A Survey on Task Scheduling of CPU-GPU Heterogeneous Cluster

ZHOU Yiheng[1], ZENG Wei[2], ZHENG Qingfang[3,4], LIU Zhilong[3,4], CHEN Jianping[2]
(1. University of Shanghai for Science and Technology, Shanghai 200093, China;
2. National Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University, Beijing 100871, China;
3. State Key Laboratory of Mobile Network and Mobile Multimedia Technology, Shenzhen 518055, China;
4. ZTE Corporation, Shenzhen 518057, China)

DOI: 10.12142/ZTECOM.202403010
https://kns.cnki.net/kcms/detail/34.1294.TN.20240724.1705.003.html, published online July 29, 2024
Manuscript received: 2024-04-07

Abstract: This paper reviews task scheduling frameworks, methods, and evaluation metrics of central processing unit-graphics processing unit (CPU-GPU) heterogeneous clusters. Task scheduling of CPU-GPU heterogeneous clusters can be carried out on the system level, node-level, and device level. Most task-scheduling technologies are heuristic based on the experts' experience, while some technologies are based on statistic methods using machine learning, deep learning, or reinforcement learning. Many metrics have been adopted to evaluate and compare different task scheduling technologies that try to optimize different goals of task scheduling. Although statistic task scheduling has reached fewer research achievements than heuristic task scheduling, the statistic task scheduling still has significant research potential.

Keywords: CPU-GPU heterogeneous cluster; task scheduling; heuristic task scheduling; statistic task scheduling; parallelization

Citation (Format 1): ZHOU Y H, ZENG W, ZHENG Q F, et al. A survey on task scheduling of CPU-GPU heterogeneous cluster [J]. ZTE Communications, 2024, 22(3): 83–90. DOI: 10.12142/ZTECOM.202403010
Citation (Format 2): Y. H. Zhou, W. Zeng, Q. F. Zheng, et al., "A survey on task scheduling of CPU-GPU heterogeneous cluster," *ZTE Communications*, vol. 22, no. 3, pp. 83–90, Sept. 2024. doi: 10.12142/ZTECOM.202403010.

1 Introduction

Cloud computing platforms have become fundamental information infrastructures in modern society[1]. A large number of computing servers in one cluster are connected by high-speed communication networks and provide high concurrency for users' remote accesses. central processing unit (CPU) and graphics processing unit (GPU) servers are dominantly used as core computing resources and virtualized into different computing resource pools to cater to various services. Computing tasks are calculated in the servers and provide various services at the Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) levels[2]. Tasks running on suitable servers can save cost and finish computing as soon as possible. Such task assignment is an important technology called task scheduling[3]. Bad task scheduling will waste computing resources, cost more, and lead to a bad user experience[4].

CPU servers are generally dominant in cloud platforms due to their universal computing architecture. However, with the rapid development of artificial intelligence (AI) technologies, the GPU has been more and more widely used in heterogeneous clusters. Cluster heterogeneity is heterogeneous among computing servers, and between CPU and GPU within a server. In most cases, one cluster may have both the CPU server and the GPU server simultaneously. Consequently, task scheduling in CPU-GPU heterogeneous clusters is more complex and different[5].

Good task scheduling should have the following characteristics:

• Maximizing clusters' system goals, e.g. throughput, energy cost, Quality of Service (QoS), etc;

• Balancing networks and storage with computing, and preventing network jams, long-time idleness, hot end, and so on;

• Assigning tasks to the best-matched CPUs or GPUs as far as possible.

2 Framework of Task Scheduling for Clusters

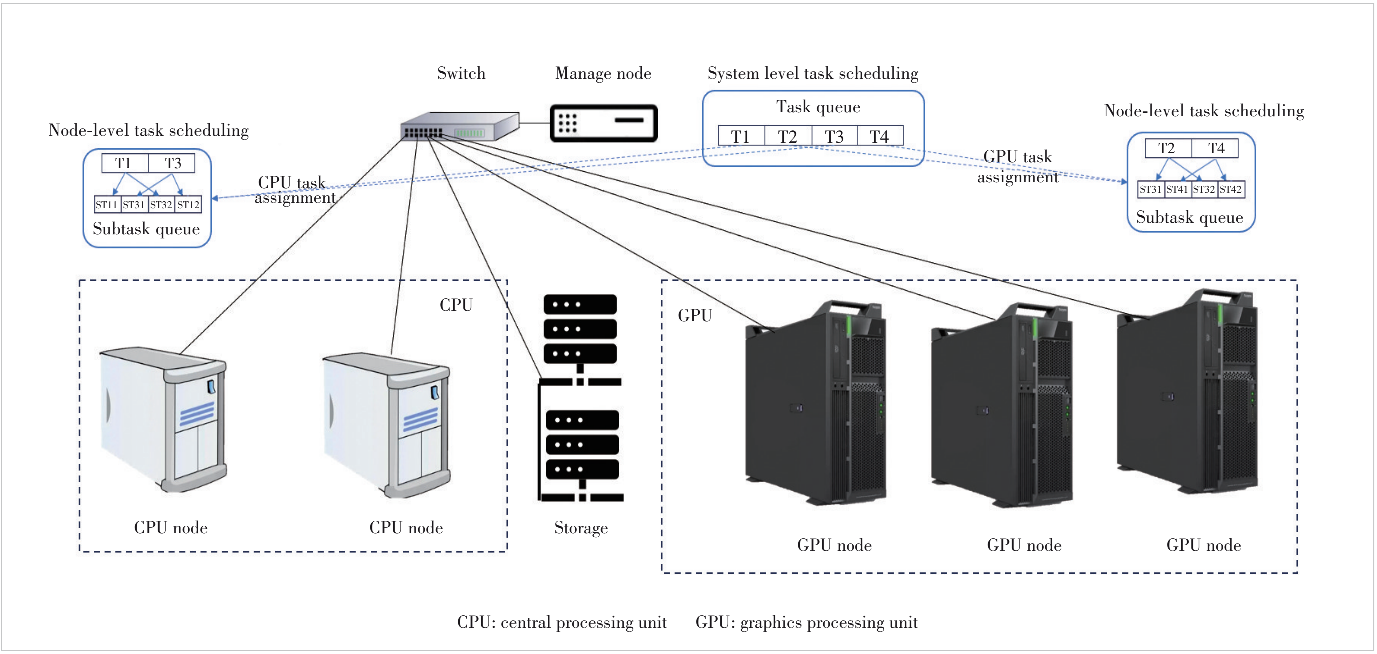
Task scheduling in cloud environments is a hot issue because of the prevalence of cloud computing. ARUNARANI et al. presented a comprehensive literature survey of task scheduling strategies and the associated metrics suitable for cloud

