2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application

An Energy-Efficient Task Scheduling Method for CPU-GPU Heterogeneous Cloud

Hui Zhao[1,2], Jianhua Li[1], Guobin Zhang[1], Shangshu Li[1], Jing Wang[1*]
[1]*Xidian University, China*
[2]*Key Laboratory of Smart Human-Computer Interaction and Wearable Technology of Shaanxi Province, China*
{hzhao,wangjing}@mail.xidian.edu.cn

***Abstract*—Given the problems of high energy consumption and unreasonable resource utilization in current heterogeneous cloud computing platforms, this paper proposes an energy-efficient task scheduling strategy. First, by analyzing the relationship between the energy consumption of physical machines and the utilization rate of CPU-GPU resources, we can establish the task scheduling model and energy consumption model. Second, a task scheduling method HCGTS (Heterogeneous CPU-GPU Task Scheduling) for energy consumption optimization of CPU-GPU heterogeneous cloud platform is proposed. Different from traditional subtasks, we divide the multi-task data flows into CPU tasks, GPU tasks and pending tasks, which are scheduled and executed by different processors to implement multi-task parallel scheduling. In addition, the execution order of subtasks is dynamically adjusted to realize the load balance of CPU and GPU, thereby reducing the task completion time and overall energy consumption. Finally, the simulation experiment is completed on the cloud simulation software. Through comparing with the other three algorithms, the HCGTS algorithm has better performance in terms of execution time, resource utilization and energy consumption optimization.**

***Index Terms*—Heterogeneous cloud, task scheduling, energy-efficient, resource utilization, load balancing**

I. Introduction

With the development and maturity of applications such as artificial intelligence and deep learning, traditional cloud platforms need to further improve their computing capabilities. Therefore, GPU processors and related technologies known as high-density parallel computing units are introduced into cloud platforms. The current computing architectures are mostly hybrid systems in which parallel GPUs and multi-core CPUs work together [1], [2]. However, due to the complexity of the heterogeneous cloud platform structure, unreasonable task scheduling will lead to unbalanced system load and lower execution efficiency, increasing the total execution time of tasks and generating additional energy consumption. The research on energy consumption optimization of cloud platform has become a research hotspot. There have been many researches to solve the problem of high energy consumption of cloud platforms. For example, Tarafdar et al. [3] proposed an independent, deadline-sensitive task scheduling method ACOEM in a heterogeneous cloud environment, which combined heuristic search and positive information feedback through Ant Colony Optimization (ACO). But they didn't consider the dependencies between tasks. The Hybrid Scheduling Algorithm (HS) proposed by Walia et al. [4] was an energy-efficient task scheduling algorithm based on the Genetic Algorithm (GA) and the Flower Pollination Algorithm (FPA), which can improve performance in heterogeneous environments while meeting user needs, but it was insufficient in load balancing and fair scheduling. Ghanavati et al. [5] proposed and evaluated a task scheduling algorithm, which found an optimal trade-off between the system makespan and the consumed energy required, so as to realize the aim of reducing the total system makespan and energy consumption.

However, existing studies in task scheduling do not consider the energy consumption optimization problem in heterogeneous environment. Moreover, researches mentioned above do not consider the impact of load balancing and the optimal working state of CPU-GPU on energy consumption optimization. And this paper proposes an energy-optimized task scheduling strategy for CPU-GPU heterogeneous cloud platforms.

The key contributions of this paper can be summarized as follows:

- A task scheduling model and energy consumption model for CPU-GPU heterogeneous cloud platform are proposed. By calculating the optimal utilization of CPUs and GPUs in PMs and taking task deadline and optimal CPU-GPU resource utilization as constraints, this paper establishes a heterogeneous cloud-oriented task scheduling and energy consumption model to minimize the energy consumption.
- A multi-task scheduling algorithm HCGTS for CPU-GPU heterogeneous cloud is proposed. The multi-task data flow is divided according to the task type, and the load balance among the PMs is realized through reasonable task allocation, so the energy consumption of the heterogeneous cloud platform is reduced. Besides, by dynamically scheduling pending tasks, both the CPUs and GPUs of the PMs can work in the best state, further reducing the energy consumption of the PM.
- The simulation experiment is completed on the cloud simulation software, and the HCGTS proposed in

979-8-3503-1993-4/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00199

this paper is compared with the task scheduling algorithms such as Min-Min, Max-Min, and PSO under multiple indicators. The experimental results prove the superiority of the HCGTS in terms of energy consumption, execution efficiency, and load balancing.

