2012 15th International IEEE Conference on Intelligent Transportation Systems
Anchorage, Alaska, USA, September 16-19, 2012

A Cellular Automata based Evacuation Model on GPU Platform *

Q. Miao, Y. Lv and F. Zhu

***Abstract*— A simulation model of crowd evacuation based on the cellular automata is introduced. The implementation on GPU is focused in details. This model takes advantage of GPU to complete part of computing by certain allocation. Taking Guangzhou Tianhe stadium as an example, the model shows that the performance of computing has been significantly improved by parallel computing technique of GPU, which obtained nearly 15 times speed-up.**

I. Introduction

Nowadays, unexpected public incidents such as natural disasters, accidents, disasters, public health and social security incident are serious threats to human society. These unexpected incidents are likely to cause casualties, property damage, ecological destruction, and then further threats on economic and social stability of whole country. Especially, in densely populated areas such as a stadium, unexpected incident can easily cause stampede events. In the event of a disaster unruffled, the correct and orderly evacuation of the disaster site can save people's lives and property. So it is necessary and important to develop and evaluate the emergency response plans. How to evaluate the rationality, feasibility, optimality of the emergency response plans, is the primary concern of government and research institutes. Evaluating by practical exercise is often expensive and difficult to operate. Computer simulation becomes an optional evaluation method to improve the efficiency of evacuation plans.

Scholars from various countries have done a lot of research work in the field of evacuation simulation. In [1], Helbing put forward an evacuation model using force principle. Wang and Mao introduce a people evacuation virtual reality system in [2]. Qin and Su [3] studied the human behavior in the evacuation process . Xiao [4] and Yue [5] describe the simulation model using the Lattice Boltzmann method and cellular automata method. But in the actual simulation, the traditional serial execution on CPU usually results in low efficiency especially when dealing with large scale crowd. Parallel computing is one of the most effective ways to improve the simulation speed. However, the high performance clusters are usually expensive and difficult to popularize in the small libraries. PC cluster can be used as an alternative, but the communication efficiency and maintainability remains as bottle neck. With the development of computer hardware, graphics processing unit (GPU) has drawn a lot of attention. The large number of threads concurrent execution of GPU provides the user with a low-cost ultra-large-scale parallel computing platform. At the same time, the software programming environment, such as CUDA [6], has been developing fast during the last five years and now becomes mature enough as stable software tool set.

This paper introduces the crowd evacuation model for the GPU parallel computing. Section 2 describes the proposed model based on cellular automata principle, including the staff of modeling and rules set. Section 3 focuses on the GPU-oriented parallel algorithm based on the CUDA environment. Section 4 gives the performance evaluation based on an example in Tianhe Sports Center of Guangzhou province where the 16th Asian Games hold. Different scales of crowd evacuation are simulated to give the comparison of computational efficiency.

II. Evacuation Model based on Cellular Automata

A. Brief Introduction on Cellular Automata

Cellular Automata (CA) is a discrete space-time local dynamics model, a typical method for the study of complex systems, especially suitable for complex spatial and temporal dynamics simulation study. Different from the general kinetic model, cellular automata is not determined by the strict definition of the physical equations or functions, but rather of a series of model construction rules. Any model to meet these rules can be regarded as a cellular automaton model. In this model, each cell spread in a Lattice Grid takes a limited number of discrete states, follows the same rules of synchronous updates based on identified local rules. A large number of cellular with simple interactions constitute a complex dynamic system. It is characterized by discrete time, space, states. Each variable takes only a finite number of states and its rules of state changing are local both in time and space.

B. Evacuation Model Utilizing Cellular Automata

Three aspects of the evacuation model are introduced as follows:

(1) Modeling of the evacuation space

The modeling of the evacuation space is a process of space discretization, or dividing flat space into grids. In this model, based on the most simple two-dimensional cellular automata model, the plane is divided into square grids. Any cell at any time has one of the four states: obstruction, exit port, occupied

*Resrach supported by National Natural Science Fund of China (61004090, 60902098, 11102222).

