3rd Lab - CES 27 - Distributed Programming

October 10th, 2019

Prof. Celso Massaki HIRATA, hirata@ita.br

Prof. JULIANA de Melo Bezerra, juliana@ita.br

Igor Bragaia[1], igor.bragaia@gmail.com

[1]Computer Engineering undergraduate student at Aeronautics Institute of Technology (ITA)

This project aims to implement MapReduce algorithm using both sequential and distributed approach.

TASK 1 - Sequential MapReduce

TASK 1.1 Complete mapFunc function to identify words from each slice of initial file

```go
func mapFunc(input []byte) (result []mapreduce.KeyValue) {
    var (
        text          string
        delimiterFunc func(c rune) bool
        words         []string
    )

    text = string(input)

    delimiterFunc = func(c rune) bool {
        return !unicode.IsLetter(c) && !unicode.IsNumber(c)
    }

    words = strings.FieldsFunc(text, delimiterFunc)

    result = make([]mapreduce.KeyValue, 0)

    for _, word := range words {
        keyvalue := mapreduce.KeyValue{
            Key: strings.ToLower(word),
            Value: "1",
        }
        result = append(result, keyvalue)
    }

    return result
}
```

wordcount.go

TASK 1.2 Implement reduceFunc function allowing correct word couting

```go
func reduceFunc(input []mapreduce.KeyValue) (result []mapreduce.KeyValue) {
    var mapAux map[string]int = make(map[string]int)
    for _,item := range input {
        _, ok := mapAux[item.Key]
        value, _ := strconv.Atoi(item.Value)
        if ok {
            mapAux[item.Key] += value
        } else {
            mapAux[item.Key] = value
        }
    }

    result = make([]mapreduce.KeyValue, 0)
    for k,v := range mapAux {
        keyvalue := mapreduce.KeyValue{
            Key: k,
            Value: strconv.Itoa(v),
        }
        result = append(result, keyvalue)
    }

    return result
}
```

wordcount.go

TASK 1.3 Validate sequential approach

We run

```
./wordcount --mode sequential --file files/teste.txt --chunksize 100 --reducejobs 2
```

terminal

Thus, for the following input file teste.txt

```
Teste para ver o correto funcionamento da contagem de palavras. Por exemplo, a palavra teste deve
ocorrer apenas tres vezes, sendo que a ultima ocorrencia e esta: teste.
```

teste.txt

We have the following two files at /result folder

```
{"Key":"ver","Value":"1"}
{"Key":"sendo","Value":"1"}
{"Key":"que","Value":"1"}
{"Key":"por","Value":"1"}
{"Key":"e","Value":"1"}
{"Key":"de","Value":"1"}
{"Key":"esta","Value":"1"}
{"Key":"ocorrencia","Value":"1"}
{"Key":"teste","Value":"3"}
{"Key":"o","Value":"1"}
{"Key":"da","Value":"1"}
{"Key":"a","Value":"2"}
{"Key":"palavra","Value":"1"}
{"Key":"vezes","Value":"1"}
```

result-0

```
{"Key":"funcionamento","Value":"1"}
{"Key":"palavras","Value":"1"}
{"Key":"apenas","Value":"1"}
{"Key":"tres","Value":"1"}
{"Key":"ultima","Value":"1"}
{"Key":"para","Value":"1"}
{"Key":"contagem","Value":"1"}
{"Key":"exemplo","Value":"1"}
{"Key":"deve","Value":"1"}
{"Key":"ocorrer","Value":"1"}
{"Key":"correto","Value":"1"}
```

result-1

Note that result-0 plus result-1 files count correctly all words and its frequency at teste.txt file.

TASK 1.4 Run the code for different input files, chuncksize and reduce jobs

Let's consider the following input file

```
palavra1 palavra2 palavra2 palavra3 palavra3 palavra3
```

novoteste.txt

Thus, for chuncksize=100 and reducejobs=2, we have

```
 ./wordcount --mode sequential --file files/novoteste.txt --chunksize 100 --reducejobs 2
```

terminal

From terminal log, we have

```
2019/10/10 01:10:46 Running in sequential mode.
2019/10/10 01:10:46 Running RunSequential...
2019/10/10 01:10:46 Fanning in file map/map-0
2019/10/10 01:10:46 Fanning out file result/result-0
2019/10/10 01:10:46 Fanning out file result/result-1
```

terminal log

We have the following two files at /result folder

```
{"Key":"palavra2","Value":"2"}
```

result-0

```
{"Key":"palavra1","Value":"1"}
{"Key":"palavra3","Value":"3"}
```

result-1

Note that result-0 plus result-1 files count correctly all words and its frequency at novoteste.txt file.

