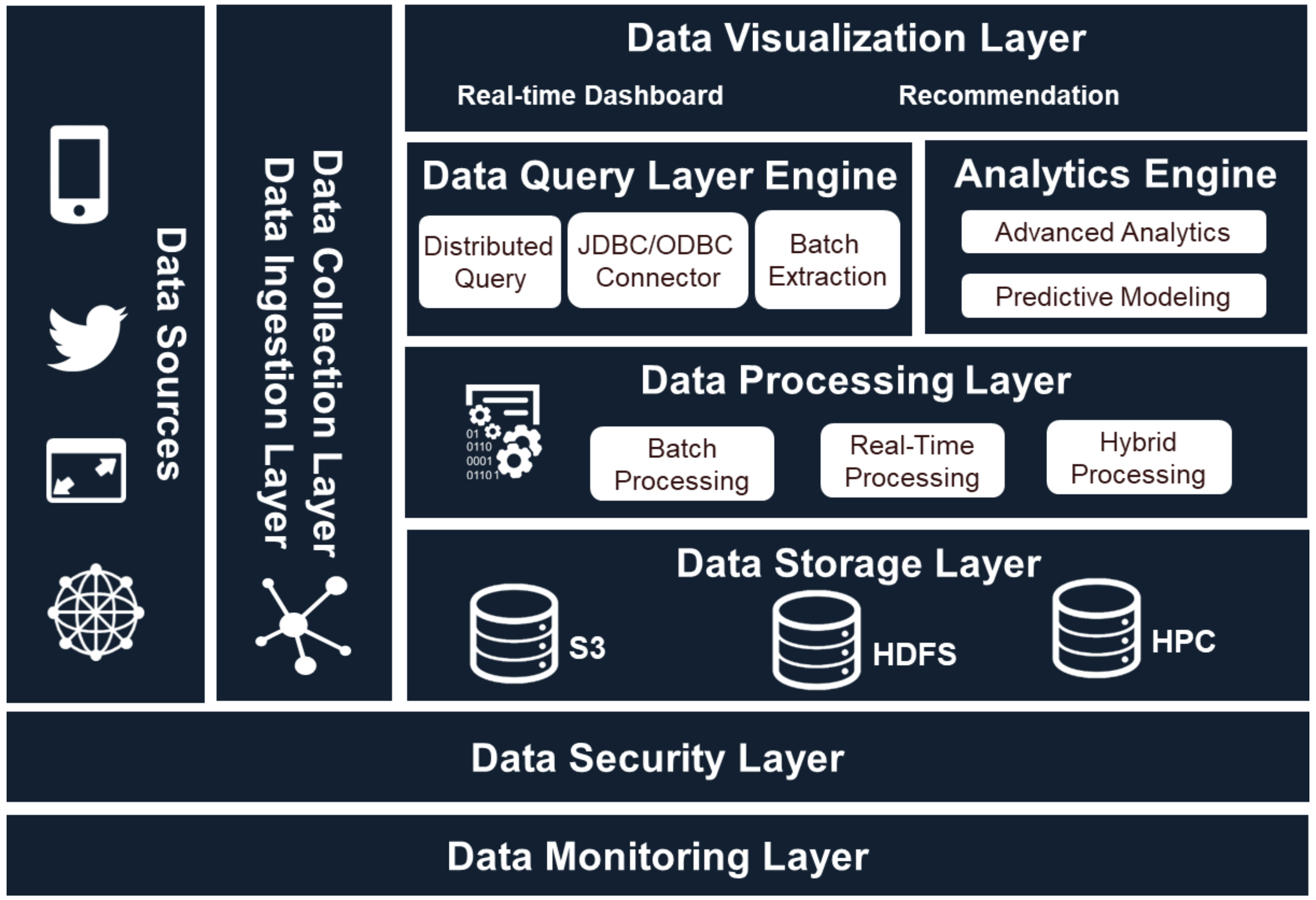
Data Schools

Module 3: Introduction To IoT

Big Data & IoT Analytics

How Do We Handle Big Data / IoT?
- Big Data Framework



IoT and Big Data Analytics

Data Ingestion Layer

Big Data Ingestion involves connecting to various data sources, extracting the data, and detecting the changed data. It's about moving data - and especially the unstructured data - from where it is originated, into a system where it can be stored and analyzed.

