SCIENCE CHINA
Information Sciences

• **Supplementary File** •

SMAF: A Secure and Makespan-aware Framework for Executing Serverless Workflows

Shuai ZHANG[1], Yunfei GUO[1], Zehua GUO[2*], Hongchao HU[1,3] & Guozhen CHEN[1,3]

[1]*Information Engineering University, Zhengzhou* 450002, *China;*
[2]*Beijing Institute of Technology, Beijing* 100081, *China;*
[3]*Purple Mountain Laboratories, Nanjing* 211111, *China*

Appendix A Experimental results

We focus on the makespan and security performance defined in our model to evaluate the effectiveness of our algorithm. These two metrics can be obtained based on the scheduling plan. As the Linux container is used as our function environment, the vulnerabilities of Linux contained are collected from NVD (after 2019). In our evaluation, a real-world scientific workflow which is called Montage is used. Contrastive analysis is used to show the advantage of our framework. HEFT and Cold Start based Heterogeneous Earliest Finish Time (CSHEFT) are considered as criteria to be compared with SMAF, as follows.

HEFT: In this algorithm, function scheduling strategy is decided with the aim of optimizing makespan, while the running environment refresh is not considered. This algorithm can represent the scenario where security performance is not considered.

CSHEFT: In this algorithm, all running environments will be refreshed after the execution of function. This is also known as "cold start" in serverless computing. Therefore, this algorithm can represent the scenario where HEFT is directly applied to the serverless workflow without the consideration of environment refresh strategy.

To evaluate the effectiveness of SMAF, makespan and security performance is measured to compare SMAF with HEFT and CSHEFT. The weight of security performance is set to satisfy the makespan constraint. The average evaluation results of synthetic workflow and Montage workflow are shown in Fig. 1. Comparing with HEFT algorithm, we observe that the makespan of SMAF algorithm is a little longer , but the security performance is greatly enhanced, especially for the scenario with little available containers. When there are adequate containers for the workflow functions, the effect of our framework is weaken. This is because HEFT will tend to schedule functions to different containers with enough containers, which also reduce the risk of reusing containers. As for the CSHEFT algorithm, no new security threats will be introduced at the execution of workflow, thus CSHEFT can obtain the best security performance in these algorithms. However, the makespan is much longer as the cold start of all functions. If there is a constraint for makespan, the only way for CSHEFT is to increase the number of available running environments. Besides, inferring from Fig. 1, increasing the number of containers can not always decrease makespan, beacuse of the data transmission time between functions. These factors will limit the adoption of CSHEFT. Comparing the results for different workflows, we observe that workflow with more functions also need more running environments. As SMAF can select the environment refresh strategy carefully, a better tradeoff between security and makespan can be obtained with limited resources. When a suitable number of containers is set (3 containers for synthetic workflow and 5 containers for Montage workflow), the security performance has around 15 times increase, with around 0.44% makespan cost comparing with traditional HEFT algorithm.

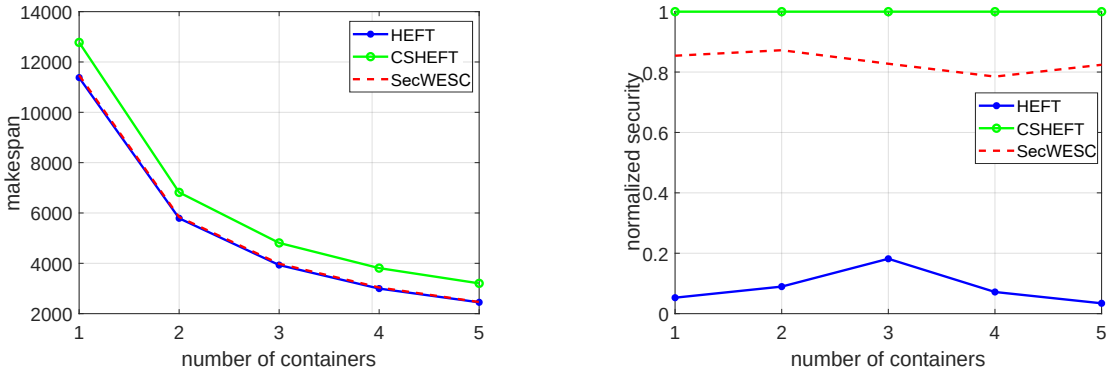

Figure A1 The average evaluation results of Montage workflow. (a) The makespan of Montage workflow with 1000 functions; (b) The makespan of Montage workflow with 1000 functions.

* Corresponding author (email: guolizihao@hotmail.com)