Umeå University
Department of Computing Science

Parallel Programming 7.5 p
5DV152

Exercises, Chapter/Topic 5

Submitted 2017-02-03
Author: Lorenz Gerber (dv15lgr@cs.umu.se lozger03@student.umu.se)
Instructor: Lars Karlsson / Mikael Ränner

Contents

Table 1 This table shows the identity values for various operators in C

operator	identity value
&&	1
||	0
&	$\sim$0
|	0
^	0

1 Introduction

This report is part of the mandatory coursework. It describes the solutions for several chosen exercises from the course book [**?**].

2 Identities for various reduction operators - 5.4

The identity values for various operators are shown in table 1.

3 Rounding errors - 5.5

It was assumed that in this exercise, one does not have to worry about how floats actually are represented in computers with sign, exponent and signigicant field. The interpretation is just based on how many digits are used to represent the given numbers. Hence in a), the sequence of values in the variable sum is $0.0, 2.0, 4.0, 8.0, 1010.0$. The last value is based on the fact that 1008.0 will be rounded from four digits in the register to 1010 with 3 digits in the variable. For b), sum of thread 0 is $0.0, 2.0, 4.0$, and sum for thread 1 is $0.0, 4.0, 1000.0$. After the reduce of sum from both threads, sum will be 1000.

It can be seen that the results from serial and parallel computation differ. This is because for floats, the sequence of addition matters when values are rounded due to too high precision.

4 Default scheduling - 5.6

A program was written to obtain the default scheduling of MP in for loops. Instead of indicating a range, indivdual indices for each thread are given. This is more flexible for various thread/iteration combinations when one thread can process two sequences of indices that don't follow each other. The source code can be found in appendix **??**.

2(1)

5 Loop-carried dependence - 5.8

6 Thread-safe string tokenizer - 5.16

7 Histogram A5.1

8 Count sort A5.3

9 Backward substitution A5.4

A C Source Code of Exercise 5.6

```
/**
 * omp_default.c
 *
 * MP program to determine standard
 * distribution of threads to for
 * loop indices.
 * @usage ./omp_default <threads> <iterations>
 */
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char* argv[]){
  int thread_count, iterations, i, j;
  int *assignments;
  int my_rank;

  thread_count = strtol(argv[1], NULL, 10);
  iterations = strtol(argv[2], NULL, 10);

  assignments = (int*) malloc(iterations * sizeof(int));

  # pragma omp parallel for num_threads(thread_count)
  for(i = 0; i < iterations; i++){
    my_rank = omp_get_thread_num();
    assignments[i] = my_rank;
  }

  for(i = 0; i < thread_count; i++){
    printf("Thread %d: Iterations ", i);
    for(j = 0; j < iterations; j++){
      if(assignments[j] == i){
printf("%d ", j);
      }
    }
    printf("\n");
  }

  return 0;
```

}