Java & Unikernels
Run What you Need

August 17th, OJUG
by Aaron Grothe

Introduction

What is a Unikernel?

A Unikernel is a single address space program. It combines what we normally think of as an operating system and a program into one running executable. So it can usually be run on a hypervisor or bare metal.

There are several different approaches to accomplish this we'll talk about them a bit more later in this talk.

Introduction (Continued)

If you have questions/comments please feel free to ask them anytime. You don't have to hold them until the end of the talk.

If there are other resources similar to these that you think might be useful to people please let the group know.

Hopefully this will be an interactive and productive session.

OPS.city

Let's build a unikernel to show how it goes

We'll use ops.city's product for this since it is pretty easy to use.

Ops is an orchestration layer that runs on top of the nanovms unikenel infrastructure

How to install

```
% curl https://ps.city/get.sh -sSfL | sh
```

This will install the basic components for the nanovms software

OPS.city - (cont'd)

Now we need an application

We'll compile a spring/boot echo example:

Ref for the program

https://github.com/raonigabriel/spring-echo-example

OPS.city - (cont'd)

Initially we'll run it locally to confirm that it works

```
% mvn spring-boot:run
```

We'll give it a quick couple of tests

```
% curl http://localhost:8080/sample.css
% curl http://localhost:8080/sample.js
```

So we've got it running locally at port 8080

We'll stop it and go on to next test

OPS.city - (cont'd)

We built it locally now we'll build it as a docker image

First need to package the jar

```
% mvn package
```

Now build the docker package

```
% sudo docker build --no-cache -f ./Dockerfile . -t \
spring-echo-example:1.1.0
```

OPS.city - (cont'd)

Now it is time to run it

```
% docker run --rm -it -p 8080:8080
spring-echo-example:1.1.0
```

We'll give it a quick couple of tests

```
% curl http://localhost:8080/sample.css
% curl http://localhost:8080/sample.js
```

So we've got it running in docker at port 8080

We'll stop it and go on to next test

OPS.city - (cont'd)

Now lets go ahead and create a Unikernel

```
% ops image create -i spring-echo-example --package \
eyberg/java:19.6.625
```

So now we have a unikernel lets list the image info

```
% ops image list
```

Should see spring-echo-example with a very recent createdat

OPS.city - (cont'd)

Now it is time to run the unikernel

```
% ops pkg load eyberg/java:20.0.1 -p 8080 -c config.json
```

We'll give it a quick couple of tests

```
% curl http://localhost:8080/sample.css
% curl http://localhost:8080/sample.js
```

So we've got it running in unikernel at port 8080

OPS.city - (cont'd)

What is it running under the hood???

```
% ps -ef grep qemu
```

Yes it is fugly, but we can cut/paste all of this and run it independently as well

We'll cut and paste it to show how it can work outside of ops

OPS.city - (cont'd)

So what have we done so far???

Compiled a simple spring boot application three different ways on our system

1. Ran it locally
2. Put in a docker container and ran that version
3. Created a unikernel of the sample application and gave it a quick spin using ops
4. Ran the unikernel outside of ops as well

OPS.city - (cont'd)

Time to compare sizes

- How big is the docker environment

```
% docker image ls
```

About 132MB for the environment

Keep in mind this requires a kernel and the rest of a supporting environment, so the true size is a lot larger

OPS.city - (cont'd)

Time to compare sizes

- How big is the unikernel environment

% ops image ls

About 371Mb for the entire system

This is for a fully functioning environment. It is the O/S, the kernel, the java, the program and everything else

OPS.city - (cont'd)

Also OPS.city can do things like automatically deploy to the google cloud platform.

OPS.city also has instructions for deploying to other clouds like Digital Ocean and so on.

Some Ops example programs

Repo for bunch of ops examples

https://github.com/nanovms/ops-examples

In the java folder there are examples for the following

- regular example
- spring/boot
- vertx
- Quarkus

Some Ops example programs

Not all of these examples are complete, or working

They do give you an idea of some of the capabilities of ops

Ops.city Capabilities

Integrates with various Cloud Vendors

Azure, GCP, AWS, Digital Ocean, OpenStack, OCI, Vultr and others

Can control the whole life cycle of the program

In some ways it is an orchestration tool as well

Other Unikernels

Another option that is getting quite a bit of traction is Unik (pronounce you-neek).

