IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

# A Hybrid Approach for GTS Using Min-Max Algorithm on GPU and CPU

**Dipali V. Patil [1], Kishor N. Shedge [2]**

M.E. Student, Department of Computer Engineering, SVIT, Nashik, India [1]

Assistant Professor, Department of Computer Engineering, SVIT, Nashik, India [2]

**Abstract:** In field of the game theories and artificial intelligence Game-tree-search is the classical problem. The general use of GTS algorithm is in the real time applications having much higher complexity like video games, chess, connect4/connect6 etc. Different algorithms for game tree are used to search out the player's next best move on the game tree in minimum time. Main focus of system is on increasing massive parallelism abilities of GPUs to accelerate the speed of game tree algorithms and propose general parallel game tree algorithm on the GPUs. In game tree search, GPU surpasses CPU if there is highest level of parallelism is achieved due to its searching is in BFS manner and CPU is in DFS manner so that CPU didn't produce improvement. Here combination of DFS and BFS technique is main focus and appropriate selection will be the depth-first-search on CPU and use breadth-first-search on GPU and looks like hybrid CPU and GPU solutions.

**Keywords:** SIMD, MIMD, GPU, Connect6, Parallel Computing.

## I. INTRODUCTION

Many applications [4] [5] [6] have get advantage from the parallelism capability of GPU. Some AI issues will be simply resolved by GPU due to its SIMD design special for parallelism. GPU is stands for graphical process unit. Single instruction multiple data (SIMD) design of system having several process elements (PE) that perform constant operation on multiple information points at the same time and it exploits the information level parallelism. On the SIMD, single instruction computations are performed at a one time. CUDA development toolkit supports the parallel work and enforced on GPU. In Artificial Intelligence, game tree search is the vital approach and GTS is employed to seek out the best move for computer games. Parallel computation task on the GPU is performed as a concurrently execution thread blocks set. These are organized into a 1d grid or 2d grid. 1d, 2d or 3d grid with every thread selected by distinctive combination of indices. The hardware schedules the execution of blocks on the multiprocessors as units of thirty two threads referred to as warps. Computing on graphics process units handles computation just for computer graphics and handled by GPU, however computation in applications historically handled by the C.P.U.

### A. Game Tree

Game tree is the directed graph whose nodes are positions and edges are the moves. Complete game tree of game is the game tree beginning at the initial position and having all possible moves from every position. The Fig.1 shows the primary two levels, within the game tree for the game tick-tack-toe. Three choices of move has offered for 1st player: in the center, at the edge, or in the corner and also the second player has four choices for the reply if 1st player played in the center, otherwise two choices and game is continue. GTS is combinatorial problem thus difficult to search out an optimum solution for many type of games like Chess and Connect6; thus focus is use better GTS algorithms to get close-optimal solutions.

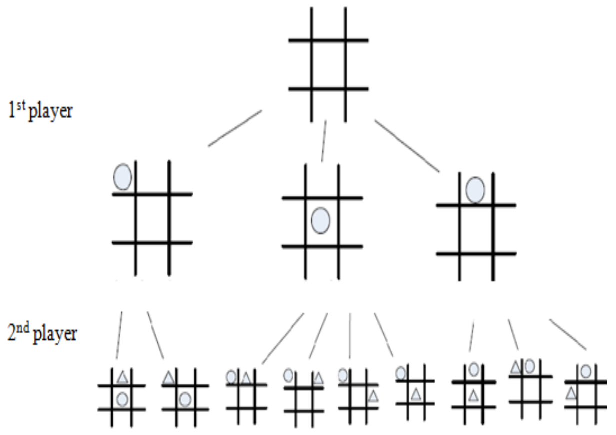

Fig1 Game tree of tick-tack-toe

## II. LITERATURE SURVEY

Since 1980's, number of parallel game tree search algorithms have been proposed. Different algorithms for the Game trees are described below.

Brockington and Schaeffer [9] were provide APHID: Asynchronous parallel game tree searching method. For finding out the minimax value as compare with the synchronous algorithms, asynchronous algorithms are better and efficient. APHID makes the algorithm easy to integrate into a sequential game-tree-searching program.

