IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

A SEMI AUTOMATIC TRANSFORMATIONAL TECHNIQUE FOR TRANSFORMING SINGLE THREADED PROGRAM INTO MULTI THREADED PROGRAM

Vinay T R[1], Ajeet A Chikkamannur[2]

[1]Assistant Professor, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bengaluru, Karnataka, India and affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India.

[2]Professor, Department of Computer Science and Engineering, RL Jallapa Institute of Technology, Doddaballapur, Bengaluru Rural, India

Abstract: Computer hardware technology has shown tremendous growth from simple uniprocessor to bigger and faster uniprocessors. In the recent past, these uniprocessors are replaced by multi-processor cores. To match these capabilities, the programming paradigms have also shifted from single-threaded sequential programming to multi-threaded parallel programming. But Legacy systems which were developed primarily to run on uniprocessors were left behind. This paper proposes to transform these applications to run on multi-processors at the same abstraction level (Implementation level) using semi-Automatic techniques. Here the legacy program is analysed based on dependency between functions and its signatures. They are then divided into smaller sub-programs (Parallel threads), which are executed on multi-core machines. The execution time of these restructured legacy programs were analyzed and observed ameliorated efficiency.

Keywords: Parallel threads; Re-engineering; Multicore machines; Slicing

1. INTRODUCTION

In the last decade, the computer hardware industry has seen faster and faster uniprocessors. In the recent past, their place is replaced by multi-processors containing multiple processor cores, that are expected to run multiple threads in parallel. Upgrading the hardware from uniprocessor to multiprocessor is primarily driven by microprocessor power utilization(faster processor consumes more power), limits on transistors and limitations of architectural design of uniprocessors. To match these capabilities, the programming paradigms have also shifted from single-threaded sequential programming to multi-threaded parallel programming. But Legacy systems which were developed primarily to run on uniprocessors were left behind. This legacy software's would run as a single threaded application. When up-gradation of hardware takes place from single core to multi-core. The difficult task here is to revamp these single threaded program into multi threaded parallel program, so that they can utilize these multi-cores available and run in parallel, thus improving the execution time of the legacy programs. This improvement affixes the procurement of new hardware technology.

All newer applications were designed and implemented as multi-threaded parallel applications keeping in mind that these will run on multi-processors. To match the features of multi-core machines, the software Designers, Engineers', programmers shifted from sequential programming to Parallel programming paradigm. Thus all newer/ modern applications were utilizing the advantage of multi-core machines and their efficiency (execution time) was also improved. Now the question comes what to do about the applications which were designed and developed for uniprocessors- legacy systems. The answer to this can be categorised into two paths, viz, Forward engineering and Re-engineering.

In Forward Engineering: The stake holders will evaluate the cost and time involved to modify the existing single threaded program into multi threaded program. If it is feasible, they will go for modifying it or completely discard this old software and start to build the new system. Here the software development team starts working from requirement phase to Design and Development of the new system following all the Software development life cycle phases. This method as some limitations such as collecting and validating the requirements. During design phase, the design engineers have to spend lot of time validating their design for completeness and it takes lot of human resources, design time and the infrastructure facilities. It also depends on current skill level of software development team and thus have exposure to Risk of failure of software system. The clients/customers will be sceptical to use this new system as they would be comfortable with the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

old system. To gain the confidence of the customers for this new system will be challenging task and time consuming. Acceptance testing has to be re-carried out by customer.

The second way is to Re-engineering [1,2] the existing software. Re-engineering is a process of modifying the code without affecting its original intent. Specifically use Reverse-engineering method to analyse the code and then re-implement it to suite for multiprocessors. In other words, it is migrating legacy system (single threaded application) into multithreaded parallel application. The modification is done at the same abstraction level (implementation part). The same is shown in figure 1. The advantages of this methodology is that it takes less time to migrate as the software engineer's will only recode it to suit for multi threading. The chances of failure are eliminated here as only existing system is modified rather than building new system from beginning. The cost of migrating is also low compared to building all new system as the cost of collecting, understanding and validating the requirements phase is not carried out. For other phases such as re-coding, testing has to be carried out thoroughly for confirming to correctness and completeness. The confidence of the clients/customers will be same as the old system as look and feel of the legacy system would not be modified.

