Resolving Out of memory errors in Beam pipelines

Guide

Presented by:
Zeeshan Khan
Cloud Data Engineer
Google Cloud Consulting
Dataflow Memory usage

**Worker operational memory**

OS and system processes. Less than 1 GB

<table>
<thead>
<tr>
<th>DoFn 1 memory usage</th>
<th>DoFn 2 memory usage</th>
<th>DoFn 3 memory usage</th>
<th>DoFn 4 memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK process memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worker operational memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beam SDK Process</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dataflow Memory usage

**SDK process memory**

In memory objects and data. Shared across DoFns.

- Side inputs
- ML models
- In memory singeltons
- Python objects created with the apache_beam.utils.shared module
Dataflow Memory usage

DoFn Memory Usage

DoFn is an Apache Beam SDK class that defines a distributed processing function.

<table>
<thead>
<tr>
<th>DoFn 1 memory usage</th>
<th>DoFn 2 memory usage</th>
<th>DoFn 3 memory usage</th>
<th>DoFn 4 memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- SDK process memory
- Worker operational memory
- Beam SDK Process
Dataflow Memory usage

Java: 1 SDK process per worker
Python: 1 SDK process per vCPU

<table>
<thead>
<tr>
<th>DoFn 1 memory usage</th>
<th>DoFn 2 memory usage</th>
<th>DoFn 3 memory usage</th>
<th>DoFn 4 memory usage</th>
</tr>
</thead>
</table>

SDK process memory

Worker operational memory

Beam SDK Process
Best practices for memory efficient Beam pipelines
1: Use Apache Beam built-in I/O connectors for reading files

# The input PCollection of Strings.
input = ...

class DoCompute(beam.DoFn):
    def process(self, element):
        textfile = open("/file_path/test.txt", 'r')
        lines = textfile.read().splitlines()
        return [lines]

output = input | beam.ParDo(DoCompute())
2 : Redesign operations when using GroupByKey PTransforms

```java
PCollection<KV<String, String>> input = ...;

PCollection<KV<String, Iterable<String>>> output =
    input.apply(GroupByKey.<String, String>create());
```
3) Reduce ingress data from external sources

Recommended to batch requests to external storage systems and API.

Reduce batch size to reduce the amount of data returned for each call.

```python
input | beam.GroupIntoBatches(3)
```
4) Share objects across threads (cache)

Method of the DoFn:
- Setup
- StartBundle
- Process
- FinishBundle
- Teardown

<table>
<thead>
<tr>
<th></th>
<th>DoFn 1 memory usage</th>
<th>DoFn 2 memory usage</th>
<th>DoFn 3 memory usage</th>
<th>DoFn 4 memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDK process memory</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Worker operational memory</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sharing an in-memory data object across DoFn instances can improve space and access efficiency.

Pass data objects as singleton to share it across DoFn within an Beam SDK process using the `apache_beam.utils.shared` library.
Make more memory available
Make more memory available

- Use a machine type with more memory per vCPU.
- Use a machine type with more vCPUs (Java and Go streaming pipelines)
- Reduce the number of threads.  
  `--number_of_worker_harness_threads`
- Use only one Apache Beam SDK process (Python streaming and Python Runner v2 pipelines).  
  `--experiments=no_use_multiple_sdk_containers`
- Use vertical autoscaling on Dataflow
Thank you!

Special thanks:
Abby Motley
Kenneth Knowles
Prathap Kumar Parvathareddy
Riju Kallivalappil
Valentyn Tymofieiev
Vince Gonzalez
Zach Zimmerman