AI技术和云技术在视频监控中的应用

云计算模型

陈一帅

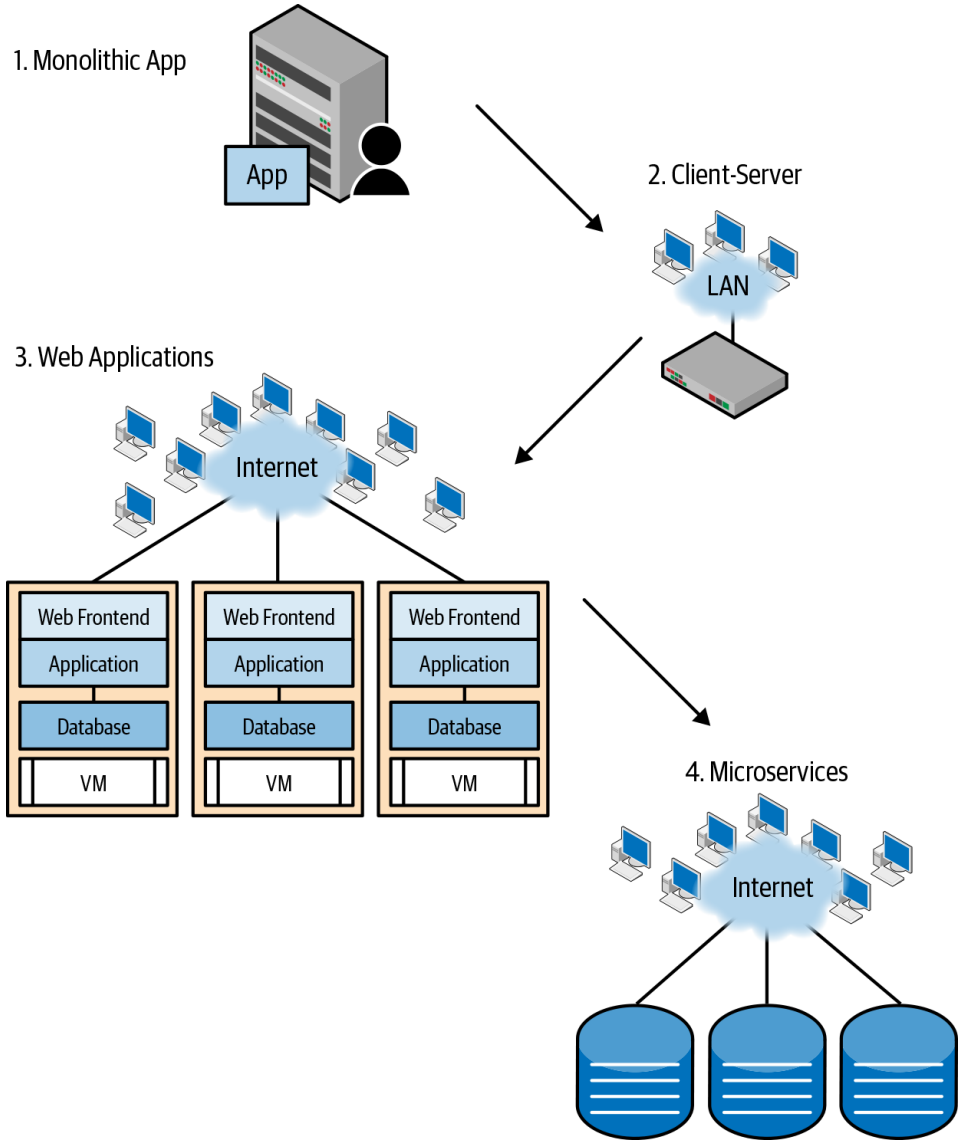
yschen@bjtu.edu.cn

北京交通大学电子信息工程学院

内容

- 虚拟化
- 虚拟机
- 容器
- 微服务

云计算应用开发环境


1. Monolithic App
App
2. Client-Server
LAN
3. Web Applications
Internet
Web Frontend
Application
Database
VM
Web Frontend
Application
Database
VM
Web Frontend
Application
Database
VM
4. Microservices
Internet

虚拟化

云计算基于虚拟化

背景知识

- 程序
  - 一些指令
- 操作系统（OS）
  - 允许多个用户程序同时运行，分享资源
  - 进行状态管理、程序上下文切换、I/O 访问控制
- 程序不能进行状态管理、I/O 访问指令，因为会接入其它程序的状态
  - 要通过操作系统执行这些指令