- Challenges: with IoT, volume and variance of data sources
- Parameters: velocity, size, frequency, formats
- Key principles: network bandwidth, right tools, streaming data
- Tools: Example: Apache Flumes, Apache Nifi

Data Collection (Integration) Layer

kafka

In this Layer, more focus is on transportation data from ingestion layer to rest of Data Pipeline. Here we use a messaging system that will act as a mediator between all the programs that can send and receive messages.

- Kafka works with Storm, Hbase, Spark for real-time analysis and rendering streaming data
  - Building Real-Time streaming Data Pipelines that reliably get data between systems or applications
  - Building Real-Time streaming applications that transform or react to the streams of data.
- Data Pipeline is the main component of data integration

Data Processing Layer

In this Layer, data collected in the previous layer is processed and made ready to route to different destinations.

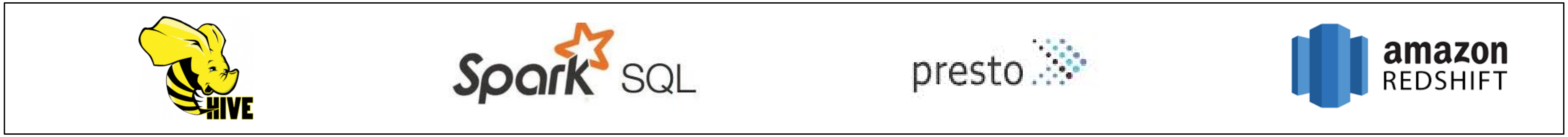
- **Batch processing system** - A pure batch processing system for offline analytics (Sqoop).
- **Near real time processing system** - A pure online processing system for on-line analytic (Storm).
- **In-memory processing engine** - Efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets (Spark)
- **Distributed stream processing** - Provides results that are accurate, even in the case of out-of-order or late-arriving data (Flink)

Data Storage Layer

Next, the major issue is to keep data in the right place based on usage. A combination of distributed file systems and NoSQL databases provide scalable data storage platforms for Big Data / IoT

- **HDFS** - A Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers.
- **Amazon Simple Storage Service (Amazon S3)** - Object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web.
- **NoSQL** – Non-relational databases that provide a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

Data Query (Access) Layer



This is the layer where strong analytic processing takes place. Data analytics is an essential step which solved the inefficiencies of traditional data platforms to handle large amounts of data related to interactive queries, ETL, storage and processing

- **Tools** – Hive, Spark SQL, Presto, Redshift
- **Data Warehouse** - Centralized repository that stores data from multiple information sources and transforms them into a common, multidimensional data model for efficient querying and analysis.
- **Data Lake** - Cloud-based enterprise architecture that structures data in a more scalable way that makes it easier to experiment with it. All data is retained

Data Visualization Layer

This layer focus on Big Data Visualization. We need something that will grab people's attention, pull them in, make your findings well-understood. This is the where the data value is perceived by the user.

- **Dashboards** – Save, share, and communicate insights. It helps users generate questions by revealing the depth, range, and content of their data stores.
  - Tools - Tableau, AngularJS, Kibana, React.js
- **Recommenders** - Recommender systems focus on the task of information filtering, which deals with the delivery of items selected from a large collection that the user is likely to find interesting or useful.

Introduction

- Modern data pipelines receive data at a high ingestion rate
- Volume, variety and velocity are important considerations for real time analytics in the context of Big Data / IoT
- To maximize the benefit of IoT data, we need an integrated platform to leverage the ability to collect, analyze and act upon this streaming data in real-time.
- Stream and real time processing frameworks together with analytics accommodating the variety, velocity and volume of big data generated by the Internet of Things

