浅谈分布式（1~12）

> The open source, in-memory data store used by millions of developers as a database, cache streaming engine, and message broker.
>
> Redis 的初心.最初就是用来作为一个"消息中间件"的(消息队列). 分布式系统下的生产者消费者模型
> 当前很少会直接使用 Redis 作为消息中间件(业界有更多更专业的消息中间件使用)

一、Redis 概述与定位

Redis 是一种开源、内存型数据存储系统，广泛被开发者用作数据库、缓存、流处理引擎和消息代理。

最初定位为“消息中间件”（消息队列），但目前在生产环境中较少直接使用 Redis 作为消息队列，因为有更专业的替代方案。

Redis 本质是在内存中存储变量，适用于分布式系统，在单机环境下直接使用变量存储通常更优。

二、Redis vs MySQL

Redis 优势：

访问速度快，适用于对性能要求高的场景。

可作为数据库使用，读写性能远高于 MySQL。

Redis 劣势：

存储空间有限，不如 MySQL 可扩展。

更多互联网产品对性能要求并不极端，MySQL 仍能满足需求。

结合使用场景：

可采用 Redis + MySQL 组合方案，利用“二八原则”（20% 热点数据满足 80% 访问需求）。

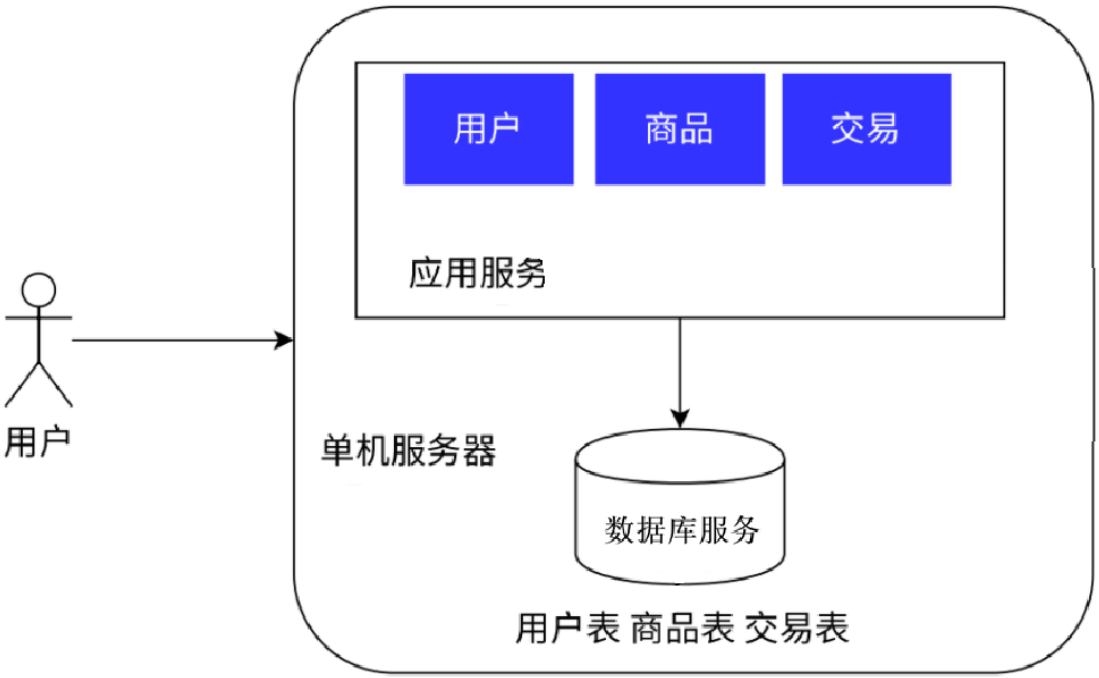
需注意数据同步问题，系统复杂度会提升。

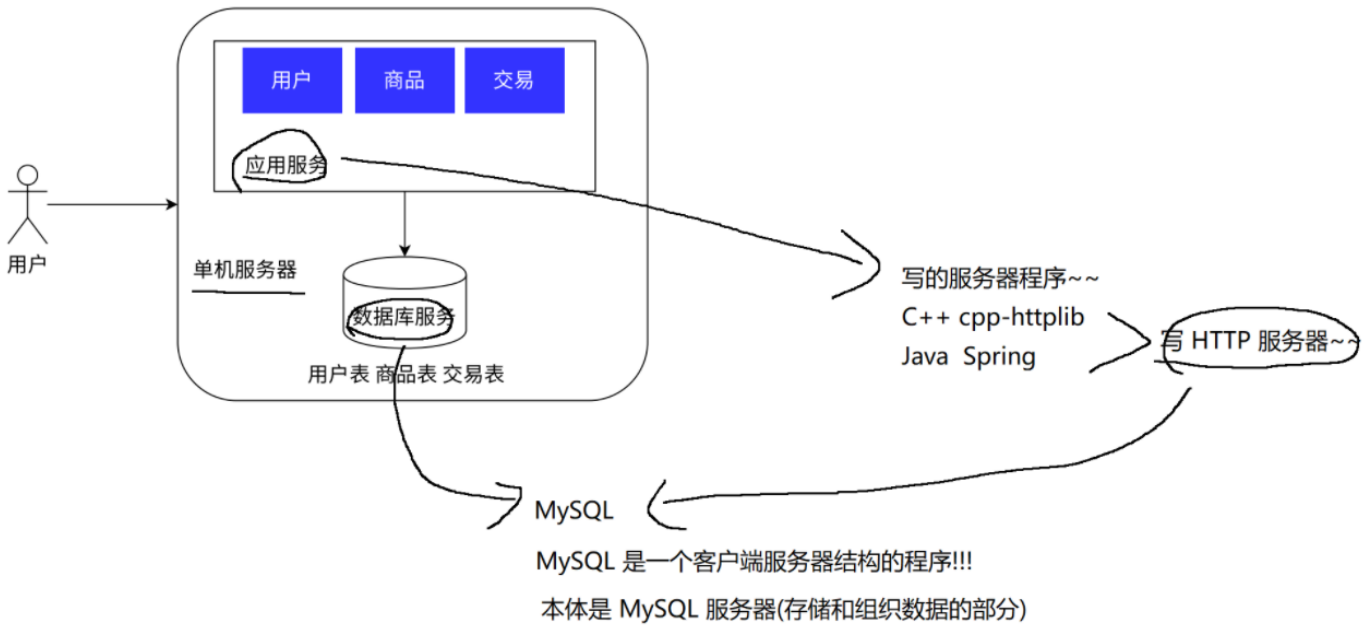
三、分布式系统概念

