2022 International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC) | 978-1-6654-6287-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/AIIPCC57291.2022.00098

GPU Acceleration of Monte Carlo Tree Search Algorithm for Amazon chess and Its Evaluation Function

Yikai Sun
Computer School of Beijing Information Science and Technology University, BISTU
Beijing, China
sunyikai2021@outlook.com

Dong Yuan
Computer School of Beijing Information Science and Technology University, BISTU
Beijing, China
YUANDONG20211@outlook.com

Ming Gao
Computer School of Beijing Information Science and Technology University, BISTU
Beijing, China
gaomingshsf@hotmail.com

Penghui Zhu
Computer School of Beijing Information Science and Technology University, BISTU
Beijing, China
941509604@qq.com

***Abstract*—The evaluation function is a very important part of the Monte Carlo Tree Search (MCTS) algorithm for Amazon chess. This paper introduces the principle of Monte Carlo Tree Search algorithm, and realizes GPU accelerated calculation of Amazon chess situation's evaluation function, based on CUDA computing platform. Using the CPU code and CUDA code, two experiments were carried out to evaluate the function calculation accuracy and the effect of kernel function call parameters on the GPU acceleration performance. The results show that the evaluation function is feasible and effective for GPU acceleration, and the speedup ratio of GPU to CPU reaches 20.0.**

Keywords-component; Amazon chess； artificial intelligence； Monte Carlo search tree； parallel operation

I. INTRODUCTION

Amazon chess is a new board game [1] invented by Argentinian Walter Zamkauskas in 1988 [2]. Although its rules are very simple, its playing methods are very difficult [3]. This is mainly because there are many situations [4] that can be selected in each step of the game [5], usually more than 500 [6], and the number of steps in each game can reach more than 80 [7]. For this reason, Amazon chess has become a good experimental sample [8] for testing the search strategy algorithm [9]. Since its invention [10], it has attracted the attention of many computer scientists [11] and computer students [12].

The main difficulty of Amazon chess lies in its complexity: This is mainly because there are a lot of situations that can be selected for each step of the game, usually more than 500, and the number of steps can be more than 80. Taking 500 situations in each step as an example, the total number of situations after four steps will reach more than 60 billion, which will undoubtedly be a great challenge to the traditional minimax search algorithm. In addition, some real tricks can only be seen after 10 steps or even 20 steps, and the first few steps look more like bad tricks. For the traditional minimax algorithm, it is extremely difficult to choose such a clever trick.

Monte Carlo (MC) method [13] is a calculation method to obtain the numerical results of targets through repeated random sampling. By combining tree search (TS) algorithm and appropriate situation evaluation function, Monte Carlo Tree Search (MCTS) algorithm has achieved very good results in the algorithms of board games with a large situation branching ratio[14-16] (such as the game of go and Amazon chess) and real-time strategy games (such as StarCraft [17] and Dota2) . For example, in 2016, the AlphaGo program [18] of DeepMind, a British artificial intelligence company, defeated the Korean world go champion Li Shiyu with a score of 4:1, and in 2018, the OpenAI Five program [19] of openai, an American artificial intelligence company, defeated the Dota2 world champion with an overwhelming advantage, all based on the Monte Carlo tree search algorithm.

The effect of Monte Carlo tree search algorithm depends largely on the number of situations it can search. The larger the number of situations it can search, the easier it is for the algorithm to find "tricks" that are difficult to find at ordinary times and can win over the opponent. In other words, the stronger the computing power, the better the effect of Monte Carlo algorithm. Due to the different computing architectures, computer Graphics Processing Unit (GPU) has higher computing efficiency than Central Processing Unit (CPU). Using GPU to accelerate the Monte Carlo tree search algorithm can significantly improve the actual application effect of the algorithm [20,21]. Therefore, this paper chooses to use GPU to accelerate the evaluation function, which is very important in Monte Carlo tree search algorithm for Amazon chess, so as to provide a basis for improving the practical application effect of the algorithm.

The Compute Unified Device Architecture (CUDA) is a parallel computing platform launched by NVIDIA, the manufacturer of graphics processing unit (GPU) in the United

The College of Computer Science (5112210832), the Innovation and Entrepreneurship Training Program for College Students of Beijing Information Science and Technology University.

