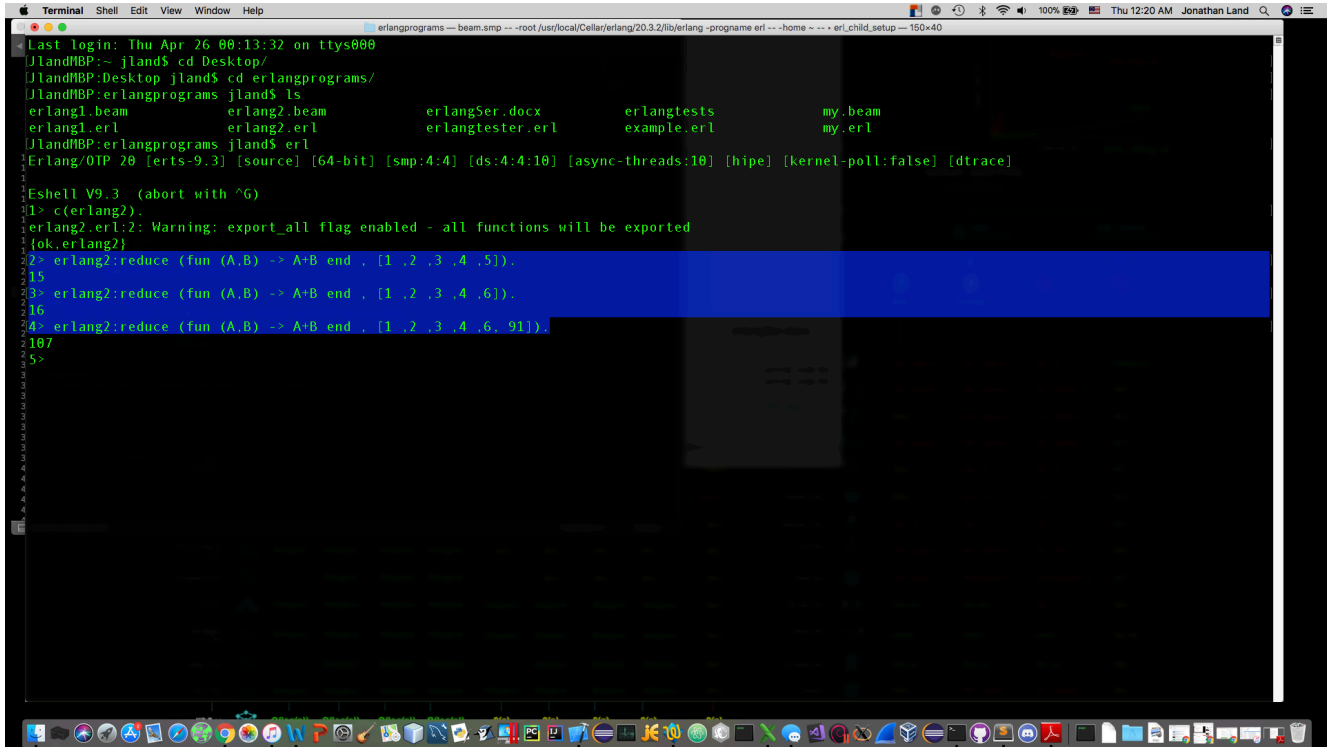


Erlang Parallelization

Jonathan Land

Parallelized program and screenshots: This program demonstrates associative and commutative operations being execute on even and odd sized lists using Erlang.



```
Terminal Shell Edit View Window Help
erlangprograms -- beam.smp -- -root./usr/local/Cellar/erlang/20.3.2/lib/erlang -progname erl -- -home -- -erl_child_setup -- 150x40

Last login: Thu Apr 26 00:13:32 on ttys000
jlandMBP:~ jland$ cd Desktop/
jlandMBP:Desktop jland$ cd erlangprograms/
jlandMBP:erlangprograms jland$ ls
erlang1.beam      erlang2.beam      erlangSer.docx    erlangtests       my.beam
erlang1.erl       erlang2.erl       erlangtester.erl  example.erl       my.erl
jlandMBP:erlangprograms jland$ erl
Erlang/OTP 20 [erts-9.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:10] [hipe] [kernel-poll:false] [dtrace]
1
1
1 Eshell V9.3 (abort with ^G)
11> c(erlang2).
