# Container-based Service State Management in Cloud Computing

Authors

**Shubha Brata Nath, Sourav Kanti Addya, Sandip Chakraborty and Soumya K Ghosh**

**Department of CSE**
**IIT Kharagpur | NITK Surathkal**

# Table of contents

# Introduction

# Problem Statement

To design an efficient mechanism such that an idle container can be resumed quickly to prevent the loss of the application's quality of service (QoS).

# Solution

- We propose an exciting direction to significantly boost up the consolidation ratio of a data-center environment by effectively managing the container's states
- We observe that many cloud based application services are event-triggered, so they remain inactive unless some external service request comes.
- We exploit the fact that the containers remain in an idle state when the underlying service is not active, and thus such idle containers can be checkpointed unless an external service request comes.
- We have implemented the system, and the evaluation is performed in Amazon Elastic Compute Cloud. The experimental results have shown that the proposed algorithm can manage the containers' states, ensuring the increase of consolidation ratio.

# System Architecture

# System Architecture

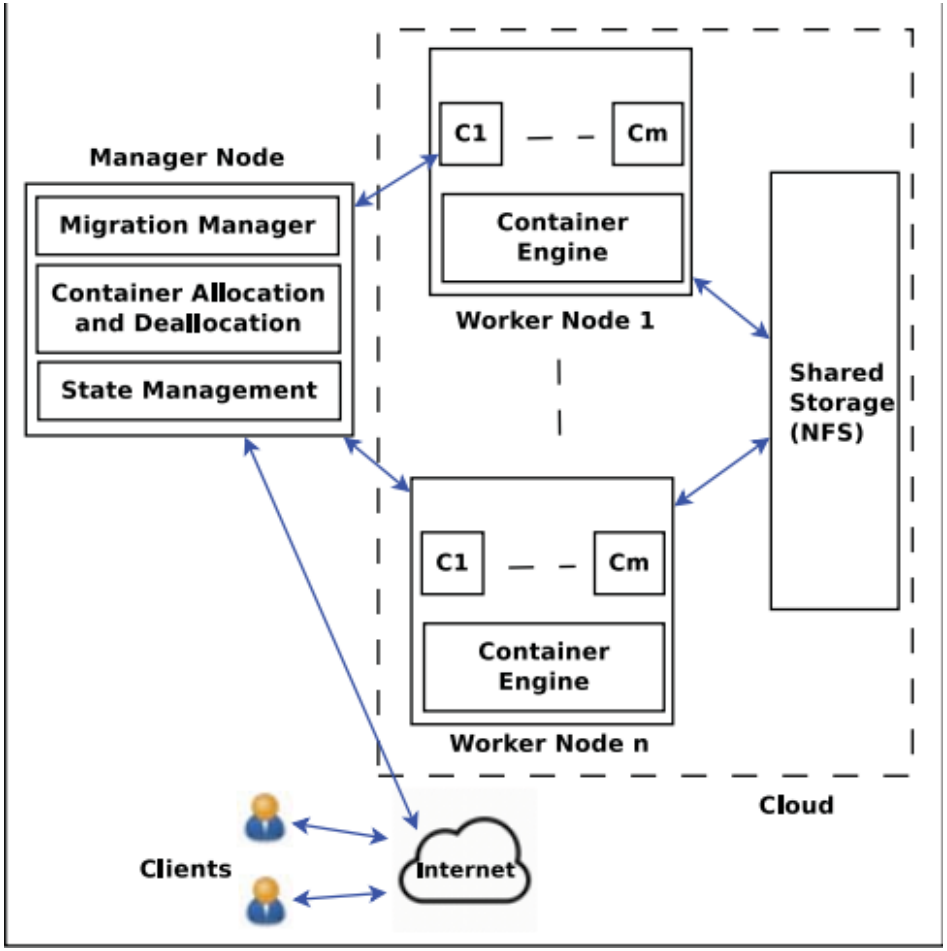


Figure: System architecture

# System Architecture

- Clients: Clients send service request to the manager node of the cloud data center.
- Manager Node: The manager node is connected to all the worker nodes of the cloud data center. The manager node has a container allocation and deallocation module to place the services in the cloud data center. This node does the aggregation of the results. Also, the final result is sent to the clients. We need to find the status of the container state (active/idle). This work is performed by the state management module.
- Cloud Worker Node: The cloud worker node has container engine (https://docs.docker.com/get-started/ overview/) to run different services as a container.
- Shared Storage using Network File System: The cloud worker nodes share a storage to save the state of the checkpointed containers. This shared storage is designed using NFS. NFS server is configured in a worker node that needs to share its directory with other worker nodes.

