Big Data Processing is the collection of methodologies or frameworks enabling access to enormous amounts of information and extracting meaningful insights. Initially, Big Data Processing involves data acquisition and data cleaning. Once you have gathered the quality data, you can further use it for Statistical Analysis or building Machine Learning models for predictions.

5 Stages of Big Data Processing

- Data Extraction
- Data Transformation
- Data Loading
- Data Visualization/BI Analytics
- Machine Learning Application

Stage 1: Data Extraction

This initial step of Big Data Processing consists of collecting information from diverse resources like enterprise applications, web pages, sensors, marketing tools, transactional records, etc. Data processing professionals extract information through many Unstructured and Structured Data Streams. For instance, in building a Data Warehouse, extracting entails merging information from multiple sources, subsequently verifying the information by removing incorrect data. To decide future decisions based on the outcomes, the data collected during the data collection phase of Big Data Processing must be labeled and accurate. This stage establishes a quantitative standard as well as a goal for improvement.

Stage 2: Data Transformation

The data transformation phase of Big Data Processing defines changing or modifying data into required formats which helps in building different insights and visualizations. There are many transformation techniques like Aggregation, Normalization, Feature Selection, Binning and Clustering, and concept hierarchy generation. Using these techniques for Big Data Processing, developers transform Unstructured Data into Structured Data and Structured Data into a user-understandable format. Business and Analytical operations become more efficient as a result of the transformation, and firms can make better data-driven choices.

Stage 3: Data Loading

The converted data is transported to the centralized database system in the load stage of Big Data Processing. Before loading the data, index the database and remove the constraints to make the process more efficient.

Using Big Data ETL, the process of loading became automated, well-defined, consistent, and Batch-driven or Real-time.

Stage 4: Data Visualization/BI Analytics

Data Analytics tools and methods for Big Data Processing enable firms to visualize huge datasets and create dashboards for gaining an overview of the entire business operations. Business Intelligence (BI) Analytics answer fundamental business growth and strategy questions. BI tools make predictions and what-if analyses on the transformed data that help stakeholders understand the depth patterns in data and the correlations between the attributes.

Stage 5: Machine Learning Application

The Machine Learning phase of Big Data Processing is primarily concerned with the creation of models that can learn to evolve in response to the new input. The learning algorithms allow for more quickly analyzing large amounts of data.

- The first type of Machine Learning is **Supervised Learning**, which uses labelled data for training the models and predicting the outcomes. Data patterns are used in Supervised learning to identify new information output for the labels. This method is often used in applications that utilize historical data to predict future outcomes.
- **Unsupervised Learning** is the second type where the data is unlabeled and trained by the algorithm. Unsupervised Machine Learning is utilized against information that doesn't have any historical labels.
- **Reinforcement Learning** is the final type in which there is no primary data that can be inserted as input to models. The algorithms have to figure out the decisions on their own based on observations or situations that happen surrounding them. The decisions are manipulated with a reward function so that the models try to make the correct decisions.

The Machine Learning phase of Big Data Processing enables automatic recognition patterns and can perform feature extraction in complicated unstructured information without any human interference, making this a significant resource for Big Data research.

What Makes Hevo's Big Data Pipeline Unique?

Providing a high-quality ETL solution can be a difficult task if you have a large volume of data. Hevo's automated, No-code platform empowers you with everything you need to have for a smooth data replication experience.

Check out what makes Hevo amazing:

- **Fully Managed**: Hevo requires no management and maintenance as it is a fully automated platform.
- **Data Transformation**: Hevo provides a simple interface to perfect, modify, and enrich the data you want to transfer.
- **Faster Insight Generation**: Hevo offers near real-time data replication so you have access to real-time insight generation and faster decision making.
- **Schema Management**: Hevo can automatically detect the schema of the incoming data and map it to the destination schema.
- **Scalable Infrastructure**: Hevo has in-built integrations for 100+ sources **(with 40+ free sources)** that can help you scale your data infrastructure as required.
- **Live Support**: Hevo team is available round the clock to extend exceptional support to its customers through chat, email, and support calls.

6 Best Big Data Tools

- Apache Spark
- Hadoop
- Altas.ti
- HPCC
- Apache Cassandra
- Strom

1) Apache Spark

Apache Spark is a Big Data Processing and Machine Learning Analytics Engine that operates at lightning speed. Spark provides an API that is easy to use and handles large datasets for fast analytics queries. It also provides several libraries which support SQL Queries, Graph Processing, and building Machine Learning models. These conventional packages help developers work more efficiently while creating complicated workflows.

