Inter-container Communication Aware Container Placement in Fog Computing

El Houssine Bourhim, Halima Elbiaze, Mouhamad Dieye
Université du Québec À Montréal, Montreal, Quebec, Canada

***Abstract*—In recent years, fog computing has increasingly become popular with the advent of Internet of Things (IoT) applications characterized by strict Quality of Service (QoS) requirements. To deploy applications, applications are typically decomposed into services then embedded with fog nodes. However, an overlooked aspect in container placement strategies is the heterogeneous inter-container network communication technologies and their impact on application performances in fog networks. We propose and evaluate in this paper, a near optimal genetic algorithm based container placement strategy that takes into account Remote Direct Memory Access as well host and overlay mode for inter-container communication to ensure application response time requirements.**

Keywords: Fog computing, RDMA, Genetic algorithms, container placement, Service chaining, Inter-container communication

I. Introduction

Recent years has seen an increasing prevalence of applications with stringent latency (e.g under 10ms) requirements in order to properly operate. As such, deployment and execution over cloud is unsuitable as latency worsens from the overhead induced by inter-cloud communications. Fog computing is an emerging paradigm, introduced to tackle these challenges by extending the processing, storage and networking capabilities to the network edge. By enabling processing near end-user locations, low latency can be achieved [1].

A technique to provision network applications consist of decomposing applications into chained micro-service which are then deployed across the network. However, given applications' low-latency requirements and in the context of fog computing, heavy virtualization overhead combined with high virtual machines (VM) startup times are major concerns. Moreover, as fog nodes typically have less capacity (in terms of processing, storage, communication, etc.) compared to cloud nodes, the number of virtual machines that can be deployed is limited. Increasingly, containerization is being advocated especially for fog computing. It operates on the premise of making virtual instances share the kernel of an operating system while retaining only relevant libraries, drivers and binaries for a given application, thereby reducing the resources needed during the computational process.

A critical task to minimize costs and ensure satisfaction of end-users' Quality of Service (QoS) requirements namely response time and isolation consist of optimally placing containers within heterogeneous fog nodes (i.e, CPU, storage, communication fabric, etc.). However, key challenges must however be addressed.

First, an overlooked aspect is the inter-container network communication technology which has been shown to have significant impact on application performance [2], particularly when container migration is considered. Containers can communicate mainly through three modes: (1) Host mode, (2) Overlay mode and more recently (3) Remote Direct Memory Access (RDMA) [3]. In summary, in host mode, containers are binded within their host's network namespace. In that, all of the network interfaces defined on the host are accessible to the container thereby allowing near bare metal performance and avoid the use of NAT [2]. Overlay mode (e.g, Weave, Flannel, Calico, etc.) uses networking tunnels to allow communication across hosts, thus enabling containers to operate as though in an intra-host network. Each host runs a software router which connects all containers on the host. It is also referred as bridge mode as the hosts are connected to each other through a bridge. RDMA tackles the data transfer overhead which constitutes one of the major factors to network latency. It reduces network latency and host's CPU utilization by allowing access to data from the memory of a host to another without any OS involvement. As kernel related overhead is reduced, an increase in throughput is typically observed. This is confirmed in our experimental evaluations.

Another important aspect to consider relates to isolation requirements for the sake of system stability and security, especially given that end-user devices are sometimes used as fog nodes. Because containers share a single underlying OS, it is more challenging to provide isolation in a container-based system. Containers mainly rely on the use of namespace and cgroups to provide isolation, which enables several services to be run in isolation on different containers on the same host. Unfortunately, documented inefficiencies within the kernel have shown notable detrimental impact on shared container performances [4]. Each inter-container network communication technology has a varying level of isolation benefits with overlay and host mode providing respectively the most and least isolation. There is however a performance trade-off [5]. These considerations significantly influence the container placement strategy.