This work is supported by ZTE-University-Institute Fund Project under Grant No. IA20230629009.

computing environments[3], in which the methods, applications, and parameter-based measures utilized for task scheduling are discussed. QoS, ant colony optimization, particle swarm optimization, genetic algorithms, multi-processors, fuzzy algorithms, clustering, deadline constraints, and cost-based algorithms were summarized and analyzed. As mentioned by the survey, from 2003 to 2018, a large number of studies on different techniques to solve scheduling problems were conducted. YANG et al. reviewed task scheduling algorithms for cloud computing[6]. They divided scheduling algorithms into single-objective optimization algorithms and multi-objective task scheduling algorithms. They also analyzed the representative algorithms of each method, and compared and summarized the advantages and disadvantages of different algorithms. The authors in Ref. [7] categorized scheduling methods into traditional scheduling strategies, heuristic-based intelligent algorithms, emerging swarm intelligence algorithms, and hybrid algorithms, which accomplished a review of nature-inspired optimization techniques for scheduling tasks in cloud computing. SINGH et al. reviewed the meta-heuristics techniques for scheduling tasks in cloud computing[4], and presented the taxonomy and comparative review of these algorithms. Methodical analysis was presented based on swarm intelligence and bio-inspired techniques[4]. Since multi-objective optimization can deal with multiple conflicting goals, HOSSEINZADEH et al. presented a comprehensive survey and overview of the multi-objective scheduling approaches designed for various cloud computing environments[8]. They classified the scheduling schemes into different types such as reducing execution cost, reducing makespan, reducing SLA violation, and meeting deadlines, regarding applied multi-objective optimization algorithms. PRITY et al. provided a review of nature-inspired optimization techniques for scheduling tasks[9]. A novel classification taxonomy and comparative review of these techniques were presented. JAWADE et al. gave a compact analytical survey on task scheduling[10], which vividly explained different approaches utilized for task scheduling in diverse works.

Since AI has played an increasingly significant role, clusters with GPU become common and important. Amazon AWS, Microsoft Azure, Ali Cloud, Huawei Cloud, Baidu Cloud, and so forth, provide GPU computing services in their clouds. The hardware in such clouds is the CPU-GPU cluster. Tasks and devices in the CPU-GPU cluster are heterogeneous. PRADHAN et al. compared and described various task scheduling methods in heterogeneous cloud environments[5]. They categorized scheduling algorithms into heuristics and hybrid methods. Heuristics algorithms were categorized into static and dynamic scheduling. Dynamic scheduling was then categorized into online and batch modes.

In general, task scheduling has two stages. System-level task scheduling is carried out in the first stage, whose goal is to optimize system performance, such as load balance, total computing efficiency, power consumption, system response, and temperature constraints. After tasks are assigned to nodes, task scheduling is carried out within the nodes in the second stage, whose goal is to optimize computing node performance, such as makespan, node computing efficiency, and node temperature. Fig. 1 shows the two-stage task scheduling framework.



▲Figure 1. Two-stage task scheduling framework

3 Task Scheduling of CPU-GPU Heterogeneous Cluster

Compared with CPU clusters, the CPU-GPU cluster shows more heterogeneity in terms of architecture and task types. The task scheduling framework for CPU-GPU heterogeneous cluster is shown in Fig. 2.

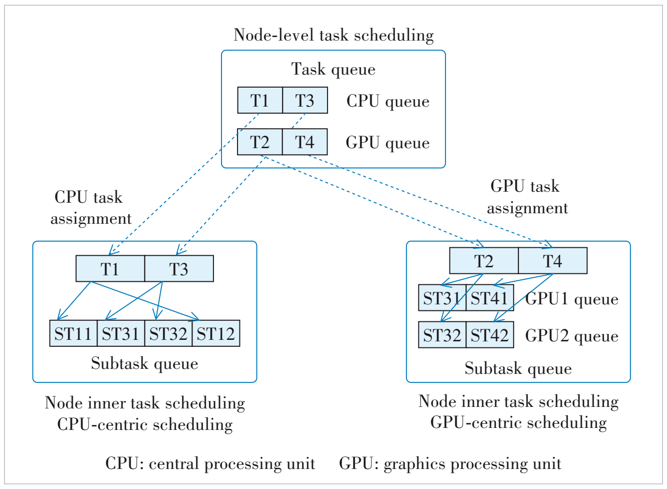
In node-level task scheduling, tasks could be classified as CPU tasks and GPU tasks depending on the type of device used. In node inner task scheduling, GPU tasks can be assigned to different GPU devices since there are multiple GPUs in the node. Based on different strategies and methodologies, the task scheduling for the CPU-GPU heterogeneous clusters still can follow heuristic methods and statistic methods partition.