978-1-6654-6287-7/22/$31.00 ©2022 IEEE
DOI 10.1109/AIIPCC57291.2022.00098

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

States, in 2007. It allows users to accelerate applications using NVIDIA's GPU. Because CUDA-C, the programming language of CUDA computing platform, is very similar to the standard C language, the CPU code usually needs only a small amount of modification, and the GPU acceleration program of CUDA-C can be implemented. For this reason, CUDA computing platform has been widely used in the field of high-performance computing. The early version of AlphaGo program and OpenAI Five program also use CUDA to accelerate the program.

In order to improve the operation performance of Monte Carlo tree search algorithm for Amazon chess, this paper uses CUDA-C language to implement the GPU acceleration program of the situation evaluation function, and analyzes the accuracy of the calculation results and the calculation performance under different parameters through experiments, which verifies that GPU can be well applied to the acceleration calculation of the evaluation function.

II. THEORIES

A. *Rules of Amazon chess*

Amazon chess usually moves on a 10 x 10 chess board by two people in turn, with four pieces on each side. The movement rules of the pieces are the same as those of the queen in chess, that is, they can move freely in eight directions centered on the pieces, but they cannot cross obstacles. The initial position of the chess piece is shown in Figure 1. Each step of Amazon chess consists of two steps. First, select one of your own pieces to move on the chessboard, and then select a position in eight directions centered on the moved piece to place an arrow. The arrow is similar to the chess piece in go, and the chess piece cannot be moved again immediately after it is moved (Fig. 2). When all four pieces of the opponent cannot move, the opponent is defeated. If both parties cannot move, it is a draw.

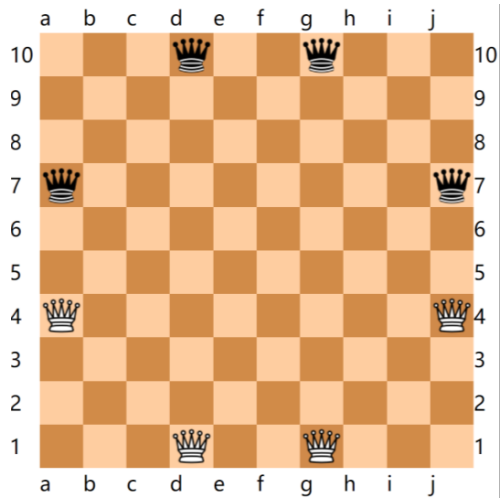

Figure 1. Amazon chess's board and the initial position of pieces

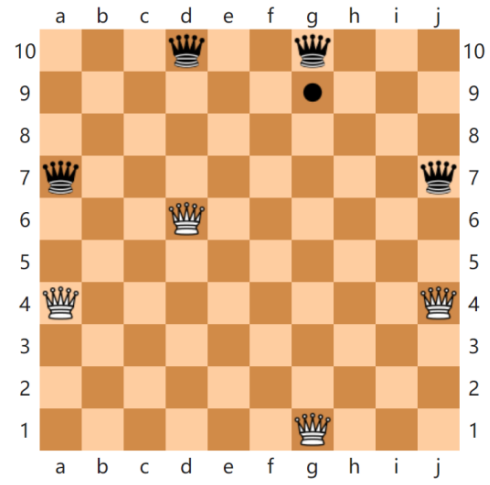

Figure 2. the board and pieces of Amazon chess. The white chess moves from position d1 to position d6, and an arrow is placed on position d9

B. *Monte Carlo Tree Search algorithm*

Since each step of Amazon chess has a large branching ratio, if we need to consider the situation after many steps of the current situation, the search space will increase sharply. Therefore, the traditional minimax search algorithm [12, 22, 23] can calculate a limited number of steps. In this way, it is difficult to find some "tricks" that need to be considered after many steps. In contrast, Monte Carlo tree search algorithm can search more steps.

The Monte Carlo tree search algorithm [24-26] includes four stages of "selection", "extension", "simulation" and "back propagation" (Fig. 3). In the selection stage, the algorithm starts from the root node of the currently constructed search tree and selects one of the child nodes from top to bottom; In the extension stage, a new child node is added to the selected node; In the simulation stage, calculate the new node; Finally, in the back propagation stage, the calculated value is updated to the path from the bottom to the root node.

