Assigments week 9

Joe Armstrong

February 25, 2014

Contents

1 Introduction

These problems are for lectures F12 and F13, held in week 9 of 2014. The problems are in two categories:

- **normal** solve these to get *Godkänd*.

- **advanced** solve these to get *Väl Godkänd*.

The sections and subsections in this paper are marked appropriately.

2 Parallel Maps

`smap(F, L)` (sequential map) maps a function `F` over a list of items `L` It is defined as follows:

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

In these exercises we will write a number of parallel mapping functions.

2.1 normal: pmap_max(F, L, MaxWorkers)

Write a function `pmap_max(F, L, MaxWorkers)` that produces the same return value as `smap(F, L)` by evaluating up to `MaxWorkers` elements of the list in parallel. Assume there are no exceptions raised when evaluating `F`.

2.2 normal: pmap_any(F, L)

Write a function `pmap_any(F, L)` that maps `F` over the list `L` in parallel but where the items in the returned list occur in the order they were computed.

Assume that:

```
fib(1) -> 1;
fib(2) -> 1;
fib(N) -> fib(N-1) + fib(N-2).
```

Then:

```
> pmap_any(fun fib/1, [35,2,20]).
[1,6765,9227465]
```

Why is this? `fib(2)` is computed quickly so is first in the list. `fib(35)` comes last since it takes a long time to compute. If we compute everything in parallel we should get the answer to `fib(2)` first.

2.3 normal: pmap_any_tagged(F, L)

This is similar to the last exercise, with the addition that the returned list should contain a copy of the input item that was processed. For example:

```
> pmap_any_tagged(fun fib/1, [35,2,20]).
[{2,1},{20,6765},{35,9227465}]
```

2.4 normal: pmap_any_tagged_max_time(F, L, MaxTime)

This is similar to the last exercise, but with a maximum total time (`MaxTime`) by which all parallel computations must terminate. `MaxTime` is a time in milliseconds. All on-going computations what have not terminated within time `MaxTime` must be killed. This means `pmap_any_tagged_max_times` will always return within `MaxTime` milliseconds.

For example:

```
> pmap_any_tagged_maxtime(fun fib/1, [35,2,456,20],2000).
[{2,1},{20,6765},{35,9227465}]
```

Note that the computation of `fib(456)` takes a very long time, and was aborted.

Aside: How long would it take to calculate fib(456) using the above program? Try estimating this. Can you think of a much faster way to compute fib(N)?

2.5 advanced: pmap_timeout(F, L, Timeout, MaxWorkers)

This map function returns the elements in the list in the same order as the original list. The arguments should be computed in parallel. If a computation fails, the return value in the list should be the atom `error`.

If the the computation of an individual element takes more that `Timeout` milliseconds then the computation should be aborted and the return value in the list should be the atom `timeout`.

No more than at most `MaxWorkers` processes should run in parallel.

For example:

```
> pmap_timeout(fun fib/1, [10,20,2000,abc], 2000, 3).
[55,6765,timeout,error]
```

The above call means "compute `fib(I)` for the elements `I` in the list `[10,20,200,abc]`. Return `timeout` if the computation of `fib(I)` takes more than 2 seconds, `error` if the computation of `fib(I)` fails and allow no more than 3 computations of `fib(I)` to run in parallel."

2.6 advanced: pmap_maxtime(F, L, MaxTime, MaxWorkers)

This is similar to the last exercise, but in this case `MaxTime` is to be interpreted as the *Total* time for all computations to take.

3 Components

3.1 normal: Simple Middle Man

Write a middle man to convert between metric and US systems of measurement.

For example if we send a term like `{km, 3}` to the input port, it should send `{miles,1.864}` to the output port.

See the lecture notes for a description of the middle man component.

3.2 advanced: Components

Implement the tracker and the tracker-client protocol described in the lecture notes for lecture F13.

Note: This problem has a rather vague specification, so you will have to make a number of design decisions to implement this. Document the design decisions you had to make and motivate your choices.