International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, June-2022

Parallelization of Minimax Algorithm for Game Theory using OpenMP

K Hari Hara Sudhan[°] ,S Gokul Srinath[:]
[°]Vellore Institute of Technology, Chennai, Tamil Nadu, India
[:]Vellore Institute of Technology, Chennai, Tamil Nadu, India

***Abstract*—Parallelism has long been used in elite registers, but it is gaining popularity due to biological limitations that limit repeated scalability. Parallel registering has become the dominant attitude in PC design in recent years, as multi-attention processors, as well as power utilisation with the help of PCs, have become a burden. OpenMP is a multiprocessing package that enables shared memory multiprocessing. The OpenMP programming paradigm is symmetric multi-processors, which means that while programming with OpenMP, all strings, percent memory, and statistics are available at the same time. OpenMP parallel programming identifies quantities that should be executed in parallel using a large request. A separate piece of code will be used to specify strings from the rest of the programme in order to keep walking in the same direction. The rule string is made up of the ace string. The slave strings continue to execute the same code in parallel. Our project's title was "Parallelization of the Minimax Calculation." To parallelize the minimax calculation of a game tree. The activity that is being played here is tic-tac- toe. The recommended parallel computational model combines the parallel algorithmic good models "administrator workers" and "offbeat emphases," and keeps each degree of the sport tree separate at width.**

Keywords: Parellelization, OpenMP, Minimax algorithm

I. INTRODUCTION

Search algorithms are an important aspect of many algorithms in computer science that have a wide range of applications, including database systems, expert systems, robot control systems, and theorem-provers. The search engines are at the heart of most game-playing systems. Branch and bound, minimax algorithm, alpha-beta pruning, and other search algorithms have been proposed to improve search efficiency in a variety of practical applications. A game tree is a tree with vertices signifying alternative game layouts and edges denoting possible moves from one position to another in game theory. The problem of tree searching is basic and computationally costly. The Minimax algorithm is a game tree search combinatorial optimization algorithm. A sequential game tree search algorithm searches the game tree with a single CPU. Multiple processors can be used in parallel computing to search at greater search depths in an acceptable amount of time.

Hyperthreading and multicore CPUs, as well as shared-memory clusters, have been widespread in recent years. As a result, shared memory parallel programming approaches are gaining traction as a genuine competitor to message passing. The hybrid (multi-level) parallel programming model combines the advantages of high-level parallelism provided by message-passing and low-level parallelism provided by loop level multithreaded parallelism. The effectiveness of the parallel minimax algorithm for game tree search on a multicomputer platform is studied in this work. A tic-tac-toe game was utilised as a case study. The proposed parallel computational model is based on the "manager-worker" parallel programming paradigm. The performance parameters achieved by message passing and multithreading parallel programming models are compared to those obtained by a hybrid (multilevel) programming model. On the basis of the experimental results, speedup and efficiency are investigated. The goal of the exper- imental results analysis and parallel performance profiling is to look into the algorithm's scalability in relation to the multi computer's size.

II. LITERATURE SURVEY

A. Alpha Beta Pruning

The Alpha Beta approach introduces parallelism by creating a concurrent processing programme for several child nodes at each level of the minimax tree. If the best move can be considered first among all the moves, the rest of the moves can be ignored right away. Unfortunately, the time consumed by sequential Alpha Beta pruning may be outweighed by the a priori quality of estimation. The root splitting algorithm serves as the foundation for our parallel Alpha-Beta implementation. The major goal of this technique is to ensure that each node, with the exception of the root, has just one processor assigned to it. To maintain the Alpha Beta pruning effect, we divided the root's offspring nodes into clusters, with each cluster served by only one processor.To maintain the Alpha Beta pruning effect, we divided the root's offspring nodes into clusters, with each cluster served by only one processor. Each child is then treated as the root of a sub search tree, which will be visited in order. When a child finishes computing, it sends its result to the root node, which judges which result is the best. During implementation, the mutual exclusion and critical section requirements are taken into account. Another optimization is implemented in addition to root splitting. Rearranging the children of a given node in such a way that the algorithm can begin investigating the most promising branches first, eliminating the undesired branches early and making the procedure faster. Furthermore, this

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, June-2022

approach is compared to one of the optimizations, beam search, which constructs a search tree using breadth-first search. It produces all the successors of the states at the present level at each level of the search tree, then sorts them in increasing order of their heuristic costs. By ignoring the least promising child nodes, the algorithm only investigates a subset of those child nodes, making the search faster.

