Exercises for lab 6 of CS2101a

Instructor: Marc Moreno Maza, TA: Xiaohui Chen

Thursday 17 October 2013

1 Exercise 1

In this exercise, we propose to use the notion of *parallel reduction* seen in class in order to compute the dot product of two vectors.

1. In preparation for that, write first a function `dotproduct_serial(U,V)` computing the dot product of the vectors of `U` and `V`
2. Next, write a function `dotproduct_parallel(U,V)` computing the dot product of the vectors of `U` and `V` using parallel reduction and thus Julia's construct `@parallel`
3. Run the following tests and record the timings

```
N = 1000000
U = [rand() for i=1:N]
V = [rand() for i=1:N]
@time dotproduct_parallel(U,V)
@time dotproduct_serial(U,V)

N = 100
U = [rand(100,100) for i=1:N];
V = [rand(100,100) for i=1:N];
@time dotproduct_parallel(U,V)
@time dotproduct_serial(U,V)
```

4. Explain these experimental results.

2 Exercise 2

Consider the Julia's function below for multiplying two square matrices of order `n`. Using Julia's construct `@spawnat` and `fetch` make a parallel version of that function that uses 4 processors.

```
function four_quadrant_mat_mul_serial(A, B, n)

        C = zeros(n, n)
        d = div(n,2)
        e = d+1
        C[1:d, 1:d] = A[1:d, 1:d] * B[1:d, 1:d] +
                      A[1:d, e:n] * B[e:n, 1:d]
        C[1:d, e:n] = A[1:d, 1:d] * B[1:d, e:n] +
                      A[1:d, e:n] * B[e:n, e:n]
        C[e:n, 1:d] = A[e:n, 1:d] * B[1:d, 1:d] +
                      A[e:n, e:n] * B[e:n, 1:d]
        C[e:n, e:n] = A[e:n, 1:d] * B[1:d, e:n] +
                      A[e:n, e:n] * B[e:n, e:n]
        C
end
```

3 Exercise 3

In this exercise, we propose to use the producer-consumer model studied in class in order to achieve the following: given a positive integer h print all prime numbers p such that the largest power of 2, say 2^e smaller than p satisfies $p - 2^e \leq h$. For small h, say $h \leq 64$, and p in the order of 2^{30}, such prime numbers are called *Fourier primes* and play an important role in areas like cryptography. Here are some hints:

- In Julia, the function `isprime` tests whether its argument is prime or not.
- The producer can be a *stream of primes*, that is, a function producing all primes one after another
- To test whether a prime number p writes $2^e + i$ for some positive integers e and i simply test if the following Julia's Boolean expression is true: `isinteger(log(p-i)/log(2))` .

4 Exercise 4

Modify the `pmap` Julia's code seen in class such that this function returns the following additional information: for each processor the number of items (from the input data list) this processor has handled.