Q. Miao is with the Graduate University of Chinese Academy of Sciences, Beijing, China. (Phone: +86-10-88256650; fax: +86-10-88259429; e-mail: miaoqh@gucas.ac.cn).

Y. Lv is with Institute of Automation, Chinese Academy of Sciences, Beijing, China. (e-mail: yisheng.lv@ia.ac.cn).

F. Zhu is with Institute of Automation, Chinese Academy of Sciences, Beijing, China. (e-mail: fenghua.zhu@ia.ac.cn).

and empty. One person can only occupy one cell. Therefore, the grid size depends mainly on the square size of a human body occupied.

(2) Rules of individual person

Behavior simulation of the personnel in the evacuation process such as how to avoid a collision, bypass the queue, and go back are difficult problems of evacuation simulation study. In this paper, four rules of individual person are considered to model a person's action in evacuation process:

- moving direction rule

To simplify the problem, we define that people can only choose from the surrounding eight grid position as a next step. The selection criterion is choosing the grid that gets the person most close to the exit port. As an example shown in Figure 1, assuming a person is in grid 5, and if the rest cells are empty, then the possible next step can be one of the 8 cells. But the evacuee chooses the one that closest to an exit.

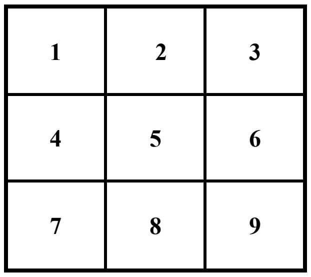

Figure 1. Example of moving direction

- conflict and competition rule

A CPU based evacuation model simulates each person in a sequential mode. That is, persons are processed one by one, without any conflicts in a certain cell. But in a GPU based evacuation model, each thread in the GPU is on behalf of an evacuation person. When people move in parallel within one time step, there are conflicts of avoidance or competition for a certain cell.

- bypass rule

In the evacuation process, a person has to make choice of next move of the cellular grid. Under normal circumstances, the person walks towards the grid cell nearest to an exit. However, when the nearest grid cell is not empty, a person can choose another cell that is little farther to the exit. For this model, a weight is assigned to each of the 8 possible adjacent cells according to their distance to a certain exit area. Then the weights are sorted in sequence. When moving, a person chooses a cell with most small value. If this cell is occupied, the cell with second smaller weight is chosen as an alternative. If all 8 cells are occupied, the person has to stay in the current cell waiting for chance.

- exhumation rule

In program design, a person may repeat the move, fall into the infinite cycle of back and forth along a certain coordinate. In order to avoid the exhumation phenomenon, we established a circular linked list to record the historical location coordinates for each person. The linked list stores the last five cells that a person just visited. When the next choice of the grid cell is the same as one of the previous five cells, a new cell is selected as a substitute choice.

- building borders

Building such as walls, flowerbeds relate to the cells that marked as obstruction. For the cells along with the border of buildings, there are less 8 choices for the next step. If a person walks into the building, he will be rebounded back to the opposite direction.

- Exit port of space

To determine whether the personnel out of the evacuation area, this model uses the coordinates of the boundary to do judgment. You can determine that a person has been evacuated to the end if the evacuation of personnel coordinate and the exit of the horizontal axis or vertical axis are equal.

I. Implementation on CUDA Platform

GPUs have evolved to the point where many real-world applications are easily implemented on them and run significantly faster than on multi-core systems. Future computing architectures will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs. CUDA is a parallel computing software platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the general purpose graphics processing unit (GPGPU). Our work depends on the GPU with the support by CUDA.

A. Data storage

Due to the specifications of modern GPU architecture, memory on GPU must be allocated and managed carefully to maximize the simulation performance. The data need to distribute on GPU memory include information of the cell grid, individual person, and positions of exit or destinations.

