



# 高性能计算与云计算

## 第十一讲 云计算平台

胡金龙，董守斌

华南理工大学计算机学院  
广东省计算机网络重点实验室

Communication & Computer Network Laboratory (CCNL)

# 课程内容

---

- 云计算概念
- 云存储
- 典型云平台

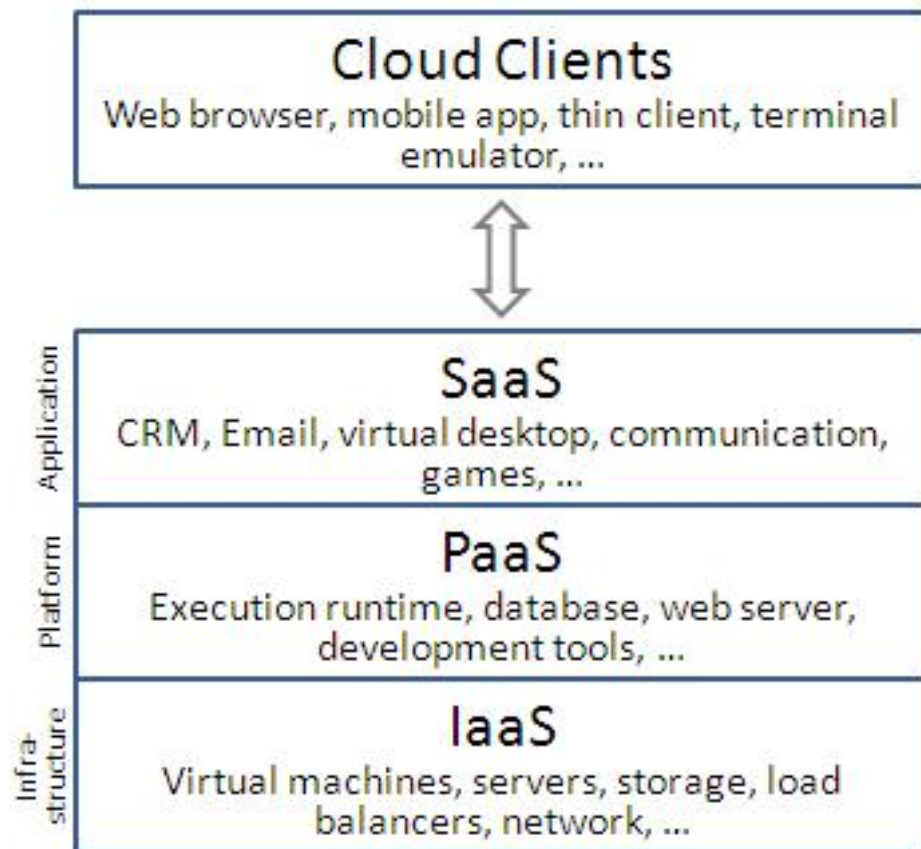
# 云计算的定义：5个特征

- 没有一致认可的定义
- 如，NIST定义：
  - 云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池（资源包括网络，服务器，存储，应用软件，服务），这些资源能够被快速提供，只需投入很少的管理工作，或服务供应商进行很少的交互
- 五个特征：
  - 按需应变自助服务
  - 随时随地用任何网络设备访问
  - 多人共享资源池
  - 快速重新部署灵活度
  - 可被监控与量测的服务

# 云计算的定义：3种服务模式

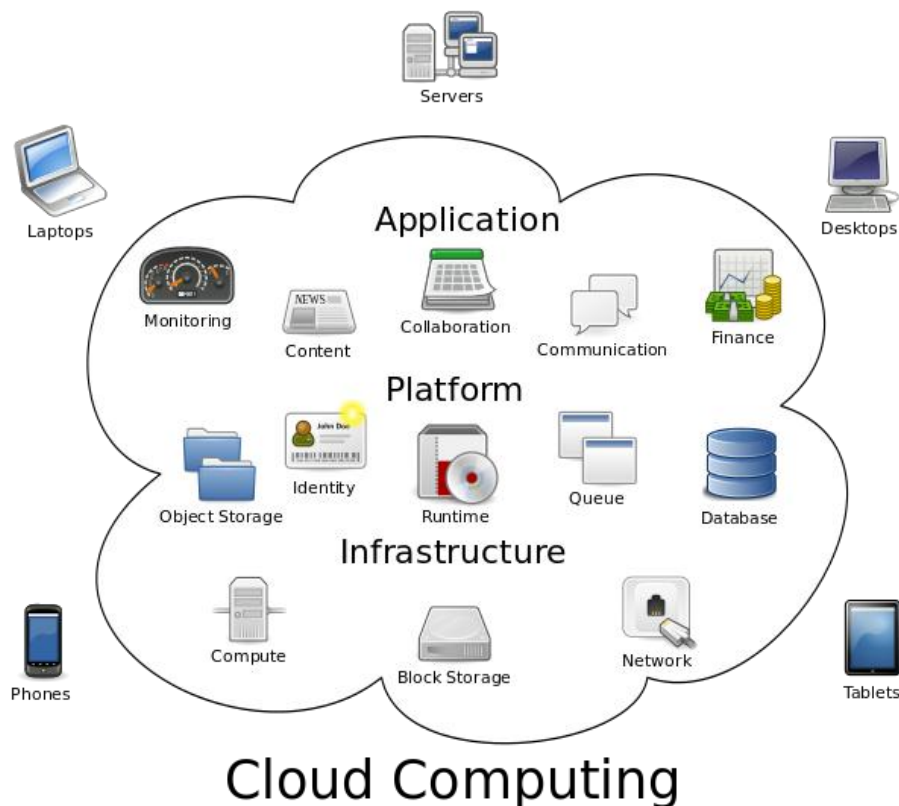
- 云计算的三种服务模式：

- 软件即服务（SaaS）
- 平台即服务（PaaS）
- 基础架构即服务（IaaS）



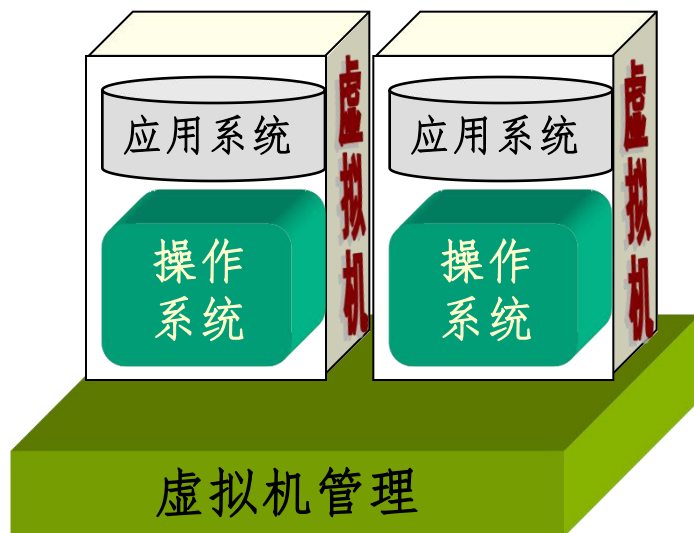
# 云计算的基础：虚拟化

- 云计算描述了一种基于互联网的新的IT服务增加、使用和交付模式，通常涉及通过互联网来提供动态易扩展而且经常是虚拟化的资源。



# 什么是虚拟化（Virtualization）？

- 在虚拟化技术中，可同时运行多个操作系统，而且每一个操作系统中都有多个程序在运行，每一个操作系统都运行在一个虚拟的CPU或者是虚拟主机上



# 虚拟化技术的历史

虚拟化技术将物理资源转化为便于切分的资源池，符合云计算的基本条件；

虚拟化给资源以动态调配的能力，符合云计算按需分配的要求；



**Amazon**采用虚拟化技术提供云计算平台，取得了商业上的成功，虚拟化技术成为云计算的基石；

**2006**

**1960's**

IBM推出虚拟化技术，提高了昂贵的大型机的利用率

**1999**

VMware公司解决了X86虚拟化问题，推出了X86平台的虚拟机软件，使虚拟化技术开始走向普通用户

**2003**

开源虚拟化技术Xen推出，使虚拟化技术的研究和应用更加普及

**2005**

Intel和AMD推出支持虚拟化技术的处理器和芯片组，实现了硬件辅助虚拟化技术

# 服务器虚拟化

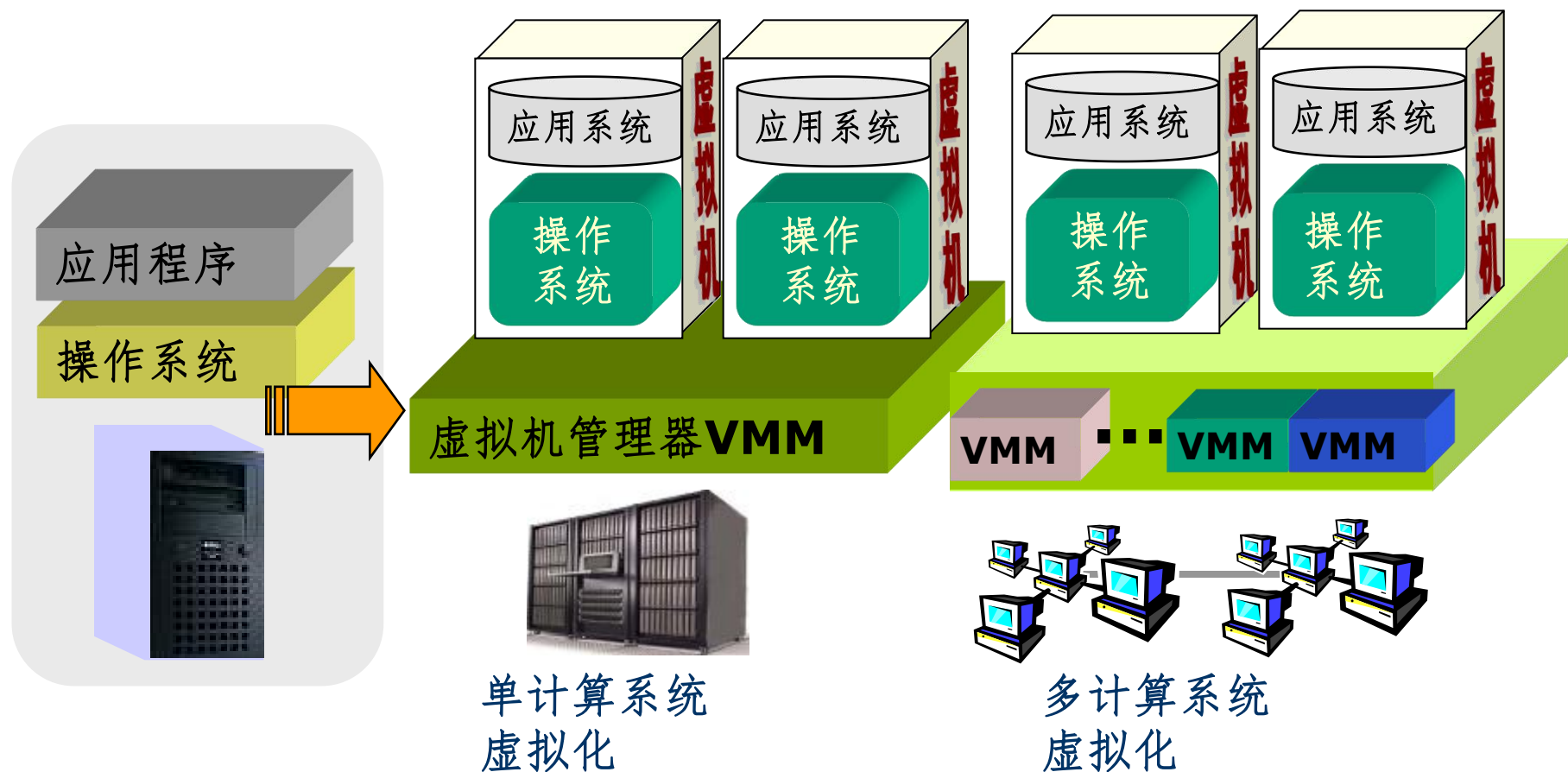
- 将服务器物理资源抽象成逻辑资源，让一台服务器变成几台甚至上百台相互隔离的虚拟服务器，不再受限于物理上的界限
- 让CPU、内存、磁盘、I/O等硬件变成可以动态管理的“资源池”，
- 可提高资源的利用率，简化系统管理，实现服务器整合，让IT对业务的变化更具适应力



# 虚拟化计算系统体系结构

传统计算系统  
计算模式

虚拟化计算系统  
计算模式



# 服务器虚拟化的类型

- **拆分 (Partition)**

- 某台计算机性能较高，而工作负荷小，资源没有得到充分利用。可以将这台计算机拆分为逻辑上的多台计算机，同时供多个用户使用。这样可以使此服务器的硬件资源得到充分的利用。
- 目的：提高资源利用率

- **整合 (Consolidation)**

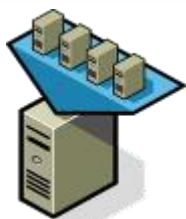
- 如有大量性能一般的计算机，可应用虚拟整合技术，将大量性能一般的计算机整合为一台计算机，以满足客户对整体性能的要求
- 目的：通过整合，获得高性能，满足特定数据计算要求。

- **迁移 (Migration)**

- 将一台逻辑服务器中的闲置的一部分资源动态的加入到另一台逻辑服务器中，提高另一方的性能。
- 目的：实现资源共享，实现跨系统平台应用等。

# 虚拟技术：四大特性

## 分区



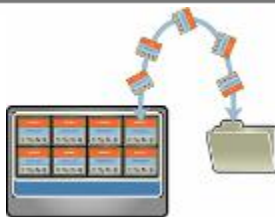
在单一物理服务器上同时运行  
多个虚拟机

## 隔离



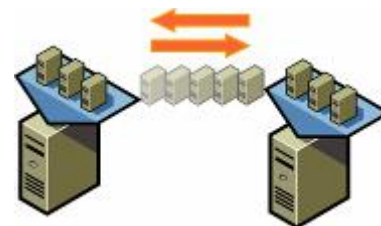
在同一服务器上的虚拟机之间  
相互隔离

## 封装



整个虚拟机都保存在文件中，而且  
可以通过移动和复制这些文件的方  
式来移动和复制该虚拟机

## 相对于硬件独立



无需修改即可在任何服务器上  
运行虚拟机

# 虚拟化技术的优势

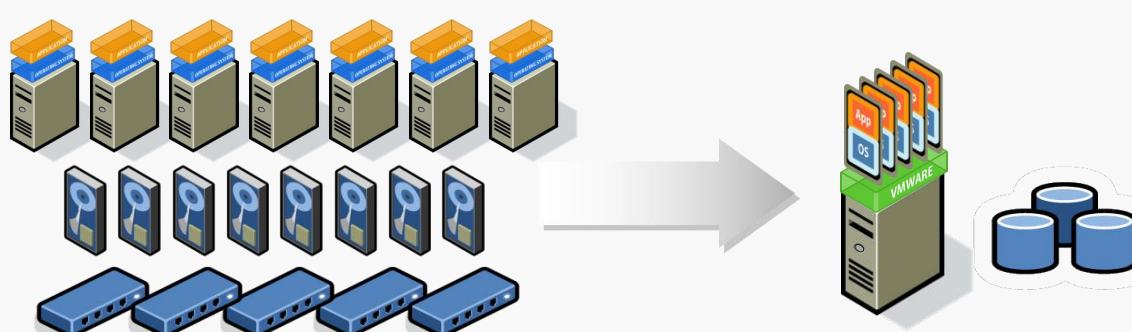
---

- 更高的资源利用率
- 降低管理成本
- 提高使用的灵活性
- 提高安全性
- 更高的可用性
- 更高的可扩展性
- 互操作性
- 改进资源供应

# 应用案例：数据中心整合

## 例子：某北美公共设施公司

	整合之前	整合之后
服务器	1,000 台	80 台
存储	270 TB DAS	140 TB SAN 和 NAS
网络	3,000 个电缆/端口	300 个电缆/端口
设备	200 个服务器机架 400 个电源开关	10 个服务器机架 20 个电源开关