单机架构：

只有一台服务器，负责所有业务与数据存储。

许多公司的产品仍采用单机架构，现代单机性能已足够支撑高并发与大数据存储。

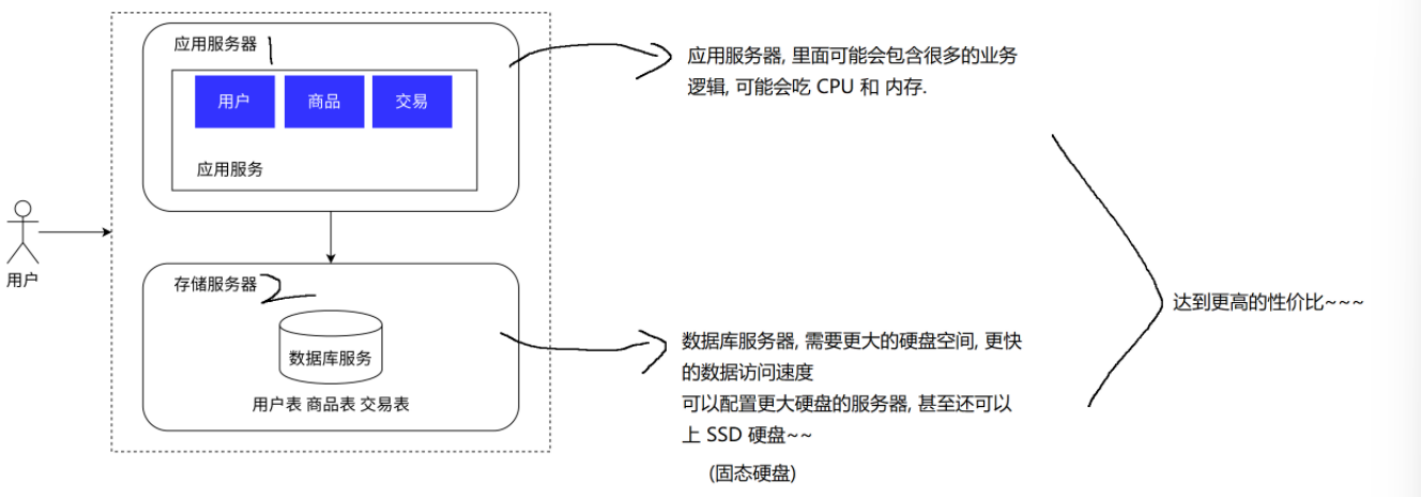




分布式系统出现原因：

当业务增长导致单机资源（CPU、内存、硬盘、网络等）不足时，需引入多台主机。

资源瓶颈可能导致请求处理变慢或出错。

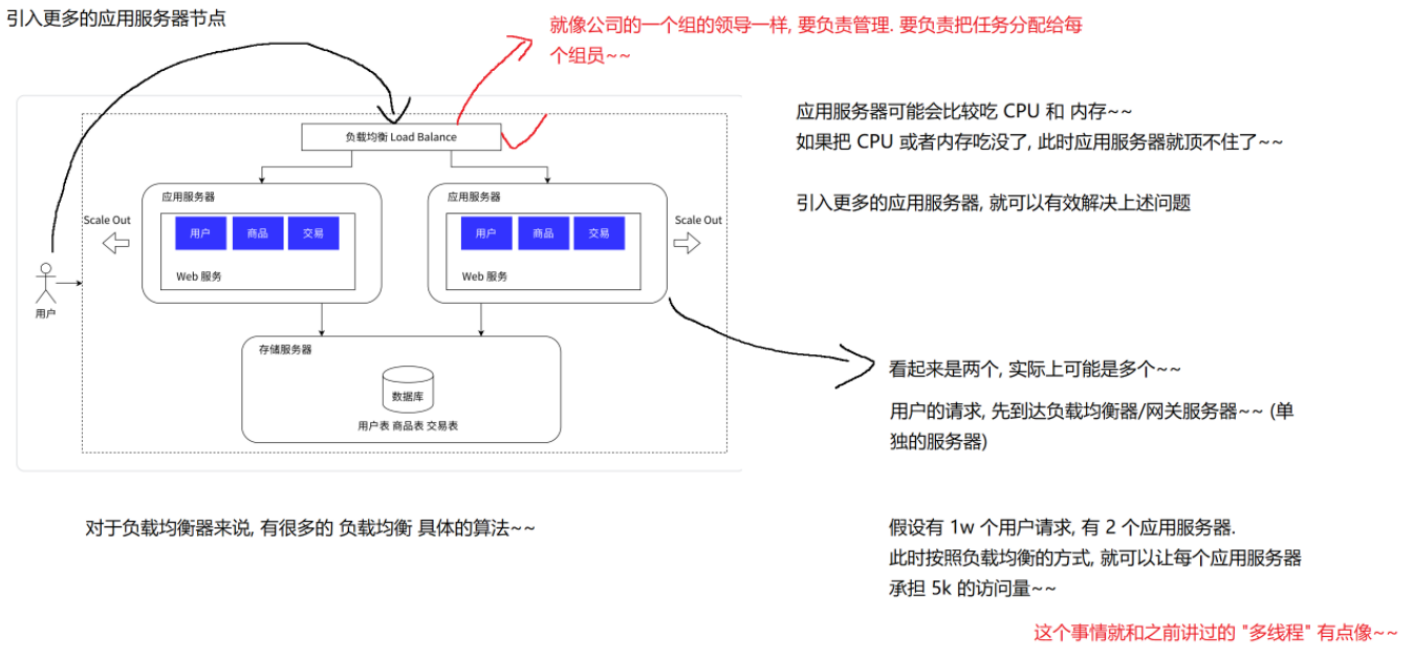


扩展方式：

开源：增加硬件资源（简单粗暴，但有上限）。

节流：软件优化（需定位性能瓶颈，对开发能力要求高）。

主板扩展能力有限，最终需引入多台主机，此时系统成为“分布式系统”。



分布式系统带来的影响

系统复杂度大幅提升。

出现 Bug 的概率增加，可能导致加班和年终奖风险。