Streaming and Real-Time Data Processing

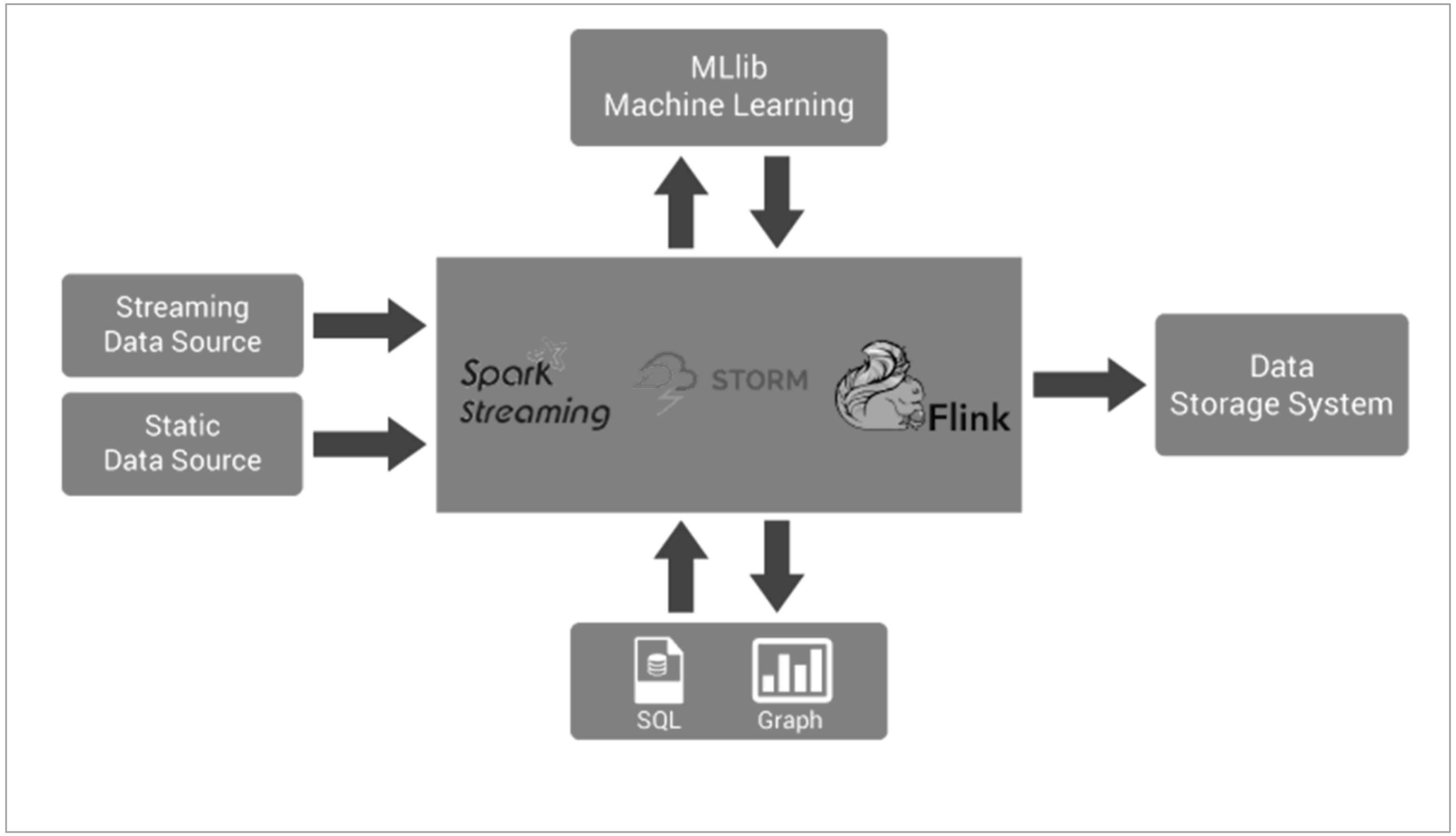
Stream Processing

Refers to a method of continuous computation that happens as data is flowing through the system. There are no compulsory time limitations in stream processing.

Real-time Processing

But Real-time data needs to have tight deadlines in the terms of time. So we normally consider that if our platform is able to capture any event within 1 ms, then we call it as real-time data or true streaming.

Stream & Real Time Processing Frameworks



Processing of huge volumes of data is not enough. We need to process them in real-time so that decisions are taken immediately whenever any important event occurs.

