Practical course: Advanced System Programming

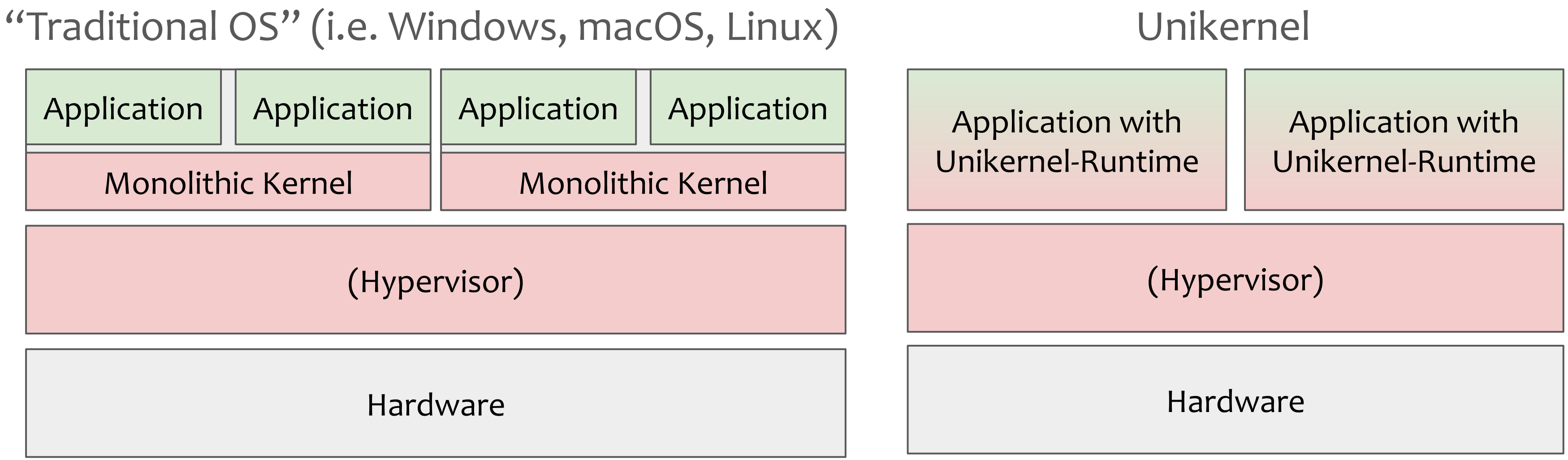
Unikernels / Unikraft

https://dse.in.tum.de/

Jörg Thalheim

Introduction - What Are Unikernels?

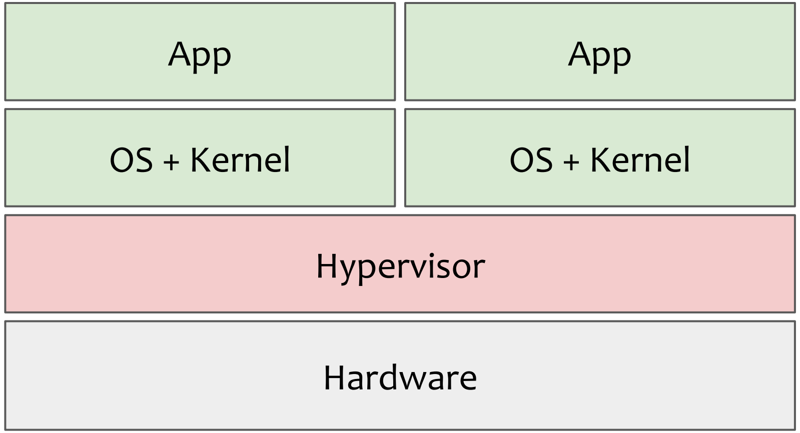
- **Definition of Unikernels:** Specialized, lightweight, and secure application images that combine the application and the operating system (OS) libraries into a single running kernel.
- **Single-Process System:** Operates by running a single application process in the space of a kernel, eliminating the need for a separate OS layer.



Next up: What was the motivation to start creating Unikernel?

