Unikernel support for the deployment of light-weight, self-contained, and latency avoiding services

University of St Andrews

Ward Jaradat, Alan Dearle, and Jonathan Lewis

Overview

- Motivation
- Goals
- Unikernels
- Xen
- MiniOS
- Stardust
- Demo
- Future research directions

Motivation

We seek to construct distributed systems dynamically from interconnecting application services

Motivation

In such systems, long network latency and high bandwidth present performance challenges especially if large datasets have to be transmitted across the network

Motivation

One solution is to deploy services (i.e. computation) closer to data sources*, **but this solution is itself hindered by the network latency, bandwidth, and the size of the deployments**

Jaradat, W., Dearle, A. and Barker, A., 2016. Towards an autonomous decentralised orchestration system. *Concurrency and Computation: Practice and Experience, 28*(11), pp.3164-3179.

Goals

- Construct light-weight and immutable components that encapsulate services

- Be able to deploy such components rapidly at the most appropriate geolocations

- Be able to compose such components using typed communication channels specified in a high-level language

- Be able to orchestrate such components dynamically

Unikernels

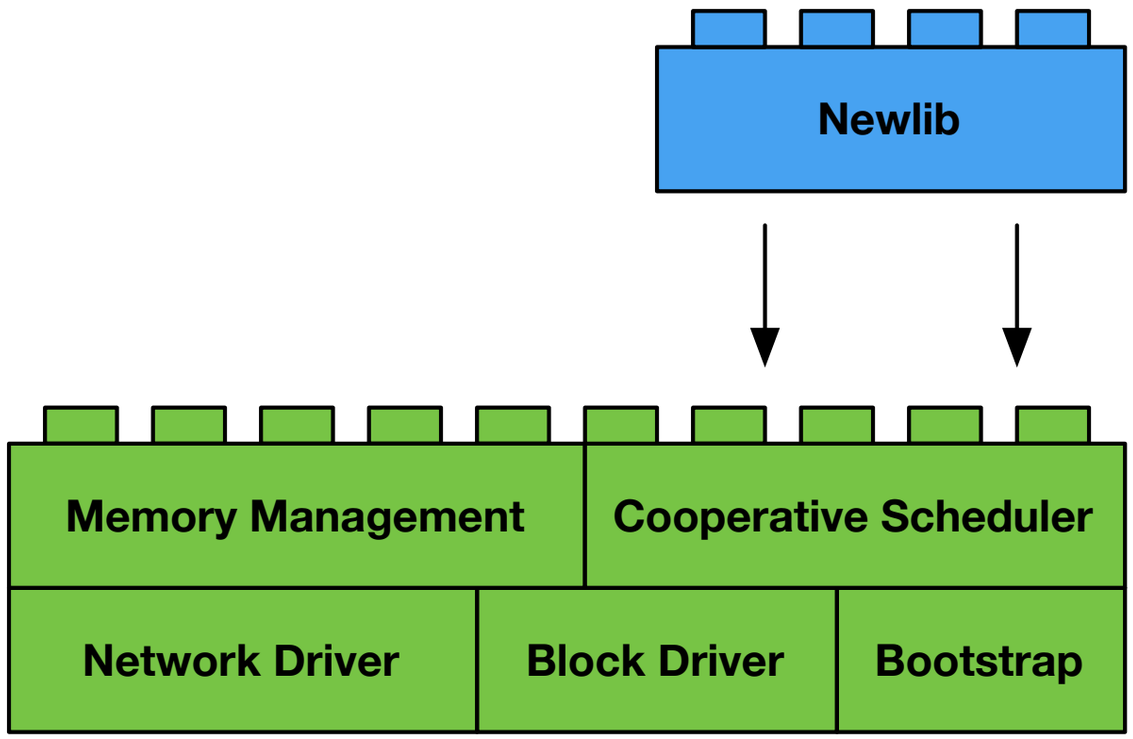
- Small library operating systems
- Support static linking of applications against the kernel
- Provide a single-address space execution environment
- Permit services to be treated as first class components that can be composed easily and deployed dynamically

Xen

- Unikernels typically run on Xen
- Xen is a hypervisor that permits multiple virtual machines to execute on the same computer hardware in parallel
- Xen provides a primitive unikernel called **MiniOS**