3.1 Heuristic Methods

Heuristic task scheduling relies on human-designed models to strategies and performs task assignments[11 – 31]. Task assignments follow human-defined rules and are implemented by mathematical models and algorithms.

Early task scheduling was still based on the Hadoop framework[11]. The hybrid map method minimized the overall MapReduce job execution time by using profiles collected from dynamic monitoring of the map task behavior.

Later, energy consumption was taken into consideration in task scheduling that focused on system-level energy optimization[12]. The coarse-grained and fine-grained strategies of the Waterfall model were migrated into the scheme. The energy efficiency problem was translated into static power consumption loss and resource utilization problems. According to the heterogeneity of the tasks and task types, buddy allocation was proposed to improve energy efficiency. HUO et al. abstracted computing resources into many identical virtual CPUs and formulated the scheduling problem into an optimization problem with integer variables and nonlinear constraints[12]. The energy



▲ Figure 2. Task scheduling framework for CPU-GPU heterogeneous cluster

consumption minimization problem was formulated as an integer nonlinear programming problem, necessitating the determination of both a task assignment plan and a tailored resource allocation plan.

Similar to energy consumption, temperature becomes an optimized goal of task scheduling, especially for big clusters. Temperature, reliability, and computing performance were taken into account to reduce node performance differences and improve throughput per unit time in clusters[13]. Temperature heat islands caused by slow nodes could be prevented by optimizing scheduling. CAO and WANG proposed a novel task-scheduling model for GPU clusters with temperature limitations[14]. GPU and temperature were both considered during task scheduling. A state matrix was designed to monitor the GPU cluster and provide status information for the scheduler. Compared with the benchmark scheduling model, the loss of scheduling performance is more acceptable.

In order to deal with the problem of unbalanced workload, task classification and packing were both used to match CPU and GPU loads[15]. Tasks were classified into six classes according to parallelism degrees and workloads, and then assigned to minimize the execution time[16]. The approach is forming a good match between the task distribution and the architecture of the heterogeneous cluster through the task classifications and combinations. Besides task classification, CHEN et al. proposed a multi-granularity partition approach to synchronizing data flow graphs and task partition on CPU and GPU tasks[17]. This method can satisfy load balancing and improve the utilization rate of the cluster. CI et al. designed an adaptive scheduling strategy to alleviate imbalance and underutilization[18], which logically treated all GPUs in the cluster as a whole. Every cluster node maintains a local information table of all GPUs. Once a GPU call request is received, a node will select a GPU to run the task adaptively based on this table. This strategy could significantly improve the GPU utilization rate and reduce mean waiting time.

Some task scheduling strategies focus on makespan optimization. This can maximize the parallel degree between CPUs and GPUs. Ref. [19] presented a scheduling algorithm using a generic methodology. The main idea of the approach is to determine an adequate partition of the set of tasks on the CPUs and the GPUs using a dual approximation scheme. ZHU et al. separated data into different data splits first, and then the scheduler assigned tasks to the CPU or GPU according to their computing resources[20]. In this case, the data size becomes a measure of processing time.

Different types of tasks may be dominant in different clusters, and task-type-oriented scheduling strategies are then designed. A parallel version of the min-min heuristic method, which is an advanced parallel cellular genetic algorithm (CGA), was proposed for large instances of tasks in a cluster[21]. For short tasks, SHAO et al. implemented a container-based batch computing system, which accepted and executed users' jobs through container images and specified configurations[22]. A shortest-job-first-based scheduling policy was used to ensure the priority of the short tasks and to prevent long tasks from starving. For gaming tasks, ZHANG et al. proposed a fine-grained scheduling framework that decomposed game workloads into small and independent render tasks and dispatched the small tasks to different machines[23]. The scheduling objective was to maximize a utility function. If all resources are fully used by games, the utility function achieves its maximum value. The proposed approach requires only 26.4% of the servers compared with packing algorithms. Since task scheduling strategies based on artificial intelligence become more and more prevalent in clusters, efficient machine learning task scheduling has attracted increasing research interests[24–28]. CHEN et al. focused on convolutional neural networks (CNN)-based task scheduling[24]. The scheduling strategy leverages an analytical prediction model to optimize the allocation of computing resources for impending tasks, thereby enhancing system efficiency and prioritizing user satisfaction. To improve performance and reduce energy consumption, CHEN et al. proposed a prediction method to predict the completion time and energy consumption of deep training tasks first, and then used the GPU allocation strategy algorithm that depended on the prediction of completion time and energy consumption to assign tasks[25]. For more time-consuming training tasks, HAN et al. proposed a method to eliminate network contention by jointly optimizing network topology and communication patterns in distributed training[26]. CHEN et al. proposed a training-inference joint scheduling framework, called DeepBoot, to support training tasks and utilize the idle GPUs in the inference cluster[27]. DeepBoot could overcome the unbalanced GPU utilization stemming from the periodic difference in training and inference workload. CHEN et al. proposed a QoS-aware scheduling framework for a deep learning R&D platform[28]. The framework provides lightweight offline profiling and online dynamic scheduling on GPU clusters. Using the lightweight offline profiler, the framework could provide a prediction model according to the domain-specific information of deep learning tasks derived from a comprehensive characterization.

