Title example

——————subtitle here

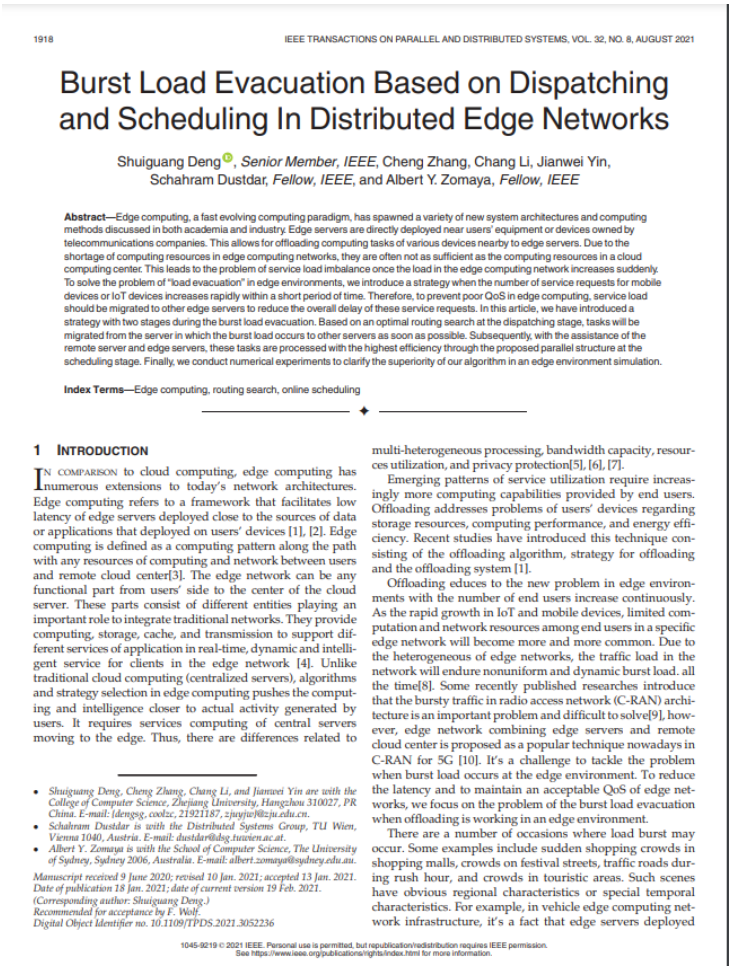
your name

your institute

youremail@example.com

20xx 年 x 月 x 日

Paper Introduction Example



1918 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 32, NO. 8, AUGUST 2021

Burst Load Evacuation Based on Dispatching and Scheduling In Distributed Edge Networks

Shuiguang Deng, *Senior Member, IEEE*, Cheng Zhang, Chang Li, Jianwei Yin, Schahram Dustdar, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—Edge computing, a fast evolving computing paradigm, has spawned a variety of new system architectures and computing methods discussed in both academia and industry. Edge servers are directly deployed near users' equipment or devices owned by telecommunications companies. This allows for offloading computing tasks of various devices nearby to edge servers. Due to the shortage of computing resources in edge computing networks, they are often not as sufficient as the computing resources in a cloud computing center. This leads to the problem of service load imbalance once the load in the edge computing network increases suddenly. To solve the problem of "load evacuation" in edge environments, we introduce a strategy when the number of service requests for mobile devices or IoT devices increases rapidly within a short period of time. Therefore, to prevent poor QoS in edge computing, service load should be migrated to other edge servers to reduce the overall delay of these service requests. In this article, we have introduced a strategy with two stages during the burst load evacuation. Based on an optimal routing search at the dispatching stage, tasks will be migrated from the server in which the burst load occurs to other servers as soon as possible. Subsequently, with the assistance of the remote server and edge servers, these tasks are processed with the highest efficiency through the proposed parallel structure at the scheduling stage. Finally, we conduct numerical experiments to clarify the superiority of our algorithm in an edge environment simulation.