Although they share some similarities in term of research issues, container placement problems for cloud networks and fog networks are distinct by the fact that fog nodes are more volatile, less reliable and less capable than cloud nodes. These considerations have significant impact on service placement. Very few works have tackled problems related to container placement in fog computing, among them [6] where

978-3-903176-24-9

a service placement policy is proposed, leveraging a two-phase partition based algorithm with the objective to increase service availability and QoS satisfaction rate. To our knowledge, [7] is the only work investigating communication aspects between containers in the context of container placement in large scale data centers. Also, none of the existed work have considered, in the context of fog computing networks, heterogeneous inter-container network communication fabric and their impacts on the satisfaction of stringent application performance and isolation requirements. Thus, we propose in this paper an efficient container placement heuristic derived from Genetic algorithm (GA) that aims to minimize response time while taking into account fog node and inter-container network communication fabric heterogeneity as well as isolation requirements for applications deployed on fog computing networks.

The remainder of this paper is organized as follows. Section II describes our system model. In Section III, we provide an Integer Linear Programming formulation of the container placement problem in fog computing networks. We then detail our proposed heuristics in section IV to solve the previously described problem. Finally, we present our simulation results in Section V before concluding in Section VI.

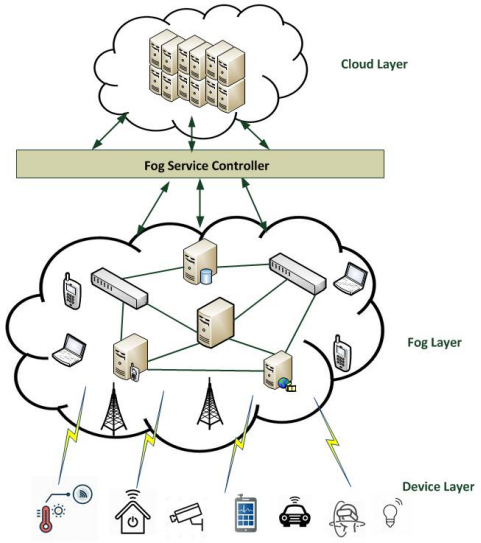


Fig. 1: Fog computing architecture

II. System Model

We present in this section, our system model depicted in Fig. 1, based on the hierarchical three-layer fog architecture. It is composed of: (1) a device layer which consists of different IoT devices, (2) a fog layer located at the edge of the network consisting of a considerable collection of fog nodes able to compute, transmit and store data, and (3) a cloud layer consisting of multiple high-performance servers and storage devices. Our container placement process is shown in Fig. 2, centered around a fog service controller residing between the fog layer and cloud layer. Its role consist of managing and controlling fog resources within a predetermined geographic area.

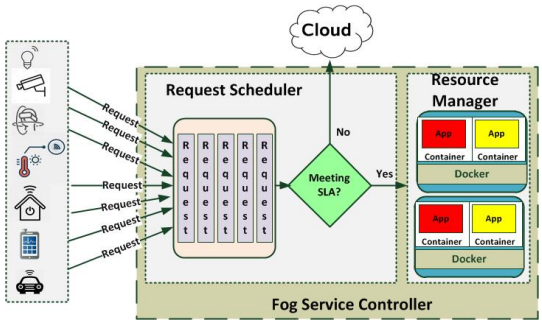


Fig. 2: Fog Service Controller

We consider a scenario where an application is decomposed into a set of chained containerized micro-services, to be deployed within the fog layer. The fog service controller has two components: a request scheduler and a resource manager. The request scheduler redirects incoming requests either to the cloud or fog layer, depending on whether or not the required application QoS requirements can be satisfied based on a predefined threshold. When feasible, the resource manager determines and effectively allocates the required containers to fog nodes in the fog layer. Furthermore, as fog nodes may be heterogeneous in terms of communication protocols supported depending on its physical network interface, the chaining of containerized micro-services also requires various inter-container network communication fabric (e.g RDMA, overlay, etc.) to be taken into account given its potential impact on application performance and isolation requirements [2, 5]. Hence, RDMA-based technology could be available for certain high capacity nodes (in terms of RAM, CPU, PCIe resources, etc.) such as base stations, high-end routers or workstations in contrast to host-based and overlay-based being accessible to lower capacity nodes (access points, mid-range PC, etc.).

