Markov Decision Process Parallel Value Iteration Algorithm On GPU

Peng Chen[1,a], Lu Lu[2,b]

[1,2]Department of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, People's Republic of China

[a]c.peng01@mail.scut.edu.cn,[b]lul@scut.edu.cn

Keywords: GPU, parallel algorithm, OpenCL, Markov decision process (MDP)

Abstract. This paper defines an Out Of Play model based on Markov Decision Process. The best path for playing can be found and recommended by using this model, and a value iteration algorithm of Markov Decision Process is used to implement the model. In this paper, the implementation of this model with CPU is presented. And then, in order to improve the performance of the value iteration algorithm, a parallel value iteration algorithm on GPU is designed and showed. For the calculation of a large amount of data, the experimental results show that the parallel value iteration algorithm on GPU improves performance far more than that of the serial value iteration algorithm on CPU.

Introduction

Graphic Processing Unit (GPU) attracts more and more attention in general-purpose computing with the development of graphics hardware. But now, GPU is not only used in graphic, it is also considered as a powerful technique for obtaining inexpensive, high performance parallelism [1, 2]. General-Purpose GPU is a highly parallel, multithreaded, many-core processor with a very high computational power and memory bandwidth [3]. GPU architecture is designed for optimization of massively parallel computing, because of this architectural difference, a GPU is in general more advantageous for large-scale parallel data processing applications than general-purpose CPUs[4], and the high performance computing community leveraging a GPU can yield performance increases of several orders of magnitude[5,6]. High-performance computing with GPUs is called GPU computing [7]. So using GPU for parallel computing will become a new focus for the purpose of speeding up the calculation.

Markov Decision process is a stochastic dynamic system based on the theory of Markov process and decision-making process. In Markov decision process, the ultimate goal is to find an action for every state so that the performance of the system is the best. In this paper, in order to reach higher performance, a parallel implementation on GPU is given, and OpenCL is selected to program.

Markov Decision Process

Markov Model. Markov Decision process [8][9] can be defined as a four tuple <S,A,T,R>. In the tuple, S is a finite set of states, and A is a finite set of actions; T is probability distribution (means the probability of transition from state to state by taking action a); R is the reward function, and means the reward got when taking action from state to state . In MDP, there is a parameter which is a discount factor and is used to reduce the interference from the future actions.

Value Iteration Algorithm. In order to solve Markov model, there are many algorithms proposed. In this paper, the value iteration algorithm of MDP is selected for studying.

The main idea of value iteration algorithm [8] is iteration. In the algorithm, the target is to find the optimal policy via the optimal value iteration. First, we give an initial value function for every state, then we update the value function of every state to a next value function for every iteration until satisfying a condition. Fig 1 [8] is the pseudo of value iteration algorithm.


```
Initialize V (s) arbitrarily
Loop until policy good enough
    Loop for s∈S
        Loop for a∈A
            V (s, a) =R (s) +γ ∑T(s,a,s')V(s')
        End loop
            V (s) =max (V (s, a))
    End loop
End loop
```

Fig.1: The pseudo of value iteration algorithm

Out of Play Model. According to Markov Model and the value iteration algorithm, this paper gives a MDP model—Out Of Play model. This model can be described as follows: Go out for playing, but we can't decide where to go, or we don't know which transportation (how to go: bus or walk) can be chosen to reach the destination. In fact, this can be described as a MDP. In the scene, choosing where to go and how to go are random and uncertain, and according to the uncertain destination and mode of transportation, a MDP model can be created. In the model, the different destinations can be described as the finite set of states, and the different transportations can be described as the finite set of actions. Every state has an initial reward which represents the fun level. Every state can take different actions to get to another state, and this meets a probability distribution. In the model, a satisfied optimal path must be found. Fig 2 is the whole system model.