- definition of cell grid

The cell grid is a structure of two dimensional array representing the evacuation environment. It has two features, one is huge in size, and the other is dynamic during the evacuation process. For the cell grid will be queried by all persons in the environment, we put it in the global memory. But the access to the cell grid cannot satisfy the coalesced access principle of CUDA, which results in low performance. What's more, for NVidia G80/G200 series GPU card, there is no cache for the global memory.

In order to promote the memory access efficiency, data read and write of global memory must be aligned, with a width of 256 Byte. Without alignment, the read and write process will be scattered into multi operations, which greatly affect the efficiency. On the other side, if the r/w operations of multi half warp can be managed to satisfy the coalesced access principle, operations will done in one time step, which greatly improves the performance.

- information of individual person

Traditionally, in order to use the cache on CPU effectively, we tend to use the AOS (array of structs) to organize data of all evacuation people. On the contrary, in order to meet the requirements of coalesced access principle on GPU, we use a different data structure called the SOA (struct of Array). The SOA mode is shown in the following pseudo-code.

```
typedef struct Persons{
  int numPersons;
  int *x;
  int *y;
  int *goal;
  int *isFree;
……
}Persons;
```

Figure 2. Struct for a person in SOA mode

Among code in Figure 2, the *x, y* are coordinates of a person in the cell grid; *goal* refers to the each goal export; *isFree* is a flag to mark if a person has fled the evacuation area. This storage method ensures that the storage and loading of stream processor provide the largest possible bandwidth.

- positions of evacuation exit

Constant memory in GPU is the hardware dedicated to graphics computing, and mainly used to speed up access to the constant. It is off-chip memory but has a cache, which makes it about ten times faster than global memory. As the position of exit and buildings remain unchangeable in the simulation process, so we store them in the constant memory on GPU to take advantage of the constant cache to improve performance.

The only thing we need to pay attention is that the size of constant memory is much smaller than the global memory, 64K as the maximum. But the storage of export information, the target export position and the offset vector in the case of this paper are relatively small, therefore does not exist the danger of memory leak.

B. *Selecting positions of next step*

The algorithm on GPU mainly focuses on the initial location of each person, determines the priority of candidate cells to move, and chooses one from the candidates in the next step.

- Personnel and position initialization

The initialization function traverses each export of the dangerous place. If there is empty cell among the 8 cells around an export, a person is allowed to occupy this position, until no one stayed in the dangerous place.

- Priority of candidate cells

The model compares distances from each cell of the 8 candidate positions to its target location, sorts them using bubble sort algorithm directly. An array can be used as storage of candidates. But after complied, they are allocated in the local memory by the NVCC complier. The local memory is a part of GPU memory space, with a low access speed. In our case, there are at most 8 candidates, the total size will not exceed the shared memory capacity, and therefore, we put them in the shared memory to take advantage of the bandwidth.

- Select and enter the next position

In case of traditional serial simulating on CPU, persons are treated one by one without conflicts with each other. That is, each person deals with a static evacuation environment when taking decision and actions. But on parallel computing platform, things get complex. When a person finds an empty cell around him and gets ready to enter, others may have entered that cell first. Persons were simulated on different threads (or stream cores), which results in conflicts among people. This paper uses atomic function atomicCAS to deal with this problem. Atomic function is used to ensure that only one thread at a time to change the function of a memory location. The role of atomicCAS is to compare the current state of one cell, if the cell is not currently being occupied, then occupy the cell and returns the memory location. In essence, the atomic function is a serial process, which can cause performance degradation, but this is inevitable.

