Universal Machine Learning with Apache Beam

Robert Bradshaw <robertwb@apache.org>
Maximilian Michels <mxm@apache.org>

Contents

Apache Beam

What is Apache Beam?

- A unified **batch** and **stream** distributed processing API
- A set of **SDK frontends**: Java, Python, Go, Scala, SQL, …
- A set of **Runners** which can execute Beam jobs into various backends: Local, Apache Flink, Apache Spark, Apache Gearpump, Apache Samza, Apache Hadoop, Google Cloud Dataflow, …

Beam Vision

Provide a **comprehensive portability framework** for data processing pipelines, one that allows you to write your pipeline once in your **language of choice** and run it with minimal effort on the **execution engine of choice**.

The Beam Model: A Simple Pipeline

```
PCollection<KV<String, Integer>> scores = input
    .apply(Sum.integersPerKey());
```

The Beam Model: Asking the Right Questions

What, Where, When, How

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
           .triggering(AtWatermark()
              .withEarlyFirings(AtPeriod(Duration.standardMinutes(1)))
              .withLateFirings(AtCount(1))
           .accumulatingFiredPanes())
    .apply(Sum.integersPerKey());
```

The Beam Model: Consistent across languages

```
scores = (input
    | WindowInto(FixedWindows(120)
                 trigger=AfterWatermark(
                     early=AfterProcessingTime(60),
                     late=AfterCount(1))
                 accumulation_mode=ACCUMULATING)
    | CombinePerKey(sum))
```

The Beam Model: IO is just [P]Transforms

```
input = root | ReadFromText("/path/to/text*") | Map(lambda line: ...)
scores = (input
    | WindowInto(FixedWindows(120)
                 trigger=AfterWatermark(
                     early=AfterProcessingTime(60),
                     late=AfterCount(1))
                 accumulation_mode=ACCUMULATING)
    | CombinePerKey(sum))
scores | WriteToText("/path/to/outputs")
```

The Beam Model: Execute on choice of Runner

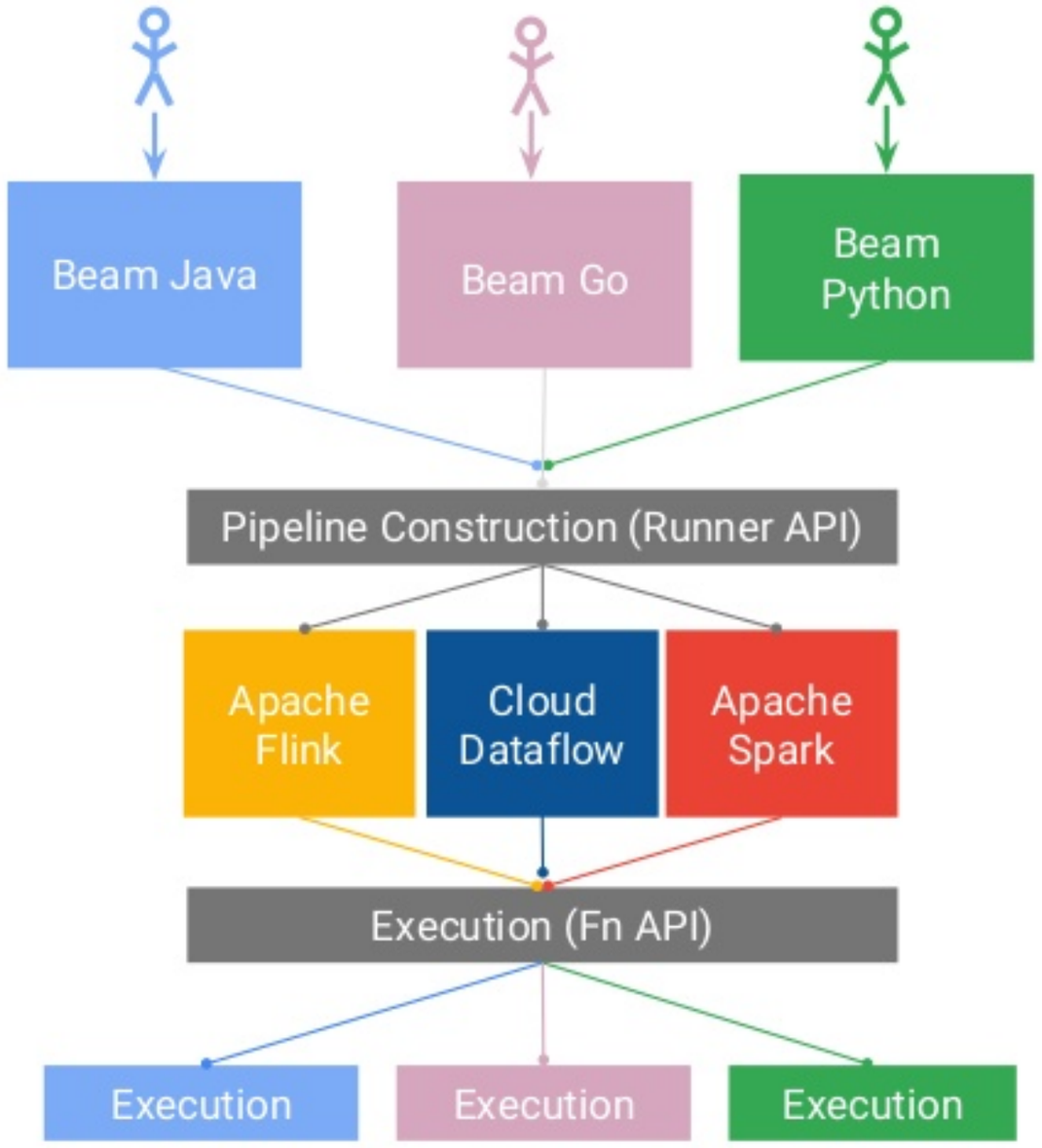
```
def pipeline(root):
  input = root | ReadFromText("/path/to/text*") | Map(lambda line: ...)
  scores = (input
      | WindowInto(FixedWindows(120)
                   trigger=AfterWatermark(
                       early=AfterProcessingTime(60),
                       late=AfterCount(1))
                   accumulation_mode=ACCUMULATING)
      | CombinePerKey(sum))
  scores | WriteToText("/path/to/outputs")

