Applied Mathematics, Modeling and Computer Simulation
C.-H. Chen et al. (Eds.)

doi:10.3233/ATDE231015

Integrated Management and Control System of Infrastructure Construction Based on Adaptive Grouping Particle Swarm Optimization Algorithm

Wei CUI [a,1], Xiao LIAO[a], Linqi KANG[b], Zhenan XU[a]
[a] *State Grid Information & Telecommunication Group Co.LTD., Beijing, China*
[b] *Beijing Yingda Chang'an Risk Management Consulting CO.LTD., Beijing, China*

Abstract. With the development of cloud computing technology, infrastructure resources are showing a trend of diversified and ubiquitous deployment. The integrated management and control system for infrastructure sites involves fixed cameras, edge-side smart devices, and cloud-based intelligent analysis centers. The important issues that need to be addressed by this system are how to coordinate resources in various aspects, achieve cloud-edge collaborative computing, minimize costs, and provide real-time warnings of hazards. This article proposes an adaptive grouping-based particle swarm optimization algorithm to address the problem of traditional particle swarm optimization easily falling into local optimal solutions. A cloud-edge multi-level resource scheduling model is designed to achieve integrated resource scheduling solutions for cloud-edge computing power. The model is applied in the comprehensive management and control system of infrastructure sites, and the results confirm that the system can achieve real-time warning and instant response for dangerous behaviors on site, while minimizing overall costs.

Keywords. cloud edge collaboration; resource scheduling ; adaptive grouping ; Particle Swarm Optimization

1. Introduction

New business and new scenarios of power informatization construction emerge in an endless stream, and the traditional IT infrastructure resources can no longer meet the business needs. With the development of mobile Internet technology, infrastructure resources show a trend of diversified and ubiquitous deployment. Various types of computing resources such as uav, intelligent terminal, edge Internet of Things agent, cloud data center have developed rapidly, and cloud computing[1], Fog calculation[2], Sea computing[3], Edge calculation[4] The concept has been put forward successively.

There are numerous safety hazards in infrastructure construction sites, and with the continuous improvement of project management and safety management requirements, there is an urgent need for an artificial intelligence-based risk diagnosis, warning, and

1 Corresponding Author, Wei CUI, State Grid Information & Telecommunication Group Co.LTD., Beijing, China; E-mail: abcui@qq.com.

decision-making analysis service. The comprehensive management system for infrastructure construction sites requires coordinated cooperation between cameras, edge IoT agents, and cloud servers. How to coordinate the resources of the cloud, edge, and end sides, and realize cloud-fog-edge collaborative computing is one of the urgent problems to be solved in this application.

2. Overview Domestic and foreign relevant technologies

Cloud computing technology refers to the physical or logical aggregation of general computing resources into a unified resource pool. Cloud computing platforms provide automated installation, deployment, management, and operation of applications, effectively reducing the difficulty of use. Enterprises can deploy applications using software and hardware resources in the cloud without high investment and management costs[6]. According to their purposes, cloud computing technologies can be divided into three layers: SaaS, PaaS, and IaaS, each with different resource allocation and scheduling strategies.

Edge computing is a new computing model that performs computation at the network edge. Edge devices have certain data calculation and analysis processing capabilities, and can offload some computing tasks to edge computing nodes to reduce the bandwidth pressure caused by massive data transmission to cloud servers. Due to the natural computational of edge devices, edge computing is only a supplement to cloud computing and cannot replace it. Fog computing and maritime computing are both types of edge computing concepts. Fog computing technology was first proposed by Cisco for IoT scenarios, introducing an intermediate layer between terminal devices and cloud data centers to provide computing, storage, and other services. Its name is fog computing because "fog is cloud closer to the ground". Its concept is similar to edge computing and can be understood as local cloud nodes serving time-sensitive applications. Maritime computing technology refers to fully utilizing the computing capabilities of various IoT devices to achieve interconnection and self-organized computing of everything. Its essence is the information interaction between things. Compared with fog computing, maritime computing is closer to end users and achieves mass device access, data collection, and preliminary intelligent analysis through intelligent algorithms.

