



# Lecture 18: 课程整体回顾

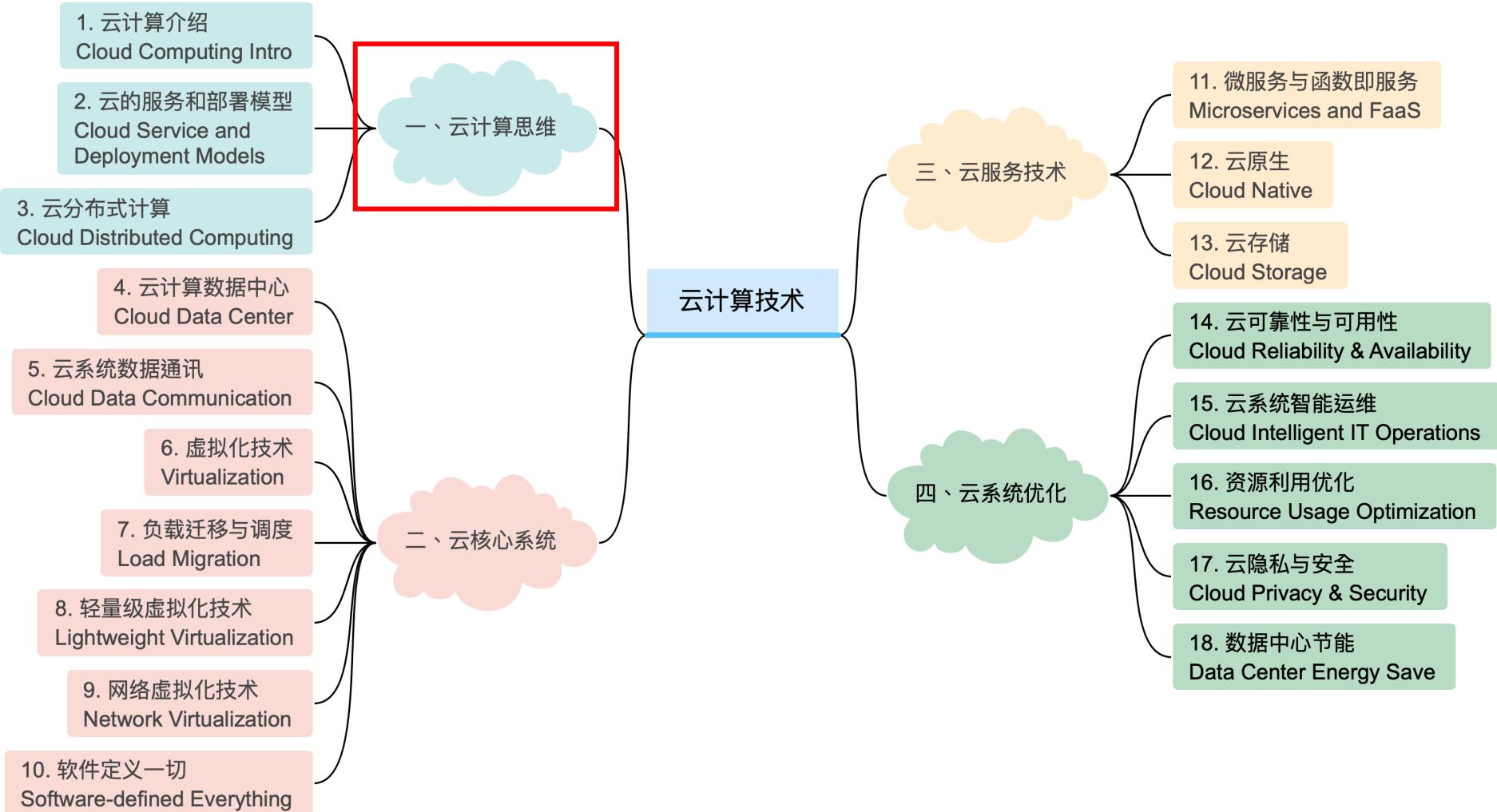
SSE316: 云计算技术  
Cloud Computing Technologies

---

陈壮彬

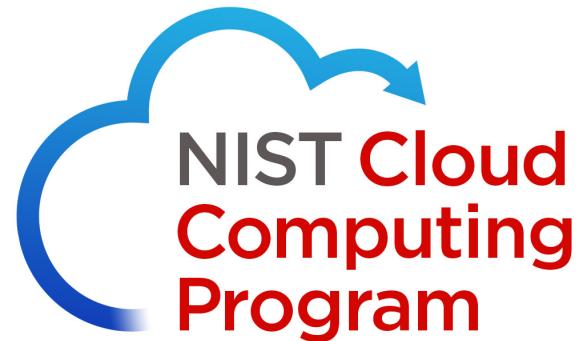
软件工程学院

chenzhb36@mail.sysu.edu.cn



# 云计算的定义

□ 美国国家标准与技术研究院 (NIST)



Cloud computing is a model for enabling **ubiquitous, convenient, on-demand** network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with **minimal management effort or service provider interaction**.

云计算是一种模型，可以实现**随时随地、便捷地、按需地**从**可配置计算资源共享池**中获取所需的资源（例如，网络、服务器、存储、应用程序及服务），资源可以快速供给和释放，**使管理的工作量和服务提供者的介入降低至最少**。

# 云计算的五个基本属性

Resource Pooling (计算资源汇聚)

Broad Network Access (广域网接入)

On-demand Self-service (按需服务)

Rapid Elasticity (敏捷弹性)

Measured Service (量化服务)

# 云计算的重要社会意义



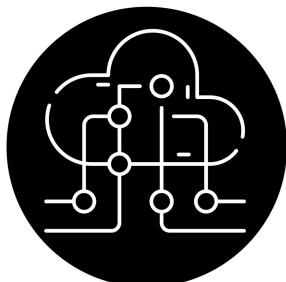
## 国计民生角度看云计算

将计算和存储简化为像公共的水和电一样**易用**能用的资源，只要连上网络即可按量付费使用。



## 商业经济角度看云计算

云计算旨在优化计算资源和数据服务在的**供需**关系，是一种基于互联网的新型IT资源供给模式。



## 信息技术角度看云计算

使用户专注于核心业务，降低计算技术的**门槛**，而不是让IT成为他们发展的困扰。

# 使用云计算的优势

## □更低的投资与开销

- 企业无需建设大规模计算基础设施而产生过高的初始投资
- 价格更具吸引力



## □更低的复杂性

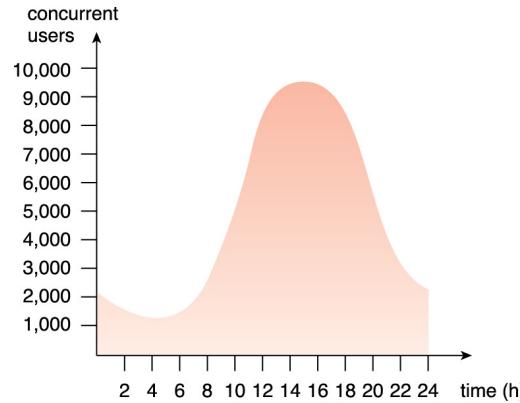
- 云提供者抽象IT资源，提供“就绪可用（ready-to-use）”或“现成（off-the-shelf）”的解决方案
- 简化和加快IT资源的开发、部署和管理
- 减少运维负担



# 使用云计算的优势

## □更高的可扩展性

- 短时按需细粒度地使用 IT 资源，并在不需要时释放资源
- 无需在前期进行大量的容量规划工作（通常难以准确预测）



一个组织一天的  
IT 资源需求变化

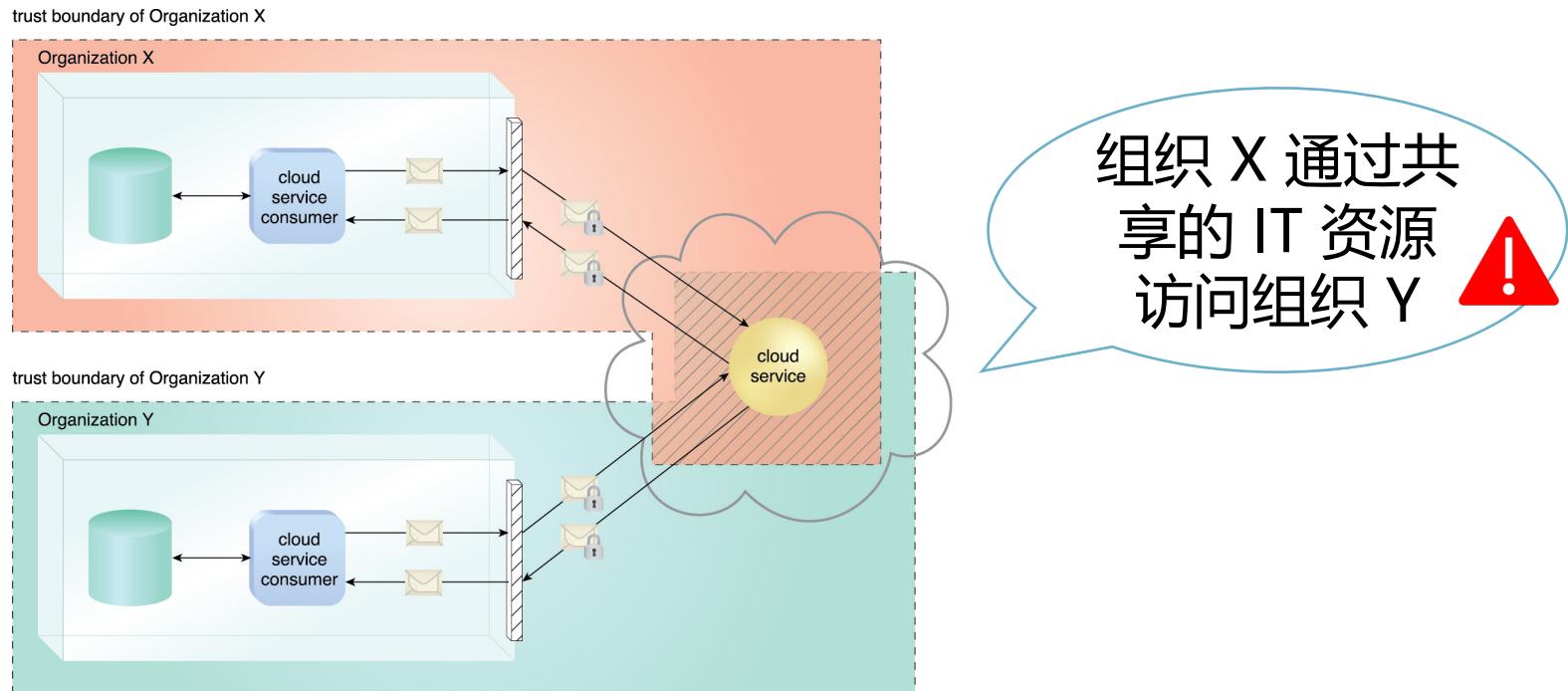
## □更好的可靠性和可用性

- 云提供者提供“可靠的”IT资源，降低意外情况的概率
- 云提供者提供“可恢复的”IT资源，缩短意外情况持续的时间（广泛支持故障转移）

# 风险与挑战

## 口更多的安全漏洞

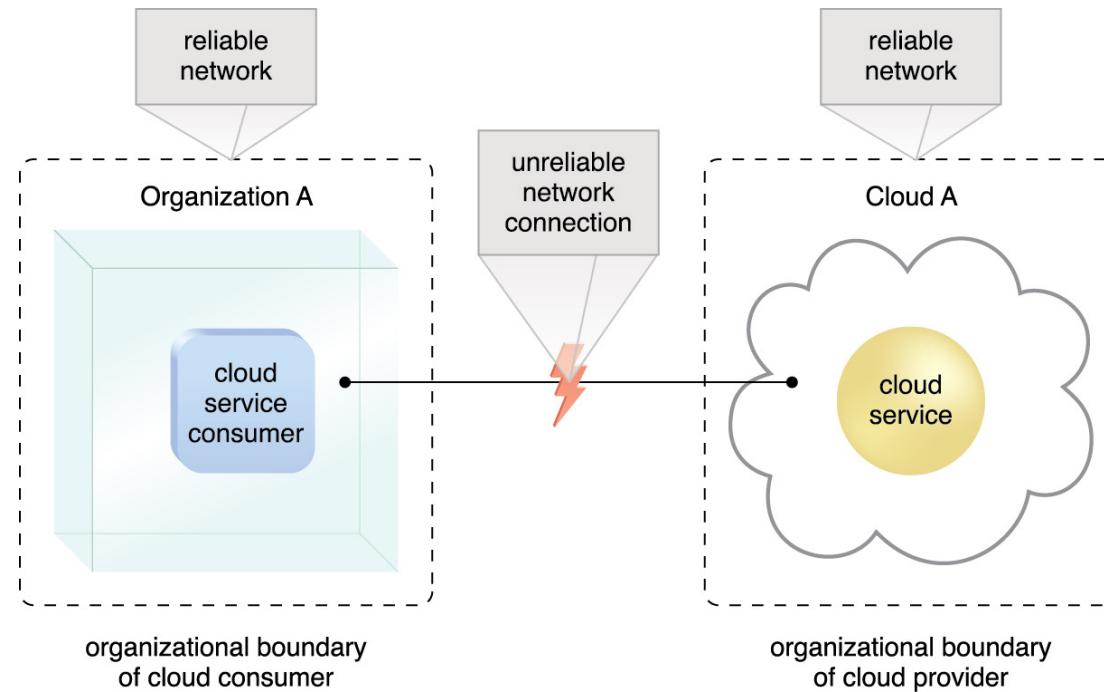
- 远程使用 IT 资源需要用户将信任边界拓展到外部云，云提供者要分担数据安全的责任
- 云提供者拥有访问用户数据的特权
- 不同云用户共享云IT资源，存在重叠的信任边界



# 风险与挑战

## 降低的运营管理控制

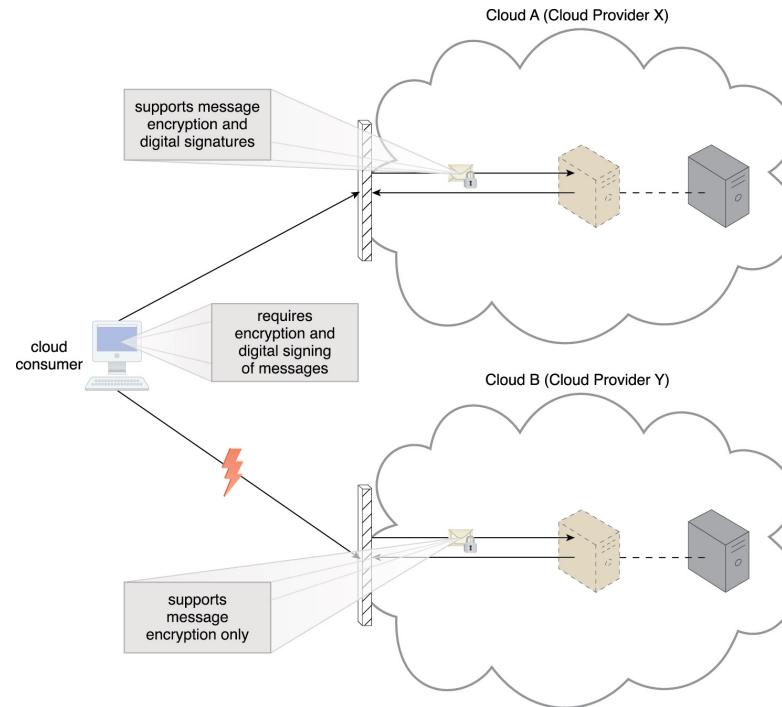
- 云用户丧失对 IT 资源管理的部分权限和灵活性
- 不可靠的云提供者可能不会遵守其 SLA 保证
- 云用户与提供者之间的地理距离使服务质量受限于网络质量（如延迟波动和带宽）



# 风险与挑战

## □云提供者之间有限的可移植性

- 云计算行业尚未建立工业标准，不同云提供者存在一定程度的定制化
- 当云用户依赖云提供者的某些私有特性时，在不同云提供者之间进行用户资源和数据的迁移就成为了问题



# 风险与挑战

## 不同地区的法规法律问题

- 数据隐私和存储的行业或政府法规：如英国法律规定英国公民的数据只能留在境内
- 数据的获得和公开：有些国家规定某些类型的数据必须向政府或数据主体公开

### TikTok 回应安全质疑：美国用户数据存于美国



雷锋网

2019/10/28 18:18 优质科技领域创作者 来自广东

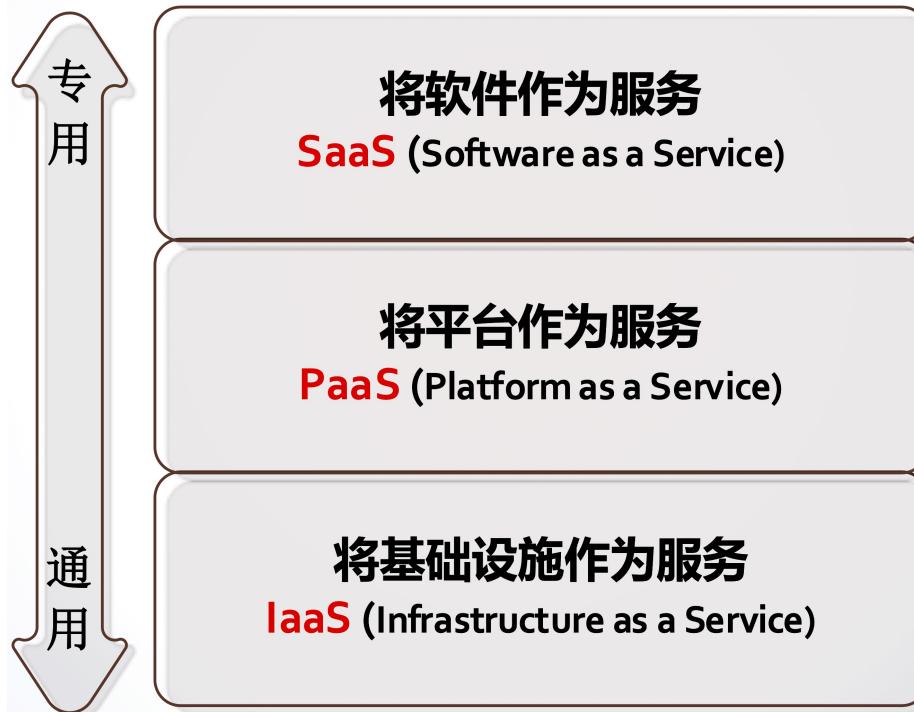
◆ 雷锋网消息，近日，TikTok（抖音海外版）在其官网上发布文章，对美国议员的质疑的内容审查和数据安全



# 服务模型

云服务模型 (Cloud Service Model) , 也称  
云交付模型 (Cloud Delivery Model)

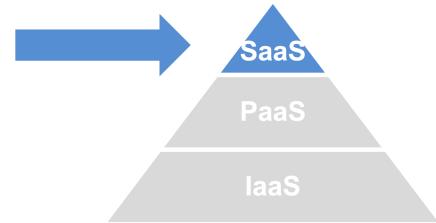
指的是云计算的运营方式, 以所有权、大小、访问方式区分



三种标准的服务模型

# 软件即服务 (Software as a Service)

**SaaS** 为用户通过网络提供顶层软件服务



产品

供应商提供的各类在线应用

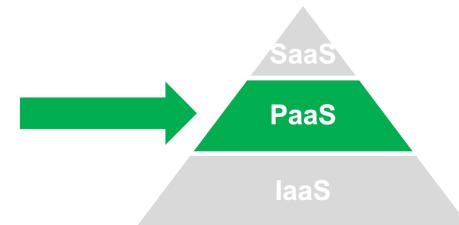
用户  
权力

**你能：**通过轻量级的客户端访问的软件，借助有限的环境参数设定来实现应用的简单个性化...

**你不能：**管理和控制底层基础设施，接触系统开发平台，对所部署的应用进行根本性优化改变...

# 平台即服务 (Platform as a Service)

PaaS 为用户提供应用的运行时支撑环境



产品

提供基本操作系统、编程环境、重要的公共库函数，各类用于高层次开发的系统工具

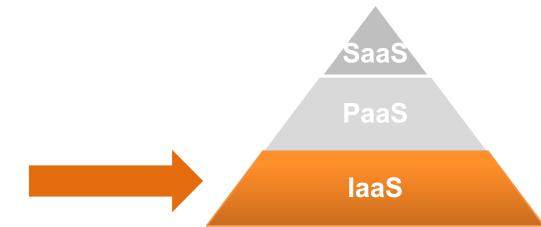
用户  
权力

你能：搭建应用，对所部署的应用进行测试，以及基于应用运行环境的相关参数进行优化部署...

你不能：管理和控制底层的基础计算设施，包括网络、服务器、操作系统、或数据存储等

# 基础设施即服务 (Infrastructure as a Service)

IaaS 为用户提供基本的底层计算资源



产品

提供计算服务器(物理设备)、存储服务器(物理设备)、虚拟计算节点(虚拟设备)和存储节点等

用户  
权力

你能：配置操作系统、搭建应用环境、管理存储设备、测试各类代码，以及部分网络环境管控权

你不能：管理和控制底层硬件基础设施的配置，比如服务器的开关机，控制服务器功耗分配等

# 服务等级协议

## 服务等级协议 Service Level Agreement

24 x 365 运行机制

5个“9” 可用性要求

任务响应时间

...

### Cloud Services Agreement

Using this agreement, Client may order Cloud Services. This agreement and applicable Attachments and Transactional Documents (TDs) are the complete agreement (Agreement) regarding transactions under this Agreement.

responsible for their actions, omissions, statements, or offerings.

Neither party may assign the Agreement, in whole or in part, without the prior written consent of the other. Assignment of IBM rights to receive payments and by IBM in conjunction with the sale of the portion of IBM's business that includes a service is not restricted.

The Agreement applies to IBM and Client and their respective Enterprise companies who avail themselves of the Agreement. The parties shall coordinate the activities of Enterprise companies under the Agreement. Enterprise companies include (i) companies within the same country that Client or IBM control (by owning greater than 50% of the voting shares), and (ii) any other entity that controls, is controlled by or is under common control with Client or IBM. All rights reserved. © 2007 IBM Corporation. All rights reserved.

