Implementation of "Towers of Hanoi" Algorithm using Cilk 5.2: Parallelizing a Recursive Algorithm

Heber Irizarry and Julio Blasini
Advisor: Dr. Jaime Seguel

Department of Mathematics
University of Puerto Rico, Mayaguez, PR 00681
heber@sil.uprm.edu, jblasini@math.uprm.edu

Abstract

The "Towers of Hanoi" puzzle is one of the programs that has been studied, played and coded by many programmers around the world. Due to the magical complexity of the problem different approaches could be presented but some of them are only the ones that get the best achievement to resolve the task. The legend of the Towers of Hanoi appeared in a column of the Scientific American magazine:

"*In the great temple of Brahma in Benares, on a brass plate under the dome that marks the center of the world, there are 64 disks of pure gold that the priests carry one at a time between these diamond needles according to Brahma's immutable law: No disk may be placed on a smaller disk. In the begging of the world all 64 disks formed the Tower of Brahma on one needle. Now, however, the process of transfer of the tower from one needle to another is in mid course. When the last disk is finally in place, once again forming the Tower of Brahma but on a different needle, then will come the end of the world and all will turn to dust.* "

It can be represented as follows: There are 3 needles and a tower of disks on the first needle, with the smaller on the top and the bigger on the bottom. The idea is to move the whole tower to the last needle, by moving only one disk at a time and by observing not to put a bigger disk atop of a smaller one. To solve the problem we could use a recursive procedure. We want to show the different behaviors between a serial code and a parallel code in a 4-Processor computer running Linux (Kernel: 2.2.10). This last one written in CILK 5.2. The solution of the Hanoi Towers Problem is inherently recursive. Although recursive algorithms usually translate a serial code, the particularities of the Cilk environment allowed us to produce a higher parallel method.

1. Introduction

The "Towers of Hanoi" problem can be solved in a very simple recursive procedure.

We have to move tower A with n disks to tower C using tower B as an auxiliary tower (Fig. 1). To do this first we assume that in order to move all the disks to tower C, we have to move the biggest disk there, and this pole should be empty before doing so. Now we cannot move the biggest disk anywhere until all other disks are first removed. And we cannot be moved to pole C because we won't be able to move the biggest disk there. We should then move all other disks to pole B. After these steps, then we can complete this key step of moving the next big disk to pole C and go on with the remaining disks.

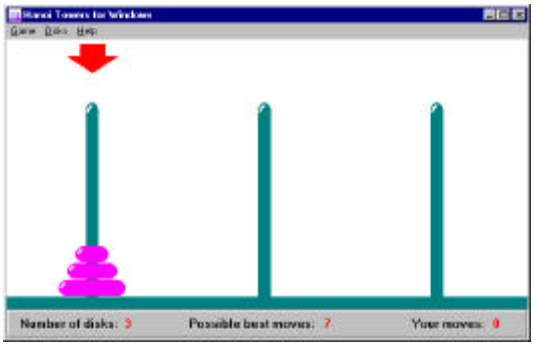

Figure 1. Towers of Hanoi (GUI version)

2. Algorithms

The following codes implement two different computer environments. The first one is written in Cilk 5.2 and the second one is a normal serial C code.

2.1 CILK 5.2 Code

```
#include <stdio.h>
#include <stdlib.h>
#include <cilk.h>

cilk void towers(int n, char del_p, char
al_p, char p_aux)
{
  if ( n == 1 )
    return ;

  spawn towers(n-1, del_p, p_aux, al_p);
  spawn towers(n-1, p_aux, al_p, del_p);
}

cilk main(int argc, char *argv[])
{
  int N = atoi(argv[1]);

  spawn towers(N, 'A', 'C', 'B');

  sync;

  return 0;
}
```

