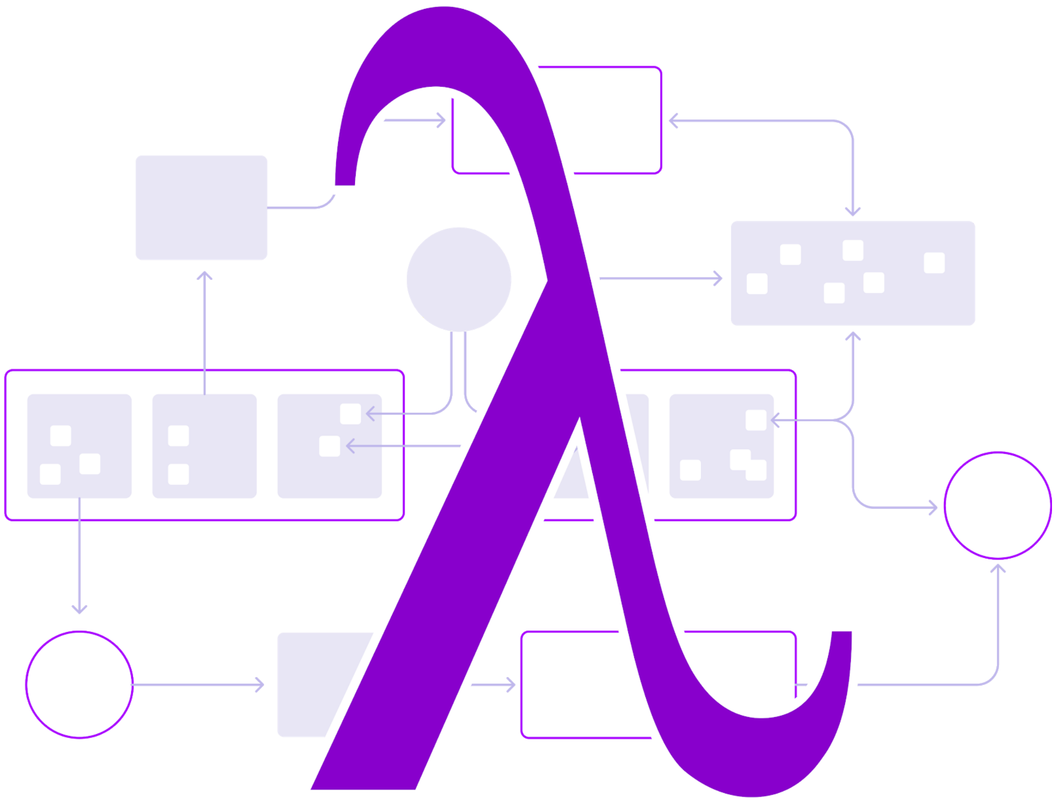
SOLUTION GUIDE

Lambda Architecture: A Modern, Simplified Approach with MemSQL

Nithin Krishna Reghunathan, Technical Evangelist



February 2020

Table of Contents

1. Introduction to Lambda Architecture **3**

2. Overview of Lambda **4**

3. Benefits of Lambda **6**

4. Workflow of Lambda Architecture **7**

5. Lambda for Big Data Systems **8**

6. MemSQL: A Complete Solution for Lambda **9**

6.1 MemSQL Overview 9

6.2 Why MemSQL is a Great Solution for Lambda 10

6.2.1 Highly Scalable 11

6.2.2 High-Performance for Both OLTP and OLAP Workloads 11

6.2.3 Multi-Model, Multi-Mode 12

6.2.4 Data Ingestion 13

7. Customer Success Stories **13**

7.1 Real-Time Analytics at Comcast 14

7.2 How Tapjoy Powers the Mobile Ad Platform 15

8. Conclusion **16**

9. Future of Lambda: Kappa Architecture **17**

1. Introduction to Lambda Architecture

Nathan Marz introduced the term Lambda architecture (LA) in 2011. He described a scalable and fault-tolerant data processing architecture, based on his experience in handling distributed data processing systems at Backtype and Twitter. The principle of this architecture is based on lambda calculus, a universal model of computation, so it is named Lambda architecture.