Some research focuses on the methodology of scheduling algorithms[29–31]. ZHANG and WU presented the weighted system-level scheduling algorithm (WSLSA) which involved the weights of the processor[29]. Due to the doubly linked list data structure for system-level tasks, the algorithm could assign and remove a task in a single direction (denoted by WSLSA-S) from the task list or it could also assign and remove a task in both directions (denoted by WSLSA-B) from the task list. ITURRIAGA et al. presented a parallel implementation on CPU/GPU of two variants of a stochastic local search method to efficiently solve the scheduling problem in heterogeneous computing systems[30]. Both methods are based on a set of simple operators to keep the computational complexity as low as possible. A two-level dynamic scheduling algorithm of CPU and GPU cooperative computing in heterogeneous clusters was proposed[31]. The algorithm could dynamically distribute data according to each node's computing capability and schedule tasks dynamically between the CPU and GPU in the node.

3.2 Statistic Methods

Different from heuristic task scheduling, statistic methods learn scheduling strategies from clusters of system data. The system data include CPU utilization information, GPU utilization information, host memory utilization information, GPU memory utilization information, node uplink traffic rate, node downlink traffic rate, global load throughput of GPU, global store throughput of GPU, etc. With the burgeoning development of AI, deep reinforcement learning and deep neural networks are employed in tasking scheduling[32–35].

By using the deep Q-network, the two-stage scheduling model was adopted to learn to perform the current optimal scheduling actions online according to the runtime status of cluster environments, the characteristics of video tasks, and the dependencies between video tasks[32]. The interference-aware workload parallelization (IAWP) method assigns subtasks with dependencies to the appropriate computing units, taking the interference of subtasks on the GPU by using neural collaborative filtering into account[33]. To make the learning of neural networks more efficient, pre-training is adopted in the two-stage scheduler. The transfer learning technology is used to efficiently rebuild the task scheduling model referring to the existing model.

Since prediction-based schedulers are limited in terms of their prediction accuracy and offline-profiling overhead, the Q-learning framework was designed to model the R&D scenarios and was proposed to build a series of implementations including state space, action space, reward function, and update scheme for task scheduling[34]. The learning agent could learn from the feedback on task performance independently and continuously to adjust online task scheduling decisions. The Q-learning-based scheduler significantly improves the task average normalized throughput and makespan. Moreover, the proposed scheduler is more suitable for long-term deep-learning R&D scenarios.

The deep network can be used to produce task scheduling strategy candidates first[35]. A set of feasible solutions is then generated through cross-variance and other operations. The optimal solutions are screened out and stored in the empirical buffer area. Finally, the neural network parameters are optimized through the empirical buffer samples.

Compared with heuristic task scheduling methods, statistic methods have demonstrated fewer achievements. This is because task scheduling is dynamic and non-deterministic polynomial (NP)-hard. It is not easy to obtain a suitable deep network to efficiently describe all behaviors of task scheduling.

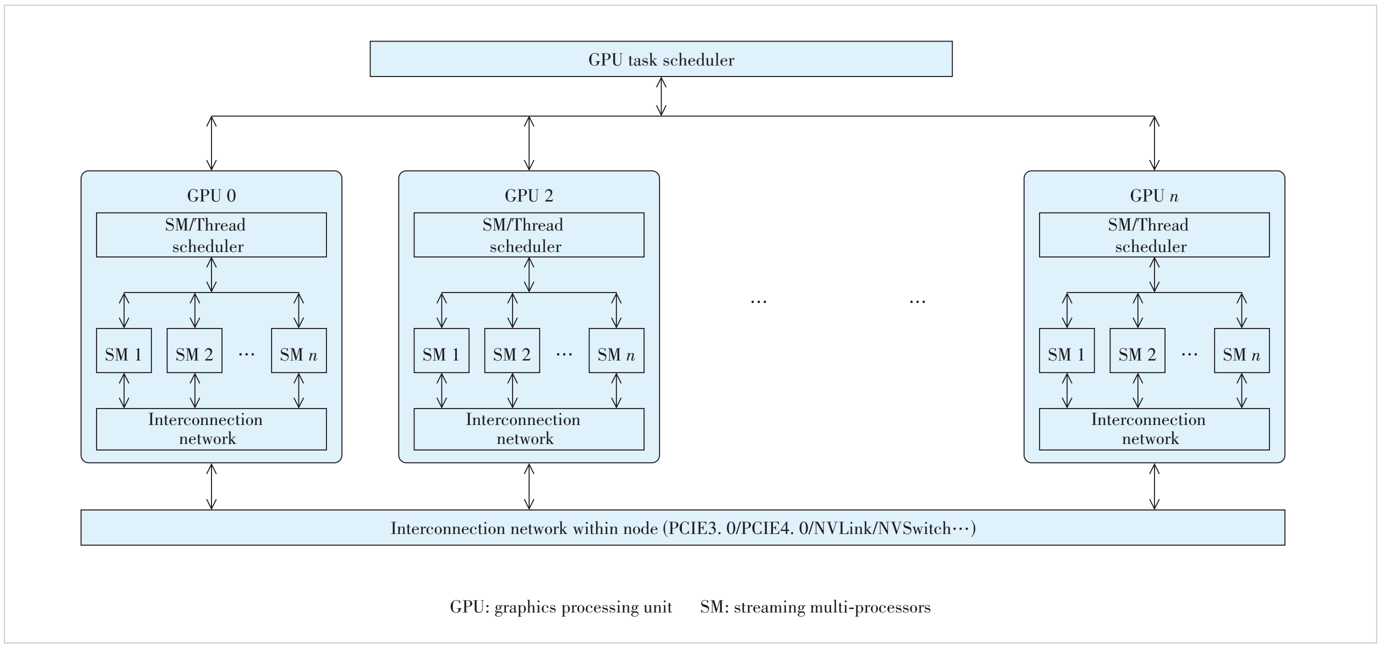
Along with the evolution of machine learning technologies, more and more algorithms will emerge and provide better performance.

