Machine Learning Enhanced CPU-GPU Simulation Platform for 5G System

Yuling Ouyang[1,2], Caiyuan Yin[1], Ting Zhou[1,2(✉)], and Yan Jin[3,4]

[1] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, China
{ouyangyuling,yincaiyuan2018,zhouting}@sari.ac.cn
[2] Shanghai Frontier Innovation Research Institute, Shanghai, China
[3] College of Sciences, Shanghai Institute of Technology, Shanghai, China
jinyan@sit.edu.cn
[4] Key Laboratory of Wireless Sensor Network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China

Abstract. The exponential growth of mobile terminals and the explosion of data volume are promoting the continuous evolution of mobile communication network and also increasing the complexity of the system. Meanwhile, 5G system-level simulation also requires more complex operations and more data processing. Conventional system simulation platform based on CPU can not satisfy the computing power requirement of system-level simulation of 5G. For tremendously shorten the execution time, we proposed to develop the CPU-GPU based parallelization platform, which adopts Logistic Regression algorithm to optimizing the use of computational resources. Numerical results demonstrate the effectiveness in terms of reducing execution time and guaranteeing reliability of system-level simulation result in 5G scenarios.

Keywords: CPU · GPU · System-level simulation · Logistic regression · enhanced Mobile Broadband (eMBB)

1 Introduction

With the popularity of intellectual mobile terminals, the mobile communication services will rise 1000-fold by 2020 [1–3]. To meet application requirements, ITU officially unveiled its vision for 5G, identifying three scenarios for enhanced Mobile Broadband (eMBB), ultra Reliable & Low Latency Communication (uRLLC) and massive Machine Type of Communication (mMTC) [4]. In 5G systems, various techniques are also taken into consideration including mmwave communications, massive multi-input multi-output (MIMO), enhanced device-to-device (D2D) communications, enhanced small cells, non-orthogonal multiple access (NOMA), multi-band carrier aggregation, dual connectivity, and so on [5].

C. T. Calafate et al. (Eds.): MONAMI 2021, LNICST 418, pp. 25–38, 2022.
https://doi.org/10.1007/978-3-030-94763-7_3

In order to study various technologies of 5G, system-level simulation is inevitably used to ensure the realistic feasibility of theoretical results. In the eMBB scenario, the layered heterogeneous network architecture will bring more complex topology structure and more intensive data operation to system simulation [6,7]. Conventional CPU serial mode applied to 4G system-level simulation has obvious disadvantage of low efficiency, which cannot meet the requirements of 5G system-level simulation.

There are three major approaches to deal with large scale simulation in literature. The first approach proposed CPU-based parallel distributed simulation platforms [8–10], while Multi-core CPUs are expensive and suitable for handling logical rather than massive data operations. The overhead may increase drasticallly in mobile environment if the network topology and machines mapping is not dynamically managed. The second approach aims to realize the simulation entirely on the GPU [11–13]. However, the GPU is not fully X86 compliant and did not support CPU features, needs a specific software architecture to disclose its power and did not support memory lock mechanism. The third approach aims to increase the efficiency of the simulation locally by offloading the most CPU-intensive part of the simulation from the CPU to a dedicated co-processor [14,15], such as GPU. In this case, whether the computing resources can be reasonably utilized will be the key factor to determine the simulation efficiency [16].

Motivated to greatly improve the simulation efficiency, we propose a method of introducing machine learning to optimize the allocation of computing resources. In conclusion, our main contributions can be summarized by the following three aspects:

- A parallelization simulation platform with CPU and GPU is proposed. The platform aims to reduce the operation time by allocating resources reasonably to parallelized modules.
- The classification algorithm of parallel modules is designed with the idea of machine learning. According to the Logical operational complexity, Numerical operational complexity and execution times of the parallel module, the appropriate weight coefficients are given by training. Finally, the parallel modules are classified by fitting function [17,18].
- Experiments on large-scale 5G system-level simulation of eMBB scenario have demonstrated that the proposed CPU-GPU based parallelization simulation platform consistently outperformed standard CPU computational baselines for operation time.

The remainder of this paper is organized as follows. Section 2 describes details of the architecture of CPU+GPU parallel computing platform proposed by us. Section 3 is dedicated to the Logistic Regression algorithms and parallel gain analysis, respectively, utilized for allocation of computing resources and theoretical location of the optimization target. Comprehensive experiments on 5G system-level simulation are carried out in Sect. 4. This work is concluded by summarizing our main contributions in Sect. 5.