Lambda architecture aims to satisfy the needs for a robust system that is fault-tolerant, being able to serve a wide range of workloads and use cases, in which low-latency - that is, fast - reads and updates are required. The resulting system scales out horizontally as well as scaling up vertically in a linear fashion.

LA is a generic pattern that is capable of delivering common requirements for most modern big data applications. This pattern has the flexibility to handle both real-time (transactional –– OnLine Transaction Processing (OLTP)) workloads and historical (analytical –– OnLine Analytical Processing (OLAP)) workloads simultaneously.

Lambda architecture is a way of processing humongous amount of data (i.e. “Big Data”) by providing a platform to concurrently access batch-processing and real-time streaming methods with a hybrid approach. Like the physical aspect of the greek letter(ƛ), the Lambda architecture forks into two paths: one is a streaming path, the other a batch path.

The LA pattern was built around a few key principles:

- **Linear scalability**: Capability to scale out as well as up, so as to cater to more different kinds of use cases

- **Fault-tolerant**: Safeguard the system from software and hardware failures, as well as human errors
- **Extensibility**: Easy to manage and easy to extend, adding new data elements and new features.

Leading internet-scale companies, like Pinterest, Zynga, Akamai, and Comcast, are using a memory-optimized database to achieve the streaming, high-speed data component of the Lambda Architecture.

Various types of NoSQL distributed, in-memory datastores have been used over the last 15 years in an attempt to provide the low latency reads and writes of the speed layer but at the cost of losing relational join functionality, durability, immediate consistency, and resilience in the form of built-in backup and recovery capabilities.

Also, for batch processing, many companies started with Hadoop using map-reduce and have moved to to try Apache Spark and cloud-based object stores like S3 and Ceph, yet face latency and application complexity issues with those choices.

2. Overview of Lambda

The main layers constituting the Lambda architecture are the batch layer, the serving layer, and the speed layer. The following diagram represents the overview of the LA pattern.