2) Hadoop

Apache Hadoop is a Java-based open-source, robust, and fault-tolerant Big Data Processing platform from the Apache software foundation. Hadoop is built to handle any type of information, including Organized, Semi-structured, and Unstructured data. Each task in Hadoop is broken into small sub-tasks, which are then allocated to each data node in the Hadoop cluster. In a Hadoop cluster, each data node processes a modest quantity of data, resulting in low network traffic.

3) Altas.ti

With accessible research tools and best-in-class technology, ATLAS.ti helps you find meaningful insights. This may be used in academia, market research, and customer experience study, including qualitative and combined methodologies analysis.

4) HPCC

HPCC's Big Data Processing solution was created by LexisNexis risk solutions company that provides data processing services under a common platform, structure, and scripting languages. It represents one of the most effective big data solutions available, allowing users to complete jobs using significantly minimum programming.

5) Apache Cassandra

The Apache Cassandra database is commonly utilized to organize large volumes of information effectively. It is the best tool for businesses that can't afford to lose their data when the data center is down. Cassandra is a NoSQL Database that allows you to transfer data horizontally across clusters seamlessly. It has the capacity for huge scalability and is not limited to joins or predefined schemas.

6) Strom

Apache Storm is a master-slave architectural computation system. It's ideal for analyzing large volumes of data in a small period of time. The Storm is the leading tool in real-time intelligence due to its low latency, scalability, and ease of deployment. Since Strom is open-source, it is used by small-scale as well as large-scale businesses.

What is a Big Data Pipeline? Key Components, Architecture & Use Cases

Big Data Pipelines can be described as subsets of ETL solutions. Like typical ETL solutions, they can dabble with semi-structured, structured, and unstructured data. The flexibility allows you to extract data from technically any source. Big Data Pipelines can also leverage the same transformations and load data into various repositories, including Data Lakes, relational databases, and Data Warehouses.

This article delves into the various salient aspects of a **Big Data Analysis Pipeline**: its key features, examples of architecture, best practices, and use cases. It also briefly introduces a Data Pipeline before diving into the nitty-gritty of Big Data Pipelines.

What is a Data Pipeline?

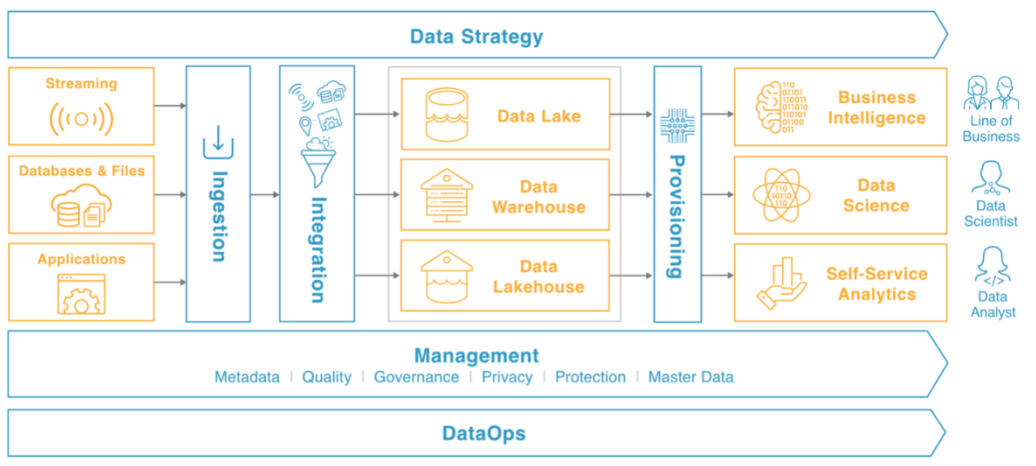


Image Source

A Data Pipeline can be described as a sequence of data processing steps. If the data is not currently loaded into the platform, it must be ingested at the beginning of the pipeline. This is followed by a series of steps, each providing an output that serves as the input for the next step. This continues until the pipeline is complete. In a couple of cases, you can run independent steps concurrently.

