# Study of Platform Passenger Evacuation Simulation Based on GPU

Zhiyong Cai[1]; Qianni Deng[1]

[1] Shanghai Jiaotong University, Minhang, Shanghai, 200240

maxavier@sjtu.edu.cn

**Keywords:** GPGPU; CUDA; Pedestrian Evacuation; Parallel Computing; Simulation

**Abstract.** Pedestrian Evacuation Simulation can be used to support developing the evacuation solutions for subway stations. Although the pedestrian evacuation based on general purpose processors acquire good effect when the simulation scale is very large. We investigated the effect of the layout of subway stations on the pedestrian evacuation time through simulation. Based on the cellular automaton model, we partitioned the platform space into a two-dimensional grid structure. The walking path of pedestrians can be represented as movement on the grid according to a specified rule. Although the weighted shortest path algorithm is an appropriate routing rule for pedestrians, it puts pressure on the CPU simulation performance. To solve this problem, we introduced GPGPU as the parallel computing hardware to accelerate the simulation process. Experimental results show that the computation time of each time-step of the simulation on GPU is 20 times faster than on CPU. The real time simulation can support rapid decision-making and department response to emergencies.

## Introduction

Urban Rail Transit station is a crowded place relatively closed. Event of fire, accidents, business interruption and other emergencies, the crowd chaos, passengers panic is difficult to control, and it may cause the evacuation extremely difficult and may even squeeze stampede and other major accidents. How to issue safety warning before the accident and contingency measures is a major issue for rail operation to solve.

Although some practical approach, such as pre-disaster evacuation drills can help to improve the design of buildings [1,2], to reduce the accident rate effect. But in a real accident, personnel-intensive exercise is often unpredictable and reproduction, evacuation drills when developing programs often lack flexibility and reliability. Computer simulation dense crowd evacuation process is a low-cost, safe and effective method. By simulating the evacuation behavior at the station, you can analyze the performance of ultra large scale safe evacuation crowd gathering station to reach emergency help develop solutions.

There are already some software [3-4] was developed to simulate the evacuation of the building. The software can simulate the behavior of different lines of people, on the utilization of space, facilities and services as well as the ability to assess the performance of buildings evacuated. Simulation software commonly used are microscopic model for the individual to establish a simple rule to simulate the movement of the entire complex system. However, for large-scale simulation of complex systems behavior of each individual calculation is very large, it is difficult for real-time simulation using a general purpose processor (CPU).

In order to solve the above problems, there have been some studies using multi-processor or a graphics processing unit to enhance the performance of the simulation. Coutry and Musse [5], who proposed a pedestrian based graphics processing unit and social force model of evacuation simulation models. Richmond [6,7] proposed a rule definition framework, the use of a scripting language similar to C language, programmers describe only the rules of conduct of the individual, it does not need to know the GPU underlying data storage and communication mechanisms. The results of this work show that: the performance of the simulation system is highly dependent on each individual degree of synchronization in the simulation calculations. Synchronization is higher, the better the performance. Other studies also showed that: the simulation program makers often need to understand the hardware architecture of the GPU, fully developed evacuation matching algorithms and hardware structure, in order to obtain good performance.

This study is a special case of evacuation scene: pedestrian subway platform evacuation simulation. Issues need to be addressed: the subway platform is a rectangular area, open to the train in both directions share the same platform. Given the layout of the site, the number of stairs and capacity per unit time, given the number of lines is calculated at the same time a train arrives and two passengers on the platform of the maximum time for evacuation.

Problems of this study apply to most of the subway station, practical. According to the site's layout features, based on the metal cellular automata model, select the appropriate individual parallel accelerated evacuation routing algorithm [8], take advantage of hardware features to get the evacuation simulation GPU real-time on the GPU.

## Background

**GPU.** Compared with the traditional CPU, GPU has significantly different in terms of hardware structure, calculation methods. Most GPU transistors are used to perform unit, can accommodate thousands of computing threads no logical relationship, good numerical highly parallel. Currently, GPU floating-point arithmetic processing capacity has reached a few times or even a hundred times over the same period of the CPU, GPU general computing technology has become a mainstream parallel computing technology, it has been successfully applied to fluid simulation, database applications, intelligent information processing systems and data mining and other commercial applications.

CUDA is NVIDIA's GPU common computing platform, which uses a general purpose parallel computing architecture to use GPU hardware. The basic idea is to try to develop CUDA thread-level parallelism, so that these threads can be dynamically scheduled for execution in hardware plurality of operational units. CUDA program can be divided into two parts, namely a series of runs on the GPU Kernel program running on the CPU serial program. Wherein, Kernel responsible for parallel computing part, CPU serial program is responsible among other tasking Kernel. Kernel program execution in hardware is performed by multiple threads in parallel.