## 虚拟化对其产生的影响

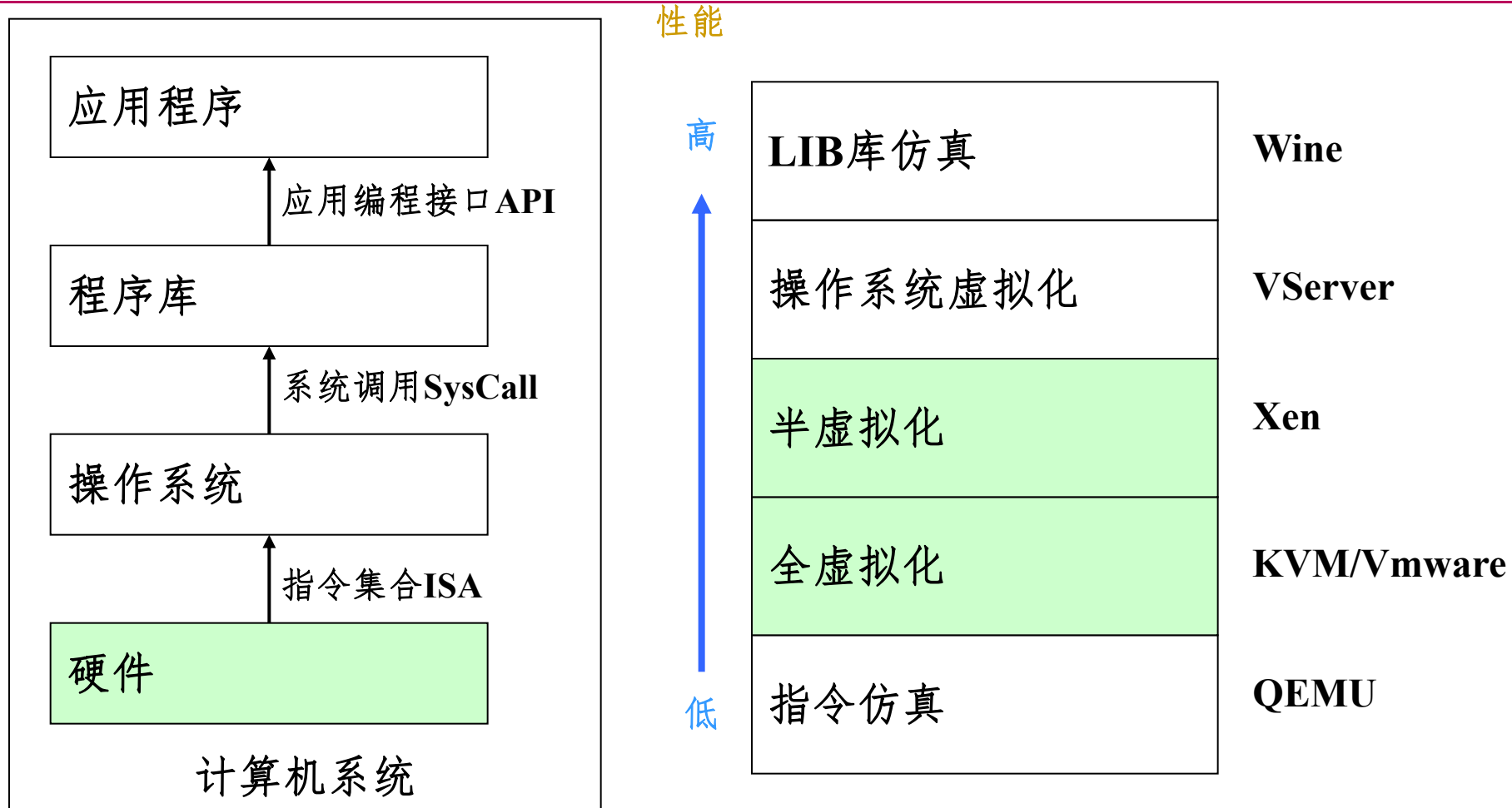
### 硬件成本节省

- 数据中心空间、电力和制冷成本节省 70-80%
- 2 年节省 800 万美元

### 运营效率

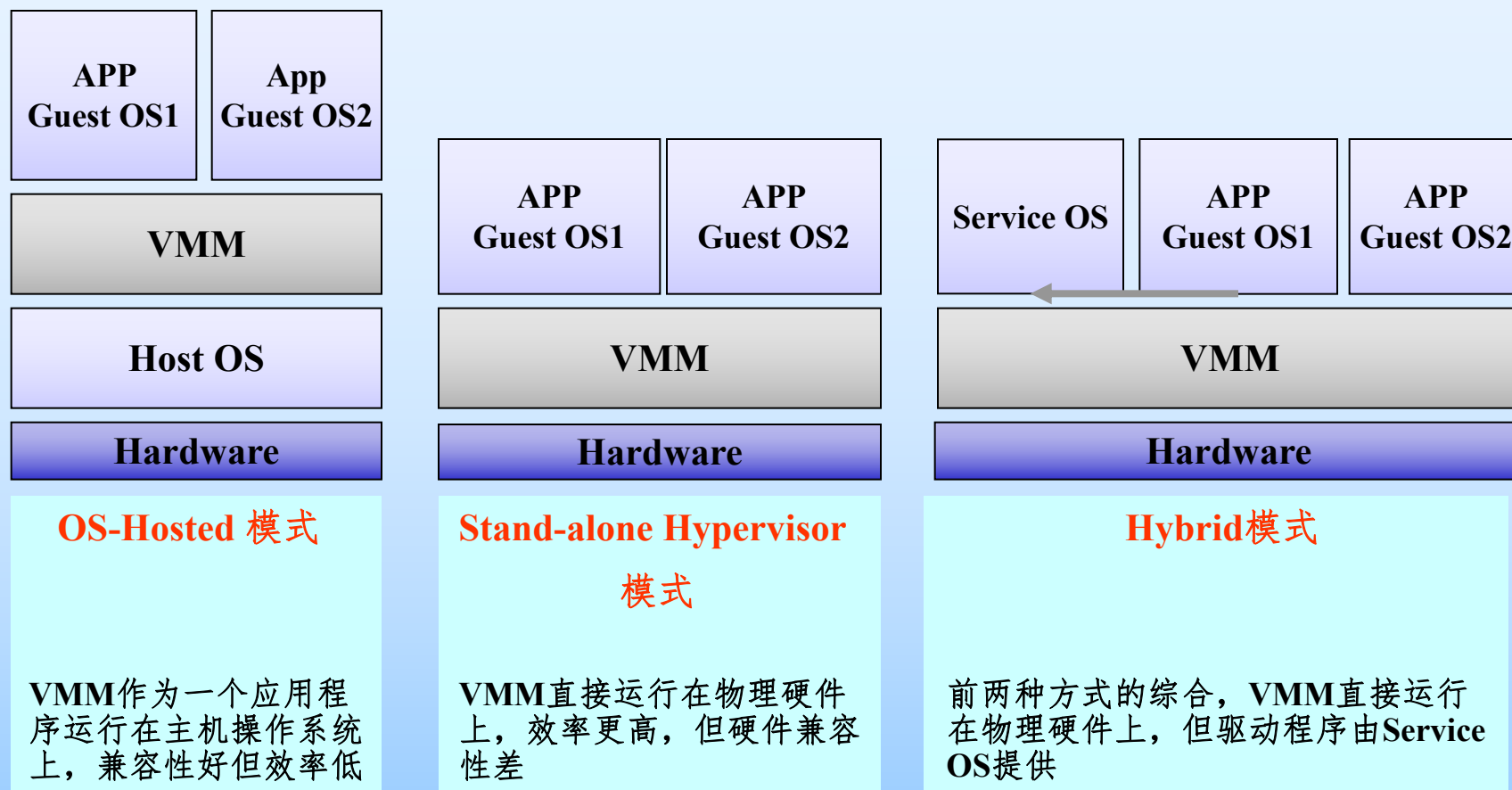
- 服务器重建和应用程序载入时间从 20-40 小时缩短到 15-30 分钟
- 每年节省 10,000 工时

# 如何实现虚拟化？

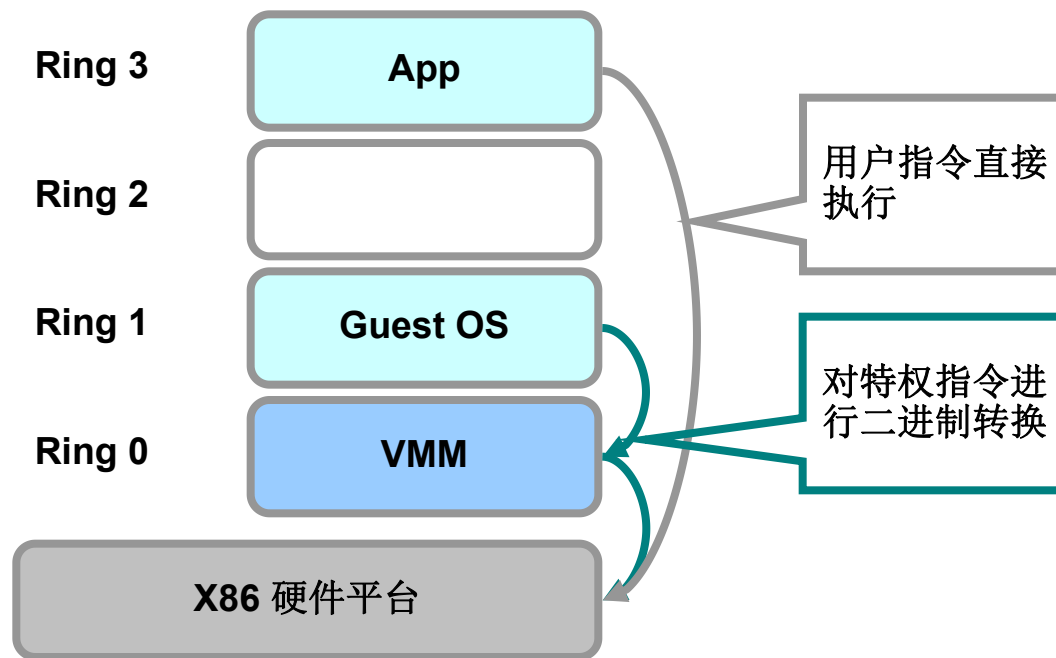


# 虚拟化关键组件VMM (Virtual Machine Manager)

VMM又称为Hypervisor，负责为虚拟机统一分配CPU、内存和外设，调度虚拟资源；



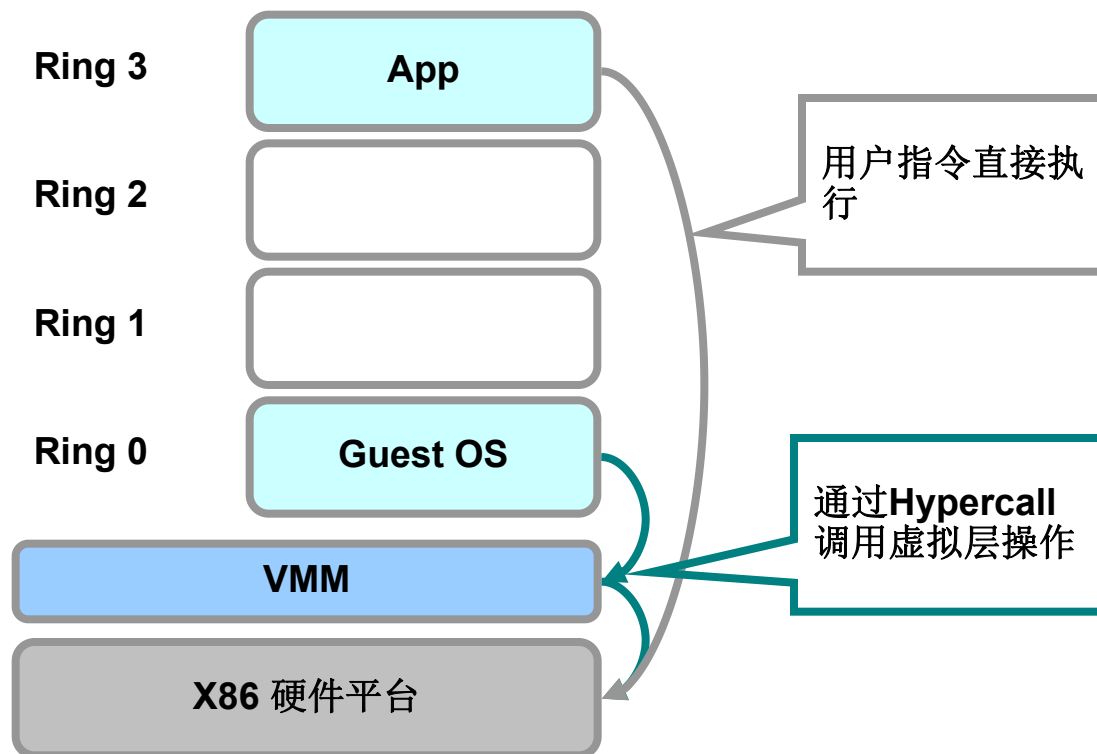
# 全虚拟化技术



- 客户操作系统运行在Ring 1级，VMM运行在Ring 0级，VMM提供给操作系统各种虚拟资源（虚拟BIOS、虚拟设备和虚拟内存管理等）。对于不能虚拟化的特权指令，通过二进制转换方式转换为同等效果的指令序列运行，而用户级指令可直接运行。
- 客户操作系统与底层硬件资源完全隔离，操作系统不感知运行在虚拟机上，也不需要修改操作系统，虚拟机具有较好的隔离性和安全性。

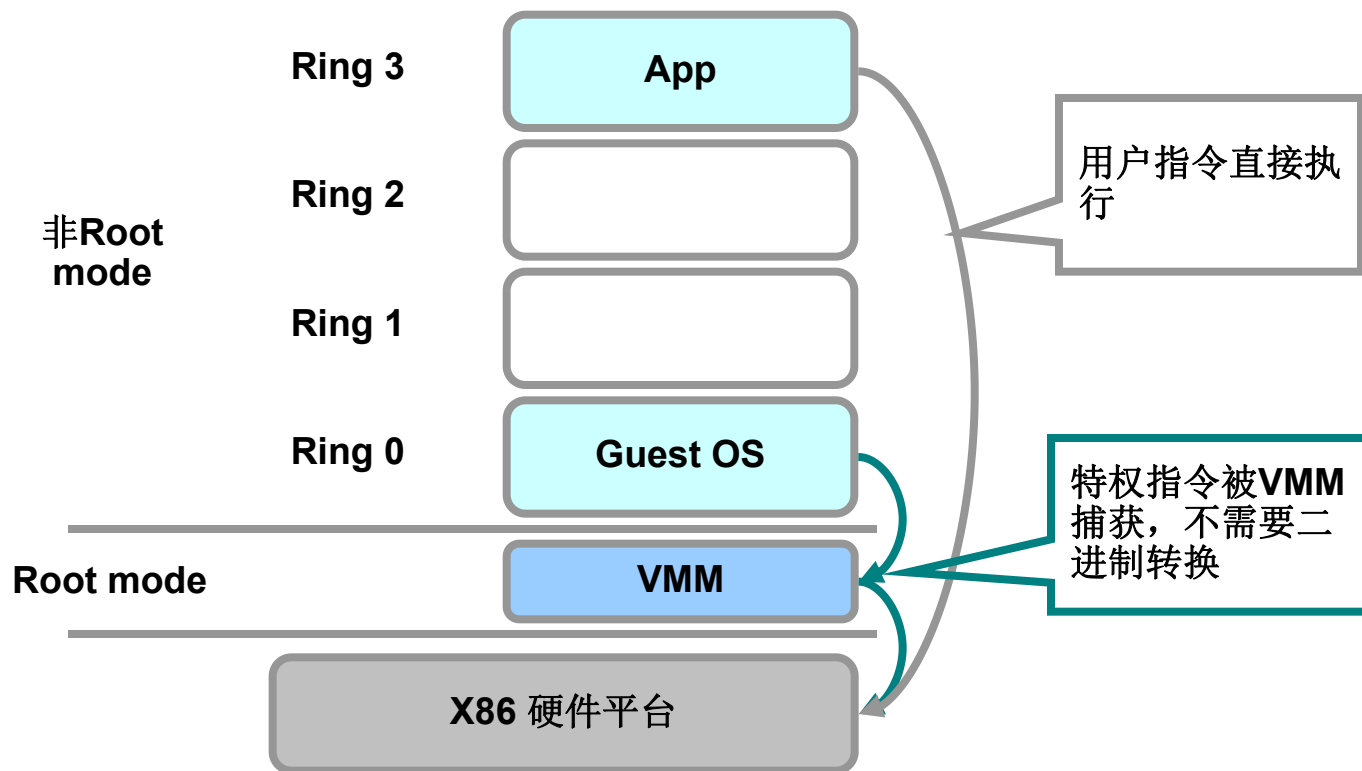


# 半虚拟化技术



- 这种方式需要修改操作系统内核，将不能虚拟化的指令替换为hypercall，hypercall直接与虚拟层通信，虚拟层提供内核操作的关键接口，如内存管理、中断处理和时间管理等。
- 这样显著减少了虚拟化开销，性能较高，但是由于需要修改操作系统内核，对于非开放的操作系统，则无法支持。

