Journal of King Saud University – Computer and Information Sciences 34 (2022) 3934–3947

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Container scheduling techniques: A Survey and assessment

Imtiaz Ahmad, Mohammad Gh. AlFailakawi *, Asayel AlMutawa, Latifa Alsalman

Department of Computer Engineering, College of Engineering & Petroelum, Kuwait University, P.O. Box 5969, safat, 13060, Kuwait

ARTICLE INFO

Article history:
Received 14 October 2020
Revised 4 February 2021
Accepted 5 March 2021
Available online 11 March 2021

Keywords:
Containers technology
Optimization techniques
Scheduling algorithms
Resources management
and Performance evaluation

ABSTRACT

Containers have emerged as the most promising lightweight virtualization technology in providing cloud services due to its flexible deployment, portability, and scalability especially in micro-services, smart vehicles, IoTs, and fog/edge computing. An important and vital role in cloud container services is played by the scheduler's component to optimize performance and reduce cost due to the diverse nature of the workload and cloud resources. Despite the immense traction of containers in cloud computing, there is no comprehensive survey that covers container scheduling techniques. In this timely survey, we investigate the landscape of the state-of-the-art container scheduling techniques aiming to inspire more research work in this active area of research. The survey is structured around classifying the scheduling techniques into four categories based on the type of optimization algorithm employed to generate the schedule namely mathematical modeling, heuristics, meta-heuristics and machine learning. Then for each class of scheduling algorithms, we analyze and identify key benefits and pitfalls, together with key challenges of the available techniques based on the performance metrics. Finally, this paper highlights fertile future research opportunities to realize the full potential of the emergent container technology.

Contents

* Corresponding author.
E-mail addresses: Imtiaz.ahmad@ku.edu.kw (I. Ahmad), alfailakawi.m@ku.edu.kw (M.Gh. AlFailakawi), asayel.almutawa@grad.ku.edu.kw (A. AlMutawa), latifah.alsalman@grad.ku.edu.kw (L. Alsalman).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.03.002
1319-1578/

1. Introduction

Cloud computing has become one of the most prevalent technologies in recent years by providing economies, society, and individuals with on-demand computing services over the Internet (Varghese and Buyya, 2018). With the recent growth of diverse and complex cloud workloads such as Internet of Things (IoT) devices, machine learning driven applications, streaming audio/video services, and cloud storage, the demand for various cloud services has increased dramatically. Furthermore, the growth of cloud services is predicted to expand even further because of the rising popularity of micro-services (Jamshidi et al., 2018), autonomous vehicles (Wang and Bao, 2019), and smart infrastructures (Jaiswal et al., 2018). Virtualization technology is considered the backbone of cloud computing, a technology that is capable of decoupling applications from the underlying infrastructure by allowing sharing of resources (CPUs, memory and network) to run diverse applications in an isolated manner (Alouane and El Bakkali, 2016).

As an emerging virtualization solution, containers are gaining tremendous popularity in recent years (Pahl et al., 2017) and are replacing traditional virtual machines (VMs) due to many promising properties such as shared host OS, quick launch time, portability, scalability, and fast deployment (Chae et al., 2019). Containers enable applications to encapsulate all necessary dependencies (code, runtime, system tools, and system libraries) into a sandbox in order to build a platform-independent run-time environment to improve productivity and portability (Watada et al., 2019). Several container technologies exist including Docker, LXC, and Kubernetes among others (Bernstein, 2014). In addition, some cloud service providers are running containers on top of VMs to improve containers isolation, enhance functionality, and simplify system management (Mavridis and Karatza, 2019). Container technologies are skyrocketing in popularity among developers and in deploying all kinds of micro-services and applications such as smart vehicles, IoTs and fog/edge computing (Casalicchio and Iannucci, 2020; Jamshidi et al., 2018; Morabito et al., 2017; Santo et al., 2019; Wang and Bao, 2019). Therefore, many cloud service providers have started to offer container-based cloud services to furnish its growing demand. Examples include Google Container Engine (cloud.google.com), Amazon Elastic Container Service (aws.amazon.com), and Azure Container Service (azure.micorsoft.com).

Container technology is revolutionizing the cloud computing paradigm (Buyya et al., 2018). From cloud service provider's perspective, running containerized applications creates an abstraction layer that handles cluster management. Docker Swarm (https://docker.com) and Google Kubernetes (https://kubernetes.io) are the dominant container orchestration platforms to provide container-based infrastructure with automatic deployment, scaling, and operation of containers on the underlying cluster. The typical structure of a cluster of containers consists of a manager and worker nodes. Worker nodes are responsible for running containers with workloads that are submitted by users; on the other hand, the manager node is responsible for orchestrating both the cluster and containers deployment on worker nodes. In addition, the manager node maintains the state of the cluster by continuously checking nodes status. An important and crucial role in container orchestration is played by the scheduler's component which is responsible for distributing workload across available nodes in the cluster and managing life cycle of containers.

The intractable container scheduling problem plays a vital role in cloud service performance and cost. Schedulers are often pluggable components of orchestrators. In Docker Swarm and Kubernetes orchestration platforms, three main scheduling strategies typically exist for deploying containers on computing nodes. Binpack strategy uses few nodes to pack as many containers as possible by scheduling container on the most loaded node that still have enough resources to run the given containers. Random strategy on the other hand schedules containers on any randomly selected node. Sparse strategy schedules containers on the least loaded node in order to distribute the load evenly across the cluster. Recently, new scheduling strategies have been proposed in Docker Swarm based on classes (Cérin et al., 2017). More details about the taxonomy and classification of schedulers in orchestration platforms can be found in the work of Rodriguez and Buyya (2019). The widespread use of containers in providing cloud services introduces many new challenges in the area of run-time resource scheduling.

Container scheduling can take on different forms depending on the underlying technology. For example, Fig. 1 shows a generalized block diagram of containers scheduling. Depending on implementation details, the incoming tasks (T_i) from users can be scheduled directly on container (C_j) running on physical machine (PM) or on virtual machine (VM) running on the physical hardware. Nevertheless, the responsibility of the scheduler is to identify the best placement of these incoming tasks to find the best possible schedule by taking into consideration various performance metrics such as utilization, makespan, power, . . .etc.

Container scheduling has become a critical part for the cost-efficient operation of modern applications on the cloud due to the diverse nature of the workloads and available cloud resources (Fazio et al., 2016). Researchers have set forth many state-of-the-art scheduling algorithms to achieve various performance

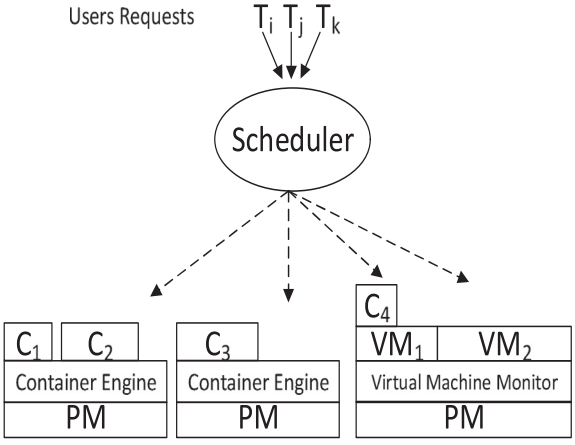

Fig. 1. Scheduler block diagram.

objectives such as reliability, response time, energy consumption, cost, load balancing, availability, and resource utilization among others. There exist many surveys discussing VM scheduling (Arunarani et al., 2019; Kumar et al., 2019; Adhikari et al., 2019) whereas there are only limited surveys dealing with resource management in container based systems (Maenhaut et al., 2019). Despite the tremendous popularity of containers, there is no comprehensive survey that covers container scheduling techniques to identify active area of research needed to improve containers scheduling. Therefore, in order to advance research in this important and growing area of research, it is crucial to study the landscape of existing container scheduling techniques and understand their strengths and limitations.

This timely survey highlights existing container scheduling techniques focusing on research trends and gaps. First of all, we classify existing scheduling techniques based on the type of optimization algorithm employed to generate schedules and describe the associated performance metrics suitable for cloud environment. Then for each class of scheduling algorithms, we analyze and report key benefits and limitations of these algorithms with respect to the performance metrics. The main goal of this work is to familiarize researchers with the latest trends in container scheduling. However, designing an efficient-general purpose container scheduling technique is a challenging task due to various factors such as dynamic workload and diverse resource requirements among others. Based on the detailed analysis of the state-of-the-art container scheduling techniques, we report several challenges that opens interesting research opportunities for future work.

The rest of this paper is structured as follows. Classification of container scheduling and common performance metrics are discussed in Section 2. Scheduling techniques are surveyed and evaluated in Section 3. Challenges and future research directions are described in Section 4. Finally, Section 5 concludes the paper.

