课程安排

课程计划

1. Redis 基础
2. Redis 高级
3. Redis 集群
   - 主从复制
   - 哨兵模式
   - 集群
4. 企业级解决方案

- 主从复制简介
- 主从复制工作流程
- 主从复制常见问题

主从复制简介

你的“Redis”是否高可用

单机redis的风险与问题

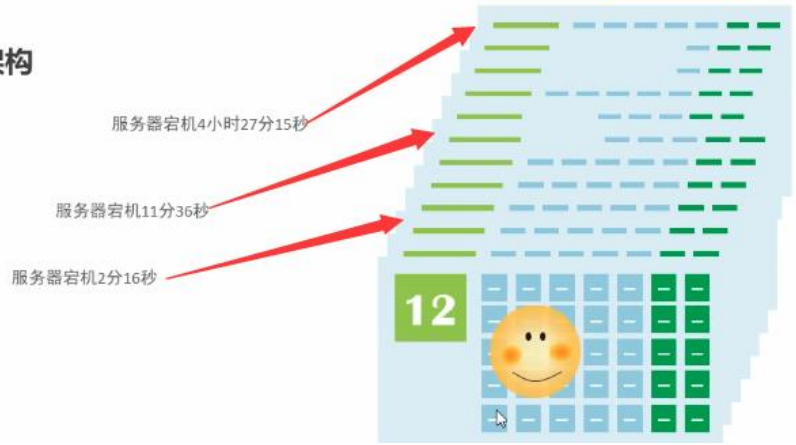
- 问题1.机器故障
  - 现象：硬盘故障、系统崩溃
  - 本质：数据丢失，很可能对业务造成灾难性打击
  - 结论：基本上会放弃使用redis.
- 问题2.容量瓶颈
  - 现象：内存不足，从16G升级到64G，从64G升级到128G，无限升级内存
  - 本质：穷，硬件条件跟不上
  - 结论：放弃使用redis
- 结论：

  为了避免单点Redis服务器故障，准备多台服务器，互相连通。将数据复制多个副本保存在不同的服务器上，连接在一起，并保证数据是同步的。即使有其中一台服务器宕机，其他服务器依然可以继续提供服务，实现Redis的高可用，同时实现数据冗余备份。

主从复制简介

互联网“三高”架构

- 高并发
- 高性能
- 高可用



4小时27分15秒+11分36秒+2分16秒 = 4小时41分7秒 = 866467秒

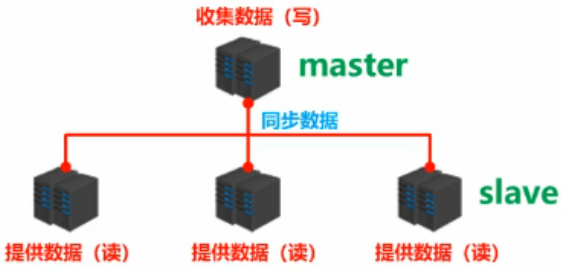
1年 = 365*24*60*60 = 31536000

可用性 = $\frac{31536000 - 866467}{31536000} * 100\% = 97.252\%$

业界可用性目标5个9，即99.999% ，即服务器年宕机时长低于315秒，约5.25分钟

多台服务器连接方案

- 提供数据方：master
  主服务器，主节点，主库
  主客户端
- 接收数据方：slave
  从服务器，从节点，从库
  从客户端
- 需要解决的问题：
  数据同步
- 核心工作：
  master的数据复制到slave中



主从复制

主从复制即将master中的数据即时、有效的复制到slave中

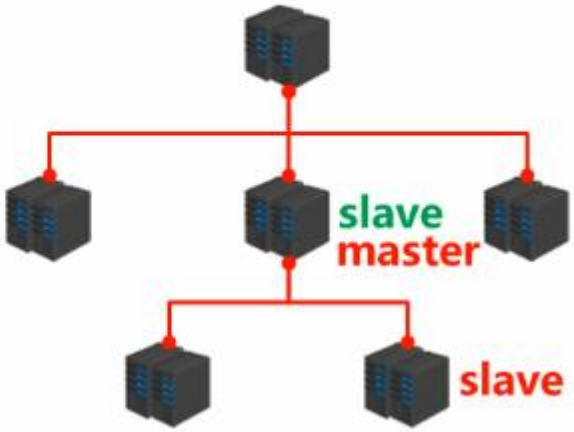
特征：一个master可以拥有多个slave，一个slave只对应一个master