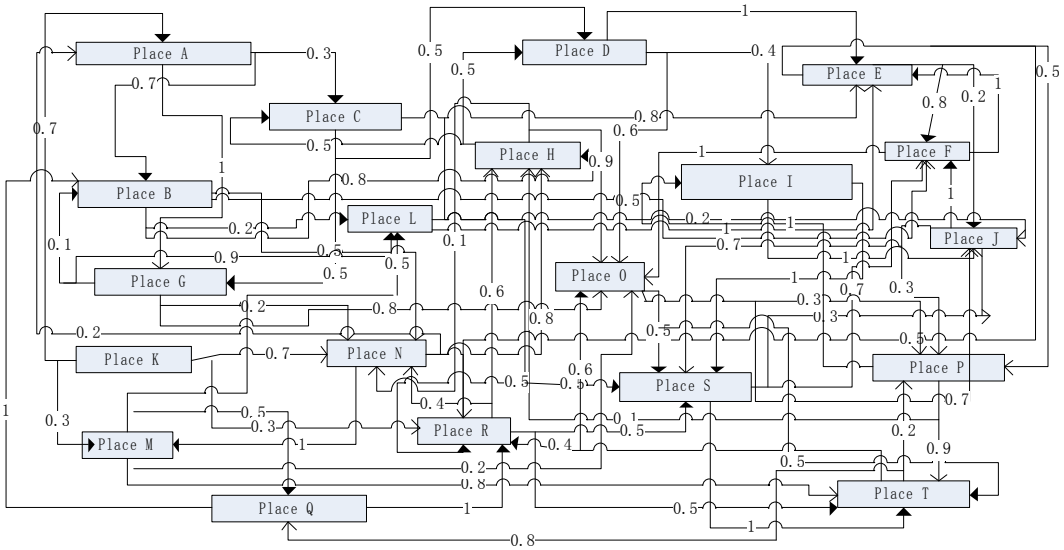

Fig.2: The model of Out of Play

In the Out Of Play model: γ = o.5;

The finite set of states: S={place A, place B, place C, place D, place E, place F, place G, place H, place I, place J, place K, place L, place M, place N, place O, place P, place Q, place R, place S, place T};

The finite set of actions: A={bus, walk};

In the order to describe this model clearly, s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s15, s16, s17, s18 and s19 are used to represent place A, place B, place C, place D, place E, place F, place G, place H, place I, place J, place K, place L, place M, place N, place O, place P, place Q, place R, place S, place T respectively. a1 and a2 are used to represent bus and walk respectively. The initial reward of every state is as follows:

R(s0)=70, R(s1) = 100, R(s2) = 95, R(s3) = 100, R(s4) = 120, R(s5) = 150, R(s6) = 80, R(s7) =180, R(s8) =105, R(s9)=60, R(s10) =75, R(s11) =110, R(s12) = 75, R(s13)=170, R(s14)=85, R(s15)=90,

