SmartDis: Near-Optimal Task Scheduling in Multi-edge Networks

Weiwei Miao[1], Zeng Zeng[1], Chuanjun Wang[1], and Zhuzhong Qian[2(✉)]

[1] State Grid Jiangsu Electric Power Co., Ltd. Information and Telecommunication Branch, Nanjing, China

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

qzz@nju.edu.cn

Abstract. In multi-edge networks, as the bandwidth and computing resources of edge servers are limited, transmission and processing of large amounts of data could bring significant pressure, leading to violations of service agreements. Thus, it is very important to schedule tasks in edge network efficiently for better performance. In this paper, we formulate the problem as minimizing the overall completion time of tasks in edge networks. Since the problem can be proved to be NP-hard, we propose a novelty algorithm, SmartDis, for scheduling tasks accross multiple edges. The main idea of SmartDis is to select offload slots of tasks based on the principle of choosing the smallest sum of added value of the overall completion time. We show theoretically that the system transmission time of SmartDis is within a constant times of the optimal result, as long as the data upload is scheduled according to the transmission order. The evaluation results illustrate that SmartDis is superior to other cross-domain job scheduling algorithms at this stage, achieving a performance improvement of at least 25%.

Keywords: Multi-edge network · Edge bandwidth · Task schedule

1 Introduction

Multi-edge computing is a distributed computing framework that brings applications closer to data sources. It has become one of the most promising ways to improve response times and bandwidth availability for IoT devices. In a typical multi-job scenario, raw data is first collected from different devices and gathered in an edge server for processing. Edge computing improves the efficiency by dispatching data close to each edge server for distributed execution. Recent research results show that under the distributed execution mode, 90% of the

Supported by the Science and Technology Project of State Grid Corporation of China, Research on Key Technologies of Edge Intelligent Computing for Smart IoT System (Grand No. 5210ED209Q3U).

H. Jiang et al. (Eds.): ICECI 2020, LNICST 368, pp. 15–34, 2021.
https://doi.org/10.1007/978-3-030-73429-9_2

job completion time is shortened to 33% [10], and wide area network bandwidth usage is reduced by a factor of 250 [23,24]. Furthermore, for jobs like query, the speed can be increased by 3–19 times, and the transmission cost of the WAN can be reduced by 15–64% [16]. This framework can bring significant advantage for delay-sensitive tasks and data-privacy tasks, such as automatic driving [14], public security [1,6], customized healthcare [4,15], and unmanned retail [21].

However, since the bandwidth and computing resources in edge networks are limited, it is a critical issue to efficiently schedule and offload tasks. Figure 1 describes an edge network environment, which is composed of a terminal access point (Access Point, AP) and several edge computing slots. Terminals upload the data to the edge servers through the AP. In this scenario, the bottleneck is the upload link bandwidth of the AP, which is less studied in previous researches. The classic multi-machine scheduling problem [2,5,8,12] and the Current Open Shop problem [9,13,18] mainly deal with jobs that can be execyted after release, and hence can't cope with such challenges.

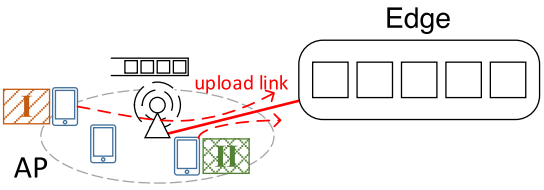


Fig. 1. Edge network environment.

In edge computing, due to the wide distribution of the edge regions and the uneven density of users in different locations, the data processing requirements naturally vary alongside the locations and time. The deviation also exists in the service capability of the edge. Therefore, completing tasks as quickly as possible on a single edge does not necessarily shorten the job completion time. Even providing services for certain subtasks is "wasting" resources, because the sibling tasks of these subtasks on other edges are being delayed due to the imbalance in execution caused by the above deviations. Therefore, to get a better average job completion time, it is more important to give priority to other tasks, such as the task that can dominate the completion time of the job.

In this paper, we propose a novel polynomial-time algorithm SmartDis, for efficient scheduling and offloading tasks in multi-edge computing. Considering the bandwidth constraints, the problem is compounded as a joint scheduling problem of network and computing resources. We first theoretically prove the NP-hardness of the problem. Since the offloading occurs after the data upload is completed, this paper naturally dismantles the problem into two sub-problems of optimizing data upload time and optimizing offload calculation time. The main idea of SmartDis is to arrange the order for the data upload request at each edge AP based on the primal dual method, to minimize the global data upload delay: When the data upload of a certain task is completed, SmartDis immediately selects the offload

slot for the task based on the principle of the smallest value added of the sum of the completion time, and arranges the scheduling timing for tasks on the slot.

We rigorously prove the theoretical performance guarantee of SmartDis. If the upload sequence of tasks is arranged according to SmartDis's sequencing strategy, the global data upload delay can be guaranteed to be within a constant times of the optimal value. Finally, we conduct extensive experiments via both implementation and simulation to validate the practical performance of SmartDis. We simulate job specifications in an real commercial cluster environment and compare with several scheduling mechanisms, including traditional scheduling mechanisms (FCFS and SRPT), recent scheduling mechanisms designed for concurrent execution of multiple data centers (Reordering and SWAG). Then we conduct evaluations on execution performance, fairness, sensitivity, and scheduling execution cost. Compared with the heuristic scheduling algorithm based on the SRPT class, SmartDis can improve the average job completion time by up to 33%, while keeping the additional calculation and communication overhead low.

2 Related Work

Since the WAN is a key bottleneck in cross-domain analysis, existing work is aimed at coordinating data distribution and task scheduling among multiple data-centers to reduce the WAN transmission time. Literature [16] emphasizes the difference between the upstream and downstream bandwidth of each cluster. If the low bandwidth carries high data volume, the receive/send operation will often become the bottleneck of the job. Literature [16] therefore optimizes the placement of two stages' data and tasks to avoid higher bottlenecks in the data transmission process, thereby reducing job completion time. Literature [23] aimed at minimizing WAN bandwidth consumption, and adjusted the query-execution plan and data backup scheme of SQL jobs. This move uses the cheap storage resources in a single data-center to cache the intermediate results of queries, aiming at avoiding redundant data transmission. The above research work aims to reduce the WAN data transmission delay. However, purely optimizing WAN transmission delay will cause uneven load on clusters, so the task will be backlogged in some "hot spots" clusters, causing execution bottlenecks.