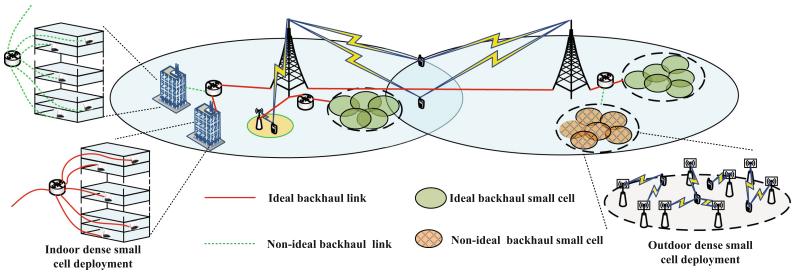


Fig. 1. Typical application scenarios for 5G eMBB.

2 5G System Simulation Parallel Framework

Wireless networks beyond 5G (fifth-generation) and 6G (sixth-generation) are supporting a large scale of novel usage scenarios and applications with high reliability, low latency, and higher frequencies. Accurate scenario characterization and simulation modelling are fundamental to evaluate the designed technologies and system performance [19]. A typical eMBB scenario is shown in Fig. 1. In order to efficiently design and implement 5G simulation system, it is necessary to deeply analyze the characteristics and implementation schemes of various application scenarios and candidate technologies [20,21].

The design of the whole platform includes two parts: 5G system simulation framework design and resource parallel architecture design. In Fig. 2 shows the parallel simulation analysis process in the form of flow chart. On the design of the 5G system simulation framework, in view of 5G network virtualization, definable network, as well as complex scenarios and business types, we have adopted the dynamic modeling technology, layering and modeling the networking, analyzing the dependencies between the internal modules of the program to decoupling the program. Further to componentization and interface decoupling design for all levels of modular simulation object model, at the same time, the simulation model components and simulation parameters are obtained by mapping based on the simulation scenarios and business model. Then dynamic configuration is carried out according to the simulation parameters, which is combined into a specific simulation process. The coupling between the conceptual model and the implementation model is solved through the design of layering, encapsulation and interface decoupling.

To meet the simulation requirements for large-scale node deployment and mass data processing in 5G system simulation, the platform adopts distributed CPU+GPU heterogeneous parallel architecture to solve intensive computing simulation in the resource parallel architecture design. The CPU has been characterized as strong single-core processing capacity and fast computational speed, it is benefit for complex logical operations. Single-chip GPU usually has thousands of stream processor cores, its single-precision floating-point computing power is 10 times higher than CPU, which is suitable for data parallel processing brought by hundreds or even thousands of nodes accessing in 5G system. Multicore CPU

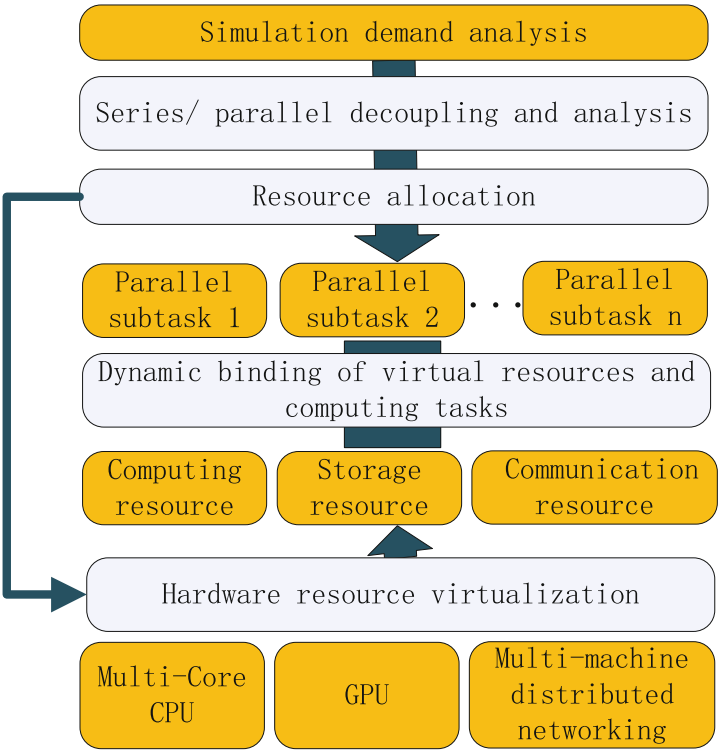


Fig. 2. Framework of the parallel simulation platform.

parallel solution can improve the computational and processing ability of simulation platform to a certain extent, but under limited resources circumstance, how to make full use of the computer capacity to achieve the maximum simulation efficiency is a key issue.

