National College of Ireland

Enhance Microservices Placement by Using Workload Profiling Across Multiple Container Clusters

MSc Research Project
Cloud Computing

Shamir Ahamed A M
Student ID: 21154929

School of Computing
National College of Ireland

Supervisor: Dr. Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shamir Ahamed A M
Student ID:	21154929
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Sean Heeney
Submission Due Date:	15/12/2022
Project Title:	Enhance Microservices Placement by Using Workload Profiling Across Multiple Container Clusters
Word Count:	7600
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shamir Ahamed
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhance Microservices Placement by Using Workload Profiling Across Multiple Container Clusters

Shamir Ahamed A M
21154929

Abstract

Companies use microservices to break up large, centralized applications into smaller, more manageable pieces that can be deployed and run in their own containers. Microservices are used by businesses to hasten product creation and upkeep, improve performance forecasting, and increase scalability. Using a task-based method, We may move only the additional workloads away from a stressed service and into a less taxed one, effectively balance the network. when there is a rise in demand for a service. Multiple versions of the system are frequently constructed to accommodate the high volume of responses. sharing network traffic amongst multiple far-flung computers. In this article, We looked at how to balance the workload of containerized microservices using a variety of flexible techniques. As a result of using cloud-based computing capabilities, microservices can make use of a distributed deployment model for their resources.
After reviewing the literature on existing methods and algorithms for load balancing, the findings indicate that further exploration concludes that further study is warranted. In this study, I have validated the PSO (Particle-Swarm-Optimization), SJF (Shorted Job First), FCFS (First Come, First Served), and RR (Round Robin) methods for microservice load balancing evaluation. A statistical analysis shows that the proposed technique is useful for reducing execution latency by picking the best load-balancing algorithm.

Keywords: MicroServices,Workloads, Task Scheduler,ContainerCloudSim,Cloudlet, VMs,Containers, Nodes.

Contents

1 Introduction

In recent times, microservices are becoming increasingly popular in software applications. Containers replication greatly improves the reuse of developed microservice elements. Functions in a software application are separated into independent services in a microservices.Wan et al. (2018).

Microservices has simplified the testing and deployment of complicated systems by breaking down big monolithic app stack into the more controllable, independent logical groups or elements that can run in cloud environments.

Microservices needed a system with lightweight deployment capabilities due to its loose coupling and flexibility to be launched separately.

Container technologies has quickly surpassed other options as the preferred platform

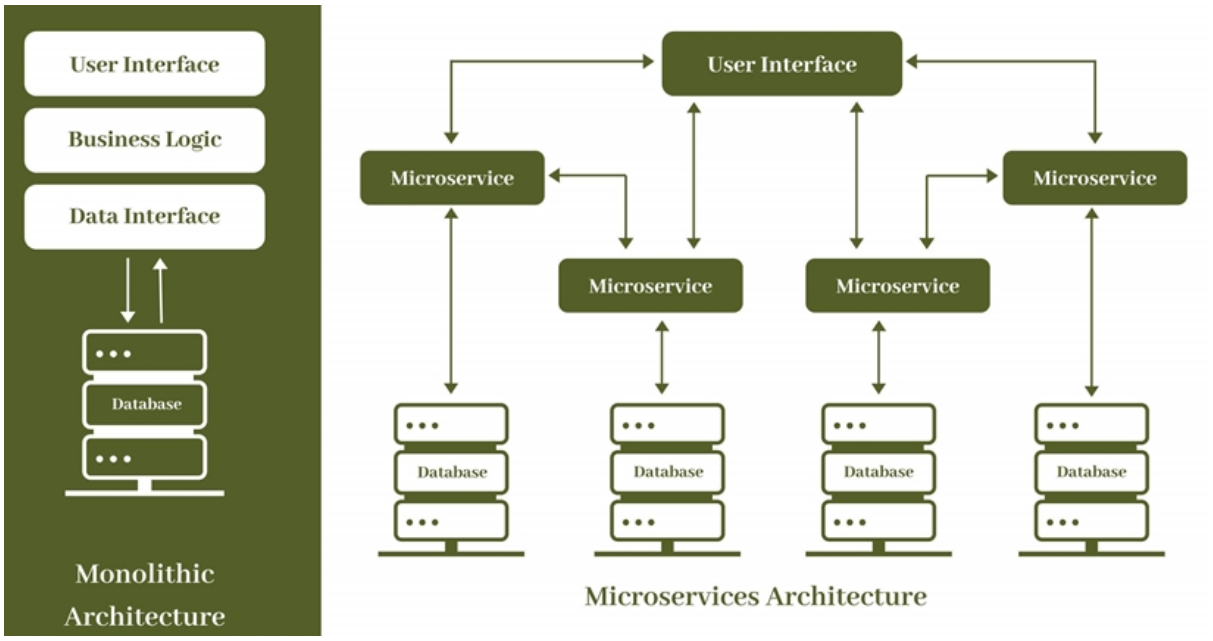

Figure 1: Microservices vs Monolithic Architecture *Microservices vs. Monolithic Architecture* (2021)

for deploying software to multiple servers. The container encloses the microservice or program logically, protecting it from the outside world.

Cloud service users and suppliers alike can reap the benefits of virtualization and dynamic scheduling of tasks. Reduced resource consumption (improved utilisation ratio), expedited job completion, and higher resource utilization are all the results of well-scheduled tasks (minimizes the makespan). Task scheduling has taken on greater significance as a result of the potential shortage of the cloud's resources brought on by the persistent growth in workloads at datacenters. More study is needed in the emerging topic of cloud real - time tasks to better fit incoming activities with available resources and enhance Quality of Service (QoS) criteria.Houssein et al. (2021) The performance of a cloud computing system might be significantly impacted by improperly scheduled jobs. In the cloud, tasks are scheduled in a way that is different from the normal.Ibrahim et al. (2021) This occurs because work time, processing power, and processing costs are just a few of the many variables considered throughout the scheduling.t Kaur and Verma (2012). Scheduling tasks entails assigning priorities to open jobs and distributing them among

accessible resources in accordance with those priorities.Houssein et al. (2021)Figure 1
This allows for simple deployment of containerized services and apps to any number of destinations, including on-premises servers, public clouds, and even personal devices.Liu et al. (2020)Söylemez et al. (2022)
The following are this paper's primary contributors:

- Using the principles of diversity and bandwidth allocation, a plan for deploying microservices is outlined. Through distributed deployment, it guarantees that no service is overburdened, improves speed, and makes better use of available bandwidth by reducing the dependency of microservices on a single node.

- The trials' end purpose is to evaluate how well the chosen algorithms performs. The suggested method demonstrably outperforms the competition.

