# Runtime Load Balancing in Docker Containers

**Prof. A. J. Kadam[1] Ms. Neha D. Pawar[2] Mr. Goraksha S. Vedpathak[3] Mr. Vaibhav A. Wadekar[4] Mr. Satyam T. Thorat[5]**

[1,2,3,4,5]Department of Computer Engineering

[1,2,3,4,5]All India Shree Shivaji Memorial Society's College of Engineering, Pune-411001, India

*Abstract*— In modern service-oriented computing paradigm, cloud computing and virtualization have played an important role. In order to achieve flexible deployment and high availability more conventional services are being migrated. Thus, our project contains introduction to a schedule algorithm based on fuzzy inference system (FIS), for global container resource allocation by evaluating node statuses using FIS.

***Key words:*** Docker, Container, Cloud, Dev-Ops

## I. INTRODUCTION

Problem of effectively managing CPU utilization when many containers share a single set of resources. The problem is exacerbated in environments where an application is designed to maximize performance by utilizing resources as efficiently as possible. For example, in high-performance computing, applications are routinely optimized for cache access and locality. The performance of such applications can be destroyed in virtualization environments by removing the benefits of cache locality; moreover, stalls can result for synchronized processes or threads. [12] The results obtained from experiment show that the presented infrastructure and schema derive optimal resource configurations and significantly improve the performance of cluster. [1] Since we observe that structure of load balancing is not clear, we use fuzzy logic in order to model the variant requirements of memory, disk space and bandwidth. As a result, the fuzzy algorithm performs better than other scheduling algorithms in terms of response time, data center processing time etc [7]. The objective of the project is to dynamically allocate the container at times of load. Impaired in recognition of objects.

### A. Docker

Docker was released as an open source project by "dot Cloud", a platform as a service company, in 2013. Docker relies on Linux kernel features, such as namespaces and c groups, to ensure resource isolation and to package an application along with its dependencies. This packaging of the dependencies enables an application to run as expected across different Linux operating systems—supporting a level of portability that allows a developer to write an application in any language and then easily move it from a laptop to a test or production[1] server—regardless of the underlying Linux Distribution. Docker provides an integrated user interface. It provides a greater level of simplicity.

### B. Docker Engine

Docker Engine is the underlying client-server technology that builds and runs containers using Docker's components and services.[4] When people refer to Docker, they mean either Docker Engine -- which comprises the Docker daemon, a REST API and the CLI that talks to the Docker daemon through the API or the company Docker Inc., which offers various editions of containerization technology around Docker Engine.

### C. Container

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings [4]. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, [5] for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.

## II. BACKGROUND

Historically, as server processing power and capacity increased, bare metal applications weren't able to exploit the new abundance in resources. Thus, VMs were born, designed by running software on top of physical servers to emulate a particular hardware system. A hypervisor, or a virtual machine monitor, is software, firmware, or hardware that creates and runs VMs. It's what sits between the hardware and the virtual machine and is necessary to virtualize the server. Within each virtual machine runs a unique guest operating system. VMs with different operating systems can run on the same physical server. Containers sit on top of a physical server and its host OS—for example, Linux or Windows. Each container shares the host OS kernel and, usually, the binaries and libraries. [6]

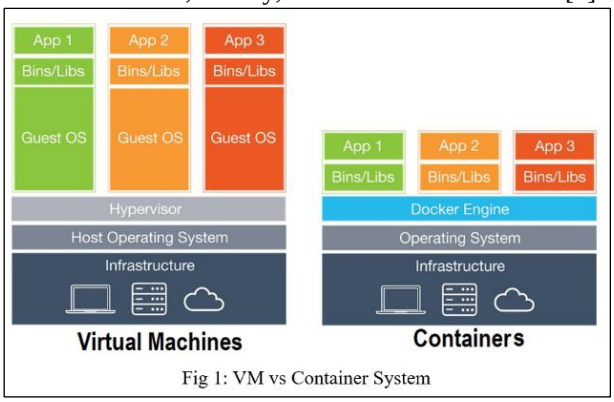



Fig 1: VM vs Container System

Virtual machines and containers differ in several ways, but the primary difference is that containers provide a way to virtualize an OS so that multiple workloads can run on a single OS instance. With VMs, the hardware is being virtualized to run multiple OS instances. Containers' speed, agility, and portability make them yet another tool to help streamline software development.

## III. RELATIVE MATHEMATICS

### *A. System Description*

*1) Input:*

Let 'S' be the

S= {D1, D2}

Where,

D1:(Set of container images)

D2:(set of all running container instances)

*2) Output:*

O = {D, L, A}

D= {Container Details}

L= {log files}

A= {alerts}

*3) Functions*

S= {F1, F2, F3, F4}

F1= {Data collection}

F2={Analysis}

F3={Generation}

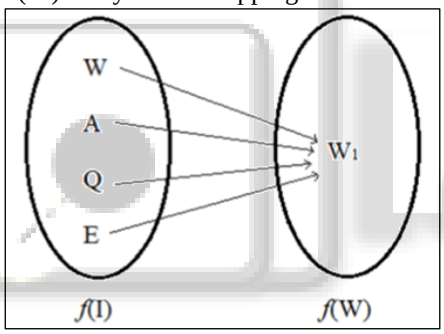
F4={Storing}

- Success Conditions: Containers Stabilized
- Failure Conditions: Error message for failure

### *B. Mapping Diagram*

f(r) be the rule which predicates work type provided that set of Input function

f (r): f (I) f (W) many to one mapping




*1) Dependency:*

- Relational dependency is: Output function is dependent on Input attributes
- Goal Function Input attributes
- i.e. f (W) f (I)

## IV. ALGORITHM/ARCHITECTURE

In a typical production environment, large groups of remote servers in a data center work under complex hierarchical layers with cross-domain cooperation. Resources are not only shared by multiple users but are also dynamically re-allocated [1] among containers on demand. Therefore, effective job scheduling, [8] load balancing and resource allocation become critical to ensure the scalability and high availability of a virtualization computing environment.