Our objective in this paper consist of finding optimal locations to deploy containerized micro-services within the fog layer where heterogeneous fog nodes (capacity-wise) and inter-container network communication fabric varieties are considered and such that delay and isolation requirements are ensured.

III. Problem Formulation

In this section, we formulate the problem of container placement and service chaining in fog computing environment as an ILP optimization problem.

Let U be the set of all participating end users/IoT device connected to the fog system. Let S be set of services for which container placement in the fog is required. Let C denote the set of containers to be deployed on the fog nodes. We assume the fog network to be composed by a set F of fog nodes. Each fog node is characterized by its capacities in terms of CPU P, RAM M and storage L.

Our model considers a full mesh network topology for the fog layer, for simplicity purposes. However, note that it could be trivially extended to take into account a routing layer.

For simplicity, assume a container to host a single service. Hence, consider $s_{k,i} \in S_k$, a given service i composing an application A_k, deployed within a container i. Let $X_{s_{k,i}}^{f_j} = \{0, 1\}$ denote whether a service i of an application A_k hosted by a container i is placed on a fog node f_j. Let $|A_k|$ denotes the number of services composing the application A_k.

We summarize the container placement problem in a fog computing environment as follows :

$$\min \ R_{A_k}, \quad \forall \ A_k \in A \tag{1}$$

$$\sum_{f_j}^{F} \sum_{s_{k,i}}^{A_k} X_{s_{k,i}}^{f_j} = |A_k|, \quad \forall A_k \in A \tag{2}$$

$$\sum_{s_{k,i}}^{A_k} C_{s_{k,i}}^{\alpha} \cdot X_{s_{k,i}}^{f_j} \leq R_{f_j}^{\alpha}, \quad \alpha = \{P, M, L\}, \forall f_j \in F \tag{3}$$

$$R_{A_k} \leq D_{A_k}, \quad \forall A_k \in A \tag{4}$$

The objective function (1) aims to minimize the response time for each application in our application set A while ensuring the QoS requirements in terms of delay threshold abiding by the capacity and communications constraints are guaranteed.

In (2), we ensure that all services composing a given application are placed within the fog network. Through constraint (3), we make sure capacity (CPU, RAM and storage) limitation constraints are respected for corresponding fog resources.

We capture through (4), the fact that service placement must be done such that the resulting response time do not violate the predefined application deadline D_{A_k} with R_{A_k} denoting the application response calculated by equation (5).

$$R_{A_k} = D_{A_k}^t + M_{A_k} + d_{A_k}, \quad \forall A_k \in A \tag{5}$$

$D_{A_k}^t$ denotes the required deployment time, M_{A_k} represents the application makespan duration required for services to be executed on fog computing nodes and d_{A_k}, further detailed below, denotes the communication delay dependent both of container placement and the corresponding inter-container network communication mechanism used between containers.

$$d_{A_k} = 2 \cdot \sum_{j=1}^{F} d(U_k, X_{s_{k,1}}^{f_j}) + \sum_{i=1}^{n-1} \sum_{j=1}^{F} \sum_{j'=1}^{F} d(X_{s_{k,i}}^{f_j}, X_{s_{k,i+1}}^{f_{j'}}) + d(X_{s_{k,n}}^{f_j}, X_{s_{k,1}}^{f_{j'}}), \tag{6}$$

$$d(X_{s_{k,i}}^{f_j}, X_{s_{k,i+1}}^{f_{j'}}) = \begin{cases} d_{f_j,f_{j'}} & \text{if } X_{s_{k,i}}^{f_j} = 1 \text{ and } X_{s_{k,i+1}}^{f_{j'}} = 1 \\ d_{f_j,f_j} & \text{if } X_{s_{k,i}}^{f_j} = 1 \text{ and } X_{s_{k,i+1}}^{f_j} = 1 \end{cases}$$

The application communication delay d_{A_k} includes the delay between the end-user and the fog node on which the container hosting the first service of the service chain is placed, the delay between subsequent containers, the delay between the last container to the first container and finally the delay between the first container back to the end-user. Further note that the intra-node delay as well as the inter-node communication delay are also considered in addition to the delay induced by different inter-container communications mechanisms.