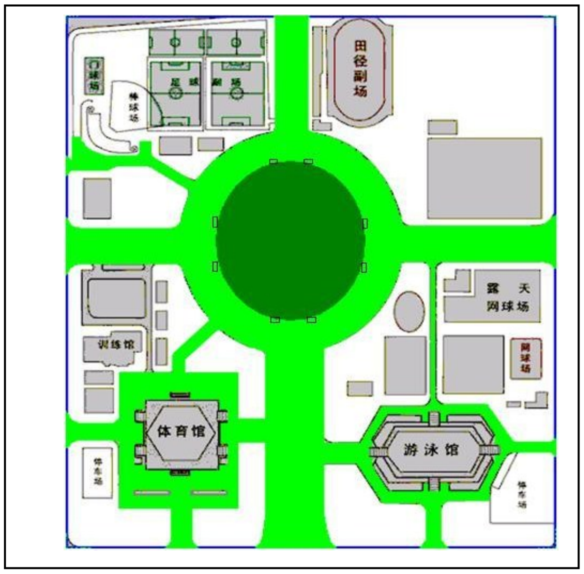

Figure 3. Map of Tianhe Sport Center

II. TEST RESULTS AND ANALYSIS

In this paper, we take the Tianhe Sports Center of Guangzhou province, which is the main stadium of the 16th Asian Games hold in 2011. Tianhe Center has 8 gates around the building, as illustrated in Figure 3.

The rectangular area shown in figure 3 is the evacuation space, which is nearly 400m in width and 500m in length. We choose 0.5m as a grid cell size, and then the whole area can be modeled with an 800*1000 array. Each element in the array maps to a grid cell, which can be occupied by only one person. When taking simulation, this model was simplified. The inner evacuation process in the stadium is neglected and the passageways are reduced to 8 gates and road in four directions.

We designed a set of simulations with different number of audiences. From the model, we can observe the evacuation process under different conditions, or how many people have evacuated at a given time. An individual person can be tracked and its trace can be drawn. After certification and modification based on these test sets, we took simulations on both serial computing and the new parallel GPU computing platform to see the performance.

A. *model certification with different people*

We took simulations with a number of people at 500, 2000, 5000, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, and 100000 respectively.

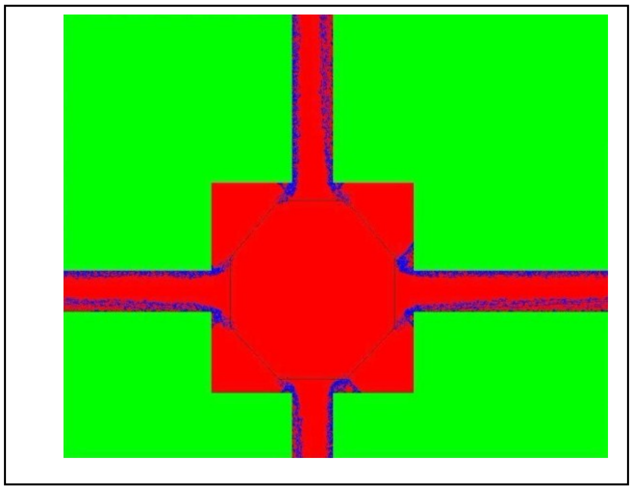
Figure 4. Distribution of 50000 people after 500 seconds

Some of the simulation results are shown in figures from number 4 to number 7. The octagon in the center of figure is Tianhe center, and its 8 vertices represent the 8 exit ports. The red color area represents the evacuation space, and the green color area is regarded as obstacles such as buildings. The curve shows the trace of one person and the cloud of dots show the distribution of all people at a given time. Figure 4 and figure 5 show the whole distribution of all 50,000 people after 500 seconds and after 1000 seconds separately.

