A Proposal of contributing to an open source project Solo5

Yanzhi Li
yli16@earlham.edu
Computer Science Department
Earlham College
Richmond, Indiana

KEYWORDS

Linux,Library Operating System,Operating Systems, Unikernels

1 ABSTRACT

Unikernel is a newly developed technology that will benefit cloud computing, system optimization, high performance computing, etc. Currently, its development is still preliminary while the problems of unikernel are insufficient. With more researchers involved, unikernels will become the next stage of system virtualization, or even the next evolution of operating systems. This project aims to do research in an idea of running unikernels as processes on Linux and proposes to solve issues of the corresponding open source project.

2 INTRODUCTION

The first unikernel-like systems were designed in late 1990s. However, because of the hardware requirement and the low demand of their potential, unikernels were paused and faded from researchers' sight. The technology behind unikernels has been developed over the last five years. Unlike a complete operating system such as Linux, which has an expanded kernel to support a wide range of functionality, a unikernel is stripped off all the unnecessary parts. In other word, it is a specialized, executable image that is deployed directly on hardware and linked with target application[14]. A similar idea of a lightweight runtime environment is the container which is now in great demand. However, Filipe et al. [12] pointed out that containers offer weaker isolation than VMs. Thus, the unikernel seems to become the next stage of system virtualization. Moreover, the idea of implementing unikernels on Linux has become a major branch of unikernel studies. Since this is newly-developed technology, researchers have not found many problems on this topic.

Dan et al. created an open source project called Solo5, which is a general sandboxed execution environment suitable for running applications built using various unikernels[16]. Their research provides a practical and accessible way to implement unikernels and utilize their advantages. I am interested in system virtualization and motivated to find a general methods of improving application performance. Unikernels seem to be the best fit. This proposal aims to contribute to the Solo5 open source project. Currently its goal is to solve three issues of the project including: Proper support for cross-compiling, Better error reporting in the ELF loader, Sandbox elftool. The current three issuses where chosen randomly and may change depending on my further investigation of the Solo5 source code.

This paper includes the following sections and presents them in this order: Abstract, Introduction, Related Work, Design, Budget, Timeline, and References.

3 RELATED WORK

The related work section will be divided into two parts. The first part introduces the concept of unikernels, why they are important and what benefits they have. This part will also talk about some implementations of the unikernels. The second part introduces some ideas associating unikernels with the Linux OS.

3.1 What are unikernels

The work in category concerns primarily with unikernel as a separate micro system from a complete operating system. Lankes et al. viewed unikernels as a totally independent system image to be used for extreme scale computing[8]. Projects under this category are application oriented. In other words, their unikernels care only about how to be applicable.

Unikernels are machine images constructed by using library operating systems. They are composed of the minimal set of libraries which correspond to the OS constructs required for some specific applications to run. In other words,unikernels strip away everything that are unnecessary. They are specifically designed for certain applications and do not provide any other options. Besides, unikernels can directly run on the, virtual or real, hardware without an intervening operating system[16]. Since they only use a small set of the resources required by a complete operating system, they are lightweight and platform-independent.

The idea of such a micro system came from the demand for an improvement of the traditional operating system virtualization. Anil and David in their paper stated that the traditional operating system virtualization,while being very useful,is built upon an already layered software stack and thus adds more burden to the overall system[11]. They pointed out a typical virtual machine that contains a full operating system image was reasonable in several years ago because high cost of building such a system, i.e., a single system to perform multiple tasks was desirable. However,nowadays most deployed VMs ultimately perform a single function such as acting as a database or Web server. The demand for a single-purpose virtual machine is a reflection of the inexpensive cost of building new virtual computers. Thus come the unikernels. The new technology is capable of delivering: system security, small footprints,high application optimization, near instant boot times, good resource utilization. Unikernels could dramatically improve the performance of the target application. A similar purposed technology of lightweight virtualization is the container, which is now widely used in many fields such as cloud computing. Google also ran most of its services in containers. However, Filipe et al expressed the concerns for the security problem of the containers in their paper[12]. The API that a container use to interact with the host OS is fundamentally difficult to secure, even with many isolation mechanisms

introduced in the past few years. In addition,containers are vulnerable to DoS attack.Thus they proposed to replace containers with unikernels, which provides high isolation as a complete VM and no less efficiency.

