Scalable Container placement in Edge Computing

Vivek Mehta
Vellore Institute of Technology, Vellore, India
vivek.mehta2018@vitstudents.ac.in vivek.rm17@gmail.com

Abstract

Containers and Edge computing both play pivotal roles in technological development. They have become the backbone of application development. This paper proposes a novel strategy to place containers in Edge nodes. It is based on class-constrained BestFit algorithm, a bin packing algorithm. The novel algorithm mainly focuses on scalability and resource consumption of containers. The novel algorithm is then experimented and compared against other algorithms.

Index Terms

Containers, Edge Computing, Scalability, Resources, BestFit, Cloud Computing, Container placement

I. Introduction

EDGE computing has become the new paradigm of cloud computing, reshaping the IT sector. It allows computer processes and storage of data to happen closer to user. Therefore, it has become the preferred choice for low latency applications. Edge computing also

Containers are light weight packages of software that contain essential elements to run in multiple environment. They are extensively used to deploy applications on cloud infrastructure. Compared to Virtual Machines, Containers are light weight and share the operating system kernel of the host.

Placement of Containers in Edge computing nodes has been one of the toughest problems faced by the community. Containers and Edge computing are relatively new technologies. Edge computing differs from traditional cloud computing as it is of fragile nature. It lacks proper cooling mechanism. Since, edge nodes are scattered throughout the location, it is difficult to maintain them. Scattered nodes also increase the cost of migration. Edge nodes can easily be damaged if they run computing intensive application as it might not have the resources required by the application. They cannot run at full capacity for longer periods of time. Edge nodes also differ from cloud computing in terms of heterogeneity as nodes may consist of different architecture and operating system in different locations.

Applications are now developed giving importance to scalability. Application services can receive surge of requests according to the type and time of the day. These services need to account for this surge, consuming more resources to handle it. The reputation of a company depends on how well the requests are handled. Therefore, edge service providers have to place containers in-order to handle the surges and not lose efficiency.

This paper attempts to solve the placement problem using class constrained bin packing algorithm. The paper places containers in edge nodes considering parameters such as resource consumption and scalability of containers. Containers are classified according to their ability to scale as defined by the developers. This novel approach is then compared to classic bin packing algorithms [21].

The remainder of the paper is organized as follows. Section 2 discusses related works for this paper. Section 3 discusses the novel algorithm and its architecture. Section 4 discusses the experimentation of the algorithm. Simulation results and observations are presented in Section 5. Section 6 concludes the paper along with a discussion of possible future works.

II. Previous Works

The placement problem has been extensively studied and researched. Multiple algorithms have been researched and surveyed. Bin packing algorithms [21] along with heuristic algorithms such as Ant Colony Optimization are most studied algorithms. [10] talks about different algorithms for virtual machine placement as well as parameters and challenges that determine the effectiveness of the algorithm.

Various research has been conducted using heuristic optimisation algorithms to solve the placement problem. For example, [9] proposes novel Ant Colony Optimisation algorithm. It exchanges idle virtual machines with active virtual machines to reduce power consumption. However, the algorithm is quadratic and doesn't consider latency as a parameter. The authors of [11] have used particle swarm algorithm based on annealing algorithm. It uses the simulated annealing algorithm to find the optimal solution in the large solution space.

Machine learning and Deep learning have also been used to find an optimal solution. [13,14] have used machine learning and deep learning models to find optimal mapping of virtual machine to physical machine. It discusses how virtual machines should be placed together in-order to reduce resource competition, while not incurring large network costs.

[7] proposes a mapping of containers inside virtual machines to physical machines according to their utilisation of virtual machines and physical machines both. The proposed algorithm tries to reduce resources wasted.[8] talks about on demand provisioning for a single OS-Application pair. However, it ignores energy consumption, locking of resources and variety in OS-Application pair.

The paper [2] looks at the placement of containers in edge computing keeping in mind the distributed attribute of the applications and other factors related to edge computing. The problem is re framed as a multi-objective optimization model or network-graph model. Containers can be placed optimally if the communication between containers can be understood. The authors of the papers [3,4,15,16] have provided with algorithms which are communication aware. The algorithm suggested tries to reduce the cost of communicating with other containers and services.

Paper [5] propose that the virtual machines should be placed in appropriate hosts to reduce the network traffic. It takes virtual machine resource demands, traffic and network topology as input. The algorithm maps virtual machines OpenFlow controls and hop counts for energy saving process. Authors of [6] solves the placement problem with respect to service providers. It looks at the power consumption of CPU.

There is limited research on placement of containers, aware of the scaling factor. The authors of papers [1,4] proposes algorithms which takes into account the increase of resource consumption while scaling up the service. However, the algorithm proposed in [1] relies on historical data of the services, which might not be available during placement.