虚拟化

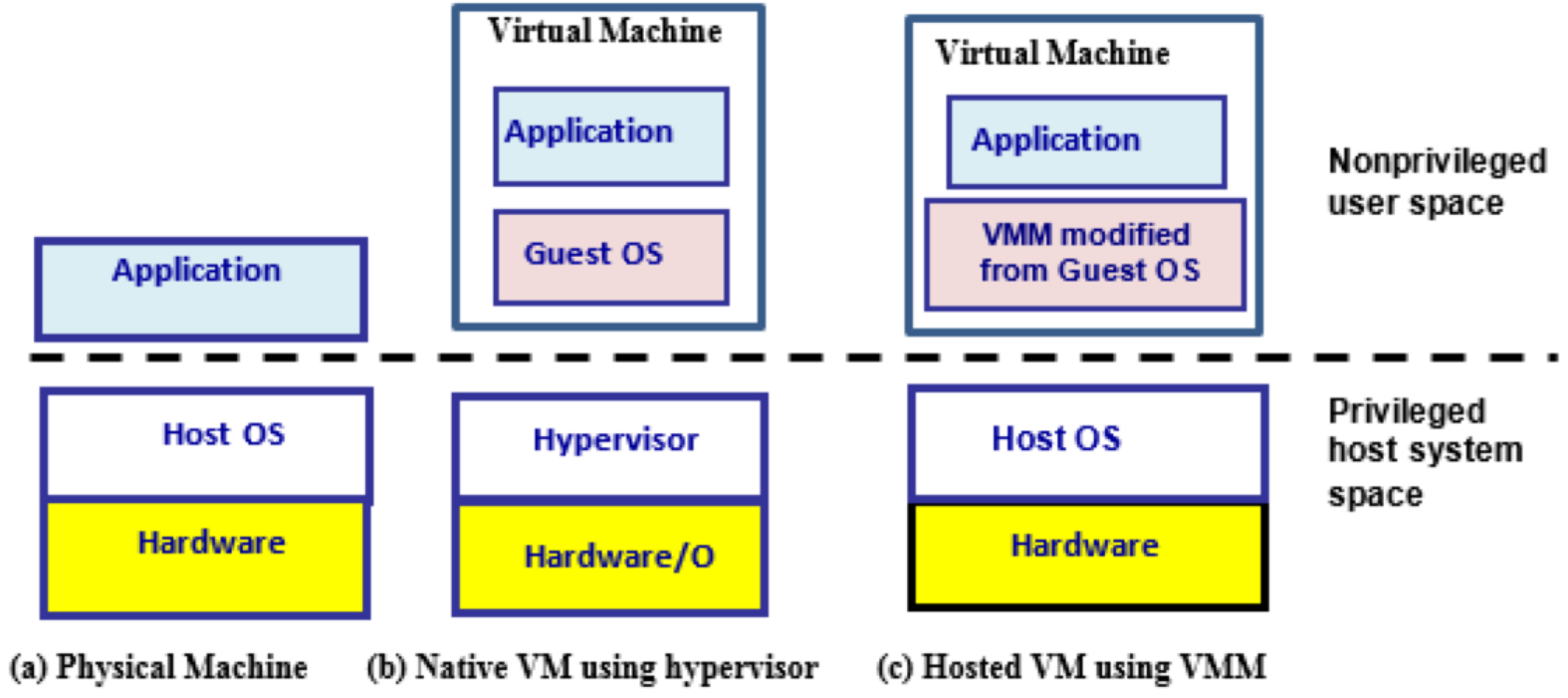
- 提供看起来真实，但实际上是在软件中处理的这些指令的过程，被称为虚拟化
- 其他类型的虚拟化（例如虚拟内存）在操作系统的指导下由硬件直接处理
- 1960 年代末和 1970 年代初，IBM 和其他公司在虚拟化方面做了很多工作，最终证明可以虚拟化整个计算机
  - R. J. Creasy. The origin of the VM/370 time-sharing system. IBM Journal of Research and Development, 25(5):483–490, 1981.

