**Articles about:**

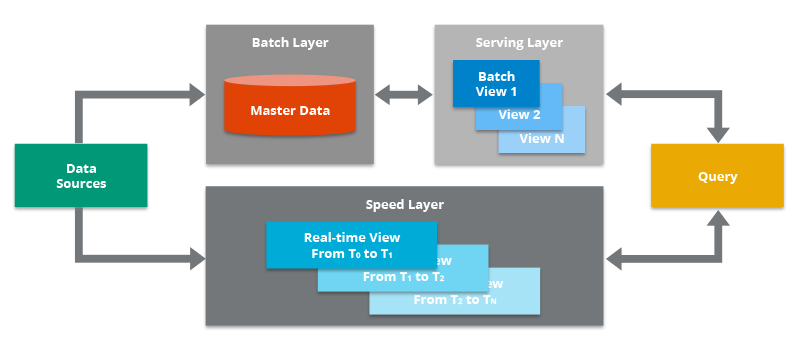
1. Lambda Architecture and similar approaches of the stream processing system. Why can’t you do both real-time processing and also handle the reprocessing when code changes? The excitement about the Lambda Architecture.
2. Problem: migrating millions of active objects and refactoring thousands of lines of code. In this article explains how we safely did one large migration of our hundreds of millions of Subscriptions objects.

**Keywords:** CAP theorem, MapReduce, migration, low-latency, ad-hoc, Apache Kafka, Hadoop.

**What Lambda Architecture tries to solve?**

Lambda architecture is a way of processing massive quantities of data (i.e. “Big Data”) that provides access to batch-processing and stream-processing methods with a hybrid approach. Lambda architecture is used to solve the problem of computing arbitrary functions.

The Lambda Architecture is aimed at applications built around complex asynchronous transformations that need to run with low latency (say, a few seconds to a few hours). A good example would be a news recommendation system that needs to crawl various news sources, process, and normalize all the input, and then index, rank, and store it for serving.  
  
**The lambda architecture itself is composed of 3 layers:**



*Batch Layer*

New data comes continuously, as a feed to the data system. It gets fed to the batch layer and the speed layer simultaneously. It looks at all the data at once and eventually corrects the data in the stream layer. This layer is built using a predefined schedule, usually once or twice a day. The batch layer has two very important functions:

* To manage the master dataset
* To pre-compute the batch views.

*Serving Layer*

The outputs from the batch layer in the form of batch views and those coming from the speed layer in the form of near real-time views get forwarded to the serving. This layer indexes the batch views so that they can be queried in low-latency on an ad-hoc basis.

*Speed Layer (Stream Layer)*

This layer handles the data that are not already delivered in the batch view due to the latency of the batch layer. In addition, it only deals with recent data in order to provide a complete view of the data to the user by creating real-time views.

**Pros**

* Batch layer of Lambda architecture manages historical data with the fault tolerant distributed storage which ensures low possibility of errors even if the system crashes.
* It is a good balance of speed and reliability.
* Fault tolerant and scalable architecture for data processing.

**Cons**

* It can result in coding overhead due to involvement of comprehensive processing.
* Re-processes every batch cycle which is not beneficial in certain scenarios.
* A data modelled with Lambda architecture is difficult to migrate or reorganize.

**What is the critique given against Lambda Architecture?**

**Kappa Architecture**

In 2014 Jay Kreps started a discussion where he pointed out some discrepancies of Lambda architecture that further led the big data world to another alternate architecture that used less code resource and was capable of performing well in certain enterprise scenarios where using multi layered Lambda architecture seemed like extravagance.

*Kappa Architecture is similar to Lambda Architecture* without a separate set of technologies for the batch pipeline. Rather, all data is simply routed through a stream processing pipeline. All data is stored in a messaging bus (like Apache Kafka), and when reindexing is required, the data is re-read from that source.

**Pros**

* Kappa architecture can be used to develop data systems that are online learners and therefore don’t need the batch layer.
* Re-processing is required only when the code changes.
* It can be deployed with fixed memory.
* It can be used for horizontally scalable systems.
* Fewer resources are required as the machine learning is being done on the real time basis.

**Cons**

* Absence of batch layer might result in errors during data processing or while updating the database that requires having an exception manager to reprocess the data or reconciliation.

Kappa Architecture cannot be taken as a substitute of Lambda architecture on the contrary it should be seen as an alternative to be used in those circumstances where active performance of batch layer is not necessary for meeting the standard quality of service. This architecture finds its applications in real-time processing of distinct events.

One advantage of the Lambda Architecture, however, is that much larger data can be stored and processed more efficiently in Hadoop for large-scale historical analysis.

**Keywords explanation.**

The **CAP theorem** (also called Brewer’s theorem) states that a distributed database system can only guarantee two out of these three characteristics: Consistency, Availability, and Partition Tolerance.

* A system is said to be consistent if all nodes see the same data at the same time.
* Availability in a distributed system ensures that the system remains operational 100% of the time. Every request gets a (non-error) response regardless of the individual state of a node.
* Partition Tolerance states that the system does not fail, regardless of if messages are dropped or delayed between nodes in a system.

**Hadoop** is an open source, Java based framework used for storing and processing big data. The data is stored on inexpensive commodity servers that run as clusters. Its distributed file system enables concurrent processing and fault tolerance.

**MapReduce** is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.,

**Data migration** is the process of moving data from one location to another, one format to another, or one application to another. Generally, this is the result of introducing a new system or location for the data.

**Low latency** describes a computer network that is optimized to process a very high volume of data messages with minimal delay (latency). These networks are designed to support operations that require near real-time access to rapidly changing data.

**Apache Kafka** is an open-source distributed publish-subscribe messaging platform that has been purpose-built to handle real-time streaming data for distributed streaming, pipelining, and replay of data feeds for fast, scalable operations.

Kafka is a broker-based solution that operates by maintaining streams of data as records within a cluster of servers. Kafka servers can span multiple data centres and provide data persistence by storing streams of records (messages) across multiple server instances in topics.  
  
  
**4 steps of data migration.**

1. **Dual writing** to the existing and new tables to keep them in sync.
2. **Changing all read paths** in our codebase to read from the new table.
3. **Changing all write paths** in our codebase to only write to the new table.
4. **Removing old data** that relies on the outdated data model.

**What can we achieve if we follow the pattern for data migration provided in the article?**

If we follow this pattern, we can achieve to run the migration safely. Also it would allow us to transition data stores while operating our services in production without any downtime. All the changes were incremental. At each step of the way, this pattern gained confidence in the safe migration.