Literature [11] proposed that the task execution sequence within a data center may still extend the completion time of the entire job. The reason is that the completion of the job depends on the most lagging subtask, so the subtasks completed in advance can be postponed appropriately.It is closest to the research work of this paper. It points out that if the subtasks' sibling tasks have a higher delay in other data centers, reducing the subtask's completion time does not speed up job completion. Therefore, it is appropriate to postpone this subtask and give resources to other competitors with "faster" sibling tasks, which may have a better average delay globally. The literature first proposed an auxiliary Reordering mechanism to adjust the existing scheduling order for the imbalance of delay between sibling tasks. The basic idea behind it is to try to delay the tasks in the queue, as long as the delay does not increase the overall completion time

of the job, but provides opportunities for other jobs to shorten the completion time. The specific operation is (1) Find the end task of the queue with the longest completion time in each data center. (2) The job to which the task belongs is inferior to other jobs. (3) Extract the task of the job from each queue and update the queue completion time. Repeat the above steps until all jobs are reordered. Since Reordering is a conservative method (used to adjust the existing scheduling), its result depends on the original algorithm's room for improvement. Literature [11] further proposes a complete scheduling algorithm SWAG, and it does not need Reordering assistance. The scheduling principle is to give priority to the jobs that make the longest queue with a least value added. The specific operation is (1) Calculate the increment of the completion time of each queue when executing each job. (2) Select the job that makes the longest queue with a least value added,and schedule it first (3) Update the length of each queue after executing the job. Repeat the above steps to sort all jobs.

However, the above work is based on several assumptions: first, data centers have the same number of computing slots; second, all slots have the same configuration and computing performance. While, in a multi-edge system, the scale of each edge cluster and the configuration of computing slots are heterogeneous (note that the processing time of a task on different edge servers is independent and different in this paper). This will lead to the coupling of scheduling and offloading, that is, the competition and imbalance when tasks are offloaded to different slots are different. The above scheduling algorithm that ignores heterogeneity is therefore not applicable.

3 Model and Problem Formulation

3.1 System Model

During execution, the multi-edge system requires a logically centralized coordinator, which is deployed either in the cloud or on a strong edge. According to the constraints of specific scenarios, the coordinator formulates corresponding decisions and delivers them to the edge for execution. In Fig. 2, a job is submitted to the coordinator. Due to the massive, redundant, and low-quality raw data produced by terminal. The tasks of the job are usually dispatched to the edge near the raw data.

In a wide area network environment, there are several edge APs, forming a set $\mathcal{P}$. Each AP is connected to an edge cluster $p \in \mathcal{P}$ deployed nearby, and let $\mathcal{S}_p$ be the set of computing slots at edge p. Set the uplink bandwidth of the AP to $\mathcal{B}_p$. A batch of analysis jobs $\mathcal{J} = \{1, 2, ..., n\}$ continuously arrives at the system, r_j is the arrival time of job $j \in \mathcal{J}$. The initial stage of each job consists of several tasks, which are responsible for processing the geographically distributed raw data. v_j^p is the amount of data that job j needs to process at edge p. Job j assigns a task u_j^p to edge p to perform data processing, and $d_s^{p,j}$ is the processing time of the task on slot $s \in \mathcal{S}_p$.

Before the task u_j^p performs data processing, data needs to be uploaded from the terminal device to the edge cluster. Let $\mathcal{T}_j^p$ be the data upload completion time,

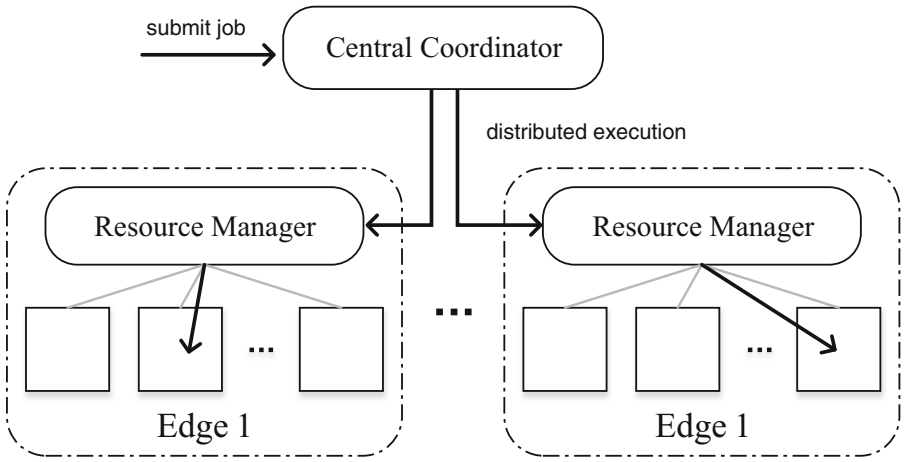


Fig. 2. Data analysis jobs run in the multi-edge system.

$$\mathcal{T}_j^p = r_j + wn_j^p + \frac{v_j^p}{\mathcal{B}_p}, \tag{1}$$

Among them, wn_j^p is the time that the data of the task is queued up at the access point p to be uploaded. After the data upload is completed, the edge cluster scheduler plans the slots used for offloading and scheduling timing for the task. Considering the limited computing power of the edge server, this paper sets the server to only handle one task at a time. If the task is offloaded to the server $s \in \mathcal{S}_p$ to perform data processing, let $\mathcal{C}_j^p$ be the completion time of the task u_j^p, then

$$\mathcal{C}_j^p = \mathcal{T}_j^p + wc_s^{p,j} + d_s^{p,j}, \tag{2}$$

Among them,$wc_s^{p,j}$ is the queuing delay of the task on the server s. Let $\mathcal{C}_j = \max_{p \in \mathcal{P}} \mathcal{C}_j^p$ represent the completion time of job j. For these jobs executed on multiple edges,this paper coordinates the data upload order of tasks(wn_j^p) , the decision of the slot where the task is offloaded (the slot s that executes the task), and the scheduling timing on the slot ($wc_s^{p,j}$), to minimize the sum of the completion time $\sum_j \mathcal{C}_j$. Geo-TORS (Task Offloading and Resource Scheduling for Geo-distributed Jobs) refers to this problem.

3.2 Problem Formulation

Since the offload occurs after the data upload is completed, this paper naturally disassembles the Geo-TORS problem into two problems, which are the optimization of job data upload time (problem I) and the optimization of job offload and calculation time (problem II).