职责：

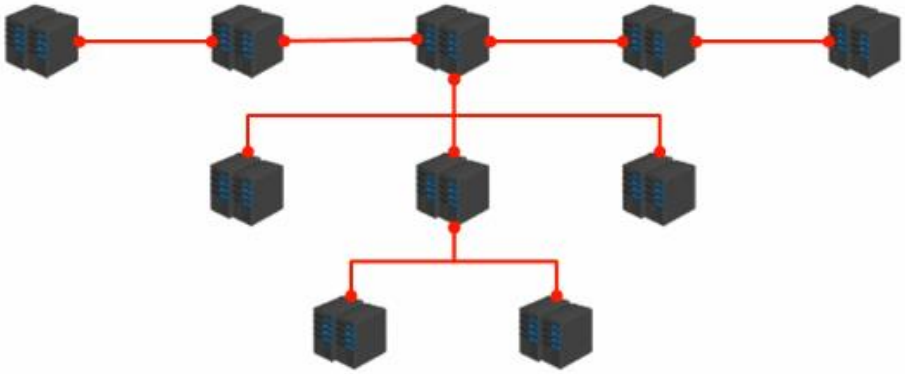
- master:
  - 写数据
  - 执行写操作时，将出现变化的数据自动同步到slave
  - 读数据（可忽略）
- slave:
  - 读数据
  - 写数据（禁止）

高可用集群

高可用集群



高可用集群



主从复制的作用

- 读写分离：master写、slave读，提高服务器的读写负载能力
- 负载均衡：基于主从结构，配合读写分离，由slave分担master负载，并根据需求的变化，改变slave的数量，通过多个从节点分担数据读取负载，大大提高Redis服务器并发量与数据吞吐量
- 故障恢复：当master出现问题时，由slave提供服务，实现快速的故障恢复
- 数据冗余：实现数据热备份，是持久化之外的一种数据冗余方式
- 高可用基石：基于主从复制，构建哨兵模式与集群，实现Redis的高可用方案

目录 Contents

主从复制工作流程

总述

- 主从复制过程大体可以分为3个阶段
  - 建立连接阶段（即准备阶段）
  - 数据同步阶段
  - 命令传播阶段

主从复制工作流程

阶段一：建立连接阶段

- 建立slave到master的连接，使master能够识别slave，并保存slave端口号

主从复制工作流程

建立连接阶段工作流程

步骤1：设置master的地址和端口，保存master信息
步骤2：建立socket连接
步骤3：发送ping命令（定时器任务）
步骤4：身份验证
步骤5：发送slave端口信息
至此，主从连接成功！

状态：
slave：
保存master的地址与端口
master：
保存slave的端口


master


slave

①发送指令：**slaveof ip port**
②接收到指令，响应对方
③保存master的IP与端口
masterhost
masterport
④根据保存的信息创建连接master的**socket**
⑤周期性发送命令：**ping**
⑥响应**pong**
⑦发送指令：**auth password**
⑧验证授权
⑨发送指令：**replconf listening-port <port-number>**
⑩保存slave的端口号

主从复制工作流程

主从连接（slave连接master）

- 方式一：客户端发送命令

```
slaveof <masterip> <masterport>
```

- 方式二：启动服务器参数

```
redis-server -slaveof <masterip> <masterport>
```

- 方式三：服务器配置

```
slaveof <masterip> <masterport>
```

- slave系统信息
  - master_link_down_since_seconds
  - masterhost
  - masterport
- master系统信息
  - slave_listening_port(多个)

主从复制工作流程

主从断开连接

- 客户端发送命令

```
slaveof no one
```

主从复制工作流程

授权访问

- master配置文件设置密码

```
requirepass <password>
```

- master客户端发送命令设置密码

```
config set requirepass <password>
config get requirepass
```