Regardless of cloud-based resources, edge-side resources, or terminal-side resources, virtualization is currently a common solution for achieving unified scheduling of heterogeneous resources. Virtualization technology first appeared in IBM, where virtual machine controllers generate many independent virtual machine instances on top of physical hardware, and a control program is used to hide the actual physical characteristics of the computing platform, providing users with an abstract, unified, simulated computing environment. Docker technology is a lightweight virtualization solution based on containers, which uses a sandbox mechanism to allow developers to package their applications and dependencies into a container. Docker containers can be directly published to meet user requirements for platform portability, simplifying deployment time. Compared to traditional virtual machine image schemes, Docker technology has very low management overhead and can deploy hundreds of Docker containers on a single server.

Scheduling management is the study of how to satisfy all demands when there is a conflict between resource supply and demand, while meeting the timing logic relationships and resource constraints of tasks. It requires minimizing the task objective function under these constraints. Resource scheduling algorithms can be divided into single-dimensional aggregation and multi-dimensional aggregation based on different constraints.

The problem can be considered as a classical packing problem with K bins and n items to be loaded in the bins, each item i needs to occupy a unit of Vi, that is, to find a strategy that can load all items in the minimum number of bins. For resource scheduling, the containers to be allocated can be viewed as items, while the physical servers can be viewed as various boxes. Common scheduling algorithms for the one-dimensional packing problem include priority adaptive algorithms, optimal adaptive algorithms, descending-order priority adaptive algorithms, and descending-order optimal use algorithms [12].

In real-world environments, containerized applications have diverse resource requirements such as CPU, memory, I/O, network resources, and so on. Therefore, it is necessary to extend the single-dimension constraint problem into a multi-dimension constraint problem by introducing multi-dimensional resource factors. Considering the topological dependencies between data storage resources and computational resources, and network resources during task execution, efforts should be made to select computational resources that are geographically closer when choosing storage resources.

At present, the commonly used methods include exact methods and heuristic algorithms. The exact methods include dynamic programming, while the heuristic algorithms include simulated annealing algorithm, genetic algorithm, ant colony algorithm, etc. Due to the NP-hard nature of such problems, it is not possible to obtain the optimal solution using exact methods within a reasonable time when the task is complex. However, heuristic algorithms can obtain relatively good solutions in a short time, which can meet the needs of engineering [13].

To solve the problem of server aggregation with multiple constraints, Literature[14] proposes a general mathematical model for multi-dimensional resource aggregation, which minimizes the number of physical servers as the optimization goal, covering basic constraints of multi-dimensional resources as well as constraints of extended problems such as combination quantity limit and mutual exclusion. Literature[15] proposes a cost-aware cloud-edge collaborative scheduling algorithm, and literature[16] proposes an approximate Jacobi alternating direction method of multipliers for scheduling cloud-edge collaborative computing tasks.

3. Multi-level resource scheduling modeling at the cloud edge end

To achieve cloud-edge integrated multi-level resource scheduling, it is necessary to achieve a unified abstraction of heterogeneous resources, which involves abstracting intelligent terminals, drones, IoT agents, servers, terminal devices, and various levels of networks as computational resources, algorithmic resources, storage resources, network resources, and location resources. These resources are provided as infrastructure to upper-level business applications. All resources are flexibly scheduled by a unified management system, and elastic allocation of computing power is carried out to meet

business requirements while maximizing resource utilization and saving equipment maintenance costs.