TABLE I: Notations

C	Set of Containers
c_i	ith container
$class(c_i)$	Class of ith container
E	Set of Edge nodes
e_j	jth edge node
R_{max}	Resource consumption of jth node at peak
R_{cc}	Current resource consumption of jth node
R_{total}	Total resource available of jth node
W_i	Resource required by ith container
X_{ij}	1 if c_i is placed in e_j; else 0
N_e	Number of edge nodes active and running
M	Number of container migrations

III. Algorithm

A. Problem Formulation

Container placement is the process by which new containers are mapped to edge nodes. Placement of containers determine the performance of the service. It also determines the efficiency of the edge node. Sub-optimal placement of containers can hamper the service.

Scaling is a process by which they increase their engagement with their service, consuming more resources. These resources are then given up when the user engagement goes down i.e. the application scales down. Scaling of application can happen at any moment, depending on the nature of the application and location along with various other factors.

Containers are categorised according to their ability to scale in the future. This is pre-determined. Containers are placed into edge nodes according to it's class and resources available on the edge node. During the scaling process, containers of class A will scale up, consuming more resources. This can lead to other scaling containers to migrate to other edge nodes in order to satisfy their resource requirement.

Table 1 defines all the variables.

B. Objective

The main objective is to reduce the number of edge nodes by optimally placing containers.

$$min \sum_{n=1} N$$

At the same time, the paper also tries to reduce total resource loss (TRL) of the edge node. It is equivalent to the resources left unused in edge nodes which are in active state.

$$\text{TRL} = \sum_{j \epsilon N} (R_{Total)(j)} - R_{cc}(j))$$

$$min\ \text{TRL}$$

During the scaling, many containers will be moved from one edge node to another. This process is called by migration (M). Migrations can be be time consuming and hamper the service of the application. The algorithm will try to reduce the number of migrations during the scaling process.

$$min \sum M$$

C. Constraints

Resources available at any given edge node at a given time should be greater than the resource required by the container

$$R_{total} \text{ - } R_{cc} > W_{i(min)}$$

D. Online and Offline Algorithms

Online and Offline algorithm differs in their processing of the input. Online algorithms [19] process its input in a serial fashion while offline algorithm process the data as a whole. Offline algorithm [20] requires all the data before hand to run the algorithm. It perform all types of operations to fit the data for the algorithm. On the other hand, online algorithms can run without getting all the data. It processes the data as it gets it and runs the algorithm.

E. Class-constrained Best Fit Algorithm

BestFit algorithm is a solution to online bin packing [21] problem where it determines the minimum number of bins required to pack certain items. While placing items in the bin, it will find the bin with tightest space left which can accommodate the item.

Algorithm 1 BestFit

```
for each container item $i$ do
    Choose edge node $e_j$ with $argmin\ R_{Total} - W_i$   ▷ Edge node with tightest spot which can accomodate the container
    Check Constraints
end for
```

Our algorithm is a modified version of best fit algorithm. It determines the class of the container and then looks at the resource consumption during the scaling process. Container will only be packed in the edge node if it satisfies all the constraints.

Algorithm 2 Novel Class-constrained BestFit

```
for each container item $i$ do
    if $class(c_i)$ == A then
        Choose edge node $e_j$ with $argmin\ \|R_{Total} - R_{max} + W_{i(max)}\|$
        Check Constraints
    else if $class(c_i)$ == B then
        Choose edge node $e_j$ with $argmin\ \|R_{Total} - R_{max} + W_i\|$
        Check Constraints
    end if
end for
```

NOTE 1: The resource requirement of container $W_{i(max)}$ is the maximum resource required by the container when scaled up. This is defined by the developer of the application system.

IV. Experimentation

Experiment for the algorithm was coded using go programming language [17] . Go is a statically typed, compiled programming language. It is syntactically similar to C/C++, but with memory safe concurrency and automatic garbage collector. The

TABLE II: Notations for simulation

$C(A)$	Set of Containers of class A
$C(B)$	Set of Containers of class B
BF	BestFit algorithm
$CCBF$	Novel Class-Constrained BestFit algorithm
N	Number of Edge nodes required for placing all containers
TRL	Total Resource Loss after placement
M	Number of Migrations during scaling

TABLE III: Placement of containers before scaling

S.l.No	C(A)	C(B)	No. of Edge nodes	BF(N)	BF(TRL)	CCBF(N)	CCBF(TRL)
1	27	23	30	21	310	21	310
2	61	59	70	47	170	47	170
3	89	86	90	67	260	67	260
4	143	132	150	103	410	103	410
5	187	173	200	137	590	137	590

TABLE IV: Placement of containers after scaling

S.l.No	C(A)	C(B)	No. of Edge nodes	BF(N)	BF(TRL)	BF(M)	CCBF(N)	CCBF(TRL)	CCBF(M)
1	27	23	30	27	420	13	27	420	13
2	61	59	70	63	1000	37	62	850	36
3	89	86	90	88	1070	51	87	950	49
4	143	132	150	140	2200	81	138	1700	79
5	187	173	200	184	2260	106	182	2360	104

program written to experiment can be found in the GitHub link [16] . The main purpose of programming the simulation was to get familiar with other algorithm and enhance basic understanding of the algorithm.

