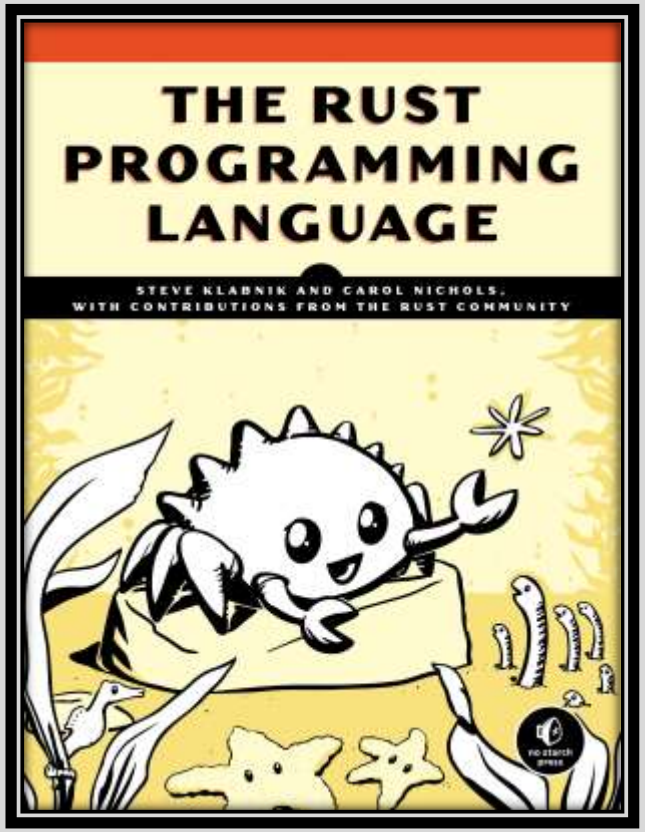
# 幺内核与模块组合式操作系统

杨德睿 · 启元实验室

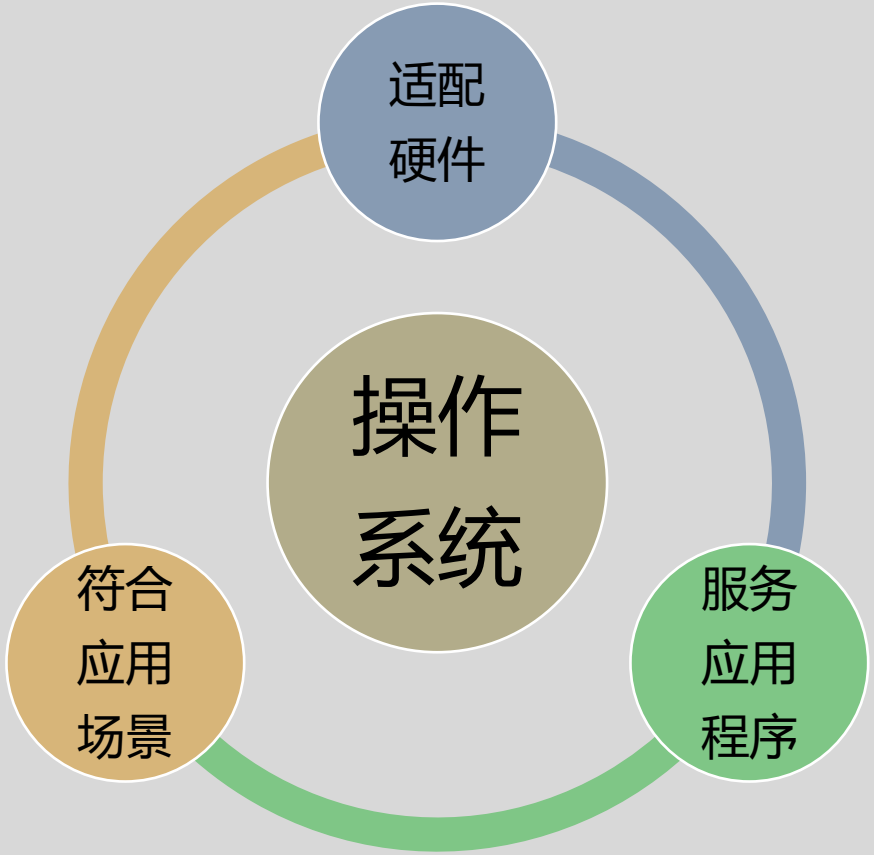
2022/10

# 序言：“实验性”

- 形式上不成熟
- 内容上不成熟
- 分享我们的“想法”
- 期待更集中的讨论
- 期待验证和反馈



# 目标：构造定制化的操作系统



# 定制开发

方法与困难性

## 独立仓库

- 直接改代码

## 并入主线

- 编译选项
- `#if`
- `#[cfg(…)]`


# 定制开发

方法与困难性

| 一般软件 | 操作系统 |
| --- | --- |
| 问题域本身的复杂性：软件是为了解决现实生活中的问题，而现实生活中存在着大量的需求。并且需求是会不断发生变化的，这种外部的复杂性导致了软件的复杂性 | 操作系统运行的硬件、服务的软件本身的多样性 |
| 管理开发过程的困难性：开发功能复杂的系统必须团队协作，但团队协作会带来更多的沟通，协调，这方面的困难也增加了软件的复杂度 | 协作开发带来的复杂性 |
| 软件中随处可见的灵活性 | 子问题没有最优解 |
| 描述离散系统行为的问题 | 为了处理大量 corner case，必须枚举出各种情况 |

# 复杂性和模块化

“复杂性是软件的基本特征。”

