ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Node Based Parallel Game Tree Search Algorithm Using Accelerated GPU

Rutuja U. Gosavi[1], Prof. Archana S. Vaidya[2]

Student, Dept of Computer Engineering, G.E.S's R. H. Sapat College of Engineering, Nasik, India[1]

Assistant Professor, Dept of Computer Engineering, G.E.S's R. H. Sapat College of Engineering, Nasik, India[2]

ABSTRACT: Game tree Search algorithm is used to search the best move in game tree. GTS is a combinatorial problem in which it is hard to find optimal solution from huge possible solutions. Focus of the system is to take advantage of GPU's massive parallelism capability to accelerate the speed of game tree algorithm and propose a concise and general parallel game tree algorithm on GPUs. GPU computing is getting popular among scientific community because of cheap and high performance computational power. Proposed system is implemented for Connect4 and Connect6 game using CUDA and MPI programming environment. It is found that parallelization tasks on SIMD processors of graphics cards perform better during searching and evaluating a GTS. In this CPU is responsible for maintain tree structure of game tree and GPU is responsible for evaluating node simultaneously. Thus choice is to use combination of CPU-GPU solution with DFS-BFS search respectively. Comparison is done with serial implementation for Connect4 and Connect 6 games.

KEYWORDS: Connect4/Connect6, CUDA, Game Tree Search, GPU, Parallel Computing

I. INTRODUCTION

GPU is a powerful support for massive parallel computing for real time applications. The GPU is processor as a multiple core having support for many of threads running parallel. GPUs are outcome of many processors with core aligned in a way that forms a unit of hardware. Cores may be a hundred or more as per specification. General purpose CPU tied applications which have important data in-dependency suits to the devices. So, parallel data or parallel codes perform efficiently since hardware can be classified as SIMD. Parallel work is implemented on GPU with the CUDA development environment.

CUDA stands for Compute Unified Device Architecture. It is framework or platform for parallel computation and also programming model created by NVIDIA and implemented by the GPUs. As CUDA is framework for parallel work it offers direct access of memory and virtual instruction set to the developers. Many applications are getting benefits from GPU massive parallelism capacity [2][3][4]. GPU used for solving Artificial Intelligence queries successfully [5]. GPU is another way of solving Artificial Intelligence queries or problems that are generally compute intensive because of its SIMD architecture specialized for parallel computing. The SIMD is termed as Single Instruction Multiple Data architecture. It is used to performing same operation or instruction on multiple data simultaneously. Hence such devices perform concurrent computation but uses only one instruction, so these exploit a data level parallelism. GTS is key approach in AI as it used to choose nest move or best move in computer games or real time applications.

A. *Game Tree Search on GPU*

Game tree search is digraph, nodes in digraph indicate position and lines or edges denote moves in a game as it mentioned in a game theory. From game of point view, the complete game tree is the hierarchical game tree structure begins at the initial position as a node and containing all possible moves from each position. It is very hard to find optimal solution for taking best move for many computer games as it contains exponential time complexity, games like Connect4/Connect6 [6], Sim, Chess [7], Havannah etc. The focus on GTS algorithm to obtain near-optimal solutions using node based approach. It is used to speed up or accelerating the GTS algorithms for the computer.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

B. *Necessity of Parallelism*

To satisfy a demand of reduction in a computational time for games, parallel computing needed to introduced an improve GTS algorithm performance. General purpose CPU-based parallelism approaches have been studied for many years [8][9]. In CUDA programming model parallelism is achieved through its set of parallel threads or multi-threaded architecture. These threads are organized into number of blocks and combination of blocks forms grids of thread blocks.