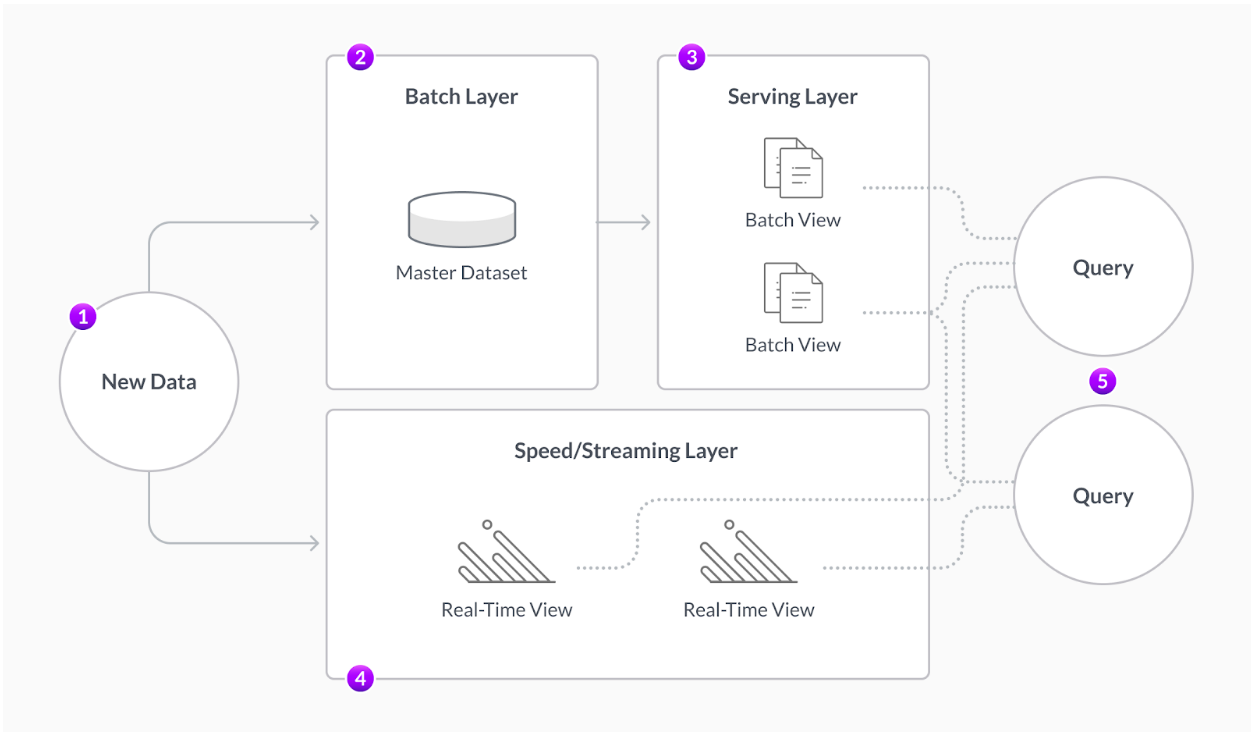


Figure 1. Overview of Lambda architecture

New Data

New data is continuously fed into the data system. The data gets fed to the speed layer and batch layer simultaneously.

Batch Layer

The batch layer is responsible for two major functions: (i) managing the master dataset, and (ii) pre-computing the batch views.

Serving Layer

The serving layer receives data from the batch layer, and is primarily responsible for indexing the batch views so that they can be queried in a low-latency, ad hoc way. The batch views from the batch layer and real-time views from the speed layer gets forwarded to the serving layer.

Speed Layer (Stream Layer)

The speed layer handles high frequency updates to the serving layer. This layer covers the data that is not being delivered in the batch view due to its latency. It only deals with the most recent data, so as to provide a complete view of the data to the user with real-time views.

3. Benefits of Lambda

Some of the key benefits of Lambda architecture as follows:

- **Zero hassle for server management:** You do not have to install, maintain, or administer any software
- **Automated scalability:** You can scale on demand based on the capacity requirement
- **Automated high availability:** Refers to the fact that serverless applications have built-in availability and fault tolerance capability
- **Real-time insights:** React to business scenarios with your best judgement by leveraging real-time data

4. Workflow of Lambda Architecture

The following figure shows the complete workflow of Lambda architecture:

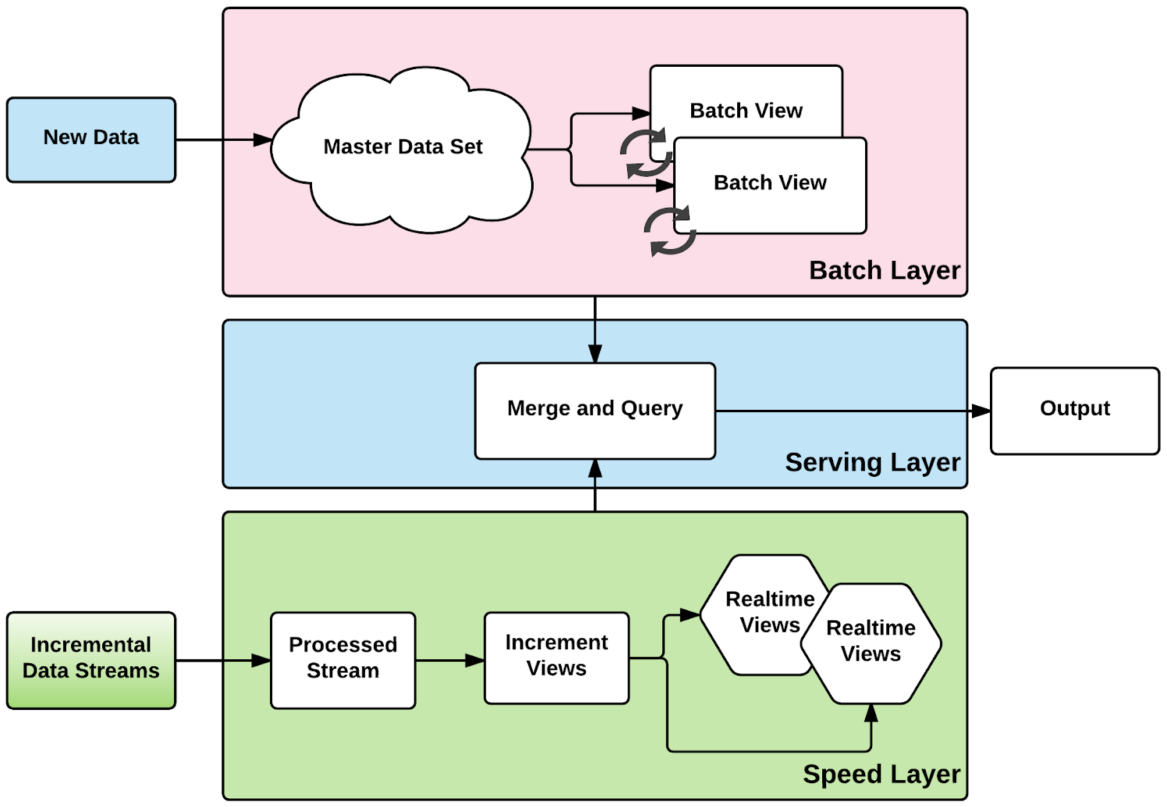


Figure 2.Workflow of Lambda architecture

As per the workflow, the master data is always managed in the batch layer. As new data arrives, it is fed to both, the speed layer and the batch layer. As they approach the batch layer, batch views are generated at regular intervals and recomputed from scratch each time. Similarly, the speed (or stream) layer that handles the most recent data generates the real-time (or stream) views, using the new data that arrives at the speed layer. When queried, the serving layer comes in action by merging both the speed and batch views together to generate appropriate results.

5. Lambda for Big Data Systems

Big data systems should be capable of handling workloads having a good mix of structured, semi-structured, and unstructured data. The ability to process highly concurrent batch and real-time data has been evolving as a common requirement for the modern data-driven organizations. The Lambda architecture pattern is capable of handling batch and real-time data, making it a great fit for hybrid big data systems. LA also has the flexibility to plug in or plug out various data generation sources based on demand.

Modern organizations have reacted to the data explosion scenario by adapting new strategies in data management. However, managing streaming data has been considered to be one of the most challenging tasks in recent times.

Most enterprises rely on a combination of streaming and historical data, which require sophisticated big data tools and solutions. Out of the most common sets of principles used to manage real-time and historical data, the Lambda architecture has been widely accepted as a proven technology for various big data applications.

Some of the better-known real-time examples of Lambda architecture are discussed below:

- **Twitter:** Performs sentiment analysis using tweets.
- **Crashlytics:** Powers mobile analytics to generate meaningful analytical results
- **Stack Overflow:** Uses batch views to derive analytics for voting

6. MemSQL: A Complete Solution for Lambda

6.1 MemSQL Overview

MemSQL is an operational database built for performing both transactions and analytics to support the demands of modern applications, analytical systems, and ML/AI at scale. MemSQL uses a cloud-native, distributed architecture to deliver maximum performance and elastic scale.

Note that "cloud-native" does not mean "cloud-only"; in fact, truly "cloud-native" infrastructure and apps, such as MemSQL, are completely flexible, being able to run on any cloud or on-premises. MemSQL does so by offering both multi-cloud and hybrid options, ranging from a database-as-a-service, to Kubernetes-based hybrid and private deployments, to traditional on-premises installations on VMs or commodity hardware.

MemSQL can ingest millions of events per second, with support for ACID transactions, while simultaneously supporting analytics, applications, machine learning, and AI queries on trillions[1] of data rows. MemSQL can support running transactional and analytical workloads under high concurrency, all while still supporting the standard ANSI SQL query language. You can read this technical whitepaper to learn more about the concepts behind the MemSQL data platform.

If you're already familiar with running MemSQL on-premises, you can now enjoy the ultra-high performance and elastic scalability of MemSQL in the cloud.

