Computers and Electrical Engineering 61 (2017) 327–338

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Profit-aware scheduling in task-level for datacenter networks

Xiaoyi Tao, Heng Qi*, Wenxin Li, Keqiu Li, Yang Liu

Dalian University of Technology, China

ARTICLE INFO

Article history:
Received 31 January 2016
Revised 22 July 2016
Accepted 25 July 2016
Available online 16 September 2016

Keywords:
Data center networks
Task scheduling
Application performance
Maximize profit

ABSTRACT

Applications perform massive and diverse tasks in data centers. Tasks completion condition seriously affects application performance. However, most existing flow-level or task-level scheduling methods treat flows in isolation, meanwhile, few works discuss the efficiency of task-level scheduling from the perspective of the task profit.

In this paper, we introduce a profit-aware task-level scheduling scheme named PAT, whose target is to maximize the profit of completing tasks within their reasonable time. To this end, a maximizing profit optimization model is proposed on task-level, and an efficient approximate scheduling algorithm is presented. Furthermore, a situation of absent deadline information is discussed and an ePAT method is presented to solve this situation. Based on the proposed algorithm, we design and implement PAT and ePAT. Some comprehensive experiments are conducted to evaluate the performance of our methods. The experimental results show that our methods bring higher profit than other scheduling methods.

1. Introduction

Data centers are being used as the crucial computing and storage infrastructures for online services. Many Internet companies such as Google, Microsoft and Amazon, have their own data centers. Application performance is significant for these online services, on account of their strict limitation on latency requirements, and even a sub-second delay can seriously impact user experience and application performance [1]. Online applications generate a large number of requests and aggregate the responses computing in the back-end, since the result must wait for all of the responses to be finished or reduce application profit on user experience. This implies that a task is successfully accomplished if and only if all of its flows in one task completed before its deadline. Accordingly, task-level scheduling methods are significant for user experience and application profit in data center networks.

To the best of our knowledge, existing scheduling methods in data center networks can be classified into two categories: flow-level scheduling and task-level scheduling.

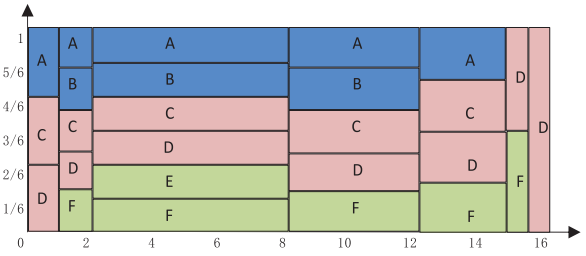
- Flow-level scheduling methods focus on minimizing completion time and finishing flows within deadline only based on flow-level information. Traditionally, fair sharing approaches approximately divide bandwidth equally on bottleneck in a fair share manner. The scheduling methods [2,3] are based on transport level rate control, whose targets are to reduce the number of flows missing deadlines inheriting the weakness of first come first serve (FCFS). Centralized flow priority scheduling methods [4,5] reduce flow completion time. To improve network resource efficiency, Hedera [6] dynamically schedules elephant flows and short flows.

* Corresponding author.
E-mail addresses: taoxiaoyi@mail.dlut.edu.cn (X. Tao), hengqi@dlut.edu.cn (H. Qi), liwenxin@mail.dlut.edu.cn (W. Li), keqiu@dlut.edu.cn (K. Li), yangliu.dlut@gmail.com (Y. Liu).

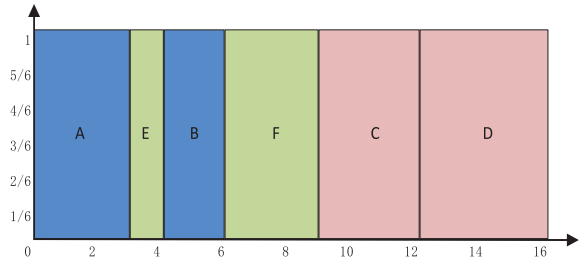
http://dx.doi.org/10.1016/j.compeleceng.2016.07.013
0045-7906/

Task ID	Flow ID	Size	Start time	Deadline
1	A	3	0	5
	B	2	1	
2	C	3	0	16
	D	4	0	
3	E	1	2	5
	F	3	1	

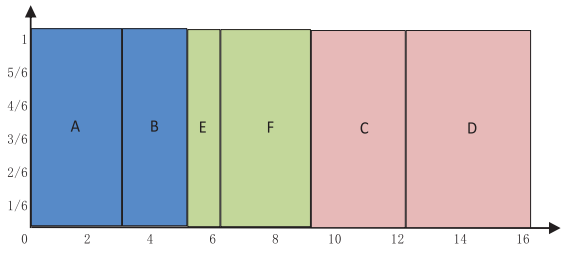
(a) The concurrent flows conditions for one bottleneck link



(b) Bandwidth fair share



(c) Flow-level priority scheduling



(d) Task-level priority scheduling

Fig. 1. An example of current works vs task-level schedule.

- Task-level scheduling methods organize flows of one task and schedule them together in order to reduce average completion time. Baraat [7] works on decentralized task-level scheduling and schedules flows in one task together. Coflow [8] abstracts the network plane and Varys [9] schedules coflows to reduce coflow completion time (CCT). As far as we know, however, no existing task-level mechanism is in place to guarantee the task completion for deadline.

Forementioned flow-level and task-level scheduling methods all neglect the integrity of a task. Since each task in data centers contains tens to hundreds of flows, all of which need to be finished before a task is considered to be completed. These scheduling methods could leading a situation that most flows in a task have been finished and be waiting for lag flows completion or return results immediately to users with performance damage.

Prior works either focus on reducing flow completion time or task completion time. Application performance is crucial for both users and data center organizations. Nonetheless both flow-level and task-level scheduling methods ignore the integrated task performance. To guarantee application performance, we firstly define profit as a variable to describe task performance. At the mean time, profit is a representation of task completion status and network efficiency, thus utilizing the task profit to ensure that task integrity is feasible. Additionally, flows belonging to diverse applications have different characteristics on flow sizes and latency. Naturally flows are capable to inheriting application properties. Thus, these have motivated task-level scheduling.