3 Classification Algorithm and Parallel Gain Analysis

3.1 Classification Algorithm

Different from the traditional CPU parallel scheme, our proposed scheme needs to decide whether to allocate CPU resources or GPU resources according to the computing characteristics of each module through machine learning. Since the performance of CPU and GPU is different, logistic regression algorithm in machine learning is considered to classify parallel computing modules. This means that an existing sample space is necessary for us to apply logistic regression algorithm to determine whether a program module should be run on CPU or GPU.

Table 1. Parameters related to execution time.

Parameters	Notations
Numerical operational complexity	x_{no}
Logical operational complexity	x_{lo}
Execution times	x_{et}

Therefore, the first step is to establish a relatively reliable data set by actually running the program, which is a heavy but necessary step. This data set is composed of two parts $\boldsymbol{\Phi} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_j, \cdots, \mathbf{x}_m]$, $\mathbf{Y} = [y_1, y_2, \cdots, y_j, \cdots, y_m]$, $\boldsymbol{\Phi}, \mathbf{Y} \in \mathbb{N}^{1\times m}$, $\mathbf{x}_j$, y_j respectively represents a program module and its corresponding classification, and m represents the number of elements in data set $\boldsymbol{\Phi}$ and $\boldsymbol{Y}$. Each element $\mathbf{x}_j$ contains parameters that affect the execution time of the program module, as detailed in Table 1. So $\mathbf{x}_j$ can be represented as $\mathbf{x}_j = [x_{no_j}, x_{lo_j}, x_{et_j}]^{\mathrm{T}}$, each $\mathbf{x}_j$ corresponds to a element y_j in $\boldsymbol{Y}$ that is either 0 or 1. When the value is 0, it means that the sample program is suitable for running on the CPU; otherwise, it is suitable for running on the GPU.

We introduce the weight vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^{\mathrm{T}}$, so the correspondence between $\boldsymbol{\Phi}$ and $\mathbf{Y}$ can be expressed as a fitting function

$$\mathbf{H} = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Phi}}}, \tag{1}$$

where $\mathbf{H} = [h_1, h_2, \cdots, h_j, \cdots, h_m]$, h_j represents the estimation result of the fitting function for the value of y_j.

To better show the error between y_j and h_j, we define a cost function

$$\begin{aligned} cost(y_j, h_j) &= \begin{cases} -ln(h_j), & y_j = 1 \\ -ln(1-h_j), & y_j = 0 \end{cases} \\ &= -y_j ln(h_j) - (1-y_j) ln(1-h_j), \end{aligned} \tag{2}$$

when y_j is equal to 1, if the prediction of h_j is also 1, then the loss is 0, and the prediction is correct. Conversely, if the prediction is 0, the loss will be infinite. Extended it to m samples to obtain the loss function

$$\boldsymbol{\mathcal{J}} = -\frac{1}{m}[\sum_{j=0}^{m}(y_j ln(h_j) + (1-y_j) ln(1-h_j))]. \tag{3}$$

the loss function is a convex function, which is convenient to get the parameter $\boldsymbol{\theta}$, so as to minimize the loss function $\boldsymbol{\mathcal{J}}$.

We want to solve the problem of minimizing $\boldsymbol{\mathcal{J}}$, and we first give an arbitrary initial value of $\boldsymbol{\theta}$, then change the value of $\boldsymbol{\theta}$ to make $\boldsymbol{\mathcal{J}}$ smaller, and keep changing the process of making $\boldsymbol{\mathcal{J}}$ smaller until $\boldsymbol{\mathcal{J}}$ is approximately equal to the minimum.

Algorithm 1. Logistic Agression algorithm steps.