MyRunner().run(pipeline)
```

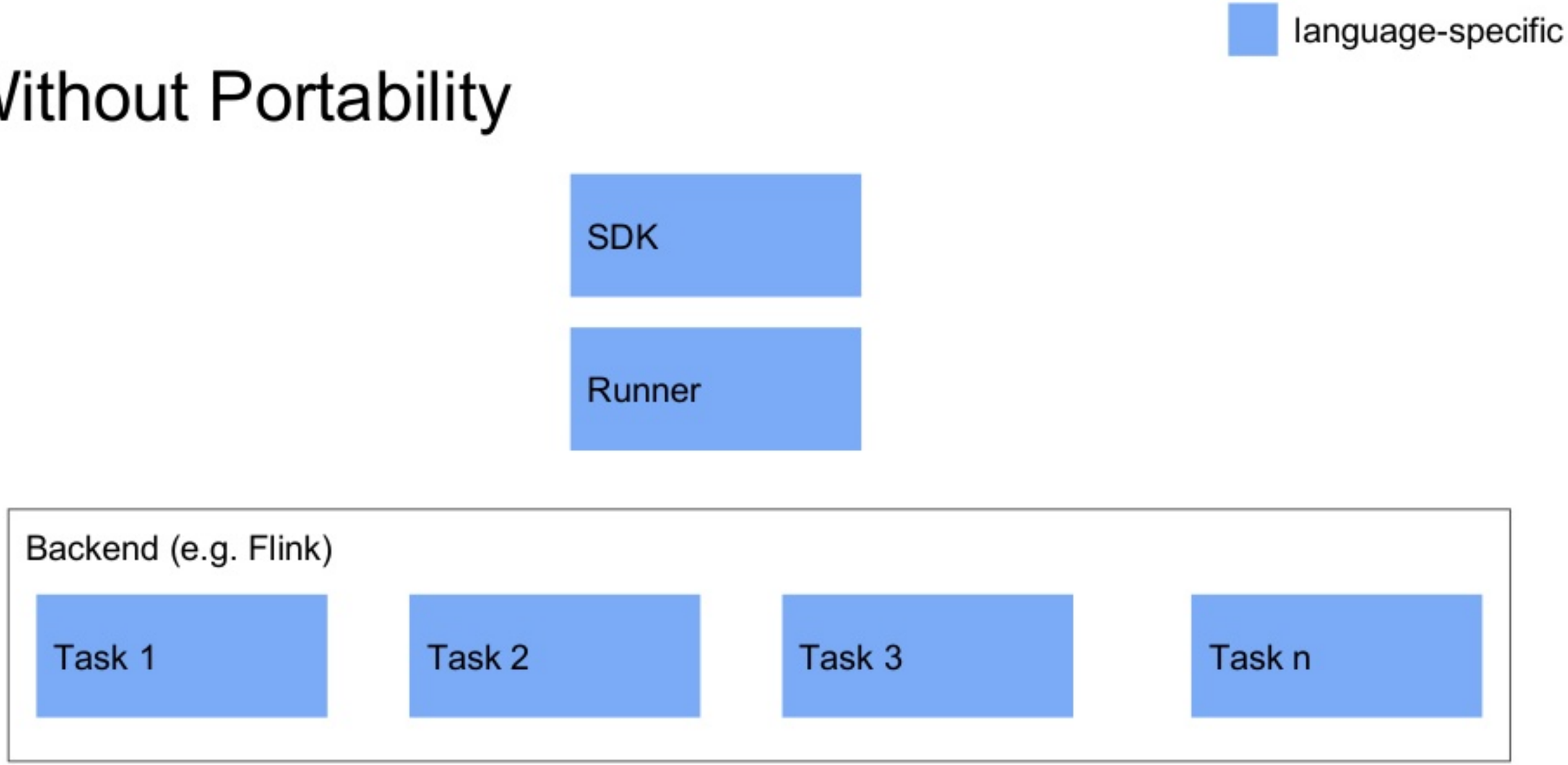
Portability

Realizing the Beam Vision

- Data processing frameworks usually only support a single language (e.g. Java)
- Portability in Beam means Pipelines can be written and executed in any supported SDK (Java/Python/Go)
- Pipelines also contain language-specific code (e.g. map/reduce functions)
- Libraries of the language can be used (!)