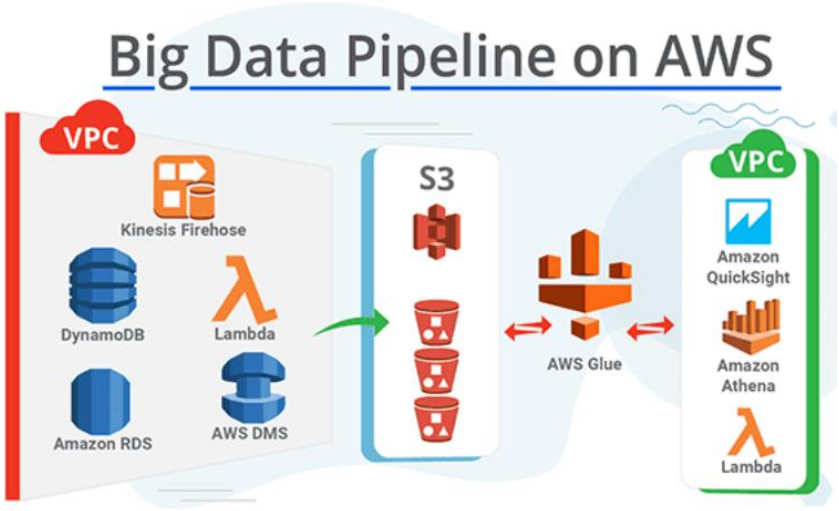
Data Pipelines contain three key elements: a source, a processing step or a set of steps, and a destination (also known as a sink). These pipelines enable data flow from an application to a Data Warehouse, for instance, from a Data Lake into a payment processing system or an analytics database. Data Pipelines can also have the same source and sink, so the pipeline is purely about changing the data set. Any time the data is

processed between points A and B (or points C, B, and D), a Data Pipeline bridges those points.

As organizations look to build applications with small code bases that serve a specific purpose, they are moving data between more and more applications, making the efficiency of Data Pipelines a critical consideration in their development and planning.

Data Pipeline architectures can explain how they're set up to enable data collation, delivery, and flow. Data can either be moved through stream processing or batch processing. In Batch Processing, data clusters are migrated from the source to the sink on a regularly scheduled or a one-time basis. On the other hand, Stream Processing allows for real-time data movement. It continuously collects data from sources like events from sensors and messaging systems or change streams from a database.

What is a Big Data Pipeline?



As the variety, volume, and speed of data have considerably grown in recent years, developers and architects have had to adapt to Big Data. Simply put, Big Data means that there is a huge volume to deal with. This massive volume of data can open opportunities for use cases such as real-time reporting, alerting, and predictive analytics, among other examples.

The most poignant difference between regular Data Pipelines and Big Data Pipelines is the flexibility to transform vast amounts of data. A Big Data Pipeline can process data in streams, batches, or other methods, with their set of pros and cons. Irrespective of the method, a Data Pipeline needs to be able to scale based on the organization's needs to serve as an effective Big Data Pipeline. Without scalability, the system might take weeks or days to complete its job.

Replicate Data in Minutes Using Hevo's No-Code Data Pipeline

Hevo Data, a Fully-managed Data Pipeline platform, can help you automate, simplify & enrich your data replication process in a few clicks. With Hevo's wide variety of connectors and blazing-fast Data Pipelines, you can extract & load data from 150+ Data Sources (**including 50+ Free Sources**) straight into your Data Warehouse or any Databases.

To further streamline and prepare your data for analysis, you can process and enrich raw granular data using Hevo's robust & built-in Transformation Layer without writing a single line of code!

Hevo is the fastest, easiest, and most reliable data replication platform that will save your engineering bandwidth and time multifold. Try our 14-day full access free trial today to experience an entirely automated hassle-free Data Replication!

Key Features of a Big Data Pipeline

Here are a few key features that allow a Big Data Pipeline to stand out:

Scalable Cloud-Based Architecture

Big Data Pipelines depend on the Cloud to allow users to automatically scale storage and compute resources down or up. While traditional pipelines aren't designed to handle multiple workloads concurrently, Big Data Pipelines house an architecture in which compute resources are distributed across independent clusters. Clusters can grow in size and number quickly and infinitely while maintaining access to the shared dataset. It is easier to predict the data processing time as new resources can be added instantly to support spikes in data volume.