The simulation was conducted in Apple MacBook Pro 13”, 2018 with 2.3 GHz Intel Core i5, 8 GB 2133 MHz LPDDR3.

The data used to simulate the experiment was randomly generated and stored in a CSV file. This data consists of both container resource requirements, class of the container as well as edge node resource capacity. The number of containers and edge numbers varied in the simulation.

V. Result and Analysis

All the simulation data was stored in an Microsoft excel file.

NOTE 2: During the experimentation, all the containers of class A are scaled up incrementally.

Table II data shows placement of containers using BestFit algorithm and our novel class constrained BestFit algorithm without enabling scaling. Both the algorithm performs equally with number of edge nodes required and Total Resource loss being the same. However, when the scaling is enabled, as shown by the data in Table III, our novel algorithm, class-constrained BestFit algorithm performs better than BestFit algorithm. The number of edge nodes required and migrations of containers during scaling is less in comparison to BestFit algorithm.

As the number of containers and edge nodes increase, the superiority of the novel algorithm increases. The difference in the performance is low when the number of edge nodes and containers are less (can insert a better word). The graph clearly suggests that the lines diverge with increase in containers and edge nodes.

NOTE 3: Custom algorithm in the figure [1,2,3] legend is our novel class-constrained BestFit Algorithm.

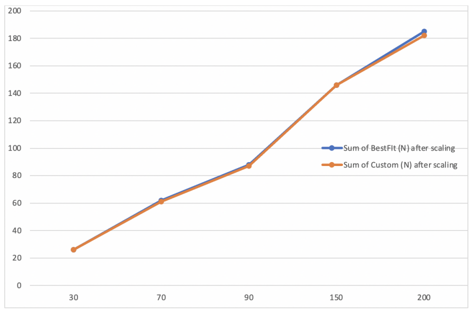

Fig. 1: Number of edge nodes required to accommodate all the containers during scaling

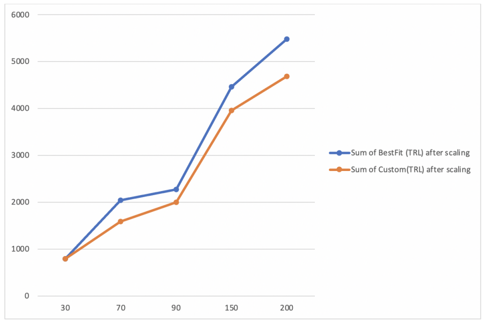

Fig. 2: Total resource loss of edge nodes after accommodating all the containers during scaling

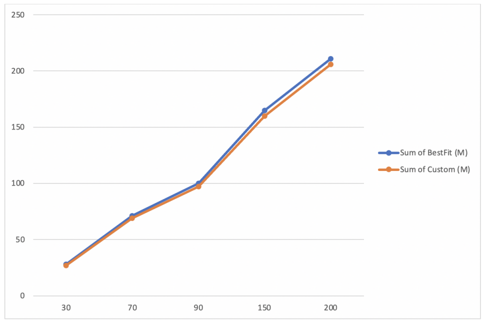

Fig. 3: Migrations of containers during scaling of containers

VI. Conclusion

In this paper, a novel algorithm is presented for container placement in edge computing. The proposed algorithm ensures optimal placement while also keeps in mind the scalability i.e. increase in resource consumption in mind, to reduce the number of migrations and number of edge nodes used. The novel algorithm is then compared with Bin packing BestFit algorithm [21]. When scaling is not enabled, both the algorithm performs equally. On the other hand, when scaling is enabled, the novel algorithm turns out to be superior with increasing containers and edge nodes.

In this study, we compared our novel algorithm with BestFit algorithm only. In the future, we will compare it with state-of-art algorithms with real world data for better understanding. Furthermore, the algorithm can also be expanded to consider various other factors which are essential to container placement. Different factors like latency, power consumption can be tested to make the algorithm fit for real world applications.

