processes

MDPI

Review

Evolving Container to Unikernel for Edge Computing and Applications in Process Industry

Shichao Chen [1,2] and Mengchu Zhou [1,3,4,*]

[1] The Institute of Systems Engineering and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Macau 999078, China; shichao.chen@ia.ac.cn
[2] The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[3] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA
[4] Center of Research Excellence in Renewable Energy and Power Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: zhou@njit.edu

Abstract: Industry 4.0 promotes manufacturing and process industry towards digitalization and intellectualization. Edge computing can provide delay-sensitive services in industrial processes to realize intelligent production. Lightweight virtualization technology is one of the key elements of edge computing, which can implement resource management, orchestration, and isolation services without considering heterogenous hardware. It has revolutionized software development and deployment. The scope of this review paper is to present an in-depth analysis of two such technologies, Container and Unikernel, for edge computing. We discuss and compare their applicability in terms of migration, security, and orchestration for edge computing and industrial applications. We describe their performance indexes, evaluation methods and related findings. We then discuss their applications in industrial processes. To promote further research, we present some open issues and challenges to serve as a road map for both researchers and practitioners in the areas of Industry 4.0, industrial process automation, and advanced computing.

Keywords: big data analytics; lightweight virtualization; cloud computing; edge computing; industrial process; Industry 4.0; Internet of things; machine learning; process industry; fault diagnosis

Citation: Chen, S.; Zhou, M. Evolving Container to Unikernel for Edge Computing and Applications in Process Industry. *Processes* **2021**, *9*, 351. https://doi.org/10.3390/pr9020351

Academic Editor: Luis Puigjaner

Received: 2 February 2021
Accepted: 10 February 2021
Published: 14 February 2021

1. Introduction

Industry 4.0 represents a new industrial revolution, enabling suppliers and manufacturers to leverage new technologies, i.e., Internet of Things (IoT), Big Data analytics, Edge Computing, Cloud Computing, and Cyber-Physical Systems to improve various processes ranging from wafer fabrication and electronic manufacturing to oil refinery and pharmaceutical production [1]. It promotes the development of manufacturing towards informatization, digitalization, and intellectualization. Edge computing and cloud computing play an important role in realizing the vision that industry 4.0 promises. In particular, edge computing can handle the data locally and provide delay-sensitive services. Cloud computing can deal with large-scale aggregated data, e.g., data mining, training of deep learning models, in different applications of industrial processes. Virtualization technologies are key elements of edge computing and cloud computing.

Virtualization technologies have been in use for years. It makes large expensive mainframes of computing easily shared among different user applications. It can enable users to run multiple operating systems on a single physical server. In this physical server, each operating system runs as a self-contained computer [2]. Virtualization is becoming increasingly important in different scenarios (e.g., computing, storage, and networking). It can improve system efficiency, reliability, and availability, reduce cost, and provide great

flexibility to users. In order to be shared over diverse applications, the virtualization of Information Technology (IT) infrastructure enables the consolidation and pooling of IT resources. It abstracts physical computing resources logically, producing a computing environment that is not limited by the configuration and architecture of physical hardware [3]. It is the creation and orchestration of small virtual computational chunks in the form of an abstract computing platform. Virtualization technologies are widely used in cloud computing [4,5], which can offer an efficient method to harness the cloud power by fragmenting a cloud physical host into small manageable virtual portions [6]. They make cloud computing services simple, convenient, and cost-effective. Hypervisor (e.g., VMware and VirtualBox [7]) has been widely used in hardware virtualization of cloud computing. However, there are some problems, such as high resource overhead [8], long start-up time [9], and large attack surface [10,11]. To overcome its disadvantages, lightweight virtualization technology (e.g., Container and Unikernel) with fast deployment and high efficiency is now applied to cloud computing and edge computing [12,13]. Docker container [14] is gaining great attraction in the IT community, since it allows users to deploy applications in most environments faster and more efficiently than using virtual machines (VMs). Container can use only one kernel for multiple isolated environments or operation systems. Container-based application virtualization is viewed as an appropriate isolation solution with less overhead than VMs. Container has several advantages, e.g., rapid development, portability across different machines, and simplified maintenance [14]. They solve the problems of traditional VMs. As a result of their ease-of-use and performance enhancements, such containers as Docker [15], OpenVZ [16], and Linux Container (LXC) [17], are being widely adopted in industry, academia, and other scientific communities. Undoubtedly, Container-based virtualization delivers a lightweight and efficient environment, but raises some security concerns as it allows an isolated process to utilize an underlying host kernel [18]. Moreover, Docker container is not suitable for IoT applications with frequent interaction of small data and resource-constrained IoT devices [19].

In order to solve the problems of VMs and the low security of Container in the applications, Madhavapeddy et al. propose a lightweight virtualization technology called Unikernel [20]. It has high level security, simplified architecture, and high efficiency. In addition to its container features, it can take full advantage of the resource management and isolation techniques of Hypervisor to provide high-level security. It can also be deployed directly on bare metal hardware without any system dependencies, which is beneficial to the application of an edge computing paradigm in IoT scenarios. Hence it promises to be a virtualization technology beyond containers. Edge computing is an extension of cloud computing at the edge network [21], and it promotes the IoT development. Lightweight virtualization technology is a key to facilitating the realization of edge computing. This paper focuses on the research and applications of lightweight virtualization technology, Container and Unikernel, in edge computing. In Section 2, the research and applications of Container for edge computing are summarized. The applicability of Unikernel for edge computing are illustrated and the comparison between Container and Unikernel is depicted in Section 3. We describe the evaluation metrics and results of lightweight virtualization technologies in Section 4. The applications of Container and Unikernel to industrial processes are discussed in Section 5. Open issues related to lightweight virtualization technologies are analyzed in Section 6. The conclusion of this review paper is concluded in Section 7.

2. Container for Edge Computing

Container-based virtualization can be considered as one of the lightweight alternatives to Hypervisor-based virtualization. Traditional VMs has been applied for a decade in cloud computing with resource virtualization and isolation. VMs are based on Hypervisor, which operates at the hardware level and supports standalone VMs. In each VM instance, a full operating system (OS) is installed on top of the virtualized hardware. Thus, the image files of based on VMs are large and its overhead is non-negligible.

Container avoids the virtualization of hardware and drivers [22]. It implements the virtualization at the OS level. It shares the same OS kernel with the host machine, making it possible to isolate standalone applications that own independent virtual network interfaces, independent process space, and separate file systems. The shared kernel feature allows Container to run a higher density of virtualized instances with small image volume on a single machine. Docker Container is popular and has achieved much more practical use recently, which is a high-level platform. It introduces a container engine, which allows easily one to build, run, manage, and remove containerized applications. It has been widely used for deployment, live migration, orchestration, and isolation of applications in edge computing. A large number of container applications are managed by different orchestration tools and cluster managers such Google Borg, Docker Swarm Manager, and Kubernetes [23]. To realize the resource management of edge nodes with relatively low computing power, Park et al. [24] propose a method of dynamic container layer replacement for a serverless architecture-based Function-as-a-Service, considering a resource-limited environment on edge nodes. Its experimental results show that it can improve boot-up latency by using their proposed method, and provide faster service than container creation. The boot-up latency of the proposed method is lower than that required to create the container. The smaller the size of the dynamic container, the much lower the boot-up latency. Mendki [25] uses Docker container-based analytics services to process the data locally in edge computing. Their feasibility is verified by setting up a deep learning framework on Raspberry Pi for real-time analysis of surveillance video. Its performance benchmarking shows that its overhead is negligible in terms of central processing unit (CPU) processing compared with the bare metal deployment. Deploying the analytics solution in Docker container can provide ease of service management and orchestration for edge nodes. Anand et al. [26] use Docker container to deploy a practical, edge analytics framework in resources-constrained heterogenous environments. It provides an agnostic logical abstraction layer residing over existing hardware and software layers enabling ease of orchestration. Through the framework and use case, it demonstrates how to employ an edge analytics framework that integrates existing systems agnostically and seamlessly. To solve the problems of live migration for offloading services in mobile edge computing environment, Ma et al. [27] propose an edge computing platform architecture, which uses Docker container to support seamless migration of offloading services. In contrast to the state-of-the-art service handoff method in edge computing, the system yields 80 percent (56 percent) reduction in handoff time under 5 Mbps (20 Mbps) network bandwidth conditions. In edge computing, virtualized resources can support and enhance service provisioning. However, migration of edge-enabled services poses significant challenges in the edge computing environment. Bellavista et al. [28] propose an edge computing platform architecture that supports service migration through Docker Container among heterogeneous edge devices. Their experimental results confirm that proactive migration can significantly minimize the service downtime in the case of layered services, by imposing a very limited overhead on the overall support infrastructure. Other studies [29–31] use Container for live migration in a mobile edge computing environment, which can reduce the service downtime to ensure the quality of services (QoS) for users. In terms of security concerns in edge computing, Maurantonio et al. [32] discuss the security of Container in different application scenarios, e.g., Augmented Reality, Smart Home, Smart Cities, E-health, and smart factories. Container can leverage the flexibility given by the additional layers between application images and hardware to provide seamless patching, and ease the need for updates. It is less vulnerable to be attacked than Real Time Operating System (RTOS). Soltesz et al. [33] provide insights into resource, security and isolation for avoiding crosstalk unwanted snooping and fault propagation between containerized systems, although container usage for provisioning security isolation may not seem favorable [34,35]. Table 1 summarizes the studies of Container for different functions.

