

EE4221 Cloud Computing Systems

Laboratory assignment 4
Google App Engine MapReduce

Objectives

- Get familiar with Google App Engine MapReduce
 - How to configure a simple mapreduce job?
- Manual contains set of initial questions
- Quiz at 16:30 (maximum 20 minutes)

MapReduce Application

← → ↻ localhost:8080

MapReduce Example

Step 1: Upload and choose your input file:

test@example.com's files:

☐	name	uploaded on	source	wordcount link	content link
☒	three_files	2015-06-12 01:40:22.409000	uploaded by user	wordcount	content
☐		2015-10-12 04:37:57.324000	uploaded by user		
☒	input	2015-10-12 04:38:46.051000	uploaded by user	wordcount	content
☒	inout	2015-10-12 04:40:14.952000	uploaded by user	wordcount	content
☒	test	2015-11-05 04:49:24.280000	uploaded by user	wordcount	content

Upload a zip file containing your input text files.

From your computer: No file chosen

Give it a name:

Step 2: Run your MapReduce job:

Input File: none selected

Once your MapReduce job finishes, come back to this page and you'll see a link next to the input file you selected, depending on which job you ran.

About Python syntax

- No semicolon!
- Python provides no braces to indicate blocks of code. Blocks of code are denoted by line indentation.
- String literals: `mystring = "hello"`
- A list an unordered collection of objects : `mylist = [1, 2, 3,2]`
- A *set* an unordered collection of distinct objects:
 - `myset = {1, 2, 3}`
 - `myset = set(mylist)` *Produces a collection of distinct objects!*
- For loop: **for x in collection**
- Get length:
 - `int num_of_elements = len(mylist)`
 - `int num_of_characters = len(mystring)`
- Printing formatting:
 - `"\n"` is used to create new line
 - `"%s: %d\n" % (string, number)` *Output example: day: 5*

WordCount

- input: [file1: hello world map reduce] [file2: hello cloud computing] [file3: hello cloud]
- map: <hello, 1>, <world, 1>, <map, 1>, <reduce, 1>, <hello, 1>, <cloud, 1>, <computing, 1>, <hello, 1>, <cloud, 1>
- shuffle: <hello, <1,1,1> >, <world, <1> >, <map, <1> >, <reduce, <1> >, <cloud, <1,1> >, <computing, <1> >
- reduce: <hello, 3>, <world, 1>, <map, 1> <reduce, 1>, <cloud, 2>, <computing, 1>

WordCount

```
class WordCountPipeline(base_handler.PipelineBase):
```

```
    """A pipeline to run Word count demo. Args: blobkey: blobkey to process as string. Should be a zip archive with text files inside.
    """
```

```
    def run(self, filekey, blobkey):
```

```
        logging.debug("filename is %s" % filekey)
```

```
        bucket_name = app_identity.get_default_gcs_bucket_name()
```

```
        output = yield mapreduce_pipeline.MapreducePipeline(
```

```
            "word_count",
```

```
            "main.word_count_map",
```

```
            "main.word_count_reduce",
```

```
            "mapreduce.input_readers.BlobstoreZipInputReader",
```

```
            "mapreduce.output_writers.GoogleCloudStorageOutputWriter",
```

```
            mapper_params={
```

```
                "blob_key": blobkey,
```

```
            },
```

```
            reducer_params={
```

```
                "output_writer": {
```

```
                    "bucket_name": bucket_name,
```

```
                    "content_type": "text/plain",
```

```
                }
```

```
            },
```

```
            shards=1)
```

```
        yield StoreOutput("WordCount", filekey, output)
```

WordCount

```
def word_count_map(data):  
    """Word count map function."""  
    (entry, text_fn) = data  
    text = text_fn()  
  
    logging.debug("Got %s", entry.filename)  
    for s in split_into_sentences(text):  
        for w in split_into_words(s.lower()):  
            yield (w, 1)
```

```
def word_count_reduce(key, values):  
    """Word count reduce function."""  
    yield "%s: %d\n" % (key, len(values))
```

ContentCount

- Determine in which file(s) each of the words occur
- Input: [file1: hello world map reduce] [file2: hello cloud computing] [file3: hello cloud]
- Final result: <hello, <file1, file2, file3>>, <world, <file1>>, <map, <file1>> <reduce, <file1>>, <cloud, <file2, file3>>, <computing, <file2>>
- Design map and reduce functions (content_map, content_reduce)
- Revise the mapreduce job specification
- Set number concurrent workers to 1
- Output format should be list of key-value pairs (string, string) in the following format
 - cloud: ['input/in1.txt', 'input/in2.txt']
 - computing: ['input/in3.txt']

```

class ContentPipeline(base_handler.PipelineBase):
    """A pipeline to run Content Count.
    Args:
        blobkey: blobkey to process as string. Should be a zip archive with
        text files inside.
    """
    def run(self, filekey, blobkey):
        bucket_name = app_identity.get_default_gcs_bucket_name()
        output = yield mapreduce_pipeline.MapreducePipeline(
            "index",
            "main.NAME_OF_MAP_FUNCTION",
            "main.NAME_OF_REDUCE_FUNCTION",
            "mapreduce.input_readers.INPUT_READER",
            "mapreduce.output_writers.OUTPUT_WRITER",
            mapper_params={
                "blob_key": blobkey,
            },
            reducer_params={
                "output_writer": {
                    "bucket_name": bucket_name,
                    "content_type": "text/plain",
                }
            },
            shards=NUMBER_OF_SHARDS)
        yield StoreOutput("Index", filekey, output)

```