2. Scheduling classification and performance metrics

This work summarizes the work done between January 2017 and January 2021 from scientific journals including IEEE, ACM, Elsevier, Springer as well as other international journals and conferences. We have searched different keywords related to container scheduling to collect papers. Finally, 55 papers were selected for the review based on their relevance to the topic. Among the selected papers, one paper was from 2021, 10 papers from 2020, 23 from 2019, 14 from 2018 and 7 from 2017. In this section, we first classify the surveyed scheduling techniques based on the optimization approach used to find the solution. Then, we describe common performance metrics used to evaluate the quality of the solution.

2.1. Scheduling classification

The surveyed scheduling techniques were found to come from four different categories with varying characteristics in terms of solution quality and run time performance. Majority of the works examined propose algorithms that fit in one of the following four categories:

1. Mathematical modeling
2. Heuristics
3. Meta-heuristics
4. Machine learning

Mathematical modeling techniques model the scheduling problem as set of constrained equations such as Integer Linear Programming (ILP) formulation and then use standard techniques to find optimal solution for the problem (Sierksma and Zwols, 2015). However, due to their prohibitive computational cost, ILP formulations can only be used for small size problems.

As container scheduling problem is an NP-hard problem, there is no polynomial complexity algorithm to find optimal schedule for large size problems. Therefore, the majority of the reported techniques are applying some heuristics to find approximate solutions to the problem (Christensen et al., 2017). Heuristic algorithms are generally of low complexity and generate a satisfactory schedule in a reasonable amount of time.

Meta-heuristics are a popular class of population-based optimization algorithms inspired by the intelligent processes and behaviors arising in nature (Hussain et al., 2019). Two important characteristics of such algorithms are selection of the fittest and adaptation to the environment. Meta-heuristics are widely used to solve optimization problems in several disciplines.

Machine learning is an active field of research with a lot of success in a wide range of applications in many disciplines and is very promising when used for container scheduling (Pouyanfar et al., 2018). Machine learning algorithms are successful because of the availability of big data to train the model. However, these algorithms have not been explored fully for container scheduling. Compared with other heuristics, one can benefit from machine learning techniques to improve solution accuracy and effectiveness by making intelligent scheduling decisions.

2.2. Performance metrics

In order to devise efficient schedule for containers, all previous works utilize some type of objectives to quantify the performance of the realized solution and evaluate the efficiency of the proposed approach. Some of the well known optimization objectives used include energy efficiency, availability, resource utilization, load balancing, scalability, cost, and makespan/latency. There exist other optimization objectives that can be used in certain application environment or with specific data center characteristics. Readers are referred to the recent survey to examine such domain specific optimization objectives (Gill and Buyya, 2018).

An optimization objective is used to measure certain aspects of the solution generated by an algorithm. In some papers, the optimization objective may consist of a single-objective while others may be multi-objectives depending on optimization needed for the problem at hand. Generally, the more objectives used in the cost function for the considered optimization problem, the more complex the decision-making process becomes. Therefore, different trade-offs are normally set in place to balance the performance of the proposed algorithm and the quality of the generated solution. The following are considered the most common objectives utilized in cost function definition of containers scheduling problem.

Energy efficiency: Energy consumption refers to the amount of power consumed when deploying containers on worker nodes. This objective tries to find schedules that minimize overall power consumption of the cluster. Minimizing power consumption reduces the cost of operating data centers by reducing energy and cooling cost resulting in increased revenues for large scale companies such as Microsoft and Google. Furthermore, energy efficiency is also a key metric for reducing carbon footprint and improve sustainability aspects (Gill and Buyya, 2018). Therefore, many scheduling techniques try to optimize energy usage by consolidating container deployment and/or using renewable energy sources whenever possible to enhance energy characteristics of the solutions generated.

Cost: The overall cost of application execution depends on the cost of various services such as computation, storage, and

communication cost. Computation cost refers to the time spent running an application on available cores in the cluster. The more time the application spends on a processor the higher the cost. Moreover, communication costs refers to the cost paid to network providers to provide telecommunications services needed to run the application. Increase in communication needed between node/clusters increases communication cost.

Availability: It refers to the amount of time an application is available to the user. Availability is one of the most important aspects of cloud computing where the user should have access to the application/service whenever he/she wishes. Such objective must guarantee that the schedule generated must provide some type of fault tolerance for the various services provided by the application which requires redundancy in the number of containers deployed.

Resource utilization: This metric refers to the efficiency of utilizing available resources on a worker node in terms of core, memory, and network bandwidth. For this metric, the higher the utilization of worker node resources, the more energy efficient and cost-effective the node becomes.

Load balancing: It is the process of evenly distributing the workload across computing resources such that no computing node becomes overloaded while others are idle. This objective also impacts response time, cost, and throughput. This metric is an important one especially in container applications that use micro-service architectures due to the dynamic fluctuation of the workload.

Scalability: Containers scalability is another key metric that deals with the ability to continually provide the needed service even when there is increased demand on the system whether from a single application or different ones. This metric measures the capability of the system to be reconfigured to increase available resources or to deploy additional containers on the cluster to accommodate for workload changes.

Makespan/Latency: It is the total time required to run the application from beginning to the end. A good scheduler always tries to reduce the makespan. This metric impacts throughput and/or cost and is considered a crucial one for real time applications that require low latency or have hard deadlines.

Throughput: This metric is calculated by dividing the total number of tasks of an application by the amount of time required to execute the tasks. Throughput provides an overall view of the performance of the system (processor, memory or a network), but cannot guarantee lower latency and cost.

Security: It is an ability of the orchestration to protect both data and services from malicious attacks or software bugs by encryption and access control mechanisms. This metric is more effective in online transaction processing, central financial services and productivity applications. Special care must be taken in case of containers since they are exposed to critical security threats.

Network Bandwidth: It is the number of bits transferred in one second on the network. This metric can help in evaluating network delay during communication among containers and handling network congestion.

Carbon Footprint: It is the amount of carbon dioxide (CO_2) released into the atmosphere as a result of the energy consumption specially produced through the burning of fossil fuels. This metric is important for sustainability aspects. Therefore, it is important to reduce energy consumption and harness clean energy to decrease carbon footprints and the use of fossil fuels.

3. Container scheduling techniques

In this section, we review recent container scheduling techniques proposed in the past four years and report their strengths, limitations, and any unique characteristic when appropriate. In the discussion that follows, a limitation of a technique refers to a missing optimization objective(s) in problem formulation and may impact the performance of the scheduler.

3.1. Mathematical modeling techniques

Integer Linear Programming (ILP) is a mathematical modeling technique that represents the optimization of a linear function using a set of linear constraints. Branch-and-bound methods have been the main tools for solving problems modeled using ILP formulation. The ILP fomulation of container scheduling problem is NP-complete in complexity, thus, large scale problems cannot be solved in polynomial time. Therefore, many times, heuristics are normally preferred over ILP formulation in order to realize solution in a timely manner.

Zhang et al. (2017) proposed the first linear programming model for the deployment of containers to computing nodes. The model considered different optimization criteria such as energy consumption and network cost. Results obtained in comparison with Docker Swarm binpack strategy showed a roughly 45% reduction in cost.

Zhou et al. (2018) proposed an ILP-based scheduling framework for container based cloud services. The formulation takes into consideration tasks deadline, resource constraints, and allowed partial task execution in order to minimize function value. The proposed technique is robust, however does not address energy reduction or other optimization objectives.

Similarly, Wan et al. (2018) reported an ILP formulation to minimize deployment cost of micro-service based applications in a Docker container environment. The technique considered networking cost and solved the scheduling problem in a distributed manner. However, this technique results in increased computational complexity and does not consider other important optimization criteria such as energy or response time.

Kaur et al. (2020) presented a multi-objective ILP-based formulation that minimizes carbon footprint, interference, and energy consumption for IoT applications in edge computing system. The proposed technique leads to considerable reduction in energy consumption and carbon footprint as compared to the solution found by the Kubernetes scheduler.

Lu et al. (2019) have designed an efficient offline as well as online scheduling technique that ensures deadline-based tasks to complete with minimum overhead delay for preemptive tasks. Results demonstrated the effectiveness of the proposed approach in meeting the deadlines and in reducing total task interruption overhead.

Alahmad et al. (2019) proposed an ILP technique to maximize availability by satisfying five specific constraints namely: Mean Time to Failure (MTTF), Mean Time to Repair (MTTR), affinity, anti-affinity, and redundancy. Experimental results showed that the proposed technique achieved high application availability as compared with three Docker scheduling techniques. The same authors proposed earlier (Alahmad et al., 2018) a heuristic approach to increase service availability as well.