The outline of the paper is organized as follows. In Section II, task scheduling model and energy consumption model are given. Then, the multi-task scheduling algorithm HCGTS is described in Section III. The experiments that verify energy consumption, execution efficiency, and load balancing are provided in Section IV. And the conclusions and future direction are given in Section V.

II. Modeling and Formulation

A. Task Scheduling Model

Reasonable task scheduling in heterogeneous cloud platforms is crucial to optimize the indicators of the system.

A multi-task set can be divided into multiple subtasks, and each subtask is represented by $\theta_m = \{CR_m^c(\text{mips}), CR_m^g(\text{mips}), RD(m)\}$; the symbol $CR_m^c(\text{mips})$ refers to the actual processing power required by the subtask on the CPU, $CR_m^g(\text{mips})$ represents the actual processing power required by the subtask on the GPU, the symbol $RD(m)$ is used to represent the deadline for subtask.

The CPU part execution time of the m-th subtask θ_m assigned to the virtual machine $VM_{i,j}$ is defined as:

$$T_{i,j}^c(k,m) = \frac{CR_m^c(\text{ mips })}{VM_{i,j}^c(\text{ mips })} \tag{1}$$

where k is the predecessor node of the m-th subtask, the symbol $VM_{i,j}$ indicates that the j-th virtual machine is deployed on the i-th PM PM_i.

Similarly, the GPU part execution time of the m-th subtask θ_m assigned to the virtual machine $VM_{i,j}$ is defined as:

$$T_{i,j}^g(k,m) = \frac{CR_m^g(\text{ mips })}{VM_{i,j}^g(\text{ mips })} \tag{2}$$

For a virtual machine, since the CPU and GPU can execute in parallel, the execution time of the virtual machine depends on the maximum completion time of the CPU and GPU. Therefore, the execution time of the m-th subtask is defined as:

$$T_{i,j}(k,m) = \text{MAX}\left\{T_{i,j}^c(k,m), T_{i,j}^g(k,m)\right\} \tag{3}$$

Each subtask θ_m has an earliest occurrence time $FD_{i,j}(m)$. If the subtask is the first task assigned to the virtual machine $VM_{i,j}$, it has no predecessor node task, so $m = 1, k = 0$, and then its earliest occurrence time is 0; if the subtask is not the first task assigned to the virtual machine $VM_{i,j}$, then there is a predecessor node, and the resulting relationship is given by:

$$\begin{aligned} FD_{i,j}(m) =& MAX\left\{FD_{i,j}(k) + T_{i,j}(k,m)\right\}, \\ & m > 1, k \in \varphi \end{aligned} \tag{4}$$

Each subtask θ_m also has a latest completion time $LD_{i,j}(m)$. For example, for the m-th subtask, its latest completion time is the maximum value of $\{q^1, q^2, ..., q^r\}$, where q^r denotes the r-th successor task of the m-th subtask.

$$LD_{i,j}(m) = MAX\left\{FD_{i,j}\left(q^x\right) | x = 1, 2, ..., r\right\} \tag{5}$$

At the same time, the latest completion time of each subtask should be completed within its deadline, which is:

$$LD_{i,j}(m) \leq RD(m) \tag{6}$$

B. Energy consumption model

The CPU and GPU resource utilization of the PM changes dynamically, and its CPU and GPU resource utilization is defined as:

$$PM_i^{cu} = \frac{\sum \sigma_i VM_{i,j}^c(\text{mips})}{PM_i^c(\text{mips})} \tag{7}$$

$$PM_i^{gu} = \frac{\sum \sigma_i VM_{i,j}^g(\text{mips})}{PM_i^g(\text{mips})} \tag{8}$$

where the symbol σ_i refers to a collection of virtual machines on a PM PM_i, $PM_i^c(\text{mips})$ represents the CPU processing power of the PM while $PM_i^g(\text{mips})$ represents the GPU processing power of the PM, PM_i^{cu} indicates the CPU resource utilization of PM and PM_i^{gu} indicates the GPU resource utilization of PM, $VM_{i,j}^c(\text{mips})$ and $VM_{i,j}^g(\text{mips})$ indicate the CPU and GPU processing capabilities of the virtual machine $VM_{i,j}$, respectively.

