第一章 绪论

翁楚良

https://chuliangweng.github.io

2023 春 ECNU

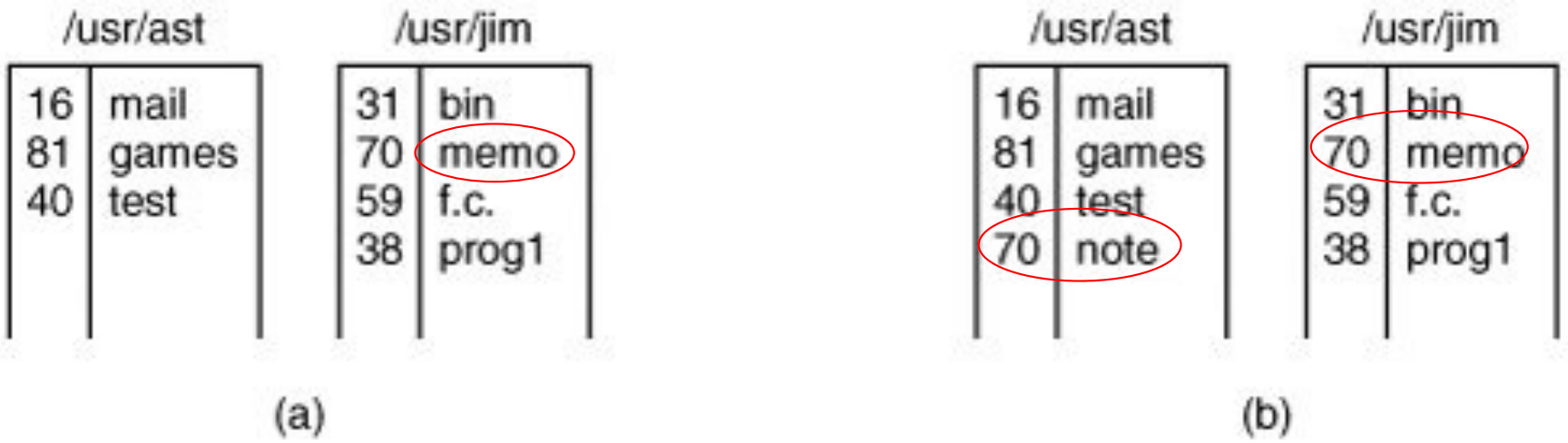
系统调用的分类

- 进程管理: fork, waitpid, getpid, …
- 信号管理: kill, alarm, pause, sigaction, …
- 文件管理: creat, open, close, read, write, dup, …
- 目录管理: mkdir, mount, link, umount, chdir, …
- 安全管理: chmod, chwon, umask, getuid, …
- 时间管理: time, stime, utime, times

目录管理系统调用

- LINK系统调用允许同一个文件按不同路径名出现
  - 一种典型的应用是允许开发小组的几个成员共享一个文件，同时该文件出现在每个人自己的目录下。
  - 每个文件都有一个唯一的数字: i-节点（i-node）号
  - i节点中存放有文件所有者以及该文件所占用的磁盘块等信息

```
link("/usr/jim/memo", "/usr/ast/note");
```



挂装(mount)文件系统

- mount系统调用可将两个文件系统合并成一个
  - 存在于RAM盘上的根文件系统，其中包含有常用命令的可执行文件及其他常用文件
  - 在插入一张存有数据的光盘后，使用mount系统调用就可以将光盘上的文件系统安装到根文件系统下

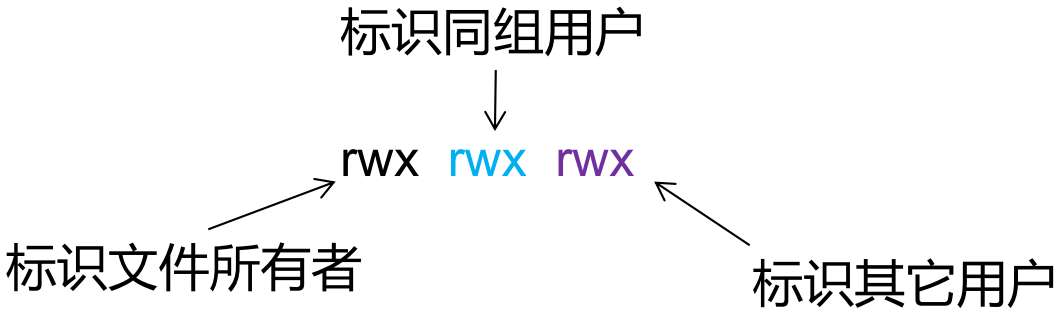
```
mount("/dev/cdrom0", "/mnt", 0);
```

系统调用的分类

- 进程管理: fork, waitpid, getpid, …
- 信号管理: kill, alarm, pause, sigaction, …
- 文件管理: creat, open, close, read, write, dup, …
- 目录管理: mkdir, mount, link, umount, chdir, …
- 安全管理: chmod, chwon, umask, getuid, …
- 时间管理: time, stime, utime, times

访问控制

- 每个文件都有一个包含11个比特的保护方式码，其中的9比特标识文件所有者、同组用户和其他用户的操作权限



chmod系统调用可以改变文件的保护方式

chmod("file", 0644);

- 另两位是设置组标识位和设置用户标识位
  - 例：系统命令 passwd (设置用户标识位)
    - 将用户的新口令写入口令文件中(一般是/etc/passwd或/etc/shadow)，而只有超级用户才具有对该文件的写许可权

系统调用的分类

- 进程管理: fork, waitpid, getpid, …
- 信号管理: kill, alarm, pause, sigaction, …
- 文件管理: creat, open, close, read, write, dup, …
- 目录管理: mkdir, mount, link, umount, chdir, …
- 安全管理: chmod, chwon, umask, getuid, …
- 时间管理: time, stime, utime, times

时间管理系统调用

- time系统调用返回当前距1970年1月1日零时的时间，以秒为单位
- stime用来设置系统时间（仅由超级用户执行）
- utime允许文件所有者（或超级用户）修改存储在文件i-节点中的时间
  - 例如touch命令就使用UTIME将文件时间设为当前时间
- times系统调用，它返回进程的计账信息

