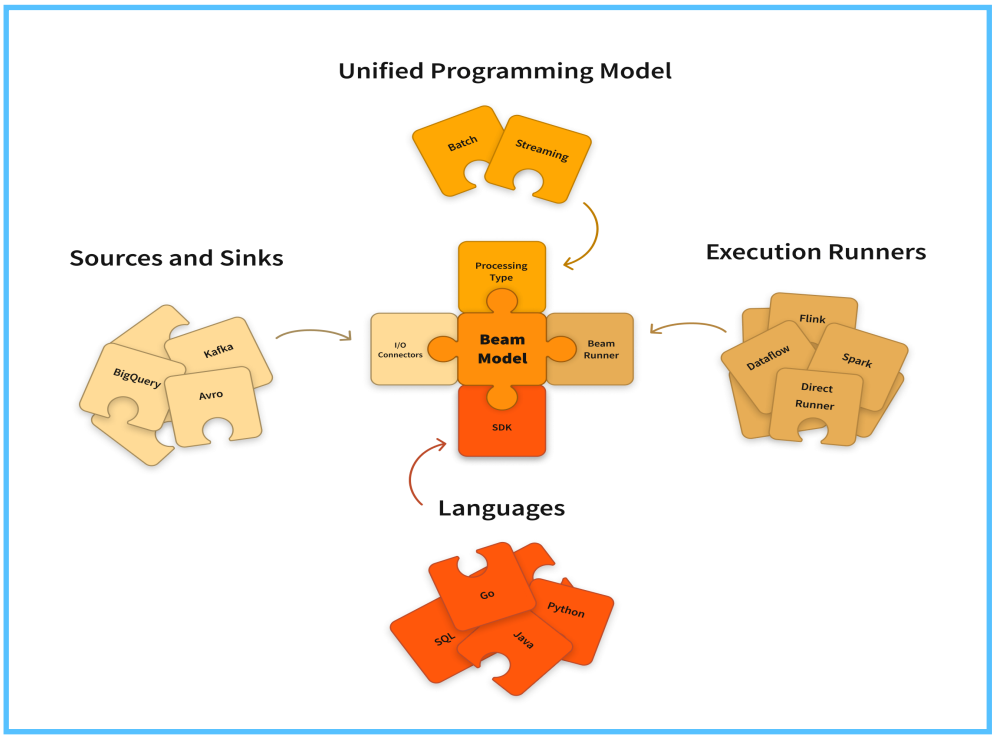
Apache Beam Overview

- Apache Beam (**B**atch + str**EAM**) is an open source, unified model for defining both batch and streaming data-parallel processing pipelines.
- It allows developers to write portable and expressive pipelines that can be executed on various distributed processing backends.



- Beam is particularly useful for **embarrassingly parallel** data processing tasks, in which the problem can be decomposed into many smaller bundles of data that can be processed independently and in parallel.
- Beam supports multiple language-specific SDKs like **Java, Python** and **Go** for writing pipelines in the Beam Model.
- Using one of the open source Beam SDKs, you build a program that defines the pipeline. The pipeline is then executed by one of

Beam's supported distributed processing back-ends, which include Apache Flink, Apache Spark, and Google Cloud Dataflow.

- Beam currently supports the following runners:
    - Direct Runner
    - Apache Flink Runner
    - Apache Nemo Runner
    - Apache Samza Runner
    - Apache Spark Runner
    - Google Cloud Dataflow Runner
    - Hazelcast Jet Runner
    - Twister2 Runner

History of Apache Beam

- Originally developed by **Google**, the Beam model was introduced to the Apache Software Foundation in 2016.
- Since then, it has undergone numerous improvements to enhance performance, and support for additional languages and runners.

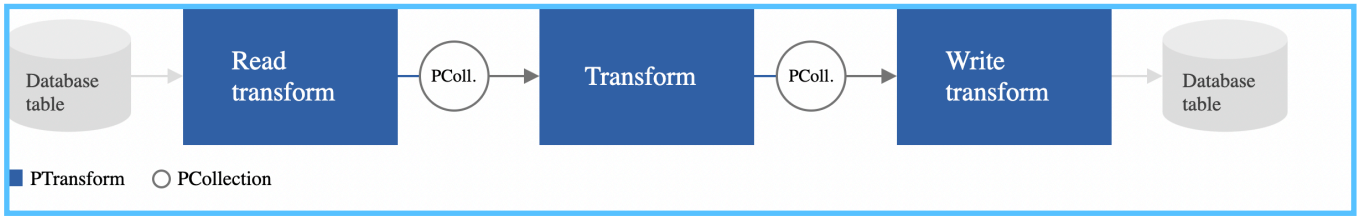
Features of Apache Beam

The unique features of Apache beam are as follows:

- **Unified -** Use a single programming model for both batch and streaming use cases.
- **Portable -** Execute pipelines in multiple execution environments. Here, execution environments mean different runners. Ex. Spark Runner, Dataflow Runner, etc

- **Extensible -** Write custom SDKs, IO connectors, and transformation libraries.
- **Open source -** Community-based development, and can read data from a variety of sources, including on-prem and cloud.
- **Connectors -** Read from and write to different systems, including Google Cloud services and third-party technologies such as Apache Kafka.

Key concepts in the Beam programming model



Pipeline -

- A Pipeline encapsulates your entire data processing task, from start to finish.
- This includes reading input data, transforming that data, and writing output data.
- All Beam driver programs must create a Pipeline. When you create the Pipeline, you must also specify the execution options that tell the Pipeline where and how to run.

PipelineRunner -

- Specifies where and how the pipeline should execute in Dataflow runner, Direct runner and Spark runner.