Cloud-based Data Pipelines are elastic and agile. They let businesses take advantage of various trends as well. For instance, a company that expects a summer sales spike can easily add more processing power when required and doesn't have to plan weeks for this scenario. In the absence of elastic Data Pipelines, businesses can find it difficult to quickly adapt to trends.

Fault-Tolerant Architecture

Data Pipeline failure is a real possibility while the data is in motion. To mitigate the impacts on mission-critical processes, today's Data Pipelines provide a high degree of availability and reliability.

Big Data Pipelines are designed with a distributed architecture that alerts users and provides immediate failover in the event of application failure, node failure, and failure of certain other services.

If a node goes down, another node within then the cluster can immediately take over without needing major interventions.

Transforming High Volumes of Data

Since semi-structured and unstructured data make up around 80% of the data collated by companies, Big Data pipelines should be equipped to process large volumes of unstructured data (including sensor data, log files, and weather data, to name a few) and semi-structured data (like HTML, JSON, and XML files). Big Data Pipelines might have to migrate and unify data from sensors, apps, log files, or databases. Often data has to be standardized, enriched, filtered, aggregated and cleaned – all in near real-time.

Real-Time Analytics and Data Processing

Big Data Pipelines should transform, ingest, and analyze data in near real-time so that businesses can quickly find and act on insights. To start with, data needs to be ingested without delay from sources including IoT devices, databases, messaging systems, and log files. For databases, log-based Change Data Capture (CDC) serves as the gold standard for producing a stream of real-time data. Real-time Data Pipelines provide decision-makers with the latest data at their disposal.

Self-Service Management

Big Data Pipelines are constructed using tools that are linked to each other. From Data Warehouses and Data Integration platforms to Data Lakes and programming languages, teams can leverage various tools to easily maintain and develop Data Pipelines in a self-service and automated manner.

Traditional Data Pipelines usually need a lot of effort and time to integrate a vast set of external tools for data transfer, data extraction, and analysis.

Ongoing maintenance can be time-consuming and causes bottlenecks that introduce new complexities.

Big Data Pipelines also democratize access to data. Tackling all types of data is more automated and easier than before, allowing businesses to take advantage of data with less in-house personnel and effort.

Streamlined Data Pipeline Development

Big Data Pipelines are developed following the principles of DataOps, a methodology that brings various processes and technologies to shorten development and delivery cycles. Since DataOps deals with automating Data Pipelines across their entire lifecycle, pipelines can deliver data on time to the right stakeholder.

By aligning pipeline deployment and development, you make it easier to scale or change pipelines to include new data sources.

Exactly-Once Processing

Data duplication and data loss are a couple of common issues faced by Data Pipelines. Big Data Pipelines offer advanced checkpointing abilities that ensure no events are processed or missed twice. Checkpointing tracks the events processed and how far they go down different Data Pipelines.

Checkpointing meshes with the data replay feature that’s provided by various sources, letting you rewind to the correct spot if a failure occurs. For sources without a data replay feature, Data Pipelines with persistent messaging can checkpoint and replay data to ensure that it has been processed only once.

What Makes Hevo’s Data Replication Process Best in Class?

Replicating data can be a mammoth task without the right set of tools. Hevo’s Automated No-Code Platform empowers you with everything you need to have a smooth Data Collection, Processing, and Replication experience. Our platform has the following in store for you!

- **Blazing-fast Setup:** Straightforward interface for new customers to work on, with minimal setup time.
- **Built-in Connectors:** Support for 150+ Data Sources, including Power BI, Databases, SaaS Platforms, Files & More. Native Webhooks & REST API Connector available for Custom Sources.
- **Data Transformation:** Hevo provides a simple interface to perfect, modify, and enrich the data you want to transfer.

- **Faster Insight Generation**: Hevo offers near real-time data replication, so you have access to real-time insight generation and faster decision-making.
- **Schema Management**: Hevo can automatically detect the schema of the incoming data and map it to the destination schema.
- **Built to Scale:** Exceptional Horizontal Scalability with Minimal Latency for Modern-data Needs.
- **Live Support:** The Hevo team is available round the clock to extend exceptional customer support through chat, email, and support calls.

Importance of a Big Data Pipeline

Like various components of data architecture, Data Pipelines have also evolved to back Big Data. Big Data Pipelines are Data Pipelines that are built to accommodate one or more of the three key traits of Big Data.

The speed of Big Data makes it engaging to construct streaming Data Pipelines for Big Data. This allows data to be extracted and transformed in real-time to accomplish an action.