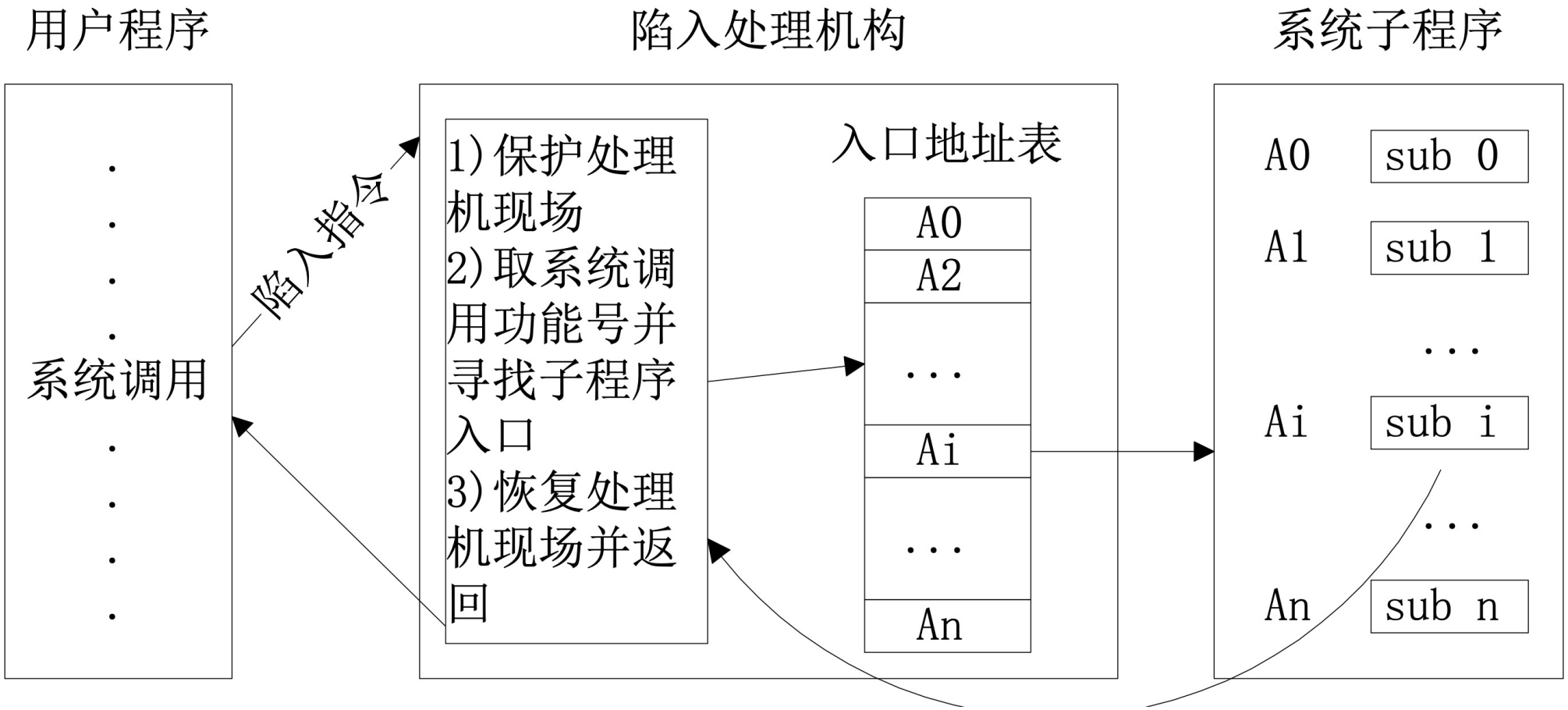
第一章绪论 提纲

- 1.1 什么是操作系统
- 1.2 操作系统的发展历史
- 1.3 操作系统基本概念
- 1.4 操作系统系统调用
- 1.5 操作系统组织结构
- 1.6 常用操作系统简介

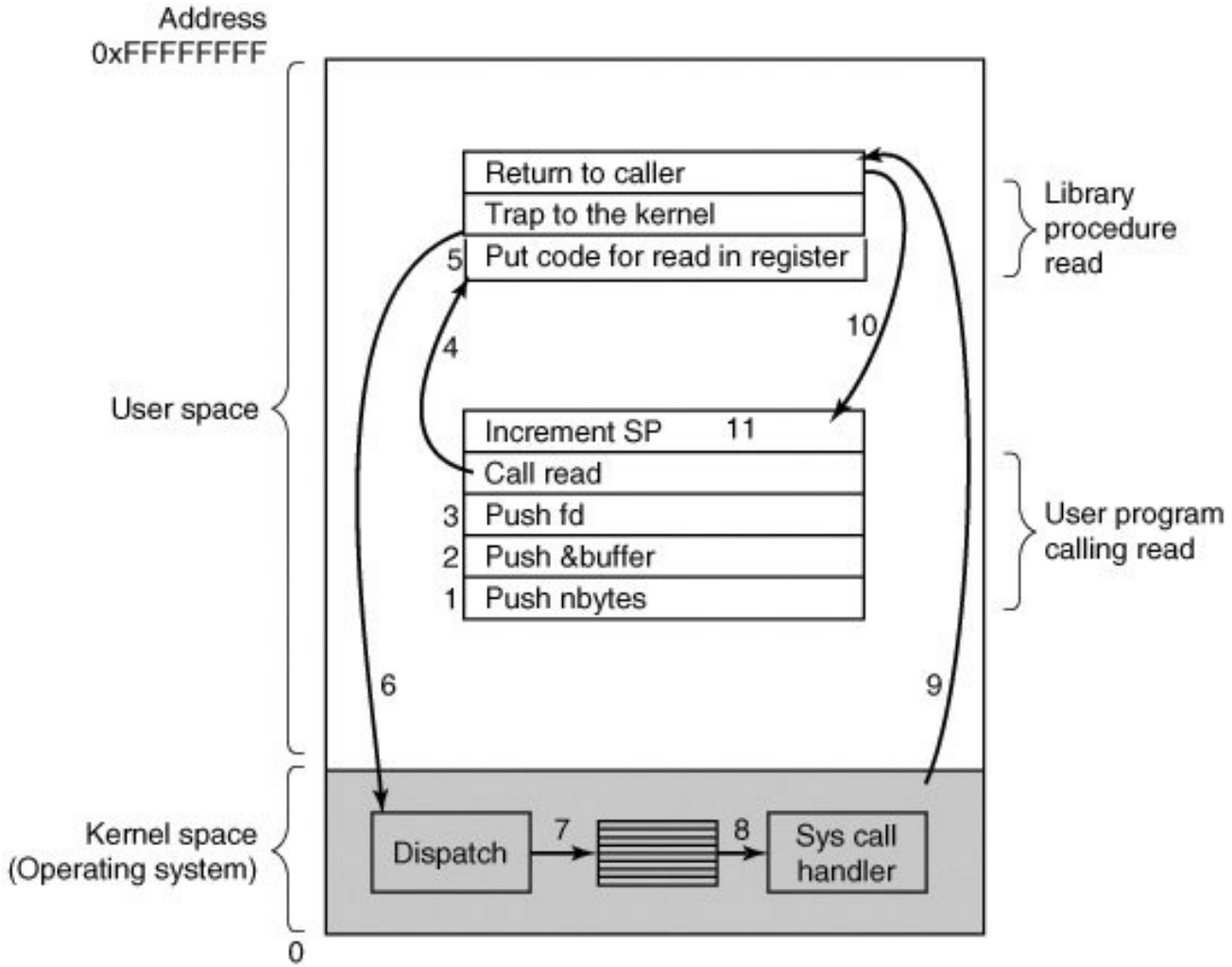
整体结构

- 整体式系统是最常用的组织方式，整个操作系统是一堆过程的集合，每个过程都可以调用任意其他过程。
- 系统中的每一过程都有一个定义完好的接口，即它的入口参数和返回值，而且相互间的调用不受约束
- CPU有两种状态:核心态和用户态
  - 核心态：供操作系统使用，该状态下可以执行机器的所有指令
  - 用户态：供用户程序用，该状态下I/O操作和某些其他操作不能执行。
- 操作系统提供的服务（系统调用）的调用过程
  - 先将参数放入预先确定的寄存器或堆栈中，然后执行一条特殊的陷入指令，即访管指令或核心调用（kernel call）指令

系统调用的实现过程


用户程序
陷入处理机构
系统子程序
.
.
.
.
系统调用
.
.
.
.
陷入指令
1)保护处理机现场
2)取系统调用功能号并寻找子程序入口
3)恢复处理机现场并返回
入口地址表
A0
A2
...
Ai
...
An
A0
sub 0
A1
sub 1
...
Ai
sub i
...
An
sub n

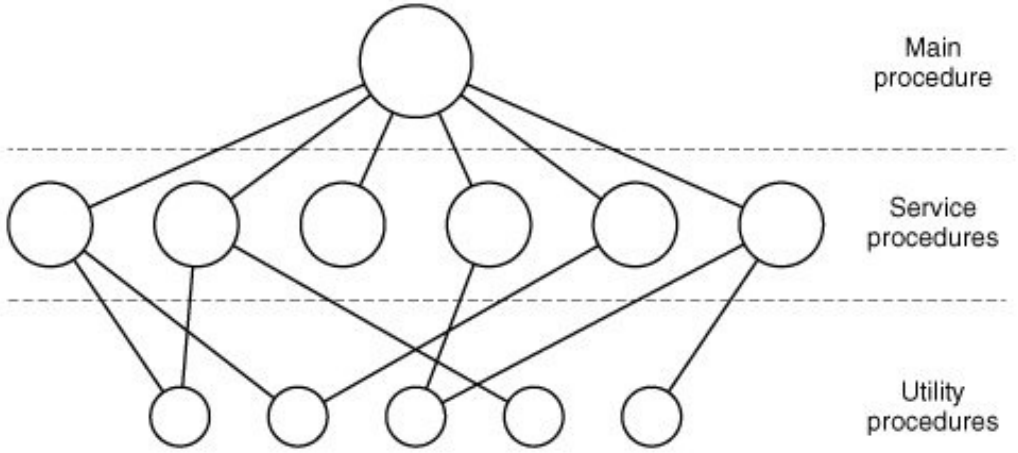
系统调用步骤实例



系统调用read(fd, buffer, nbytes)