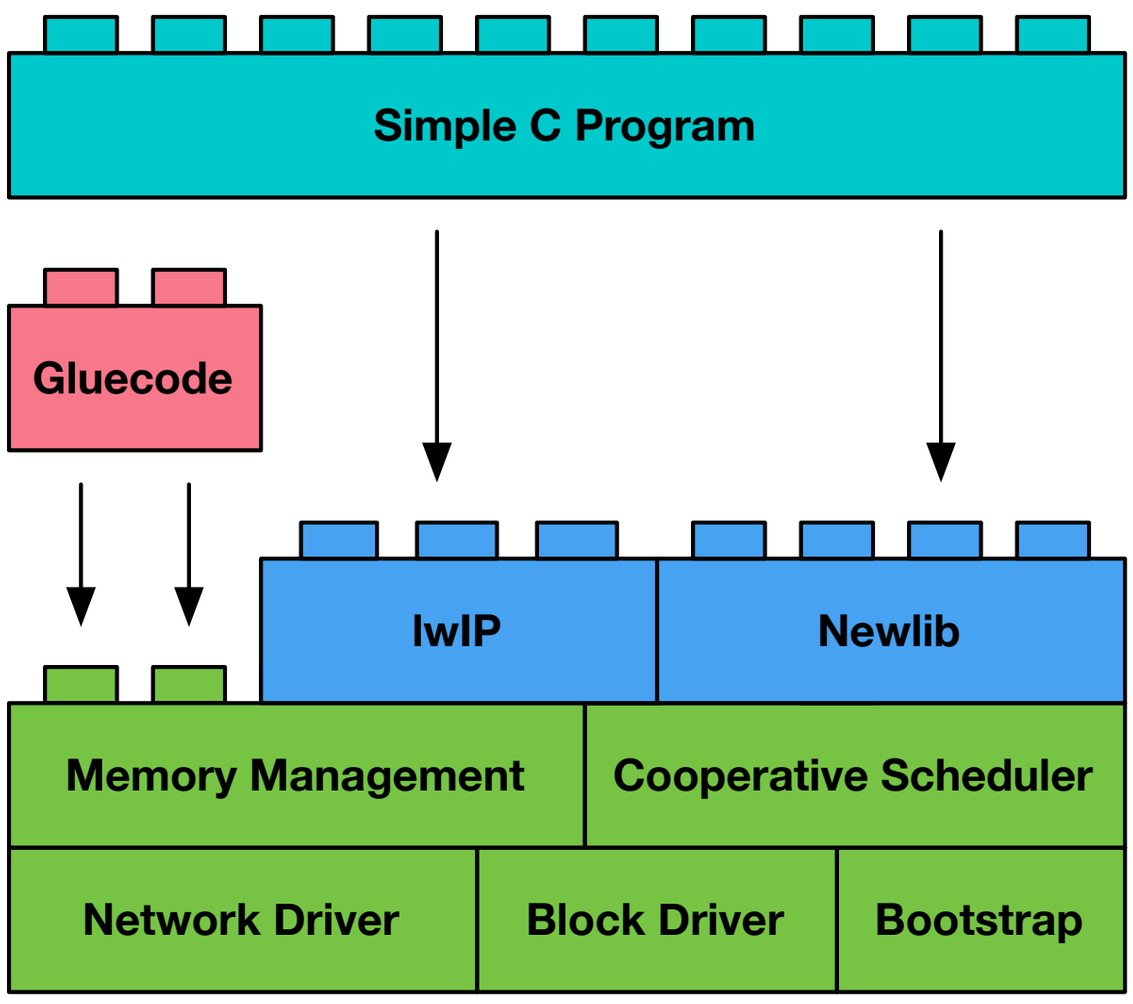
MiniOS

MiniOS provides a simple **memory management module** and a **cooperative thread scheduler**. It provides partial standard C support through its integration with **Newlib**.



