ISSN:0975-9646

A Review on Parallelization of Node based Game Tree Search Algorithms on GPU

Ms. Rutuja U. Gosavi[1], Prof. Payal S. Kulkarni[2]

[1]*Student,Computer Engg. Department, GES's R. H. Sapat COE Nashik Pune University, India*
[2]*Professor,Computer Engg. Department, GES's R. H. Sapat COE Nashik Pune University, India*

Abstract: **Game tree search is a classical problem in the field of game theory and artificial intelligence. Focus of the system is on how to leverage massive parallelism capabilities of GPUs to accelerate the speed of game tree algorithms and propose a concise and general parallel game tree algorithm on GPUs. Comparison can be done for classical CPU-based game tree algorithms with pruning and without pruning. The purpose is to identify possibilities of tasks parallelization when searching and assess game search trees that would perform better on SIMD processors of graphics cards. For game tree search CPU did not produce improvement but GPU surpasses a single CPU if high level of parallelism is achieved; because its searching is in BFS manner whereas CPU precedes BFS approach. The focus is on combination of DFS and BFS. Thus selection will be the depth-first search on CPU due to memory limit and use breadth-first search on GPU. Approach can be made for multi-computer environments, which look into better pruning algorithms for the GPU-based GTS algorithms. GTS algorithm used for games having much higher complexity to find best real time response, such as CHESS, SUDOKU, and HEX to weigh up the effectiveness of parallelization on the GPU. It intends to look into hybrid CPU-GPU solutions with heuristic GPU activation for deeper searches.**

Keywords: **Alpha-Beta Pruning, CHESS, HEX, GPU, GTS, Parallel Computing, SIMD, SUDOKU.**

I. INTRODUCTION

Graphical processing unit accelerators have emerged as a powerful support for massively parallel computing. Parallel work is implemented on GPU with the CUDA development toolkit.

CUDA stands for Compute Unified Device Architecture it is a parallel computing platform and programming model created by NVIDIA and implemented by the GPUs that they produce. CUDA gives developers direct access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs.

Lots of applications have get benefits from the massive parallelism capability of GPU [2] [3] [4]. GPU used to solve some specific AI problems successfully [5].It is a way to solve compute-intensive AI problems due to its SIMD architecture specialized for parallel computing.

Single Instruction Multiple Data architecture describes computers with multiple processing elements that perform the same operation on multiple data points simultaneously. Thus, such machines exploit data level parallelism; there are simultaneous computations, but only a single instruction at a given moment. Game tree search is important in artificial intelligence because one way to pick the best move in a real time games.

A. WHY GTS ON GPU?

In game theory, a game tree search is a directed graph whose nodes are positions in a game and whose edges are moves. The complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position; the complete tree is the same tree as that obtained from the extensive-form game representation.

Game Tree Search is a combinatorial game theory problem, it is difficult to find an optimal solution for many computer games like Connect6 [6] and Chess [7] due to their exponential time complexity. It focuses on two topics: one is to find better GTS algorithms to obtain near-optimal solutions; another one is to use advanced computing technologies to accelerate the speed of GTS algorithms for applications asking for real-time response, e.g., online computer game, decision tree, expert system and etc.

B. WHY PARALLELISM?

To satisfy such a requirement like to reduce computational time for games, parallel computing technologies introduced to improve the performance of GTS algorithms. CPU-based parallelism methods have been studied for many years [8] [9].

The parallel computation on the GPU is performed as a set of concurrently executing thread blocks, which are organized into a 1D or 2D grid. They can be 1D, 2D or 3D with each thread designated by a unique combination of indices. The hardware schedules the execution of blocks on the multiprocessors in units of 32 threads called warps.
Computing on graphics processing units is the utilization of a GPU, which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the CPU.

The use of multiple graphics cards in one computer, or large numbers of graphics chips, further parallelizes the already parallel nature of graphics processing.

GPU works on a SIMD approach and it is best method to find the best move in a computer game. Graphics processing Unit achieve in some specific tasks higher performance than conventional processors. These basic tasks are decomposed into higher number of smaller subtasks that can be processed con-currently by graphics card. It improves the computational time to find answer which is the feasible optimal solution.