基本结构组成

- 操作系统的一种基本结构：
- 一个用来调用被请求服务例程的主程序
- 一组执行系统调用的服务例程
- 一组支持服务例程的实用过程



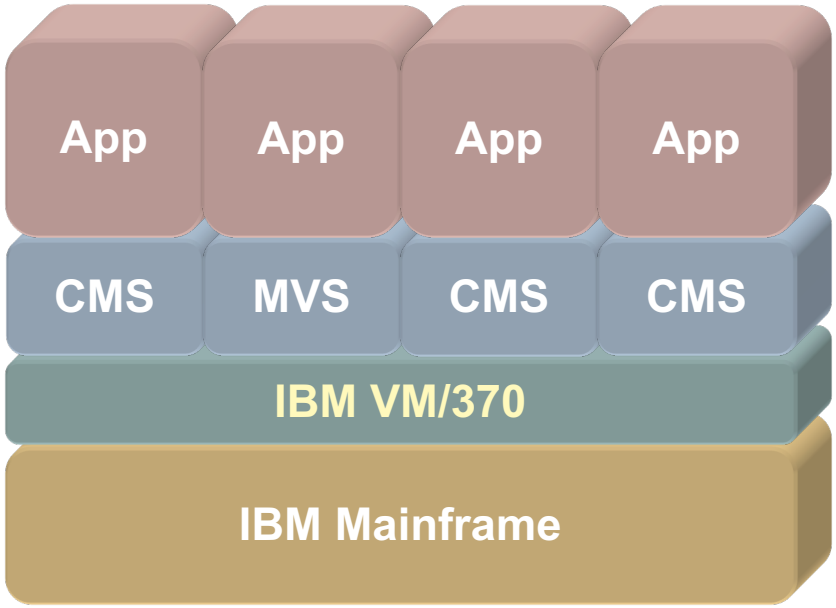
虚拟机

- 全虚拟化技术
- 半虚拟化技术
- 基于容器的虚拟化技术

全虚拟化技术 (Full Virtualization)

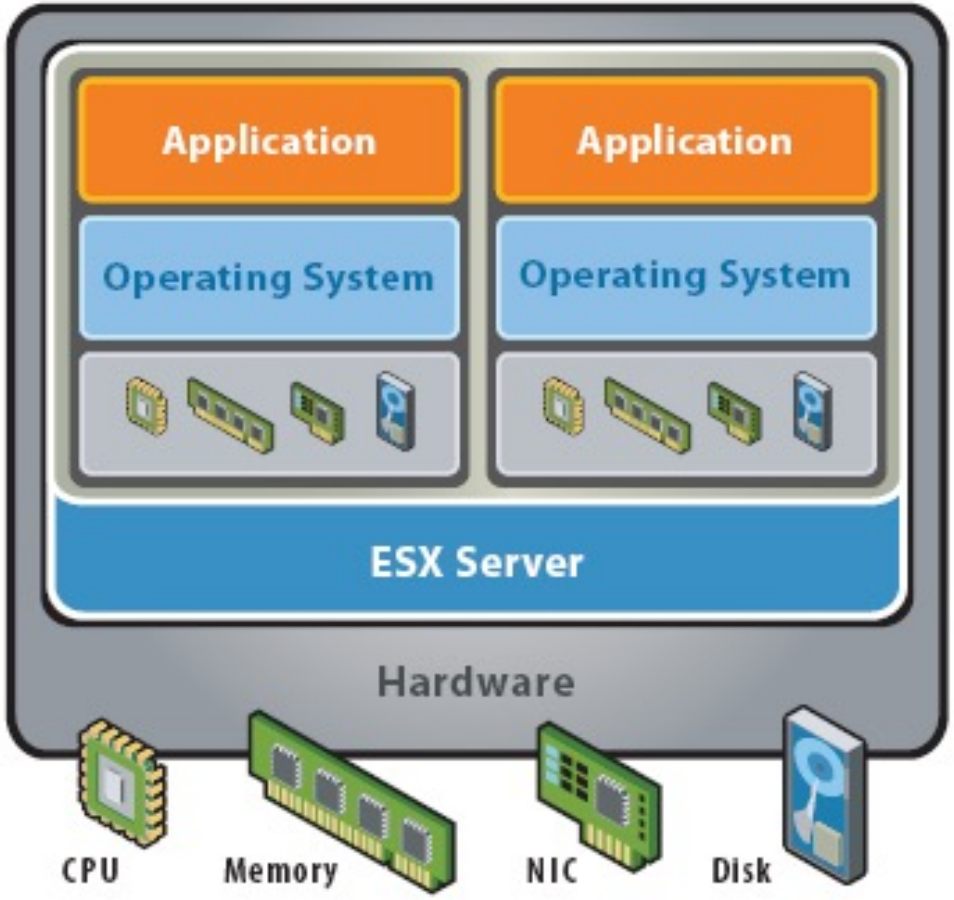
- 20世纪70年代早期，IBM的虚拟化技术在IBM370主机系列上取得了商业上的成功
  - IBM大型机非常昂贵，需要在多个用户之间共享，而用户对软件的需求特别是操作系统是不一样
  - 通过虚拟化技术，能够将一个大型机虚拟成多个大型机，并且和硬件接口完全一致，因此可以安装多个操作系统