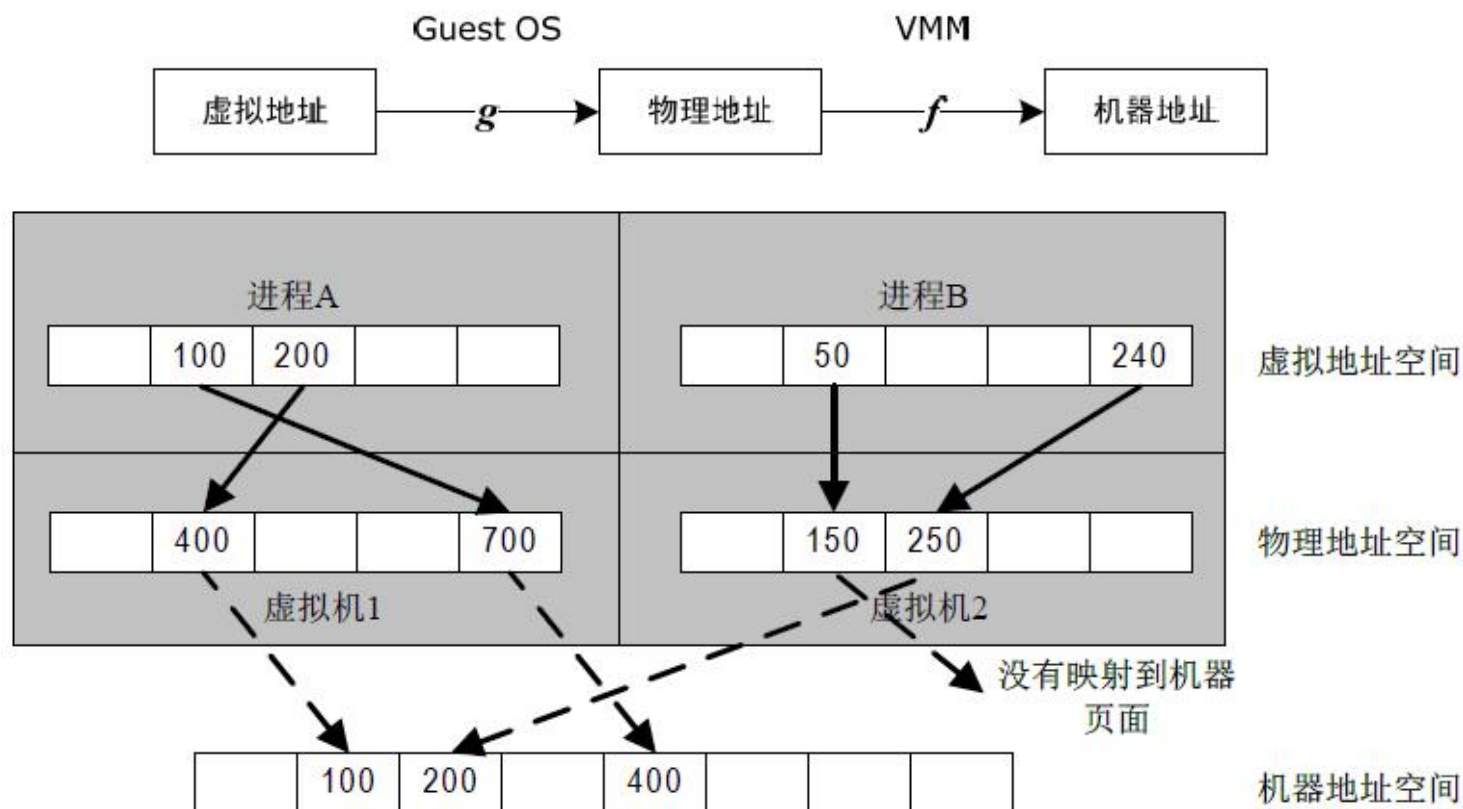
# 硬件辅助虚拟化



- 在Intel的VT-x技术中，CPU在Ring 0级之下还提供了**Root Mode**，VMM运行在Root Mode下。特权指令自动被VMM捕获，不需要进行二进制转换或调用Hypercall。
- Intel还对外设提供了VT-d和VT-c等技术，提供对外设虚拟化的支持。

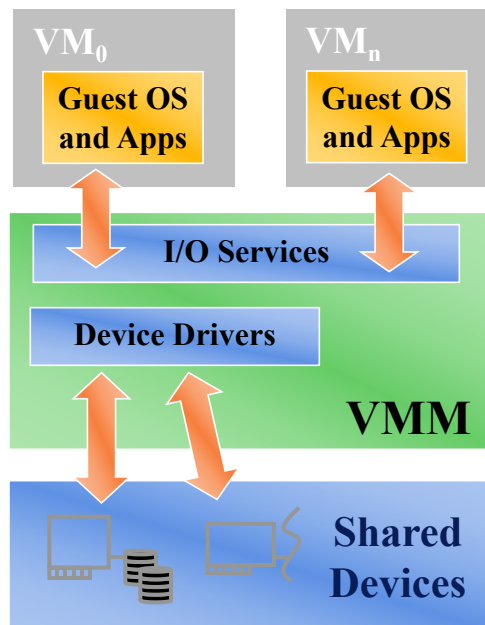
# 内存虚拟化

- VMM 通常采用分块共享的思想来虚拟计算机的物理内存。即将机器内存分配给虚拟机，并维护机器内存和虚拟机所见的“物理内存”的映射关系，使这些内存存在虚拟机看来是从地址0开始、连续的物理地址空间



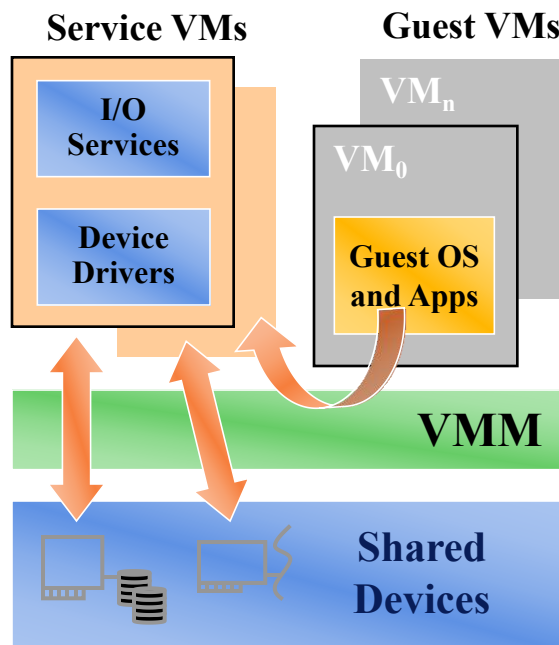
# I/O设备虚拟化模式

## Monolithic Model



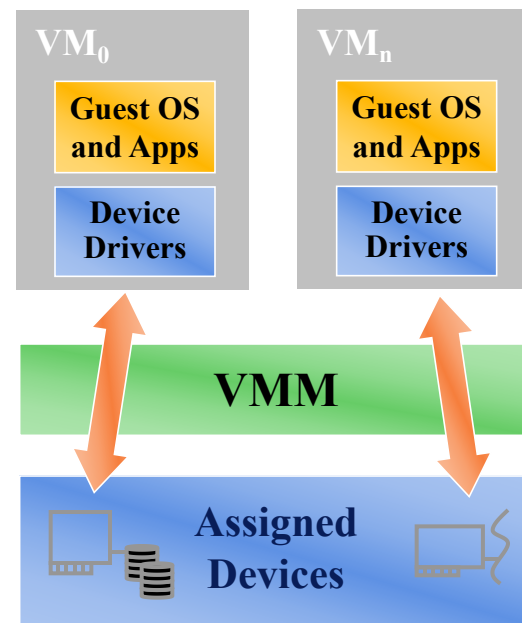
- 较高性能
- IO设备共享
- 支持虚拟机迁移
- 虚拟层过于复杂

## Service VM Model



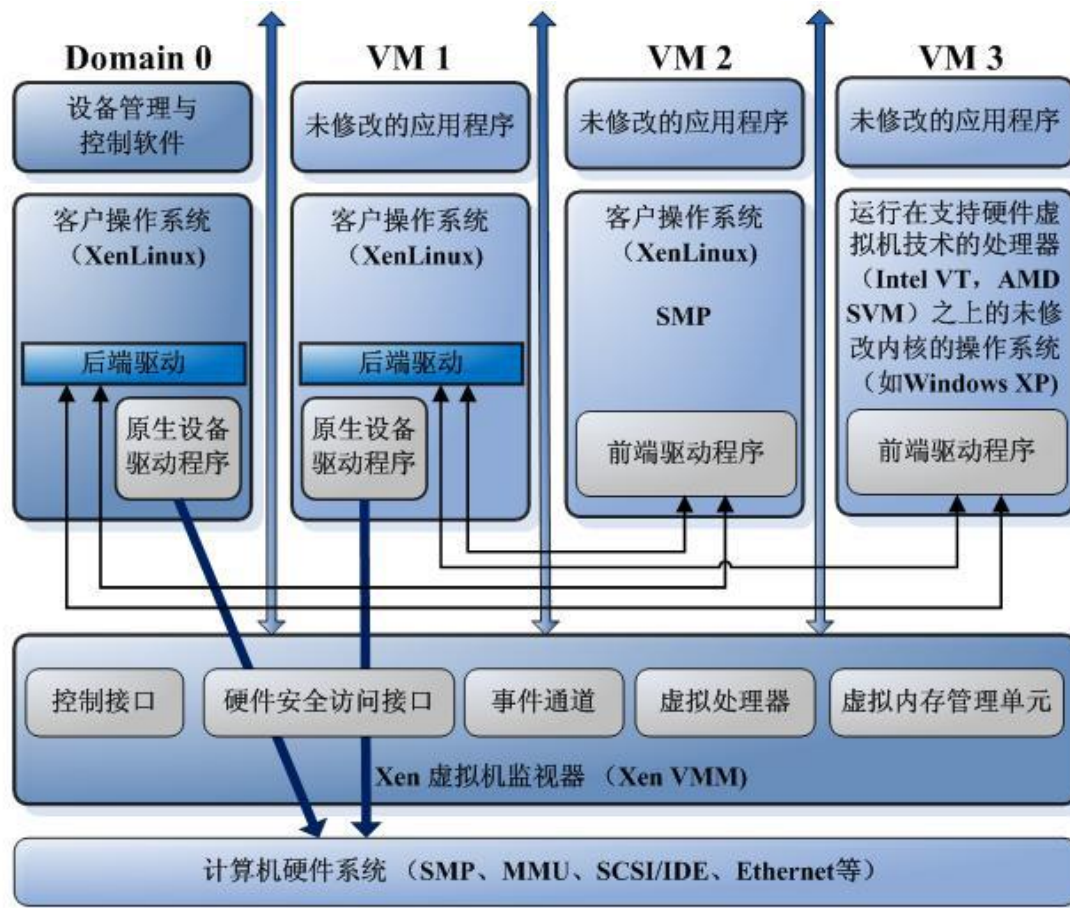
- 高安全性
- IO设备共享
- 支持虚拟机迁移
- 性能受影响

## Pass-through Model



- 最高性能
- 虚拟层更简洁
- 需要设备支持共享
- 迁移能力受影响

# 开源虚拟机—Xen



- Xen是由剑桥大学计算机实验室发起的开源虚拟机项目；
- 支持半虚拟化和全虚拟化（需要硬件支持）；
- Xen Hypervisor 是虚拟机管理器，负责CPU调度和内存分区，不负责网络和设备IO；
- Domain 0，负责管理其他虚拟机，提供管理接口；
- Domain U PV Guest：半虚拟化虚拟机；
- Domain U HVM Guest：全虚拟化虚拟机；
- Xen Hypervisor的管理接口可通过Libxenctrl库调用，来实施管理功能；

<https://www.xenproject.org>

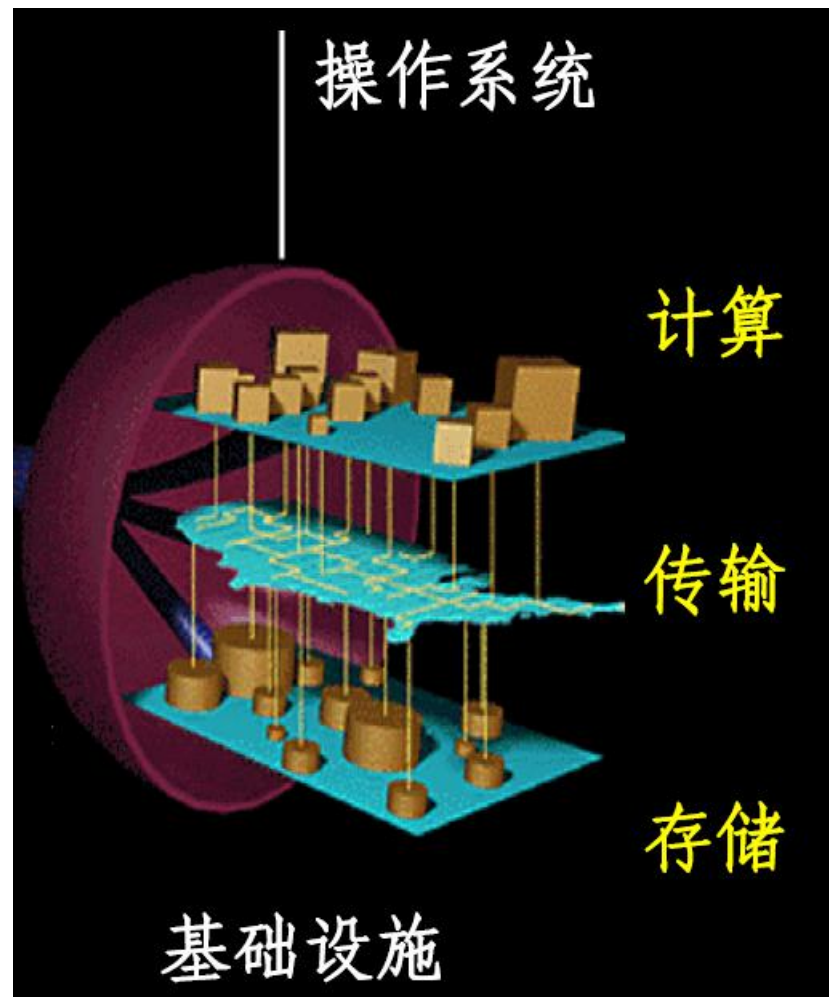
# 课程内容

---

- 云计算概念
- 云存储
  - 云存储概述
  - 分布式文件系统--GFS
- 典型云平台

# 存储：IT的基石

- IT的基础架构
  - 计算
  - 存储
  - 通信
- 存储：已经从服务器的附属设备发展成为独立的设备

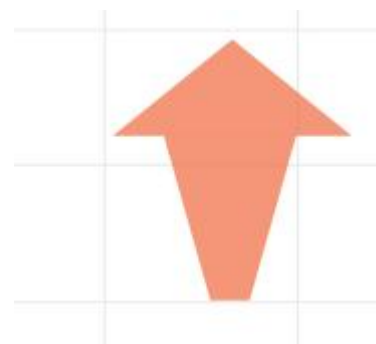


# 存储的发展对比计算的发展

- 单处理器---多处理器---多计算机---云计算
- 随机存储、硬盘、光盘---阵列---存储网---广域存储网



新型光电存储原理（器件、设备）  
(Flash, SSD, MRAM)