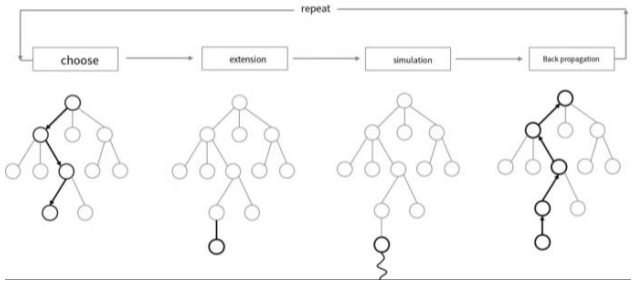

Figure 3. Basic steps of MCTS algorithm

However, the pure Monte Carlo tree search algorithm (selecting the child node with the highest score at the current node) is not ideal, partly because this greedy strategy may miss the potential moves after the non optimal child node. In order to solve this problem, this paper adopts the "upper confidence bounds applied to trees" algorithm [27] to guide the selection of nodes. As shown in formula (1), the UCT formula is composed of two terms, wherein the first term X_i represents the

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

success rate of the child node and represents the utilization of the constructed search tree information by the algorithm (exploitation), and the second item $\sqrt{\frac{lnN}{n_i}}$ represents the exploration of the algorithm on the non visited nodes (exploration). The constant C is used to adjust the proportion of the algorithm to the two strategies. Note that the value of the root formula is infinite because of the unreachable child nodes, which ensures that the unreachable child nodes under the current node will be accessed at least once.

$$UCT = X_i + c\sqrt{\frac{\ln N}{n_i}} \tag{1}$$

Where X_i is the winning rate of the child node

N is the number of times the parent node is accessed

n_i is the number of times the child node is accessed

c is a constant

C. Evaluation function

Similar to many board games, Amazon chess can be roughly divided into three stages: opening, middle and ending [28]. The strategy of each stage is also slightly different. Therefore, the situation evaluation function should also include strategies in different stages. One of the goals of Amazon chess is to have more open space than the opponent. When a space on the chessboard can only be reached by one party, the space can be considered as the territory of that party.

According to these ideas, J. Lieberum proposed the following evaluation function of Amazon chess [28]:

$D_1^j(a)$ is defined as the minimum number of steps for a player $j(j=1,2)$ to reach the open space a through the Queen's walk in chess, and $D_2^j(a)$ is the minimum number of steps for a player to reach the open space a through the King's walk in chess. The King's walk is to move within the eight squares with the Center removed within its 3×3 range. Fig. 4 and Fig. 5 respectively show and in a certain situation. Wherein, the upper left corner of the open space is j = 1 and the lower right corner is j = 2.

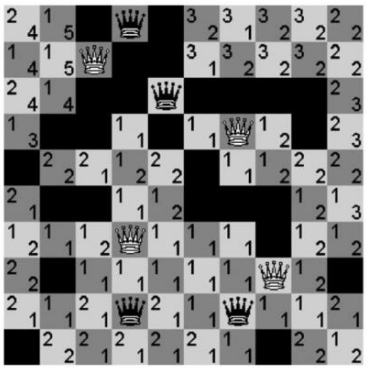

Figure 4. The minimum number of steps for the queen to reach the specified space

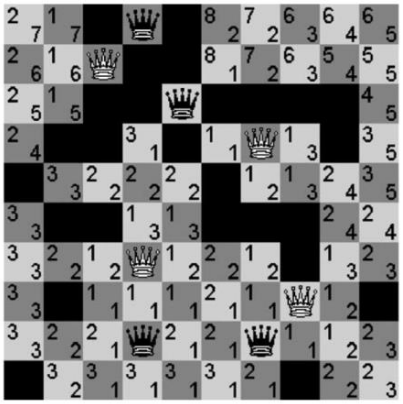

Figure 5. The minimum number of steps for the king to reach the specified space

When there is $D_1^1 < D_1^2(a)$ on the open space a, generally, it means that player 1 is easier to reach a than player 2. However, in the opening stage, due to the small number of arrows, the open spaces on the chess board may have very small values, but in fact they cannot reach these open spaces in one step. In comparison, $D_2^j(a)$ has better local characteristics and more stable numerical values.

