PROGRAMAÇÃO DE ENTRADAS BATCH EM CLUSTERS

Primeiro projeto de pesquisa

Equipe

Thiago Paiva

19/0020377

Problema

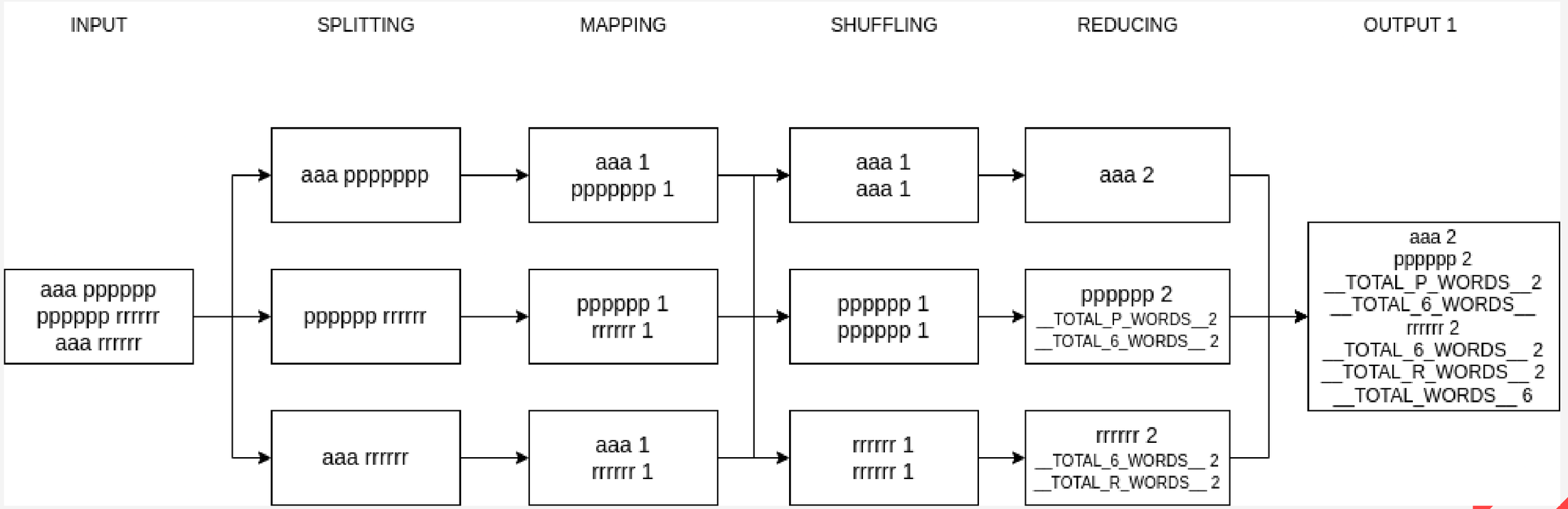
Contar palavras
Contar palavras que começam com S
Contar palavras que começam com P
Contar palavras que começam com R
Contar palavras com 6 caracteres
Contar palavras com 8 caracteres
Contar palavras com 11 caracteres

