

Modelo MapReduce {

[usando Elixir]

< Cleiane Bondade >

< Gabriel da Costa >

< Moisés Araújo >

< Raphael Cezar Sabbado >

}



map.ex

reduce.ex

Estrutura { do projeto }

```
> _build
> deps
✓ lib
  🔥 app.ex
  🔥 io.ex
  🔥 particao.ex
  🔥 reduce.ex
  🔥 teste-desempenho.ex
  🔥 usuario.ex
> test
  ≡ bench-aux.txt
  🔗 benchmark.exs
  ≡ input.txt
  🔗 mix.exs
  ≡ mix.lock
  ⓘ README.md
  ≡ sherk.txt
```

Testes { automatizados

▼ test

- 🔗 app_test.exs
- 🔗 nosso-reduce_test.exs
- 🔗 particao_test.exs
- 🔗 test_helper.exs

```
cleiane@Reily-PC2020:~/funcional/map-reduce-elixir-master$ mix test
.....
Finished in 0.05 seconds (0.00s async, 0.05s sync)
7 tests, 0 failures

Randomized with seed 563075
```

Modelo MapReduce {



}

01 [Partição]

Entrada:

```
[No, meio, do, caminho, tinha, uma,  
pedra,tinha, uma, pedra, no, meio,  
do, caminho,tinha, uma, pedra, no,  
meio, do, caminho, tinha, uma,  
pedra.]
```



Saída:

```
["No", "meio", "do", "caminho", "tinha", "uma"]
```

```
["pedra", "tinha", "uma", "pedra", "no", "meio"]
```

```
["do", "caminho", "tinha", "uma", "pedra", "no"]
```

```
["meio", "do", "caminho", "tinha", "uma",  
"pedra."]
```

02 [Map]

"caminho"



```
def map(elemento) do
  {elemento, 1}
end
```



```
{"caminho", 1}
```

03 [Concorrência]

- `Task.async(Enum, :map, [particao, funcaoMap])`
- `Task.async(NossoReduce, :reduce, [particao, funcaoReduce])`
- `Task.await_many()`



03 [Agrupamento]

- Junção dos elementos
`List.flatten/1`;
- Agrupamento de valores por chaves
em um Map; `Enum.group_by/3`
- Conversão de Map para List;
`Map.to_list/1`
- Particionar grupos novamente;

Saída:

```
[  
  {"No", [1]},  
  {"caminho", [1, 1, 1]},  
  {"do", [1, 1, 1]},  
  {"meio", [1, 1, 1]},  
  {"no", [1, 1]},  
  {"pedra", [1, 1, 1]},  
  {"pedra.", [1]},  
  {"tinha", [1, 1, 1, 1]},  
  {"uma", [1, 1, 1, 1]}  
]
```


map.ex

reduce.ex

02 [Reduce]

```
Partição: [  
  {"No", [1]},  
  {"caminho", [1, 1, 1]},  
]
```

```
def reduce(chave, lista) do  
  {chave, Enum.count(lista)}  
end
```

```
def reduce(particao, reduceUsuario) do  
  Enum.map(particao, fn {chave, lista} ->  
    reduceUsuario.(chave, lista)  
  end)  
end
```

Saída: {"No", 1} {"caminho", 3}

Otimização { < Partição >

Comparison:

particionar recursivo 50 part	167.31
particionar nao-recursivo 50 part	157.72 - 1.06x slower +0.36 ms
particionar nao-recursivo 300 part	128.94 - 1.30x slower +1.78 ms
particionar nao-recursivo 15 part	109.08 - 1.53x slower +3.19 ms
particionar recursivo 15 part	69.38 - 2.41x slower +8.44 ms
particionar recursivo 300 part	6.74 - 24.81x slower +142.32 ms

Name	Memory usage
particionar recursivo 50 part	12.21 MB
particionar nao-recursivo 50 part	12.21 MB - 1.00x memory usage +0 MB
particionar nao-recursivo 300 part	13.12 MB - 1.07x memory usage +0.91 MB
particionar nao-recursivo 15 part	13.96 MB - 1.14x memory usage +1.75 MB
particionar recursivo 15 part	14.75 MB - 1.21x memory usage +2.54 MB
particionar recursivo 300 part	135.50 MB - 11.10x memory usage +123.29 MB

Benchmark {


Comparison:


async-15	51.28			
async-30	50.67	- 1.01x	slower	+0.23 ms
async-100	48.78	- 1.05x	slower	+1.00 ms
async-5	45.97	- 1.12x	slower	+2.25 ms
sync	30.54	- 1.68x	slower	+13.24 ms

Comparison:

async-15	9.50 MB			
async-30	9.61 MB	- 1.01x	memory usage	+0.126 MB
async-100	9.56 MB	- 1.01x	memory usage	+0.0774 MB
async-5	9.46 MB	- 1.00x	memory usage	-0.02283 MB
sync	14.71 MB	- 1.55x	memory usage	+5.23 MB

Limitações {

[] Dados de entrada

[] Tolerância a falhas

[] Variedade de testes

[] Processamento distribuído

}

Execução {

< Entrada >

```

input2.txt
6520 azul gato luna comida toalha flor album gato ca
6521 gente casa bola album gato cachorro gente gente
6522 azul gato luna comida toalha flor album gato ca
6523 gente casa bola album gato cachorro gente gente
6524 azul gato luna comida toalha flor album gato ca
6525 gente casa bola album gato cachorro gente gente
6526 azul gato luna comida toalha flor album gato ca
6527 gente casa bola album gato cachorro gente gente
6528 azul gato luna comida toalha flor album gato ca

```

< Saída >

```

iex(13)> App.demo("input2.txt",4)
[
  {"album", 21084},
  {"amigo", 7028},
  {"arquivo", 6526},
  {"azul", 10542},
  {"bola", 3514},
  {"cachorro", 7028},
  {"casa", 14056},
  {"comida", 3514},
  {"flor", 3514},
  {"gato", 10542},
  {"gente", 24098},
  {"luna", 3514},
  {"mar", 7028},
  {"novidade", 3514},
  {"papel", 3012},
  {"pink", 4518},
  {"rosa", 3514},
  {"rua", 7028},
  {"toalha", 4520},
  {"toalhagente", 500}
]

```

1
2
3
4
5
6 **Obrigado!**
7
8
9

10 CREDITS: This presentation template was
11 created by **Slidesgo**, including icons by
12 **Flaticon**, and infographics & images by **Freepik**

13 < Please keep this slide for attribution >
14