# Redis的背景知识（13~18）

> Redis是一个在内存中存储数据的中间件.用于作为数据库，用于作为数据缓存.在分布式系统中能够大展拳脚~

## 一、 分布式系统演化过程小结

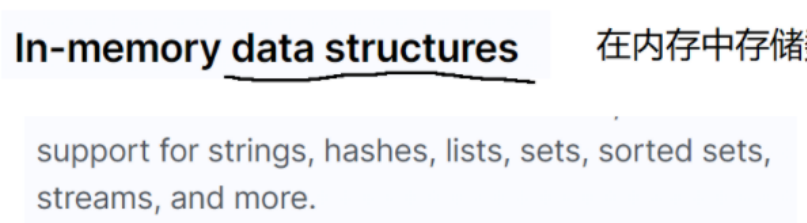
1. **单机架构**：最基础阶段，应用程序与数据库服务器部署在同一台机器上。
2. **数据库和应用分离**：将应用程序和数据库服务器分别部署在不同的主机上。
3. **引入负载均衡（应用服务器集群）**：
   - 通过负载均衡器将请求均匀分发给集群中的每个应用服务器。
   - **优势**：当集群中某个主机宕机时，其他主机仍可承担服务，从而提高了整个系统的**可用性**。
4. **引入读写分离（数据库主从结构）**：
   - 一个节点作为**主节点**负责写数据，其余 N 个节点作为**从节点**负责读数据。
   - 主节点负责将修改后的数据同步给从节点。
5. **引入缓存（冷热数据分离）**：
   - 基于“二八原则”，进一步提升服务器处理请求的能力。Redis 通常在此阶段扮演缓存角色。
   - **带来的问题**：需要解决数据库和缓存之间的数据一致性问题。
6. **引入分库分表**：使数据库能够进一步扩展存储空间。
7. **引入微服务**：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、更小的服务器。

**核心结论**：真实的演化过程与业务发展密切相关。业务是更重要的，技术只是支撑。所谓的分布式系统，本质就是想办法引入更多的硬件资源。

## 二、 Redis 的特性与优点

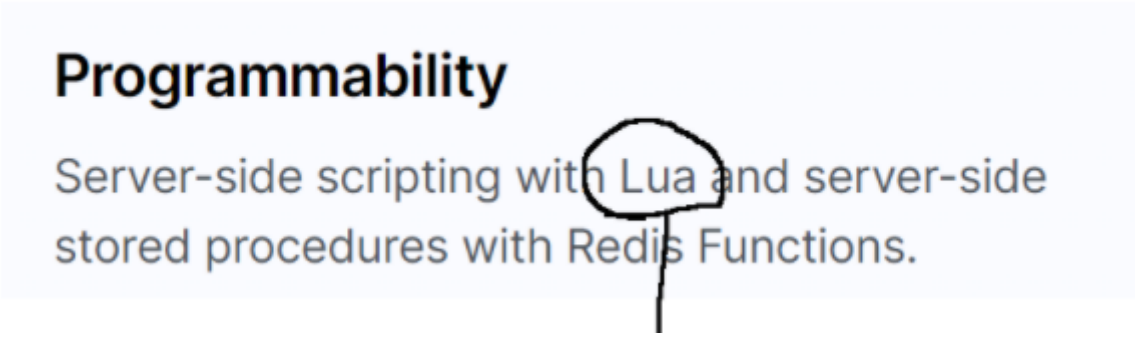
Redis 是一个在内存中存储数据的中间件，具有以下核心特性：

---



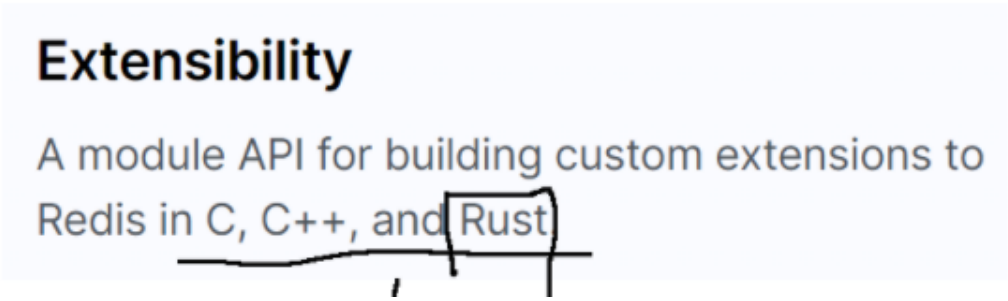
- **内存数据结构 (In-memory data structures)**：
  - 不同于 MySQL 的“表”结构（关系型），Redis 采用“键值对”方式（非关系型）。
  - **Key** 永远是 String 类型；**Value** 可以是 String, Hash, List, Set, Sorted Set, Stream 等。

