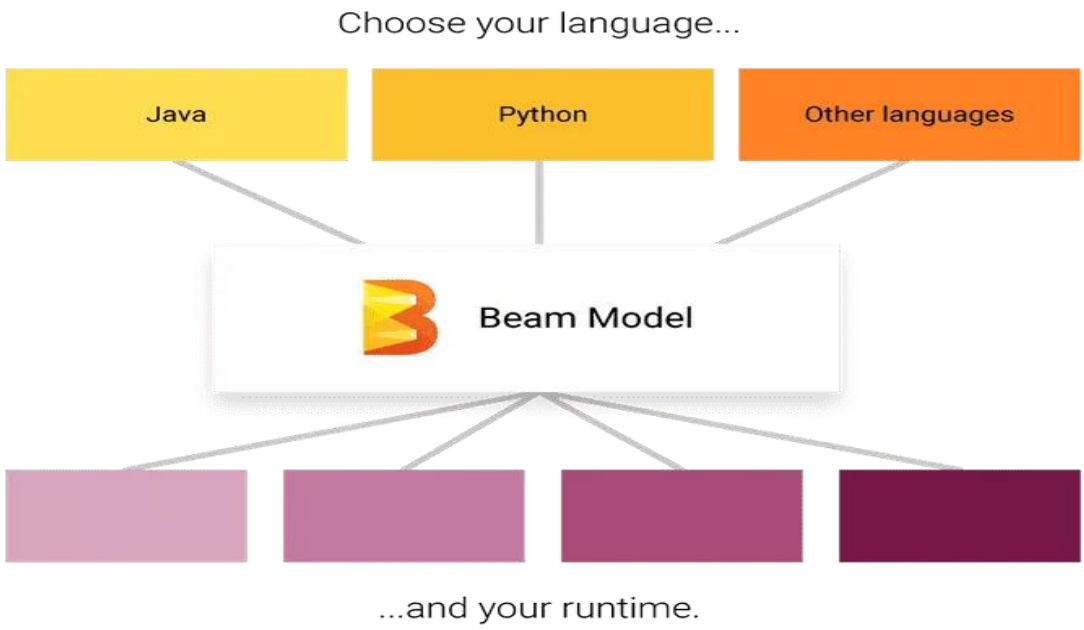
What is Apache Beam?



- Apache Beam (Batch + strEAM) is an open-source, unified model for defining batch and streaming data-**parallel processing pipelines.**
- Unified model as in beam exposes **single API abstraction layer** for both batch and stream processing jobs, unlike other frameworks with different APIs for batch and stream processing



Unique Features of Apache Beam

- Unified Model: Apache Beam provides a unified programming model for both batch and stream processing. This allows developers to write code that works for both scenarios, promoting code reuse and simplifying development.

- Portability: Beam pipelines can be executed on various processing engines, such as Apache Spark, Apache Flink, Google Cloud Dataflow, and more. This enables users to choose the most suitable execution engine for their use case.

- Parallel Processing: Beam processes data in parallel, taking advantage of the underlying processing engine's capabilities to distribute work across multiple nodes and cores, improving performance.

- Event Time Processing: Beam supports event time processing, allowing developers to handle out-of-order data based on event timestamps. This is crucial for scenarios where data arrives at different times.

- Windowing: Beam provides windowing capabilities for managing and processing data within specific time intervals or windows, facilitating tasks like sessionization and aggregations.

- Exactly-Once Processing: Beam supports exactly-once processing semantics, ensuring that each piece of data is processed exactly once, even in the presence of failures.

- Dynamic Scaling: Beam pipelines can dynamically scale up or down based on the incoming data volume, enabling efficient resource utilization.

- Backpressure Handling: Beam provides mechanisms to handle backpressure, which occurs when the rate of data production exceeds the rate of data processing. This ensures stability and prevents resource exhaustion.

- Extensibility: Beam allows users to create custom transformations, sources, and sinks, enabling integration with various data sources, APIs, and storage systems.

- State Management: Beam supports distributed state management, allowing developers to maintain and use stateful information across different processing stages.

- Checkpointing: For stream processing, Beam supports checkpointing, which enables fault tolerance by periodically saving the pipeline's state to a durable storage system.