By analyzing the relationship between PM resource utilization and energy consumption, it can be simplified as:

$$C = \alpha + \beta \log_2(1 + x) \tag{9}$$

where C represents the energy consumption of the PM, α and β are constants and both are greater than 0, x is the resource utilization of the CPU and GPU and ranges from 0 to 1.

Substituting the parameters into formula (9) can obtain the energy consumption of the CPU and GPU of the current PM when they are working:

$$\begin{aligned} PM_i^{c-work} =& PM_i^{c-idle} + [PM_i^{c-max} - PM_i^{c-idle}] \\ & \times \log_2(1 + PM_i^{cu}) \end{aligned} \tag{10}$$

$$\begin{aligned} PM_i^{g-work} =& PM_i^{g-idle} + [PM_i^{g-max} - PM_i^{g-idle}] \\ & \times \log_2(1 + PM_i^{gu}) \end{aligned} \tag{11}$$

where PM_i^{c-work} indicates the energy consumption of the CPU of the PM when it is working, PM_i^{c-idle} denotes the energy consumption when the CPU is idle in

the PM, PM_i^{c-max} refers to the maximum CPU power consumption when the PM is working at 100%.

The energy consumption of each PM PM_i during work is generated from the earliest execution time $FD_{i,j}$ of the task to the latest completion time $LD_{i,j}$, and the total energy consumption of its CPU and GPU during work can be defined as:

$$W_i^c = \int_{FD_i^c}^{LD_i^c} PM_i^{c-work} = PM_i^{c-work} \times (LD_i^c - FD_i^c) \tag{12}$$

$$W_i^g = \int_{FD_i^g}^{LD_i^g} PM_i^{g-work} = PM_i^{g-work} \times (LD_i^g - FD_i^g) \tag{13}$$

where W_i^c and W_i^g respectively represent the energy consumption of the CPU and GPU working in the i-th PM PM_i.

Then the total energy consumption of PMs in the heterogeneous cloud platform is defined as:

$$W_i = W_i^c + W_i^g + W_i^{\text{other}} \tag{14}$$

where W_i is the total energy consumption of the i-th PM in the heterogeneous cloud platform $(1 <= i <= M)$, W_i^c and W_i^g respectively represent energy consumption when working for the CPU and GPU in this PM, W_i^{other} denotes energy consumption for other functional units in the PM (including application, release of virtual machines, task transfer, etc.). Therefore, the total energy consumption of the PM can be given by:

$$W = \sum_{\vartheta} W_i \tag{15}$$

Combining the above equations and taking minimizing energy consumption of heterogeneous cloud platforms as the optimization goal, it can be defined as follows:

$$\text{MIN } (W) \tag{16}$$

$$\text{s.t.} \begin{cases} \frac{\partial W_i^c}{\partial PM_i^{cu}} = 0 \\ \frac{\partial W_i^g}{\partial PM_i^{gu}} = 0 \end{cases} \tag{17}$$

III. Task scheduling algorithm

A. Division of multi-task data flow

Existing studies have shown that various subtasks have corresponding parallel characteristics [12]. In the task scheduling process, the subtasks are divided into stateful and stateless according to whether the subtasks have dependencies on scheduling. The execution of stateful subtasks depends on the result of the previous operation, and the parallelism is low; while stateless subtasks have no dependencies on the previous and subsequent scheduling, so that they can run concurrently on the GPU. The degree of parallelism of subtasks can be obtained by calculating the ratio of the current PM resources to the resources required by the subtasks. The higher the degree of parallelism, the more favorable it is to exert the parallel computing power of the GPU.