## 服务等级协议 (SLA)

Agreed to:

Client Company Name:

Agreed to:

IBM Company Name:

By \_\_\_\_\_  
Authorized signature

By \_\_\_\_\_  
Authorized signature

Title:

Title:

Name (type or print):

Name (type or print):

Date:

Date:

Client number:

Agreement number:

Enterprise number:

Client address:

IBM address:

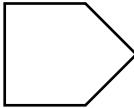
# 服务等级协议

## □ 服务等级协议 Service Level Agreement

- 云提供者与云用户间的一份协议
- 协议设定了承诺提供的云服务等级的期望值

### 常见服务等级协议内容

- 页面加载时间
- 事务处理时间
- 事件解决时间
- 安全隐私保障
- 突发事件响应计划
- 数据所有权申明
- 标准，价格，责任等



**Service Level Indicator, SLI**  
**服务等级指标**是具体的指标  
“响应时间、吞吐量、错误率”

**Service Level Objective, SLO**  
**服务等级目标**是指标的目标  
“响应时间小于100ms”

# 华为云不同云服务模型的 SLA

## 文字识别OCR SaaS



服务可用性	代金券补偿金额
99.00%≤服务可用性<99.90%	月度服务费的10%
服务可用性<99.00%	月度服务费的25%

## 弹性云服务器 ECS IaaS



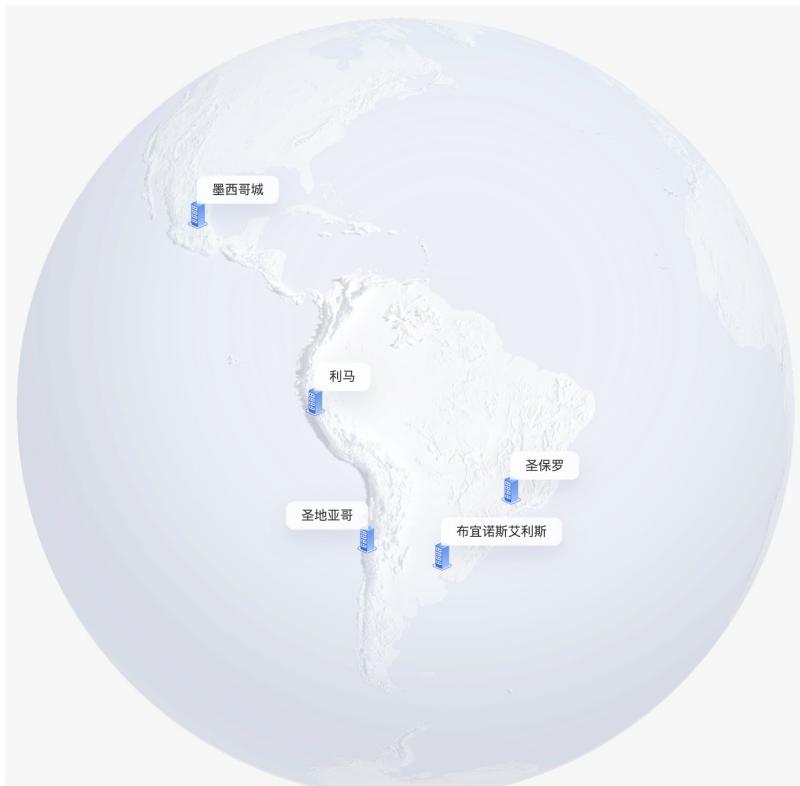
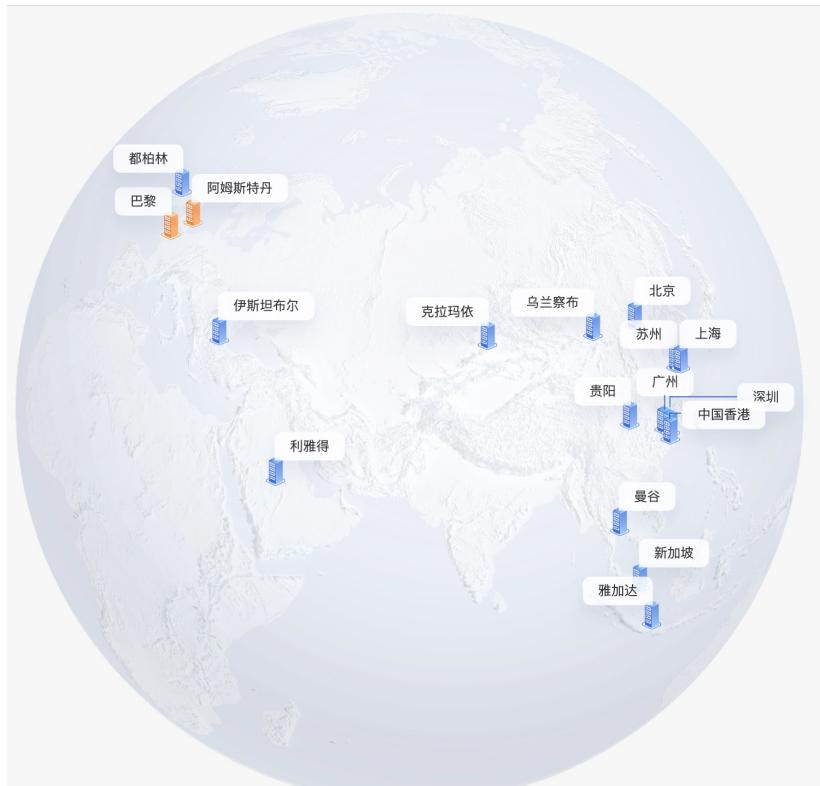
服务可用性	代金券补偿金额
99%≤服务可用性<99.975%	月度服务费的10%
95%≤服务可用性<99%	月度服务费的30%
服务可用性<95%	月度服务费的100%

## AI开发平台 ModelArts PaaS



服务可用性	代金券补偿金额
95.00%≤服务可用性<99.95%	月度服务费的10%
服务可用性<95.00%	月度服务费的25%

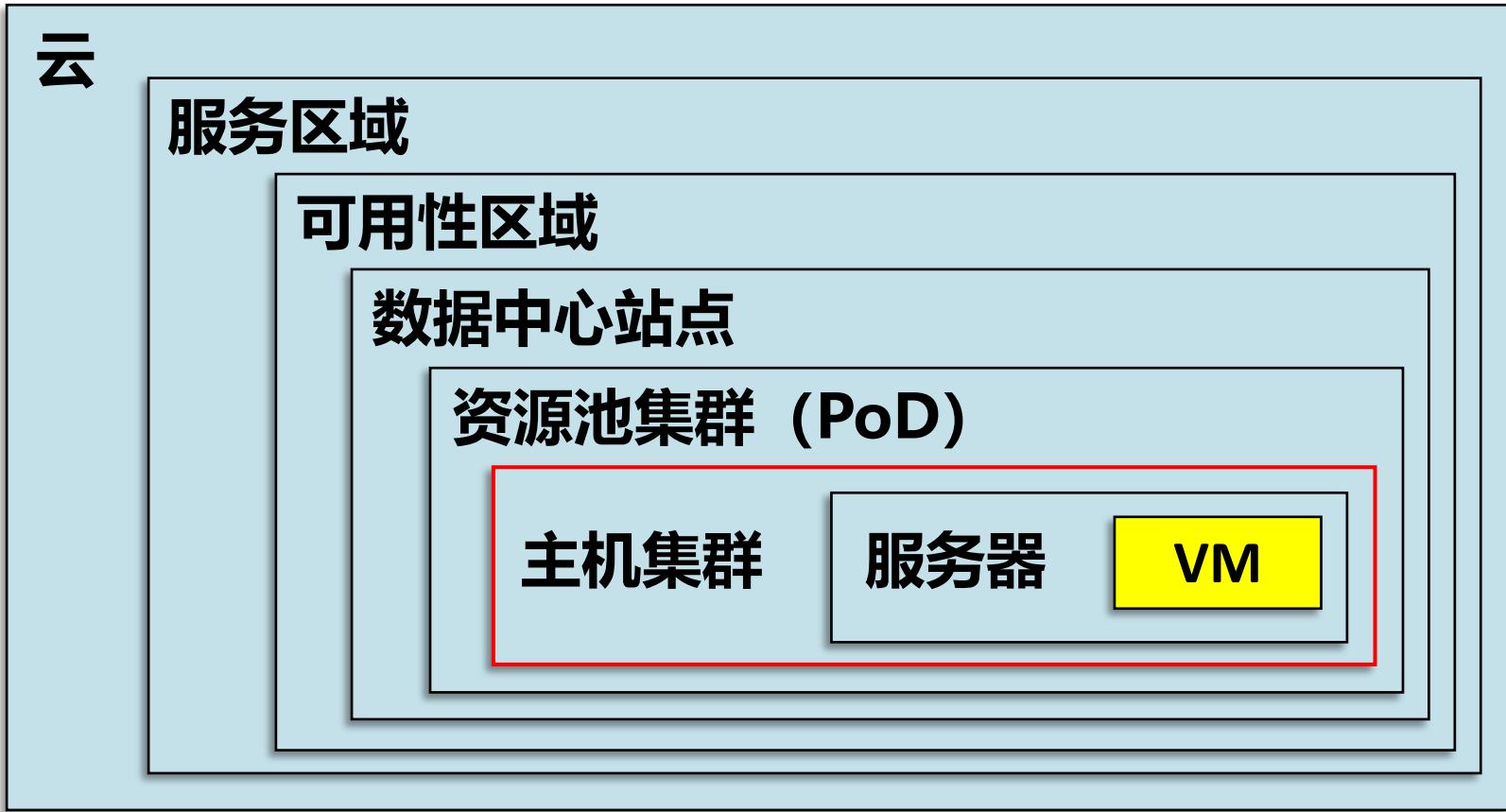
# 大型云计算厂商拥有覆盖全球的数据中心



华为云全球数据中心  
<https://www.huaweicloud.com/>

# 云的分级视图

面向管理员的分级视图

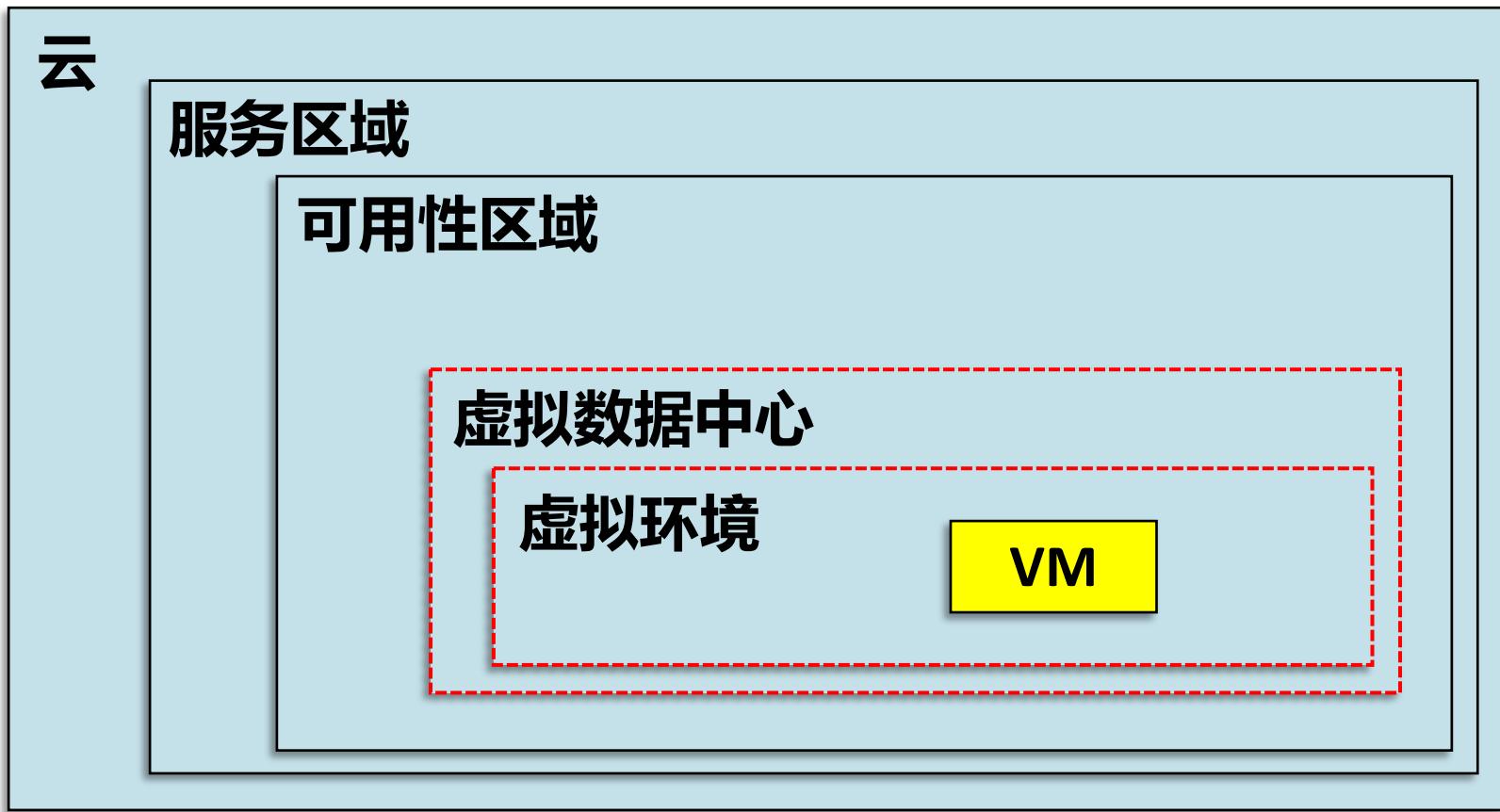


口 相同类型的服务器还可以构成一个主机集合

- 某一服务器主机可以被标记为多个资源属性标签

# 云的分级视图

面向租户的分级视图



在租户签约虚拟机或容器服务时，可以看到隶属的虚拟机实例  
在租户签约物理机的情况下，可以看到隶属的物理机实例

# 云的部署模型

## 部署模型 (Deployment Models) 定义云计算的部署和访问方式

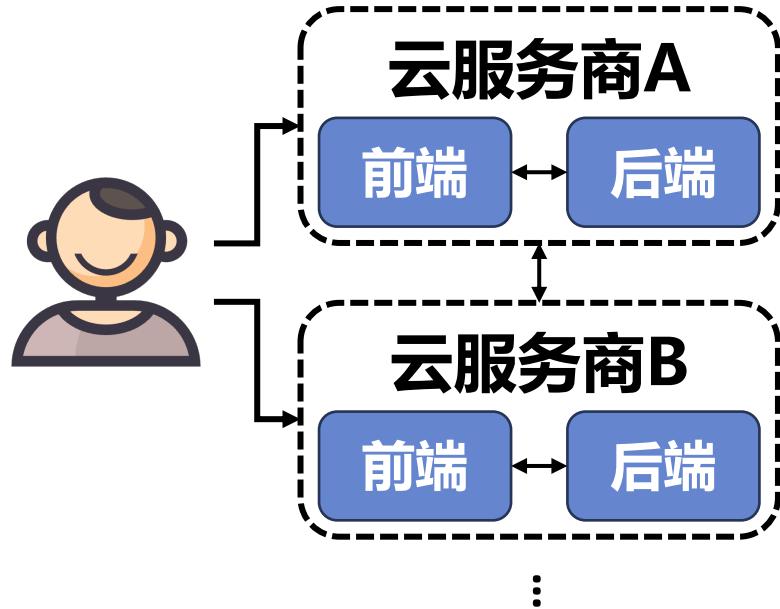
私有云 (Private Cloud)

社区云 (Community Cloud)

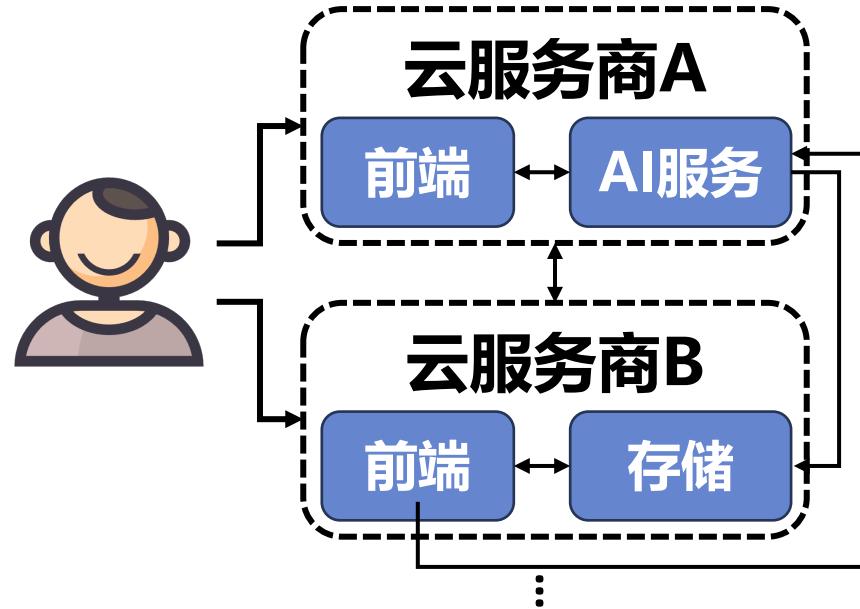
公有云 (Public Cloud)

混合云 (Hybrid Cloud)

# 多云和聚云



(a) 多云



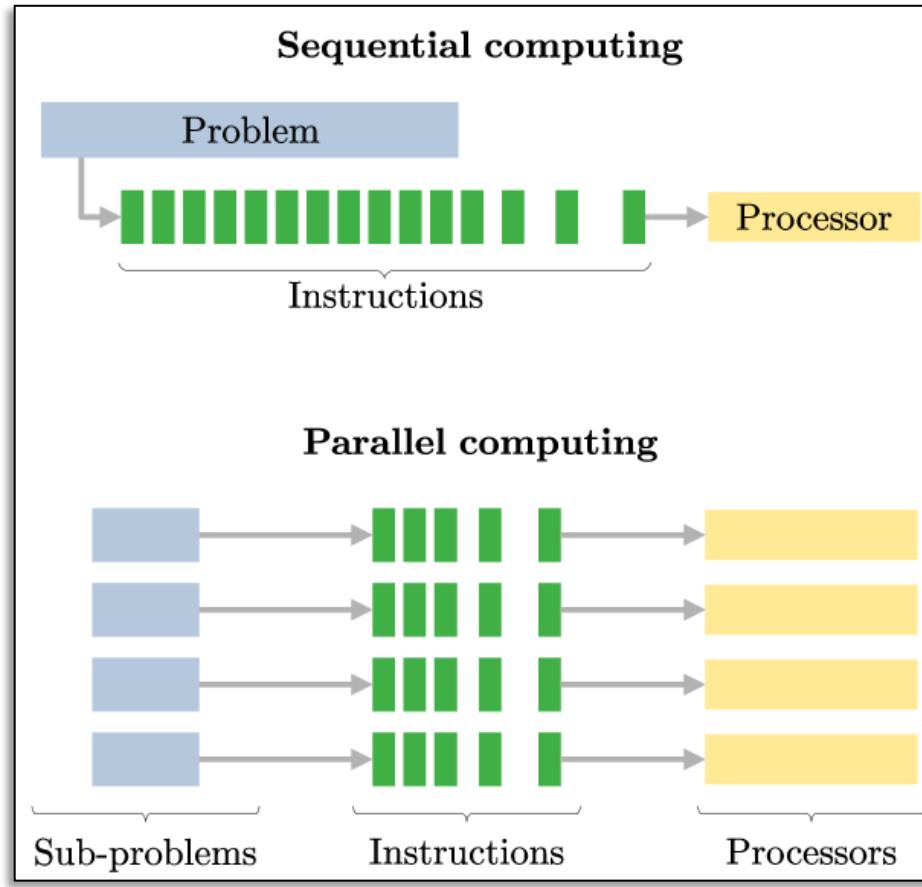
(b) 聚云

**多云 (Multi cloud)** 以与云功能特性无关的方式使用多个云提供商

**聚云 (Poly cloud)** 在于有机地使用多个云提供商的多种专业功能

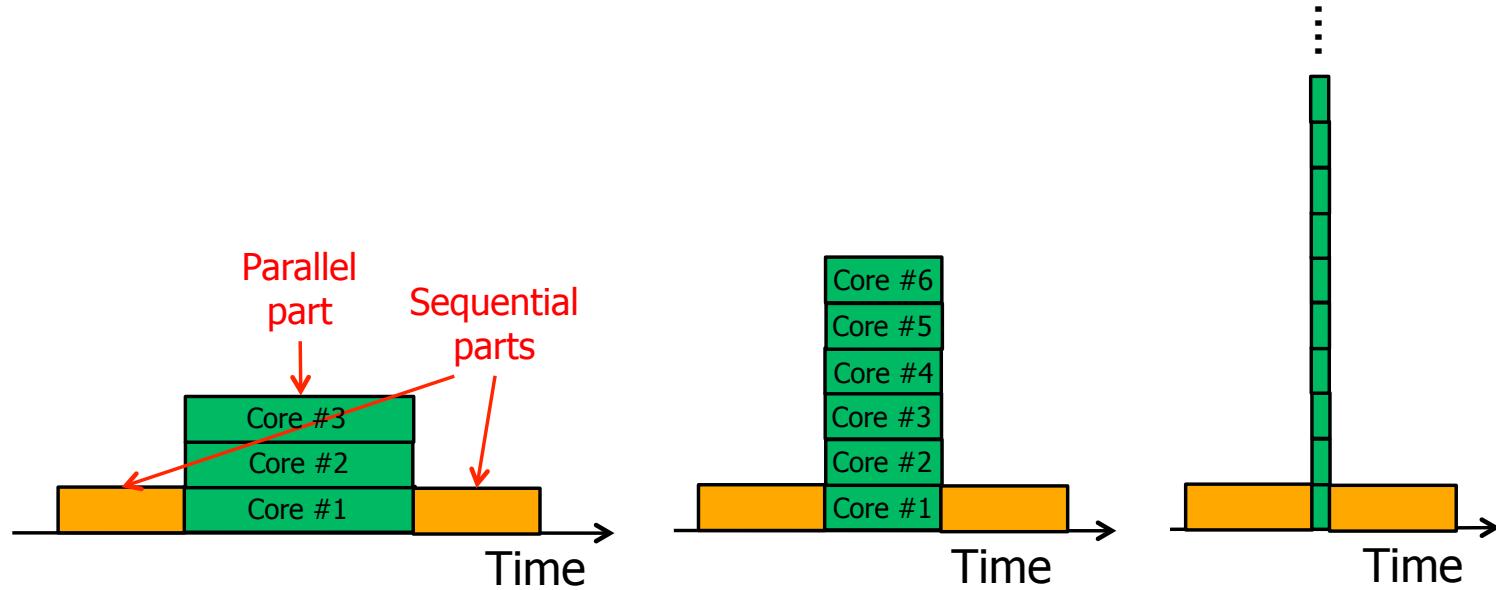
# 如何提高计数的效率?

## 并行计算! Parallel Computing!



并行计算是一种**计算模型**，指同时使用多种计算资源解决计算问题，以提高计算速度和效率

# 阿姆达尔定律 Amdahl' s Law

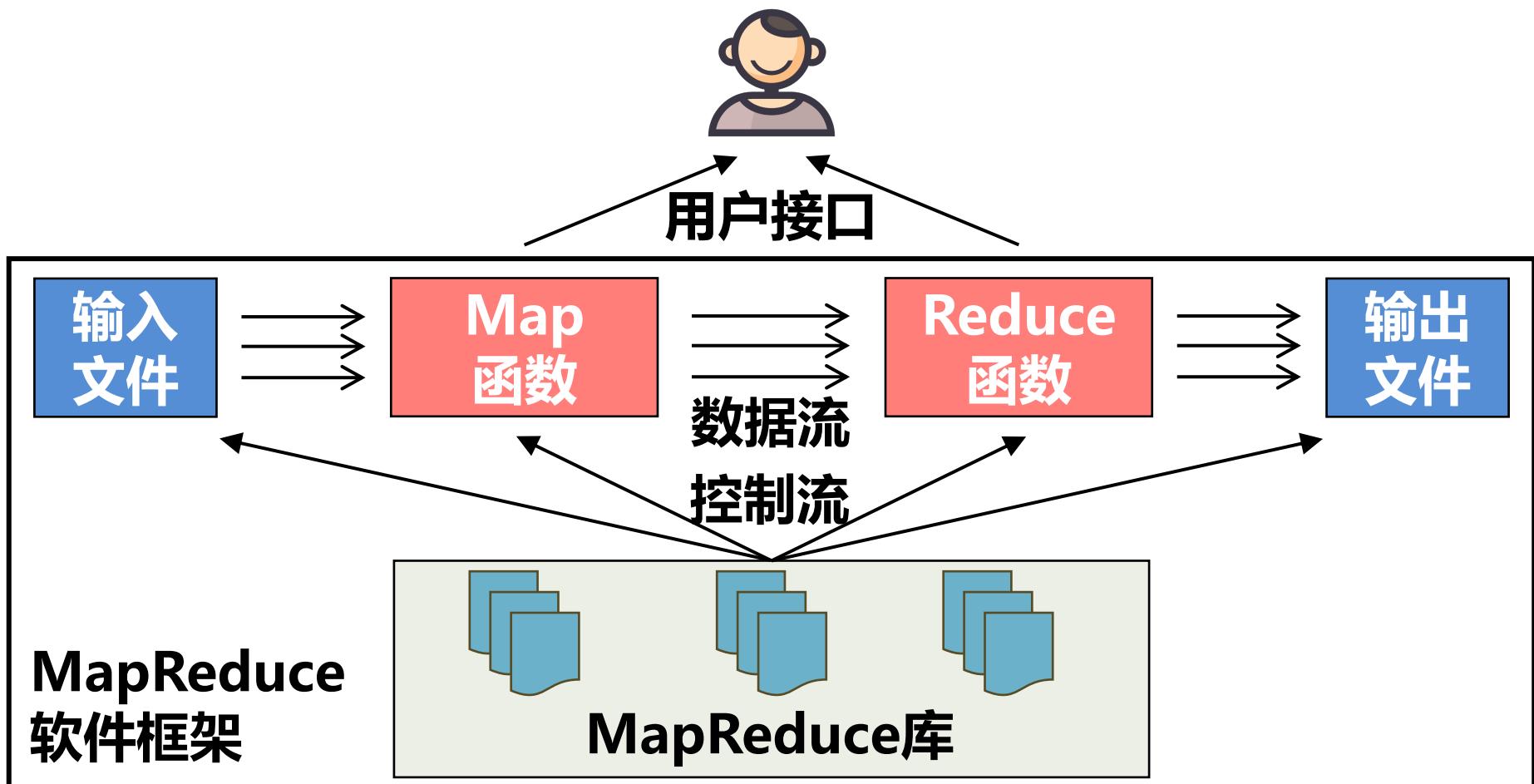


□通常，一个程序并不是所有部分都可以并行化处理

□假设某程序可并行处理的部分占比为  $f$ ，并行处理的节点个数为  $p$ ，则并行的加速比  $S$  为：

$$S = \frac{1}{(1 - f) + \frac{f}{p}}$$

# MapReduce是一个软件框架，简化复杂编程



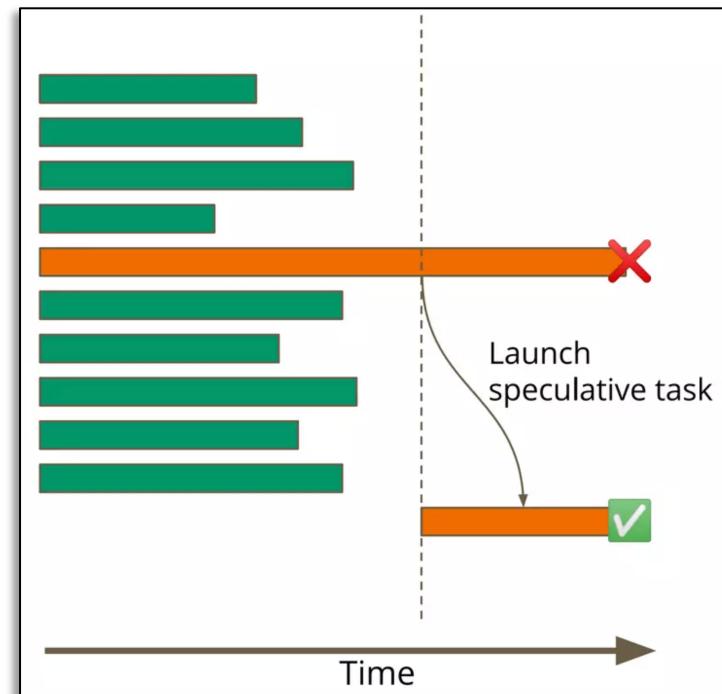
底层硬件基础设施 (对用户不可见)

# MapReduce 深度优化

## 利用空闲节点避免应用进度滞后

MapReduce 的一个关键优点是自动处理故障，若一个节点崩溃，则在另一台机器上重新运行其任务

同样重要的是，若一个节点  
**可用但表现不佳**，称之为滞  
后节点 (straggler)，则会在  
另一台机器上运行其任务的  
副本（也称为“备份执行”），  
以更快地完成计算



# MapReduce 深度优化

## 利用空闲节点避免应用进度滞后

如何选择任务进行备份执行?