Problem I: Optimize the Upload Time of Jobs' Data. This sub-question determines the data upload completion time T pj for each task of the job. By

arranging the task upload sequence, the sum of these job upload times is optimized. Its formal definition is as follows:

$$\min \sum_{j=1}^{n} \mathcal{T}_j \tag{3}$$

$$\text{s.t. } \mathcal{T}_j \geq \mathcal{T}_j^p, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{4}$$

$$\mathcal{T}_j^p \geq r_j, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{5}$$

$$\mathcal{T}_k^p \geq \mathcal{T}_j^p + \frac{v_k^p}{B_p} \text{ or } \mathcal{T}_j^p \geq \mathcal{T}_k^p + \frac{v_j^p}{B_p}, \qquad \forall p \in \mathcal{P}, \forall j, k \in \mathcal{J} \tag{6}$$

Among them, $\mathcal{T}_j$ is the data uploading completion time of job j, and constraint (6) indicates that the task data is uploaded in strict order and no preemption is allowed.

Problem II: Optimize Jobs' Offloading and Calculation Time. After the data upload is complete, the task u_j^p waits for offload and processing. Let $r_j^p = \mathcal{T}_j^p$ be the time when the task u_j^p can start execution. At time r_j^p, select a computing slot $s \in \mathcal{S}_p$ to offload the task u_j^p, and use variables $x_s^{p,j} \in \{0, 1\}$ to represent the task's offloading decision. Then arrange the processing order of this task on slot s, and finally optimize the sum of the completion time of the whole job. Its formal definition is as follows:

$$\min \sum_{j=1}^{n} \mathcal{C}_j \tag{7}$$

$$\text{s.t. } \sum_{s \in \mathcal{S}_p} x_s^{p,j} = 1, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{8}$$

$$\mathcal{C}_j \geq \mathcal{C}_j^p, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{9}$$

$$\mathcal{C}_j^p \geq r_j^p, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{10}$$

$$\mathcal{C}_k^p \geq \mathcal{C}_j^p + \sum_{s \in \mathcal{S}_p} x_s^{p,j} d_s^{p,k} \text{ or } \mathcal{C}_j^p \geq \mathcal{C}_k^p + \sum_{s \in \mathcal{S}_p} x_s^{p,j} d_s^{p,j}, \qquad \forall p \in \mathcal{P}, \forall j, k \in \mathcal{J} \tag{11}$$

Among them, the constrained formula (8) ensures that a task can and must be offloaded to a slot. Constraint (11) shows that task execution strictly follows the determined scheduling sequence, and preemption is not allowed.

Theorem 1. *The Geo-TORS problem is NP-hard.*

Proof. If the coupling between tasks in this problem is neglected(I.e. tasks belonging to the same job jointly determine the completion time of this job), the problem can be simplified to the problem of offloading and resource scheduling of independent tasks in literature [22]. It has been proved that the offloading and resource scheduling of independent tasks are NP-hard. Therefore, the Geo-TORS problem is at least NP-hard.

4 The SmartDis Algorithm

This paper designs tasks offloading and scheduling algorithm SmartDis. The algorithm's brief overview is that when a distributed execution job is released, SmartDis first arranges the transmission order for the data upload request of the job at each edge AP based on the primary dual method, to minimize the global data transmission delay (GeoOrder component). When the data transmission of a certain task is completed, SmartDis immediately selects the offload slot for the task based on the principle of the smallest value added of the sum of the completion time, and arranges the scheduling timing for tasks on the slot (GeoSRPT+LeastDelta component).

Offloading the task immediately is due to the following two considerations. First, the main scheduler has limited memory. If a large number of tasks are cached in the task pool and only the resources are idle, the scheduling will be performed. During this process, the data in the memory may be frequently switched, which will cause extra time. The second is that in a heterogeneous server cluster, the execution time of tasks on each server is independent. In the face of different idle slots, the priority of tasks changes. If the task pool model is adopted, no matter whether the scheduling is performed regularly or the scheduling is performed after idle resources are available, the scheduling algorithm needs to be executed once for the entire task pool. This is time-consuming.

4.1 Determining Upload Orders

Based on the primary-dual method, when a new job is released, the SmartDis algorithm performs job sequencing for each edge AP's data upload requests, and each edge AP should transmit data according to the ordered job sequence. The transmission ordering strategy GeoOrder proposed in this paper is near optimal, which ensures that the data transmission delay does not exceed 3 times the optimal value.

The main challenge of GeoOrder design is how to capture the impact of the data transmission between jobs in the system under the constraints of upload bandwidth at each edge. This paper achieves this goal based on the job weight scaling step derived from the primary-dual design framework. The execution steps of GeoOrder are summarized as follows in Algorithm 3. Later, this paper introduces the design process of GeoOrder based on the primary-dual framework and the performance analysis of GeoOrder in detail.

Algorithm 1: GeoOrder-simplified version

Input: $\mathcal{J}$ (unordered job set)
$\tilde{v}_j^p$ (Amount of data not uploaded by job j at edge p)
w_j (The weight of job j)
Output: σ (Assignment of jobs)

1 **for** $k = n$ *to* 1 **do**
2 $b \leftarrow \arg\max_p \sum_{j \in \mathcal{J}} \tilde{v}_j^p$ (Find bottleneck AP)
3 $\sigma(k) \leftarrow \arg\min_{j \in \mathcal{J}} \frac{w_j}{\tilde{v}_j^p}$ (Select the job with the largest amount of unuploaded data after weighting)
4 $w_j \leftarrow w_j - w_{\sigma(k)} \times \frac{\tilde{v}_j^p}{\tilde{v}_{\sigma(k)}^p}$ (Job weight scaling)
5 $\mathcal{J} \leftarrow \mathcal{J} \backslash \sigma(k)$ (Update job collections that have not been scheduled)
6 **return** σ

GeoOrder (1) finds the bottleneck AP, that is, the edge AP with the largest amount of unuploaded data (Algorithm 4.1, line 2) (2) Selects the job with the largest amount of unuploaded data among all upload requests of the bottleneck AP. And place the job at the end of all out-of-order jobs (Algorithm 4.1 line 3) (3) scale the weight of all out-of-order jobs to capture the impact of the job sequencing operation in step (2) on the data transmission delay of the remaining jobs (Algorithm 4.1 line 4). Repeat the above steps to sequence all jobs.