The tasks are divided into three sets according to whether they are stateful and the degree of parallelism, including CPU task set, pending task set and GPU task set. Among them, the set of pending tasks is dynamically adjusted according to the resource efficiency between the CPU and the GPU to realize load balanced.

At first, by traversing each subtask and counting the parallelism of them, all subtasks can be sorted in ascending order. If the subtasks have the same degree of parallelism, they are sorted according to their computational load. The greater the amount of computation, the larger the weighted proportion of parallelism obtained by the subtask. After sorting, the stateful subtasks can be directly added to the CPU set, and the stateless subtasks that reach the maximum parallelism value can be directly added to the GPU set; repeat the above process until all subtasks are allocated. The remaining subtasks are kept in the pending area.

B. Load Adjustment for Multitasking Data Streams

In order to realize the load adjustment of multi-task data flow, it is necessary to establish an estimation model, which can obtain the overall workload of the CPU and GPU, and then dynamically adjust the workload in the system to achieve load balancing.

$$WK_i^c = \sum_{\delta_i} CR_m^c(\text{mips}) \times P_m \tag{18}$$

where δ_i represents the set of CPU demand processing power on all PMs PM_i, P_m indicates the degree of parallelism of subtasks θ_m, WK_i^c indicates the workload of all CPUs in the i-th PM.

$$WK_i^g = \sum_{\tau_i} CR_m^g(\text{mips}) \times s \times \frac{P_m + r}{r} \tag{19}$$

where τ_i represents the set of GPU demand processing power on all PMs PM_i, r is the number of cores in a single GPU, s indicates the single-core performance ratio between CPU and GPU, which is used as a reference to estimate the load on different devices. WK_i^g indicates the workload of all GPUs in the i-th PM.

After areas are allocated for the multi-tasking scheduling module, there is an adjustable area. The nodes in this area are regarded as candidate nodes for reasonable scheduling, so that the load of computing units such as CPU and GPU in the system is balanced as much as possible, which can improve the operation efficiency and reduce the execution time of the cloud platform system.

The HCGTS algorithm is divided into two sub-algorithms for execution. In Algorithm 1, the task requirements and the unscheduled task set are input, and the task scheduling result of each PM is output, and then it is used as the input of Algorithm 2 to obtain the total energy consumption generated by the PM.

Algorithm 1 HCGTS-1

Input: Actual needs of tasks θ_m, unmapped task set $\varphi = \{\theta_1, \theta_2, \ldots, \theta_m\}$
Output: Scheduling results of each PM ∂_i

Initialization;
while Not Empty(φ) **do**
  for each job θ_m **do**
    Calculate CPU and GPU partial execution time $T^c_{i,j}(k,m)$ and $T^g_{i,j}(k,m)$;
    Set the actual execution time according to (3) and calculate $FD_{i,j}(m)$, $LD_{i,j}(m)$ according to (4), (5);
    if $LD_{i,j}(m) \leq RD(m)$ **then**
      Task θ_m is assigned to $VM_{i,j}$;
    else
      Change for another $VM_{i,j}$;
    end if
  end for
end while
Calculate PM_i^{cu}, PM_i^{gu}, FD_i, LD_i;
return best scheduling result of each PM ∂_i

Algorithm 2 HCGTS-2

Input: Scheduling results of each PM ∂_i, result of M PMs set ∂
Output: Minimum total energy consumption W

Initialization;
while Not Empty(∂) **do**
  for each PM PM_i **do**
    Calculate the energy consumption PM_i^{c-work} and PM_i^{g-work} of the CPU and GPU of the current PM when it is working;
    Calculate the energy consumption of the CPU and GPU working in the i-th PM PM_i according to (12) and (13);
    $W_i^c = PM_i^{c-work} \times \frac{\sum_{\delta_i} CR_m^c(\text{ mips })}{PM_i^{cu} \times PM_i^c(\text{ mips })}$;
    $W_i^g = PM_i^{g-work} \times \frac{\sum_{\tau_i} CR_m^g(\text{ mips })}{PM_i^{gu} \times PM_i^g(\text{ mips })}$;
    Then the total energy consumption of PMs in the heterogeneous cloud platform is given in (14);
  end for
end while
Adjust the frequency according to the results: PM_i^{cu}, PM_i^{gu};
Energy consumption is accumulated by (15) to (17);
return Minimum total energy consumption W