需进行服务分离（如应用服务与数据库服务分离）。

需引入负载均衡机制，将请求分发到多个应用服务器节点。

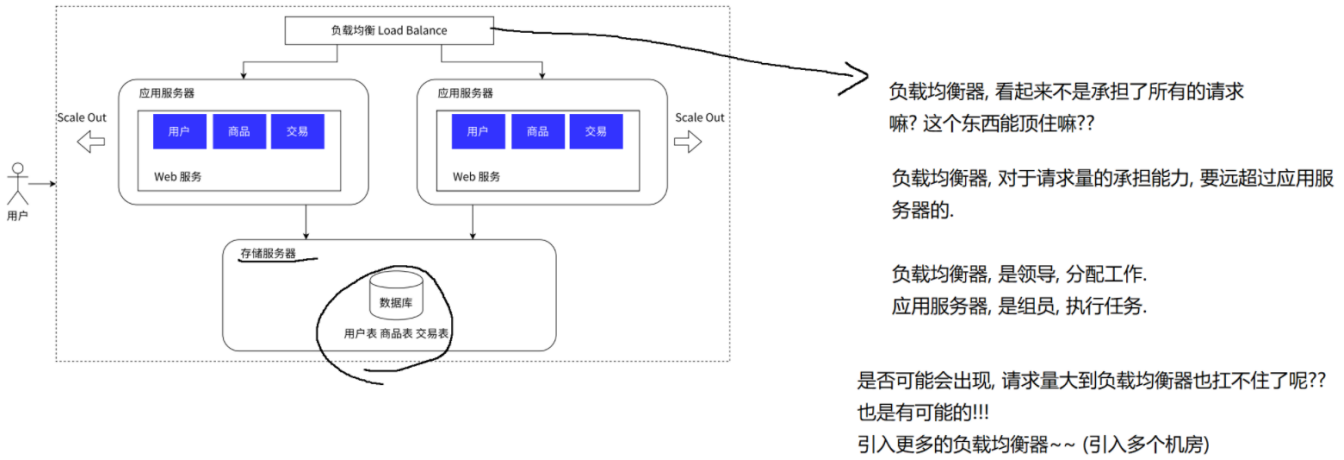
分布式系统是“无奈之举”，应在单机无法满足需求时再考虑。

引入多台主机需软件架构相应调整，如负载均衡、服务拆分、数据同步等。

系统设计需权衡性能、复杂度与维护成本，避免过度设计。

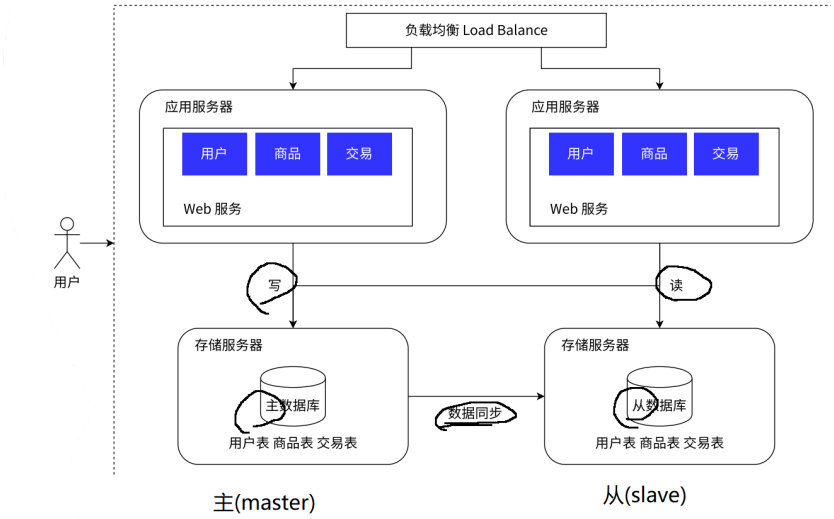
数据库主从架构

- 数据库采用**一主多从**架构：
  - **主服务器**：通常只有一个，负责写操作。
  - **从服务器**：可以有多个，负责读操作。
  - **负载均衡**：从数据库通过负载均衡的方式让应用服务器访问，分担读压力，提高系统并发能力。