IV. Heuristics

The problem described above is NP-hard to solve as it generalizes the bin-packing problem with variable bin sizes and prices. Furthermore, the service placement problem has been specifically shown to be NP hard [8, 9]. Hence, calling for an efficient heuristic.

A. Greedy Algorithm

We first propose a greedy algorithm for the problem described above. Our greedy strategy consists of searching and placing the first container in fog nodes near the end-users while keeping in mind of the constraints (2), (3) and (4). We subsequently conduct an exhaustive search to identify the most suitable fog nodes for the remaining containers with regards to the objective function defined in equation (1) until every container is placed. Details of our greedy algorithm are given in Algorithm 1.

B. Genetic Algorithm

We present in this section the design of our hybrid genetic algorithm (GA) shown in Algorithm 2. Typically, each potential solution in GA is considered as an individual and represented by a chromosome, which in turn consists of genes encoding specific characteristics of the individual. In our work, a chromosome corresponds to a container placement plan for a given application. Each gene in a chromosome denotes a certain placement of a container on a specific fog resource.

The first step of a GA is to derive an initial population from which the evolution starts. The selection operator consist of selecting individuals with the aim of creating offsprings for the next generation. We adopt a tournament-based selection mechanism [10] in which individuals (initially randomly picked) compete based on their fitness. The fitness metric measures for an individual, the potential to contribute to the optimal solution. In this work, it is computed based on the delay. The next step is the crossover operation in which parent chromosomes are split at a crossover point and re-combined to create new offsprings. Given here that a chromosome constitutes a container placement plan, the crossover operation enables sufficient exploration of other fog nodes, thus avoiding local optima and induce convergence. Our adopted strategy is the single-point crossover operation[11]. To introduce randomness and further introduce diversity between offsprings generated after crossover, a mutation operation may occur, with a certain probability, on specific genes of a chromosome. Afterwards, the new population is evaluated with the objective of finding container placement schemes that have optimal fitness value. The new population goes through another iteration of selection, crossover, mutation, and evaluation until no evolution occurs for a predefined number of generations.

Algorithm 1: Greedy algorithm

Input: C, F, Delay matrix
Output: Mapping matrix c_i, f_j
$F_{alloc} \leftarrow \emptyset$; $C_{dep} \leftarrow \emptyset$
for *each* $c_i \in C$ do
  # the fog nodes are sorted in ascending order
  for *each* $f_j \in F$ do
    # verify that constraints (2), (3) and (4) are respected
    if f_j *can host* c_i then
      $F_{alloc} \leftarrow F_{alloc} \cup \{f_j\}$
      $C_{dep} \leftarrow C_{dep} \cup \{c_i\}$
      $C \leftarrow C \setminus \{c_i\}$
  if $C_{dep} \neq \emptyset$ then
    Update Placement with $Placement(C_{dep}, F_{alloc})$
return Placement(C_{dep}, F_{alloc})

Algorithm 2: CPGA Container Placement based on Genetic Algorithm

Input: C, F, Delay matrix
Output: Mapping matrix c_i, f_j
 # Set of parameter:
 # population size (pop_size), number of generation (gen_num), tournament size (k), crossover probability (p_c), mutation probability (p_m)
 # Set of containers C, and set of fog nodes F are used to generate the initial Population
$POP \leftarrow$ population of pop_size number of random individuals
$Best \leftarrow \infty$
$Iter \leftarrow 1$
while *terminating condition not true* ($Iter < gen\_num$) do
  # Selection
  $P_1 \leftarrow$ Tournament_selection(POP)
  $P_2 \leftarrow$ Tournament_selection(POP)
  # Crossover or mutation operator
  $R \leftarrow random(0, 1)$
  if $R < p_c$ then
    $POP\prime \leftarrow$ Crossover(P_1, P_2)
  if $R < p_m$ then
    $POP\prime \leftarrow$ Mutation(P_1)
  # update the population
  $POP \leftarrow POP \cup POP\prime$
  # evaluation
  $newBest \leftarrow \min_{i \in \mathcal{POP}} fitness(i)$
  if $newBest < Best$ then
    $Best \leftarrow newBest$
  $Iter \leftarrow Iter + 1$
return *Best Individual*

V. Experimental Results

A. Evaluation environment

In our simulations, we assume that the fog computing nodes are within the same domain, thus under a single administrative control. Furthermore, we also consider the fog nodes to be static. We leave mobility and multi-domain/operator aspects for future research. Also note that each end-user request is processed according to their order of arrival.