Table 1. Studies of Container for different functions.

Reference	Migration	Orchestration	Security Isolation	Summary of Findings
Park et al., 2019 [24]		√		① Realizing the resource management of edge nodes; ② Performing dynamic container layer replacement; ③ Improving boot-up latency; ④ Providing faster services than container creation.
Mendiki, 2019 [25]		√		① Providing analytics services based on Docker in edge computing; ② Verifying a deep learning framework on Raspberry Pi; ③ Docker' overhead being negligible.
Anand et al., 2017 [26]		√		① Deploying an edge analytics based on Docker; ② Easing the services orchestration.
Ma et al., 2019 [27]	√			① Proposing an edge computing platform based on Docker to support seamless migration of services; ② Reducing handoff time.
Bellavista et al., 2019 [28]	√			① Proposing an edge computing platform based on Docker for proactive migration; ② Reducing the service downtime.
Elgazar and Harras, 2019 [29]; Maheshwari et al., 2018 [30]; Dupont et al., 2017 [31]	√			① Being suitable for live migration in mobile edge computing; ② Reducing service downtime.
Caprolu et al., 2019 [32]			√	① Comparing the security among Container, Unikernel, and RTOS; ② Being less vulnerable to security attacks than RTOS.
Soltesz et al., 2007 [33]			√	① Achieving security and isolation between containerized systems.
Bernstein, 2014 [34]			√	① Comparing security between Container and VMs; ② Presenting a cluster manager for Docker Container.
Combe et al.,2016 [35]			√	① Being more flexible than VMs; ② Being vulnerable to security attacks.

According to [27,28], resources, e.g., computing, storage, and networking ones, can be virtualized by Container without regard to their heterogeneousness. Container is running on OS and their images occupy some memory, and edge devices in IoT edge computing have no OS, and are resource-constrained. They are suitable for edge computing but not for IoT edge computing. Container can be utilized for image deployment, resource management, and orchestration services, which only imposes little time to the systems. In addition, Container directly shares the kernel with their host machines. They occupy fewer resources and have lower virtualization overhead than VMs. Container-virtualization technologies used in edge nodes with relatively rich resources produces an almost negligible impact for edge computing systems' overhead. In terms of security isolation, Container is able to protect Container-specific information from unwanted leakage to some extent. However, Container-based applications share the same system core, which challenges system security.

In addition, Container has been applied in edge computing platforms. Next, we present a review of the work concerning the combination of an edge computing platform and Container. ParaDrop is a research project in Wisconsin Wireless and NetworkinG Systems (WiNGS) laboratory at the University of Wisconsin-Madison (Madison, WI, USA) [36]. It is suitable for IoT applications and uses Container (Docker) to isolate the operating environment of different applications. A single edge server can run multiple tenant applications. All applications on the gateway are deployed and revoked by a cloud server. EdgeX Foundry is founded by the Linux Foundation to create an interoperable, plug and play, and modular IoT edge computing ecosystem. It is a standardized microservice framework focusing on IoT applications, and its design meets the independence of hardware and OS. All microservice applications in EdgeX Foundry can run in various operating systems in the form of Container [37]. FocusStack [38] is developed to support the deployment of complex applications to IoT devices. Container on edge devices supports its OpenStack services, including virtual network access and application-based granularity configuration. CloudPath [39] is an edge computing system to support the on-demand allocation and dynamic deployment of a multi-level architecture. Its PathExecute module has a container architecture and supports lightweight application functions. AirBox is a secure, lightweight system with scalable edge functions. Its edge functions are deployed through system-level containers [40]. Central office Re-architected as a Datacenter (CORD) is an open-source project for network operators. It can reconstruct the existing network edge integration implementation by using a software-defined-network (SDN), network function virtualization (NFV), and cloud computing technology. OpenStack in CORD is used to manage computing and storage resources, create and configure VMs, and provide an Infrastructure-as-a-Service (IaaS) function. Docker as a Container engine uses Container technology to instantiate services provided to users [41]. AKraino Edge Stack is an open-source project for high-performance edge services, and provides an overall solution for edge infrastructure. It includes an application, middle, and infrastructure layers. The application layer is dedicated to creating an ecosystem of virtual network function (VNF) to promote the development of edge applications [42]. Azure IoT Edge is a fully hosted service built on the Azure IoT center launched by Microsoft. Its IoT Edge modules run as Docker, which can deploy Azure services, third-party services or custom code to IoT Edge nodes, which are locally executed at the nodes [43]. OpenEdge [44] is an open-source edge computing system developed by Baidu. It adopts modular and containerized design. KubeEdge [45] is an open-source edge computing system that relies on container arrangement and scheduling capabilities based on kubernetes to achieve cloud-edge collaboration.

After introducing the existing virtualization techniques of these edge computing systems, we can conclude that Container, especially Docker, are widely used in edge computing systems due to their rapid deployment and resource management services. Yet some edge computing systems adopt the virtualization mode of combining VMs and Container to manage the hardware resources and application services. Table 2 shows the illustration of virtualization technologies used in an edge computing system.

Table 2. Illustration of Container used in edge computing platforms.

Platform	Virtualization Technique	Application Scenarios
ParaDrop [36]	Container	IoT
EdgeX Foundry [37]	Container	IoT
FocusStack [38]	Container	IoT
CloudPath [39]	Container	Mobile
AirBox [40]	Container	IoT
CORD [41]	VM and Container	No Limit
AKraino Edge Stack [42]	VM and Container	No Limit
Azure IoT Edge [43]	Container	No Limit
OpenEdge [44]	Container	No Limit
KubeEdge [45]	Container	No Limit

Container can be utilized in edge nodes with relatively sufficient resources, e.g., edge servers. It is not suitable for edge nodes without OS and enough resources, especially for edge devices in IoT edge computing environment. In addition, security is another vulnerability of Container. We demand other lightweight virtualization technologies for edge computing. Unikernel to be discussed next, stands out as such lightweight virtualization technology.

3. Unikernel for Edge Computing

Unikernel [46] is a single-purpose appliance that is specialized at compile time into standalone kernel and sealed against modification after deployment. Additionally, it provides increased security through a reduced attack surface and better performance by reducing unnecessary components from the applications. It was designed initially for cloud computing, but its small footprint and flexibility make it suitable for edge computing, especially upcoming IoT edge computing. The attack surface of Unikernel is strictly confined to the application embedded within. It does not include a uniform operating system layer, and everything is directly compiled into the application layer. Therefore, each Unikernel may have a different set of vulnerabilities, which implies that an attacker that can penetrate one may not threaten to others. In addition, Unikernel is principally designed to be stateless. Therefore, edge intelligent algorithms (e.g., compression, encryption, and NFV) can be executed easily with it.

There are many research projects about Unikernel, mainly including MirageOS Unikernel [47], IncludeOS [48], OSv Unikernel [49], ClickOS [50], and others [51–57]. Table 3 summarizes their characteristics, in terms of programming languages, supporting platforms, characteristics, and application scenarios.