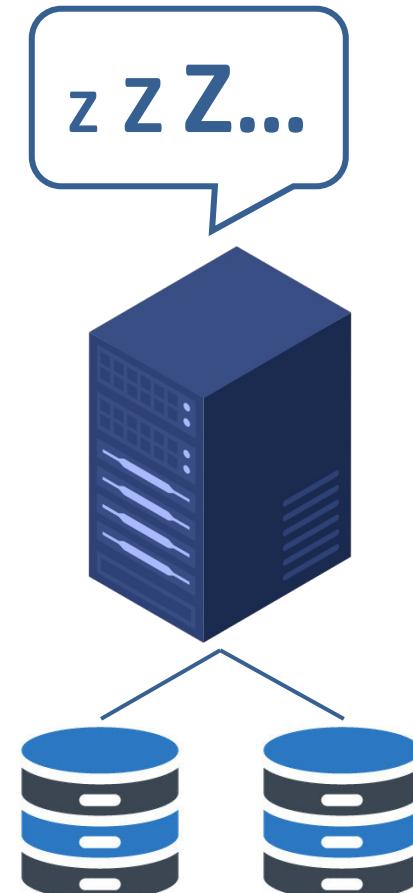
已半途失败的任务

未开始执行的任务

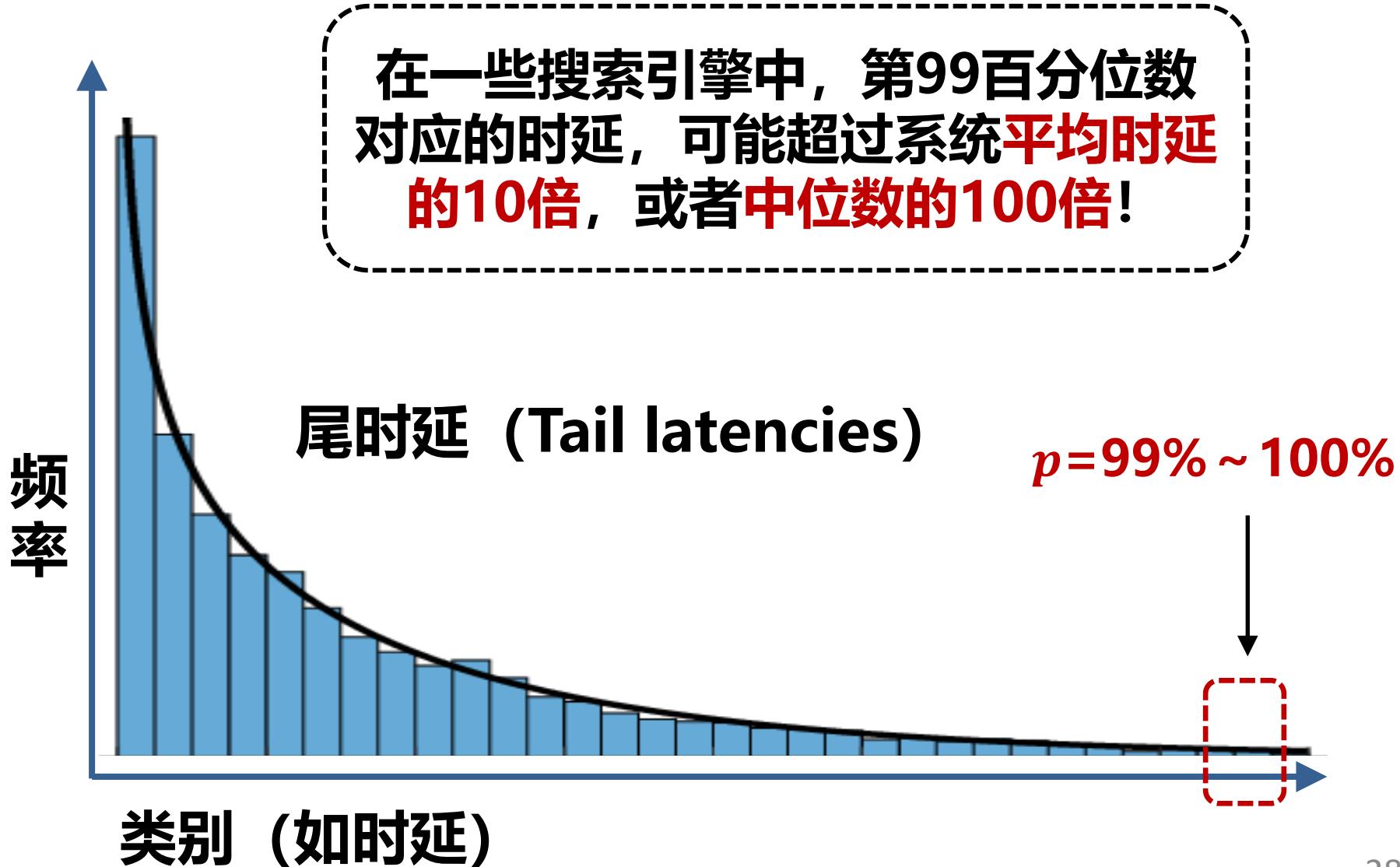
推测执行

优先级

创建副本

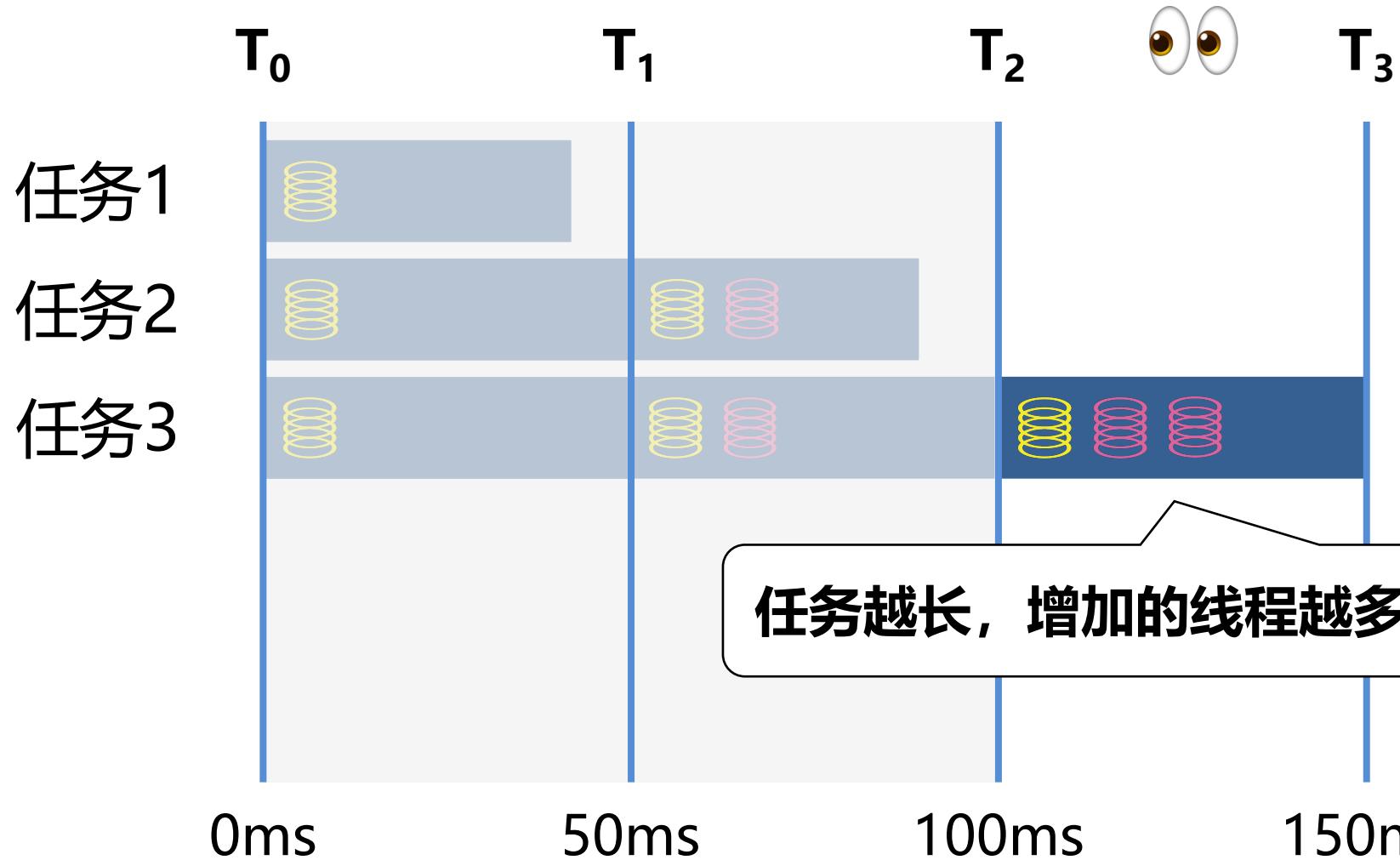


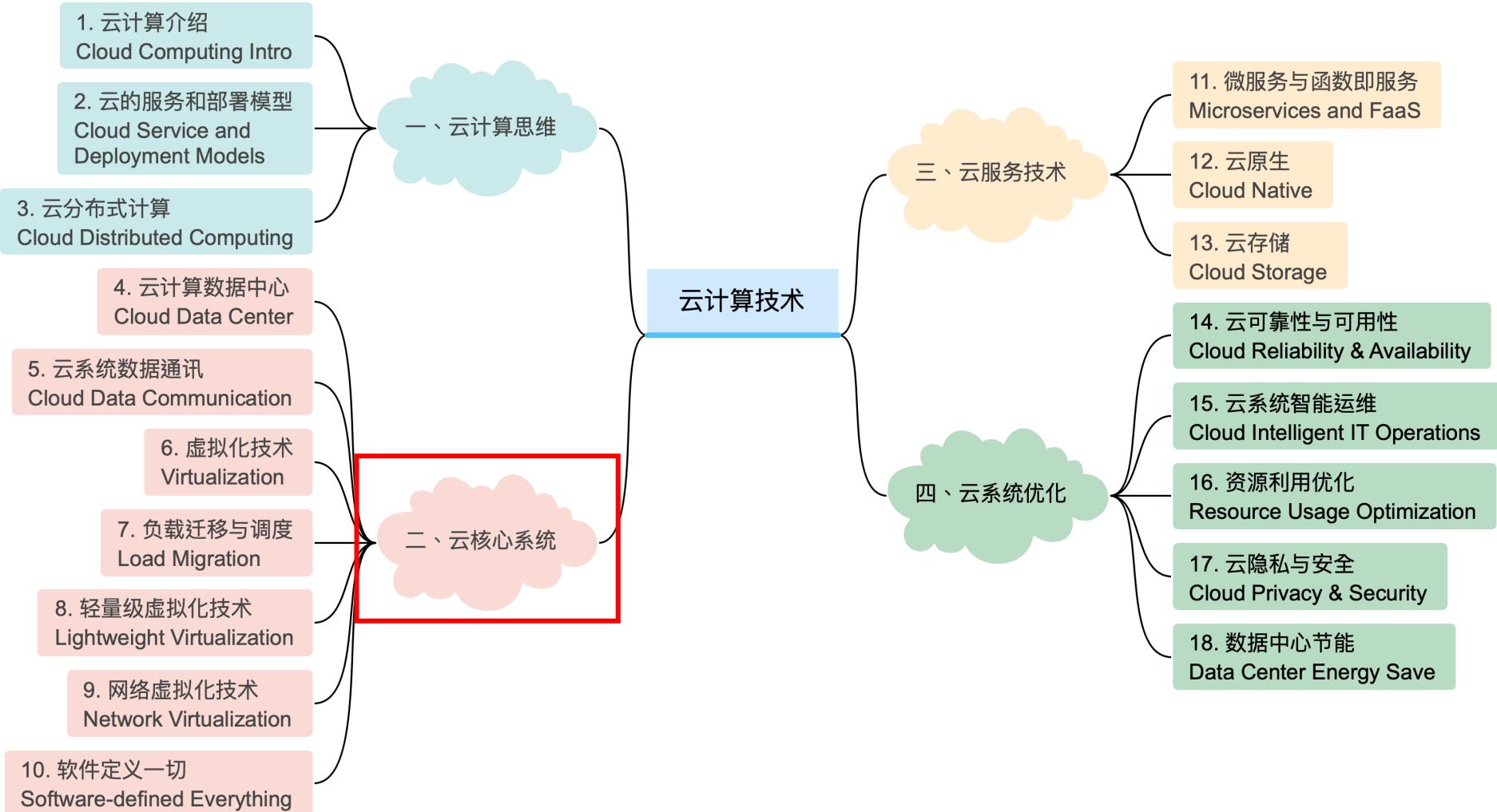
# 计算中的长尾



# 改善长尾问题

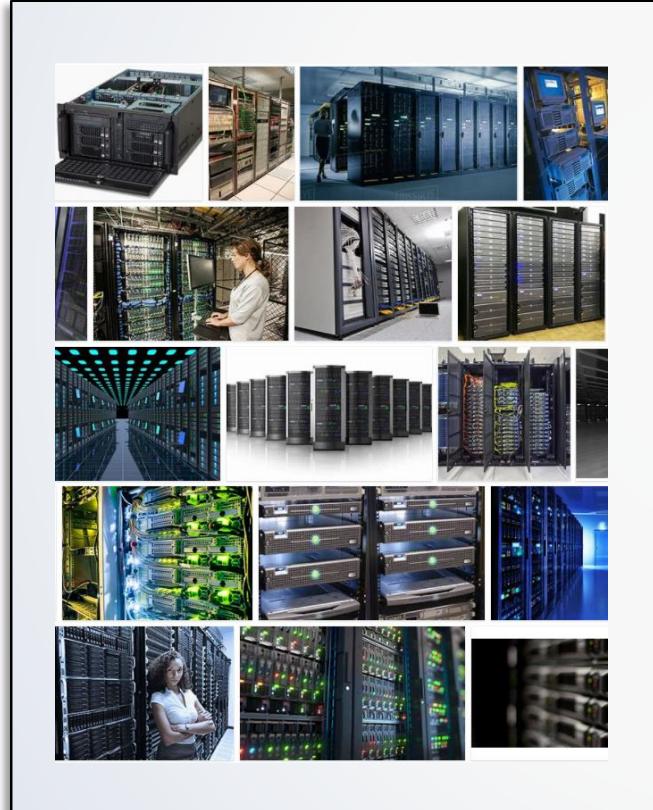
## Few-to-Many (FM) 漸增式调度算法





# 数据中心特点

IDC 使用商用设备(Commodity hardware)降低成本



价格相对低廉

易于大量获取

可替代性较好

灵活性较高

优势

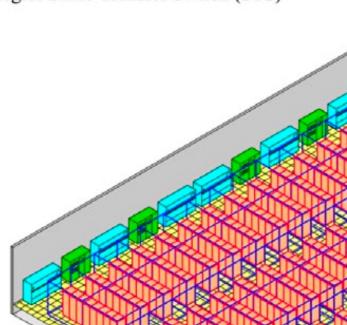
市面现成的硬件设备 commodity off-the-shelf (COTS) hardware

# 数据中心剖析

## 电力分配单元

### Power Distribution Unit (PDU)

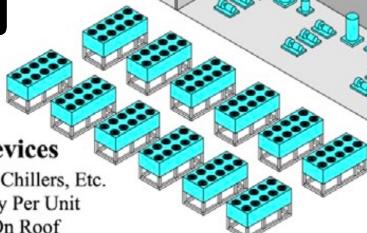
- Typical Capacities Up To 225 kVA Per Unit
- Redundancy Through Dual PDU's With Integral Static Transfer Switch (STS)



### Colocation Suites

- Modular Configuration For Flexible Suite Sq.Ft. Areas.
- Suites Consist Of Multiple Cabinets With Secured Partitions (Cages, Walls, Etc.)

## 机架托管套间



### Heat Rejection Devices

- Drycoolers, Air Cooled Chillers, Etc.
- Up To 400 Ton Capacity Per Unit
- Mounted At Grade Or On Roof
- N+1 Design

## 热量排放设备

## 计算机空气处理单元

### Computer Air Handling Unit (CRAC)

- Up To 30 Ton Sensible Capacity Per Unit
- Air Discharge Can Be Upflow Or Downflow Configuration
- Downflow Configuration Used With Raised Floor To Create A Pressurized Supply Air Plenum With Floor Supply Diffusers

### Individual Colocation Computer Cabinets

- Typ. Cabinet Footprint (28"W x 36"D x 84"H)
- Typical Capacities Of 1750 To 3750 Watts Per Cabinet

## 个体机柜

## 应急柴油发电机

### Emergency Diesel Generators

- Total Generator Capacity = Total Electrical Load To Building
- Multiple Generators Can Be Electrically Combined With Paralleling Gear
- Can Be Located Indoors Or Outdoors At Grade Or On Roof.
- Outdoor Applications Require Sound Attenuating Enclosures

## 燃油储存罐

### Fuel Oil Storage Tanks

- Tank Capacity Dependant On Length Of Generator Operation
- Can Be Located Underground Or At Grade Or Indoors

## 不间断电源系统

### UPS System

- Uninterruptible Power Supply Modules
- Up To 1000 kVA Per Module
- Cabinets And Battery Strings Or Rotary Flywheels
- Multiple Redundancy Configurations Can Be Designed

### Electrical Primary Switchgear

- Includes Incoming Service And Distribution
- Direct Distribution To Mechanical Equipment
- Distribution To Secondary Electrical Equipment Via UPS

## 电气主开关设备

## 泵房

### Pump Room

- Used To Pump Condenser/Chilled Water Between Drycoolers And CRAC Units
- Additional Equipment Includes Expansion Tank, Glycol Feed System
- N+1 Design (Standby Pump)

# 空调系统节能技术

## 1. 高温回风空调系统

根据不同出水温度下的制冷和能耗，对应的出水温度（即空调回风温度）提高 $1^{\circ}\text{C}$ ，空调系统约节能3%。

## 2. 低能耗加湿系统

将纯净的水直接喷洒在多孔介质或者空气中，形成颗粒极小的水雾，由送风气流送出。

## 3. 自然冷空调系统

使用室外自然风直接带走机房的IT设备的散热，减少了机械制冷系统中最大的压缩耗能环节，压缩机制冷系统的EER由 $2 \sim 3.5$ 提高到 $10 \sim 15$ ，节能空间巨大。

# 用电效能

## 数据中心能源效率衡量标准：用电效能 PUE



绿网联盟

### Power Usage Effectiveness (PUE)

$$PUE = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$

基准是2，比值越接近1，表示数据中心的能源利用率越高

# 全球最节能的 5 个数据中心

1. 雅虎“鸡窝”式数据中心

PUE = 1.08

2. Facebook 数据中心

PUE = 1.15

3. 谷歌比利时数据中心

PUE = 1.16

4. 惠普英国温耶德数据中心

PUE = 1.16

5. 微软都柏林数据中心

PUE = 1.25

# 集装箱数据中心节能技术

从绿色节能的角度看，集装箱数据中心也采用了诸多良好的设计提高数据中心的能效比

1 缩短送风距离

2 提高冷通道温度

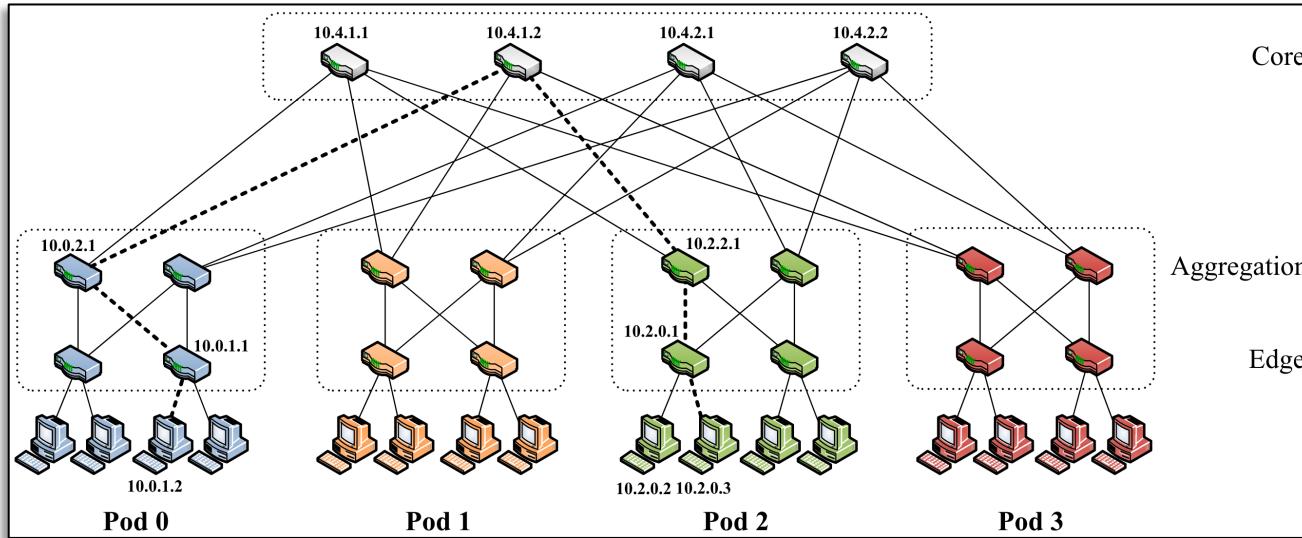
3 冷/热通道完全隔离

4 隔热保温材料

5 自然冷却功能



# Fat-tree 网络架构



An example Fat-tree architecture with  $k=4$

## □ $k$ -ary Fat-tree:

- ✓ 每个交换机有  $k$  个端口（编号 0-3），网络包含  $k$  个 pod
- ✓ 每个 pod 包含两层交换机（汇聚和接入），每层有  $k/2$  个交换机
- ✓ 每个接入交换机分别与  $k/2$  个主机和  $k/2$  个汇聚交换机相连
- ✓ 网络中共有  $(k/2)^2$  个核心交换机，所有核心交换机的第  $i$  个端口连接第  $i$  个 pod
- ✓ 汇聚交换机的端口以  $k/2$  的划窗连接核心交换机
- ✓ 能支持  $k^3/4$  个主机，当  $k = 4$  时， $k^3/4 = 4^3/4 = 16$