Why it is important?

- The reason why Beam came into existence is that it can be used in parallel data processing tasks which can be fragmented into several smaller chunks of data which are processed in parallel and independently.

- The advantage of using beam SDK is that it can transform a dataset of any size whether the input data is finite i.e., came from a batch data source or an infinite data set from a streaming source of data.

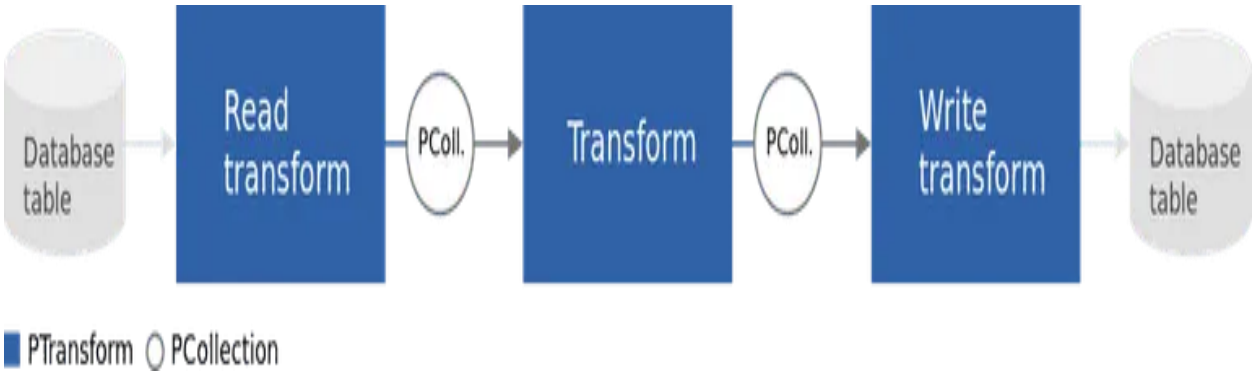
- Other benefits of using Beam SDK are that it uses the same class to the representation of both the bounded and unbounded data.

Beam SDKs and Runners

- Beam Runners translate the beam pipeline to the API compatible backend processing of your choice. Beam currently supports runners that work with the following backends.
- Apache Spark
- Apache Flink
- Apache Samza
- Google Cloud Dataflow
- Hazelcast Jet
- Twister2
- Direct Runner to run on the host machine, which is used for testing purpose.

Basic Concepts in Apache Beam

- Apache Beam has three main abstractions. They are



Pipeline

- A pipeline is the first abstraction to be created. It holds the complete data processing job from start to finish, including reading data, manipulating data, and writing data to a sink.
- Every pipeline takes in options/parameters that indicate where and how to run.

PCollection:

- A pcollection is an abstraction of distributed data.
- A pcollection can be bounded, i.e., finite data, or unbounded, i.e., infinite data.
- The initial pcollection is created by reading data from the source.
- From then on, pcollections are the source and sink of every step in the pipeline.