TABLE I
CONFIGURATION DESCRIPTION OF TWO DIFFERENT TYPES OF PMS

Server	Frequency (GHz)	Number of cores	Memory (GB)	Storage
IBM	2.96	6	10	20000
HP	2.86	4	10	20000

TABLE II
VGPU TYPES SUPPORTED BY GRID K1

Graphics card	Types of vGPU	Memory (MB)	Number of vGPU
NVIDIA GRID K1	$GRIDK120Q$	512	32
	$GRIDK160Q$	204	8
	$GRIDK180Q$	4096	4

IV. PERFORMANCE EVALUATION

A. Experimental environment

All the experiments involved in this paper are performed on cloud simulation software, and the research focuses on the energy consumption optimization of heterogeneous cloud platforms. The experiment quantifies the energy consumption generated by the server according to the resource utilization of CPU and GPU. The energy consumption value under different resource utilization uses the real energy consumption data provided by the standard performance evaluation company SPEC. The PM configurations of the two different types of servers are shown in Table I.

In the experimental design of this paper, PMs under different load states are used, one PM is in an idle state, and the other PM is in a working state, which can prove the effectiveness of the algorithm proposed in this paper. It can be seen from the GPUCloudSim simulation process that a data center needs to be created, and the data center contains tasks to be executed that should include GPU-type tasks. Besides, the configured simulation environment should support the simulation of GPUs. The graphics card used in this experiment is NVIDIA GRID K1, and the maximum power consumption is 130W. Table II lists the vGPU types and related parameter information supported by this graphics card.

B. Analysis of results

In order to test the optimization effect of the task scheduling algorithm proposed in this paper on the cloud platform, the experiment is divided into two parts. The first part verifies the relationship between the number of tasks and energy consumption, so as to reduce the extra energy consumption caused by the idle time; the second part verifies the optimization effect of the HCGTS on task execution time, resource utilization and energy consumption.

In the first part of the experiment, in order to verify the relationship between the number of tasks and the energy consumption of the heterogeneous cloud platform, a control experiment with the number of tasks sufficient to make the PM run at full load is set up.

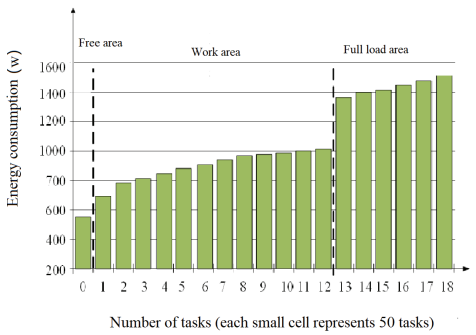

Fig. 1. The energy consumption diagram of the three states of the PM

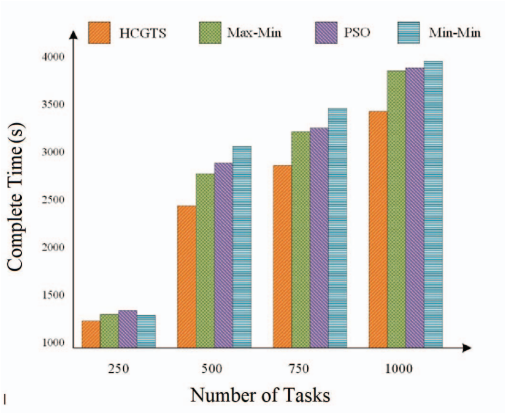

Fig. 2. Schematic diagram of task completion time (the first case)

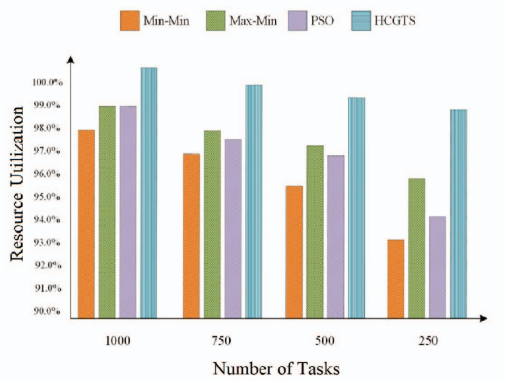

Fig. 3. Schematic diagram of resource utilization (the first case)