[1]Learn how MemSQL processing shatters the trillion rows per second mark:
https://www.memsql.com/blog/memsql-processing-shatters-trillion-rows-per-second-barrier/

6.2 Why MemSQL is a Great Solution for Lambda

The biggest challenge of the Lambda architecture is that it duplicates what already is "Big Data" into the batch and speed layers, which increases the memory and storage requirements. The MemSQL value proposition is that you can use it in the batch and speed layers in the same database, which simplifies the data infrastructure management. Alternatively, you can avoid data duplication by simply not duplicating data, or creating materialized views (which is what happens in practice in the batch and speed layers with other datastores), because MemSQL is fast enough to ingest the raw, new data while simultaneously performing the aggregations and the additional work needed by queries due to its dual storage (soon to be SingleStore), distributed query execution, and query optimization capabilities.

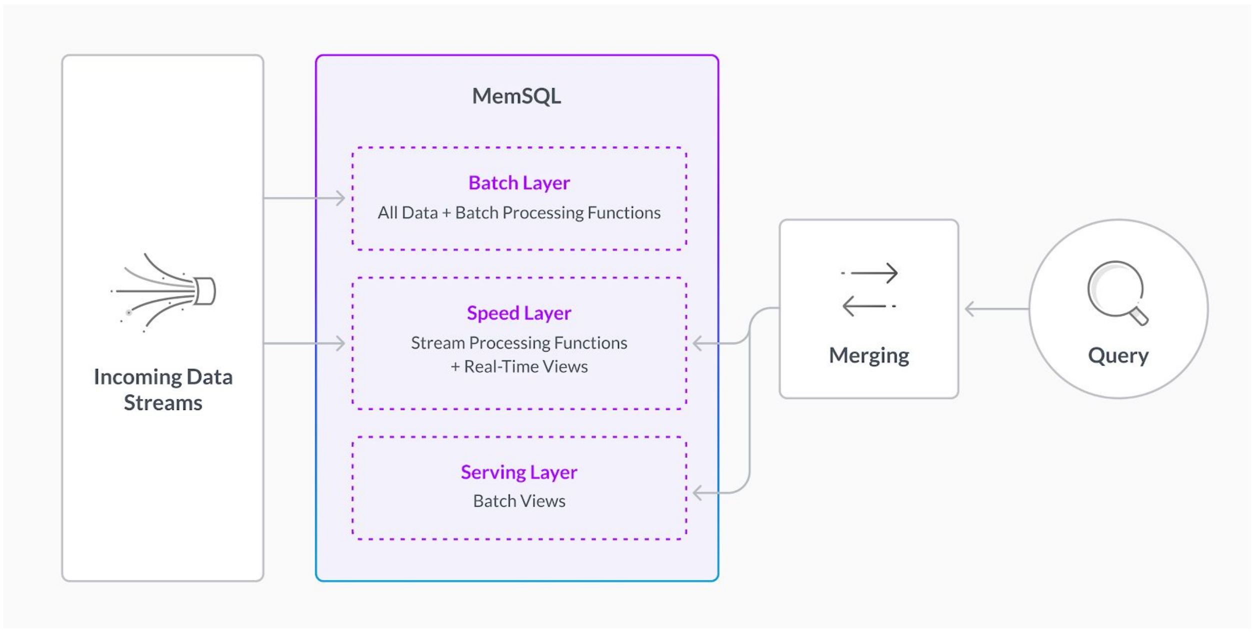


Figure 2. Lambda architecture simplified with MemSQL

MemSQL delivers real-time analytics on a rapidly changing data set, making it an ideal match for the characteristics of the Lambda architecture speed layer. Other data stores have limitations that inhibit high-speed data ingest, lack analytical capabilities (especially SQL support), or cannot scale. MemSQL offers a complete solution with its ability to handle tens of millions of transactions per second while performing complex multi-table join queries in parallel.

Let’s dig into some of the key innovations that make MemSQL an ideal solution for simplifying the Lambda architecture.

6.2.1 Highly Scalable

MemSQL uses a distributed, shared-nothing architecture that scales on commodity hardware and local storage, supporting petabytes of data. MemSQL is a memory-first, relational database that also offers a disk-based compressed columnstore. In-memory optimization provides high-speed data ingestion while simultaneously delivering analytics on the rapidly changing data set. The disk-based columnstore provides historical data management and access to historical data trends to leverage, in combination with the “hot” real-time data, to deliver real-time analytics.