The volume of Big Data needs Data Pipelines to be scalable since the volume can vary over time. Realistically, there can be many Big Data events that occur concurrently or very close together, so the Big Data Pipeline should be able to scale to transform considerable volumes of data simultaneously.

The variety of Big Data needs Big Data Pipelines to have the ability to transform and identify data in many different formats – unstructured, structured, and semi-structured.

Components of a Big Data Pipeline

To run a Big Data Pipeline seamlessly here are the three components you'll need:

- Compute
- Messaging
- Storage

Compute

The compute component allows your data to get processed. Some common examples of Big Data Compute frameworks are as follows:

- Apache Flink
- Apache Spark
- Hadoop MapReduce
- Apache Heron
- Apache Storm

These compute frameworks are responsible for running the algorithms along with the majority of your code. For Big Data frameworks, compute components handle running the code in a distributed fashion, resource allocation, and persisting the results.

There are all different levels of sophistication on the compute side of a Data Pipeline. From the code standpoint, this is where you'll be spending the majority of your time. This can also take the blame for the misconceptions around compute being the only technology that is required.

Messaging

Messaging components are used to migrate events or knowledge from point A to point B in real time. Some commonly used instances of messaging platforms are as follows:

- Apache Pulsar
- Apache Kafka
- RabbitMQ (doesn't scale)

It is recommended to leverage messaging components when there is a need for real-time systems. These messaging frameworks can be used to extract and propagate a large amount of data. This propagation and extraction are pivotal to real-time systems because it solves mobility issues. From a coding and architecture perspective, you will be spending an equal amount of time on both. This will require you to gain a better understanding of user access patterns and use cases where the code can be leveraged.

Some technologies can be a mix of two or more components. However, there are important nuances that you need to be aware of. For instance, Apache Pulsar is primarily a messaging component but can also be used for storage and compute needs.

Storage

Storage components are responsible for the permanent persistence of your data. A couple of examples of simple storage are as follows:

- Cloud filesystems like Amazon S3
- HDFS
- Local Storage (doesn’t scale)

For simple storage needs, people would just dump their files into a directory. As it becomes slightly more difficult, you can begin using partitioning. This will put files in directories with particular names. A commonly used partitioning method is to use the date of the data as part of the directory name.

Types of Big Data Pipelines

Here are the different types of commonly used Big Data Pipelines in the marketplace:

- ETL
- ELT
- Reverse ETL

ETL

ETL is the most common Data Pipeline architecture, one that has been the standard for several decades. It takes raw data from various sources, transforms it into a single pre-defined format, and loads it to the sink – typically a Data Mart or an enterprise Data Warehouse.

Typical use cases for ETL Pipelines include the following:

- Collecting high volumes of data from different types of external and internal sources to offer a holistic view of business operations
- Extracting user data from several touchpoints to have all the information on customers in a single place (usually in the CRM system)
- Linking disparate datasets to allow deeper analytics.

The primary downside of the ETL architecture is that you need to rebuild your data pipeline every time business rules are modified. To address the problem, another approach gained prominence over the years — ELT.

ELT

ELT differs from ETL in the sequence of steps; loading takes place before transformation here. Due to this small change, instead of modifying large amounts of raw data, you first move it directly to a Data Lake or Data

Warehouse. This will allow you to structure and process your data as needed — at any moment, partially or fully, numerous times or just once.

ELT architecture would come in handy for the following use cases:

- Scenarios where the speed of Data Ingestion plays a key role.
- Situations where you aren't sure what you're going to do with the data and how would you go about transforming it.
- ELT architecture also comes in handy where large volumes of data are involved.

Reverse ETL

Your decision to transition to Operational Analytics using reverse ETL can become a turning point for your business. It is one step forward to become more data-driven and adapt your products and services to better suit your customers.

Operational Analytics can be facilitated if you either build or buy Reverse ETL. It is essential for your data management stack to make effective and factual decisions. This ensures efficient use of workflows, operational decisions, and systems that drive benefits to your business.

A dynamic trend as such isn't something new. As the volume of business data and time constraints grow, executive **decision-making is being pushed down to operational teams**. It is now more vital than ever to provide your employees and teams with accurate data to make operational decisions swiftly and implement them to reap the rewards.