The number of generations is set to 500 for our GA-based container placement heuristic in addition of the population size being set constantly at 100. As previously mentioned, the selection mechanism used is the tournament selection strategy with a size equal to 2. The crossover and mutation operation rate is respectively set at 0.07 and 0.01. An optimal container placement plan is selected after 500 generations.

We run our simulations using a physical machine running Ubuntu 18.04, with Intel Xeon(R) E5 with 32 CPU cores, 2GHZ and 64GB of RAM.

In our evaluation, we consider a fog network with ten fog nodes, the network topology is generated using BRITE [12], we consider twenty different applications, each application leveraging three services deployed on three containers.

For this work, in order to find optimal container placement schemes to satisfy stringent application QoS requirements, we compared the performances of three optimization algorithms previously described : ILP implemented using the CPLEX library [13] and also serving as the baseline for our evaluations, greedy and finally a GA-based container placement algorithm named CPGA. A summary of our experimental parameters is shown in Tables I, II and III.

B. Results and Discussion

We mainly focus on investigating the impact of various inter-container networking technologies with regards to application performance and isolation requirements. More specifically, we look at the following metrics to evaluate the effectiveness of a container placement plan : (1) the achieved response time by the fog layer to an application request for service; (2) the number of activated fog nodes to measure resource usage as well as projected the energy consumption; and finally algorithm execution time to indicate scalability potential as the number of requests increases over time. We simulated three scenarios, summarized in Table I in which the available inter-container network communication fabric is varied. We first investigated using ILP, the average application response time for each simulated scenario with regards to the number of request received by the fog service controller, as shown in Fig.3. It can be clearly observed that lower response time are achieved in scenarios with

Scenario	Number of nodes (RDMA)	Number of nodes (Overlay)	Number of nodes (Host)
Scenario 1	4	4	2
Scenario 2	2	4	4
Scenario 3	0	5	5

TABLE I: Scenarios

Fog nodes	CPU (MIPS)	RAM (MB)	DISK (MB)
Node1	1000	1000	3000
Node2	500	250	500
Node3	2000	1500	4000
Node4	1000	1000	3000
Node5	1500	1000	2000
Node6	1000	1000	1000
Node7	500	500	1000
Node8	2000	2000	4000
Node9	1000	1000	2000
Node10	500	500	1000

TABLE II: Fog resources [14]

Services	CPU (MIPS)	RAM(MB)	DISK(MB)	$M_{A_k}(s)$
Application A_1: D_{A_1} = 20 s, $D^t_{A_1}$ = 15 s [15]				
Container1	200	50	50	0.8
Container2	200	50	50	0.8
Container3	200	50	50	0.8