Define the evaluation variable t_j representing the territory, as shown in formula (2). When it is easier for player 1 to reach open space a, t_j is 1, and vice versa is -1, where k is a constant, which is used to express the advantage of the first drop square when $D_i^j(a)$ is the same. Generally, $|k|\leq 1/5$ is taken, and k=0.1 is taken in this paper.

$$t_i = \sum_a \Delta(D_i^1(a), D_i^2(a)) \tag{2}$$

where

$$\Delta(n,m) = \begin{cases} 0 & if\ n = m = \infty \\ k & if\ n = m < \infty \\ 1 & if\ n < m \\ -1 & if\ n > m \end{cases}$$

However, $\Delta\left(D_i^1(a), D_i^2(a)\right)$ does not consider that the value of represents the difference of advantage between players, so the evaluation variables c_1 and c_2 are defined, as shown in formulas (3) and (4).

$$c_1 = 2\sum_a (2^{-D_1^1(a)} - 2^{-D_1^2(a)}) \tag{3}$$

$$c_2 = \sum_a \min(1, \max(-1, (D_2^2(a) - D_2^1(a))/6) \tag{4}$$

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

Then, as shown in formula (5), the weight variable w is defined. The value of the weight variable usually decreases with the progress of the chess game.

$$w = \sum_{a} 2^{-\left|D_1^1(a) - D_1^2(a)\right|} \tag{5}$$

Finally, as shown in formula (6), combine t_1, t_2, c_1, c_2 into one overall evaluation function:

$$t = f_1(w)t_1 + f_2(w)c_1 + f_3(w)c_2 + f_4(w)t_2 \tag{6}$$

Where, f_i is the division function of the weight between each variable, i.e. $\sum f_i = 1$ .The partition function used in this paper is shown in formula (7):

$$\begin{cases} f_1 = \dfrac{5}{w+5} \\ f_2 = f_3 = \dfrac{1}{2}\left(1 - \dfrac{5}{w+5} - \dfrac{w}{(w+20)}\right) \\ f_4 = \dfrac{w}{(w+20)} \end{cases} \tag{7}$$

The relationship between the partition function and the weight variable w in formula (6) is shown in Fig. 6. Since the value of w decreases with the progress of the chess game, the value of f_i will move from the right side of the x-axis to the left side on the graph. That is, f_1 gradually decreases and f_2 , f_3 , f_4 gradually increase.

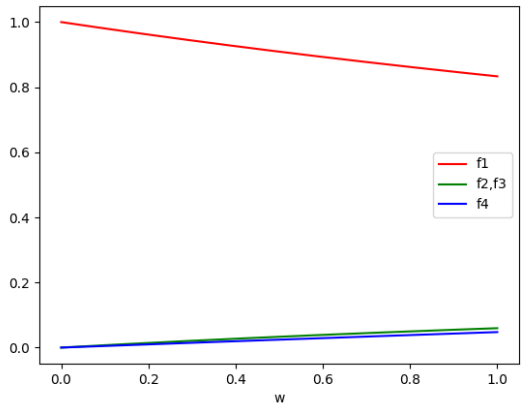

Figure. 6. Variation of partition function with weight variable w

D. GPU acceleration of evaluation function

Compared with the CPU generally having several to a dozen computing cores, the GPU with CUDA function usually has thousands of computing cores [29]. Therefore, for computing intensive tasks, the computing speed of the GPU greatly exceeds that of the CPU [30-34] (Fig. 7). However, although CUDA GPU can execute many instructions at the same time, the computing speed of a single CUDA core is much slower than that of a traditional CPU core. This is because, unlike the traditional CPU, CUDA adopts a system architecture called Single Instruction Multiple Thread (SIMT). A group of threads called a Warp executes the same piece of code. If code execution in a thread in a warp branches, some threads are suspended to execute branch code. Therefore, if there are many control branches in CUDA code, the efficiency will be greatly reduced [35].

A CUDA program includes two parts: host side (CPU) code and device side (GPU) code. The host code is mainly responsible for the input and output, and the scheduling of the whole program.

In CUDA's programming model (see Figure 8), the device side functions called on the host side are called kernel functions. The kernel function is executed in a thread block composed of many threads in the GPU. Different thread blocks do not interfere with each other. The threads in a thread block can exchange data through the shared memory with a high speed [36, 37].The set of all thread blocks of a kernel function is called a grid. The kernel functions share the same GPU global memory in the grid.