The unikernels introduced in this section are application/function oriented. In other words, they are designed for some particular applications or providing some specific functionalities. According to Raza et al, there are two approaches of creating a new unikernel: a clean slate approach where the kernel is largely built from scratch, and a strip down approach where people stripped an an existing kernel codebase of functionality that are unnecessary for the unikernel. One of the most famous unikernel implementation exploring the clean-slate design space is the MirageOS, which is extremely specialized and limited to OCaml-based applications[11]. Meanwhile, strip-down unikernels are better at porting software by preserving the general-purpose libraries and interfaces of a legacy kernel codebase. One representation of this type is the RumpRun unikernel which contains a heavily-reduced version of NetBSD[14].

In addition, unikernel communities have now developed many different unikernel implementations for other languages, some of which are able to support common applications and runtimes like nginx, redis, Node.js express, Python, etc[16]. For example, HalVM8 is an unikernel based on the famously pure and lazy Haskell language [11].

With lightweight characteristics and strong isolation, unikernels have substantial advantages for a wide class of applications. They are well suited for microservices, network function virtualization , and High-performance Computing.

3.2 Unikernels on Linux

Rather than designing a new unikernel for a specific application, work under this category focus on how to import current unikernel implementation to the Linux environment. Some papers proposed the idea of incorporating unikernels into current Operating System Linux and making them compatible with Linux applications. Raza et al argued that unikernels' advantages represent the next natural evolution for Linux[14]. While Pierre et al pointed out that the barrier to their widespread adoption is the difficulty/impossibility to port existing applications to current unikernels, they created an unikernel HermiTux which is the first unikernel providing binary-compatibility with Linux applications [11]. These people aimed to make unikernels backward compatible so that the unikernel could be more acceptable and more people will be involved in its development.

Raza et al. demonstrated that Linux can be turned into a unikernel successfully[14]. Pierre et al. built HermiTux , a unikernel that runs native Linux executables by providing binary compatibility, relieving application programmers from the effort of porting their software[11]. Dan et al. used a different approach by implementing unikernels as processes[16]. They all showed the possibility of merging the new technology with the complete operating system.

4 DESIGN

4.1 Solo5 Structure

Dan et al. implemented unikernels as processes on Linux[16]. Figure 1 shows an overview of unikernels as processes. They created a tender process, which has two main tasks: setup and "exit" handling. They also built a prototype system, called nabla, to demonstrate unikernels running as processes and implemented nabla as part of the Solo5 unikernel ecosystem, as shown in Figure 2.

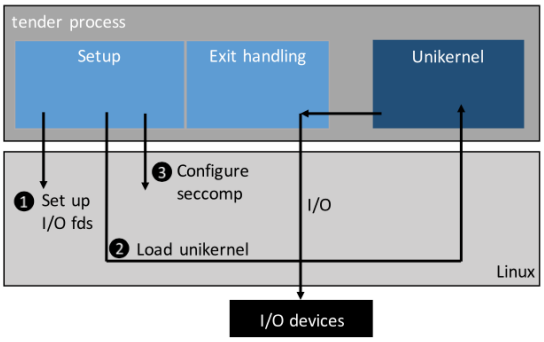

Figure 1: Unikernel isolation using a tender process with seccomp technology[16]

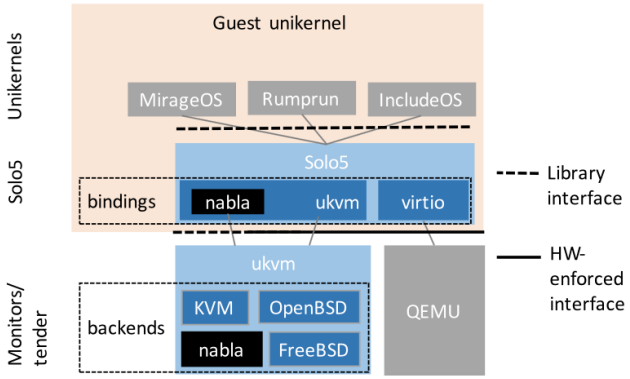

Figure 2: The Solo5 unikernel ecosystem. Nabla is implemented as a new Solo5 binding and a new backend for the ukvm monitor[16]

