Distributed Redundancy Scheduling for Microservice-based Applications at the Edge

1st Hailiang Zhao
College of Computer Science and Technology
Zhejiang Univeristy
Hangzhou, China
hliangzhao@zju.edu.cn

2nd Shuiguang Deng
College of Computer Science and Technology
Zhejiang Univeristy
Hangzhou, China
dengsg@zju.edu.cn

3rd Zijie Liu
College of Computer Science and Technology
Zhejiang Univeristy
Hangzhou, China
liuzijie@zju.edu.cn

4th Jianwei Yin
College of Computer Science and Technology
Zhejiang Univeristy
Hangzhou, China
zjuyjw@zju.edu.cn

5th Schahram Dustdar
Distributed Systems Group
Technische Universität Wien
Vienna, Austria
dustdar@dsg.tuwien.ac.at

[J1C2 Presentation Abstract at IEEE SERVICES 2021 for IEEE Transactions on Services Computing DOI 10.1109/TSC.2020.3013600]

Multi-access Edge Computing is booming as a promising paradigm to push the computation and communication resources from cloud to the edge to provision services and to perform computations. With container technologies, mobile devices with small memory footprint can run composite microservice-based applications without the time-consuming backbone transmission. Service placement at the edge is of importance to put MEC from theory into practice. However, current state-of-the-art research does not sufficiently take the composite property of services into consideration but study the to-be-placed services in an atomic way. Besides, although Kubernetes has certain abilities to heal container failures, high availability cannot be ensured due to heterogeneity and variability of edge sites.

To solve the above problems, in this paper, we propose a distributed redundant placement framework, named Sample Average Approximation-based Redundancy Placement (SAA-RP), for the microservice-based applications with sequential combinatorial structure. For this kind of application, if all of the candidates are placed on one edge site, network congestion is inevitable. Therefore, we adopt a distributed placement scheme, which is naturally suitable for the distributed edge. Redundancy is the core of SAA-RP, which allows that *one candidate to be dispatched to multiple edge sites*. By creating multiple candidate instances, it boosts a faster response to service requests. To be specific, it alleviates the risk of a long delay incurred when a candidate is assigned to only one edge site. With one candidate deployed on more than one edge site, requests from different end users at different locations can be balanced, so as to ensure the high availability of service and the robustness of the provision platform. Currently, the main strategy of job redundancy usually releases the resource occupancy after completion, which is not befitting for geographically distributed edge sites. In addition, job redundancy is not always a win and might be dangerous sometimes, since practical studies have shown that creating too many instances can lead to unacceptably high response times and even instability. Thus, in SAA-RP, we do not release the candidate instances but periodically update them based on the observations of service demand status during that period.

SAA-RP works as follows. Firstly, we derive expressions to decide each service candidate should be dispatched with how many instances and which edge sites to place them. Specifically, we calculate the response time for each microservice instance based on its dispatched edge site and the execution details of its predecessors. We thoroughly discuss the data transmission cost under different scenarios. Then, by collecting user requests for different service composition schemes, we model the distributed placement problem as a discrete stochastic optimization problem and approximate the expected service response time through long-term Monte Carlo sampling. During each sampling, we solve the deterministic problem based on a subroutine GA-based Server Selection. SAA-RP makes up with the shortcoming of the default scheduler of Kubernetes when encountering the MEC. It takes both the service request pattern of end users and the heterogeneity of the distributed edge sites into consideration.

The performance of SAA-RP on the overall service response time is verified with extensive simulations based on a real-world dataset. We design several benchmark policies in two scenarios, where redundancy is allowed and not allowed, respectively. When redundancy is not allowed, each service candidate can only be dispatched to one edge site. We also choose a state-of-the-art service dispatching algorithm for MEC, named GenDoc, for comparison. In our default settings, the simulation results show that our algorithm outperforms these benchmark policies at least 16.21% and GenDoc 10.81%, respectively. The results also show that SAA-RP is robust to the change of the scale of applications.