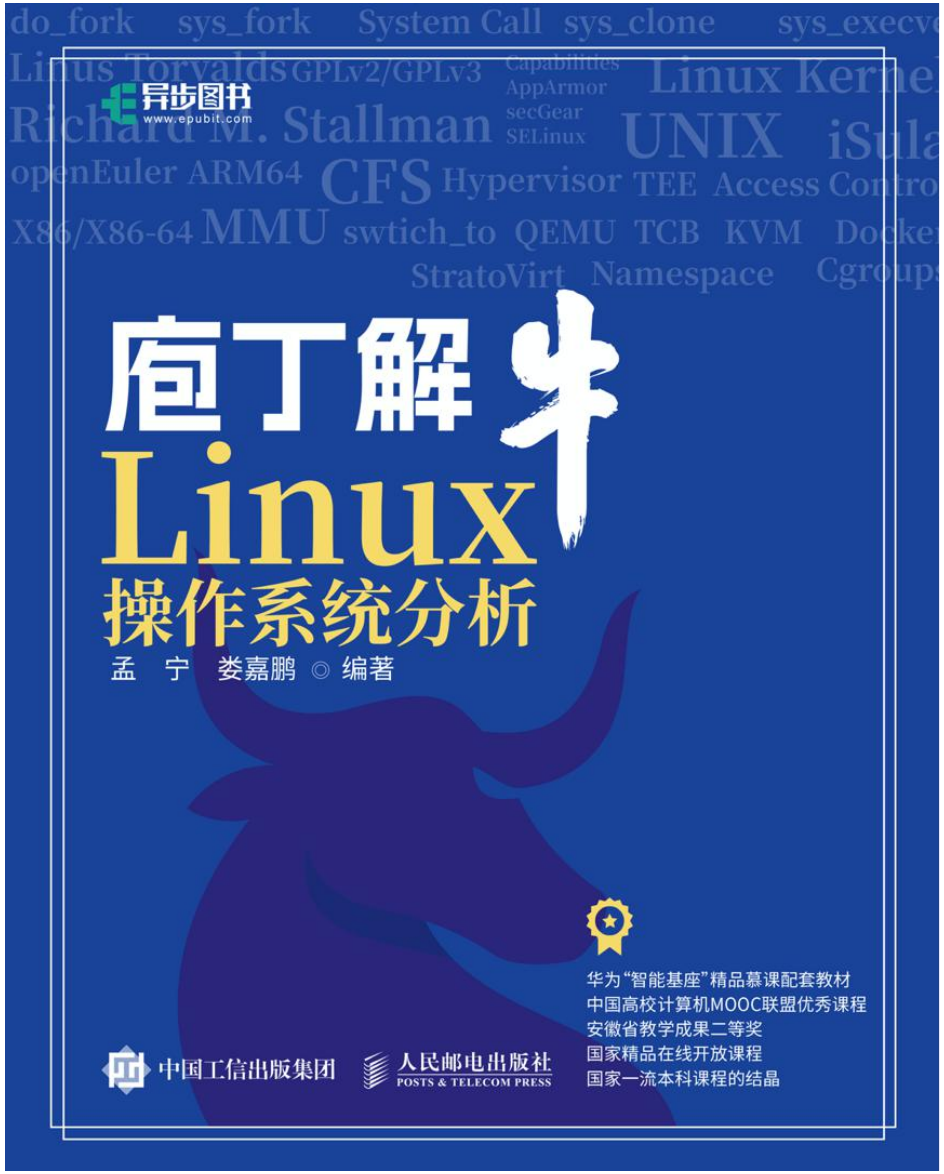
CCF ChinaSoft
CCF 中国软件大会

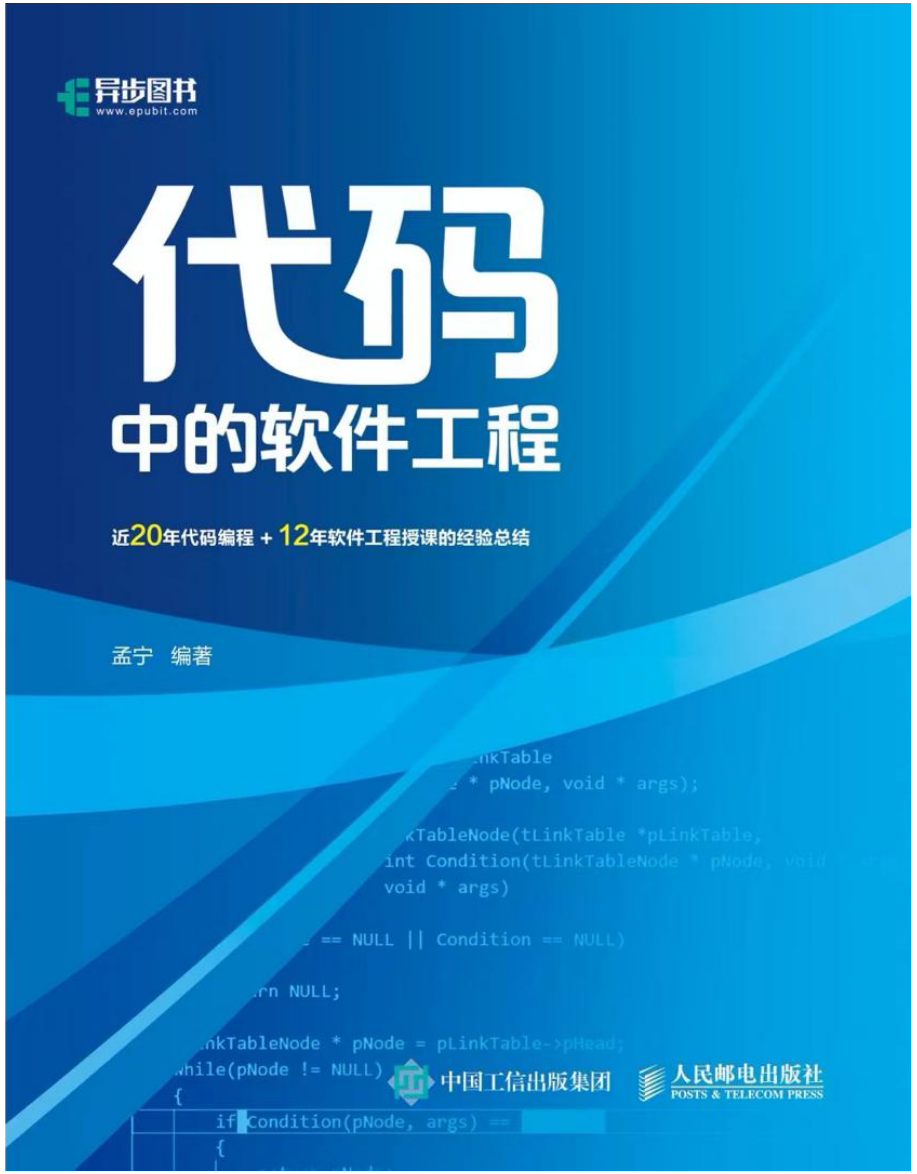
操作系统的过去、现在和未来展望

孟宁
mengning@ustc.edu.cn
系统软件研究中心
中国科学技术大学软件学院
苏州市产业技术研究院区块链技术研究所

报告日期：2022.11.27

自我介绍


异步图书
www.epubit.com
庖丁解牛
Linux
操作系统分析
孟 宁 娄嘉鹏 ◎ 编著
华为“智能基座”精品慕课配套教材
中国高校计算机MOOC联盟优秀课程
安徽省教学成果二等奖
国家精品在线开放课程
国家一流本科课程的结晶
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS


异步图书
www.epubit.com
代码
中的软件工程
近20年代码编程 + 12年软件工程授课的经验总结
孟宁 编著
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