4.2 Project Modules

The project consists of four steps as shown in Figure 3. First, I will do some prelimary work to create an environment for building Solo5. Second, I will pull from the Solo5 repository and build it on the VM. Also, I will replicate Dan et al.'s work in the paper. Third, I will choose three issues to solve or optimize the source code based on my investigation. Finally, I will conduct experiments on two applications' performance and I will communicate with other contributors to receive feedback on my project. I will evaluate my results according to feedback.

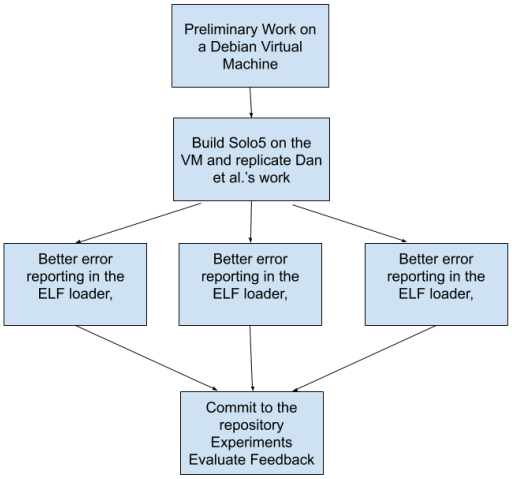

Figure 3: Project Framework

4.3 Project Issues

I am building Solo5 on my VM and replicating Dan et al.'s work in their paper[16]. The challenge of each issue is still unknown. The current three issues are chosen randomly and could be possibly out of the scope of mine. As a result, they may change according to my further investigation of the source code.

4.4 Preliminary Work

To build Solo5, I created a new Debian virtual machine on VirtualBox. Besides, configurations need to be done and some software required by Solo5 need to be installed on the VM. These include:

- Git Software
- Network Configuration to enable internet access
- A C11 compiler; recent versions of GCC and clang are supported
- GNU make
- Full host system headers (on Linux, kernel headers are not always installed by default)
- Pkg-config and libseccomp >= 2.3.3 are required
- Makefile Configuration

4.5 Experiments Design

The experiments for the project results consists of two parts. First, after commit changes to the Github, I will receive feedback from other contributors, which I will use to evaluate my project. Second, I will use the unikernels to run two applications, Atlas and Mothur, and test whether unikernels could improve their performance. Based on observations, I will discuss with other contributors to evaluate my work.

5 BUDGET

There is currently no expense for this project.

6 TIMELINE

Table 1 shows the timeline of the project.

Jan 15th - Feb 1st	Learn about the open source code
Feb 2nd -Feb 16th	Evaluate the difficulty of each issues
Feb 16th- Mar 1st	Find three issues that are feasible
Mar 1st -Mar 14th	Build a Linux VM and install Solo5
Mar 14 -Mar 28th	Find solutions to the issues
Mar 25th -April	Solve the issues and make report.
April	Add myself to the contributor list

Table 1: Timeline.

ACKNOWLEDGEMENT

I would like to thank Prof.Xunfei to for her help and advice for my project and Prof. Charlie for his suggestions on unikernels.

REFERENCES

[1] Bob Duncan, Andreas Happe, and Alfred Bratterud. 2016. Enterprise IoT Security and Scalability: How Unikernels Can Improve the Status Quo. In *Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC '16)*. ACM, New York, NY, USA, 292–297. https://doi.org/10.1145/2996890.3007875 event-place: Shanghai, China.

[2] Henrique Fingler, Amogh Akshintala, and Christopher J. Rossbach. 2019. USETL: Unikernels for Serverless Extract Transform and Load Why Should You Settle for Less?. In *Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '19)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/3343737.3343750 event-place: Hangzhou, China.

[3] Takayuki Imada. 2018. MirageOS Unikernel with Network Acceleration for IoT Cloud Environments. In *Proceedings of the 2018 2Nd International Conference on Cloud and Big Data Computing (ICCBDC'18)*. ACM, New York, NY, USA, 1–5. https://doi.org/10.1145/3264560.3264561 event-place: Barcelona, Spain.