4 Task Scheduling of GPU

Modern GPU computing allows application programmers to exploit parallelism using new parallel programming languages such as CUDA and OpenCL and a growing set of familiar programming tools, leveraging the substantial investment in parallelism that high-resolution real-time graphics require and AI applications[36]. Different from the CPU, the GPU cannot be partitioned into arbitrary virtual GPUs with cores. GPUs also cannot compute without cooperation with the CPU. Therefore, GPUs show different characteristics of computing. GPUs offer the capability to handle multiple tasks at the process level, that is, multiple processes can run on one GPU. However, frequent process switching may destroy the hardware pipeline of GPUs. Usually, a server may have several GPUs. Different GPUs are treated as different devices although such GPUs are connected by high-speed interconnection networks. As a result, tasks can be scheduled among GPUs and processed within a GPU server, which is device-level task scheduling. The task scheduling framework of GPU is shown in Fig. 3.

Most research focuses on task scheduling within GPU[37–45]. The goal of task scheduling within GPU is to exploit the throughput of the GPU kernel and give the best separation of SM or threads. In the early stage, AUGONNET et al. presented StarPU, a runtime system that efficiently exploited heterogeneous multicore architectures[37]. StarPU provides a uniform execution model and a high-level framework to design scheduling policies. StarPU permits dynamically selecting the strategies at runtime, thus letting the programmer try and choose the most efficient strategy. This makes it possible to benefit the scheduling without setting restrictions or making excessive assumptions. Later, ZHONG and HE proposed Kernelet, a runtime system that improved the throughput of concurrent kernel executions on the GPU[38]. Kernelet embraces transparent memory management, PCIe data transfer techniques, and dynamic slicing and scheduling techniques for kernel executions. A novel Markov chain-based performance model to guide the scheduling decision was proposed in Kernelet. Recently, ZOU et al. proposed RTGPU that combined fine-grain GPU partitioning on the system side with a novel scheduling algorithm on the theory side[39]. RTGPU leverages a precise timing model of the GPU applications with the persistent threads technique and improves fine-grained utilization through interleaved execution. The RTGPU real-time scheduling algorithm can provide real-time guarantees of meeting deadlines for GPU tasks with better scheduling ability.

Besides the scheduling framework, some research focuses on scheduling strategies[40–41]. LOPEZ-ALBELDA et al. employed a scheduling theory to build a model that took into account the device capabilities, workload characteristics, constraints, and objective functions[40]. The heuristic called NEH-GPU, which combines an existing heuristic with a GPU task execution model, has been developed. HUANG et al. proposed a dynamic GPU task balance scheduling called the coefficient of balance and equipment history ratio value (CB-HRV) task scheduling[41], which was developed to reduce system energy consumption during task execution by allocating tasks based on workload balance, thereby improving GPU en-


GPU task scheduler
GPU 0
GPU 2
GPU n
SM/Thread scheduler
SM 1
SM 2
SM n
Interconnection network
Interconnection network within node (PCIE3. 0/PCIE4. 0/NVLink/NVSwitch…)
GPU: graphics processing unit   SM: streaming multi-processors

▲**Figure 3. Task scheduling framework of GPU** ergy usage. The CB-HRV algorithm is more balanced, and it allows the computing device to be utilized more reasonably and efficiently.

For special types of tasks, an efficient task scheduling frame was designed[42 – 44]. LI et al. proposed a two-level scheduling strategy to distribute irregular tasks and enable resource sharing on GPUs, by managing tasks and threads hierarchically[42]. The framework manages both tasks and threads in the two levels to allow for helpful resource sharing. KWON et al. proposed Nimble, a deep-learning execution engine that runs GPU tasks in parallel with minimal scheduling overhead[43]. Nimble introduces a technique called ahead-of-time (AoT) scheduling, which pre-runs the given neural network once according to the generated stream mapping and records all the GPU tasks as an execution trace. AoT scheduling also intercepts memory allocation/free requests from the base framework and reserves the GPU memory allocated in the pre-run. At the end of the AoT scheduling, Nimble packs the execution trace and reserves the memory into a task schedule. CHEN et al. presented Atos, a task-parallel GPU dynamic scheduling framework that was especially targeted at dynamic irregular applications[44]. Atos exposes additional concurrency by supporting task-parallel formulations of applications with relaxed dependencies, achieving higher GPU utilization. Atos also offers implicit task-parallel load balancing in addition to data-parallel load balancing, providing users the flexibility to balance between them to achieve optimal performance.

However, there is a relative scarcity of research on task scheduling for muti-GPU systems. TANG et al. proposed AEML, an acceleration engine for multi-GPU load-balancing in distributed heterogeneous environments[45]. AEML could effectively integrate GPUs into the distributed processing framework and achieve good load balance among multiple heterogeneous GPUs. To achieve the best load-balancing among multiple heterogeneous GPUs, the AEML model utilizes four techniques: a fine-grained task mapping mechanism, a device resource unified management scheme, a novel resource-aware GPU task scheduling strategy, and a feedback-based stream adjustment scheme.

5 Evaluation

The goal of task scheduling is to exploit all potential parallelism of heterogeneous clusters composed of multi-CPUs and multi-GPUs. Some metrics are adopted to evaluate and compare different methods, such as makespan, load balance, resource utilization, energy, speedup and QoS[46 – 47]. The scheduling methods will optimize several metrics simultaneously. We summarize the motioned scheduling technologies in Table 1.