R(s16) =80, R(s17)=95, R(s18)=85, R(s19)=75
The probability from one state to another state:
T(s0,a1,s6)=1,T(s0,a2,s1)=0.7, T(s0,a2,s2)=0.3; T(s1,a1,s5)=0.5, T(s1,a1,s13)=0.5, T(s1,a2,s7)=0.8, T(s1,a2,s11)=0.2;T(s2,a1,s4)=0.8,T(s2,a1,s9)=0.2,T(s2,a2,s3)=0.5,T(s2,a2,s6)=0.5;T(s3,a1,s8)=0.4, T(s3,a1,s14)=0.6,T(s3,a2,s4)=1;T(s4,a1,s15)=0.5,T(s4,a1,s17)=0.5,T(s4,a2,s5)=0.8,T(s4,a2,s9)=0.2; T(s5,a1,s14)=1,T(s5,a2,s4)=1;T(s6,a1,s13)=0.2,T(s6,a1,s14)=0.8,T(s6,a2,s1)=0.2,T(s6,a2,s11)=0.9; T(s7,a1,s13)=0.1,T(s7,a1,s14)=0.9,T(s7,a2,s2)=0.5,T(s7,a2,s3)=0.5;T(s8,a1,s9)=1,T(s8,a2,s18)=1;T (s9,a1,s15)=0.3,T(s9,a1,s19)=0.7,T(s9,a2,s5)=1;T(s10,a1,s13)=0.7,T(s10,a1,s17)=0.3,T(s10,a2,s0)= 0.7,T(s10,a2,s12)=0.3;T(s11,a1,s4)=1,T(s11,a2,s17)=0.5,T(s11,a2,s18)=0.5;T(s12,a1,s14)=0.2,T(s1 2,a1,s19)=0.8,T(s12,a2,s11)=0.5,T(s12,a2,s16)=0.5;T(s13,a1,s0)=0.2,T(s13,a1,s7)=0.8,T(s13,a2,s12) =1;T(s14,a1,s9)=0.7,T(s14,a1,s15)=0.3,T(s14,a2,s18)=0.5,T(s14,a2,s19)=0.5;T(s15,a1,s7)=0.1,T(s1 5,a1,s19)=0.9,T(s15,a2,s8)=1;T(s16,a1,s1)=1,T(s16,a2,s17)=1;T(s17,a1,s7)=0.6,T(s17,a1,s13)=04, T(s17,a2,s18)=0.5,T(s17,a2,s19)=0.5;T(s18,a1,s5)=0.7,T(s18,a1,s9)=0.3,T(s18,a2,s19)=1;T(s19,a1, s15)=0.2,T(s19,a1,s16)=0.8,T(s19,a2,s14)=0.6,T(s19,a2,s17)=0.4

Parallel Computing On GPU

Find the Optimal Policy of Out of Play. In order to find the optimal policy of Out of Play, the four steps are as follows.

(1) For every state, the value function should be computed, and we can find the optimal value function according to the values. The optimal value function [10] could be defined as follows.

V*(s) =. . (1)

(2) In this paper, γ is set to be 0.5, so the formula is as follows.

V*(s) =. . (2)

(3) For every state, after computing the optimal value function, find the state whose value function changes greatest. So get the maximum difference as below:

. (3)

Where n is the number of states.

(4) According to the above formulas, finding an optimal value for every state so that a unique action exits for every state is necessary. But for an action taken to a state, the state can jump to many states. The next state jumped is not unique. So the next unique state should be determined to jump to. To achieve this effect, choosing the state whose transition probability is the greatest.

. (4)

Where and are the serial numbers of the states, and means the taken action when the serial number is .

Value Iteration Algorithm With OpenCL. OpenCL has its execution model. In this paper, it's necessary to combine the value iteration algorithm with OpenCL. Fig.3 is the hybrid execution model of value iteration algorithm of MDP.