[4] Antti Kantee. 2015. OPINION The Rise and Fall of the Operating System. 40, 5 (2015), 4.

[5] Ricardo Koller and Dan Williams. 2017. Will Serverless End the Dominance of Linux in the Cloud?. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS '17)*. ACM, New York, NY, USA, 169–173. https://doi.org/10.1145/3102980.3103008 event-place: Whistler, BC, Canada.

[6] Simon Kuenzer, Anton Ivanov, Filipe Manco, Jose Mendes, Yuri Volchkov, Florian Schmidt, Kenichi Yasukata, Michio Honda, and Felipe Huici. 2017. Unikernels Everywhere: The Case for Elastic CDNs. In *Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '17)*. ACM, New York, NY, USA, 15–29. https://doi.org/10.1145/3050748.3050757 event-place: Xi'an, China.

[7] S. Kuenzer, S. Santhanam, Y. Volchkov, F. Schmidt, F. Huici, Joel Nider, Mike Rapoport, and Costin Lupu. 2019. Unleashing the Power of Unikernels with Unikraft. In *Proceedings of the 12th ACM International Conference on Systems and Storage (SYSTOR '19)*. ACM, New York, NY, USA, 195–195. https://doi.org/10.1145/3319647.3325856 event-place: Haifa, Israel.

[8] Stefan Lankes, Simon Pickartz, and Jens Breitbart. 2016. HermitCore: A Unikernel for Extreme Scale Computing. In *Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '16)*. ACM, New York, NY, USA, 4:1–4:8. https://doi.org/10.1145/2931088.2931093 event-place: Kyoto, Japan.

[9] Anil Madhavapeddy, Thomas Leonard, Magnus Skjegstad, Thomas Gazagnaire, David Sheets, Dave Scott, Richard Mortier, Amir Chaudhry, Balraj Singh, Jon

Ludlam, Jon Crowcroft, and Ian Leslie. [n.d.]. Jitsu: Just-In-Time Summoning of Unikernels. ([n. d.]), 15.

[10] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: Library Operating Systems for the Cloud. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13)*. ACM, New York, NY, USA, 461–472. https://doi.org/10.1145/2451116.2451167 event-place: Houston, Texas, USA.

[11] Anil Madhavapeddy and David J. Scott. 2014. Unikernels: The Rise of the Virtual Library Operating System. *Commun. ACM* 57, 1 (Jan. 2014), 61–69. https://doi.org/10.1145/2541883.2541895

[12] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. 2017. My VM is Lighter (and Safer) Than Your Container. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. ACM, New York, NY, USA, 218–233. https://doi.org/10.1145/3132747.3132763 event-place: Shanghai, China.

[13] Pierre Olivier, Daniel Chiba, Stefan Lankes, Changwoo Min, and Binoy Ravindran. 2019. A Binary-compatible Unikernel. In *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2019)*. ACM, New York, NY, USA, 59–73. https://doi.org/10.1145/3313808.3313817 event-place: Providence, RI, USA.

[14] Ali Raza, Parul Sohal, James Cadden, Jonathan Appavoo, Ulrich Drepper, Richard Jones, Orran Krieger, Renato Mancuso, and Larry Woodman. 2019. Unikernels: The Next Stage of Linux's Dominance. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS '19)*. ACM, New York, NY, USA, 7–13. https://doi.org/10.1145/3317550.3321445 event-place: Bertinoro, Italy.

[15] Florian Schmidt. 2017. Uniprof: A Unikernel Stack Profiler. In *Proceedings of the SIGCOMM Posters and Demos (SIGCOMM Posters and Demos '17)*. ACM, New York, NY, USA, 31–33. https://doi.org/10.1145/3123878.3131976 event-place: Los Angeles, CA, USA.

[16] Dan Williams, Ricardo Koller, Martin Lucina, and Nikhil Prakash. 2018. Unikernels As Processes. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC '18)*. ACM, New York, NY, USA, 199–211. https://doi.org/10.1145/3267809.3267845 event-place: Carlsbad, CA, USA.