内容目录

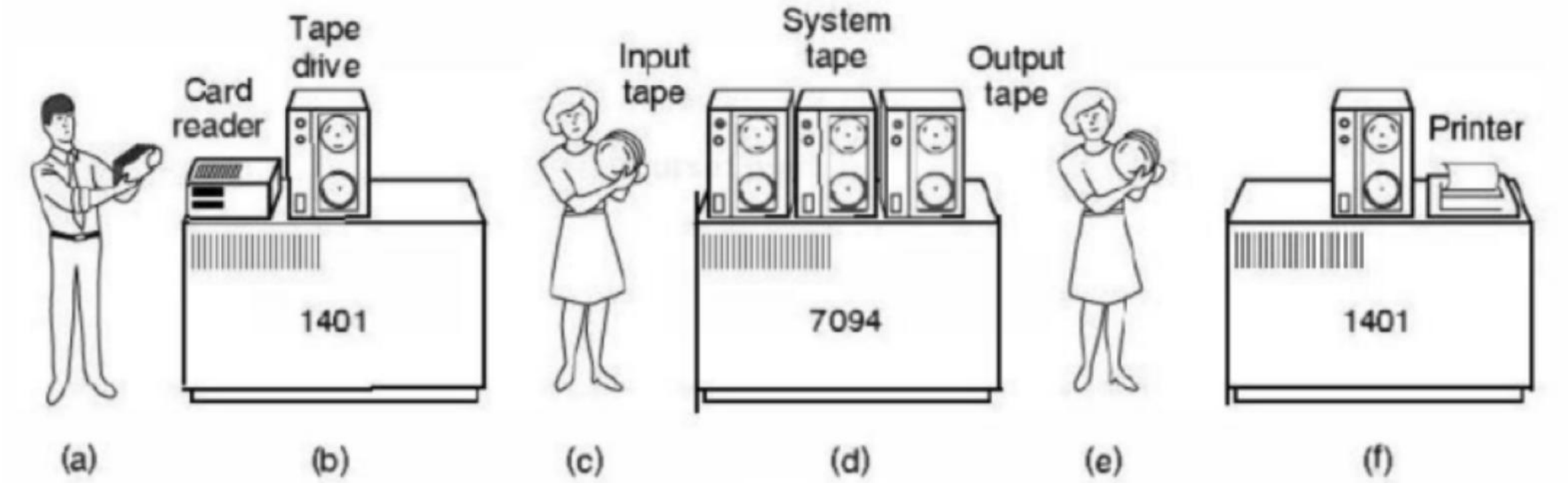
- 封闭式计算站里的操作员和批处理系统
- 分时和多道程序系统
- 系统调用——用户程序和操作系统的隔离
- 经典操作系统
- 分布式操作系统
- 操作系统面对的主要挑战
- 操作系统的未来展望

IBM 701开放式计算站(open shop)

- 最初并没有操作系统。比如在IBM 701开放式计算站(open shop)上它是由使用者手动操作的，工作效率自然十分低下，为了有效利用 IBM 701，每个用户被分配了一个最低 15 分钟的计算时间。在这 15 分钟里，使用者通常要为配置设备而花掉 10 分钟时间，等准备工作完成的时候，经常最多只有 5 分钟的时间来完成实际的计算工作，大约浪费了 2/3 的时间。每个月因浪费的计算时间造成的损失高达 14.6 万美元。要知道那时可是 1954 年！
- 为了减少计算时间的浪费并使开发人员从电脑机房中解放出来，开发人员被要求在穿孔卡上准备程序和数据，然后提交到计算中心去执行。开放式计算站(open shop)由此变成了封闭式计算站(closed shop)。

IBM 7094封闭式计算站(closed shop)

- 封闭式计算站(closed shop)里的操作员——最初的操作系统



图：IBM 7094工作流程示意图

批处理（Batch Processing）系统

- 当主计算机7094执行程序的时候，两台卫星计算机同时做着输入和输出的工作。操作员完成了现代操作系统需要做的主要工作，比如决定程序运行的顺序，也就是任务调度。如果你想让你的程序得到优先执行，你得通过操作员想办法插队。
- 这种计算模式就是批处理（Batch Processing）系统，IBM 7094所使用的共享操作系统(SHARE operating system)是早期的批处理系统。一般批处理系统先做好准备工作，然后由操作员以分批的方式来执行程序。这时的计算机还没有以交互方式使用，因为早期的计算机非常昂贵要有效利用计算机的机时，批处理方式可以最大限度地充分利用计算机机时。

分时（Time Sharing）系统

- 批处理费时费力，而且给了计算机管理员太大的权力，黑客痛恨权威的传统，很大一部分原因归咎于计算机管理员有权决定哪个程序优先运行。
- 1962年MIT的Project MAC（人工智能实验室的前身）引入了分时的概念，之前曾被称为偷时（Time stealing），因为它利用一个程序运行中间的空隙执行另外一个程序。同时电子打字机、电视机和鼠标陆续引入计算机系统作为人机交互设备，再也不用靠打孔和打印来和计算机交互。

