WITS UNIVERSITY

School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS3008A: Parallel Computing
Exercise 6

2021-9-19

1 Objectives

- Understanding OpenMP `for`, `task` and `sections/section` constructs.
- Use OpenMP `task`, `sections/section`, `for`, `parallel`, and synchronization constructs to parallelize a serial program.

2 Problems

1. Implement

   (a) *quicksort* algorithms and

   (b) computing the *Fibonacci numbers*,

   respectively, both in serial and parallel. For your parallel version, use OpenMP `task` and `sections/section` constructs respectively. Compare the performances of these 3 implementations, i.e., serial, parallel using `sections/section` construct, parallel using `task` construct, by running the programs for different problem sizes and thread numbers. Put your results in a table like the one shown in Table 1. Consider the following question:

   (a) What are the differences of `sections/section` and `task` constructs?

   (b) Understand the usage of various clauses of `task` and `sections/section` constructs by applying them in your implementations where applicable, and observe the impacts on the performance of your programs.

2. Scan operation, or all-prefix-sums operation, is one of the simplest and useful building blocks for parallel algorithms Blelloch 1990. Given a set of elements, $[a_0, a_1, \cdots, a_{n-1}]$, the scan operation associated with addition operator for this input is the output set $[a_0, (a_0 +$

Table 1: Example table for benchmarking the performance of *quicksort*

Size	Seq	Par(sections/section)					Par(task)				
		1 thread	2 threads		4 threads		1 thread	2 threads		4 threads	
		Times(s)	Time(s)	Speedup	Time(s)	Speedup	Time(s)	Time(s)	Speedup	Time(s)	Speedup
10^5											
10^6											
10^7											
10^8											

$a_1), \cdots, (a_0 + a_1 + \cdots + a_{n-1})]$. For example, the input set is $[2,1,4,0,3,7,6,3]$, then the scan with addition operator of this input is $[2,3,7,7,10,17,23,26]$.

It is simple to compute scan operations in serial, see Listing 1.

```
1  scan(out[N], in[N]) {
2    i=0;
3    out[0]=in[0];
4    for(i=1;i<N;i++){
5      out[i]=in[i]+out[i-1];
6    }
7  }
```

Listing 1: Sequential algorithm for computing scan operation with '+' operator

Sometimes it is useful for each element of the output vector to contain the sum of all the previous elements, but not the element itself. Such an operations is called prescan. Tha is, given the input $[a_0, a_1, \cdots, a_{n-1}]$, the output of prescan operation with addition operator is $[0, a_0, (a_0 + a_1), \cdots, (a_0 + a_1 + \cdots + a_{n-2})]$.

The algorithm for scan operation in Listing 1 is inherently sequential, as there is a loop carried dependence in the `for` loop. However, Blelloch 1990 gives an algorithm for calculating the scan operation in parallel (see Blelloch 1990, Pg. 42). Based on this algorithm, (i) implement the parallel algorithm for prescan using OpenMP; and (ii) implement an OpenMP parallel program for scan operation based on the prescan parallel algorithm.

References

Blelloch, Guy E. (Nov. 1990). *Prefix Sums and Their Applications*. Tech. rep. CMU-CS-90-190. School of Computer Science, Carnegie Mellon University.