Numerous researches have examined the optimal resource management for cloud technology. Included in these creations is the realization that cloud-native applications' smooth functioning depends on their having quick and simple access to resources provided by microservices. It may be difficult to provide new microservices enough capacity to thrive. Han et al. (2020). Analysis of a Microservice, This profiling will reveal the workload- and granularity-dependent resources usage of various applications. Devices designed for 's responsiveness often keep an eye on servers, both real and virtual. Furthermore, as cloud-native apps become more reliant on containers, understanding the data stored in containers that make up the number of benefits is more important then ever before. Since application programs are made up of many interconnected microservices, their monitoring platform must be able to track beyond just the underlying hardware.

1.1 Research-Question

What are the most appropriate load balancer strategies and algorithms to consider in order to improve response time and boost call speed across different microservices in a multiple microservices architecture?

2 Literature Review

2.1 Scheduling

Scheduling automates decision making and forges stronger linkages, and it provides backend support for a wide variety of uses, including but not limited to games, NLP, data science, etc. This complicates the ability of CSPs to use a variety of QoS models, including those for public cloud, services, virtual machine templates, implementation strategies, excitation system, etc. Unknowns, insufficient operational excellence, valuable opportunities, latency, delay, implementation capacity, cache delay, setup and regulatory oversight, virtual machines, platform support, fault types, violence, and there own limitations are all topics that have been studied in latest business analytics research. Multiple studies have shown that an on-premises QoS environment is preferable for industry 4.0, and scientists are learning to pinpoint particular issues via careful measurement and analysis. The standard operating procedures for using machine learning in business, healthcare, presentation, and certifications; sleep identification; signature detection; metal sensing; etc. will be the primary topic.

Job planning and process automation in the cloud might seem like an image of allocating values with one or more tasks, based on the authors and the nature of the work. Changes in the virtual machine (VM) status supplied by service providers are reflected in real-time management tools and comprehensive assistance from cloud suppliers. This virtual machine (VM) has to be up to the task of completing the project. The complexity of the process is established by the form that input and output data take. The next cloud scheduling Li et al. (2022) will be affected by whether or not the scheduling model was static or dynamic.

The primary purpose is to identify the most cost-effective Virtual to lessen the possibility of using the cloud-based weather board application and maximize the full use of all available resources. Honey bee provincial expansion (BCO), insect settlement enhancement (Insect), bat streamlining, and molecule multiplicity enhancement are only few of the many knowledge tactics utilized to choose the top Virtual (PSO). However, most of these methods have little mathematical use. Honey bee settlement enhancement (BCO) is a populace-based calculation roused by the way of behaving of bumble bees, however, it has two fundamental inconveniences: (1) its sluggish combination and (2) many control boundaries. Kruekaew and Kimpan (2022) ANT is used to figure out the best approach to a problem based on how a person would act if they were looking for honey bees. It is not necessary to do the same investigation on every single person. The technique of critical thinking known as molecule pillar improvement, which use objects to reproduce a population and channels to hunt down a solution, is used to determine the optimal course of action. However, it cannot handle the exchange problem and cannot use identical integers concurrently. Even yet, the two players are now using equal molecule channel development (PPSO) to select Virtual Ms for practical settings, which is used by both and overcomes the shortcomings of the current system. In order to save processing time, PPSO divides the population into smaller subpopulations and performs its own computation on each subpopulation separately. That's why it's crucial to respond to requests from all of your partners in a timely manner.

2.2 Workflow Scheduling in Cloud

Geological Mapping, Astrophysics, Quantum theory, and Bio-informatics are just a few examples of fields where workflow modeling has been widely adopted to simulate large-scale scientific and technical applications. Directed acyclic graphs (DAGs) consisting of nodes and edges are used to depict a Workow. Large-scale scientific applications are represented by a network in which nodes stand for computing operations and edges for data/control relationships. Depending on the specifics of the scientific use, the workow might be somewhat compact or rather large. Researchers needed a highperformance compute cluster, such as computer clusters, grid computing, or the most recent iteration, cloud computing, to conduct various workows of varied sizes.

The effectiveness of scientific process tasks is tested in a simulation environment. Additionally, the effect of several system and workload characteristics on application performance is studied. This review of performance shows how successful it is. As cloud computing grows in popularity, more and more applications, especially those with complex execution flows and resource requirements, are being deployed to the cloud (referred to as multi-stage jobs). Many businesses and academic organizations rely on MapReduce to provide Big Data analyses, usually in tandem with cloud or cluster computing. In order to meet service level agreements (SLAs) in a distributed computing system with a finite

amount of resources, this study focuses on developing an effective matching and scheduling approach for processing a continuous flow of multi-stage activities (workflows).Khan et al. (2021)

2.3 Survey of Microservices

When implementing containerized apps as microservices, the intricacy of distributed infrastructure can make it difficult to detect and resolve outages for mission-critical use cases, such as industry 4.0 processes. Observability is to help operators manage and control large - scale distributed infrastructure and microarchitectures by the measurement of end-to-end simulation of the effect.

Building, delivering, and maintaining user applications is being transformed by the combination of Development and Operations (DevOps), a design philosophy and developing technologies. In contrast to a monolithic program, which builds all of its parts into a single, cohesive structure, a microservices architecture separates the application into a set of independent services that collaborate to carry out a larger whole. Lightweight RESTful App Programming Interfaces (APIs) are extensively used to facilitate communication across microservices in the context of actual application execution. Keeping track of things, whether successfully or not, is not a new phenomenon. Existing monitoring systems have considerable challenges when it comes to meeting the operation monitoring needs of container orchestration deployed across distributed IT infrastructures. Analyzing metrics from individual bare-metal servers, VM instances, or application containers won't give a diverse operations (Ops) staff a complete picture of performance issues unless they also understand the topological connections between these components. A decentralized infrastructure is one in which many modules run independently on various machines spread out over a network and communicate with one another to accomplish their goals. The need to provide a high level of service quality (QoS) for end users in order to deliver a positive Quality of Experience (QoE) is a major driving force behind the current state of distributed infrastructure development. The overarching trend of computer advancements over time. Usman et al. (2022)

2.4 Microservice vs Monolithic Service Architecture

Using a monolithic design, developers may build applications with dozens or even hundreds of independently functioning services that share a common codebase. In a shared office space, this might cause a lot of problems for the team. Thus, many businesses are adopting a microservices design to facilitate communication and collaboration across their design teams. Compared to the microservices design, the monolithic architecture achieved a 6 percentage point greater bandwidth in the concurrent tests. There was little to no noticeable distinction between the two architectures throughout the load testing scenario. Additionally, a third test evaluating the performance of microservices systems built using various service discovery technologies like Consul and Eureka revealed that the apps using Consul achieved higher results. In contrast to microservices design, monolithic applications have a single source of code that incorporates all of the necessary services. They use protocols like Web services, HTML pages, and REST API to talk to other systems and end users. It is believed that the use of microservices architecture would improve the procedures of supportability, re - usability, adaptability, reliability, and automatic deployment. comparisons were made between the efficiency of microservices

running in a containerized setting and a virtual machine (VM) setting. Throughput, reaction time, and CPU use were measured and compared across different Amazon cloud settings. This article shows that virtual machine (VM)-based settings on Aws cloud services beat container-based settings by a factor of 125 percent. This is notably true when comparing response times.