- slave客户端发送命令设置密码

```
auth <password>
```

- slave配置文件设置密码

```
masterauth <password>
```

- 启动客户端设置密码

```
redis-cli -a <password>
```

主从复制工作流程

阶段二：数据同步阶段工作流程

- 在slave初次连接master后，复制master中的所有数据到slave
- 将slave的数据库状态更新成master当前的数据库状态

主从复制工作流程

数据同步阶段工作流程

步骤1：请求同步数据
步骤2：创建RDB同步数据
步骤3：恢复RDB同步数据

主从复制工作流程

数据同步阶段工作流程

之后 master 会记录 slaver 部分复制的时候从复制缓冲区拿到了第几条指令，那么下次复制的时候直接从这条指令的下一条开始即可。

主从复制工作流程

数据同步阶段master说明

1. 如果master数据量巨大，数据同步阶段应避开流量高峰期，避免造成master阻塞，影响业务正常执行
2. 复制缓冲区大小设定不合理，会导致数据溢出。如进行全量复制周期太长，进行部分复制时发现数据已经存在丢失的情况，必须进行第二次全量复制，致使slave陷入死循环状态。

```
repl-backlog-size 1mb
```

3. master单机内存占用主机内存的比例不应过大，建议使用50%-70%的内存，留下30%-50%的内存用于执行bgsave命令和创建复制缓冲区

主从复制工作流程

数据同步阶段slave说明

1. 为避免slave进行全量复制、部分复制时服务器响应阻塞或数据不同步，建议关闭此期间的对外服务

```
slave-serve-stale-data yes|no
```

2. 数据同步阶段，master发送给slave信息可以理解master是slave的一个客户端，主动向slave发送命令
3. 多个slave同时对master请求数据同步，master发送的RDB文件增多，会对带宽造成巨大冲击，如果master带宽不足，因此数据同步需要根据业务需求，适量错峰
4. slave过多时，建议调整拓扑结构，由一主多从结构变为树状结构，中间的节点既是master，也是slave。注意使用树状结构时，由于层级深度，导致深度越高的slave与最顶层master间数据同步延迟较大，数据一致性变差，应谨慎选择

主从复制工作流程

阶段三：命令传播阶段

- 当master数据库状态被修改后，导致主从服务器数据库状态不一致，此时需要让主从数据同步到一致的状态，同步的动作称为命令传播
- master将接收到的数据变更命令发送给slave，slave接收命令后执行命令

主从复制工作流程

命令传播阶段的部分复制

- 命令传播阶段出现了断网现象
  - 网络闪断闪连　　忽略
  - 短时间网络中断　　部分复制
  - 长时间网络中断　　全量复制

- 部分复制的三个核心要素
  - 服务器的运行id（run id）
  - 主服务器的复制积压缓冲区
  - 主从服务器的复制偏移量

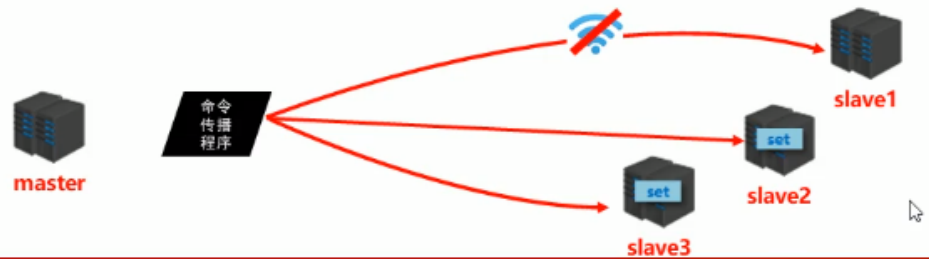
主从复制工作流程

服务器运行ID（runid）

- 概念：服务器运行ID是每一台服务器每次运行的身份识别码，一台服务器多次运行可以生成多个运行id
- 组成：运行id由40位字符组成，是一个随机的十六进制字符
  例如：fdc9ff13b9bbaab28db42b3d50f852bb5e3fcdce
- 作用：运行id被用于在服务器间进行传输，识别身份
  如果想两次操作均对同一台服务器进行，必须每次操作携带对应的运行id，用于对方识别
- 实现方式：运行id在每台服务器启动时自动生成的，master在首次连接slave时，会将自己的运行ID发送给slave，slave保存此ID，通过info Server命令，可以查看节点的runid

