Application of Docker Swarm cluster for testing programs, developed for system of devices within paradigm of Internet of things

Y A Shichkina, M S Kupriyanov, S O Moldachev

Saint Petersburg Electrotechnical University "LETI", 5, Professora Popova St., St. Petersburg, 197376, Russia

E-mail: strange.y@mail.ru

Abstract. Today, a description of various Internet devices very often appears on the Internet. For the efficient operation of the Industrial Internet of things, it is necessary to provide a modern level of data processing starting from getting them from devices ending with returning them to devices in a processed form. Current solutions of the Internet of Things are mainly focused on the development of centralized decisions, projecting the Internet of Things on the set of cloud-based platforms that are open, but limit the ability of participants of the Internet of Things to adapt these systems to their own problems. Therefore, it is often necessary to create specialized software for specific areas of the Internet of Things. This article describes the solution of the problem of virtualization of the system of devices based on the Docker system. This solution allows developers to test any software on any number of devices forming a mesh.

1. Introduction

Internet of Things (IoT) is a system of interconnected computer devices, mechanisms and computers, objects, animals or people that have unique identifiers and the ability to transfer data over the network without human intervention. An example of a participant in an IoT ecosystem can be a car with built-in sensors that warn the driver of the danger, or it can be a person with a medical bracelet, which collects health data and then sends them to the doctor.

But, the Internet of things is not just the devices connected to the Internet. This is a system of devices that has three properties: first - connection to the Internet, second - the interaction of devices, third is the ability to analyze the situation and make a decision for a person, the ability to guess his future actions and desires.

In fact, people started talking about the Internet of things in the beginning of the 20th century.

So, in 1926 Nikola Tesla in an interview given to Collier's magazine said that in the future, the radio will be transformed into a "big brain". All things will become a part of a single whole, and management tools will easily fit in a pocket.

The first Internet thing appeared in 1990. It was a toaster.

In 1999, at the meeting of the developers of sensory technologies, a new loud term was required, meaning a big step forward. Then Kevin Ashton first proposed the term "Internet of Things".

In 2003, about 6.3 billion people lived on the planet, and 500 million devices were connected to the Internet. If one divides the number of connected devices by the number of people on the planet, one will get that for each person there is less than one device. Thus, according to Cisco's definition, in
2003, the Internet of things had not existed yet. Smartphones that time only appeared on the market. Let us recall that Steve Jobs announced the iPhone only four years later - in 2007.

In 2010, the number of connected devices grew to 12 billion, while the world population was 6.8 billion people. Thus, for the first time in history, each person has had more than one connected device.

Therefore, scientists believe that the transition from the "Internet of people" to the "Internet of things" occurred in 2009.

Today there is also the industrial Internet of things, when interaction through the Internet occurs at the level of industrial machines and systems. But, despite the fairly long development of IoT technologies, there are still many problems in this area.

For the efficient operation of the Industrial Internet of things, one needs to provide a modern level of data processing from getting them from devices to returning them to devices in a processed form.

According to forecasts for 2020, there will be 20 million smart devices in the world that can communicate with each other. When developing software for IoT, it is necessary to have the device itself, which is not always possible. The solution to this problem is virtualization.

Many devices of the Internet of things have low computing powers, but if these devices are combined into a cluster, the total capacity will be sufficient for solving various applied problems. An example of such task is real-time recognition of car numbers. Its solution consists of two stages:

1. union of devices into a cluster;
2. distribution of computing tasks by cluster nodes.

This article describes these two stages. A possible solution for creating a test stand from various IoT devices for testing software and the functioning of a system of devices based on the Docker platform is considered. The obtained results were tested on the task of testing the developed software for recognizing car numbers on images and video.

The content of the article is as follows. Section 1 describes the main stages and trends in the development of IoT. Section 2 describes similar research in the field of IoT. In Section 3, the existing methods of virtualization are considered and the way of virtualizing the level of the operating system chosen by the authors, namely Docker, is justified. Section 4 shows the use of the Docker platform to create a virtual environment for testing software for analyzing images or video in identifying vehicle numbers. Section 5 focuses on the ability of Docker to consolidate individual physical or virtual machines into clusters. This mode allows users to emulate situations when several devices are in the same network. Section 6 provides an example of how a web service works to distribute tasks on cluster nodes. In Section «Conclusion», the results of research are summed up.