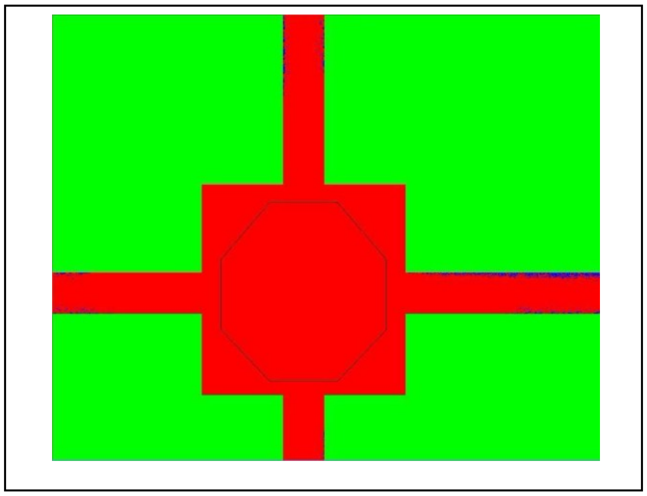
Figure 5. Distribution of 50000 people after 1000 seconds

From the two figures we can see that the proposed model generates an evacuation process similar to the process of audience exit, in the macro view.

Curves in figure 6 and figure 7 illustrate the traces of people with id 9998 and 49999. We can see that the person choose an exit port first, and then walk outward with the crowd. He can change his route if too many people in front him, which can be observed in both figures.

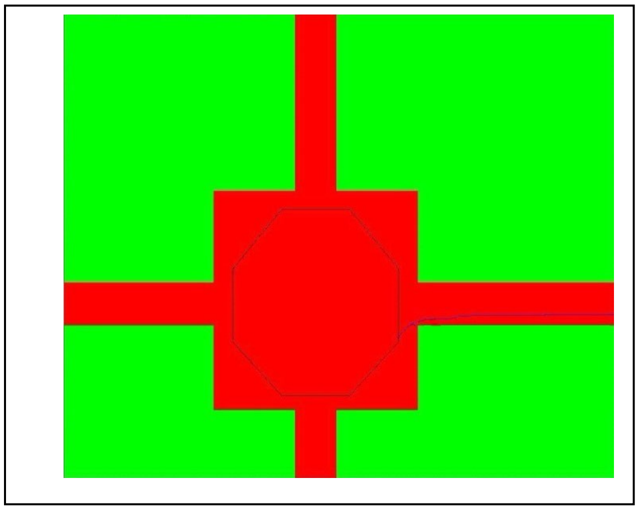
Figure 6. Trace of the No.49999 person

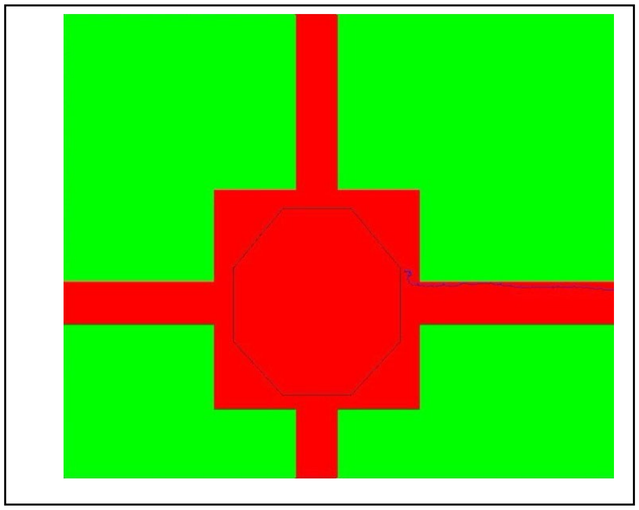
Figure 7. Trace of the No. 9998 person

B. *Performance Comparison on Two Platform*

We tested the model on three different platforms, two of them are CPU platform: Intel (R) Xeon 5482 (3.2 GHz) and T7250 (2.0 GHz) separately. And the other is the NVidia GTX 295 GPU, with 480 stream processors. We selected a total of 12 sets of data to run on the three above platform respectively. The test cases and the performance measured by the simulation time for the same set of cases. The speedup is defined as the ratio of simulation time of serial CPU platform to time of GPU platform for the same case. The speedups for all the 12 cases are shown in figure 8.

