UniKernel – developing a minimized OS poster presentation

Jian Wang
4708 Marshall Dr. W, Vestal, NY 13850

Introduction

According to unikernel.org, UniKernel is specialized, single-address-space machine images constructed by using library operating systems. They provide minimized cores for applications to increase performance speed. Compared with previous operating system architecture, UniKernel has less compatibility with applications: a UniKernel image supports one complex application.

Here we introduce Lupine Linux, a UniKernel based on Linux, to show how UniKernel solves all problems.

UniKernel Architecture

In UniKernel, one platform or application on one image which we focus on has the most priority. Furthermore, if one application is running in user space it has limited privilege. Therefore, a simple and effective method to solve this problem is directly running the application in privilege mode and eliminating user space. In privilege mode, privilege and resource scrambling behaviors are appropriately guaranteed.

Because all behaviors are done in privilege mode, it allows operating systems the opportunity to eliminate system calls. Though risk may increase when we diminish an abundance of functions from the kernel, it can be prevented, saving time. For cloud computing platforms, since every guest OS uses the same drivers, kernels for UniKernel standard images can be similar. For the cloud computing industry, UniKernel solves substantial problems which data centers face every day.

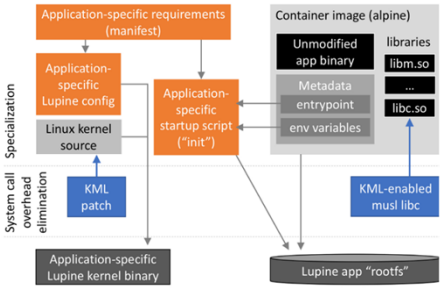

Figure 1 Lupine Linux architecture

In Lupine Linux, the Kernel Module Linux patch is used to eliminate user space. When an application is running in the root, it has the highest privilege, but kernel modules and drivers have the same privilege as the application while running. With the patch, Lupine Linux runs applications in kernel mode. As a result, system calls and user space are eliminated. Thurs, image size is reduced. Figure 1 shows the architectures of Lupine Linux, an example of UniKernel.

Design Your Lupine Linux

To design your own image, you need to modify the Linux kernel and disk files separately.

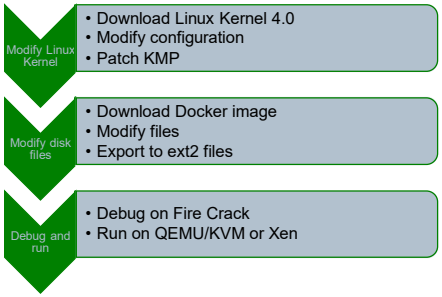

Figure 2 Steps to build Lupine Linux.

Build Your Lupine Linux

Lupine Linux provides shell scripts officially.
https://github.com/hckuo/Lupine-Linux

Setup

1. Clone project: git clone https://github.com/hckuo/Lupine-Linux.git
2. Update submodule: git submodule update –init
3. Pull a docker image that contains the environment for building Linux 4.0 kernel, and create a tag linuxbuild:latest. Run the below commands:

```
docker pull a74731248/linuxbuild:latest
docker tag a74731248/linuxbuild:latest linuxbuild:latest
```

4. Run make command in the content load_entropy to generate load_entropy file
5. Build the Lupine unikernel of your interest by following one of the below steps:

```
sh scripts/build-kernels.sh
sh scripts/build-with-configs.sh configs/<specific_config> <app_config>
```

For Eg:

```
sh scripts/build-with-configs.sh configs/lupine-djw-kml.config configs/apps/nginx.config
```

6. Build rootfs:

```
sh image2rootfs.sh app tag ext2, the tag must be alpine, because Lupine use musl libc rather than glibc.
```

Example:

```
sh image2rootfs.sh nginx alpine ext2
```

Try Your Lupine Linux

In Firecracker

```
run sh firecrackerd.sh in first shell
run sh firecracker-run.sh <path_to_kernel> <path_to_rootfs> init=<init_script> in second shell.
```

The init can be /bin/sh or some scripts that you want to run after Lupine boots up.

Evaluation

Lupine OS boots considerably faster than any virtual machine. The Lupine Linux official team published data and charts to prove increased performance.
The following graph shows the boot time of the Lupine Linux "Hello World" image.

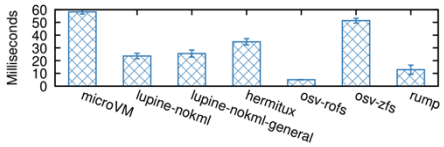

Figure 3 Boot time for hello world.

Another important indicator that is used to measure performance is memory occupation. As seen in Figure 3, which shows memory footprint comparison.

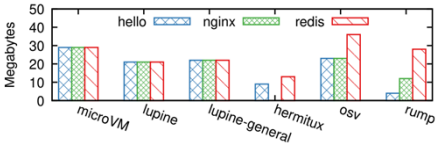

Figure 3 Memory footprint.

Lupine Linux costs 2/3 of memory resources as the micro VM. The system call is the most important feature of the Linux operating system, but it also causes crucial overhead. Figure 4 shows system call latency.

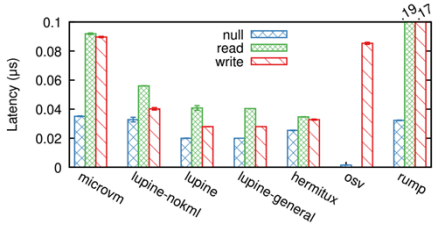

Figure 4 System call latency via lmbench.

Lupine Linux clearly states increased performance when it is compared with virtual machines. Resource consumption is reduced. Moreover, with the help of Kernel Mode Patch, Lupine Linux eliminates overhead successfully.

Something still need to dig

Build Image
Steps for Lupine Linux to build an image are still time-consuming. Here are several cons:

- Linux Kernel 4.0-rc1 must be used but is supported by the official shell scripts, which is in beta.
- Images must be used but are officially supported to export Docker™ images as disk files.

Architecture Support
Lupine Linux only supports limited hardware and software architecture officially.
For hardware and virtualization platforms, the con is that the following must be used:

- X86 hardware architecture due to Kernel Mode Patch.
- QEMU/KVM-related virtualization platforms.

For software-related functions, it has the following disadvantages:

- The ext2 file system must be used but is supported officially as the disk file format.
- Linux Kernel versions lower than 4 are officially supported due to Kernel Mode Patch being no longer upgraded, but Linux Kernel version 5 series are reaching end of life soon.

Conclusion

Lupine Linux is good at performance, but still needs to be developed. It is not friendly for new users of the Linux kernel.

In addition, it still does not have official support on different hardware architectures, which needs to be developed until becomes a product.

For industry, Lupine Linux can easily achieve success as a customized product. It runs fast and is lightweight. Because customers cannot reach the backend of machines and instances, they can accept any shape of products as if they can run fluently or even faster than traditional products. Since all hardware resources are uniform, maintenance jobs are simplified on Lupine Linux.

Contact information

Jian Wang

4708 Marshall Dr. W
Vestal, NY 13850

T: 607 222 3023
E: jwang375@binghamton.edu