**Cellular Automata.** Cellular automata model plane is divided into equal-size square grid, the equivalent of one RMB each grid cell, the grid or by pedestrian occupied or occupied by the obstacle, or is empty. Each cell can only accommodate a pedestrian and each pedestrian can only occupy one RMB cells. Pedestrians must not cross the barrier. Different cellular are given different weights according to the state. Evacuation process is discretized into equal time steps. Within each time step, shown in Figure 1, can be moved to the pedestrian 8 directions, translational or vertical direction to move the price of 1, the cost of moving the diagonal direction is 1.4. Pedestrians will be to determine the next moment the movement direction and movement according to the state the cost of their moving target and its neighbor lattice points, or stop waiting, or moving a cell.

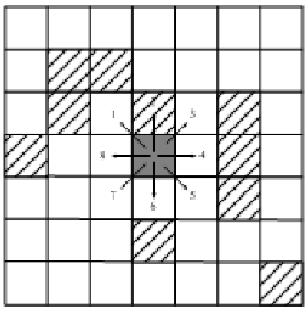

Fig. 1 Cellular Automata Model

**Routing Algorithms.** The pedestrian evacuation from the location to export the routing problem can be abstracted as calculated for each cell in the single-source shortest path problem with the right to export (single source shortest path: SSSP), namely: a given weighted directed graph G (V; E; W) and each source node S, node v to find the shortest path S. Dijkstra's algorithm is SSSP problem common solution. The basic idea is the source node as the center outward layers expand, the source node to calculate the minimum weight of each layer path node until the expansion until all nodes. Although there are other search suboptimal shortest path heuristic search algorithms such as A * algorithm, convergence speed faster than Dijkstra's algorithm, but Dijkstra algorithm is more appropriate parallel multi-threading implemented on a single processor on the GPU.

## Routing Algorithm Based on GPU Parallel

When pedestrians are moving in the next position, they are not only to consider whether the move away from the outlet near, will also consider whether the path to the location in front of congestion. Figure 2 shows an example, the figure of the staircase has two exits, pedestrian move to the stairs, the next possible move Cellular can be A, B or C, where B and C to the stairs two outlets in Europe s the shortest distance. But if we consider the crowded situation, A and C to overcrowding in the shortest path on the exit stairs better than B, even for pedestrians the most convenient location next move is B, but in reality, he is more likely to choose C as a next moving position.

Therefore, in each time step of the simulation process, need to calculate the most appropriate route to move to evacuation outlet instead shortest path and that is every pedestrian selected in the "optimal route" on the neighbor node on all pedestrian cells as the direction of the next move. To calculate the "optimal route", we define cell "dynamic distance parameter": Dcell = (the path to the exit of the cell weights of all cellular mobile expense + path) Min, said the current the cell to export minimum cost required.

Below we discuss how to compute all the cells in the GPU dynamic distance parameters. A plurality of internal GPU computing unit, assign one thread to calculate each cell at a time step of dynamic parameters to complete the distance, multiple threads execute in parallel. Calculated as follows:

For each cell, steps A and B:

A. If the dynamic distance parameter of the cell has been updated in the previous step, the cell of each unit to perform a neighbor:

Dynamic distance parameter neighbor units= current cell dynamic distance parameter + the cost of the current cell to the mobile neighbor cells;

If the result is less than the distance between the original dynamic neighbor cell parameters, to the neighbor cells send out updates from the parameters of instruction

Accept all its neighbor cells modify parameters dynamic distance instruction, select one of the minimum, if the distance is less than the current value of the dynamic parameters of the cell, to update the current cell parameters dynamic distance.

For all cells A and B in parallel repeat steps until no cell dynamic distance parameter updates.

We use a two-dimensional array to represent the station platform Bij individual cells, Wij is the weight of the cell on, Dij represents the cell to export dynamic distance parameter, Uij for dynamic distance parameter cell to export the updated, MijIt represents the cell mask code. S for all export nodes set. In each iteration, a node according to their own mask bit to determine whether the new dynamic distance calculation parameters neighbors. Step 2 implements the algorithm A, algorithm implements 3 steps B, Algorithm 1 Algorithm 2 and 3 repeated calls to complete the dynamic parameters of all units from a time step of the calculation.