B. Parallel Alpha-Beta Algorithm

The alphabeta algorithm's key source of parallelism is the simultaneous processing of many sibling nodes at any level of the game tree. If the best move is examined first, the rest of the moves can be denied with maximum efficiency in parallel. Regrettably, the a priori quality of a manoeuvre can only be judged heuristically. In the case of an unfavourable move order, parallel processing of nodes normally introduces some search overhead, which may exceed the amount of serial search. The PV-split approach, in which parallelism is used at the nodes on the principal variation, also known as PV- nodes or type 1 nodes, is the foundation for our parallel alpha- beta implementation. The leftmost child of each PV-node is searched on the CPU to determine the lower bound on the PV- node value in the GPU-based variation.The descendants of the remaining PV nodes are searched in parallel on the GPU with the narrower window. Multiple thread blocks on the GPU are used to parallelize the processing of sister nodes, however this is merely the high degree of parallelization provided by the GPU-based alpha-beta. On a lower level, the threads of the two-dimensional block process each node in parallel, with a single block per node. The block's dimensions may be smaller than the board's because: • using the smaller block is desirable when it results in higher GPU occupancy • The number of threads in the block is limited to 1024, which requires the use of smaller blocks for boards larger than 32×32 squares. If the block and board are of the same dimension, then the thread with index pair (x, y) is mapped directly to the square (x, y) of the board If the block isn't big enough to fit on the board. The alpha-beta technique is traditionally implemented as a recursive operation, however recursion is only supported by the latest generation of graphics cards. As a result, we used a manually controlled stack of nodes in shared memory to implement the iterative alpha-beta version. Node evaluation, move creation, and move execution are portions of the alpha- beta algorithm that are done in parallel by all threads in a block. A single thread handles the remaining administrative tasks, such as checking for cuts and updating scores. In parallel move generation, where each thread designates the corresponding square as a legitimate move, and parallel move execution, where each thread calculates the flipping of the corresponding square, thread divergence is the primary source of decreased efficiency. During the board evaluation phase, each thread assesses the associated square separately, after which the board value is calculated by summing the square values using the parallel reduction approach.

C. Hybrid Approach for Game Tree Search

According to prior studies, some difficult challenges exist while working with GPU, such as low pruning efficiency of parallel GTS algorithms, complexity of algorithm design for SIMD architecture, and low performance of divergence on GPU for rule-based computer games. To tackle these GTS problems, a node-based parallel technique that makes use of the GPU's capability can be used. Approach based on nodes:

