



redis 核心技术与应用实践

作者：谢国恩

2018.9.16



「Why-What-How」

01

认识Redis

02

应用场景

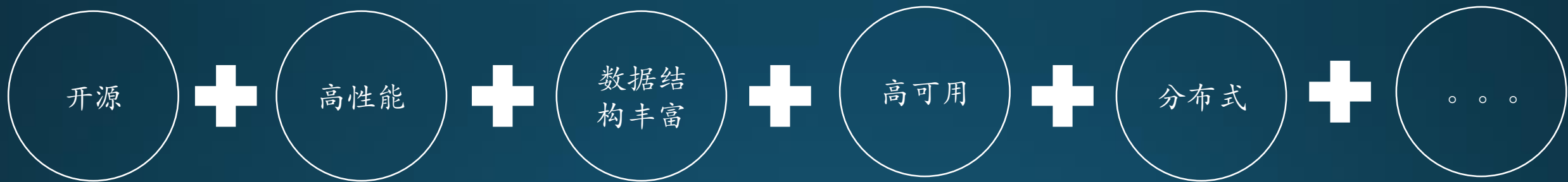
03

核心技术与应用



1

「Why」认识Redis



高级KV存储系统





2

「What」 应用场景

缓存

Top N

计数器

Uniq操作

Pub/Sub

队列

GEO

数据淘汰



3

「How」 核心技术与应用



队列

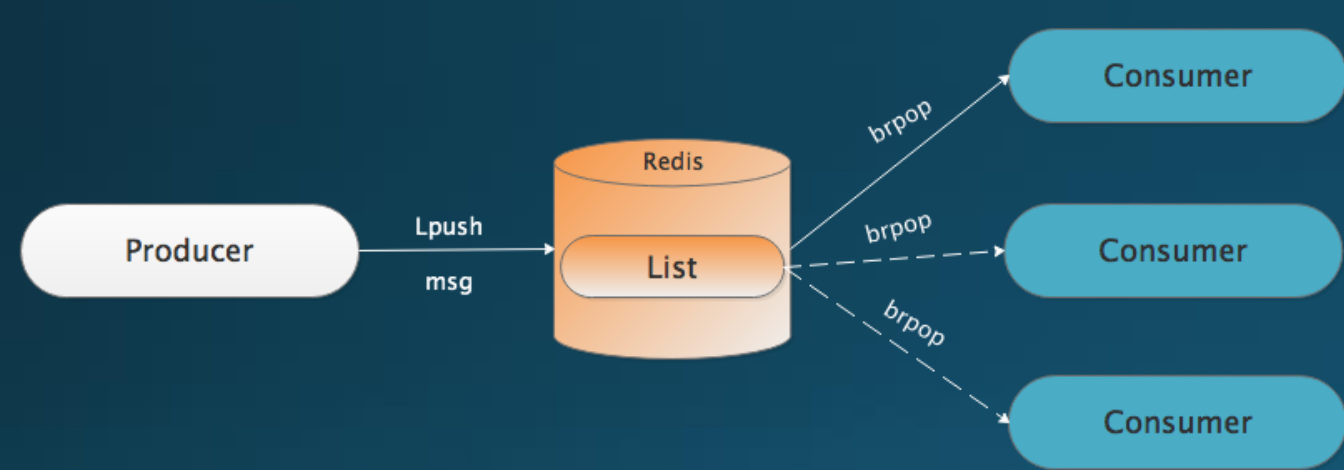
Redis-List
发布/订阅

持久化

RDB&AOF

高可用

redis主从 + 哨兵
(sentinel) + 漂移
VIP

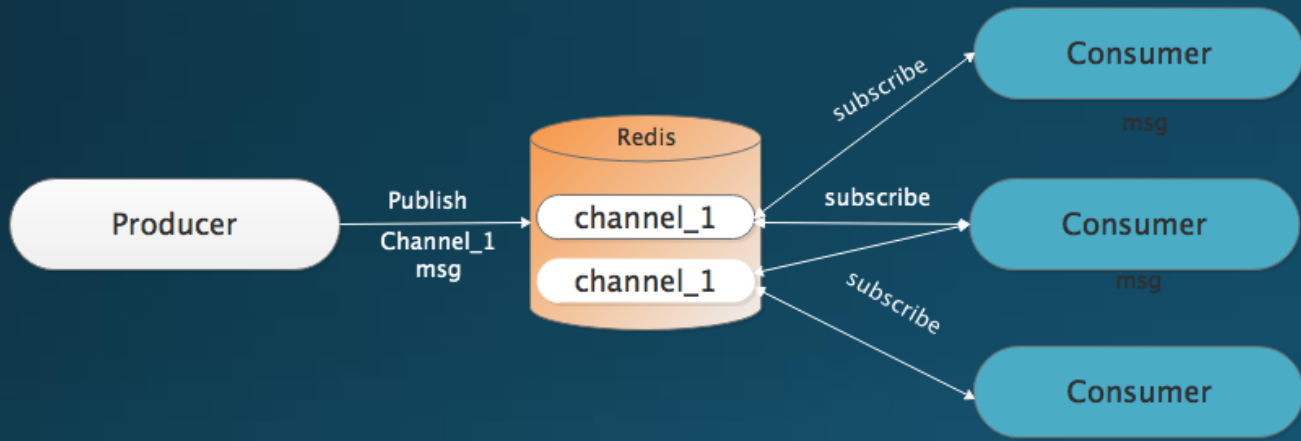


优点:

- 1、持久化
- 2、支持集群
- 3、使用简单

缺点:

- 1、一对一
- 2、高可用或崩溃后的处理机制需要自己实现
- 3、生产快于消费可能会导致内存溢出问题，进程 crash



优点:

- 1、一对多
- 2、支持集群
- 3、接口使用简单

缺点:

- 1、无持久化机制，即发即弃
- 2、没有保证pub/sub消息性能的方案
- 3、生产快于消费可能会导致内存溢出问题，进程 crash



- 定时执行
- Fork子进程，执行临时文件写入
- 替换，二进制文件压缩存储



- AOF持久化以日志的形式记录服务器所处理的每一个写、删除操作

RDB

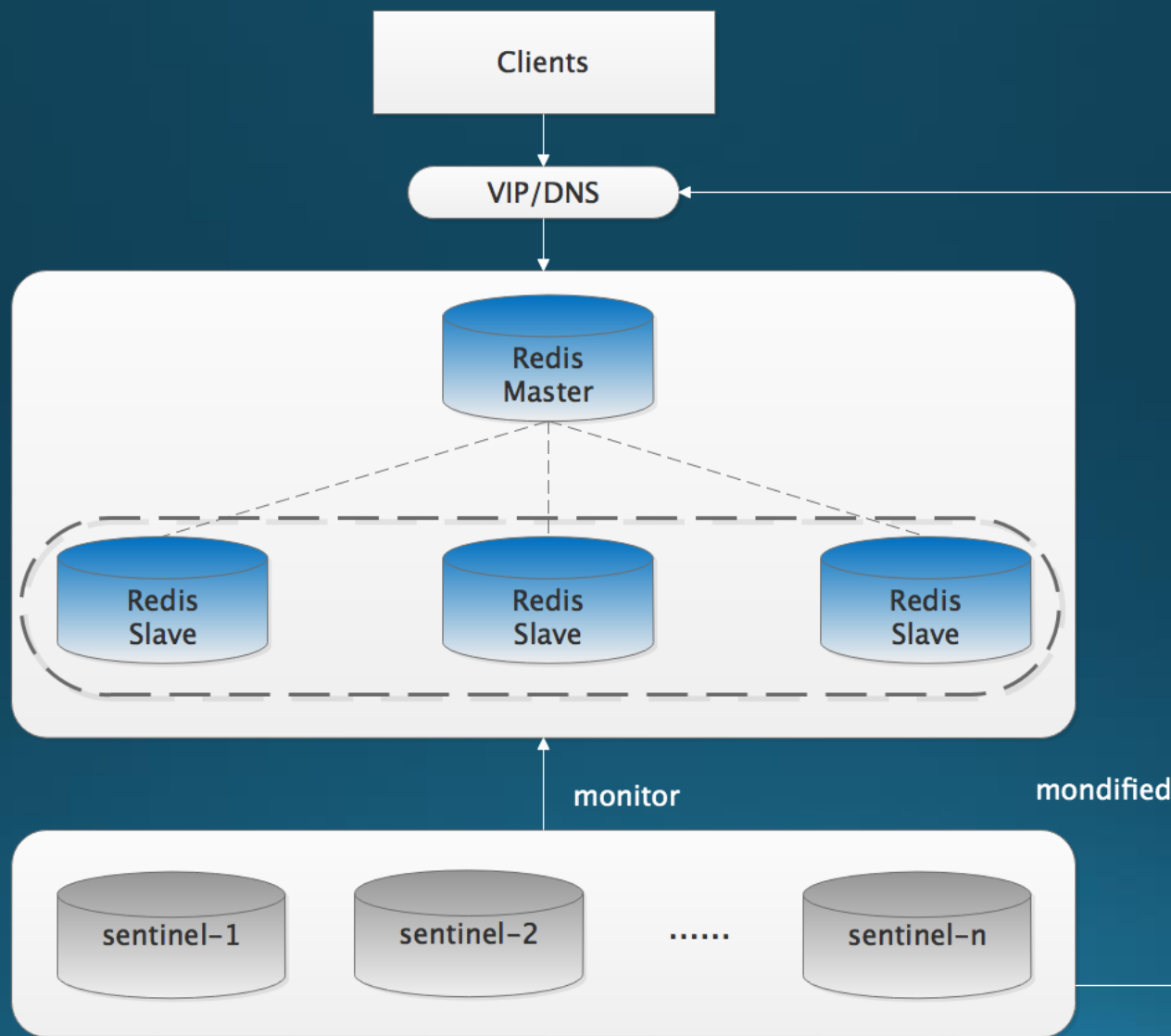
A1、单文件存储
A2、性能最大化
A3、启动效率高

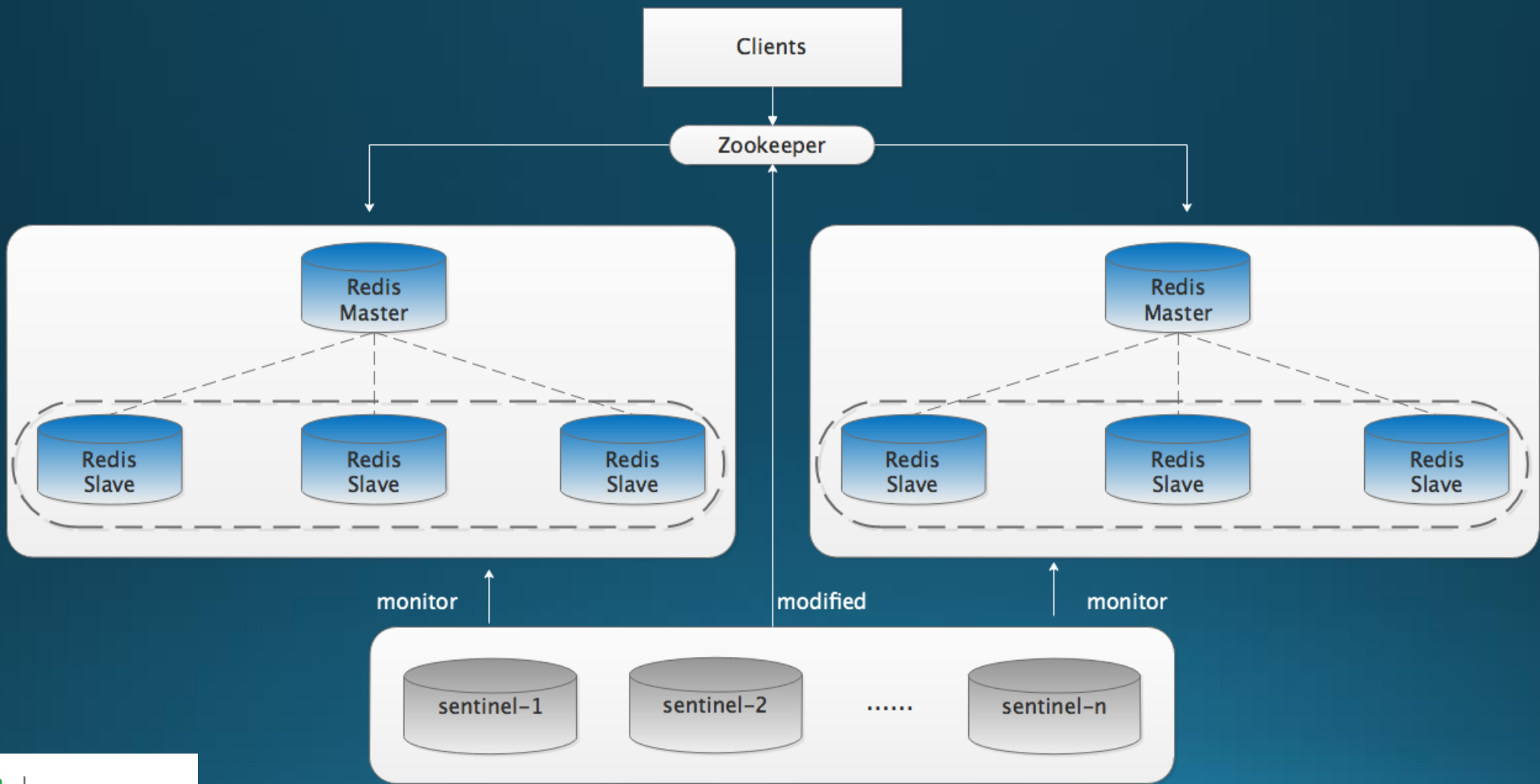
B1、单位时间内的数据丢失
B2、数据量大Fork时间过久

AO
F

A1、更高数据安全性：三种数据同步策略、rewrite机制
