

CS 327E Class 5

October 7, 2019

No Quiz Today

Milestone 4 Feedback

Did you run into any major obstacles with the assignment?

- A. My group had problems identifying entity types.
- B. My group had problems decomposing large tables.
- C. A and B.
- D. My group did **not** face any major problems.

Beam/Dataflow Setup

<https://github.com/cs327e-fall2019/snippets/wiki/Beam-Dataflow-Setup-Guide>

Beam/Dataflow Setup Outcome

Did you successfully complete your setup?

- A. Yes, the Wordcount jobs ran without errors.
- B. No, I got stuck during the setup and need help.
- C. I'm still setting things up and need more time to finish.

Dataflow Concepts

- A system for processing arbitrary computations on large amounts of data
- Can process batch data and streaming data using the same code
- Uses Apache Beam, an open-source programming model
- Designed to be very scalable, millions of QPS

Apache Beam Concepts

- A model for describing data and data processing operations:
 - `Pipeline`: a data processing task from start to finish
 - `PCollection`: a collection of data elements
 - `Transform`: a data transformation operation
- SDKs for Java, Python and Go
- Executed in the cloud on Dataflow, Spark, Flink, etc.
- Executed locally with Direct Runner for dev/testing

Beam Pipeline

- Pipeline = A directed acyclic graph where the nodes are the `Transforms` and the edges are the `PCollections`
- General Structure of a Pipeline:
 - Reads one or more data sources as input `PCollections`
 - Applies one or more `PTransforms` on `PCollections`
 - Outputs resulting `PCollection` as one or more data sinks
- Executed as a single unit
- Run in batch or streaming mode

PCollection

- `PCollection` = A collection of data elements, either bounded or unbounded
- Elements can be made up primitive and complex types
- `PCollections` are distributed across machines
- `PCollections` are immutable
- Created from a data source or a `PTransform`
- Written to a data sink or passed to another `PTransform`

PTransform

All operations on data in beam are different kinds of PTransforms

- Element-wise:
 - maps 1 input to (1, 0, many) outputs
 - Examples: `ParDo`, `Map`, `FlatMap`
- Aggregation:
 - reduces many inputs to (1, fewer) outputs
 - Examples: `GroupByKey`, `CoGroupByKey`
- Composite: combines element-wise and aggregation
 - `GroupByKey` \rightarrow `ParDo`

PTransform Properties

- Serializable
- Parallelizable
- Idempotent

ParDo

- ParDo = “Parallel Do”
- Maps 1 input to (1, 0, many) outputs
- Takes as input a `PCollection`
- Applies the user-defined `ParDo` to the input `PCollection`
- Outputs results as a new `PCollection`
- Typical usage: filtering, formatting, extracting parts of data, performing computations on data elements

Hello World Example

```
1 import apache_beam as beam
2 from apache_beam.io import ReadFromText
3 from apache_beam.io import WriteToText
4
5 # DoFn to perform on each element in the input PCollection.
6 class ComputeWordLengthFn(beam.DoFn):
7     def process(self, element):
8         words = element.strip().split(' ')
9         result_list = []
10        for word in words:
11            result_list.append((word, len(word)))
12        return result_list
13
14 # Create a Pipeline using a local runner for execution.
15 with beam.Pipeline('DirectRunner') as p:
16
17     # create a PCollection from the file contents.
18     in_pcoll = p | 'Read' >> ReadFromText('input.txt')
19
20     # apply a ParDo to the PCollection
21     out_pcoll = in_pcoll | beam.ParDo(ComputeWordLengthFn())
22
23     # write PCollection to a file
24     out_pcoll | 'Write' >> WriteToText('output.txt')
```

Hello World Example

```
1 import apache_beam as beam
2 from apache_beam.io import ReadFromText
3 from apache_beam.io import WriteToText
4
5 # DoFn to perform on each element in the input PCollection.
6 class ComputeWordLengthFn(beam.DoFn):
7     def process(self, element):
8         words = element.strip().split(' ')
9         result_list = []
10        for word in words:
11            result_list.append((len(word), word))
12        return result_list
13
14 # Create a Pipeline using a local runner for execution.
15 with beam.Pipeline('DirectRunner') as p:
16
17     # create a PCollection from the file contents.
18     in_pcoll = p | 'Read' >> ReadFromText('input.txt')
19
20     # apply a ParDo to the PCollection
21     word_pcoll = in_pcoll | 'ParDo' >> beam.ParDo(ComputeWordLengthFn())
22
23     # apply GroupByKey to the PCollection
24     out_pcoll = word_pcoll | 'GroupByKey' >> beam.GroupByKey()
25
26     # write PCollection to a file
27     out_pcoll | 'Write' >> WriteToText('output.txt')
```

Hands-on Exercises

```
git clone https://github.com/cs327e-fall2019/snippets.git
```

Best Practices:

1. Know basic UNIX commands (e.g. `ls`, `cp`, `mv`, `rm`, etc.)
2. Start with some initial working code. See [snippets repo](#) for working examples.
3. Test and debug **each** new `PTransform` block before adding more logic.
4. Write temporary and final `PCollections` to log files.
5. If you get stuck, go to OHs. If you can't make OHs, make an appointment with one of the TAs.
6. Start assignments **early**. The Beam Python documentation is sparse and learning Beam requires *patience*, *perseverance*, and *experimentation*.

Milestone 5

- 1) Requirements: [assignment sheet](#)
- 2) Data Modeling Questions: [sign-up sheet](#)
- 3) Beam Setup Questions: [sign-up sheet](#)