6.2.2 High-Performance for Both OLTP and OLAP Workloads

In MemSQL, data is sharded automatically amongst nodes in a cluster. Sharding optimizes query performance for both distributed aggregate queries and filtered queries with equality predicates. You can add nodes and redistribute shards (also known as partitions) as needed to scale your workload.

MemSQL also supports storing and processing data using an in-memory rowstore or a disk-based columnstore. The in-memory rowstore provides optimum real-time performance for transactional workloads. The disk-based columnstore is best for analytical workloads across large historical datasets.

Rowstores are the default table type, but columnstores can be created by specifying a columnstore index type. A combination of the MemSQL rowstore and columnstore engines simplify your technology stack by allowing you to merge real-time and historical data in a single query.

In addition to handling mixed workloads, MemSQL supports high concurrency for simultaneous users. A distributed query optimizer evenly divides the processing workload to maximize the efficiency of CPU usage. Query plans are compiled to machine code and cached to expedite subsequent executions. A key feature of these compiled query plans is they do not pre-specify values for the parameters. This allows MemSQL to substitute the values upon request, which enables all subsequent queries of the same structure to run quickly. Moreover, with MemSQL using Multi-Version Concurrency Control (MVCC) and lock-free data structures, data remains highly accessible, even amidst a high volume of concurrent updates and queries.

MemSQL delivers a number of query performance capabilities to reduce response time. These include:

- True shared-nothing horizontal scaling and distributed parallel query execution
- Columnstore query processing for fast aggregations and optimized disk consumption
- Query compilation for faster execution
- Query vectorization for highly efficient processing by leveraging streamlined, array-oriented processing and on-chip CPU instructions

6.2.3 Multi-Model, Multi-Mode

MemSQL supports the ingestion of structured, semi-structured data, and unstructured data. Flexibility to align data to a structure in support of analytics meets the business requirements of your organizations. Real-time analytics requires a defined data structure, which MemSQL supports through a fully relational model. Furthermore, MemSQL supports the ingestion of semi-structured (JSON) data into key-value pairs.

Full ANSI SQL support makes MemSQL readily accessible to data analysts, business analysts, and data scientists, reducing application code requirements. Plugging data visualization and query tools into the analytics architecture delivers immediate value from data to the business. MemSQL also has extended SQL, including JSON support. Traversing a JSON document is similar to SQL, with extensions to traverse the key-value pairs.

6.2.4 Data Ingestion

MemSQL can load data continuously or in bulk from a variety of sources. Popular loading sources include files, Kafka clusters, cloud repositories such as Amazon S3, HDFS, and other databases. As a distributed system, MemSQL ingests data streams using parallel loading to maximize throughput.

MemSQL Pipelines is an easy-to-use, built-in capability that extracts, transforms, and loads external data using sources such as Kafka, S3, Azure Blob, and filesystems. It is ideal for scenarios where data from a supported source must be ingested and processed in real-time. This makes MemSQL Pipelines a good alternative to third-party middleware for basic ETL operations that must be executed as fast as possible, thus eliminating traditional long-running processes such as overnight batch jobs.

For bulk data load operations, MemSQL offers a LOAD DATA function that imports files in parallel to maximize performance.

7. Customer Success Stories

In this section, we will take a look at our customers successfully leveraging MemSQL for their lambda architecture, built for real-time data processing and exploration.

7.1 Real-Time Analytics at Comcast

Our first example comes from the Comcast Xfinity data team, who built a data processing infrastructure that focuses on real-time operational analytics. Using a combination of MemSQL and Hadoop, Comcast can proactively diagnose potential issues in an instant and deliver the best possible video experience. The Comcast architecture writes one copy of data to a MemSQL instance and a separate copy to Hadoop. This enables Comcast to run real-time analytics on massive, ever-changing datasets, while also making their analytics infrastructure more performant. Instead of just logging all Xfinity data and analyzing it hours or days later, Comcast has the power to get both viewership and infrastructure monitoring metrics the moment they occur. HDFS provides a quasi-infinite data store where they can run machine learning jobs and other "offline" analytics