多道程序（Multiprogramming）

- 多道程序（Multiprogramming）从根本上改变了操作系统。在大型机时代之后，进入小型机（minicomputer）的时代，这时计算机的成本更低，同时硬件功能也日趋强大。这些变化同时也使得处理器开始能够支持程序的并发执行和控制。**中断技术**能够使得一个处理器控制多道程序的并发执行，这时多道程序操作系统开始出现。
- 多道程序操作系统不是一次只运行一个程序，而是将大量程序加载到内存中并在它们之间快速切换，从而提高了 CPU 的利用率。这种切换是非常重要的，因为 I/O 访问速度很慢，在 I/O 访问过程中 CPU 空闲，因为 CPU 在等待输入数据或者等待输出完成，输入和输出虽然都在计算机内部，但是和前文所述 IBM 7094 的卫星计算机 1401 所做的工作并没有本质的区别。通过快速切换不同的用户程序，CPU的利用率可以得到大大提高。

系统调用（System Call）

- 系统调用（System Call）概念的诞生是操作系统发展的一个重要里程碑。Atlas Supervisor 是第一个提出现代操作系统诸多概念的系统，它率先采用了系统调用的工作机制。
- 系统调用将用户程序和操作系统内核之间进行了隔离，通过添加一些特殊的硬件指令和硬件状态让用户程序进入操作系统的过程更加正式、可控，这样有效保护操作系统提供的底层代码，使得整个系统更加稳定，不会让用户程序的错误造成整个系统的崩溃。
- 系统调用机制的采用使得操作系统超越了早期操作系统只是提供一些底层 API 函数库的状况，因为通过系统调用提供了有效的保护机制。

经典操作系统

- Unix家族——BSD/FreeBSD、Sun Solaris、IBM AIX、HP HPUX…
- Linux——openEuler、Debian/Ubuntu、Redhat、Android…
- 微软OS——DOS、Windows系列…
- 苹果OS——macOS、iOS

分布式操作系统

- 基于计算和存储的分布式，Hadoop、Spark、Hive…
- 基于虚拟机的分布式，Openstack、VMware vSphere、Citrix…
- 基于操作系统虚拟化容器的分布式，Kubernetes
- 基于交互方式的终端分布式，openHarmony、Fuchsia

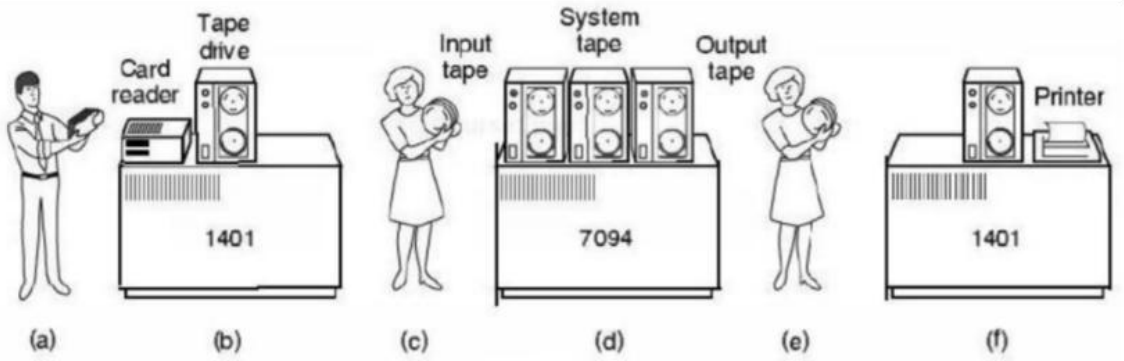
操作系统面对的主要挑战

- 万物都可以嵌入一颗芯片导致了终端的多样性；
- 异构算力蓬勃发展，比如CPU、NPU、GPU、DPU…xPU；
- 软硬件协同优化越来越重要，以CPU为中心的ISA作为软硬件标准界面受到挑战；
- 分布式管理的边界不断扩张，边缘计算、泛在计算、函数计算…

“The farther back you can look, the farther forward you are likely to see. ”——by Winston Churchill

你能看到多远的过去，就能看到多远的未来!