MiniOS

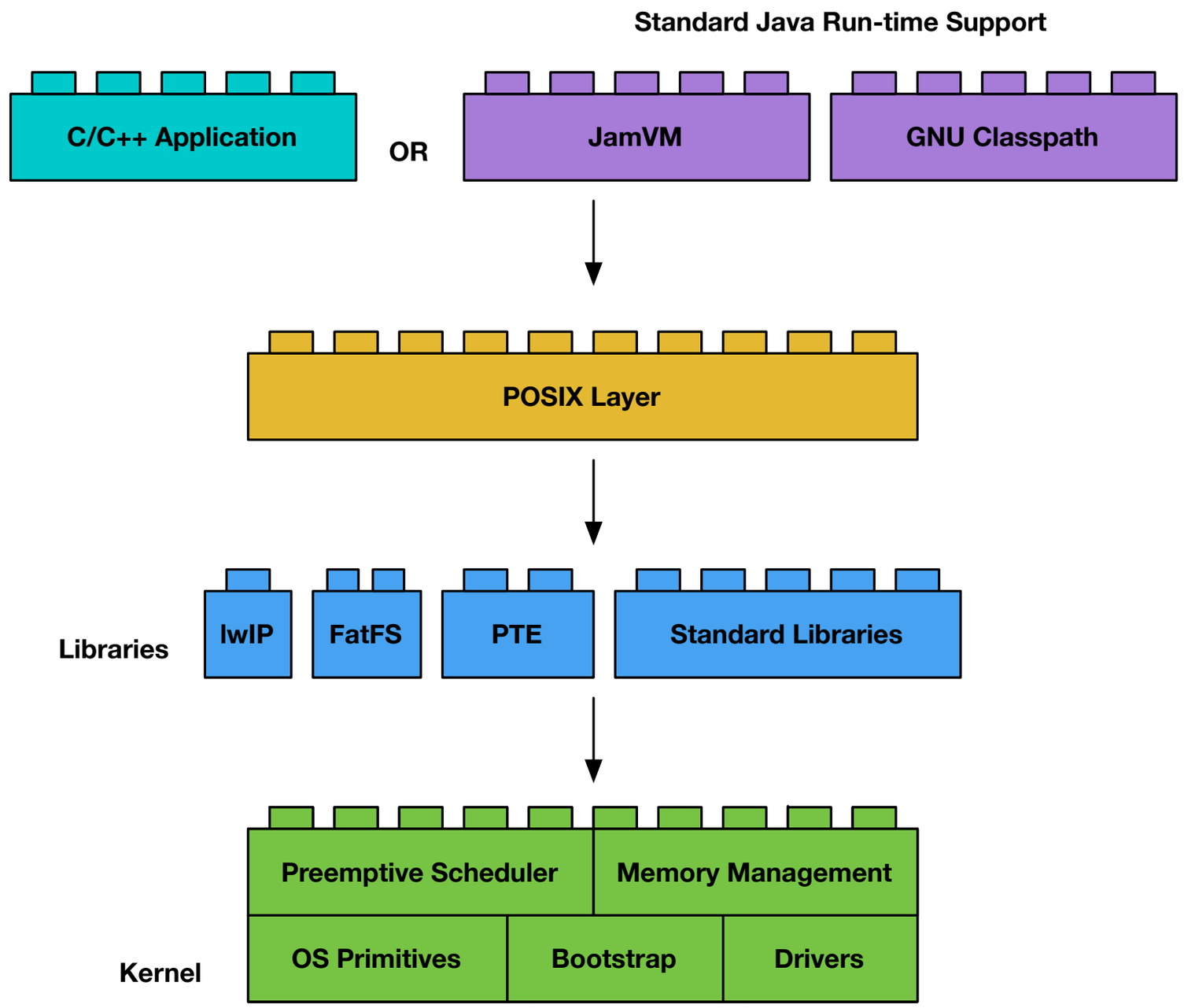
It also provides a TCP/IP network stack using **lwIP** and implements some code that enables a **program in C** to be linked against the kernel.