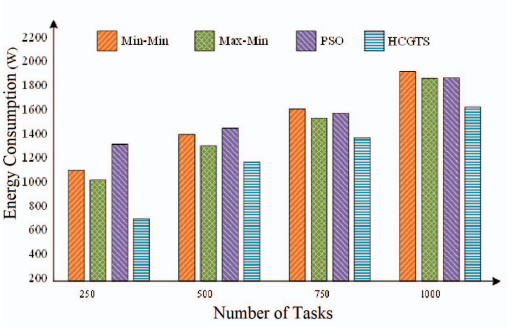

Fig. 4. Schematic diagram of energy consumption (the first case)

As shown in Fig. 1, when there is no task scheduling, the PM generates a lot of idle energy consumption. At this time, although the PM has no tasks to execute, it still needs to perform data communication and cooling down, and the idle energy consumption accounts for 35.1% of the energy consumption in the overall load state. Therefore, a reasonable task scheduling algorithm can reduce the unnecessary energy consumption of the computer due to idle waiting.

In the second part of the experiment, in order to verify the relationship among the number of tasks, resource utilization and energy consumption, ensure the authenticity of the experimental results, the number of tasks set in the experiment is from 250 to 1000, incremented by 250 each time. The HCGTS is compared with the scheduling algorithms such as Max-Min, Min-Min and PSO. By comparing the resource utilization and the energy consumption under different task numbers, the superiority of the algorithm proposed in this paper is verified.

1) The number of CPU-type tasks is less than the number of GPU-type tasks: The experimental results are shown in Fig. 2, Fig. 3, and Fig. 4. It can be seen that the HCGTS scheduling algorithm proposed in this paper has the shortest task completion time, the highest resource utilization level, and the lowest energy consumption. The Max-Min and PSO algorithms are next, and the Min-Min algorithm has the worst performance.

2) The number of CPU-type tasks is more than the number of GPU-type tasks: Fig. 5, Fig. 6, and Fig. 7 show the schematic diagrams of the experimental results in which the number of CPU-type tasks is more than the number of GPU-type tasks. In this experiment, as the picture shows, the HCGTS scheduling algorithm performs better than the other three algorithms in terms of energy consumption, resource utilization and task completion time, followed by the Min-Min algorithm, and the worst performance is the Max-Min algorithm.

V. Conclusions and Future Work

This paper firstly classifies, researches and summarizes the current scheduling algorithms, and then takes this as a starting point to study the relationship between energy consumption and the resource utilization of CPU-GPU heterogeneous cloud, so as to establish an energy consumption model. Secondly, taking the subtask deadline as the constraint, a task scheduling model is established, and a multi-task scheduling algorithm is designed. Finally, a simulation experiment is carried out on the cloud simulation software, and the experimental results show that the

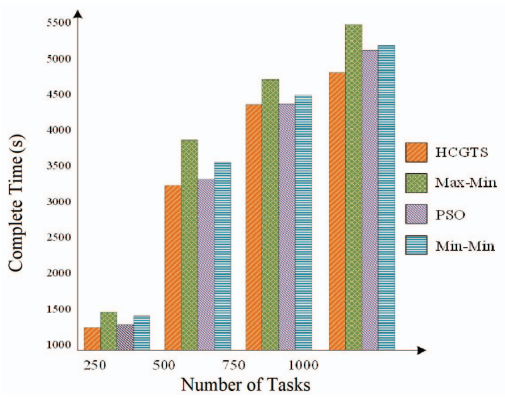

Fig. 5. Schematic diagram of task completion time (the second case)