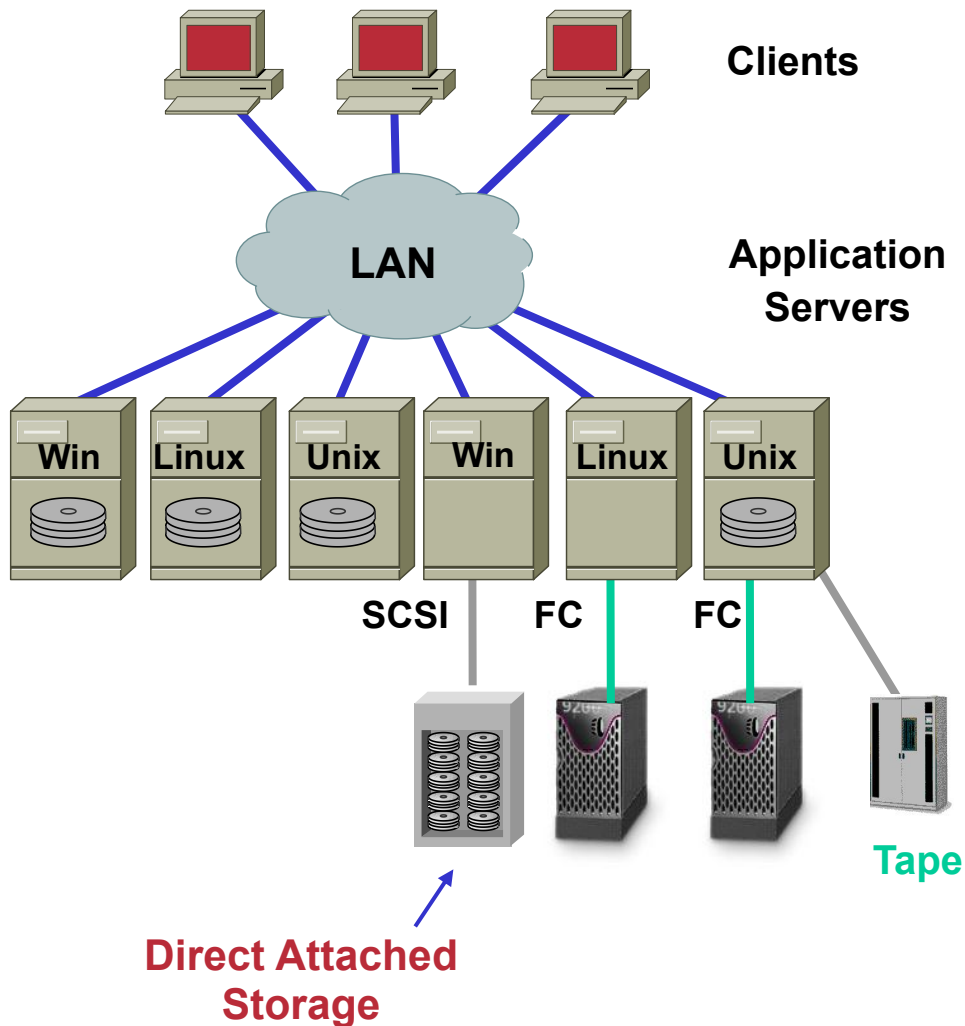
大规模网络存储系统  
(NAS, SAN, OBS)



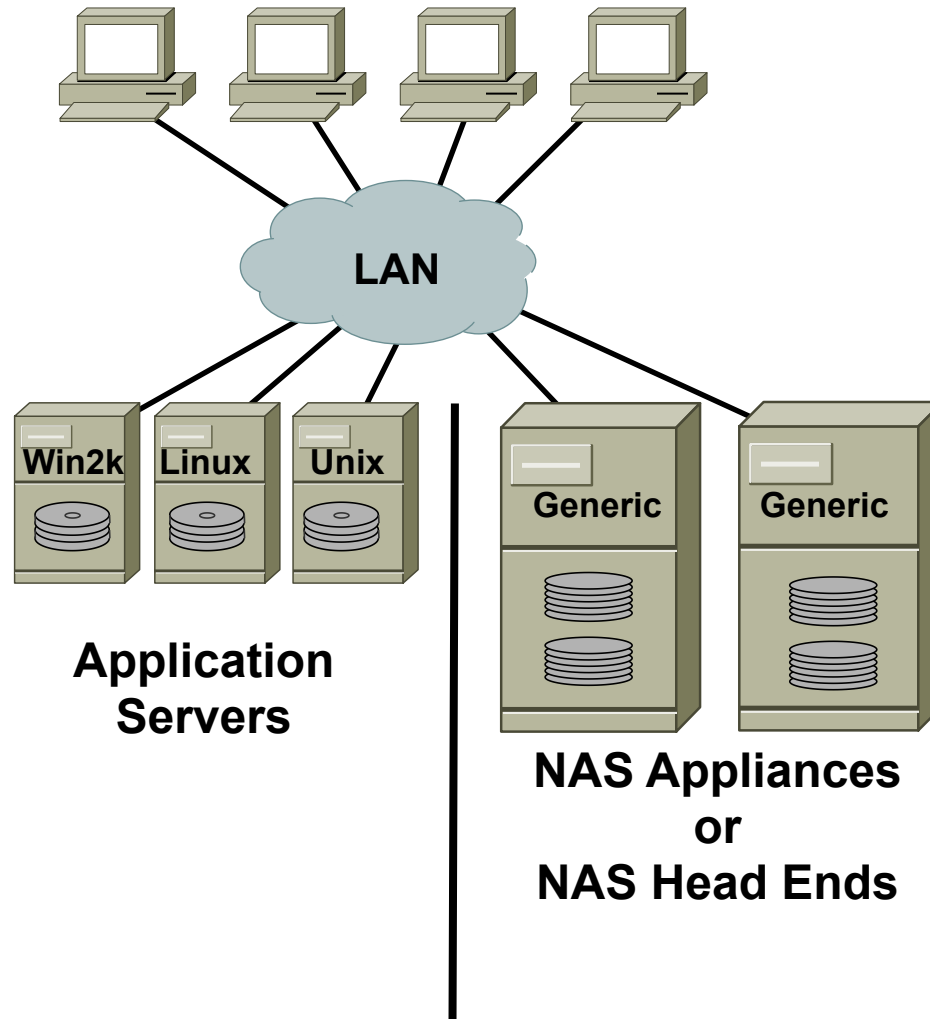
# 新型光电存储设备

- **Flash**: 又称闪存，是一种非易失性存储器**NVM**（**Non Volatile Memory**），允许在操作中被多次擦或写。闪存以块为读写单位，使其能够达到更高的集成度
- **SSD**（**Solid-State Disk**）：固态硬盘。非易失性存储器的数据访问速度介于易失性存储器和传统硬盘之间。将多个大容量闪存模块集成在一起，可制成以闪存为存储介质的固态硬盘
- **MRAM**（**Magnetoresistive Random Access Memory**）：磁随机存取存储器。下一代存储技术**MRAM**与**DRAM**（动态随机存储）存储技术相比，耗电量仅为**DRAM**三分之一，但读写速度却达到**DRAM**十倍，更适合于应用到智能手机和平板电脑

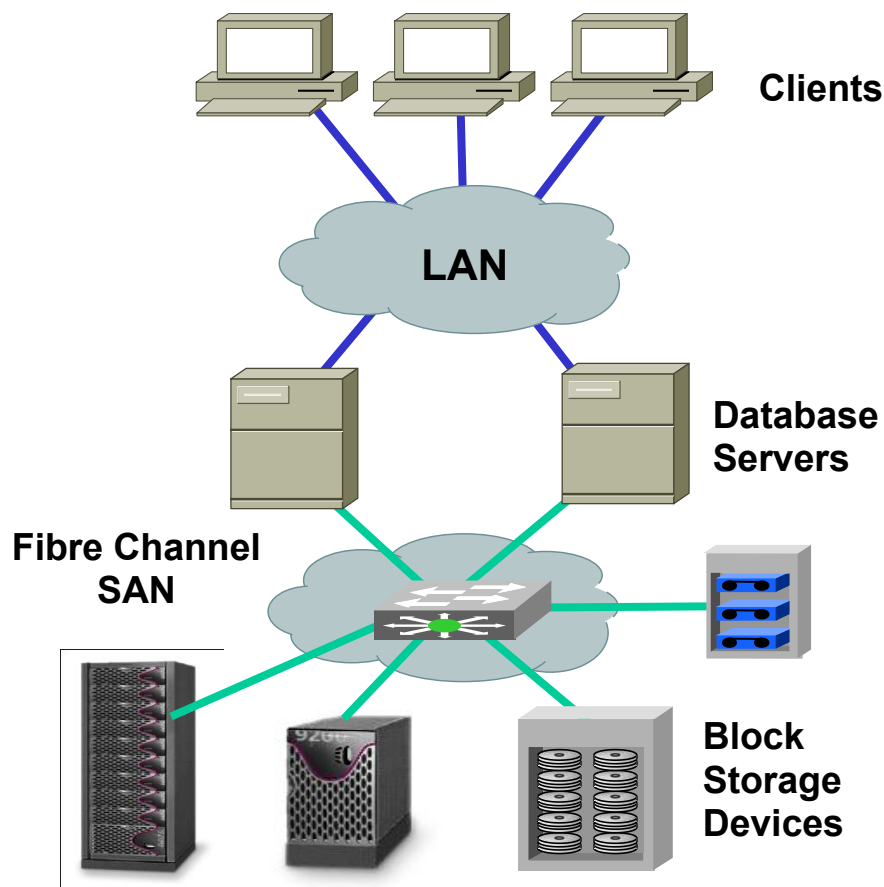
# 直连存储DAS (Direct Attached Storage)



# 网络连接存储NAS (Network Attached Storage)



# 存储区域网SAN (Storage Area Network)



Storage Area Network (SAN)

- 存储在块（**block**）级别而不是在文件级别
- 很高性能
- 存储是共享的
- 有很好的管理工具
- 互操作问题

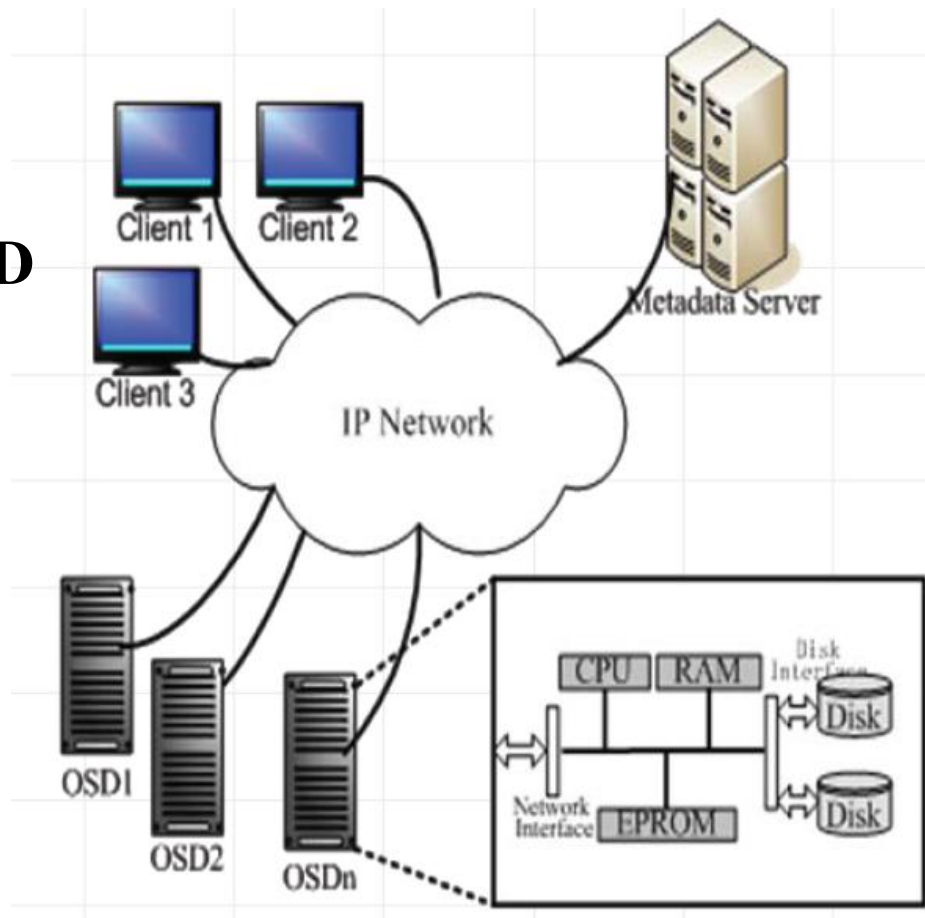
# 基于对象存储OBS (Object Based Storage)

- 主要模块

- 元数据服务器MDS (Metadata Server)
- 基于对象的存储设备OSD (Object-based Storage Device)
- 客户端 (Clients)

- OSD

- 对象管理
- 设备安全管理
- 网络通信



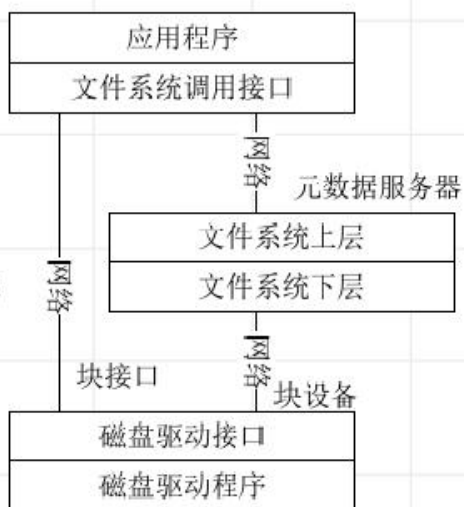
# 存储模式比较



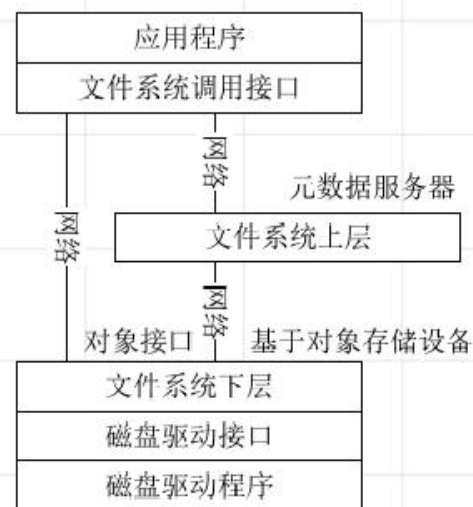
直连存储(DAS)



附网存储(NAS)

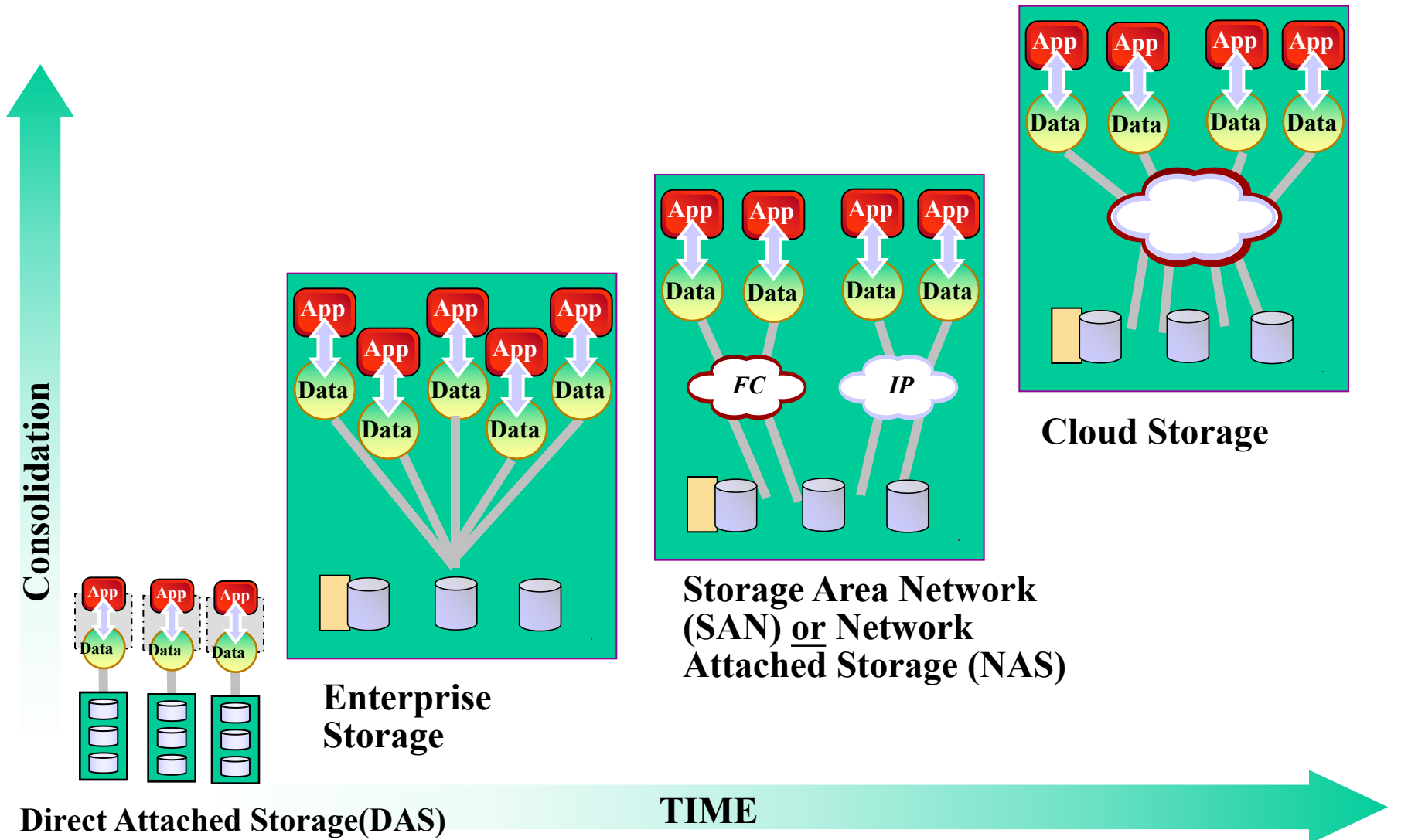


存储区域网(SAN)