Due to its small image file size and high security, Unikernel has been under active research and development since its inception in 2013, especially for edge computing. Expending it from cloud computing to edge computing, researchers focus on the issues related to migration, orchestration, network, and isolation for edge computing. To enable service migration in mobile edge environment, Ramirez et al. [58] develop a practical framework for service management in vehicular networks. Docker and Unikernel are used as the migration techniques for the migration of a Network Memory Server. Experimental results show that the average migration time with Unikernel is less than one with Docker, and Unikernel can support new applications and services in highly mobile environment. To provide reliable network storage in highly mobile environments, Ezenwigbo et al. [59] explore how services can be migrated as users travel around. They use migration techniques, e.g., Docker and Unikernl, to implement the migration of a simple Network Memory Server. Their results show that the migration time based on Unikernel is less than other virtualization technologies in proactive and reactive service migration scenarios.

In [60], a fog-enabled cellular vehicle-to-everything architecture is proposed, which provides resources at core, edge and vehicle layers. This architecture enables the connection of VMs, Container and Unikernel to form an Application-as-a-Service function chain, which can efficiently manage and orchestrate all the underlying physical resources. In a cellular Vehicle-to-everything (C-V2X) use case, the live migration and scaling functionalities are evaluated, and the experimental results demonstrate that the proposed scheme maximizes the accepted requests, without violating the applications' service level agreement. To support the composition and deployment of machine learning-based data analytics in IoT devices, Zhao et al. [61] design a Zoo system to address these challenges. MirageOS, a Unikernel technology, is utilized for the model deployment. Deploying Unikernel is proved to be of low memory footprint, and thus quite suitable for resource-constrained IoT devices.

Table 3. Summary of different Unikernel based products.

Unikernel	Programming Languages	Supporting Platforms	Characteristics	Application Scenarios
MirageOS [47]	OCaml	Xen, FreeBSD, POSIX	Supporting secure and high performance network services.	Cloud computing, edge computing
HalVM [51]	Haskell	Xen	Implementing advanced lightweight VM on Xen by developers.	Cloud computing, edge computing
LING [52]	Erlang	Xen	Supporting for concurrency, distribution, and fault tolerance; High security and avoiding most attacks.	Cloud computing
Clive [53]	Go	Xen, KVM	Being designed for distributed and cloud computing environments.	Cloud computing, edge computing
ClickOS [50]	C++	Xen	Building a multifunctional and high performance software middleware platform.	Network function virtualization
IncludeOS [48]	C++	KVM/VirtualBox	Supporting for full virtualization.	Cloud computing, edge computing
Drawbridge [54]	C	Windows "picoprocess"	Combining picoprocess and library operating system to improve the performance of applications and isolation security.	Desktop applications on Windows
Runtime.js [55]	Javascript	Xen, KVM	Realizing the management of low-level CPU and memory; Running JavaScript using an embedded V8 engine.	Cloud computing
OSv [49]	Java, C, C++, Node.js	Xen, KVM, VMware, VirtualBox	Supporting a variety of programming language; Being compatible with existing Java programs; Being supported by multiple hypervisors.	Cloud computing, edge computing
Rumprun [56]	C, C++, Erlang, Go, Java, Javascript, PHP, Bsash, Pythons	Xen, KVM, bare metal	Supporting from bare metal ARM hardware to hypervisor; Supporting applications written in various languages.	Cloud computing, edge computing
HermitCore [57]	C, C ++, Fortran, Go	Xen, KVM, x86_64	Supporting to expand into multi-core processor VM system; Supporting limit scale computing.	Cloud computing

An orchestration framework is proposed to enable edge-cloud collaborative computing for road context assessment [62]. Mirage OS Unikernel is utilized for developing this orchestration platform due to its multiple advantages in terms of isolation, memory footprint and fine-grained function encapsulation. Experimental results illustrate the Unikernel's boot time is substantially lower than Amazon Firecracker microVM's. In addition, it is suitable for processing a small amount of information. To efficiently exploit the resources of constrained edge devices through fine-grained computation offloading, Fine-Grained edge offloading with Unikernels (FADES) is proposed [63]. It takes advantage of MirageOS Unikernel to isolate and embed application logic in concise Xen-bootable images. Its performance is evaluated under various hardware and network conditions. The results show that FADES can effectively strike a balance between running complex applications in the cloud and simple operations at the edge.

Valsamas et al. [64] propose an elastic content distribution platform, which serves the Internet content using tiny Unikernel-based VMs. It provides a dynamic deployment service at the edge. It is demonstrated that the proposed platform is valid. Virtualization technologies are widely used in NFV. In [65], VMs, Container and Unikernel are utilized to deploy virtualized network functions at the network edge. Their performances are evaluated by deploying two services, i.e., Apache and Redis with them. Experimental

results show that Unikernel has a small image size and very small memory consumption. Moreover, Unikernel can eliminate the overhead of context switching, applications with high context switching between user and kernel mode can outperform than other two. Filipe et al. [66] also compare the use of two virtualization technologies, e.g., Container and Unikernel, for virtual network function (VNF) instantiation in edge computing. They develop a failure detection and recovery mechanism to ensure VNF reliability. The experimental results show that the mechanism can ensure near zero downtime. In a resource-scarce isolated environment, multiple virtualization techniques including VMs, Container, Unikernel, and kata-containers are explored to deploy network functions [67]. The performance of NFV virtualization by deploying web services is analyzed. Experimental results show that Unikernel is secure, lightweight and is suitable for running applications requiring many interactions among various smart devices or smart objects. Table 4 summarizes the studies of Unikernel for different functions.

Table 4. Studies of Unikernel for different functions.

Reference	Migration	Orchestration	Network	Security Isolation	Summary of Findings
Ramirez et al., 2020 [58]	√				① Enabling service migration by using Unikernel in mobile edge environment; ② Comparing the average migration time between Docker and Unikernel.
Ezenwigbo et al., 2020 [59]	√				① Using Docker and Unikernel to implement the migration; ② Requiring less migration time based on Unikernel than other virtualization technologies.
Sarrigiannis et al., 2020 [60]	√	√			① Proposing a fog-enabled cellular vehicle-to-everything architecture; ② Using Unikernel to efficiently manage and orchestrate all the physical resources.
Zhao et al., 2018 [61]	√	√			① Designing a Zoo system to implement the deployment of machine learning-based data analytics; ② Using MigrageOS for model deployment.
Cozzolino et al., 2020 [62]		√		√	① Using MigrageOS for services orchestration.
Cozzolino et al., 2017 [63]				√	① Proposing FADES for computation offloading; ② Using MirageOS to isolate images.
Valsams et al., 2018 [64]		√			① Proposing an elastic content distribution platform; ② Using Unikernel-based VMs to serve Internet content.
Behravesh et al., 2019 [65]			√		① Utilizing Unikernel to deploy virtualized network functions; ② Using it to eliminate the overhead of context switching.
Filipe et al., 2019 [66]			√		① Comparing the performance of VNF between Container and Unikernel; ② Developing a failure detection and recovery mechanism.
Aggarwal and Thangaraju,2020 [67]			√		① Comparing the performance of NFV among VMs, Container, and Unikernel. ② Concluding that Unikernel is suitable for running applications demanding interaction among devices.
Caprolu et al., 2019 [32]				√	① Comparing the security among Container, Unikernel, and RTOS; ② Concluding that Unikernel has the high-level security.

According to the above analysis and discussion, we conclude that Unikernel has a smaller image size and very small memory consumption. It can be used for the migration in a mobile edge computing environment, especially Vehicular Networks. It can quickly respond to user requests. Since its image size is small, it can run on the edge devices with highly limited resources. It can also reduce the attack surface that can help guarantee code integrity and ease of updates, and keep high security isolation. Its OS overhead is negligible, and it is suitable for running applications with high context switching, processing a small amount of information.

VMs, Container and Unikernel virtualization technologies, are expected to co-exist for cloud computing, edge computing and IoT edge computing. We can choose an appropriate virtualization technology to meet different requirements. The three virtualization architectures are shown in Figure 1. Table 5 summarizes their main characteristics the comparison among VMs, Container and Unikernel.

