RESEARCH ARTICLE OPEN ACCESS

Achieving Load Balancing through Program Slicing

P.A.Tijare*, Dr. P.R.Deshmukh**
*(PhD Student, CSE, Sipna College of Engineering & Technology, Amravati, MS, India
Email: pritishtijare@rediffmail.com)
** (Department of Computer Science & Engineering, Amravati, MS, India
Email: pr_deshmukh@yahoo.com)

ABSTRACT
Implementing load balance in parallel program is very important. It may reduce running time and improve performance of program. This paper proposes a slicing algorithm in which we did not use any slicing criteria but we use slicing point. It is designed only for iterative programs as most of the programs or applications are developed for performing repetitive tasks. We found better results by achieving load balancing through program slicing.
***Keywords*:** Load balancing, parallel computing, partitioning, slicing

I. INTRODUCTION

Load balancing is nothing but the distribution of a load on the host system to other computing resources present in the network. The computing resources can be computer, computer clusters, network of workstations, CPUs, disk drives etc. The main aim to achieve load balancing is to utilize the resources in the network, minimize response time, maximize throughput and also to minimize the overload of any single system. Reliability and availability through redundancy can be achieved through load balancing [1]. Load balancing can be achieved by using program slicing. Program slicing is a method to simplify the programs by concentrating on selected part of semantics [2]. In program slicing, program is decomposed by analyzing their data flow and control flow [3]. Slicing transfers program to a minimal form by maintaining the original program behavior. Such minimal form is called as ‘slice’. Different sorts of slicing methods exist such as Forward Slicing, Backward Slicing, Static Slicing, Dynamic Slicing, Conditioned, Amorphous etc. The distinctive slicing strategies have different application domains such as software maintenance, program analysis, software testing, software optimization etc.

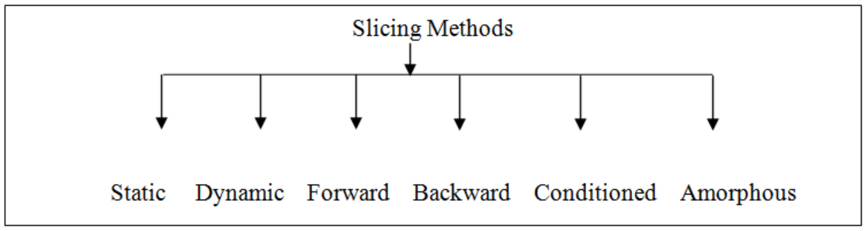

Figure 1: Variants of Slicing

While going for program slicing one should consider various parameters as follows:

Table 1: Parameters for Program Slicing

Parameter	Meaning
Slicing Point	Point where the programmer is interested repeat the number of the time till the logic of the program is complete
Slicing Variable	Variable mentioned in the slicing criteria
Scope	Intra procedural or inter procedural
Slicing Direction	Forward or Backward
Abstraction Level	Procedure or Statement
Information Type	Static or Dynamic
Result	Equivalent to the program of few set of statements from the program

1. Program Representation

To have an intermediate representation of a program is the best way to understand large programs. For computing slice, the program source code has to be transformed into intermediate representation. In most of the slicing algorithms, programs are represented by directed graph such as Control Flow Graph (CFG), Data Dependence Graph (DDG), Control Dependence Graph (CDG), Program Dependence Graph (PDG), System Dependence Graph (SDG), Extended System Dependence Graph (ESDG) etc. [4,5,6]

2. Need of Slicing

Program slicing techniques are used for various applications like program understanding, program verification, parallelization of a program, software probability analysis, program integration, compiler optimization etc.

II. PARALLEL COMPUTING

Parallel computing is nothing but the computation in which many computations or the process execution are carried out in parallel way. Frequently, large problems are divided into smaller parts, and then these small parts can be solved at the same time. There are several variants of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling.

To solve any computational problem, the simultaneous use of multiple compute resources, is nothing but parallel computing. Basic concept of parallel computing is to break down a computational task in several very similar sub tasks that can be processed independently and the results are combined after the completion of the task. One of the greatest barriers to achieve parallel computing is communication and synchronization in between different sub tasks. Effective use of Parallel Computers in practical applications can be achieved by proper Knowledge of Algorithm, Computer Architecture and Parallel Languages.

1. **To provide concurrency:** As single computing resource can only solve one thing at a time. Multiple computing resources can be doing many things simultaneously at a time.
2. **To solve larger problems:** Some computations are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially when limited computer memory is available

2.1 Partitioning

To design a parallel program, is to break the problem into discrete groups of work that can be distributed to multiple tasks. This is known as partitioning. There are two basic ways of partitioning computational work among parallel tasks: ***domain decomposition*** and ***functional decomposition***

To save time and/or money: As the number of resources are more to solve a task will reduce its time to complete, with potential cost savings.

To use non-local resources: Using computing resources on a network, when local compute resources are insufficient for demand.

To limit to serial computing: There are many reasons to apply significant constraints to build ever faster serial computers:

2.2 Parallelism

Parallel processing is an important part of any high performance computing model. Parallel Processing involves the use of computing resources such as CPU and Memory to complete the task. Parallel processing involves the division of a task into several sub tasks and making the system work on each of these smaller tasks in parallel. If multiple nodes or processor are engage in doing a computational task it execution time is faster than the single processor. Parallel processing improves the response time and throughput by utilizing all the computing resources in the network.