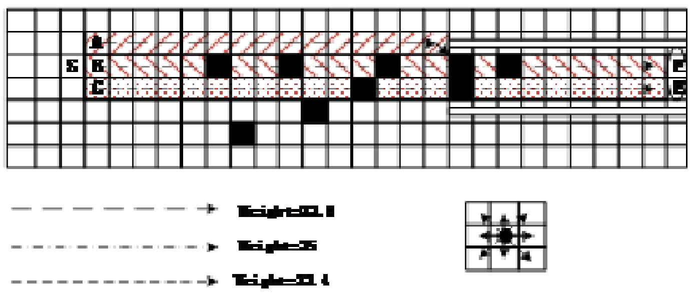

Fig. 2 Routing Algorithm Example

Algorithm 1 with CUDA update within a time step away from all the cells of the dynamic parameters

```
Algorithm 1CUDA Shortest Path to Exits ( $C_{ij}$, S)
1.  Take parameters for $W_{ij}$, if $C_{ij}$ is taken over by people then $W_{ij} \leftarrow 1$else $W_{ij} \leftarrow 0$, if $C_{ij}$ is taken over by other obstacle then $W_{ij} \leftarrow \infty$
2.  Initialize $M_{ij} \leftarrow$ false, $D_{ij} \leftarrow \infty$
3.  for each exit vertex $s_i$ in S, M[$s_i$] $\leftarrow$ true, D[$s_i$] $\leftarrow$ 0,U[$s_i$] $\leftarrow$ 0, //S
4.  while $M_{ij}$ not Empty do
5.  for each cell$C_{ij}$ in parallel do
6.  Invoke CUDA SPSE KERNEL1($C_{ij}$;S; $W_{ij}$; $M_{ij}$; $D_{ij}$, $U_{ij}$)
7.  Invoke CUDA SPSE KERNEL2($C_{ij}$;S; $W_{ij}$; $M_{ij}$; $D_{ij}$, $U_{ij}$)
8.  end for
9.  end while
```

Algorithm 2 calculates parameters dynamic distance around node Kernel program

```
Algorithm 2    CUDA SPSE KERNEL1($C_{ij}$; S; $W_{ij}$; $M_{ij}$; $D_{ij}$, $U_{ij}$)
1.  tid    ← getThread ID
2.  if M[tid] then
3.  M [tid] ← false
4.  for all neighbors nid of tid do
5.  if U[nid] > Cost[tid, nid]+D[tid] then
6.  U[nid]←Cost[tid, nid]+D[tid]
7.  end if
8.    end for
9.  end if
```

Algorithm 3 updates from the parameters and the mask code of Kernel program

```
Algorithm 3    CUDA SPSE KERNEL1($C_{ij}$;S; $W_{ij}$; $M_{ij}$; $D_{ij}$, $U_{ij}$)
1.  tid ←getThreadID
2.  if D[tid] >U[tid] then
3.  D[tid] ← U[tid]
4.  M[tid]← true
5.  end if
6.  U[tid] ← D[tid]
```

Fig. 3 is a time step of each cell updates from the parameters, and then select each pedestrian movement position the next time step of the dynamic distance parameter and move around the node cost of the process. Exit stairways as a starting point, layer by layer out updates from the parameters of nodes in each layer, each layer, each node to calculate and modify the next level dynamic distance parameter neighbor nodes, these calculations are completely parallelized . At each time step, call the algorithm a, after each cell is calculated dynamically from the latest parameters, one pedestrian moving to the cell under the basis of these parameters. Simulation of pedestrian evacuation process is repeated every time step until all pedestrians have left the outlet. Simulation process is shown in Figure 4.

## Analysis of Results

Test Environment: 4 GB of memory, 3.0GHz clock speed of Intel Core i5-2320 processor; 1GB of memory to NVIDIA GeForce GT 620 graphics processing unit. Software: Windows 7 operating system, C ++ CUDA 6.5 and Visual Studio2013 on.

**Dynamic Distance Parameter Calculation.** The use of CPU and GPU, 120m*15m (7200/cellular) and 200m*15m (12000 /Cellular) the size of the evacuation of the platform 1000 performance test. Simulation platform number is two stairs, stair unit time pass rate 2/sec. Table 1 shows the computation time of a time step, namely: the CPU

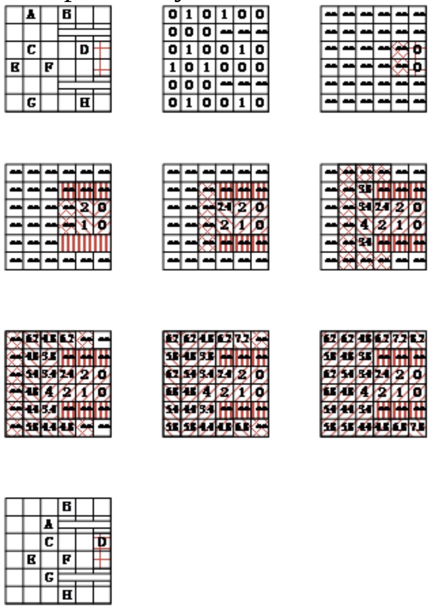

Fig. 3 A time step within each cell update parallel process parameters dynamic distance

And on all cellular GPU computing average performance is in dynamic distance parameters of the time step. We can see that a single time step on the GPU can get 10 times the performance speedup.