From Table 1, it can be clearly seen that most algorithms focus on makespan and resource utilization. This is because that makespan is very important for cluster users' feeling. Resource utilization optimization can improve cluster system effi-

▼Table 1. Summary of scheduling technologies based on evaluation metrics

Task Scheduling Technology	Evaluation Metrics					
	Speed-up	Energy	Load balance	Makes-pan	Resource utilization	QoS
Hybrid map[11]	√					
Energy efficient task scheduling[12]		√				
Energy-minimized scheduling[13]		√				
Task scheduling with temperature constraint[14 – 15]		√				
PTA&WSLSA[16]			√			
Multi-granularity partition[17]			√			
Adaptive and transparent task scheduling[18]			√			
Dual approximation technique[19]				√		
Data partition[20]				√		
Large instance sheduling[21]	√					
Short task scheduling[22]				√		
Fine-grained scheduling[23]					√	
CNN-based task scheduling[24]				√	√	
GAS[25]		√		√		√
Isolated scheduling[26]	√				√	√
DeepBoot[27]					√	
QoS guarantee scheduling[28]						√
Greedy heuristics[29]			√			
Local serach[30]				√	√	
CPU and GPU cooperative scheduling[31]					√	
Learning driven scheduling[32 – 33]				√	√	
Q-learning[34]				√		
Dynamic priority task scheduling[35]				√		√
StarPU[37]					√	
Kernelet[38]	√				√	
RTGPU[39]					√	
Heuristics for concurrent task scheduling[40]				√	√	
Task balance scheduling[41]		√				
Two-level task Scheduling[42]	√					
Nimble[43]	√					
Atos[44]	√					
AEML[45]			√		√	

CNN: convolutional neural network
CPU: central processing unit
GAS: GPU allocaion strategy
GPU: graphics processing unit
PTA: packing task algorithm
QoS: Quality of Service
WSLSA: weighted system-level scheduling algorithm

ciency, which is the main purpose of task scheduling technologies. Energy-based task scheduling is mainly designed for GPU clusters since GPUs are highly energy-consuming. Heuristic methods are dominant since statistic-based approaches require running data from the real clusters which are hard to collect. However, learning-based scheduling can provide more intelligent scheduling strategies and will attract more attention in future research.

6 Conclusions

Task scheduling is a long-term hot research topic in companies with cloud computing and AI's flourishing. Most task scheduling strategies are heuristic and based on experts' experience. Statistic strategies have attracted researchers' interest recently. Different from task scheduling of CPU clusters, task scheduling of CPU-GPU clusters is more complex due to heterogeneous system composition. Task-oriented scheduling focuses on short tasks, gaming tasks, deep learning tasks, etc. This paper also reviews task scheduling strategies within GPU. This is because GPU has process-level parallel ability and incompatible tasks will decrease GPU's workflow and parallelism.

This paper describes a task scheduling framework for CPU-GPU heterogeneous clusters and a task scheduling framework for GPU servers with multiple GPUs. From the two frameworks, we can clearly see that task scheduling can be separated into the system level, the node level, and the device level. Although research achievements in statistic task scheduling are less than heuristic task scheduling, statistic task scheduling is still a highly potential technology.