The design process of the primary dual algorithm and GeoOrder performance analysis. Let the variable $\mathcal{T}_j^p = r_j + wn_j^p + \frac{v_j^p}{\mathcal{B}_p}$ be the time when the data transfer of job j is completed at the edge p, and the variable $\mathcal{T}_j$ be the time when the data transfer of job j is completed. w_j is the weight of job j, and the default value is 1. With the goal of minimizing the weighted data transmission time, the formal data flow scheduling problem is as follows:

$$\min \sum_{j=1}^{n} w_j \mathcal{T}_j \tag{12}$$

$$\text{s.t. } \mathcal{T}_j \geq \mathcal{T}_j^p, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{13}$$

$$\mathcal{T}_j^p \geq r_j, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{14}$$

$$\mathcal{T}_k^p \geq \mathcal{T}_j^p + v_k^p \text{ or } \mathcal{T}_j^p \geq \mathcal{T}_k^p + v_j^p, \qquad \forall p \in \mathcal{P}, \forall j, k \in \mathcal{J} \tag{15}$$

In this paper, the nonlinear Primal problem (Eq. (15) is a nonlinear constraint) is further rewritten as the linear programming LP-Primal. Equation (20) is similar to the linear constraints introduced by Wolsye [25] and Queyranne [17] for the One-Machine-N-Job scheduling problem:

$$\sum_{j \in \mathcal{S}} p_j \mathcal{C}_j \geq \frac{1}{2} \left[\left(\sum_{j \in \mathcal{S}} p_j \right)^2 + \sum_{j \in \mathcal{S}} p_j^2 \right], \forall \mathcal{S} \subseteq [n] \tag{16}$$

The variable $\mathcal{C}_j$ is the completion time of job j , p_j is the processing time of job j , $[n] = \{1, 2, ..., n\}$. Queyranne [17] shows that the convex hull of a feasible solution

to a completion time vector in a scheduling problem can be fully described by this linear constraint.

$$\min \sum_{j=1}^{n} w_j \mathcal{T}_j \tag{17}$$

$$\text{s.t. } \mathcal{T}_j \geq \mathcal{T}_j^p \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{18}$$

$$\mathcal{T}_j^p \geq r_j, \qquad \forall p \in \mathcal{P}, \forall j \in \mathcal{J} \tag{19}$$

$$\sum_{j \in \mathcal{S}} v_j^p \mathcal{T}_j^p \geq \frac{1}{2} \mid \left[\sum_{j \in \mathcal{S}} v_j^p \right]^2 + \sum_{j \in \mathcal{S}} \left(v_j^p \right)^2 \qquad \forall p \in \mathcal{P}, \forall \mathcal{S} \subseteq [n] \tag{20}$$

This paper considers the duality problem of the LP-Primal problem and analyzes the performance of the data flow scheduling algorithm proposed next in this paper. This paper introduces the dual variable α_j^p for the constrained expression (18), the dual variable γ_j^p for the constrained expression (19), and the dual variable β_S^p for the constrained expression (20). The LP-Dual problem is as follows:

$$\max \sum_p \sum_j \gamma_j^p r_j + \frac{1}{2} \sum_S \sum_p \beta_S^p \left[\left(\sum_{j \in S} v_j^p \right)^2 + \sum_{j \in S} \left(v_S^p \right)^2 \right] \tag{21}$$

$$\text{s.t. } \sum_p \alpha_j^p \leq w_j \qquad \forall j \in [n] \tag{22}$$

$$\sum_{S \ni j} \beta_S^p v_j^p \leq \alpha_j^p - \gamma_j^p, \qquad \forall p \in \mathcal{P}, \forall j \in [n] \tag{23}$$

$$\alpha_j^p \geq 0, \gamma_j^p \geq 0, \qquad \forall p \in \mathcal{P}, \forall j \in [n] \tag{24}$$

$$\beta_S^p \geq 0, \qquad \forall p \in \mathcal{P}, \forall S \subseteq [n] \tag{25}$$

Primal-Dual algorithm GeoOrder full version description see Algorithm 4.2 for details. Algorithm 4.1 has the following simplifications in Algorithm 4.2, so that it can run at a certain time (specifically the time when a new job is released) to sequence all jobs that have not uploaded data in the current system: (1) Ignore the job release time r_j (2) The input is the amount of data that the job has not uploaded at each moment on each edge.

For the convenience of analysis, after GeoOrder calculates the job arrangement σ, this paper renumbers the job so that $\sigma(k) = k, \forall k \in [n]$. Next, this paper transfers data according to the sequence of jobs generated by GeoOrder. The approximate ratio of the weighted data transmission completion time ($\sum_{j=1}^{n} w_j \mathcal{T}_j$) to the optimal transmission completion time ($\sum_{j=1}^{n} w_j \mathcal{T}_j^{OPT}$) is less than or equal to 3.

Algorithm 2: GeoOrder

Input: $\mathcal{J}$ (unordered job set)
v_j^p (Amount of data not uploaded by job j at edge p)
w_j (The weight of job j)
Output: σ (Assignment of jobs)

for $k = n$ *to* 1 **do**
  $b_k \leftarrow \arg\max_p \sum_{j\in\mathcal{J}} v_j^p$ (Find bottleneck AP)
  $r_{max} \leftarrow \max_{j\in\mathcal{J}} r_j$ (Find the latest release time)
  if $r_{max} \leq \frac{1}{2}\sum_{j\in\mathcal{J}} v_j^{b_k}$ **then**
    $\sigma(k) \leftarrow \arg\min_{j\in\mathcal{J}} \frac{w_j - \sum_{l>k}\beta_l v_j^{b_l}}{v_j^{b_k}}$ (Select the job with the largest amount of unuploaded data after weighting)
    $\beta(k) \leftarrow \frac{w_{\sigma(k)} - \sum_{l>k}\beta_l v_j^{b_l}}{v_{\sigma(k)}^{b_k}}$ (Update related parameters)
  else
    $\sigma(k) \leftarrow \arg\max_{j\in\mathcal{J}} r_j$ (Select the job with the closest arrival time)
    $\gamma_{\sigma(k)} \leftarrow w_{\sigma(k)} - \sum_{l>k}\beta_l v_{\sigma(k)}^{b_l}$ (Update related parameters)
  $\mathcal{J} \leftarrow \mathcal{J}\backslash\sigma(k)$ (Update job collections that have not been scheduled)
return σ

Theorem 2. *If the order generated by the GeoOrder algorithm is used,*

$$\sum_{j=1}^{n} w_j \mathcal{T}_j \leq 3\sum_{j=1}^{n} w_j \mathcal{T}_j^{OPT}$$

Proof. We first construct a feasible solution for the LP-Dual problem and compare the performance between SOL_{Dual} of the feasible solution with SOL_{Primal} of the LP-Primal solution. Since SOL_{Primal} equals $\sum_{k=1}^{n} w_k T_k$, we can get the following equation according to weak duality theorem,

$$SOL_{Dual} \leq OPT_{Dual} \leq OPT_{Primal}$$

Thus, we can show that the primal solution of SOL_{Primal} is within a constant times of the optimal solution.