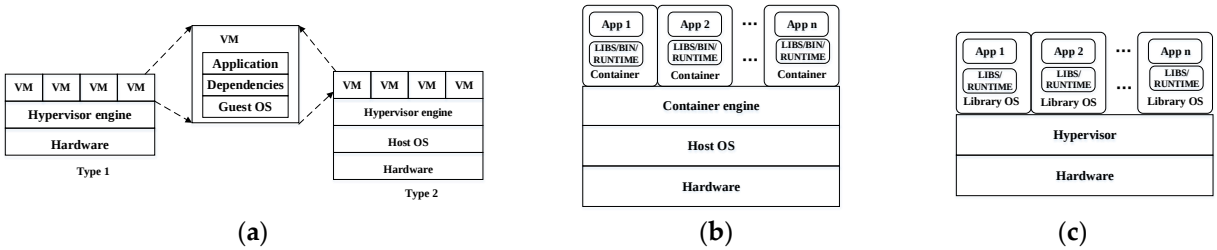

Figure 1. Virtualization Architectures: (**a**) Virtual machines; (**b**) Container; an (**c**) Unikernel.

Table 5. Comparison among VM, Container and Unikernel.

	Virtual Machine • **KVM** • **QEMU**	**Container** • **Docker** • **Open VZ** • **LXC**	**Unikernel** • **MirageOS** • **IncludeOS** • **ClickOS** • **OSv**
Instantiation Time	~5/10 s	~800/100 ms	~<50 ms
Start-Up Time	Slow	Medium	Fast
Image Size	~1000 MBs	~50 MBs	~<5 MBs
Memory Footprint	~100 MBs	~5 MBs	~8 MBs
Programming Language	No	No	Yes
Hardware Portability	High	High	Medium
Security	High	Medium	High
Live Migration Support	Yes	Yes	Requires manual implementation
Application Scenario	Cloud computing	Cloud computing, Edge computing	Cloud computing, Edge computing, IoT edge computing
Multitenancy	Yes	Yes	Yes

4. Evaluation for Lightweight Virtualization Technologies

A number of metrics can be used to evaluate the performance of Lightweight Virtualization technologies, e.g., CPU performance, memory performance, Disk Input/Output (I/O) performance, and Network I/O performance.

Watada et al. [68] present a performance comparison between Container and Unikernel. They use some standard benchmarks called Sysbench (CPU performance); Iperf (for checking network bandwidth) and STREAM (measuring sustained bandwidth of entire cache hierarchy). Their experimental evaluation is done by using HP-Blade server with 64-bit Ubuntu 16:04. They choose tiny OSv and Rumprun VMs on top of Xen and kvm as

unikernels. The CPU performance for lightweight virtualization technologies (e.g., Docker, LXC/LXD, OSv, and Rumprun) are tested by Sysbench. Their experimental results show that CPU performance of Docker is near that of the native system for a single instance, but performance significantly drops down for multiple instances. OSv and Rumprun reveal the worse performance than other containers. Their network performances are evaluated by iperf. They have tested them via two instances (one acting as server and the others as client). Their experimental results show Docker container and OSv are promising. In terms of memory performance, Rumprun offers the better performance than others. In [65], the performance of Container and Unikernel are also evaluated, including image size, memory utilization, CPU utilization, the time serving each request, and transfer rate. Experimental verification is done by using Intel Next Unit of Computing (NUC) device equipped with a Kingston SODIMM DDR4 RAM with 16 GB capacity and Intel(R) Core (TM) i7-7567U CPU with 3.5 GHz clock rate. Ubuntu 18.04.1 LTS is utilized as the host OS (64-bit Ubuntu 16:04) for all the platforms. Docker container engine (version18.06.1-ce) is installed for running containers on the system. The experiment has two instances, i.e., Apache Hyper Text Transport Protocols (HTTP) server and Redis. Experimental results [65] show the image size of Rumprun unikenels for both services are significantly lower than Docker container. This is because unikernels only contain the dependencies required to run the application. Additionally, the memory usage of containers is much less than Rumprun unikernels. The main reason is that containers can have the efficient and dynamic usage of memory. However, Rumprun unikernels have the fixed size memory allocation. In the idle mode, the CPU utilization of the services based on Container and Unikernel is very low. It increases drastically, when more and more service requests arrive and the corresponding services are performed. This is especially true for Rumprun unikernels aas caused by their poor process management. By evaluating the transfer rate of the service, Rumprun performs poorly and has a lower transfer rate than containers.

In another study [67], the performances of Container and Unikernel are evaluated in terms of the image size, boot time, memory utilization and CPU utilization. Its system comprises of a Xen server with DDR4 4 GB RAM and 10 GB storage capacity. Ubuntu 18.04 LTS is used as the host OS for all platforms. Unikernel uses QEMU, and Docker engine (version 18.09.5) is installed for running docker containers. The Apache Benchmarking tool is installed on the host operating system to send requests and analyze their performances. The experimental results and findings of [67] are similar to those of [65].

In summary, Container has satisfactory performances in almost all aspects on servers with rich resources. It provides near real-time and good resource utilization, and its overheads are negligible. Its image size is bigger than Unikernel's, and it is not free from an issue regarding isolation and security. From the perspective of its maturity and performance evaluation, Container is highly suitable for edge servers with sufficient resources in edge computing. Unikernel offers promising features such as significantly reduced memory footprint, fast booting, high-level security, efficient resource utilization, and many more. Unikernel offers important advantages for those cases with many IoT devices and especially fit to IoT edge computing whose nodes have highly limited processing power and storage facilities. It is not suitable for processing the services with large volumes of data. However, to achieve the desired technical maturity, much work remains to be done, including microprocessor stability, process management, and persistent storage.

5. Applications to Industrial Processes

With the development of artificial intelligence, IoT, digital twin [69], and parallel intelligence [70], the manufacturing industry is moving towards the goal of smart manufacturing. A number of edge computing frameworks or applications based on virtualization technologies are deployed to different industrial processes, e.g., semiconductor manufacturing [71], robotic assistance for emergency management [72], explosion prevention in mining industry [73], maintenance management [74,75], Fabric defect detection for textile production [76], oil and gas production [26,77], spectroscopic inspection for olive [78], and

Augmented Reality for shipbuilding [79]. In this section, we focus on illustrating equipment fault diagnosis and computation of scheduling tasks. Additionally, we depict the reasons for adopting a specific lightweight virtualization technology for this application.

5.1. Fault Diagnosis Processes

Fault diagnosis in industry can improve the production efficiency, and reduce equipment maintenance cost. Machine learning (ML) has been applied to fault diagnosis [80,81]. Figure 2 is a data-driven and edge-cloud collaboration-based fault diagnosis system for an industrial process (semiconductor manufacturing). It includes three tiers, i.e., edge devices, edge servers, and cloud data centers layers. The models of fault diagnosis are deployed to edge servers by Container and Unikernel. Thus, it can reduce the delay time of fault detection by edge computing. In this system, models are trained in cloud data centers rich with resources, and the processes of inferring faults are executed in edge servers. Additionally, the unidentified fault data can be transmitted to the cloud data center for updating models, and then updating the corresponding models in edge servers. In edge nodes and servers, we need to install Docker container environment, and then Docker-based model image can be deployed and executed quickly. It greatly facilitates the deployments and execution of applications. Unikernel can also be used to deploy models to edge nodes with limited resources, and it has high security isolation.

