COMP 5426: Assignment #1

Due on Friday, April 17, 2015

B.Zhou 18:00

Shujian Zhou SID:450180358

April 17, 2015

Contents

1 Problem Definition and Requirements

1.1 Definition of Collapse Sets of Integers

The collapse of a set of integers indicates the sum of all integers in the set, while the collapse of a single integer is the sum of all its digits.
For the purpose of acquiring the ultimate collapse of an integer, which is a single digit, we should do the calculation recursively. In other words, we shall implement this procedure step by step, process the result of previous calculation, till we get a result as a single digit.

1.2 Requirements of Task

In this assignment, we are expected to write a program using the Master/Workers parallel programming paradigm will be employed to work out the ultimate collapse of the provided integer. In detail, this paradigm consists of two entities, the Master process and some of Worker processes. The Master process is responsible for decomposing the whole task into a number of small ones and distributed them to all Worker processes. on the other hand, it will also collect the partial results from Worker processes to produce the final result.In each time, Master would send a integer with fixed length to a Worker process, while the Worker process just execute the task and return the result. It can also be seen that the communication would only take place between Master process and every Worker process.

2 Parallel Algorithm Design

2.1 Applied Mathematical Theorem and Proof

Theorem 1. $UC(n) = (n-1)\ mod\ 9 + 1$

Proof. Assume N as an integer which consists of n digits, And it can be presented as:

$$\begin{aligned} N &= \sum_{i=0}^{n-1} q_i \cdot 10^i \\ &= \sum_{i=0}^{n-1} q_i + (\sum_{i=0}^{n-1} q_i \cdot 10^i - \sum_{i=0}^{n-1} q_i) \\ &= \sum_{i=0}^{n-1} q_i + (\sum_{i=0}^{n-1} q_i \cdot (10^i - 1)) \end{aligned} \tag{1}$$

$\because (10^i - 1)$ is a multiple of 9
$\therefore \sum_{i=0}^{n-1} q_i \cdot (10^i - 1)$is a multiple of 9 as well.

Therefore, N can be presented as

$$\sum_{i=0}^{n-1} q_i + 9\alpha \tag{2}$$

α presents $\sum_{i=0}^{n-1} q_i \cdot (10^i - 1)$ above. And any integer can be presented as

$$N = 9k + a \tag{3}$$

Here, we combine ② and ③.

$$N = \sum_{i=0}^{n-1} q_i + 9\alpha \quad = 9k + a \tag{4}$$

Then, we will have

$$\sum_{i=0}^{n-1} q_i = 9(k - \alpha) + a$$

This is the result of first collapse as $C(N)$.For $C(N)$, it can be processed as shown above:

$$\begin{aligned} C(N) &= \sum_{i=0}^{n-1} d_i \cdot 10^i \\ &= \sum_{i=0}^{n-1} d_i + (\sum_{i=0}^{n-1} d_i \cdot 10^i - \sum_{i=0}^{n-1} d_i) \\ &= \sum_{i=0}^{n-1} d_i + 9\beta \end{aligned} \tag{5}$$

Then, we repeat the procedure implemented on $\sum_{i=0}^{n-1} q_i$, we get:

$$\sum_{i=0}^{n-1} d_i = 9(k - \alpha - \beta) + a \tag{6}$$

We proceed with this procedure with P times and we get:

$$C(N_p) = 9k_{p+1} + a$$

If $k_{p+1} = 0$

$$UC(N) = C(N_p) = a$$

As $N = 9k + a$, $N \bmod 9 = a$
If $k_{p+1} = 1$ and $a = 0$, there will be:

$$UC(N) = C(N_p) = 9$$

As demonstrated above, we could prove that the equation

$$UC(n) = (n-1) \, mod \, 9 + 1$$

holds. □

Theorem 2. *The equation $UC(Z) = UC[UC(X) + UC(Y)]$ holds, where $Z = \sum_{i=0}^{n-1} q_i \cdot 10_i$, $X = \sum_{i=0}^{k-1} q_i$, $Y = \sum_{i=k}^{n-1} q_i$.*

Proof. As every integer can be written in such format as $N = 9k + a$, we can present X,Y,Z in such way below:

$$\begin{cases} Z = & 9k + a \\ X = & 9k_1 + a1 \\ Y = & 9k_2 + a2 \end{cases}$$

So, $Z = 9(k_1 + k_2) + a_1 + a_2$. And it can be seen that:

$$\begin{aligned} UC(z) &= a \\ &= UC(a_1 + a_2) \end{aligned} \tag{7}$$

Because $(a_1 + a_2)\ mod\ 9 = a$, the theorem can be proved to hold. $\square$

In this way, we can get the collapse of a integer by using the equation $UC(n) = (n - 1) \bmod 9 + 1$. Furthermore, owing to Theorem 2 provided above, we can divide the whole task in a number of parts and distributed to different processes.

2.2 Designing of Program

The program applied in this task employed these two theorems above. Specifically, this algorithm is applied in "Collapse()" function which implements the calculation of collapse.

At the beginning of the program, it will examine the number of process. The program will run sequentially while there is only one process. If there are multiple processes, the program will execute to another branch, in which Master process will divide the task and allocate them to those Worker processes. A variable "AllocateJobs" will be used to record the number of process which have been assigned for a task. When one of Worker processes return its result to Master process, Master process will not only retrieve the result, but also examine if there is remaining digits that have not been processed. Worker processes will stop working until Master get the knowledge that all digits have been processed.

3 Implementation and Testing

This program has been tested on all given samples and it computed correct result in every scenario. Here is the testing record of this program which shows an interesting thing that the running time increases as we employ more processors. I would like to make an assumption that the communication between Master and Worker consumes more time than that spent on computation. Because when we change the "Block size" from 1024 to 2048, which would decrease communication between Master and Worker. For the scenario of 5 processes and 9999999.tst, the running time decreases from 0.7175 to 0.3839.

Test sample	Num of procs	Time (Sec)
10001.tst	1	0.000256
	5	0.002878
9999999.tst	1	0.150396
	5	0.7175

4 Manual

Firstly extract and change direction to the folder, then compile the source code by simply typing "make" in terminal. After this, run the program by typing "mpirun -np numprocs collapse < sample.tst" command, the result and running time will be shown on screen.