2. Related Works

Due to the fact that the Internet of things is a very rapidly developing field, in the last 5 years there have appeared a lot of research related to this field. Most of these research are related to application aspects, with reliability and security, with the processing of a large data flow, with the reliability and performance of networks.

So, throughout the last decade, many researchers have dedicated their contributions to handling the bottleneck issue. At the service level, a data reduction could be accomplished by aggregation and compression techniques, regardless of application information, for example, in [1]. Along with a reducing amount, the quality of service (QoS) such as throughputs and delays could also be specified and controlled [2]. Above the service level, the solutions on application level exploit user-specified requirements to draw only requested data streams (i.e., drop unrequested streams) [3-5]. Solution of the problem of bottlenecks on the local gateway is considered in article [6].

The problem of information compression is discussed in [7], where it is said that although the typical network bandwidth available in mobile devices has been improving it remains limited with the rise in communication activity. Therefore, data could be optimised on the device to make it more suitable for the available network bandwidth. A scalable real-time data sharing system can be built by
using existing message formats, messaging architectures and compression techniques. The article considers the problem of the bandwidth limitation and scalability issue with special focus on the impact of message compression in such networks [7].

The problems of processing large data are discussed in [8]. Paper [8] focuses on the specific problem of Big Data classification of network intrusion traffic. It discusses the system challenges presented by the Big Data problems associated with network intrusion prediction. The prediction of a possible intrusion attack in a network requires continuous collection of traffic data and learning of their characteristics on the fly.

When the problem concerns large amounts of data, it is always relevant to the question of the application of high-performance systems and special boards. A number of publications are also devoted to this [9-11].

However, employing multicore in mobile devices triggers a new issue. As the number of cores increases, the power consumption used by cores becomes a significant portion in mobile devices. The increased power consumption puts pressure on the energy resource in a mobile device due to the limitation in battery capacity [12]. In addition, it causes the high internal temperature in a mobile device [13], having a potential to result in the thermal runaway [14].

Sensor networks offer a powerful combination of distributed sensing, computing and communication. They lend themselves to countless applications and, at the same time, offer numerous challenges due to their peculiarities, primarily the stringent energy constraints to which sensing nodes are typically subjected. The distinguishing traits of sensor networks have a direct impact on the hardware design of the nodes at least four levels: power source, processor, communication hardware, and sensors. [23]

A lot of research is devoted to information security issues. The proposed in [15] method can detect and report the possible IoT attacks with three types: jam-attack, false-attack, and reply-attack automatically.

Because the resources of IoT’s front devices are constrained, many security mechanisms are hard to be implemented to protect the IoT networks. Some lightweight encryption methods are considered as the core technology to build the security mechanism of IoT [16], but considering the increments of the hacker’s computation ability (the usage of Cloud Computing, Distributed Computing, Quantum computation, etc.), those lightweight cryptography methods are going to be crushed in the foreseeable future. Other kinds of security enforcement methods, such as the intrusion detection system, should be considered to protect the IoT networks [17].

And of course a lot of research is being done in the field of applying the technology of the Internet of things to different spheres of human activity. There is a mechanism for determining the health of the elderly on the basis of daily activities [18], a new medical information system called Alarm Net designed for smart healthcare [19], use of sensors and sensor networks to reduce GHG emissions and to evaluate their impact on the environment [20] and other studies.

All these studies are undoubtedly very important, but there is an immediate group of issues without which the Internet of things simply can not exist. These questions concern the organization of the Internet of things. First of all, these are networks [21,22].

Getting nearer to the world of the Internet of Things, a variety of promising applications, for instance, health monitoring, product monitoring, structure monitoring, smart appliance control, and smart buildings, have emerged. Those applications can be driven by up to petabyte scale of data streams. A high volume of collected values and metadata, as well as a variety and velocity of the streams, are challenging.

Towards the world of the Internet of Things, people utilize knowledge from sensor streams in various kinds of smart applications. The number of sensing devices is rapidly increasing along with the amount of sensing data.

In view of the large difference in sensory devices, research in the field of building and testing systems from various interconnected devices of the Internet of things is very important today.