基于对象存储(OBS)

# 存储技术的发展



# 云存储的定义

- 云计算将IT相关的能力以服务的方式提供给用户，允许用户在不了解提供服务的技术、没有相关知识以及设备操作能力的情况下，通过Internet获取需要服务
- 云存储是指通过集群、网络、分布式文件系统等技术，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能的系统
- 云存储是一个以数据存储和管理为核心的云计算系统



# 云存储与传统存储的区别

- 功能需求:

- 传统存储系统: 面向如高性能计算、事务处理等应用;
- 云存储系统: 面向多种类型的网络在线存储服务

- 性能需求:

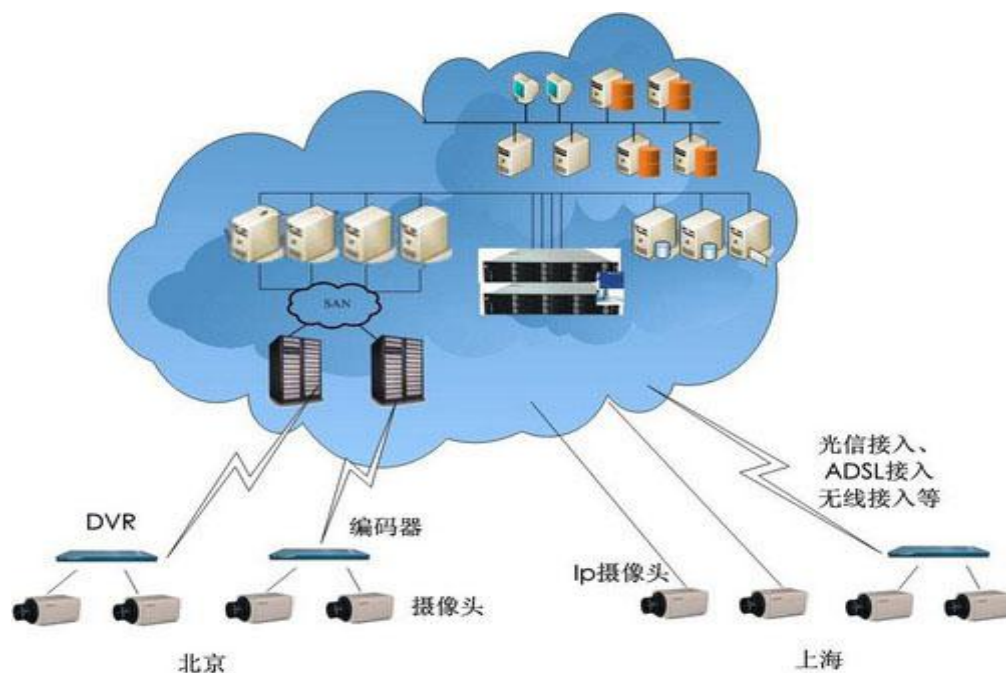
- 传统存储: 数据存取效率、数据安全、可靠性
- 云存储: 由于用户规模大、服务范围广、网络环境复杂多变等特点, 实现高质量的云存储服务必将面临更大的技术挑战

- 数据管理:

- 传统存储: **POSIX**的传统文件访问
- 云存储系统: 不仅要提供类似于**POSIX**的传统文件访问, 还要能够支持海量数据管理并提供公共服务支撑功能, 以方便云存储系统后台数据的维护。

# 云存储主要应用场景

- 个人级云存储
  - 网络硬盘
  - 在线文档编辑
- 企业级云存储
  - 存储空间的租赁
  - 数据备份和容灾
  - 视频监控系统



# 云存储的优点

- 可靠性（**Reliability**）
  - 99.999%服务时间（**uptime**）
  - 在广域的数据中心复制数据（**Replication of data across the wide network of datacenters**）
- 可扩展性（**Scalability**）
  - 按需服务（**On-demand services**）
  - 按需付费（**Pay as much as you need**）
- 简洁（**Simplicity**）
  - 云存储的操作者自动配置用户所需资源
- 节省费用（**Cost Efficient**）
  - 无需前期投资
- 安全（**Security**）
  - 由于复制，可抵御 **DDOS**
  - 提供安全的解决方案

# 云存储的挑战

- 网络可用性（**Network availability**）
  - 当网络连接中断时会发生什么？
- 传输数据的性能
  - 时延，可容忍性？
- 安全（**Security**）
  - 数据位置（**Data location**）？
  - 钓鱼网站（**Phishing site**）？
  - 物理安全性？
- 信任（**Trust**）
  - 敏感数据信息的泄漏
- 其他
  - 不付费，是否会导致数据丢失？
  - 谁拥有数据？
- 数据安全性
- 数据完整性
- 能耗
- 存取速度
- 经济性
- 可靠性

# 课程内容

---

- 云计算概念
- 云存储
  - 云存储概述
  - 分布式文件系统--GFS
- 典型云平台

# GFS设计动机

- **Google**需要一个支持海量存储的文件系统
  - 购置昂贵的超级计算机或大量廉价的服务器？
- 为什么不使用当时现存的文件系统？
  - **Google**所面临的问题与众不同
  - 不同的工作负载，不同的设计优先级（廉价、不可靠的硬件）
  - 需要设计与**Google**应用和负载相符的文件系统

# 实际应用需求

- 很高的模块故障率
  - 廉价的商用模块随时都可能崩溃
- 相关多的大文件
  - 上百万
  - 每个是100MB或更大，一般是几GB大小
- 文件只写一次，一般被追加
  - 可能是并发的
- 大规模的流式读（streaming reads）
- 低时延下的高吞吐率（high sustained throughput favored over low latency）

# GFS 设计理念

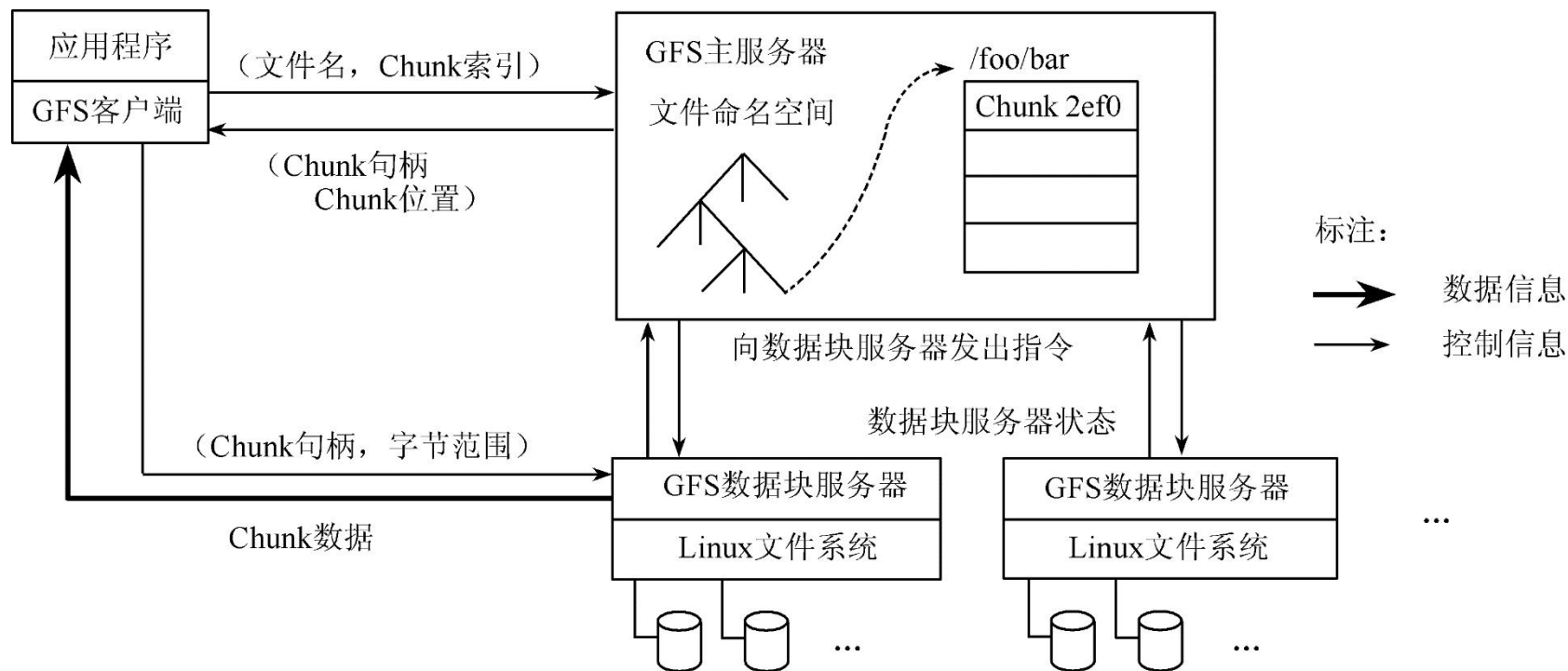
- GFS将容错的任务交给文件系统完成，利用软件的方法解决系统可靠性问题，使存储的成本成倍下降。GFS将服务器故障视为正常现象
- 采用多种方法，从多个角度，使用不同的容错措施，确保数据存储的安全、保证提供不间断的数据存储服务
  - 文件以 **chunks** 为单元存储，大小64MB
  - 通过复制（**replication**）保证可靠性
  - 单个**master**协调存取，集中化管理

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, The Google File System, 19th ACM Symposium on Operating Systems Principles , 2003



# 系统架构（1）

- 单个**Master**（Single master）
- 多个**Chunkserver**（Multiple chunkservers）



## 系统架构（2）

---

- **Client**（客户端）：应用程序的访问接口
- **Master**（主服务器）：**管理**节点，在逻辑上只有一个，保存系统的元数据，负责整个文件系统的管理
- **Chunk Server**（数据块服务器）：负责具体的**存储**工作。数据以文件的形式存储在**Chunk Server**上

# 单个Master

- 分布式系统中单个Master
  - 单点故障（Single point of failure）
  - 扩展性瓶颈（Scalability bottleneck）
- GFS的解决方案
  - 引入Shadow masters
  - 最小化Master的操作
    - 只用于元数据管理，与数据移动无关
      - 客户端会缓存元数据
    - chunk size较大
    - 在数据修改（data mutations）过程中，master会授权一个primary replica
- Simple, and good enough!

# 元数据（Metadata）

- 全局的元数据存储在**Master**
  - 文件和**Chunk**的命名空间（**namespaces**）
  - 映射**files**到 **chunks**
  - 每个**chunk**的复制的位置
- 所有信息可放入内存（**64bytes/chunk**）
  - 快速
  - 易于存取
- 用操作日志( *operation log* )记录重要的元数据变更
  - 永久化到本地磁盘
  - 复制（**replicated**）
  - 用于快速恢复的检查点(**checkpoints for faster recovery**)

# GFS的特点

- 客户端首先访问Master节点，获取交互的Chunk Server信息，然后访问这些Chunk Server，完成数据存取工作。这种设计方法实现了控制流和数据流的分离
- Client与Master之间只有控制流，而无数据流，极大地降低了Master的负载
- Client与Chunk Server之间直接传输数据流，同时由于文件被分成多个Chunk进行分布式存储，Client可以同时访问多个Chunk Server，从而使得整个系统的I/O高度并行，系统整体性能得到提高

# Mutations (1)

- 修改（Mutation） = write or append

- 必须对所有复制者（replica）操作

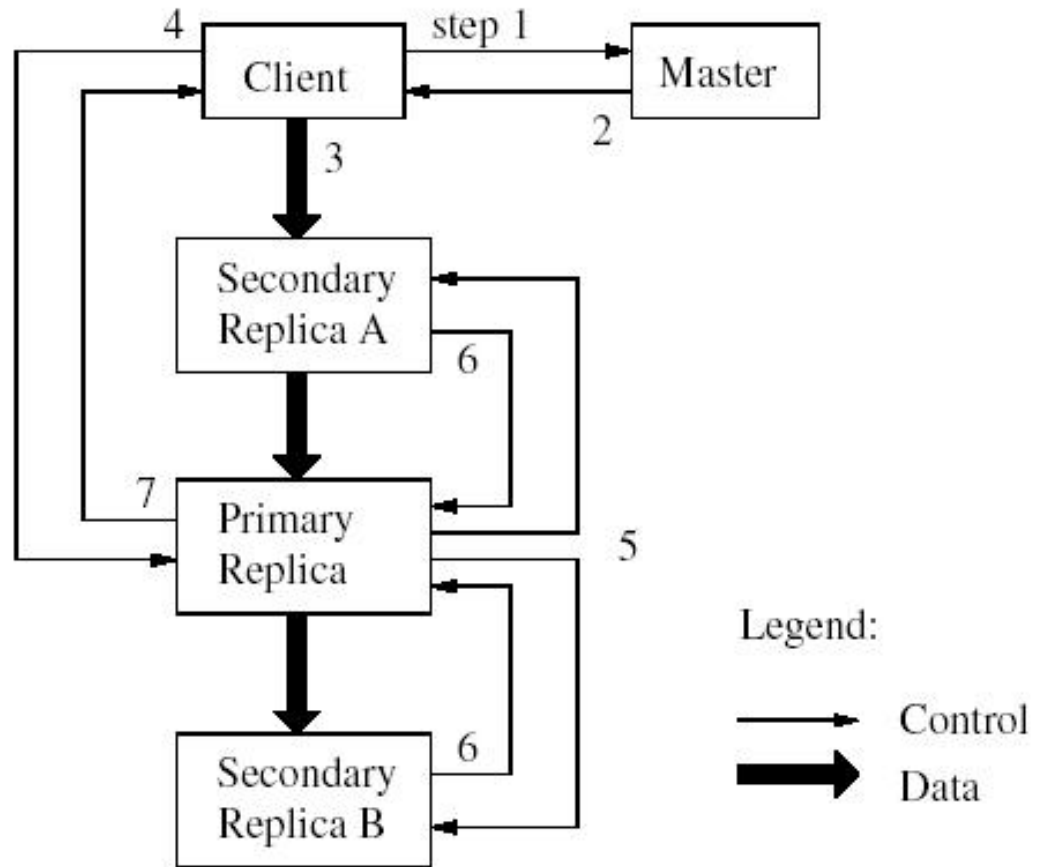
- 租约（Lease）机制：

- master 选择一个replica 做primary，授予租约（lease）

- primary 定义了 mutation 的顺序，所有复制按这个顺序

- 数据流与控制流分离

- 目标：最小化master 的操作



# Mutations (2)—传送数据到内存

## 1. Client向Master发送请求。

- Master（记录着整个集群中的replica小组，每个小组中有哪些机器，每个机器的IP地址等）选择一个replica做primary，并通知lease的授予时间和有效期