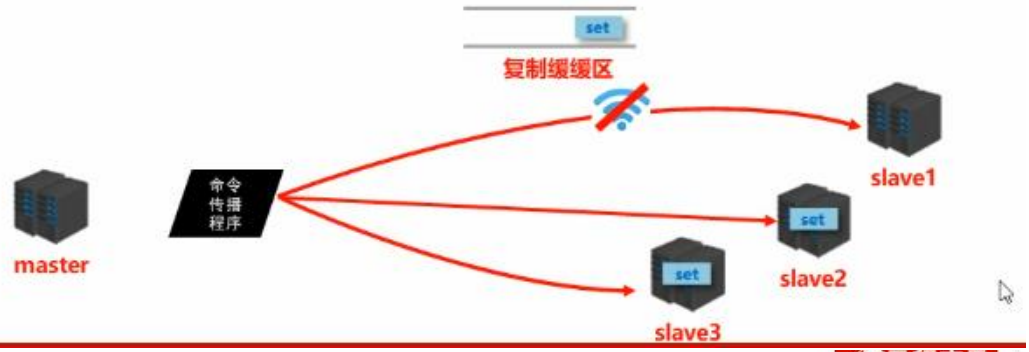
主从复制工作流程

复制缓冲区



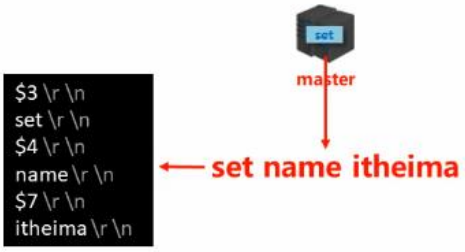
复制缓冲区

- 概念：复制缓冲区，又名复制积压缓冲区，是一个先进先出（FIFO）的队列，用于存储服务器执行过的命令，每次传播命令，master都会将传播的命令记录下来，并存储在复制缓冲区



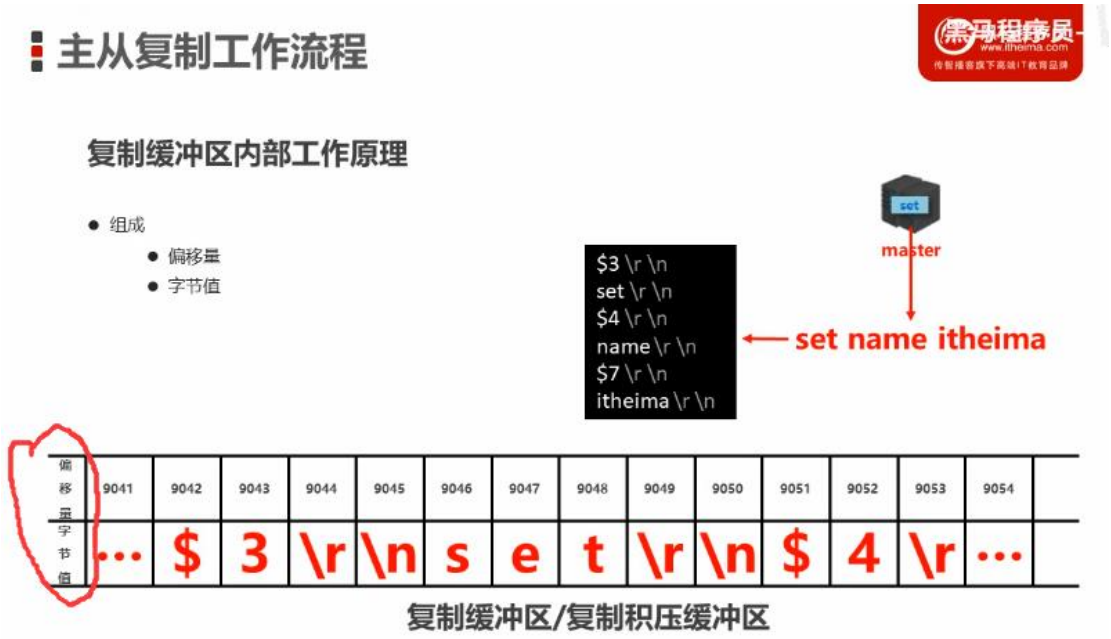
主从复制工作流程

复制缓冲区内部工作原理



字节值	…	$	3	\r	\n	s	e	t	\r	\n	$	4	\r	…	

复制缓冲区/复制积压缓冲区



master 和 slaver 各自都要记住自己的 offset，那么在下次遇到 slaver 丢包的时候，master 能对比自己的 offset 和 slaver 的 offset，就可以知道丢了哪些数据了。