In this area, it is necessary to identify research related to virtualization.
Given the fact that many companies are adopting a virtual desktop infrastructure (VDI) for private cloud computing to reduce costs and enhance the efficiency of their servers, some studies were devoted to the search of various desktop virtualization solutions based on systems, such as VMware, Citrix, and Microsoft [24].

With the advancement of computing and network virtualization technology, the networking research community is interested in network emulation. Compared with network simulation, network emulation can provide more relevant and comprehensive details. In [25] EmuStack, a large-scale real-time emulation platform for Delay Tolerant Network (DTN) is proposed.

Emulab [26] is one of the well-known testbeds using the container virtualization in Linux. Due to the efficiency of container, Emulab possesses a good support for scalability. Although these technologies introduced in Emulab are not the latest now, the design philosophies are still helpful for current researchers to design large-scale test bed. Additionally, Lantz et al. [27] designed Mininet based on the container virtualization technique including processes and network namespaces technique. Mininet can support SDN and run on a single computer. Handigol et al. [28] further improved Mininet performance with enhancements to a resource provisioning, isolation, and monitoring system. Besides, Handigol replicated a number of previously published experimental results and proved that Linux Container (LXC) [29] technology is not only lightweight but also possesses a good fidelity and performance. In order to perform an in-depth performance evaluation of LXC, Xavier et al. [30] conducted a number of experiments to evaluate various compute virtualization technologies and finally proved that LXC virtualization has a near-native performance on CPU, memory, disk, and network.

In this article, the authors want to show the application of the Docker system for virtualization.

3. Choosing a way of virtualization

Today there are different ways of virtualization:

1. Emulation of equipment. This type of virtualization involves the joint development of hardware and embedded software. Using this approach, a virtual machine (VM) of hardware is created on the host system for simulating the required hardware.

In order for the developers of embedded software not to depend on the availability of real hardware, they can use virtual equipment for testing and developing software.

The main problem in the case of hardware emulation is a strong slowdown in the performance of running programs in a virtual environment. This happens due to making software simulation of hardware. In this case, the slowdown can reach 100 times. Nevertheless, hardware emulation has significant advantages over other ways of virtualization. For example, when using this type of virtualization may be used an operating system that is designed for PowerPC® on a system with an ARM processor, and can also manage multiple virtual machines, each of which will be modeled by another processor.

2. Full (hardware) virtualization. This virtualization model uses middleware software, a hypervisor, or a virtual machine manager. The hypervisor is the connecting node between the hardware and the guest OS.

The virtual machine manager must have installed protection, because the hardware of the guest OS does not belong to it, but is shared by the manager. Hardware virtualization is used to create corporate systems. In this case, large vendors such as VMware, IBM and Microsoft are creating their own virtualization platforms based on hardware virtualization technologies Intel VT (VT-x), AMD-V.

3. Paravirtualization is another way of virtualization, which has some similarities with full virtualization. This virtualization approach uses a hypervisor only to share access to hardware resources. In this case, the code that concerns virtualization is combined in the OS. This approach helps the complete removal of such operations as recompilation or interception, because the guest OS and the host operating system are combined during the virtualization process. But to achieve this result, it is necessary that the guest operating system be modified for the hypervisor and this fact is a
disadvantage of this type of virtualization, because not every OS has open source code for modification.

But, despite the significant disadvantage due to the necessary changes in the kernel code of the guest OS, paravirtualization provides performance comparable to the performance of real systems. In this case, as in the case of full virtualization, different operating systems can simultaneously be supported.

4. Operating system level virtualization. If this type of virtualization is used, the guest operating systems are created above the host operating system. Therefore, this method supports a single operating system and, in general, guest OS (containers) are isolated from each other. In order to share resources between containers, it is necessary to make changes to the core of the host operating system (for example, OpenVZ), but this minus is leveled by native performance, without imposing any costs for virtualization of devices. This approach is used in Virtuozzo / OpenVZ, Linux Containers (LXC), KVM, Docker.

From the above discussed methods of virtualization to develop and test the software, the authors used virtualization of the operating system level, namely the Docker software. This method allowed us to cover a much larger number of devices compared to the ways in which the equipment is emulated.

