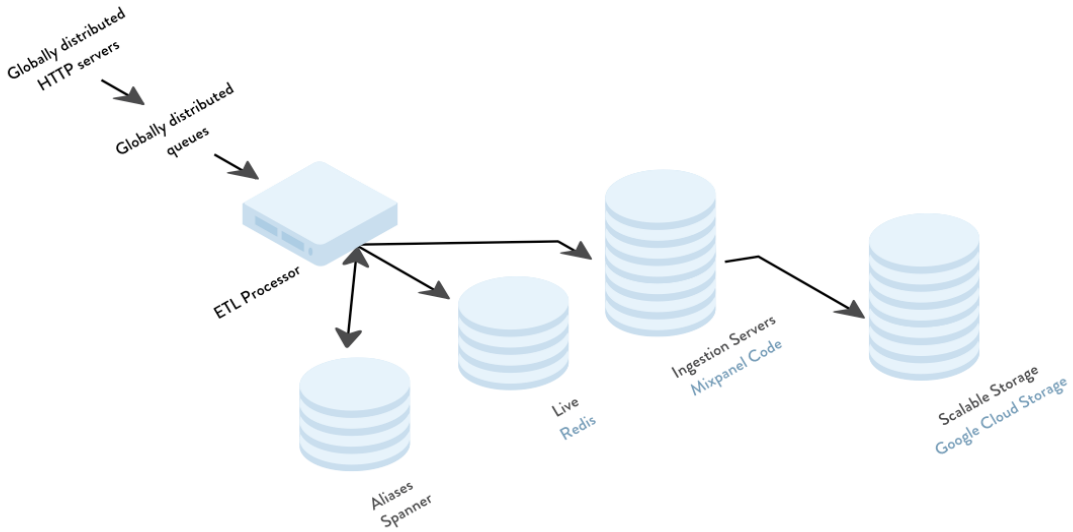


*Figure 3: All records pass from http servers to an alias resolver to queues for storage.*

The ETL processor also sends each record to a key-value store for Live reports. Live reports provide customers with real-time views of records for faster debugging and understanding of the data.

## STORAGE

Mixpanel's data store is a distributed, user-centric event database, built for interactive-speed analytics on billions of events.

### COLUMN STORE

Mixpanel's storage was originally row-oriented: there was one json blob to describe each event or people record. However, column-oriented data layout has become the standard for analytics data stores. Analytics queries compute aggregations over many records, but usually only need a few columns of each record. Column-oriented data stores (column stores) allow queries to access only the columns needed for a query and they allow much faster scanning of the data in a single column, e.g., to filter on column values.

In 2015, we wrote a new column-oriented data store for Mixpanel data. We wrote our own for two main reasons: functionality and performance.

**Functionality:** Mixpanel data has a flexible schema. We needed to handle a set of properties that is not declared and may change with every new record. Most data stores are very strict about schema modifications.

**Performance:** Mixpanel data and queries are user-centric. By controlling the data layout and data distribution, we can optimize for this data and these queries. Knowing the data layout results in lower query planning overhead, fewer stages in query plans, and less data sent over the network, which in turn leads to faster queries.

- The user id is the sharding key for distributing data across storage servers. Therefore, all events for a single user are on the same storage server.
- Query processing can take advantage of having all of the data for each user on the same server. For example, DAU computations can count users on each server independently. Similarly, funnel computations can process the entire event history for a user on the one server where those events are stored.

The column store compresses data 10x and allows queries to access only the columns they need, resulting in faster and more efficient queries. Query latency improved by a factor of nearly 5 when we switched to a column store and query server CPU utilization dropped by a factor of 10. We therefore need much less hardware to process the same query load. We also had enough query capacity to add many new features that run queries, such as Notifications and Automatic Segmentation.