主从复制工作流程

复制缓冲区

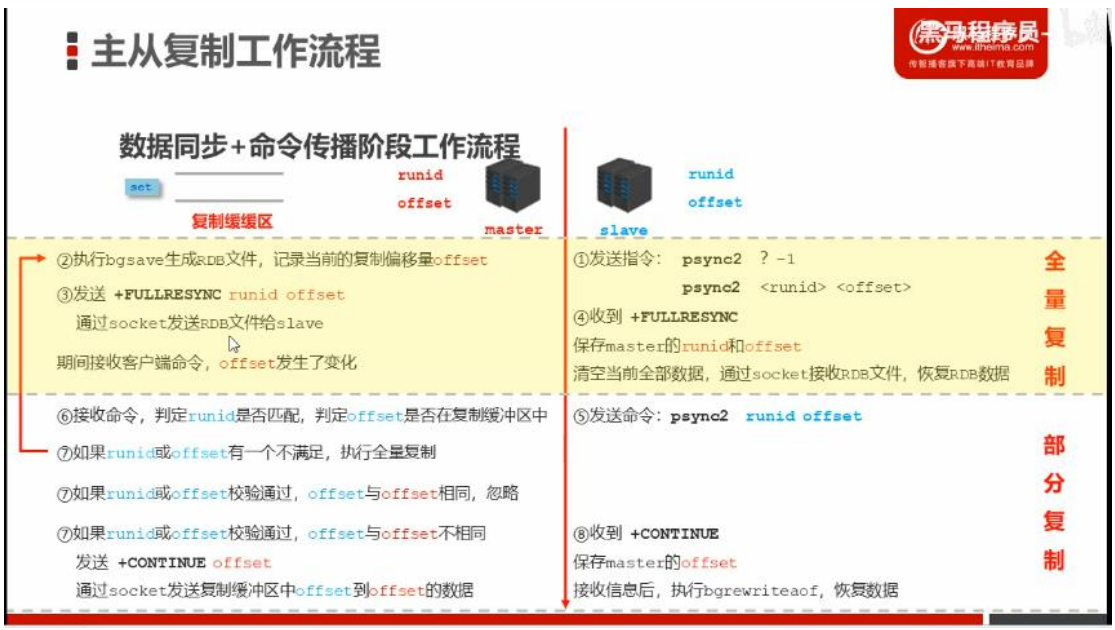
- 概念：复制缓冲区，又名复制积压缓冲区，是一个先进先出（FIFO）的队列，用于存储服务器执行过的命令，每次传播命令，master都会将传播的命令记录下来，并存储在复制缓冲区
  - 复制缓冲区默认数据存储空间大小是1M，由于存储空间大小是固定的，当入队元素的数量大于队列长度时，最先入队的元素会被弹出，而新元素会被放入队列
- 由来：每台服务器启动时，如果开启有AOF或被连接成为master节点，即创建复制缓冲区
- 作用：用于保存master收到的所有指令（仅影响数据变更的指令，例如set，select）
- 数据来源：当master接收到主客户端的指令时，除了将指令执行，会将该指令存储到缓冲区中

主从复制工作流程

主从服务器复制偏移量（offset）

- 概念：一个数字，描述复制缓冲区中的指令字节位置
- 分类：
  - master复制偏移量：记录发送给所有slave的指令字节对应的位置（多个）
  - slave复制偏移量：记录slave接收master发送过来的指令字节对应的位置（一个）
- 数据来源：
  master端：发送一次记录一次
  slave端：接收一次记录一次
- 作用：同步信息，比对master与slave的差异，当slave断线后，恢复数据使用