In this paper, we propose a task-level scheduling method to improve the task performance with the property of task profit. To simplify presentation, an example is presented to demonstrate the potential benefits of task-level scheduling methods comparing to existing flow-level methods. Task-level scheduling takes the task information into account for the integrity of tasks. As shown in Fig. 1, we suppose that there are three tasks with six flows. Each flow is represented as a 5-tuple [task ID, flow ID, size, starttime, deadline]. Meanwhile, we suppose that these flows are transferred in one bottleneck link, and the network capacity cannot satisfy all the flows deadline demands. In traditional solutions, fair sharing is the main method which is shown in Fig. 1(b). The completion time of these six flows are (44/3, 12, 44/3, 16, 8, 46/3) respectively in fair sharing manner. Comparing the finish time with deadline 1(a), it is clearly to figure out that only flow C and flow D meet their deadlines, as well as the average flow completion time is 13.4 by applying the fair sharing. In Fig. 1(c), flows are scheduled by priorities which are associated with deadlines. As shown in the Fig. 1(a), both tasks 1 and 3 miss their deadlines. Priority flow scheduling reduces the average completion time to 7.7. Compared to fair sharing method, it averagely saves 42% on completion time. Flow-level priority scheduling methods solely schedule flows on the basis of flow-level priority without task information. In this case, only one flow of tasks 1 and 3 is completed, which means they obtain zero profit from these tasks.

Comparing to flow-level scheduling, the task-level scheduling as shown in Fig. 1(d), the average flow completion time is 7.8 which is proximal to priority scheduling methods. However, the average task completion time of flow-level priority scheduling is 10 while task-level scheduling is 9.6. Moreover, task-level method reduces the number of switches among the

flows which might bring some lag during switching in this scenario. Task 1 meets its deadline in task-level scheduling, and obtains more profit than flow-level scheduling.

According to the above analysis, we design PAT (Profit-Aware Scheduling in Task-level), a method targeting at maximizing profit of completing tasks within their deadlines. PAT employs a centralized controller which has global task-level information including task number, requirement and deadline. The core of PAT is an $\frac{1}{2}$-approximation algorithm on maximizing profit on the basis of task time and task profit, which helps the controller make decisions on schedule policy. In PAT, the controller identifies the flows information, task information and making admission control on flows. Our algorithm gives a good attempt on guaranteeing task integrated. Through extensive simulation, we finds that PAT provides benefits over existing benefits on task completion rate, task profit and task completion time. Finally, we discuss a simpler situation to loose the maximum profit constraints to reveal the validity of PAT. Forementioned method schedules flows based on prior knowledge of flows and complex switch functions. We make attempts to schedule tasks with no prior knowledge. To this end, we present ePAT, an extension of PAT based on simple information during the working period.

In short, our main contributions of this paper are as follows:

- We address a task-level scheduling for application performance guarantee issue in data centers. Specially, we target a profit goal to guarantee the integrated task completion.
- We propose PAT and ePAT, a profit-aware method for task-level scheduling, which contains two key components: task identifier and profit calculator.
- We conduct comprehensive experiment to evaluate the performance of PAT and ePAT, to compare with existing priority flow scheduling solutions under typical data center traffic. The result shows that PAT and ePAT outperforms the flow-level priority scheduling in data center networks. PAT can save 15% task completion time on average, while increasing 20% task profit. The ePAT averagely saves 9% task completion time, and increasing 19%task profit.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the work model of PAT. In Section 3, we show the detailed design of PAT and ePAT. We present the evaluation in Section 4. In Section 5, we discuss the related works of this paper. Section 6 concludes the paper and our future work.

2. System model

In this section, we firstly present an overview of maximizing profit problem. Then, we describe the detail of our model.

2.1. Problem statement

Applications in data centers generally generate complex and a large number of tasks with multiple flows. From the nature of applications, each task has a strict limitation on deadline and all of its flows need to be finished before the task considered completed. In this way, the completion of integrated tasks is critical for application performance. So as to qualify the application performance, we define a profit to measure task completion quality. The dominated goal of our work is to guarantee the completion of integrated tasks in the form of maximizing profit on scheduling tasks. Generally a problem on task-level scheduling is difficult to solve. To schedule task efficiently, we resort to a centralized controller.

2.2. Model formulation

With the aim of guaranteeing application performance, we formulate a task-level model that mainly focuses on integrated of tasks in the scheduling. We assume that task size, deadline and priority information are known in our model. Meanwhile, each task in our model has an identifier denoted by the symbol t_i. The arriving time of task t_i is indicated by a_i, and d_i is the deadline of t_i. The duration of task t_i is the interval between a_i and d_i. We use $f_{i,j}$ to denote flows in task t_i, and n_i is the number of flows in t_i. For each $f_{i,j}$ belonging to task t_i, it has the same deadline d_i. Even so flows in one task have different arriving time. Each flow in tasks has a value on accomplishment which is presented by profit. Specially, if a flow accomplished before its deadline, the organizer will obtain the flow profit. In the mean time, we denote $\alpha_{i,j}(t)$ as a binary variable that indicates whether to schedule flow $f_{i,j}$ at current time slot. Here, we suppose that once a flow is scheduled before its deadline, it will be considered as completed. As the network resource is restricted, network capacity C can be allocated to the scheduled flows. Let $s_{i,j}$ indicate the size of flow $f_{i,j}$.

Considering the nature of tasks, each task has a weight ω_i which is associated with task value as well as the priority ρ_i of task. Regarding priority, realtime tasks have higher priority than background tasks. Due to the applications in data center clouds pay different prices for their resource. Service price and profit determined by the providers are relevant in this situation. According to our survey, in the cloud [10], mobile netorks[11] and Cyber-Physical systems[12], the price is based on the computation resource, service time and service type. In the meantime, service price is affected by the quality of experience which is application performance in this paper. Correspondingly, we define profit is a definition on provider service price based on service type. Hence the task profit belonging to these applications is affected by their prices which can be denoted by task weight ω and priority ρ . Consequently, the profit P_i of task t_i as shown in Eq. (1) is determined by task priority ρ_i and task weight ω_i.

$$P_i = \omega_i \rho_i. \tag{1}$$

Accordingly, flow profit $P_{i,j}$ is determined by two parts: task profit P_i and task completion condition for each flow. Task completion condition is important for the integrity of task performance. Here, the task completion condition is denoted by $\gamma_{i,j}$ which is affected by time urgency of task t_i. For each unaccomplished flows in task t_i, the value of $\gamma_{i,j}$ is the same at the current time slot. As well as $\gamma_{i,j}$ is a dynamical indicate over time slots. Flow profit $P_{i,j}$ can be dynamically modified according to current time and deadline, which means scheduling some urgent tasks' flows can bring in more profit.

$$P_{i,j} = \gamma_{i,j} P_i. \tag{2}$$

We define that profit is gained by successfully scheduling flows as

$$G_{i,j} = P_{i,j} \alpha_{i,j}(t). \tag{3}$$

In order to introduce the overall task profit maximizing issue with deadline limitation, we formulate a maximizing profit problem for data center networks. Task profit and scheduling revenue is formulated in Eqs. 2 and 3. Here we use the summation of flows profit to denote the overall task profit which is the objective to maximize.

