

Resolving Out of memory errors in Beam pipelines

Guide

Presented by :

Zeeshan Khan

Cloud Data Engineer

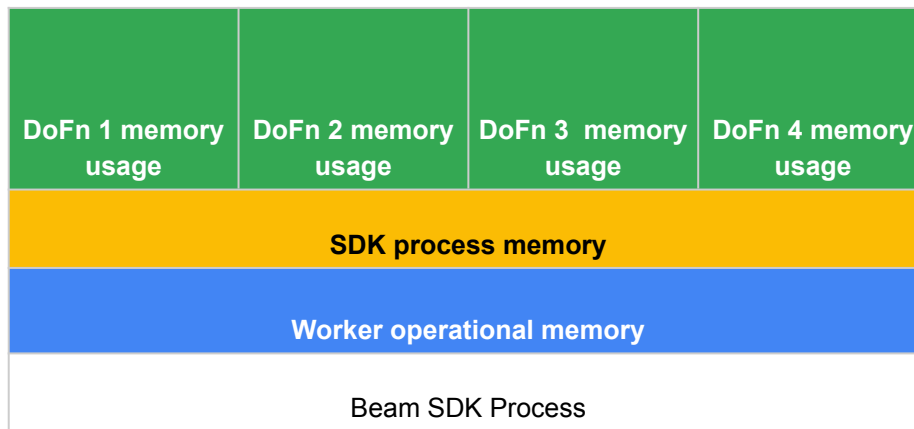
Google Cloud Consulting



Dataflow Memory usage

Worker operational memory

OS and system processes. Less than 1 GB

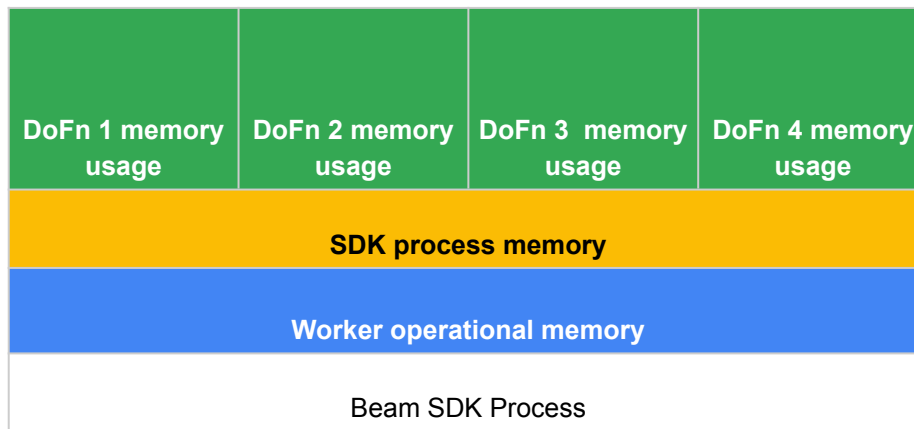


Dataflow Memory usage

SDK process memory

In memory objects and data. Shared across DoFns.

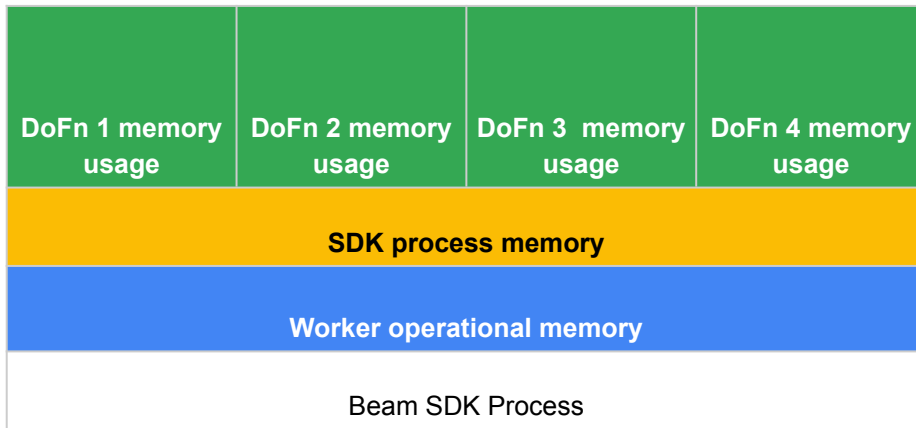
- Side inputs
- ML models
- In memory singeltons
- Python objects created with the `apache_beam.utils.shared` module