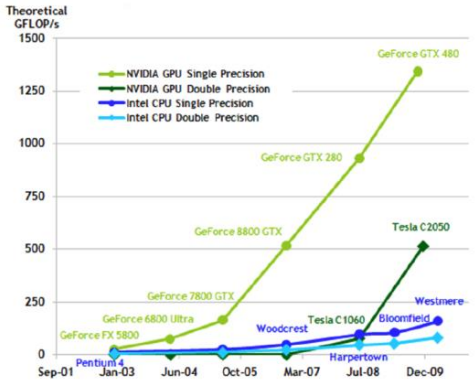

Figure 7. Comparison of computing power between CPU and GPU

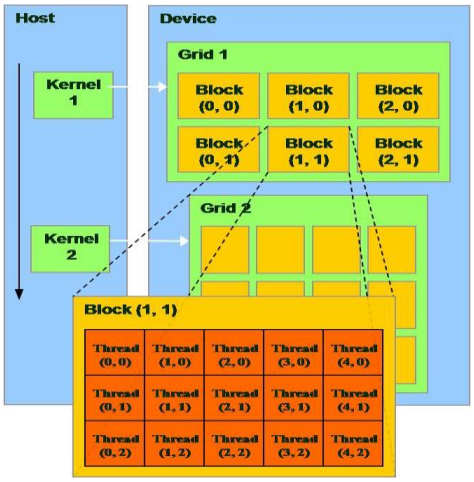

Figure 8. Basic system architecture of CUDA

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

III. EXPERIMENTS

The experiment is divided into two parts to evaluate the calculation accuracy of the function and the influence of the kernel function's call parameters on the acceleration performance of GPU. The code runs on a notebook computer. The CPU is Intel Core2 i7-8850h with six CPU cores, and the memory is 32GB. It runs on the windows 10 operating system. The GPU acceleration card is NVIDIA Quadro p3200, with 1792 CUDA cores and 14 multiprocessors running at 1.3GHz. The video memory is 6GB. The software programming development environment uses Microsoft Visual Studio 2015 and CUDA computing platform software package 10.1 (Figure 9).

In order to compare and analyze the GPU acceleration effect of the evaluation function, write a program to randomly generate the situation to be calculated and save it in the file moves.txt (Figure. 10). For the sake of simplicity, this paper starts from the beginning, in this paper, from the beginning, the chess pieces are randomly selected, and the legal moves are also randomly selected. The coordinates of the moves are saved in the file. Then write CPU and CUDA programs respectively, calculate the values of all situations in turn, and save them in the output file scores_ Cpu.txt and scores_ Gpu.txt for subsequent analysis (Figure 11).