Solução Hadoop Saida

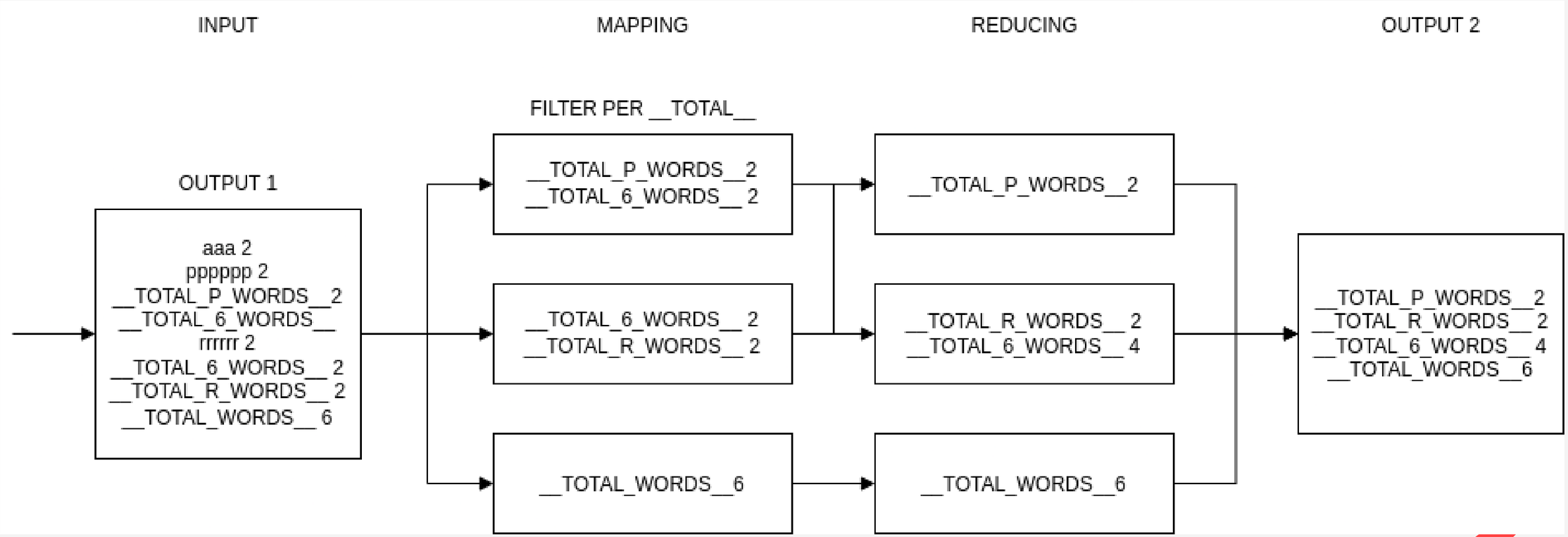
```
→  p1 head output1/*
==> output1/part-00000 <==
a 319933
aa 12293
aaa 449
aaaa 13
aaaaa 2
aaaaaaeskhqy 1
aaaaabjng 1
aaaaacawwwg 1
__TOTAL_11_WORDS__ 1
aaaaactwulp 1

==> output1/_SUCCESS <==
→  p1
```

```
→  p1 cat output2/*
__TOTAL_11_WORDS__ 8334756
__TOTAL_6_WORDS__ 8333788
__TOTAL_8_WORDS__ 8333314
__TOTAL_P_WORDS__ 3846306
__TOTAL_R_WORDS__ 3847588
__TOTAL_S_WORDS__ 3845863
__TOTAL_WORDS__ 100000000
```

Solução Hadoop Parte 1



Solução Hadoop Parte 2



Solução Hadoop Parte 1 - MAPPER

mapper.py > ...

```python
#!/bin/python3
import sys

for line in sys.stdin:
    if not line.strip():
        continue
    words = line.split()
    words_mapped = map(lambda w: f'{w.lower()} 1\n', words)
    print("".join(words_mapped), end='')

```

Solução Hadoop Parte 1 - REDUCER

```
22  for line in sys.stdin:
23      if not line.strip():
24          continue
25
26      word, count = line.split()
27
28      try:
29          count = int(count)
30      except ValueError:
31          continue
32
33      count_words += count
34
35      if word == current_word:
36          current_count += count
37          continue
38
39      print_word_info()
40      current_word = word
41      current_count = count
42
43
44  print_word_info()
45  print('__TOTAL_WORDS__', count_words)
46
```

```
1   #!/bin/python3
2   import sys
3
4   current_word = None
5   current_count = 0
6   count_words = 0
7
8
9   def print_word_info():
10      if not current_word:
11          return
12
13      print(current_word, current_count)
14
15      if (fstLetter := current_word[0].upper()) in 'SPR':
16          print(f'__TOTAL_{fstLetter}_WORDS__ {current_count}')
17
18      if (ln := len(current_word)) in [6, 8, 11]:
19          print(f'__TOTAL_{ln}_WORDS__ {current_count}')
20
```

Solução Hadoop Parte 2 - MAPPER

mapper2.py > ...

```python
#!/bin/python3
import sys

for line in sys.stdin:
    if not line.strip():
        continue
    if line.startswith('__TOTAL_'):
        print(line)

```

Solução Hadoop Parte 2 - REDUCER

```
15  for line in sys.stdin:
16      if not line.strip():
17          continue
18
19      word, count = line.split()
20
21      try:
22          count = int(count)
23      except ValueError:
24          continue
25
26      if word == current_word:
27          current_count += count
28          continue
29
30      print_word_info()
31      current_word = word
32      current_count = count
33
34
35  print_word_info()
36
```