IBM 370虚拟机结构



用一个虚拟机管理软件处在操作系统和硬件之间，从而虚拟和管理所有的硬件资源

VMWare ESX Server



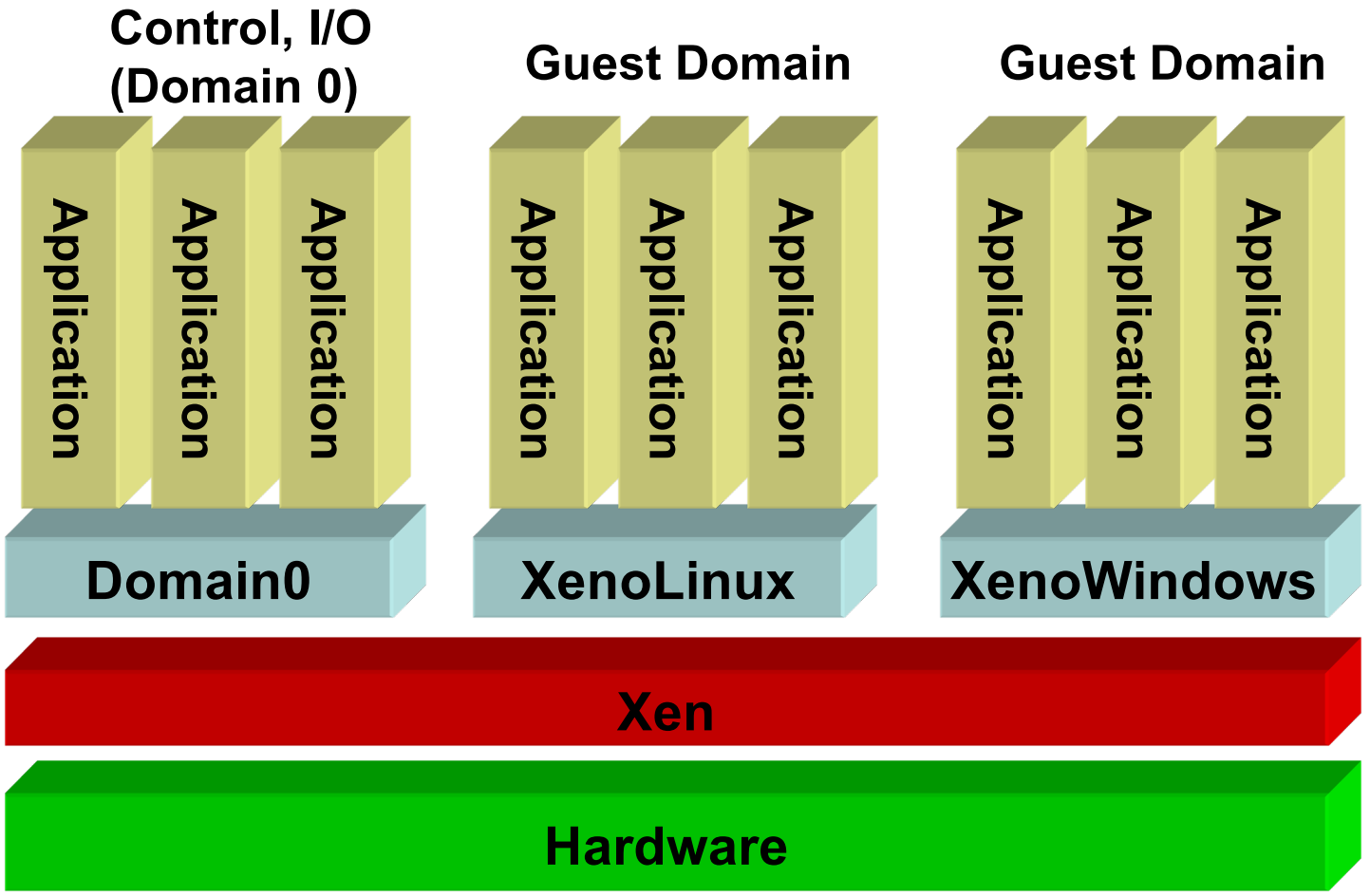
●能在一个节点上安装多个不同类型的操作系统。

●虚拟硬件设备要消耗资源，大量代码需要被翻译执行，造成了性能的损耗。

部分虚拟化技术(Para-virtulization)

- 在硬件支持不完全的情况下，仍然能够提供虚拟化
- 需要修改客户操作系统以实现部分虚拟化，修改的操作系统能够知道虚拟机的存在
- 应用程序不需要修改

Xen

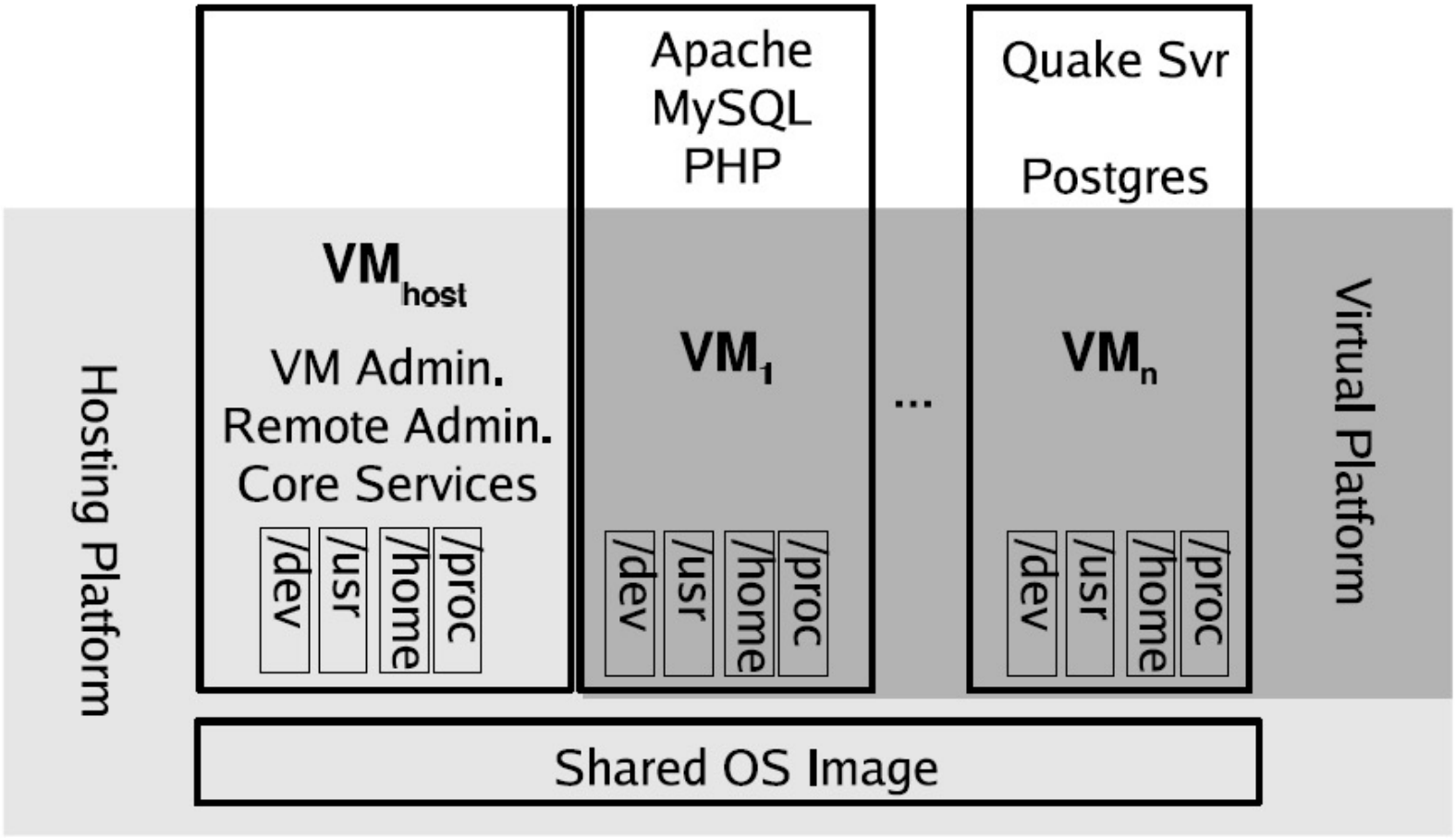

Control, I/O
(Domain 0)
Guest Domain
Guest Domain
Application
Application
Application
Application
Application
Application
Application
Application
Application
Domain0
XenoLinux
XenoWindows
Xen
Hardware

基于容器的虚拟化技术

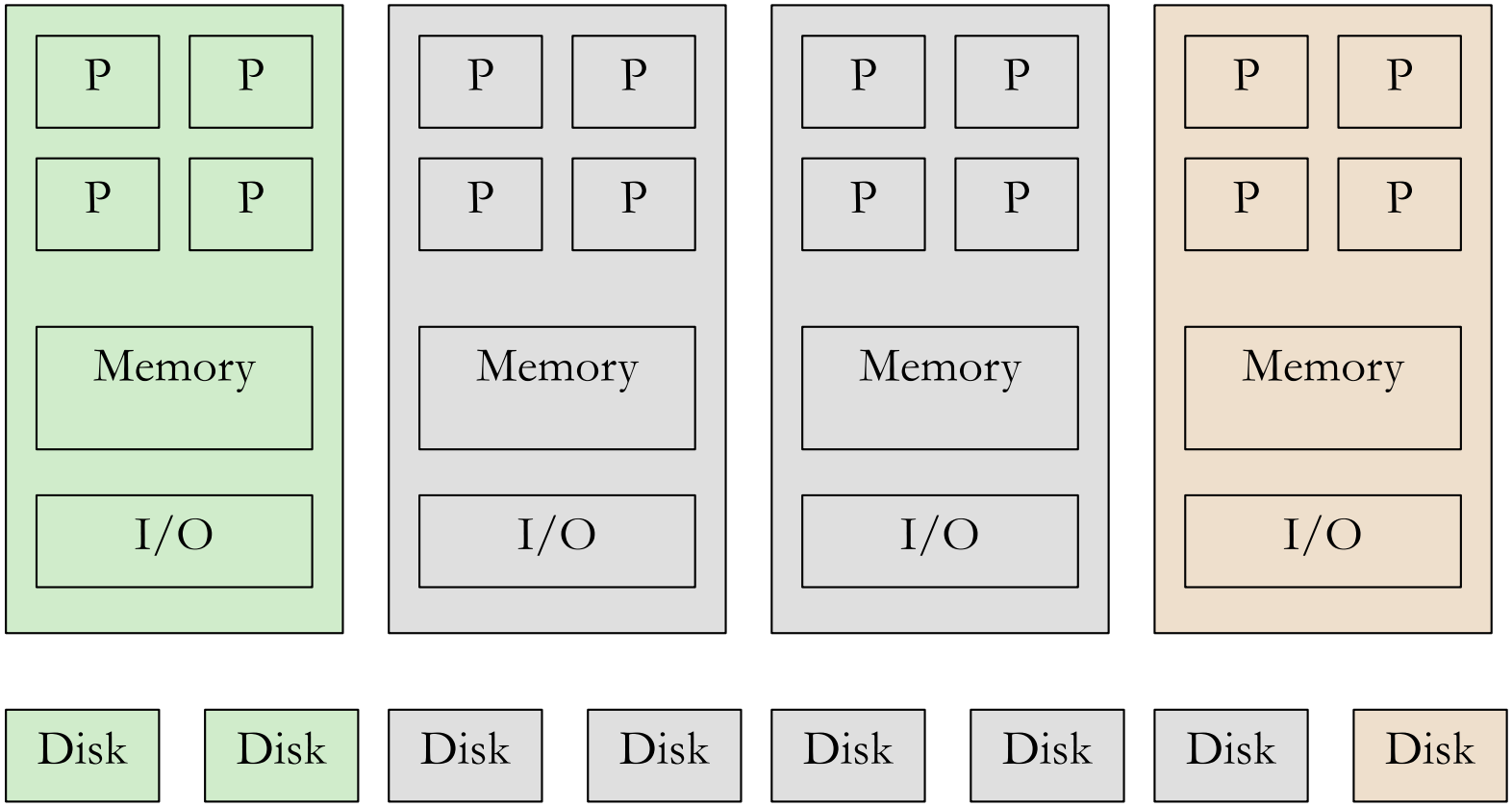
Container-based Operating System Virtualization