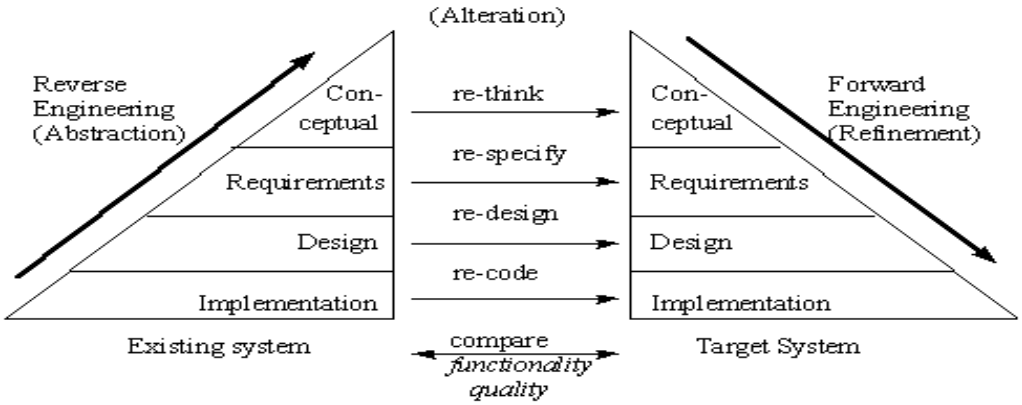

Fig. 1: Comparison between Forward and Re-engineering Approach.

The challenge in this methodology is that the legacy system should exhibit or posse's inherent parallelism. Additional programming effort by software engineers is required as they have to study, understand the legacy code and then recode it for multi-threaded model. In essence, it is software engineers' job to make full use of multiprocessors.

To aid this migration process, a novel methodology is designed where it will help the migration engineers to identify the independent and dependent functions using semi-automated technique.

2. LITERATURE SURVEY

Many works from academia and industry has been take up on issues regarding migration strategies. Most of these studies are done by specifically taking a particular legacy system.

i. Speculative multi threaded Architecture [11] splits a sequential program into smaller fragments called tasks. These tasks are then executed in parallel on multiple-cores. Hence they speculate that tasks are independent. The architecture provides hardware support to detect dependencies and roll-back misspeculations[11].

The limitations of this method are: Extra hardware units are used for identifying dependencies between tasks and to roll-back. It leads to great design complexity and high power consumption by hardware units.

ii. Executing Single threaded program on multi-core machine.

When the single threaded program runs on multi-core machine, it uses just one core for its execution. The other cores are not utilized at all. So the price and time involved for procuring and installing multiprocessor and running single threaded program on multi-core machine is not feasible. So just be upgrading the hardware technology and running old software will not make can improvement in efficiency. So once the hardware technology improved, the corresponding software's has to be redesigned to suit the hardware technology.

iii. Program Slicing Techniques[8]:

A computer program slice consists of the parts of a original computer program that (potentially) affect the values determined at some point of interest. Such a point of interest is mentioned as a slicing criterion, and is typically specified by a pair (program point, set of variables). The parts of a program that have a direct or indirect effect on the values computed at a slicing criterion C constitute the program slice with respect to criterion C. The task of computing program

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

slices is called program slicing. Weiser[10] defined a program slice S as a reduced, executable program obtained from a program P by removing statements, such that S resembles part of the behaviour of P.

iv. Werner Teppe[6], "ARNO Project: Challenges and Experiences in a Large-scale Industrial Software Migration Project"

The paper presents the process involved in migrating a large software project, the tools used for migrating the data and re-implementing the tasks. Here testing played a significant role and consumed significant part of resources. Finally the acceptance testing by client is of paramount important here.

v. Andreas Menychtas[7] et al. "ARTIST Methodology and Framework: A approach for the migration of legacy software"

The authors emphasis on first conducting Migration Feasibility Assessment before taking up Migration Project. If the expenditure is very high, projects that never end and migration risk and failures.

3. PROPOSED METHODOLOGY

The methodology used to restructure the legacy program at implementation level to suit the target environment (multi-core machine). This is depicted in the below given figure-2.

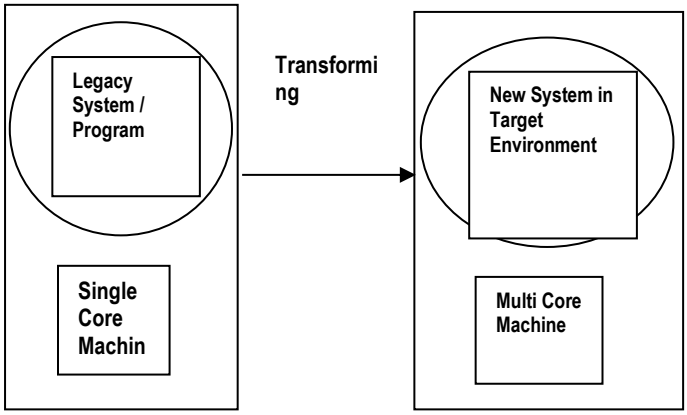

Fig. 2: Specific Model for Re-engineering

This method of Re-engineering has three specific advantages viz Reduced Time, Reduced Development cost and Reduced risk (risk of software failure).