Thus, for chuncksize=100 and reducejobs=3, we have

```
./wordcount --mode sequential --file files/novoteste.txt --chunksize 100 --reducejobs 3
```

terminal

From terminal log, we have

```
2019/10/10 01:14:20 Running in sequential mode.
2019/10/10 01:14:20 Running RunSequential...
2019/10/10 01:14:20 Fanning in file map/map-0
2019/10/10 01:14:20 Fanning out file result/result-0
2019/10/10 01:14:20 Fanning out file result/result-1
2019/10/10 01:14:20 Fanning out file result/result-2
```

terminal log

We have the following three files at /result folder

```
{"Key":"palavra3","Value":"3"}
```

result-0

```
{"Key":"palavra2","Value":"2"}
```

result-1

```
{"Key":"palavra1","Value":"1"}
```

result-2

Note that result-0 plus result-1 plus result-2 files count correctly all words and its frequency at novoteste.txt file.

Finally, if we have chuncksize=10 and reducejobs=3, for example, we have

```
./wordcount --mode sequential --file files/novoteste.txt --chunksize 10 --reducejobs 3
```

terminal

From terminal log, we have

```
2019/10/10 01:18:17 Running in sequential mode.
2019/10/10 01:18:17 Running RunSequential...
2019/10/10 01:18:17 Fanning in file map/map-0
2019/10/10 01:18:17 Fanning in file map/map-1
2019/10/10 01:18:17 Fanning in file map/map-2
2019/10/10 01:18:17 Fanning in file map/map-3
2019/10/10 01:18:17 Fanning in file map/map-4
2019/10/10 01:18:17 Fanning in file map/map-5
2019/10/10 01:18:18 Fanning out file result/result-0
2019/10/10 01:18:18 Fanning out file result/result-1
2019/10/10 01:18:18 Fanning out file result/result-2
```

terminal log

What means we have more map files however our result files are exactly the same than previous results from chuncksize=100 and reducejobs=3.

TASK 2 - Distributed MapReduce

TASK 2.1 Complete handleFailingWorkers function code.

```go
func (master *Master) handleFailingWorkers() {
    for {
        worker := <- master.failedWorkerChan

        master.workersMutex.Lock()

        delete(master.workers, worker.id)

        master.workersMutex.Unlock()

        fmt.Printf("Removing worker %d from master list.\n", worker.id)
    }
}
```

master.go

In order to make sure correct behavior from this code, let's run a master and a worker in two different terminals

```
./wordcount --mode distributed --type worker --port 50001 --fail 3
```

first terminal

```
./wordcount --mode distributed --type master --file files/novoteste.txt --chunksize 102400
--reducejobs 5
```

second terminal

Running worker repeatedly until all jobs have been finished, we have the following correct output at second terminal

```
2019/10/10 01:23:55 Running in distributed mode.
2019/10/10 01:23:55 NodeType: master
2019/10/10 01:23:55 Reduce Jobs: 5
2019/10/10 01:23:55 Address: localhost
2019/10/10 01:23:55 Port: 5000
2019/10/10 01:23:55 File: files/novoteste.txt
2019/10/10 01:23:55 Chunk Size: 102400
2019/10/10 01:23:55 Running Master on localhost:5000
2019/10/10 01:23:55 Scheduling Worker.RunMap operations
2019/10/10 01:23:55 Accepting connections on 127.0.0.1:5000
2019/10/10 01:23:57 Registering worker '0' with hostname 'localhost:50001'
2019/10/10 01:23:57 Running Worker.RunMap (ID: '0' File: 'map/map-0' Worker: '0')
2019/10/10 01:23:57 1x Worker.RunMap operations completed
```

```
2019/10/10 01:23:57 Scheduling Worker.RunReduce operations
2019/10/10 01:23:57 Running Worker.RunReduce (ID: '0' File: 'reduce/reduce-0' Worker: '0')
2019/10/10 01:23:57 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '0')
2019/10/10 01:23:58 Operation Worker.RunReduce '1' Failed. Error: dial tcp 127.0.0.1:50001: connect:
connection refused
Removing worker 0 from master list.
2019/10/10 01:24:52 Registering worker '0' with hostname 'localhost:50001'
2019/10/10 01:24:52 Running Worker.RunReduce (ID: '2' File: 'reduce/reduce-2' Worker: '0')
2019/10/10 01:24:52 Running Worker.RunReduce (ID: '3' File: 'reduce/reduce-3' Worker: '0')
2019/10/10 01:24:52 Running Worker.RunReduce (ID: '4' File: 'reduce/reduce-4' Worker: '0')
2019/10/10 01:24:53 Operation Worker.RunReduce '4' Failed. Error: dial tcp 127.0.0.1:50001: connect:
connection refused
Removing worker 0 from master list.
2019/10/10 01:24:54 Registering worker '0' with hostname 'localhost:50001'
Trying to recover Worker.RunReduce (ID: '1' File: 'reduce/reduce-1')
2019/10/10 01:24:54 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '0')
Trying to recover Worker.RunReduce (ID: '4' File: 'reduce/reduce-4')
2019/10/10 01:24:54 Running Worker.RunReduce (ID: '4' File: 'reduce/reduce-4' Worker: '0')
2019/10/10 01:24:54 5x Worker.RunReduce operations completed
2019/10/10 01:24:55 Closing Remote Workers.
2019/10/10 01:24:55 Done.
```

