Scalable Efficient Big Data Pipeline Architecture

Architecture for High-Throughput Low-Latency Big Data Pipeline on Cloud

📅 Mar 3, 2020 · ☕ 11 min read
🏷 #BigData #DataPipeline #Cheatsheet

What's on this Page

For deploying big-data analytics, data science, and machine learning (ML) applications in the real world, analytics-tuning and model-training is only around 25% of the work. Approximately 50% of the effort goes into making data ready for analytics and ML. The remaining 25% effort goes into making insights and model inferences easily consumable at scale. The big data pipeline puts it all together. It is the railroad on which heavy and marvelous wagons of ML run. Long-term success depends on getting the data pipeline right.

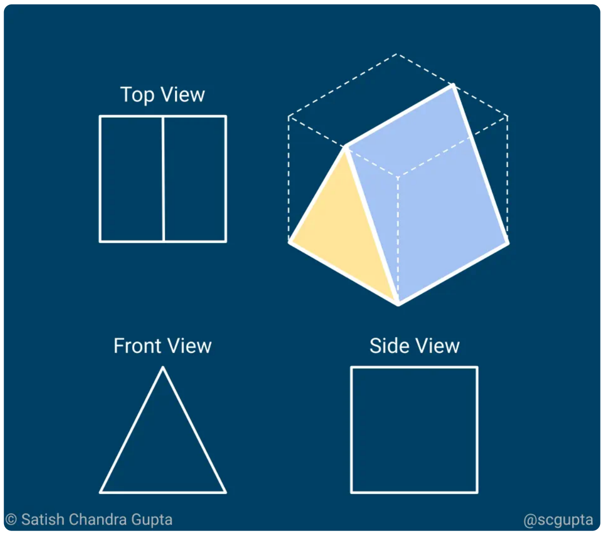
This article gives an introduction to the data pipeline and an overview of big data architecture alternatives through the following four sections:

- **Perspective:** By understanding the perspectives of all stakeholders, you can enhance the impact of your work. This section explains various perspectives and

- **Pipeline:** In this section, you will learn about the conceptual **stages** of a big data pipeline passing through the **data lake** and the **data warehouse**.
- **Possibilities:** In this section, you will learn about the **lambda architecture** for balancing scale and speed, and **technology choices** for the key components of the big data architecture. You will also get a glimpse of **serverless** pipelines on **AWS**, **Azure**, and **Google Cloud**.
- **Production:** This section offers tips for your big data pipeline deployment to be successful in production.

What's on this Page

Perspective



Perspective: View depends on the vantage point.

There are three stakeholders involved in building data analytics or machine learning applications: data scientists, engineers, and business managers.

From the data **science** perspective, the aim is to ***find*** the *most robust* and computationally *least expensive* model

From the **engineering** perspective, the aim is to ***build*** things that *others can depend on;* to innovate either by building *new things* or finding *better* ways to build existing things that function *24x7* without much human intervention.

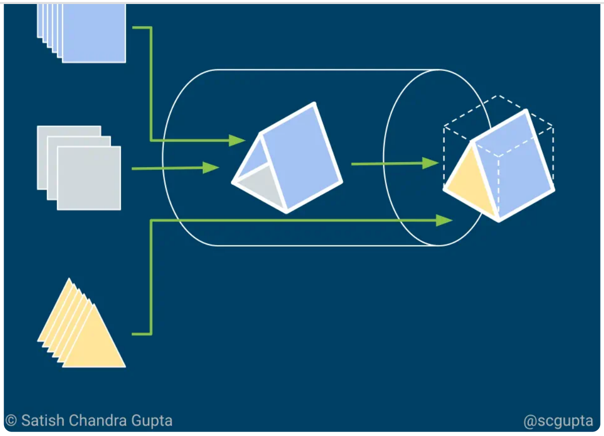
From the **business** perspective, the aim is to ***deliver*** *value* to customers; science and engineering are means to that end.

In this article, we will focus on the engineering perspective, and specifically the aspect of processing a huge amount of data needed in ML applications, while keeping other perspectives in mind. Desired engineering characteristics of a data pipeline are:

- **Accessibility:** data being easily accessible to data scientists for hypothesis evaluation and model experimentation, preferably through a query language.
- **Scalability:** the ability to scale as the amount of ingested data increases, while keeping the cost low.
- **Efficiency:** data and machine learning results being ready within the specified latency to meet the business objectives.
- **Monitoring:** automatic alerts about the health of the data and the pipeline, needed for proactive response to potential business risks.