In comparison with synchronous searching methods, APHID having the better speedup.

P. Borovska and M. Lazarova [10] was proposed the minimax algorithm. This algorithm for the game tree search and divided into two stages i.e. first stage is for first player i.e. computer. Second stage for the second player i.e. human. The minimax algorithm try to find out the best move for first player i.e. computer even if second player i.e. human plays the best move over it. When it chooses the computer move it maximizes the computer score, at the other side minimizing that score by choosing the best move for the human player when human move is choose.
M. S. Campbell and T. A. Marsland [11] was proposed the algorithm that was negamax algorithm. Both the minimax and Negamax algorithms are similar with only one slight difference is that, it use the maximization function on place of using both maximization and minimization functions. That process is done by negating value that is returned from children from the opponent's point of view instead of searching for minimum score.

D. E. Knuth and R. W. Moore [12] was proposed the algorithm that is Alpha Beta algorithm. This algorithm form by doing some smart modification in the MiniMax and NegaMax algorithms. Moore and Knuth proved the things that is, time needed to search the tree can be reduced by pruning the many branches of the game tree and gives the same output as similar to the MiniMax or NegaMax algorithm. In the alpha beta algorithm, cutting the uninteresting branches of the game tree is the basic idea.

V. Manohararajah [13] presented the principle variation splitting algorithm. PVS is a tree based parallel GTS algorithm using multiple processor. In this PVS algorithm, the initial branch is marked by 1 as a principle node [24]. In game tree, nodes should be serially searched by first processor P0 before beginning of parallel search of other nodes. Other processor has to wait, for finishing the searching of previous one. One’s all processor finished their task, best move to player return by PVS. Drawback of PVS, processor who has completed their task needs to wait for another processor.

V. Manohararajah [13] presented the Enhanced principle variation splitting algorithm. EPVS avoid limitation of PVS algorithm and use the multiprocessor platform. In the EPVS algorithm, subtrees are assigned to idle processor from other busy processor So that efficiency and performance is increased. Extra communication overhead will be comes along with EPVS method.

R. M. Hyatt [14] was proposed Dynamic Tree Splitting algorithm for parallel GTS. Peer-to-peer model for multi-processor systems is used for DTS. In this split-points list (SP-LIST) were maintained by which all processors find uncalculated nodes to process. DTS algorithm is usable and scalable compared with PVS and EPVS.

## III. MOTIVATIONS

Major goal of GTS is that finding the best move of the player's that maximizes his/her probabilities of winning. For several computer games, hard to search out an optimum solution as a result of GTS may be a combinatorial problem within the field of game theory and it additionally having an exponential time complexity. Hence, looking for close to optimum solution is very important factor to accelerate the speed of GTS for real time applications like real time games on computer. Main motivation to use the GPU is that, it processes the thousands of game tree nodes in parallel and plenty of applications gets benefits from its parallelism capability. GPU having a lot of computing power, low power consumption and huge memory bandwidth; these factors make them a lot of applicable. CPU have few cores with various cache and it can handle only few software threads at just the once however on the opposite facet GPU having lots of core thus it can handle thousands of threads in parallel. Thus it's necessary to research that GTS will get benefit from GPU and compare with GPU-based approach with CPU-based approaches

## IV. PROPOSED SYSTEM

Some of the challenging problems arise when we are working with GPU unit and according to previous studies i.e. Low pruning efficiency of the parallel GTS algorithms, Complexity of the algorithm design for SIMD architecture, Low performance of divergence on GPU for rule-based computer games. To solve these GTS challenges, following node based parallel method to utilize the potential of GPU can be utilized efficiently.

### A. Node-based Approach

1. Adopt node-based parallel computing for the game tree search.
The tree-based approach isn't suite for the GPU design. The node based approach is assigning a group of nodes from one or multiple subtrees to processors, on other aspect the tree-based approach is assigned to processors. The utilization of method isn't only taking benefits of the high concurrency of GPU equally avoiding the complexity of tree splitting.