# 升级版转发协议 – 两级路由表

## □ TCP/IP 路由表

- Destination: 目标主机 IP 地址
- Gateway: 到达目标主机的网关或下一跳地址

Routing tables						
Destination	Gateway	Flags	Refcnt	Use	Interface	
140.252.13.65	140.252.13.35	UGH	0	0	emd0	
127.0.0.1	127.0.0.1	UH	1	0	lo0	
default	140.252.13.33	UG	0	0	emd0	
140.252.13.32	140.252.13.34	U	4	25043	emd0	

## □ 两级路由表

### ▪ 第一级是前缀查找

✓ 用于将拓扑向下路由到端主机

### ▪ 第二级是后缀查找

✓ 用于向核心路由

✓ 扩散和分散交通

✓ 通过对相同的端主机使用相同的端口来维护数据包顺序

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

→

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

# 虚拟化的优势：整合服务器

口传统数据中心服务器资源利用率低

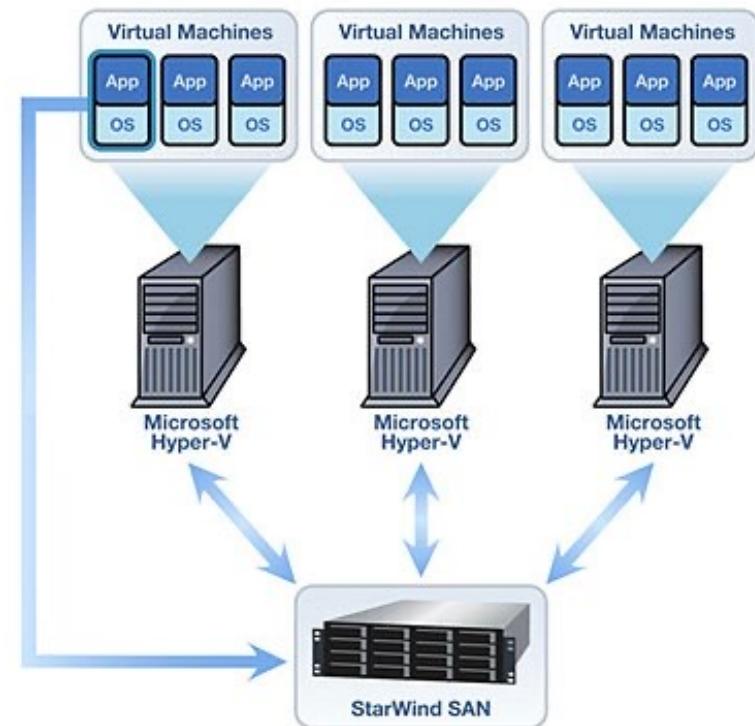
- CPU 平均利用率仅为**约 20%**

口提高物理机资源利用率

- 一个物理机上整合多个虚拟机

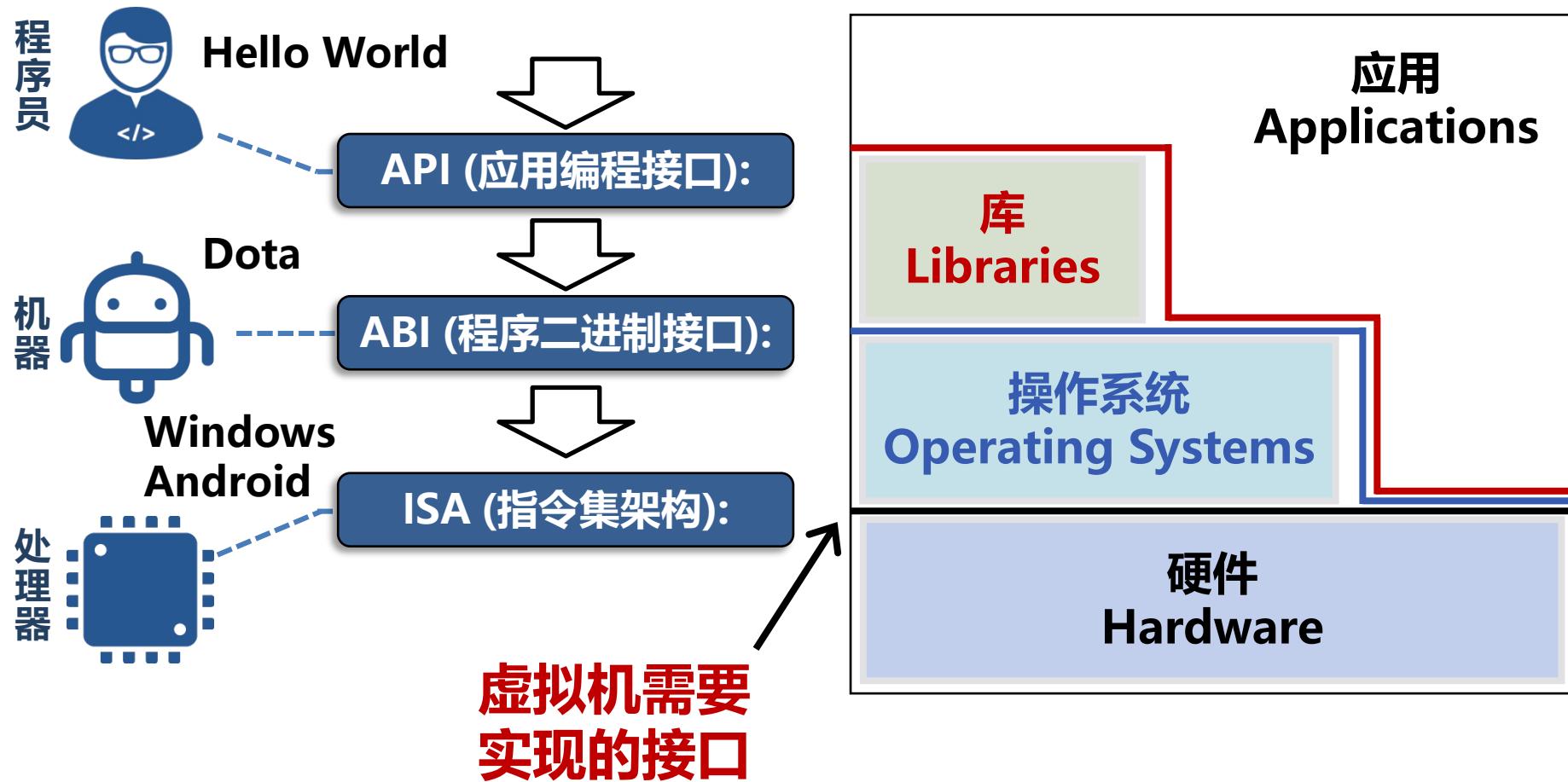
口降低云提供商成本

- 基于用户**错峰使用**特性，"超售"系统资源



# 不同接口层次面对的对象

关键资源下沉，逐层通过 **接口（Interface）** 交互



# 这个过程有什么问题？

- 捕捉下陷指令并处理是为了防止影响整个系统
- 然而，并不是所有对系统有影响的指令均会下陷！

## 特权指令 Privileged instructions

在用户态执行时会触发下陷的指令，包括主动触发下陷的指令和不允许在用户态执行的指令。

## 敏感指令 Sensitive instructions

管理系统物理资源或者更改CPU状态的指令：读写特殊寄存器、读写敏感内存、I/O指令。

当所有敏感指令都是特殊指令，在非特权级执行时都会发生下陷时，才为可虚拟化架构

# 各种虚拟化技术的比较

	全虚拟化	硬件辅助	半虚拟化
技术	二进制翻译	虚拟化扩展	Hypercall
兼容性	好	好	差
性能	较好	好	分场合

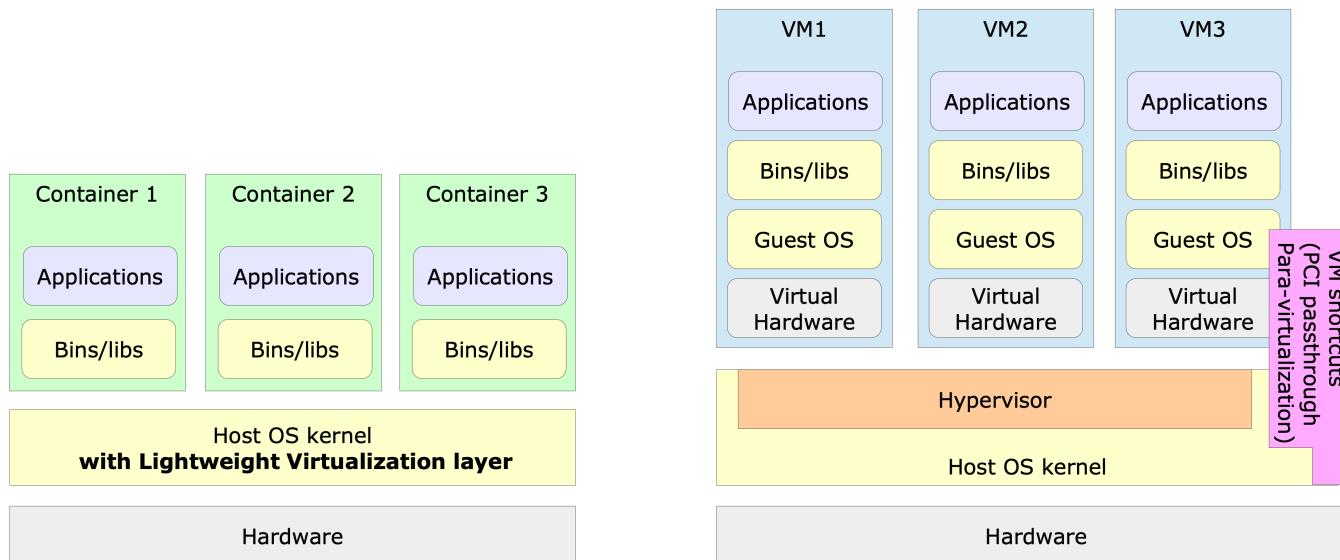
# Containers overview

□ 容器提供了轻量级虚拟化（基于 cgroups 和 namespaces）

- 允许隔离进程和资源，且没有完全虚拟化的复杂性

□ Linux 容器是一种操作系统级虚拟化方法

- 用于在单个主机上运行多个独立的Linux系统（容器）
- Linux 内核在所有容器中共享



Containers vs. Hypervisors

# Docker in a nutshell



# Kubernetes

开源的容器编排 (orchestration) 系统，用于自动化部署、扩展和管理容器化应用程序

口K8s 的主要功能包括：

- 服务发现和负载均衡
- 存储管理
- 自动部署和回滚
- 容器健康管理
- 密钥和配置管理



kubernetes

How to pronounce Kubernetes?

coo-ber-net-ees

(even when it is shortened as K8s)

# Kubernetes 关键理念

## □ API 与实现解耦

- 只要符合 API 规范，那么相同对象就可以根据上下文（例如，云提供商）以不同的方式实现

## □ 声明式配置 (Declarative configuration)

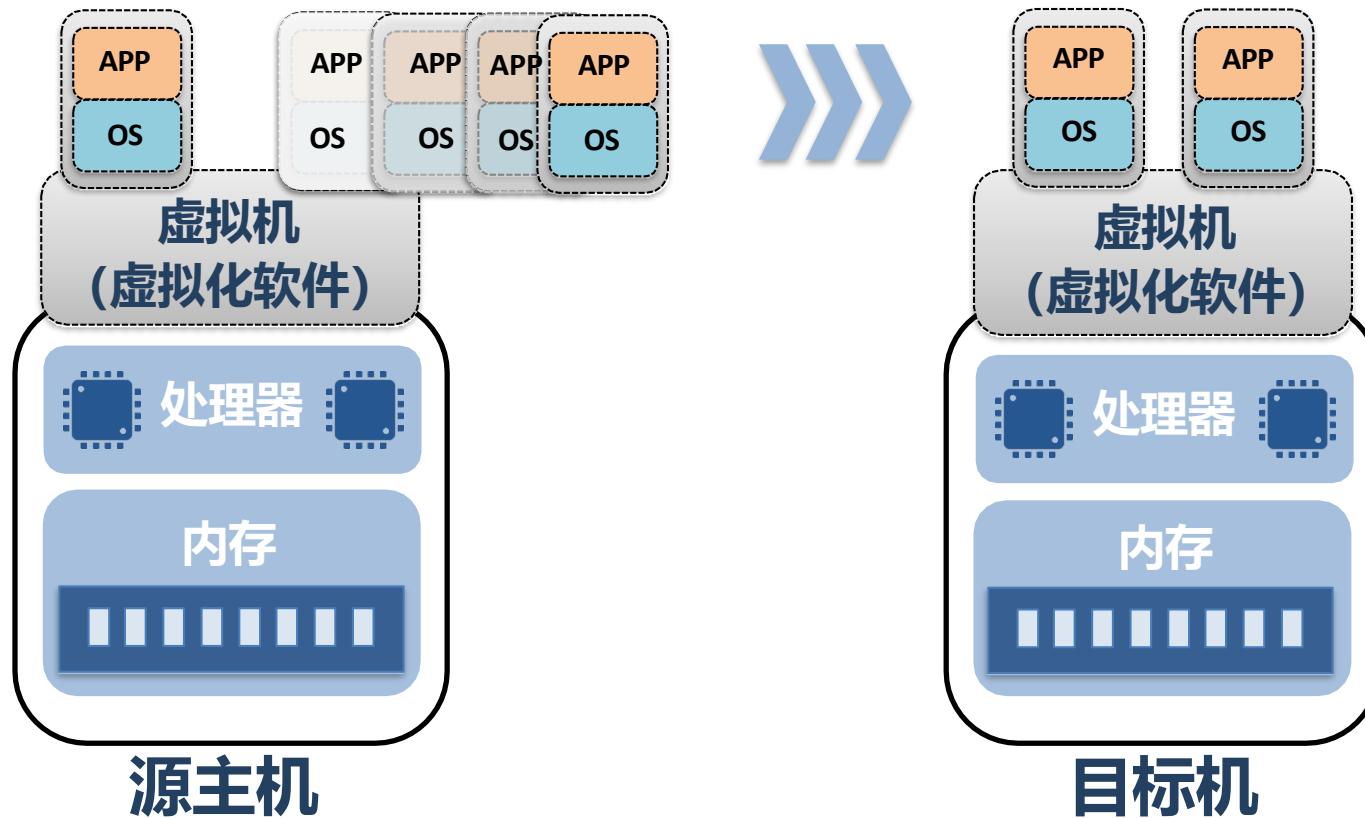
- 遵循“基础设施即代码 (Infrastructure as Code)”的理念
- 仅描述期望的系统状态，无需定义控制流

## □ 以控制循环为导向的方法 (Control Loop-oriented)

- 使对象的状态收敛于期望的状态
- 持续检查状态是否符合期望

# 迁移条件

负载迁移能力靠虚拟化软件支撑



# 迁移条件

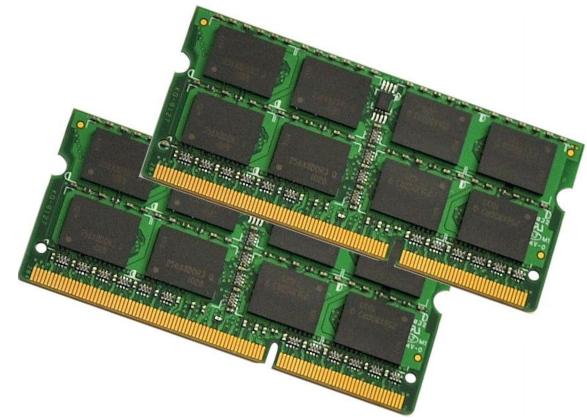
□ 三种需要迁移的资源



**Storage**



**Network**



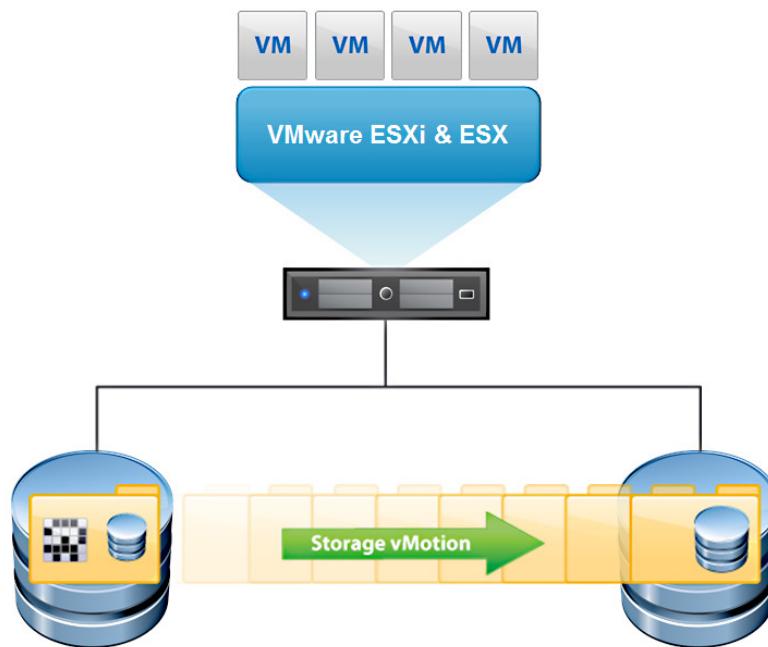
**Memory**

# 存储迁移

口迁移存储的最大障碍在于需要占用大量时间和网络带宽

口通常的解决办法是共享数据和文件系统，而非真正迁移

口目前大多数集群使用 NAS (Network Attached Storage, 网络连接存储) 作为存储设备共享数据



# 内存在线迁移

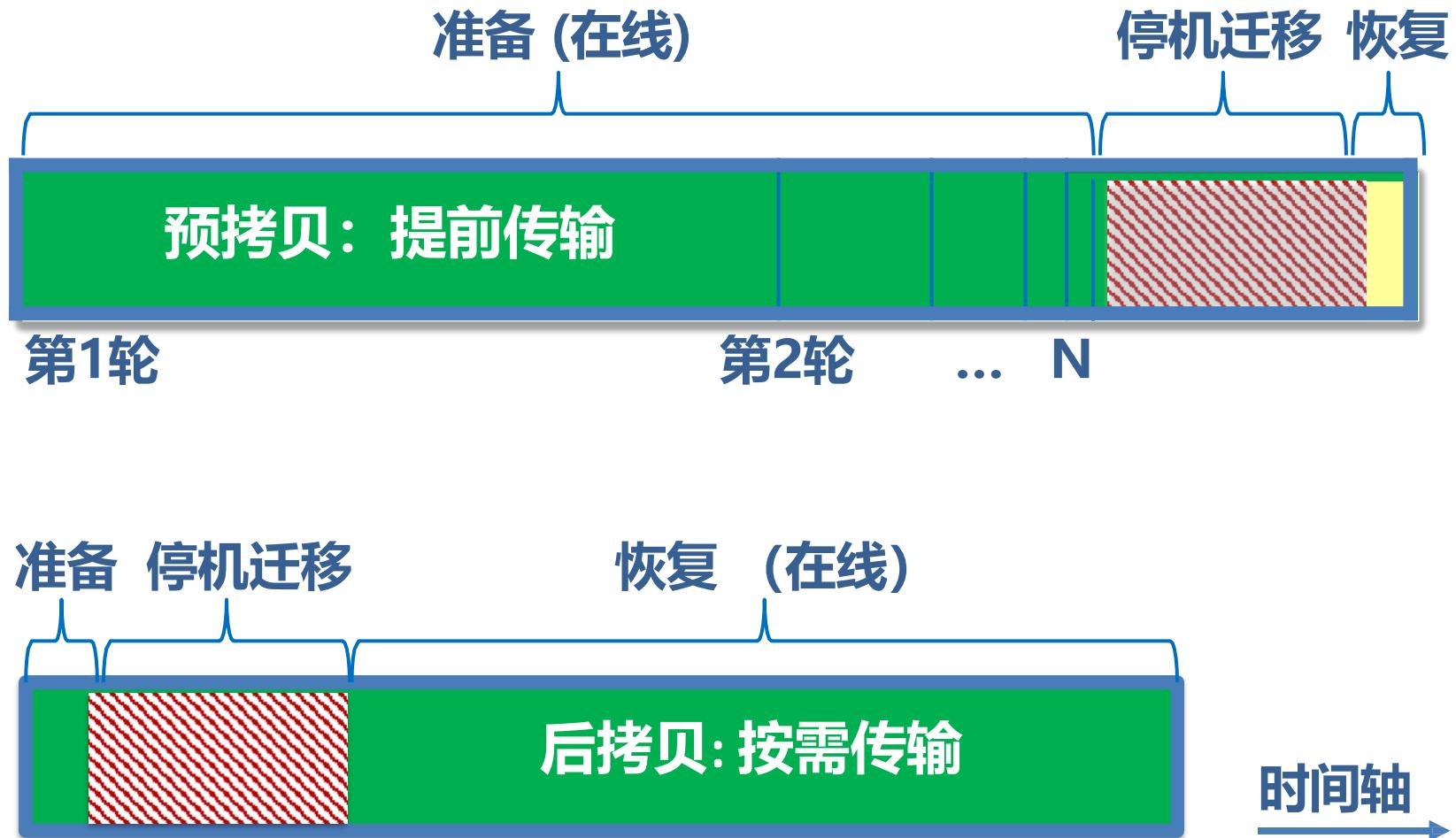
## 方法一：Pre-copy 预拷贝

- VMM 拷贝源虚机的所有内存页到目的节点
- 若在此期间出现 dirty pages，则重新拷贝
- 当 dirty pages 的数量较少时，短暂暂停虚机并拷贝剩下的所有页面

## 方法二：Post-copy 后拷贝

- 先暂停虚机，并将最小的执行状态集合（CPU状态、寄存器等）传输到目的节点
- 虚机在目的节点启动
- 源节点同步将剩下的内存页传输到目的节点
- 若期间虚机在目的节点的缺页中断将被重定向回源节点

# 内存在线迁移



IP地址: 192.168.0.1  
MAC地址: AAAA  
子网掩码: 255.255.255.0  
默认网关: 192.168.0.254

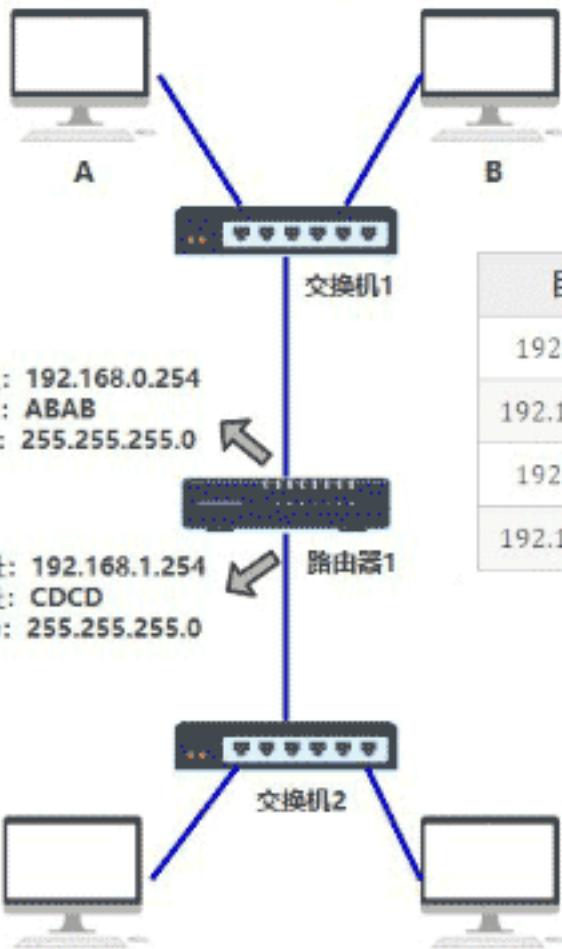
IP地址: 192.168.0.2  
MAC地址: BBBB  
子网掩码: 255.255.255.0  
默认网关: 192.168.0.254