Rodrigues et al. (2019) proposed a multi-objective MILP (Mixed Integer Linear Programming) approach to minimize energy consumption by consolidating containers to few nodes and maximize both resource and link bandwidth utilization. Two GPU-accelerated multi-objectives heuristics were also reported for large scale scenarios.

A summary of ILP-based techniques described in this section is given in Table 1. None of the techniques is scalable or considers all optimization objectives. Moreover, since the time taken to solve these formulations increases considerably with increased problem size, these techniques are applicable to small size problems only.

Table 1
Summary of papers presenting mathematical approaches for container scheduling

Reference	Objectives							Container	Strengths	Limitations
	Energy	Availability	Utilization	Load Balancing	Scalability	Cost	Makespan	Network	Technology		
Zhang et al. (2017)	✓					✓		✓	Docker	-First ILP fomultation -Multi-objective	-Same type resources
Zhou et al. (2018)			✓			✓			–	-Deals with deadlines -Allows partial execution	-Does not consider energy
Wan et al. (2018)			✓			✓		✓	Docker	-Micro-service based apps -Reduces deployment cost	-Computation complexity -Does not consider energy
Kaur et al. (2020)	✓		✓	✓			✓		Kubernetes	-Effective in energy -IoT edge computing	-No use of clean energy
Lu et al. (2019)				✓			✓		–	-Consider deadlines -Preemptive tasks	-Few optimization criteria -Not tested in real settings
Alahmad et al. (2019)		✓	✓	✓			✓		Docker	-Formalized availability -Multi-objective	-Does not consider energy
Rodrigues et al. (2019)	✓		✓	✓				✓	–	-Robust -Multi-objective	-Computation intensive -Needs GPUs

For large size problems, heuristics are the preferred approach compared to ILP, however, ILP is beneficial to gauge heuristic solution quality.

3.2. Heuristic Techniques

Several heuristics have been proposed to solve the container scheduling problem as highlighted earlier. Most of these heuristics use bin packing technique, or a combination of existing techniques in Docker Swarm and Kubernetes to map containers to computing nodes. The heuristics are generally scalable and very fast in generating a schedule, however optimality of the solutions is not guaranteed.

Wu and Chen (2017) developed an Availability-assured Buffered-layer Prioritized scheduler (ABP) to reduce network traffic and shorten service scale-out latency in Docker Swarm. Comparison between ABP with the default algorithm in Docker Swarm showed a reduction in start-up time and faster task distribution.

Mao et al. (2017) implemented a Dynamic and Resource-Aware Placement Scheme (DRAPS) in Docker Swarm for a set of heterogeneous nodes. The scheme selects the node to deploy a container based on the available resources and the demand for that node by other services. Performance comparison with Swarm kit showed a more efficient and balanced usage of resources. However, network consumption was high.

Havet et al. (2017) proposed GENPACK container scheduling framework for the Docker environment. It exploits runtime time monitoring and garbage collection to deploy containers on cluster nodes followed by best-fit scheduling technique which was found to provide 23% energy efficient solution as compared to Docker Swarm.

Pongsakorn et al. (2017) proposed a proactive container rebalancing technique to utilize resources efficiently in LXC clusters. In their approach, containers are first classified into short-lived and long-lived ones where only long-lived ones are considered for migration when a certain preset resource threshold utilization is reached. The migration strategy is fast but suffers from limited optimization objectives and lack of proper testing under real environment.

Dziurzanski and Indrusiak (2018) proposed a market-value inspired heuristic to allocate containers in Docker Swarm with the goal to maximize total value gained from a workload. In their scheme, the execution value of a container varies with time defined by the value curve of the job. However, their scheme does not deal with memory usage or network utilization to enhance performance.

Zhang et al. (2018) proposed a container scheduling algorithm based on stretch-out and compact technique. The proposed algorithm saves network bandwidth and requires less execution time as compared to Docker. In addition, the scheme takes care of dependencies among tasks during scheduling.

Liu et al. (2018) proposed a multi-objective algorithm, namely Multiopt, which combines the advantages of the three Docker Swarm scheduling policies, spread, binpack and the random to make container scheduling more efficient. Furthermore, Multiopt took into consideration five optimization criteria to generate better schedules including CPU utilization, memory usage, network delay, association between containers and nodes, and containers clustering. Comparison with Docker Swarm policies showed a superior performance, however, the scheme is not fault-tolerant.

Song et al. (2018) proposed a topology-based GPU scheduling algorithm called Gaia for Kubernetes. The authors used resource access cost tree to make optimum dynamic scheduling decisions in order to enhance resource utilization and computational efficiency. Experimental results in production environment of Tencent showed 10% improvement in resources utilization as well as reduc-

tion in network overhead. However, Gaia scheduler is only applicable for GPU cluster environment.

Lv et al. (2019) proposed a two-stage scheduling technique that targets the reduction of communication cost as well as balancing resource utilization. During the first stage, a set of newly arriving containers are placed on servers with the goal to enhance resource utilization and minimize communication cost by using Communication Aware Worst Fit Decreasing (CAWFD) algorithm. During the second stage, containers are migrated among servers to enhance performance by using Sweep & Search algorithm. Experimental results on Baidu's data centers showed its effectiveness in reducing communication cost and improving utilization.

Mendes et al. (2019) suggested an enhanced Docker Swarm scheduling algorithm in order to increase its energy efficiency in fog/edge computing environment. Because of scarcity of resources in these environments, an oversubscription strategy was opted in order to enhance resource usage and save energy consumption. From the obtained results, it was found that the suggested solution achieved better resource utilization resulting in reduced energy. However, due to the complexity involved in decision making process, the scheduler takes more time to generate a solution as compared to Docker Swarm strategies.

Chen et al. (2019) proposed a container scheduling technique based on the popular Min-Min heuristic to improve energy efficiency of edge computing system. The authors modified the original Min-Min algorithm by allocating containers on computing nodes with the least energy consumption. Experimental results showed a 56% reduction in energy consumption as compared with the First-Fit container placement policy.

Hu et al. (2019) proposed a vector bin packing based container scheduling algorithm with multiple objectives cost function. The objectives include resource utilization, load balancing, and network dependency. The technique co-located containers with dependencies on the same host resulting in overall performance improvement as compared to Kubernetes scheduler.

Menouer and Darmon (2019) reported a scheduling technique that leverages both spread and bin packing strategies in Docker Swarm kit using a multi-criteria algorithm called "Technique for the Order of Prioritization by Similarity to Ideal Solution" (TOPSIS). The proposed technique demonstrated its potential to enhance performance as compared with Docker Swarm default strategies such as bin packing, spread, and random.

Xu et al. (2018, 2019) proposed an energy reduction technique based on brownout. In brownout, under overload condition, some of the optional services are deactivated temporarily to reduce energy consumption. The authors proposed three different container scheduling polices based on brownout paradigm that resulted in 10–40% energy saving as compared to existing techniques. However, network congestion and renewable energy sources were not considered in their approach.

Kumar et al. (2018) proposed a renewable energy-based container scheduling scheme. In the proposed technique, incoming workload is first classified into three categories based on their priority, load, and availability. Then, the workload is scheduled on nodes selected using a renewable energy-aware host selection scheme. Furthermore, a container consolidation scheme was devised to reduce energy consumption of computing nodes. The proposed approach has been evaluated using Google workload traces and result suggests that up to 15% energy saving can be achieved as compared with First-Fit or random host selection schemes.

James and Schien (2019) designed and implemented a low carbon footprint scheduling policy for Kubernetes where this technique migrates the load to countries with the lowest carbon footprint.

Fu et al. (2019) proposed ProCon, a scheduling technique that considers instant resource utilization as well as the estimation of future usage to reduce application response time. Comparison with Kubernetes shows an improvement in overall performance by 23%.

Khan et al. (2019) introduced a set of techniques to perform container consolidation with the aim to reduce energy consumption and enhance performance in data centers. The proposed techniques include an energy consumption estimation model and a migration methodology that takes into consideration both performance and energy characteristics of the final solution.

Hu et al. (2020) formulated the concurrent multi-resource container scheduling problem as a minimum-cost flow model. Experimental results revealed that the proposed technique outperformed state-of-the-art container scheduling techniques, Docker Swarm and Kubernetes, both in terms of resource efficiency and performance. However, container dependencies were not considered in their approach.

Menouer (2020) presented a multi-criteria Kubernetes Container Scheduling Strategy called KCSS with the goal to optimize makespan and power consumption. In KCSS, the author extended his earlier approach (Menouer and Darmon, 2019) by considering six criteria in an effort to find an improved way for selecting a node for container scheduling. However, KCSS did not consider fault-tolerance and container consolidation to further enhance power consumption.