terminal log

TASK 2.2 Modify given code in order to make sure that all failing operations are correctly executed.

```go
type Master struct {
    // Task
    task *Task

    // Network
    address   string
    rpcServer *rpc.Server
    listener  net.Listener

    // Workers handling
    workersMutex sync.Mutex
    workers      map[int]*RemoteWorker
    totalWorkers int // Used to generate unique ids for new workers

    idleWorkerChan   chan *RemoteWorker
    failedWorkerChan chan *RemoteWorker

    // Fault Tolerance
    failedOperationChan chan *Operation
}
```

master.go

```go
func (master *Master) schedule(task *Task, proc string, filePathChan chan string) int {
    var (
        wg        sync.WaitGroup
        filePath  string
        worker    *RemoteWorker
        operation *Operation
        counter   int
    )

    log.Printf("Scheduling %v operations\n", proc)

    counter = 0
    for filePath = range filePathChan {
        operation = &Operation{proc, counter, filePath}
        counter++

        worker = <-master.idleWorkerChan
        wg.Add(1)
        go master.runOperation(worker, operation, &wg)
    }

    wait := 1
    for wait == 1{
        select {
            case operation := <-master.failedOperationChan:
                worker = <-master.idleWorkerChan
                fmt.Printf("Trying to recover %v (ID: '%v' File: '%v')\n", operation.proc,
operation.id, operation.filePath)
                go master.runOperation(worker, operation, &wg)
            default:
                wait = 0
        }
    }

    wg.Wait()

    log.Printf("%vx %v operations completed\n", counter, proc)
    return counter
}
```

master_scheduler.go

```go
func (master *Master) runOperation(remoteWorker *RemoteWorker, operation *Operation, wg
*sync.WaitGroup) {
    var (
        err  error
        args *RunArgs
    )

    log.Printf("Running %v (ID: '%v' File: '%v' Worker: '%v')\n", operation.proc, operation.id,
operation.filePath, remoteWorker.id)

    args = &RunArgs{operation.id, operation.filePath}
    err = remoteWorker.callRemoteWorker(operation.proc, args, new(struct{}))

    if err != nil {
        log.Printf("Operation %v '%v' Failed. Error: %v\n", operation.proc, operation.id, err)
        master.failedOperationChan <- operation
        master.failedWorkerChan <- remoteWorker
    } else {
        wg.Done()
        master.idleWorkerChan <- remoteWorker
    }
}
```

master_scheduler.go

TASK 2.3 Validate distributed approach

Initially, let’s see what is the content from /result folder for a worker that does not fail. We run

```
./wordcount --mode distributed --type worker --port 50002
```

first terminal

```
./wordcount --mode distributed --type master --file files/teste.txt --chunksize 102400 --reducejobs
5
```
second terminal

Thus, for the following input file

```
Teste para ver o correto funcionamento da contagem de palavras. Por exemplo, a palavra teste deve
ocorrer apenas tres vezes, sendo que a ultima ocorrencia e esta: teste.
```
teste.txt