1. For game tree search, use node-based parallel computing.

The tree-based method is incompatible with GPU architecture. The tree-based technique assigns processors to a set of nodes from one or more subtrees, whereas the node-based approach assigns processors to a set of nodes from one or more subtrees. The usage of this method not only takes advantage of the GPU's high concurrency, but also avoids the tree splitting's complexity. 2 The use of both depth-first and breadth-first search methods.There are two methods to search the tree, the depth-first search and the breadth-first search. For GPU based GTS algorithm, selection is the depth-first search on CPU because of memory limit and use breadth-first search on GPU. In the BFS approach, all threads assess nodes in parallel, while in the DFS method, the tree structure is traversed. 3. Programming on both the CPU and the GPU in a hybrid mode. BFS and DFS algorithms are used to achieve hybrid programming using a GPU-CPU combination. The CPU is responsible for maintaining the game tree structure, doing depth first search on the created tree, and interfacing with the GPU. The GPU receives tree nodes from the CPU and is in charge of assessing all nodes in parallel, i.e. a breadth first search. As a result, the GTS algorithm employs both CPU and GPU architecture. Architecture:The most common purpose of Game Tree Search is to find the players' moves in order to maximise their chances of victory. In the Game Tree, the game is divided into a large number of possible choices, which are referred to as possible moves, or the player's next move. Many of the game options are computed sequentially by the processor using the Depth First Search method. Using a tree-based method. Because of the SIMD technique on GPU, tree-based approaches are difficult to implement on GPU. The node-based approach has an advantage over the tree-based approach since the nodes and leaves are contained in the number of possible trees generated by the CPU. Creates a tree-like representation of the number of possible moves on the CPU. The CPU is in charge of controlling execution and maintaining the gametree structure. The evaluation of all nodes and leafs takes place on the GPU unit based on the number of threads. Using this hybrid approach, you may take advantage of DFS calculation on the CPU as well as BFS evaluation of nodes on the GPU.

D. Different Heuristics for Minimax Algorithm

The minimax theorem was initially applied to a zero-sum game in which two players were aware of all previous moves. The Connect-4 game is a chess game

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, June-2022

played on a board with seven vertical columns, each with six squares. Two players alternate moves until four men are linked horizontally, vertically, or diagonally. Once a man is placed in one of the columns, he will descend to the column's lowest unoccupied square. The maximum evaluation value will be chosen by Max, while the minimum value will be chosen by Min. Finally, Min will have just two options, resulting in four possible outcomes, all of which are terminal nodes, which signify that the search has come to a conclusion since the game has ended or the maximum search depth has been reached. The evaluation value will then be assigned based on the characteristics of the circumstance. The second layer's Min will select the smallest number of its children. Minimax employs a heuristic function to assess the present state of the game. Minimax's ultimate decision is heavily influenced by the heuristic function's performance. The unavailability of a square is determined not only by the presence of an opponent chessman in that square, but also by the ease with which that square can be filled with a chessman. When we raise the search depth of a somewhat weaker heuristic with a small number of features, that "weaker" heuristic can outperform its more feature-rich opponent.

III. Methodology

This research used the tic tac toe game as a case study. Asynchronous Iterations and Manager Work Algorithms have a role. It even makes use of multithreading. Speedup, effi- ciency, and scalability in relation to the size of the multicom- puter, as well as their impact on the performance of the parallel system, have been analysed based on experimental results. Tree searching is a simple but computationally expensive problem. Parallel computing is becoming more popular by the day. Theoretically, a group of parallel processors can deliver significant speedups. In practise, technological improvements have made it possible to build such devices at a reasonable cost. Given the inherent limits of ordinary combinatorial algorithms, these prospects appear to be particularly attractive for scholars engaged in the design and development of combi- natorial algorithms. The Minimax algorithm is a combinatorial optimization algorithm that uses a game tree search. Issues with knapsacks and travelling salesmen are two instances of such problems. The Minimax algorithm can be employed if there are two players in the game and they take turns playing with a specified number of possible movements for each position. A recursive approach for identifying a player's next move is a minimax search. It scores all possible game continuations to the goal level by evaluating each conceivable combination of moves. The search then moves up the game tree, selecting the highest and lowest child scores at nodes rep- resenting the first and second players, respectively. The alpha- beta minimax method searches the tree.In this experiment, the implementations of multithreading MPI and OpenMP parallel programming are employed.