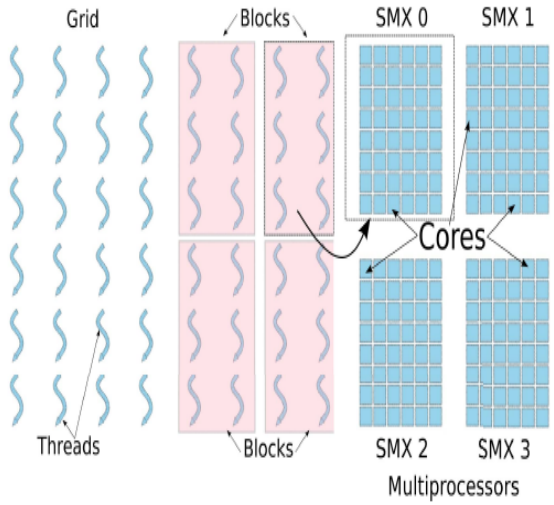

Fig 1: Structure of GPU

From structure of GPU, simply assigning instruction to the multiple threads it works on finding next best move of player GTS.

II. Background And Motivation

The common goal of GTS is finding of player's move that maximizes his/her chances of winning. Basically in combinatorial game theory problem it is hard to find near optimal solution due to its exponential time complexity. It is necessary thing to find near optimal solution and to accelerate the speed of GTS for real time responses from application. E.g. Real time games on computer.

In fact, lots of applications get benefits from massive parallelism capability of GPUs. The parallelism can be implemented at different levels of computer system using multiprocessor level, multiple computer level. GPUs have independently developed to perform data-parallel computation using multiple cores, the main fact to take advantage of this technology by performing some computation on GPUs that has traditionally been done on CPUs. A recent trend in computer architecture is the move from traditional, single core processors to multi-core processors and further to many-core or massively multi-core processors. The result of this trend is that computational problems which can take advantage of multiple threads with significant speed up. To achieve fast real-time response parallelism is necessary approach. GPU is a feasible way to improve the performance of GTS.

Thus it is essential to investigate if GTS can get benefit from GPU and compare with GPU-based approach with CPU-based approaches.

III. Related Work

According to previous studies some challenges arise like:

1. Complexity of algorithm design on SIMD architecture.
2. Low performance of divergence on GPU for rule-based computer games.
3. Low pruning efficiency of parallel GTS algorithms.

Solution of these problems depends on exploiting parallelism potential of GPU [1].

A representative methods used in computer game includes minimax algorithm, negamax algorithm with alpha-beta pruning etc. [11]. These algorithms are very widely used in searching game trees.

Principal Variation Splitting is a parallel GTS algorithm on CPU.

In PVS, the first nodes marked by 1 in the game tree should be searched serially first by a processor p0 before parallel search of other nodes begins. Let's say First node is principal node [10]. When Searching of it is finished by a processor, as per algorithm all processor starts effort to search unvisited nodes. The processor that finished the search task will inform its updated results to other processors and take the next unassigned branch to calculate. When there are no branches left and all processors finished their tasks, the PVS algorithm halts and return the best move to the player. The limitation of the PVS algorithm is that a processor who has finished its task needs to wait for other processors' finish to start up another around of calculation. It wastes computing time of till next processor gives an calculation to it.

Therefore Enhanced PVS is introduced, the idea is assigning subtrees to idle processors from other busy processors. That result in significant performance improvement for an unbalance tree and experiments prove its efficiency.

Dynamic Tree Splitting is method for parallel game tree search, which uses a peer-to-peer model on multi-processor systems. It based on a shared global list of active split-points SP-LIST, by which all processors find uncalculated nodes to process. At the beginning one processor takes the root node of the game tree and other nodes remain in the idle state. An idle processor looks up SP-LIST to find a branched node in the game tree to traverse the subtree.

If there is no split points left in SP-LIST, the idle processor broadcasts a HELP message to all processors. A busy processor receives that message and copies the state of the subtree to SP-LIST. The idle processor looks into SP-LIST again and obtains a split point and search the subtree from that point. The DTS algorithm completes after all processors stop in an idle state and no branched nodes left. As compared with PVS and EPVS, the advantage of DTS algorithm is its usability and scalability achievement. It is found that DTS algorithm can search nodes concurrently and efficiently.

A. CHALLENGES FOR GTS ALGORITHM DESIGN ON GPU

The work is to use GPU to process thousands of game tree nodes simultaneously. GPU is very sensitive to the instruction divergence caused by the control flow in a warp [1].