Pipeline

What's on this Page



Pipeline: Well oiled big data pipeline is a must for the success of machine learning.

The value of data is unlocked only after it is transformed into actionable insight, and when that insight is promptly delivered.

A **data pipeline** stitches together the end-to-end operation consisting of *capturing* the data, *transforming* it into insights, *training* a model, *delivering* insights, *applying* the model whenever and wherever the action needs to be taken to achieve the business goal.

> Data is the new oil. It's valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc. to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value.
> — Clive Humby, UK Mathematician and architect of Tesco's Clubcard

Stages in Data Processing Pipeline

What's on this Page

- **Capture** data from internal & external sources.
  Data sources (mobile apps, websites, web apps, microservices, IoT devices, etc.) are instrumented to capture relevant data.

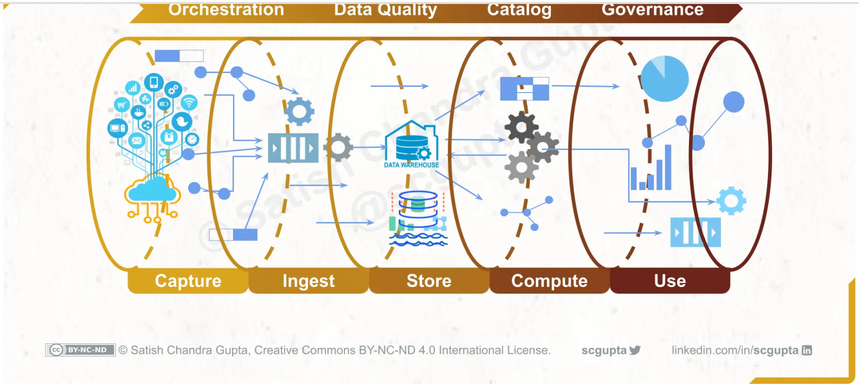
- **Ingest** data through batch jobs or streams.
  The instrumented sources pump the data into various inlet points (HTTP, MQTT, message queue, etc.). There can also be jobs to import data from services like Google Analytics. The data can be in two forms: batch blobs and streams.

- **Store** in Data Lake or Data Warehouse.
  Often raw data/events are stored in Data Lakes, and the it is cleaned, duplicates and anomalies removed, and transformed to conform to schema. Finally, this ready-to-consume data is stored it in a Data Warehouse.

- **Compute** analytics aggregations and/or ML features.
  This is where analytics, data science, and machine learning happen. Computation can be a combination of batch and stream processing. Models and insights (both structured data and streams) are stored back in the Data Warehouse.

- **Use** it in dashboards, data science, and ML.
  The insights are delivered through dashboards, emails, SMSs, push notifications, and microservices. The ML model inferences are exposed as microservices.

What's on this Page



Stages in data pipeline processing pipeline

Data Lake vs. Data Warehouse

The Data Lake contains all data in its natural/raw form as it was received usually in blobs or files. The Data Warehouse stores cleaned and transformed data along with catalog and schema. The data in the lake and the warehouse can be of various types: structured (relational), semi-structured, binary, and real-time event streams.

It is a matter of choice whether the lake and the warehouse are kept physically in different stores, or the warehouse is materialized through some kind of interface (e.g. Hive queries) over the lake. The choice is driven by speed requirements and cost constraints.

> No matter which approach is followed, it is important to retain the raw data for audit, testing and debugging purposes.

Exploratory Data Analysis

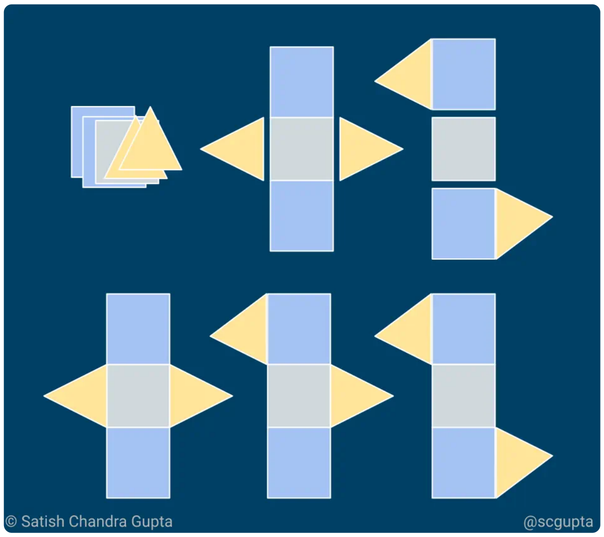
What's on this Page

expose gaps in the collected data, lead to new data collection and experiments, and verify a hypothesis.