3.1. Resource cost assessment

For resource scheduling, the first step is to evaluate the cost of application's performance, which is often initially unknown and requires the use of different cost models for evaluation. For CPU, it can be approximated by the time efficiency of the code; for disk IO, it can be initially estimated by analyzing the disk read and write operations of the application; for databases, it can be decomposed into various database operator combinations. To estimate the resources consumed, standard queries can be executed in advance, and the resources consumed by different operations can be collected to pre-evaluate the resources needed for application execution. As the data volume increases, the time consumed by operators will also increase accordingly. Coupled with the differences in internal logic of different applications, it is difficult to accurately estimate the resources consumed by each operation. Based on the pre-estimation of application operations, this article modifies the code of the container scheduling system and uses counters to store the resource consumption of different containers. The resource cost model is updated based on the cluster operation logs, thereby providing better pre-evaluation characteristics.

3.2. Application of the resource model

For the cloud-edge integrated environment, the resource providers can be divided into three categories: intelligent devices on the edge represented by drones and smart terminals, edge server devices, and cloud servers. Considering the computational power and power limitations of edge devices, some complex operations need to be uploaded to edge servers and cloud servers for computation, and the calculation results need to be downloaded.

This article only involves the allocation and scheduling of tasks between the cloud, edge, and end devices, and does not involve the internal resource scheduling of the cloud data center. For a typical application, it can be divided into several containers, each container performs several operations, and there is only data communication demand between containers. The container is used as the minimum unit for scheduling. Assuming that after evaluation, the demand of application i for j type of resources is demand q_{ij}, then the required five resource requests can be positioned as a vector $\langle q_{i1}...q_{i5} \rangle$. If the resources that can be provided by cloud servers, edge servers, and end devices are $\langle c_{11}...c_{15} \rangle$ $\langle c_{21}...c_{25} \rangle$ $\langle c_{31}...c_{35} \rangle$ respectively, a output matrix can be generated to represent the scheduled location of each container.

Figure 1. Cloud side-side scheduling model

3.3. Realistic constraints

Due to the different latency requirements for each task, the specific conditions of each node, wireless device energy consumption limitations, network bandwidth limitations, and time constraints for latency-sensitive tasks must be considered during task allocation. The specific constraints are:

1） Power limit

End-device devices can be divided into fixed wired devices and mobile wireless terminals. Wireless devices are limited by battery capacity and cannot operate high-performance applications for a long time. Within a maintenance cycle, the total battery consumption must be less than the available energy.

Local energy consumption per device can be calculated as

$$E_i = P_i * t_i \tag{1}$$

During the simplification process, the power consumption is assumed to be directly proportional to the actual computing power, without considering the energy consumed by memory activities.

$$P_i = k_i * f_i \tag{2}$$

The power factor of the wireless device is denoted as k_i , and the current computational frequency of the device is denoted as f_i. Furthermore, the total power constraint can be converted into a computational constraint, which is the total amount of computation performed on all wireless terminal devices.

$$\sum w_i < W_{total} \tag{3}$$

w_i The total amount of calculation required for the i-th task on the terminal is the total calculation amount that the battery can support during the cycle of the terminal equipment.

In which, w_i is the total amount of computation required by the i-th task on the terminal, and W_{total} is the total computational power that the battery can support during the cycle of the terminal device.

2） time limit

All the calculations must be done within a given time.

$$t_{total} = \max(t_t, t_e + t_{tran1}, t_c + t_{tran1} + t_{tran2}) \tag{4}$$

In which, t_t is the execution time of tasks on the terminal side, t_e is the execution time of tasks on the edge side, t_{trans1} is the transmission time from the terminal to the edge, t_{trans2} is the transmission time from the edge to the cloud, t_c is the execution time of tasks on the cloud side, and the final time must be within the specified time limit.

3） Network bandwidth limit

It is necessary to consider the bandwidth limitations of the link, and no single task should occupy all of the bandwidth. Therefore, set the bandwidth that each task can occupy to 90% of the current remaining bandwidth.

$$band_c < band_{max} * 0.9 \tag{5}$$

4） Resource side restrictions

For a specific task, the resources of a cloud computing center can be simplified as infinite, but the edge resources and end-device computing resources are limited.

$$\text{approach} \sum_{i} q_{ij} < \mathrm{c}_{nj} \tag{6}$$

In which, q_{ij} represents the demand of the i-th container for the j-th resource, and C_{nj} represents the total amount of the j-th resource provided by the n-th environment.