The program is presented in Eqs. (4)–(7). Clearly, the objective function in Eq. (4) defines the profit gained by successfully scheduling tasks. It should be noted that partially scheduling tasks cannot get partial profit but zero profit. Eq. (5) enforces the task integrity which makes the task to be completed in legitimate time, and Eq. (6) is the network capacity limitation. In Eq. (5), all of the finished flows during legal time are subject to the flow number n_i of task t_i. Network capacity C is an abstraction of network resource that can be allocated to flows for transferring. All of the scheduled flows in the network are strictly subject to network capacity C. To ensures the profit of data center to be maximum, we introduce a strategy that the priorities of flows can be changed with tight deadline. Since we focus on maximizing the overall profit for tasks to guarantee application performance in this paper, however the last several flows in one task seriously impact task overall performance.

$$Max \quad \sum_{i=1}^{n} \sum_{j=1}^{n_i} G_{i,j}, \tag{4}$$

$$s.t. \quad \sum_{a_i}^{d_i} \sum_{j=1}^{n_i} \alpha_{i,j}(t) = n_i, \tag{5}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n_i} s_{i,j} \alpha_{i,j}(t) \le C, \tag{6}$$

$$\alpha_{i,j}(t) \in 0, 1. \qquad \forall i, j \tag{7}$$

Obviously, our model is a NP-hard problem [13]. The optimization program above cannot get the optimal solution in constant time. Therefore the optimization function is a target that we make effort to obtain more profit within deadline for tasks. To this end, we propose an approximation algorithm to obtain the approximate result of scheduling policy. Our model is the first attempt on the view of profit for scheduling issues in data center networks. We introduce an approximation algorithm to solve the scheduling policy in next section.

3. Profit-aware task-level flow schedule (PAT)

In this section, we present the details of PAT. In Section 3.1, we briefly introduce the main devise of PAT. PAT can be divided into two parts: controller design and switch design. In Section 3.2, we introduce main function of controller, in the meantime, we present and analyse an approximate algorithm to maximize overall task profit in our model. Additionally, we discuss a loose constraints for maximum profit model. We introduce the design of switches for PAT in Section 3.3.

3.1. PAT overview

In general, PAT implements a centralized controller that is capable to recognize task information, compute profit as well as generate scheduling policy. The controller schedules flows of diverse tasks on the basis of their profit. Fig. 2 illustrates the method procedure of PAT, which contains a centralized controller. Controller is the core of PAT and orchestrate the scheduling policy for multiple tasks. It maintains a global task information, flow information and network capacity. In the mean time, it performs a calculator to maximize the overall profit. Further, the controller generates a scheduling policy on the basis of task information and overall profit. The switches in PAT execute the policy generated by the controller. PAT works as follows. When a new task request arrives, the controller first collects information of the request, then the controller computes the corresponding task profit, lastly it generates a new scheduling policy for current task requests. In the following we mainly describe the PAT's controller designs and the approximate algorithms to maximize profit.

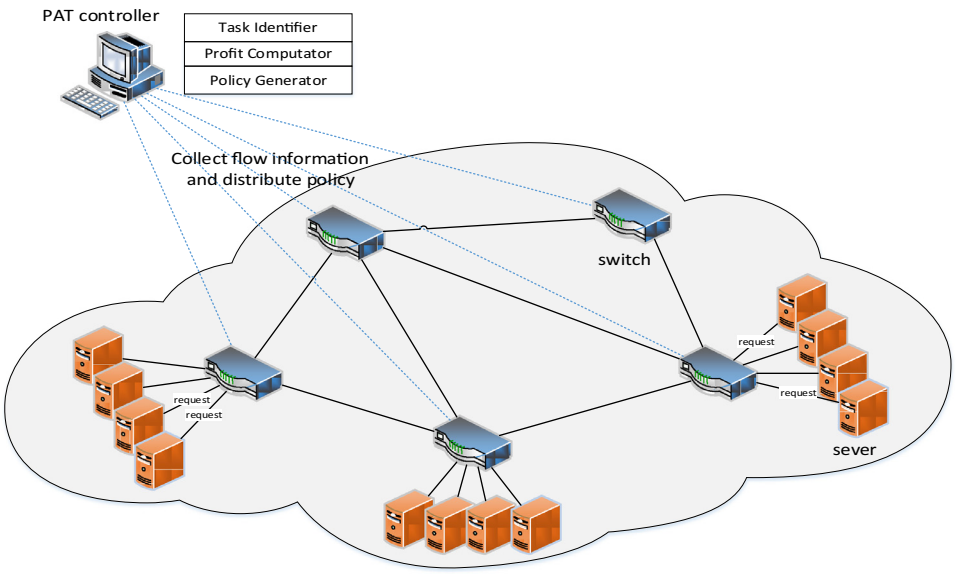


Fig. 2. The framework of PAT.

3.2. Controller design

Many data center networks are designed to have a controller, such that network can be flexibly controlled by the controller. Controller has full view of network situation as well as it recognizes tasks information and status. In addition, as we know, data center [14] has its own administrate system which realizes the application allocation and varieties in data center. In PAT, the controller has three main function: information collector, profit calculator and scheduling policy generator.