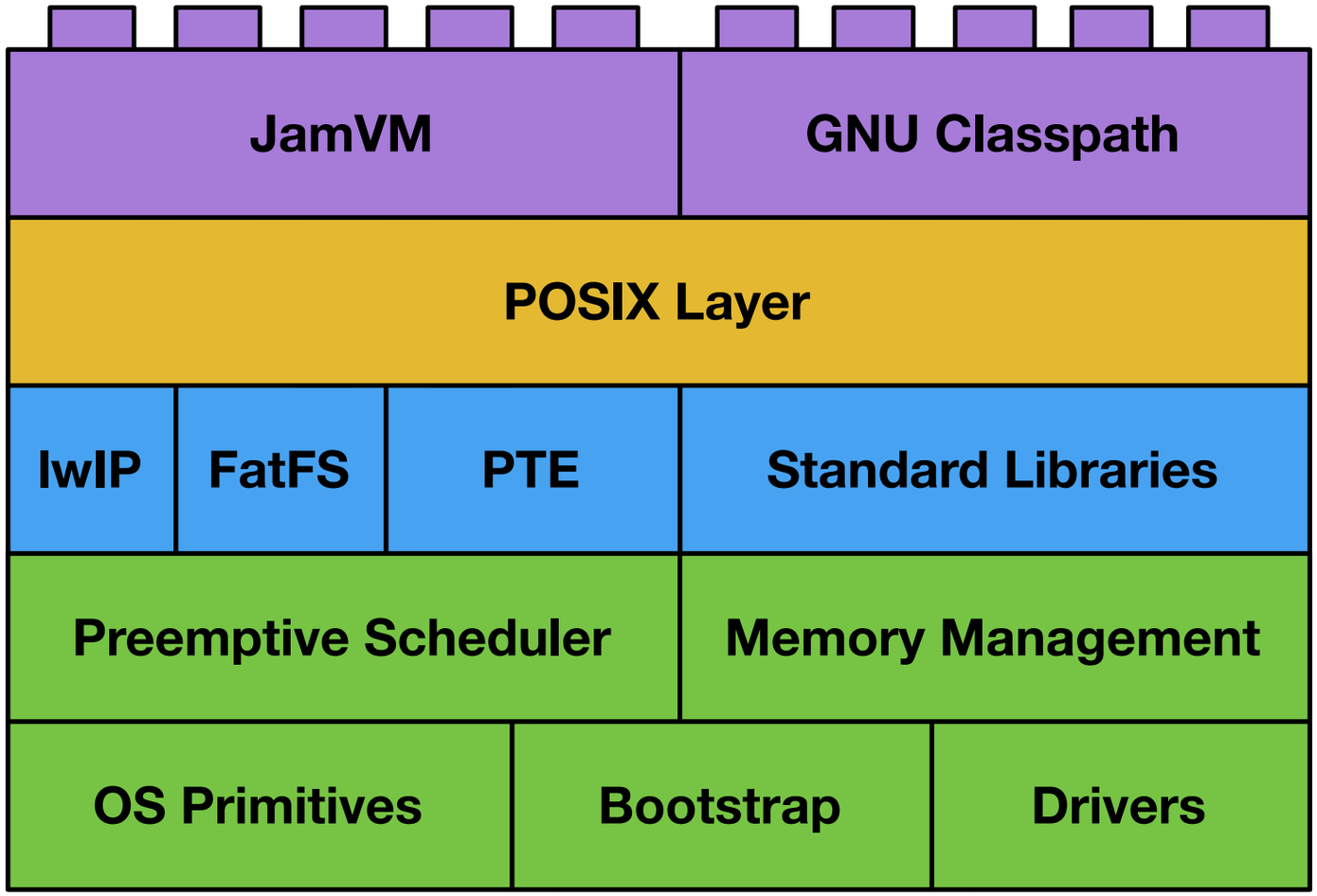
Stardust

- Stardust is a Unikernel designed to run Java applications
- It is written completely in C and has a small codebase that can be maintained easily
- It relies on static linking to combine the kernel, system libraries, and a Java virtual machine into an immutable, single-purpose virtual appliance

Stardust


Standard Java Run-time Support
C/C++ Application
OR
JamVM
GNU Classpath
POSIX Layer
Libraries
lwIP
FatFS
PTE
Standard Libraries
Kernel
Preemptive Scheduler
Memory Management
OS Primitives
Bootstrap
Drivers

Stardust


JamVM
GNU Classpath
POSIX Layer
lwIP
FatFS
PTE
Standard Libraries
Preemptive Scheduler
Memory Management
OS Primitives
Bootstrap
Drivers

Stardust

- Simple and modular design
- Supports multiple languages (C, C++, and Java)
- Limited attack surface
- Support for immutability
- Small size

Stardust

- Stardust is less than 400 KBs in size
- Stardust's disk image contents (e.g. standard .class files) are less than 20 MBs in size
- **Linux** virtual machine appliances are typically over 1 GBs
- **Containers** (e.g. BSD Jails, Docker, Linux Containers, and Ubuntu Snaps) are hundreds of MBs in size

Demo

Future Research Directions

- Benchmarking experiments
- Optimise the memory and scheduler components
- Provide an in-memory, read-only file system to support immutability (**Alastair Millican and Jonathan Lewis**)
- Support channel-based abstractions in Java
- Provide deployment and orchestration tools

Thanks!