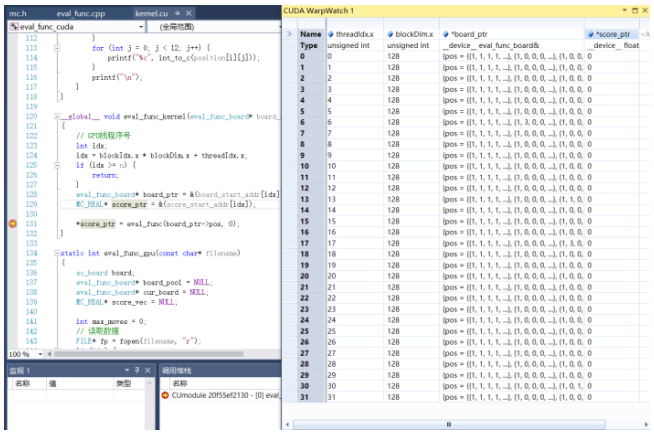

Figure 9. Debugging CUDA program with visual studio 2015

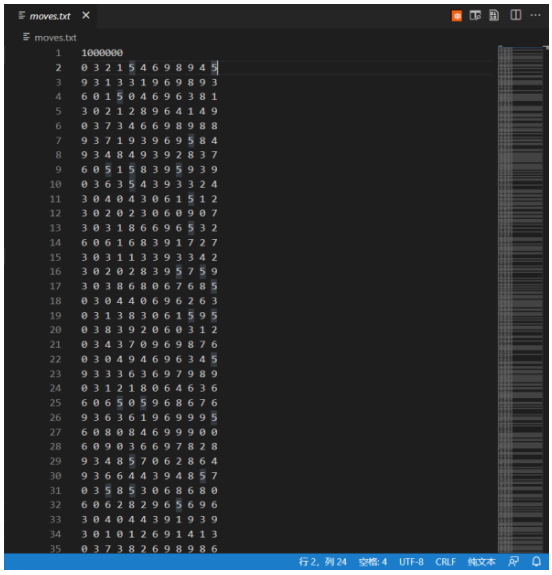

Figure 10. Input file for evaluation function calculation

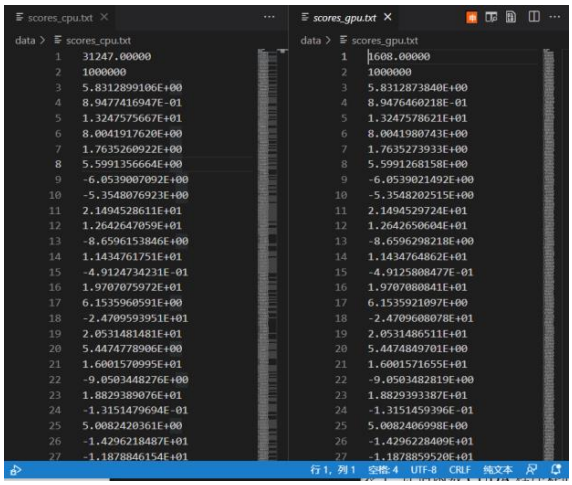

Figure 11. Error of CPU and GPU calculation results

IV. RESULTS AND DISCUSSION

A. *Accuracy of evaluation function calculation*

The general CPU uses double precision floating-point numbers for calculation, while the GPU generally uses single precision floating-point numbers, which is faster than double precision floating-point numbers. Taking NVIDIA Quadro P3200 as an example, its theoretical single precision floating-point operation speed is about 5.5 Tflops, that is, it can complete 5.5 trillion floating-point operations per second, while its double precision floating-point operation speed is only 0.17 Tflops, about 1 / 32 of the single precision floating-point operation speed. Therefore, GPU accelerated operations are usually performed using single precision floating-point numbers. In order to verify the results of GPU operation, we used the CPU version program and CUDA version program to calculate the evaluation function of the sub randomly generated

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

chess game, and compared the errors of the two. As shown in Fig. 12, the error of most results is within and , which shows that CUDA program can calculate accurately the evaluation function.

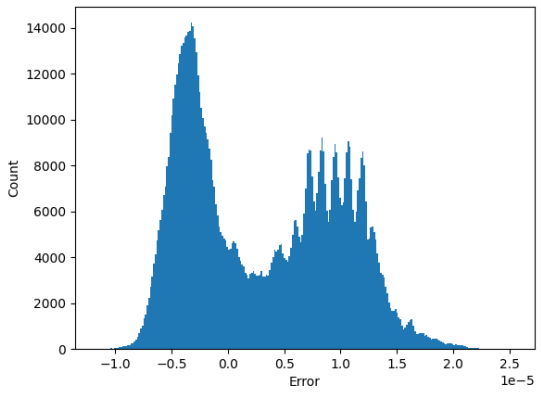

Figure 12. Error of CPU and GPU calculation results

B. The effect of kernel function's call parameters on performance

Due to the GPU architecture, the operation speed of a single GPU core is much slower than that of the CPU. In order to achieve the acceleration effect, many threads need to run in the GPU at the same time. Table 1 and table 2 show the time consumption of calculating evaluation function for times under the grid parameters of different CUDA call kernel functions, where is the number of thread blocks in the grid and is the number of threads included in a single thread block.

It can be seen from table 1 that as the number of simultaneous running threads increases, the calculation time consumption gradually decreases, reaching the minimum at, but the time consumption increases slightly since. This is mainly because the GPU accelerator used in the experiment has 1792 CUDA cores. When the number of threads is small, the GPU is not fully utilized, so the operation speed is slowed down. When the number of threads reaches a certain number, all the cores of the GPU are fully utilized, and the computing speed reaches the lowest. We note that the number of threads in the grid at the highest efficiency is 16384, which is much higher than the number of GPU physical cores. This indicates that more threads than the number of physical cores are required to call the kernel function to achieve the highest work efficiency of the GPU. Continuing to increase the number of threads can not continue to reduce the calculation time. It may be because excessive threads will increase the time spent by the system in allocating thread resources, thus increasing the total calculation time.