- 计算系统上运行着唯一的操作系统实例
- 通过在这个系统上加装虚拟化平台，可以将系统划分成多个独立隔离的容器，每个容器是一个虚拟的操作系统
- 不虚拟任何硬件设备

Linux-VServe



外核



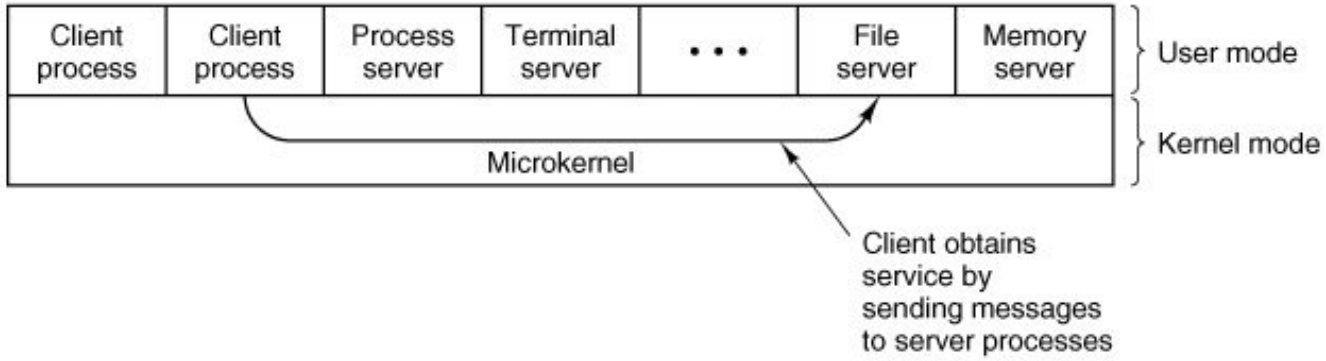
- 外核负责为虚拟机分配资源并确保资源的使用不会发生冲突
- 每个虚拟机运行自己的OS
- 每个虚拟机只能使用分配给它的那部分资源

客户/服务器模型或微内核结构

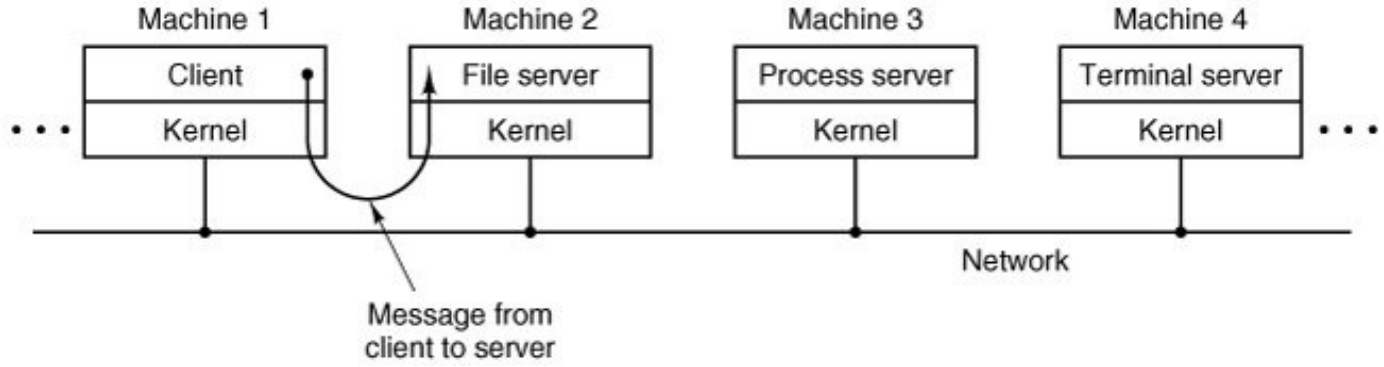
- 把操作系统分成若干分别完成一组特定功能的服务进程，等待客户提出请求；而系统内核只实现操作系统的基本功能(如：消息传递)
- 微内核(micro-kernel)：将更多操作系统功能放在核心之外，作为独立的服务进程运行
  - 服务器进程
  - 客户进程
- 本地过程调用 (LPC, Local Procedure Call)：一种进程之间请求-应答式的消息（ Message ）传递机制
- 消息：是一定格式的数据结构。①发起调用，送出请求消息②请求消息到达并进行处理③送出回答消息④整理回答消息，返回结果；如：对文件creat, read, write

系统模型

- 基本结构



- 分布式环境中的结构

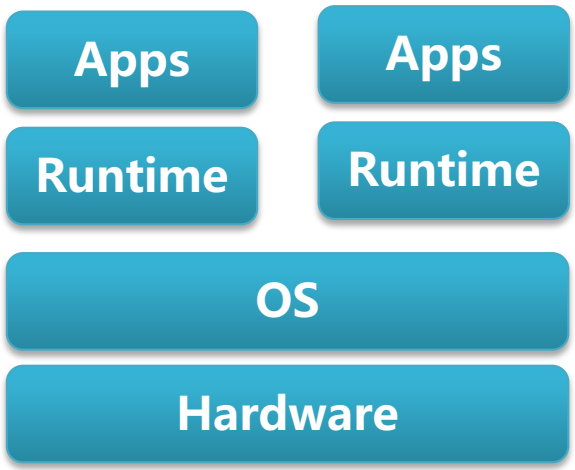


云计算基础: (轻量级)虚拟化技术

经典云计算平台:
Amazon AWS, 阿里云

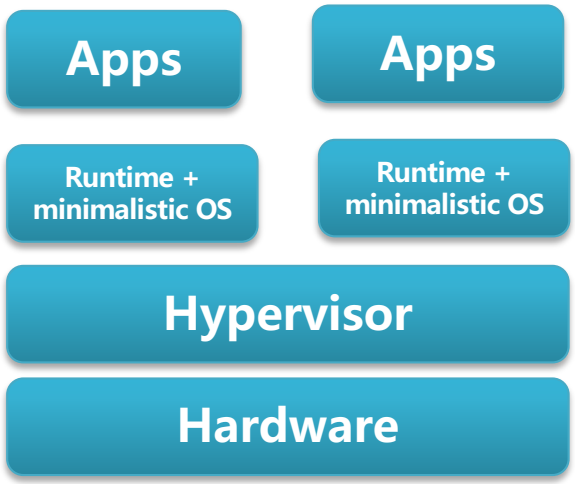
Xen, KVM等

容器技术
如何高效管理容器



Docker, CoreOS等

Unikernel
非常适用于特定类型的云服务
(如Web, DB等)