Rodrigues et al. (2020) proposed a GPU Accelerated Containers Scheduler (GPUACS), in which the joint allocation of network and containers with Quality of Service (QoS) requirements was modelled as a graph embedding problem. The authors considered multi-criteria in their scheduling decision making and used the GPU to make the schedule faster and scalable. The GPUACS was able to schedule a variety of large scale requests on a data center with more than 20,000 servers in less than 3.5 s even when considering network and time requirements. However, the authors did not consider load balancing and fault-tolerance aspects of scheduling.

A summary of all studied heuristics in this section is given in Table 2. It is observed that only few heuristics focused on the optimization of system availability. Heuristics are generally fast in generating solutions and can be integrated with other optimization techniques to further enhance the quality of the generated schedules.

3.3. Meta-heuristic techniques

Recently, meta-heuristic algorithms have become dominant for solving challenging optimization problems in diverse fields. Meta-heuristic algorithms can be divided into evolutionary algorithms (EAs) such as Genetic Algorithm (GA) or swarm intelligence algorithms such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Whale Optimization among others. In this section, we have grouped scheduling techniques based on the type of meta-heuristic employed to generate the solution.

3.3.1. ACO-based techniques

Ant colony optimization is a population-based search algorithm based on the pheromone trail laying behavior of ants in searching for food.

Kaewkasi and Chuenmuneewong (2017) presented the first Ant Colony Optimization algorithm for container scheduling with the goal to enhance resource utilization using proper load balancing. The algorithm was integrated and tested on Docker Swarm. The results were compared with Swarm greedy approach showing 15% performance enhancement. However, the algorithm considered only few optimization objectives such as resource utilization and load balancing.

Burvall (2019) presented a Multi-Objective Container Placement Ant Colony Optimization (MOCP-ACO) algorithm that modi-

Table 2
Summary of papers presenting heuristic approaches for container scheduling

Reference	Objectives								Container	Strengths	Limitations
	Energy	Availability	Utilization	Load Balancing	Scalability	Cost	Makespan	Network	Technology		
Wu and Chen (2017)		✓					✓		Docker	-Reduce latency -Enhance availability	-Homogeneous resources -Does not consider energy
Mao et al. (2017)			✓		✓			✓	Docker	-Stable performance -Migrate containers	-More network consumption
Havet et al. (2017)	✓	✓					✓		Docker	-Garbage collection based -Container consolidation	-Requires monitoring -Homogeneous resources
Pongsakorn et al. (2017)			✓	✓					LXC	-Fast container migration -Container classification	-Requires multi-objective -Need more testing
Dziurzanski and Indrusiak (2018)			✓			✓			Docker	-Consider value curve -Market-inspired heuristic	-No memory footprint size -No network cost
Zhang et al. (2018)					✓	✓	✓	✓	Docker	-DAG flow model -Reduce network bandwidth	-Homogeneous resources -Energy not considered
Liu et al. (2018)			✓	✓	✓		✓		Docker	-Multi-objective model -Based on Docker swarm	-Not fault-tolerant -No energy reduction
Song et al. (2018)			✓				✓	✓	Kubernetes	-Utilize resource efficiently -Consider network cost	-Works for GPU cluster
Lv et al. (2019)			✓	✓	✓		✓	✓	Baidu' DS	-Worst-fit bin packing -Container migration	-Two stage formulation -Energy not considered
Mendes et al. (2019)	✓		✓						Docker	-Fog/edge computing -Over-subscription strategy	-Complex decision process -Slow in schedule generation
Chen et al. (2019)	✓								–	-Use min-min algorithm -Edge computing domain	-Optimize single objective -Homogeneous resources
Hu et al. (2019)			✓	✓	✓			✓	Kubernetes	-Use container consolidation -Vector bin packing	-Not tested on real systems -Timing deadline constraints
Menouer and Darmon (2019)				✓			✓		Docker	-Spread + bin packing -Multi-criteria decision	-Not fault-tolerant -Not tested on real settings
Xu et al. (2018) and Xu and Buyya (2019)	✓		✓		✓		✓		Docker	-First brownout scheme -Energy efficient	-No network congestion -Triggering threshold value
Kumar et al. (2018)	✓		✓	✓					–	-Consolidation + migration -Use renewable energy	-Migration overhead
James and Schien (2019)	✓					✓			Kubernetes	-Considers carbon density -Select greenest data center	-Need to consider price -Migration takes time
Fu et al. (2019)			✓	✓			✓		Kubernetes	-Enhance performance -Predict resource usage	-Need monitoring -Does not consider energy
Khan et al. (2019)	✓						✓		CloudSim	-Develop energy model -Use container migration	-Estimation of run time -Renewable energy sources
Hu et al. (2020)			✓	✓	✓		✓	✓	Kubernetes	-Min-cost flow model -Heterogeneous resources	-Dependency not considered -Energy reduction
Menouer (2020)	✓						✓		Kubernetes	-Six criteria approach -Easily extendable	-No container consolidation -No fault-tolerance
Rodrigues et al. (2020)							✓	✓		-Graph embedding formulation -Multi-criteria approach	-No fault-tolerance -No load-balancing

fied the standard Ant Colony implementation to use additional objectives such as network usage and cost. MOCP-ACO performed better for all tested cases by varying the workload and number of containers (16–1024) when compared to the spread scheduling strategy in Docker Swarm.

Lin et al. (2019) proposed a multi-objective ant colony scheduling technique that optimizes resource utilization (CPU and memory), network transmission, and failure rate of micro-service based applications. Experimental results with existing techniques such as Spread and GA-based approaches showed an improved resource balancing and reliable service. However, the algorithm did not consider energy consumption.

3.3.2. GA-based techniques

Genetic algorithms are evolutionary search techniques inspired by the theory of natural evolution. In the evolution process, individuals are selected based on their fitness to produce offspring for the next generation.

Guerrero et al. (2018) were the first to propose a NSGA-II, based container scheduling technique. NSGA-II is a multi-objective optimization algorithm. The authors considered four optimization objectives in their formulation which includes network communication overhead, load balancing, failure rate, and resource utilization. Compared to Kubernetes scheduling algorithm, the proposed technique showed 40–60% enhancement in most of the targeted objectives.

Zhang et al. (2019) proposed an improved genetic algorithm using a non-linear energy model to reduce energy consumption. Two new mutation operators were devised to efficiently search for optimal solution. The proposed scheme however considered a single objective that is energy optimization.

Tan et al. (2019) proposed a two level hybrid algorithm to reduce energy consumption. Similar to genetic algorithm, genetic programming is a population based evolutionary computation technique. In this scheme, containers are first mapped to virtual machines (VMs) and then VMs are allocated to physical machines (PMs). Experimental results showed that the hybrid approach significantly reduced energy consumption as compared to heuristic only based approaches such as first-fit, best-fit, . . .etc.

Imdoukh et al. (2019) proposed a Many-Objective Genetic Algorithm (MOGAS) based on NSGA-III. NSGA-III is an improved version of NSGA-II to handle more objective functions. The authors considered five optimization criteria which include power consumption, availability, resource utilization, load balancing, and scalability. Comparison with ACO-based scheduling technique demonstrated the effectiveness of the approach.

Tan et al. (2020) extended their earlier work (Tan et al., 2019), by applying Cooperative Coevolution Genetic Programming (CCGP) hyper-heuristic approach to solve the two-level container allocation problem. The CCGP approach generated allocation rules for both levels (containers-VMs and VMs-PMs levels) simultaneously to minimize the overall energy consumption. Experiments showed considerable improvement in energy consumption as compared to the state-of-the-art algorithms.

Dhumal and Janakiram (2020) proposed C-Balancer, a scheduling framework which leverages runtime metrics of containers for efficient placement in cluster environment. In the proposed scheme, containers runtime metrics are periodically collected and then sent to GA-based optimizer to find the most stable and optimal container for node placement. The key idea behind the approach is to re-balance containers by migrating across multiple nodes using runtime metrics. The experiment conducted in the Swarm cluster showed the effectiveness of C-Balancer in improving the performance in terms of resource utilization and throughput.

3.3.3. PSO-based techniques

PSO algorithm is inspired by the behavior of birds and one of the popular meta-heuristic algorithms used due to its robust characteristics.

Li et al. (2018) proposed a PSO algorithm that improves resource utilization and load balance. In their implementation, the authors used simulated annealing algorithm with PSO to escape local minimas. Experimental results revealed that the proposed implementation improved performance by 20% as compared to the spread strategy of Docker Swarm.

Guo and Yao (2018) proposed a PSO-based scheduling algorithm that exploits the neighborhood division idea to tune the parameters of the PSO algorithm to generate better quality solutions. The algorithm considered both response time and load balancing to enhance system performance. In the proposed implementation, containers with dependencies were placed on neighboring hosts in order to reduce communication cost. Results showed a 20–25% enhancement as compared to a spread scheduling and traditional PSO algorithm. One disadvantage for the solution generated was the increase in power consumption.