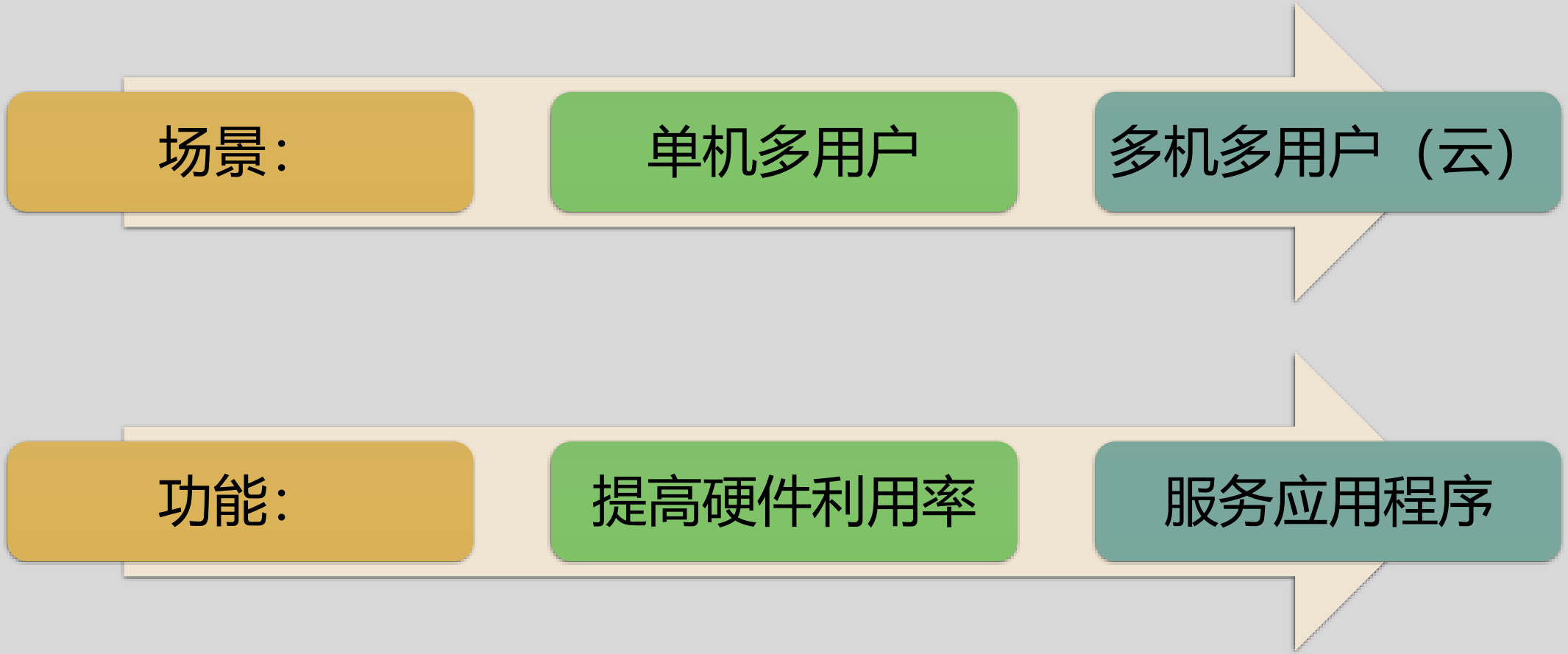
（《面向对象分析与设计》）

# unikernel

| 原文 | 译文 |
| --- | --- |
| monolithic kernel | 宏内核 |
| microkernel | 微内核 |
| unikernel | 幺内核 |

| 前缀 | 本意 | 含义 |
| --- | --- | --- |
| mono- | 宏大的；单一的 | 内核实现是一体的 |
| micro- | 微小的 | 缩小内核，内核只负责 IPC |
| uni- | 单一的 | 只能运行一个进程，不支持 IPC |

# 内核功能的变化



# uni-

# 幺内核总结

- 只能运行一个进程（广义）
- 只有一个地址空间
- 只有一个特权级
- 和应用程序（静态）链接：内核和应用程序是一个二进制文件
- 更小、更快、更安全

# unikraft：成果

- 一个新颖的微库操作系统：
  1. 完全模块化的操作系统基元，因此很容易定制幺内核并只包括相关的组件；
  2. 暴露了一套可组合的、面向性能的 API，以便使开发者容易获得高性能；
- 用现有的应用程序（如 nginx、SQLite 和 Redis）进行的评估表明：
  - 与 Linux 虚拟机相比，性能提高了 1.7–2.7 倍
  - 应用的镜像约为 1MB，运行时需要不到 10MB 的内存
  - 在虚拟机启动时间之外，启动时间约为 1ms（总启动时间为 3–40 ms）

# unikraft：设计目标

- 单一地址空间：以单一应用场景为目标，可能有不同的应用通过网络通信交流。