You can think of them as small-scale ML experiments to zero in on a small set of promising models, which are compared and tuned on the full data set.

Having a well-maintained Data Warehouse with catalogs, schema, and accessibility through a query language (instead of needing to write programs) facilitates speedy EDA.

Possibilities



Possibilities: Architecture is a trade-off between performance and cost. There are six options sown in this image to make a triangular tent shape. From top left to bottom right, the amount of glue needed to make the tent decreases. Which one will you choose to do in prod? Notice that the base of the triangle is smaller than the other side, and the light blue piece is a rectangle and not a square.

using batch programs, SQL, or even Excel sheets. What has changed now is the availability of big data that facilitates machine learning and the increasing demand for real-time insights.

Big Data Pipeline Architecture

There are several architectural choices offering different performance and cost tradeoffs (just like the options in the accompanying image). I have learned that the technically best option may not necessarily be the most suitable solution in production. You must carefully examine your requirements:

- Do you need real-time insights or model updates?
- What is the staleness tolerance of your application?
- What are the cost constraints?

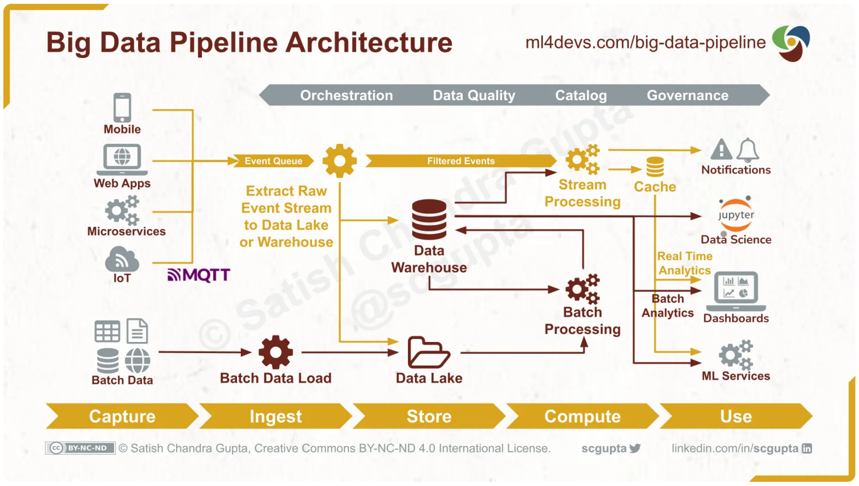
Based on the answers to these questions, you have to balance the batch and the stream processing in the Lambda architecture to match your requirements of throughput and latency. Lambda architecture consists of three layers:

- **Batch Layer:** offers high throughput, comprehensive, economical map-reduce batch processing, but higher latency.
- **Speed Layer:** offers low latency real-time stream processing, but costlier and may overshoot memory limit when data volume is high.

What's on this Page

output of the stream processing to provide comprehensive results in the form of pre-computed views or ad-hoc queries.

The underlying assumption in the lambda architecture is that the source data model is *append-only*, i.e. ingested events are timestamped and appended to existing events, and never overwritten.



Big data pipeline architecture: batch layer is in brown color, speed layer is in orange color, and both are joined in the serving layer at the last stage.

In addition, there are following aspects across the whole pipeline:

- **Orchestration:** Data pipelines are complex and have several part forming a Directed Acyclic Graph (DAG). Pipeline Orchestration is to ensure that these parts are run in right order. All required inputs for a part must have already been computed before running apart.
- **Data Quality:** Checking the statistical distribution, outliers, anomalies, or any other tests required at each part of the data pipeline.

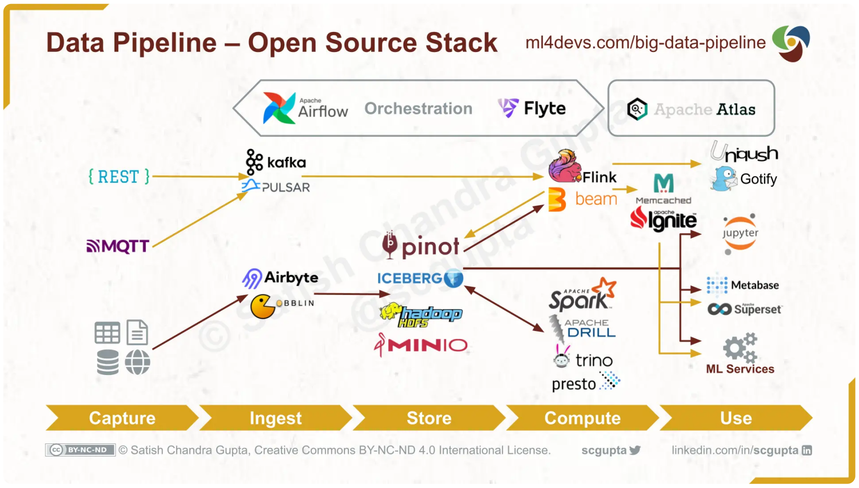
What's on this Page

data lake, topics in message queue). It creates and manages metadata and schema of the data assets so that data engineers and data scientists can understand it better.