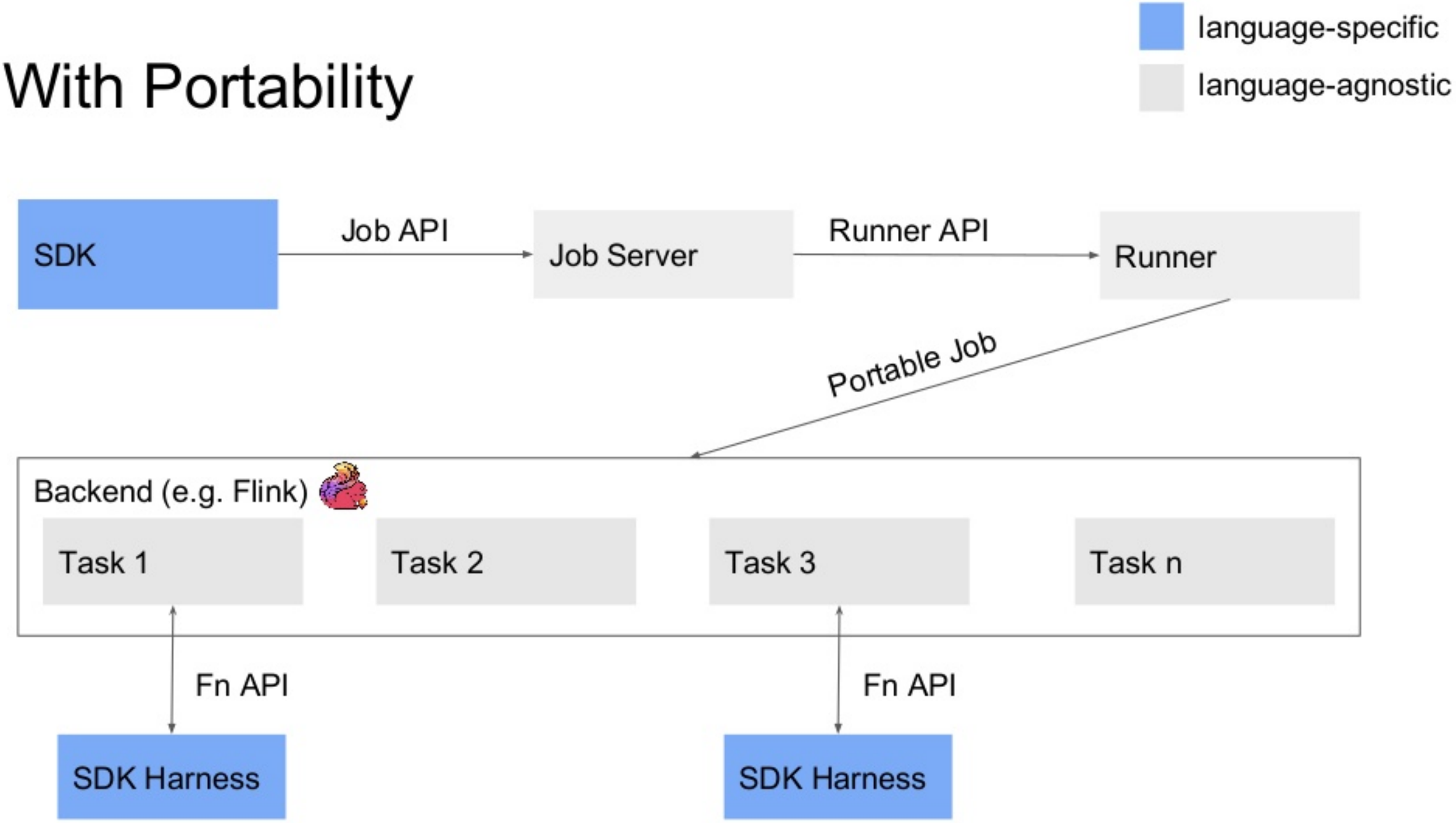


Without Portability



All components are tight to a single language

With Portability



TFX

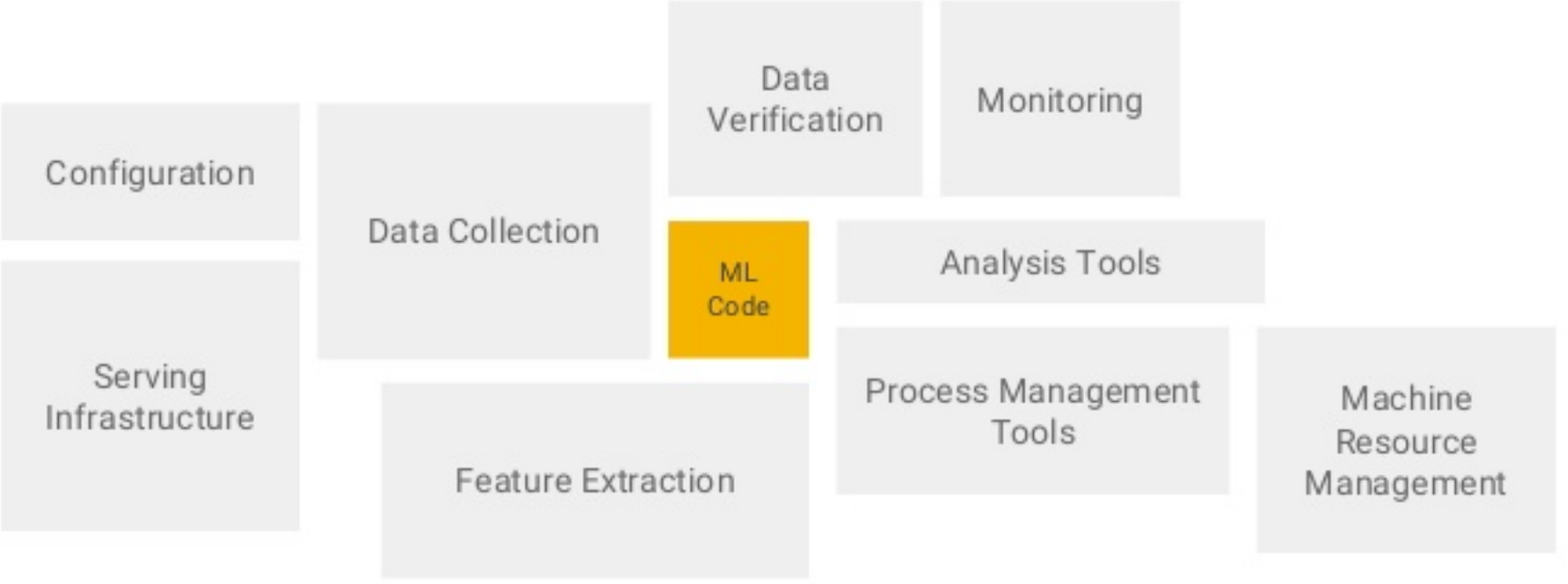
TFX: TensorFlowExtended

To do Machine Learning at scale, in addition to the actual ML...

ML
Code

TFX: TensorFlowExtended

...you have to worry about so much more



TFX: TensorFlowExtended

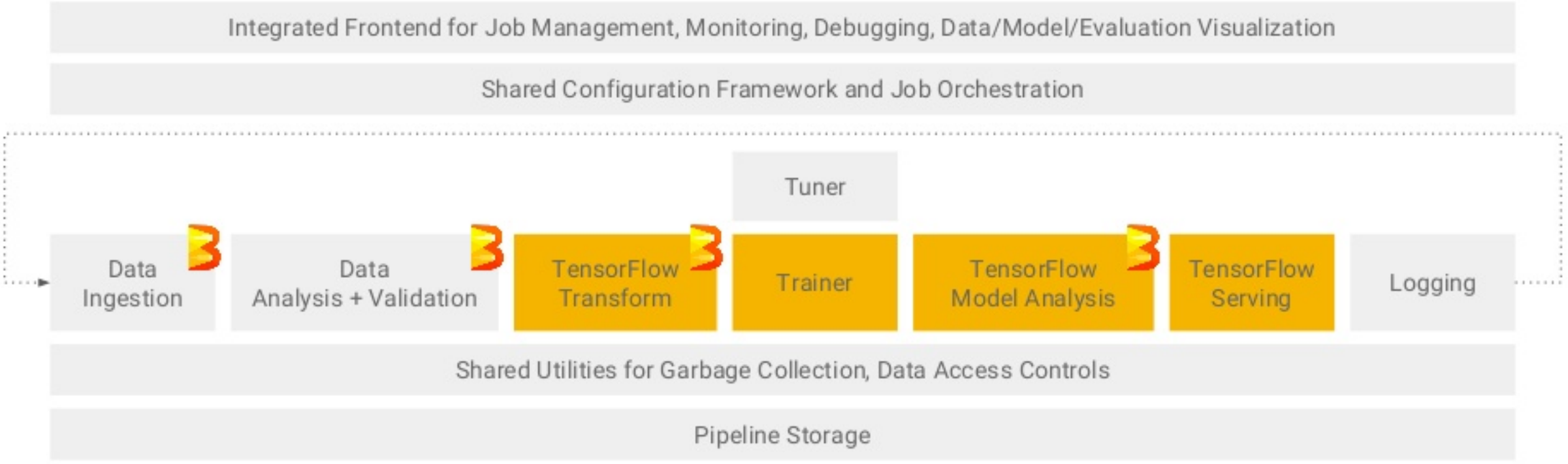