Shi et al. (2018) solved container consolidation problem by using a Two-stage Multi-type Particle Swarm Optimization (TMPSO) algorithm. The TMPSO algorithm integrated heuristic and greedy techniques to improve solution quality. Experimental results showed a significant saving in energy consumption as compared to standard and binary PSO algorithms.

Adhikari and Srirama (2019) reported an accelerated multi-objective particle swarm algorithm to schedule containers for IoT tasks with objectives to minimize energy consumption as well as computation time. In addition, the proposed technique considered minimizing CO_2 emission and temperature of computing nodes.

Liu et al. (2020) proposed particle swarm optimization based container scheduling algorithm for big data applications for Kubernetes called K-PSO. The authors modified the standard PSO with dynamic inertia weight and learning factor for fast convergence with better quality solutions. Based on the improved PSO, the authors introduced enhancements to the predicate scheduling process and the priority scheduling process of the Kubernetes native scheduler. Experimental results reported 20% improvement in resource utilization as compared with Kubernetes default scheduling strategy. However, the proposed scheme can be extended to include multi-criteria optimization objectives.

Fan et al. (2020) proposed a latency, reliability and load balancing aware scheduling (LRLBAS) algorithm based on PSO to determine container-based microservice deployment in edge computing paradigm. Comparison of experimental results with variants of PSO demonstrated effectiveness and efficiency of the LRLBAS algorithm for microservice scheduling in edge computing. The authors planned to include more objectives in the formulation in the future.

3.3.4. Other meta-heuristic techniques

Tao et al. (2017) proposed a fuzzy interface system (FIS) based container scheduling algorithm. In their implementation, authors considered factors such as CPU, memory, I/O, and network usage based cost function when deploying containers to nodes. The technique showed superior results as compared with Round-Robin scheduling algorithm, however, did not consider optimization of energy consumption.

Salp swarm is a population based meta-heuristic algorithm that mimics predatory behavior of salp swarm. Ma et al. (2019) reported an enhanced version of the salp swarm algorithm for redundant container deployment problem. The aims of the proposed technique was to optimize the availability and response time of services. Results with Docker scheduling policies and other meta-heuristics showed the efficiency of the approach.

Wei-guo et al. (2018) proposed a hybrid scheduling technique combining ACO with PSO for Kubernetes scheduler to reduce the resources cost with proper load balancing.

Vhatkar and Bhole (2019) reported a new multi-objective resource allocation scheme by combining Lion Algorithm (LA) with Whale Optimization (WOA). Four objectives were considered including availability, load balancing, reliability, and communication overhead. Experimental results proved superiority of solution quality of the proposed technique as compared with GA and NSGA-II based techniques.

As compared with existing approaches, Al-Moalmi et al. (2021) formulated the placements of containers to VMs and VMs to PMs as one optimization problem and then adapted Whale Optimization Algorithm (WOA) to solve the problem while considering minimizing power consumption and maximizing resources utilization as their objectives. Comparison of experimental results against existing methods conduced over different levels of heterogeneous environments demonstrated the superiority of the proposed method in reducing the resources needed and power consumption. However, the scheme does not consider containers or VMs migration to further reduce power consumption.

A summary of meta-heuristics scheduling techniques described in this section is given in Table 3. Meta-heuristic are more suitable for multi-objective optimization, however, due to their computation complexity when dealing with large size problems, the parallelism of such algorithms on platforms such as Apache Spark or GPU clusters is a must to reduce execution time.

3.4. Machine learning techniques

Machine learning algorithms learn from data and make informed decisions based on what they have learned. Deep machine learning is a subfield of machine learning that use a layered structure of artificial neural networks to learn and make intelligent decisions on its own. Deep learning algorithms have shown a lot of success in image processing and computer vision and is being actively used in a wide range of disciplines.

Resource optimization in general is a challenging problem. In container consolidation problem, containers are mapped to a minimum number of computing nodes to increase resource utilization and/or reduce energy consumption. Nanda and Hacker (2018) proposed one of the first deep reinforcement learning technique targeting container consolidation to enhance resource utilization. The proposed technique achieved better results as compared to shortest job first and random placement algorithms.

Lv et al. (2019) presented a technique based on a random forest regression machine learning model to predict the number of needed containers. The proposed technique accurately predict future resource requirements to quickly respond to sudden workload changes. The proposed algorithm has shown a significant improvement in terms of time and accuracy as compared with other machine learning algorithms such as support vector machine.

Menouer and Darmon (2019) proposed a machine learning scheme targeting energy consumption. In their approach, authors used hourly request frequency data to form three submission periods (high, medium, low). K-Means unsupervised clustering algorithm was used to identify submission periods, and then based on submission period, dynamic decisions were made to consolidate containers. The technique was implemented on Docker Swarm and showed a considerable reduction in power consumption as well as the number of active nodes in the cluster.

Mehta et al. (2020) proposed WattsApp, a power-aware container scheduling scheme to minimize power cap violations on a server. The importance of such method lies in the fact that power cap violation on any server leads to degrading the performance of all containers running on the server. The proposed scheme employed a neural network-based power estimation model that uses independent metrics to measure resource utilization information of the container, thus making it portable as compared with earlier methods. WattsApp continuously monitors the servers for any power cap violation and when a violation is detected, power capping is performed either by migrating container or by container resource reduction. Nevertheless, the authors have reported that resource reduction is more feasible approach because of negligible overhead as compared to migrating containers. However, performance or Service Level Agreement (SLA) of workloads and power estimation on heterogeneous platforms were not considered in their scheme.

Liu et al. (2020) have used physical machine (PM)-level usage prediction (overutilized and underutilized) and load balancing (migrating containers) to improve SLA and energy efficiency in cloud datacenters. The authors used a linear regression model for resource utilization prediction which is a very good mean to facilitate proactive container consolidation decisions. Experimental results in ContainerCloudSim toolkits showed that the proposed scheme reduced power consumption while complying with SLA.

Nath et al. (2020) formulated the problem of green container-based service consolidation as an optimization problem with the objective to minimize total energy consumption footprint of data centers. The authors applied a statistical online learning technique based on Bayesian Optimization which require less data points to solve container consolidation problem. Experimental results have shown that the proposed scheme improved total energy consumption of data centers as compared with various existing approaches.

A summary of machine learning techniques discussed in this section is given in Table 4. One can observe from the table that only a handful machine learning techniques have been reported for container scheduling and only few optimization objectives were used in these solutions. Machine learning field promises a bright future for container scheduling and further exploration is needed to exploit its full potential.

4. Challenges and future research directions

With the recent explosive growth of emerging applications such as IoTs, micro-services, driverless vehicles, and smart infrastructures, containers have attained a massive usage within the cloud computing community. Hence, container scheduling problem has become of a paramount importance to efficiently manage runtime cloud resources. Analyzing container scheduling techniques revealed that a single algorithm cannot address all key performance objectives, and thus several challenges still remain to be addressed as key research opportunities.

An interesting area of future research can be to design a security aware scheduler to avoid security threat associated with containers when deployed over cloud infrastructures. A preliminary work in this regard was proposed by Vaucher et al. (2018) using Intel Software Guard Extensions (SGX) technology, which supports the creation of trusted execution environments. A recent work by Vhatkar and Bhole (2020) included security as a constraint in container allocation optimization problem. However, more extensive work is needed in this important aspect of container scheduling.

Due to the emergence of fog/edge computing paradigm, computation and storage is being pushed close to the user to support real-time applications to promote energy efficiency, improve response time, and save bandwidth. Applications such as 5G mission critical applications, IoT, e-commerce services, autonomous vehicle and content delivery networks are major players using such paradigm. Containers are the key virtualization technology employed to offer services in fog/edge computing. Since edge devices have limited

Table 3
Summary of papers presenting meta-heuristics approach for containers scheduling