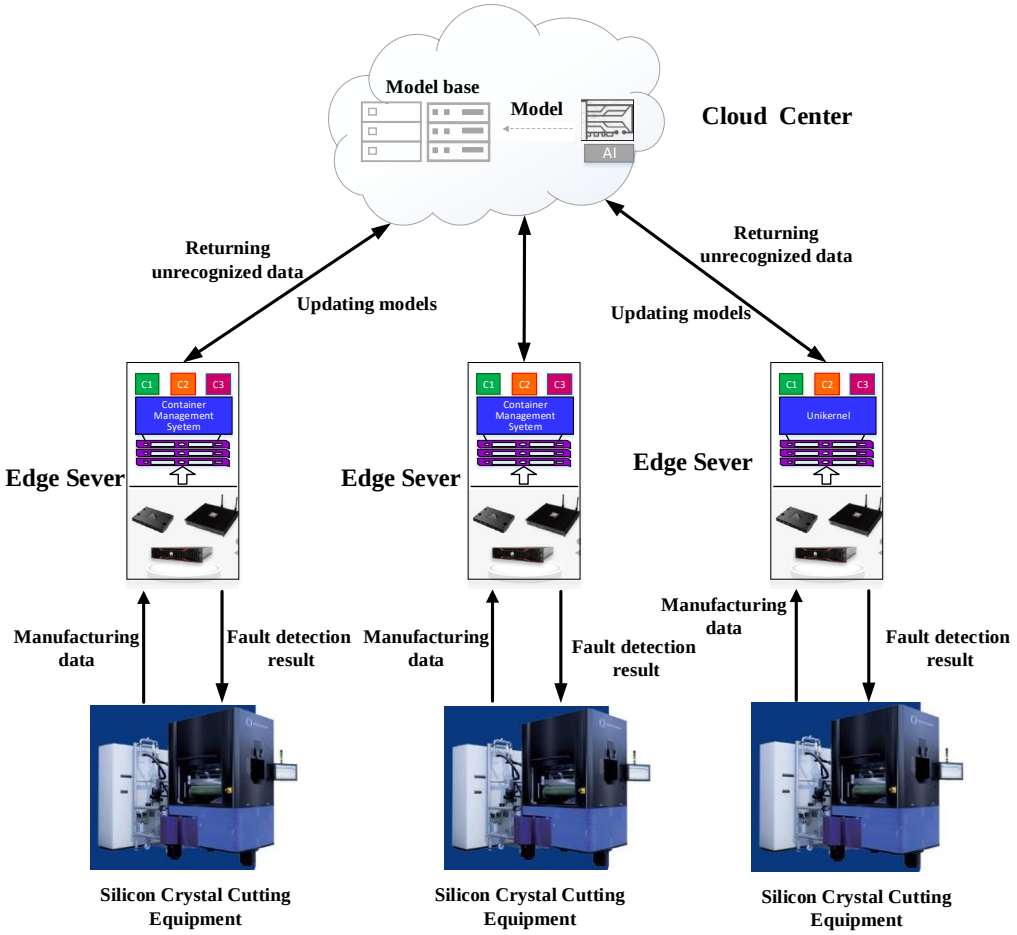

Figure 2. Data-driven and edge-cloud collaboration based a fault diagnosis system.

In our experiment [81], the Tennessee Eastman dataset is used, which contains 52 process variables and 21 process faults. Firstly, we use a dimensionality reduction algorithm called fisher discriminant analysis (FDA) to extract fault features. Then, an ensemble learning method called AdaBoost is utilized for classifying faults. We build the image files

based on Container and Unikernel for fault diagnosis, and then deploy them to Raspberry Pi 3B+. Experimental results show that Unikernel image only occupies 81 MB, thus only 11.86% of Docker's 683 MB.

It only takes 24.003 s to package the Python file into Unikernel image and execute the program, thereby requiring 68.6% of Docker's 35.022 s. Thus, Unikernel has some advantages over Container, especially for edge devices with constrained resources.

5.2. Oil Extraction Process

An oil extraction process in the Oil and Gas industry is a fault-sensitive process. It requires high reliability and extra safety measures to protect the surrounding environment. Thus, efficient and environment-friendly oil extraction is a challenging operation. To overcome these challenges and protect the environment from pollution, one needs to build smart oil fields with many devices (e.g., sensors and actuators) for achieving clean oil and gas extraction. Cloud data center can handle the generated data by devices, but impose high latency, which cannot for detecting oil spill anomalies [82], and analyzing a large amount of data to predict the oil spill spread direction and quantity [83]. Figure 3 is the system architecture of collaborative edge computing for environment-friendly oil extraction, where an edge scheduler-based an edge device in every oil extraction site is demonstrated. The system includes three tiers, i.e., IoT, edge nodes, and cloud data center layers. IoT devices, including physical sensors of smart oil fields, takes physical quantities. Edge nodes are located locally for processing data. Containers satisfy latency-sensitive operational requirements. Instead, edge computing can provide delay-sensitive services, due to its ability to process data locally. Thus, edge computing systems are utilized to each rig of smart oil fields. To overcome the limited resources of single oil rig and rapid deployment of edge computing systems, it is necessary to build a collaborative edge computing platform with nearby oil rigs at the edge, thus sharing computing resources among each other [77].

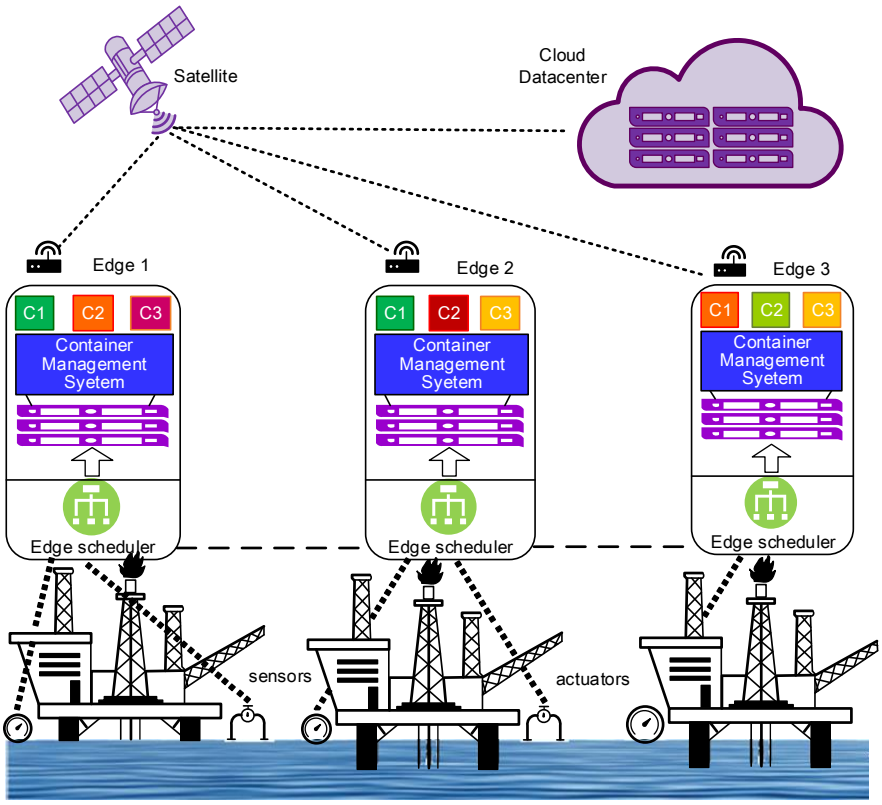

Figure 3. System architecture of collaborative edge computing for environment-friendly oil extraction.

In this scenario, the computation tasks of each rig can offload to nearby edge nodes (rigs). In this system, edge machines own limited storage, computing, and networking resources, which are placed on the platform of oil rig. They process different tasks, e.g., image processing, are used, which makes the migration and deployment of applications easy. To verify the system model, Minimum Expected completion Time (MECT) [84], Success with Computational Certainty (SCC) heuristics, and Highest Probability of Success [77] are adopted to evaluate the resulting system. Experimental results show that they can greatly reduce improve task deadline miss rate.

6. Open Issues and Challenges

In this section, we discuss the technical challenges research issues for Container and Unikernel. To promote the development in industrial applications, we must focus on the following issues of Container to achieve its convenience, faster and easier deployment, and greater elasticity.

(1) Weaker isolation. The existing Container isolation mechanism [68] is much weaker than that of Unikernel. It shares one kernel for multiple isolated environments, thus facing the risk to collapse the entire containerized environment. To solve the security problems of Container, several methods can be explored, including using trusted images, managing container secret, securing the runtime environment, and vulnerability scanning [68].

(2) Lack of tools and support. To realize the large-scale application of containers, the container monitoring and managing tools are needed. However, we have only Container orchestrators, like Kubernetes. More container management orchestrators need to be researched and developed to support the management of different containers

(3) Generalization for all services. Container is suitable for microservices and it does not well support monolithic architecture. For a monolithic architecture, Container only provides simplified a delivery mechanism by offering easy packaging technologies.

(4) Data storage. Container is not suitable for storing permanent data, i.e., data collected for IoT sensors. It is risky to storage significant data on edge nodes due to both the volatile environment of edge nodes and the security risks of containers. Therefore, important data need to be stored in centralized nodes or cloud datacenters and retrieved on demand. This may reduce the feasibility of lightweight virtualization-based edge computing in some highly data-intensive applications. To address the situation, we should improve the *Data Volumes* of Container, which are needed to be implemented in more seamless way.