IV. Modules Explanation

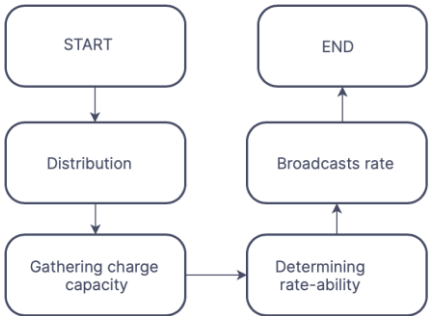


Fig. 1. Flowchart of the Modules

The parallel computational version of minimax set of regu- lations for are looking for in a pastime tree is primarily based on a mixture of parallel programming paradigms: "manager- worker" and "asynchronous iterations". The supervisor proce- dure is accountable for the following sports:

A. Procedure

- Distributes particular positions of the initial mark at the board for evaluation to the worker techniques.
- Gathers the tremendous charge capacity esteems and the first-rate actions controlled by means of manner of every of the employee processors at a given level of the sport tree.
- Determines the first-class rate ability esteem obtained thru all worker techniques.
- Broadcasts the tremendous rate bypass to all worker techniques.

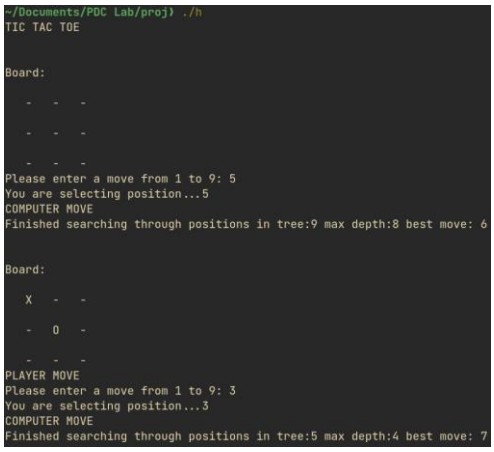
V. Software Requirements

Open MP

VI. Hardware Requirements

1. Main processor: Pentium 4 2. Processor Speed: 1000 MHz or More 3. RAM Size: 64 MB DDR or More 4. Keyboard: Standard QWERTY serial or PS/2 keyboard 5. Mouse: Standard serial or PS/2 mouse 6. Compatibility: AT/T compatible 7. Cache memory: 512 KB

VII. Outputs



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 06, June-2022

VIII. CONCLUSION

Hence, through this we will finish that parallelization mod- ified into finished in our software. It ends up achieved by way of which include numerous open MP functionalities. From the outcomes, the serial code for Program Execution time was 170ms whereas within the parallel code it has become 97ms. Future work would possibly contain examination of a class of video games just like tic-tac-toe however for areas of better dimensionality and video video games achieved for “m in a row” on k×n board

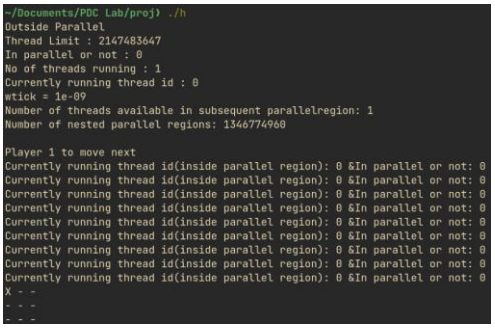


Fig. 2. Threading Process Outside Parallelization

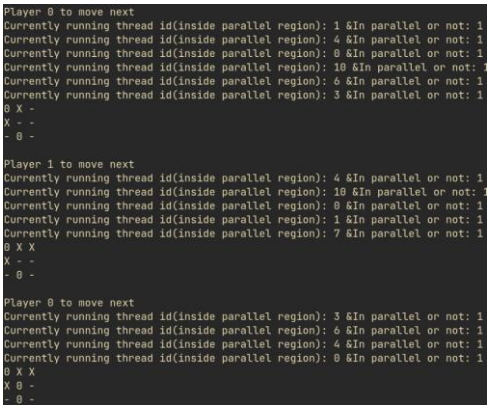


Fig. 3. Result of the Procedure

IX. REFERENCES

[1] arxiv.org/pdf/1908.11660.pdf
[2] 161.53.78.90/index.php/CIT/article/download/2029/1509
[3] ieeexplore.ieee.org/document/7033850