2.5 Microservice-Workload Profiling

In this paper Han et al. (2020) A change to Empirical workload profiling calls for a shift in how microservices are allocated. Since stage microservices deployment can learn to account for changes in demand, newline For cloud-native apps to run well, planning process must take into account the requirements of realizing its individual microservices. There are a number of obstacles that must be surmounted before new services may receive adequate funding. Most scholars rely on mathematically noticed on simulations than than attempting to answer these problems directly. Many often, the research fails to account for the resources required to run a microservice. When calculating resource requirements for various workloads,the solutions should factor in the complexities of microservices. This research provides an effective way for evaluating application relative performance to the placement of microservices through the use of empirical profiling. This section provides a brief overview.

- Workload profile-based microservice deployment framework. System enables work load analysis and distributed software development. Build resource fluctuations monitor to gather microservices resource use data. That data examines cloud-native programs that adjust to loads regularly. Profiled data guides microservice implementation.

- Profiling microservice deployment that use the refining architecture across container clusters. Practical profiling of distributed software deployment reveals a greedy-based workload-responsive heuristic. Positioning testing used three container clusters. To check the idea, deploy microservices using the new structure deployment rules on the testing ground. Trials show thatthe solutions increase service quality.

3 Methodology

A microservice is a small, self-contained service that may interact with other microservices over an application programming interface (API). Services like this are managed by self-contained, small-sized teams. Applications may be developed and expanded with less effort thanks to microservices architectures, which promotes creativity and speeds up the deployment of new features. The term "load balancing" refers to the process of dividing network traffic that arrives amongst several servers in a data centre or servers pooling in real time or in a number of servers that may be used at any one moment. This technique for dividing up the resources can be carried out on a fair scale or in accordance with established Analysis tools. To prevent any one server from being overworked to the point that its performance suffers, a load balancer acts as just a "traffic cop" ahead of the server, distributing requests evenly among all available machines. If one server goes down, the scheduler will forward all of the requests to the other servers. The scheduler

will immediately begin routing requests to the newly added server once it has been registered to the service group. The functional group will store data to be monitored using distributed software containers. The workload parameters will be analyzed by the activity scheduler, that will allocate virtual machines and containers. The effectiveness of different task schedulers with varied distributed software workloads will be analyzed and demonstrated for future study.
Improvements to microservices with multiple schedulers will simulate task schedulers with a variety of microservices workloads in order to study features with multilayer workloads across different datacenters, VMs, and containers. Container CPU-memory, storage, and network metrics will be tracked and shown for analysis.

3.1 ContainerCloudSim

Simulator creates a simulated setting in which research investigations may be tested and validated, leading to better, more real applications solutions. It's a scholarly technique for making a simulation or a working program.Singh et al. (2021). As a result, it removes the requirement for and costs associated with computational facilities for performance evaluation and modeling the research answer. This tutorial primarily focuses on the CloudSim simulation tool and its usefulness for research.
CloudSim is divided into three levels. The uppermost layer, known as the "User Code"

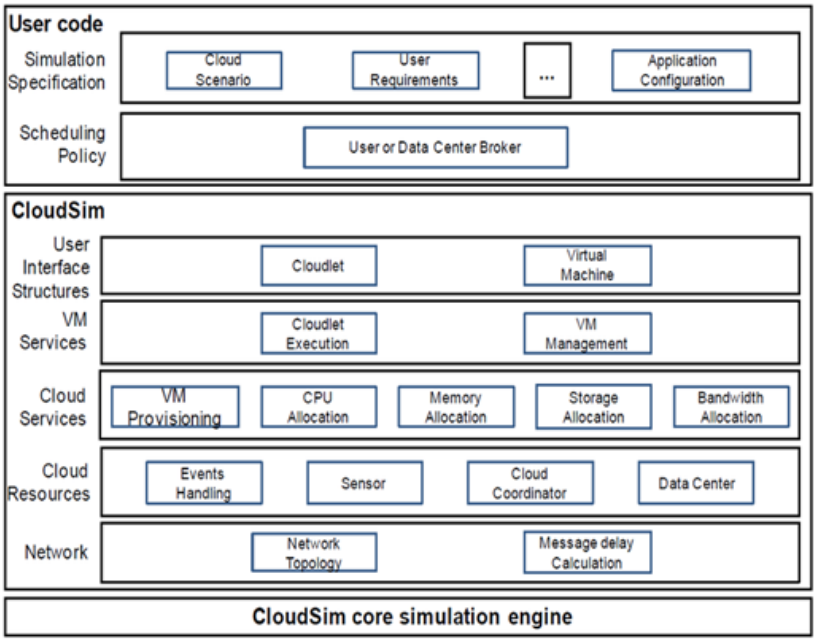

Figure 2: CloudSim Architecture Tariq and Santarcangelo (2015)

layer, is made up of the cloud's fundamental elements. This is where the simulation's requirements, such as the number of VMs, users, and the upgraded or suggested scheduling scheme. The cloudlet in this tool is known as the task, which is the user request. As a result, this layer supplies the interface as well as additional simulation experiment characteristics such as data center location, etc. The second layer, "CloudSim," also aids in the creation of a cloud-based environment by implementing a user interface that incorporates the Cloudlet and Virtual Machines.This layer allows customers to adjust the

cloud services' bandwidth, memory, and CPU. The layer can also be used to imitate the network in cloud settings. The last layer is the "Simulation Engine," which is in charge of conducting events such as cloud component formation and communication Alotaibi et al. (2021).

To better simulate cloud computing environments and services, ContainerCloudSim has been built upon the existing CloudSim modeling framework. ContainerCloudSim Piraghaj et al. (2017) models cloud environments that make use of containers. Schedule and provisioning tests for containerized applications. The majority of existing simulators, with the exception of ContainerCloudSim, are geared at providing a platform built around virtual machines. Figure 2.