REFERENCES

[1] Rahmani, S., Khajehvand, V. Torabian, M. Burstiness-aware virtual machine placement in cloud computing systems. J Supercomput 76, 362–387 (2020). https://doi.org/10.1007/s11227-019-03037-8

[2] Hu, Yang Zhou, Huan Laat, Cees Zhao, Zhiming. (2019). Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Future Generation Computer Systems. 102. 10.1016/j.future.2019.08.025.

[3] L. Lv et al., ”Communication-Aware Container Placement and Reassignment in Large-Scale Internet Data Centers,” in IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 540-555, March 2019, doi: 10.1109/JSAC.2019.2895473.

[4] Sridharan R. and Domnic S. 2021. Placement for Intercommunicating Virtual Machines in Autoscaling Cloud Infrastructure: Autoscaling and Intercommunication Aware Task Placement. J. Organ. End User Comput. 33, 2 (Mar 2021), 17–35. DOI:https://doi.org/10.4018/JOEUC.20210301.oa2

[5] Shao-Heng Wang, P. P. -. Huang, C. H. -. Wen and L. Wang, ”EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks,” The International Conference on Information Networking 2014 (ICOIN2014), 2014, pp. 220-225, doi: 10.1109/ICOIN.2014.6799695.

[6] Zhang, Xinqian Wu, Tingming Zhou, Junlong Wei, Tongquan Chen, Mingsong Hu, Shiyan Buyya, Rajkumar. (2018). Energy-Aware Virtual Machine Allocation for Cloud with Resource Reservation. Journal of Systems and Software. 147. 10.1016/j.jss.2018.09.084.

[7] Mohamed K. Hussein, Mohamed H. Mousa, and Mohamed A. Alqarni. 2019. A placement architecture for a container as a service (CaaS) in a cloud environment. J. Cloud Comput. 8, 1, Article 131 (December 2019), 15 pages. DOI:https://doi.org/10.1186/s13677-019-0131-1

[8] J. Sung, S. Han and J. Kim, ”Virtual Machine Pre-Provisioning for Computation Offloading Service in Edge Cloud,” 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 2019, pp. 490-492, doi: 10.1109/CLOUD.2019.00087.

[9] M. K. Hossain, M. Rahman, A. Hossain, S. Y. Rahman and M. M. Islam, ”Active Idle Virtual Machine Migration Algorithm- a new Ant Colony Optimization approach to consolidate Virtual Machines and ensure Green Cloud Computing,” 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), 2020, pp. 1-6, doi: 10.1109/ETCCE51779.2020.9350915.

[10] P. D. Bharathi, P. Prakash and M. V. K. Kiran, ”Virtual machine placement strategies in cloud computing,” 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, pp. 1-7, doi: 10.1109/IPACT.2017.8244949.

[11] M. Zeyu, H. Jianwei and C. Yanpeng, ”Virtual Machine Scheduling in Cloud Environment Based on Annealing Algorithm and Improved Particle Swarm Algorithm,” 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), 2020, pp. 33-37, doi: 10.1109/ICAIIS49377.2020.9194890.

[12] M. Duggan, K. Flesk, J. Duggan, E. Howley and E. Barrett, ”A reinforcement learning approach for dynamic selection of virtual machines in cloud data centres,” 2016 Sixth International Conference on Innovative Computing Technology (INTECH), 2016, pp. 92-97, doi: 10.1109/INTECH.2016.7845053.

[13] Chiang, R.C. Contention-aware container placement strategy for docker swarm with machine learning based clustering algorithms. Cluster Comput (2020). https://doi.org/10.1007/s10586-020-03210-2

[14] M. Gamsiz and A. H. Özer, ”An Energy-Aware Combinatorial Virtual Machine Allocation and Placement Model for Green Cloud Computing,” in IEEE Access, vol. 9, pp. 18625-18648, 2021, doi: 10.1109/ACCESS.2021.3054559.

[15] E. H. Bourhim, H. Elbiaze and M. Dieye, ”Inter-container Communication Aware Container Placement in Fog Computing,” 2019 15th International Conference on Network and Service Management (CNSM), 2019, pp. 1-6, doi: 10.23919/CNSM46954.2019.9012671.

[16] Michael Sindelar, Ramesh K. Sitaraman, and Prashant Shenoy. 2011. Sharing-aware algorithms for virtual machine colocation. In Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures (SPAA ’11). Association for Computing Machinery, New York, NY, USA, 367–378. DOI:https://doi.org/10.1145/1989493.1989554

[17] https://github.com/vortex-17/Container-Edge-Placement

[18] https://golang.org/doc/

[19] https://www.cs.cmu.edu/ avrim/451f13/lectures/lect1107.pdf

[20] https://xlinux.nist.gov/dads/HTML/offline.html

[21] https://en.wikipedia.org/wiki/Bin_packing_problem