分布式存储与数据扩展

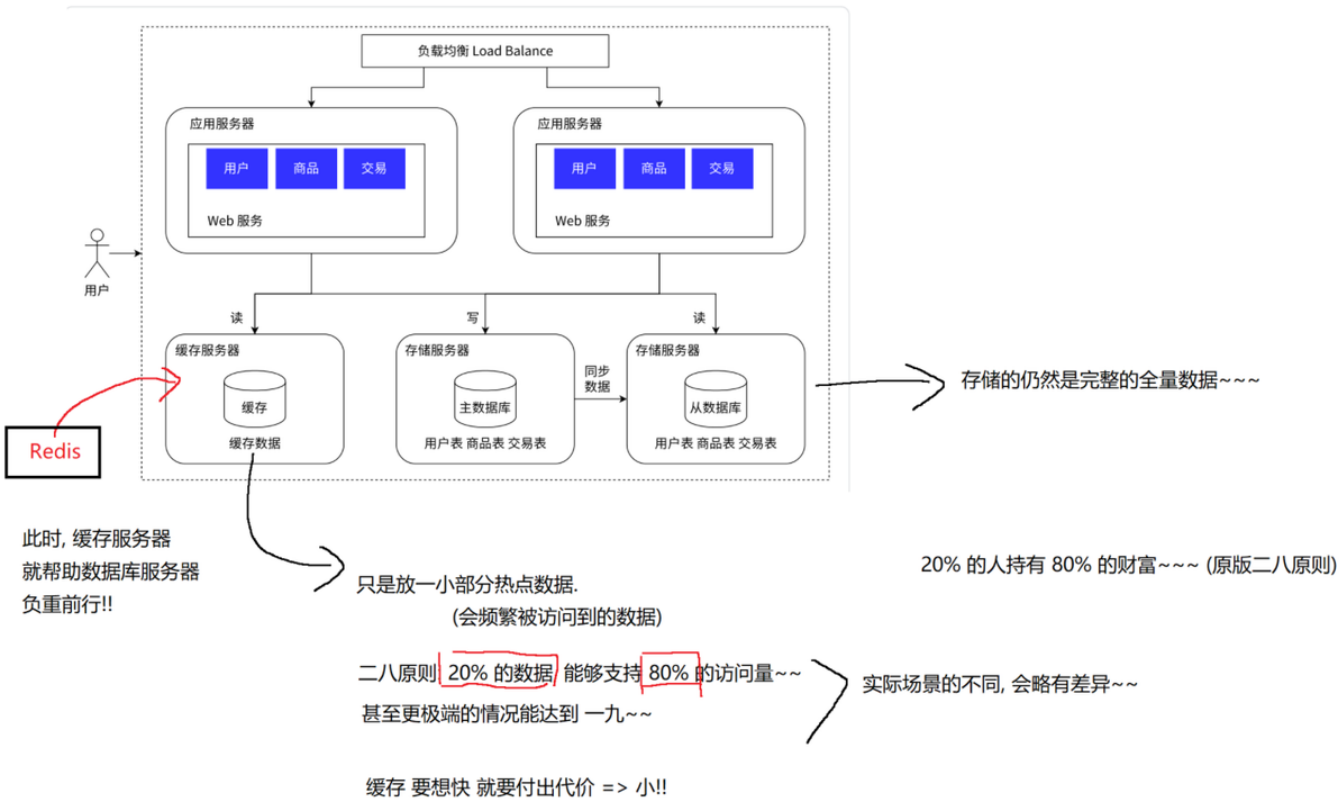
- 分布式系统不仅为应对高并发，也需应对**大数据存储问题**。
- 单台服务器的存储容量有限（即使可达几十TB），在某些场景（如短视频存储）仍可能无法满足需求。
- 解决方案：引入多台主机进行**分布式存储**，将数据分布在不同服务器上。



实际的应用场景中, 读的频率是比写 要高的!!!

主服务器一般是一个.从服务器可以有多个(一主多从)

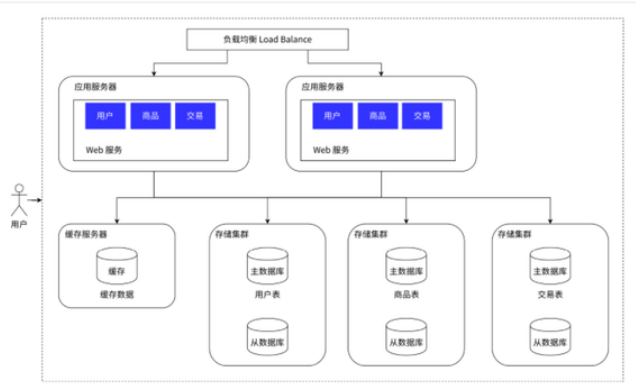
同时从数据库通过负载均衡的方式，让应用服务器进行访问



引入分布式系统，不光要能够去应对更高的请求量(并发量)，同时也要能应对更大的数据量~~

是否可能会出现，一台服务器已经存不下数据了呢??