Reference	Objectives								Container Technology	Strengths	Limitations
	Energy	Availability	Utilization	Load Balancing	Scalability	Cost	Makespan	Network			
Kaewkasi and Chuenmuneewong, 2017			✓	✓					Docker	-First ACO technique -Use resource efficiently	-No container migration -Require parameter tuning
Burvall (2019)						✓		✓	Docker	-Modified ACO technique -Consider network delay	-Does not consider energy -Slow
Lin et al. (2019)		✓	✓	✓				✓	–	-Reduce network cost -Use resources efficiently	-No container consolidation -No energy reduction
Guerrero et al. (2018)		✓	✓	✓				✓	Kubernetes	-First multi-objective GA -Based on NSGA-II	-No container migration -No energy reduction
Zhang et al. (2019)	✓								–	-Non-linear energy model -Focus on energy reduction	-Requires parameters tuning
Tan et al. (2019)	✓								–	-Genetic programming -Two level optimization	-Single objective -No cooperative coevolution
Imdoukh et al. (2019)	✓	✓	✓	✓	✓				Docker	-Consider 5 objectives -Based on NSGA-III	-Slow, need parallalization
Li et al. (2018)			✓	✓					Docker	-First PSO model -Improve performance	-Not fault tolerant -Requires parameters tuning
Guo and Yao (2018)				✓			✓	✓	CloudSim	-Fast -Improve performance	-No container migration -Consumes more memory
Shi et al. (2018)	✓		✓						–	-Multi-objective PSO -Fast	-No container migration -Requires parameters tuning
Adhikari and Srirama (2019)	✓		✓				✓		CloudSim	-Consider CO2 emission -Consider temperature	-Requires parameters tuning
Tao et al. (2017)			✓	✓			✓		Docker	-Fuzzy interface model -Use rule-based prediction	-No Communication cost -Requires better monitoring
Ma et al. (2019)		✓					✓		Docker	-First salp swarm model -Considers availability	-Communication overhead -Slow convergence
Wei-guo et al. (2018)				✓		✓			Kubernetes	-Hybrid ACO + PSO -Considers availability	-No container migration -Few optimization criteria
Vhatkar and Bhole (2019)		✓		✓				✓	–	-Hybrid whale-Lion model -Consider availability	-Lack of testing in real settings -Requires parameters tuning
Tan et al. (2020)	✓								–	-Cooperative coevolution genetic approach -Consider availability	-Single objective only -Requires parameters tuning
Dhumal and Janakiram (2020)			✓				✓		Docker	-GA based approach -Use container migration	-Requires parameters tuning
Liu et al. (2020)			✓						Kubernetes	-Modified-PSO technique -Fast convergence	-Single objective only
Fan et al. (2020)		✓		✓			✓		–	-Multi-objective PSO -Edge computing paradigm	-No energy reduction -Not tested in real-settings
Al-Moalmi et al. (2021)	✓		✓						CloudSim	-Multi-objective WOA	-No container migration -No containers consolidation

Table 4
Summary of papers presenting machine learning approaches for containers scheduling

Reference	Objectives							Container	Strengths	Limitations
	Energy	Availability	Utilization	Load Balancing	Scalability	Cost	Makespan	Network	Technology		
Nanda and Hacker (2018)			✓				✓	✓	–	-First deep learning formulation -Use resources efficiently	-No container migration -Few optimization criteria
Lv et al. (2019)			✓	✓			✓		Kubernetes	-Random forest Model -High accuracy	-Lack testing in real settings -Only predict resources
Menouer and Darmon (2019)	✓		✓	✓					Docker	-K-mean clustering model -Container migration	-Need better ML-model -Few optimization criteria
Mehta et al. (2020)	✓								Docker	-Neural network based model -Container migration	-Does not consider performance
Liu et al. (2020)	✓		✓						CloudSim	-Linear regression model -Container migration	-Does not consider communication cost
Nath et al. (2020)	✓								Docker	-Container consolidation problem formulation -Bayesian optimization method	-Needs evaluation in multi-cloud environment

computational resources and battery capacity, more resource-aware and energy-efficient container scheduling techniques are needed to serve this emerging domain of applications. Although, preliminary research work has been initiated in this direction (Kaur et al., 2020; Chen et al., 2019; Hu et al., 2019; Luo et al., 2019; Rausch et al., 2021), fog/edge computing offers a fertile field to explore real-time energy efficient container scheduling in such environment. Specifically, the emerging quantum computing based optimization (Gill et al., 2020) may be more attractive technique to explore for scheduling decision making in fog/edge computing environments (Bhatia and Sood, 2020).

It was observed from the survey of related work that traditional container scheduling heuristics lack learning components and their performance deteriorates when unexpected information is encountered during scheduling. To avoid such undesired results, deep learning techniques can provide powerful and robust way to address container scheduling, however, it requires further investigation (Nanda and Hacker, 2018; Lv et al., 2019; Yang et al., 2018; Sung et al., 2019). Machine learning can help to devise workload specific scheduling policies (Mao et al., 2019) and can provide the needed intelligence to predict future workload and resource needed for scheduling decisions. However, in order to exploit the full potential of machine learning, workload traces spanning many years are needed to have the necessary data diversity to provide sufficient learning data (Kuchnik et al., 2019).

There has been a massive adoption of micro-services in different areas driven by their key advantages of flexibility, simplicity, and scalability. Cloud native application might consists of a large number of micro-services and application behavior depends on traffic flow among micro-services which incur non-negligible overhead and causes performance degradation (Suo et al., 2018). Therefore, communication among containers is of vital importance and a challenging task. This has led to the emergence of service meshes (Li et al., 2019), a dedicated infrastructure layer to manage service-to-service communication. It would be interesting to explore container scheduling that takes into consideration the availability of service mesh.

With the popularity of containers as a key technology for cloud services, the tremendous increase in the demand for cloud services resulted in rocketing increase in energy consumption and operating cost of data centers (Gill and Buyya, 2018). Therefore, to offer sustainable cloud services, extensive work is needed that target multi-objective holistic management techniques of container scheduling that takes into account data centers resources such networks, memory, processors, cooling and storage (Li et al., 2017; Gill et al., 2019; Townend et al., 2019). Furthermore, reduction of monetary cost is highly critical for big data applications and therefore must be included in the optimization objectives to avoid financial burden (Chung et al., 2018; Jiang et al., 2019). Multi-objectives meta-heuristic algorithms may be the most suitable candidates for holistic management techniques, however, solution quality and convergence time can be greatly improved further by utilizing local search techniques to generate initial population or by using hybridized algorithms such as integrating heuristics with meta-heuristic algorithms or combining two different meta-heuristics to explore optimization search space efficiently and effectively.

5. Conclusion

The usage of containers has been increasing dramatically within the cloud computing community for providing cloud services in recent years. Therefore, efficient management of cloud resources at run time through container scheduling has become of prime importance in cloud computing. In this work, we presented a comprehensive survey to study the landscape of container scheduling

techniques. We started with a classification of scheduling techniques into four categories based on the optimization algorithm employed to generate the schedule. Then, we examined optimization objectives to quantify the performance of the generated schedules. Next, we characterized and summarized existing techniques for each category based on the performance metrics with the aim to understand their strengths and limitations. Finally, as can be observed from the emerging trends and proposed future research directions, the evolution of container technology in the future will bring forth significant new requirements leading to the new generation of research solutions for resource management and scheduling. Emerging technology trends, such as fog/edge computing and micro-services, offer novel opportunities to explore real-time energy-aware, communication-aware and security-aware schedulers for these environments. In addition, it is conceivable that the continuing success and interest for deep learning algorithms will help in devising intelligent scheduling policies by predicting future workloads and resources for container consolidation, resource provisioning, and load balancing among others. To make the best use of deep learning algorithms, there is a need to deploy more fine-grained system counters (hardware-level or kernel-level) to monitor and collect application information for prediction and decision making. Furthermore, to offer energy-efficient and sustainable cloud services, multi-objective holistic management container scheduling techniques need to be pursued to manage energy consumption, service level agreements (SLAs) and quality-of-service (QoS) at the same time. We believe that this survey will spur research in this field to exploit the full potential of container technology.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank the anonymous reviewers for their invaluable comments and constructive criticism to improve the quality of the manuscript.

References

Adhikari, M., Amgoth, T., Srirama, S.N., 2019. A survey on scheduling strategies for workflows in cloud environment and emerging trends. ACM Comput. Surv. (CSUR) 52 (4), 1–36.

Adhikari, M., Srirama, S.N., 2019. Multi-objective accelerated particle swarm optimization with a container-based scheduling for internet-of-things in cloud environment. J. Netw. Comput. Appl. 137, 35–61.

Al-Moalmi, A., Luo, J., Salah, A., Li, K., Yin, L., 2021. A whale optimization system for energy-efficient container placement in data centers. Expert Syst. Appl. 164, 113719.

Alahmad, Y., Daradkeh, T., Agarwal, A., 2018. Availability-aware container scheduler for application services in cloud. In: 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC). IEEE, pp. 1–6.

Alahmad, Y., Daradkeh, T., Agarwal, A., 2019. Optimized availability-aware component scheduler for applications in container-based cloud. In: 2019 Sixth International Conference on Software Defined Systems (SDS). IEEE, pp. 194–199.

Alouane, M., El Bakkali, H., 2016. Virtualization in cloud computing: Existing solutions and new approach. In: 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech). IEEE, pp. 116–123.

Arunarani, A., Manjula, D., Sugumaran, V., 2019. Task scheduling techniques in cloud computing: A literature survey. Fut. Gen. Comput. Syst. 91, 407–415.

Bernstein, D., 2014. Containers and cloud: From lxc to docker to kubernetes. IEEE Cloud Comput. 1 (3), 81–84.