For GPU-based GTS algorithms, the kernel of GPU is the function to calculate the node, which includes lots of control flow instructions because of rule based games. Due to the SIMD feature of GPU, the tree-based approach on CPUs cannot be easily followed in GPU. The new approach fully utilizes the potential of GPU and development of efficient GPU-based GTS algorithms.

There is need to solve following challenges:

1. Complex control logic of parallel search on GPU
   Design of GPU-based parallel GTS algorithm is much more complex than CPU-based algorithm design. On CPUs searching is done according to the alpha-beta search. However, on GPUs, necessity is to unfold the recursive search and maintain the game tree in the memory. This significantly increases the system complexity, there is need to design algorithms carefully.
2. Low performance of divergent execution on GPU for rule-based computer games. GPU have low performance in divergent execution. Most of the GTS related calculation is rule-based; there are many divergences in node calculation which leads to a low speed up and affect the overall performance of GTS algorithms.
3. Low pruning efficiency for parallel tree search. As GTS algorithms can prevent duplicated calculations on game tree nodes. The pruning depends on the accumulated knowledge of all situations in the previous nodes, which implies that calculations of different nodes may depend on each other and it is hard to process them in parallel.

The problem still exists and there is a need to keep balance between pruning efficiency and parallelism to obtain best performance for GPU-based algorithm

To solve the above challenges, a node based parallel method to utilize the potential of GPU can be adopted.

B. NODE-BASED APPROACH

To solve the challenging problems in Section III following ideas to solve GTS problems on GPU:

1. To adopt node-based parallel computing for game tree search.
   The tree-based approach is not suite for GPU architecture. The approach is assigning a set of nodes from one or multiple subtrees to processors, while the tree-based approach is while the tree-based approach is assigned to processors. The benefit of method is not only taking advantages of the high concurrency of GPU, but also avoiding the complexity of tree splitting.
2. The combination of depth-first and breadth-first search
   Commonly, there are two methods to search the tree, the depth-first and the breadth-first search
   In this approach calculating a many tree nodes in the same depth in the current game tree, which is the breadth-first search. In addition, each cycle in the search process takes in the deepest nodes of the current game tree, which is the depth-first search. In GPU based GTS algorithm, selection is the depth-first search on CPU due to memory limit and use breadth-first search on GPU by scheduling the nodes with the same depth to GPU in each cycle.
3. Hybrid programming on both CPUs and GPU.
   There are many control logic in GTS algorithms. CPU architecture is good at complex execution of programs containing control statements such as switch, condition and loops, GPU architecture good in intensive computation on data with the same type. Therefore, third method is to use both CPU and GPU architecture in GTS algorithm [1].

C. APPROACHES

Using above three ideas calculation of leaf and branches performed on GPU concurrently.

In empirical study two configurations were used games. The first one adopts the GPU-based GTS algorithm without the tree pruning. With this configuration, verification is on the stand-alone capability of GPU to process massive nodes calculations in parallel [1].

The second configuration focuses on classical pruning technologies as well as the parallel search algorithm on GPU to evaluate the practical performance [1].

From the study it was observed that the speedup drop happens when both of the branch and leaf nodes calculation is done using a node-based approach.

IV. CONCLUSION

This review paper focuses on a Parallelization of Node based Game Tree Search Algorithms on GPU. Parallel GTS algorithm presented three approaches for obtaining speedy optimal solution of real time computer games on GPU.

By use of both CPU and GPU architecture, the approach can take advantage of the capability of GPU to compute massive nodes in parallel and GPUs flexibility to accelerate tree search and pruning. This approach can be tested by implementing it for SUDOKU, HEX and CHESS games. Results of implementation can be compared with serial implementation of game tree search.

ACKNOWLEDGMENT

We are glad to express our sentiments of gratitude to all who rendered their valuable guidance to us. We would like to express our appreciation and thanks to Prof. Dr. P. C. Kulkarni, Principal, G. E. S. R. H. Sapat College of Engg., Nashik. We are also thankful to Prof. N. V. Alone, Head of Department, Computer Engg., G. E. S. R. H. Sapat College of Engg., Nashik. We 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", 2010 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] I.C. Wu and D.Y. Huang, "A New Family of k-in-a-Row Games", Advances in Computer Games, H. van den Herik, S.-C. Hsu, T. Hsu, and H. Donkers, 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, A 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 2007 international conference on Computer systems and technologies, 2007, pp. 14:1-14:6.