Table 1 step at a time on the CPU and GPU computing time (ms)

| Platform L*W | CPU | GPU | ACC |
|---|---|---|---|
| 120m*15m | 124 | 13 | 9.5 |
| 200m*15m | 147 | 15 | 9.8 |

**GPU-Based Large-Scale Passenger Evacuation Simulation Platform.** Different platform dimensions (120m * 15mm and 200m * 15mm), the number of different numbers of the evacuation process was simulated, simulation parameters stairs above. Experimental results show that, GPU performance simulation algorithm is better than using simulation on the CPU, accelerating than the increase in the number of scale increase is particularly evident. GPU-based platform simulation, when the number reached 1,000 people or more, effective than CPU-based simulations. In the evacuation of the number reached 10,000 cases, GPU-based method can reach 20 times speedup. Experiments show that the site 200m*15m scenario, population-based real-time simulation CPU only about 1,000 people, but the way GPU-based parallel processing of real-time simulation of 10,000, if the high-end graphics computing unit, real-time simulation of The number can also be increased.

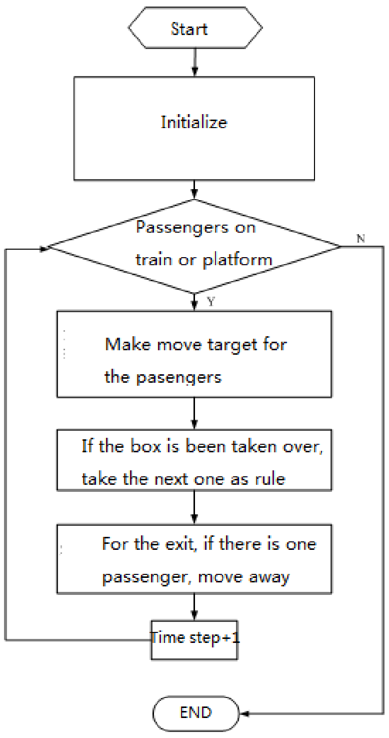
Start
Initialize
Passengers on train or platform
N
Y
Make move target for the pasengers
If the box is been taken over, take the next one as rule
For the exit, if there is one passenger, move away
Time step+1
END

Fig. 4 Simulation flowchart

Table 2. The entire evacuation simulation on the CPU and GPU computing time (s)

| 120m*15m | | | |
|---|---|---|---|
| **Evacuation number** | **CPU** | **GPU** | **ACC** |
| **1000** | **74** | **7.8** | **9.48** |
| **3000** | **147.5** | **13.4** | **11** |
| **5000** | **360.7** | **22.1** | **16.32** |
| **10000** | **884.3** | **45.7** | **19.35** |
| 200m*15m | | | |
| **Evacuation number** | **CPU** | **GPU** | **ACC** |
| **1000** | **83** | **9.1** | **9.12** |
| **3000** | **189.6** | **14.9** | **12.72** |
| **5000** | **510.3** | **28.7** | **17.78** |
| **10000** | **897.2** | **44.2** | **20.3** |

## Conclusion

For large-scale evacuation of people in subway platform, this paper implements parallel simulation algorithm based on graphics processing unit. Compared to the CPU serial way, our proposed method can be implemented in real-time simulation of when the crowd in large scale. Based on the basic idea of parallel Dijkstra's algorithm, the paper proposes a parallel dynamic distance update algorithm based on GPU to make full use of the multi-GPU computing unit, but also improve the algorithm's versatility. The algorithm also exist improvement room and you can also take advantage of the GPU memory hierarchy architecture to further improve simulation performance, reducing computation time. This will be one of our next works.

## References

[1] NPFA 130-2010, Standard for Fixed Guideway Transit and Passenger Rail Systems. National Fire Protection Association. 2010.

[2] GB50157-2010, Metro Design. Planning Press. 2010.

[3] M. Quinn, R. Metoyer, K. Hunter-Zaworski. Parallel Implementation of the Social Forces Model. In Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics (August),2003:63–74.

[4] D. Helbing, P. Molnar. Social Force Model for Pedestrian Dynamics. Physical Review E, 1995,51: 42-82.

[5] N. Courty, S. R. Musse. FASTCROWD: Real-Time Simulation and Interaction with Large Crowds based on Graphics Hardware. ACM SCA 2004-Symposium on Computer Animation, 2004.

[6] P. Richmond, Coakley Simon, Romano Daniela. A High Performance Agent based Modeling Framework on Graphics Card Hardware with CUDA, Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS),2009.

[7] P. Richmond, D. Romano. Template-driven Agent-based Modelling and Simulation with CUDA, GPU Computing Gems Emerald Edition, Morgan Kaufmann,2011:313-324.

[8] Pawan Harish, P. J. Narayanan. Accelerating Large Graph Algorithms on the GPU Using CUDA. High Performance Computing–HiPC 2007, Lecture Notes in Computer Science,2007, 4873: 197-208.