3.2.1. Task identifier

PAT controller is capable to collect the information of tasks and flows such as task identifier, task size and requirements. For task identifier, we employ a counter on the controller. When task requests arrive at controller, we increase task counter and mark it as task-id. In order to identify tasks and tasks type, task counter with tasks id and tasks priorities are designed. The module of task identifier accounts for identifying both tasks and flows. In PAT, we schedule tasks according to their profits, therefore the profit of tasks is significant for scheduling. As shown in Eq. (1), tasks profit is determined by tasks value and tasks priorities. In the view of task value, which is associate with applications requirement. Therefore, task information is crucial for PAT. The classifications of applications are basically interactive latency sensitive, interactive throughput sensitive, system control and background services. System control tasks have the requirement of lowest latency and accuracy. In the mean time, the background flows are generally long-lived flows that occupy the network resource from online service. Consequently, we divided the task priority into four levels. In application level, tasks have priorities and their corresponding profit. We encapsule the task priority into the task id. In this case, tasks have task id and its corresponding tasks priorities, which provide information for profit calculator.

3.2.2. Profit calculator

PAT schedules tasks according to their profits and task completion condition. As we mentioned before, there are two level profits which are task profit and flow profit. Task profit is constant while flow profit can be modified over time slot. As we know tasks have deadline, flow A has a longer deadline than flow B while they belong to the same task. In this case, only task-level profit identifier cannot distinguish flow A and flow B. A flow-level profit with the urgency of deadline is necessary in this situation. Specially, flows have a task-level profit information and deadline information, as well as the flow-level profit is determined by task-level and variable *gamma*. This is the initialization progress of profits. In detail, flows belong to the same application carrying different task-id, even so they have the same task profit. Application performance in data center is based on the overall task profit for this application. Here the overall task profit is the summation of the flows profits in the tasks. In the mean time, the application performance in the network can be calculated by collaborate tasks profits.

More specifically, the controller has information on task id, and the tasks priorities can be decoded from task id. At the same time, the profit of task P_i is determined by task value *omega* and task priorities *rho* which is connected to task deadline. As shown in Eq. (1), (2) and (3), overall profit is corresponding to task value, task priority and task completion condition. Since task-id has the priority and id information, thus we only need to calculate the profit from deadline infor-

mation. Accordingly, we choose to modify the vlan id field as the profit of tasks for PAT. In the mean time, the priority field is used to schedule flows.

3.2.3. Policy generator

Here we present the main function of controller scheduling policy generator. In this paper, we mainly focus on maximizing overall profit for tasks. We need to compute the decision variable $\alpha_{i,j}$ for maximum profit model. The approximate Algorithm 1 realizes maximize profit formulation, as we define that in Section 2 a scheduling policy is generated in con-

Algorithm 1 Maximum overall profit computation

Input:

Flow $f_{i,j}$ assemble F; Profit of flow $P_{i,j}$;
Number of flows in task i n_i; Size of schedule task s_i

Output:

Schedule policy set $\mathcal{A}$;
Schedule tasks profit p

1: Sort P_i/n_i in a descend sort order
2: For all flows compute profit $P_{i,j} \leftarrow \gamma * P_i$
3: Task schedule counter k
4: Schedule the valuable tasks under the network capacity limitation $C > 0$
5: Compute the summation of scheduled task profit $p \leftarrow p + P_{i,j}$
6: Schedule the partial of one task while the task size $s_i > C$
7: Partial schedule task l_i
8: maximum k, profit P
9: **return** $\mathcal{A} \leftarrow max\{\sum_{i=1}^{k} P_i, P_{k+1}\}$, profit p

troller. Algorithm 1 lists all the steps in computing profit-effective, overall task profit and schedule policy. The core of our model is task profit which is corresponding to the flow dynamic priority and task priority. In the mean time, the dynamic priority of flow is a variable calculated by task priority and deadline urgency condition determined by the controller. Moreover, PAT is a centralized network that we can apply the controller view to figure out the appropriate scheduling policy. To solve the maximum profit problem, we resort to an approximate algorithm. Next, we introduce our approximate profit algorithm.

Algorithm 1 lists all the steps followed in computing maximum profit and scheduling set $\mathcal{A}$. At the beginning of the algorithm, we sort the profit-effective value of the existing tasks in a descending order in step 1. In a limited network resource situation, scheduling more profit-effective tasks which can bring in more overall profit is a optimal choice. According to the current time slot, flow profit is computed in step 2. In step 3, we attempt to get a maximum i denoted by k which is constrained by the network resource. We schedule all of flows in the most profit-effective tasks which satisfy the constraints of task size is smaller than current network capacity C (step 4). Reduce the capacity C while the resource is used by scheduled tasks, however C is not enough for a entire task that we schedule partial flows for the most profit-effective task(step 6). For each loop in the algorithm, the partial scheduled tasks are considered at first. The controller executes this algorithm until the capacity C reduce to zero.

The Algorithm 1 greedily chooses the most profit-effective tasks, which subjects to network capacity and the nature of tasks. The approximation algorithm is proposed, and its time complexity and approximation ratio are analyzed in the next. Our algorithm is least $\frac{1}{2}\mathcal{OPT}$ in time bound $\mathcal{O}(nlog(n))$. Next, we will prove our approximation rate of algorithm is at least $\frac{1}{2}$.

Theorem 1. *Let $\mathcal{A}$ denote the set output by the algorithm. Then, $profit(\mathcal{A}) > \frac{1}{2}\mathcal{OPT}$.*

Proof. Let $\mathcal{OPT}$ denote the optimal set. If $\sum_{i=1}^{n} s_i \leq C$, then profit($\mathcal{A}$)= profit((*OPT*)). Therefore, we assume $\sum_{i=1}^{n} s_i > C$. We use k to denote the largest positive integer got by in algorithm line 3. We need to prove the following two inequalities

$$\sum_{i=1}^{k} P_i \leq \mathcal{OPT} < \sum_{i=1}^{k+1} P_i \tag{8}$$

Obviously, the first inequality holds in our model. For the second inequality, in the algorithm line 1 we sort the tasks of task profit and task size in the descending order. The most efficient method is to schedule the most valuable tasks as much as possible until exceed the network capacity. The remaining capacity assigns to the most value flows until saturate the whole network. From the network perspective, $\sum_{i=1}^{k+1} P_i$ is the upper bound of profit (*OPT*). If we loose the constraint of flows completion number in tasks Eq. (5) to smaller than n_i. Indicator I_i is denoted as tasks completion and we use $0 < I_i$