内容

- 虚拟化
- 虚拟机
- 容器
- 微服务

虚拟机

- Virtual Machine（VM）
- 在虚拟机管理程序上运行的 OS
- 是一个完整机器的软件映像，可以将其加载到服务器上并像其他程序一样运行

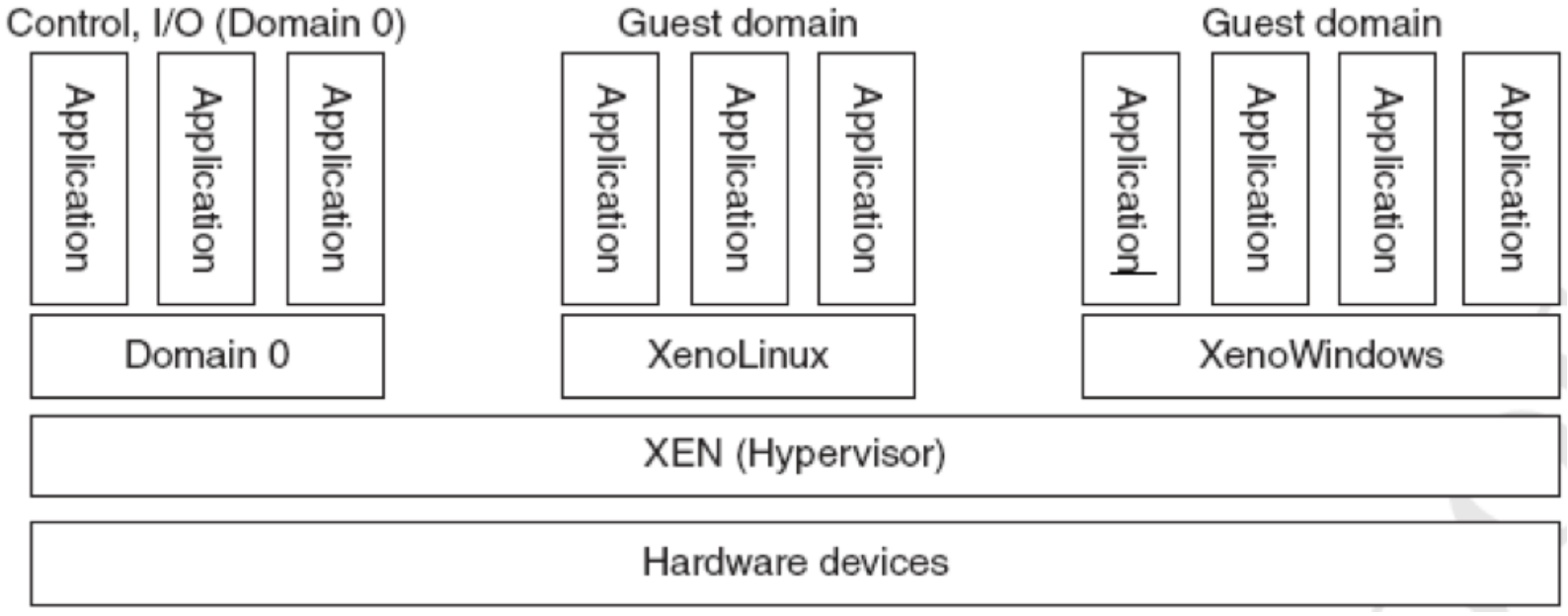


两种虚拟机

- Native 虚拟机
  - 基于虚拟机管理程序 Hypervisor
  - Hypervisor 在 OS 之下，允许多个 OS 同时运行，分享同一个硬件资源
  - 管理和分配服务器资源
  - 如 Citrix Xen，Microsoft Hyper-V，VMWare ESXi
- Hosted 虚拟机
  - 基于 VMM
  - 虚拟机管理程序作为进程运行在主机操作系统上
  - 如 VirtualBox 和 KVM

例：XEN Hypervisor

- Hypervisor 在 OS 之下
- 允许多个 OS 同时运行，分享同一个硬件资源



虚拟机带来的好处

- 云可以选择使用哪个服务器来运行请求的 VM 实例
- 如需要，可在一台服务器上同时运行多个 VM
- 同一服务器上的不同 VM 中运行的用户应用程序之间彼此之间几乎完全没有察觉
- 可以监视每个 VM 的运行状况，记录事件并重启 VM
- 一个 VM 实例崩溃，不会使整个服务器崩溃

虚拟化的好处

- 最小化硬件成本（CapEx）
  - 一台物理硬件上的多个虚拟服务器
- 能够轻松将 VM 移至其他数据中心
  - 提供灾难恢复，硬件维护
  - 跟随太阳（活跃的用户）或跟随月亮（便宜）
- 合并空闲的工作负载，释放未使用的物理资源
  - 用户流量是突发性的和异步的，提高设备利用率，省电
- 更轻松的自动化（降低 OpEx）
  - 简化的硬件和软件供应/管理
- 可扩展、灵活，支持多种操作系统

灵活的计费

- 例：亚马逊 AWS 的三种虚拟机实例的计费方式
  - On-demand instances
  - Reserved instances：1~3 年
  - Spot instances：竞价，闲时用，区域有关
- 弹性配置器
  - Ryan Chard 弹性配置器，成本降低多达 95%
  - R. Chard, K. Chard, K. Bubendorfer, L. Lacinski, R. Madduri, and I. Foster. Cost-aware cloud provisioning. In IEEE 11th International Conference on e-Science, pages 136–144, 2015.