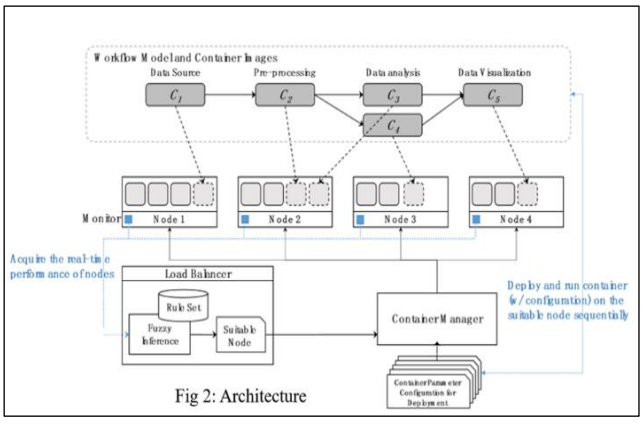



Fig 2: Architecture

In a CaaS framework, a container is the basic component that constitutes the business workflow, while a physical node (usually a server) is the fundamental carrier to deploy and execute containers. In practice, a cluster holds a large number of parallel workflows that execute independently. Each workflow varies on calculation time, data volume, network latency, submitted/completed time and other aspects, but they all share the resources in the same cluster. Therefore, it is difficult to accurately estimate the resource consumption for the entire workflow. For most static load balancing policies, the performance of the nodes is determined at the beginning of workflow execution.

First, we need data about each node to determine the state of that node. The collector collects data about each node from the /proc pseudo-file system, which contains the runtime information about the current host system, e.g. system memory, devices mounted, hardware configuration and etc. (1) CPU usage: Various statistics about the CPU status can be obtained from /proc/stat, and we use the following formula to compute the final crisp value of CPU usage in percentage It will cause all tasks in that workflow always been executed on the same node which it is assigned. Conversely, a more reasonable policy is to break down the resource allocated to the granularity of the container, because a container dedicates to create separate units for every single application, service or backing resource. In this way, the workload is dynamically distributed among the nodes, in a more balanced manner, when it receives a request for a container deployment and execution.

### *A. Fuzzy Inference System (FIS)*

Fuzzy Logic resembles the human decision-making methodology and deals with vague and imprecise information. [1] Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

A control system is an arrangement of physical components designed to alter another physical system so that this system exhibits certain desired characteristics. Following are some reasons of using Fuzzy Logic in Control Systems

While applying traditional control, one needs to know about the model and the objective function formulated in precise terms. This makes it very difficult to apply in many cases.

By applying fuzzy logic for control, we can utilize the human expertise and experience for designing a controller.

## V. CONCLUSION

In this paper we presented a fuzzy inference system OR fuzzy logic for runtime resource allocation in Docker Containers environment. There are no. of clusters containing nodes & the Containers are deployed to the nodes in the clusters. The Docker Container images can be Dynamically Monitor

## ACKNOWLEDGEMENT

The authors are thankful for the guidance and support from Prof. A. J. KADAM.

## REFERENCES

[1] Xiaodong Wang, Xiaowei Xu, Ye tao, Yinong Chen: Dynamic Resource Allocation Algorithm for Container-based Service Computing: (2017) IEEE 13th International Symposium
[2] X. Luo, Y. Lv, R. Li, and Y. Chen: Web Service QoS Prediction Based on Adaptive Dynamic Programming Using Fuzzy Neural Networks for Cloud Services: IEEE, vol. 3, pp. 2260-2269, (2015)
[3] C. Kleineweber, A. Reinefeld, and T. Schütt: QoS-aware storage virtualization for cloud file systems: 2014, pp. 19-26
[4] Model-Driven Management of Docker Containers IEEE (2016)
[5] Ruxandra Tapu, Bogdan Mocanu, Andrei Bursuc: Containers and Cloud: From LXC to Docker to Kubernetes IEEE (2013)
[6] Soon K. Bang, Sam Chung, Young Choh, Marc Dupuis: A Grounded Theory Analysis of Modern Web Applications-Knowledge, Skills, and Abilities for DevOps:
[7] Charles Anderson: DOCKER PAPER IEEE (2015)
[8] Sean McDaniel, Stephen Herbein, and Michela Taufer: A Two-Tiered Approach to I/O Quality of Service in Docker Containers IEEE (2015)
[9] https://www.tomsplanner.com/?template=new #doc=tUvNobhiSWCCouqaXppM
[10] https://hub.docker.com/explore/--Docker Hub.Dev-test pipeline automation, 100,000+ free apps, public and private registries.
[11] Linode Cloud Hosting Service: Founded: 2003
[12] Babak Bashari, Harrison Bhatti, Mohammad Ahmadi: An Introduction to Docker and Analysis of its Performance IJCSNS [2017]