---



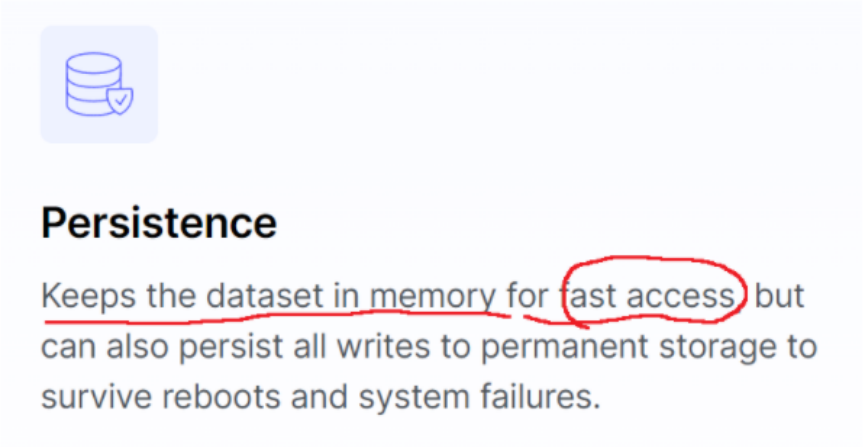

- **可编程性 (Programmability)**：支持通过简单的交互式命令操作，也可以通过 **Lua 脚本**批量执行带有逻辑的操作。
- 针对Redis的操作，可以直接通过简单的交互式命令进行操作.也可以通过一些脚本的方式，批量执行一些操作(可以带有一些逻辑)

---



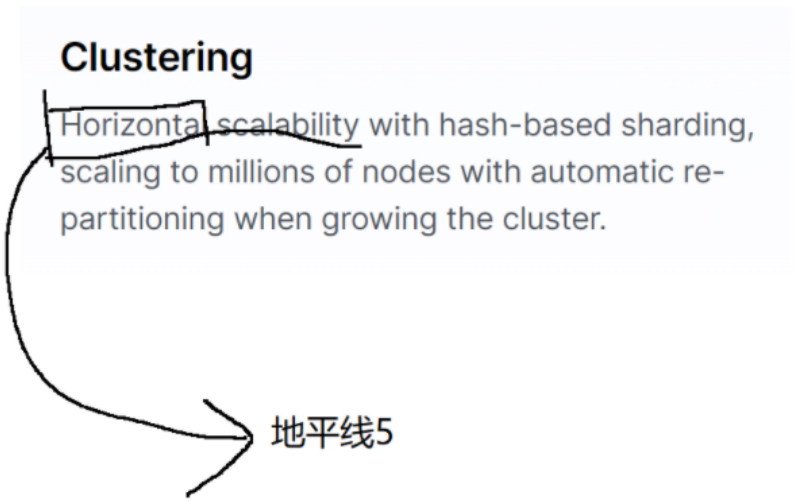

- **可扩展性 (Extensibility)**：提供了一组 API（支持 C, C++, Rust），允许开发者通过编写动态链接库（如 Windows 的 .dll 或 Linux 的 .so）来扩展更多的数据结构和命令。
- 程序员自己去扩展 Redis的功能.比如，Redis自身已经提供了很多的数据结构和命令.通过扩展，让Redis支持更多的数据结构以及支持更多的命令。