- Data centers may manage containers and virtualized workloads.
- Dynamic fog nodes for modelling and applications are distributed in real time during simulations using Element.
- Information technology facilitates the modeling of applications with interdependent duties

3.1.1 Implementation

ContainerCloudSim Meng et al. (2016) represents a simulation, or is a simulator. The entities and pieces in CloudSim communicate with one another via messaging to produce a model of the clouds and its capabilities.

After registering with CIS, Datacenter Brokers link to CIS to find Datacenters that are currently available. Cloudlets and other virtual resources are created and scheduled through communication between data centers and data center brokers. Due to the fact that two forms of virtualization cannot coexist in a single Datacenter in CloudSim, several kinds of Data centres are required. After obtaining Cloud computing Cloud resources from of the Cloud computing Broker at the beginning of the simulation period, the Infrastructure Broker immediately assigns them to virtual machines or containers. It is recommended that at the start of the simulation time, a list of all arriving fog nodes be provided into the Broker. Multiple jobs would be scheduled using time management software, and the results of these schedules would be reviewed for both data and efficiency.

3.2 PSO Task Scheduling Algorithm

Eberhart and Kennedy created and refined the standard PSO in 1995, drawing inspiration from the feeding behavior of birds. In PSO, a random solution is generated at the outset, and the local best solution is generated at the end of each iteration to find the global best solution. The solution's particles are nearly probably the optimal solution. The probability of a best solution is calculated with the help of the fitness value, and the particles has a tendency to reach that value.Dubey and Sharma (2021)

3.3 SJF Algorithm

An example of a common classical scheduling method is shortest job first (SJF). When scheduling, this method chooses the nodes with the quickest execution, making it simple to implement.Lin et al. (2022)

3.4 Round Robin Algorithm

Round robin scheduling allocates a predetermined time quantum to each process.
In this method, every process is performed cyclically, so programs that have burst time left after the period quantum expire are being sent back to a ready queue and wait until their next chance to finish execution until it finishes. FIFO processing means first-come, first-served.

3.5 First Come First Serve Algorithm

FCFS is a scheduler that instantly handles requests and tasks in the order they were put in a queue. It is the most straightforward and straightforward way to schedule things. In this kind of algorithm, the allocation goes to the process that asked for it first. A FIFO queue is used to handle it.

3.6 Proposed-Algorithm-Workflow

Under the emerging paradigms, the cloud services platform may be administered in a variety of ways that are beneficial to businesses and end users alike. There is a significant challenge in the cloud when it comes to coordinating the schedule of Microservice jobs among Host, Vms, Container, and Cloud Services. In order to address these issues, this paper proposes a method of task scheduling that is both effective and practical. I plan on comparing PSO, SJF, RR, and FCFS to examine cloud containers availability, production cycle time, order to obtain higher, and CPU use.

4 Design Specification

CloudSim Discrete Event simulators Core is being used to represent fundamental features of cloud services and offer the basic discrete event simulation functionality required for delivering the above capabilities. Message passing between instances and elements in CloudSim is a crucial

The core layer is accountable for of processes, actions, and relations between modules. Visual representation of ContainerCloudSim's primary data types. Here, we'll discuss the specifics of these courses. The core components of a ContainerCloudSim implementation are a set of simulated components and a set of modelled applications. With the help of a custom java application and some measuring in the construction process, I can emulate the desired results.

- **Datacenter-:**Datacenter represents the hardware component of cloud services. This class functions as a proxy for all servers at the physical layer.
- **Host-:**In a computer system, the Host class stands in for the hardware (servers). Their specs are broken down into three categories: memory, storage, and computing power, all of which are measured in MIPS (million instruction per second).
- **VM-:**Users may express the qualities of a virtual machine device class by its memory, processing power, and storage space. Hosts often play the role of hosting and managing virtual machines (VMs).

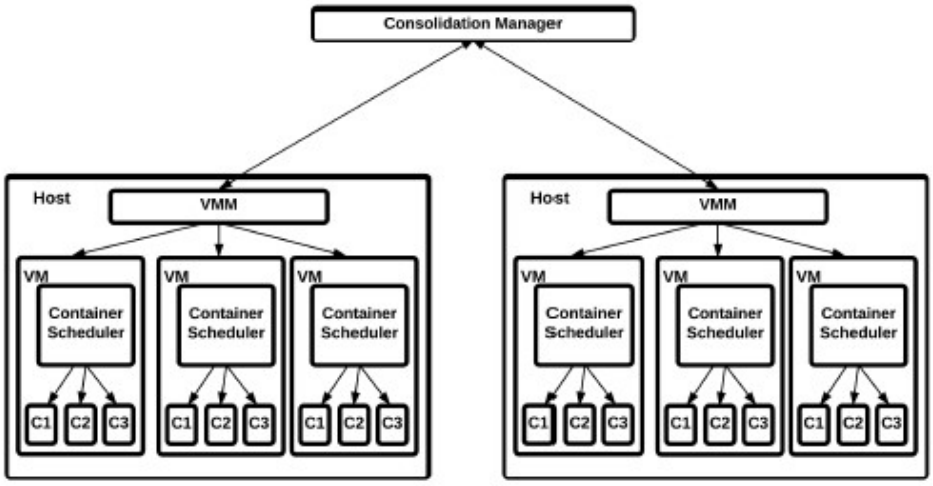

Figure 3: Illustration and Simulations of Containerized Cloud Piraghaj et al. (2017)

- **Container-:** Virtual machines (VMs) act as hosts for containers, representing their resources like memory, processing power, and disk space.
- **Cloudlet-:** Version with this object that runs in a containers or VM. The cloudlet count, the agency's starting timing, and its current state are some of the most crucial configuration options.
- **Datacenter components-:** Implementing the foundational tenets of VM/Container assignment in a facility that is able to operate both is the holy grail of modern datacenters. The introduction of Container to cloud users and virtualization software enables the Data centre to pick the most appropriate allocation method. As an added bonus, ContainerCloudSim offers the following services for simulation purposes:
- **VM Provisioning:** A property of the Host class, the VM provisioning process will determine how the host's CPU resources are divided up across running virtual machines. The Host element is, in many ways, like CloudSim. Classes that manages CPU cores and provides a model for simulation via interfaces that it implements. alternatively, cores may be shared with other VMs on the host.
- **Workload Management:**The simulation supports containers installation at both the virtual machine and container levels. Each package's systematically analyze are set at the virtual machine (VM) level. Each application services can be allocated a certain number of resources per the structure's specifications.
- **CloudletScheduler:**As with virtual machines, programs (here called Cloudlets) may be placed within containers. Extending the CloudletScheduler abstract methods allows for the implementation of alternative techniques for measuring the distribution of resources across Cloudlets inside a container. The ContainerCloudSim bundle has time-shared (ContainerCloudletSchedulerTime-Shared) deployment rules.