< 1 to replace with $I_i \in 0, 1$.

$$I_i = \begin{cases} 1 & i = 1, 2, \ldots, k, \\ \frac{C - \sum_{i=1}^{n} s_i}{s_{k+1}}, & i = k+1, \\ 0 & i = k+2, k+3, \cdots, n. \end{cases} \tag{9}$$

Here we get a maximum profit value $\hat{p}$ in linear program. Therefore, we conduct a inequality

$$\begin{aligned} \mathcal{OPT} \leq \hat{p} &= \sum_{i=1}^{k} P_i + \frac{P_{k+1}}{s_{k+1}} \left(C - \sum_{i=1}^{k} s_i \right) \\ &< \sum_{i=1}^{k} P_i + \frac{P_{k+1}}{s_{k+1}} s_{k+1} = \sum_{i=1}^{k+1} P_i. \end{aligned} \tag{10}$$

In the end, we conduct following Equation

$$\mathcal{A} = max\{P_{k+1}, \sum_{i=1}^{k} P_i\} \geq \frac{1}{2} \sum_{i=1}^{k+1} P_i > \frac{\mathcal{OPT}}{2} \tag{11}$$

Accordingly, our algorithm approximation rate is at least $\frac{1}{2}$. □

The Algorithm 1 greedily chooses the most profit-effective tasks, and the result of algorithm is acceptable and efficient.

3.2.4. Extension of PAT

Most existing data center network scheduling methods aim at minimizing completion time. To minimize completion time, most proposals assume that we know priori knowledge of flow size and deadline so does PAT. Naturally, we figure out sometimes specific flow information is not available [15]. To this end, we introduce an extension PAT without deadline and flow size information (ePAT) that aims at obtaining more profit for scheduling tasks.

In ePAT, we assume the same model described above except the flow size and deadline information. As far as we know, flow size and deadline are priori knowledge which are difficult to obtain. Further, we consider that the specific information is hard to compute. In the mean time, task information is a natural property which we assume available in ePAT. In specific, ePAT controller recognizes the task information and compute task-level profit so as to maximize the overall profit. We do not formulate a profit model for ePAT, since ePAT is similar to PAT as well as less information to formulate a model. However, in ePAT there is a similar Algorithm 2, we present the detail of the algorithms next.

Algorithm 2 Maximum overall profit computation

Input:
Flow $f_{i,j}$ assemble F; Flow number in task i n_i;
Output:
Schedule policy set $\mathcal{A}$;
Schedule tasks profit p
1: Sort task profit P_i in a descend sort order
2: Sort number of flows n_i in a ascending order
3: Schedule task in profit-effective order
4: Task schedule counter k
5: Schedule the valuable tasks under the network capacity limitation $C > 0$
6: Compute the summation of scheduled task profit $p \leftarrow p + P_i$
7: maximum k, profit P
8: **return** $\mathcal{A} \leftarrow max\{\sum_{i=1}^{k} P_i, P_{k+1}\}$, profit p

In consequence of the absence of flow information, the profit-effective is affected by the task information. Specially, as we mentioned before, the task priorities can be classified into four levels. Generally, task profit can be distinguished by task priority. Even so, the same level tasks are difficult to distinguish. At this point, we apply the number flows in the task to judge the task-effective. For example, task A and task B are on the same priority level for applications, while flow number in task A n_a is smaller than flow number in task B n_b. In this case, we predicate that task A is more profit-effective than task B. Accordingly, in ePAT we firstly order the profit-effective sequence in step 1–2. In steps 3–7, the most k profit-effective tasks is respectively scheduled on the condition of available network resources. In the mean time, we compute the maximum profit and scheduling policy. If there is no sufficient bandwidth for a entire task, we firstly schedule partial flows in the $k+1$ profit-effective task.

3.3. Switch design

SDN network [16] separates the network control plane from data forwarding plane which can improve resource utilization, network management and reduce cost. Controller has a full view of the whole network and flow information. According to the protocol of openflow [17], switches have flow tables when one unkown flow appears it trigger the packet-in and send the packet-in information to the controller. The controller computes the route and other bits in packet head and responses to switches.

Similarly, the pota switches employ a simple mechanism on priority scheduling. Our switches solely focus on the priorities of flows without considering the task information. The limitation of PAT switches is that it employs openflow switches and schedules flows based on the flow table. In specific, controller has the information of deadline and profit which are not available in flow table and packtin message. The controller is responsible for computing priorities on both task-level and flow-level. In order to obtain maximum profit, the controller executes the Algorithm 1 and schedules a set of flows need to be scheduled at current time slot. In ePAT, the Algorithm 2 is simpler than Algorithm 1. Thus, the controller is capable to implement ePAT on the basis of PAT. Then, the controller modifies the priorities of scheduling flows to absolute high priorities for authorities to scheduling flows in the switches. In the mean time, switches resolve flows to transfer on the basis priorities decided by the controller.

4. Performance evaluation

In this section we evaluate PAT and ePAT performance in mininet. Our evaluation consists of four parts. First, we use mininet to build a topology to run our experiments. Second, we generate data mining and web search workload with integrated task information including deadline and profit information. Further, we realize the Algorithms 1 and 2 on controller. In the end, we present the main result on task completion time, task completion rate and overall profit comparing to flow-level priority scheduling.

4.1. Experiment setting

Workload: Our evaluation considers two representative workload: web search and data mining. We generate workloads using python socket which send and receive flows according to Possion process on the basis of the characteristic of web search and data mining [18,19]. We generate the flow size range from 1 KB to 50MB and record them into trace file. In the mean time, these two workloads run on separate hosts in our mininet topology.

Topology: We use a hose model topology in Fig. 3 which can highlight the shortage of network resource. In addition, we could not scale up the hosts number due to memory constrains, so we employs the hose model topology as 20 hosts connected to a single switch. Hose model topology has a severe limitation on switch capacity moreover it easily causes congestion. We choose it to emphasize the significance of scheduling. Meanwhile, we emulate link bandwidth to 100 Mbps.

Performance: We consider three performance benchmarks. Comparing to flow completion time, we choose task completion time to illustrate the performance of our method. Since our method employs the overall profit to improve performance, so we define a parameter of task completion rate to illustrate the task performance. As well as profit is another quantity to directly measure task performance. To this end, we compare our method to priority schedule with task completion time, overall profit and task completion rate.