Table 1. CHANGE OF CUDA PROGRAM' S TIME OF EVALUATION FUNCTION WITH KERNEL FUNCTION CALL PARAMETERS (THE NUMBER OF THREADS IN THE GRID GRADUALLY INCREASES)

N_b	N_t	**Calculation time (s)**
16	16	14.489
32	32	5.056
64	64	2.009
128	128	1.603
256	256	1.835
512	512	1.804
1024	1024	1.756

TABLE 2. CHANGE OF CUDA PROGRAM'S TIME OF EVALUATION FUNCTION WITH KERNEL FUNCTION CALL PARAMETERS (THE NUMBER OF THREADS IN THE GRID IS FIXED AT 4096)

N_b	N_t	**Calculation time (s)**
4	1024	2.237
16	256	2.058
64	64	2.009
256	16	2.004
1024	4	4.238
4096	1	10.534

Table 2 shows the time taken to calculate the evaluation function for times under different and conditions, with the total number of threads unchanged. It can be seen from the table that the calculation time decreases gradually when increases gradually, and increases slightly when reaches 1024. This is mainly because, unlike the CPU core, a single CUDA core cannot perform calculations independently, but runs the same code in the form of 32 warp. Therefore, is small and the overall operation speed will be seriously slowed down. Note in the table that after reaches 16, the calculation time is no longer reduced. This may be mainly because the calculation of the evaluation function is complex and the code has many control branches, so many threads in the warp are in a waiting state during running.

As a comparison, the CPU code takes 31.247 s to calculate the evaluation function for times, which is far slower than the GPU accelerated calculation speed of 1.603 s, indicating that the GPU acceleration program we have written has achieved a good acceleration effect, and the GPU: CPU acceleration ratio has reached 20.0.

V. Conclusion and next step

This paper mainly introduces the principle of Monte Carlo tree search algorithm for Amazon chess, and implements the GPU program for Amazon chess situation evaluation function based on CUDA computing platform. Finally, the feasibility and effectiveness of GPU acceleration of evaluation function are verified by experiments.

From the experimental results of this paper, it can be seen that GPU acceleration has a very obvious performance advantage for the evaluation function calculation of Amazon chess. The next step is to apply GPU acceleration to the overall Monte Carlo tree search algorithm.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.

References

[1] Guo Q , Shuqin L I . Research on evaluation function computer game of Amazon[J]. Computer Engineering & Applications, 2012, 48(34): 50-54+87.

[2] Li R, Gao M. Amazon chess search algorithm design based on CNN model [J] Digital technology and applications, 2022, 40 (02): 164-166.

[3] Guo T, Qiu H, Tong B, et al. Optimization and Comparison of Multiple Game Algorithms in Amazon Chess[C]. 2019 Chinese Control And Decision Conference (CCDC), 2019: 6299-6304.

[4] Jianning Q, Hongkun Q, Yajie W, et al. Application of UCT technologies for computer games of Amazon[C]. 2016 Chinese Control and Decision Conference (CCDC), 2016: 6896-6899.

[5] Ju J, Qiu H, Wang F, et al. Research on Thread Optimization and Opening Library Based on Parallel PVS Algorithm in Amazon Chess[C]. 2021 33rd Chinese Control and Decision Conference (CCDC), 2021: 2207-2212.

[6] Li Z, Ning C, Cao J, et al. Amazon Chess Based on UCT-PVS Hybrid Algorithm[C]. 2021 33rd Chinese Control and Decision Conference (CCDC), 2021: 2179-2183.

[7] Meng D, Jianbo B, Yizhong Q, et al. Design of Amazon Chess Game System Based on Reinforcement Learning[C]. 2019 Chinese Control And Decision Conference (CCDC), 2019: 6337-6342.

[8] Tong B, Qiu H, Guo T, et al. Research and Application of Parallel Computing of PVS Algorithm Based on Amazon Human-Machine Game[C]. 2019 Chinese Control And Decision Conference (CCDC), 2019: 6293-6298.