当然会存在!!!虽然一个服务器，存储的数据量可以达到几十个TB，即使如此也可能会存不下~~ 短视频一台主机存不下，就需要多台主机来存储



针对数据库进行进一步的拆分.
分库分表~~

本来一个数据库服务器. 这个数据库服务器上有多个数据库 (指的是逻辑上的数据集合, create database 创建的那个东西)

现在就可以引入多个数据库服务器. 每个数据库服务器存储一个或者一部分数据库~~

如果某个表特别大, 大到一台主机存不下, 也可以针对表进行拆分~~

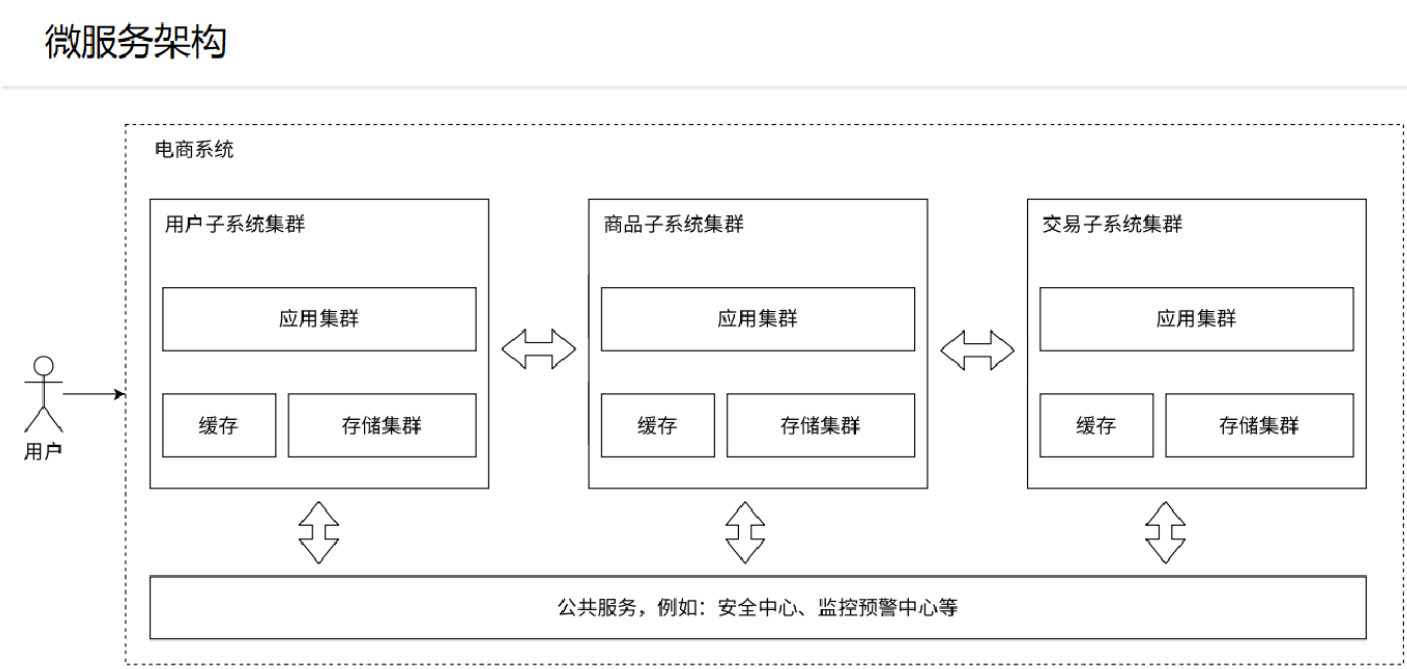
具体分库分表如何实践? 还是要结合实际的业务场景来展开~~

业务 至关重要!!!
技术 只是给业务提供支持的!!!

业务 决定了 技术 !!

