云计算回忆版总结

云计算课回忆版试题 — 完整解答与知识点总结

1. 云计算与人工智能、物联网、移动互联网、大数据的联系

云计算是上述四类技术发展的基础设施底座。

- **云计算为 AI 提供算力与弹性资源池**：深度学习训练需要大量 GPU/TPU，云平台可按需提供并通过虚拟化技术快速扩容。
- **云计算为物联网提供海量设备接入能力**：IoT 设备产生的海量数据需在云端进行汇聚、处理与分析。
- **云计算支撑移动互联网服务高并发与高可用**：APP 后端常部署在云平台，实现弹性伸缩与自动化运维。
- **云计算是大数据处理的运行平台**：Hadoop/Spark 等大数据框架大多部署于云环境，实现分布式存储与计算。

知识点总结

云计算提供基础设施层，AI/IoT/大数据/移动互联网互为依赖，最终形成云生态体系。

2. 简述 KVM 架构

KVM（Kernel-based Virtual Machine）属于 Linux 内核级虚拟化技术，其核心结构如下：- **KVM 模块**：将 Linux 内核转换为 Type-1 虚拟机监控器（Hypervisor）。

- **QEMU**：提供设备虚拟化与指令翻译。
- **硬件辅助虚拟化（Intel VT-x / AMD-V）**：CPU 将虚拟化功能下沉到硬件层，提高性能。
- **每个 VM 对应一个 Linux 进程**：利用 Linux 原生调度、内存管理、I/O 系统。

架构中，Linux 内核负责控制，KVM 模块提供虚拟化能力，QEMU 提供设备模拟。

知识点总结

KVM = Linux 内核 + KVM 模块 + QEMU + 硬件辅助虚拟化。

3. 虚拟机监视器（Hypervisor）的作用

Hypervisor 是用于创建和运行虚拟机的核心软件，其作用包括：

- **硬件资源抽象与分配**：CPU、内存、磁盘、网络的虚拟化。
- **隔离**：不同 VM 之间互不影响，实现强隔离。
- **调度**：管理 VM 对底层物理资源的调度。
- **安全管理**：监控 VM 行为、防止越权访问。

- **性能优化**：利用硬件虚拟化特性提升执行效率。

常见 Hypervisor：KVM、Xen、VMware ESXi、Hyper-V。

知识点总结

Hypervisor 是虚拟化平台的核心，实现资源抽象、隔离与调度。

4. CAP 理论

CAP 理论指出：在分布式系统中无法同时满足以下三个属性：

- **Consistency（一致性）**：所有节点读到相同的数据。
- **Availability（可用性）**：每个请求都能得到响应。
- **Partition Tolerance（分区容忍性）**：系统能在网络分区情况下继续运行。

分布式系统必须在 C 与 A 之间做权衡，但 P 必须保证。
Examples：

- CP：HBase、ZooKeeper（一致性优先）
- AP：Cassandra、Dynamo（可用性优先）

知识点总结

CAP 核心：分区容错必须要，C 和 A 无法同时兼得。

5. 软件定义网络（SDN）的定义、架构图与解释

定义： SDN 是通过控制层与数据层分离，实现网络集中控制、可编程配置的一种网络架构。

三层结构：

1. **应用层（Application）**：业务策略，如 ACL、防火墙、流量工程。
2. **控制层（Controller）**：网络大脑（如 OpenDaylight、ONOS）。
3. **数据层（Data Plane）**：可编程交换机（如 Open vSwitch，OVS）。

核心思想： 集中控制 + 可编程 + 南北向接口分层架构（OpenFlow）。

知识点总结

SDN 通过控制分离，提高灵活性与集中管理能力。

6. 轻量级虚拟化技术是什么？优势与不足？代表技术？

轻量级虚拟化 = 容器化技术。

代表：Docker、LXC、containerd、K8s 中的 Pod。

特点：

- 不模拟硬件，不需要完整 OS，只运行应用及依赖。- 共享宿主机内核，比传统 VM 更轻量。

优势：

- 启动快（秒级 vs 分钟级）- 占用资源少（无 Guest OS）- 部署与迁移方便- 更适合微服务架构

不足：

- 隔离性弱于 VM（共享内核）- 安全性略低（内核攻击面更大）- 不适合需要完整内核的应用

知识点总结

Docker=轻量，KVM/Xen=重量；容器高效但隔离弱。

7. 简述租户网络

租户网络是云平台为不同租户提供的独立虚拟网络环境。

核心功能：- 网络隔离（VLAN、VXLAN、GRE）- 虚拟路由器、虚拟交换机- 安全组（虚拟防火墙）

- DHCP、浮动 IP、负载均衡等服务

在 OpenStack 中由 Neutron 提供实现。

知识点总结

租户网络 = 独立虚拟网络环境，提供隔离、安全与灵活配置能力。

8. HDFS 是否支持并发访问和随机读写？

- **并发访问：支持** 多个客户端可同时读大文件。
- **随机写：不支持** 写入只能 append，不能修改已写数据。
- **随机读：部分支持** 可 seek，但不适用于频繁随机访问。HDFS 适用于一次写入，多次读取的大文件。

