Using Apache Beam

April, 20th

Apache Beam

- Provides you with a place to run Apache Beam jobs on the GCP
- Offers the ability to create jobs based on **templates**
- No need to address common aspects of running jobs on a cluster:
  - load balancing
  - scaling number of workers for a job
- These tasks are done automatically for both **batch** and **streaming**

Apache BEAM – Batch + strEAM

- Evolution of Google Dataflow that separates the dataflow logic from the programming issues (language, runners etc)
- Unified model for both batch and stream data processing
- Programs can be executed in different processing frameworks (via runners) using a set of different IOs

Apache BEAM – Batch + strEAM

Why to use BEAM instead of only Hadoop, Spark, Flink, GCP Dataflow etc?

→ The Apache Beam framework provides an abstraction between your application logic and the big data ecosystem

Apache BEAM – Batch + strEAM

In the BEAM ecosystem:

- **DataSource**: can be batches, micro-batches or streaming data
- **SDK**: Java or Python
- **Runner**: Apache Spark, Apache Flink, Google Cloud Dataflow, among others and DirectRunner (runs locally)
- To build a BEAM logic: Pipeline, PCollection, PTransform, ParDO and DoFn

Apache BEAM – Batch + strEAM: Components

- **Pipeline**: encapsulates the workflow of your entire data processing tasks from start to finish. Includes:
  - reading input data
  - transforming that data (almost all data transformations are supported including database operations)
  - 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
- Beam can run independently of the Google Cloud Platform

Apache BEAM – Batch + strEAM: Components

- **PCollection**: distributed data set that your Beam pipeline operates on
- data may come from a fixed source like a file, or from a continuously updating source via a subscription or other mechanism

Apache BEAM – Batch + strEAM: Components

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

Apache BEAM – Batch + strEAM: Components

- **ParDo**: for generic parallel processing
- similar to the "Map" phase of a Map/Shuffle/Reduce-style algorithm
- a ParDo transform considers each element in the input PCollection, performs some processing function (your user code) on that element, and emits zero, one, or multiple elements to an output PCollection.

Apache BEAM – Batch + strEAM: Components

- **DoFn**: applies your logic in each element in the input PCollection and lets you populate the elements of an output PCollection
- to be included in your pipeline, it's wrapped in a ParDo PTransform.

For Wednesday April 22nd: Read and discuss

Polemical: Parallel and Distributed databases or MapReduce?

[Pavlo et al., SIGMOD 2009] A Comparison of Approaches to Large-Scale Data Analysis

[Dean and Ghemawat, CACM 2010] MapReduce: A Flexible Data Processing Tool

[Stonebraker et al., 2010] mapReduce and Parallel DBmss: friends or foes?