IP地址: 192.168.0.254  
MAC地址: ABAB  
子网掩码: 255.255.255.0

IP地址: 192.168.1.254  
MAC地址: CDCD  
子网掩码: 255.255.255.0

IP地址: 192.168.1.1  
MAC地址: CCCC  
子网掩码: 255.255.255.0  
默认网关: 192.168.1.254

IP地址: 192.168.1.2  
MAC地址: DDDD  
子网掩码: 255.255.255.0  
默认网关: 192.168.1.254



目的地址	端口
192.168.0.0/24	0
192.168.0.254/32	0
192.168.1.0/24	1
192.168.1.254/32	1

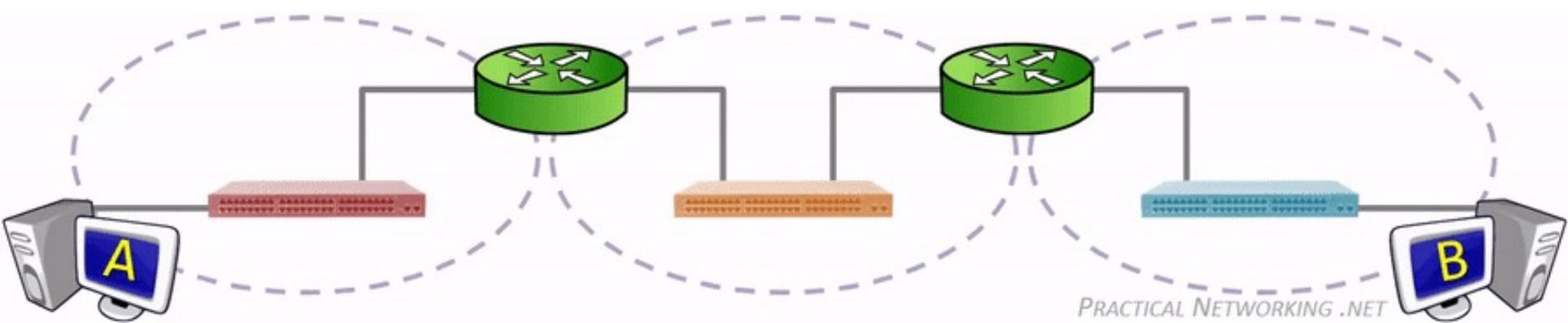
# 网络通信



路由器 (router), 三层网络, 基于 IP 地址



交换机 (switch), 二层网络, 基于 MAC 地址



## 局域网 LAN (Local Area Network)

### 局域网有什么问题?

# 局域网的问题

## 口广播风暴

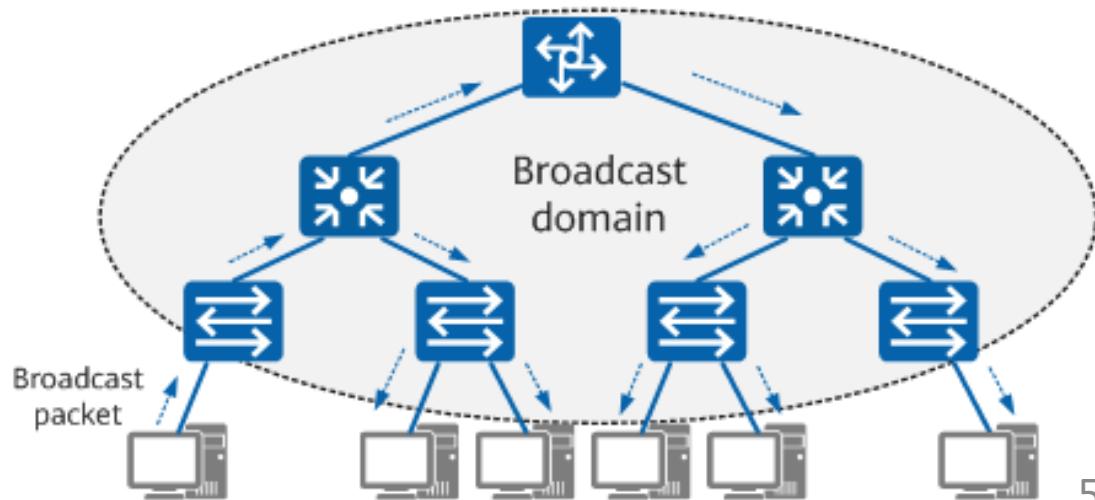
- 一个局域网为一个广播域，所有设备都可以收到广播消息
- 例如广播 ARP 包，进行服务发现等
- 当设备数量增多时，容易引发广播风暴

## 口安全和隐私

- 不同设备可以尝试访问或监听其他设备，存在安全隐患

## 口效率和管理问题

- 占用带宽且管理复杂

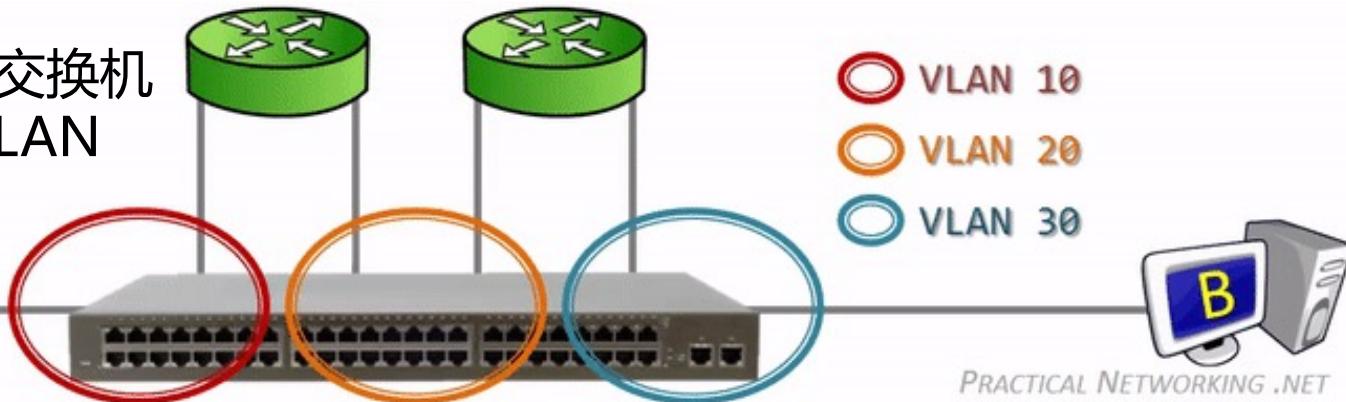


# 虚拟局域网 VLAN

## 口虚拟局域网 (Virtual Local Area Network)

- 将一个**物理二层网络**拆分为多个**逻辑虚拟网络**，相互独立
- 每个 VLAN 充当一个**单独的广播域**
- 同一 VLAN 中的主机能够直接相互通信
- 跨 VLAN 通信需要借助三层网络

可以在一个物理交换机中设置不同的 VLAN



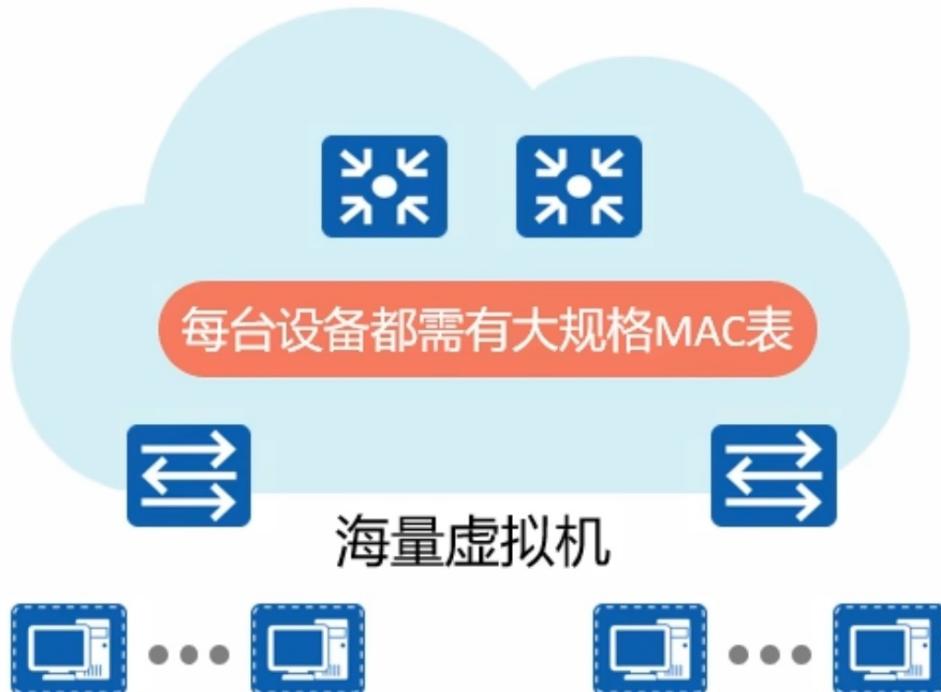
支持 VLAN 的交换机的 Mac 地址路由表格式：

VLAN# | MAC Address | Port

# VLAN 面临的问题

## □ 虚拟机规模受设备表项规则限制

- 服务器虚拟化后，VM 的数量比原有物理机发生了巨大的增长
- 二层设备的 MAC 地址表规则较小，无法满足快速增长的 VM



Vlan	Mac Address	Type	Ports
10	0001.c7ad.e316	DYNAMIC	Fa0/24
10	0002.4ab7.1701	DYNAMIC	Fa0/24
11	0001.c7ad.e316	DYNAMIC	Fa0/24
11	0002.4ab7.1701	DYNAMIC	Fa0/24
12	0001.c7ad.e316	DYNAMIC	Fa0/24
12	0002.4ab7.1701	DYNAMIC	Fa0/24
12	000d.bd94.6b58	DYNAMIC	Fa0/1
12	0060.3e5c.bald	DYNAMIC	Fa0/2
12	00e0.f934.e3c4	DYNAMIC	Fa0/3
13	0001.c7ad.e316	DYNAMIC	Fa0/24
13	0002.4ab7.1701	DYNAMIC	Fa0/24

Switch#

# VLAN 面临的问题

## □ 网络隔离能力限制

- VLAN Tag 只有 12 bits, 4094 个不同的 VID (2 个保留 ID)
- 对于大型虚拟化云服务场景，租户数量远大于 VLAN 个数
- 传统二层网络中的 VLAN 无法满足网络动态调整的需求

802.1Q Tag帧：



VLAN ID长12 bit, 仅能表示4096个逻辑单元。

大型数据中心需支持的租户规模远大于该数目

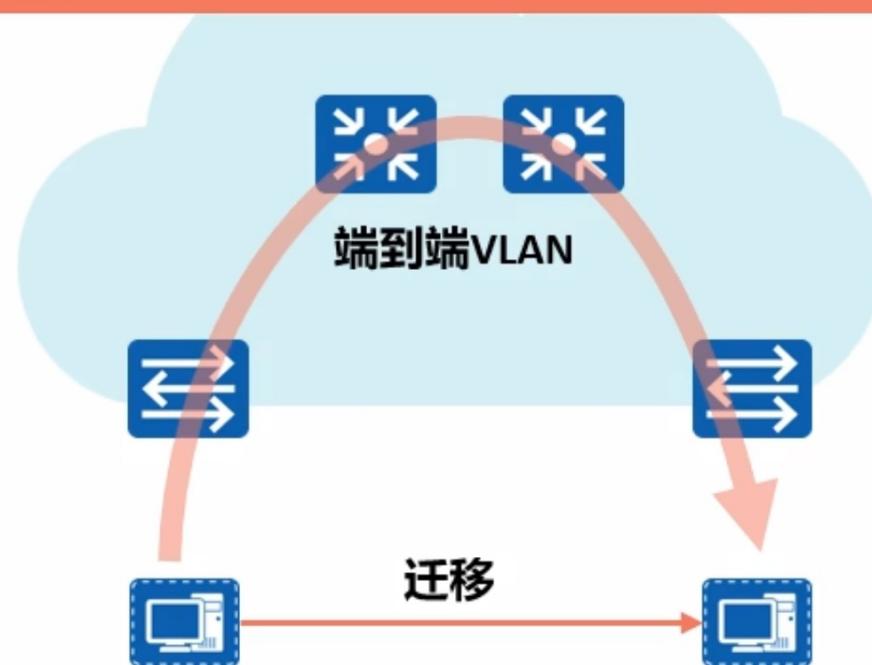
数据中心

# VLAN 面临的问题

## □ 虚拟机迁移范围受限

- 传统的二层网络将虚拟机迁移限制在较小的局部范围内
- 跨 VLAN 迁移会引起业务中断、信息丢失等

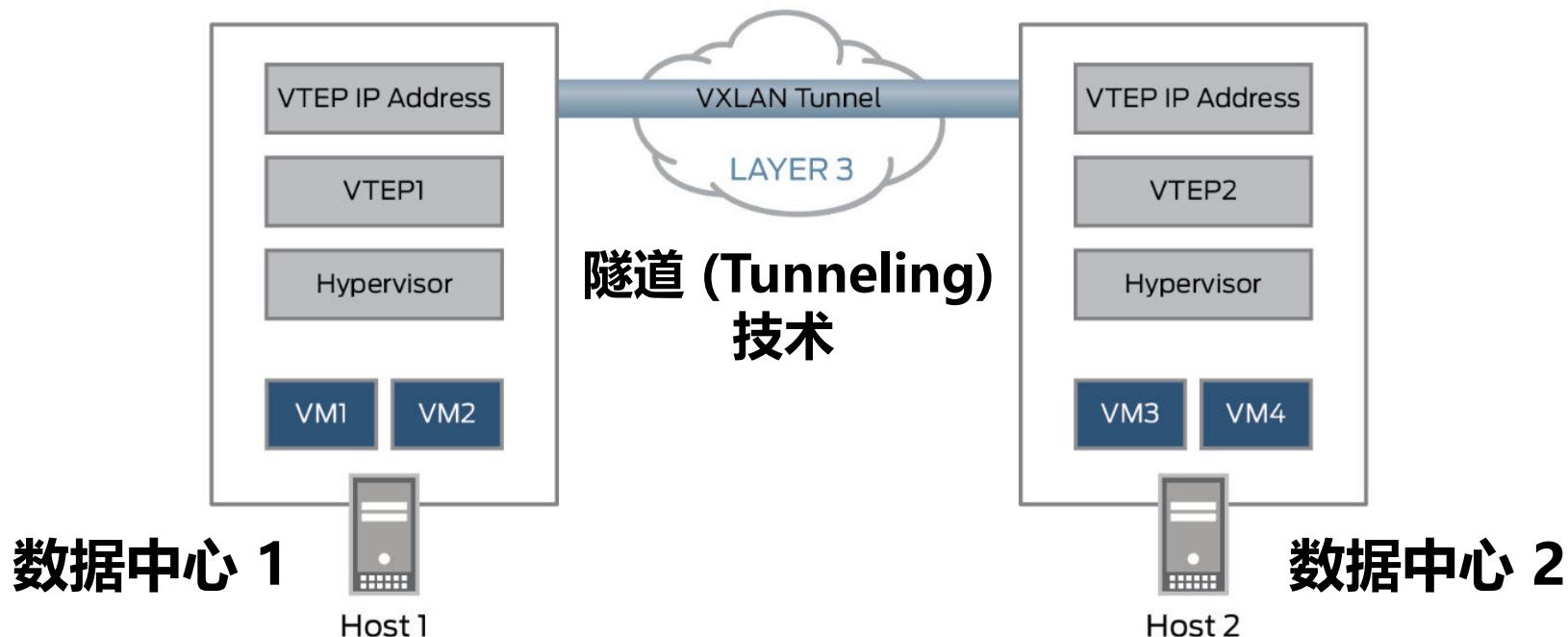
虚拟机只能在一个VLAN内迁移，VLAN范围存瓶颈



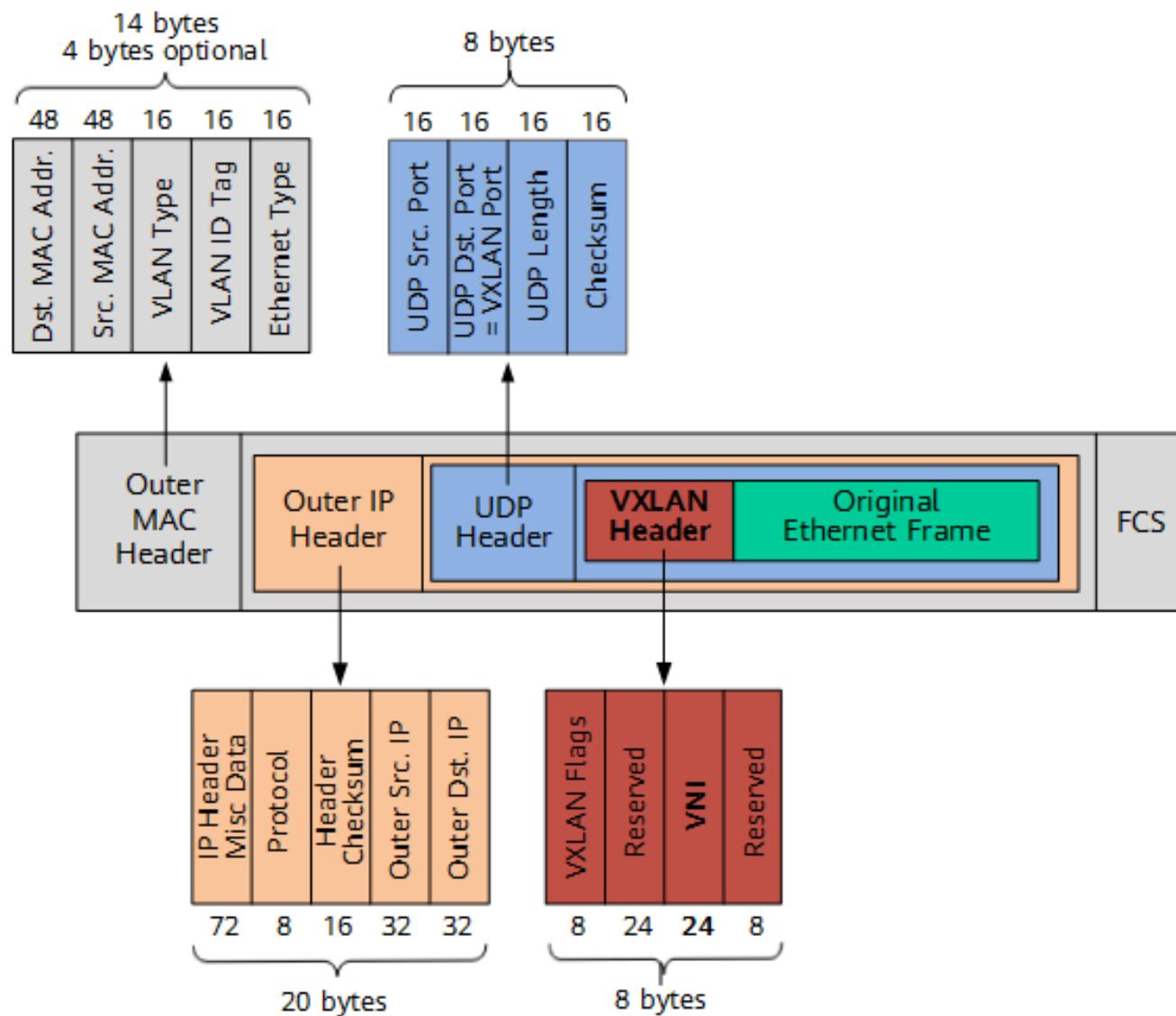
# VxLAN 简介

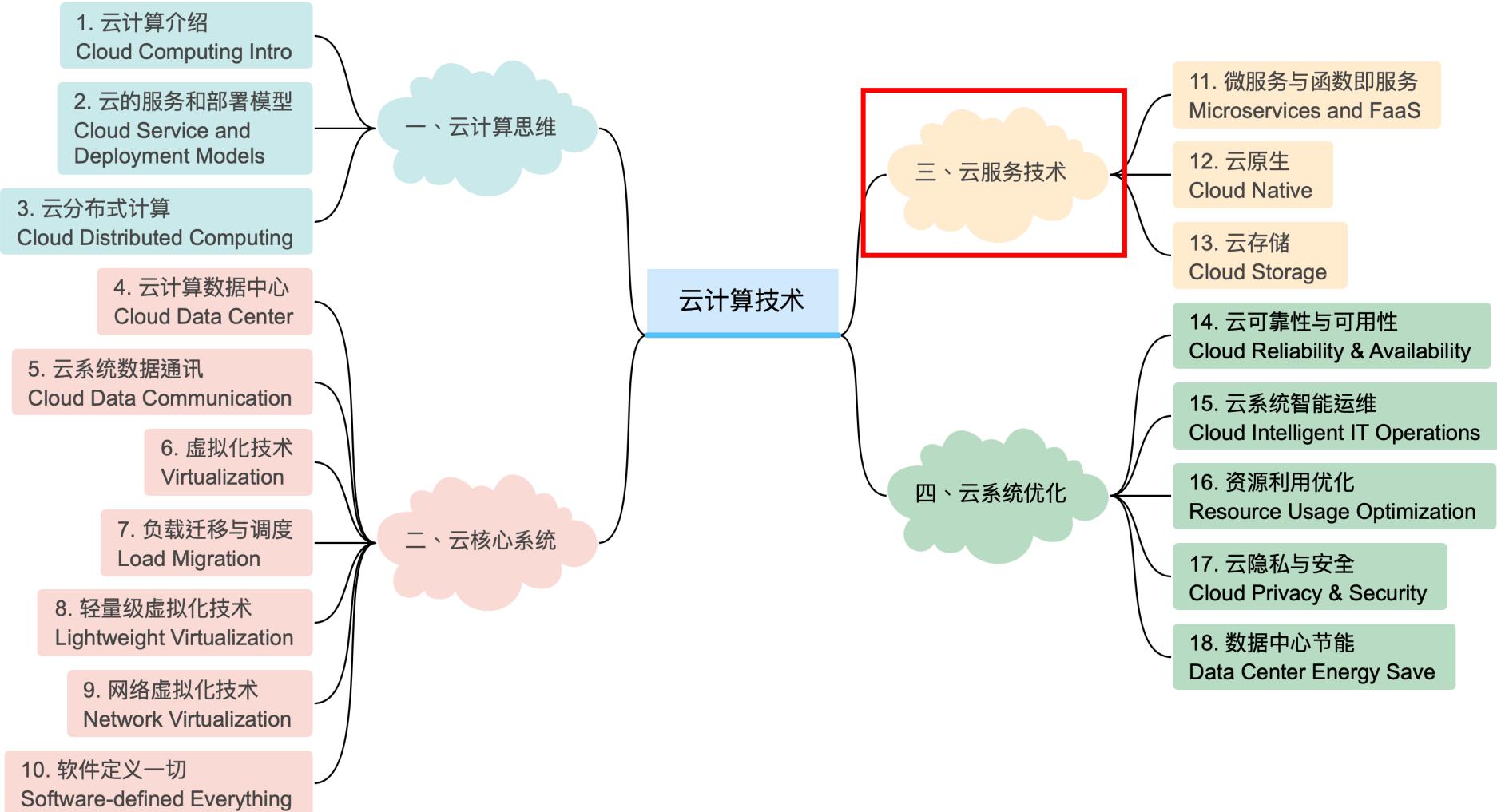
□ VxLAN 借助三层网络上扩展二层的连接

- 采用 **MAC in UDP 封装 (encapsulation)** 来延伸二层网络
- 将以太报文封装在 IP 报文之上，**像普通 IP 包一样通过路由在三层网络中传输**，无需关注虚拟机的 MAC 地址
- 通过路由网络，**虚拟机迁移不受网络架构限制**