4. Using the Docker platform to create a virtual environment

Docker is an open platform for container virtualization on Linux. As it was mentioned above, this software uses operating system-level virtualization and has a requirement for a specific version of the Linux kernel, i.e. 3.10 or higher.

Using Docker processes can be run in an isolated environment. The process running in Docker assumes that it works in a minimal environment, where except it there are only its child processes. Although the process is running in the same operating system as other normal processes, this process simply does not see them, exactly as it does not see the files and everything else outside of its isolated area.

The minimum environment in which the process runs is called the container. The container is an isolated operating system with its own network, files and devices. The container can interact with both the main operating system and with other containers, for example, mount certain folders and files, use a common network, open any necessary ports.

To create a docker container, it is necessary to use images. The image is a template that contains the operating system and the necessary applications. Images can be located either locally or in a special public registry (dockerhub). The image template is described declaratively in the Dockerfile, using a special syntax. Next, let us consider an example of an image template that contains the operating system Ubuntu and has preinstalled software OpenALPR.

```
FROM ubuntu:16.04
RUN apt-get update
&& apt-get install -y
openalpr
openalpr-daemon
openalpr-utils
libopenalpr-dev
WORKDIR /data
ENTRYPOINT ["alpr"]
```

OpenALPR - free software created in C ++ that allows one to analyze images or video to identify car numbers.

In order to create an image from the template, it is necessary to use the command «docker build». As a result, one obtained a ready environment, which can be run on an unlimited number of containers.

Below are the results obtained with the software OpenALPR, which was launched in a container:

docker build --tag openalpr

docker run openalpr
5. Combining IoT devices in the cluster

It should be noted that Docker has the ability to combine separate physical or virtual machines into clusters, this possibility is called «swarm». This mode allows users to emulate situations when several devices are in the same network.

Using Docker Swarm, it is necessary to test the prototype. To do this, it is necessary to create nodes using the Docker Machine. Docker Machine is a tool that turns virtual or physical servers into Docker nodes. At the moment, Docker Machine includes about 12 drivers for various cloud platforms, including Amazon EC2, VirtualBox, Google Cloud Platform and OpenStack.

For testing, there were 3 virtual nodes using the following command sequence:

```
docker-machine create -d virtualbox node1
docker-machine create -d virtualbox node2
docker-machine create -d virtualbox node3
```

Next, it is necessary to define the roles for the created nodes and make the network:
```
docker swarm init --advertise-addr 192.168.99.100
```
Swarm initialized: current node (wlx5xrspv4enim4c1xbmcuo) is now a manager.
```
docker network create --driver overlay --subnet 192.168.1.0/24 --opt encrypted mesh-network
```

After creating a «swarm» cluster, one can create and launch services. For testing, it is necessary to create two images. The first will contain the broker's code; the second image will contain the agent code. Next, it is necessary to create a configuration file docker-compose.yml, which will describe the services and deployment rules. To combine all services into a single network, one uses networks section. The cluster is created using the command «docker stack deploy -c docker-compose.yml mesh» (fig. 1).

```
5. Combining IoT devices in the cluster

The docker-compose.yml will contain the following rules:

1. the broker service must be deployed on a node that acts as a manager;
2. the service agent must be located on the nodes acting in the role of workers;
3. the number of instances of the service «agent» must be equal to two.

Below is the code of the broker:
```
version: "3"
services:
  broker:
    image: sergeyleti/broker
    ports:
deploy:
  mode: global
placement:
  constraints: [node.role == manager]
agent:
  image: sergeyleti/agent
deploy:
  replicas: 2
placement:
  constraints: [node.role == worker]