The Evolution of Cloud Computing: 1. VMs

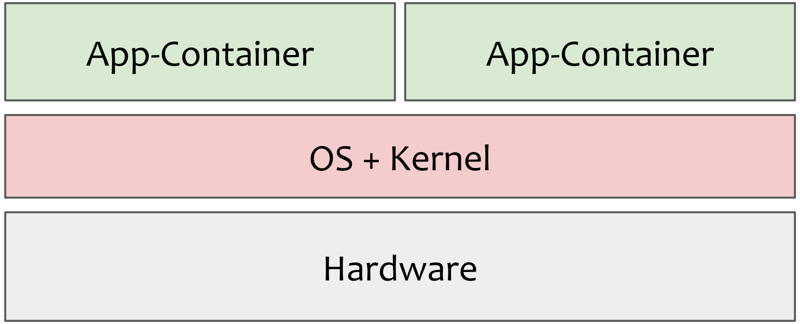
- **Definition:** Emulates a complete hardware system, running an entire operating system along with the application.
- **Characteristics:** Provides strong isolation, versatility, and is suitable for a wide range of applications.
- **Drawbacks:** Resource-intensive, with significant overhead due to the need to emulate hardware and run full OS instances.



VM: Too much overhead for applications

The Evolution of Cloud Computing: 2. Containers

- **Definition**: Packages the application and its dependencies in a virtual container that can run on any Linux server, sharing the host system's kernel.
- **Advantages**: More efficient than VMs, easier to manage, and provides faster start times.
- **Limitations**: Less isolated than VMs, dependent on the host OS's kernel, potential for security vulnerabilities.



App-Container

App-Container

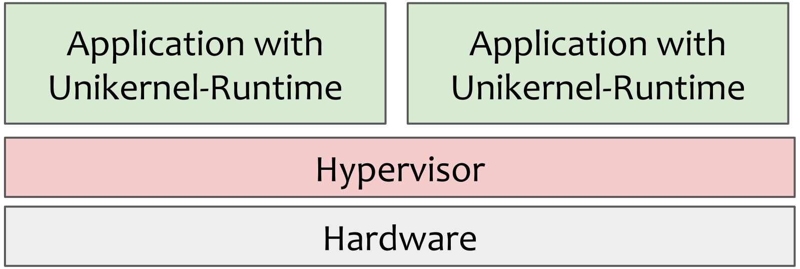
OS + Kernel

Hardware

Container: Less much overhead but also less isolation

The Evolution of Cloud Computing: 3. Unikernels

- **Emergence:** Developed as a response to the limitations of VMs and containers, aiming for even greater efficiency and security.
- **Core Concept**: Merges the application and the necessary parts of the OS into a single image that runs directly on the hypervisor or hardware, without an OS.
- **Benefits**: Extremely lightweight, fast boot times, reduced attack surface, and tailored specifically to the needs of the application.
- **Current Challenges**: Still emerging, with ongoing development in tooling, ecosystem, and adoption challenges



Application with Unikernel-Runtime
Application with Unikernel-Runtime
Hypervisor
Hardware

Unikernel: Strong Isolation and little OS-Overhead

Unikernels concepts

Unikernels: “specialized, single-address-space machine images constructed by using library operating systems” (unikernel.org).

- specialized: each image *can* fit the OS services to the application instead of relying on general purpose choices.
- single-address-space: assumes the virtual machine runs only one application or a collaborating application and simplifies the OS.
- library OS: co-locate the application and the OS at the privileged execution level effectively turning a system call into a simple library call.

Next up: When and when not to use Unikernels?

Advantages of Unikernels

- **Enhanced Performance:**
  - Fast Boot Times, Resource Efficiency, Optimized Execution
- **Robust Security:**
  - Reduced Attack Surface, Strong Isolation, Immutable Infrastructure
- **Simplicity and Minimalism:**
  - Less Complexity, Ease of Deployment, Customization

Disadvantages of Unikernels

- **Limited Tooling and Ecosystem:**
  - Emerging Tools Community and Support
- **Compatibility and Portability Issues:**
  - Operating System Services, Hardware and Platform Support
- **Development and Operational Challenges:**
  - Steep Learning Curve, Debugging Difficulties
- **Scalability and Management:**
  - Orchestration and Management, Monitoring and Logging
- **Application Suitability:**
  - Not One-size-fits-all, Migration Effort

Some of disadvantages might be resolve itself as unikernel become more mature (Unikraft)

Understanding Unikraft: A Specialized Unikernel Project

A project under the Linux Foundation, dedicated to simplifying the process of building unikernels through a modular, customizable approach.