To each grid the kernel executes concurrently. Kernel is a user defined C function which is executed in GPU. On GPU parallel computation is performed by executing thread blocks concurrently. These are arranged into a Dimensional structure like 1D, 2D or 3D manner as shown in figure 1. The CUDA threads are organized into a two-level hierarchy using unique coordinates called block ID and thread ID. Hence each thread can be uniquely identified by its ID which is represented by the built in variable blockldx and threadIdx. A group of 32 threads with consecutive thread IDs is called a Warp, which is the unit of thread scheduling. Computation on graphics cards is taking complete advantage of usage of GPU which performs operations which are related to the computer graphics, which was performed by CPU traditionally. GPU achieves a higher performance parallelism in some specific tasks as compare to the conventional processor. Thus use of multiple graphics cards in single computer, even in large numbers of graphics chips helps for parallelizing the already parallel nature of graphics processing. Thus basic tasks are decomposed into small one that can be further evaluated or processed concurrently by GPU. It improves the computational time to find answer which is the feasible optimal solution. For getting right solution we are taking advantage of SIMD nature of GPU that allocating instruction to the number of threads and they works on finding next move of the game.

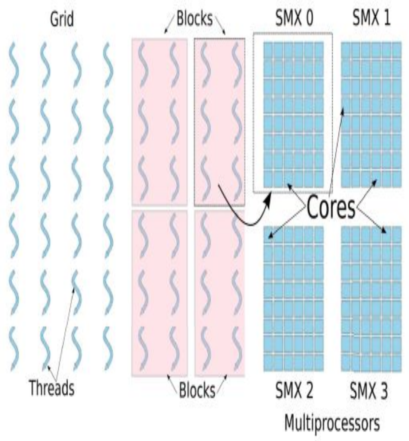


Fig. 1. The CUDA thread block structure

II. Related work

.

Basically three GTS algorithms were designed using multiple processors. Algorithmic descriptions are given below:

A. *Principal Variation Algorithm*

Before beginning of algorithm, number possible choices are generated as a game tree form. In this algorithm initial branch is marked by principal nodes. Nodes are searched serially by first processor P0 and it marks the node as a visited one. Other processors need to wait until searching finished by previous one. After finishing it others can proceed. As this algorithm uses parallel way but contains following drawbacks:
There is no synchronization among processors. Processor who finished its task needs to wait for other processors finish to start up another around of calculation. It increases time of calculation because of serial approach [8][10].

B. *Enhanced Principal Variation Algorithm*

Hence improvement was done in this algorithm to overcome the drawbacks of PVS algorithm.
The idea is to assigning subtrees to idle one. Hence the performance and efficiency is improved. The enhanced method will bring extra communication overhead[8][10].

C. *Dynamic Tree Splitting*

In this algorithm Split points list were maintained. In that list of all uncalculated nodes maintained. At beginning that list is cleared. All processors visit the nodes and update the list. If there are no points left in SP-LIST, the idle processor will broadcast a HELP message to all processors. Busy processors that receive the message will split and copy the state of the subtree to SP-LIST. Then idle processor again goes through list and searches the tree. The DTS algorithm completes after all processors stop in an idle state and no split-points left. Compared with PVS and EPVS algorithms, the advantage of DTS algorithm is its usability and scalability[8][10].

All these algorithms are developed with the help of processors. Even parallelism is achieved but processor capability is lower to evaluate.

III. Proposed Algorithm

GPU consist of capacity of processing thousands of nodes simultaneously. GPU is subtle to the instruction divergence because of warp. In GPU-based GTS algorithm, the kernel of GPU is a user defined C function used to calculate the node and also involves lots of control flow instructions because of game rules.

PVS, EPVS, DTS algorithms are implemented using tree based approach consist of drawbacks like searching is done using DFS manner, it requires more time for computation. Game is splitted into number of possible choices that are considered as possible moves which is next best move for player. Using a tree-based approach many choices of games are computed serially by processor in DFS manner. Due to the SIMD feature of GPU, the tree based approach cannot be easily adopted in GPU. Different from tree-based approach, node-based approach is advantageous in which CPU generates number of possible trees contains the nodes as well as leaf. Number of possible moves in form of tree is created on CPU. CPU is responsible for execution control as well as it is responsible for maintaining the game tree structure. Representation of the game is shown in figure 3. Shows Connect4 is a two-player connection game in which the players first choose a “X” pieces and then take turns dropping “X” discs from the top into a seven column, eight rows vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect four of one's own discs of the same piece next to each other vertically, horizontally, or diagonally before your opponent. Connect Four is a strongly solved game.