虚拟机的问题

- 成本问题
  - 每个 VM 都需要一个操作系统（OS）
  - 每个操作系统都需要许可证 ⇒ CapEx
  - 每个操作系统都有自己的计算和存储开销
  - 需要维护，更新 ⇒ OpEx
  - VM 成本 = 添加了 CapEx + OpEx
- 性能问题
  - 启动操作系统，需要额外的 Overhead
  - VM 是完整的 OS 实例，因此可能需要几分钟才能启动

内容

- 虚拟化
- 虚拟机
- 容器
- 微服务

容器

具有虚拟机的所有良好特性，同时更轻量级

介绍

- 容器是另一种虚拟化方法
- 在同一操作系统上运行许多应用
  - 这些应用共享操作系统及其开销
- 但不会互相干扰
  - 未经明确许可就无法访问彼此的资源
- 像公寓一样 ⇒ 容器

容器的隔离

- 容器将应用程序及其所有库依赖关系和数据打包到一个易于管理的单元中
- 每个容器有自己的网络，主机名，域名，进程，用户，文件系统，IPC，互不干扰
- Docker 允许在封装所有应用程序依赖项的容器中供应应用程序
  - 该应用程序可以看到一个完整的私有进程空间，文件系统和网络接口，与同一主机操作系统上其他容器中的应用程序隔离
  - 例如，容器中的“显示进程”（在 Linux 上为 ps）命令将仅显示容器中的进程

容器的轻量级

- 在操作系统上提供隔离，轻量级
- 比 VM 轻
  - 是一个应用
  - 应用和它的依赖库、配置、数据的打包
- 启动快
  - 可以在几秒钟内将应用程序初始化并运行
- 比 Process 重
  - 一个容器内能运行多个 Process

容器的低成本

- 在一个操作系统上运行多个容器
- 所有容器共享操作系统
- CapEx 和 OpEx

容器的优点

- 具有虚拟机的所有良好特性
- 可移植
  - 容器从 Image 中生成，也可以另存为 Image
  - 同一个 Image 可以在个人计算机，数据中心或云中运行
  - 可以停下来、保存并移动到另一台计算机上或以后运行
- 可扩展
  - 可以在同一台计算机或不同计算机上运行多个副本
- 灵活
  - 可根据容器构建时的设计来限制或不限制操作系统资源

Docker

- 使用 Google 的 Go 编程语言编写
- 管理容器，提供覆盖网络和安全性
- 提供容器之间的隔离，帮助他们共享操作系统
- 最初由 Docker.com 开发，现已开源
- 可从 Docker.com 下载用于 Linux，Windows 和 Mac
- 两个版本：
  - 社区版（CE）：免费进行实验
  - 企业版（EE）：用于带付费支持的部署
- Docker Swarm 和 Kubernetes 管理大量容器

Image 存储库（Registries）

- Image 存储在 Registries 中
  - 每个 Image 都有几个标签，例如 v2，最新，...
  - 每个 Image 都通过其 256 位哈希值进行标识
- 主机上有本地存储库
- 网络存储库
  - Docker Hub 存储库，经过 Docker 审查的 Image
  - 非官方存储库，未经审核的 Image，谨慎使用
- 启动 Image 时，在本地存储库中找不到的任何组件，会从指定位置下载

Docker Hub

- https://hub.docker.com/
- 公共资源，可以在其中存储你的容器，搜索和下载数百个公共容器
- 创建一个免费的 Docker 帐户，并将容器保存到 Docker Hub

```
docker push yourname/bottlesamp
```

- 然后就可以将容器下载到云中并运行

运行容器

```
Docker run -d -p 8000:8000 yourname/bottlesamp
```

- 下载
  - 首次运行时，会下载所有组件，需要时间
  - 组件下载后，缓存在本地计算机上
- 其它命令
  - ls, exec, stop, start, rm, inspect