The goal of Parallel Processing is to obtain high performance with the minimum programming efforts, minimum resource requirements and with flexible architecture. We would like to obtain good speedup over the best sequential program that solves the same problem. This requires that we ensure a balanced distribution of work among processors, reduce the amount of inter processor communication which is expensive, and keep the overheads of communication, synchronization and parallelism management low.

2.2.1 Types of Parallelism

Various forms of parallel computing are:

1. Bit-Level Parallelism
2. Data Level Parallelism
3. Instruction-Level Parallelism
4. Task Parallelism
5. Loop Level Parallelism

Loops are the main target area in our work for parallelization. Loops are contained by most of the application. Long running application contain large loop that have many iteration to perform. We can divide the large number of iterations of such loops among the different processing nodes. If the amount of work performed in each iteration is roughly the same, simply dividing the iterations evenly across processors will achieve maximum parallelism.

III. SLICING THE PROGRAM

We have designed a slicing algorithm in which we did not use any slicing criteria but we use slicing point. It is designed only for iterative programs as most of the programs or applications are developed for performing repetitive tasks. So we consider "for loop" as our slicing point in our algorithm.

In the iterative programs, same code is repeated number of times and system has to execute all the code till the condition is true. Slicing point is the point from where we can slice the program. Number of slices can be created according to requirement or demand. Calculating slicing point value is dependent on number of slices to be created and initial & last value for all iterations will vary accordingly.

Steps for proposed Slicing Algorithm are as follows:

1. Form CFG of a program.
2. Identify slicing point in the program
3. Calculate initial and last value in iteration according to slicing point
4. Divide slicing point in number of iterations from last value of iteration by the formula (LV - FV) +1 / No of slices
5. Make slice ready for execution by inserting remaining program code into newly created slice

We can create the slices of the program as per users need since we are forming the slices by giving the value to the program to divide into n number of parts. New slice of the program is considered as a new program and will be transfer for execution on another system in the network.

IV. LOAD BALANCING PROCESS

When each task received, we are partitioning the work equally. We have evenly distributed the data set among the tasks, for those operations where each task performs similar type of work. For loop iterations where the work done in each iteration is similar, evenly distribute the iterations across the tasks.

The program slicer slices the program and converted into executable files. Now these files will be transferred to remote systems present in the network. After execution of slices on remote systems, the result files generated by each remote system will be transferred back to host system. After getting the result files back, the file output and time to execute slice on each remote system will be calculated.

Since we divide the program in to slices transferred them on number of remote systems to execute parallel by achieving load balancing, we shown the time required to execute overall program is reduces.

V. RESULTS

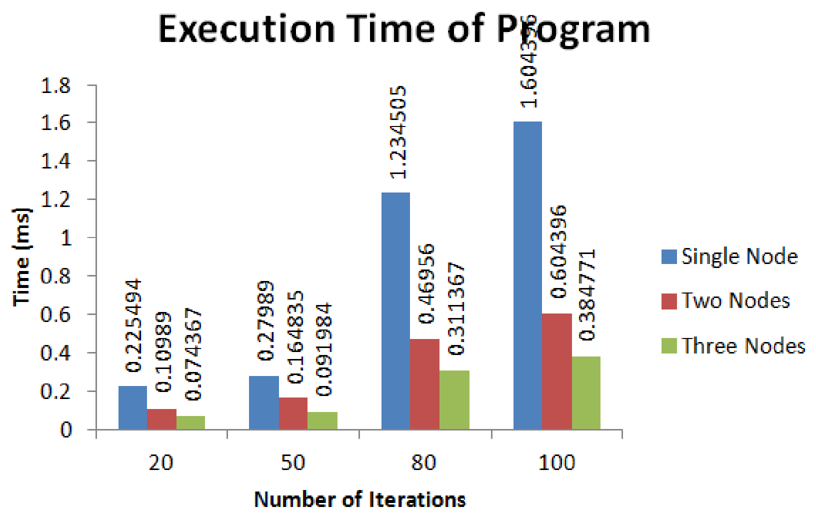

Figure: 2: Execution time of a program with different iterations

VI. CONCLUSION

Thus we have created the slices of the program as per users need by applying proposed program slicing algorithm. We are forming the slices by dividing a program into n number of parts. Thus we have achieved load balancing through program slicing in parallel computing.

REFERENCES

[1]. Amit Chhabra, Gurvinder Singh, Sandeep Singh Waraich, Bhavneet Sidhu, and Gaurav Kumar "Qualitative Parametric Comparison of Load Balancing Algorithms in Parallel and Distributed Computing Environment", Proc. World Academy of Science, Engineering and Technology (PWASET) ISSN 1307-6884, Vol 16, pp. 39-42, November 16, 2006
[2]. Mark Harman and Robert M. Hierons, "An overview of program slicing", http://www0.cs.ucl.ac.uk /staff/mharman/sf.html
[3]. Mark Weiser, "Program Slicing", IEEE transactions on Software Engineering, vol SE-10, No.4, July 1984
[4]. F. Tip., "A survey of program slicing techniques", Journal of Programming Languages 3(3):121–89, 1995
[5]. J. Ferrante, K. Ottenstein, and J. Warren. "The program dependence graph and its use in optimization", ACM Transactions on Programming Languages and Systems 9(3):319–49, 1987
[6]. K. Ottenstein and L. Ottenstein, "The program dependence graph in software development environment", SIGPLAN Notices 19(5):177–84, 1984