Here are a few use cases of Reverse ETL:

- If the marketing team needs direct access to the frequency of client purchases within its marketing automation solution to create more powerful scenarios, reverse ETL can come in handy.
- Reverse ETL can also be leveraged by the customer success team if they wish to send tailored emails based on the usage of a product.

Building an indigenous reverse ETL solution can help you better tailor your solution to your business use case and needs, but customizing your reverse ETL solution can get unwieldy and requires a ton of assessment of your current business operations.

You may be hesitant, but spending extravagant amounts of money and labor on creating a reverse ETL solution would be heedless to know that similar functionality is offered if you buy reverse ETL.

You can buy Reverse ETL tools such as Hevo Activate, a fantastic- simpler & speedier solution that runs in the cloud. Hevo Activate can operationalize your business data from data warehouses like Snowflake, Amazon Redshift, Google BigQuery, and Firebolt to target destinations like CRM, Support Apps, Project Management Apps, and Marketing Apps with no difficulty.

Key Big Data Pipeline Architecture Examples

Organizations typically depend on three types of Data Pipeline transfers:

Streaming Data Pipeline

Real-time streaming dabbles with data moving onto further processing and storage from the moment it's generated, for instance, a live data feed. The stream processing engine can provide outputs from the Data Pipeline to data stores, CRMs, and marketing applications. From an implementation point of view, streaming data processing can leverage micro-batches that can be completed within short time windows.

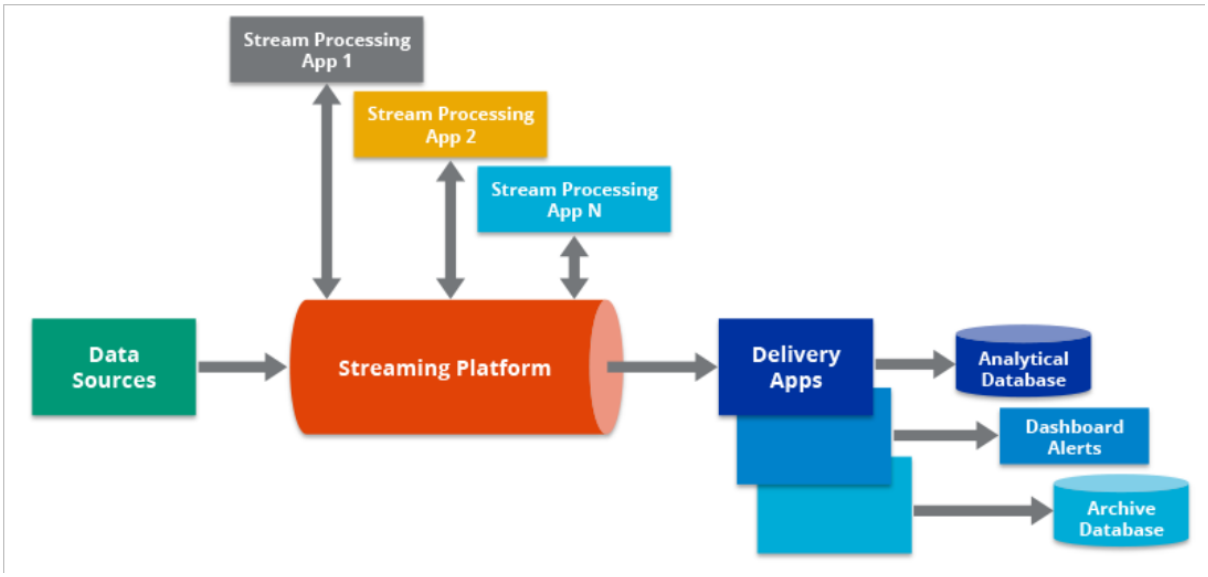


Image Source

Batch Data Pipeline

In batch processing, you simply collect data fragments in temporary storage and send them as a group on a schedule. You can execute this when access to this data isn't urgent, or there are intermittent latency issues to deal with.