# VxLAN 数据包





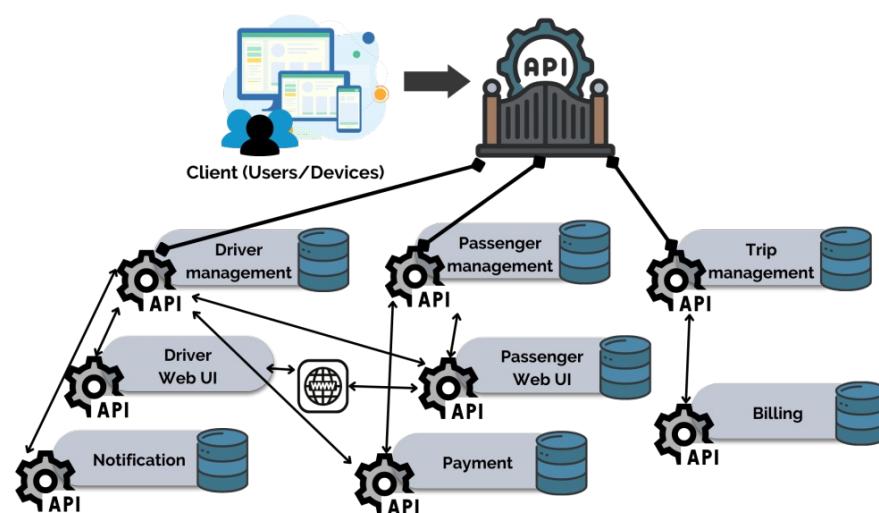
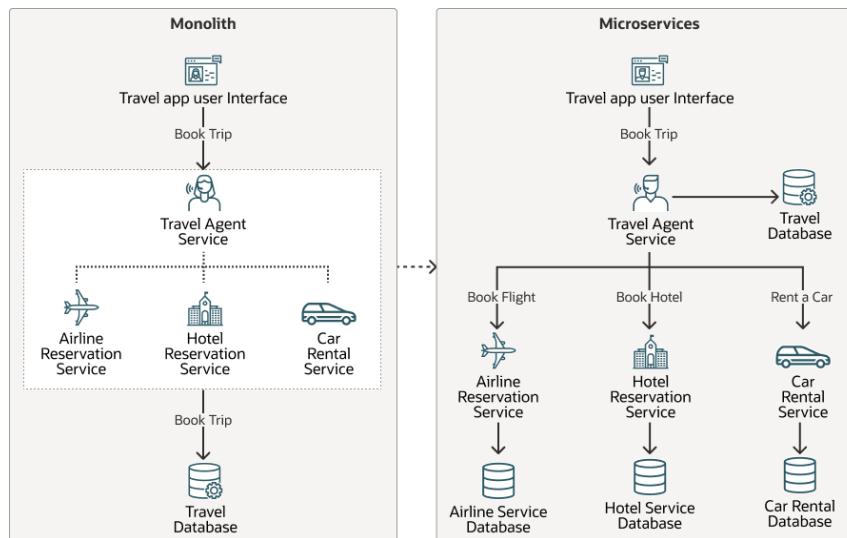
# 微服务 Microservices

## □ Microservices (from Wikipedia [2])

- Microservices are a **software development technique** that arranges an application as **a collection of loosely coupled services**

## □ Application

- A complex service provided to the user (e.g., e-commerce)



# 微服务架构的特点

“Service-oriented architecture  
composed of  
loosely-coupled elements  
that have  
bounded contexts”

# 基于微服务交互的架构评估

## Trace Analysis Based Microservice Architecture Measurement

Xin Peng\*<sup>†</sup>  
Fudan University  
China

Akasaka Isami\*  
Fudan University  
China

Chenxi Zhang\*  
Fudan University  
China

Xiaofeng Guo  
Alibaba Group  
China

Zhongyuan Zhao\*  
Fudan University  
China

Yunna Cui\*  
Fudan University  
China

该论文定义了 14 个架构度量标准来衡量微服务系统的  
**服务独立性和调用链复杂性**

### □ 服务独立性 (Service Independence)

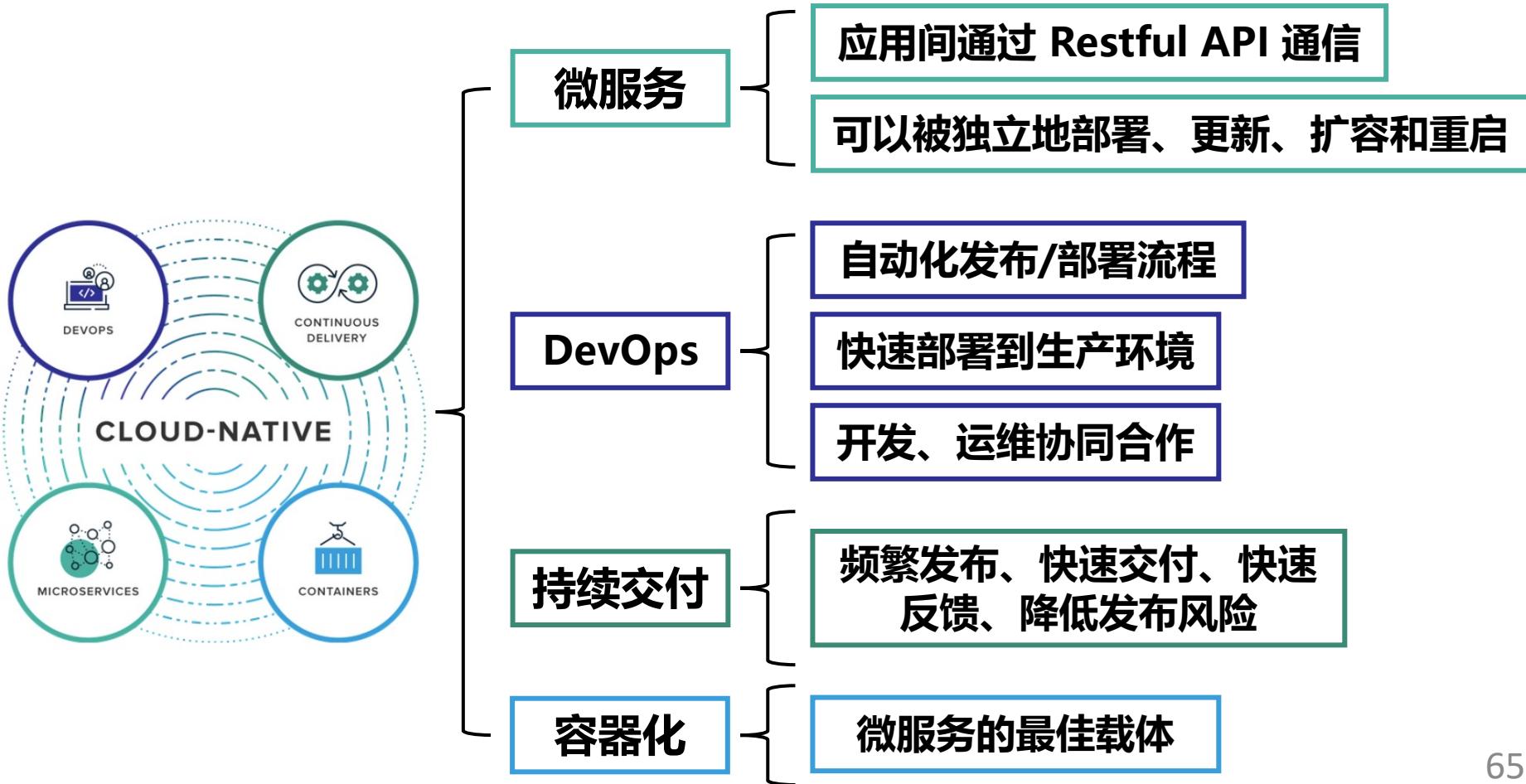
- 服务在开发 (Development)、演变 (Evolution) 和扩展 (Scaling) 方面如何独立于彼此？

### □ 调用链复杂性 (Invocation Chain Complexity)

- 服务的调用链有多复杂？

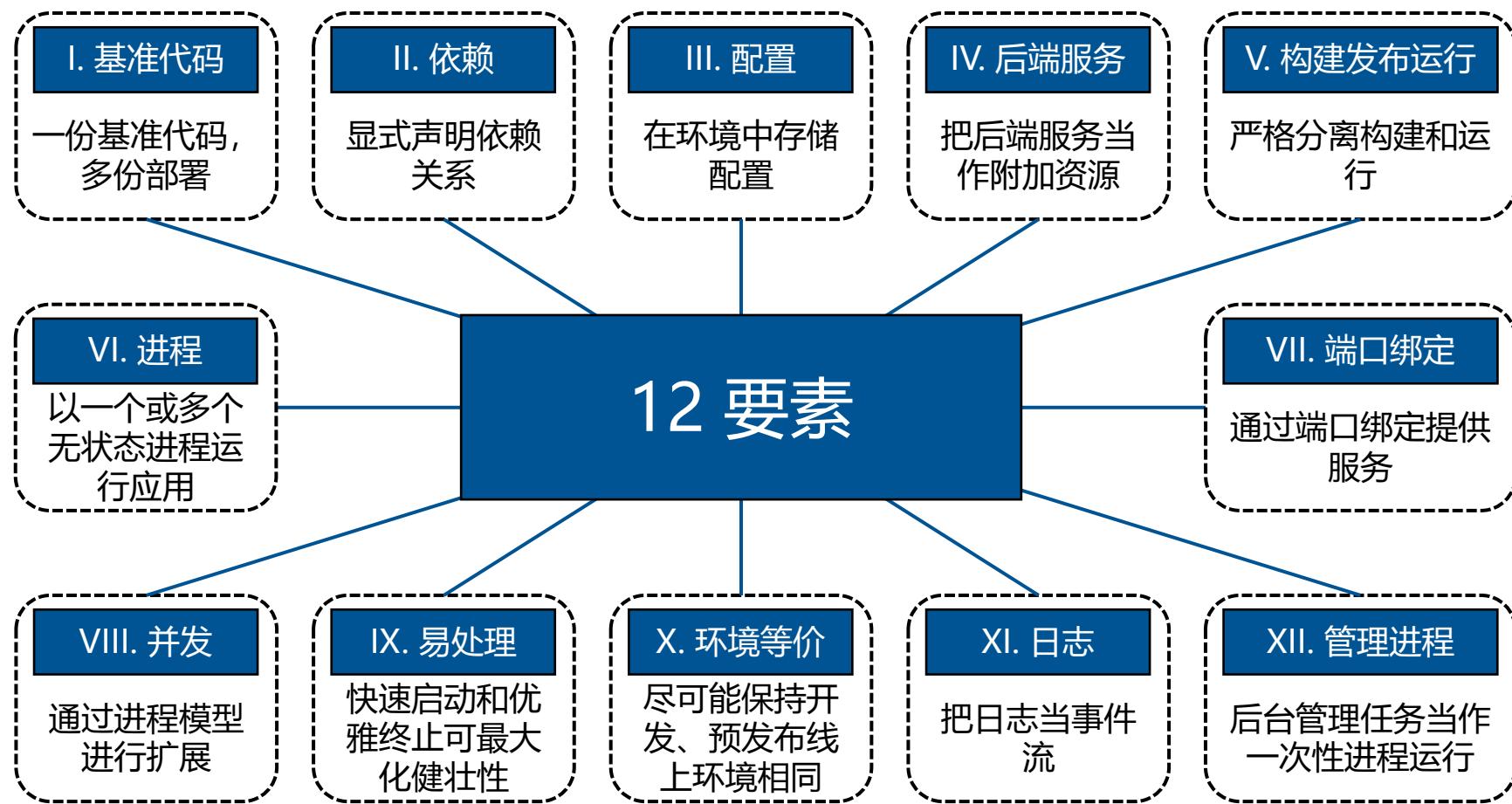
# 云原生代表技术

- 从宏观概念上讲，云原生是不同思想的集合，包含各种技术
- 这些技术能构建**容错性好、易于管理和便于观察**的松耦合系统



# 云原生应用开发实践的 12 要素

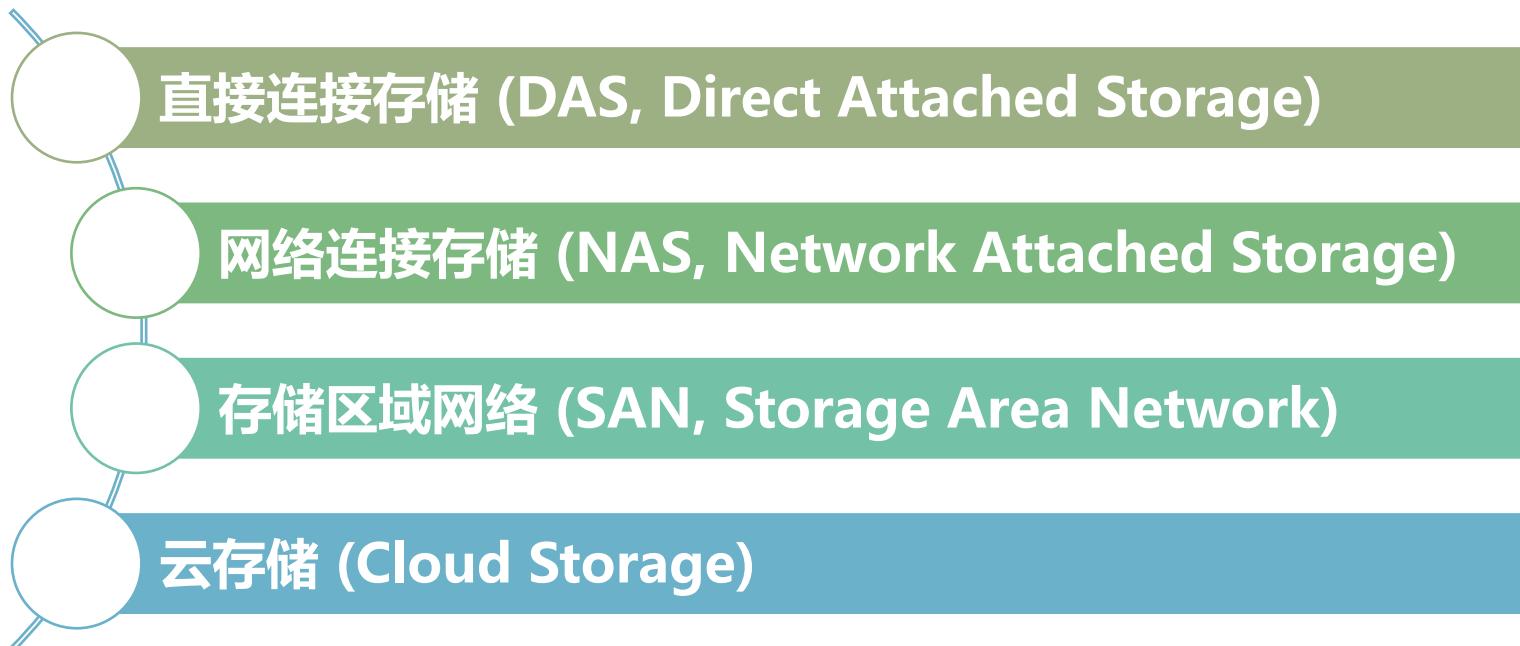
口云原生应用开发的 12 要素是由 Heroku 的联合创始人 Adam Wiggins 提出的，用于指导构建**可扩展和可维护**的云原生应用



# 存储模型

**存储模型 (Storage Model)** 是一种描述计算机数据存储结构和管理方式的概念模型

常见的存储模型包括：



# 共识 (Consensus)

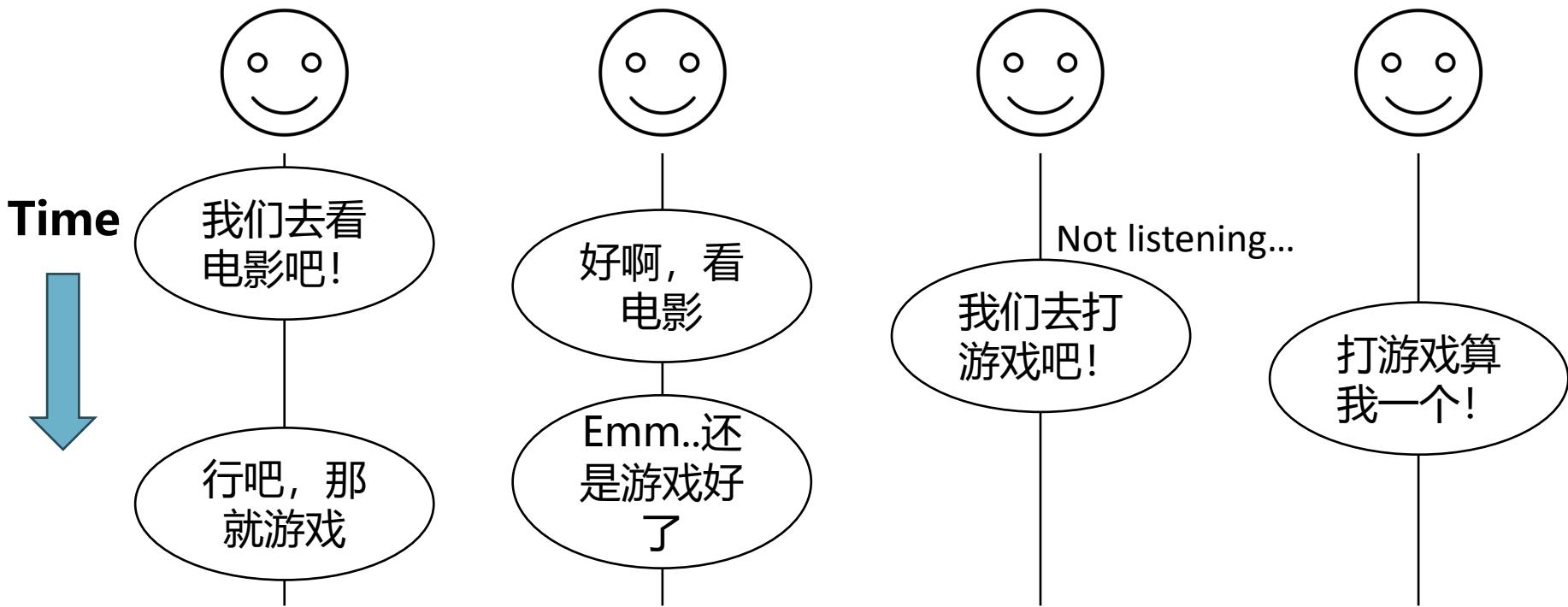


在分布式系统中，共识是指多个计算节点就某个**特定值或状态**达成一致的过程。这在分布式系统设计中非常重要，因为系统中的各个节点需要协调和同步，以**确保数据一致性和系统可靠性**

## □为什么需要共识？

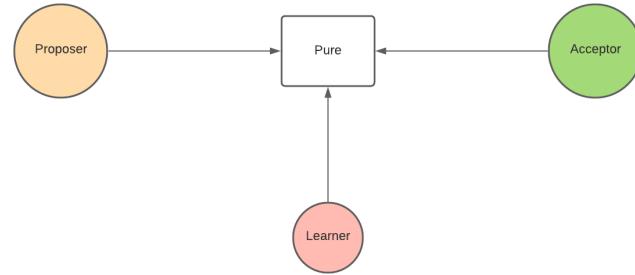
- **资源访问控制**: 确保在同一时间只有一个节点能够访问某个共享资源（互斥）
- **领导者选举**: 在分布式系统中选举出一个领导节点来协调其他节点的操作
- **事件顺序一致性**: 确保所有节点以相同的顺序处理事件，以便系统状态一致

# 达成共识是什么意思？



- 共识是就**一个结果**达成一致
- 各方希望就**任何结果**达成一致，而不仅仅就自己的提议达成一致
- 一旦多数人同意一项提案，那就是共识
- 达成的共识最终可以被所有人知道
- 通信信道可能出现故障，也就是说，**信息可能会丢失**

# Paxos 算法

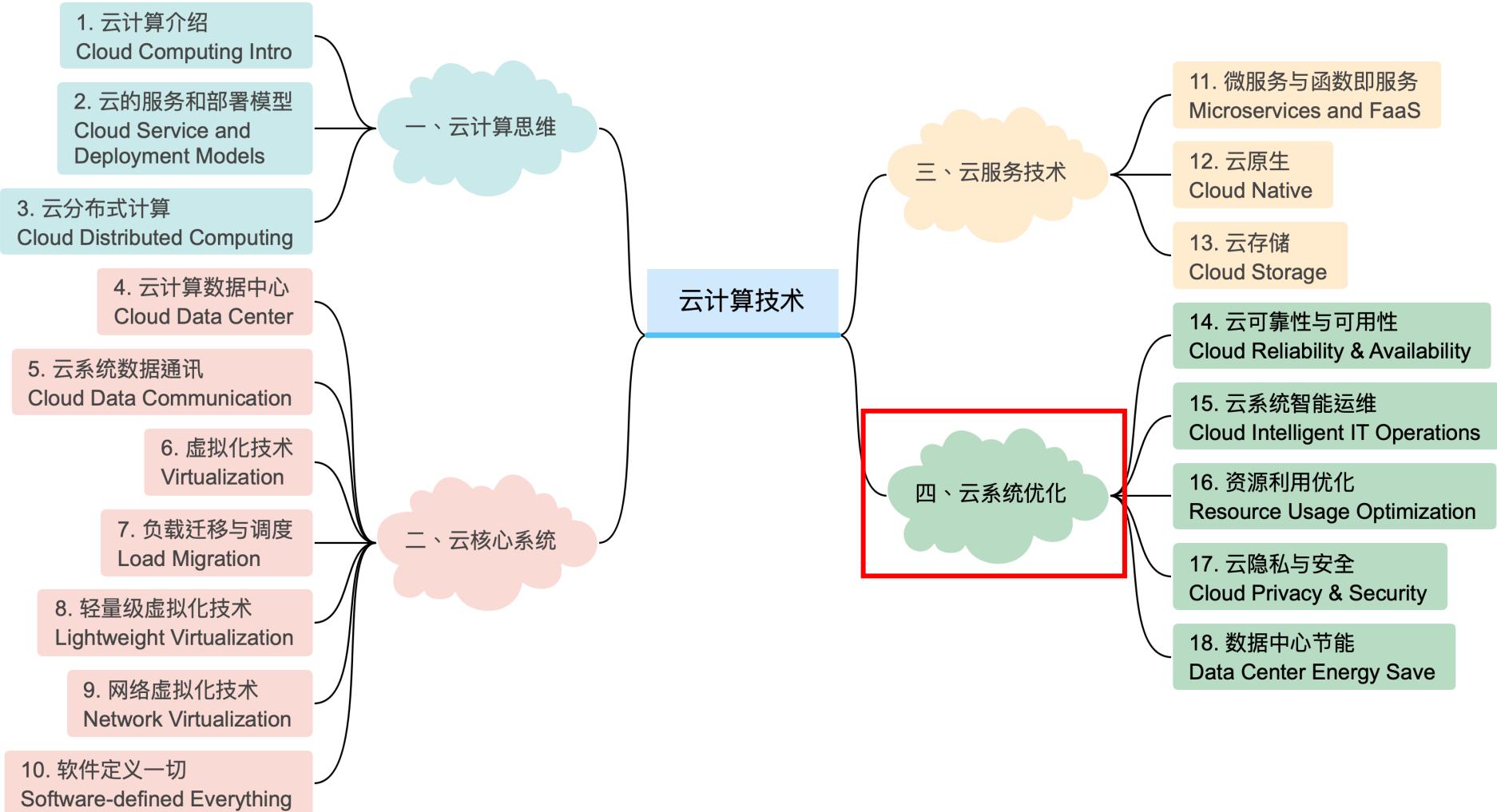


Paxos 算法用于在通过异步网络通信的一组分布式计算机之间实现共识。当运行 Paxos 的多数系统同意其中一个提议的值时，共识就达成了

Paxos 被广泛使用，并且在计算机科学中享有盛名，因为它是第一个被严格证明正确的共识算法

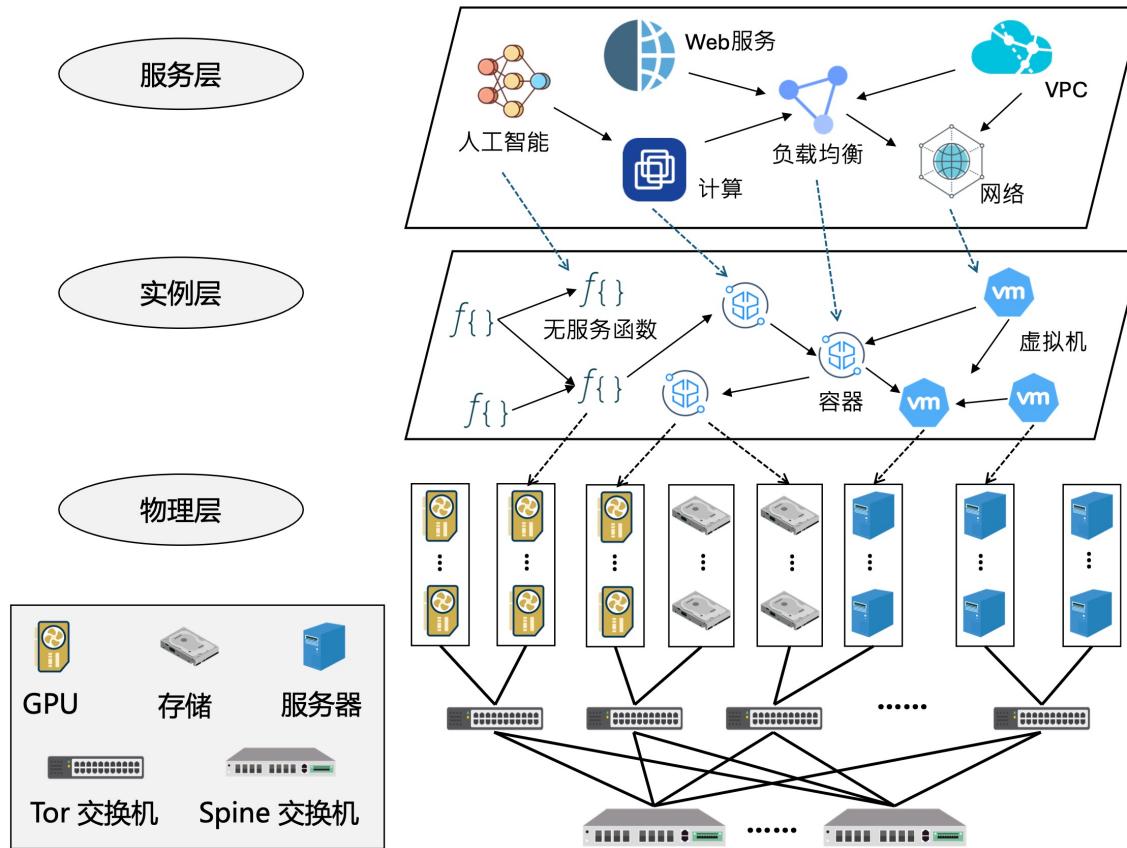
□ Paxos 提供可中止的共识 (abortable consensus)