1erlang2.erl:2: Warning: export_all flag enabled - all functions will be exported
1{ok,erlang2}
1
22> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,5]).
15
23> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6]).
16
24> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6,91]).
107
2
5>
```

```
Terminal Shell Edit View Window Help
erlangprograms -- beam.smp -- -root /usr/local/Cellar/erlang/20.3.2/lib/erlang -proname erl -- -home ~ -- -erl_child_setup -- 150x40

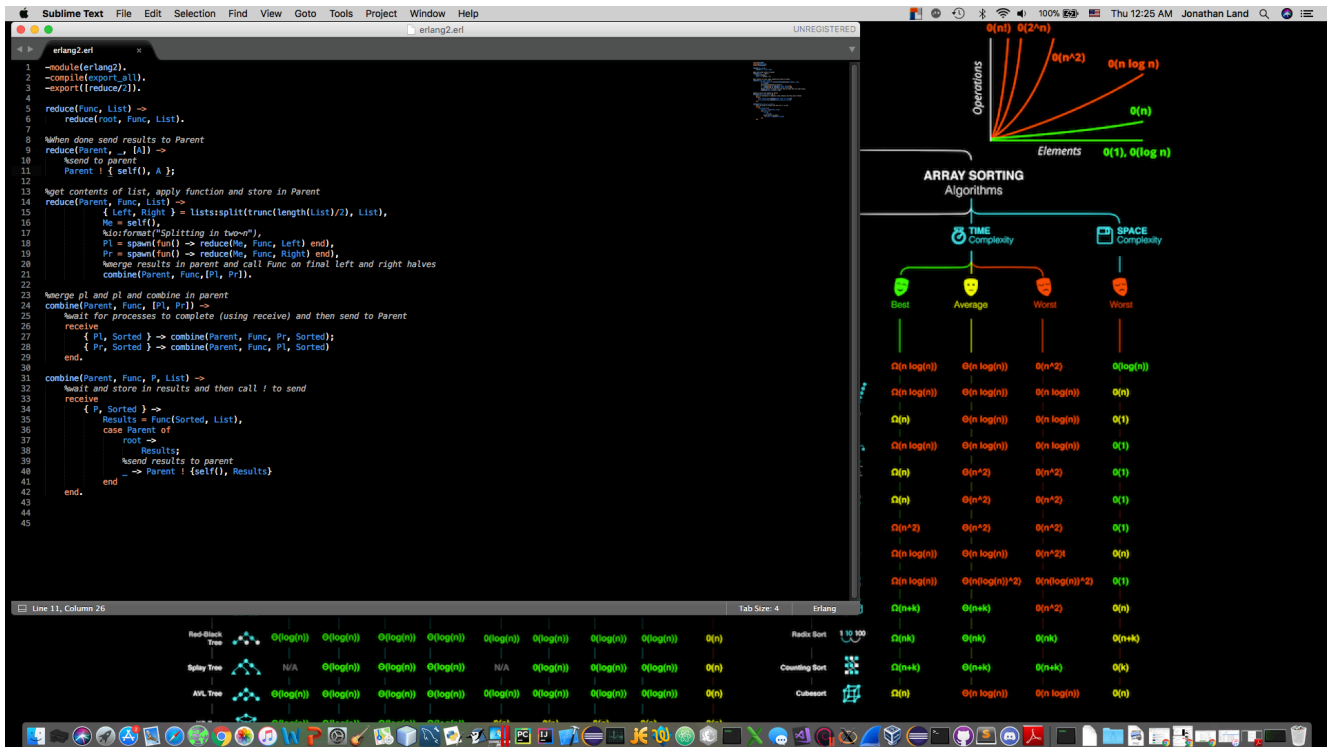
Last login: Thu Apr 26 00:13:32 on ttys000
jlandMBP:~ jland$ cd Desktop/
jlandMBP:Desktop jland$ cd erlangprograms/
jlandMBP:erlangprograms jland$ ls
erlang1.beam      erlang2.beam      erlangSer.docx    erlangtests       my.beam
erlang1.erl       erlang2.erl       erlangtester.erl  example.erl       my.erl
jlandMBP:erlangprograms jland$ erl
Erlang/OTP 20 [erts-9.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:10] [hipe] [kernel-poll:false] [dtrace]

Eshell V9.3 (abort with ^G)
1> c(erlang2).
erlang2.erl:2: Warning: export_all flag enabled - all functions will be exported
1 {ok,erlang2}
2 2> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,5]).