2. Combination of depth-first searching and breadth-first searching.
There are two strategies to search the tree, the depth-first search and also breadth-first search. For GPU based Game-tree-Search algorithm, choice is that the depth-first search on CPU due to memory limit and use breadth-first search on GPU. In BFS technique all threads evaluates node in parallel and for DFS traversing tree structure.

3. Hybrid programming on both CPU and GPU.
Hybrid programming is achieved through GPU-CPU combination severally, using BFS and DFS methods. CPU is maintaining game tree structure and perform depth first search on generated tree and conjointly

interacting with GPU. GPU takes tree nodes from the CPU is responsible for evaluating all nodes in parallel that is breadth first search is performed. Therefore, technique is to use both CPU and GPU architecture in GTS algorithmic program.

## B. Architecture of the system

The most common goal of Game Tree Search is finding of the players move so maximizes his probability of winning. In Game tree, game is spitted into several number of alternatives choices these are thought as possible moves that is next move for the player. Several of the alternatives of the games are computed as consecutive by processor in depth first Search manner using tree-based approach. Primarily tree based approach can't be simply employed in GPU as a result of the SIMD technique on GPU. Node-based approach is advantageous over tree-based approach because of during which C.P.U generating the number of possible trees contains the nodes and leaf. On the CPU, creates the number of possible moves in the style of tree. CPU is blaming for the the execution control and is responsible for the maintaining a gametree structure. On the GPU unit by number of threads, evaluation of all nodes and leafs takes place. Exploiting this hybrid approach takes a advantages of computation on the C.P.U. in the DFS manner in addition as evaluation of nodes by the GPU in a BFS manner.

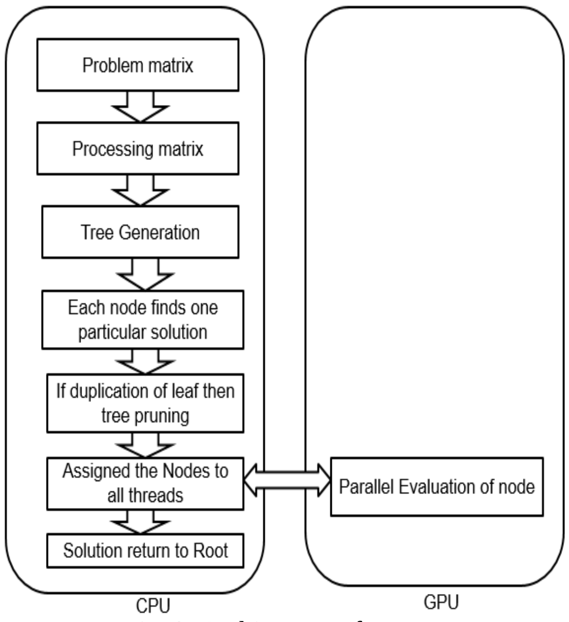
Problem matrix
Processing matrix
Tree Generation
Each node finds one particular solution
If duplication of leaf then tree pruning
Assigned the Nodes to all threads
Solution return to Root
Parallel Evaluation of node
CPU
GPU

Fig. 2. Architecture of system

Using such a combination of CPU and GPU the system design is form as shown within the Fig. 2 Input for the system in the form of problem data set referred to as matrix that is provided as an input to CPU. Once the matrix is provided as an input to the system, CPU generates variety of possible trees contains the nodes as well as leaf. CPU is performs operation like maintaining the tree structure, processing data, generation of the all nodes, and the tree pruning, conjointly performs checking of leaf nodes, and in the end solution returned to root node. Calculation of the many tree nodes is doing within the same depth in the current game tree, that breadth-first search (BFS). Additionally, every cycle in the search process will take in deepest nodes of the present game tree that is depth-first search (DFS). This means on DFS approach CPU works to calculate the nodes, since CPU will execute quicker as compare to GPU during this situation. And on the BFS approach, GPU used for calculating the branch and the leaf nodes within the parallel.