- 完全模块化的系统：所有组件，包括操作系统基元、驱动程序、平台代码和库，都应该易于根据需要添加和删除；甚至 API 也应该是模块化的。
- 单一的特权级：不应该有用户/内核空间的分离，以避免昂贵的处理器模式切换。这并不排斥区隔化 compartmentalization（例如微库），这可以以合理的开销实现。
- 静态链接：启用编译器功能，例如死代码消除（DCE）和链接时优化（LTO），以自动摆脱不需要的代码。
- 支持 POSIX：为了支持现有的或遗留的应用程序和编程语言，同时仍允许在该 API 下实现专用化。
- 平台抽象化：为一系列不同的 Hypervisor/VMM 无缝生成镜像。

# unikraft：组成

- 微库。微库是实现 unikraft 核心 API 之一的软件组件；我们将它们与库区分开来，因为它们具有最小的依赖性，可以是任意小的，例如一个调度器。所有实现相同 API 的微库都是可以互换的。一个这样的 API 包含多个内存分配器，它们都实现了 `ukalloc` 接口。此外，Unikraft 支持的库可以提供来自外部库项目（OpenSSL、musl、Protobuf 等）、应用程序（SQLite、Redis 等）、甚至平台（如 Solo5、Firecracker、Raspberry Pi 3）的功能。
- 构建系统。它提供了一个基于 Kconfig 的菜单，让用户选择在应用程序构建中使用哪些微库，让他们选择目标平台和 CPU 架构，甚至在需要时配置单个微库。然后，构建系统对所有的微库进行编译，将它们链接起来，并为每个选定的平台生成一个二进制文件。