- **ContainerAllocationPolicy:**When assigning containers to virtual machines, this abstract class stands in for the placement policy that will be used. To allocate containers, ContainerAllocationPolicy determines which available virtual machine (VM) in a given data center satisfies the container's distribution criteria, such as ram, store, and availability.

- **VmAllocationPolicy:**The abstract data type not only sets the consolidation rules at the containers and VM layers, it also implements the optimizeAllocation function that is responsible for assigning virtual machines to hosts.

- **Workload Management:**Cloud applications are characterized by their highly changeable workloads. This is because ContainerCloudSim allows for the simulation of cloud apps' changing required to fill inside a CaaS setting. To learn about container-level resource needs, I used CloudSim's preexisting Consumption Model. As an abstract class, the Utilization Model allows simulated users to acquire unique workload characteristics by overriding the getUtilization() function. As input to the getUtilization() function is the duration of the simulated, and as output is the proportion of each Cloudlet's compute time that was actually used throughout the experiment.

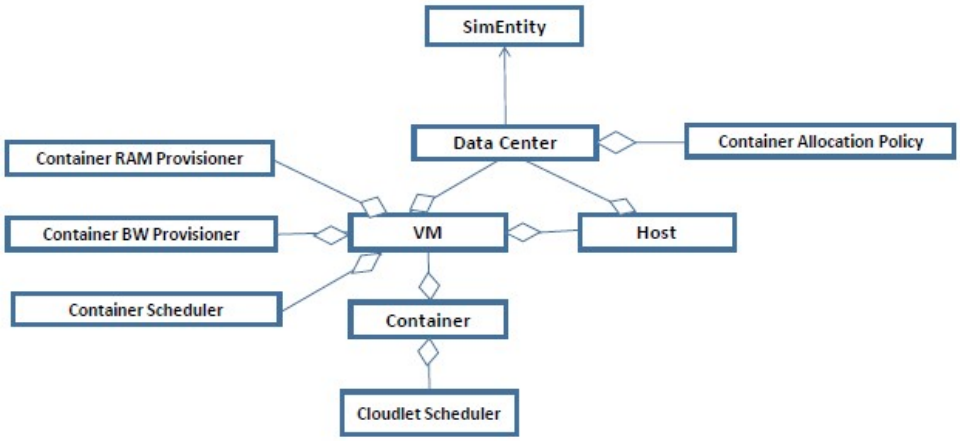

Figure 4: ContainerCloudSim classes Piraghaj et al. (2017)

4.1 Lifecycle of Simulation Modeling

Within the container carrying out the tasks, the simulating execution of job pieces is handled. In this regard, the status of the currently running job is updated at each iteration of the simulation. The Server class's updateVMsProcessing() function is invoked at every simulated sampling interval. The update VMsProcessing() function takes a simulation time value as its devices generate. Each host is then prompted to modify the execution of all of its virtual machines through a function call (updateContainersProcessing()). Each virtual machine (VM) must repeat the procedure in order to refresh the process of its containers, so each containers must repeat the process in order to update the process of its applications. The container-level technique provides the quickest expected completion time for all tasks executing on the container. The containers with the quickest turnaround

times are the ones whose results are sent back to the host VM. At the end of the day, only the VMs with the quickest total finishing time are sent back to the Datacenter from the network level. If the Datacenter class receives a time value earlier than the current time, the current time will be reset and the procedure will begin again at that moment. The next model phase then is determined based on the predicted time, as well as an activity is planned in the simulated core at that moment.

.

5 Implementation

The primary objective of the new design is to solve problems with the traditional method of scheduling tasks by using a novel approach to scheduling tasks. The system's infrastructure was acquired from Amazon's Web Services (AWS dedicated-hosts) and (AWS instance-types) and has been taken into consideration for one Datacenteres each datacenter having two host computers. Each host machine has the potential to allocate up to 20 VMs, depending on the number of cloudlets, and each Virtual Machine will have two cores, allowing for the creation of two containers simultaneously. When doing the study, I will take into account VM CPU usage,container assignment, time consumption, and memory utilization before coming to a conclusion on the performance of the Scheduling algorithms. The assignment of VMs and containers is determined by PSO,FCFS,SJF and RR algorithms.

Table 1: Configuration of the server, VMs, and containers

Server Configurations			
Server type	CPU	Memory (GB)	Population
m4	24	192	2
VM Configuration			
t2.medium	2	4	20
Container Configuration			
-	1	1	40

The Virtual Machine Manager installed on top of the physical servers relays information like the host status and the needed container list to the consolidation manager, demonstrating the potential of ContainerCloudSim for analyzing TaskScheduling. All the use cases I looked at have the same basic architecture. When a migration is scheduled, VMM notifies the consolidation management of the host's current condition and provides a list of container that will be moved. The consolidation management selects where containers will go and then asks that resources be made available there.

Depending on how the containers are mapped to the VMs, the resources are used in a variety of ways. In order to learn more about container to virtual machine mapping methods like PSO,FCFS,SJF and RR I may use ContainerCloudSim. Here, I provide a case study illustrating how ContainerCloudSim may be put to work to probe the impact of container placement methods. Three distinct placement strategies are analyzed.

5.1 Proposed Task Schedulers Architecture

In the first step, the given workload is analyzed and used to inform a characterisation model used in the process of classifying the application. Based on their respective requirements for computing power, memory, and network connectivity, applications are separated into three distinct categories. The second step of scheduling algorithms involves checking against the previously described situations to determine whether a new virtual machine (VM) should be added or an existing one used. The suggested paradigm guarantees that requests using the same resource group all belong to the same pod. Likewise, by grouping requests of the same kind together, the likelihood of duplicate requests for the same set of resources is reduced. As a result, increased resource utilisation is the implementation and demonstration of the theory, allowing for a decrease in resource contention and an improvement in performance metrics by avoiding over- and under-utilization.

5.2 Stacking of Containers

As part ofthe thesis, I investigate using the Cloudsim Simulation platform to emulate the cloud environment and undertake the necessary analysis of the container choosing policy optimization technique. Containers have been shown to be a more light virtualization technique than virtualization hypervisor layers. Containers isolate workloads while generating the overheads that hypervisors do. The virtual machines that host the containers provide an additional degree of protection and separation for unknown incoming workloads. Simulation models assist us in comparing novel resource management techniques. The simulation model allows us to replicate large - scale distributed network in order to test resources plans and policies without had to reach cloud services, which is a costly and time-consuming job. The simulator also allows for the verification and testing of different resource procedures.

