Should batch-producing DoFns return Iterables?

bhulette@apache.org
04/13/2021

Background

In Beam Python, `DoFn.process` implementations are expected to return iterables:

```
class DoubleDoFn(beam.DoFn):
  def process(self, element):
    yield element
```

This enables DoFn authors to easily create high-fanout DoFns if they need to, but it can also lead to surprising behavior if a DoFn author is unaware of this, or simply forgets, and writes a DoFn that returns an element:

```
class DoubleDoFn(beam.DoFn):
  def process(self, element):
    return element
```

If `element` is not itself iterable, then this will raise an execution-time error when the OutputProcessor tries to iterate over it. If element is iterable, then the OuptutProcessor will iterate over it, implicitly turning the ParDo into a FlatMap.

In practice this quirk doesn't seem to have caused significant issues, likely for a couple of reasons:

- Most end-users don't create DoFns directly - instead using beam.Map, or composites included in Beam or other libraries (e.g. TFX).
- DoFn authors that mistakenly **`return`** can quickly detect the issue in testing and remedy it.

Batch-producing DoFns

With s.apache.org/batched-dofns we are adding the ability to create DoFns that output objects representing batches of multiple logical elements. It's an open question whether these DoFns should always return an iterable of batches (which might contain just a single batch), or if they should always return a single batch.

Always return an iterable of batches

```
class ReaderDoFn(beam.DoFn):
  @beam.DoFn.yields_batches
  def process(self, filename):
    for batch in read_file(filename):
      yield batch

class TransformDoFn(beam.DoFn):
  def process_batch(self, batch: np.ndarray) -> np.ndarray:
    yield batch * 2
```

- Pro: Consistent with current behavior of process method.
- Con: A batch-producing DoFn is more susceptible to the issue noted above, as the batch type will often be iterable.

Always return a single batch

```
class ReaderDoFn(beam.DoFn):
  @beam.DoFn.returns_batches # This decorator would be changed
  def process(self, filename):
    …
    return batch_with_all_contents

class TransformDoFn(beam.DoFn):
  def process_batch(self, batch: np.ndarray) -> np.ndarray:
    return batch * 2
```

- Pro: Simplicity. A batch-producing DoFn is already a complex construct, and a batch already contains multiple elements.
- Con: Loss of flexibility. a batch-producing DoFn will **never** be able to fan out multiple batches. If there's a use-case that would benefit from yielding multiple batches per input, the batches will need to be combined, which might incur a cost.
    - An example of a batch-producing DoFn that fans out would be a DoFn that reads a Parquet file. A single file might contain multiple row groups, and each could be represented by a single pyarrow RecordBatch instance.

Decision: Always return an iterable of batches

At this time I'm inclined to always return an iterable of batches to keep the API as flexible as possible, but I'm open to changing this based on feedback from the community.