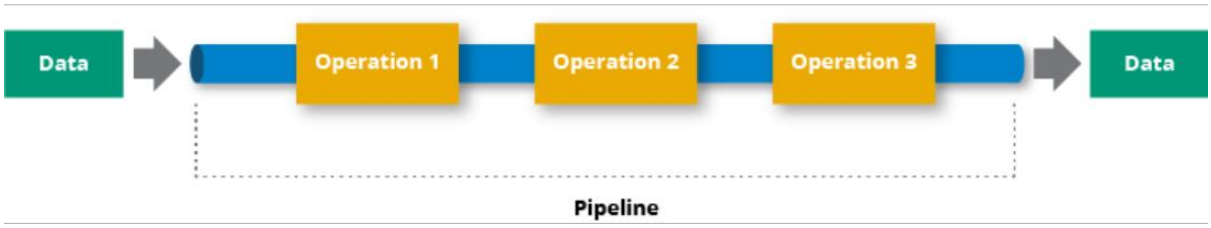


Image Source

Lambda Architecture

Lambda architecture tries to combine real-time and batch streaming by having them sync to storing data in the same file by regularly adding to it. This can be a little complicated to accomplish in-house since the real-time and batch components are independently coded and need to be in sync with the file writing.

However, AWS Lambda functions have a couple of limitations that you should be aware of. For instance, the Lambda timeout is 15 minutes, and the memory size limit is 10 GB. This method is recommended for short-term tasks that need a lot of memory per task. One key aspect of this architecture is that it encourages storing data in a raw format so that you can continuously run new Data Pipelines to rectify any code errors in prior pipelines or generate new data destinations that allow new types of queries.

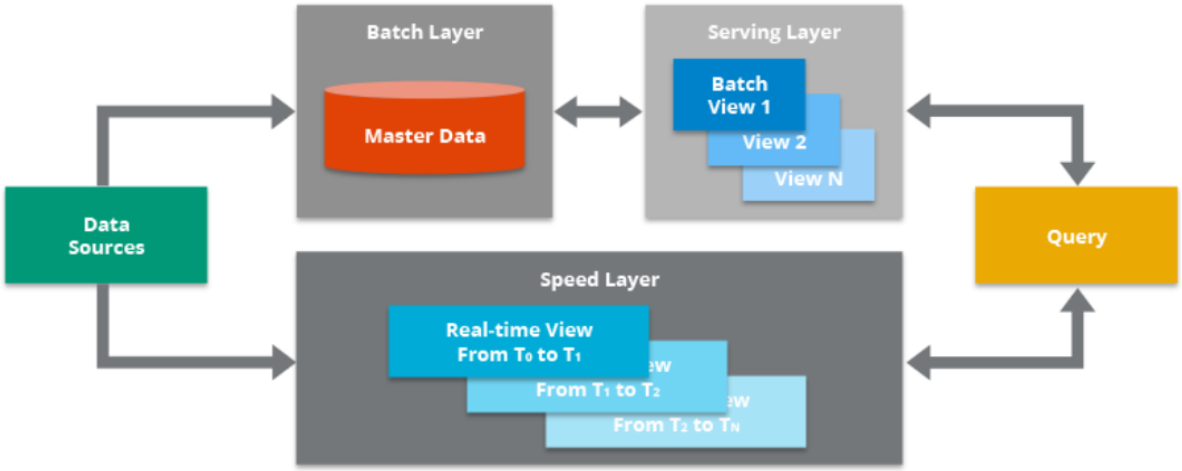


Image Source

Key Use Cases of Big Data Pipelines

There are a couple of industries that depend on Big Data more than others. These include:

- Construction companies track everything from the hours put into material costs.
- Brick-and-mortar and online retail stores that track consumer trends.
- Finance and banking institutions use Big Data to predict data trends and improve customer services.
- Healthcare organizations analyze voluminous data to find effective treatments.
- Organizations that dabble in media, entertainment, and communications leverage Big Data in various ways, such as offering real-time social media updates, improving connections between smartphone users, and improving HD media streaming.
- Colleges, schools, and universities measure student demographics, improve the student success rate, predict enrollment trends, and ascertain which educators excel.
- Energy companies leverage Big Data Pipelines to identify problems quickly to start finding solutions, handle workers during crises, and provide consumers with information that can help them use less energy.
- The government can leverage these pipelines in several ways, such as analyzing data to process disability claims, detect fraud, identify illnesses before they impact thousands of people, and track changes in the environment.
- Natural resource and manufacturing groups require Big Data Pipelines to align their activities to deliver the products that consumers need, lower overheads, and recognize potential dangers.