Although Unikernel has many advantages, e.g., faster booting, small size, and high security, it has the following problems and challenges.

(1) Unikernel's usability. Unikernel does not have a shell and not support online debugging. If Unikernel fails, we can only reboot it. It does not support online upgrades and updates either. If the application and configuration need to be updated, the user needs to recompile the source code to produce a new Unikernel and deploy a new version, which can be very costly and sometimes prohibitive. We can build a mechanism similar to Docker container's to realize the remote deployment, update, and upgrading of Unikernel.

(2) Security. Unikernel's security is guaranteed by the isolation provided by the underlying operating system or Hypervisor, and it is more secure than Container. However, it is just a process in application space, and thus it is vulnerable to various traditional attacks. Process management needs to be improved for promoting Unikernel's security. Blockchain technologies [85] can be considered.

(3) High development cost of Unikernel based on library operating system(LibOS). LibOS is the core technology of Unikernel. When developing it, we should consider not only specific application requirements and programming languages, but also the association and boundary among the underlying operating systems. To solve this

problem, we can build a platform adaption layer, which can resolve the dependencies of LibOS on the underlying host operating systems, and improve its compatibility.

(4) Construction and deployment. There are no mature compilation tools for Unikernel, and there are certain technical barriers to build and deploy Unikernel. It is very inconvenient that different unikernels need to build and generate a matching tool chain, and configure the corresponding development environment. So we can build comprehensive and easy-to-use tools for quickly compiling application into Unikernel, like Unik [86] to facilitate more applications, e.g., [87–97].

7. Conclusions

In this paper, we have summarized lightweight virtualization technologies in edge computing, and compared the characteristics of Container and Unikernel to indicate what edge computing scenarios they fit. According to their performance evaluation results, we have discussed which lightweight virtualization technologies fit to what application scenarios. We have presented their possible applications in some industrial processes in which lightweight virtualization technologies are required. Finally, we have discussed some technical challenges and open issues for future research in this area. We hope that this review article can stimulate more researchers and engineers to apply recent edge computing technologies to their various industrial processes and realize what industry 4.0 promises to bring.

Author Contributions: Conceptualization, S.C. and M.Z.; investigation, S.C.; writing—original draft preparation, S.C.; writing—review and editing, M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Deanship of Scientific Research (DSR) at King Abdulaziz University under grant no. D-422-135-1441.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Acknowledgments: We would like to acknowledge the help of Yue Liu, School of Artificial Intelligence and Automation, Beijing University of Technology, Beijing, China for her help in drawing some figures and text revision. We also appreciate the anonymous reviewers for their constructive comments that help improve this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Petrasch, R.; Hentschke, R. Process modeling for industry 4.0 applications: Towards an industry 4.0 process modeling language and method. In Proceedings of the 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016; pp. 1–5. [CrossRef]
2. Sahoo, J.; Mohapatra, S.; Lath, R. Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. In Proceedings of the 2010 Second International Conference on Computer and Network Technology, Bangkok, Thailand, 23–25 April 2010; pp. 222–226.
3. Elsayed, A.; AbdelBaki, N.; Elsayed, A. Performance evaluation and comparison of the top market virtualization hypervisors. In Proceedings of the 2013 8th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egypt, 26–27 November 2013; pp. 45–50.
4. Sotiriadis, S.; Bessis, N.; Xhafa, F.; Antonopoulos, N.; Antonopoulos, N. Cloud Virtual Machine Scheduling: Modelling the Cloud Virtual Machine Instantiation. In Proceedings of the 2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems, Palermo, Italy, 4–6 July 2012; pp. 233–240.
5. Kapse, P.V.; Dharmik, R.C. An effective approach of creation of virtual machine in cloud computing. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 10–11 February 2017; pp. 145–147.
6. Linlin, W.; Saurabh, G.; Rajkumar, B. SLA-based Admission Control for a Software-as-a-Service Provider in Cloud Computing En-vironments. *J. Comput. Syst. Sci.* **2012**, *78*, 1280–1299.

7. Vojnak, D.T.; Eordevic, B.S.; Timcenko, V.V.; Strbac, S.M. Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation. In Proceedings of the 2019 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 26–27 November 2019; pp. 1–4. [CrossRef]
8. Garg, S.; Dwivedi, R.K.; Chauhan, H. Efficient utilization of virtual machines in cloud computing using Synchronized Throttled Load Balancing. In Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 4–5 September 2015; pp. 77–80.
9. Patel, K.D.; Bhalodia, T.M. An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 6–8 May 2019; pp. 145–150.
10. Nishad, L.S.; Paliwal, J.; Pandey, R.; Beniwal, S.; Kumar, S. Security, privacy issues and challenges in cloud computing: A survey. In *Proceedings of the International Conference on Information and Communication Technology for Competitive Strategies New York*; ACM Press: New York, NY, USA, 2016; p. 47.
11. Grobauer, B.; Walloschek, T.; Stocker, E. Understanding Cloud Computing Vulnerabilities. *IEEE Secur. Priv. Mag.* **2010**, *9*, 50–57. [CrossRef]
12. Linthicum, D.S. Moving to Autonomous and Self-Migrating Containers for Cloud Applications. *IEEE Cloud Comput.* **2016**, *3*, 6–9. [CrossRef]
13. Ahmed, A.; Pierre, G. Docker Container Deployment in Fog Computing Infrastructures. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 1–8.
14. Abdelbaky, M.; Diaz-Montes, J.; Parashar, M.; Unuvar, M.; Steinder, M. Docker Containers across Multiple Clouds and Data Centers. In Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, Cyprus, 7–10 December 2015; pp. 368–371. [CrossRef]
15. Docker—Build, Ship, and Run Any App, Anywhere. Available online: http://www.docker.com/ (accessed on 14 December 2014).
16. OpenVZ Linux Containers Wiki. Available online: http://openvz.org/Main_Page (accessed on 14 December 2014).
17. Linux Containers. Available online: http://linuxcontainers.org (accessed on 11 December 2014).
18. Sarkale, V.V.; Rad, P.; Lee, W. Secure Cloud Container: Runtime Behavior Monitoring Using Most Privileged Container (MPC). In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 351–356.
19. Hajji, W.; Fung, T. Understanding the Performance of Low Power Raspberry Pi Cloud for Big Data. *Electronics* **2016**, *5*, 29. [CrossRef]
20. Madhavapeddy, A.; Mortier, R.; Rotsos, C.; Scott, D.; Singh, B.; Gazagnaire, T.; Steven, S.; Steven, H.; Jon, C. Unikernels: Library operating systems for the cloud. *ACM SIGARCH Comput. Archit. News* **2013**, *48*, 461–472. [CrossRef]
21. Chen, S.; Li, Q.; Zhou, M.; Abusorrah, A. Recent Advances in Collaborative Scheduling of Computing Tasks in an Edge Computing Paradigm. *Sensors* **2021**, *21*, 779. [CrossRef] [PubMed]
22. Felter, W.; Ferreira, A.; Rajamony, R.; Rubio, J. An updated performance comparison of virtual machines and Linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA, 29–31 March 2015; pp. 171–172.
23. Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; Wilkes, J. Large-scale clus-ter management at Google with borg. In Proceedings of the Tenth European Conference on Computer Systems, Bordeaux, France, 21–24 April 2015; p. 18.
24. Park, J.; Kim, Y.; Son, A.-Y.; Lim, Y.; Huh, E.-N. A Method of Dynamic Container Layer Replacement for Faster Service Providing on Resource-Limited Edge Nodes. In Proceedings of the 2019 IEEE 2nd International Conference on Electronics and Communication Engineering (ICECE), Xi'an, China, 9–11 December 2019; pp. 434–437.
25. Pankaj, M. Docker container based analytics at IoT edge Video analytics use case. In Proceedings of the International Conference on Internet of Things: Smart Innovation and Usages, Bhimtal, India, 23–24 February 2018.
26. Anand, N.; Chintalapally, A.; Puri, C.; Tung, T. Practical Edge Analytics: Architectural Approach and Use Cases. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 236–239.
27. Ma, L.; Yi, S.; Carter, N.; Li, Q. Efficient Live Migration of Edge Services Leveraging Container Layered Storage. *IEEE Trans. Mob. Comput.* **2018**, *18*, 2020–2033. [CrossRef]
28. Bellavista, P.; Corradi, A.; Foschini, L.; Scotece, D. Differentiated Service/Data Migration for Edge Services Leveraging Container Characteristics. *IEEE Access* **2019**, *7*, 139746–139758. [CrossRef]
29. Elgazar, A.; Harras, K. Teddybear: Enabling Efficient Seamless Container Migration in User-Owned Edge Platforms. In Proceedings of the 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, Australia, 11–13 December 2019; pp. 70–77.
30. Maheshwari, S.; Choudhury, S.; Seskar, I.; Raychaudhuri, D. Traffic-Aware Dynamic Container Migration for Real-Time Support in Mobile Edge Clouds. In Proceedings of the 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Indore, India, 16–19 December 2018; pp. 1–6.
31. Dupont, C.; Giaffreda, R.; Capra, L. Edge computing in IoT context: Horizontal and vertical Linux container migration. *Glob. Internet Things Summit (GIoTS)* **2017**, 1–4. [CrossRef]

