Advances in Smart Systems Research ISSN 2050-8662
Vol. 4. No. 1 : pp.32-37 : isrp15-009

Implementing SRN for Resilient Server on The Virtual Environment Using Container

Idris Winarno[a,1], Takeshi Okamoto[b,2], Yoshikazu Hata[a,3], Yoshiteru Ishida[a,4]

[a]Department of Computer Science and Engineering
Toyohashi University of Technology, Japan
[1]idris@sys.cs.tut.ac.jp, [3]hata@sys.cs.tut.ac.jp, [4]ishida@cs.tut.ac.jp

[b]Department of Information Network and Communication
Kanagawa Institute of Technology, Japan
[2]take4@nw.kanagawa-it.ac.jp

Abstract

Linux Container (LXC) offers fast boot and efficient resource usage to deploy a server on the virtual environment. However, this server could not stave off various threats. A self-repairing network (SRN) is required to solve the problem by simulating a resilient server against the threats by involving LXC and VMM. The simulation results showed that the LXC-based SRN outperformed the VMM-based SRN.

Keywords: Virtual Machine, resilient server, SRN, Container

1. Introduction

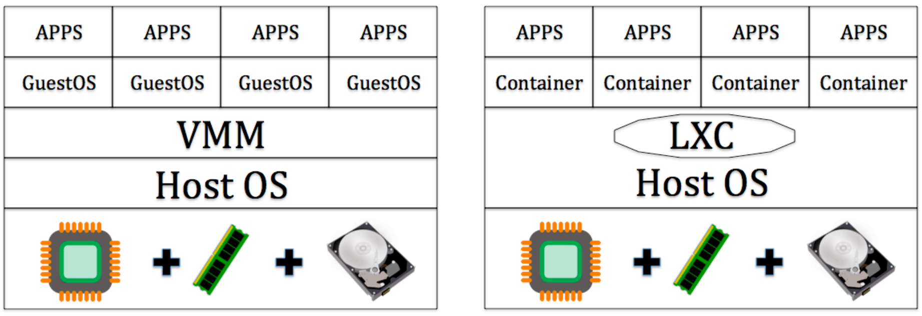
For the last 3-years, along with the growth of computer server technology, the use of virtualization has become a trend. Many industries or companies migrate their computer servers from physical or standalone machines to virtual ones. Virtualization offers many benefits [1] including CPU, memory, and I/O virtualization. By utilizing virtualization, users are able to customize their servers whatever they need to optimize the performance of the service to support their tasks.

In the other side, the growth of the computer servers' threats is never ending. Hackers always attempt to find out the weakness of the system to exploit the target resource. When we decide to move or to migrate our physical computer servers to virtual machines, it does not mean that our system is getting secure. We need to consider that virtualization technology has no proper mechanism to address the traditional computing threats [2] that occur at the application level such as a web server.

In our previous research, we introduced VMM-based SRN, which simulates a resilient server against a particular threat by involving XEN as virtual machine monitor (VMM) technology and a self-repairing network (SRN) model [3]. This paper discusses about LXC-based SRN, in which we develop the resilient server by involving Linux container (LXC) technology. We compare the performance of resilient servers between LXC-based SRN and VMM-based SRN.

2. Related work

Nowadays, we are facing the virtual technology to develop a new system or an application. VMM and LXC are come from the different model (Figure 1).



(a) (b)
Figure 1. Comparison of VMM (a) and LXC (b)

We could compare VMM and Container to several factors [4] such as guest OS, communication, security and startup time. LXC could support VMM to build a PaaS cloud computation platform [4]. We could create a virtual server easily in the short period using VMM or LXC. However, we have to keep it resilient against the attacker. A security survey for a virtual machine (VM) [5,6] reported several holes in the virtualized environment such as communication between VMs or between VMs and host, denial of service and external modification of a VM. On the other hand, SRN models are proposed for a computer network to solve the problem to deal with the network cleaning problem [7]. We implement the mutual-repair, which is one of the basic models of SRN models to repair the broken node.