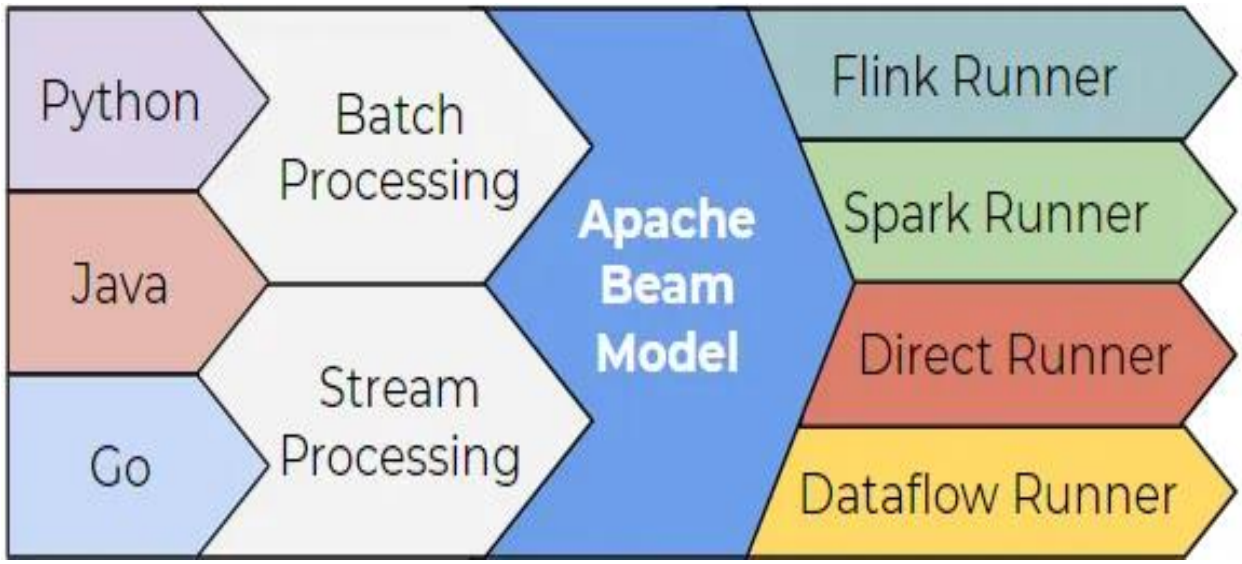
Transform

- A transform is a data processing operation.
- A transform is applied on one or more pcollections.
- Complex transforms have other transform nested within them.
- Every transform has a generic apply method where the logic of the transform sits in.

PipelineRunner

- A runner runs a Beam pipeline using the capabilities of your chosen data processing engine. Most runners are translators or adapters to massively parallel big-data processing systems, such as Direct Runner, Apache Spark or Apache Flink etc.

Apache Beam Model



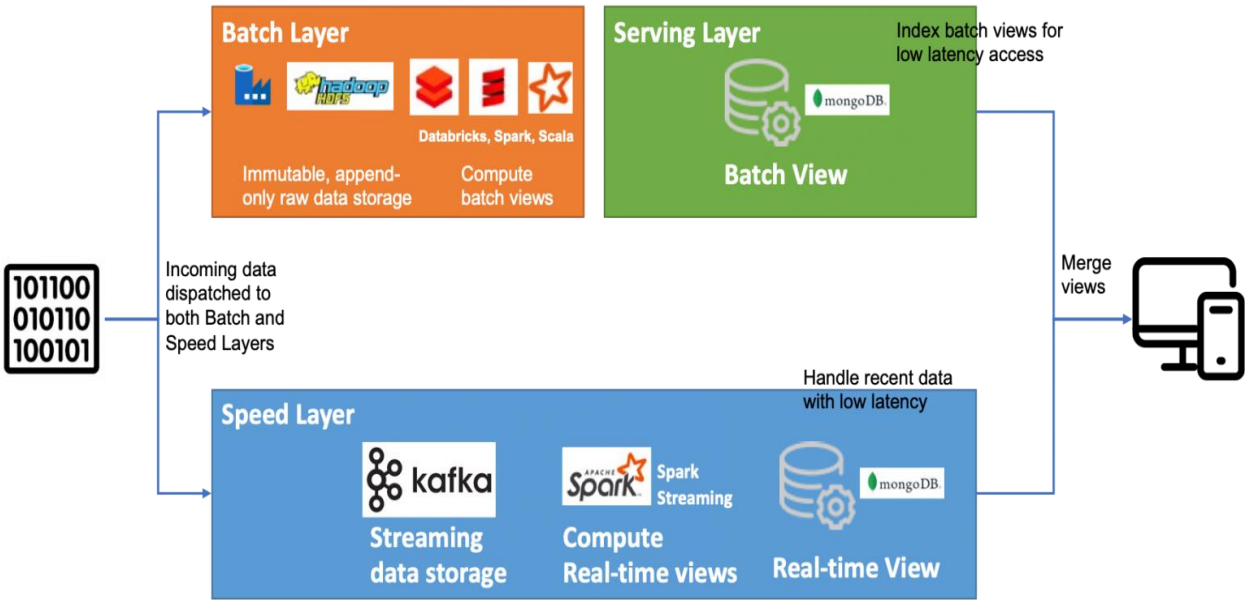
What is the Pipeline Development Life-cycle?

