Exercises for lab 7 of CS2101a

Instructor: Marc Moreno Maza, TA: Xiaohui Chen

November 28, 2012

1 Exercise 1

Remember the *merge-sort* algorithm:

`http://en.wikipedia.org/wiki/Merge_sort`

We saw last week that a naive parallelization of *merge-sort* leads poor speedup.

1. On an input array of size n, what is the work of your parallel *merge-sort*? What is the span? What do you conclude?
2. In order to increase the parallelism, we must parallelize the *merge* sub-algorithm. Propose a divide and conquer parallel algorithm for the *merge* sub-algorithm such that its span is essentially in $\Theta\log(n)$. If you cannot find a solution, look at these slides

   `http://www.csd.uwo.ca/ moreno/HPC-Slides/Analysis_of_Multithreaded_Algorithms.pdf`
3. Realize a `Cilk++` implementation of this parallel *merge* sub-algorithm and integrate it into your parallel *merge-sort*.
4. Analyze the work and the span of your program. To this end, you can conduct complexity analysis and/or use `cilview`.

2 Exercise 2

Read the section about matrix multiplication (again) in the slides

`http://www.csd.uwo.ca/ moreno/HPC-Slides/Analysis_of_Multithreaded_Algorithms.pdf`

1. Three schemes for matrix multiplication are proposed. Note that these are schemes, not complete algorithms. The second and third schemes are both divide and conquer. What is the difference between the two?
2. Which one is more suitable for an implementation targeting multicores.

3 Exercise 3

Consider the following `Cilk++` code fragment.

```
const int N = 50 * 1000 * 1000;

double A[N], B[N], C[N];

void vector_add()
{
    cilk_for (int i = 0; i < N; ++i)
        A[i] = B[i] + C[i];
}
```

1. Turn this code fragment into a complete `Cilk++` program which adds two large vectors randomly generated.
2. Analyze the work and the span of your program. To this end, you can conduct complexity analysis and/or use `cilview`.
3. In addition, measure the running of this program on $p = 4$ cores and the running time of its serial elision. Why is the speed up so low?
4. Is there a solution to this problem of poor performance?