3. Base System

In this paper, we use two base systems to obtain the result, which will be discussed in the section 5. The two base systems are the SRN as a self-repairing model and LXC as the virtual environment.

3.1. Self-Repairing Network Model

There are two types of the basic model of the SRN: self-repair and mutual-repair. Self-repair targets are to repair the node itself and the mutual-repair targets are to repair the other nodes. From these two basic models we could be combined into another model: mixed-repair and switching-repair [7]. In this paper, we are focusing on the mutual-repair in which the healthy and the broken node can be realized into virtual servers. Figure 2 illustrates a healthy node that repairs a broken node by copying the normal content, which could be called the inter-node repair.

3.2. Linux Container (LXC)

Linux Container (LXC) popularity increases significantly in the last two year [8]. LXC offers new ability where the VMM could not do. LXC is claimed to be faster and more efficient of resources (e.g. CPU and memory) than VMM [4,9,10]. Docker, Warden and OpenVZ are the examples of LXC. However, LXC has several weaknesses due to:

- Running on the Linux operating system
- Could not virtualize other than the Linux OS
- Use a shared kernel
- Vulnerable to a containment environment [11]

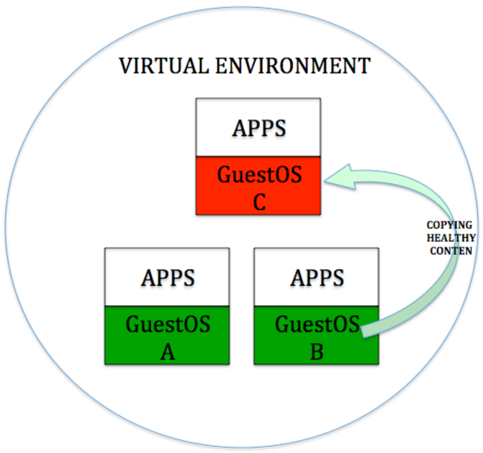


Figure 2. Mutual-repair: Guest OS B repair Guest OS C

4. Simulation Design

We design the simulation environment using Debian GNU/Linux 7 OS and Docker 1.6.2 as LXC. Figure 3 shows the logical design of the simulation. We create a new image (called Debian-ws) that consists of Debian GNU/Linux 7 OS and several additional applications. The additional applications are Tripwire (as a sensor), Apache2 (as a web server) and a script to be run periodically to detect and to respond the abnormal condition. Docker generates five containers: Debian0 to Debian4.

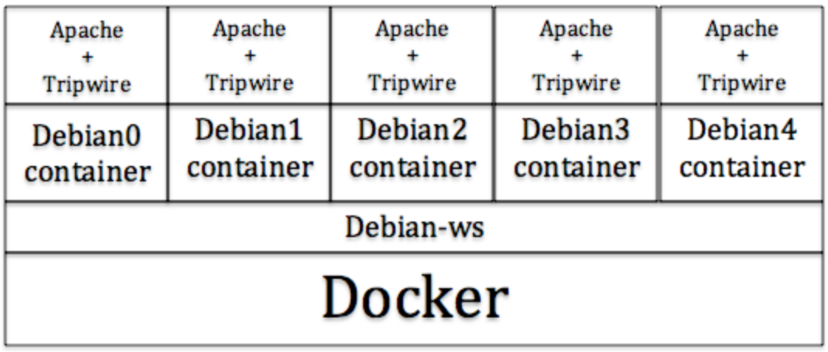


Figure 3. Docker generates five container from Debian-ws

5. Results and Analysis

Based on the simulation design as described in the section 4, we got the result as follows:

5.1. Execution time

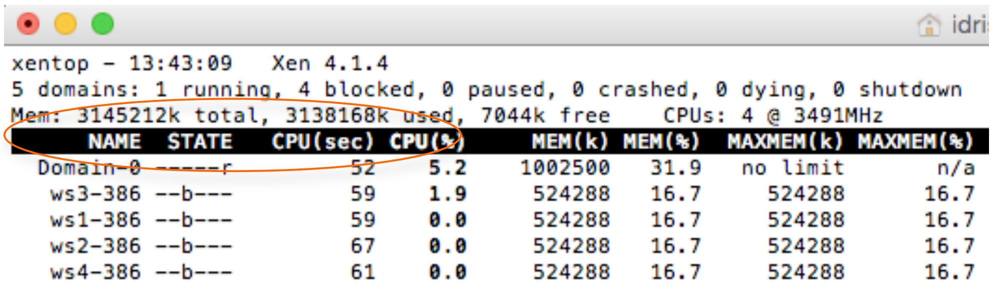
The execution time is starting to count when the broken node (container) is asked for repairing by the healthy node. The counting will be stopped when the broken node has been repaired. Table 1 shows that the execution time of VMM-based SRN is 20.75 seconds on average. Moreover, Table 2 shows that the execution time of LXC-based SRN is 2.0 seconds on average.

Hostname	Start	Stop	Duration	Origin
Ws1	1439682933	1439682953	20 sec	Ws4
Ws2	1439682731	1439682752	21 sec	Ws1
Ws3	1439682771	1439682793	22 sec	Ws1
Ws4	1439682821	1439682841	20 sec	Ws2

Table 1. Execution time of VMM-based SRN

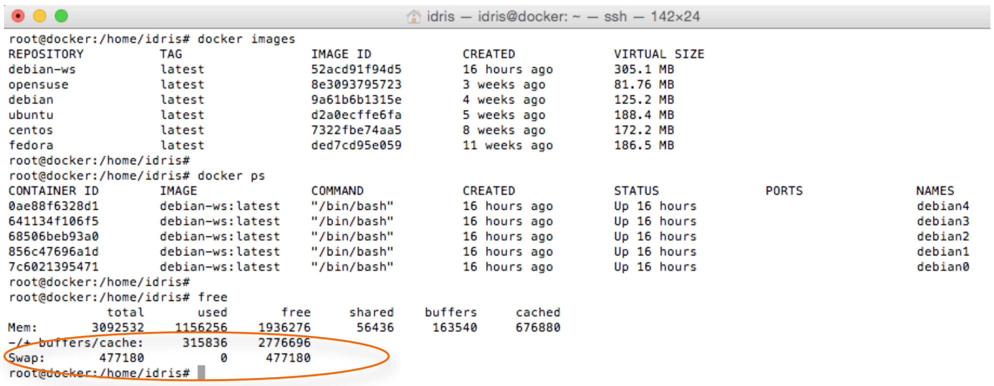
Hostname	Start	Stop	Duration	Image
Debian0	1439684766	1439684768	2 sec	Debian-ws
Debian1	1439684769	1439684771	2 sec	Debian-ws
Debian2	1439684773	1439684775	2 sec	Debian-ws
Debian3	1439684791	1439684793	2 sec	Debian-ws
Debian4	1439684795	1439684797	2 sec	Debian-ws

Table 2. Execution time of LXC-based SRN



```
xentop - 13:43:09   Xen 4.1.4
5 domains: 1 running, 4 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 3145212k total, 3138168k used, 7044k free    CPUs: 4 @ 3491MHz
      NAME  STATE   CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%)
  Domain-0 -----r         52    5.2    1002500   31.9   no limit       n/a
   ws3-386 --b---         59    1.9     524288   16.7     524288      16.7
   ws1-386 --b---         59    0.0     524288   16.7     524288      16.7
   ws2-386 --b---         67    0.0     524288   16.7     524288      16.7
   ws4-386 --b---         61    0.0     524288   16.7     524288      16.7
```