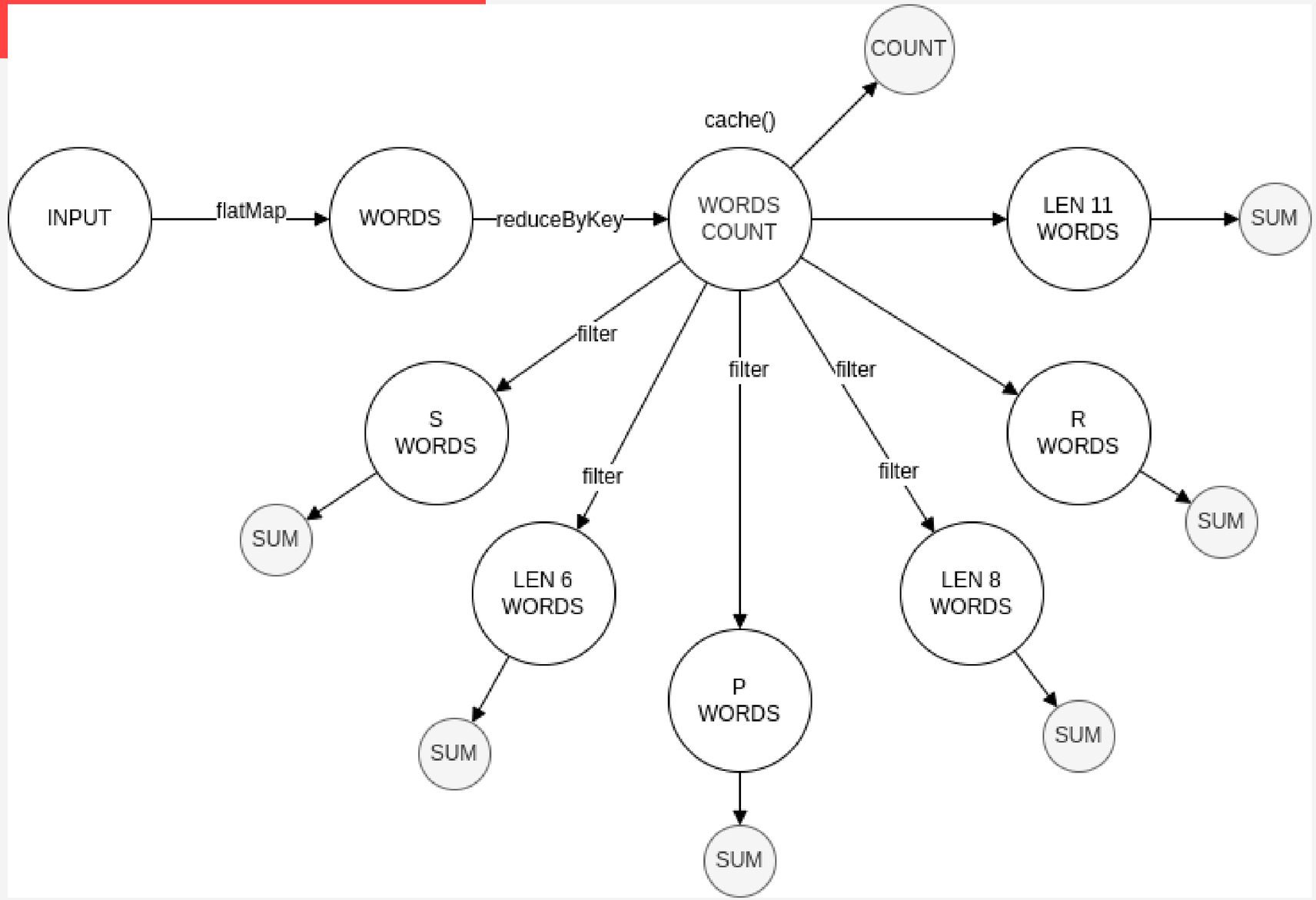
reducer2.py > ...

```
1   #!/bin/python3
2   import sys
3
4   current_word = None
5   current_count = 0
6
7
8   def print_word_info():
9       if not current_word:
10          return
11
12      print(current_word, current_count)
13
14
```

