SUMMARY

1. Big Data Architectures

Big data solutions typically involve one or more of the following types of workload:

- Batch processing of big data sources at rest: large amounts of data that are stored in a static or unchanging state, such as data stored in a database or data warehouse.
- Real-time processing of big data in motion
- Interactive exploration of big data
- Predictive analytics and machine learning

Big data architectures are different from traditional DBs. Specific big data architectures are needed when:

- Store and process large volumes of data
- transform unstructured data → structured data: analysis and reporting
- Capture, process, and analyze **unbounded streams of data in real time**, or with **low latency**

→ The most used architecture for big data is the **Lambda Architecture** (Nathan Marz in 2011).

Lambda Architecture

🔎 A **Lambda Architecture** is a big data processing architecture designed to handle massive quantities of data by taking advantage of both batch and real-time processing methods. The architecture is composed of three layers: the batch layer, the speed layer, and the serving layer. Data is processed simultaneously in both the batch and speed layers, and the results are combined and made available for querying in the serving layer. The architecture is fault-tolerant, scalable, and able to handle large volumes of input data, making it a popular choice for big data processing.

Lambda architecture depends on a data model with an append-only, immutable data source that serves as a system of record. It is intended for ingesting and processing timestamped events that are appended to existing events rather than overwriting them. State is determined from the natural time-based ordering of the data.

Requirements:

- Fault-tolerant against both hardware failures and human errors
- Support variety of use cases that include low latency querying as well as updates
- Linear scale-out capabilities
- Extensible: so that the system is manageable and can accommodate newer features easily

Queries properties:

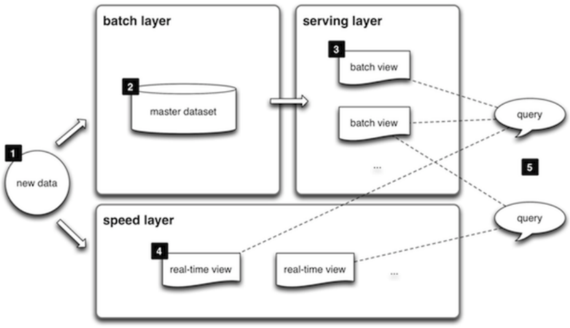
- Latency: the time it takes to run a query
- Timeliness: how up to date the query results are (**freshness** and **consistency**)
- Accuracy: Tradeoff between performance and scalability (**approximations**)

Lambda Architecture is based on 2 data paths:

- **Batch layer** (cold path): It stores all of the incoming data in its raw form and performs batch processing on the data. The result of this processing is stored as batch views.
- **Speed layer** (hot path): It analyzes data in real time. This path is designed for low latency, at the expense of accuracy.

The **Batch layer** is responsible for processing large volumes of data that are stored in a static or unchanging state, such as data stored in a database or data warehouse. The **Speed layer**, on the other hand, is responsible for processing unbounded streams of data in real time or with low latency. These two paths are then combined in the serving layer to provide a complete view of the data.

1. All data entering the system is dispatched to both the batch layer and the speed layer for processing
2. The batch layer has two functions:
    - managing the master dataset, an immutable, append-only set of raw data)
    - to pre-compute the batch views
3. The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way
4. The speed layer compensates for the high latency of updates to the serving layer and deals with recent data only



→ Any incoming query can be answered by merging results from batch views and realtime views

Lambda architecture: A more detailed view