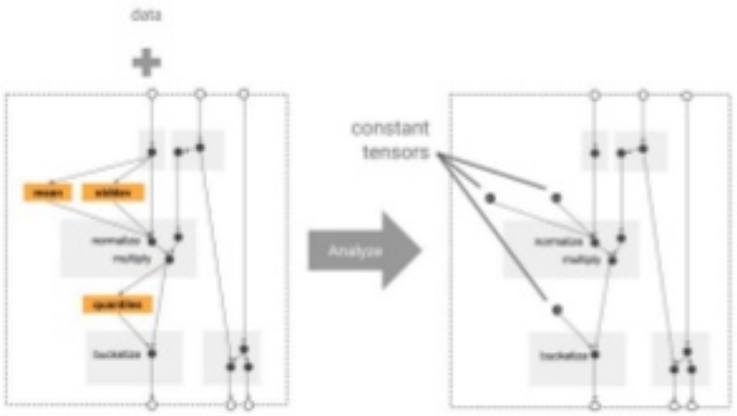


Figure 1: High-level component overview of a machine learning platform.

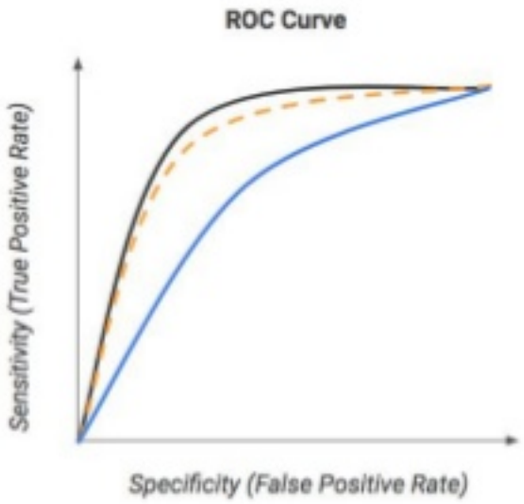
TFX: TensorFlowExtended

+

- Tensorflow Transform
  - preprocess/transform model training data
  - feature extraction and normalization
  - statistics and validation
  - avoid training/serving skew



- **Tensorflow Model Analysis**
  - analyze performance of Tensorflow models
  - slice and dice across population subsets
  - discover and investigate bias (see ml-fairness.com)



Demo

https://github.com/robertwb/model-analysis/blob/flink-demo/examples/chicago_taxi/chicago_taxi_tfma_local_playground.ipynb

Thank you

- Check out the Beam website
  - beam.apache.org
  - Documentation
  - Examples
- Mailing lists
  - user@beam.apache.org
  - dev@beam.apache.org
- Join the ASF Slack channel #beam