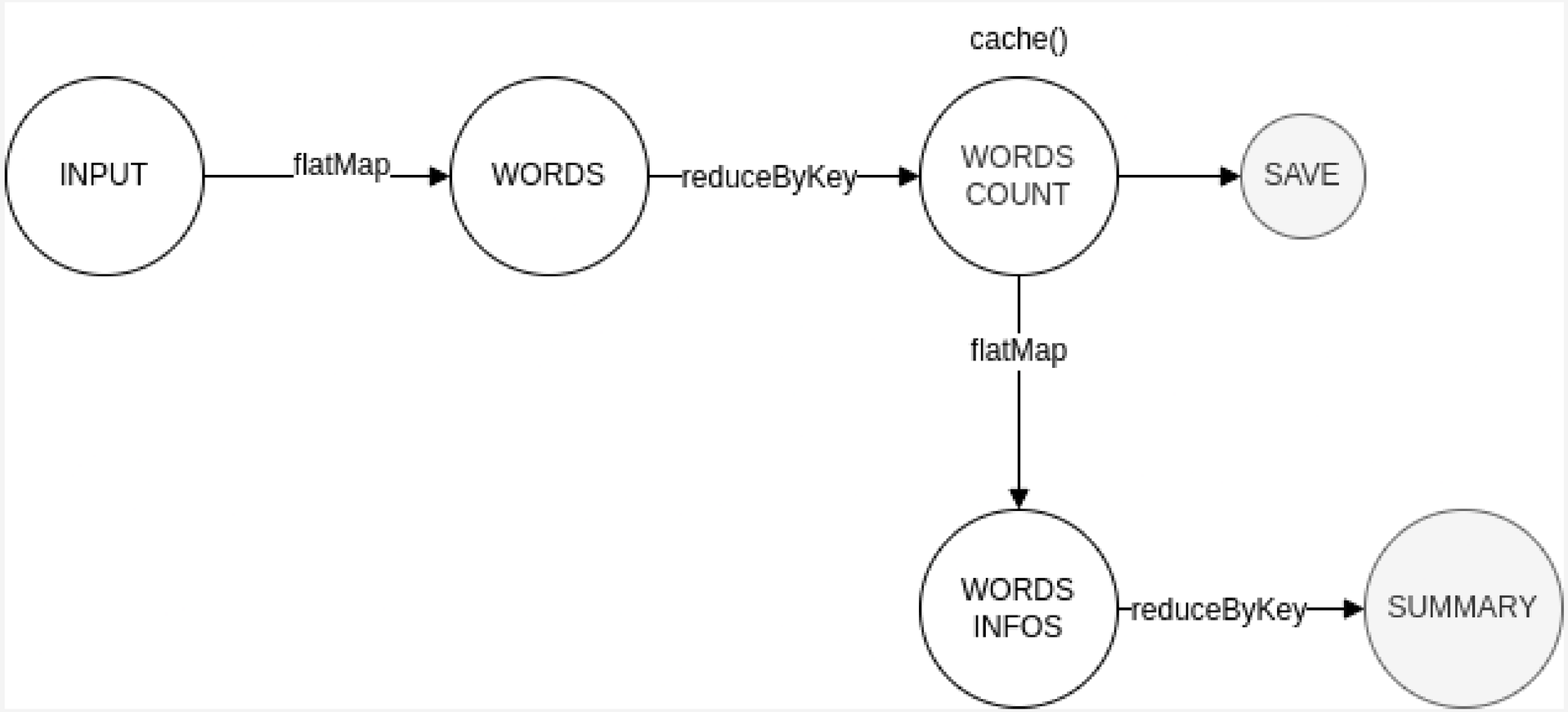
Comando Exec Hadoop

```
mapred streaming \
    -input input \
    -output output1 \
    -mapper mapper.py \
    -reducer reducer.py \
    -file reducer.py -file mapper.py \
    && \
    mapred streaming \
        -input output1 \
        -output output2 \
        -mapper mapper2.py \
        -reducer reducer2.py \
        -file reducer2.py -file mapper2.py
```

Solução Spark



Solução Spark "OTIMIZADA"



Solução Spark

```
5   # Inicializa o Spark e passa o endereco do nó master
6   conf = SparkConf().setAppName('PSPD - P1').setMaster('spark://notebook:7077')
7   sc = SparkContext(conf=conf)
8   hdfs_path = "hdfs://notebook:9000/user/thiago"
9   # Abre o arquivo do HDFS
10  file = sc.textFile(hdfs_path + "/input")
11
12  # Mapeia cada palavra do arquivo em um par (Palavra, 1)
13  # Como pode ter mais de uma palavra por linha, utiliza-se o flatMap
14  words = file.flatMap(lambda l: ((w.lower(), 1) for w in l.split()))
15
16  # Agrupa as palavras e soma as quantidades
17  words_count = words.reduceByKey(lambda a, b: a + b)
18
19  # Avisa para salvar o conjunto de dados na memoria
20  words_count.cache()
21
```