32. Caprolu, M.; Di Pietro, R.; Lombardi, F.; Raponi, S. Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Larnaca, Cyprus, 16 May 2019; pp. 116–123.
33. Soltesz, S.; Pötzl, H.; Fiuczynski, M.E.; Bavier, A.; Peterson, L. Container -based operating system virtualization: A scalable, high performance alternative to hypervisors. In Proceedings of the 2nd ACM SIGOPS/EuroSys european conference on computer systems 2007, Lisbon, Portugal, 20–23 March 2007; Volume 41, pp. 275–287.
34. Bernstein, D. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Comput.* **2014**, *1*, 81–84. [CrossRef]
35. Combe, T.; Martin, A.; Di Pietro, R. To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Comput.* **2016**, *3*, 54–62. [CrossRef]
36. Liu, P.; Willis, D.; Banerjee, S. ParaDrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge. In Proceedings of the 2016 IEEE/ACM Symposium on Edge Computing (SEC), Washington, DC, USA, 27–28 October 2016; pp. 1–13.
37. Brasil. Decreto—Lei n° 227, de 28 de Fevereiro de 1967. Dá nova Redação ao Decreto-lei n° 1.985, de 29 de Janeiro de 1940 (Código de Minas). Brasília. 1967. Available online: http://www.planalto.gov.br/ccivil_03/Decreto-Lei/Del0227.htm (accessed on 19 October 2020).
38. Amento, B.; Balasubramanian, B.; Hall, R.J.; Joshi, K.; Jung, G.; Purdy, K.H. FocusStack: Orchestrating Edge Clouds Using Location-Based Focus of Attention. In Proceedings of the 2016 IEEE/ACM Symposium on Edge Computing (SEC), Washington, DC, USA, 27–28 October 2016; pp. 179–191.
39. Mortazavi, S.H.; Salehe, M.; Gomes, C.S.; Phillips, C.; de Lara, E. Cloudpath: A multi-tier cloud computing frame work. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose, CA, USA, 12–14 October 2017; pp. 1–13.
40. Bhardwaj, K.; Shih, M.-W.; Agarwal, P.; Gavrilovska, A.; Kim, T.; Schwan, K. Fast, Scalable and Secure Onloading of Edge Functions Using AirBox. In Proceedings of the 2016 IEEE/ACM Symposium on Edge Computing (SEC), Washington, DC, USA, 27–28 October 2016; pp. 14–27.
41. Cord. 2018. Available online: https://www.open-networking.org/cord (accessed on 11 September 2020).
42. Akraino Edge Stack. 2018. Available online: https://www.akraino.org (accessed on 25 August 2020).
43. Azure IoT. 2018. Available online: https://azure.micro-soft.com/en-us/overview/iot/ (accessed on 18 July 2020).
44. Baetyl. Baetyl Architecture [EB/OL]. Available online: https://baetyl.io/zh/docs/overview/Baetyl-design (accessed on 9 September 2019).
45. KubeEdge vs. K3S: Kubernetes in Edge Computing. [EB/OL]. Available online: http://www.sohu.com/a/307671703683048 (accessed on 13 April 2019).
46. Madhavapeddy, A.; Scott, D.J. Unikernels: Rise of the Virtual Library Operating System. *Queue* **2013**, *11*, 30–44. [CrossRef]
47. Unikernel.org. Unikernels-Rethinking Cloud Infrastructure [EB/OL]. Available online: http://Unikernel.org/ (accessed on 20 January 2018).
48. Bratterud, A.; Walla, A.-A.; Haugerud, H.; Engelstad, P.E.; Begnum, K. IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services. In Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, BC, Canada, 30 November–3 December 2015; pp. 250–257.
49. Kivity, A.; Laor, D.; Costa, G.; Enberg, P.; Har'El, N.; Marti, D.; Zolotarov, V. Osv: Optimizing the operating system for virtual machines. In Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference, Philadelphia, PA, USA, 17–20 June 2014; pp. 61–72.
50. Martins, J.; Ahmed, M.; Raiciu, C.; Olteanu, V.; Honda, M.; Bifulco, R.; Huici, F. ClickOS and the art of network function virtualization. In Proceedings of the 11th USENIX Conference on Net-worked Systems Design and Implementation USENIX Association, Seattle, WA, USA, 2–4 April 2014.
51. Galois Inc. The Haskell Lightweight Virtual Machine (HaLVM) Source Archive [EB/OL]. Available online: http://galois.com/project/HaLVM/ (accessed on 20 January 2018).
52. Erlangonxen.org. Erlang on Xen: At the Heart of Super-Elastic Clouds [EB/OL]. Available online: http://erlangonxen.org/ (accessed on 20 January 2018).
53. Ballesteros, F.J. The Clive Operating System. Technical Report, 4 October 2014. Available online: http://mirror.postnix.pw/clive-man/clivesys.pdf (accessed on 20 January 2018).
54. Porter, D.E.; Boyd-Wickizer, S.; Howell, J.; Olinsky, R.; Hunt, G.C. Rethinking the library OS from the top down. *ACM SIGPLAN Not.* **2011**, *46*, 291–304. [CrossRef]
55. Runtime. JS Community. Runtime.js—JavaScript Library Operating System for the Cloud. Available online: http://runtimejs.org/ (accessed on 20 January 2018).
56. Kantee, A. Puffs-Pass-to-Userspace framework file system. *Asiabsdcon* **2007**, 29–42. Available online: https://research.aalto.fi/en/publications/puffs-pass-to-userspace-framework-system (accessed on 20 January 2018).
57. Lankes, S.; Pickartz, S.; Breitbart, J. HermitCore: A Unikernel for Extreme Scale Computing. In Proceedings of the International Workshop on Runtime & Operating Systems for Supercomputers ACM, Kyoto, Japan, 1 June 2016.
58. Ramirez, J.; Ezenwigbo, O.A.; Karthick, G.; Trestian, R.; Mapp, G. A New Service Management Framework for Vehicular Networks. In Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 24–27 February 2020; pp. 162–164.