Input: Data set $\boldsymbol{\Phi}$, $\boldsymbol{Y}$, weight vector $\boldsymbol{\theta}$, learning rate α, a set of parallel modules to be classified $\boldsymbol{\Omega}$.

Output: Classification results set $\boldsymbol{\Lambda}$.

1: Initialize the weight vector $\boldsymbol{\theta}$ and learning rate α.
2: Get the estimation result $\mathbf{H}$ of function (1) with $\boldsymbol{\theta}$ and $\boldsymbol{\Phi}$;
3: Use (3), (5), combining (4) to get the iteration function of $\boldsymbol{\theta}$;
4: After sufficient iterations, update weight vector $\boldsymbol{\theta}$;
5: Substitute $\boldsymbol{\Omega}$ for $\boldsymbol{\Phi}$ in the fitting function (1) to classify the parallel modules set $\boldsymbol{\Omega}$ by using updated weight vector $\boldsymbol{\theta}$;
6: **return** the set of the parallel module classification results $\boldsymbol{\Lambda}$.

The iterative formula for $\boldsymbol{\theta}$ is as follows

$$\theta_i := \theta_i - \alpha \frac{\partial \boldsymbol{\mathcal{J}}}{\partial \theta_i}, \tag{4}$$

where α in the formula is called the learning rate, which controls the magnitude of change during each iteration in the direction of decreasing $\boldsymbol{\mathcal{J}}$. The partial derivative of $\boldsymbol{\mathcal{J}}$ with respect to indicates the direction in which $\boldsymbol{\mathcal{J}}$ changes the most. Since we are dealing with a minimum, the gradient direction is the opposite direction of the partial derivative. Combining (1) and (3), take the partial derivative of $\boldsymbol{\mathcal{J}}$ with respect to θ, written as

$$\begin{aligned}
\frac{\partial \boldsymbol{\mathcal{J}}}{\partial \theta_i} &= -\frac{1}{m}\sum_{j=0}^{m}[(y_j \frac{1}{h_j} - (1-y_j)\frac{1}{1-h_j})\frac{\partial h_j}{\partial \theta_i}] \\
&= -\frac{1}{m}\sum_{j=0}^{m}[(y_j \frac{1}{h_j} - (1-y_j)\frac{1}{1-h_j})h_j(1-h_j)\frac{\partial \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}}{\partial \theta_i}] \\
&= -\frac{1}{m}\sum_{j=0}^{m}[y_j(1-h_j) - (1-y_j)h_j)x_i] \\
&= -\frac{1}{m}\sum_{j=0}^{m}(y_j - h_j)x_i,
\end{aligned} \tag{5}$$

where $i = 1, 2, 3$. Combining (4) and (5)

$$\theta_i := \theta_i - \alpha \frac{1}{m}\sum_{j=0}^{m}(h_j - y_j)x_i. \tag{6}$$

After sufficient iterations, $\boldsymbol{\theta}$ is obtained with a minimum value of $\boldsymbol{\mathcal{J}}$. Take $\boldsymbol{\theta}$ as the weight value to classify the parallel modules set $\boldsymbol{\Omega} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k, \cdots, \mathbf{m}_n]$, $\mathbf{m}_k$ can be represented as $\mathbf{m}_k = [x_{no_k}, x_{lo_k}, x_{et_k}]^{\mathrm{T}}$, and parameters in Table 1 can be obtained by checking the design of the program. A predicted classification result set $\boldsymbol{\Lambda} = [l_1, l_2, \cdots, l_k, \cdots, l_n]$ can be obtained

through the fitting function (1) by replacing $\boldsymbol{\Omega}$ for $\boldsymbol{\Phi}$. If l_k is close to 0, the $\mathbf{m}_k$ module should be run with CPU, otherwise it should be run with GPU. The proposed Logistic Aggression algorithm above is summarized in Algorithm 1.

3.2 Parallel Gain Analysis

The parallel gain can be defined as

$$G = \frac{T_S}{T_P} = \frac{T_S}{T_{pr} + T_c + T_{po}}, \tag{7}$$

where G denotes the parallel gain; T_S denotes the time of serial computing; T_P denotes the time of parallel computing; T_{pr} denotes the preparation time of parallel computing; T_c denotes the execution time of parallel computing; and T_{po} denotes the processing time after parallel computing.