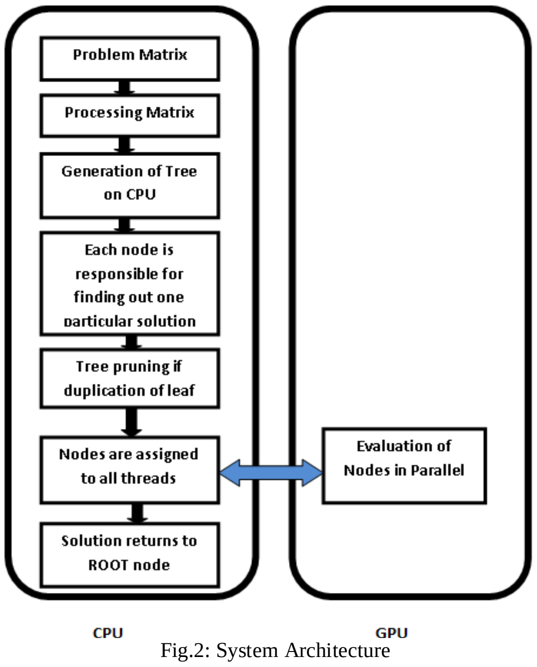
The tree starts at the initial position of the game as its root. Moves of the players are represented by “X” and “O”. The player represented by the “X”, plays game first and each node in this tree is a possible situation of the game. Possible moves by another player for “O” are generated on the CPU which is tree structure. Evaluation of all nodes, leafs are done by number of threads. Calculation of many tree nodes is done in the same depth in the current game tree, which is the BFS search. Further each cycle in the search process will take in the deepest nodes of the current game tree, which is the DFS search. That means on DFS approach CPU works to calculate nodes, since CPU will execute faster than GPU in this situation and on BFS approach GPU used for calculating the branch and the leaf nodes in parallel. By this hybrid manner, our algorithm is fully utilizes both architectures.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015


Problem Matrix
Processing Matrix
Generation of Tree on CPU
Each node is responsible for finding out one particular solution
Tree pruning if duplication of leaf
Nodes are assigned to all threads
Solution returns to ROOT node
Evaluation of Nodes in Parallel
CPU
GPU
Fig.2: System Architecture