4.2. Overall performance

In this section we show PAT and ePAT performance in hose topology with simulation workloads. We show that PAT has a similar result to priority scheme on flow completion time. Nevertheless with much higher profit and completion rate on task-level. About ePAT, it is clear that ePAT has better performance than flow scheduling, while not much worse than PAT method.

4.2.1. Flow completion time

We illustrate the flow completion time under two workloads: websearch and datamining. First, we run one workload a round and then mix these two workloads. The websearch workload have smooth and better result than datamining workload. The reason is that the flow sizes in datamining workload widely range from 1 KB to 50MB, while the webseach flow

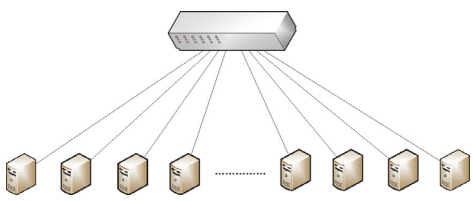
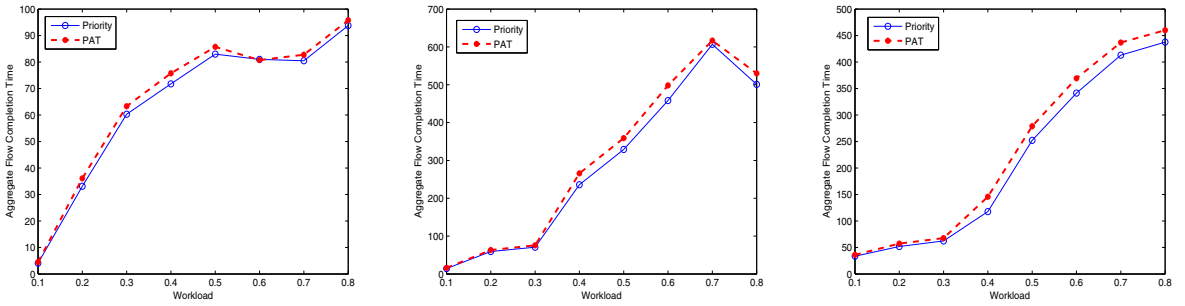
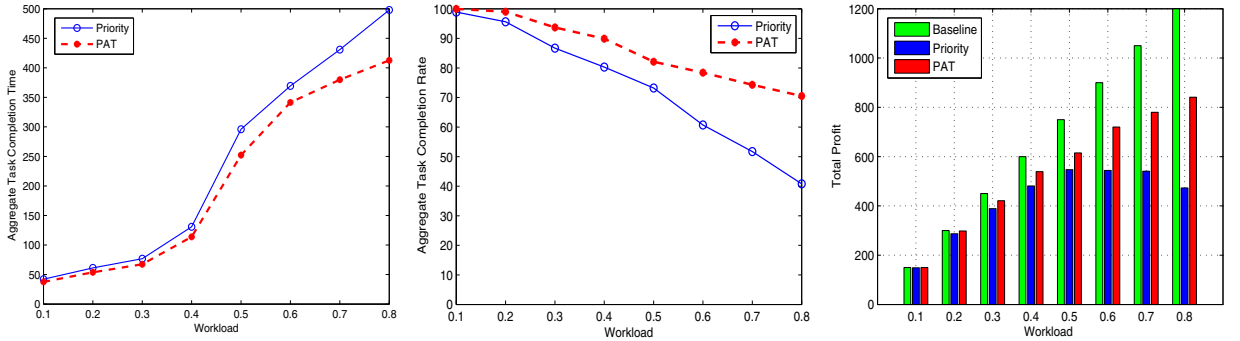


Fig. 3. Topology used in simulations.



(a) Web search workload (b) Data mining workload (c) Mix workload

Fig. 4. The average flow completion time of two workloads on web search and data mining and the mix workload.



(a) Task completion time (b) Task completion rate (c) Overall profit

Fig. 5. The overall performance on task completion time, task completion rate and profit for PAT.

sizes only range from 1 KB to 1MB. As shown in Fig. 4, the flow completion time is normalize to the best possible completion time (assume there is no congestion in the network). PAT runs two kinds of workloads, the average flow completion time compares to flow-level priority scheduling method. First, we let 10 hosts run the websearch workload as well as other 10 hosts run data mining workloads, Fig. 4(c) shows the results. The routing process is not considered in our evaluation. Comparing to Fig. 4(a) and Fig. 4(b), mixed workloads get a worse flow completion time. As shown in Fig. 4, flow completion time of PAT is not better than priority scheduling method, since PAT is a task-level scheduling method. In addition, our the results of PAT is similar to priority method in Fig. 4.

4.2.2. Task completion time

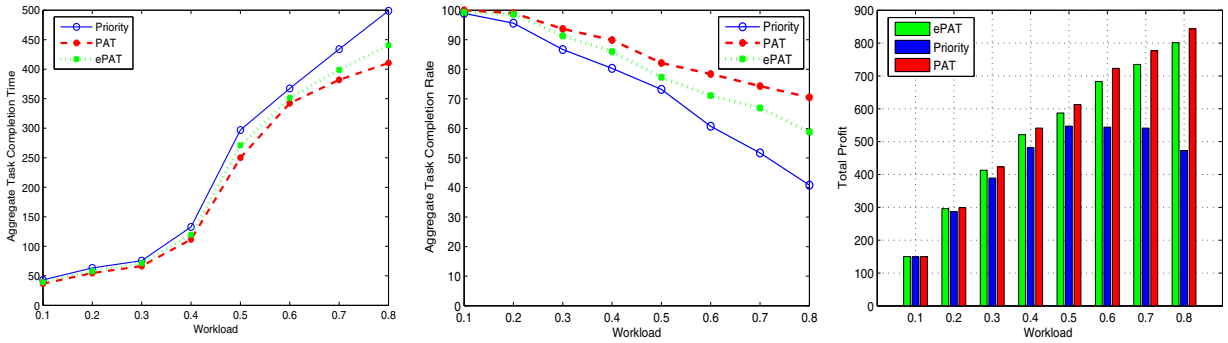
Flow completion time is a standard most scheduling methods applying, in PAT we introduce a benchmark of task completion time to illustrate our advantages. Task completion time is a significant quantity to describe task performance which is crucial for applications. In task completion time(TCT) PAT is distinctly better than priority scheduling method showing Fig. 5(a). PAT solely focuses on task-level scheduling, it centralized schedules a collaborate of flows in one task, consequently it reduces the task completion time comparing to priority scheduling method.