## 2. Master 回复Client关于 Replica 小组的信息（各机器 IP 地址、Primary等）

## 3. Client 在 Replica 小组中找一个最邻近的replica，传送文件到其内存。该replica在接收数据的同时，传递到小组中与其最邻近的replica的内存中去。

- 当小组中所有replica都收到了完整的文件数据，会分别向Client报告“文件数据已经完整收到”

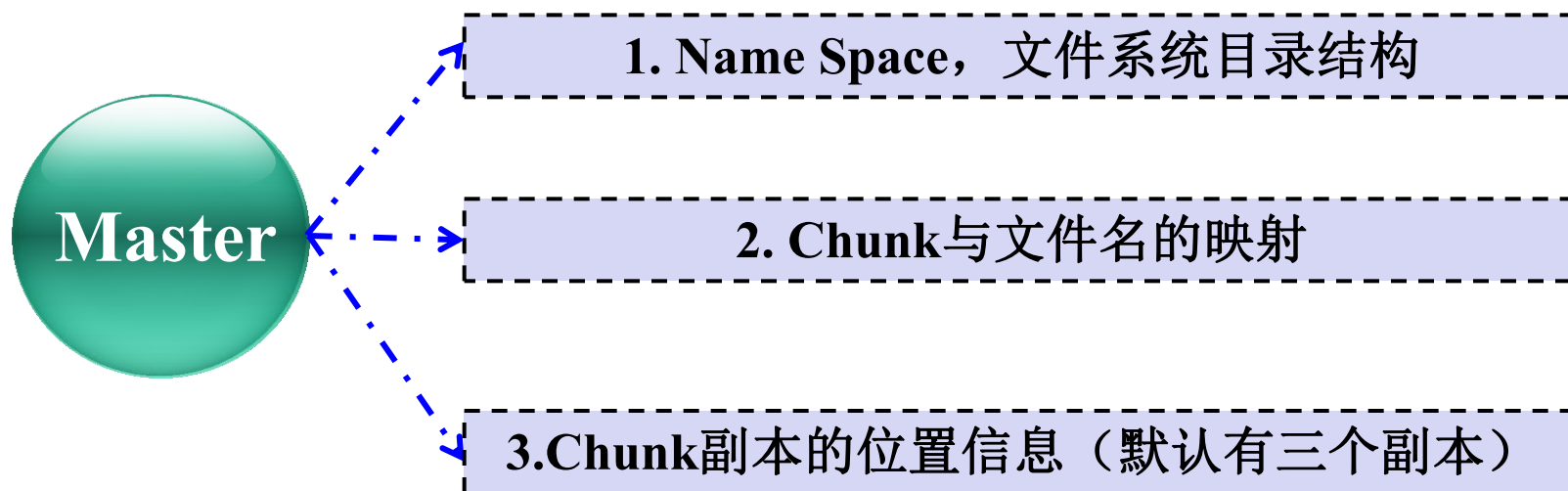
## Mutation (3)—写入硬盘

---

4. **Client** 向**primary replica**请求将各个 **replica** 内存中的有关文件，写入硬盘
5. **Primary**编写按顺序写入硬盘的请求序列，并发送给其他**replica**
6. 小组中所有 **replica**都把内存中缓存的文件，逐个按**primary**指定的顺序写入硬盘，并向 **primary**报告“文件数据已经完整写入硬盘”
7. **Primary**回复 **client**，文件操作已经完成。如出错，则报告故障



# GFS的容错机制



# Master容错

- 单个Master，对于前两种元数据，GFS通过**操作日志**来提供容错功能
- 第三种元数据信息保存在各个Chunk Server上，Master故障时，**磁盘恢复**
- GFS还提供了Master远程的**实时备份**,防止Master彻底死机的情况

# Chunk Server 容错（1）

---

- **GFS中的每一个文件被划分成多个Chunk，Chunk的默认大小是64MB**
- **Chunk Server存储的是Chunk的副本，副本以文件的形式进行存储**
- **每个Chunk又划分为若干Block（64KB），每个Block对应一个32bit的校验码，保证数据正确（若某个Block错误，则转移至其他Chunk副本）**

## Chunk Server 容错（2）

- 采用副本方式实现Chunk Server容错
  - 每一个Chunk有多个存储副本（默认为三个），分布存储在不同的Chunk Server上，用户态的GFS不会影晌Chunk Server的稳定性
  - 副本的分布策略需要考虑多种因素，如网络的拓扑、机架的分布、磁盘的利用率等
  - 对于每一个Chunk，必须将所有副本全部写入成功，才视为成功写入
- 尽管一份数据需要存储三份，好像磁盘空间的利用率不高，但综合比较多种因素，加之磁盘的成本不断下降，采用副本无疑是最简单、最可靠、最有效，而且实现难度最小的一种方法

# 便利的系统管理

- **GFS**集群中通常有非常多的节点，需要相应的技术支撑
- **GFS**构建在不可靠廉价计算机之上的文件系统，由于节点数目众多，故障发生十分频繁
- 新的**Chunk Server**加入时，只需裸机加入，大大减少**GFS**维护工作量

# 总结：GFS

---

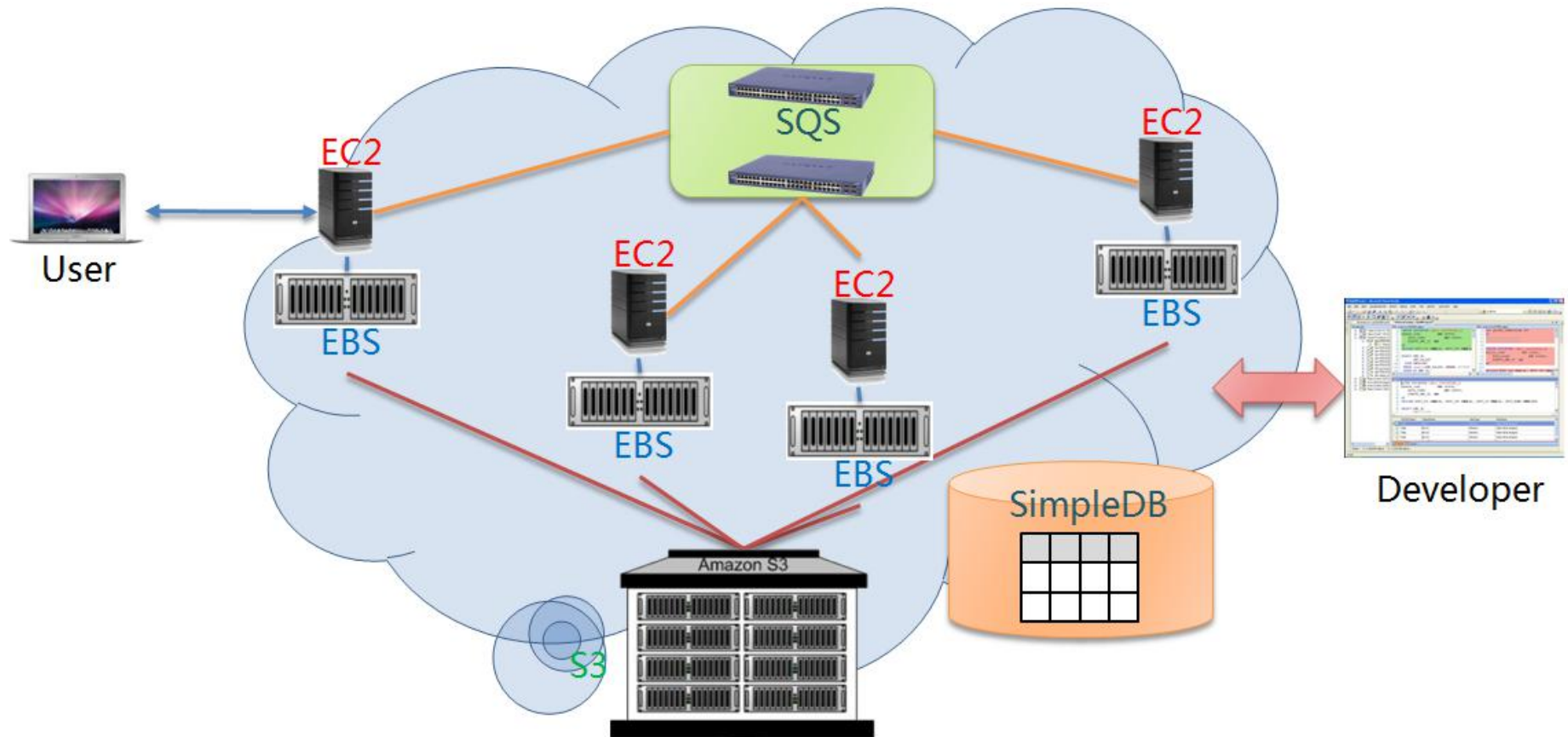
- **GFS展示了如何在商用硬件上很好地支持大规模的工作负载的处理**
  - 可容忍频繁的硬件错误
  - 优化大文件的追加和读操作（**appended and read**）
  - 兼容常用的文件读写接口
  - 方法简单有效（如单个 **master**）
- **GFS将容错的任务交给文件系统完成，利用软件的方法解决系统可靠性问题，使存储的成本成倍下降**

# 课程内容

---

- 云计算概念
- 典型云平台
  - **Amazon**
  - **Hadoop**

# Amazon Elastic Computing Cloud



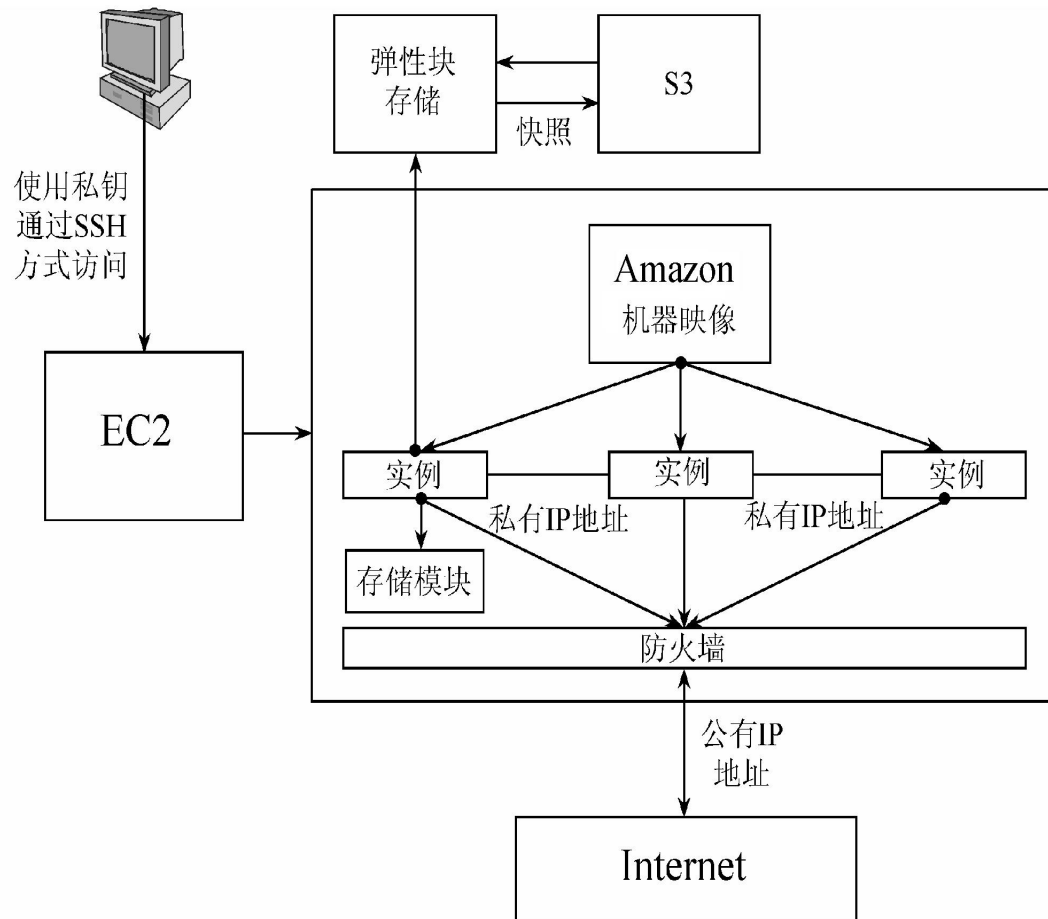
- **SQS: Simple Queue Service**
- **EC2: Running Instance of Virtual Machines**
- **EBS: Elastic Block Service, Providing the Block Interface, Storing Virtual Machine Images**
- **S3: Simple Storage Service,**
- **SimpleDB: Simplified Database**



# 弹性计算云EC2主要特性

- **灵活性**：EC2允许用户对运行实例类型、数量自行配置，还可以选择实例运行的地理位置，根据用户的需求随时改变实例的使用数量
- **低成本**：EC2使得企业不必为暂时的业务增长而购买额外的服务器等设备。EC2的服务都是按小时来收费，价格合理
- **安全性**：EC2向用户提供了一整套安全措施，包括基于密钥对机制的SSH方式访问、可配置的防火墙机制等，同时允许用户对它的应用程序进行监控
- **易用性**：用户可以根据Amazon提供的模块自由构建自己的应用程序，同时EC2还会对用户服务请求自动进行负载平衡
- **容错性**：利用系统提供的诸如弹性IP地址等机制，在故障发生时EC2能最大程度地保证用户服务仍能维持在稳定的水平

# EC2基本架构



# EC2基本概念

- 用户使用EC2服务的第一步就是要创建一个自己的AMI（Amazon Machine Image）—Amazon机器映像
- 用户创建好AMI后，实际运行的系统称为一个实例（Instance）

资 源	Small	Large	Extra Large	High-CPU Medium	High-CPU Extra Large
平台	32位	64位	64位	32位	64位
CPU	1ECU	4ECU	8ECU	5ECU	20ECU
内存	1.7GB	7.5GB	15GB	1.7GB	7GB
存储容量	160GB	850GB	1690GB	350GB	1690GB
实例类型名	m1.small	m1.large	m1.xlarge	c1.medium	c1.xlarge

实例类型和其相关配置

# S3的基本概念（1）

- 1. 对象

- 数据（任意类型）和元数据（描述数据的数据）
- 元数据是通过一对键-值（**Name-Value**）集合来定义

元数据名称	名称含义
Last-modified	对象被最后修改的时间
ETag	利用MD5哈希算法得出的对象值
Content-Type	对象的MIME（多功能网际邮件扩充协议）类型，默认为二进制/八位组
Content-Length	对象数据长度，以字节为单位

