

Linguagens para Programação Paralela

December 10, 2009

- expressividade
- facilidade de uso
- integração com programas e programadores

Modelos de Linguagens Paralelas (lista não completa)

- paralelismo de dados (data-parallel)
- GAS e PGAS
- orientação a processos



- ênfase em processos e mensagens
 - communicating sequential processes
- escalabilidade de processos: tempo e custo
- Occam (super antiga mas com derivação Occam-Pi), Erlang

Erlang - Telefonia (1)

```
start(Sys, Proc) ->
    counter(Sys, Proc, 0).

counter(Sys, Proc, Tot) ->
    receive {
        Proc ? bump =>
            Tot1 = Tot + 1,
            counter(Sys, Proc, Tot1);
        Sys ? report =>
            Sys ! Tot,
            counter(Sys, Proc, Tot);
        Sys ? reset =>
            counter(Sys, Proc, 0).
```



```
-module(sys_start).
```

```
-export([init/2]).
```

```
init(Sys,0).
```

```
init(Sys,Max) ->
```

```
    Id = spawn(line_handler:start(port(Max))),
```

```
    spawn(counter:start(Sys, Id)),
```

```
    Max1 = Max - 1,
```

```
    init(Sys, Max1).
```

- processos peso *realmente* leve
- envio assíncrono
- recebimento síncrono com *patterns*

```
rpc(Name, Request) ->
    Name ! {self(), Request},
    receive
        {Name, Response} -> Response
    end.

loop(Name, Mod, State) ->
    receive
        {From, Request} ->
            {Response, State1} = Mod:handle(Request, State),
            From ! {Name, Response},
            loop(Name, Mod, State1)
    end.
```

- características funcionais
 - listas e map
 - variáveis com atribuição única
 - não há globais

```
map(_, []) -> [];
```

```
map(F, [H|T]) -> [F(H)|map(F, T)].
```

- oportunidade de paralelismo

Erlang: pmap

```
pmap(F, L) ->
  S = self(),
  Ref = erlang:make_ref(), %% we'll match on this later
  Pids = map(fun(I) ->
              spawn(fun() -> do_f(S, Ref, F, I) end)
            end, L),
  %% gather the results
  gather(Pids, Ref).
do_f(Parent, Ref, F, I) ->
  Parent ! {self(), Ref, (catch F(I))}.
gather([Pid|T], Ref) ->
  receive
    {Pid, Ref, Ret} -> [Ret|gather(T, Ref)]
  end;
gather([], _) ->
  [].
```



- occam e occam-pi
 - fortemente baseadas em CSP
 - envio e recebimento síncrono
 - uso de canais

```
INT x, y:  
  SEQ  
    x := 4  
    y := (x + 1)  
  CHAN INT c:  
  PAR  
    some.procedure (x, y, c!)  
    another.procedure (c?)  
  y := 5
```

- `erlang` J. Armstrong. Programming Erlang: *Software for a Concurrent World*. Pragmatic Bookshelf, 2007.
- `occam-pi` P.H. Welch and F.R.M. Barnes. Communicating mobile processes: introducing occam-pi. LNCS 3525 (25 Years of CSP), 2005.