# System Architecture

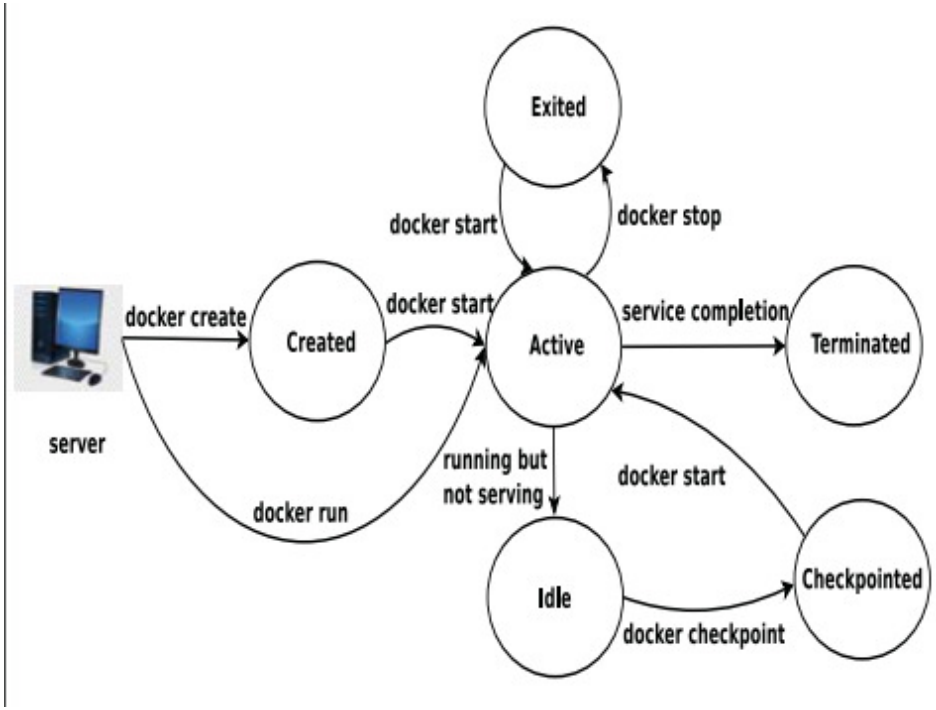


Figure: Docker Container

# Service State Management of Containerized Applications

# Service State Management of Containerized Applications

Let us consider $C$ containers and $S$ servers present in the system at a particular time instant. We denote $S = \{S_i : i \in (1, ..., s)\}$ and $C_i = \{C_j : j \in (1, ..., c)\}$ as sets of cloud data center servers and containers respectively.

## Average Startup Latency of the Services

$$Avg_{\text{startup\_latency}} = \frac{\sum_{n=1}^{c} Startup\_latency_i}{c}$$

## Average Response Time of the Services

$$Avg_{\text{resp\_time}} = \frac{\sum_{n=1}^{c} Resp\_time_i}{c},$$

# Service State Management of Containerized Applications

## Constraints

$$\sum_{j=1}^{TC_i} C_j^{CPU} \times x_{i,j} \leq Re_i^{CPU} \times y_i \quad (1)$$

$$\sum_{j=1}^{TC_i} C_j^{RAM} \times x_{i,j} \leq Re_i^{RAM} \times y_i \quad (2)$$

$$\sum_{j=1}^{TC_i} C_j^{BW} \times x_{i,j} \leq Re_i^{BW} \times y_i \quad (3)$$

# Parameters In Amazon EC2 Experiment

# Parameters In Amazon EC2 Experiment

| Parameter | Value |
|---|---|
| Number of Amazon EC2 virtual machine instances or worker nodes | 10 |
| Processor of a worker node | 2.5 GHz Intel Xeon Family |
| Number of vCPU in a worker node | 1 |
| Storage of a worker node | 8 GB |
| Operating System used | Ubuntu Server 18.04 LTS |
| Maximum available RAM of the worker node | 680 MB |
| Maximum available RAM of the manager node | 1780 MB |

# Conclusion

# Conclusion

- We have performed the container-based service state management to maximize the cloud data center's consolidation ratio. In our proposed approach, the idle containers need to be checkpointed to save the hardware resources. However, after the end of the idle period, the containers need to be resumed from the last saved state.
- The idle containers are migrated to a new server if the current server does not have sufficient resources to run it. The problem of state management of the containers is formulated as an optimization problem that maximizes the consolidation ratio as the objective function.
- The evaluation in Amazon EC2 shows that the proposed algorithm is able to maximize the consolidation ratio ensuring the state management of the containers.
- Thus, the proposed algorithm can efficiently manage the service state.

# References

# References I


[1] E. Casalicchio, “Container orchestration: A survey,” in Systems Modeling: Methodologies and Tools. Springer, 2019, pp. 221–235.

[2] F. Rossi, V. Cardellini, and F. L. Presti, “Elastic deployment of software containers in geo-distributed computing environments,” in Proc. of IEEE ISCC’19, 2019.

[3] K. Suo, Y. Zhao, W. Chen, and J. Rao, “An analysis and empirical study of container networks,” in proceedings of the IEEE Conference on Computer Communications. IEEE, 2018, pp. 189–197.

[4] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, “Exploring container virtualization in iot clouds,” in 2016 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2016, pp.1–6.

# References II


[5] A. Celesti, D. Mulfari, A. Galletta, M. Fazio, L. Carnevale, and M. Villari, “A study on container virtualization for guarantee quality of service in cloud-of-things,” Future Generation Computer Systems, vol. 99, pp. 356–364, 2019.

[6] I. M. A. Jawarneh, P. Bellavista, L. Foschini, G. Martuscelli, R. Montanari, A. Palopoli, and F. Bosi, “Qos and performance metrics for container-based virtualization in cloud environments,” in Proceedings of the 20th International Conference on Distributed Computing and Networking. ACM, 2019, pp. 178–182.

[7] W. Li, A. Kanso, and A. Gherbi, “Leveraging linux containers to achieve high availability for cloud services,” in 2015 IEEE International Conference on Cloud Engineering. IEEE, 2015, pp. 76–83.

# THANK YOU