---



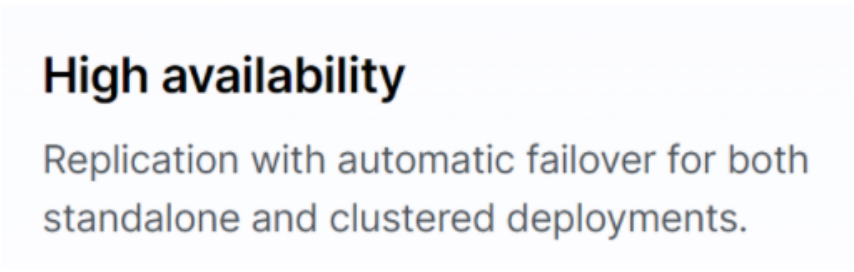

- **持久化 (Persistence)**：虽然数据存在易失的内存中，但 Redis 会将数据存储在磁盘上（内存为主，磁盘为辅）。重启时会加载磁盘备份数据，恢复到重启前的状态。
- 硬盘相当于对内存的数据备份了一下)如果Redis重启了，就会在重启时加载硬盘中的备份数据使Redis的内存恢复到重启前的状态.

---




- **集群 (Clustering)**：支持水平扩展（基于哈希的分片），类似于数据库的分库分表。通过引入多台主机部署多个节点，突破单机内存限制。

---




- **高可用 (High Availability)**：自身支持“主从”结构，从节点作为主节点的备份（冗余），并支持自动故障转移。

---

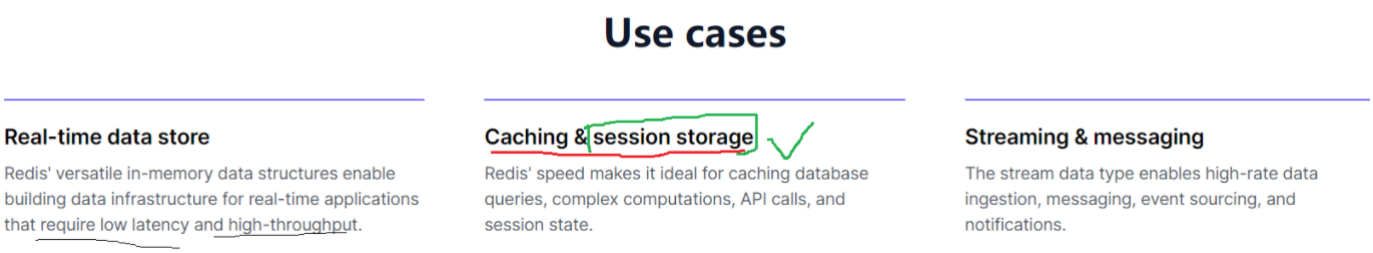
## 三、 为什么 Redis 速度快?