4.2 Computing Slot Selection and Task Scheduling

After the data transfer is completed, the task is ready to perform data processing. Let $r_j^p = \mathcal{T}_j^p$ be the time at which the data processing task u_j^p of job j at edge p can start execution. Let $\mathcal{S}_p = 1, 2, ..., m$ be the server cluster of edge p. At time r_j^p, select a computing slot $s \in \mathcal{S}_p$ to offload the data processing task u_j^p, and arrange the scheduling timing of tasks on slot s.

With the goal of minimizing the sum of the completion delays, this paper proposes a compute slot selection strategy (LeastDelta) and a task scheduling

strategy on the slot (GeoSRPT). Each edge cluster executes the same slot selection strategy. When the data transmission of a task is completed, it selects the computing slot for the task; each computing slot executes the same task scheduling strategy and sequentially processes the offloaded computing tasks.

Algorithm 3: LeastDelta&GeoSRPT

1 Node Selection (LeastDelta): When the data transmission of task μ_j^p is completed (that is $t = r_j^p$), we offload the task to the node $s^* = \arg\min_{s \in S_p} Q_{sj}^p(t)$

2 Tasks scheduling on a node (GeoSRPT): At any time t', the node s performs task $\mu_{j^*}^p$, $j^* = \arg\min_{j \in \mathcal{A}_J t^*} u_s^{p,j}(t')$

Computing Task Scheduling Strategy (GeoSRPT). For single-task jobs (or in more detail, for One-Server-One-Queue scheduling problems and preemption is allowed). The SRPT (Shortest Remaining Processing Time) scheduling strategy is optimal [20] (with the goal of minimizing the completion time of weighted jobs). Based on SRPT and considering the multi-task job distributed execution mode (Geo-execution), this paper designs GeoSRPT scheduling strategy.

Use $d_s^{p,j}(t)$ to denote the remaining execution time at time t after the task u_j^p is offloaded to slot s. Let $u_s^{p,j}(t) = \max_(i \in \mathcal{P}) d_j^i(t)$ denote at least the remaining execution time of the job to which the task belongs at time t. $d_j^i(t)$ is defined as:

$$d_j^i(t) = \begin{cases} d_{s'}^{i,j}(t), \text{ If the task } \mu_j^i \text{ has been offloaded to the node } s' \in \mathcal{S}_i. \\ t - r_j^i + d_{s'}^{i,j} s'' = \arg\min_{s \in \mathcal{S}_i} d_s^{i,j}, \text{ If the task } \mu_j^i \text{ has not been offloaded.} \end{cases}$$

At time t, for tasks that have not been completed on slot s (denoted by the set $\mathcal{A}_s(t)$), and the task u_j^p with the smallest $u_s^{p,j}(t)$ value is executed.

Slot Selection Strategy (LeastDelta). When $t = r_j^p$, the data transmission of a task is completed, and an execution slot needs to be selected for the task. LeastDelta's slot selection principle is to offload the task to a certain slot, so that the increase in the sum of the delays in the completion of the job is minimal. When the task u_j^p is offloaded to a slot s, and it is assumed that no new task will be offloaded to the slot s after time t, the increase in the overall job completion time is composed of three parts:

1. Because there are some other tasks (Type-I tasks) with a smaller $u_s^{p,j}$ value than task u_j^p on slot s, the waiting time of task u_j^p increases.
2. The processing time of the task itself.
3. Other tasks (Type-II tasks) that have a larger $u_s^{p,j}$ value than task u_j^p existing on slot s have an additional waiting time due to the execution of task u_j^p.

When no new task is offloaded to slot s after time t, the completion time of task u_j^p can be calculated and expressed as $t_{j,s}^p$(available by simulating task scheduling on slot s, the scheduling strategy is GeoSRPT). Let $c_j(t)$ record the time t, which is the maximum estimated completion time in the offloaded tasks

of job j. The maintenance of the $c_j(t)$ value will be described in detail after each offloading decision.

Let $\mathcal{A}^p_{sj}$ be the set of tasks that reach slot s at time t and have not been completed. The definition is as follows,

$$\mathcal{A}^p_{sj} = \left\{ j' \mid s^p_{j'} = s, r^p_{j'} \leq t, d^p_{j'}(t) > 0 \right\} \tag{26}$$

Among them, $s^p_{j'}$ is the slot selected for the task $u^p_{j'}$. $\mathcal{A}^p_{sj}(t)$ contains Type-I and Type-II tasks affected by task u^p_j.

Let $mathcalB1^p_{j,s}(t)$ denote the type-I task set affected by the task u^p_j, defined as follows,

$$\mathcal{B}1^p_{j,s}(t) = \left\{ j' \mid j' \in \mathcal{A}^p_{sj} : u^{p,j'}_s(t') \leq u^{p,j}_s(t') \right\} \tag{27}$$

Let $mathcalB2^p_{j,s}(t)$ denote the type-II task set affected by the task u^p_j, defined as follows,

$$\mathcal{B}2^p_{j,s}(t) = \left\{ j' \mid j' \in \mathcal{A}^p_{sj} : u^{p,j'}_s(t') \leq u^{p,j}_s(t') \right\} \tag{28}$$

Based on the above symbols, when the task u^p_j is offloaded to the slot s at $t = r^p_j$, the calculation value of the increase in the completion time of the overall job $\mathcal{Q}^p_{sj}(t)$ is as follows,

$$\mathcal{Q}^p_{sj}(t) = \max \left\{ \sum_{j' \in \mathcal{B}1^p_{j,s}(t)} d^p_{j'}(t) + d^p_j - c_j(t), 0 \right\} \tag{29}$$

$$+ \sum_{j' \in \mathcal{B}^p_2,(t)} \max \left\{ d^p_j - c_{j'}(t), 0 \right\} \tag{30}$$

Among them, the formula (29) is the increase of the completion time of the task u^p_j caused by the Type-I task, and the formula (30) is the increase of the completion time of the Type-II task caused by the task u^p_j. LeastDelta selects the slot $s^* = \arg\min_{s \in \mathcal{S}_p} \mathcal{Q}^p_{\mathcal{S}j}(t)$ for the task u^p_j to perform the calculation. After offloading, update the c_j value of the job to which the related task belongs. Update the c_j value for the job to which the task u^p_j belongs, as follows,

$$c_j = \max \left\{ c_j, t + \sum_{j' \in \mathcal{B}1^p_{j,s^*}(t)} d^p_{j'}(t) + d^p_j \right\}. \tag{31}$$

Update the c_j value for the job of Type-II tasks $u^p_{j'} | j' \in \mathcal{B}2^p_{j,s^*}(t)$ on slot s^*, as follows,

$$c_j = \max \left\{ c_{j'}, t + t^p_{j',s^*} + d^p_j \right\}. \tag{32}$$

5 Experiments

In this section, we simulate job specifications in an actual commercial cluster environment and compared to several other scheduling mechanisms, including traditional scheduling mechanisms (FCFS and SRPT), recent scheduling mechanisms designed for concurrent execution of multiple data centers (Reordering and SWAG), and SmartDis mechanism in this paper. Then we conduct evaluations on execution performance (Sect. 5.2), fairness (Sect. 5.3), sensitivity (Sect. 5.4), and scheduling execution cost (Sect. 5.5).