Solução Spark

```
infos = {
    'all': 'TOTAL_WORDS',
    's': 'TOTAL_S_WORDS',
    'p': 'TOTAL_P_WORDS',
    'r': 'TOTAL_R_WORDS',
    '6': 'TOTAL_6_WORDS',
    '8': 'TOTAL_8_WORDS',
    '11': 'TOTAL_11_WORDS',
}

def find_infos(pair):
    word, count = pair
    len_word = len(word)
    initial_letter = word[0]
    result = [(infos['all'], count)]

    if initial_letter in 'spr':
        result.append((infos[initial_letter], count))
    if len_word in [6, 8, 11]:
        result.append((infos[str(len_word)], count))
    return result

# Salva as palavras em um arquivo no HDFS
words_count.saveAsTextFile(hdfs_path + "/spark-output/words")

# Itera sobre o dataset
summary = words_count.flatMap(find_infos).reduceByKey(lambda a, b: a+b)

# Salva o sumario
summary.saveAsTextFile(hdfs_path + "/spark-output/result")
```

Comando Exec Spark

```
$SPARK_HOME/bin/spark-submit $PROJECT_HOME/sparkOtimized.py
```

Single Node

Hostname	Cpu	Núcleos	Memória	Função
notebook	I5-7300HQ 2.50Ghz	4	16GB DDR4 2666MHz	Worker

Tabela 1: Configuração da máquina em Single Node

Single Node

```
                Bytes Written=100
2022-08-06 21:02:50,648 INFO streaming.StreamJob: Output directory: /home/d/unb/2022.2/pspd/p1/outpu
t2
( bin/mapred streaming -input $FPATH/input -output $FPATH/output1 -mapper    )  792,60s user 22,68s
system 140% cpu 9:42,10 total
```

Figura 16: Execução da solução do Hadoop com medição de tempo

```
2c5-4319-bb00-1c5461942fd0
( bin/spark-submit /home/d/unb/2022.2/pspd/p1/spark.py; )  181,68s user 12,41s syste
m 21% cpu 15:01,57 total
```

Figura 17: Execução da solução do Spark com medição de tempo

- Hadoop - 9m 42s
- Spark - 15m 1s

2 Node Cluster

Hostname	Cpu	Núcleos	Memória	Funções
notebook	I5-7300HQ 2.50Ghz	4	16GB DDR4 2666MHz	NameNode DataNode ResourceManager NodeManager Master Worker
pc	Xeon E5440 2.83Ghz	4	8GB DDR2 800MHz	DataNode NodeManager Worker

Tabela 2: Configuração da máquina em modo Cluster

2 Node Cluster

FinalStatus Reported by AM:	SUCCEEDED
Started:	Dom ago 07 18:45:35 -0300 2022
Launched:	Dom ago 07 18:46:17 -0300 2022
Finished:	Dom ago 07 18:51:07 -0300 2022
Elapsed:	5mins, 31sec

FinalStatus Reported by AM:	SUCCEEDED
Started:	Dom ago 07 18:55:35 -0300 2022
Launched:	Dom ago 07 18:55:36 -0300 2022
Finished:	Dom ago 07 18:59:30 -0300 2022
Elapsed:	3mins, 54sec

Figura 20: Resultado do Job pelo Hadoop.

Figura 21: Resultado do Job pelo Spark.

- Hadoop - 9m 25s
- Spark - 13m 28s

7 Node Cluster - FGA

Hostname	Função
gpu1	ResourceNode NameNode DataNode NodeManager SecondaryNameNode Master
cm1	NodeManager Worker
cm2	Worker
cm3	NodeManager DataNode Worker
cm4	NodeManager Datanode Worker
gpu2	NodeManager DataNode Worker
gpu3	NodeManager DataNode Worker

Tabela 3: Configuração da máquina em modo Cluster FGA

2 Node Cluster

```
2022-08-08 10:33:38,696 INFO streaming.StreamJob: Output directory: output4

real    5m2.233s
user    0m16.135s
sys     0m1.037s
```