- **Key Objectives**
  - Simplification
  - Accessibility
- **Core Features**
  - Modularity
  - Compatibility
  - Performance Optimization
- **Build Process**
  - Customizable Builds
  - Automated Tooling
- **Use Cases:**
  - Cloud Services

Unikraft aims to provide a modern implementation baked by robust tooling

Working with Unikraft - Hello World example

```
# 1. Kraftfile
specification: v0.6
unikraft: stable
libraries:
  musl: stable
targets:
  - name: default
    architecture: x86_64
    platform: qemu
    kconfig:
libraries:
    musl:
    version: stable
```

```
# 2. Makefile.uk
$(eval $(call addlib,apphelloworld))

APPHELLOWORLD_SRCS-y += $(APPHELLOWORLD_BASE)/main.c
```

Working with Unikraft - Hello World example

```
// 3. main.c
#include <stdio.h>

int main(int argc, char **argv) {
  printf("Hello world\n");
  return 0;
}


# 4. Build and Run

$ kraft build
$ kraft run
Oo   Oo  ___ (_) | __ __  __ _ ' _) :_
oO   oO ' _ `| | |/ /  _)' _` | |_|  _)
oOo oOO| | | | |   (| | | (_) |  _) :_
 OoOoO ._, ._:_:_,\_._,  .__,_:_, \___)
             Prometheus 0.14.0~2565209
Hello world!
```

Task 1: Implementing a New System Call

Motivation:

- System calls, typically implemented as interrupts in general-purpose operating systems, are function calls in Unikraft.
- The syscall ABI in Unikraft aims for Linux compatibility to ease porting, but currently lacks the reboot system call (https://man7.org/linux/man-pages/man2/reboot.2.html).

Task:

- Create a system call to reboot the virtual machine

Goal:

- Get comfortable with extending unikraft
- Learn a bit about x86 internals and where to find information i.e. how rebooting works

Task 2: Developing a GDB Stub

Motivation:

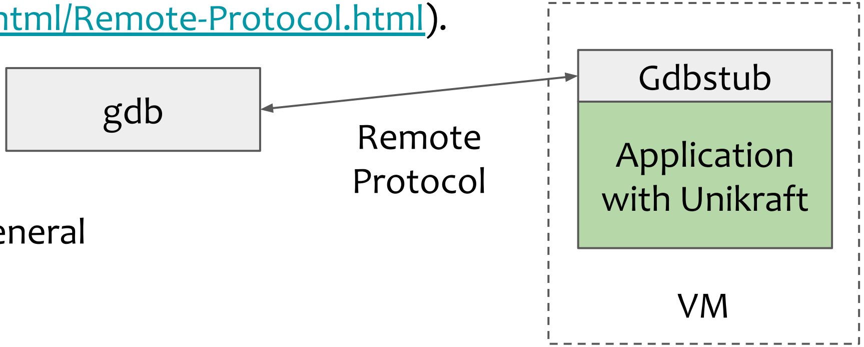
- Currently the own way to attach a debugger to unikraft apps, is by using QEMU's own gdb integration
- Downside: Does not work with multiple core application, other hypervisors or bare-metal

Task:

- Enable debugging of the operating system using console input/output.
- Implement a gdbstub based on the GDB Remote Serial Protocol (https://sourceware.org/gdb/current/onlinedocs/gdb.html/Remote-Protocol.html).

Goal:

- Learn how interrupts work
- Get some insights how GDB and debuggers work in general



Summary

- **What Are Unikernels?**
- **Advantages/Disadvantages of Unikernels**
- **Example for Unikernels: Unikraft**
- **Tasks:**
  - **Task 1: Implementing a New System Call**
  - **Task 2: Developing a GDB Stub**