PCollection -

- A PCollection represents a distributed data set that your Beam pipeline operates on.

- The data set can be bounded, meaning it comes from a fixed source like a file, or unbounded, meaning it comes from a continuously updating source via a subscription or other mechanism.

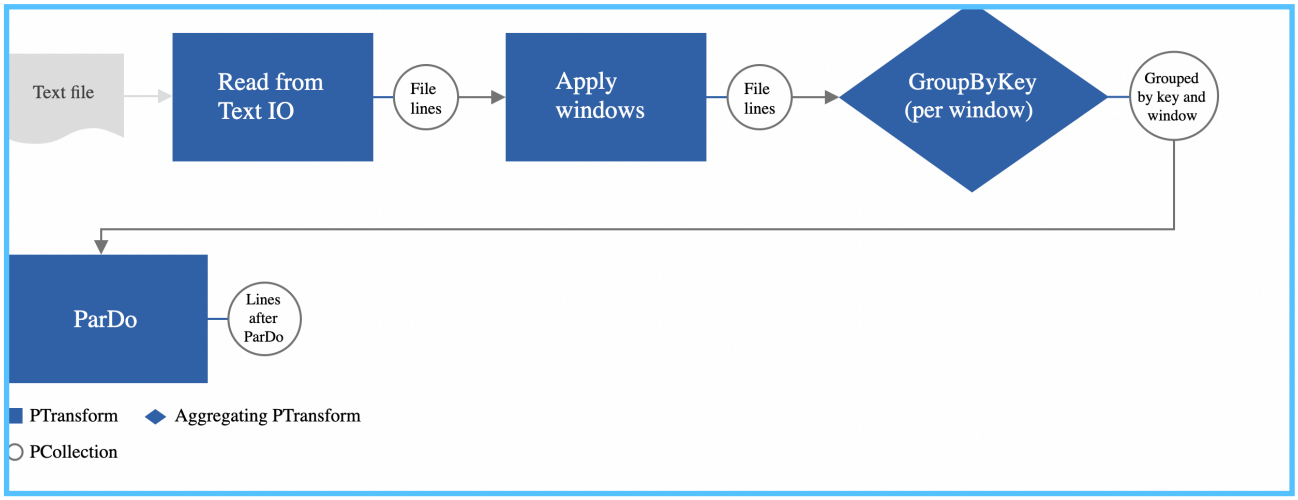
PTransform -

- A PTransform represents a data processing operation, or a step, in your pipeline.
- Every PTransform takes one or more PCollection objects as the input, performs a processing function that you provide on the elements of that PCollection, and then produces zero or more output PCollection objects.

I/O transforms -

- Beam comes with a number of “IOs” - library PTransforms that read or write data to various external storage systems.

A typical Beam driver program works as follows:



1. Create a pipeline object and Create an initial PCollection for pipeline data.
2. Apply necessary transformations using PTransforms to each PCollections.
3. Use IO’s to Read and Write data to external sources.

4. Run the pipeline using designated Pipeline Runner.

Advantages of Apache Beam

Unified Model

- Apache Beam aims to provide a unified model for batch and streaming data processing.
- This means that you can use the same Beam code to process data that is either coming in as a stream or that has already been collected into a batch.
- This can save you time and effort, as you don't need to learn two different sets of APIs.

Flexible Execution

- Apache Beam supports a variety of execution engines, including Apache Spark, Google Cloud Dataflow, and Apache Flink.
- This offers you the flexibility to choose the execution engine that best meets your needs.

Accessibility

- The Apache Beam project hosts a fantastic tutorial and execution environments for getting started with Beam quickly and testing different aspects of data flows.

Portable Code

- Theoretically, Apache Beam code can be run on any execution engine without modification.
- This means that you can develop your code once and then run it on any platform that supports Apache Beam.

- This can save you time and money, as you don't need to develop and maintain separate versions of your code for different platforms.

Scalable

- Apache Beam can scale to process large amounts of data.
- This is because Apache Beam uses a distributed architecture that can be scaled out to multiple machines.
- This can help you to process data more quickly and efficiently.

Extensible

- Apache Beam is extensible with a variety of plugins and libraries. This means that you can add new features and functionality to Apache Beam to meet your specific needs.
- For example, you can add support for new data sources or new data processing operations.

Disadvantages of Apache Beam

Flexible Execution

- The idea behind Beam is to decouple the logic of data processing from the data processing platform.
- The same job ought to be able to execute on any platform for which a Beam Runner is available.
- While this is an elegant idea, most organizations do not have the requirement to repurpose a processing job from one platform to another, and most jobs will require some degree of platform-specific tweaking to run optimally.

Portable Code

- The portability benefits of Beam come into doubt when it comes time to optimize a job for production, or to debug a job running insufficiently on a specific platform.
- A key step in both of these cases might be to dispense with the abstraction layer and implement directly on the platform to take advantage of that platform's specific capabilities.

Scalability

- There's nothing about Beam that improves scalability of the execution platform on which the Beam job runs.
- While it's nice that Beam jobs can scale, that scalability is entirely handled by the execution platform.

Extensibility

- While Apache Beam is extensible with a variety of plugins and libraries, and you can add new features and functionality to support your needs, the execution platforms supported by Beam.
- Apache Flink, Apache Spark, and Google Cloud Dataflow, are all similarly extensible.

Unified Model

- Beam's abstraction layer is well thought out and enables developers to reason about data processing instead of the execution platform specifics or details about whether the data is received in a batch or streaming fashion.
- Apache Beam also aims to provide a unified model for batch and streaming data processing under the idea that you can use the same code to process data that is coming in as a stream or a batch.

- While there is a benefit to the unification of batch and stream processing, other frameworks, such as Apache Flink, also aim to do this.