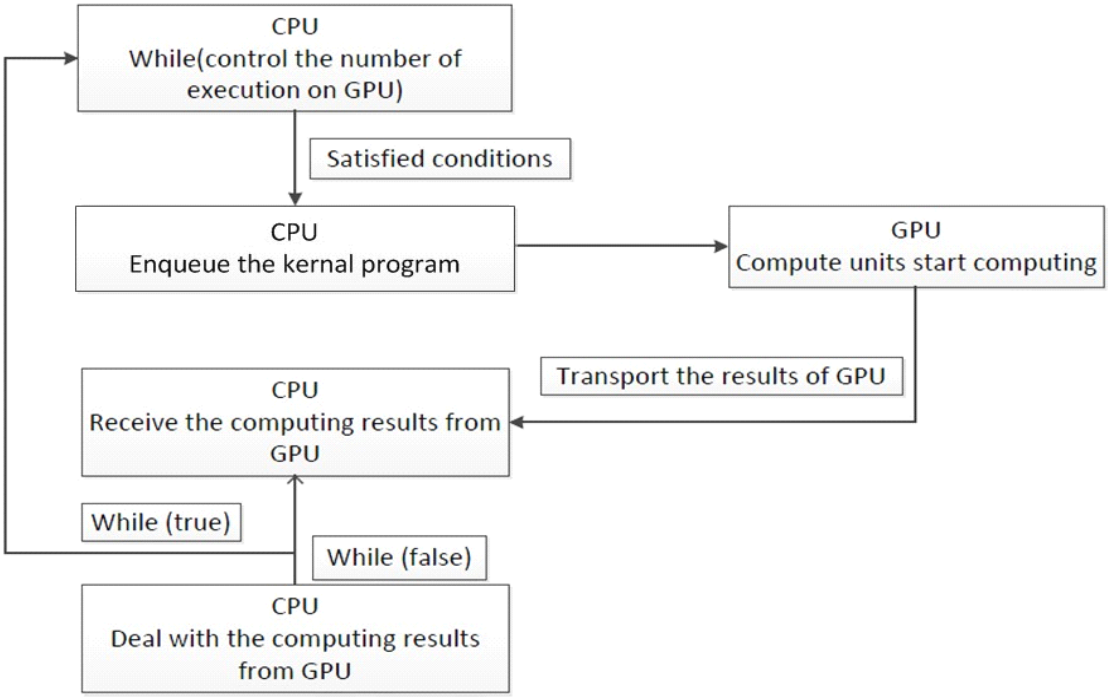

Fig.3: The model of value iteration algorithm with OpenCL

According to value iteration algorithm of MDP and the programming method of OpenCL, the pseudo code of value iteration algorithm which is used to implement the Out Of Play with OpenCL is as Fig 4.