Streaming Architectures

There are two types of architectures which are used while building real-time pipelines

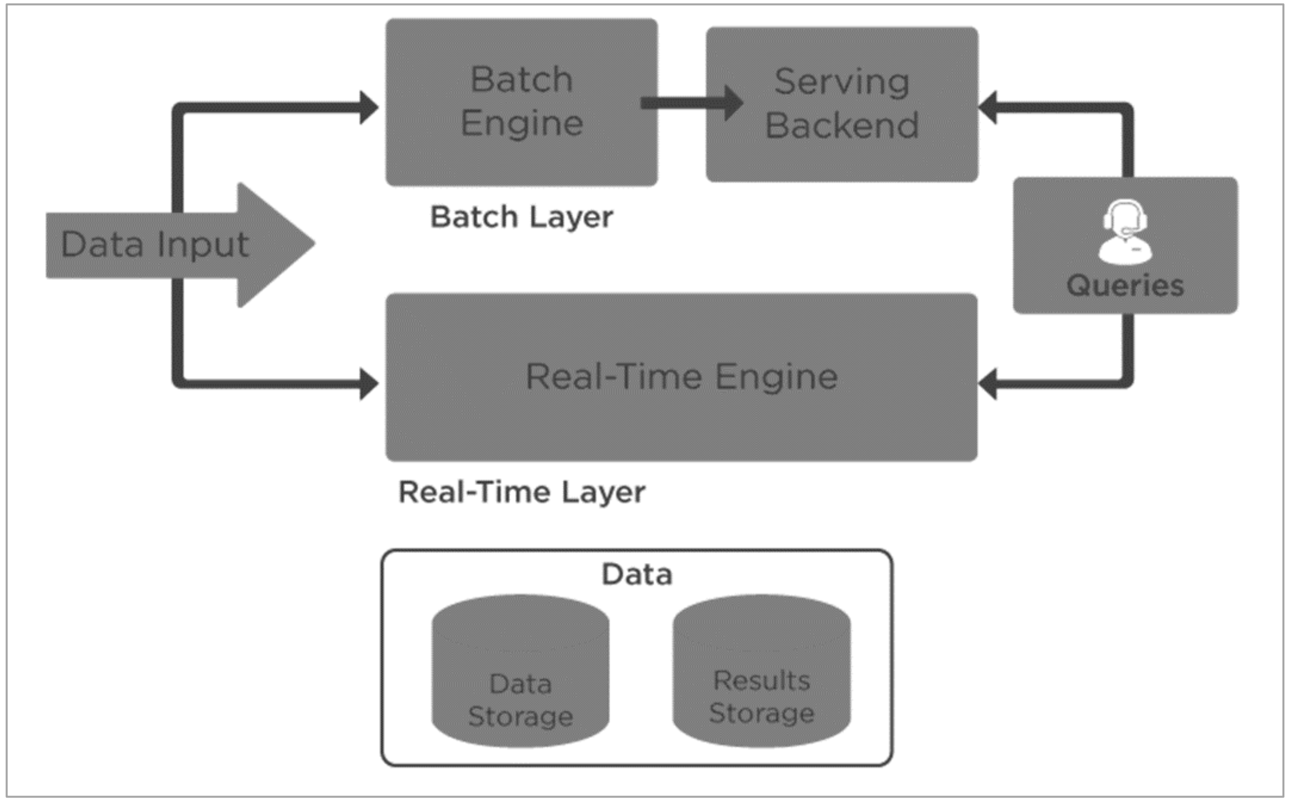
Lambda Architecture

This architecture was introduced by Nathan Marz in which we have three layers to provide real-time streaming and compensate any data error occurs if any. The three layers are Batch Layer, Speed layer, and Serving Layer.