5.1 Experimental Setup

System Scale: This paper expands the CloudSim simulator to simulate the multi-edge system, with the number of edges ranging from 20 to 500. The main evaluation experiment of this paper was run in a system environment with 100 edges and 3000 servers. The number of servers is scaled according to the ratio of the number of servers to the number of edges (for example, a system with 50 edges deployed has $3000/100 \times 50 = 1500$ servers).

Server Distribution: In order to evaluate the impact of the difference in edge size, this paper models the skewness of the server distribution between edges based on the Zipf distribution. When there is no skew, the server is evenly distributed to each edge. As the skew parameter of the Zipf distribution is higher, the degree of skew of the server's distribution between the edges is greater . The default setting of the skew parameter is 2.

Job Specifications: This paper synthesizes simulated experimental loads based on the job size specifications in Facebook's commercial Hadoop cluster and Google's working cluster [11]. Both types of workloads are dominated by small jobs. This paper is also based on high-performance computing clusters to synthesize large-scale jobs. Refer to Table 1 for details of the three types of loads. This paper adjusts the job release interval based on the Poisson process to keep the utilization rate of each load system consistent.

Table 1. Load specifications

Composition ratio	Small job (1–150)	Medium job (151–500)	Large job (501+)	Average job size (task numbers)
Facebook-like	89%	8%	3%	241
Google-like	96%	3%	2%	94
HPC-like	18%	29%	53%	582

Data Distribution: To evaluate the impact of job data distribution on scheduling algorithms, this paper uses Zipf distribution to model the skewness of job data distribution between edges. As the skew parameter of the Zipf distribution is higher, the distribution of data between edges is more skewed. There are two

extreme cases. One is that the job data is evenly distributed on the relevant edges, and the other is that the job data is concentrated on a single edge. The default setting of the skew parameter is 2.

Task Duration: This paper models job duration based on Pareto distribution with $\beta = 1.259$, and the average task duration is 2 s. This model is consistent with the fitting of the task duration in Facebook cluster in [3].In the simulation experiment, the server performance was randomly sampled from the normal distribution. In order to show the heterogeneity of server performance between edges, this paper sets different mean and variance parameters for the normal distribution model based on the edge scale. For parameter setting, refer to the results of virtual machine performance measurement in the Amazon cloud [7,19].

Evaluation Criteria: The main performance indicator that this paper focuses on is the average completion time of the job. In addition, this paper focuses on the degree of delay in the execution of the job, that is, the ideal execution time without waiting divided by the actual execution time, as an indicator to measure the fairness between the jobs.

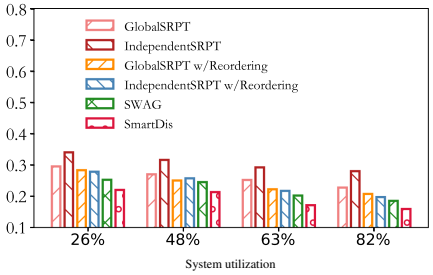
The SmartDis in this paper is compared with the First Come First Schedule strategy(FCFS),Global Shortest Remaining Processing Time (GlobalSRPT), Independent Shortest Remaining Processing Time(IndependentSRPT), and these two types of SRPT scheduling strategies adjusted by Reordering, and SWAG Scheduling strategies. In this paper,the results of FCFS are used to standardize the results of the remaining scheduling algorithms under the same experimental environment setting.

5.2 Execution Performance

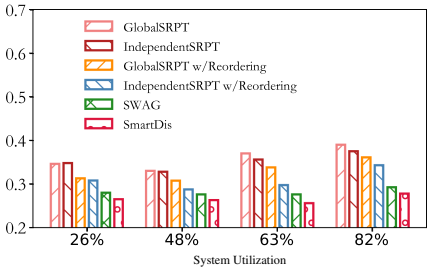
Figure 3 shows the average completion time of each scheduling algorithm under different workload specifications (Facebook-like see Fig. 3(a), Google-like see Fig. 3(b), HPC-like see Fig. 3(c)). The experimental results show that SmartDis performs better than other scheduling strategies. Compared with the heuristic scheduling strategy based on SRPT, SmartDis performance is improved by 33% (Facebook-like), 25% (Google-like) and 27% (HPC-like) at high utilization rate (82%). At low utilization (26%), the performance improvement is at least 15% more. SmartDis chooses to execute the job that minimizes the increase in the overall execution delay by sensing the imbalance between the upload and calculation requirements of each subtask of the job. Compared with the Reordering and SWAG strategies, SmartDis's performance improvement is better at the meticulous processing of the edge scale and server performance heterogeneity. Under various load conditions, SmartDis's performance improvement advantage remains above 10%.

5.3 Fairness

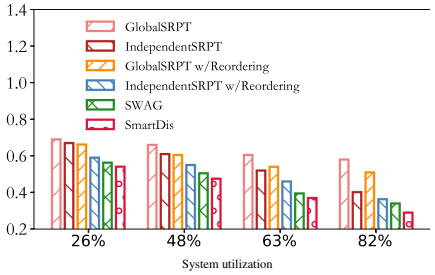
Figure 4 shows the degree of delay of jobs of different size categories under each scheduling algorithm, in order to show the degree of fairness between jobs. Since the job delay in FCFS is too large compared to other algorithms, it is ignored



(a) Simulate Facebook business Hadoop cluster job specifications



(b) Simulate Google cluster job specifications

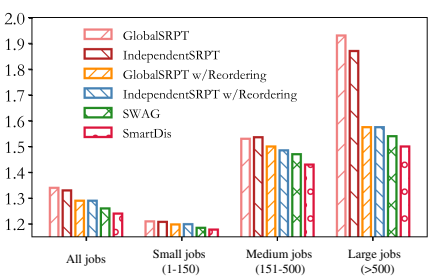


(c) Simulation of high-performance computing cluster job specifications

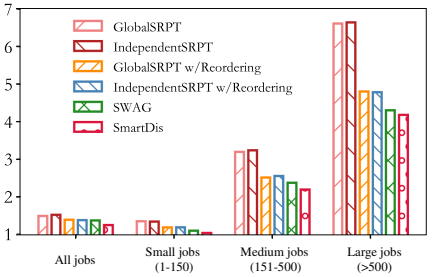
Fig. 3. Comparison of execution performance of various scheduling algorithms under different job specifications

in the results. Overall, the delay of small jobs is the smallest and the delay of large jobs is the largest under all scheduling methods.Because all algorithms essentially reduce the average job completion time by prioritizing the small jobs.

