No Quiz Today
Milestone 4 Feedback

Did you run into any major issues 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 big problems.
Beam + Dataflow Setup

https://github.com/cs327e-spring2020/snippets/wiki/Beam--&-Dataflow-Setup
Beam/Dataflow Setup Outcome

Did you successfully complete your setup?

A. Yes, the Wordcount jobs ran without errors.
B. No, I got stuck somewhere 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 of 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 → ParDo
PTransform Properties

- Serializable
- Parallelizable
- Idempotent
ParDo Transform

- 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
```python
import apache_beam as beam
from apache_beam.io import WriteToText
import logging

class MultiplyDoFn(beam.DoFn):
  def process(self, element):
    return [element * 10]

def run():
  PROJECT_ID = 'cs327e-sp2020'  # change to your project id
  options = {
    'project': PROJECT_ID
  }
  opts = beam.pipeline.PipelineOptions(flags=[], **options)

  p = beam.Pipeline('DirectRunner', options=opts)

  in_pcoll = p | beam.Create([1, 2, 3, 4, 5])

  out_pcoll = in_pcoll | 'Multiply' >> beam.ParDo(MultiplyDoFn())

  out_pcoll | 'Write results' >> WriteToText('multiplied_numbers.txt')

  result = p.run()
  result.wait_until_finish()

if __name__ == '__main__':
  logging.getLogger().setLevel(logging.INFO)
  run()
```
```python
import apache_beam as beam
from apache_beam.io import WriteToText
import logging

class SplitIntoWordsDoFn(beam.DoFn):
    def process(self, element):
        words = element.split()
        return [words]

def run():
    PROJECT_ID = 'cs327e-sp2020'  # change to your project id

    options = {
        'project': PROJECT_ID
    }
    opts = beam.pipeline.PipelineOptions(flags=[], **options)

    p = beam.Pipeline('DirectRunner', options=opts)

    in_pcoll = p | beam.Create(['Hello Beam', 'This is awesome!'])

    out_pcoll = in_pcoll | 'Split Words' >> beam.ParDo(SplitIntoWordsDoFn())

    out_pcoll | 'Write results' >> WriteToText('split_words.txt')

    result = p.run()
    result.wait_until_finish()

if __name__ == '__main__':
    logging.getLogger().setLevel(logging.ERROR)
    run()
```
GroupByKey Transform

- Takes an input PCollection where each element is a (key, value) pair
- Groups the values by unique key
- Produces an output PCollection where each element is a (key, list(value)) pair

`('Nicole', '100 Avenue A')`
`('Erik', '21 Guadalupe')`
`('Sameer', '7071 Hamilton')`
`('Nicole', '200 Avenue B')`

GroupByKey

`('Nicole', ['100 Avenue A', '200 Avenue B'])`
`('Erik', ['21 Guadalupe'])`
`('Sameer', '7071 Hamilton')`
import apache_beam as beam
from apache_beam.io import WriteToText
import logging

class ExtractFirstLetterDoFn(beam.DoFn):
    def process(self, element):
        tuple = (element[0], element)
        return [tuple]

def run():
    PROJECT_ID = 'cs327e-sp2020' # change to your project id

    options = {
        'project': PROJECT_ID
    }
    opts = beam.pipeline.PipelineOptions(flags=[], **options)

    p = beam.Pipeline('DirectRunner', options=opts)

    in_pcoll = p | beam.Create(['apple', 'ball', 'car', 'bear', 'cheetah', 'ant'])

    out_pcoll = in_pcoll | 'Extract' >> beam.ParDo(ExtractFirstLetterDoFn())

    grouped_pcoll = out_pcoll | 'Grouped' >> beam.GroupByKey()

    grouped_pcoll | 'Write results' >> WriteToText('grouped_letters.txt')

    result = p.run()
    result.wait_until_finish()

if __name__ == '__main__':
    logging.getLogger().setLevel(logging.INFO)
    run()
Hands-on Exercises

git clone https://github.com/cs327e-spring2020/snippets.git
Practice Problem 1

Run `Student_beam1.py` from `college_modeled.ipynb`
iClicker Question 1

How many records are in the resulting Student_beam table?

A. 0
B. 12
C. 15
Practice Problem 2

1. Save `Student_beam1.py` as `Student_beam2.py`
2. Fix the logic such that the Student’s status field does not get dropped from the output.
3. Run `Student_beam2.py` from `college_modeled.ipynb`
iClicker Question 2

How many records are in the resulting `Student_beam table`?

A. 0
B. 12
C. 15
Writing Beam Code:

1. Start with a working code sample.
2. Test and debug one Transform block at a time.
3. Write temporary and final PCollections to log files.
4. Start assignment early. The Beam Python documentation is sparse and learning Beam requires patience, perseverance, and experimentation.
5. If you get stuck, go to OHs. If you can’t make OHs, make an appointment with the TAs.
Milestone 5

http://www.cs.utexas.edu/~scohen/milestones/Milestone5.pdf