15
2 3> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6]).
16
2 4> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6,91]).
107
2 5>
```

```
Terminal Shell Edit View Window Help
erlangprograms -- beam.smp -- -root /usr/local/Cellar/erlang/20.3.2/lib/erlang -proname erl -- -home ~ -- -erl_child_setup -- 150x40

Last login: Thu Apr 26 00:13:32 on ttys000
jlandMBP:~ jland$ cd Desktop/
jlandMBP:Desktop jland$ cd erlangprograms/
jlandMBP:erlangprograms jland$ ls
erlang1.beam      erlang2.beam      erlangSer.docx    erlangtests       my.beam
erlang1.erl       erlang2.erl       erlangtester.erl  example.erl       my.erl
jlandMBP:erlangprograms jland$ erl
Erlang/OTP 20 [erts-9.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:10] [hipe] [kernel-poll:false] [dtrace]

Eshell V9.3 (abort with ^G)
1> c(erlang2).
erlang2.erl:2: Warning: export_all flag enabled - all functions will be exported
1 {ok,erlang2}
2 2> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,5]).
15
2 3> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6]).
16
2 4> erlang2:reduce (fun (A,B) -> A+B end , [1,2,3,4,6,91]).
107
2 5> erlang2:reduce (fun (A,B) -> A++B end , ["A", "B", "C", "D", "E "]).
"E DCBA"
2 6> erlang2:reduce (fun (A,B) -> A++B end , ["A", "B", "C", "D", "E"]).
"EDCBA"
2 7> erlang2:reduce (fun (A,B) -> A++B end , ["A", "B", "B", "B", "A", "D", ]).
* 1: syntax error before: ']'
2 7> erlang2:reduce (fun (A,B) -> A++B end , ["A", "B", "B", "B", "A", "G", "H", "I"]).
"IHGABBBBA"
2 8>
```



Erlang Code

```
-module(erlang2).
-compile(export_all).
-export([reduce/2]).
```

```
reduce(Func, List) ->
    reduce(root, Func, List).
```

```
%When done send results to Parent
reduce(Parent, _, [A]) ->
    %send to parent
    Parent ! { self(), A};
```

```
%I tried these to take care of one el in list, but it didn't work
%length ([]) ->
% Parent ! {self(), A};
```

```
%get contents of list, apply function and store in Parent
reduce(Parent, Func, List) ->
    { Left, Right } = lists:split(trunc(length(List)/2), List),
    Me = self(),
    %io:format("Splitting in two~n"),
    Pl = spawn(fun() -> reduce(Me, Func, Left) end),
    Pr = spawn(fun() -> reduce(Me, Func, Right) end),
    %merge results in parent and call Func on final left and right halves
```

```
combine(Parent, Func,[Pl, Pr]).
```

```
%merge pl and pr and combine in parent
```

```
combine(Parent, Func, [Pl, Pr]) ->
```

```
    %wait for processes to complete (using receive) and then send to Parent
```

```
    receive
```

```
        { Pl, Sorted } -> combine(Parent, Func, Pr, Sorted);
```

```
        { Pr, Sorted } -> combine(Parent, Func, Pl, Sorted)
```

```
    end.
```

```
combine(Parent, Func, P, List) ->
```

```
    %wait and store in results and then call ! to send
```

```
    receive
```

```
        { P, Sorted } ->
```

```
            Results = Func(Sorted, List),
```

```
            case Parent of
```

```
                root ->
```

```
                    Results;
```

```
                %send results to parent
```

```
                _ -> Parent ! {self(), Results}
```

```
            end
```

```
    end.
```