5.3 Container Provisioning

In the simulation model, containers are provisioned in the following ways: at virtual machines. Whenever container are added to virtual machines (VMs), the amount of processing power allocated to each container must be specified; however, at the container level, a variable amounts of resources may be allotted to each application operating inside the container. There are also abstract classes for managing memory and bandwidth consumption for containers, allowing them to be allocated dynamically to processes.

5.4 Container Consolidation

The study's goal is to validate the suggested algorithm's efficiency in a cloud data center in terms of increasing processing time and energy usage via the use of CloudSim simulation. In container cloudsim, I use the Container as a Service architecture to replicate a data center with a few hundred physical nodes, each hosting a variety of containers and one of four distinct kinds of virtual machines. Below is a table detailing the various host types and their distinguishing traits.

5.5 Simulation Scalability

ContainerCloudSim is a lightweight and scalable cloud simulation tool that uses containers. I shall run the simulation using workloads for PSO,FCFS,SJF and RR and set up the experiment with a container initial placement strategy to examine the sustainability of the built simulator. The same test is run with workloads varying their normal size. Both the number of hosts and the number of virtual machines (VMs) are assumed to be the same for the sake of this comparison.

To demonstrate the efficacy of the suggested work in a real-world setting, an application designed for that purpose must be installed in an easy manner on a simulation platform. Cloudsim simulations of the proposed scheduler in action provide for a more thorough understanding of its potential. Real implementation requires carrying out the procedures shown, while the simulation of this task may benefit from the information provided.

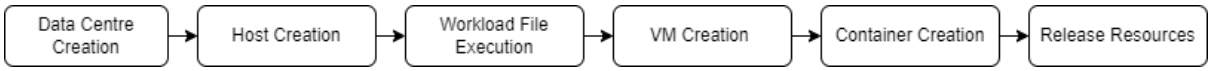

Figure 5: Simulation

5.6 Code Implementation

This part digs further into the implementation structure of the CloudContainer CloudSim model, where it keeps a separate category for each element, in order to include the Container Scheduler.

5.7 Development Structure

The Container Task Scheduler model was simulated in Cloudsim. In order to simulate the scheduling of virtual machines and containers, I will utilize the PSO,FCFS,SJF and RR algorithms.

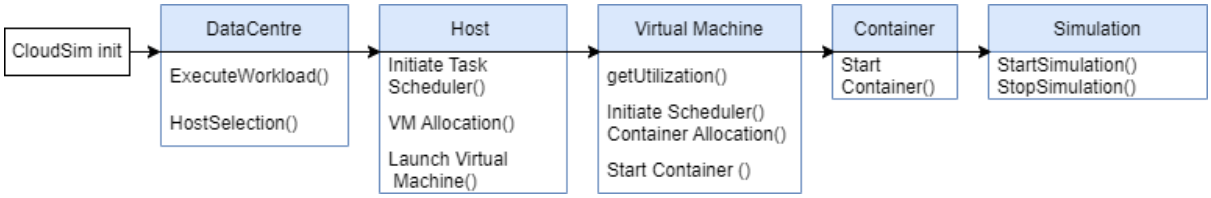

Figure 6: Development Structure

- Cloud-Sim.init - When creating any structures in CloudSim, it is necessary to initialize the CloudSim Packages.
- Container-Allocation-Policy - Determining how containers will be allocated. The allocation of Containers to Virtual Machines in the data center is controlled by this policy.

- Host-Selection-Policy - The Virtual Machine (VM) Policy Formulation. Whenever a hosts is found to be overrun, this policy defines whether virtual machines should be used.
- host-List - When compiling the host lists, I take into account the total number of nodes.
- cloudlet-List - In this implementation, a simulated work load is utilized. These load files are generated using a mathematical model. CPU, memory, storage, and network bandwidth are only some of the hardware variables evaluated.
- vmList - Compilation of the defined virtual machines
- Container-VmAllocation-Policy - In this part, I present the above-described preset algorithms as policies for allocating VMs to containers.
- cloudlet-List,container-List,vm-List - Lists of cloudlets, containers, and virtual machines (VMs) are created and sent to the Datacenter broker.
- Container-Datacenter-Broker - This class will get all of the knowledge that was been reviewed.
- submit-Cloudlet-List,submitContainerList,submitVmList - Sending a list of cloudlets, containers, and virtual machines to the datacenter provider
- CloudSim-start-Simulation,CloudSim-stop-Simulation - By providing the aforementioned information, the simulation may begin, run its course, and display the outcomes.

6 Evaluation

With ContainerCloudSim, this test case hopes to understand how a certain container selection method affects the productivity of the container consolidation procedure. The data, including the host's state and a list of containers, is sent to the consolidation manager through VMM. The consolidation manager makes the decision on where containers go, and then asks that resources be made available there. manager, which runs on a different system, makes the decision on where to relocate containers and then asks resources from the target host.
In order to get a better grasp on the data processing and time consumption, I have adopted the container model and scheduling model, assigning the vm schedular to the container model. This allows us to conduct in-depth analyses ofthe progress and provide timely updates to other modules, including the data center, the vms, and their dependencies.
As tasks are being carried out, several scheduling techniques are applied at various points. The virtual machine assignment phase makes use of Scheduling algorithms: PSO,FCFS,SJF and RR.
With the configuration of 2 datacenters, each datacenter having 2 host systems, and each host system having 10 virtual machines (VMs), and required containers being assigned as per the requirement, this setup has been done on Container cloudsim for the purposes of validating resource utilization, maintaining resource load balance, and increasing system

performance by reducing the amount of processing that needs to be done.
The container cloudsim software consolidation manager will check through the workload files and divide them once it has processed anywhere from five to ten different types of workload-related files. The workload file will be assigned to a particular virtual machine (VM), which will depend on availability and the configuration. The decision will be made by the scheduling algorithm, which is scheduled and defined in the simulation. With a variety of factors, the performance factors will assist us in determining which algorithm provides the best possible results. In the beginning, I will visualize the assignment of containers with respect to VM. This will help us to understand the scheduling progress, such as how frequently the VM is used and how many containers are assigned in parallel with the processing time of individual workloads using various scheduling algorithms, I will discover which algorithm is performing most effectively.