- **Data Governance:** Policies and processes to follow throughout the lifecycle of the data for ensuring that data is secure, anonymised, accurate, and available.

Building Data Pipelines With Open Source Stack

The following figure shows an architecture using open source technologies to materialize all stages of the big data pipeline. The preparation and computation stages are quite often merged to optimize compute costs.



Building data processing pipelines using open source technologies

Key components of the big data architecture and technology choices are the following:

- **HTTP / MQTT Endpoints** for ingesting data, and also for serving the results. There are several frameworks

What's on this Page

- **Pub/Sub Message Queue** for ingesting high-volume streaming data. Kafka is currently the de-facto choice. It is battle-proven to scale to a high event ingestion rate.
- **Low-Cost High-Volume Data Store** for data lake (and data warehouse), Hadoop HDFS or cloud blob storage like AWS S3.
- **Query and Catalog Infrastructure** for converting a data lake into a data warehouse, Apache Hive is a popular query language choice.
- **Map-Reduce Batch Compute** engine for high throughput processing, e.g. Hadoop Map-Reduce, Apache Spark.
- **Stream Compute** for latency-sensitive processing, e.g. Apache Storm, Apache Flink. Apache Beam is emerging as *the* choice for writing the data-flow computation. It can be deployed on a Spark batch runner or Flink stream runner.
- **Machine Learning Frameworks** for data science and ML. Scikit-Learn, TensorFlow, and PyTorch are popular choices for implementing machine learning.
- **Low-Latency Data Stores** for storing the results. There are many well-established SQL vs. NoSQL choices of data stores depending on data type and use case.
- **Deployment** orchestration options are Hadoop YARN, Kubernetes / Kubeflow.

Scale and efficiency are controlled by the following levers:

- **Throughput** depends on the **scalability** of the ingestion (i.e. **REST/MQTT** endpoints and **message**

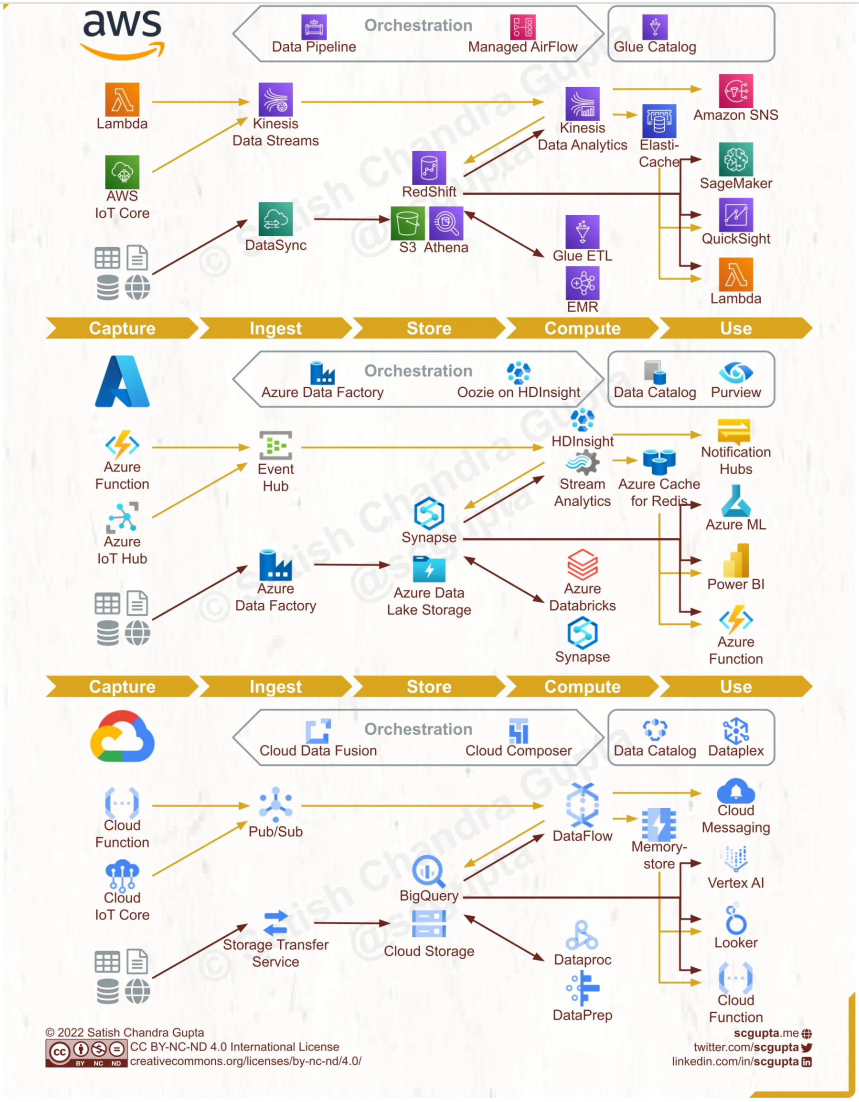
- **Latency** depends on the **efficiency** of the **message queue**, **stream compute** and **databases** used for storing computation results.