## C. Algorithm

Input:
Initial position of game tree search P0.
Output:
Best Move of player i.e. MBest.
Begin:
Step1: Set the P0 as root of the Game Tree Search
Step2: if Tree T Null
Step3: return
Step4: else
Step5: Tree formation/generation on the CPU
Step6: Node formation and structure maintain on the CPU
Step7 Depth First Search to process the tree and pruning the redundant nodes
Step8: Leaves, branch nodes are assigned to the GPU to calculate them concurrently
Step9: Calculates branch and leaf nodes in parallel as the Breadth first search
Step10: Updates parent node i.e. P0
Step11: Returns result i.e. MBest
End:

## V. EXPERIMENTAL RESULTS

For the connect4/connect6 game experiments are performed on the machine with CPU configuration Intel core i5 processor with 8GB RAM and GPU

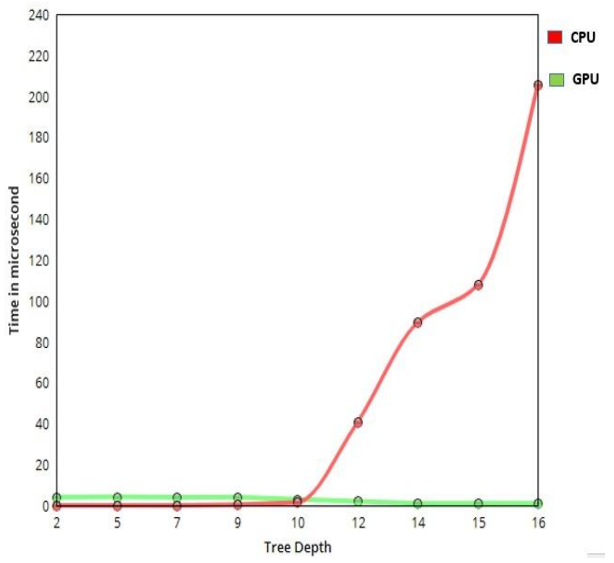

Fig. 3. Time comparision CPU vs. GPU

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

configuration NVIDIA GEFORCE GTX680 with 46 CUDA cores and 1024 threads with 2GB memory. CUDA version 6.2 is used for system.

Fig. 3 shows the time comparison CPU vs. GPU across the parameters depth and width of tree for connnect4/connect6 game. As the depth of tree increase, GPU time goes on decreasing and CPU time increases. Consider width of tree 7 in this case. CPU time exceeded as compare to the GPU.

Table I shows the time comparison CPU vs. GPU across the parameters depth and width of tree for connnect4/connect6 game. As the depth of tree increase with constant width 10 for all values of depth, GPU time goes on decreasing and CPU time increases.

TABLE I TIME COMPARISON CPU VS. GPU

| Depth | width | GPU(time in ms) | CPU(time in ms) |
|---|---|---|---|
| 8 | 10 | 8.42 | 0.7 |
| 10 | 10 | 8.41 | 17.26 |
| 20 | 10 | 8.42 | 358.5 |

Time comparison CPU vs. GPU across the parameters depth and width of tree for connnect4/connect6 game showed in Table II. As the depth of tree increase with constant width 12 for all values of depth, GPU time goes on decreasing and CPU time increases.

TABLE II TIME COMPARISON CPU VS GPU

| Depth | width | GPU(time in ms) | CPU(time in ms) |
|---|---|---|---|
| 2 | 12 | 45.47 | 0.009 |
| 5 | 12 | 44.23 | 0.194 |
| 10 | 12 | 44.53 | 377.05 |

## VI. CONCLUSION

Main focus of the system design is on the Parallelization of the Node based Game Tree Search Algorithms on CPU and GPU. Parallel algorithm for game tree search presented the approach that is node based approach for gaining the fast optimal solution of the real time computer games on the Graphics processor that is GPU. Using the node based computing and combining the DFS and BFS on the CPU-GPU units respectively.

With the help of this hybrid combination on the CPU and GPU architecture, this approach taking the capability of GPU for computing massive nodes parallely and CPUs flexibility for tree pruning. This approach can be tested on connect4/connect6 games and results show that node based algorithm for parallel implementation gains the speed over the serial implementation of the GTS. Improving the more game tree GTS algorithm and applied it with large scale cluster of GPU will be consider as future work.