Bhatia, M., Sood, S.K., 2020. Quantum computing-inspired network optimization for iot applications. IEEE Int. Things J. 7 (6), 5590–5598.

Burvall, B., 2019. Improvement of container placement using multi-objective ant colony optimization Master's thesis. KTH Royal Institute of Technology, Stockholm, Sweden.

Buyya, R., Srirama, S.N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L.M., Netto, M.A.S., Toosi, A.N., Rodriguez, M.A., Llorente, I. M., Vimercati, S.D.C.D., Samarati, P., Milojicic, D., Varela, C., Bahsoon, R., Assuncao, M.D.D., Rana, O., Zhou, W., Jin, H., Gentzsch, W., Zomaya, A.Y., Shen, H., 2018. A manifesto for future generation cloud computing: Research directions for the next decade. ACM Comput. Surv. (CSUR) 51 (5), 1–38.

Casalicchio, E., Iannucci, S., 2020. The state-of-the-art in container technologies: Application, orchestration and security. Concurr. Comput.: Pract. Exp. 32, (17) e5668.

Cérin, C., Menouer, T., Saad, W., Abdallah, W.B., 2017. A new docker swarm scheduling strategy. In: 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2) IEEE, pp. 112–117.

Chae, M., Lee, H., Lee, K., 2019. A performance comparison of linux containers and virtual machines using docker and kvm. Cluster Comput. 22 (1), 1765–1775.

Chen, F., Zhou, X., Shi, C., 2019. The container scheduling method based on the min-min in edge computing. In: Proceedings of the 2019 4th International Conference on Big Data and Computing, pp. 83–90.

Christensen, H.I., Khan, A., Pokutta, S., Tetali, P., 2017. Approximation and online algorithms for multidimensional bin packing: A survey. Comput. Sci. Rev. 24, 63–79.

Chung, A., Park, J.W., Ganger, G.R., 2018. Stratus: cost-aware container scheduling in the public cloud. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 121–134.

Dhumal, A., Janakiram, D., 2020. C-balancer: A system for container profiling and scheduling.

Dziurzanski, P., Indrusiak, L.S., 2018. Value-based allocation of docker containers. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE, pp. 358–362.

Fan, G., Chen, L., Yu, H., Qi, W., 2020. Multi-objective optimization of container-based microservice scheduling in edge computing. Comput. Sci. Inf. Syst. 00, 41.

Fazio, M., Celesti, A., Ranjan, R., Liu, C., Chen, L., Villari, M., 2016. Open issues in scheduling microservices in the cloud. IEEE Cloud Comput. 3 (5), 81–88.

Fu, Y., Zhang, S., Terrero, J., Mao, Y., Liu, G., Li, S., Tao, D., 2019. Progress-based container scheduling for short-lived applications in a kubernetes cluster. Proceeding of the IEEE International Conference on Big Data (BigData'19), 1–10.

Gill, S.S., Buyya, R., 2018. A taxonomy and future directions for sustainable cloud computing: 360 degree view. ACM Comput. Surv. (CSUR) 51 (5), 1–33.

Gill, S.S., Garraghan, P., Stankovski, V., Casale, G., Thulasiram, R.K., Ghosh, S.K., Ramamohanarao, K., Buyya, R., 2019. Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge. J. Syst. Soft. 155, 104–129.

Gill, S.S., Kumar, A., Singh, H., Singh, M., Kaur, K., Usman, M., Buyya, R., 2020. Quantum computing: A taxonomy, systematic review and future directions.

Guerrero, C., Lera, I., Juiz, C., 2018. Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. J. Grid Comput. 16 (1), 113–135.

Guo, Y., Yao, W., 2018. A container scheduling strategy based on neighborhood division in micro service. In: NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1–6.

Havet, A., Schiavoni, V., Felber, P., Colmant, M., Rouvoy, R., Fetzer, C., 2017. Genpack: A generational scheduler for cloud data centers. In: 2017 IEEE International Conference on Cloud Engineering (IC2E) IEEE, pp. 95–104.

https://docker.com, Accessed 5-Jan-2021.

https://kubernetes.io, Accessed 5-Jan-2021.

Hu, Y., De Laat, C., Zhao, Z., et al., 2019. Multi-objective container deployment on heterogeneous clusters. In: Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.(CCGRID), pp. 592–599.

Hu, Y., Zhou, H., de Laat, C., Zhao, Z., 2020. Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Fut. Gen. Comput. Syst. 102, 562–573.

Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y., 2019. Metaheuristic research: a comprehensive survey. Artif. Intel. Rev. 52 (4), 2191–2233.

Imdoukh, M., Ahmad, I., Alfailakawi, M., 2019. Optimizing scheduling decisions of container management tool using many-objective genetic algorithm. Concurr. Comput.: Pract. Exp. 32, (5) e5536.

Jaiswal, K., Sobhanayak, S., Turuk, A.K., Bibhudatta, S.L., Mohanta, B.K., Jena, D., 2018. An iot-cloud based smart healthcare monitoring system using container based virtual environment in edge device. In: 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR). IEEE, pp. 1–7.

James, A., Schien, D., 2019. A low carbon kubernetes scheduler. Proceedings of the 6th International Conference on ICT for Sustainability (ICT4S 2019), 1–10.

Jamshidi, P., Pahl, C., Mendonça, N.C., Lewis, J., Tilkov, S., 2018. Microservices: The journey so far and challenges ahead. IEEE Soft. 35 (3), 24–35.

Jiang, F., Ferriter, K., Castillo, C., 2019. PIVOT: Cost-Aware Scheduling of Data-Intensive Applications in a Cloud-Agnostic System. Tech. rep, Chapel Hill, North Carolina.

Kaewkasi, C., Chuenmuneewong, K., 2017. Improvement of container scheduling for docker using ant colony optimization. In: 2017 9th international conference on knowledge and smart technology (KST). IEEE, pp. 254–259.