Considering a sequential program made up of user-defined functions. The methodology scans the entire program, analyzes it, i.e., performs physiology of computer program. It builds Function-Signature table and from this table it categorizes the functions into two groups, Dependent functions and independent functions. It takes signature of function as a decision criterion. The schematic diagram of the overall process is given show in below in figure -3.

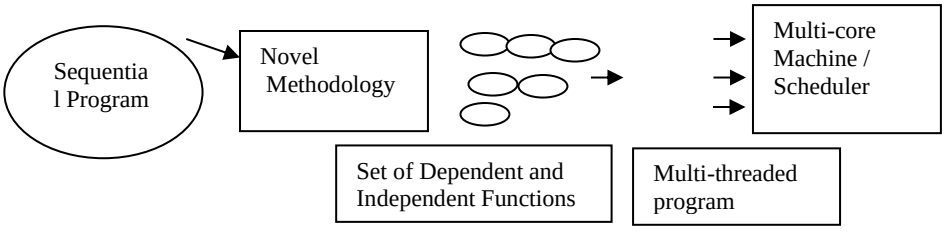

Fig. 3: Process flow graph

The Function-Signature table is a table consisting of function names as row heading and all the function parameter and return value (Signature of a function) as column heading. The last column is named as Relation-Status, where if there is dependency between functions, it is marked as D and if it is completely independent it is marked as IND.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

Signature of a function / Function Names	Parameter 1	Parameter 2	. . .	Return variable	Relation-Status
Function 1	**a**	**b**			**D**
Function 2				**b**	**D**
					...
Function N	**s**	**h**		**j**	**IND**

Table 1. The Function-Signature Table.

The algorithm for the construction of a novel Function-Signature table is given below.

Algorithm : To Construct Function-Signature Table

Input: A Legacy program consisting of numerous user-defined functions.
Expected Output: Entries into Function-Signature table and identifying relation/dependency between functions.

Step 0: Start
Step 1: Extract all variables used in the program, say $\mathbf{V_A}$.
Step 2: Construct a Table, where the row heading is function-name and column heading are actual parameters of the function (input variables of the function) and last-but-one column is the return variable of the function. The last column heading is the Relation-Status column to indicate whether this function is dependent or independent of each other.
Step 3: for each function call statement in the program, extract their actual parameters and return variable and insert it into new row of the table.
After the above step, the Function-Signature table will be filled with the corresponding values. From next step, the above constructed Table will be taken as input.
Step 4: For (i = first-row to last-row – 1) {
For(j = i + next-row to last-row) {
Compare(i and j rows actual parameters and their return variables) {
If(i's any one function parameter and j's return variable or vice-versa are same)
Then they are dependent and marked in the last column as Dependent (D).
End for (Inner loop)
End for (outer loop)

Step 5: if any row is not marked, it means they are independent functions. They are marked as IND.
Step 6: Now based on this marking, Segregate all Independent and Dependent functions.
Step 7: Transform the above Independent functions as Parallel threads and Dependent functions as sequential threads in program order.
Step 8: Schedule them to run on multi-core machine, so that they can be executed in parallel making use of all available multi-cores.
Step 9: Stop.
The complexity of this algorithm is $\mathbf{O(F^2)}$, where **F** is the number of user defined functions in the given program.

4. CASE STUDY AND RESULTS

Consider a sample legacy bench mark application and applying the above algorithm to find dependent and independent functions and constructing Function-Signature Table for the above application.
The sample functionality of this legacy program is as follows:

1. Begin
2. Declare and Assign a Vector Y[N] = {Assign Random Values} and N is the size of the vector.
3. Declare and Assign a Vector X[N] = {Assign Random Values}.
4. Declare and Assign a Vector Z[N] = {Assign Random Values}
5. Read a scalar value, Say A.
6. Read a Scalar Value, Say B.
7. Y[ ] = VectorMultiplication(Y,N,A) // Y[ ] = A * Y[ ]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

8. X[ ] = VectorAddition(X,N,B) // X[ ] = B + X[ ]
9. Z[ ] = VectorSubtraction(Z,N,X) // Z[ ] = X[ ] – Z[ ]
10. Print Vector Y
11. Print Vetor X
12. End.

Function Name	Parameter 1	Parameter 2	Parameter 2	Return variable	Relation-Status
VectorMultiplication( )	**Y**	**N**	**A**	**Y**	**IND**
VectorAddition **( )**	**X**	**N**	**B**	**X**	**D**
VectorSubtraction( )	**Z**	**N**	**X**	**Z**	**D**

Table 2. The Function-Signature Table of sample application.