4.2.3. Task completion rate

PAT is a method aiming at maximizing overall profit to improving task performance for applications. Here we apply the task completion rate to be a standard to estimate the scheduling result. We define the overall performance of task completion condition as task completion rate. The deadline information is used to justify whether the tasks are completed. We run the mix workloads, and calculate the task completion rate according to the trace file shows in Fig. 5(b) which clearly shows the benefits from PAT in task completion rate. Comparing to Fig. 4, we observe that PAT sacrifices 8.1% flow-level latency for a incremental of 21% task completion rate, which directly improves user experience and application performance. PAT schedules integrated flows in one task, consequently task completion rate is better than flow-level priority scheduling methods.

4.2.4. Task profit

Task profit is a definition we proposed in this paper as shown in Eq. (1) and (2). Profit is obtained through successfully scheduling flows in tasks. The profit of web search and data mining tasks in our evaluation are normalized to integer. Additionally web search workload has a profit of 2 as well as data mining workload has a profit of 1. In the Algorithm 1, we can calculate the maximum overall profit which is shown in Fig. 5(c). In Fig. 5(c), PAT significantly improves overall profit in



(a) Task completion time (b) Task completion rate (c) Overall profit

Fig. 6. The overall performance on task completion time, task completion rate and profit for ePAT.

the limited network condition. However, flow-level priority scheduling method ignores the integrity of tasks. The differences between flow-level scheduling and PAT are that PAT guarantees the integrity of tasks and obtains the total task profit under the limited network, while flow-level scheduling method focuses on individual of flows without global view of tasks.

4.2.5. ePAT performance

We have shown that PAT has a good performance on three benchmarks: task completion time, task completion rate and overall profit. For the reason of difficulty of priori information on deadline and flow size, we present an extension of PAT (ePAT) that schedules flows in diverse tasks regardless of deadline and flow size information. Specially, ePAT solely schedules flows according to their task information, while PAT is able to schedule flows with specific flow information on deadline and size. In our method, ePAT is a loosen constrains from PAT, so the ePAT is easily implements from PAT. As shown in Fig. 6, the result of ePAT is better than the flow priority scheduling, while PAT's result is better than ePAT under the three benchmarks. The ePAT improves 9% task completion time comparing to the flow-level scheduling, while gets a 19% improvement on overall profit. Less information makes ePAT performance is not as good as PAT. Obviously, PAT is aware of global and specific information on flow level which makes a good performance on scheduling. The ePAT sacrifices a little performance for less computation and more practical method.

5. Related work

Latency is crucial for interactive data center applications. Many works in data center networks mainly focus on reducing flow completion time or task completion time. Specially, reducing flow completion time or task completion time is basically similar, as well as deadline is a special issue in latency. In this section, we briefly discuss some works relevant to PAT.

Hedera [6] presents the elephant flows occupying the bandwidth resource which may cause short flows starving for network resource. HULL [20] makes effort to keep the queue low occupation by using congestion control algorithm. HULL reserves some bandwidth for short flows and controls packets in order to avoid elephant flow take up all the bandwidth resources. Consequently, HULL obtains small latencies. Low network occupation can reduce latency, meanwhile, redundancy can also receive good result. In the endurable network condition, short flows redundancy [21,22] can reduce completion times. In data center topology [23], fattree provides multipath between any two hosts which provides opportunity for replicate flows to minimize flow completion times. Reduce long tail flow completion time means the applications get better worst performance, DeTail [24] designs to lower the long-tailed flow completion times for different data center workflows. L^2DCT [25] is a transport protocol to minimize flow completion time by modulate the congestion window size based on the estimate flow sizes.

DCTCP [1] keeps the queue occupation under the threshold by introducing adaptive congestion control based on ECN. Low buffer utilization can partly reduce latency and miss deadline rate. D^3 [2] first proposes rate control scheme using deadline information, and assigns the rate by computing desire rate which uses the information of flow size and flows deadline or estimated completion time. D^2TCP [3] is a deadline-aware TCP protocol extended by DCTCP. It implements to adjust the window size based on both deadline information and the congestion status. These methods mostly improved latency, however, they have limitations that they cannot precisely estimate the right flow rate to meet deadline when the earlier deadline flows arrive late. Moreover, bandwidth reservation and rate adjustment methods are hard to handle traffic bursty which is common in data center networks.

Having recognized the rate control strategy limitations, some subsequent works pay attention to bandwidth preemption and the quality of service. PDQ [4] and pFabric [5] are the state-of-the-art approaches in priority schedule. PDQ provides a distributed flow schedule mechanism based on assigning rates to flows with flow priority. While pFabric introduces a priority queue to schedule flows. As a result of task consists of multiple flows, the integrity of one task is also important for data center applications. Maximizing overall profit and guarantee the integrity of tasks are the main goals in our work.

Data centers have a huge amount of flows. All of these flows in one task need to finish before deadline or task completion. Considering this, some effort has been made in data center networks. Coflows [8,9] proposed an application-level network abstraction and schedule problem. Coflow makes a sketch for the application into network abstraction and uses the abstraction to optimize some application-level quality of service. On the basic of coflow, Varys [9] proposes a schedule problem to minimize flow completion time with rate allocation method. On task-level scheduling, baraat [7] works on decentralized task-level scheduling, as well as flows in one task loosely synchronize by different pace. While we focus on a centralized control method to make use of controller and increase task-level profits.

6. Conclusion and future work

PAT is a centralized method which is designed to accomplish tasks within their deadlines. It employs the deadline and task information to achieve maximum profit while scheduling tasks in data center networks. Tasks have a value of accomplishment, and flows in the tasks can dynamically affect the overall profit of the tasks with scheduling. Hence PAT can recognize the completion urgency for better profit. Besides PAT has good adaptability for the nature of multiple types of flows in data centers. In addition, we introduce a situation without deadline information to guarantee task performance. We present ePAT a missing deadline information maximizing profit method. Our evaluation shows that the profit of task is a reasonable representation of the quality of service. PAT and ePAT fit for both different and uniform flows. They significantly benefits the task completion rate, task completion time and overall profit comparing to existing priority scheduling methods.