6.1 VM and Container Scheduling

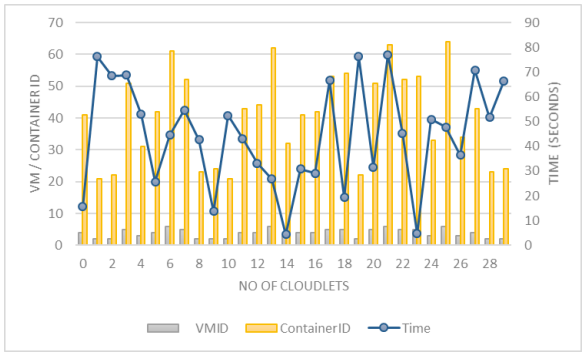

Figure 7: FCFS Schedler

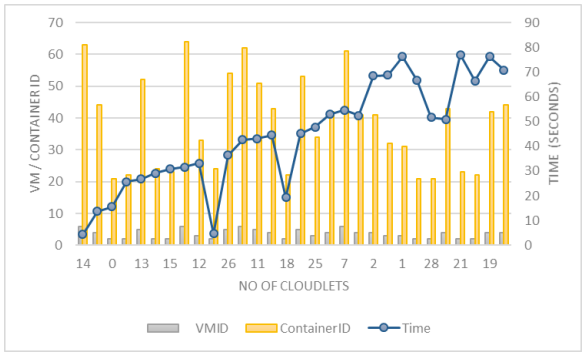

Figure 8: Round Robin Scheduler

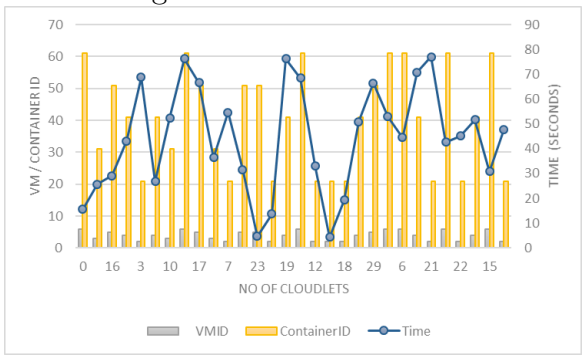

Figure 9: SJF Scheduler

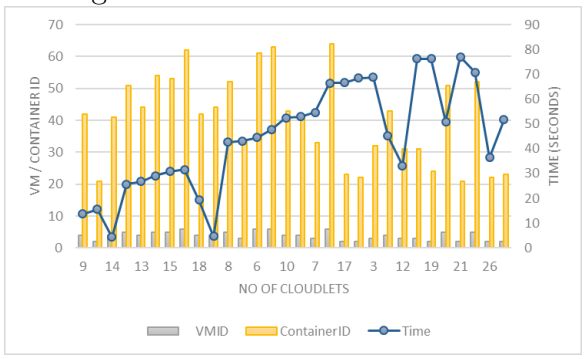

Figure 10: PSO Scheduler

Based on visual analysis. Once the procedure is complete, the next available virtual machine (VM) and container (container) will be assigned by FCFS. Round Once the time limit has been reached, Robin will cease processing that task on the VM container and initiate a new one; the old one will be returned to the queue. This is determined by Robin's analysis of the cloudlet. To begin a new task, SJF will first locate a shorted-availability container with available resources, then wait for that container to complete its execution. Once the PSO has finished analyzing all of the cloudlets and their workload and the availability of containers, it will begin processing the workload in the order calculated previously and using the predefined VM container.

6.2 Processing Time with Various alogrithm

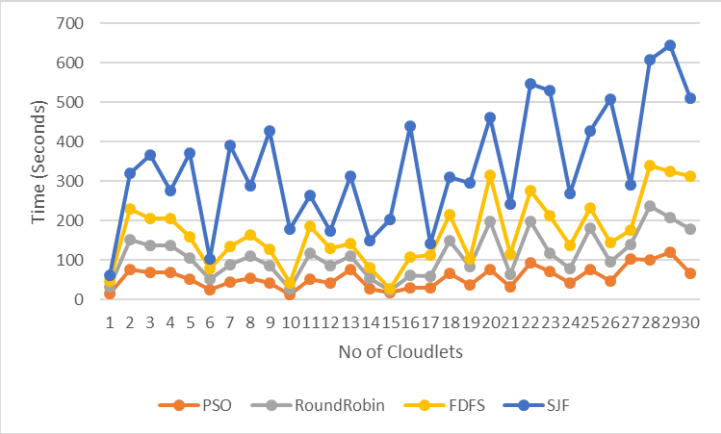

Figure 11: Processing Time Vs FCFS, RR, SJF, PSO Scheduler

6.2.1 Makespan

The time taken from the beginning of a project to its completion is called its makespan. The goal of a multi-mode resources restricted task scheduling challenge is to develop the short logic scheduling feasible by making the most effective use of available resources and allocating the fewest possible extra resources in order to meet the required minimum makespan. I analyzed the Processing Time of FCFS, RR, SJF, and PSO Scheduler using

Table 2: Processing Time and Makespan for FCFS, RR, SJF, PSO Scheduler

	FDFS	Round Robin	SJF	PSO
Processing Time (Seconds)	1723.64	1689.33	5035.51	1674.77
Make Span	3794	4247	3974	2900

the same workload on a cloud simulator with the same sort of data center having the same host machine and virtual machines with containers.The visualization of processing time shows PSO on top, then RR, then FDFS, then SJF. The SJF scheduler is slower than alternatives.

6.3 Processing time and Makespan evaluation with Schedulers

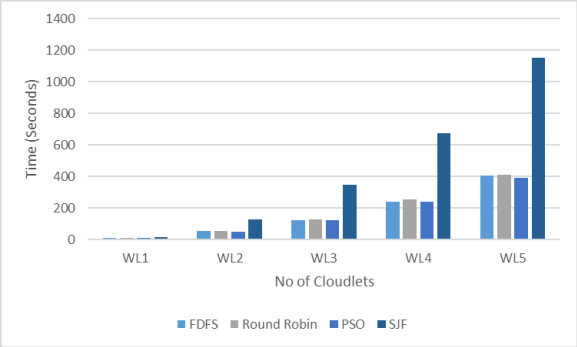

Figure 12: Processing time

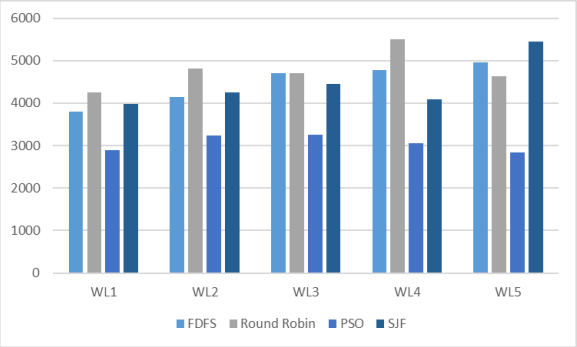

Figure 13: Makespan

On a cloud simulator with the same kind of data center, using the same host machine and virtual machines with containers, I compared the Processing Time and make span of FCFS, RR, SJF, and PSO Scheduler with multiple workloads. There is no fluctuation in processing time while working with little cloudlets, but this changes dramatically as the demand does. I have modeled a wide range of workloads, each with its own unique processing time. After graphing the data for each of the aforementioned task schedulers, I came to the conclusion that PSO Scheduler performed the best overall.