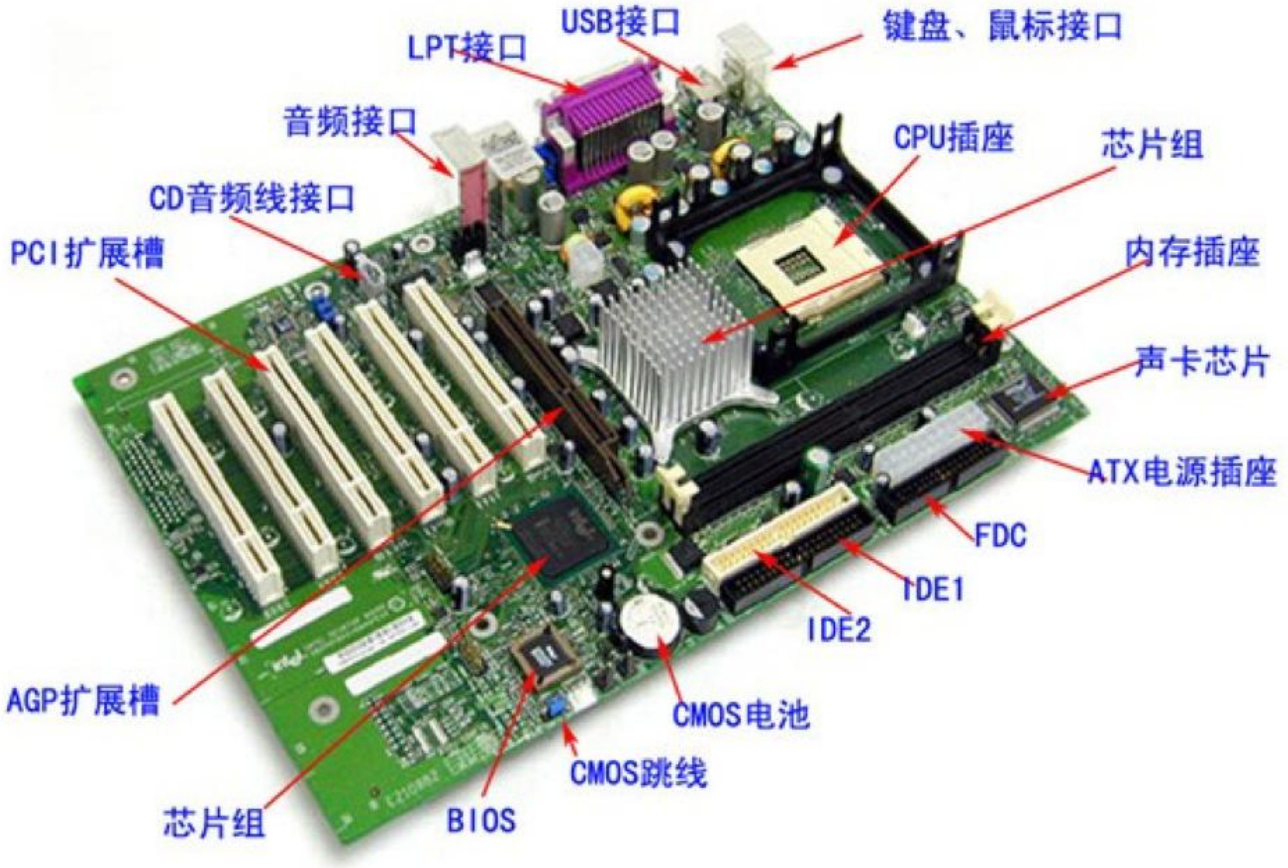
从分布式到集中分层



System Call

OS Kernel

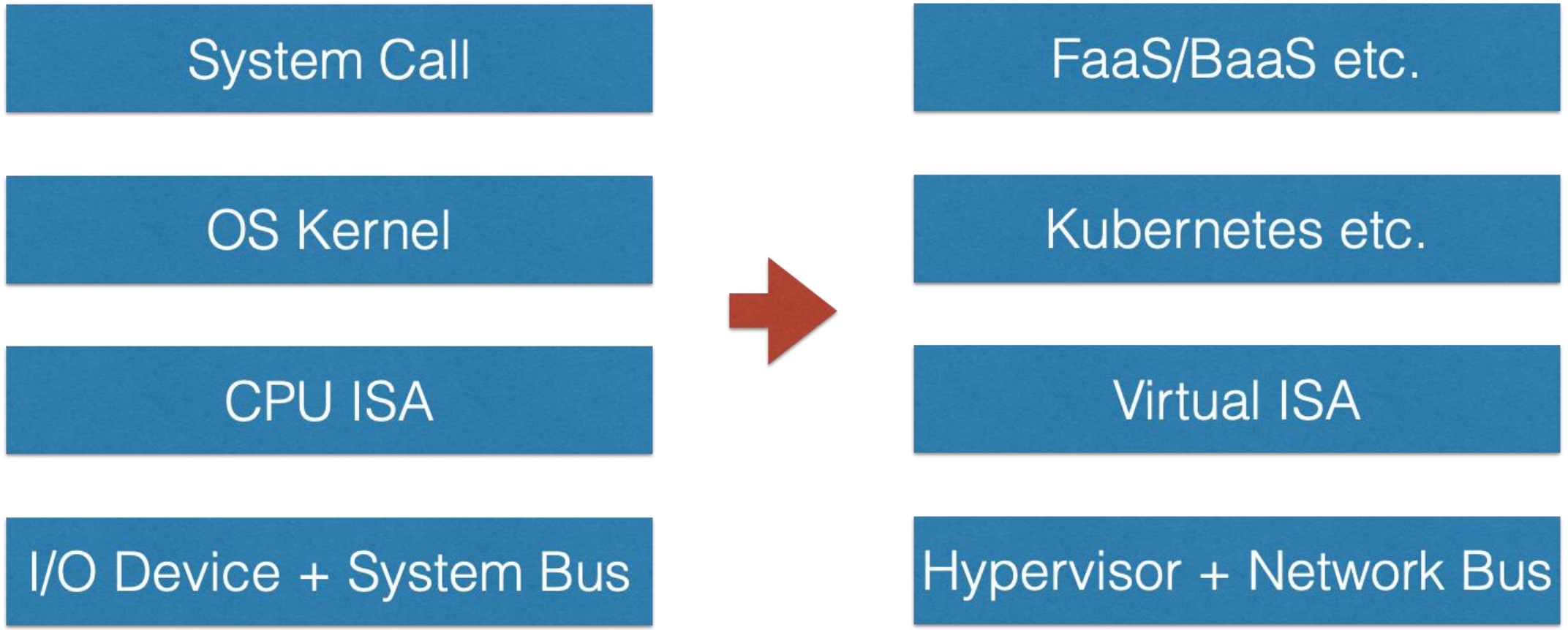
CPU ISA



System Bus

操作系统的未来展望

- 从分布式到集中分层，从分布式到下一次集中分层？



Hypervisor OS

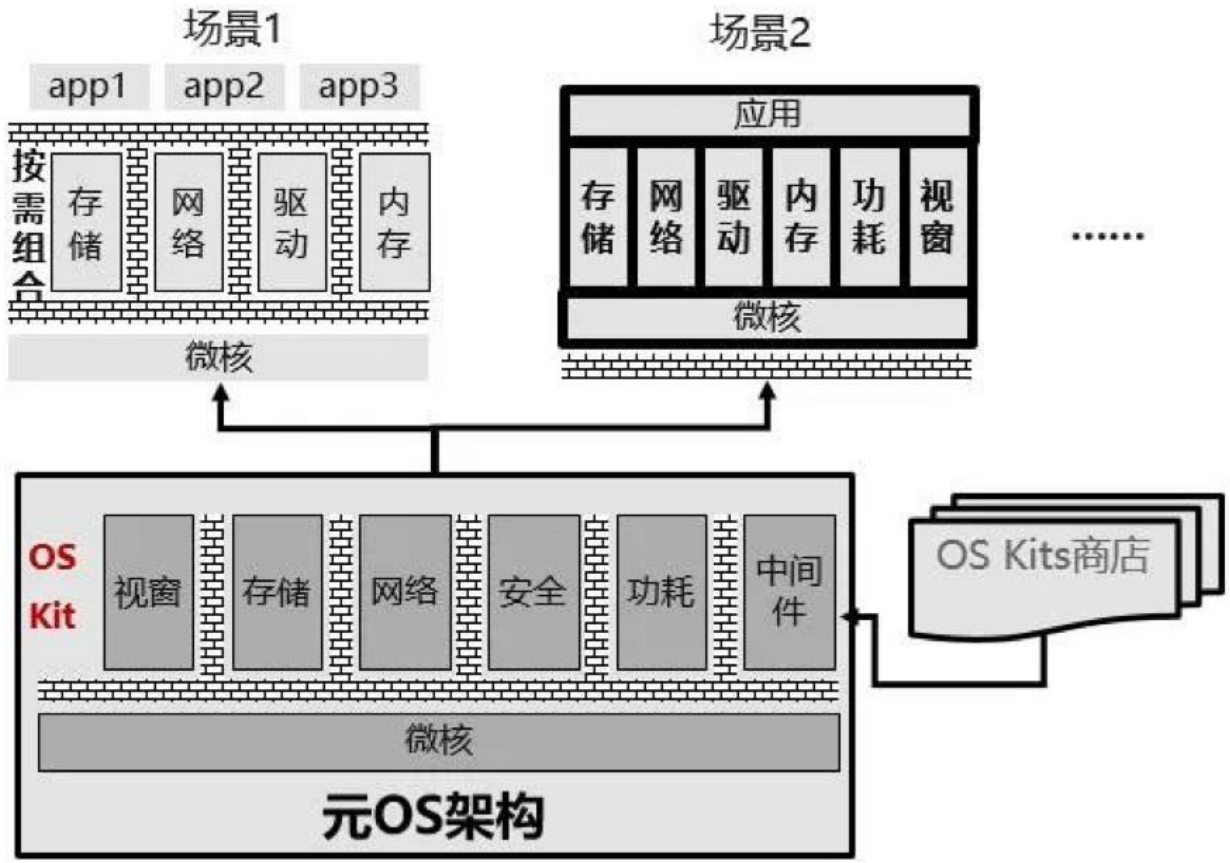
场景禁锢的操作系统

