ATIVIDADE 010: MORTE POR CONCORRÊNCIA

Paradigmas de Linguagens de Programação

- Anna Alicia
- Caio Andrade
- Fabrizio Honda
- José Jordan
- Vanessa Câmara

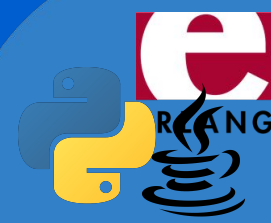
UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

DESCRIÇÃO DA ATIVIDADE

Morte por Concorrência em Java

1) Crie uma versão em Python
2) Crie uma versão em Erlang
3) Rode os 3 programas com
    - 100 processos e 100 mensagens
    - 1.000 processos e 1.000 mensagens
    - 10.000 processos e 1.000 mensagens
    - 100.000 processos e 10.000 mensagens

Anote e apresente os tempos.



VERSÃO PYTHON

```python
from threading import Thread
import sys
import time

class MiniThread(Thread):

    def __init__(self, th_n, m):
        Thread.__init__(self)
        self.th_n = th_n
        self.msg = m


    def run(self):
        while (self.msg):
            time.sleep(0.000001)
            self.msg -= 1


threads = int(sys.argv[1])
times = int(sys.argv[2])

for i in range(threads,0,-1):
    t = MiniThread(i, times)
    t.start()
```

VERSÃO ERLANG

```erlang
-module(morte).
-export([send/2]).


send(M, N) ->

  H = lists:foldl(
    fun(Id, Pid) -> spawnlink(fun() -> loop(Id, Pid, M) end) end,
    self(),
    lists:seq(N, 2, -1)),
    {, Time} = statistics(runtime),
    io:format("~p processes spawned in ~p ms~n", [N, Time]),
  statistics(runtime),
  H ! M,
  loop(1, H, M).

loop(Id, Pid, M) ->
  receive
    1 ->
      {_, Time} = statistics(runtime),
      io:format("~p messages sent in ~p ms~n", [M, Time]),
      exit(self(), ok);
    Index ->
      erlang:yield(),
      Pid ! Index - 1,
      loop(Id, Pid, M)
  end.
```

VERSÃO JAVA

```java
class MiniThread extends Thread {
    int n, t;
    MiniThread(int m, int th) { n=m; t=th; }
    public void run() {
        do {
            yield();
            n--;
        } while (n>0);
    }
}

public class Death {
    public static void main(String[] argv) {
        int threads = Integer.parseInt(argv[0]);
        int times   = Integer.parseInt(argv[1]);
        for(int i=threads;i>0;i--) {
            MiniThread t = new MiniThread(times, i);
            t.start();
        }
    }
}
```

COMPARAÇÕES

	100p 100m	**1.000p 1.000m**	**10.000p 1.000m**	**100.000p 10.000m**
Python	108 ms	3,077 s	31,990 s	Indefinido
Java	0,126 ms	0,503 s	4,139 s	5 m 48,119 s
Erlang	9 ms / 0 ms	410 ms / 4 ms	3742 ms / 1 ms	System Limit

PRINT PYTHON

```
annamilani@DESKTOP-TF89JP0:/mnt/c/Users/milan/Documents/PLP$ time python morte.py 100 100

real    0m0.108s
user    0m0.016s
sys     0m0.047s
annamilani@DESKTOP-TF89JP0:/mnt/c/Users/milan/Documents/PLP$ time python morte.py 1000 1000

real    0m3.077s
user    0m1.875s
sys     0m13.500s
annamilani@DESKTOP-TF89JP0:/mnt/c/Users/milan/Documents/PLP$ time python morte.py 10000 1000

real    0m31.990s
user    0m23.219s
sys     2m34.656s
```

PRINT JAVA

```
jordan@DESKTOP-5QGE0PB:/mnt/c/Users/jorda/Documents/PLP/erlang$ time java Death 100 100

real    0m0.126s
user    0m0.110s
sys     0m0.063s
```

```
jordan@DESKTOP-5QGE0PB:/mnt/c/Users/jorda/Documents/PLP/erlang$ time java Death 1000 1000

real    0m0.503s
user    0m0.673s
sys     0m0.643s
```

```
jordan@DESKTOP-5QGE0PB:/mnt/c/Users/jorda/Documents/PLP/erlang$ time java Death 10000 1000

real    0m4.139s
user    0m6.201s
sys     0m5.831s
```

```
jordan@DESKTOP-5QGE0PB:/mnt/c/Users/jorda/Documents/PLP/erlang$ time java Death 100000 10000

real    5m48.119s
user    9m32.668s
sys     12m56.218s
```

PRINT ERLANG

```
3> morte:send(100, 100).
100 processes spawned in 9 ms
100 messages sent in 0 ms
```

```
2> morte:send(1000, 1000).
1000 processes spawned in 410 ms
1000 messages sent in 4 ms
```

```
3> morte:send(1000, 10000).
10000 processes spawned in 3742 ms
1000 messages sent in 1 ms
```

OBRIGADO PELA ATENÇÃO!

- Link do GitHub:
  https://github.com/caioandrademota/programacao-concorrente