Then T_{pr} can be written as

$$T_{pr} = T_{pms} + T_{ptd}, \tag{8}$$

where T_{pms} denotes the time of parallel management mechanism starting; T_{ptd} denotes the time of parallel task distribution, they are related to context of computing environment and computing tasks. The impact of the computation environment context (CPU frequency, GPU frequency, memory bandwidth, memory delay, transmission bandwidth, transmission delay) is fixed, but the computing tasks are related to computing resources required for parallelizing tasks, allocation scheme of the computing task will affect T_{ptd}

T_c is the maximum execution time for every parallel task, written as

$$T_c = max(T_{pte}(i)), i \in [1, Task_{number}], \tag{9}$$

where $T_{pte}(i)$ is the execution time of the parallel task i.

T_{po} is the time overhead required by serial computation task to obtain computation results from parallel service process, including data transfer time and memory access time. T_{po} is related to the space occupied by the parallel task results and the context of the computing environment.

By integrating (7), (8) and (9), G can written as

$$\begin{aligned} G &= \frac{T_S}{T_{pms} + T_{ptd} + max(T_{pte}(i)) + T_{po}} \\ &= \frac{1}{\frac{T_{pms}}{T_S} + \frac{T_{ptd}}{T_S} + \frac{max(T_{pte}(i))}{T_S} + \frac{T_{po}}{T_S}}. \end{aligned} \tag{10}$$

Let $T_{pms}/T_S = k_{pms}$, $T_{ptd}/T_S = k_{ptd}$, $T_{po}/T_S = k_{po}$ and $max(T_{pte}(i))/T_S = k_{pte}$, then G is written as

$$G = \frac{1}{k_{pms} + k_{ptd} + k_{pte} + k_{po}}. \tag{11}$$

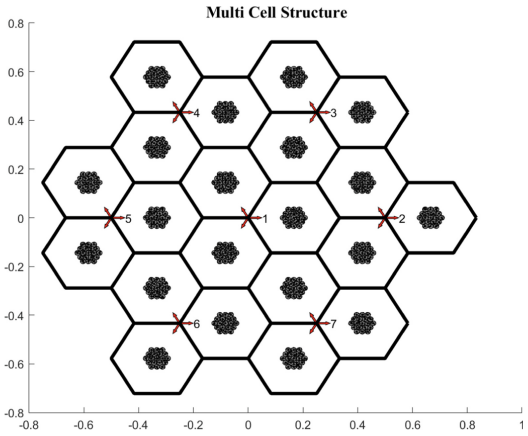


Fig. 3. eMBB simulation scenario diagram.

- *Case1:* When the amount of computation is small, more computational power is used to handle the logical operations between modules. The preparation time and result feedback time will account for a large proportion. Which means $k_{pms} \gg 0$, $k_{ptd} \gg 0$ and $k_{po} \gg 0$. When $(k_{pms} + k_{ptd} + k_{po}) < (1 - k_{pte})$, parallel computation has gains, otherwise the computing efficiency will decrease after the parallelization.
- *Case2:* When the amount of computation is large, the computational power is concentrated on the processing of data. The preparation time and result feedback time will account for a relatively small proportion of the total time. As the amount of computation goes to infinity, $k_{pms} \to 0$, $k_{ptd} \to 0$ and $k_{po} \to 0$. Then there is: $G \approx 1/k_{pte} = T_S/max(T_{pte}(i))$, which is an ideal parallel effect and the upper bound of the parallel acceleration gain.

Our platform is designed for the *Case2*, k_{pte} becomes the main factor that affects parallel gain. Since T_S is a constant, we need to consider minimizing $max(T_{pte}(i))$ for increase the parallel gain. CPU specialize in processing program tasks with complex instruction scheduling, looping, branching, logical judgment, and execution. Its parallel advantage is at the program execution level. GPU specialize in data-intensive computing tasks and illogical parallel computing, such as floating point operations and matrix operations. By designing a reasonable allocation scheme of computing resources and assigning CPUs and GPUs the computing tasks suitable for their characteristics, the maximum execution time T_c for all parallel task will be compressed as much as possible, so as to improve the overall computing efficiency theoretically.

Table 2. Simulation parameters.