图片总结了 Redis 保持高性能的五个原因：

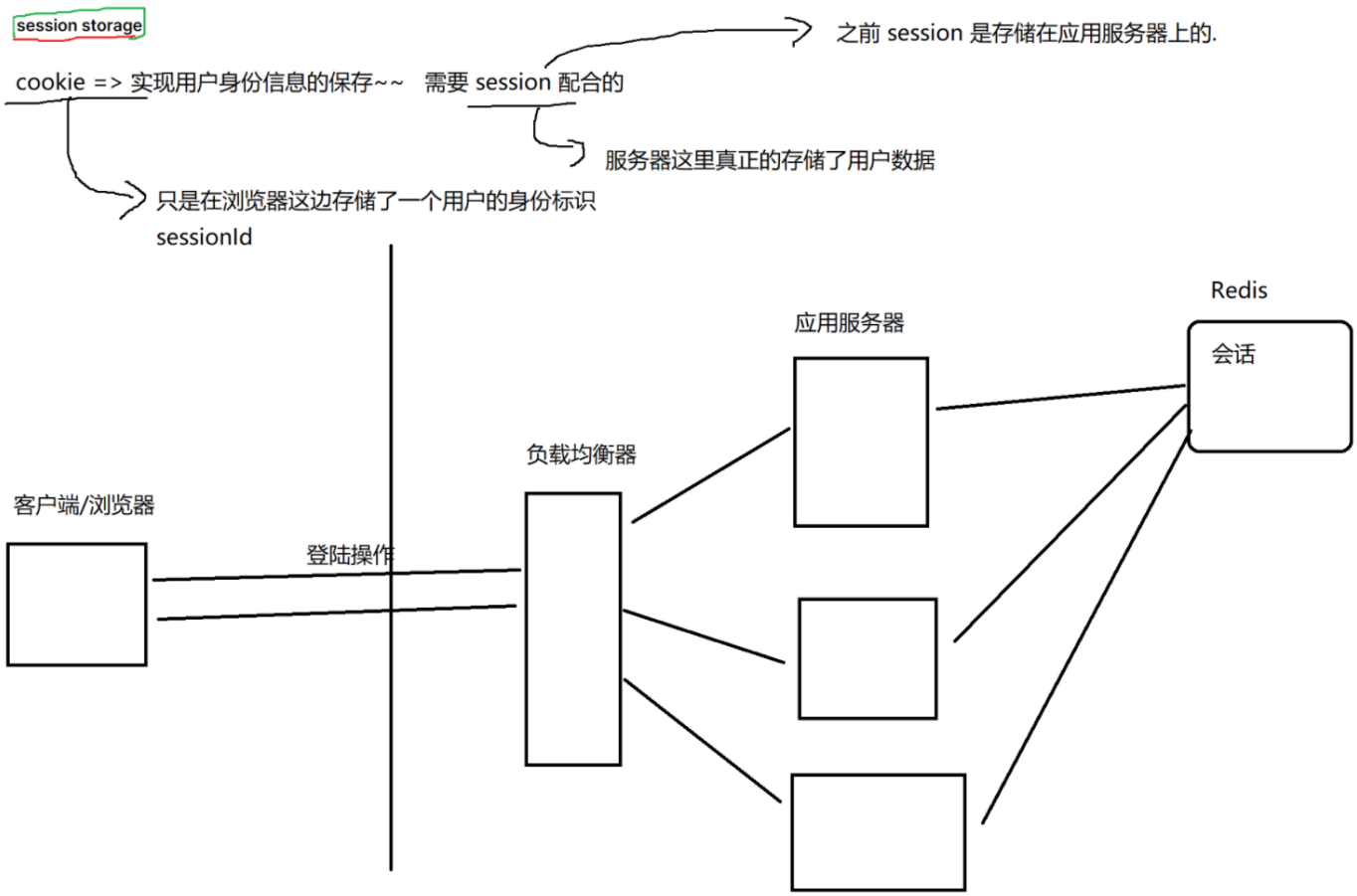
1. **内存存储**：数据在内存中，访问速度远快于访问硬盘数据库。
2. **逻辑简单**：核心功能都是比较简单的操作内存数据结构的逻辑。
3. **IO 多路复用 (epoll)**：在网络层面上，使用一个线程管理多个 socket。
4. **单线程模型**：减少了不必要的线程竞争开销。虽然多线程能利用多核 CPU，但 Redis 的核心任务是内存操作，不吃 CPU。
5. **C 语言开发**：虽有此说法，但作者指出 MySQL 也是 C 开发的，因此这不一定是其“快”的决定性理由。

---

## 四、 Redis 的应用场景




- **实时数据存储（作为数据库）**：
  - 用于对性能要求极高的场景（如广告搜索），将所有需要检索的数据存储在内存中。
  - **注意**：这需要消耗大量的硬件资源（充值），且此处 Redis 存的是全量数据，不可丢失。
- **缓存与会话存储 (Caching & Session Storage)**：
  - **缓存**：将 MySQL 中的热点数据拎出来存在 Redis 中（二八原则）。即使 Redis 数据丢失，也可以从 MySQL 重新加载。
  - **分布式会话**：传统 session 存放在应用服务器上。在多服务器环境下，通过 Redis 集中存储会话数据，解决负载均衡后的会话丢失问题（应用重启会话不丢失）。




**如何解决上述问题?**

1. 想办法让负载均衡器，把同一个用户的请求始终打到同一个机器上(不能轮询了，而是要通过userld之类的方式来分配机器)
2. 把会话数据单独拎出来，放到一组独立的机器上存储(Redis) (应用程序重启了，会话不丢失)

- **消息队列 (Streaming & Messaging)**：
  - 基于 Redis 可以实现网络版本的生产者-消费者模型。
  - **优势**：解耦、削峰填谷。如果系统对消息队列功能依赖不多且不想引入额外依赖（如 Kafka, RabbitMQ），Redis 是一个好的选择。

---

## 五、Redis的劣势

> 存储大规模数据

- **学习建议**：

a. 不仅要学习技术，更要通过业务项目来加深理解。

b. 读万卷书，行万里路，业务场景不同，分布式系统的实践方式差异巨大。

c. 通过阅读官方文档、完成高质量作业和博客来巩固知识。