- 当客户端向 Paxos 提出一个值时，若有一个竞争的并发提议获胜，那么提议的值可能会失败。客户端需再次向另一轮 Paxos 算法提出该值



# 云系统运维

云系统运维是指在云计算环境中**管理和维护系统、应用和基础设施的实践**，涵盖了从**部署、监控、优化到故障排除等各方面的工  
作**，以确保云资源的高效、安全、可靠运行



每一个组件的故  
障率都不为零

# 监控数据

口系统监控数据的典型类型



指标 (Metric)



日志 (Log)



追踪 (Trace)



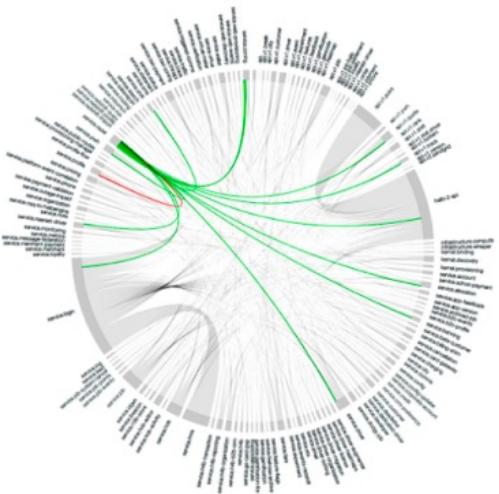
告警 (Alert)



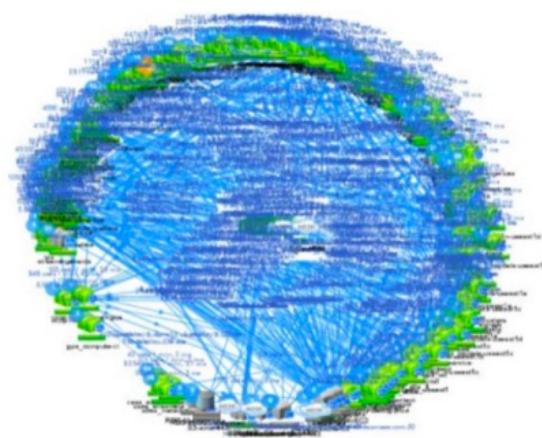
工单 (Ticket)

# 现代微服务系统

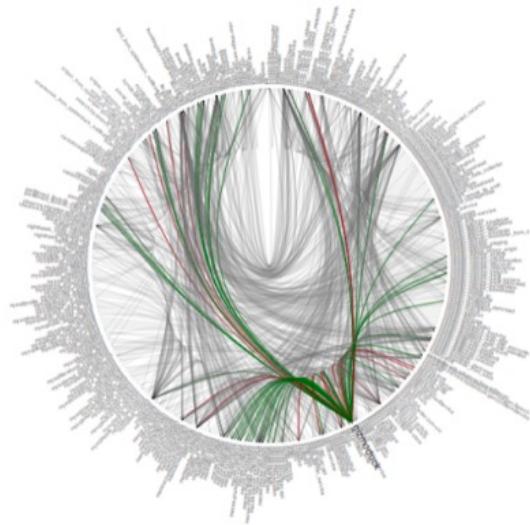
450+ microservices



500+ microservices



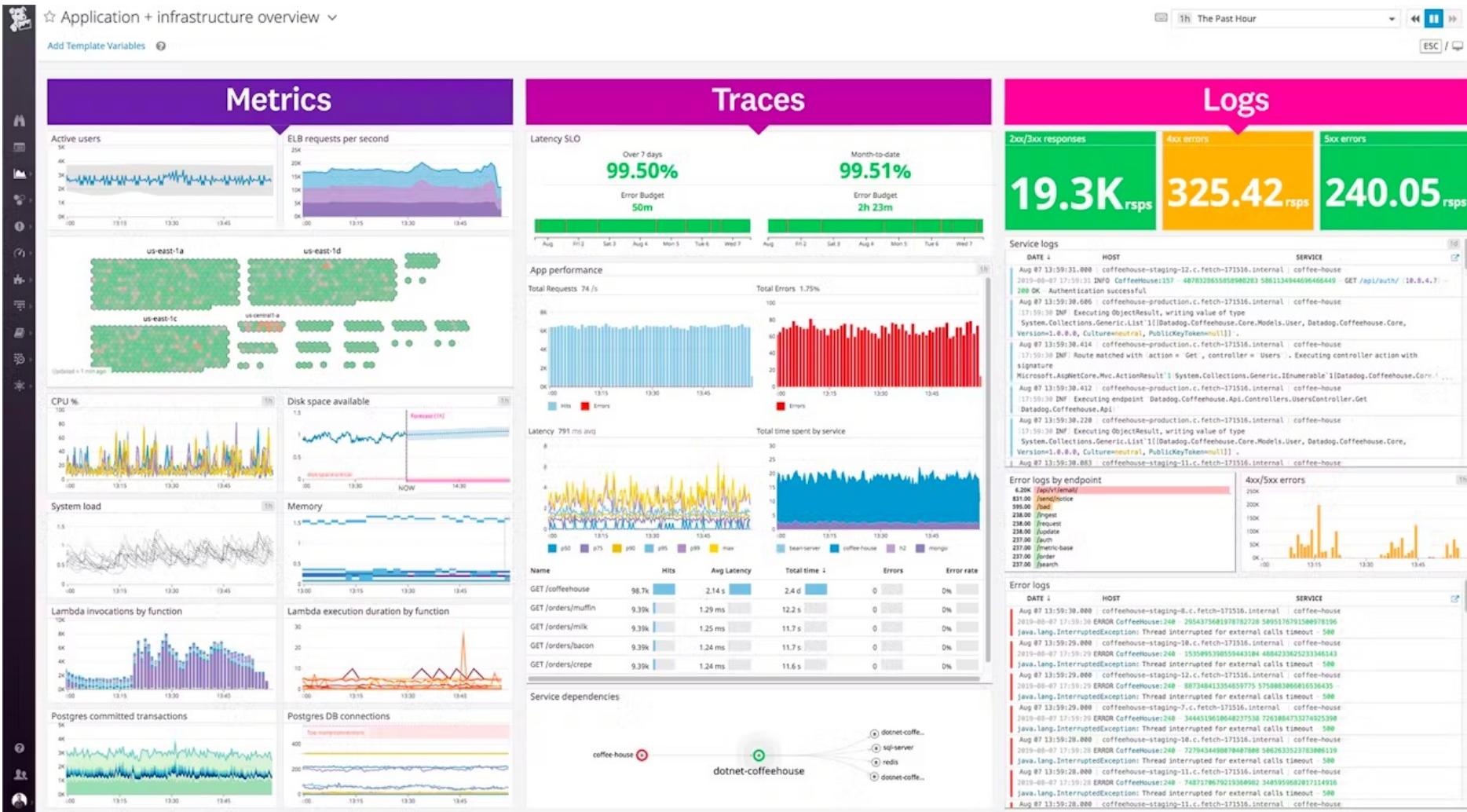
500+ microservices



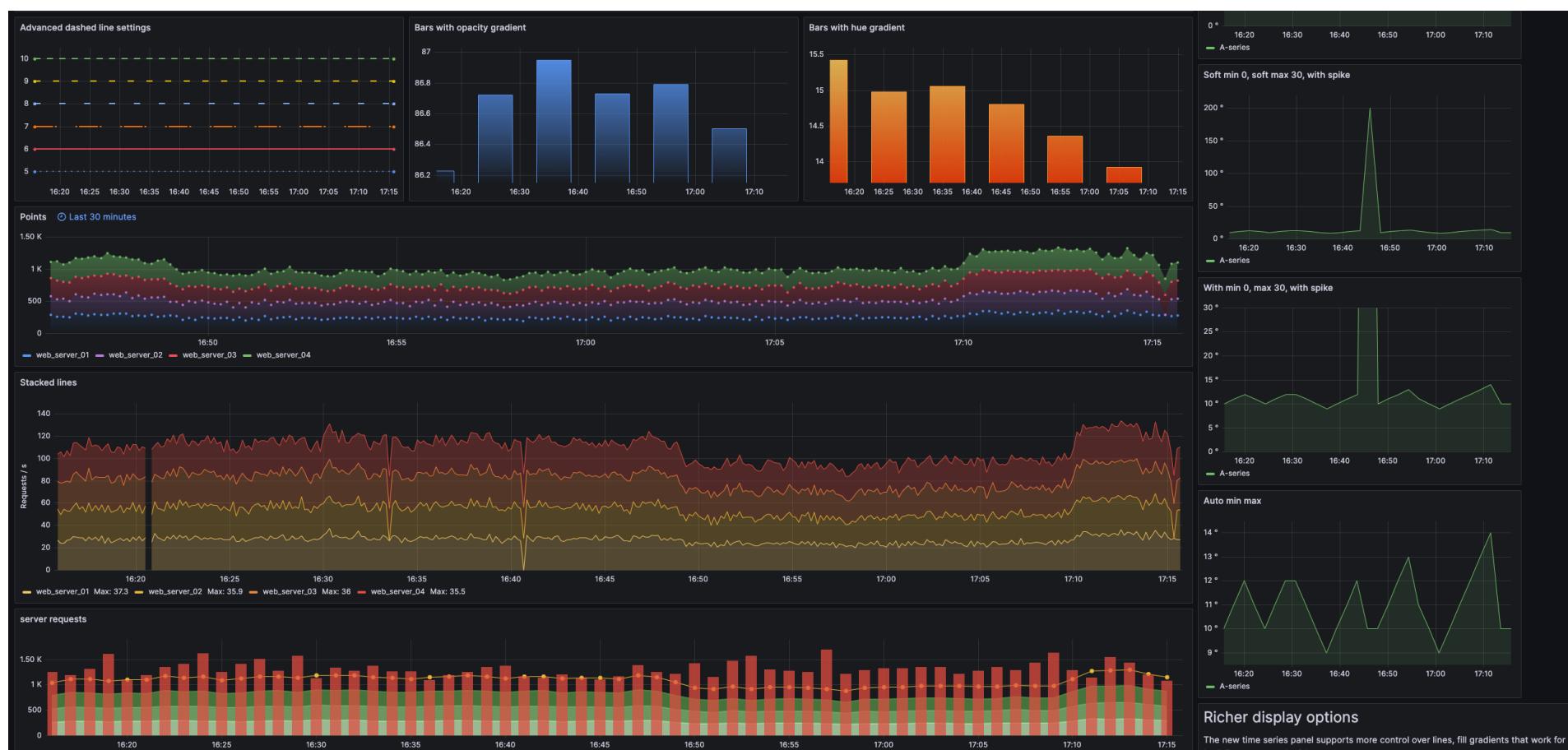
NETFLIX



# 维服务系统监控数据



# 百万级监控曲线

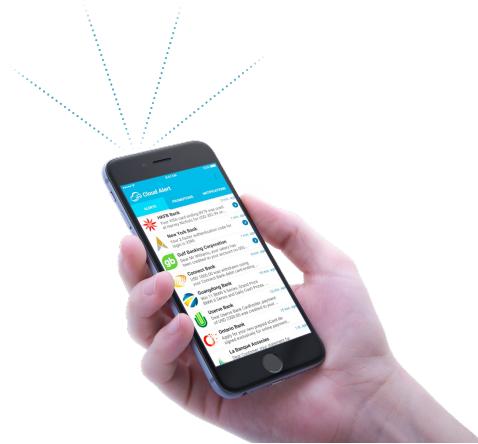


# 海量日志

## 口搜索 Exception

```
081110 020724 29 INFO dfs.FSNameSystem: BLOCK* NameSystem.delete: blk_2568309208894455676 is added to invalidSet of 10.251.31.160:50010
081110 022226 6281 INFO dfs.DataNode$DataXceiver: 10.250.19.16:50010 Served block blk_1078000656626961731 to /10.250.19.16
081110 023456 6415 WARN dfs.DataNode$DataXceiver: 10.251.67.225:50010:Got exception while serving blk_-6900989714336081087 to /10.251.25.237:
081110 024834 6371 INFO dfs.DataNode$DataXceiver: 10.251.126.227:50010 Served block blk_-8306714721294235181 to /10.251.126.227
081110 030331 6561 WARN dfs.DataNode$DataXceiver: 10.251.42.191:50010:Got exception while serving blk_-8023826090828946372 to /10.251.214.130:
081110 030942 6646 WARN dfs.DataNode$DataXceiver: 10.251.31.5:50010:Got exception while serving blk_-1367876730256254709 to /10.251.67.225:
081110 031019 6604 INFO dfs.DataNode$DataXceiver: 10.251.126.83:50010 Served block blk_-3860894070657427592 to /10.251.126.83
081110 032126 6555 WARN dfs.DataNode$DataXceiver: 10.251.26.8:50010:Got exception while serving blk_-7983508786213002472 to /10.251.38.197:
081110 035357 6606 INFO dfs.DataNode$DataXceiver: 10.251.71.97:50010 Served block blk_5454332143498402824 to /10.250.15.67
081110 040800 6739 INFO dfs.DataNode$DataXceiver: 10.250.6.214:50010 Served block blk_-3384560576963801177 to /10.250.6.214
081110 042826 6827 INFO dfs.DataNode$DataXceiver: 10.251.67.4:50010 Served block blk_-2901225370888235702 to /10.251.91.84
081110 045413 6957 INFO dfs.DataNode$DataXceiver: 10.251.215.50:50010 Served block blk_39636985674744747 to /10.251.107.196
081110 050300 6683 INFO dfs.DataNode$DataXceiver: 10.251.30.134:50010 Served block blk_2039230511363331616 to /10.251.65.203
081110 051323 6956 INFO dfs.DataNode$DataXceiver: 10.251.71.146:50010 Served block blk_8482211101480751895 to /10.251.71.146
081110 054642 7145 INFO dfs.DataNode$DataXceiver: 10.251.215.16:50010 Served block blk_-5919767995096301121 to /10.251.215.16
081110 060453 7193 INFO dfs.DataNode$DataXceiver: 10.251.199.225:50010 Served block blk_8457344665564381337 to /10.251.199.225
081110 060934 7211 INFO dfs.DataNode$DataXceiver: 10.251.66.102:50010 Served block blk_2986720270598512615 to /10.251.66.102
081110 065635 7324 INFO dfs.DataNode$DataXceiver: 10.251.90.64:50010 Served block blk_-5719934513583495857 to /10.251.199.245
081110 070326 7430 WARN dfs.DataNode$DataXceiver: 10.251.75.228:50010:Got exception while serving blk_-7680599654910200999 to /10.251.75.228:
081110 070334 7327 INFO dfs.DataNode$DataXceiver: 10.251.111.37:50010 Served block blk_-6050976999174805557 to /10.251.111.37
081110 070347 7574 INFO dfs.DataNode$DataXceiver: 10.250.10.6:50010 Served block blk_159841462205739245 to /10.251.90.239
081110 070614 7513 INFO dfs.DataNode$DataXceiver: 10.250.19.16:50010 Served block blk_-7873739190764609698 to /10.251.197.161
081110 070921 7744 INFO dfs.DataNode$DataXceiver: 10.251.39.144:50010 Served block blk_-8187008844253719581 to /10.251.91.32
081110 071154 7627 INFO dfs.DataNode$DataXceiver: 10.251.214.112:50010 Served block blk_4081177399275502985 to /10.251.110.68
081110 071426 7742 INFO dfs.DataNode$DataXceiver: 10.251.42.191:50010 Served block blk_3515154079719300106 to /10.251.42.191
081110 071657 7700 WARN dfs.DataNode$DataXceiver: 10.251.38.24:50010:Got exception while serving blk_-5547569777499890340 to /10.251.195.52:
081110 072121 7820 INFO dfs.DataNode$DataXceiver: 10.251.123.99:50010 Served block blk_5385121219895615240 to /10.251.71.146
081110 072124 7772 WARN dfs.DataNode$DataXceiver: 10.251.42.246:50010:Got exception while serving blk_-7658293778087733436 to /10.251.30.179:
081110 080546 7970 WARN dfs.DataNode$DataXceiver: 10.251.111.130:50010:Got exception while serving blk_3169060243663461885 to /10.251.214.32:
081110 080555 8227 INFO dfs.DataNode$DataXceiver: 10.250.11.194:50010 Served block blk_3087787567144441647 to /10.251.91.84
081110 080718 7850 INFO dfs.DataNode$DataXceiver: 10.250.10.100:50010 Served block blk_-3657665801189425193 to /10.250.10.100
081110 080724 8080 INFO dfs.DataNode$DataXceiver: 10.251.74.79:50010 Served block blk_-3457731723401426942 to /10.251.74.79
081110 080814 8123 INFO dfs.DataNode$DataXceiver: 10.251.42.84:50010 Served block blk_61050615579757068 to /10.251.42.84
081110 080847 8088 WARN dfs.DataNode$DataXceiver: 10.251.26.8:50010:Got exception while serving blk_-8522942048313632858 to /10.251.31.85:
081110 080922 7633 INFO dfs.DataNode$DataXceiver: 10.251.203.246:50010 Served block blk_365496398062338141 to /10.251.203.246
081110 080949 7994 INFO dfs.DataNode$DataXceiver: 10.250.19.227:50010 Served block blk_39798727517691718643 to /10.250.19.227
081110 081044 8125 WARN dfs.DataNode$DataXceiver: 10.251.123.1:50010:Got exception while serving blk_-272707591443354058 to /10.251.198.33:
081110 081054 8108 WARN dfs.DataNode$DataXceiver: 10.251.90.239:50010:Got exception while serving blk_-8679916835272129336 to /10.250.15.198:
081110 081337 8145 WARN dfs.DataNode$DataXceiver: 10.251.66.112:50010:Got exception while serving blk_610688431796192596 to /10.251.66.102:
081110 081515 8312 WARN dfs.DataNode$DataXceiver: 10.251.31.5:50010:Got exception while serving blk_633289272972950039 to /10.251.123.1:
081110 081643 8095 INFO dfs.DataNode$DataXceiver: 10.251.195.52:50010 Served block blk_6655622109568310643 to /10.251.195.52
081110 081643 8247 WARN dfs.DataNode$DataXceiver: 10.250.17.225:50010:Got exception while serving blk_-5935642747315643391 to /10.251.199.150:
081110 081741 8169 WARN dfs.DataNode$DataXceiver: 10.251.215.70:50010:Got exception while serving blk_-20269367189114433 to /10.251.30.179:
081110 082013 8326 INFO dfs.DataNode$DataXceiver: 10.251.43.115:50010 Served block blk_-7364557883931785608 to /10.251.43.115
081110 082043 8305 INFO dfs.DataNode$DataXceiver: 10.251.215.16:50010 Served block blk_2322432806134104317 to /10.251.215.16
081110 082444 8172 INFO dfs.DataNode$DataXceiver: 10.251.109.209:50010 Served block blk_4848669047361069041 to /10.251.26.177
081110 082702 8223 WARN dfs.DataNode$DataXceiver: 10.250.15.198:50010:Got exception while serving blk_-1851265222873801714 to /10.251.195.52:
081110 082706 8552 WARN dfs.DataNode$DataXceiver: 10.251.198.33:50010:Got exception while serving blk_-8495670552887053546 to /10.250.10.223:
081110 082737 8341 WARN dfs.DataNode$DataXceiver: 10.250.19.227:50010:Got exception while serving blk_-7372087176866857012 to /10.251.110.68:
081110 082954 8543 WARN dfs.DataNode$DataXceiver: 10.251.70.112:50010:Got exception while serving blk_4357276972386184626 to /10.251.74.79:
081110 083045 8495 WARN dfs.DataNode$DataXceiver: 10.251.215.16:50010:Got exception while serving blk_-4590972095204776122 to /10.251.30.6:
081110 083121 8197 INFO dfs.DataNode$DataXceiver: 10.251.106.214:50010 Served block blk_-827787362772158374 to /10.251.112.79
081110 083231 8530 INFO dfs.DataNode$DataXceiver: 10.250.10.176:50010 Served block blk_3797494971676497497 to /10.251.29.239
081110 083328 8416 INFO dfs.DataNode$DataXceiver: 10.251.126.22:50010 Served block blk_80558786054060864 to /10.251.126.22
081110 083453 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_3141363517520802396
081110 085933 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_-4117999745005013424
081110 091216 7593 WARN dfs.DataNode$DataXceiver: 10.251.107.50:50010:Got exception while serving blk_4524198807982839635 to /10.251.122.79:
081110 091550 8683 WARN dfs.DataNode$DataXceiver: 10.251.111.209:50010:Got exception while serving blk_7505828172725463922 to /10.251.111.209:
081110 091657 8720 WARN dfs.DataNode$DataXceiver: 10.251.70.211:50010:Got exception while serving blk_424255210146453297 to /10.251.203.179:
081110 091800 8380 INFO dfs.DataNode$DataXceiver: 10.250.14.224:50010 Served block blk_666713934549639791 to /10.250.14.224
081110 092131 8815 INFO dfs.DataNode$DataXceiver: 10.251.30.179:50010 Served block blk_-2975629975082443857 to /10.251.30.179
081110 092151 8601 WARN dfs.DataNode$DataXceiver: 10.251.126.22:50010:Got exception while serving blk_1686195200514944346 to /10.250.6.223:
081110 092459 8650 INFO dfs.DataNode$DataXceiver: 10.250.9.207:50010 Served block blk_435546062720483068 to /10.250.9.207
081110 092815 8872 INFO dfs.DataNode$DataXceiver: 10.250.6.191:50010 Served block blk_5952254363678329024 to /10.250.6.191
081110 093020 8827 WARN dfs.DataNode$DataXceiver: 10.251.70.211:50010:Got exception while serving blk_-66793171485085225 to /10.251.203.246:
081110 093643 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_9188832735514090334
081110 093831 8571 INFO dfs.DataNode$DataXceiver: 10.251.106.10:50010 Served block blk_2273334621241206674 to /10.251.106.10
081110 094019 8808 WARN dfs.DataNode$DataXceiver: 10.251.125.193:50010:Got exception while serving blk_3790492230047189408 to /10.251.199.159:
081110 094657 7835 INFO dfs.DataNode$DataXceiver: 10.251.107.50:50010 Served block blk_-228572986739318683 to /10.251.70.5
```

# 告警运维



Wake up at 3am!