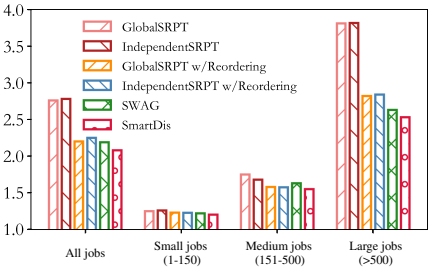
The difference in fairness is mainly reflected in the difference in the degree of delay in large jobs. First, it is observed that GlobalSRPT and IndependentSRPT are almost identical in maintaining fairness, so the following will only analyze the gap between IndependentSRPT and other scheduling algorithms. From the perspective of load type, the gap between the delay of large jobs in the Google-like load and the overall job delay is more significant. This is because almost all jobs in the Google-like are small jobs, resulting in a small number of large jobs being queued for too long due to the preference for small jobs. Even so, the big jobs in the SmartDis still maintain relatively low latency compared to others. In Facebook-like and HPC-like, the delay of large jobs under the IndependentSRPT is at least 38% higher than the delay of their overall jobs. While, IndependentSRPT's large job delay adjusted by Reordering is controlled to not exceed 28%. SWAG can control this gap to not exceed 23%, and SmartDis further controls this gap to within 20%. Therefore, it can be concluded that SmartDis does not significantly sacrifice the performance of large jobs when improving performance.



(a) Simulate Facebook business Hadoop cluster job specifications



(b) Simulate Google cluster job specifications



(c) Simulation of high-performance computing cluster job specifications

Fig. 4. Fairness comparison of scheduling algorithms under different job specifications

5.4 Sensitivity

The Impact of the Degree of Data Dispersion: Figure 5 presents the general trend that as the data skewness increases, the performance of the algorithm increases first and then decreases. When the job data is evenly distributed on the relevant edges, there is less room for optimization. When the data distribution starts to be unbalanced, multi-edge job scheduling collaboration can reduce job completion time. However, when the skewness exceeds a certain level, the imbalance of data distribution becomes so severe that most of the data of the same job is only distributed on a few edges. In this case, too much collaborative work is not required.

The Impact of Edge Scale Differences: Figure 6 shows that with the increase in edge scale heterogeneity, SmartDis performance has always maintained an advantage, and compared with other algorithms, SmartDis is more sensitive to edge scale differences. This is because SmartDis's scheduling proactively senses the imbalance in the execution delay between sibling tasks due to the difference in the overall service capabilities of the edge caused by the difference in the number of servers on the edge, and is committed to reducing the increase in global completion time caused by the imbalance.

The Effect of the Number of Edges: In Fig. 7, as the number of edges increases, the performance improvement of Reordering, SWAG, and SmartDis all show an increasing trend, because more edges provide more coordination opportunities for execution.

The Effect of Task Duration Estimation Accuracy: In the experiment, the estimation error is introduced based on the uniform distribution with the original task duration as the mean. The results in Fig. 8 show that with the increase in the accuracy of task duration estimation, the performance of each algorithm will be slightly improved, but it is not obvious. This is because due to the existence of various interference factors in the execution process, the original set time of the task is originally quite different, so the estimation error is not enough to seriously affect the scheduling decision. SmartDis maintains the best performance under different estimation accuracy, and it is robust to the estimation error of task duration.

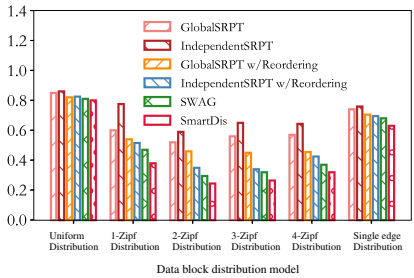


Fig. 5. The influence of the degree of data dispersion on each scheduling algorithm

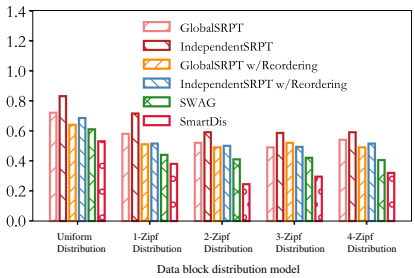


Fig. 6. Influence of the difference in size between edges on each scheduling algorithm

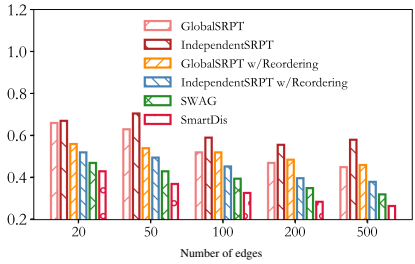


Fig. 7. Influence of the number of edges on each scheduling algorithm

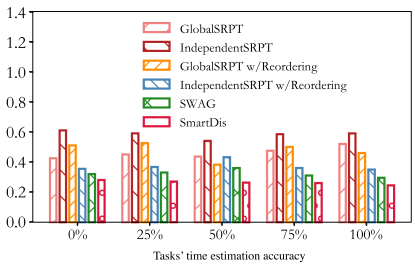
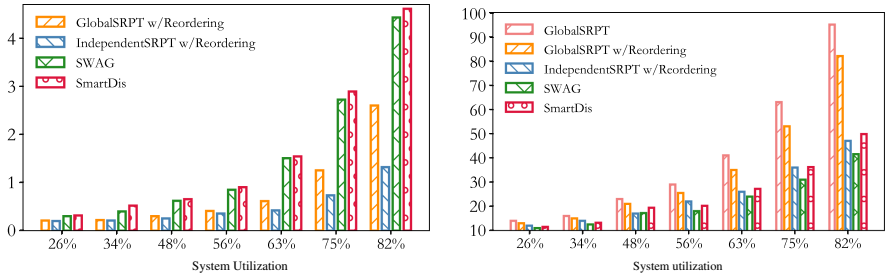


Fig. 8. Influence of the accuracy of task time estimation on each scheduling algorithm

5.5 Scheduling Execution Cost

Comparison of Running Time: The experiment measures the length of time to run the algorithm at each scheduling decision point. Figure 9(a) shows the running time of the algorithm under different system utilization rates. FCFS, GlobalSRPT, and IndependentSRPT are not shown in the results due to their minimal running time compared to others. The results show that even at high utilization rates (82%), Smart's scheduled runtime (4.61 ms) is much smaller than the average task duration (2 s).