Index Terms—Edge computing, routing search, online scheduling

1 Introduction

In comparison to cloud computing, edge computing has numerous extensions to today's network architectures. Edge computing refers to a framework that facilitates low latency of edge servers deployed close to the sources of data or applications that deployed on users' devices [1], [2]. Edge computing is defined as a computing pattern along the path with any resources of computing and network between users and remote cloud center[3]. The edge network can be any functional part from users' side to the center of the cloud server. These parts consist of different entities playing an important role to integrate traditional networks. They provide computing, storage, cache, and transmission to support different services of application in real-time, dynamic and intelligent service for clients in the edge network [4]. Unlike traditional cloud computing (centralized servers), algorithms and strategy selection in edge computing pushes the computing and intelligence closer to actual activity generated by users. It requires services computing of central servers moving to the edge. Thus, there are differences related to multi-heterogeneous processing, bandwidth capacity, resources utilization, and privacy protection[5], [6], [7].

Emerging patterns of service utilization require increasingly more computing capabilities provided by end users. Offloading addresses problems of users' devices regarding storage resources, computing performance, and energy efficiency. Recent studies have introduced this technique consisting of the offloading algorithm, strategy for offloading and the offloading system [1].

Offloading educes to the new problem in edge environments with the number of end users increase continuously. As the rapid growth in IoT and mobile devices, limited computation and network resources among end users in a specific edge network will become more and more common. Due to the heterogeneous of edge networks, the traffic load in the network will endure nonuniform and dynamic burst load. all the time[8]. Some recently published researches introduce that the bursty traffic in radio access network (C-RAN) architecture is an important problem and difficult to solve[9], however, edge network combining edge servers and remote cloud center is proposed as a popular technique nowadays in C-RAN for 5G [10]. It's a challenge to tackle the problem when burst load occurs at the edge environment. To reduce the latency and to maintain an acceptable QoS of edge networks, we focus on the problem of the burst load evacuation when offloading is working in an edge environment.

There are a number of occasions where load burst may occur. Some examples include sudden shopping crowds in shopping malls, crowds on festival streets, traffic roads during rush hour, and crowds in touristic areas. Such scenes have obvious regional characteristics or special temporal characteristics. For example, in vehicle edge computing network infrastructure, it's a fact that edge servers deployed

• *Shuiguang Deng, Cheng Zhang, Chang Li, and Jianwei Yin are with the College of Computer Science, Zhejiang University, Hangzhou 310027, PR China. E-mail: {dengsg, coolzc, 21921187, zjuyjw}@zju.edu.cn.*
• *Schahram Dustdar is with the Distributed Systems Group, TU Wien, Vienna 1040, Austria. E-mail: dustdar@dsg.tuwien.ac.at.*
• *Albert Y. Zomaya is with the School of Computer Science, The University of Sydney, Sydney 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 9 June 2020; revised 10 Jan. 2021; accepted 13 Jan. 2021. Date of publication 18 Jan. 2021; date of current version 19 Feb. 2021.
(Corresponding author: Shuiguang Deng.)
Recommended for acceptance by F. Wolf.
Digital Object Identifier no. 10.1109/TPDS.2021.3052236

- S. Deng, C. Zhang, C. Li, J. Yin, S. Dustdar and A. Y. Zomaya, " Burst Load Evacuation Based on Dispatching and Scheduling In Distributed Edge Networks," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 8, pp. 1918-1932, 1 Aug. 2021, doi: 10.1109/TPDS.2021.3052236.
- Innovations
  - point one
  - point two

Contents

Contents

The reference exmaple

- The reference are used as follows:
  - In the article[1] , the author proposed a new algorithm.
  - In order to do something， article[2]designed the example protocol.
  - Someone[3] used the example algorithm to address the problem.

[1] Kengo Sasaki, Satoshi Makido, and Akihiro Nakao. "Vehicle Control System for Cooperative Driving Coordinated Multi -Layered Edge Servers". In: 2018 IEEE 7th International Conference on Cloud Networking (CloudNet). 2018, pp. 1–7.

[2] Paolo Romano and Francesco Quaglia. "Design and Evaluation of a Parallel Invocation Protocol for Transactional Applications over the Web". In: IEEE Transactions on Computers 63.2 (2014), pp. 317–334.