Cloud Data Pipeline on AWS, Azure, and Google Cloud

With the advent of serverless computing, it is possible to start quickly by avoiding DevOps. Various components in the architecture can be replaced by their serverless counterparts from the chosen cloud service provider.

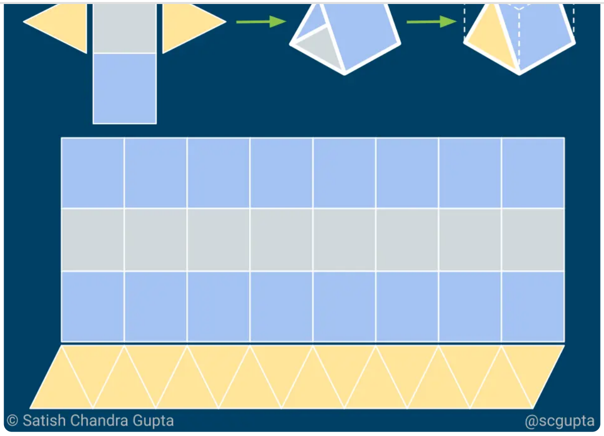
Typical cloud data pipelines on **Amazon Web Services**, **Microsoft Azure**, and **Google Cloud Platform (GCP)** are shown below. Each maps closely to the general big data pipeline architecture discussed in the previous section. You can use these as a reference for shortlisting technologies suitable for your needs.

What's on this Page



Cloud data pipeline on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP)

Production



In production, simplicity quite often trumps cleverness. You may notice that the option chosen to make the triangular tent shape is not the one needing the least amount of glue. How the needed parts will be produced, and the simplicity of the overall operation is important for reducing the potential for errors.

Production can be the graveyard of un-operationalized analytics and machine learning. If you do not invest in 24x7 monitoring of the health of the pipeline that raises alerts whenever some trend thresholds are breached, it may become defunct without anyone noticing.

> Be mindful that engineering and OpEx are not the only costs. While deciding architecture, consider time, opportunity, and stress costs too.

Operationalising a data pipeline can be tricky. Here are some tips that I have learned the hard way:

- **Scale Data Engineering before scaling the Data Science team.** ML wagons can't run without first laying railroads.
- **Be industrious in clean data warehousing.** ML is only as good as data. Be disciplined in defining the schema of the data being collected, cataloging it. In

What's on this Page

- **Start simple.** Start with serverless, with as few pieces as you can make do. Move to a full-blown pipeline, or your own deployment, only when RoI is justifiable. Bootstrap with minimal investment in the computation stage. Go even "compute-less" by implementing computations by scheduling a bunch of SQL queries and cloud functions. That will get the whole pipeline ready faster, and give you ample time to focus on getting your data strategy in place, along with data schemas and catalogs.
- **Build only after careful evaluation.** What are the business goals? What levers do you have to affect the business outcome? What insights will be actionable? Collect data and build ML based on that.

Summary

Key takeaways are:

- Tuning analytics and machine learning models is only 25% effort.
- Invest in the data pipeline early because analytics and ML are only as good as data.
- Ensure easily accessible data for exploratory work.
- Start from business goals, and seek actionable insights.

What's on this Page

What's on this Page

Share on

WRITTEN BY

Satish Chandra Gupta

Data/ML Practitioner

← Python Microservices: Choices, Key Concep...

Python Microservices: Build and Test REST ... →