主从复制工作流程



主从复制工作流程

心跳机制

- 进入命令传播阶段候，master与slave间需要进行信息交换，使用心跳机制进行维护，实现双方连接保持在线
- master心跳：
  - 指令：PING
  - 周期：由repl-ping-slave-period决定，默认10秒
  - 作用：判断slave是否在线
  - 查询：INFO replication　　获取slave最后一次连接时间间隔，lag项维持在0或1视为正常
- slave心跳任务
  - 指令：REPLCONF ACK {offset}
  - 周期：1秒
  - 作用1：汇报slave自己的复制偏移量，获取最新的数据变更指令
  - 作用2：判断master是否在线

主从复制工作流程

心跳阶段注意事项

- 当slave多数掉线，或延迟过高时，master为保障数据稳定性，将拒绝所有信息同步操作

```
min-slaves-to-write 2
min-slaves-max-lag 8
```

slave数量少于2个，或者所有slave的延迟都大于等于10秒时，强制关闭master写功能，停止数据同步

- slave数量由slave发送**REPLCONF ACK**命令做确认
- slave延迟由slave发送**REPLCONF ACK**命令做确认

主从复制工作流程

注意在数据同步阶段的部分复制中 slaver 发送指令

给 master 用的是 psync2，而在命令传播阶段用的是 replconf ack。

主从复制常见问题

频繁的全量复制（1）

伴随着系统的运行，master的数据量会越来越大，一旦master重启，runid将发生变化，会导致全部slave的全量复制操作

内部优化调整方案：

1. master内部创建master_replid变量，使用runid相同的策略生成，长度41位，并发送给所有slave
2. 在master关闭时执行命令 **shutdown save**，进行RDB持久化,将runid与offset保存到RDB文件中
   - repl-id repl-offset
   - 通过redis-check-rdb命令可以查看该信息
3. master重启后加载RDB文件，恢复数据

   重启后，将RDB文件中保存的repl-id与repl-offset加载到内存中
   - master_repl_id = repl master_repl_offset = repl-offset
   - 通过info命令可以查看该信息

作用：

本机保存上次runid，重启后恢复该值，使所有slave认为还是之前的master

```
-rw-r--r--. 1 root root   344 Oct 14 07:09 dump.rdb
[root@localhost data]# redis-check-rdb dump-6379.rdb
```

主从复制常见问题

频繁的全量复制（2）

- 问题现象
  - 网络环境不佳，出现网络中断，slave不提供服务
- 问题原因
  - 复制缓冲区过小，断网后slave的offset越界，触发全量复制
- 最终结果
  - slave反复进行全量复制
- 解决方案
  - 修改复制缓冲区大小

```
repl-backlog-size
```

- 建议设置如下：
  1. 测算从master到slave的重连平均时长second
  2. 获取master平均每秒产生写命令数据总量write_size_per_second
  3. 最优复制缓冲区空间 = 2 * second * write_size_per_second

主从复制常见问题

频繁的网络中断（1）

- 问题现象
  - master的CPU占用过高 或 slave频繁断开连接
- 问题原因
  - slave每1秒发送REPLCONF ACK命令到master
  - 当slave接到了慢查询时（keys * ，hgetall等），会大量占用CPU性能
  - master每1秒调用复制定时函数replicationCron()，比对slave发现长时间没有进行响应
- 最终结果
  - master各种资源（输出缓冲区、带宽、连接等）被严重占用
- 解决方案
  - 通过设置合理的超时时间，确认是否释放slave

```
repl-timeout
```

该参数定义了超时时间的阈值（默认60秒），超过该值，释放slave

频繁的网络中断（2）

- 问题现象
  - slave与master连接断开
- 问题原因
  - master发送ping指令频度较低
  - master设定超时时间较短
  - ping指令在网络中存在丢包
- 解决方案
  - 提高ping指令发送的频度

```
repl-ping-slave-period
```

超时时间repl-time的时间至少是ping指令频度的5到10倍，否则slave很容易判定超时

主从复制常见问题

数据不一致

- 问题现象
  - 多个slave获取相同数据不同步
- 问题原因
  - 网络信息不同步，数据发送有延迟
- 解决方案
  - 优化主从间的网络环境，通常放置在同一个机房部署，如使用阿里云等云服务器时要注意此现象
  - 监控主从节点延迟（通过offset）判断，如果slave延迟过大，暂时屏蔽程序对该slave的数据访问

```
slave-serve-stale-data  yes|no
```

开启后仅响应info、slaveof等少数命令（慎用，除非对数据一致性要求很高）