四、微服务架构

- **背景**：传统单体应用服务器业务复杂、代码臃肿，难以维护。
- **微服务定义**：将一个复杂的服务器按功能拆分为多个**功能单一、独立部署的小型服务**。



之前应用服务器，一个服务器程序里面做了很多的业务.这就可能会导致这一个服务器的代码变的越来越复杂

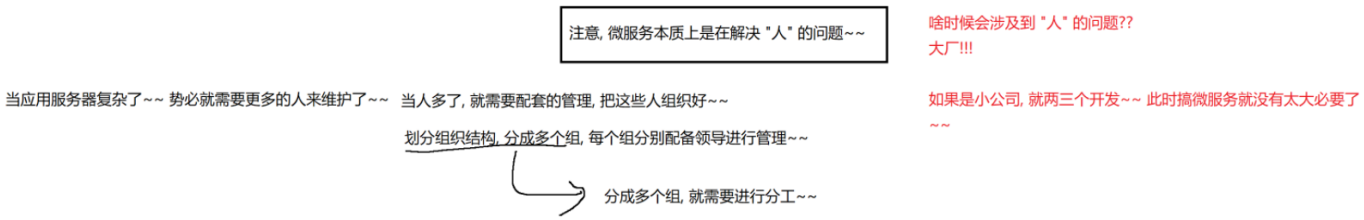
为了更方便于代码的维护，就可以把这样的一个复杂的服务器，拆分成更多的，功能更单一，但是更小的服务器

(微服务) ||

服务器的种类和数量就增加了

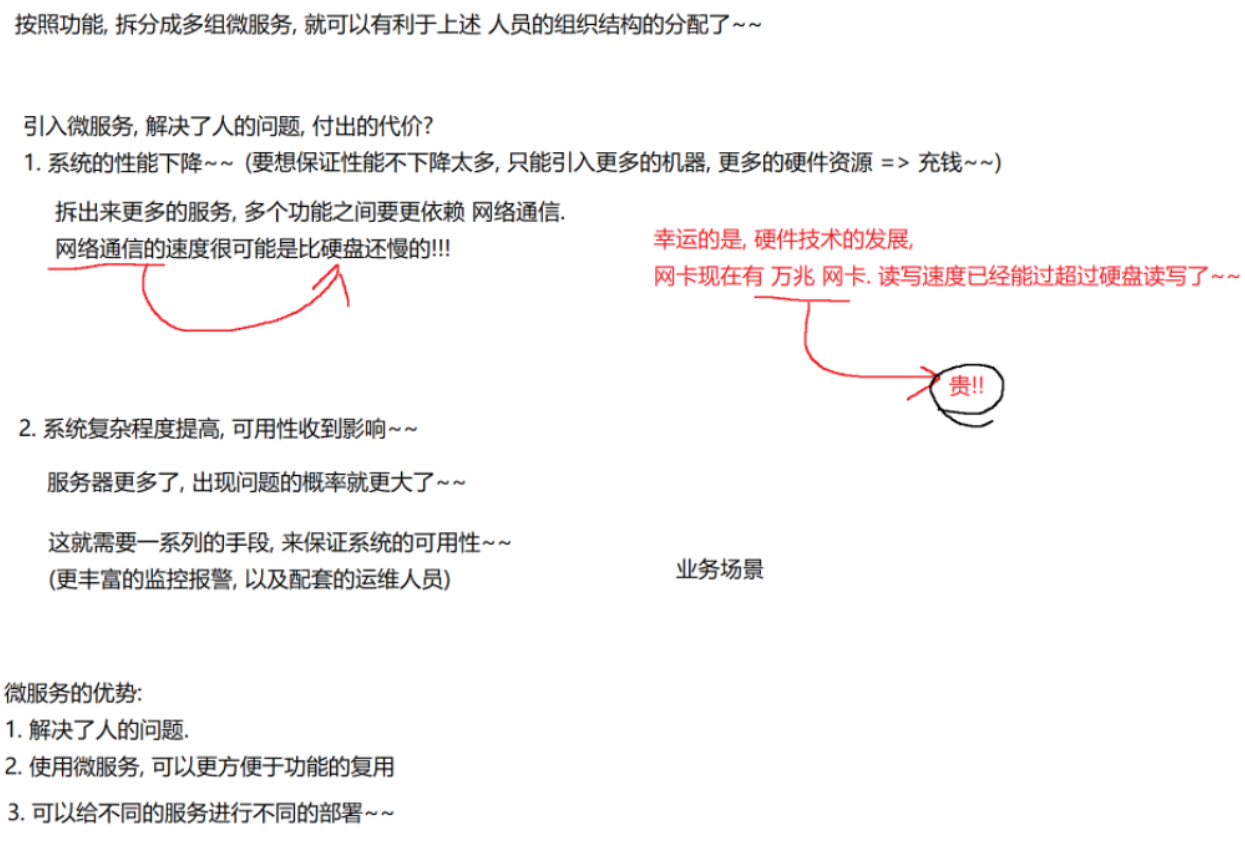
微服务的优势

1. **解决人的问题**：便于团队按功能模块分工，提升开发与维护效率。
2. **功能复用**：微服务可被多个系统或模块复用。
3. **独立部署**：不同服务可独立部署、升级与扩展，提高灵活性。