内容

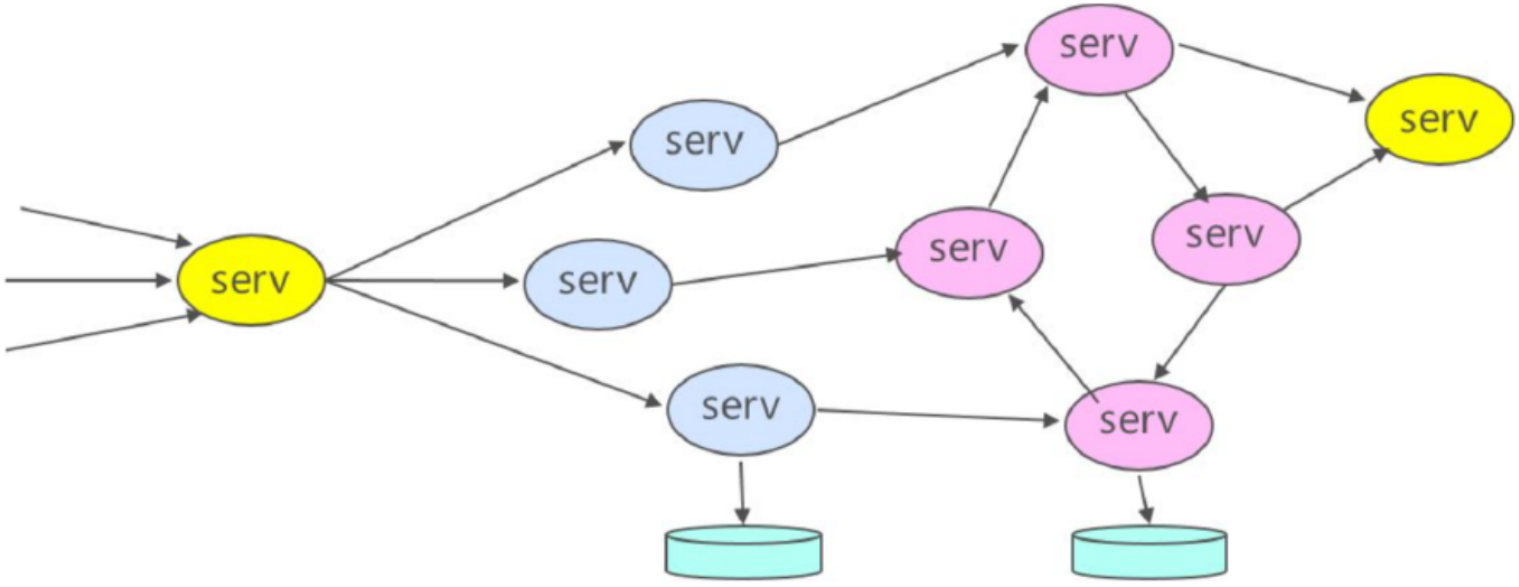
- 虚拟化
- 虚拟机
- 容器
- 微服务

微服务

- 大型云应用程序的主要设计范式
- 消息 + 参与者
  - 计算由许多进行消息通信的参与者执行
  - 每个参与者都有自己的内部专用内存，并且在收到消息后开始行动
  - 根据消息，它可以更改内部状态，也可以向其他参与者发送消息
  - 参与者可实现为功能单一的 Web 服务 （Unix 设计原则）
- 如果服务无状态，就是微服务
  - 微服务通常被实现为容器实例

微服务系统模型

- 微服务构成群（Swarm）
- 微服务利用云中的虚拟网络进行异步通信



微服务适合大规模互联网应用

- 大型在线服务的需求
  - 需要支持数千个并发用户
  - 在维护和升级的同时，还必须保持每天 24 小时在线
- 微服务集群，可伸缩

微服务组件设计

- 将应用程序划分为小型、独立的微服务组件
- 每个微服务必须能独立于其他微服务进行管理
  - 复制，扩展，升级和部署
- 每个微服务仅具单一功能，在有限上下文中运行
  - 责任有限，对其他服务的依赖也有限
- 所有微服务都应能支持不断发生故障和恢复
- 尽可能重用现有可信服务，如数据库，缓存和目录

微服务组件间通信

- 组件之间通过简单，轻便的机制进行通信
- 通信机制
  - REST Web 服务调用
  - RPC 机制（例如 Google 的 Swift）
  - 高级消息队列协议（AMQP）

微服务的广泛应用

- 微服务理念已被广泛采用
  - Netflix，Google，Microsoft，Spotify 和 Amazon
- 云中的应用被设计成可扩展的服务
  - 如 Web 服务器或移动应用程序的后端
  - 接受来自远程客户端的连接，并根据客户端请求执行一些计算并返回响应
  - 处理来自远程传感器的事件，通知控制系统如何响应
  - 如当地球传感器检测到以明显方式发生的地面震动时，会发出地震警告
- 大数据系统

微服务的管理

- 大规模微服务应用，需要管理大量分布式通信服务
- 管理平台
  - 启动实例
  - 停止实例
  - 创建新版本
  - 扩展实例数量

管理工具

- Mesos
- Docker
  - Kubernetes: Google 云容器调度程序
  - Swarm: Docker 容器调度程序

小结

- 虚拟化
- 虚拟机
- 容器
- 微服务