A2、redis-check-aof工具-解决数据一致性的问题

B1、文件大恢复慢
B2、性能弱





主从

规避单点

哨兵

虚拟IP (Virtual IP, VIP) :
主从切换-回调脚本-Client端
友好

VIP

Sentinel: 监控Redis Server服务的进
程

优点

- A1、官方社群推出、部署简单
- A2、主从模式下高可用
- A3、易于扩展、突破瓶颈

不足

- A1、复杂性
- A2、资源浪费
- A3、局限性

改进

- A1、合理设置参数，防止误切
- A2、各节点时间同步
- A3、Redis建议使用pipeline和multi-keys操作，减少RTT次数，提高请求效率
- A4、自行搞定配置中心（zookeeper）

- Redis持久化磁盘IO方式及其带来的问题
- 安全问题：非root用户运行、开启危险命令认证...
- 容量问题：合理使用内存分配策略（no-eviction、allkeys-random ...）
- big-key、hot-key问题
- 避免使用阻塞操作（如：flushall、flushdb、keys *等）
- 配置上：建议开启tcp-keepalive, tcp-backlog
- Key过期问题：合理设置过期时间

- 01 充分了解，关注竞品、替代方案
- 02 掌握其应景场景，只有更好、更合适
- 03 技术是实践科学，“知道” - “做到”，需长期刻意练习

Thank you

PPT by Hsieh