知识点总结

HDFS = 只追加写、支持顺序读、多客户端读、不支持随机写。

9. OpenStack 部署的服务及架构

常见服务：

- Keystone：认证
- Nova：计算

- Neutron：网络
- Glance：镜像
- Cinder：块存储
- Swift：对象存储
- Horizon：Web 控制台

架构： Controller 节点（Keystone/Nova-api/Neutron-server）+
Compute 节点（Nova-compute + OVS）+
Storage 节点（Cinder/Swift）

知识点总结

OpenStack = 计算+网络+存储+认证+镜像的云平台组件集合。

IaaS、PaaS、SaaS 定义与差别

- **IaaS**：提供虚拟机、存储、网络，如 AWS EC2、OpenStack。用户管理 OS 与应用。
- **PaaS**：提供开发运行环境，如 Heroku、Google App Engine。用户只需部署代码。
- **SaaS**：在线应用服务，如 Gmail、Salesforce。用户仅使用服务。

差别在资源管理粒度与用户参与程度。

知识点总结

IaaS=基础设施；PaaS=平台环境；SaaS=应用服务。

云体、云栈、云计算、云平台的定义及关系

- **云体**：具体的计算资源载体（机房、服务器）。
- **云计算**：把资源虚拟化、服务化的思想。
- **云平台**：云计算运行的软件系统（OpenStack、AWS）。
- **云栈**：云平台的技术组件集合。

关系：云体是物理基础 → 云计算是理念 → 云栈是技术组件 → 云平台是完整实现。

知识点总结

四者关系：物理→理念→组件→平台。

硬件辅助虚拟化、全虚拟化、半虚拟化

硬件辅助虚拟化（VT-x / AMD-V）
CPU 提供虚拟化指令，性能高，KVM 使用。

全虚拟化 模拟完整硬件，不修改 Guest OS，性能较低（如 QEMU）。

半虚拟化
Guest OS 需修改，借助 Hypercall 与 Hypervisor 交互，性能更好（如 Xen）。

知识点总结

性能：半虚拟化 > 硬件辅助 ≈ 全虚拟化（较慢）。

SDN 图及解释

架构同前：应用层/控制层/数据层，控制器集中管理，通过 OpenFlow 下发流表。

知识点总结

SDN = 可编程 + 控制分离。

轻量化虚拟化技术性能/资源/安全性比较

性能：接近原生 > 虚拟机 资源开销：最低（无 Guest OS） 安全性：弱于 VM（共享内核）

知识点总结

Docker 性能好、资源小，但安全性弱于 KVM。

结构化 / 半结构化 / 非结构化数据与适用存储系统

- **结构化**：关系模型，如 MySQL、PostgreSQL
- **半结构化**：JSON、XML，如 MongoDB、HBase
- **非结构化**：图片、视频，如 HDFS、对象存储（OSS/S3）

知识点总结

三类数据的应用场景与存储系统不同。

MapReduce 各阶段输出结果

阶段：

1. Map：输入 → KV
2. Shuffle：按 key 分组
3. Reduce：对组内 value 进行合并

每阶段输出 KV 格式不同。

知识点总结

Map=分散；Shuffle=排序分组；Reduce=聚合。

本地聚合（Combiner）的作用与与 Reducer 区别

- 本地聚合：在 Map 端对同 key 进行局部合并，减少网络传输。 - Reducer：全局聚合，处理最终结果。

知识点总结

Combiner 是可选、局部、优化；Reducer 是必须、全局。

写出 Map→Shuffle→Reduce 的三阶段 KV 示例

以词频统计为例：Map 输出：("hello",1)
Shuffle：("hello",[1,1,1])
Reduce：("hello",3)

知识点总结

典型三步：映射→分组→合并。

Hadoop 简述

Hadoop 是分布式存储（HDFS）+ 分布式计算（MapReduce）框架，用于大数据处理。

知识点总结

HDFS=大文件存储；MR=批处理计算。

HDFS 不适合大量小文件原因及处理策略

原因：- 每个文件块都有元数据，NameNode 内存压力大。- 并非为随机读取优化。

解决策略：- 小文件合并（HAR、SequenceFile、Parquet）- 对象存储（如 HBase、OSS、S3）

知识点总结

关键：NameNode 内存瓶颈。

元数据类型及 NameNode/DataNode 管理方式

元数据包括：- 文件名、权限- Block 列表- Block 位置

NameNode：管理所有元数据并持久化为 FsImage + EditLog
DataNode：存储数据块并向 NameNode 汇报心跳与块信息

知识点总结

NameNode=元数据；DataNode=数据块。

HMaster 与 HRegionServer 职责与交互

HMaster：- 管理 Region 分配- 元数据维护- 负载均衡

HRegionServer：- 管理 Region

- 提供读写服务
- memstore/flush/compaction

交互：Master 分配 Region，RegionServer 提供读写服务。

知识点总结

HMaster=管理；RegionServer=服务。

设计一个 HBase 表

示例：用户行为分析表 表名：user_action
RowKey：userID + timestamp 列族：info（action_type, device, ip）

知识点总结

HBase 表需根据查询模式设计 RowKey 与列族。