References

[1] BOHN R B, MESSINA J, LIU F, et al. NIST cloud computing reference architecture [C]//IEEE World Congress on Services. IEEE, 2011: 594 – 596. DOI: 10.1109/SERVICES.2011.105
[2] CAITHNESS N, DRESCHER M, WALLOM D. Can functional characteristics usefully define the cloud computing landscape and is the current reference model correct? [J]. Journal of cloud computing, 2017, 6(1): 10. DOI: 10.1186/s13677-017-0084-1
[3] ARUNARANI A, MANJULA D, SUGUMARAN V. Task scheduling techniques in cloud computing: a literature survey [J]. Future generation computer systems, 2019, 91: 407 – 415. DOI: 10.1016/j.future.2018.09.014
[4] SINGH P, DUTTA M, AGGARWAL N. A review of task scheduling based on meta-heuristics approach in cloud computing [J]. Knowledge and information systems, 2017, 52(1): 1 – 51. DOI: 10.1007/s10115-017-1044-2
[5] PRADHAN R, SATAPATHY S C. Advances in intelligent systems and computing: task scheduling in heterogeneous cloud environment—a Survey [M]. Singapore: Springer, 2020
[6] YANG G, ZHAO X, HUANG J. Survey on task scheduling algorithms for cloud computing [J]. Computer systems and applications, 2020, 29 (3): 11 – 19, DOI: 10.15888/j.cnki.csa.007261
[7] WANG H, WANG H H. Survey on task scheduling in cloud computing environment [C]//The 7th International Conference on Intelligent Informatics and Biomedical Science. IEEE, 2022: 286 – 291. DOI: 10.1109/ICIIBMS55689.2022.9971622
[8] HOSSEINZADEH M, GHAFOUR M Y, HAMA H K, et al. Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review [J]. Journal of grid computing, 2020, 18(3): 327 – 356. DOI: 10.1007/s10723-020-09533-z
[9] PRITY F S, GAZI M H, ASLAM UDDIN K M. A review of task scheduling in cloud computing based on nature-inspired optimization algorithm [J]. Cluster computing, 2023, 26(5): 3037 – 3067. DOI: 10.1007/s10586-023-04090-y
[10] JAWADE P B, SAI KUMAR D, RAMACHANDRAM S. A compact analytical survey on task scheduling in cloud computing environment [J]. International journal of engineering trends and technology, 2021, 69(2): 178 – 187. DOI: 10.14445/22315381/ijett-v69i2p225
[11] SHIRAHATA K, SATO H, MATSUOKA S. Hybrid map task scheduling for GPU-based heterogeneous clusters [C]//The 2nd International Conference on Cloud Computing Technology and Science. IEEE, 2010: 733 – 740. DOI: 10.1109/CloudCom.2010.55
[12] HUO H P, SHENG C C, HU X M, et al. An energy efficient task scheduling scheme for heterogeneous GPU-enhanced clusters [C]//International Conference on Systems and Informatics. IEEE, 2012: 623 – 627. DOI: 10.1109/ICSAI.2012.6223074
[13] HU B, YANG X C, ZHAO M G. Energy-minimized scheduling of intermittent real-time tasks in a CPU-GPU cloud computing platform [J]. IEEE transactions on parallel and distributed systems, 2023, 34(8): 2391 – 2402. DOI: 10.1109/TPDS.2023.3288702
[14] CAO Y P, WANG H F. A task scheduling scheme for preventing temperature hotspot on GPU heterogeneous cluster [C]//International Conference on Green Informatics. IEEE, 2017: 117 – 121. DOI: 10.1109/ICGI.2017.20
[15] WANG H F, CAO Y P. Task scheduling of GPU cluster for large-scale data process with temperature constraint [C]//International Conference on Computer Engineering and Networks. CENet, 2020: 110-117.10.1007/978-3-030-14680-1_13
[16] ZHANG K L, WU B F. Task scheduling for GPU heterogeneous cluster [C]//International Conference on Cluster Computing Workshops. IEEE, 2012: 161 – 169. DOI: 10.1109/ClusterW.2012.20
[17] CHEN W B, YANG R R, YU J Q. Multi-granularity partition and scheduling for stream programs based on multi-CPU and multi-GPU heterogeneous architectures [J]. Computer engineering & science, 2017, 39(1): 15. DOI: 10.3969/j.issn.1007-130X.2017.01.002
[18] CI Q Y, LI H R, YANG S W, et al. Adaptive and transparent task scheduling of GPU-powered clusters [J]. Concurrency and computation: Practice and experience, 2022, 34(9): e5793. DOI: 10.1002/cpe.5793
[19] KEDAD-SIDHOUM S, MONNA F, MOUNIÉ G, et al. Scheduling independent tasks on multi-cores with GPU accelerators [C]//European Conference on Parallel Processing. Berlin, Heidelberg: Springer, 2014: 228-237.10.1007/978-3-642-54420-0_23
[20] ZHU Z Y, TANG X C, ZHAO Q. A unified schedule policy of distributed machine learning framework for CPU-GPU cluster [J]. Journal of northwestern polytechnical university, 2021, 39(3): 529 – 538. DOI: 10.1051/jnwpu/20213930529
[21] PINEL F, DORRONSORO B, BOUVRY P. Solving very large instances of the scheduling of independent tasks problem on the GPU [J]. Journal of parallel and distributed computing, 2013, 73(1): 101 – 110. DOI: 10.1016/j.jpdc.2012.02.018
[22] SHAO J L, MA J M, LI Y, et al. GPU scheduling for short tasks in private cloud [C]//IEEE International Conference on Service-Oriented System Engineering. IEEE, 2019: 215 – 2155. DOI: 10.1109/SOSE.2019.00037
[23] ZHANG W, LIAO X F, LI P, et al. Fine-grained scheduling in cloud gaming on heterogeneous CPU-GPU clusters [J]. IEEE network, 2018, 32(1): 172 – 178. DOI: 10.1109/MNET.2017.1700047
[24] CHEN Z Y, LUO L, QUAN W, et al. Multiple CNN-based tasks scheduling across shared GPU platform in research and development scenarios [C]//The 20th International Conference on High Performance Computing

and Communications. IEEE, 2018: 578 – 585. DOI: 10.1109/HPCC/SmartCity/DSS.2018.00107

[25] CHEN Y W, HAN J C, ZHOU H, et al. GAS: GPU allocation strategy for deep learning training tasks [C]//IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles. IEEE, 2022: 880 – 887. DOI: 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00133

[26] HAN X C, JIANG W H, CAO P R, et al. Isolated scheduling for distributed training tasks in GPU clusters [EB/OL]. (2023-8-10)[2023-9-30]. https://doi.org/10.48550/arXiv.2308.05692

[27] CHEN Z Q, ZHAO X K, ZHI C, et al. DeepBoot: dynamic scheduling system for training and inference deep learning tasks in GPU cluster [J]. IEEE transactions on parallel and distributed systems, 2023, 34(9): 2553 – 2567. DOI: 10.1109/TPDS.2023.3293835

[28] CHEN Z Y, QUAN W, WEN M, et al. Deep learning research and development platform: characterizing and scheduling with QoS guarantees on GPU clusters [J]. IEEE transactions on parallel and distributed systems, 2020, 31(1): 34 – 50. DOI: 10.1109/TPDS.2019.2931558

[29] ZHANG K L, WU B F. Task scheduling greedy heuristics for GPU heterogeneous cluster involving the weights of the processor [C]//IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum. IEEE, 2013: 1817 – 1827. DOI: 10.1109/IPDPSW.2013.38

[30] ITURRIAGA S, NESMACHNOW S, LUNA F, et al. A parallel local search in CPU/GPU for scheduling independent tasks on large heterogeneous computing systems [J]. The journal of supercomputing, 2015, 71(2): 648 – 672. DOI: 10.1007/s11227-014-1315-6

[31] GAO Y, GU W J, DING Y H, et al. Design and implementation of CPU and GPU cooperative scheduling algorithm with heterogeneous clusters [J]. Computer engineering and design, 2020, 41(2): 10. DOI: CNKI:SUN:SJSJ.0.2020-02-045