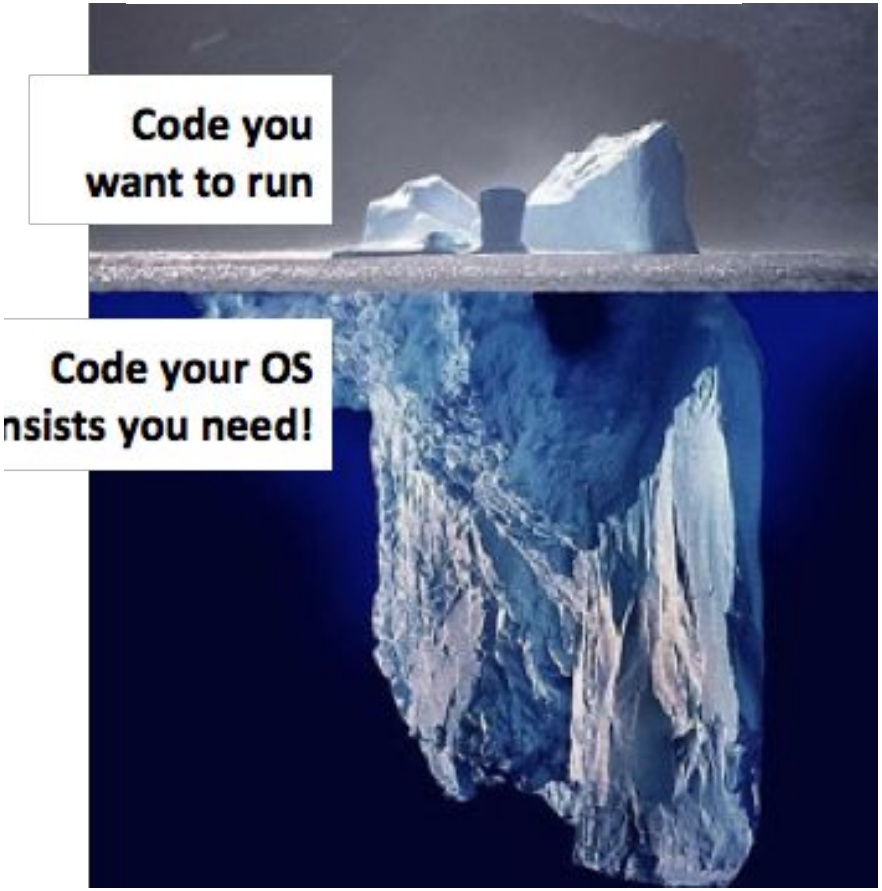
MirageOS, ClickOS等

Unikernel的基本概念

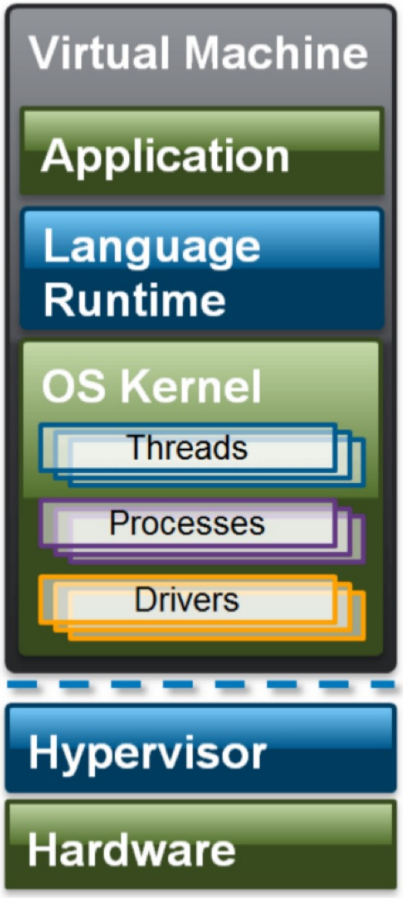
Linux: over 25 million lines of code

Debian: 65 million lines of code

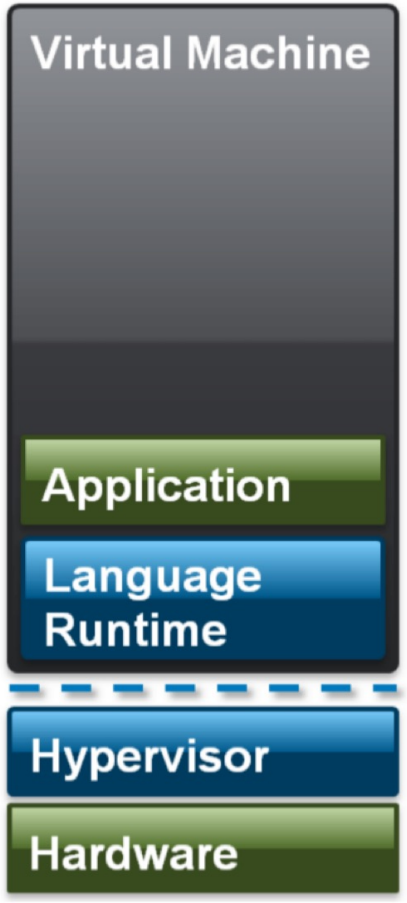
OS X : 85 million lines of code



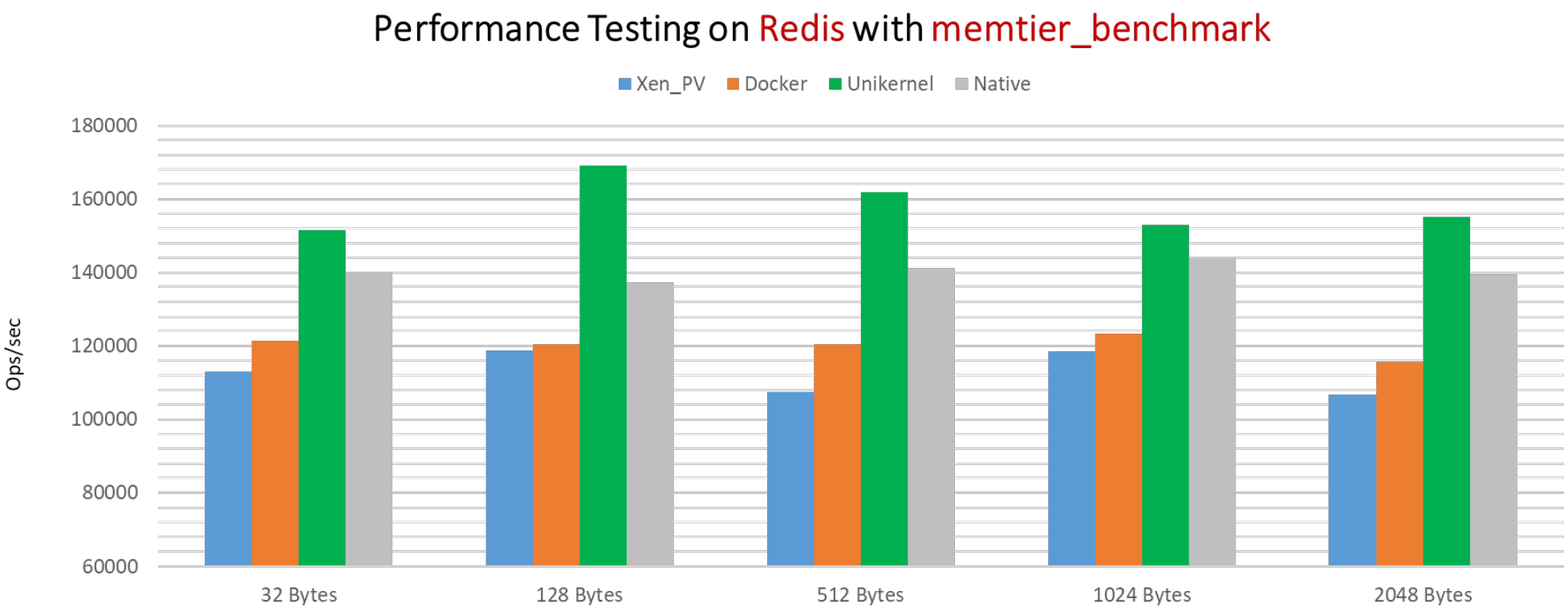
Typical Cloud Stack



Light-weight Stack With Unikernel

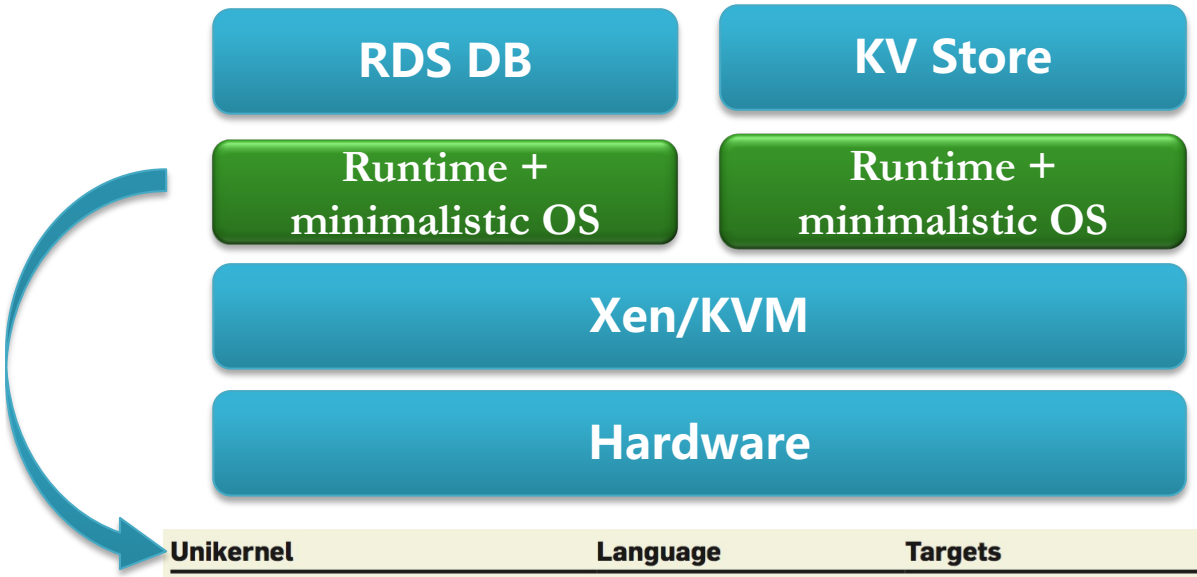


性能实测



通过实测，Unikernel云化方案是性能最好的，是云平台未来重要的发展方向

轻量级虚拟化技术：面向大数据处理



- 生成Unikernel的编译工具链
- 轻量级I/O虚拟化