微服务带来的挑战与代价

1. **系统性能下降**：
   - 服务拆分后，服务间依赖**网络通信**，而网络通信速度可能成为瓶颈。
   - 图片中标注（位置：[160, 160, 298, 192]）提示：“网络通信的速度很可能是比硬盘还慢的！”
   - **缓解方式**：使用更高速网络设备（如万兆网卡），或增加硬件资源（“充钱”）。
2. **系统复杂度提高，可用性受影响**：
   - 服务数量增多，故障概率增加。
   - 需要加强**监控、报警与运维体系**，保障系统可用性。



业务场景与组织结构适配

- 微服务架构有利于**按功能划分团队**，每个团队负责一个或一组微服务，提升组织协作效率。
- 架构调整需与**团队结构、业务发展**相匹配。

总结建议

- 微服务架构是**为解决复杂系统维护与团队协作问题**而引入的，并非适用于所有场景。
- 引入微服务需权衡**性能、复杂度、运维成本与团队能力**。
- 在高速网络硬件发展的背景下，网络通信瓶颈逐渐缓解，但仍需在架构设计中充分考虑通信开销与可用性保障。

五、概念补充

软件架构基础

- **应用 (Application) / 系统 (System)**：指代一个或一组服务器程序，是提供服务的整体。
- **模块 (Module) / 组件 (Component)**：应用内部的细分单位，指代其中每一个独立的功能部分。
- **中间件 (Middleware)**：与具体业务无关的通用服务。例如：**数据库、缓存、消息队列**等。

多机协作模式

- **分布式 (Distributed)**：侧重于**物理层面**，引入多个物理主机协同完成一系列工作。
- **集群 (Cluster)**：侧重于**逻辑层面**，也是引入多个主机协作，但对外通常表现为一个整体。
- **主 (Master) / 从 (Slave)**：分布式系统中常见的结构。其中一个节点为主，其余为从，从节点的数据通常从主节点同步而来。

系统评价指标

1. 可用性 (Availability)

系统的“第一要务”，衡量系统能够正常提供服务的时间比例。

- **计算公式**：$可用性 = \frac{系统整体可用时间}{总时间}$
- **常见标准**：
  - 4个9：99.99% 可用。
  - 5个9：99.999% 可用（极高标准）。

2. 性能指标

- **响应时长 (Response Time, RT)**：衡量服务器处理单个请求的速度，数值**越小越好**。
- **吞吐 (Throughput) vs 并发 (Concurrent)**：
  - 衡量系统处理请求的能力。
  - 是衡量服务器性能高低的关键方式。

六、 分布式系统演化小结

这份资料总结了系统为了应对更高并发和更大规模数据，在硬件资源和架构设计上的演进过程：

---

1. 基础架构阶段

- **单机架构**：应用程序与数据库服务器部署在同一台机器上。
- **数据库和应用分离**：将应用程序和数据库分别部署在不同的主机上，初步缓解性能压力。

2. 应用层扩展

- **引入负载均衡与集群**：
  - 通过负载均衡器将请求均匀分发给应用服务器集群。
  - **核心价值**：提高可用性。当集群中某个主机宕机时，其他主机仍可承担服务。

3. 数据库层扩展

- **读写分离（主从结构）**：
  - **主节点 (Master)**：负责写数据。
  - **从节点 (Slave)**：负责读数据，数据从主节点同步而来。
- **分库分表**：为了进一步扩展存储空间和处理能力，对数据库进行水平或垂直拆分。

4. 性能与业务优化

- **引入缓存（冷热数据分离）**：
  - 利用**二八原则**（20% 的热点数据承担 80% 的访问量），提升系统处理能力。
  - **典型工具**：Redis 常在分布式系统中扮演此类角色。
  - **挑战**：需要解决数据库与缓存之间的数据一致性问题。
- **微服务架构**：从业务功能角度出发，将应用服务器拆分为功能更单一、更简单的微小服务。

核心结论

- **技术服务于业务**：真实的演化过程与业务发展密切相关，业务才是第一位的。
- **分布式本质**：所谓的分布式系统，其核心思路就是通过技术手段**引入更多的硬件资源**来支撑业务。