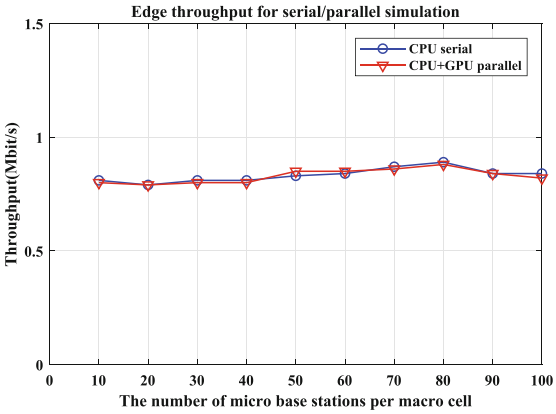
Parameters	Macro cell	Small lell
Layout	Cellular grid, 7 macro base stations, 3 sectors per macro base station	Small base stations are evenly distributed in the cellular
Bandwidth	20 MHz	
Carrier frequency	2.0 GHz	3.5 GHz
BS transmitting power	46 dBm	24 dBm
Channel profile	3D-UMx channel	
Pass loss model	128.1 + 37.6log(R) R in km	140.7+36.7log(R) R in km
Penetration loss	20 dB	
Shadow standard deviation	8 dB	10 dB
Antenna pattern	3D Directional antenna	2D Omni-directional antenna
The antenna pattern	$A_H(\varphi) = -min\left\{12(\frac{\varphi}{\varphi_{3dB}})^2, A_m\right\}$ where $A_m = 20$,$\varphi_{3dB} = 70^\circ$	$A_H(\phi) = 0$
Antenna gain	8 dBi	5 dBi
UE antenna gain	0 dBi	
Antenna configuration	BS: 1Tx, UE: 2Rx	
UE release	The user number of per sector is 20, randomly distributed	
Distance between BSs	200 m	20 m
Traffic model	Full buffer	

4 5G System-Level Simulation and Results Evaluation

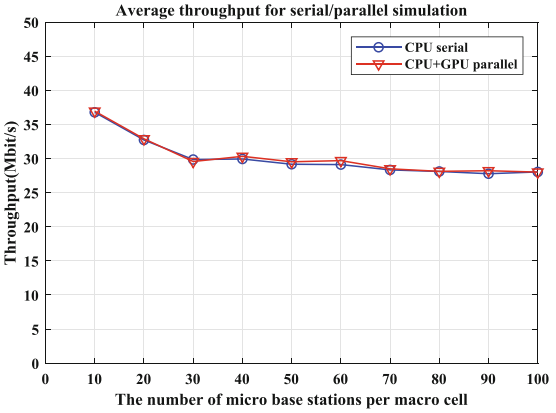
In this section, we first introduce the 5G system-level simulation program used to verify CPU-GPU heterogeneous parallel mechanism. Subsequently, the heterogeneous parallel mechanism of CPU-GPU was compared with the traditional serial mechanism from the two dimensions of simulation error and operation efficiency, proving the effectiveness of the heterogeneous parallel mechanism of CPU-GPU.

4.1 5G System-Level Simulation

The simulation scene adopts macro base station and micro base station joint deployment mode, micro base station is uniformly deployed in the coverage area of the macro base station. 5G system-level simulation deployment diagram is shown in Fig. 3. The macro base station and the micro base station are deployed at different frequencies to avoid interference between them. The output of the 5G system-level simulation program is cell edge throughput and average throughput. Major simulation parameters are list in Table 2 [22].



(a) Edge throughput for serial/parallel simulation.



(b) Average throughput for serial/parallel simulation.

Fig. 4. Throughput for serial/parallel simulation.

4.2 Results Evaluation

With the parameters provided in Table 2, simulation is carried out respectively in serial and parallel simulation mode. The serial simulation is calculated using CPU, and the parallel simulation is completed on our proposed CPU-GPU platform. Gradually increasing the number of micro base stations and observe the curve of edge throughput and average throughput. Each time the number of micro base stations was increased, the program was run three times and the average result was taken. Figure 4 shows the throughput of the system in both serial and parallel simulations. Comparing the edge throughput and average throughput of the two simulation modes, edge throughput as shown in Fig. 4(a), and the average throughput as shown in Fig. 4(b). The results of serial simulation tend to be consistent with those of parallel simulation, and the error is mainly