Kaur, K., Garg, S., Kaddoum, G., Ahmed, S.H., Atiquzzaman, M., 2020. Keids: Kubernetes based energy and interference driven scheduler for industrial iot in edge-cloud ecosystem. IEEE IoT J. 7 (5), 4228–4237.
Khan, A.A., Zakarya, M., Buyya, R., Khan, R., Khan, M., Rana, O., 2019. An energy and performance aware consolidation technique for containerized datacenters. IEEE Trans. Cloud Comput., 1–18
Kuchnik, M., Park, J.W., Cranor, C., Moore, E., DeBardeleben, N., Amvrosiadis, G., 2019. This is why ml-driven cluster scheduling remains widely impractical. Tech. rep.
Kumar, M., Sharma, S., Goel, A., Singh, S., 2019. A comprehensive survey for scheduling techniques in cloud computing. J. Netw. Comput. Appl. 143, 1–33.
Kumar, N., Aujla, G.S., Garg, S., Kaur, K., Ranjan, R., Garg, S.K., 2018. Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. IEEE Trans. Ind. Inf. 15 (5), 2947–2957.
Li, L., Chen, J., Yan, W., 2018. A particle swarm optimization-based container scheduling algorithm of docker platform. In: Proceedings of the 4th International Conference on Communication and Information Processing, pp. 12–17.
Li, W., Lemieux, Y., Gao, J., Zhao, Z., Han, Y., 2019. Service mesh: Challenges, state of the art, and future research opportunities. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, pp. 122–1225.
Li, X., Garraghan, P., Jiang, X., Wu, Z., Xu, J., 2017. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. IEEE Trans. Paral. Distrib. Syst. 29 (6), 1317–1331.
Lin, M., Xi, J., Bai, W., Wu, J., 2019. Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud. IEEE Access 7, 83088–83100.
Liu, B., Li, J., Lin, W., Bai, W., Li, P., Gao, Q., 2020. K-pso: An improved pso-based container scheduling algorithm for big data applications. Int. J. Netw. Manage. e2092.
Liu, B., Li, P., Lin, W., Shu, N., Li, Y., Chang, V., 2018. A new container scheduling algorithm based on multi-objective optimization. Soft Comput. 22 (23), 7741–7752.
Liu, J., Wang, S., Zhou, A., Xu, J., Yang, F., 2020. Sla-driven container consolidation with usage prediction for green cloud computing. Front. Comput. Sci. 14 (1), 42–52.
Lu, W., Li, B., Wu, B., 2019. Overhead aware task scheduling for cloud container services. In: 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, pp. 380–385.
Luo, J., Yin, L., Hu, J., Wang, C., Liu, X., Fan, X., Luo, H., 2019. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot. Fut. Gen. Comput. Syst. 97, 50–60.
Lv, J., Wei, M., Yu, Y., 2019. A container scheduling strategy based on machine learning in microservice architecture. In: 2019 IEEE International Conference on Services Computing (SCC). IEEE, pp. 65–71.
Lv, L., Zhang, Y., Li, Y., Xu, K., Wang, D., Wang, W., Li, M., Cao, X., Liang, Q., 2019. Communication-aware container placement and reassignment in large-scale internet data centers. IEEE J. Select. Areas Commun. 37 (3), 540–555.
Ma, B., Ni, H., Zhu, X., Zhao, R., 2019. A comprehensive improved salp swarm algorithm on redundant container deployment problem. IEEE Access 7, 136452–136470.
Maenhaut, P.-J., Volckaert, B., Ongenae, V., De Turck, F., 2019. Resource management in a containerized cloud: Status and challenges. J. Netw. Syst. Manage. 28, 1–50.
Mao, H., Schwarzkopf, M., Venkatakrishnan, S.B., Meng, Z., Alizadeh, M., 2019. Learning scheduling algorithms for data processing clusters. In: Proceedings of the ACM Special Interest Group on Data Communication, pp. 270–288.
Mao, Y., Oak, J., Pompili, A., Beer, D., Han, T., Hu, P., 2017. Draps: Dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster. In: 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC). IEEE, pp. 1–8.
Mavridis, I., Karatza, H., 2019. Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. Fut. Gen. Comput. Syst. 94, 674–696.
Mehta, H., Harvey, P., Rana, O., Buyya, R., Varghese, B., 2020. Wattsapp: Power-aware container scheduling.
Mendes, S., Simão, J., Veiga, L., 2019. Oversubscribing micro-clouds with energy-aware containers scheduling. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 130–137.
Menouer, T., 2020. KCSS: Kubernetes container scheduling strategy. J. Supercomput., 1–27
Menouer, T., Darmon, P., 2019. Containers scheduling consolidation approach for cloud computing. In: International Symposium on Pervasive Systems, Algorithms and Networks. Springer, pp. 178–192.
Menouer, T., Darmon, P., 2019. New scheduling strategy based on multi-criteria decision algorithm. In: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE, pp. 101–107.
Morabito, R., Farris, I., Iera, A., Taleb, T., 2017. Evaluating performance of containerized iot services for clustered devices at the network edge. IEEE IoT J. 4 (4), 1019–1030.
Nanda, S., Hacker, T.J., 2018. Racc: resource-aware container consolidation using a deep learning approach. In: Proceedings of the First Workshop on Machine Learning for Computing Systems, pp. 1–5.
Nath, S.B., Addya, S.K., Chakraborty, S., Ghosh, S.K., 2020. Green containerized service consolidation in cloud. In: ICC 2020–2020 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.
Pahl, C., Brogi, A., Soldani, J., Jamshidi, P., 2017. Cloud container technologies: a state-of-the-art review. IEEE Trans. Cloud Comput. 7 (3), 677–692.
Pongsakorn, U., Watashiba, Y., Ichikawa, K., Date, S., Iida, H., et al., 2017. Container rebalancing: Towards proactive linux containers placement optimization in a data center. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). Vol. 1. IEEE, pp. 788–795.
Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.-L., Chen, S.-C., Iyengar, S., 2018. A survey on deep learning: Algorithms, techniques, and applications. ACM Comput. Surv. (CSUR) 51 (5), 1–36.
Rausch, T., Rashed, A., Dustdar, S., 2021. Optimized container scheduling for data-intensive serverless edge computing. Fut. Gen. Comput. Syst. 114, 259–271.
Rodrigues, L.R., Koslovski, G.P., Pasin, M., Pillon, M.A., Junior, O.C.A., Miers, C.C., 2020. Time-constrained and network-aware containers scheduling in gpu era. Fut. Gen. Comput. Syst. 117, 72–86.
Rodrigues, L.R., Pasin, M., Alves Jr, O.C., Miers, C.C., Pillon, M.A., Felber, P., Koslovski, G.P., 2019. Network-aware container scheduling in multi-tenant data center.
Rodriguez, M.A., Buyya, R., 2019. Container-based cluster orchestration systems: A taxonomy and future directions. Soft.: Pract. Exp. 49 (5), 698–719.
Santo, W., Junior, R.M., Ribeiro, A., Silva, D., Santos, R., 2019. Systematic mapping on orchestration of container-based applications in fog computing. In: 2019 15th International Conference on Network and Service Management (CNSM). IEEE Computer Society, Los Alamitos, CA, USA, pp. 1–7.
Shi, T., Ma, H., Chen, G., 2018. Energy-aware container consolidation based on pso in cloud data centers. In: 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 1–8.
Sierksma, G., Zwols, Y., 2015. Linear and integer optimization: theory and practice. Chapman and Hall/CRC.
Song, S., Deng, L., Gong, J., Luo, H., 2018. Gaia scheduler: A kubernetes-based scheduler framework. In: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE, pp. 252–259.
Sung, T.T., Chockalingam, V., Yahja, A., Ryu, B., 2019. Neural heterogeneous scheduler.
Suo, K., Zhao, Y., Chen, W., Rao, J., 2018. An analysis and empirical study of container networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, pp. 189–197.
Tan, B., Ma, H., Mei, Y., 2019. A hybrid genetic programming hyper-heuristic approach for online two-level resource allocation in container-based clouds. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 2681–2688.
Tan, B., Ma, H., Mei, Y., Zhang, M., 2020. A cooperative coevolution genetic programming hyper-heuristic approach for on-line resource allocation in container-based clouds. IEEE Trans. Cloud Comput.
Tao, Y., Wang, X., Xu, X., Chen, Y., 2017. Dynamic resource allocation algorithm for container-based service computing. In: 2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS). IEEE, pp. 61–67.
Townend, P., Clement, S., Burdett, D., Yang, R., Shaw, J., Slater, B., Xu, J., 2019. Improving data center efficiency through holistic scheduling in kubernetes. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, pp. 156–15610.
Varghese, B., Buyya, R., 2018. Next generation cloud computing: New trends and research directions. Fut. Gen. Comput. Syst. 79, 849–861.
Vaucher, S., Pires, R., Felber, P., Pasin, M., Schiavoni, V., Fetzer, C., 2018. Sgx-aware container orchestration for heterogeneous clusters. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, pp. 730–741.
Vhatkar, K.N., Bhole, G.P., 2019. Optimal container resource allocation in cloud architecture: A new hybrid model. Journal of King Saud University-Computer and Information Sciences.
Vhatkar, K.N., Bhole, G.P., 2020. Improved rider optimization for optimal container resource allocation in cloud with security assurance. Int. J. Pervasive Comput. Commun. 16, 235–258.
Wan, X., Guan, X., Wang, T., Bai, G., Choi, B.-Y., 2018. Application deployment using microservice and docker containers: Framework and optimization. J. Network Comput. Appl. 119, 97–109.
Wang, Y., Bao, Q., 2019. Adapting a container infrastructure for autonomous vehicle development.
Watada, J., Roy, A., Kadikar, R., Pham, H., Xu, B., 2019. Emerging trends, techniques and open issues of containerization: A review. IEEE Access 7, 152443–152472.
Wei-guo, Z., Xi-lin, M., Jin-zhong, Z., 2018. Research on kubernetes' resource scheduling scheme. In: Proceedings of the 8th International Conference on Communication and Network Security, pp. 144–148.
Wu, Y., Chen, H., 2017. Abp scheduler: Speeding up service spread in docker swarm. In: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). IEEE, pp. 691–698.
Xu, M., Buyya, R., 2019. Brownoutcon: A software system based on brownout and containers for energy-efficient cloud computing. J. Syst. Soft. 155, 91–103.

Xu, M., Toosi, A.N., Buyya, R., 2018. Ibrownout: an integrated approach for managing energy and brownout in container-based clouds. IEEE Trans. Sustainable Comput. 4 (1), 53–66.

Yang, R., Ouyang, X., Chen, Y., Townend, P., Xu, J., 2018. Intelligent resource scheduling at scale: a machine learning perspective. In: 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). IEEE, pp. 132–141.

Zhang, D., Yan, B.-H., Feng, Z., Zhang, C., Wang, Y.-X., 2017. Container oriented job scheduling using linear programming model. In: 2017 3rd International Conference on Information Management (ICIM). IEEE, pp. 174–180.

Zhang, R., Chen, Y., Dong, B., Tian, F., Zheng, Q., 2019. A genetic algorithm-based energy-efficient container placement strategy in caas. IEEE Access 7, 121360–121373.

Zhang, W., Liu, Y., Wang, L., Li, Z., Goh, R.S.M., 2018. Cost-efficient and latency-aware workflow scheduling policy for container-based systems. In: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, pp. 763–770.

Zhou, R., Li, Z., Wu, C., 2018. Scheduling frameworks for cloud container services. IEEE/ACM Trans. Netw. 26 (1), 436–450.