[3] Xu Chen et al. "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing". In: IEEE/ACM Transactions on Networking 24.5 (2016), pp. 2795–2808.

The example of formula reference

- The reference of formula is as follows:
  - $x = r_1, r_2 \ldots r_N$, find the optimal solution:
  - thus the $P2$:

$$\mathcal{P}_2 : \min_{x \in \mathcal{X}} \max(r_1, \ldots, r_N) \tag{8}$$

- this is a random sentence:
  - $H = h : X \rightarrow Q$,where Q is the optimal solutions.
  - $S_0 = (x_0, y_0) \ldots (x_U, y_U)$,

Contents

The example of mixed text and image

- Item1
  - Item1.1 This is a randomly generated sentence
  - Item1.2 This is a randomly generated sentence
  - Item1.3 This is a randomly generated sentence
- Item2

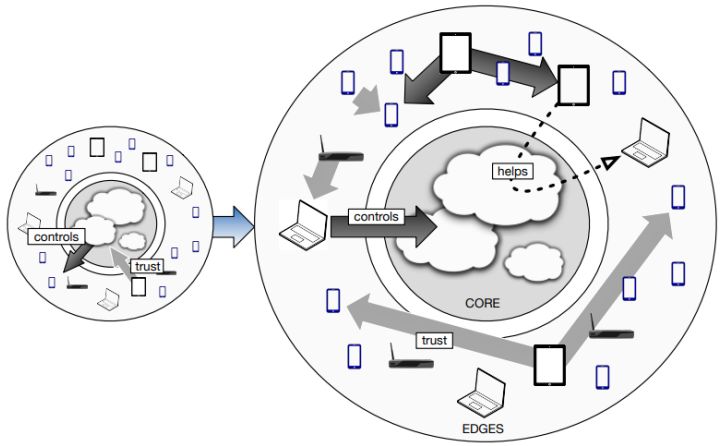


Figure 1: Centralized cloud model (left) versus Edge-centric Computing (right).

The example of combining formula and image

- index set of tasks as $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$
- the index set of edge servers as $\mathcal{M} \triangleq \{1, 2, \ldots, M\}$
- We denote the indicator of a task latency in burst load as $I_i^{\dagger}(t)$.While $I_i^{\dagger}(t) = 1$ means waiting.
  - The latency time of the ith task denoted as:

$$T_1^i = \sum_{t>0} I_i^{\dagger}(t), \qquad (1)$$

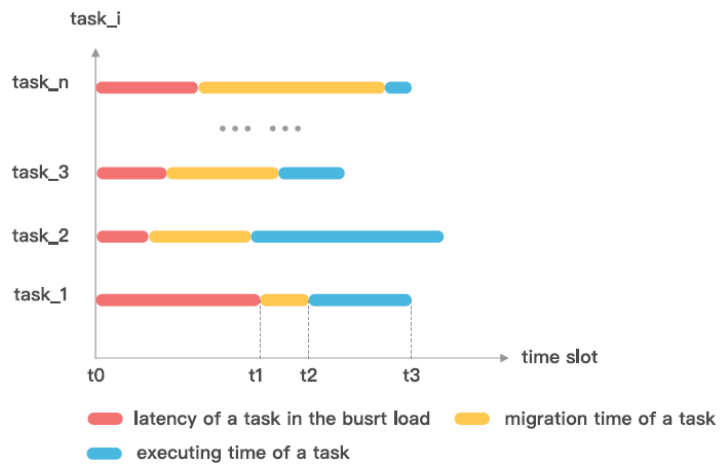
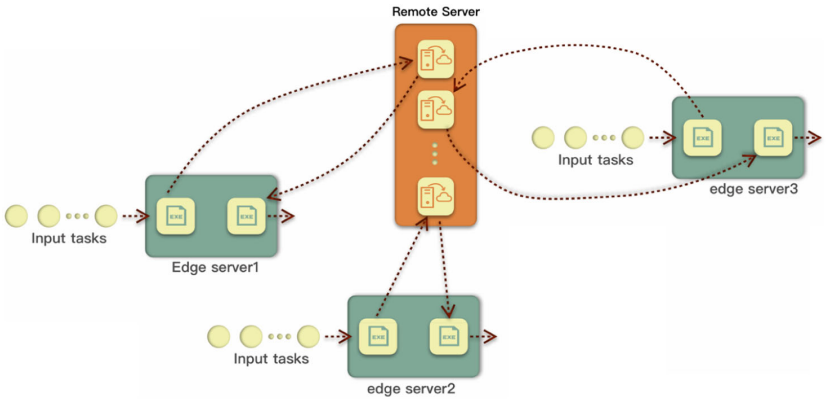
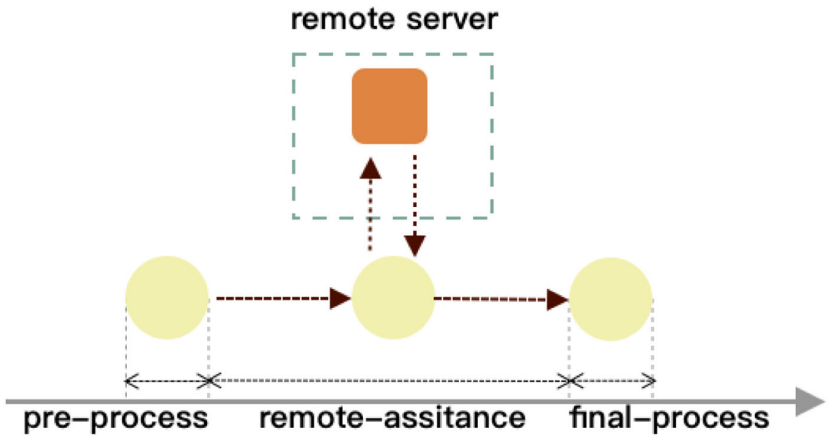


Figure 2: The flowchart of tasks when evacuation

The example of two images



(a) Task scheduling for executing and requesting data



(b) Task processing sequence.

The example of single image

Algorithm 1. Optimal Routing for Task Dispatching (OPTD)

Generate the window size W, the initial solution size $\mathcal{U}$,the parameter of routing selection γ
Initial $S_0 = \{(x_0, y_0), \ldots, (x_\mathcal{U}, y_\mathcal{U})\}, V_t = \emptyset$
for $t = 1$ to $\mathcal{T}$ **do**
  $(x_t, y_t) = (x_{min}, y_{min}), I_v = I, I_f = \emptyset$
  for $i = (t-1) * W$ to $t * W$ **do**
    if $\lambda < \alpha$ **then**
      $\tilde{x}_t^i = \lceil x_t^i / \gamma \rceil, I_v = I_v \setminus \{i\}, I_f = I_f \cup \{i\}$
    else
      $\tilde{x}_t^i = rand(1, |\mathcal{R}|)$
    end if
  end for
  $\tilde{y}_t = c(\tilde{x}_t)$
  if $\tilde{y}_t < y_t$ **then**
    construct h_t from the updated I_v and I_f
  else
    randomly choose i with the size of $|I_f|$ to replace the elements in I_f, $I_v = I \setminus I_f$, construct h_t
  end if
  for $t = 1$ to $\mathcal{U}$ **do**
    Sample x_h from h_t, $y_h = c(x_h), V_t = V_t \cup (x_h, y_h)$
  end for
  $y_{min} = \min\{y_h, y_t, \tilde{y}_t\}, (x_h, y_h) \in V_t$
end for
return (x_{min}, y_{min})

Reference

[1] Kengo Sasaki, Satoshi Makido, and Akihiro Nakao. "Vehicle Control System for Cooperative Driving Coordinated Multi -Layered Edge Servers". In: 2018 IEEE 7th International Conference on Cloud Networking (CloudNet). 2018, pp. 1–7.

[2] Paolo Romano and Francesco Quaglia. "Design and Evaluation of a Parallel Invocation Protocol for Transactional Applications over the Web". In: IEEE Transactions on Computers 63.2 (2014), pp. 317–334.

[3] Xu Chen et al. "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing". In: IEEE/ACM Transactions on Networking 24.5 (2016), pp. 2795–2808.

Acknowledgement

- thank y’ all.