系统默认元数据

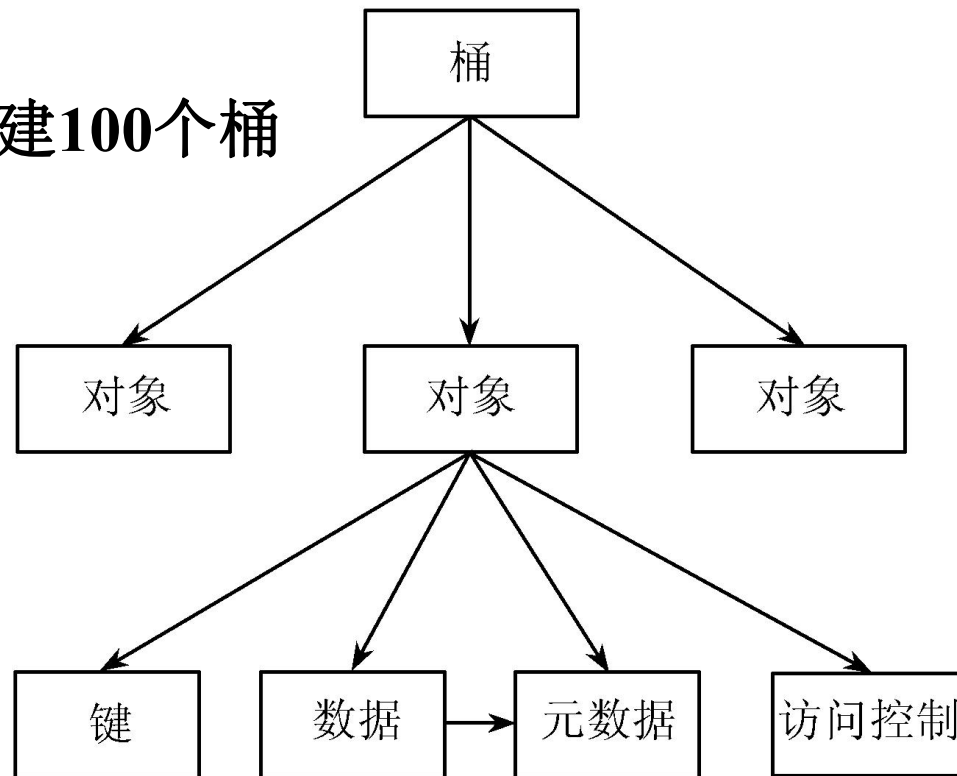
# S3的基本概念（2）

- 2. 键

- 对象的唯一标示符

- 3. 桶

- 存储对象容器（最多创建100个桶，不限桶中数量）



# 基本操作

- 目前S3支持的主要操作包括：**Get**、**Put**、**List**、**Delete**和**Head**

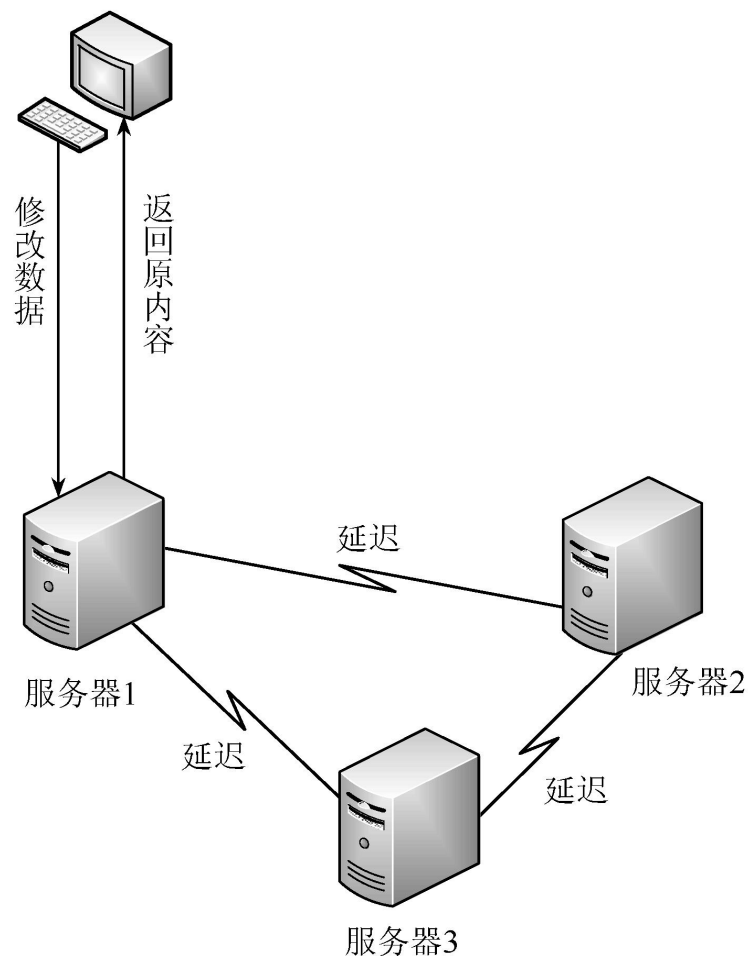
操作目标	Get	Put	List	Delete	Head
桶	获取桶中对象	创建或更新桶	列出桶中所有键	删除桶	无
对象	获取对象数据和元数据	创建或更新对象	无	删除对象	获取对象元数据

# 数据一致性模型（1）

- S3系统采用冗余存储
- 优势：某些服务器出现故障时用户仍然可以对其数据进行操作
- 弊端：用户在操作时可能会出现以下几种情况
  - 一个进程写入一个新的对象并立即尝试读取它，但在该改变被传送到S3的多个服务器前，服务器对该操作可能返回“键不存在”
  - 一个进程写入一个新的对象并立即尝试列出桶中已有的对象，但在该改变被传送到S3的多个服务器前，该对象很可能不会出现在列表中
  - 一个进程用新数据替换现有的对象并立即尝试读取它，但在该改变被传送到S3的多个服务器前，S3可能会返回以前的数据
  - 一个进程删除现有的对象并立即尝试读取它，但在该改变被传送到S3的多个服务器前，S3可能会返回被删除的数据
  - 一个进程删除现有的对象并立即尝试列出桶中的所有对象，但在该改变被传送到S3的多个服务器前，S3可能会列出被删除的对象

## 数据一致性模型（2）

- 为了保证用户数据的一致性而采取的一种折中手段
- 在数据被充分传播到所有的存放节点之前返回给用户的仍是原数据





# 课程内容

---

- 云计算概念
- 典型云平台
  - Amazon
  - Hadoop

# Hadoop概述

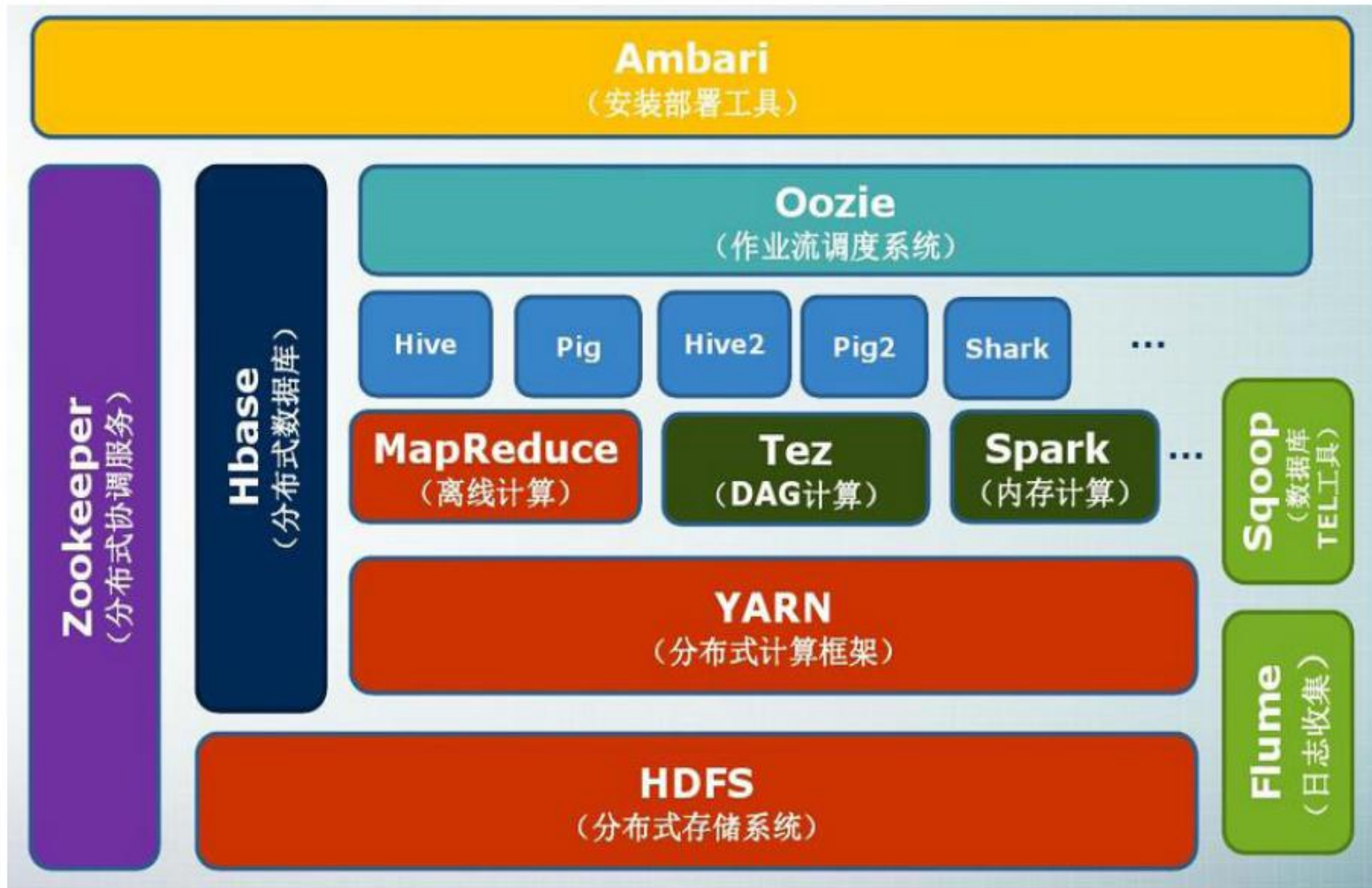
---

- **Hadoop**是Apache软件基金会旗下的一个开源分布式计算平台，为用户提供了系统底层细节透明的分布式基础架构
- 基于Java语言开发，具有很好的跨平台特性，并且可以部署在廉价的计算机集群中
- 核心是分布式文件系统**HDFS**和**MapReduce**
- 几乎所有主流厂商都围绕**Hadoop**提供开发工具、开源软件、商业化工具和技术服务，如谷歌、雅虎、微软、思科、淘宝、百度、网易、华为、中国移动等，都支持**Hadoop**

# Hadoop的发展历史

- Hadoop源自始于2002年的Apache Nutch项目（开源的网络搜索引擎）并且也是Lucene项目的一部分
- 2004年，Nutch基于GFS开发了分布式文件系统NDFS（Nutch Distributed File System），即HDFS前身
- 2005年，Nutch开源实现了谷歌的MapReduce
- 2006年2月，Nutch中的NDFS和MapReduce开始独立出来，成为Lucene项目的一个子项目，称为Hadoop
- 2007年，雅虎Hadoop集群系统（4000个处理器、1.5PB容量）
- 2008年1月，Hadoop正式成为Apache顶级项目
- 2008年4月，Hadoop成为最快排序1TB数据的系统，用一个910个节点的集群进行排序，只用了209秒
- 2009年5月，Hadoop把1TB数据排序时间缩短到62秒
- 已发展成为目前最具影响力的开源云计算平台

# Hadoop项目结构



# Hadoop项目结构

组件	功能
<b>HDFS</b>	分布式文件系统
<b>MapReduce</b>	分布式并行编程模型
<b>YARN</b>	资源管理和调度器
Tez	运行在YARN之上的下一代Hadoop查询处理框架
Hive	Hadoop上的数据仓库
<b>HBase</b>	<b>Hadoop上的非关系型的分布式数据库</b>
Pig	一个基于Hadoop的大规模数据分析平台，提供类似SQL的查询语言Pig Latin
Sqoop	用于在Hadoop与传统数据库之间进行数据传递
Oozie	Hadoop上的工作流管理系统
<b>Zookeeper</b>	<b>提供分布式协调一致性服务</b>
Storm	流计算框架
Flume	一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统
Ambari	Hadoop快速部署工具，支持Apache Hadoop集群的供应、管理和监控
Kafka	高吞吐量的分布式发布订阅消息系统
<b>Spark</b>	<b>类似于Hadoop MapReduce的通用并行框架</b>

# Hadoop的改进与提升

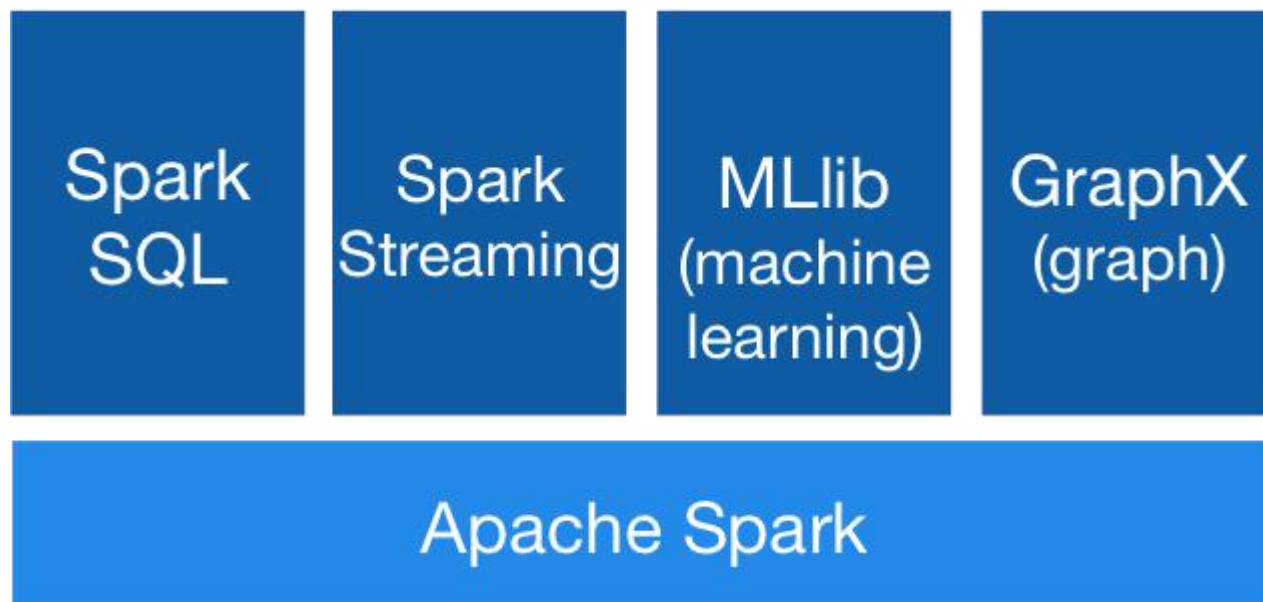
组件	Hadoop1.0的问题	Hadoop2.0的改进
<b>HDFS</b>	单一名称节点，存在单点失效问题	设计了 <b>HDFS HA</b> ，提供名称节点热备机制
	单一命名空间，无法实现资源隔离	设计了 <b>HDFS Federation</b> ，管理多个命名空间
<b>MapReduce</b>	资源管理效率低	设计了新的资源管理框架 <b>YARN</b>
	延迟高，而且不适合执行迭代计算	基于内存的分布式并行编程框架 <b>Spark</b> ，具有较高的实时性，并且较好支持迭代计算

# Spark发展历史

- Spark最初由美国加州伯克利大学（UCBerkeley）的AMP实验室于2009年开发，是基于内存计算的大数据并行计算框架，可用于构建大型的、低延迟的数据分析应用程序
- 2013年Spark加入Apache孵化器项目后发展迅猛，如今已成为Apache软件基金会最重要的三大分布式计算系统开源项目之一（Hadoop、Spark、Storm）
- Spark在2014年打破了Hadoop保持的基准排序纪录
  - Spark/206个节点/23分钟/100TB数据
  - Hadoop/2000个节点/72分钟/100TB数据
  - Spark用十分之一的计算资源，获得了比Hadoop快3倍的速度

# Spark基础架构

---





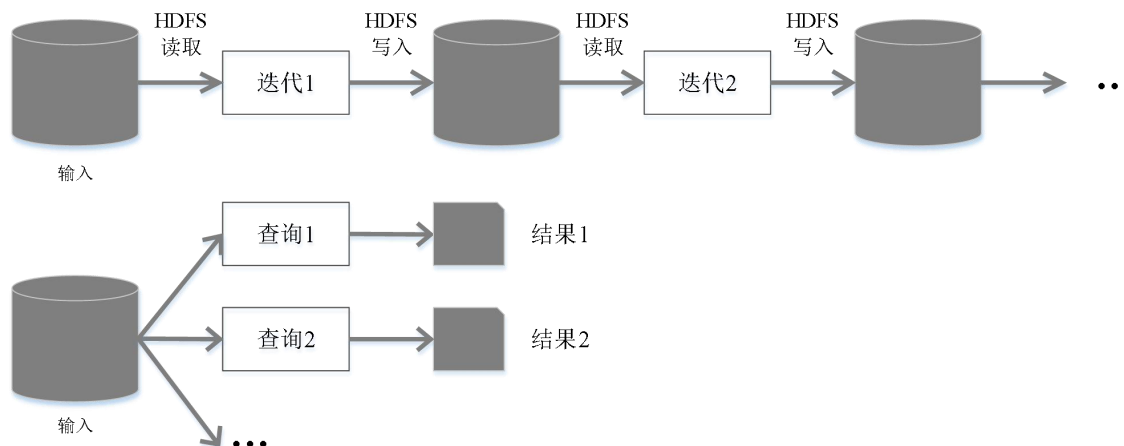
# Spark特点

- 运行速度快：使用**DAG**执行引擎以支持循环数据流与内存计算
- 容易使用：支持使用**Scala**（多范式编程语言，运行于**Java**平台）、**Java**、**Python**和**R**语言进行编程，可以通过**Spark Shell**进行交互式编程
- 通用性：**Spark**提供了完整而强大的技术栈，包括**SQL**查询、流式计算、机器学习和图算法组件
- 运行模式多样：可运行于独立的集群模式中，可运行于**Hadoop**中，也可运行于**Amazon EC2**等云环境中，并且可以访问**HDFS**、**Cassandra**、**HBase**、**Hive**等多种数据源