Kappa Architecture

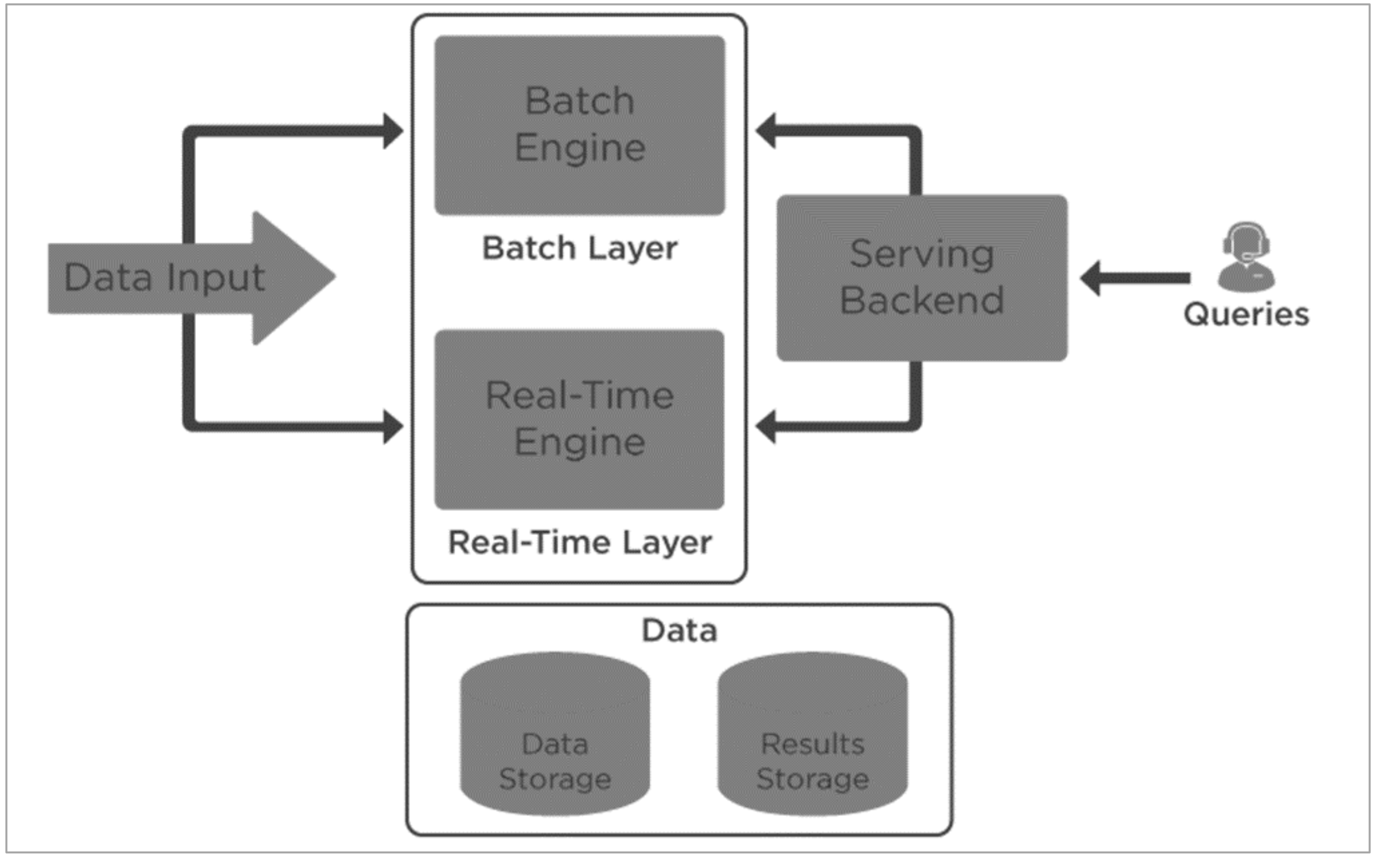
One of the important motivations for inventing the Kappa architecture was to avoid maintaining two separate code bases for the batch and speed layers. The key idea is to handle both real-time data processing and continuous data reprocessing using a single stream processing engine.

Lambda Architecture



- The batch layer has two major tasks: (a) managing historical data; and (b) recomputing results such as machine learning models.
- The speed layer is used in order to provide results in a low-latency, near real-time fashion.

Kappa Architecture



- The key idea is to handle both real-time data processing and continuous data reprocessing using a single stream processing engine.

ANY
QUESTIONS