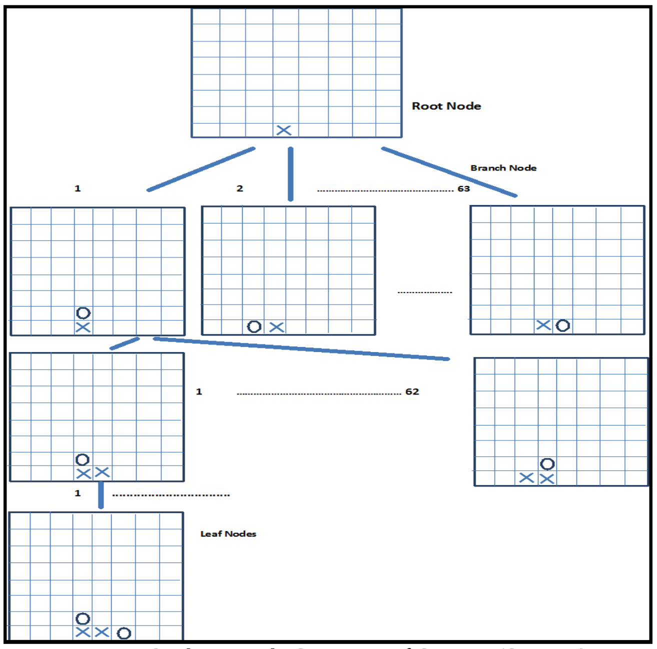


Fig. 3: The Two-Ply Game Tree of Connect4/Connect6

The pruning procedure is another key component of our node-based algorithm which is used to prune redundant nodes. Finally solution is returned to the root node. Such a hybrid approach takes an advantage of computation on CPU in DFS manner and evaluation of nodes by GPU in BFS manner. This approach can be applied for HEX, CHESS games. The game like Connect4/Connect6 is compared with serial implementation and single GPU implementation. Approach is made for improving the GTS algorithm and extends it to the large-scale clusters with GPU environments. In this work a well-established HPC framework, Message Passing Interface is used to minimize amount of latency and handle the communication between the devices. The work is responsible for dividing node generation task on multiple CPU using MPI. Each CPU is uses GPU for node evaluation task.

A. System Architecture

The system architecture of parallel evaluation with combination of CPU-GPU is shown in the fig 2.
The system works with combination of both, where CPU does calculation and GPU evaluates nodes in parallel way. To solve three mentioned problems in SECTION 2 following system architecture is used.

B. Methodology

The most common goal of game tree search is finding of players move that maximizes his chance of winning. To achieve this goal it is necessary to create game tree as much as possible generated by legal moves of both players. Consequently the game tree is calculated and it is chosen the move that leads to highest price. Node-based GTS algorithm does functionality as discussed for Connect4/Connect6.
From figure 2, CPU and GPU are shown differently. Problem data set is nothing but matrix which is provided as an input. CPU performs operation like maintaining tree structure, processing of data, generation of all nodes, tree pruning, checking of leaf nodes, solution returned to the root node. Where number of tree nodes are evaluated by threads in GPU section.

C. Tree Generation on CPU

The use of DFS and BFS approach will provide not only calculate nodes on GPU, but also avoid the exponential growth of the space complexity through the parallel search process and prune nodes after calculation on

GPU. When matrix is provided as an input to the system, CPU generates number of possible trees contains the nodes as well as leaf. That means generation of possible move of the player is created on the CPU. CPU is responsible for maintaining the game tree structure.

D. Parallel Evaluation on GPU

Calculation of many tree nodes is done in the same depth in the current game tree, which is the breadth-first search. In addition, each cycle in the search process will take in the deepest nodes of the current game tree, which is the depth-first search. That means on DFS approach CPU works to calculate nodes, since CPU will execute faster than GPU in this situation and on BFS approach GPU used for calculating the branch and the leaf nodes in parallel.

E. Algorithm

```
Node-based GTS Algorithm
Input: Initial position for GTS P0
Output: Best Move $M_{Best}$
Begin:
Step1: Set P0 as the root of the GTS
Step2: if Tree T Null then
Step3: return
Step4: End if
Step5: else
Step6: T formation on CPU
Step7: Node formation and structure maintain on CPU
Step8: Depth First Search to process tree and prunes redundant nodes
Step9: Leaves, branch nodes are assigned to GPU to calculate concurrently
Step10: Calculates branches and leafs nodes in parallel as Breadth first search
Step11: Updates parent node P0
Step12: Returns result $M_{Best}$
End
```

Algorithmic strategy mentions tree generation and parallel evaluation. The pruning procedure is another key component of Node-based algorithm. That is after calculating scores for all leaves, algorithm update the parent node and check its brother nodes to cut off some nodes as per the pruning procedure. This algorithm is used for achieving good pruning efficiency.

IV. Experiment and Results

Connect 4 and connect 6 game is performed on a machine contains CPU configuration Intel ® Core i3-4150 CPU @ 3.50GHz x 4 with 7.7 GiB memory. Operating system is UBUTNTU 14.04 LTS of 64 bit. GPU configuration consist of Quadro 6000 Graphics Processor, 448 CUDA cores, Total memory 6144 MB.