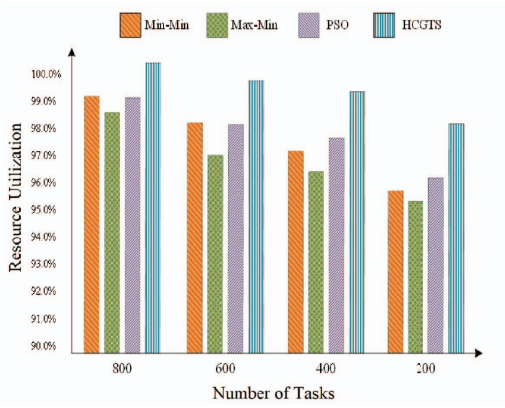

Fig. 6. Schematic diagram of resource utilization (the second case)

HCGTS scheduling algorithm has better performance in terms of execution time, resource utilization and energy consumption optimization.

In this paper, the deadlines and other information of tasks are known in advance, but in the current practical application, we need to face some unexpected situations, and we need to consider tasks such as preemption and fault handling. The research on complex scheduling in heterogeneous cloud environments will be further explored in future work.

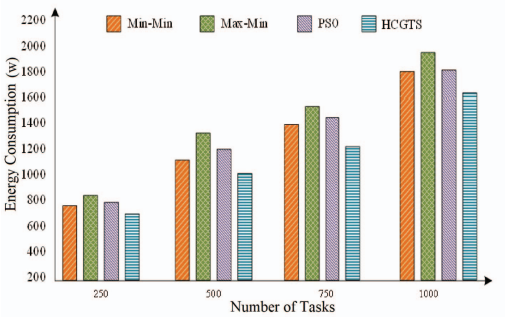

Fig. 7. Schematic diagram of energy consumption (the second case)

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for the Central Universities (JB210312, JB210309), the Key Research and Development Program of Shaanxi (2021GY-014, 2021GY-086) and the National Natural Science Foundation of China (61972302).

REFERENCES

[1] L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, "Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems," IEEE Access, vol. 8, 2020, pp. 205099-205110.

[2] R. Montella, D. Di Luccio, C. G. De Vita, G. Mellone, M. Lapegna, G. Laccetti, S. Kosta, and G. Giunta, "Enabling the cuda unified memory model in edge, cloud and hpc offloaded gpu kernels," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2022, pp. 834-841.

[3] A. Tarafdar, M. Debnath, S. Khatua, and R. K. Das, "Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment," Journal of Grid Computing, vol. 19, no. 2, 2021, pp. 1-25.

[4] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, and H. Kaur, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments," IEEE Access, vol. 9, 2021, pp. 117325-117337.

[5] S. Ghanavati, J. H. Abawajy, and D. Izadi, "An energy aware task scheduling model using ant-mating optimization in fog computing environment," IEEE Transactions on Services Computing, 2020, pp. 1.

[6] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is np-hard," 2008 Real-Time Systems Symposium, 2008, pp. 397-406.

[7] H. Zhao, G. Qi, Q. Wang, J. Wang, P. Yang, and L. Qiao, "Energy-efficient task scheduling for heterogeneous cloud computing systems," 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019, pp. 952-959.

[8] H. Zhao, F. Meng, N. Feng, Q. Wang, B. Wan, and J. Wang, "An energy-efficient task scheduling strategy based on improved fireworks algorithm in heterogeneous cloud," 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), 2021, pp. 197-204.

[9] K. Pradeep, A. L. Javid, N. Gobalakrishnan, C. J. Raman, and N. Manikandan, "Cwoa: Hybrid approach for task scheduling in cloud environment," The Computer Journal, no. 7, 2021, p. 7.

[10] H. Zhang, X. Geng, and H. Ma, "Learning-driven interference-aware workload parallelization for streaming applications in heterogeneous cluster," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, 2020, pp. 1-15.

[11] Y. Alahmad, T. Daradkeh, and A. Agarwal, "Proactive failure-aware task scheduling framework for cloud computing," IEEE Access, vol. 9, 2021, pp. 106152-106168.

[12] S. A. Lohi, N. Tiwari, V. Namdeo, and S. A. Lohi, "Analysis and review of effectiveness of metaheuristics in task scheduling process with delineating machine learning as suitable alternative," 2020 International Conference on Innovative Trends in Information Technology (ICITIIT), 2020, pp. 1-6.