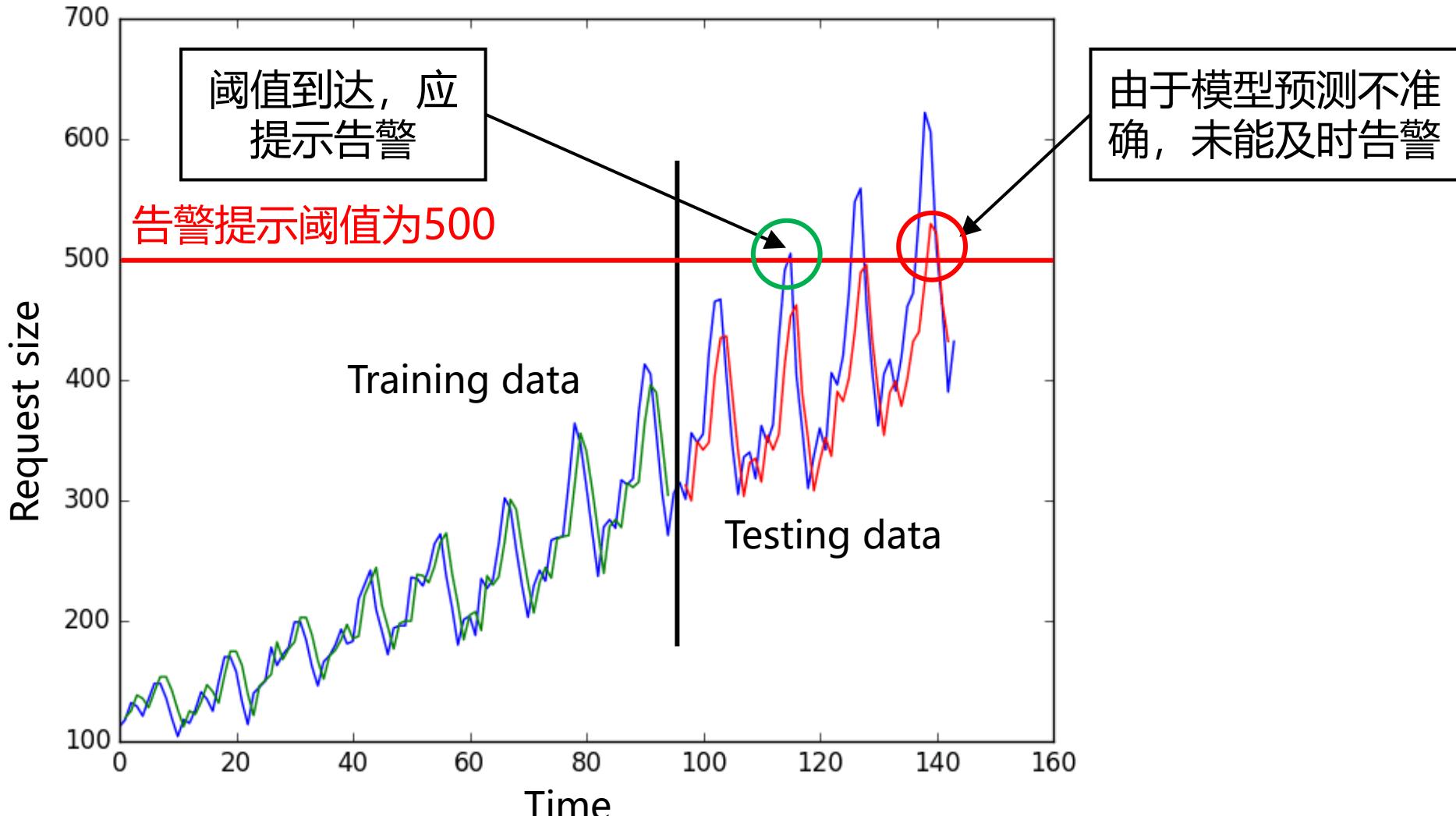
# 云系统智能运维

AIOps (Artificial intelligence for IT Operations): 引入机器学习，以数据驱动的方法进行云系统运维

## 口典型智能运维任务

- ✓ 异常检测 (Anomaly detection)
- ✓ 故障分析 (Failure analysis)
- ✓ 故障预测 (Failure prediction)
- ✓ 根因定位 (Root cause localization)
- ✓ 软件测试 (Software testing)
- ✓ 自动修复 (Auto-healing)
- ✓ ...

# 基于 LSTM 的指标异常检测



# 日志挖掘



判断系统的某次任务执行是否发生异常

Traditional  
machine learning

Deep learning  
models

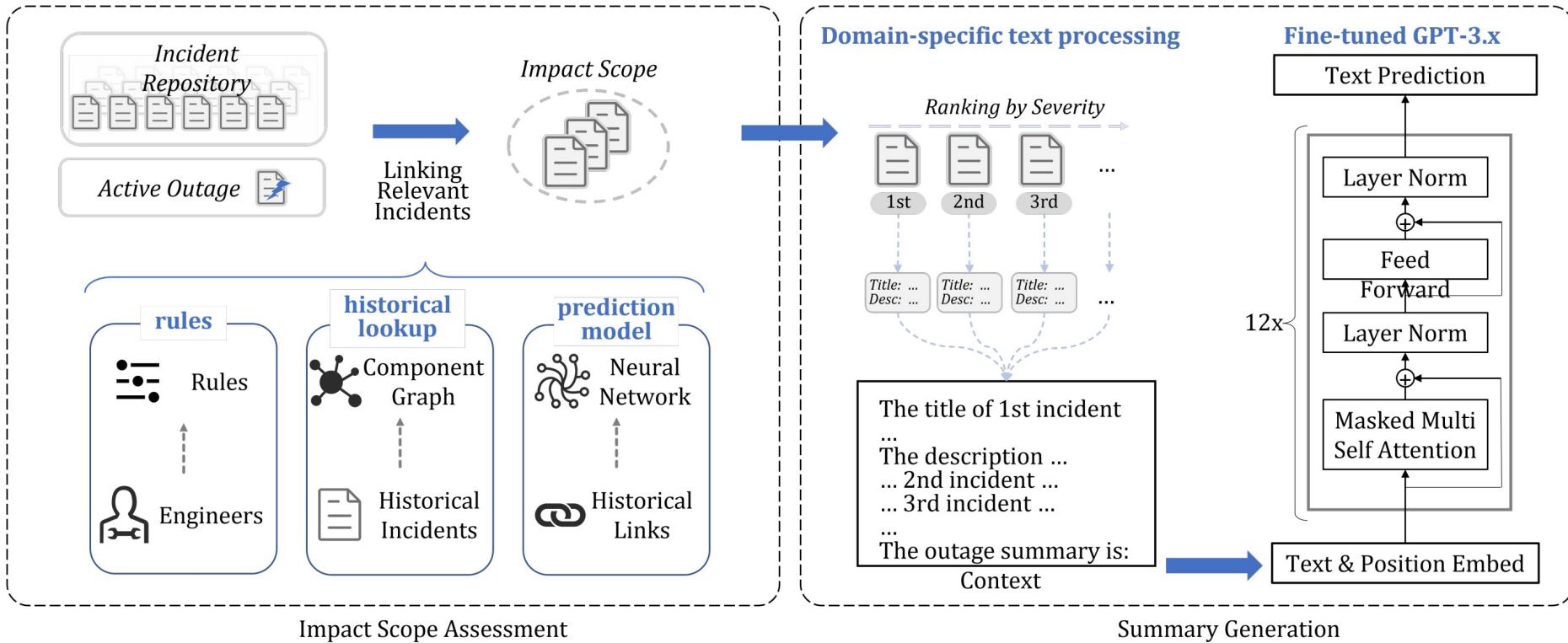
# 知识图谱方向



关系

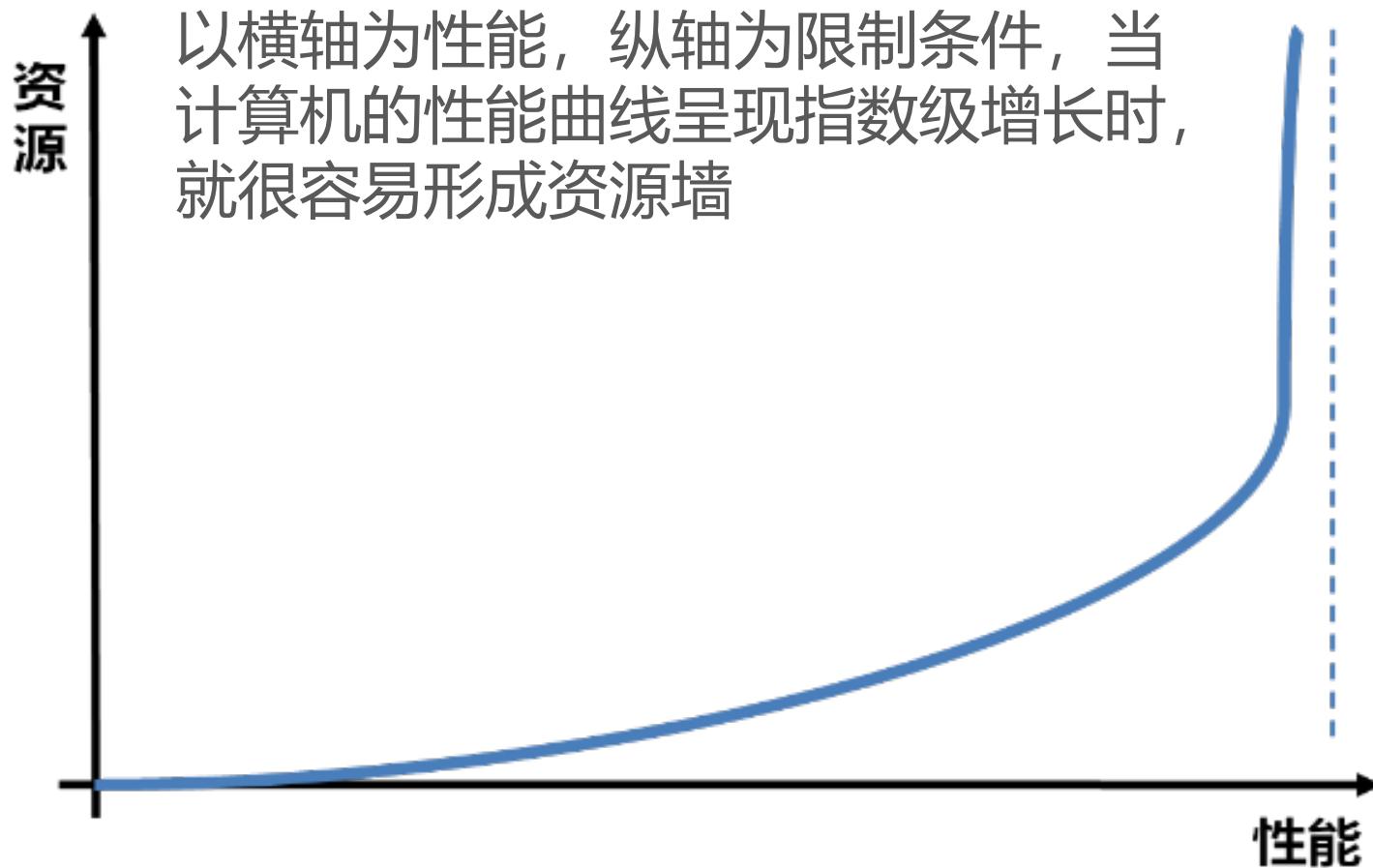
- has symptom
- has root cause
- has child

# 基于大模型的工单总结



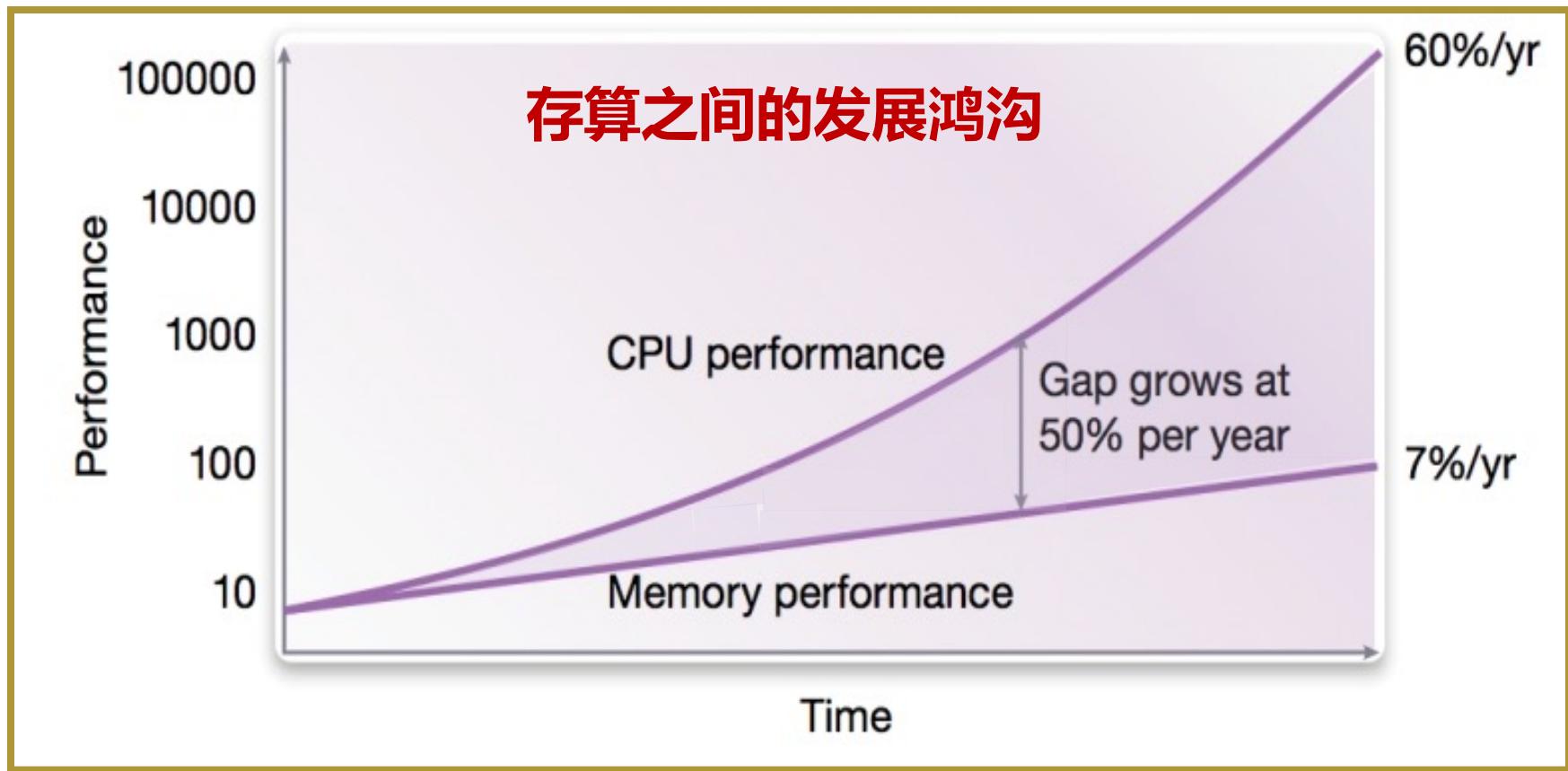
# 访存/功耗墙

“墙”：限制计算机性能的无形资源瓶颈



# 访存/功耗墙

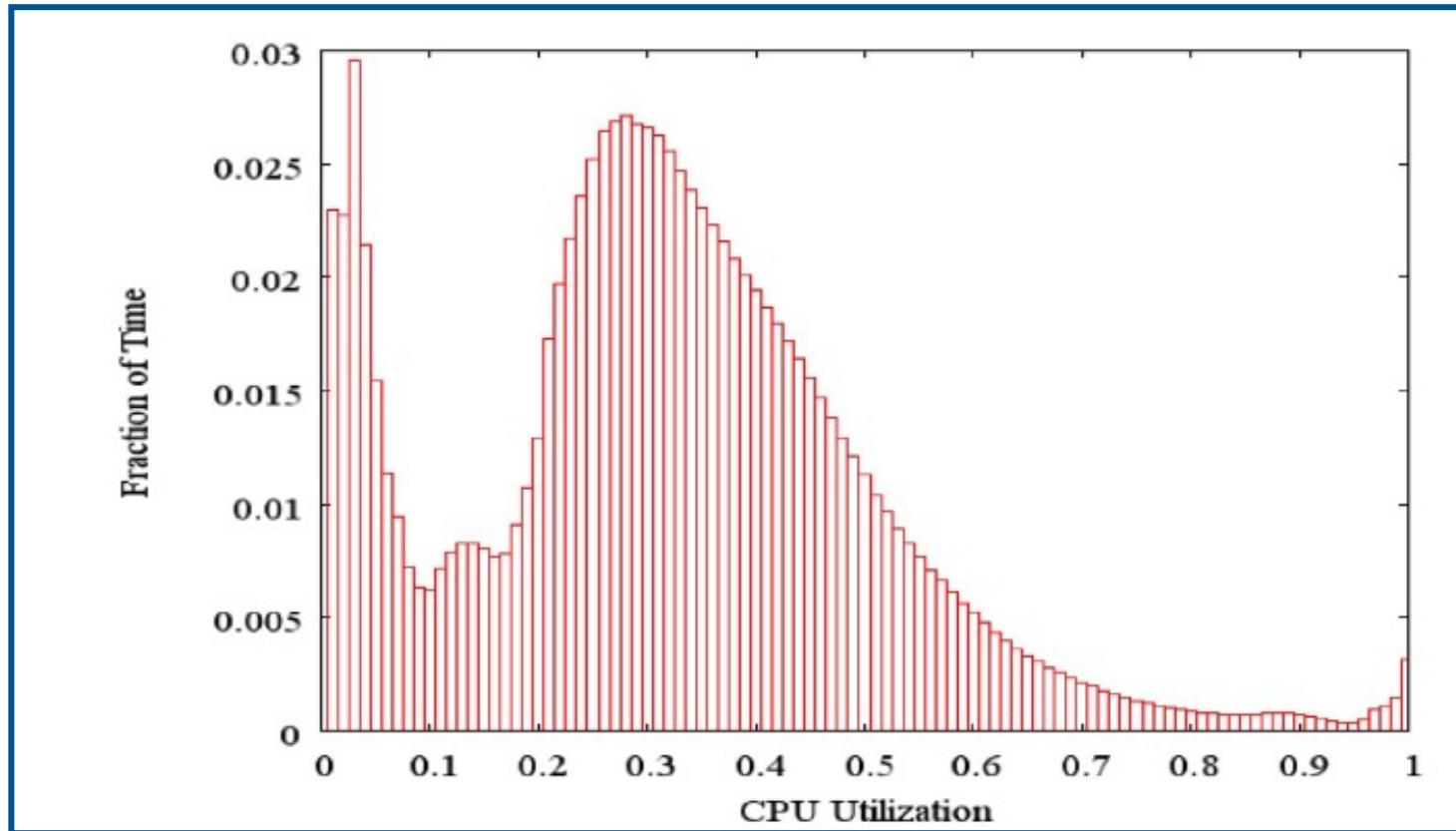
造成“墙”的原因往往在于技术发展速度不均衡



# 利用率问题

## 数据中心资源闲置问题

Google 5000台服务器半年的服务器负载分布



低使用率是许多数据中心面临的一个关键问题

# 资源利用优化策略 – 弹性伸缩

资源弹性伸缩 (Autoscaling) 是一种云  
资源管理策略，它根据实际工作负载需求  
动态调整计算资源的数量

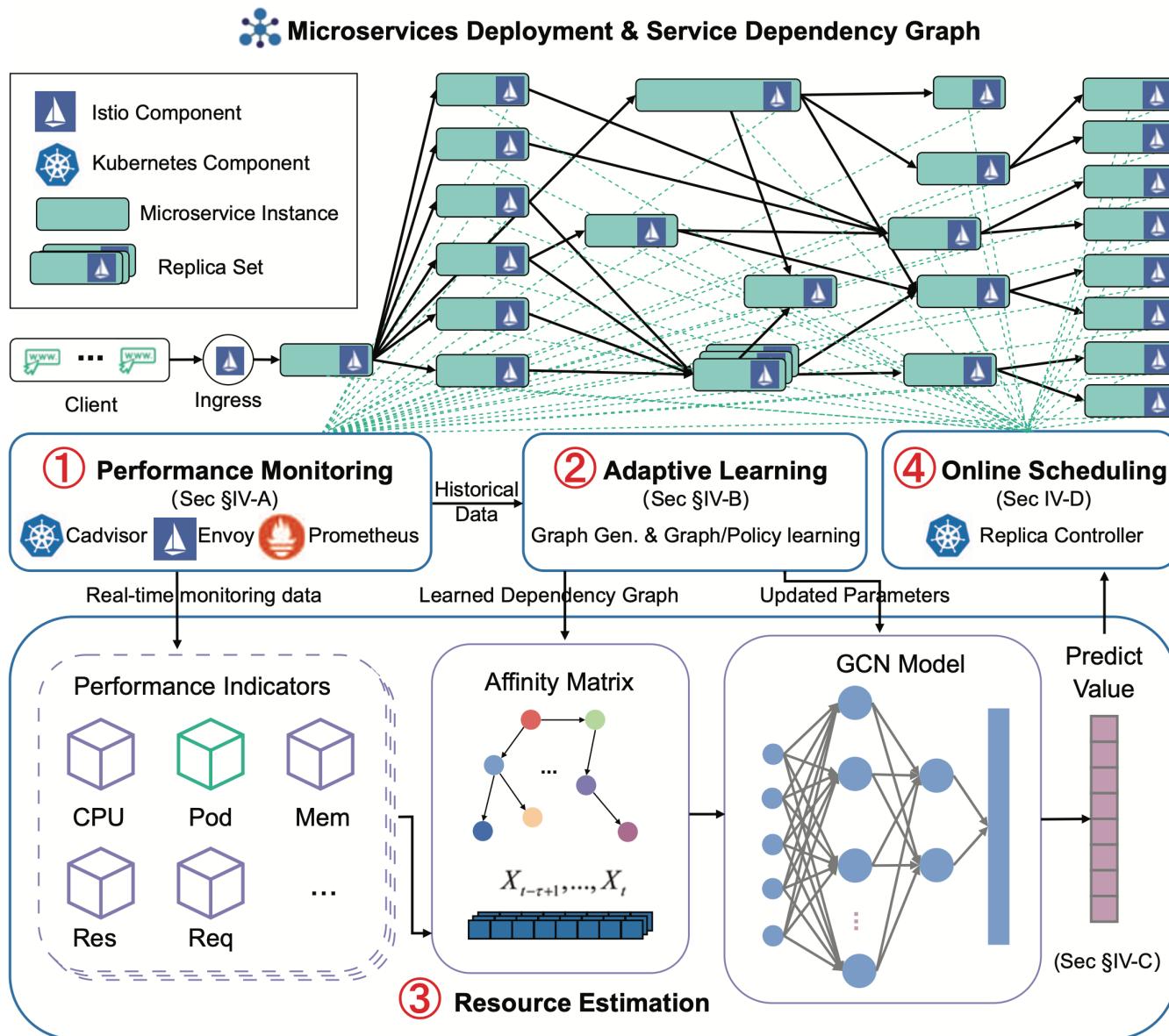


# 资源利用优化策略 – 负载均衡

负载均衡 (Load Balancing) 是一种策略，  
通过在多个计算资源之间分配工作负载，  
实现性能优化和可靠性提升

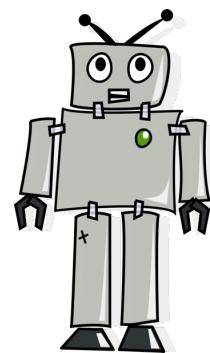
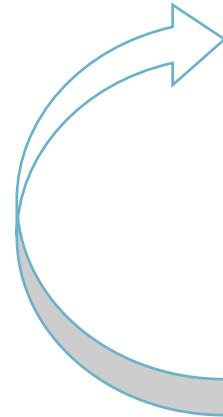


# 微服务负载预测系统概览



# 强化学习流程

**Observation  
Function input**



**Find a policy  $f$  maximizing  
the total reward:  
 $Action = f(Observation)$**

**Action  
Function output**



**Environment (Space invaders)**

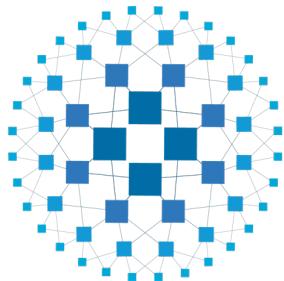
# 负载均衡的实现方式

## 口硬件负载均衡器

- 硬件负载均衡器是专用设备，提供高性能和高可靠的负载均衡服务，但成本较高，灵活性较差。常见的厂商有 F5、Citrix 等

## 口软件负载均衡器

- 软件负载均衡器运行在通用服务器上，灵活性高，易于扩展和管理。常见的开源软件有 HAProxy、Nginx、Apache Traffic Server 等



HAProxy





中山大學 软件工程学院  
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬  
软件工程学院  
[chenzhb36@mail.sysu.edu.cn](mailto:chenzhb36@mail.sysu.edu.cn)