We have used CUDA toolkit 4.0 version to implement algorithms. As global memory transaction requires maximum time for computation hence shared memory is used for respected game. Connect4/Connect 6 is developed in such way that it is a scalable as per hardware of system. Like we can extend game’s length width wise and height wise also.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

TABLE I: SHOWS RESULTS OF CONNECT4 GAME

Board Width	Board Height	Tree max depth	Average CPU time (seconds)	Average GPU time (seconds)
8	8	8	20.70	0.266
8	8	10	40.78	0.261
8	8	12	1356.60	0.230
8	8	14	3160.50	0.229
8	8	16	4260.42	0.254

Table I describes board width, height, tree depth parameters are tested for Connect4 game. Board width, height, tree depth is 8-8-8 respectively but results of CPU-GPU are taken in terms of increased tree depth 8-10-12-14-16 as given in column three. Fourth and fifth column gives average computational time which is required to take next move of connect4 game. From column three it is observed that computation time of CPU exceeds as tree depth grows.

TABLE II: SHOWS RESULTS OF CONNECT4 GAME

Board Width	Board Height	Tree max depth	Average CPU time (seconds)	Average GPU time (seconds)
10	10	8	101.94	2.705
10	10	10	317.30	2.703
10	10	12	4352.76	2.677
10	10	14	5008.94	2.718
10	10	16	7179.56	2.706

Table II describes board width, height, tree depth parameters are tested for Connect4 game. Board width, height, tree depth is 10-10-8 respectively but results of CPU-GPU are taken in terms of increased tree depth 8-10-12-14-16 as given in column three. Fourth and fifth column gives average computational time which is required to take next move of connect4 game. From column three it is observed that computation time of CPU exceeds as tree depth grows.

TABLE III: SHOWS RESULTS OF CONNECT6 GAME

Board Width	Board Height	Tree max depth	Average CPU time (seconds)	Average GPU time (seconds)
8	8	8	20.33	0.225
8	8	10	409.45	0.240
8	8	12	2593.02	0.278
8	8	14	2590.40	0.181
8	8	16	2700.52	0.333

Table III describes results of Connect6 game in which board width, height, tree depth parameters are tested. Board width, height, tree depth is 8-8-8 respectively but results of CPU-GPU are taken in terms of increased tree depth 8-10-12-14-16 as given in column three. Fourth and fifth column gives average computational time which is required to take

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

next move of connect4 game. From column three it is observed that computation time of CPU exceeds as tree depth grows.

TABLE IV: SHOWS RESULTS OF CONNECT6 GAME

Board Width	Board Height	Tree max depth	Average CPU time (seconds)	Average GPU time (seconds)
10	10	8	72.83	2.617
10	10	10	696.27	2.756
10	10	12	2720.80	2.652
10	10	14	2789.89	2.942
10	10	16	2856.78	2.950

Table IV describes results of Connect6 game in which board width, height, tree depth parameters are tested. Board width, height, tree depth is 10-10-8 respectively but results of CPU-GPU are taken in terms of increased tree depth 8-10-12-14-16 as given in column three. Fourth and fifth column gives average computational time which is required to take next move of connect4 game. From column three it is observed that computation time of CPU exceeds as tree depth grows.