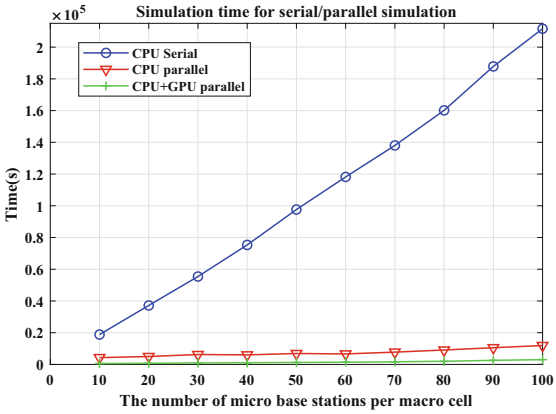


Fig. 5. Simulation time for serial/parallel simulation.

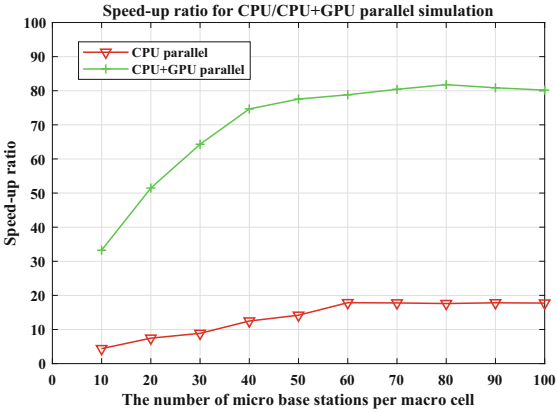


Fig. 6. Speed-up ratio for CPU/CPU+GPU parallel simulation.

related to some random processes in the program, such as noise and other factors. This demonstrates the reliability of our parallel computing platform, which is comparable to traditional simulation platforms in the accuracy of simulation results.

We also recorded the elapsed time under serial simulation and our parallel simulation, and took the average of the results of three times experiments. Meanwhile, the time consumption under the CPU parallel computing scenario is added for comparison. Simulation time is shown in Fig. 5. We can observe that under the three simulation scenarios, the curve of time duration changing with the number of base stations is almost linear. But it is clear that the gradient of the CPU serial scenario is much higher than the gradient of the other two scenarios. Our parallel simulation scenarios consumes the least amount of time.

To better understand the perform difference between our parallel simulation and CPU parallel simulation, we depict the acceleration ratio of the two scenarios in Fig. 6. The acceleration ratio here is the comparison of simulation time, and the object of comparison is the CPU serial simulation scene. Our parallel simulation platform has more obvious accelerate effect compared with CPU parallel simulation. In addition, when the number of micro base stations does not exceed 50, the acceleration ratio increases sharply, but tends to be stable after 60. In the early stage, with the increase of the number of micro base stations, the run time of serial modules that can not parallelized account decreasing proportion in the total running time. When the number of micro base stations reaches a certain scale, the proportion of run time of remained serial modules to total running time tends to be stable.

5 Conclusion

System-level simulation in 5G scenario have become significantly important for the research of new technologies in communication industries. However, large-scale system-level simulations commonly lead to high computational cost, which make efficient implementations of targeted simulation of 5G application scenario challenging tasks. To tremendously shorten the execution time, we proposed to develop the CPU-GPU based parallelization platform. In particular, the proposed platform adopts Logistic Regression algorithm to classify parallel modules for optimizing the use of computational resources. The simulation program under 5G system-level simulation is used to verify the validity of the platform. Numerical results demonstrate the effectiveness in terms of reducing execution time and guaranteeing reliability of system-level simulation results in 5G scenarios.

Acknowledgement. This work was supported in part by the Program of Shanghai Academic/Technology Research Leader (No. 21XD1433700), the Science and Technology Commission Foundation of Shanghai (No. 20DZ1101200), and the Youth Innovation Promotion Association of CAS.