We have the following correct output at second terminal

```
2019/10/10 01:45:34 Running in distributed mode.
2019/10/10 01:45:34 NodeType: master
2019/10/10 01:45:34 Reduce Jobs: 5
2019/10/10 01:45:34 Address: localhost
2019/10/10 01:45:34 Port: 5000
2019/10/10 01:45:34 File: files/teste.txt
2019/10/10 01:45:34 Chunk Size: 102400
2019/10/10 01:45:34 Running Master on localhost:5000
2019/10/10 01:45:34 Scheduling Worker.RunMap operations
2019/10/10 01:45:34 Accepting connections on 127.0.0.1:5000
2019/10/10 01:45:35 Registering worker '0' with hostname 'localhost:50002'
2019/10/10 01:45:35 Running Worker.RunMap (ID: '0' File: 'map/map-0' Worker: '0')
2019/10/10 01:45:35 1x Worker.RunMap operations completed
2019/10/10 01:45:36 Scheduling Worker.RunReduce operations
2019/10/10 01:45:36 Running Worker.RunReduce (ID: '0' File: 'reduce/reduce-0' Worker: '0')
2019/10/10 01:45:36 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '0')
2019/10/10 01:45:36 Running Worker.RunReduce (ID: '2' File: 'reduce/reduce-2' Worker: '0')
2019/10/10 01:45:36 Running Worker.RunReduce (ID: '3' File: 'reduce/reduce-3' Worker: '0')
2019/10/10 01:45:36 Running Worker.RunReduce (ID: '4' File: 'reduce/reduce-4' Worker: '0')
2019/10/10 01:45:36 5x Worker.RunReduce operations completed
2019/10/10 01:45:36 Closing Remote Workers.
2019/10/10 01:45:36 Done.
```
third terminal log

Then we look at /result folder contents

```
igorbragaia@igorbragaia:~/go/src/labMapReduce/wordcount$ ls -l result
total 24
-rw-rw-r-- 1 igorbragaia igorbragaia  82 out 10 01:45 result-0
-rw-rw-r-- 1 igorbragaia igorbragaia  55 out 10 01:45 result-1
-rw-rw-r-- 1 igorbragaia igorbragaia 138 out 10 01:45 result-2
-rw-rw-r-- 1 igorbragaia igorbragaia 172 out 10 01:45 result-3
-rw-rw-r-- 1 igorbragaia igorbragaia 254 out 10 01:45 result-4
-rw-rw-r-- 1 igorbragaia igorbragaia 701 out 10 01:45 result-final.txt
```
third terminal log

Now, let's introduce a worker that fails at reduce operation. We run

```
./wordcount --mode distributed --type worker --port 50001 --fail 3
```
first terminal

```
./wordcount --mode distributed --type worker --port 50002
```
second terminal

```
./wordcount --mode distributed --type master --file files/teste.txt --chunksize 102400 --reducejobs
5
```
third terminal

We have the following correct output at third terminal, regarding correct failure treatment

```
2019/10/10 01:55:52 Running in distributed mode.
```

```
2019/10/10 01:55:52 NodeType: master
2019/10/10 01:55:52 Reduce Jobs: 5
2019/10/10 01:55:52 Address: localhost
2019/10/10 01:55:52 Port: 5000
2019/10/10 01:55:52 File: files/teste.txt
2019/10/10 01:55:52 Chunk Size: 102400
2019/10/10 01:55:52 Running Master on localhost:5000
2019/10/10 01:55:52 Scheduling Worker.RunMap operations
2019/10/10 01:55:52 Accepting connections on 127.0.0.1:5000
2019/10/10 01:55:53 Registering worker '0' with hostname 'localhost:50001'
2019/10/10 01:55:53 Running Worker.RunMap (ID: '0' File: 'map/map-0' Worker: '0')
2019/10/10 01:55:54 1x Worker.RunMap operations completed
2019/10/10 01:55:54 Scheduling Worker.RunReduce operations
2019/10/10 01:55:54 Running Worker.RunReduce (ID: '0' File: 'reduce/reduce-0' Worker: '0')
2019/10/10 01:55:54 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '0')
2019/10/10 01:55:54 Registering worker '1' with hostname 'localhost:50002'
2019/10/10 01:55:54 Running Worker.RunReduce (ID: '2' File: 'reduce/reduce-2' Worker: '1')
2019/10/10 01:55:54 Running Worker.RunReduce (ID: '3' File: 'reduce/reduce-3' Worker: '1')
2019/10/10 01:55:55 Operation Worker.RunReduce '1' Failed. Error: dial tcp 127.0.0.1:50001: connect:
connection refused
Removing worker 0 from master list.
2019/10/10 01:55:55 Running Worker.RunReduce (ID: '4' File: 'reduce/reduce-4' Worker: '1')
Trying to recover Worker.RunReduce (ID: '1' File: 'reduce/reduce-1')
2019/10/10 01:55:55 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '1')
2019/10/10 01:55:55 5x Worker.RunReduce operations completed
2019/10/10 01:55:55 Closing Remote Workers.
2019/10/10 01:55:55 Done.
```