- 内存管理优化
- Unikenerl的VCPU调度管理
- Unikernel多运行实例的性能隔离
- 基于Unikernel的云数据安全增强

Unikernel	Language	Targets
Mirage[13]	OCaml	Xen, kFreeBSD, POSIX, WWW/js
Drawbridge[17]	C	Windows "picoprocess"
HalVM[8]	Haskell	Xen
ErlangOnXen	Erlang	Xen
OSv[2]	C/Java	Xen, KVM
GUK	Java	Xen
NetBSD "rump"[9]	C	Xen, Linux kernel, POSIX
ClickOS[14]	C++	Xen

目前开源的相关研究工作，主要应用于某种高级语言，均是处于早期发展阶段

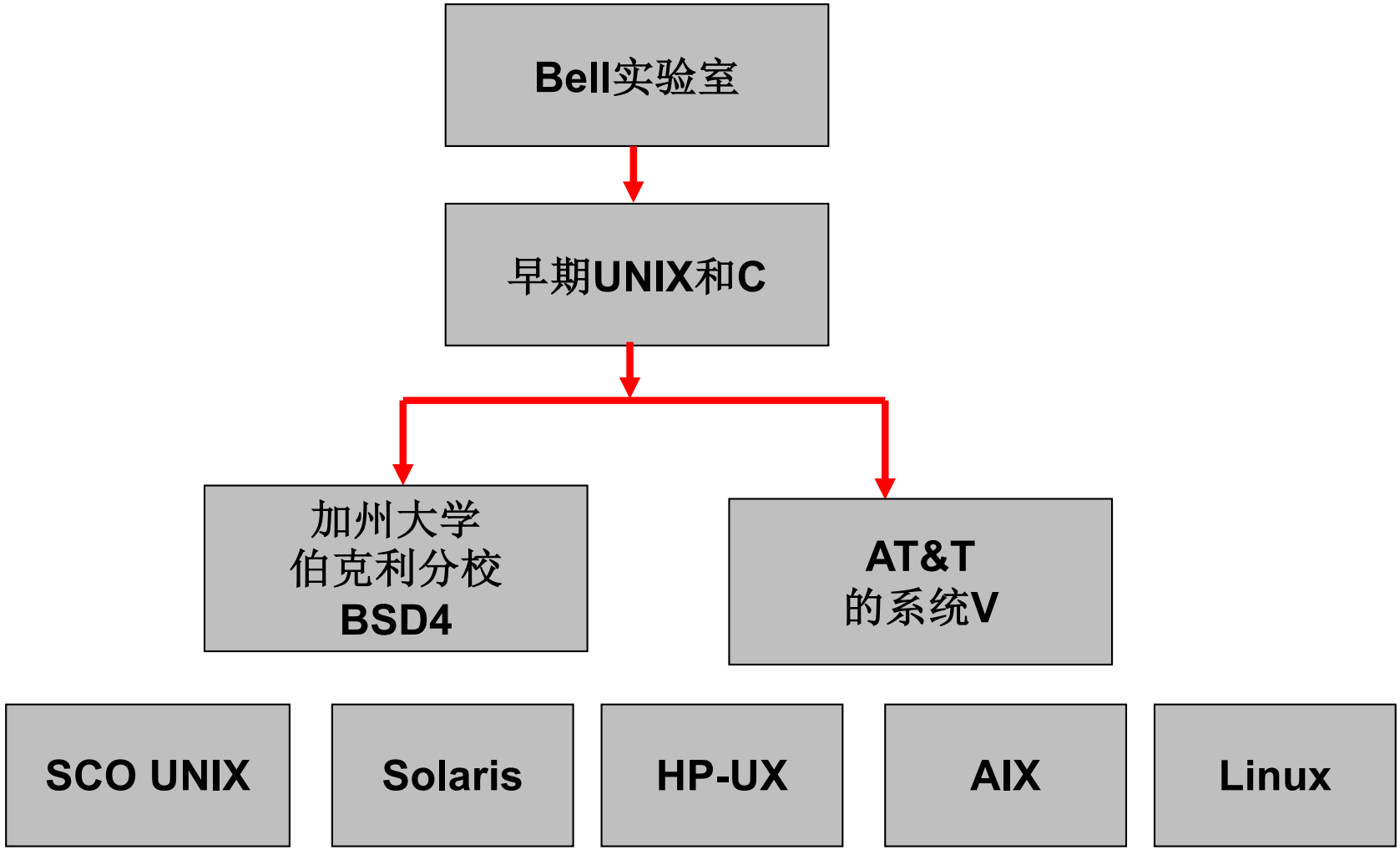
第一章绪论 提纲

- 1.1 什么是操作系统
- 1.2 操作系统的发展历史
- 1.3 操作系统基本概念
- 1.4 操作系统系统调用
- 1.5 操作系统组织结构
- 1.6 常用操作系统简介