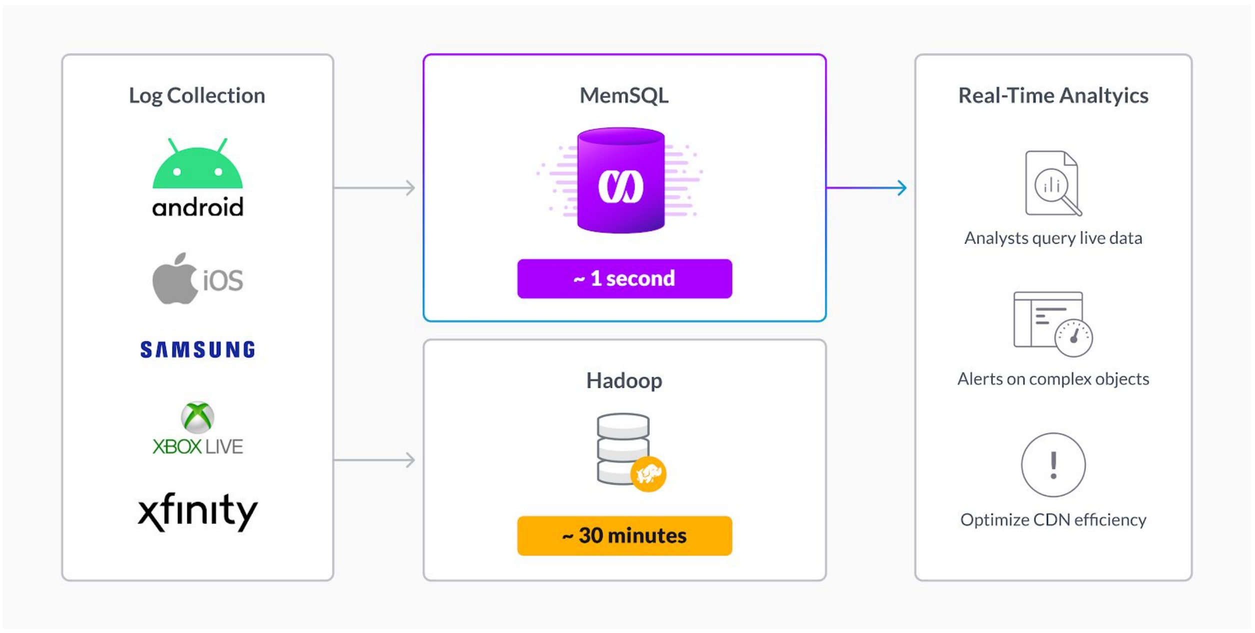


Fig 3. Real-time analytics at Comcast

Watch the Comcast team's recorded session and blog from Strata+Hadoop World to learn how Comcast architected their Xfinity platform to work with millions of users, process enormous volumes of data and, at the same time, perform advanced real-time analytics.

7.2 How Tapjoy Powers the Mobile Ad Platform

Tapjoy, the mobile app industry's leading mobile marketing automation and monetization platform, is processing and analyzing real-time and historical data concurrently to power its ad platform.