The above table 2 clearly informs that the function VectorMulitplication( ) is independent one and the funtions VectorAddition( ) and VectorSubtraction( ) are dependent as these two functions have common variable (X) which is passed as a parameter in one function and a return variable in another function.

4.1 Program Analysis

The above legacy program is executed as a single threaded program and execution time is noted for different values of N (Vector size) .
Next, Two OpenMP threads are created, one thread is assigned to function VectorMultiplication( ) and the other thread is assigned to execute VectorAddition( ) and VectorSubtraction( ) . OpenMP directive will execute their threads in parallel. Here also execution time is noted for different values of N.

4.1.1 The Worst-case execution time

Data size (N)	Legacy Program in ms	Multi threaded Parallel Program (using OpenMP directive) in ms
100	76	82
1000	756	742
5000	3865	3028
10000	7259	6442

Table 3: Execution Time of Legacy Program and its restructured parallel Program.

Here the observation is that after transforming a single threaded program into multi threaded parallel program and running them on a multi core machine, there is improvement in execution time. Initially for less number of data size, there will be no improvement as creating parallel threads and maintaining them will take time. Only when the data size is big enough, we can observe the improvement.

CONCLUSION

The objective of this work is to develop a semi automatic technique for transforming single threaded program into multi threaded parallel program. The dependency between functions within a given program is analyzed based on construction and invention of Function-Signature Table. Here the decision criterion is based on function's actual parameter and its return variable. These information is compared with other function's details. Based on this, all Independent functions are transformed as Parallel threads and all dependent functions are grouped into a single sequential thread. These threads are executed on multi-core machine in program order. The transformation is done at the implementation level only thereby saving time, resources (human & infrastructure) and risk of failure is reduced. This also ameliorates the efficiency of legacy systems. Further this methodology can be extended for migration of legacy systems into cloud environment.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 12, December 2021

DOI: 10.17148/IJARCCE.2021.101251

REFERENCES

[1] Dr.Shivanand M.Handigund. Reverse Engineering of Legacy COBOL Systems.Doctoral dissertation, IIT, Bombay.
[2] Dr.Shivanand M.Handigund, Rajkumar N.Kulkarni "An Ameliorated Methodology for the design of Object Structures from legacy 'C' Program" , 2010 International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 13
[3] Pankaj Jalote. An Integrated Approach to Software Engineering, Third Edition, Narosa Publishing House.
[4] G.A.Venkatesh. The semantic approach to program slicing. In Proceedings of the ACMSIGPLAN'91 Conference on Programming Language Design and Implementation, pages 107–119, 1991. SIGPLAN Notices 26(6).
[5] R. Gupta and M.L. Soffa. A framework for generalized slicing. Technical report TR-92-07, University of Pittsburgh, 1992.
[6] Werner Teppe[6], "ARNO Project: Challenges and Experiences in a Large-scale Industrial Software Migration Project", IEEE Transaction 2009.
[7] Andreas Menychtas et al. "ARTIST Methodology and Framework: A approach for the migration of legacy software", IEEE Transaction 2014.
[8] Ferrante J., Ottenstein K. J., Warren J. D., The Program Dependence Graph and its use in Optimization, ACM Transactions on Programming Languages and Systems, Vol. 9, No. 3, July 1987.
[9] Ottenstein K. J., Ottenstein L. M., The program dependence graph in a software development environment, Proceedings of ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, Pages 85-97, July 1995.
[10] Frank Tip. A Survey of Program Slicing Techniques.
[11] T. D. Brown Jr., "C for Basic Programmers", Tata McGraw Hill Publishing Company Limited, New Delhi, 1992
[12] Vinay.T.R and Ajeet A C. "A Methodology for Migration of Software from Single-core to Multi-core Machine" International Conference on Computational Systems and Information Systems for Sustainable Solutions. IEEE Conference Publications: Pages: 367 - 369, DOI: 10.1109/CSITSS.2016.7779388 ,Bengaluru. 6-8th OCT. 2016.
[13] Gurindar S. Sohi and T.N. Vijaykumar, "Speculatively Multithreaded Architectures" Springer Series Edition, ISSN 1558-9412.

AUTHORS PROFILE

Vinay T R is currently working as Assistant Professor, Department of Computer Science and Engineering, NMIT, Bengaluru. He is having more than 08 Years of Teaching experience. He received his B.E and M.Tech degree in Computer Science and Engineering from Visveshwaraya Technological University, Belgaum. His area of interest are High Performance Security, Parallel Computing and Software Engineering.

Ajeet A Chikkamannur is currently working as Professor, Department of Computer Science and Engineering, RLJIT, Bengaluru. He is having over 20 Years of teaching experience and holds a doctorate in area of Database Management System from Visveshwaraya Technological University, Belgaum. His research area are Expert Systems, Software Engineering.