third terminal log

Finally, we verify that result/result-final.txt file has exactly same size for previous case with no failing worker

```
igorbragaia@igorbragaia:~/go/src/labMapReduce/wordcount$ ls -l result
total 24
-rw-rw-r-- 1 igorbragaia igorbragaia  82 out 10 01:55 result-0
-rw-rw-r-- 1 igorbragaia igorbragaia  55 out 10 01:55 result-1
-rw-rw-r-- 1 igorbragaia igorbragaia 138 out 10 01:55 result-2
-rw-rw-r-- 1 igorbragaia igorbragaia 172 out 10 01:55 result-3
-rw-rw-r-- 1 igorbragaia igorbragaia 254 out 10 01:55 result-4
-rw-rw-r-- 1 igorbragaia igorbragaia 701 out 10 01:55 result-final.txt
```

third terminal log

Now, let’s introduce a worker that fails at map operation. In order to do that, we modify chuncksize and reducejobs. We run

```
./wordcount --mode distributed --type worker --port 50001 --fail 2
```

first terminal

```
./wordcount --mode distributed --type worker --port 50002
```

second terminal

```
./wordcount --mode distributed --type master --file files/teste.txt --chunksize 50 --reducejobs 2
```

third terminal

We have the following correct output at third terminal, regarding correct failure treatment

```
2019/10/10 22:44:53 Running in distributed mode.
2019/10/10 22:44:53 NodeType: master
2019/10/10 22:44:53 Reduce Jobs: 2
2019/10/10 22:44:53 Address: localhost
2019/10/10 22:44:53 Port: 5000
```

```
2019/10/10 22:44:53 File: files/teste.txt
2019/10/10 22:44:53 Chunk Size: 50
2019/10/10 22:44:53 Running Master on localhost:5000
2019/10/10 22:44:53 Scheduling Worker.RunMap operations
2019/10/10 22:44:53 Accepting connections on 127.0.0.1:5000
2019/10/10 22:44:53 Registering worker '0' with hostname 'localhost:50001'
2019/10/10 22:44:53 Running Worker.RunMap (ID: '0' File: 'map/map-0' Worker: '0')
2019/10/10 22:44:53 Running Worker.RunMap (ID: '1' File: 'map/map-1' Worker: '0')
2019/10/10 22:44:54 Operation Worker.RunMap '1' Failed. Error: dial tcp 127.0.0.1:50001: connect:
connection refused
Removing worker 0 from master list.
2019/10/10 22:45:04 Registering worker '1' with hostname 'localhost:50001'
2019/10/10 22:45:04 Running Worker.RunMap (ID: '2' File: 'map/map-2' Worker: '1')
2019/10/10 22:45:04 Running Worker.RunMap (ID: '3' File: 'map/map-3' Worker: '1')
Trying to recover Worker.RunMap (ID: '1' File: 'map/map-1')
2019/10/10 22:45:05 Running Worker.RunMap (ID: '1' File: 'map/map-1' Worker: '1')
2019/10/10 22:45:05 4x Worker.RunMap operations completed
2019/10/10 22:45:05 Scheduling Worker.RunReduce operations
2019/10/10 22:45:05 Running Worker.RunReduce (ID: '0' File: 'reduce/reduce-0' Worker: '1')
2019/10/10 22:45:05 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '1')
2019/10/10 22:45:05 2x Worker.RunReduce operations completed
2019/10/10 22:45:05 Closing Remote Workers.
2019/10/10 22:45:05 Done.
```

third terminal log

Finally, we verify that result/result-final.txt file has exactly same size for previous case with no failing worker

```
igorbragaia@igorbragaia:~/go/src/labMapReduce/wordcount$ ls -l result
total 12
-rw-rw-r-- 1 igorbragaia igorbragaia 374 out 10 22:45 result-0
-rw-rw-r-- 1 igorbragaia igorbragaia 327 out 10 22:45 result-1
-rw-rw-r-- 1 igorbragaia igorbragaia 701 out 10 22:45 result-final.txt
```

third terminal log

TASK 2.4 Test distributed approach using different input file and different values for chuncksize and reducejobs