Reference: https://github.com/solo-io/unik

Supports OSv - which is a general purpose unikernel, can run unmodified linux binaries, but the unik makes it a lot nicer

Types of Unikernels

- Generic Unikernels - these are able to run general programs. Can be in many different locations, other examples of this include RumpRUN
- Language Specific Unikernels - these are designed to support one specific language/runtime. E.g. Clive for Go programming language. Kind of a glorified Read-Evaluation-Print-Loop (REPL)
- Reduced O/S. An example of this is Hermitux, that is able to run Linux executables with a reduced O/S size
- Other - there are a lot of other types included as well

Some Other Unikernels

- Clive - Operating System written in Go programming language
- ClickOS - Supposed to be very fast, boot 20 milliseconds, minimal Xen OS
- HalVM - Haskell Compiler Tool Suite
- IncludeOS - library used to generate OS images, designed to run C++ code
- RumpRun - NetBSD based system
- Hermitux - takes posix executables from Linux and runs them under a reduced O/S

Unikernel Linux

- Work has been done and published proposing a set of extensions for the Linux kernel to generate Unikernels
- Interesting part of this is the approach they plan on taking as it would be a simple build of a Linux kernel with the UKL option to generate a self-contained bootable executable
- Is still very early in design, but looks very interesting. Compares to Hermitux
- Leverages off of previous designs such as User Mode Linux (UML) to help with design

Benefits of Unikernels

- Less code
- Smaller environment
- Works well in a devops type of environment
- Reduced attack surface
- Better Security?
- IoT???

Drawbacks of Unikernels

- “Unikernels are unfit for production” - article by Bryan Cantrill that is pretty tough
  - Debugging can be tough, hello printf
- Limitations of program (no memory swapping, processes)
- Can be tough to do complicated programs
- Early days, e.g. Rumprun has been unsupported for quite some time
- Once a unikernel is compromised the attacker has full privs to the environment
- No concept of UserIDs, User permissions, memory checks and so on

Security of Unikernels?

- This is complicated
- There is a really good paper on this by the NCC group titled "Assessing Unikernel Security"
  - Some security features aren't addressed in some Unikernels
    - ASLR - Address Space Layout Randomization
    - W^X, NX protection not enabled
  - The CEO of OPS.City had a blog post rebutting most of the findings of the paper
- Some of the issues are less dangerous when running on a VM since it is running in Ring 3 instead of Ring 0

History of Unikernels

- Unikernels go back to the 1990s and have been around in various forms since then
- With rise of containers and DevOPs, unikernels are being reevaulated
- Some people say that Forth is the original Unikernel
    - Still being used by Nasa :-)
    - Forth goes back to the 60s, btw

Future of Unikernels

- There will be some consolidation in this area and more startups and companies closing
- Internet of Things (IoT) offers some potential new places for the deployment of Unikernels
- When a large company announces a project using a Unikernel will be interesting
- Kubernetes plus Unikernels might be very interesting :-)

Is it ready for production?

It is worthwhile to consider for experimentation. I don't know if I would deploy real production items on a unikernel yet.

Have spent some time talking with a friend who works at a Supercomputing center about Unikernels and they may gave it a spin if they ever have the time

Q & A

Any Questions?

Thanks for listening.

Links

OPS.city

https://ops.city

Repo for examples ops examples

https://github.com/nanovms/ops-examples

Unik

https://github.com/solo-io/unik

Links

- Clive - http://lsub.org/ls/clive.html
- ClickOS - http://cnp.neclab.eu/projects/clickos/
- HalVM - http://galois.com/project/halvm/
- IncludeOS - http://www.includeos.org/
- RumpRun - http://rumpkernel.org/
- UniK - https://github.com/emc-advanced-dev/unik
- Hermitux - https://ssrg-vt.github.io/hermitux/

Links.

Assessing Unikernel Security

https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2019/april/assessing-unikernel-security/

Unikernels are unfit for production

https://www.joyent.com/blog/unikernels-are-unfit-for-production

Forth Programing Language

https://en.wikipedia.org/wiki/Forth_(programming_language)

Links.

Unikernels: The Next Stage of Linux Dominance

https://www.cs.bu.edu/~jappavoo/Resources/Papers/unikernel-hotos19.pdf