TABLE III: Application resource demands

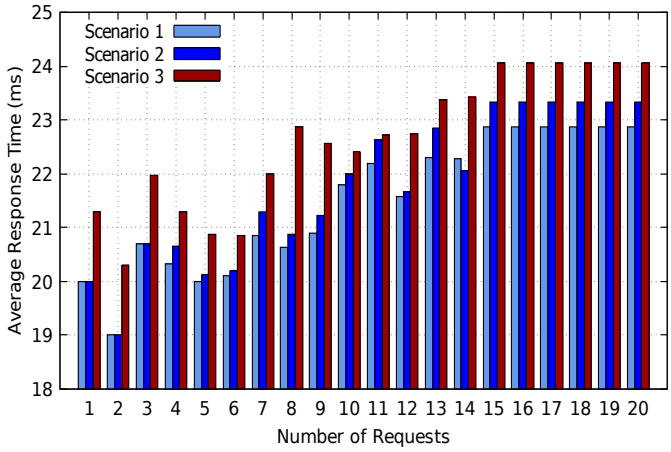


Fig. 3: Average response time per scenario

greater number of RDMA-enabled fog nodes. The impact of RDMA as an inter-container network communication fabric is therefore evident in terms of network and application performance improvement, although as a tradeoff, higher prior capital expenditure to upgrade high-capacity fog nodes (given compatible network interfaces are required) must be also be considered. The performance gain can be explained from RDMA's ability to leverage zero-copy mechanisms with kernel bypassing to reduce unnecessary CPU processing.

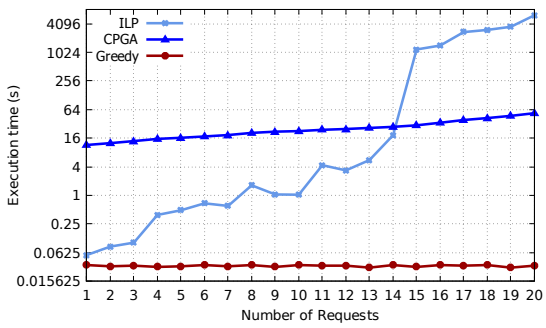
We also compared the performances of each container placement algorithm (i.e ILP, greedy and CPGA) previously described using scenario 2 as our simulation basis. Fig 4a illustrates in log-scale axis the execution time of each tested algorithm as the number of requests increases. Unsurprisingly, ILP become more and more computationally complex as the input variables increases with incoming application requests. In contrast, Greedy and CPGA show more stable evolution in terms of execution time indicating a potential suitability for large scale fog environments. It can be noted that Greedy exhibits the lowest execution time, explained by its relative simplicity in its container placement search which focuses on finding the nearest compatible fog nodes to deploy containers. However, its limits are fully displayed in Fig. 4b where we observed that the average response time achieved with respect to the number of requests by Greedy are far greater compared to CPGA and ILP. Hence highlighting the Greedy algorithm's inability to find near optimal container placement strategy as the number of incoming requests escalates. In contrast, CPGA reveals a better capability to discover and deploy containers at near optimal locations within the fog layer. In fact, we observe an average disparity of 10-15 ms between CPGA and the optimal response time obtained by ILP. We hypothesize that the crossover and mutation operators leveraged by CPGA enables better exploration of the search space, thus avoiding being stuck in local optima. It is worth noting that CPGA's execution time is also dependent of the number of generations considered. The tradeoff then being that CPGA may not converge to optimal schemes with fewer number of generations. To highlight this, we show in Fig. 4c the average response time achieved by CPGA with respect to the number of generations used. We note globally that CPGA converges towards near-optimal container placement schemes after 500 generations, thereby justifying why we set it to 500 generations for other simulations.

We further investigated the number of nodes used based on the number requests received for each algorithm tested. The results are shown in Fig. 4d where ILP seems to consume the fewest amount of fog resources. Naturally, as Greedy aggressively seek neighbour fog nodes for container placement result in high resource consumption. Obviously, high resource consumption has significant impacts, notably greater energy consumption and operating expenditures. Note that all tested algorithms converge to similar amount of nodes used as the number of requests increases in the long term.

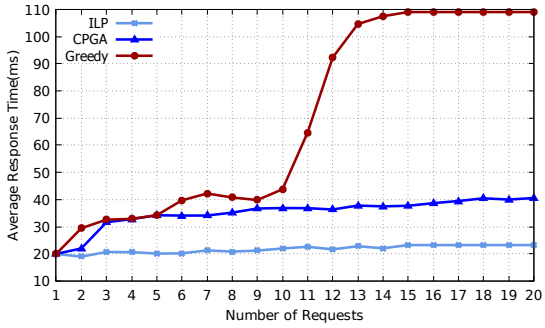
VI. Conclusion and Future Work

We presented in this work, a GA-based container placement strategy that takes into account heterogeneous fog nodes in addition to various inter-container network communication fabric. We investigated their impacts on application performance through realistic scenarios using ILP, greedy and CPGA approaches. Experimental evaluations has shown our algorithm CPGA to outperform other approaches tested. It has also highlighted the significant performance benefits in terms of application response times introduced by RDMA. By considering different inter-container network communication fabric with varying level of isolation provided, our container placement proposal also have accounted for isolation requirements.