Table 3: Processing Time and Makespan for FCFS, RR, SJF,PSO Scheduler

Processing Time (Seconds)				
Workload	FDFS	Round Robin	PSO	SJF
WL1	11.86	9.03	8.8	16.12
WL2	53.61	52.6	48	128.63
WL3	125.24	128	123	347
WL4	242	255	238	673
WL5	405	413	393	1151
Makespan				
WL1	3794	4247	2900	3974
WL2	4142	4814	3240	4245
WL3	4711	4699	3263	4457
WL4	4784	5501	3053	4087
WL5	4960	4632	2839	5451

7 Conclusion

As the number of virtual machines (VMs) and containers (Containers) using a given host CPU increases, I have found that their performance slightly degrades. As the workload file sizes grow above a certain limit, though, the process time shifts. To evaluate the efficacy and effectiveness of container consolidation with the chosen Scheduler, I compare identical workloads with a similar test executed on virtual machines hosted within containers. When compared to the time required by the newly developed approach to execute an operation is much different. After being subjected to several types of workloads, the

average processing times have been studied. Processing time analysis places PSO at the top, followed by RR, FDFS, and finally SJF. The goal is not to demonstrate the superior scheduler, but rather to demonstrate how we may simulate with schedulers using old log files and understand the conclusion for use in future implementation. .

7.1 Future Directions

In additional, the work that will be carried out in the future in relation to this may include the implementation of these algorithms on the real code base and the observation of the outcomes of the application. The fact that the results of the simulation have been verified with the output values has provided us with some useful insights into the possible behavior of this algorithms in real life. Because the workload will be almost

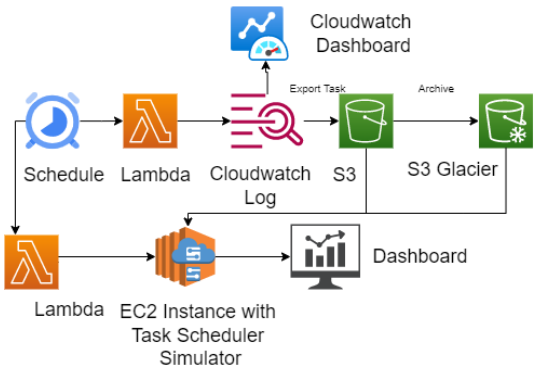

Figure 14: Time Vs FCFS, RR, SJF, PSO Scheduler

identical after the implementation after a long long run, the workload log files will be gathered and stored at a central destination. After that, I will be able to build a system that will fetch all of the workload files from real container based platform and simulate with multiple custom schedulers and visualize with the base of the simulation, I will be able to implement the schedulers in real time and can increase the performance by decreasing the processing time.

References

Alotaibi, A., AlZain, M., Masud, M. and Zaman, N. (2021). Cspm: A secure cloud computing performance management model, *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* **12**: 2114–2127.

Dubey, K. and Sharma, S. (2021). A novel multi-objective cr-pso task scheduling algorithm with deadline constraint in cloud computing, *Sustainable Computing: Informatics and Systems* **32**: 100605.
URL: *https://www.sciencedirect.com/science/article/pii/S2210537921000937*

Han, J., Hong, Y. and Kim, J. (2020). Refining microservices placement employing workload profiling over multiple kubernetes clusters, *IEEE Access* **8**: 192543–192556.

Houssein, E. H., Gad, A. G., Wazery, Y. M. and Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends, *Swarm and Evolutionary Computation* **62**: 100841.

Ibrahim, I. M. et al. (2021). Task scheduling algorithms in cloud computing: A review, *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* **12**(4): 1041–1053.

Kaur, S. and Verma, A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment, *International Journal of Information Technology and Computer Science (IJITCS)* **4**(10): 74.

Khan, M. G., Taheri, J., Al-dulaimy, A. and Kassler, A. (2021). Perfsim: A performance simulator for cloud native microservice chains, *IEEE Transactions on Cloud Computing* .

Kruekaew, B. and Kimpan, W. (2022). Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning, *IEEE Access* **10**: 17803–17818.

Li, J., Zhang, R. and Zheng, Y. (2022). Qos-aware and multi-objective virtual machine dynamic scheduling for big data centers in clouds, *Soft Computing* **26**(19): 10239–10252.

Lin, Z., Li, C., Tian, L. and Zhang, B. (2022). A scheduling algorithm based on reinforcement learning for heterogeneous environments, *Applied Soft Computing* **130**: 109707.
URL: *https://www.sciencedirect.com/science/article/pii/S1568494622007566*

Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H. and Li, Z. (2020). Microservices: architecture, container, and challenges, *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, pp. 629–635.

Meng, Y., Rao, R., Zhang, X. and Hong, P. (2016). Crupa: A container resource utilization prediction algorithm for auto-scaling based on time series analysis, *2016 International conference on progress in informatics and computing (PIC)*, IEEE, pp. 468–472.

Microservices vs. Monolithic Architecture (2021). https://www.hitechnectar.com/blogs/microservices-vs-monolithic/. [Online; accessed 2022-12-11].

Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N. and Buyya, R. (2017). Containercloudsim: An environment for modeling and simulation of containers in cloud data centers, *Software: Practice and Experience* **47**(4): 505–521.

Singh, A. P., Pradhan, N. R., Luhach, A. K., Agnihotri, S., Jhanjhi, N. Z., Verma, S., Kavita, Ghosh, U. and Roy, D. S. (2021). A novel patient-centric architectural framework for blockchain-enabled healthcare applications, *IEEE Transactions on Industrial Informatics* **17**(8): 5779–5789.

Söylemez, M., Tekinerdogan, B. and Kolukısa Tarhan, A. (2022). Challenges and solution directions of microservice architectures: A systematic literature review, *Applied Sciences* **12**(11): 5507.

Tariq, M. and Santarcangelo, V. (2015). Simulator's requirements for modeling and simulation of cloud environment, *Sei Sigma e Qualità* **6**.

Usman, M., Ferlin, S., Brunstrom, A. and Taheri, J. (2022). A survey on observability of distributed edge & container-based microservices, *IEEE Access* .

Wan, X., Guan, X., Wang, T., Bai, G. and Choi, B.-Y. (2018). Application deployment using microservice and docker containers: Framework and optimization, *Journal of Network and Computer Applications* **119**: 97–109.