References

1. 3rd Generation Partnership Project (3GPP): NR; NR and NG-RAN Overall Description; Stage-2 (Release 15), 3GPP TS 38.300 V15.12.0 (2021)
2. 3rd Generation Partnership Project (3GPP): Study on New Radio Access Technology; Radio access architecture and interfaces (Release 14), 3GPP TS 38.801 V14.0.0 (2017)
3. 3rd Generation Partnership Project (3GPP): Study on Scenarios and Requirements for Next Generation Access Technologies (Release 14), 3GPP TS 38.913 V14.3.0 (2017)
4. International Telecommunication Union (ITU): Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond, ITU-R M.2083-0 (2015)
5. Cho, S., Chae, S., Rim, M., Kang, C.G.: System level simulation for 5G cellular communication systems. In: International Conference on Ubiquitous and Future Networks 2017, Milan, Italy, pp. 1–4. IEEE (2017)
6. Hwang, I., Song, B., Soliman, S.S.: A holistic view on hyper-dense heterogeneous and small cell networks. IEEE Commun. Mag. **51**(6), 20–27 (2013)
7. Ge, X., Tu, S., Mao, G., Wang, C., Han, T.: 5G ultra-dense cellular networks. IEEE Wirel. Commun. **23**(1), 72–79 (2016)
8. Dhiyagu, D., Shanmughasundaram, R.: Dependency and utilization aware task allocation for multi-core embedded processors. In: 2019 Innovations in Power and Advanced Computing Technologies, Vellore, India, pp. 1–5. IEEE (2019)
9. Jian-zhen, C., Bin, L., Dan-ping, S.: The multi-core CPU parallel computation for CFD simulation of flowmeter. In: 2008 International Symposium on Information Science and Engineering, Shanghai, China, pp. 367–370. IEEE (2008)
10. Singh, T., Srivastava, D.K., Aggarwal, A.: A novel approach for CPU utilization on a multicore paradigm using parallel quicksort. In: Proceedings of 3rd International Conference on Computational Intelligence & Communication Technology, Ghaziabad, India, pp. 1–6. IEEE (2017)
11. Yao, W., Li, J., Tan, B., Hao, S.: Interference management scheme of ultra dense network based on clustering. In: Proceedings of IEEE 2nd Information Technology, Chengdu, China, pp. 374–377. IEEE (2017)
12. Jin, S., et al.: Understanding GPU-based lossy compression for extreme-scale cosmological simulations. In: 34th IEEE International Parallel and Distributed Processing Symposium, New Orleans, LA, USA, pp. 105–115. IEEE (2020)
13. Huang, Y., Li, Y., Zhang, Z., Liu, R.W.: GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries. IEEE Internet Things J. **7**, 10794–10812 (2020)
14. Di, Y., Weiyi, S., Ke, S., Zibo, L.: A high-speed digital signal hierarchical parallel processing architecture based on CPU-GPU platform. In: Proceedings of IEEE 17th International Conference on Communication Technology, Chengdu, China, pp. 355–358. IEEE (2017)
15. Raju, K., Chiplunkar, N.N., Rajanikanth, K.: A CPU-GPU cooperative sorting approach. In: 2019 Innovations in Power and Advanced Computing Technologies, Vellore, India, pp. 1–5. IEEE (2019)
16. Xu, T., Zhou, T., Tian, J., Sang, J., Hu, H.: Intelligent spectrum sensing: when reinforcement learning meets automatic repeat sensing in 5G communications. IEEE Wirel. Commun. **27**(1), 46–53 (2020)
17. Sang, J., Zhou, T., Xu, T., Jin, Y., Zhu, Z.: Deep learning based predictive power allocation for V2X communication. IEEE Access **9**, 72881–72893 (2021)

18. Shang, X., Hu, H., Li, X., Xu, T., Zhou, T.: Dive into deep learning based automatic modulation classification: a disentangled approach. IEEE Access **8**, 113271–113284 (2020)
19. He, D., Ai, B., Guan, K., Wang, L., Zhong, Z., Kürner, T.: The design and applications of high-performance ray-tracing simulation platform for 5G and beyond wireless communications: a tutorial. IEEE Commun. Surv. Tutor. **21**(1), 10–27 (2019)
20. Xu, T., Zhang, M., Zhou, T.: Statistical signal transmission technology: a novel perspective for 5G enabled vehicular networking. IEEE Wirel. Commun. **24**(6), 22–29 (2017)
21. Zhou, T., Xu, B., Xu, T., Hu, H., Lei, X.: User-specific link adaptation scheme for device-to-device network coding multicast. IET Commun. **9**(3), 367–374 (2015)
22. Li, K., Xu, J., Yang, Y.: System simulation modeling and key technology evaluation in 5G. ZTE Technol. J. **22**(3), 41–46 (2016)