# unikraft：架构

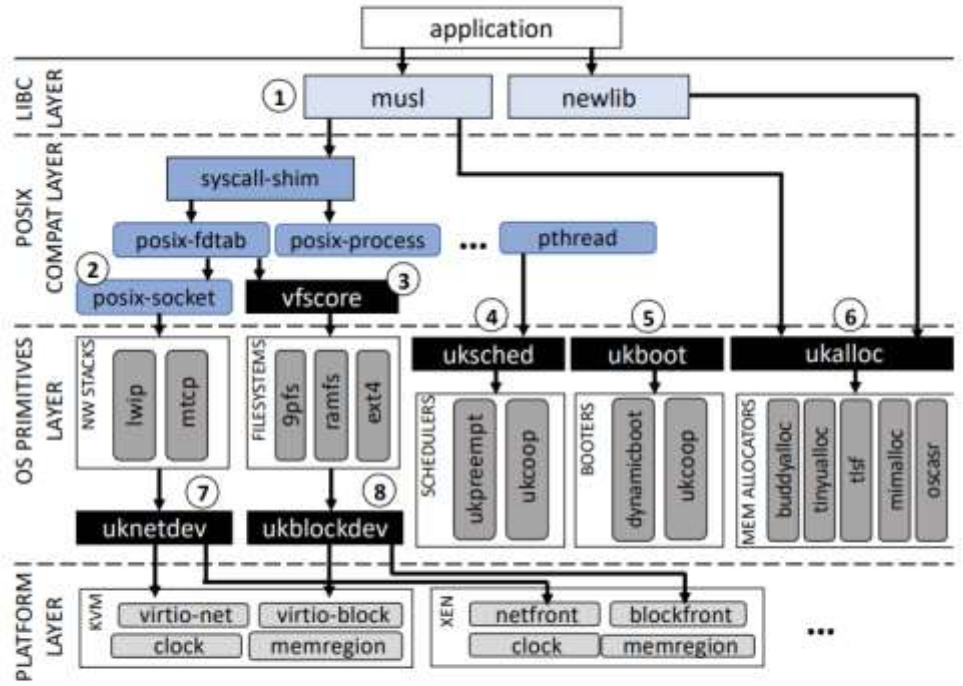



**Figure 4.** The Unikraft architecture (APIs in black boxes) enables specialization by allowing apps to plug into APIs at different levels and to choose from multiple API implementations.

# unikraft：应用程序

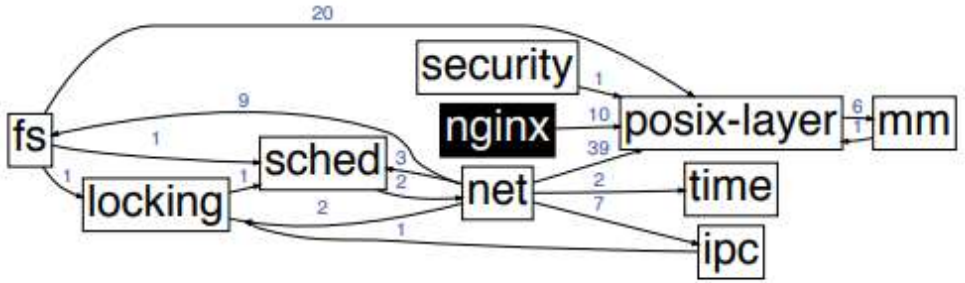



**Figure 2.** Nginx Unikraft dependency graph

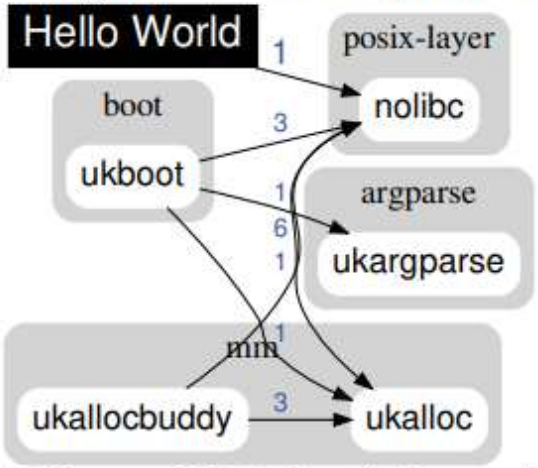



**Figure 3.** Helloworld Unikraft dependency graph

# 任务一详细描述

- 目标：实现一个 RISC-V 上的 libos 或 unikernel
- 要求：
  - 运行在 RISC-V 上
  - libos：应用程序和内核是静态链接在一起的同一个二进制文件，用函数调用替换系统调用，需要说明应用程序怎么编写
  - unikernel 指的是内核里只运行一个应用程序，需要说明如何定制内核，但不要求模块化（有更好）

# 扩展讨论

- 如何将幺内核和正常内核统一起来?


欢迎提问！