Figure 4. Memory usage of XEN with four guest OSs.



```
root@docker:/home/idris# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
debian-ws           latest              52acd91f94d5        16 hours ago        305.1 MB
opensuse            latest              8e3093795723        3 weeks ago         81.76 MB
debian              latest              9a61b6b1315e        4 weeks ago         125.2 MB
ubuntu              latest              d2a0ecffe6fa        5 weeks ago         188.4 MB
centos              latest              7322fbe74aa5        8 weeks ago         172.2 MB
fedora              latest              ded7cd95e059        11 weeks ago        186.5 MB
root@docker:/home/idris#
root@docker:/home/idris# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
0ae88f6328d1        debian-ws:latest    "/bin/bash"         16 hours ago        Up 16 hours                             debian4
641134f106f5        debian-ws:latest    "/bin/bash"         16 hours ago        Up 16 hours                             debian3
68506beb93a0        debian-ws:latest    "/bin/bash"         16 hours ago        Up 16 hours                             debian2
856c47696a1d        debian-ws:latest    "/bin/bash"         16 hours ago        Up 16 hours                             debian1
7c6021395471        debian-ws:latest    "/bin/bash"         16 hours ago        Up 16 hours                             debian0
root@docker:/home/idris#
root@docker:/home/idris# free
             total       used       free     shared    buffers     cached
Mem:       3092532    1156256    1936276      56436     163540     676880
-/+ buffers/cache:     315836    2776696
Swap:       477180          0     477180
root@docker:/home/idris#
```

Figure 5. Memory usage of Docker with five containers.

5.2. Memory usage

We allocate 3GB physical memory for each simulation for VMM-based SRN and LXC-based SRN. Figure 4 shows that VMM-based SRN was running four guest OSs (ws1-386 to ws4-386) and consumed more than 90% of physical memory, while LXC-based SRN was running five containers (Debian0 to Debian4) and consumed only approximately 37.4% of physical memory (Figure 5).

From the simulation results (the execution time and the memory usage), LXC-based SRN performance is faster and more efficient of resource (memory) rather than VMM-based SRN. This is because LXC-based SRN which shares a kernel. The shared kernel on the LXC could reduce the startup time. Nevertheless, VMM offers more security since it needs a modified kernel [4].

6. Conclusion

We could conclude that LXC-based SRN, which is implementing LXC technology, outperforms VMM-based SRN, which is implementing VMM technology. LXC-based SRN takes 2 seconds to recover the abnormal node while VMM-based SRN takes 20.75 seconds longer. VMM-based SR consumes much memory than LCX-based SRN where more than 90% is used by VMM-based SRN while LXC-based SRN only use approximately 37.4%. However, we have to consider that one of the weaknesses of the LXC is only running in the Linux environment. In the other hand, to protect against growing threats; we have to involve the diversity by creating heterogeneous nodes and involving a self-reconfiguration model.

7. References

[1] Rosenblum M, Garfinkel T, et al, Virtual machine monitors: Current technology and future trends, Computer vol. 38 no. 5, (2005)

[2] Garfinkel T, Rosenblum M, When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments, in Proceedings of the 10th conference on Hot Topics in Operating Systems vol.10, (2005)

[3] Winarno I, Ishida Y, Simulating Resilient Server Using XEN Virtualization, 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, (2015)

[4] Rajdeep D, Raja AR, et al, Virtualization vs Containerization to support PaaS, 2014 IEEE International Conference on Cloud Engineering, (2014)

[5] Reuben JS, A Survey on Virtual Machine Security, Seminar on Network Security Autumn 2007. Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, (2007)

[6] Moghadam SS, A Survey of Virtualization security, International Journal of Scientific & Engineering Research vol. 4 Issue 9, (2013)

[7] Ishida Y, Tanabe K, Dynamics of Self-Repairing Networks: Transient State Analysis on Several Repair Types, International Journal of Innovative Computing, Information and Control 10, (2014)

[8] Merkel D, Docker: Lightweight Linux Containers for Consistent Development and Deployment, Linux Journal vol. 2014 no. 239, (2014)

[9] Felter W, Ferreira A, et al, An Updated Performance Comparison of Virtual Machines and Linux Containers, IBM Research Report, (2014)

[10] Joy AM, Performance Comparison Between Linux Containers and Virtual Machines, 2015 International Conference on Advances in Computer Engineering and Application, (2015)

[11] Evading from linux containers. http://blog.bofh.it/debian/id_413