3.4. Final objective function

Given the constraints of minimizing overall cost, the main costs considered are the cloud server cost, the communication cost from end-devices to edge servers, the usage cost of edge servers, and the communication cost from edge servers to the cloud. As for a specific application, the choice can be to execute simple calculations on local terminals, perform certain computational tasks on edge servers, or transmit data via the network to the cloud for computation. Finally, the results are transmitted back.

$$\text{The total cost is: } Res = \sum k_{in} Cost_{i\,\mathrm{j}} \tag{7}$$

In which, k_{in} represents the total amount of the i-th resource consumed by the n-th container, and $Cost_{ij}$ represents the unit cost of the i-th resource in the j-th environment.

4. Particle swarm algorithm based on adaptive grouping

Traditional particle swarm optimization algorithms are simple to implement and have good convergence properties, and are widely used for single-objective optimization problems, but they are prone to local optima. To avoid the problem of a particle swarm algorithm becoming trapped in a local optimal solution, this article proposes a new method, the adaptive group-selection based particle swarm optimization (AGPSO), which mimics the idea of social group convergence. That is, each group of particles does not strictly converge towards the absolute optimal point, but rather classifies and aggregates towards multiple small group centers, and uses an adaptive inertial coefficient algorithm to achieve a balance between convergence speed and accuracy.

The specific process is as follows:

1. Randomly initialize all particle groups N.
2. Calculate the cost of all particles based on the total cost function.
3. Randomly initialize M cluster centers and assign particles to clusters based on their distances.
4. Perform optimal value search within a specific cluster.
5. Calculate the change in particle position based on the particle's current position and the distances to the cluster center and the absolute optimal value.
6.f convergence is achieved, stop; otherwise, repeat step 4.

The core of the algorithm is the location numbering algorithm. Due to the discrete nature of the task assignment problem, a specific computational task can only be assigned to the end, edge, or cloud center side. This article uses a probability algorithm to determine the probability of position change through particle adaptation functions.

Where the probability of change for the particle j is

$$\rho_j = (res_j - \text{res}_{\text{best}}) / res_j \tag{8}$$

$$\theta_j = (res_j - \text{res}_{\text{lbest}}) / res_j \tag{9}$$

As for each particle's matrix variables k_{in}, it is:

$$k_{in} = \begin{cases} k_{best} & \text{if } rand() < \rho_j \text{ and } rand\ 2() > \lambda \\ k_{lbest} & \text{if } rand() < \theta_j \text{ and } rand\ 2() <= \lambda \\ k_{in} & \text{else} \end{cases} \tag{10}$$

In which, local adaptability λ is:,

$$\lambda = \frac{\text{res}_{best}}{res_{lbest}} \tag{11}$$

5. Architecture of an integrated management and control system for English cloud edge collaboration infrastructure

The cloud-edge-terminal collaborative infrastructure construction site comprehensive control system follows the SG-CIM model and the unified data standard for infrastructure construction, and relies on the integrated national grid cloud for deployment and application. It mainly consists of three parts: the cloud-based intelligent analysis center, the edge-side server control components, and the end-side data acquisition components. By using algorithms such as deep learning and image recognition, it constructs identification models for dangerous behaviors such as unauthorized climbing without safety helmets and climbing in violation of regulations, and realizes real-time monitoring and intelligent handling of risks throughout the entire infrastructure construction process. The overall architecture is as follows:

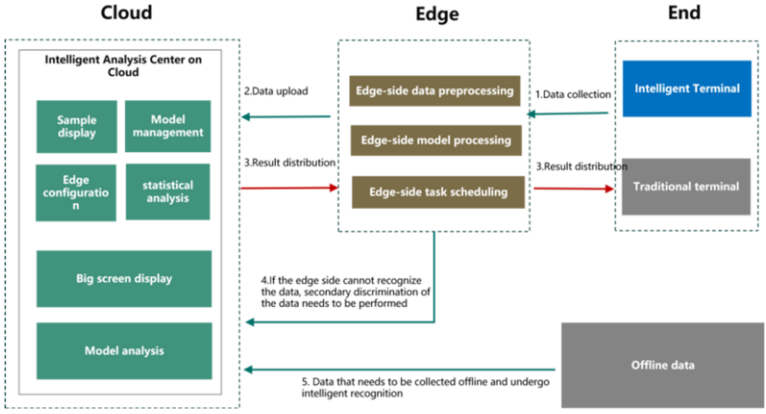

Figure 2 Application architecture diagram of cloud side-end collaborative transmission line inspection

The edge side can be mainly divided into intelligent terminals and traditional terminals. Intelligent terminals such as smart cameras have a certain data analysis ability

and can execute tasks with high real-time requirements according to business needs. If they cannot handle a task, they will transfer it to the edge side for processing. On the other hand, ordinary cameras and traditional terminals are mainly used for data collection and mainly send collected data to edge devices for processing.

On the edge side, customized edge artificial intelligence processing servers can obtain data collected by end devices and use cloud-based computational models to perform data preprocessing and some intelligent analysis tasks. Due to hardware limitations on the edge side, data that cannot be recognized by the edge side will be submitted to the cloud-based business system for secondary identification. According to the edge-side scheduling algorithm, tasks that are not suitable for edge computing will also be submitted to the cloud for analysis.

On the cloud side, the cloud intelligent analysis center is equipped with a large-scale GPU server cluster, mainly responsible for conducting data model training and managing edge-side devices and artificial intelligence analysis models. The cloud server processes the data and computational tasks uploaded from the edge side and provides data display on large screens.

6. Algorithm simulation and validation and analysis

This article uses Docker containers as the application carrier, with all applications running inside the containers. Docker technology is currently the most widely used container virtualization technology, which can package an application and its dependent runtime environment into a standard container, making it easy for the application to run on different platforms. It can solidify the runtime container into a disk image that includes all configuration information, making it easy to share across clusters, greatly improving the efficiency of application migration across clusters and reducing the cost of container migration.

The overall scheduling idea is as follows:

1. The application estimates its required resources based on historical records and standard models, and submits them to the control system.

2. During runtime, log the actual resources and time consumed by the application. If the actual consumption of resources and time of an application submitted by a user differs significantly from the declared resources, lower the priority of all applications submitted by that user.

3. Use particle swarm optimization algorithm to calculate the positions where all containers should be located.

4. Calculate the total cost of using the application.

In this article, a simulation analysis was conducted using a common PC server with 4 cores and 16GB of memory. Ten cloud-edge collaborative computing tasks were randomly generated and scheduled using four different algorithms: Cloud Computing Cluster Priority (CF), Edge Computing Cluster Priority (EF), Random Allocation (Ran), and the algorithm selected in this project (PSO). The specific simulation parameters are as follows:

Table 1. Simulation Parameters Table

order number	type	reference value
1	Power-based computing server cost	zero point two five yuan /vcpu.h

2	GPU Type Server Cost	eight yuan /cpu.h
3	Edge-side server cost	one yuan /vcpu.h
4	Terminal-to Edge Side Network Cost	zero point one yuan /GB
5	Edge-end-to-cloud network costs	zero point eight yuan /GB
6	Edge-side-to-cloud bandwidth	100Mbps
7	Terminal-to-edge-side bandwidth	100Mbps

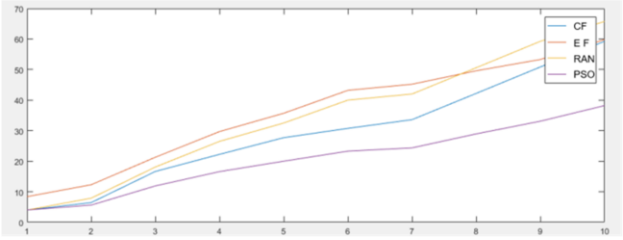

Figure 3 Total system cost simulation diagram

Simulation results demonstrate that with the continuous increase of tasks, the improved algorithm of group-based adaptive particle swarm optimization outperforms other algorithms significantly, and can effectively minimize scheduling costs.