### COLUMN STORE DATA LAYOUT

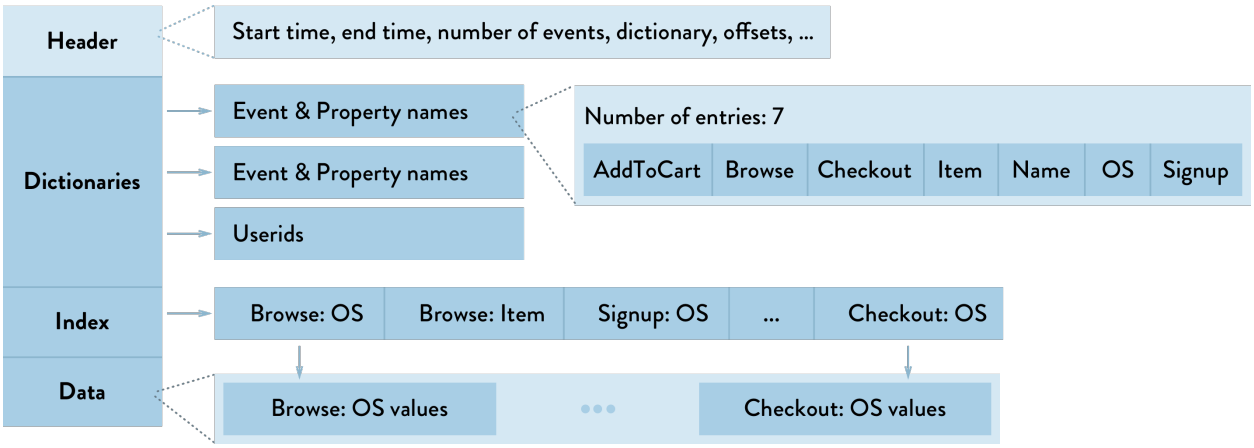


*Figure 4: Column store layout of the data from Figure 2*

We now describe the data layout of our column store.

The column store creates one file for each customer project for each day. Each file has four sections, as illustrated in Figure 4:

- *Header*: start and end timestamps, total number of events, locations of the remaining sections, and other metadata.
- *Dictionaries*: 3 separate dictionaries for property names, property values, and userids.
- *Index*: location of each property for each event, sorted by event name then property name.
- *Data (Columns)*: the number of values plus the actual values for all properties of all events.

Each dictionary is compressed with the LZ4 compression algorithm and the smaller version (compressed or uncompressed) is stored in the file. Each data section stores the values for one property, sorted by the event timestamp. The values themselves are stored either as the absolute (integer, string, float, boolean, or list) values or as indexes into the dictionary, whichever compresses better for the property. In both cases, run-length encoding of repeated values and byte-packing of integers and indexes are applied to save space.

### DATA SHARDING

Data is divided into shards (disjoint sets of records). The sharding key is the user identifier, which ensures that all records for a user are on the same storage server. We maintain multiple replicas of the data across two geographic zones to support failover. For additional reliability, periodic snapshots of all data are backed up to Google Cloud Storage (GCS). Historical data is also persisted in GCS for cost-effective, large-scale storage.

During the ingestion process, storage servers in each zone consume events for the shards housed on that server, writing them to row-based files to make them immediately available to queries. In the background, an indexing process converts these row-based files into our column storage format. This both results in greater compression and better query performance. Queries can read data from both the row-based and column-based storage formats.

### SCALABILITY AND RELIABILITY

Hundreds of shards provide high scalability for both ingesting and querying data. Our system is equipped to reshard project data as they grow, so that we can scale horizontally with our customers.

Multiple replicas of each shard mean that data is available as long as any replica is up. Mixpanel can therefore handle most machine upgrades and failures without data loss or unavailability. Historic uptime for both ingestion and queries is 99.98%.

## QUERY

### QUERY INTERFACES

Mixpanel customers have multiple options for querying their data. They can query the Mixpanel UI in a web browser, the Mixpanel mobile app on a phone, or the Mixpanel Query API server from any computer.

Customers create and view data reports in the Mixpanel UI. Each report sends one or more queries to the Mixpanel Query API server to get data and then displays a custom visualization of the result. Many powerful report types are built into the UI, including the segmentation, funnel, cohorts, and retention reports we discussed above. Customers can also create and view their metrics in the Mixpanel mobile app.

Finally, customers can send queries directly to the Query API server. Some customers write queries in scripts and send them periodically to track their key metrics. Other customers embed Mixpanel queries in their applications, to surface data from Mixpanel to their users. Both use cases are supported along with queries from the UI.

### QUERY PROCESSING

Mixpanel's query stack is built in-house for high-performance, real-time, user-centric analytics queries. Queries are received by the Distributed Query Server (DQS), which validates and plans how to distribute the query. The query is distributed to many Local Query Servers (LQS) to compute in parallel. The query is run both on historical data and on live ingested data, to provide real-time insights. Each LQS then streams the results back to the DQS, which performs the aggregation and returns a final result back to the client.