networks:
  default:
  external:
    name: mesh-network

6. An example of the functioning of a web service for task distribution on cluster nodes

After the cluster is created, the broker of task and its web interface will be available on port 54859. View information about the available devices (Fig. 2) by clicking on the link http://<broker_ip>:54589/devices/info/all. It can be seen that agents are started only on two nodes; this is confirmed by the available information about the resources.

```json
{
  "192.168.1.1": {},
  "192.168.1.10": {},
  "192.168.1.2": {
    "cpu": {
      "percents_free": 99.0,
      "percents_load": 1.0
    },
    "memory": {
      "free": 144165684,
      "total": 1044127744,
      "use": 899932100
    },
    "processes": {
      "count": 2
    }
  },
  "192.168.1.3": {
    "cpu": {
      "percents_free": 96.6,
      "percents_load": 3.4
    },
    "memory": {
      "free": 144175312,
      "total": 1044127744,
      "use": 900378432
    },
    "processes": {
      "count": 2
    }
  },
  "192.168.1.4": {
    "cpu": {
      "percents_free": 96.0,
      "percents_load": 4.0
    },
    "memory": {
      "free": 144162816,
      "total": 1044127744,
      "use": 899665928
    },
    "processes": {
      "count": 2
    }
  },
  "192.168.1.5": {},
  "192.168.1.6": {}
}
```

Figure 2. The status of all devices in the network.
To test the system, it is necessary to create a task by clicking on the link http://<broker_ip>:54589/task/spawn, fill out the form and send the task for execution. Figure 3 shows a screenshot of the task creation form, which determines the vehicle number from the photo, using the ALPR (Automatic license-plate recognition) technology.

![Task creation form](image)

**Figure 3.** The task creation form

The car number recognition software is added to the agent image and is available in the container. The result of solving this problem is shown in Figure 4.
Figure 4. The result of the task of recognizing the number of the car

The task distribution statistics is available at http://<broker_ip>:54589/tasks/stat. The screenshot with the output of the result is presented below in Figure 5.

Figure 5. The task distribution statistics

This example illustrates the possibility of testing software developed for IoT. Docker Swarm cluster allows one to add an unlimited number of nodes, and Docker allows one to run an unlimited number of containers on these nodes, which allows one to do full-scale testing that is close to real conditions.

7. Conclusion

IoT technology at the moment is a promising technology that can radically change the world. But today, in this new paradigm of communication between people and things, there are many problems that need to be solved, and one of them is the possibility of testing software in conditions close to real.

Based on the results of the done work, one can conclude that Docker and container virtualization is ideally suited for the development and testing of software in the field of IoT. Isolation of processes in containers allows increasing the level of security of IoT devices, the ability to create various configurations of virtual environments, testing more thoroughly and comprehensively the created software and reducing the number of defects.
Acknowledgments
The paper has been prepared within the scope of the state project “Initiative scientific project” of the main part of the state plan of the Ministry of Education and Science of the Russian Federation (task № 2.6553.2017/8.9 BCH Basic Part).

References
[1] Bagale J N, Shiyanbola A, Moore J P T, and Kheirkhahzadeh A D 2014 Towards a real-time data sharing system for mobile devices, 8th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST '14, Oxford, UK) pp. 147–152
[2] Ni Q, Romdhani I, and Turletti T 2004 A survey of QoS enhancements for IEEE 802.11 wireless LAN, Wireless Communications and Mobile Computing 4(5) 547–566
[3] Sun Y, Wu T-Y, Zhao G, and Guizani M 2015 Efficient rule engine for smart idling systems, IEEE Transactions on Computers 64(6) 1658–1669
[4] Estrin D, Govindan R, Heidemann J, and Kumar S 1999 Next century challenges: scalable coordination in sensor networks, 5th Annual International Conference on Mobile Computing and Networks (MobiCom ’99, Seattle, DC, USA) pp. 263–270
[5] Hunkeler U, Truong H L, and Stanford-Clark A 2008 MQTT-S—a publish/subscribe protocol for wireless sensor networks, 3rd IEEE/Create-Net International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE ’08, Bangalore, India) pp. 791–798
[6] Sunyanaan Choochotkaew, Hirozumi Yamaguchi, and Teruo Higashino, Two-Tier 2017 Vol Prioritization System on Requirement-Based Data Streaming toward IoT, Mobile Information Systems 2017 7892545
[7] Bagale J N, Shiyanbola A, Moore J P T, and Kheirkhahzadeh A D 2014 Towards a real-time data sharing system for mobile devices, 8th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST '14, Oxford, UK) pp. 147–152
[8] Zhang Y, Yu R, Xie S, Yao W, Xiao Y, and Guizani M 2011 Home M2M networks: architectures, standards, and QoS improvement, IEEE Communications Magazine 49(4) 44–52
[9] Baun C 2016 Mobile clusters of single board computers: an option for providing resources to student projects and researchers, SpringerPlus 5 360
[10] Guan T, Wang Y, Duan L, and Ji R 2015 On-device mobile landmark recognition using binarized descriptor with multifeature fusion, ACM Transactions on Intelligent Systems and Technology 7 12
[11] Zhu M and Shen K 2016 Energy discounted computing on multicore smartphones, USENIX Annual Technical Conference (ATC ’16, Denver, Colo, USA)
[12] Carroll and G. Heiser 2014 Unifying DVFS and offlining in mobile multicore, 20th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS ’14) pp. 287–296
[13] Tawara Y, Idehara A, and Yamamoto H 2010 DVFS and power-off controls on a multicore operating system, 10th International Forum on Embedded MPSoC and Multicore (MPSoC ’10, Gifu, Japan)
[14] Kim J M, Kim Y G, and Chung S W 2015 Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices, IEEE Transactions on Computers 64 286–292
[15] Yulong Fu, Zheng Yan, Jin Cao, Ousmane Koné, and Xuefei Cao 2017 An Automata Based Intrusion Detection Method for Internet of Things, Mobile Information Systems 2017 1750637
[16] Boggs N, Wang W, Mathur S, Coskun B, and Pincock C 2013 Discovery of emergent malicious campaigns in cellular networks, 29th Annual Computer Security Applications Conference (ACSAC ’13, New Orleans, La, USA) pp. 29–38
[17] Wang C X, Gao X, You X, et al., 2014 Cellular architecture and key technologies for 5g wireless communication networks, IEEE Communications Magazine 5(2) 122–130
[18] R. K. Pragnya and J. K. Chaitanya, “Wireless sensor network based healthcare monitoring system for homely elders,” International Journal of Advances in Engineering & Technology, vol. 6, no. 5, pp. 2078–2083.