e.g. 服务器操作系统

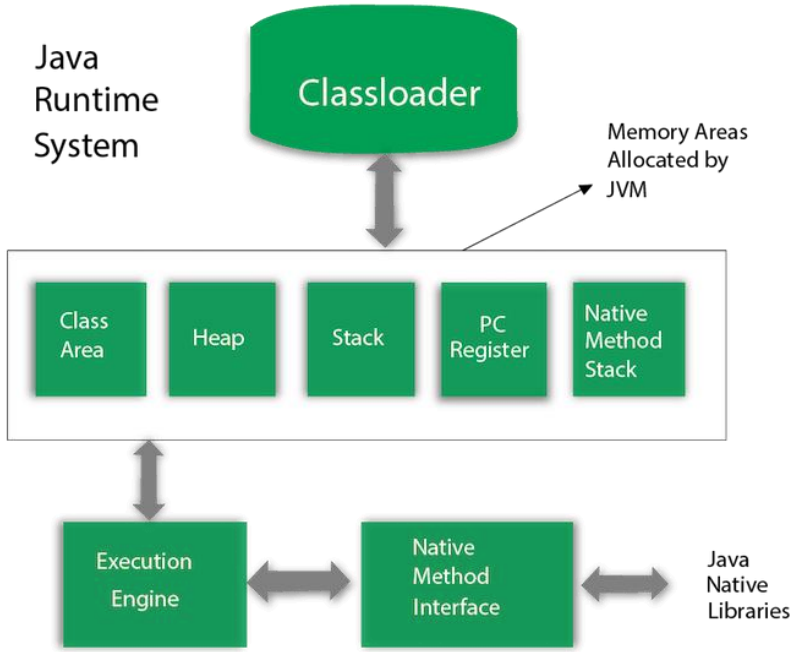
e.g. 手机操作系统

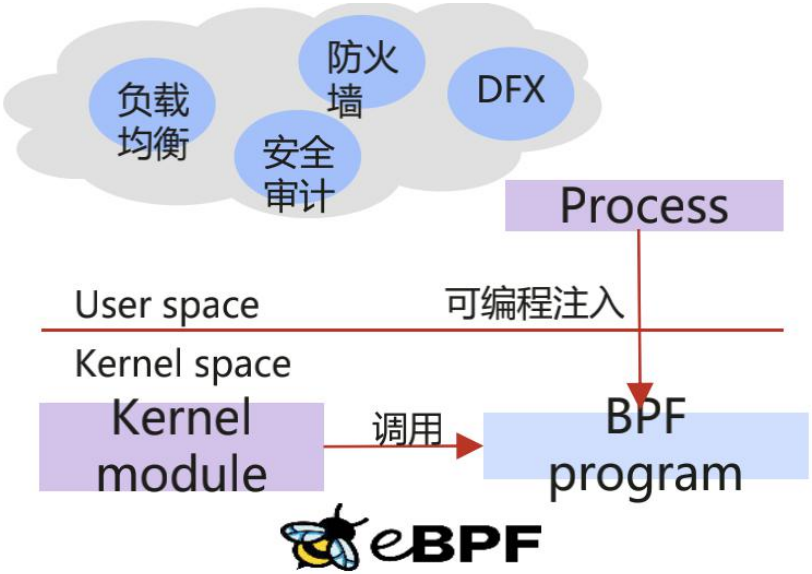
其它...越来越多种类的操作系统

元OS架构：One OS Kit for All



平衡高效、安全、易用的全系统可编程技术

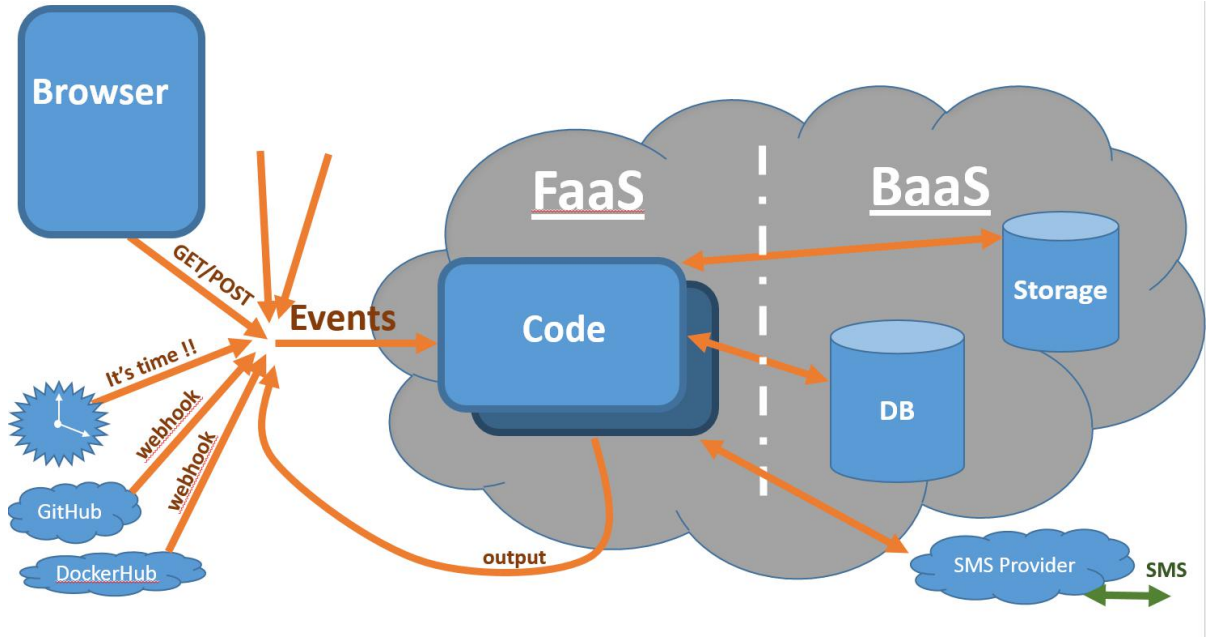




Virtual ISA

云原生落地困难重重

- Kubernetes解决了应用编排和调度自动化问题，但是Kubernetes的使用不仅要改变开发习惯，还要学习很多新概念新技术。
- 云原生落地的难点在于使用。将云原生底层的复杂技术包装成应用开发者熟悉的应用层概念至关重要。基于FaaS的应用层编程框架可能是一个有前景的演化方向。



想 都是问题

做 才有答案

站着不动，永远是观众！

面向万物智联的泛在操作系统技术体系

- 从分布式到集中分层，从分布式到下一次集中分层？
- 一、云原生IDE开源框架
  - VS Code + Toolchains + OS，从而集中到IDE中以软件代码的形式进行抽象分层，兼容不同的芯片和OS
  - 分布式编译技术加速构建过程
- 二、基于SDP的仿真测试开源框架
  - 在虚拟产品中运行和测试可以提高软件开发效率和质量
  - 隔离硬件特性的差异
- 三、基于WASM的FaaS框架和应用开发框架
  - 通过WASM字节码可以兼容众多编程语言生态
  - WASM的跨平台、轻量化和高性能对接K8s分布式编排部署能力
- 软件定义一切的愿景，需要有一个工具生态去实现它！

在软件定义一切的时代，我们该如何定义软件？

Thank You