7. Conclusion

This article applies the adaptive grouped particle swarm optimization algorithm to achieve global scheduling and planning of cloud and edge computing capabilities, and designs a cloud-edge collaborative integrated management system for infrastructure sites. The results indicate that compared with other scheduling strategies, this algorithm can significantly reduce resource usage costs and effectively support real-time fault detection in infrastructure sites.

References

[1] Sadeeq M M, Abdulkareem N M, Zeebaree S R M, et al. IoT and Cloud computing issues, challenges and opportunities: A review[J]. Qubahan Academic Journal, 2021, 1(2): 1-7.
[2] Costa B, Bachiega Jr J, de Carvalho L R, et al. Orchestration in fog computing: A comprehensive survey[J]. ACM Computing Surveys (CSUR), 2022, 55(2): 1-34.
[3] Laghari A A, Wu K, Laghari R A, et al. A review and state of art of Internet of Things (IoT)[J]. Archives of Computational Methods in Engineering, 2021: 1-19.
[4] Mansouri Y, Babar M A. A review of edge computing: Features and resource virtualization[J]. Journal of Parallel and Distributed Computing, 2021, 150: 155-183.
[5] Huang Zhen. The Power Digital Space and The New Power System [M]. Beijing, China Electric Power Press.2022:8
[6] Sandhu A K. Big data with cloud computing: Discussions and challenges[J]. Big Data Mining and Analytics, 2021, 5(1): 32-40.
[7] Cao K, Liu Y, Meng G, et al. An overview on edge computing research[J]. IEEE access, 2020, 8: 85714-85728.
[8] JIA Wei-jia, ZHOU Xiao-jie.Concepts, issues, and applications of fog computing[J].Journal on Communications,2018,39, 153–165.

[9] Yang Y. Multi-tier computing networks for intelligent IoT[J]. Nature Electronics, 2019, 2(1): 4-5..
[10] Pham X Q, Nguyen T D, Huynh-The T, et al. Distributed Cloud Computing: Architecture, Enabling Technologies, and Open Challenges[J]. IEEE Consumer Electronics Magazine, 2022.
[11] BReis D, Piedade B, Correia F F, et al. Developing docker and docker-compose specifications: A developers' survey[J]. Ieee Access, 2021, 10: 2318-2329.
[12] Afrin M, Jin J, Rahman A, et al. Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2021, 23(2): 842-870..
[13] XU Wen-zhong, PENG Zhi-ping, ZUO Jing-long.Research on Cloud Computing Resource Scheduling Strategy Based on Genetic Algorithm, Computer Measurement & Control, 2015, 23(5): 1653-1656.
[14] JIANG Hua, ZHNAG Le-qian, WANG Xin.Cloud Computing Resource Scheduling Strategy Based on Multidimensional Evaluation Model and Improved Ant Colony Algorithm[J].Computer Measurement & Control, 2015, 23(7): 2559-2562.
[15] LIU Ze-ning, LI Kai, WU Lian-tao, et al.CATS: Cost Aware Task Scheduling in Muti-tier Computing Networks[J].Journal of Computer Research and Development, 2020, 57(9): 1810–1822.
[16] MA Lu, LIU Ming, LI Chao, et al.Wei-jia, ZHOU Xiao-jie.A Cloud-Edge Collaborative Computing Task Scheduling Algorithm for 6G Edge Networks[J].Journal of Beijing University of Posts and Telecommunications, 2020, 43(06): 66-73.