Best Practices to Build Big Data Pipelines

Here are a couple of best practices to keep in mind when building a Big Data Pipeline:

- They should be scalable so that they can be sized down or up based on the demands of system warrants. Sizing down is often neglected but is important for managing cloud hosts.
- These Data Pipelines should be highly extensible since this would allow them to incorporate as many things as possible.
- They should also be idempotent so that they can generate the same result no matter how many times the same initial data is used. Logging should take place at the completion and inception of every step.
- You should start simple, for instance, with serverless, with as few pieces as needed. You can then move to a full-blown pipeline or your deployment, only when the Return on Investment (ROI) is justifiable. A piece of helpful advice is to bootstrap with minimal investment in

the computational stage. You can even start a “compute-free” pipeline by executing computations by scheduling various cloud functions and SQL queries. This would get the whole pipeline ready faster, giving you ample time to handle your data strategy, along with data catalogs and data schemas.

Data Aggregation

Data aggregation is any process whereby data is gathered and expressed in a summary form. When data is aggregated, atomic data rows -- typically gathered from multiple sources -- are replaced with totals or summary statistics. Groups of observed aggregates are replaced with summary statistics based on those observations. Aggregate data is typically found in a data warehouse, as it can provide answers to analytical questions and also dramatically reduce the time to query large sets of data.

Data aggregation is often used to provide statistical analysis for groups of people and to create useful summary data for business analysis. Aggregation is often done on a large scale, through software tools known as *data aggregators*. Data aggregators typically include features for collecting, processing and presenting aggregate data.

Data aggregation can enable analysts to access and examine large amounts of data in a reasonable time frame. A row of aggregate data can represent hundreds, thousands or even more atomic data records. When the data is aggregated, it can be queried quickly instead of requiring all of the processing cycles to access each underlying atomic data row and aggregate it in real time when it is queried or accessed.

As the amount of data stored by organizations continues to expand, the most important and frequently accessed data can benefit from aggregation, making it feasible to access efficiently.

What does data aggregation do?

Data aggregators summarize data from multiple sources. They provide capabilities for multiple aggregate measurements, such as sum, average and counting.

Examples of aggregate data include the following:

- Voter turnout by state or county. Individual voter records are not presented, just the vote totals by candidate for the specific region.
- Average age of customer by product. Each individual customer is not identified, but for each product, the average age of the customer is saved.
- Number of customers by country. Instead of examining each customer, a count of the customers in each country is presented.

Data aggregation can also result in a similar effect to data anonymization -- as individual data elements with personally identifiable details are combined and replaced with a summary representing a group as a whole. An example of this is creating a summary that shows the aggregate average salary for employees by department, rather than browsing through individual employee records with salary data.

Aggregate data does not need to be numeric. You can, for example, count the number of any non-numeric data element.

Before aggregating, it is crucial that the atomic data is analyzed for accuracy and that there is enough data for the aggregation to be useful. For example, counting votes when only 5% of results are available is not likely to produce a relevant aggregate for prediction.

How do data aggregators work?

Data aggregators work by combining atomic data from multiple sources, processing the data for new insights and presenting the aggregate data in a summary view. Furthermore, data aggregators usually provide the ability to track data lineage and can trace back to the underlying atomic data that was aggregated.

Collection. First, data aggregation tools may extract data from multiple sources, storing it in large databases as atomic data. The data may be extracted from internet of things (IoT) sources, such as the following:

- social media communications;

- news headlines;
- personal data and browsing history from IoT devices; and
- call centers, podcasts, etc. (through speech recognition).

Processing. Once the data is extracted, it is processed. The data aggregator will identify the atomic data that is to be aggregated. The data aggregator may apply predictive analytics, artificial intelligence (AI) or machine learning algorithms to the collected data for new insights. The aggregator then applies the specified statistical functions to aggregate the data.

Presentation. Users can present the aggregated data in a summarized format that itself provides new data. The statistical results are comprehensive and high quality.

Data aggregation may be performed manually or through the use of data aggregators. However, data aggregation is often performed on a large-scale basis, which makes manual aggregation less feasible. Furthermore, manual aggregation risks accidental omission of crucial data sources and patterns.

Uses for data aggregation

Data aggregation can be helpful for many disciplines, such as finance and business strategy decisions, product planning, product and service pricing, operations optimization and marketing strategy creation. Users may be data analysts, data scientists, data warehouse administrators and subject matter experts.

Aggregated data is commonly used for statistical analysis to obtain information about particular groups based on specific demographic or behavioral variables, such as age, profession, education level or income.