Comparison of Extra Traffic: Extra traffic is defined as the information required by the scheduling algorithm to be transmitted from each edge to the global scheduler. Note that this does not include the basic and necessary information required for jobs. Figure 9(b) shows the traffic of each scheduling algorithm. FCFS and IndependentSRPT do not require the edge to provide any other information to the global scheduler, so their traffic is zero. The amount of traffic essentially depends on the current number of jobs in the system. SmartDis does its best to schedule jobs that can be completed quickly, keeping the number of jobs blocked in the system at a low level, so its traffic is acceptable.



(a) Comparison of running time of various scheduling algorithms

(b) Comparison of traffic volume of various scheduling algorithms

Fig. 9. Fairness comparison of scheduling algorithms under different job specifications

6 Conclusion

In the era of big data, the amount of data continues to grow at an alarming rate. This paper emphasizes the network and computing resource competition problems encountered by cross-domain big data analysis jobs in the edge environment. A resource coordination algorithm SmartDis is proposed to schedule subtasks across regions on multiple edges. SmartDis can achieve a near-optimal average completion time. Furthermore, the time-consuming transmission can prove the efficient approximate ratio. This paper conducts extensive experiments based on job execution specifications in real clusters to evaluate the performance of SmartDis in a wide range of scenarios. Compared with the heuristic scheduling

algorithm based on the SRPT class, SmartDis improves the average completion time up to 33%, and keeps the additional calculation and communication overhead low.

References

1. Aazam, M., Huh, E.N.: E-HAMC: Leveraging fog computing for emergency alert service. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 518–523. IEEE (2015)
2. Anand, S., Garg, N., Kumar, A.: Resource augmentation for weighted flow-time explained by dual fitting. In: Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, pp. 1228–1241. SIAM (2012)
3. Ananthanarayanan, G., Hung, M.C.C., Ren, X., Stoica, I., Wierman, A., Yu, M.: GRASS: trimming stragglers in approximation analytics. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2014), pp. 289–302 (2014)
4. Asif-Ur-Rahman, M., et al.: Toward a heterogeneous mist, fog, and cloud-based framework for the internet of healthcare things. IEEE Internet Things J. **6**(3), 4049–4062 (2018)
5. Chadha, J.S., Garg, N., Kumar, A., Muralidhara, V.: A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 679–684 (2009)
6. Chen, N., Chen, Y., Song, S., Huang, C.T., Ye, X.: Smart urban surveillance using fog computing. In: 2016 IEEE/ACM Symposium on Edge Computing (SEC), pp. 95–96. IEEE (2016)
7. Dejun, J., Pierre, G., Chi, C.-H.: EC2 performance analysis for resource provisioning of service-oriented applications. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave-2009. LNCS, vol. 6275, pp. 197–207. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16132-2_19
8. Garg, N., Kumar, A.: Minimizing average flow-time: Upper and lower bounds. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 603–613. IEEE (2007)
9. Garg, N., Kumar, A., Pandit, V.: Order scheduling models: hardness and algorithms. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 96–107. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77050-3_8
10. Hajjat, M., Maltz, D., Rao, S., Sripanidkulchai, K.: Dealer: application-aware request splitting for interactive cloud applications. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 157–168 (2012)
11. Hung, C.C., Golubchik, L., Yu, M.: Scheduling jobs across geo-distributed datacenters. In: Proceedings of the Sixth ACM Symposium on Cloud Computing, pp. 111–124 (2015)
12. Im, S., Moseley, B.: An online scalable algorithm for minimizing lk-norms of weighted flow time on unrelated machines. In: Proceedings of the Twenty-sScond Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 95–108. SIAM (2011)
13. Mastrolilli, M., Queyranne, M., Schulz, A.S., Svensson, O., Uhan, N.A.: Minimizing the sum of weighted completion times in a concurrent open shop. Oper. Res. Lett. **38**(5), 390–395 (2010)

14. Mohan, P., Thakurta, A., Shi, E., Song, D., Culler, D.: GUPT: privacy preserving data analysis made easy. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 349–360 (2012)
15. Mutlag, A.A., Abd Ghani, M.K., Arunkumar, N.A., Mohammed, M.A., Mohd, O.: Enabling technologies for fog computing in healthcare IoT systems. Future Gener. Comput. Syst. **90**, 62–78 (2019)
16. Pu, Q., Ananthanarayanan, G., Bodik, P., Kandula, S., Akella, A., Bahl, P., Stoica, I.: Low latency geo-distributed data analytics. ACM SIGCOMM Comput. Commun. Rev. **45**(4), 421–434 (2015)
17. Queyranne, M.: Structure of a simple scheduling polyhedron. Math. Program. **58**(1–3), 263–285 (1993). https://doi.org/10.1007/BF01581271
18. Roemer, T.A.: A note on the complexity of the concurrent open shop problem. J. Sched. **9**(4), 389–396 (2006). https://doi.org/10.1007/s10951-006-7042-y
19. Schad, J., Dittrich, J., Quiané-Ruiz, J.A.: Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. **3**(1–2), 460–471 (2010)
20. Schrage, L.: Letter to the editor-a proof of the optimality of the shortest remaining processing time discipline. Oper. Res. **16**(3), 687–690 (1968)
21. Singh, S., Singh, N.: Internet of things (ToT: Security challenges, business opportunities and reference architecture for e-commerce. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), pp. 1577–1581. IEEE (2015)
22. Tan, H., Han, Z., Li, X.Y., Lau, F.C.: Online job dispatching and scheduling in edge-clouds. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
23. Vulimiri, A., Curino, C., Godfrey, P.B., Jungblut, T., Padhye, J., Varghese, G.: Global analytics in the face of bandwidth and regulatory constraints. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2015), pp. 323–336 (2015)
24. Vulimiri, A., Curino, C., Godfrey, P.B., Jungblut, T., Karanasos, K., Padhye, J., Varghese, G.: Wanalytics: Geo-distributed analytics for a data intensive world. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1087–1092 (2015)
25. Wolsey, L., et al.: Formulating single machine scheduling problems with precedence constraints. Université catholique de Louvain, Center for Operations Research, Technical report (1989)