Our future work concludes two aspects. First, the definition of profit in this paper is an abstract concept which is related to the task priority and task value. Flow profit is determined by task classification and completion condition. Actually, the number of accomplished flows in one task and deadline emergency can affect the profit of scheduling flows. We take attempts to improve our profit model to fit for the practical settings. Second, we will further polish PAT and its approximation rate on algorithm, and then implement it on testbed in a real data center network environment.

Acknowledgments

This work is supported by the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61225010); the State Key Program of National Natural Science of China(Grant No. 61432002); NSFC Grant Nos. 61272417, 6130018961370199 and 61173161; Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20130041110019), and the Fundamental Research Funds for the Central Universities (Grant. DUT15QY20).

References

[1] Alizadeh M, Greenberg A, Maltz DA, Padhye J, Patel P, Prabhakar B, et al. Data center tcp (dctcp). ACM SIGCOMM Comput.Commun.Rev. 2011;40:63–74.
[2] Wilson C, Ballani H, Karagiannis T, Rowtron A. Better never than late: meeting deadlines in datacenter networks. ACM SIGCOMM Computer Communication Review 2011;41:50–61.
[3] Vamanan B, Hasan J, Vijaykumar T. Deadline-aware datacenter tcp (d2tcp). ACM SIGCOMM Comput. Commun. Rev. 2012;42:115–26.
[4] Hong C-Y, Caesar M, Godfrey P. Finishing flows quickly with preemptive scheduling. ACM SIGCOMM Comput. Commun. Rev. 2012.
[5] Alizadeh M, Yang S, Sharif M, Katti S, McKeown N, Prabhakar B, et al. pfabric: Minimal near-optimal datacenter transport SIGCOMM; 2013.
[6] Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: Dynamic flow scheduling for data center networks. NSDI; 2010.
[7] Dogar FR, Karagiannis T, Ballani H, Rowstron A. Decentralized task-aware scheduling for data center networks. SIGCOMM; 2014.
[8] Chowdhury M, Stoica I. Coflow: a networking abstraction for cluster applications. In: Proceedings of the 11th ACM Workshop on Hot Topics in Networks; 2012.
[9] Chowdhury M, Zhong Y, Stoica I. Efficient coflow scheduling with varys. SIGCOMM; 2014.
[10] Li H, Dong M, Ota K, Guo M. Pricing and repurchasing for big data processing in multi-clouds. IEEE Trans. Emerging Top. Comput. 2016;4:266–77.
[11] Mianxiong D, Xiao L, Zhuzhong Q, Anfeng L, Tao W. Qoe-ensured price competition model for emerging mobile networks. IEEE Wireless Commun. 2015;22:50–7.
[12] Xiao L, Mianxiong D, Kaoru O, Patrick H, Anfeng L. Service pricing decision in cyber-physical systems: insights from game theory. IEEE Trans. Serv. Comput. 2016;9:186–98.
[13] El-Rewini H, Lewis TG, Ali HH. Task scheduling in parallel and distributed systems. Prentice-Hall, Inc.; 1994.
[14] Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, et al. B4: Experience with a globally-deployed software defined wan SIGCOMM; 2013.
[15] Bai W, Chen L, Chen K, Han D, Tian C, Sun W. Pias: Practical information-agnostic flow scheduling for data center networks. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks; 2014.
[16] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, et al. Openflow: Enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 2008;38:69–74.
[17] Consortium O.S., et al. Openflow switch specification version 1.1. 0. 2011.
[18] Kandula S, Sengupta S, Greenberg A, Patel P, Chaiken R. The nature of data center traffic: measurements & analysis. SIGCOMM; 2009.
[19] Benson T, Anand A, Akella A, Zhang M. Understanding data center traffic characteristics. ACM SIGCOMM Comput. Commun. Rev. 2010;40:92–9.
[20] Alizadeh M, Kabbani A, Edsall T, Prabhakar B, Vahdat A, Yasuda M. Less is more: trading a little bandwidth for ultra-low latency in the data center. NSDI; 2012.
[21] Vulimiri A, Godfrey PB, Mittal R, Sherry J, Ratnasamy S, Shenker S. Low latency via redundancy. In: Proceedings of the ACM conference on Emerging networking experiments and technologies; 2013.
[22] Xu H, Li B. Repflow: Minimizing flow completion times with replicated flows in data centers. INFOCOM; 2014.
[23] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. ACM SIGCOMM Comput. Commun. Rev. 2008;38:63–74.
[24] Zats D, Das T, Mohan P, Borthakur D, Katz R. Detail: Reducing the flow completion time tail in datacenter networks. ACM SIGCOMM Comput. Commun. Rev. 2012.
[25] Munir A, Qazi IA, Uzmi ZA, Mushtaq A, Ismail SN, Iqbal MS, et al. Minimizing flow completion times in data centers INFOCOM; 2013.

Xiaoyi Tao received the bachelor's degree from the school of Software Engineering, Dalian University of Technology, China, in 2011. Currently, she is a Ph.D. candidate in the School of Computer Science and Technology, Dalian University of Technology, China. Her research interests include datacenter networks, SDN networks and cloud computing.

Heng Qi received the bachelors degree from Hunan University in 2004 and the masters and doctoral degree from Dalian University of Technology in 2006 and 2012, respectively. He is a Lecturer at the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include computer network, wireless network and multimedia computing.

Wenxin Li received the bachelor's degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2012. Currently, he is a Ph.D. candidate in the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include datacenter networks and cloud computing.

Keqiu Li received the bachelors and masters degrees from the Department of Applied Mathematics at the Dalian University of Technology. He received the Ph.D. degree from the School of Information Science, Japan Advanced Institute of Science and Technology. He also has postdoctoral experience with University of Tokyo. He is a Professor in Dalian University of Technology, Computer Science and Technology.

Yang Liu received the bachelor's degree from the school of Computer Science and Technology, Dalian Maritime University, China. She is currently a master student in the School of Computer Science and Technology, Dalian University of Technology, China. Her research interests include cloud computing and datacenter networks.