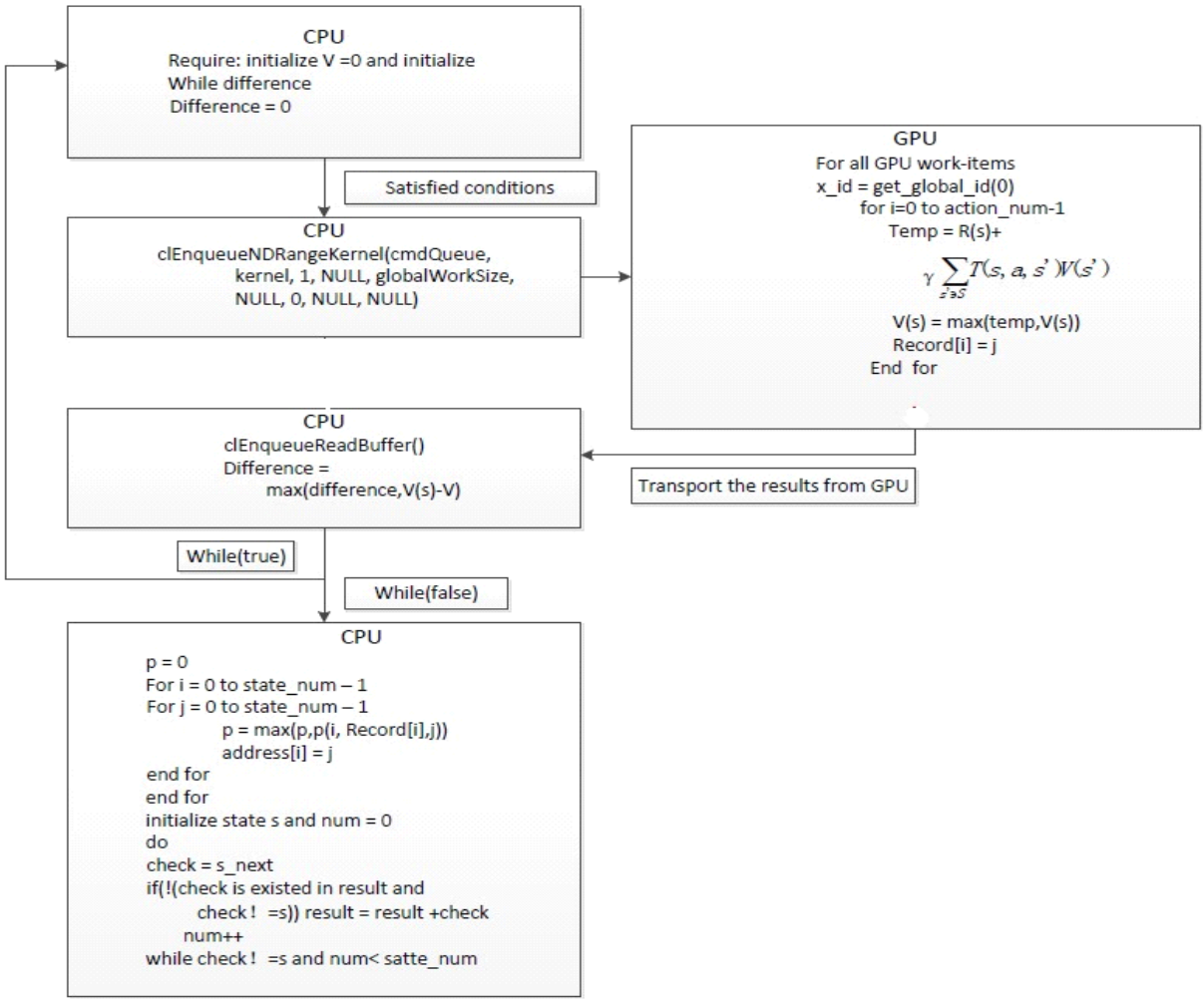

Fig.4: Pseudo code of value iteration algorithm in OpenCL

Display the Results

Part 3 has given a pseudo code of value iteration algorithm with OpenCL. After implementing the Out Of Play model, this part shows the experimental results.

Fig 5 is the result of the Out Of Play model implemented by MFC. In the program, after selecting the starting place, the system can give a recommended path with the modes of transportation, and it is an optimal path.

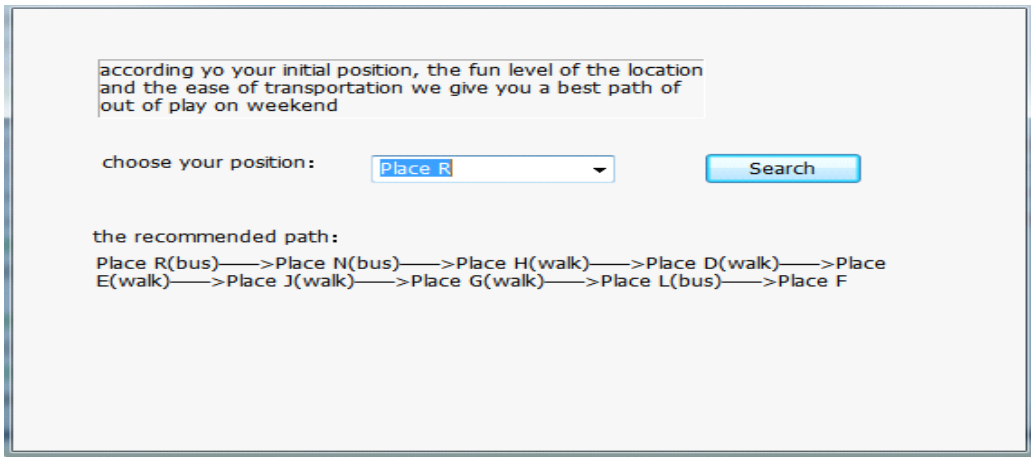

Fig.5: The result of Out Of Play

Performance Comparison

In this paper, the main improvement is to change the value iteration algorithm of MDP to parallel algorithm on GPU and implement the Out Of Play model by using parallel value algorithm of MDP on GPU. As is expected, the performance of parallel value iteration algorithm improves a lot.

(1) For different states

In this scene, after implementing the system with different number of states, but with the same , the experimental data shows the time consumed is extremely different. Table 1 shows the time used for different number of states. According to the experimental results, apparently, the time of CPU increases faster than GPU with the growth in the number of states. When the number of states reaches 10000, the time consumed by CPU is 20 times more than that of the time consumed by GPU. The performance of the model implemented on GPU improves much than that of CPU.