From the curve of GT295/T7250, we can see that the speedup reaches max value 14.85 for the case with 30000 people. For the cases have less than 30000 people, the curve increase rapidly. For the cases with more than 30000 people, the speedup remains table just with a slightly decrease. From the curve of GT295/X5482, we can see that the speedup reaches max value 9.15 for the case with 50000 people. The difference of CPU frequency results in the different performance, for Xeon 5482 has a much higher frequency.

The two curves both have summits when dealing with 30000-50000 people. The reason is that when simulating with less people, less stream processors are used, which cannot exploit the whole capacity of GPU. As the number of people increasing, more and more stream processors take part in the computing. If the ratio of people number to stream processor number is integer power of 2 (such as 1,2,4), the whole parallel capacity of GPU is exploited, which results in summit speedup.

The slightly decrease of speedup when people number larger than 50000 is due to the specific architecture of GPU and its Single Instruction Multi Data (SIMD) mode, a warp of 16 threads run under a same instruction. For our model, each person is assigned a thread. But when a person has already evacuated, its thread still runs. Thus all 16 threads within a warp need to run unnecessary.

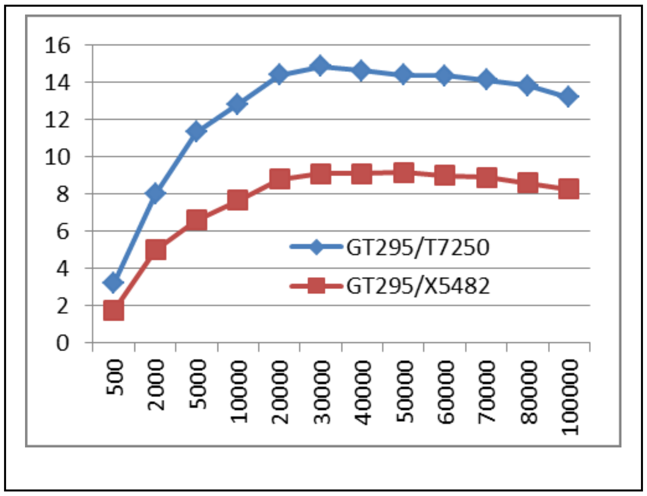

Figure 8. Speedup on GPU based on the 12 cases.

III. Conclusion

From the experiments we can draw two conclusions when using GPU to improve the performance.

- The performance improvement of the proposed evacuation model depends on the full use of all stream processors in GPU.
- The evacuation performance improvement is proportional to the number of built-in stream processors in GPU.

The work introduced in this paper is a preliminary try is to use the GPU parallel computing for evacuation simulation. A significant improvement in performance was achieved, but the proposed model still needs to improve. Though GPU has much stronger parallel capacity than CPU, its ability of logic control is relatively poor. So the model of person is designed relatively simple. However, due to the hardware architecture of the GPU is in continuous improvement, and the corresponding CUDA software is becoming mature, GPU is one of the most promising platforms in the near future.

Acknowledgment

This work was supported by projects of National Natural Science Fund of China (61004090, 60902098).

References

[1] Helbing D, Farkas I, Vicse T. Simulating dynamical features of escape panic [J]. Nature, 2000, 407(28):487-490.
[2] Z. Wang, T. Mao, J. Jiang and S. Xia. Guarder: Virtual Drilling System for Crowd Evacuation Under Emergency Scheme, Journal of Computer Research and Development, 47(6):969-978, 2010.
[3] W. Qin, G. Su, X. Li. Technology for simulating crowd evacuation behaviors. International Journal of Automation and Computing, 2009, 6(4):351-355.
[4] Y. Xiao, W. Feng, and Y. Chao. Large-Scale Flow Evacuation Research Based on Lattice Boltzmann Method, Computer technology and development, Vol.21, No. 7, July, 2011.
[5] H. Yue, C. Shao, and Z. Yao. Pedestrian evacuation flow simulation based on cellular automata, ACTA PHYSICA SINICA, 2009, 58(7).
[6] NVidia, CUDA C Programming Guide, website: http://www.nvidia.com/object/cuda_home_new.html