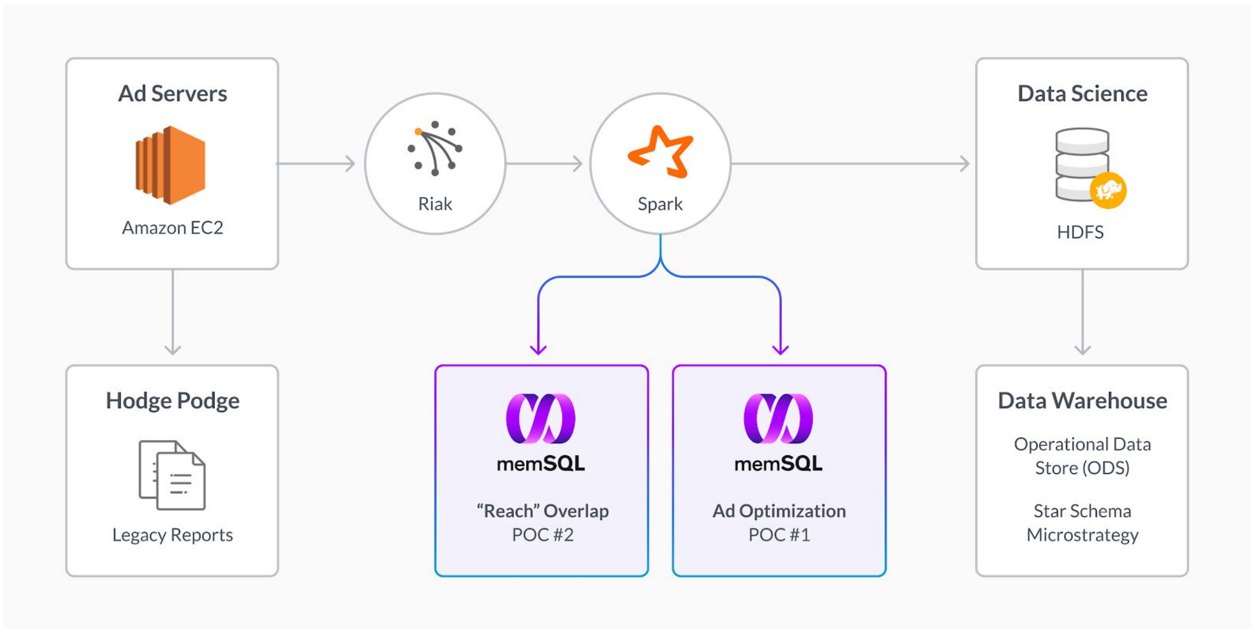


Fig 4. Tapjoy's Database Architecture

Tapjoy optimizes ad performance by taking advantage of the speed and scalability of in-memory computing. With the processing power to run 60,000 queries at a response time of less than ten milliseconds, Tapjoy is able to cross-reference user data and serve higher-performing ads to more than 500 million global users.

Read the full post about Tapjoy moving to MemSQL on their engineering blog: http://eng.tapjoy.com/blog-list/moving-to-memsql

Additional resources:

Learn how Tapjoy is Powering its Mobile Ad Platform with MemSQL:
https://www.memsql.com/blog/tapjoy-is-powering-its-mobile-ad-platform-with-memsql/

Tapjoy Presenting at the In-Memory Computing Summit 2015:
https://www.youtube.com/watch?v=K6brONAzRyc

8. Conclusion

Today's applications are built with a mindset to scale with no limits. Modern digital enterprises have started adopting memory-optimized, distributed data systems to process near-real-time applications in the most efficient way. The hybrid approach of the Lambda architecture helps by dividing a big data system's work into real-time and batch processing of data. MemSQL fulfills the speed layer of the Lambda architecture, providing in-memory performance to ingest and process streaming data, and the batch layer, by supporting bulk or batch updates to the compressed columnar storage on disk.

Traditional business intelligence and analytics tools were designed for a world of periodically updated static sources. They often require duplication of data into data warehouses or proprietary data stores, and lack streaming query capabilities. In short, if you seek an architecture that is more reliable in updating the data lake, as well as efficient in operationalizing machine learning models to predict upcoming events in a robust manner, then you should consider the Lambda architecture. It delivers increased throughput, reduced latency, and a reduction in errors.

9. The Future of Lambda: Kappa Architecture

In Kappa architecture, the batch layer is removed and the speed layer is enhanced to offer reprocessing capabilities. The Kappa architecture simplifies data processing by reducing the Lambda architecture's overhead that comes from handling two separate code bases and data stores for stream and batch processing. Kappa architecture delivers the best of both worlds by maintaining a single code base, and a single data store, while still allowing historical data querying and analysis through stream replays when code changes.

The Kappa architecture is particularly suited for real-time applications because it focuses on the speed layer. This architecture can be used for a variety of Big Data use cases such as real-time analytics of user activities, network and social data collected by a telco operator, real-time analytics in an Internet of Things (IoT) implementation, etc. MemSQL can fulfill the goals of the Kappa architecture, supporting real-time applications through modern streaming and analytics by delivering fast, usable insights on streaming data. Many companies have recently started adopting the Kappa architecture to re-define their streaming platform, which can be successfully achieved with the power of MemSQL.