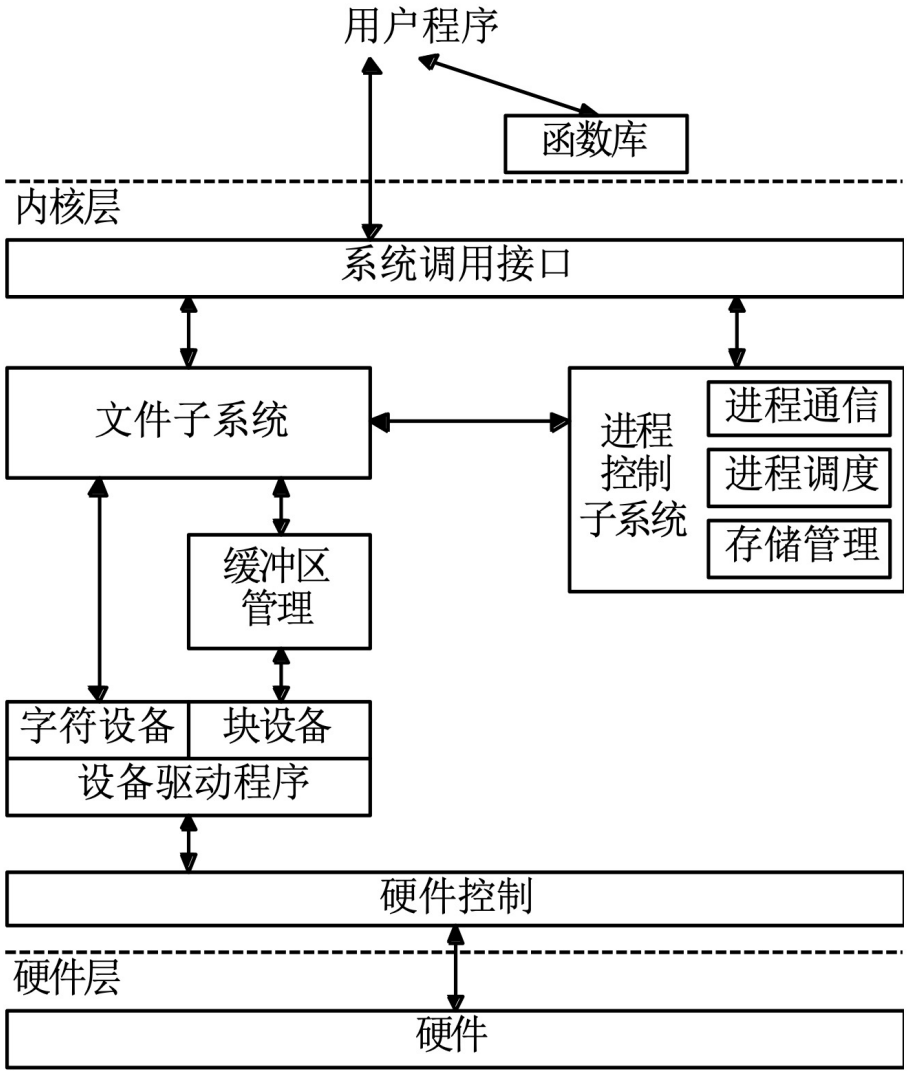
UNIX

- 1965年：MIT的Multics，由于规模和进展而没有达到目标；
- 1969年：AT&T，PDP-11上的16位操作系统；
- 1974年：UNIX系统正式发表(第五版)，在大学得到使用和好评；
- 1980年：University of California at Berkeley为VAX11发表BSD4.0；以后，UNIX就以AT&T和Berkeley为主分别开发，有多种变种；
- 1989年：UI (UNIX International)发表UNIX system V Res4.0；使BSD和System V在用户界面上统一；
- 1991年：芬兰大学生Linus Benedict Torralds开发了第一个Linux版本。
- 1994年：Linux 1.0，现在的最新内核版本是5.16.12 (2022-03-06)

- **UNIX系统：可运行UNIX应用软件的操作系统。**


Bell实验室
早期UNIX和C
加州大学
伯克利分校
BSD4
AT&T
的系统V
SCO UNIX
Solaris
HP-UX
AIX
Linux

经典UNIX结构

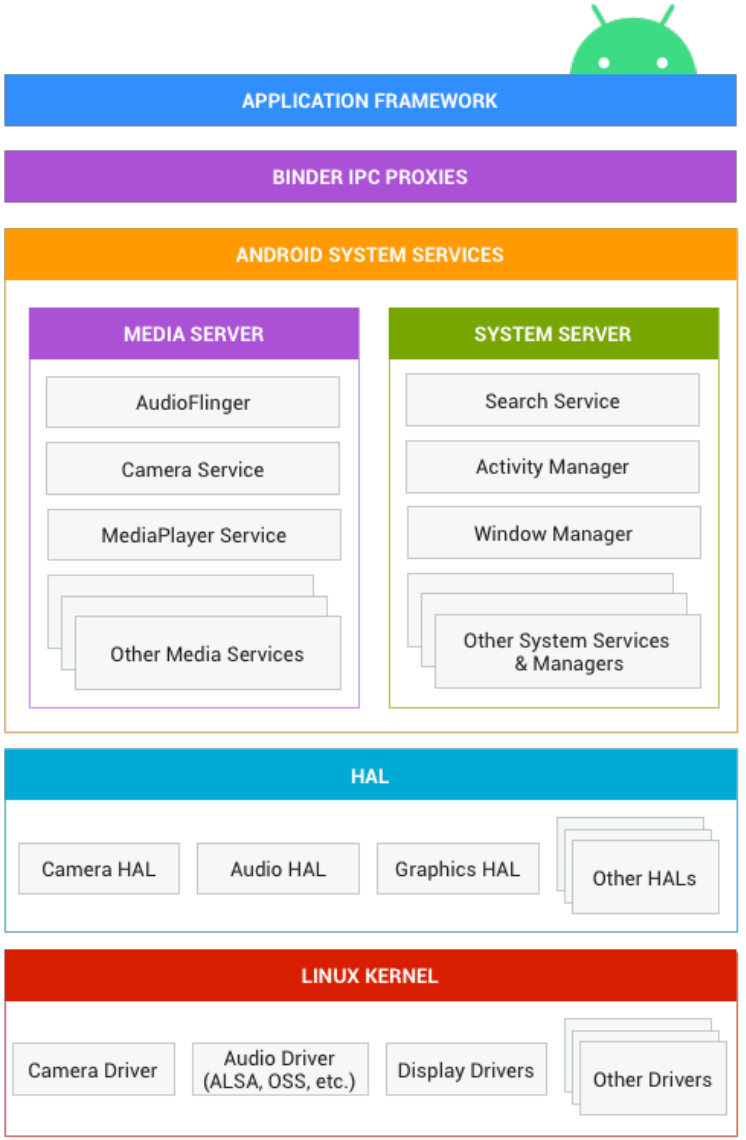

用户程序
函数库
内核层
系统调用接口
文件子系统
进程
控制
子系统
进程通信
进程调度
存储管理
缓冲区
管理
字符设备
块设备
设备驱动程序
硬件控制
硬件层
硬件

UNIX系统的特点

- 字符用户界面和图形用户界面GUI(X Window)
- 抢先式多任务，多线程。支持动态链接。支持对称式多处理
- 虚拟存储：段页式，有存储保护
- 文件系统：多级目录，文件卷可以在子目录下动态装卸。无文件属性，可有别名
- 采用设备文件的形式（读写，参数控制）。设备驱动程序修改后需要重新编译连接生成内核
- 支持多种硬件平台
- 易移植：主要代码用C语言写成
- 变种很多，很难标准化

Android

- 为移动设备设计
  - 智能手机和平板
- 基于Linux内核
  - 基于longterm版
- 定制用户的界面
  - 可以定制各种发行版本
- “开源”操作系统
  - 不是所有代码都是开源的



硬件抽象层(HAL)

- Android系统通过两层来支持硬件设备，一层实现在用户空间中，另一层实现在内核空间 中。
  - 系统的硬件抽层（ Hardware Abstract Layer， HAL ）运行在用户空间中，它向下屏蔽硬件驱动模块的实现细节，向上提供硬件访问服务。
- Linux 内核源代码遵循 GPL协议、Android系统源代码遵循 Apache License协议
  - 折中的方案是将对硬件的支持分别实现在内核空间和用户空间中
  - 内核空间仍然是以硬件驱动模块的形式来支持，不过只提供简单的硬件访问通道
  - 用户空间以硬件抽象层模块的形式来支持，它封装了硬件的实现细节和参数

小结

- 1.1 什么是操作系统
- 1.2 操作系统的发展历史
- 1.3 操作系统基本概念
- 1.4 操作系统系统调用
- 1.5 操作系统组织结构
- 1.6 常用操作系统简介

作业

- 1,3,4,6,9,10,17,18,21,26,27,31,32 (现代操作系统)