Let's consider following input file

```
palavra1 palavra2 palavra2 palavra3 palavra3 palavra3 palavra1 palavra2 palavra2 palavra3 palavra3
palavra3 palavra1 palavra2 palavra2 palavra3 palavra3 palavra3
palavra4 palavra4 palavra5 palavra4 palavra4 palavra5 palavra4 palavra4 palavra5
```

novoteste.txt

Then we run master with chuncksize=1000 and reducejobs=5 and three workers which two of them will fail

```
./wordcount --mode distributed --type worker --port 50001 --fail 2
```

first terminal

```
./wordcount --mode distributed --type worker --port 50002 --fail 2
```

second terminal

```
./wordcount --mode distributed --type worker --port 50003
```

third terminal

```
./wordcount --mode distributed --type master --file files/novoteste.txt --chunksize 1000
--reducejobs 5
```

fourth terminal

We have the following correct output at fourth terminal, regarding correct failure treatment

```
2019/10/10 22:07:15 Running in distributed mode.
2019/10/10 22:07:15 NodeType: master
2019/10/10 22:07:15 Reduce Jobs: 5
2019/10/10 22:07:15 Address: localhost
2019/10/10 22:07:15 Port: 5000
2019/10/10 22:07:15 File: files/novoteste.txt
2019/10/10 22:07:15 Chunk Size: 1000
2019/10/10 22:07:15 Running Master on localhost:5000
2019/10/10 22:07:15 Scheduling Worker.RunMap operations
2019/10/10 22:07:15 Accepting connections on 127.0.0.1:5000
2019/10/10 22:07:16 Registering worker '0' with hostname 'localhost:50001'
2019/10/10 22:07:16 Running Worker.RunMap (ID: '0' File: 'map/map-0' Worker: '0')
2019/10/10 22:07:16 1x Worker.RunMap operations completed
2019/10/10 22:07:16 Scheduling Worker.RunReduce operations
2019/10/10 22:07:16 Running Worker.RunReduce (ID: '0' File: 'reduce/reduce-0' Worker: '0')
2019/10/10 22:07:16 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '0')
2019/10/10 22:07:16 Registering worker '1' with hostname 'localhost:50002'
2019/10/10 22:07:16 Running Worker.RunReduce (ID: '2' File: 'reduce/reduce-2' Worker: '1')
2019/10/10 22:07:16 Running Worker.RunReduce (ID: '3' File: 'reduce/reduce-3' Worker: '1')
2019/10/10 22:07:17 Operation Worker.RunReduce '1' Failed. Error: dial tcp 127.0.0.1:50001: connect:
connection refused
Removing worker 0 from master list.
2019/10/10 22:07:17 Operation Worker.RunReduce '3' Failed. Error: dial tcp 127.0.0.1:50002: connect:
connection refused
Removing worker 1 from master list.
2019/10/10 22:07:19 Registering worker '2' with hostname 'localhost:50003'
2019/10/10 22:07:19 Running Worker.RunReduce (ID: '4' File: 'reduce/reduce-4' Worker: '2')
Trying to recover Worker.RunReduce (ID: '1' File: 'reduce/reduce-1')
2019/10/10 22:07:19 Running Worker.RunReduce (ID: '1' File: 'reduce/reduce-1' Worker: '2')
Trying to recover Worker.RunReduce (ID: '3' File: 'reduce/reduce-3')
2019/10/10 22:07:19 Running Worker.RunReduce (ID: '3' File: 'reduce/reduce-3' Worker: '2')
2019/10/10 22:07:19 5x Worker.RunReduce operations completed
2019/10/10 22:07:19 Closing Remote Workers.
2019/10/10 22:07:19 Done.
```

fourth terminal log

Finally, if we look at /result folder, we have the correct expected word counting from our novoteste.txt input file.

```
{"Key":"palavra3","Value":"9"}
{"Key":"palavra2","Value":"6"}
{"Key":"palavra5","Value":"3"}
{"Key":"palavra1","Value":"3"}
{"Key":"palavra4","Value":"6"}
```

/result