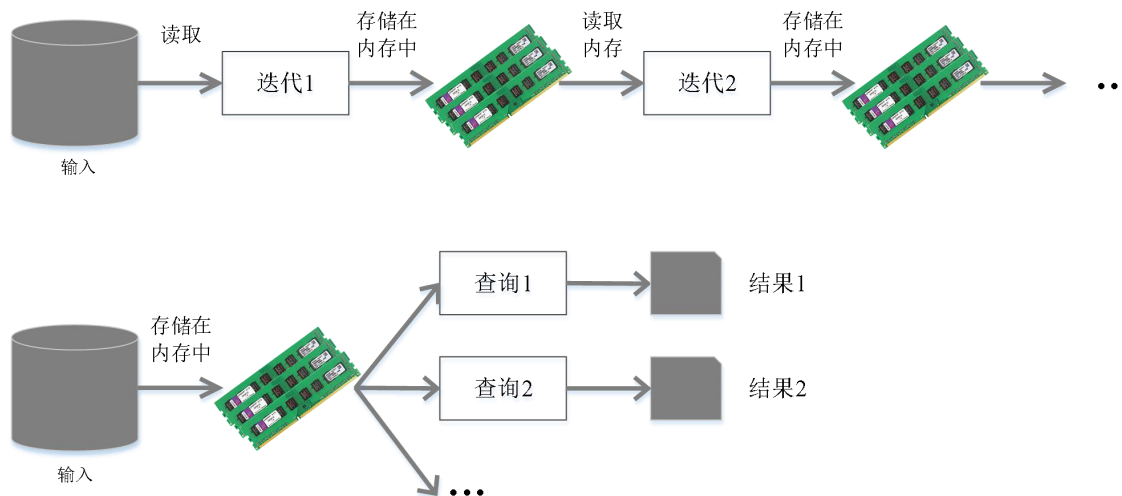
# Spark与Hadoop MapReduce的对比

- **Hadoop MapReduce存在缺点：**
  - 表达能力有限、延迟高
  - 任务之间的衔接涉及IO开销、磁盘IO开销大
  - 在前一个任务执行完成之前，其他任务就无法开始，难以胜任复杂、多阶段的计算任务
- **相比于Hadoop MapReduce，Spark主要优点：**
  - Spark的计算模式也属于**MapReduce**，但不局限于Map和Reduce操作，还提供了多种数据集操作类型，编程模型比Hadoop MapReduce更灵活
  - Spark提供了**内存计算**，可将中间结果放到内存中，对于迭代运算效率更高
  - Spark**基于DAG的任务调度执行机制**，要优于Hadoop MapReduce的迭代执行机制

# Spark与Hadoop的执行流程对比



(a) Hadoop MapReduce执行流程

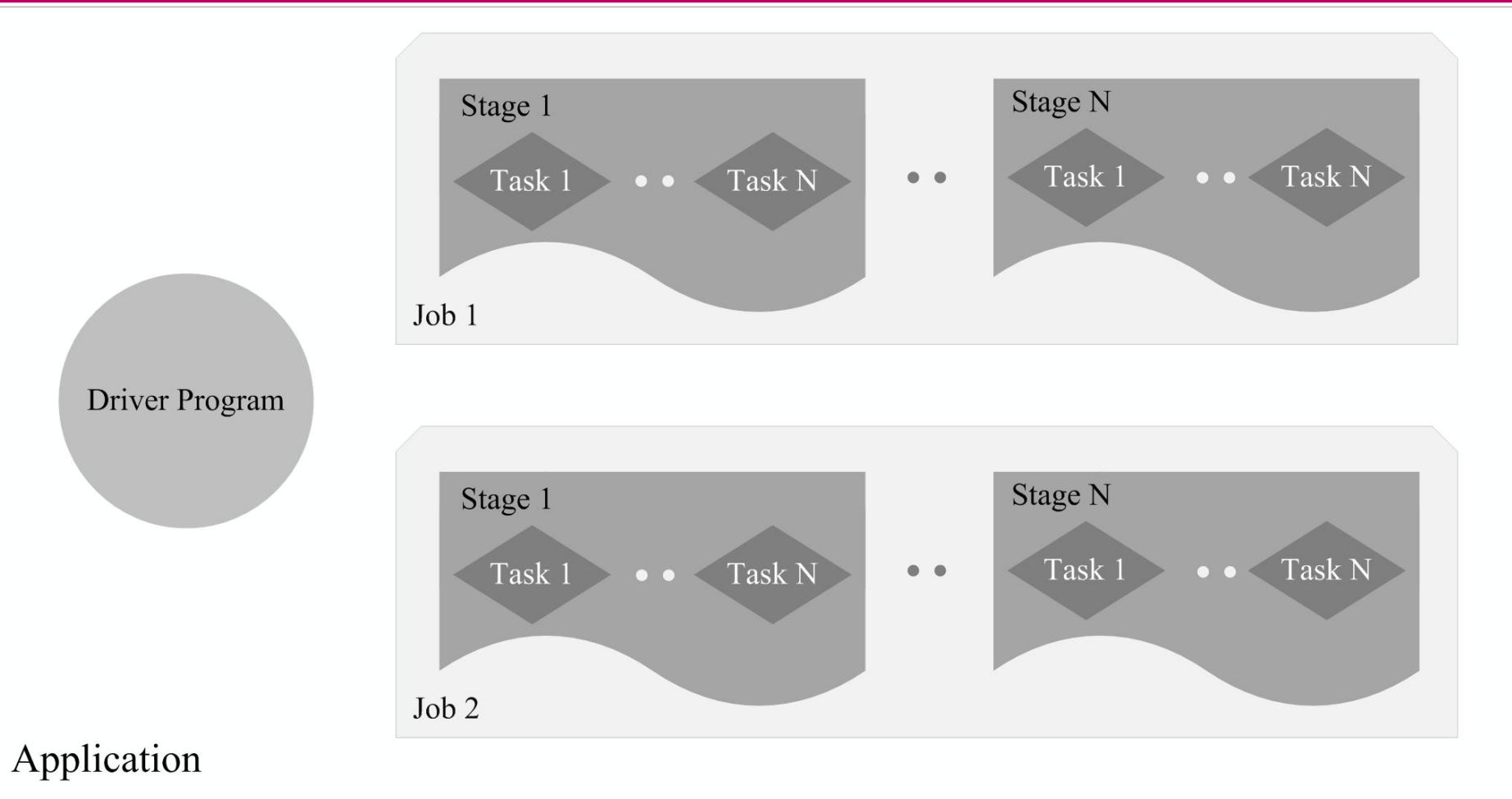


(b) Spark执行流程

# 基本概念

- **RDD**: 是Resilient Distributed Dataset（弹性分布式数据集）的简称，是分布式内存的一个抽象概念，提供了一种高度受限的共享内存模型
- **DAG**: 是Directed Acyclic Graph（有向无环图）的简称，反映RDD之间的依赖关系
- **Executor**: 是运行在工作节点（WorkerNode）的一个进程，负责运行Task
- **Application**: 用户编写的Spark应用程序
- **Task**: 运行在Executor上的工作单元
- **Job**: 一个Job包含多个RDD及作用于相应RDD上的各种操作
- **Stage**: 是Job的基本调度单位，一个Job会分为多组Task，每组Task被称为Stage，或者也被称为TaskSet，代表了一组关联的、相互之间没有Shuffle依赖关系的任务组成的任务集

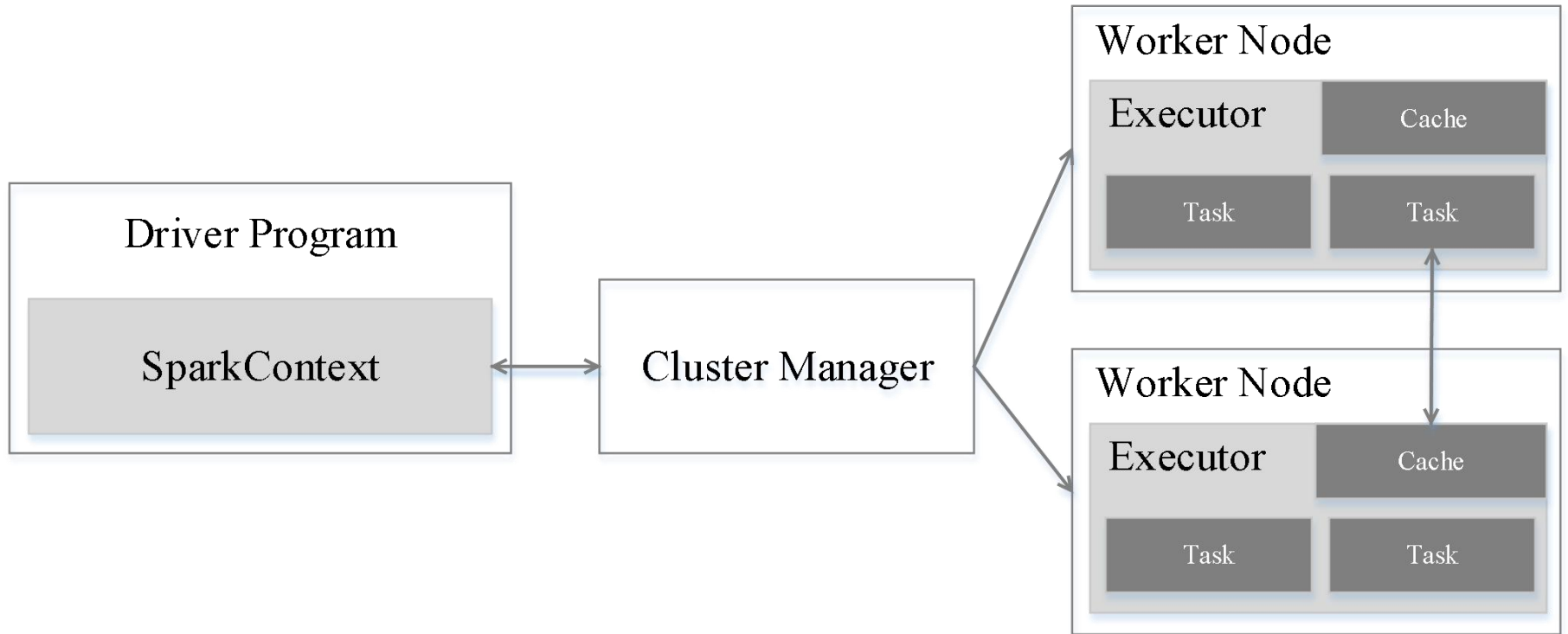
# 概念之间关系



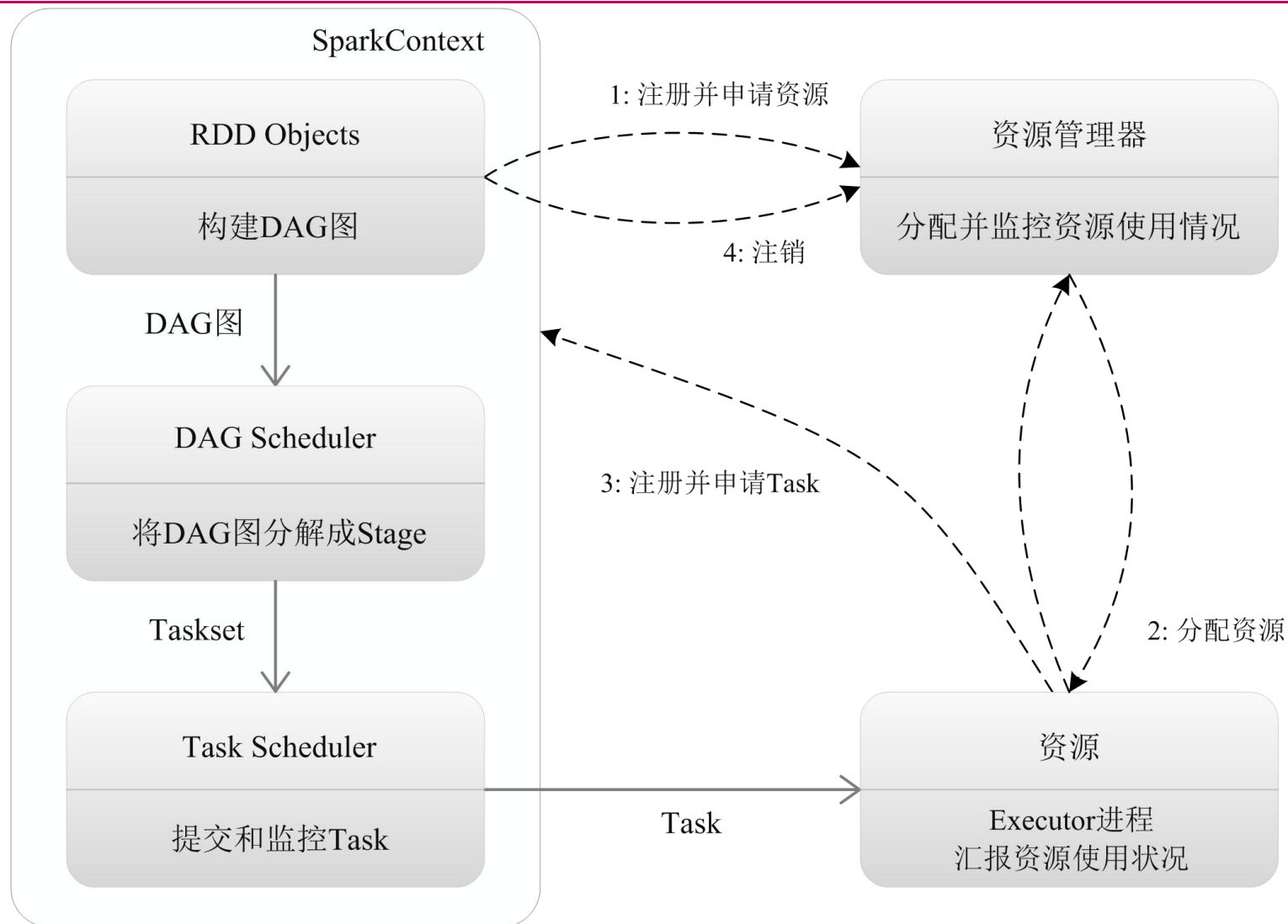
# 架构设计

- Spark运行架构包括集群**资源管理器**（Cluster Manager）、运行作业任务的**工作节点**（Worker Node）、每个应用的**任务控制节点**（Driver）和每个工作节点上负责具体任务的**执行进程**（Executor）
- 与Hadoop MapReduce计算框架相比，Spark的Executor有两个优点：
  - 利用多线程来执行具体的任务，减少任务启动开销
  - Executor中有一个BlockManager存储模块，会将内存和磁盘共同作为存储设备，有效减少IO开销

# Spark运行架构



# 运行基本流程





# Hadoop/Spark

---

- **Hadoop**是以一种可靠、高效、可伸缩的方式对大数据进行分布式处理的软件框架
  - 高可靠性
  - 高效性
  - 高可扩展性
  - 高容错性
  - 成本低
  - 运行在Linux平台上
  - 支持多种编程语言
- **Spark**是基于内存的分布式并行编程框架，有较高的实时性，可较好地支持迭代计算

# 课程小结

---

- 虚拟化技术

- 虚拟化原理，虚拟化技术的类型
- VMM：全虚拟化、半虚拟化、硬件虚拟化

- 典型云计算平台

- 工业界：Amazon
- 开源：Hadoop

# 结束语

- 高性能计算与云计算是当前高速发展的热点技术
  - 受到工业界、学术界和各国政府的高度关注
  - 大数据时代的到来，新一轮信息化浪潮的来临，隐含着巨大的机遇
- 同时，也有众多挑战，本课程只是一个开始
- 希望：未来继续学习、研究，运用大数据技术

# 推荐网站和读物

---

- 《云计算》（第三版）
  - 第2、3、5、6、7章
- **Spark**
  - <https://spark.apache.org>
- **Spark Programming Guide**
  - <https://spark.apache.org/docs/latest/programming-guide.html>