[9] Chen X, Yang L. Research on evaluation function in Amazon chess [J] Computer knowledge and technology, 2019, 15 (08): 224-226

[10] Li Z, Li Y, Ran G, et al. Design of Amazon chess game system based on PVS search algorithm [J] Smart computers and applications, 2018, 8 (05): 86-88.

[11] Wang C, Ding M. Interface design and implementation of personalized Amazon chess [J] Intelligent computer and application, 2017, 7 (02): 78-80.

[12] Zhang L. Research on Amazon chess game system based on minimax search algorithm [D] Northeast University, 2010.

[13] Metropolis N, Ulam S. The Monte Carlo Method[J]. Journal of the American Statistical Association, 1949, 44(247): 335-341.

[14] Coulom R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search[C]. Computers and Games, 2007: 72-83.

[15] Gelly S, Kocsis L, Schoenauer M, et al. The grand challenge of computer Go: Monte Carlo tree search and extensions[J]. Commun. ACM, 2012, 55(3): 106‑113.

[16] Gelly S, Silver D. Monte-Carlo tree search and rapid action value estimation in computer Go[J]. Artificial Intelligence, 2011, 175(11): 1856-1875.

[17] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782): 350-354.

[18] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.

[19] Berner C, Brockman G, Chan B, et al. Dota 2 with Large Scale Deep Reinforcement Learning[J]. arXiv, 2019.

[20] Chen J X. The Evolution of Computing: AlphaGo[J]. Computing in Science & Engineering, 2016, 18(4): 4-7.

[21] Rocki K, Suda R. Large-Scale Parallel Monte Carlo Tree Search on GPU[C]. 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011: 2034-2037.

[22] Shannon C E. XXII. Programming a computer for playing chess[J]. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1950, 41(314): 256-275.

[23] 2004.Von N, Morgenston, Wang W, et al. Game theory and economic behavior [M] Beijing: life, reading and new knowledge Sanlian bookstore, 2004.

[24] Browne C B, Powley E, Whitehouse D, et al. A Survey of Monte Carlo Tree Search Methods[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(1): 1-43.

[25] Chaslot G, Bakkes S, Szita I, et al. Monte-Carlo Tree Search: A New Framework for Game AI[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2021, 4(1): 216-217.

[26] Świechowski M, Godlewski K, Sawicki B, et al. Monte Carlo Tree Search: a review of recent modifications and applications[J]. Artificial Intelligence Review, 2022.

[27] Kocsis L, Szepesvári C. Bandit Based Monte-Carlo Planning[C]. Machine Learning: ECML 2006, 2006: 282-293.

[28] Lieberum J. An evaluation function for the game of amazons[J]. Theoretical Computer Science, 2005, 349(2): 230-244.

[29] Garland M. Parallel computing with CUDA[C]. 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), 2010.

[30] Després P, Jia X. A review of GPU-based medical image reconstruction[J]. Physica Medica, 2017, 42: 76-92.

[31] Pratx G, Xing L. GPU computing in medical physics: A review[J]. Medical Physics, 2011, 38(5): 2685-2697.

[32] Bridges R A, Imam N, Mintz T M. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods[J]. ACM Comput. Surv., 2016, 49(3): Article 41.

[33] Jia X, Ziegenhein P, Jiang S B. GPU-based high-performance computing for radiation therapy[J]. Physics in Medicine and Biology, 2014, 59(4): R151-R182.

[34] Mittal S, Vetter J S. A Survey of Methods for Analyzing and Improving GPU Energy Efficiency[J]. ACM Comput. Surv., 2014, 47(2): Article 19.

[35] Han T D, Abdelrahman T S. Reducing branch divergence in GPU programs[C]. Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, 2011: Article 3.

[36] Kim Y, Shrivastava A. CuMAPz: A tool to analyze memory access patterns in CUDA[C]. 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 2011: 128-133.

[37] Rosen P. A Visual Approach to Investigating Shared and Global Memory Behavior of CUDA Kernels[J]. Computer Graphics Forum, 2013,32(3pt2):161-170.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on May 05,2023 at 02:38:49 UTC from IEEE Xplore. Restrictions apply.