Table 1: time consumed on CPU and GPU with different number of states

The number of states	1000	2000	4000	10000
CPU time [ms]	2304	768	3328	23552
GPU time [ms]	1024	512	768	1024

(2) For different Δ

In this scene, after implementing the system with different , but with the same number of states, the experimental results show the time consumed is very different. Table 2 shows the time consumed for different . According to the experimental data, apparently, the time consumed by CPU increases much faster than that of GPU with the reduction of . When is 0.0025, the time consumed by CPU is 4 times more than the time consumed by GPU. The performance of the model implemented by GPU improves much more than by CPU.

Table 2: time consumed on CPU and GPU with different

	25	2.5	0.25	0.025	0.0025
CPU time [ms]	512	1280	1792	2560	2816
GPU time [ms]	512	512	512	768	768

Conclusion

This paper presents how to improve the performance of value iteration algorithm of MDP. Of course, for the calculation of the large amount of data, parallel computing is a pathway to improve performance, especially parallel computing on GPU. So we change the value iteration algorithm of MDP to parallel algorithm on GPU and implement an Out Of Play model by the parallel algorithm. According to the data from experiment, the performance of value iteration algorithm of MDP improves a lot.

In the future, we can optimize the parallel algorithm on GPU according to the platform optimization of OpenCL, so that the performance can improve much more. Except that, we can change other Markov Decision Process algorithm to parallel algorithm on GPU.

Reference

[1] Ryoo, S., Rodrigues, C., Stone, S., et al.: Program optimization carving for GPU computing. Journal of Parallel and Distributed Computing 68(10),2008,pp.1389–1401.

[2] Maciej Zbierski. A Simulated Annealing Algorithm for GPU Clusters. Springer-Verlag Berlin Heidelberg,2012,pp.750-759.

[3] Marek Błażewicz, Miłosz Ciżnicki, Piotr Kopta,Krzysztof Kurowski, and Paweł Lichocki.: Two-Dimensional Discrete Wavelet Transform on Large Images for Hybrid Computing Architectures: GPU and CELL. Springer-Verlag Berlin Heidelberg.2012,pp.481-490.

[4] Kim, J., Hyeon, S., & Choi, S. (2010). Implementation of an SDR system using graphics processing unit. IEEE Communication Magazine, 48,2012,pp.156–162.

[5] Gong C, Liu J, Chen H, Xie J, Gong Z (2011) Accelerating the Sweep3D for a graphic processor unit. JInform Process Syst 7(1):2011,pp.63–74.

[6] Sathappan OL, Chitra P, Venkatesh P, Prabhu M. Modified genetic algorithm for multiobjective task scheduling on heterogeneous computing system. IJITCC 1(2),2011, pp.146–158.

[7] Hiroyuki Takizawa, Ryusuke Egawa, and Hiroaki Kobayashi. A Prototype Implementation of OpenCL for SX Vector Systems. High Performance Computing on Vector Systems 2011, Part 1, pp.41-50.

[8] Qi, Zhang, Guangzhong, Sun, Yinlong, Xu. Parrallel Algorithms for Soving Markov Decision Process[C]. Germany:Springer-Verlag, 2009,pp.466-477.

[9] M. L. Puterman. Markov decision processes : discrete stochastic dynamic programming. John Wiley & Sons, New York, 1994.

[10] Qi, Feng, Xuezhong Zhou, Houkuan Huang, Jian Yu, Yin Zhang, Xiaolin Tong, Runshun Zhang. A MDP Solution for Traditional Chinese Medicine[C]. 2010 3rd International Conference on Biomedical Engineering and Informatics, pp.2250-2254.

Acknowledgment

This paper is supported by Guangdong Production, Education & Research Project (2012B091000050), Guangzhou Production, Education & Research Project (2011Y5-0004).