Computers and Electrical Engineering 000 (2016) 1–12

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 xxx

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.

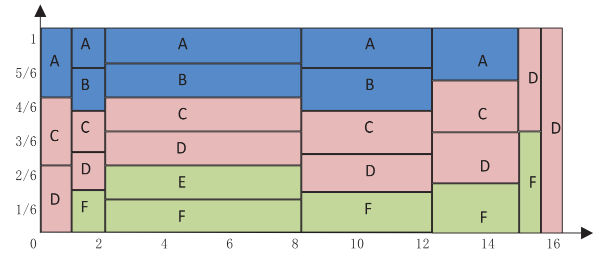
[*] Corresponding author.

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).

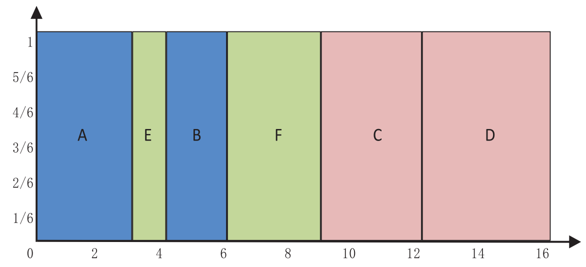
http://dx.doi.org/10.1016/j.compeleceng.2016.07.013
0045-7906/© 2016 Elsevier Ltd. All rights reserved.

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	

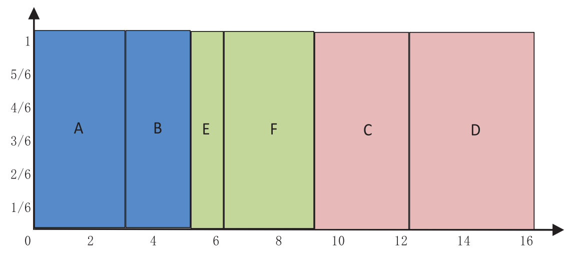
(a) The concurrent flows conditions for one bottleneck link



(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