Dataflow Memory usage

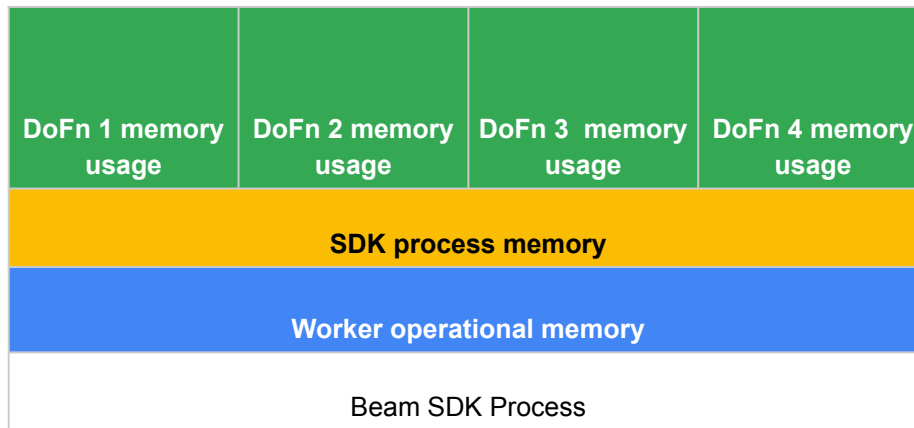
DoFn Memory Usage

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



Dataflow Memory usage

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



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

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

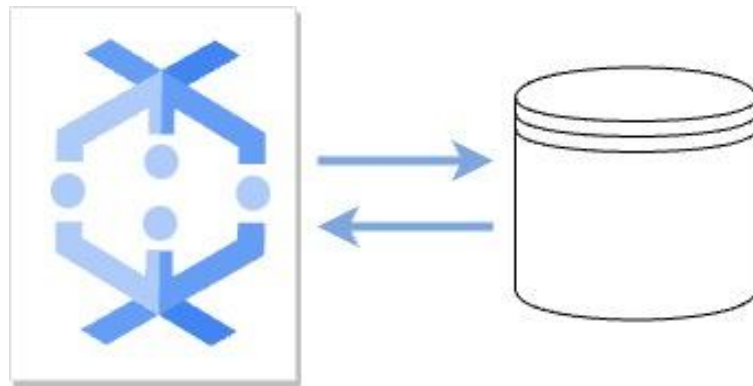
```
PCollection<KV<String, Iterable<String>>> output =  
    input.apply(GroupByKey.<String, String>create());
```


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

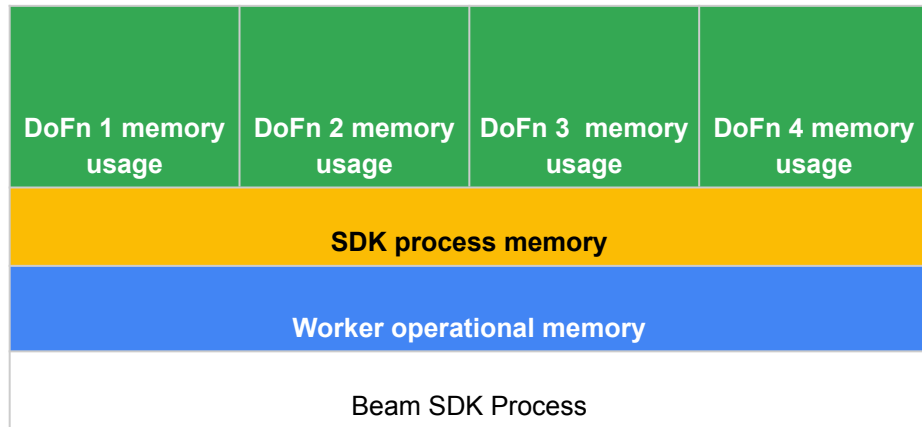
```
input | beam.GroupIntoBatches(3)
```



4) Share objects across threads (cache)

Method of the DoFn :

- Setup
- StartBundle
- Process
- FinishBundle
- Teardown



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