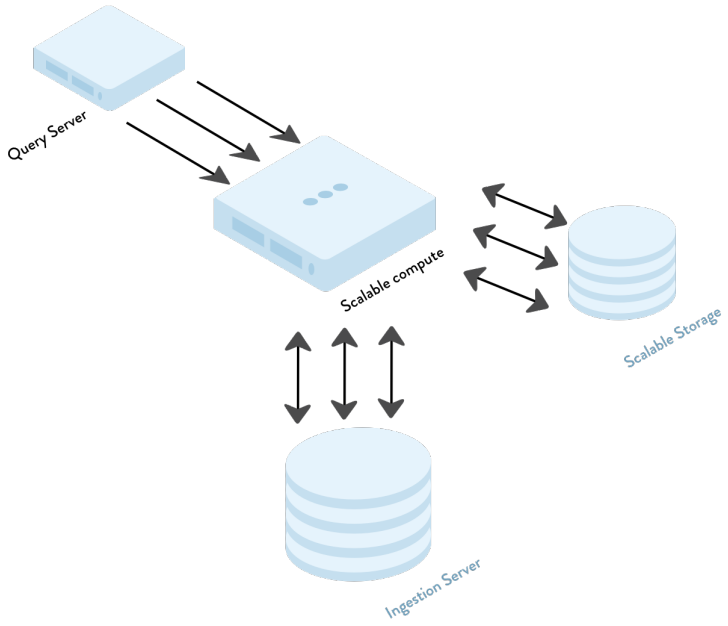


*Figure 5: Query execution*

In 2017, we re-architected this query engine to be cloud-native on Google's Cloud Platform. As part of this migration, we de-coupled our storage and compute layers, so that we could scale the two independently. As our customers track greater volumes of data, our storage layer scales seamlessly to reliably store it. As our customers make larger numbers of complex queries, we use the thousands of CPU cores available to process terabytes of data in seconds. Our massively parallel processing (MPP) architecture, coupled with our user analytics specific optimizations, give us the unique advantage of speed and scale.

Our infrastructure also forms the foundation for our more advanced features, like Messages and Automatic Insights. Messages are made possible by our real-time capabilities, so that customers can take timely, automated action on their data. Our advanced machine learning features like Anomaly Detection and Automatic Segmentation work well because our models are trained daily on the trillions of data points we've collected, which would be impossible without the scalable infrastructure that underlies Mixpanel.

## SECURITY

### SECURITY IN THE MIXPANEL TECHNOLOGY STACK

Security is an integral part of Mixpanel's technology stack. This manifests in a number of ways:

**Logical data isolation:** All customer data is tagged with information about the project it is associated with, and this information is validated at every layer of the architecture to ensure that requests for data are authorized and only return data for that project.

**Data protection:** Our production environment is firewalled from the public Internet in restricted-access networks, and all customer data is encrypted at rest.

**Auditing:** All access to customer data and production systems is restricted to individuals with a documented business need, and access to customer data is reviewed on a regular basis for signs of abuse.

For more details on Mixpanel's security and privacy, see the Mixpanel Security and Architecture whitepaper.

## SUMMARY

Mixpanel provides a fast, scalable, reliable, and secure product analytics platform. The following chart summarizes how Mixpanel's architecture design provides this speed, scale, reliability, and security.

This architecture supports the Mixpanel product analytics platform for nearly 20,000 customers, including BMW, DocuSign, US Bank, and Microsoft. We know that they rely on us to help them answer their most important user and product questions, so we take the performance, reliability, and security of our system very seriously. Nearly a third of our engineers build the ingestion, storage, and query systems we described in this white paper and everyday, we deliver improvements in both functionality and performance.

mixpanel

| | INGESTION | STORAGE | QUERY |
|---|---|---|---|
| Speed | Separate queues for high-volume and low-volume customers.<br>*99% of data points are ingested and stored under a minute.* | Custom distributed and column-oriented data store, built to scale on Google Cloud Platform | Distributed processing. Only reads columns needed. Local data caching. Work divided among shard replicas.<br>*< 1 second latency for most queries* |
| Scale | Many http servers and queues. Scale horizontally with demand by adding new servers.<br>*> 8 Trillion new data points a year* | Many data shards. | Local processing of each data shard.<br>*Up to 250 Billion events per query* |
| Reliability | Globally distributed ingestion centers. Remote copies for fault tolerance.<br>*99.98% historic uptime* | At least 2 replicas of each data shard. | Query processing can use any shard replica.<br>*99.98% historic uptime* |
| Security | Up to TLS 1.2 with 256-bit AES encryption.* | 256-bit data encryption<br>*SOC 2 and ISO 27001 certified Data Centers; GDPR and EU Privacy Shield compliant*<br>*Depending on client support | Up to TLS 1.2 with 256-bit AES encryption.* |

*Figure 6: Mixpanel's architecture is designed for speed, scale, reliability, and security*