## REFERENCES

[1] Liang Li, Hong Liu, HaoWang, Taoying Liu, Wei Li, "A Parallel Algorithm for Game Tree Search using GPGPU", IEEE Transaction on Parallel and Distributed Systems, aug, 2015.
[2] K. Rocki and R. Suda, "Parallel minimax tree searching on GPU," in Parallel Processing and Applied Mathematics, vol. 6067, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., Berlin, Germany: Springer, pp. 449–456, 2010
[3] D. Strnad and N. Guid, "Parallel alpha-beta algorithm on the GPU," Journal in Computing and Information Technology, vol. 19, no. 4, pp. 269–274, 2011.
[4] X. Huo, V. T. Ravi, W. Ma, and G. Agrawal, "Approaches for parallelizing reductions on modern GPU," International Conference on High Performance Computing (HIPC), pp. 1–10, 2010.
[5] W. Ma and G. Agrawal, "An integer programming framework for optimizing shared memory use on GPU," International Conference on High Performance Computing, pp. 1–10, 2010.
[6] J. Soman, M. K. Kumar, K. Kothapalli, and P. J. Narayanan, "Efficient Discrete Range Searching primitives on the GPU with applications", International Conference on High Performance Computing (HiPC), pp. 1-10, 2010.
[7] C. E. Shannon, "Programming a computer for playing chess", Philosophical Magazine Series 7, 41(314):256-275, 1950.
[8] G. Karypis and V. Kumar, "Unstructured tree search on SIMD parallel computers," IEEE Transaction on Parallel and Distributed Systems, vol. 5, no. 10, pp. 1057–1072, 1994.
[9] M. G. Brockington and J. Schaeffer, "APHID: Asynchronous parallel game-tree search," Journal of Parallel and Distributed Computing, vol. 60, no. 2, pp. 247–273, 2000.
[10] P. Borovska and M. Lazarova, "Efficiency of parallel minimax algorithm for GTS," in Processing International Conference on Computer Systems and Technologies, pp. 14:1–14:6, 2007.
[11] M. S. Campbell and T. A. Marsland, "A comparison of minimax tree search algorithms," report on Artificial Intelligence, University of Alberta, Canada, vol. 20, no. 4, pp. 347–367, 1983.
[12] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," Artificial Intelligence, vol. 6, no. 4, pp. 293–326, 1975.
[13] V. Manohararajah, "Parallel alpha-beta search on shared memory multiprocessors," Master‘s thesis, Graduate Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada, 2001.
[14] R. M. Hyatt, "The dynamic tree-splitting parallel search algorithm," ICCA Journal, vol. 20, no. 1, pp. 3–19, 1997.
[15] M. G. Brockington, "A taxonomy of parallel game-tree search algorithms," ICCA Journal, vol. 19, pp. 162–174, 1996.
[16] R. M. Karp and Y. Zhang, "On parallel evaluation of game trees," in Proccesing 1st Annual ACM Symposium on Parallel Algorithms and Architecture, pp. 409–420, 1989,
[17] H. Wang, S. Potluri, D. Bureddy, C. Rosales, and D. K. Panda, "GPU-aware MPI on RDMA-enabled clusters: Design, implementation and evaluation," IEEE Transaction on Parallel and Distributed System, vol. 99, 2014.
[18] J. E. McClure, H. Wang, J. F. Prins, C. T. Miller, and W. Feng, "Petascale application of a coupled CPU-GPU Algorithm for simulation and analysis of multiphase flow solutions in porous medium systems," in Processing 28th IEEE International Parallel and Distributed Processing Symposium, pp. 583–592, 2014.
[19] K. Rocki and R. Suda, "Large-scale parallel monte carlo tree search on GPU," in Processing IEEE International Parallel and Distributed Processing Symposium, pp. 2034–2037, 2011.
[20] Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-Stover, "GPU cluster for high performance computing," in Processing ACM/IEEE Conference on Supercomputing, pp. 47–53, 2004.
[21] J. Zhang, H. Wang, H. Lin, and W. Feng, "cuBLASTP: Fine-Grained parallelization of protein sequence search on a GPU," in Processing 28th IEEE International Parallel and Distributed Processing Symposium, pp. 251– 260, 2014.