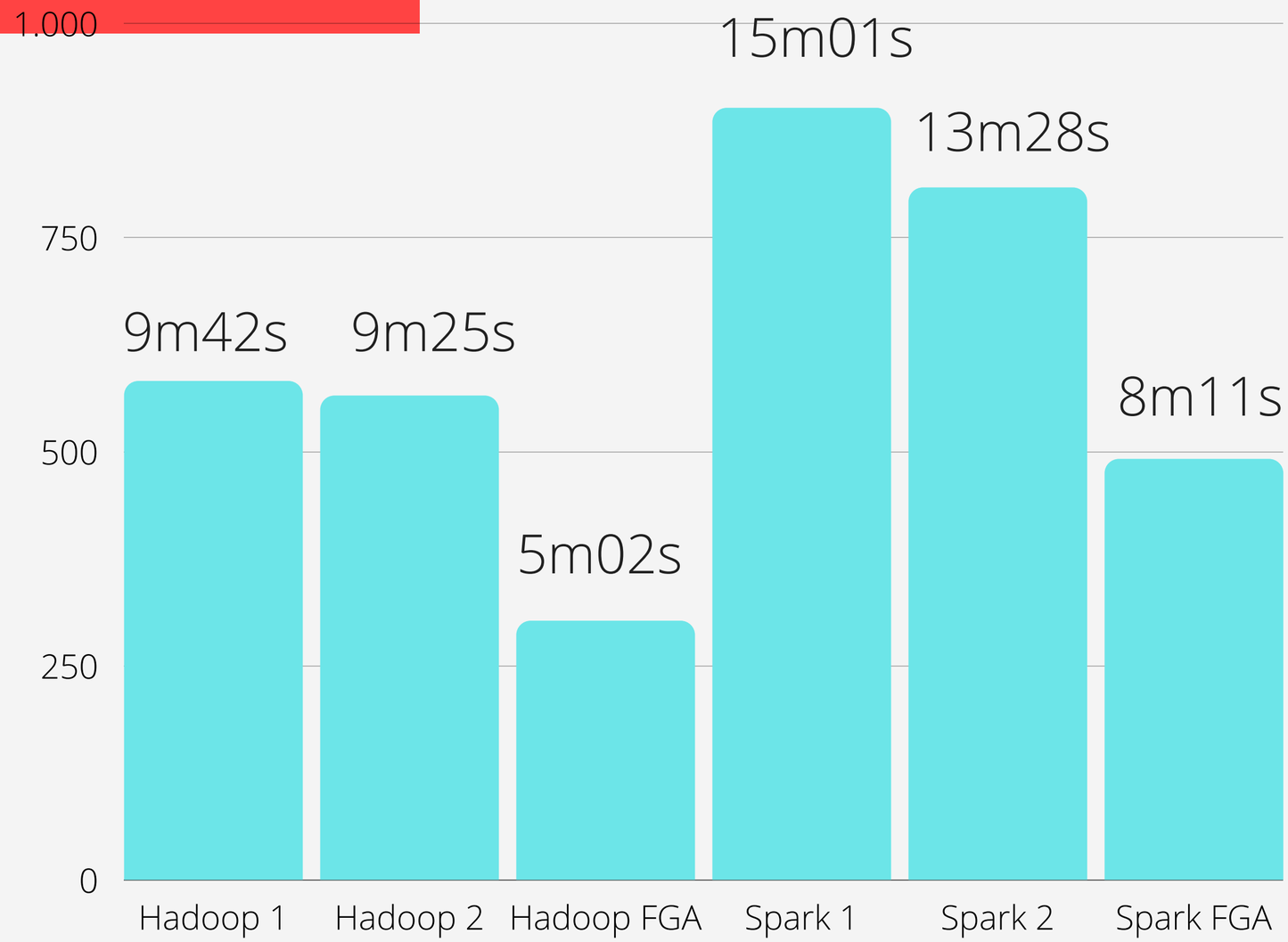
Figura 2: Resultado do Job pelo Hadoop.

```
22/08/08 10:57:44 INFO ShutdownHookManager:
9-1329-465f-8f79-90f5516a3a27

real    8m11.098s
user    0m20.178s
sys     0m1.248s
```

Figura 22: Resultado do Job pelo Spark

- Hadoop - 5m 2s
- Spark - 8m 11s

Desempenho Geral



Obrigado pela atenção