- There exist three significant stages while designing a pipeline, and that is as follows:
- Developing a pipeline - Designing a beam pipeline involves the information about how to determine a pipeline structure, how to choose that which transform must apply to the data, and how to determine the input and output data.

Following are the things while designing a pipeline -

- Where the input data is stored i.e., how many sets of input data it has, it also determines which kind of read transforms we will need to apply at the beginning of the pipeline.
- What does the input data look like i.e., the data can be a plain text or rows in a database table?
- What to do with the data, it involves how we change and manipulate our data and to determine how to build core transforms such as ParDo.
- Where should the data go i.e., what kind of write transforms we will need to apply at the end of the pipeline.

How Apache Beam Works and Architecture?



- Pipeline Definition: Developers define their data processing pipeline using the Apache Beam SDK. This includes specifying transformations, data sources, sinks, and other processing steps.
- Graph Representation: The pipeline definition is transformed into a directed acyclic graph (DAG) that represents the sequence of transformations and data flows.

- Execution Engine: The user specifies the execution engine where the pipeline will run. Apache Beam supports various execution engines like Apache Spark, Apache Flink, Google Cloud Dataflow, and more.
- Translation Layer: Apache Beam translates the pipeline's logical representation into executable code that is compatible with the chosen execution engine.
- Distributed Processing: The execution engine distributes the pipeline tasks across a cluster of machines, performing parallel processing and optimizations.
- Data Processing: The execution engine processes the data according to the defined transformations and logic, performing tasks like mapping, filtering, aggregation, and more.
- Windowing and Time Handling: If windowing is used, data is organized into windows based on time intervals. Event time processing and windowing are applied to ensure accurate data processing.
- Output: Processed data is sent to the specified output sinks, which could be databases, storage systems, APIs, or other data destinations
- Fault Tolerance: Apache Beam's execution engines ensure fault tolerance by managing checkpoints, maintaining state, and handling failures gracefully.
- Scaling: The execution engine can dynamically scale the pipeline based on incoming data volume and available resources, ensuring efficient resource utilization.
- Completion and Cleanup: Once the pipeline completes, resources are released, and cleanup tasks are performed.
- Monitoring and Metrics: Apache Beam provides monitoring and metrics capabilities to track the progress, performance, and health of the pipeline.

benefits of using Apache Beam

- It is a unified model for data processing pipelines.
- It can be used to process data on a variety of platforms, including Apache Spark, Apache Flink, and Google Cloud Dataflow.
- It provides a variety of features for data processing, such as batch processing, streaming processing, and machine learning.
- It is free to apply and open-source.

drawbacks of using Apache Beam:

- It can be complex to learn and use.
- It can be slow for some applications, especially those that use a lot of data.
- It is not as popular as some other data processing frameworks, such as Spark and Flink.

use cases of Apache Beam:

- Real-Time Analytics: Apache Beam is used to process and analyze streaming data in real time. It can be applied in scenarios like monitoring social media feeds, tracking user activities, and generating real-time insights.
- Batch ETL (Extract, Transform, Load): Beam enables efficient extraction, transformation, and loading of large volumes of batch data. It's commonly used to preprocess and clean data before storing it in data warehouses or databases.
- Event Time Processing: Apache Beam provides features for event time processing, which is critical for scenarios where events occur at different times but are processed in the correct order.
- IoT Data Processing: With the rise of Internet of Things (IoT), Beam can be used to ingest and process data from various sensors and devices in real time.
- Clickstream Analysis: For websites and applications, Beam can help analyze clickstream data to understand user behavior and make data-driven decisions to improve user experiences.
- Fraud Detection: Beam can be used to detect patterns and anomalies in streaming data, helping to identify potential fraudulent activities in real time.
- Financial Data Processing: Beam is applicable in the financial industry for processing and analyzing stock market data, transactions, and trading activities.
- Recommendation Systems: In e-commerce and entertainment, Beam can process user interaction data to generate personalized recommendations for products, movies, music, and more.
- Data Enrichment: Beam can enrich data by joining streams or batches of data with external datasets, providing additional context for analysis.
- Machine Learning Pipelines: Beam can be used to preprocess and transform data before feeding it into machine learning models. It supports data preparation steps like feature engineering and normalization.