[32] ZHANG H T, TANG B C, GENG X, et al. Learning driven parallelization for large-scale video workload in hybrid CPU-GPU cluster [C]//The 47th International Conference on Parallel Processing. ACM, 2018. DOI: 10.1145/3225058.3225070

[33] ZHANG H T, GENG X, MA H D. Learning-driven interference-aware workload parallelization for streaming applications in heterogeneous cluster [J]. IEEE transactions on parallel and distributed systems, 2021, 32(1): 1 – 15. DOI: 10.1109/TPDS.2020.3008725

[34] CHEN Z Y. Research on deep learning task scheduling based on small scale GPU cluster platform [M]. Hunan, China: National University of Defense Technology, 2019

[35] QI Y F, He X. Dynamic priority task scheduling algorithm based on deep learning [J]. Computer systems and applications, 2023, 32(7): 195 – 201. DOI: 10.15888/j.cnki.csa.009169

[36] KECKLER S W, DALLY W J, KHAILANY B, et al. GPUs and the future of parallel computing [J]. IEEE micro, 2011, 31(5): 7 – 17. DOI: 10.1109/MM.2011.89

[37] AUGONNET C, THIBAULT S, NAMYST R, et al. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures [M]//Lecture Notes in Computer Science. Berlin, Germany: Springer, 2009: 863 – 874. DOI: 10.1007/978-3-642-03869-3_80

[38] ZHONG J L, HE B S. Kernelet: high-throughput GPU kernel executions with dynamic slicing and scheduling [J]. IEEE transactions on parallel and distributed systems, 2014, 25(6): 1522 – 1532. DOI: 10.1109/TPDS.2013.257

[39] ZOU A, LI J, GILL C D, et al. RTGPU: real-time GPU scheduling of hard deadline parallel tasks with fine-grain utilization [J]. IEEE transactions on parallel and distributed systems, 2023, 34(5): 1450 – 1465. DOI: 10.1109/TPDS.2023.3235439

[40] LÓPEZ-ALBELDA B, LÁZARO-MUÑOZ A J, GONZÁLEZ-LINARES J M, et al. Heuristics for concurrent task scheduling on GPUs [J]. Concurrency and computation, 2020, 32(20): e5571. DOI: 10.1002/cpe.5571

[41] HUANG Y H, GUO B, SHEN Y. GPU energy optimization based on task balance scheduling [J]. Journal of systems architecture, 2020, 107: 101808. DOI: 10.1016/j.sysarc.2020.101808

[42] LI J, LIU L, WU Y, et al. Two-level task scheduling for irregular applications on GPU platform [J]. International journal of parallel programming, 2017, 45(1): 79 – 93. DOI: 10.1007/s10766-015-0387-0

[43] KWON W, YU G-I, JEONG E, et al. Nimble: lightweight and parallel GPU task scheduling for deep learning [C]//The 34th International Conference on Neural Information Processing Systems. Curran Associates, 2020: 8343 – 8354. DOI: https://dl. acm. org/doi/10.5555/3495724.3496423

[44] CHEN Y X, BROCK B, PORUMBESCU S, et al. Atos: a task-parallel GPU scheduler for graph analytics [C]//The 51st International Conference on Parallel Processing. ACM, 2022. DOI: 10.1145/3545008.3545056

[45] TANG Z, DU L F, ZHANG X D, et al. AEML: an acceleration engine for multi-GPU load-balancing in distributed heterogeneous environment [J]. IEEE transactions on computers, 2022, 71(6): 1344 – 1357. DOI: 10.1109/TC.2021.3084407

[46] LIMA J V F, GAUTIER T, DANJEAN V, et al. Design and analysis of scheduling strategies for multi-CPU and multi-GPU architectures [J]. Parallel computing, 2015, 44: 37 – 52. DOI: 10.1016/j.parco.2015.03.001

[47] ALEBRAHIM S, AHMAD I. Task scheduling for heterogeneous computing systems [J]. The journal of supercomputing, 2017, 73(6): 2313 – 2338. DOI: 10.1007/s11227-016-1917-2

Biographies

ZHOU Yiheng has received his bachelor's degree in robotic engineering from University of Shanghai for Science and Technology, China in 2024. He has been working as a research assistant at Peking University, China since 2023. His research interests include computer vision (detection and pose estimation), robotic arm control, and artificial intelligence (AI computing platform).

ZENG Wei (weizeng@pku. edu. cn) is currently a research professor at the School of Computer Science, Peking University, China. He was a senior researcher at NEC Laboratories, China from 2005 to 2015. He worked as a visiting scholar at Stanford University, USA in 2012. He received his PhD degree in computer science and engineering from Harbin Institute of Technology, China in 2005. His research interests include computer vision (object detection and segmentation), artificial intelligence (AI computing platform), and media analysis (video retrieval). He is the author or coauthor of over 40 refereed journals and conference papers. He was the reviewer of ICCV2023, CVPR2023, ACM MM2022, ICPR2020, BigMM2020, and so forth.

ZHENG Qingfang received his BS degree from Shanghai Jiaotong University, China in 2002 and PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 2008. He is now the chief scientist of cloud video product and deputy director of the Video Technology Committee of ZTE Corporation. His current research interests include video communication, computer vision and artificial intelligence.

LIU Zhilong is currently working with the Video Systems Department, ZTE Corporation as a senior system architecture engineer. He graduated from University of Electronic Science and Technology of China with a master's degree in 2015. His research interests include media transmission, real-time audio and video networks (RTN), and video computing networks.

CHEN Jianping is currently a senior research engineer at the School of Computer Science, Peking University, China. His research interests include media analysis, AI platform, and video retrieval.