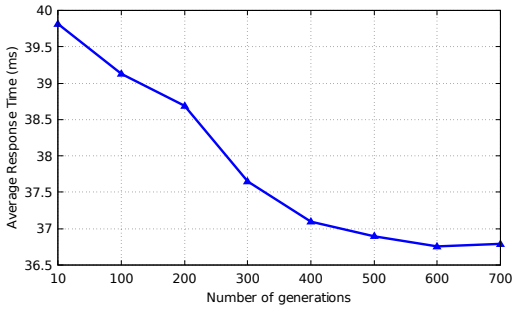
In our future work, we aim to implement a more realistic scenario considering the mobility of fog nodes and to improve the system model for resource provisioning in terms of reliability and availability of services.



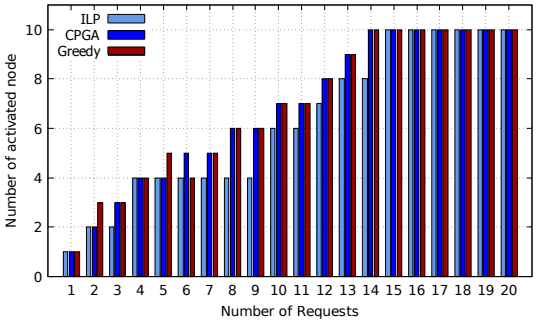
(a) Execution time



(b) Response Time



(c) Convergence of CPGA



(d) Resource usage

Fig. 4: Performance evaluation of container placement strategies.

REFERENCES

[1] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[2] U. Abbasi, E. H. Bourhim, M. Dieye, and H. Elbiaze, "A performance comparison of container networking alternatives," *IEEE Network*, vol. 33, no. 4, pp. 178–185, 2019.

[3] P. MacArthur and R. D. Russell, "A performance study to guide rdma programming decisions," in *IEEE Int. Conf. on High Performance Computing and Communication*, June 2012, pp. 778–785.

[4] G. Borello, "Container isolation gone wrong," Apr 2017. [Online]. Available: https://sysdig.com/blog/container-isolation-gone-wrong/

[5] T. Yu, S. A. Noghabi, S. Raindel, H. Liu, J. Padhye, and V. Sekar, "Freeflow: High performance container networking," in *ACM Workshop on Hot Topics in Networks*, 2016.

[6] I. Lera, C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet of Things Journal*, 2018.

[7] L. L. et al, "Communication-aware container placement and reassignment in large-scale internet data centers," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 540–555, 2019.

[8] L. Tang, H. Yang, R. Ma, L. Hu, W. Wang, and Q. Chen, "Queue-aware dynamic placement of virtual network functions in 5g access network," *IEEE Access*, vol. 6, pp. 44 291–44 305, 2018.

[9] X. Huang, S. Ganapathy, and T. Wolf, "Evaluating algorithms for composable service placement in computer networks," in *IEEE Int. Conf. on Communications*, June 2009, pp. 1–6.

[10] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University computing*, vol. 15, no. 2, pp. 56–69, 1993.

[11] C. L. Bridges and D. E. Goldberg, "An analysis of reproduction and crossover in a binary-coded genetic algorithm," *Grefenstette*, vol. 878, pp. 9–13, 1987.

[12] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *IEEE Int. Sym. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2001.

[13] IBM, *CPLEX Optimizer*, (accessed April 12, 2019). [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[14] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, pp. 1275–1296, 2017.

[15] A. Ahmed and G. Pierre, "Docker container deployment in fog computing infrastructures," in *2018 IEEE International Conference on Edge Computing (EDGE)*, July 2018, pp. 1–8.