59. Ezenwigbo, O.A.; Ramirez, J.; Karthick, G.; Mapp, G.; Trestian, R. Exploring the Provision of Reliable Network Storage in Highly Mobile Environments. In Proceedings of the 2020 13th International Conference on Communications (COMM), Bucharest, Romania, 18–20 June 2020; pp. 255–260.
60. Sarrigiannis, I.; Contreras, L.M.; Ramantas, K.; Antonopoulos, A.; Verikoukis, C. Fog-Enabled Scalable C-V2X Architecture for Distributed 5G and Beyond Applications. *IEEE Netw.* **2020**, *34*, 120–126. [CrossRef]
61. Zhao, J.; Tiplea, T.; Mortier, R.; Crowcroft, J.; Wang, L. Data Analytics Service Composition and Deployment on IoT Devices. In Proceedings of the 16th Annual International Conference on Digital Government Research, Delft, The Netherlands, 30 May–3 June 2018; Association for Computing Machinery (ACM): New York, NY, USA, 2018; pp. 502–504.
62. Cozzolino, V.; Ott, J.; Ding, A.Y.; Mortier, R. ECCO: Edge-Cloud Chaining and Orchestration Framework for Road Context Assessment. In Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, Australia, 21–24 April 2020; pp. 223–230.
63. Vittorio, G.; Ding, A.Y.; Ott, J. FADES: Fine-Grained Edge Offloading with Unikernels. In Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems, Los Angeles, CA, USA, 25 August 2017.
64. Valsamas, P.; Skaperas, S.; Mamatas, L. Elastic Content Distribution Based on Unikernels and Change-Point Analysis. European Wireless 2018. In Proceedings of the 24th European Wireless Conference, Catania, Italy, 2–4 May 2018; pp. 1–7.
65. Behravesh, R.; Coronado, E.; Riggio, R. Performance Evaluation on Virtualization Technologies for NFV Deployment in 5G Networks. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019; pp. 24–29.
66. Filipe, J.B.; Meneses, F.; Rehman, A.U.; Corujo, D.; Aguiar, R.L. A Performance Comparison of Containers and Unikernels for Reliable 5G Environments. In Proceedings of the 2019 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, 19–21 March 2019; pp. 99–106.
67. Aggarwal, V.; Thangaraju, B. Performance Analysis of Virtualisation Technologies in NFV and Edge Deployments. In Proceedings of the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bengaluru, India, 2–4 July 2020; pp. 1–5.
68. Watada, J.; Roy, A.; Kadikar, R.; Pham, H.; Xu, B. Emerging Trends, Techniques and Open Issues of Containerization: A Review. *IEEE Access* **2019**, *7*, 152443–152472. [CrossRef]
69. Wang, Q.; Jiao, W.; Wang, P.; Zhang, Y. Digital Twin for Human-Robot Interactive Welding and Welder Behavior Analysis. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 334–343. [CrossRef]
70. Wang, F.-Y.; Wang, X.; Li, L.; Li, L. Steps toward Parallel Intelligence. *IEEE/CAA J. Autom. Sin.* **2016**, *3*, 345–348. [CrossRef]
71. Ghahramani, M.; Qiao, Y.; Zhou, M.; Hagan, A.O.; Sweeney, J. AI-based modeling and data-driven evaluation for smart manu-facturing processes. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1026–1037. [CrossRef]
72. Mahmud, R.; Toosi, A.N.; Ramamohanarao, K.; Buyya, R. Context-Aware Placement of Industry 4.0 Applications in Fog Computing Environments. *IEEE Trans. Ind. Inform.* **2019**, *16*, 7004–7013. [CrossRef]
73. Fang, L.; Ge, C.; Zu, G.; Wang, X.; Ding, W.; Xiao, C.; Zhao, L. A Mobile Edge Computing Architecture for Safety in Mining Industry. In Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 19–23 August 2019; pp. 1494–1498.
74. Ashjaei, M.; Bengtsson, M. Enhancing smart maintenance management using fog computing technology. In Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 December 2017; pp. 1561–1565.
75. O'Donovan, P.; Gallagher, C.; Bruton, K.; O'Sullivan, D.T. A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications. *Manuf. Lett.* **2018**, *15*, 139–142. [CrossRef]
76. Zhu, Z.; Han, G.; Jia, G.; Shu, L. Modified DenseNet for Automatic Fabric Defect Detection With Edge Computing for Minimizing Latency. *IEEE Internet Things J.* **2020**, *7*, 9623–9636. [CrossRef]
77. Hussain, R.F.; Salehi, M.A.; Semiari, O. Serverless Edge Computing for Green Oil and Gas Industry. In Proceedings of the 2019 IEEE Green Technologies Conference (GreenTech), Lafayette, LA, USA, 3–6 April 2019; pp. 1–4.
78. Konishi, T.; Nakamichi, T.; Kamikawa, R.; Yamasaki, Y. Spectroscopic Inspection Optimization for Edge Computing in Industry 4.0. In Proceedings of the 2020 22nd International Conference on Transparent Optical Networks (ICTON), Bari, Italy, 19–23 July 2020.
79. Fraga-Lamas, P.; Fernandez-Carames, T.M.; Blanco-Novoa, O.; Vilar-Montesinos, M.A. A Review on Industrial Augmented Reality Systems for the Industry 4.0 Shipyard. *IEEE Access* **2018**, *6*, 13358–13375. [CrossRef]
80. Li, H.; Hu, G.; Li, J.; Zhou, M. Intelligent Fault Diagnosis for Large-Scale Rotating Machines Using Binarized Deep Neural Networks and Random Forests. *IEEE Trans. Autom. Sci. Eng.* **2021**. [CrossRef]
81. Chen, S. Collaborative Scheduling Models of Computing Tasks and Lightweight Virtualization Technology for Edge Computing. Ph.D. Thesis, Macau University of Science and Technology, Macau, China, 2021.
82. Liu, P.; Li, X.; Qu, J.J.; Wang, W.; Zhao, C.; Pichel, W. Oil spill detection with fully polarimetric spill detection with fully polarimetric UAVSAR data. *J. Mar. Pollut. Bull.* **2011**, *62*, 2611–2618. [CrossRef]
83. Fingas, M.; Brown, C.E. *Oil Spill Remote Sensing: A Review*; Elsevier: Amsterdam, The Netherlands, 2011; pp. 111–169. ISBN 9781856179430.

84. Salehi, M.A.; Smith, J.; Maciejewski, A.A.; Siegel, H.J.; Chong, E.K.; Apodaca, J.; Briceno, L.D.; Renner, T.; Shestak, V.; Ladd, J.; et al. Stochastic-based robust dynamic resource Allocation for inde-pendent tasks in a heterogenous computing system. *J. Parallel Distrib. Comput. (JPDC)* **2016**, *97*, 96–111. [CrossRef]
85. Zhang, P.; Zhou, M. Security and Trust in Blockchains: Architecture, Key Technologies, and Open Issues. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 790–801. [CrossRef]
86. Weiss GitHub.cf-unik/unik: The Unikernel Compilation and Deployment Platform [EB/OL]. Available online: https://github.com/cf-unik/unik (accessed on 20 January 2018).
87. Huang, Z.; Xu, X.; Zhu, H.; Zhou, M. An Efficient Group Recommendation Model with Multiattention-Based Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4461–4474. [CrossRef] [PubMed]
88. Liu, H.; Chatterjee, I.; Zhou, M.; Lu, X.S.; Abusorrah, A. Aspect-Based Sentiment Analysis: A Survey of Deep Learning Meth-ods. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 1358–1375. [CrossRef]
89. Fortino, G.; Messina, F.; Rosaci, D.; Sarne, G.M.L. ResIoT: An IoT social framework resilient to malicious activities. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1263–1278. [CrossRef]
90. Luo, X.; Wu, H.; Yuan, H.; Zhou, M. Temporal Pattern-Aware QoS Prediction via Biased Non-Negative Latent Factorization of Tensors. *IEEE Trans. Cybern.* **2020**, *50*, 1798–1809. [CrossRef]
91. Guo, X.; Zhou, M.; Liu, S.; Qi, L. Lexicographic Multiobjective Scatter Search for the Optimization of Sequence-Dependent Se-lective Disassembly Subject to Multiresource Constraints. *IEEE Trans Cybern.* **2020**, *50*, 3307–3317. [CrossRef]
92. Fu, Y.; Zhou, M.; Guo, X.; Qi, L. Scheduling Dual-Objective Stochastic Hybrid Flow Shop with Deteriorating Jobs via Bi-Population Evolutionary Algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 5037–5048. [CrossRef]
93. Zhao, Z.; Liu, S.; Zhou, M.; Guo, X.; Qi, L. Decomposition Method for New Single-Machine Scheduling Problems from Steel Production Systems. *IEEE Trans. Autom. Sci. Eng.* **2019**, *17*, 1–12. [CrossRef]
94. Zhou, M.; Cao, Z.; Wang, Y. Robust fault detection and isolation based on finite-frequency H-/H∞ unknown input observers and zonotopic threshold analysis. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 750–759. [CrossRef]
95. Cao, Z.; Lin, C.; Zhou, M.; Huang, R. Scheduling Semiconductor Testing Facility by Using Cuckoo Search Algorithm with Re-inforcement Learning and Surrogate Modeling. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 825–837. [CrossRef]
96. Jin, G.; Deng, M. Operator-based robust nonlinear free vibration control of a flexible plate with unknown input nonlinearity. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 442–450. [CrossRef]
97. Wu, Y.; Deng, M. Experimental study on robust nonlinear forced vibration control for an L-shaped arm with reduced control inputs. *IEEE/ASME Trans. Mechatron.* **2018**, *22*, 2186–2195. [CrossRef]