[19] Virone G, Wood A, Selavo L et al. 2006 An assisted living oriented information system based on a residential wireless sensor network. 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2 ’06, Arlington, Tex, USA) pp. 2–4

[20] OECD 2009 Smart sensor networks: technologies and applications for green growth, OECD Digital Economy Papers, vol. 167 (OECD Publishing, Paris, France)

[21] Ni Q, Romdhani L, and Turletti T 2004 A survey of QoS enhancements for IEEE 802.11 wireless LAN. Wireless Communications and Mobile Computing 4(5) 547–566

[22] Estrin D, Govindan R, Heidemann J, and Kumar S 1999 Next century challenges: scalable coordination in sensor networks. 5th Annual International Conference on Mobile Computing and Networks (MobiCom ’99, Seattle, DC, USA) pp. 263–270

[23] Puccinelli D and Haenggi M 2005 Wireless sensor networks: applications and challenges of ubiquitous sensing. IEEE Circuits and Systems Magazine 5(3) 19–31

[24] Doowon Jeong, Junghuem Park, Sangjin Lee, and Chulhoon Kang 2015 Investigation Methodology of a Virtual Desktop Infrastructure for IoT. Journal of Applied Mathematics 2015 689870 https://www.hindawi.com/journals/jam/2015/689870/

[25] Haifeng Li, Huachun Zhou, Hongke Zhang, Bohao Feng, and Wenfeng Shi 2016 EmuStack: An OpenStack-Based DTN Network Emulation Platform. Mobile Information Systems 2016 6540207

[26] Hibler M, Ricci R, Stoller L et al. 2008 Large-scale virtualization in the emulab network testbed, USENIX Annual Technical Conference (USENIX ’08, Boston, Mass, USA), pp. 113–128

[27] Lantz B, Heller B, and McKeown N 2010 A network in a laptop: rapid prototyping for software-defined networks. 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Monterey, Calif, USA)

[28] Handigol N, Heller B, Jeyakumar V, Lantz B, and McKeown N 2012 Reproducible network experiments using container-based emulation. 8th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT ’12, ACM, Nice, France), pp. 253–264

[29] LXC https://linuxcontainers.org/

[30] Xavier M G, Neves M V, Rossi F D, Ferreto T C, Lange T, and De Rose C A F 2013 Performance evaluation of container-based virtualization for high performance computing environments. 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP ’13), pp. 233–240