2.2 Serial Code

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

int counter;

void towers(int n,char del_p,char al_p,char
p_aux)
{
  if (n==1)
  return ;

 towers(n-1, del_p, p_aux, al_p);
 towers(n-1, p_aux, al_p, del_p);
}

int main(int argc, char *argv[])
{
  struct timeval tv1, tv2;
  double ttime;
  counter= 0;

  int N=atoi(argv[1]);

  gettimeofday(&tv1, NULL);

  towers(N, 'A', 'C', 'B');
 /*where N is the number of disks */

  gettimeofday(&tv2, NULL);

  ttime = (double) tv2.tv_sec -
          (double) tv1.tv_sec;

  printf ("Time: %lf\tSteps: %d\n", ttime, counter);

  return 0;
}
```

3. Data Analysis

Table 1 demonstrates the increase of the speed up as the number of disks increases. We can show this effect in the following mathematical analysis.

PARALLEL CODE			
Disks	Parallelism	Speed Up	Total Execution Time(s)
12	1.000	0.8	0.01
14	1.667	1.176	0.06
16	13.000	3.059	0.23
18	57.000	3.738	0.83
20	225.000	3.930	3.21
24	1855.000	3.991	51.46

Table 1. Parallel Code

Let x be a natural binary number ($0 < x <= 2^y-1$) with at most y digits. Moreover, let x have exactly y digits, $x \in \mathbb{N}$ with the rightmost digits having index 1 and the leftmost having index y. If we increment x by one, then exactly one digit changes from '0' to '1'. It can be proved that we need 2^y-1 moves to solve the Towers of Hanoi puzzle, for a tower of height y. The successive values of x will determine which disk to move. To determine where to move the disk we have to take into consideration that disk 1, for example, is the smallest, and remember where the disk was before and where is located now. Since there is only one possibility left we conclude that all of the disks move in cycles. The same will happen for the rest of disks. If we take a disk different from 1, the only possible positions are dictated by disk 1, remembering that we cannot move on top of disk 1 and that we cannot leave that disk in the same old place, there is again just one possible move.

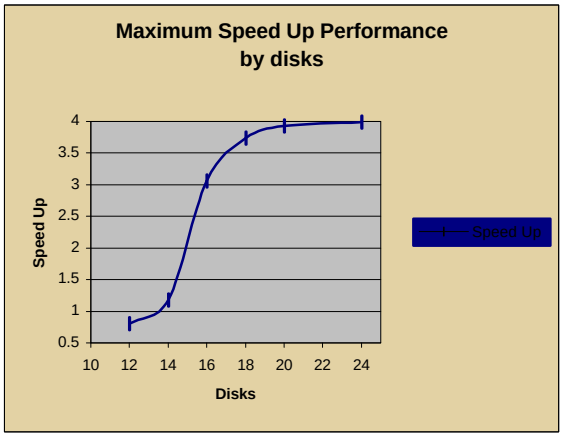

Figure 2

4. Conclusions

As stated in the introduction, Cilk 5.2 allowed us to transform a recursive method into a highly parallel one.

One of the most important advantages of the parallel code is that we can reduce execution time efficiently. When running the parallel code with just one processor the results were faster and better execution time than a serial code in one processor.

We have encountered that the parallelism and the speed up are increased significantly with the number of disks. Indeed, as the number of disks tends to infinity we approach the perfect speed up of 4 in a 4-processor machine. A multiprocessor machine works more efficiently when the number of disks is incremented constantly. A very important fact is that the variable should not be tested with a value of disks = 64. Our computation shows that it will take approximately ¼ billion years to get the answer.

Bibliography

- R. Douglas Hofstadter. Metamagical themas. Scientific American, 248(2): 16-22, March 1983
- Cilk 5.2 Reference Manual Supercomputing Technologies group, MIT http://supertech.lcs.mit.edu/cilk/
- Introduction to Parallel Computing: Design, Analysis of Algorithm The Benjamin/Cummings Publishing Corp. Inc. 1994 V. Kumar, A. Grama, A. Grupta, G. Karypis