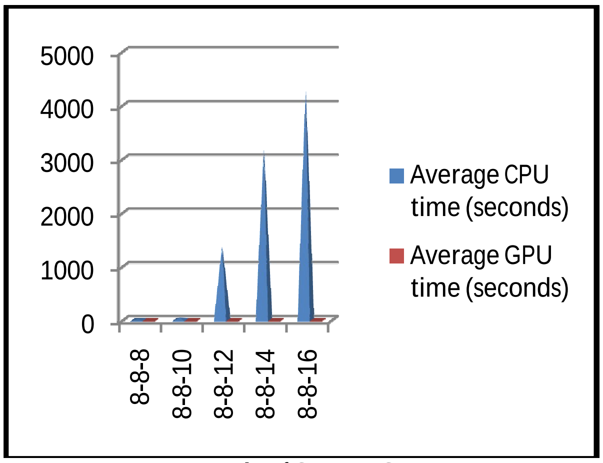


Fig 4: Results of Connect4 Game

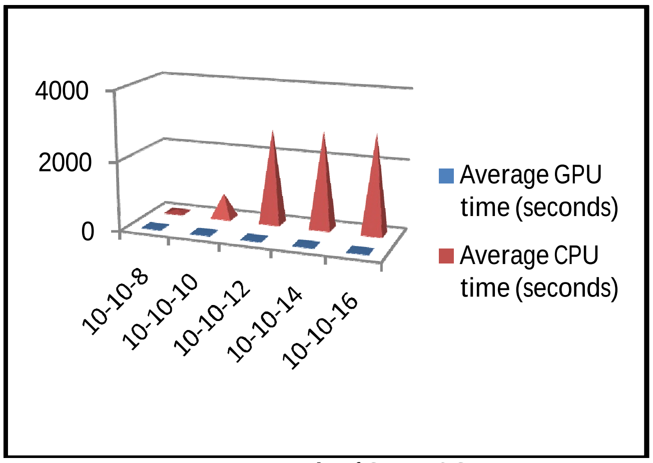


Fig 5: Results of Connect6 Game

Fig 4 and Fig 5 shows result of Table I/IV which consist of Board Width-Height-Depth at X-Axis. Graphical representation shows drastic computational time effect between CPU and GPU. Results are improved with the use of GPU. As we can see Speed up factor is achieved 7341.389X and 1255.427X for connect 4 and 6709.016X and 644.6214X for connect6 game.

V. Conclusion and Future Work

System focuses on a Parallelization of Node based Game Tree Search Algorithm on GPU. Parallel GTS algorithm presented node based approach for obtaining speedy optimal solution of real time computer games on GPU using node-based parallel computing for GTS and combination of DFS-BFS on CPU-GPU respectively.

Connect4/Connect6 game is tested on Quadro 6000 Graphics Processor having 448 CUDA cores. From graphical representation it is seen that node based GTS algorithm gains the speed over traditional approach. Future scope can be extended for large scale clusters with GPU environments and applicable for other games.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

ACKNOWLEDGEMENT

I am glad to express our sentiments of gratitude to all who rendered their valuable guidance to me. I would like to express my appreciation and thanks to Prof. Dr. P. C. Kulkarni, Principal, G. E. S's. R. H. Sapat College of Engg., Nashik. I am also thankful to Prof. N. V. Alone, Head of Department, Computer Engg., G. E. S R. H. Sapat College of Engg., Nashik. I thank the anonymous reviewers for their comments.

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,31 July,2014
2. X. Huo, V. T. Ravi, W. Ma, and G. Agrawal, "*Approaches for parallelizing reductions on modern GPU*", International Conference on High Performance Computing (HiPC), 2010, pp. 1-10.
3. W. Ma and G. Agrawal, "*An integer programming framework for optimizing shared memory use on GPU*", International Conference on High Performance Computing (HiPC), 2010, pp. 1-10.
4. 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), 2010, pp. 1-10.
5.A. Bleiweiss, "*GPU Accelerated Pathfinding*", The 23rd ACM Symposium on Graphics Hardware, 2008, pp. 65-74.
6.H.van den Herik, S.-C. Hsu, T. Hsu, H. Donkers, I.C. Wu and D.Y. Huang, "*A New Family of k-in-a-Row Games*", Eds. Springer Berlin Heidelberg, 4250:180-194, 2006
7.C. E. Shannon, "*Programming a Computer for Playing Chess*", Philosophical Magazine Series 7, 41(314):256-275, 1950.
8.M. G. Brockington, "*Taxonomy of Parallel Game-Tree Search Algorithms*", 1996.
9.R. M. Karp and Y. Zhang, "*On parallel evaluation of game trees*", The first annual ACM symposium on Parallel algorithms and architectures, 1989, pp. 409-420.
10.V. Manohararajah, "*Parallel alpha-beta search on shared memory multiprocessors*", 2001.
11.P. Borovska and M. Lazarova, "*Efficiency of parallel minimax algorithm for GTS*", The international conference on Computer systems, 2007