服务注册中心

HUSKAR

ABOUT ME

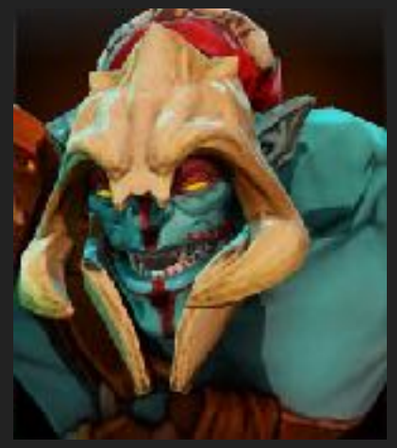
- 张江阁，2016 年 3 月加入饿了么，前半年服务于北研（现北京中心技术创新部），后加入框架部负责 Huskar 项目，在北京组建了框架部的三人小分队。
- 框架工具部：上海技术中心的业务支持部门，提供SOA 框架、多活中间件、数据库和缓存中间件等基础服务。

Huskar - 服务注册中心
Corvus - Redis 集群中间件
Eless AppOS - 容器构建部署平台

Trace - 调用链跟踪平台
Sam - 带服务发现的负载均衡器
API Router - 多活路由层

DAL DRC - 数据库中间件
MaxQ - AMQP 消息队列

什么是 HUSKAR

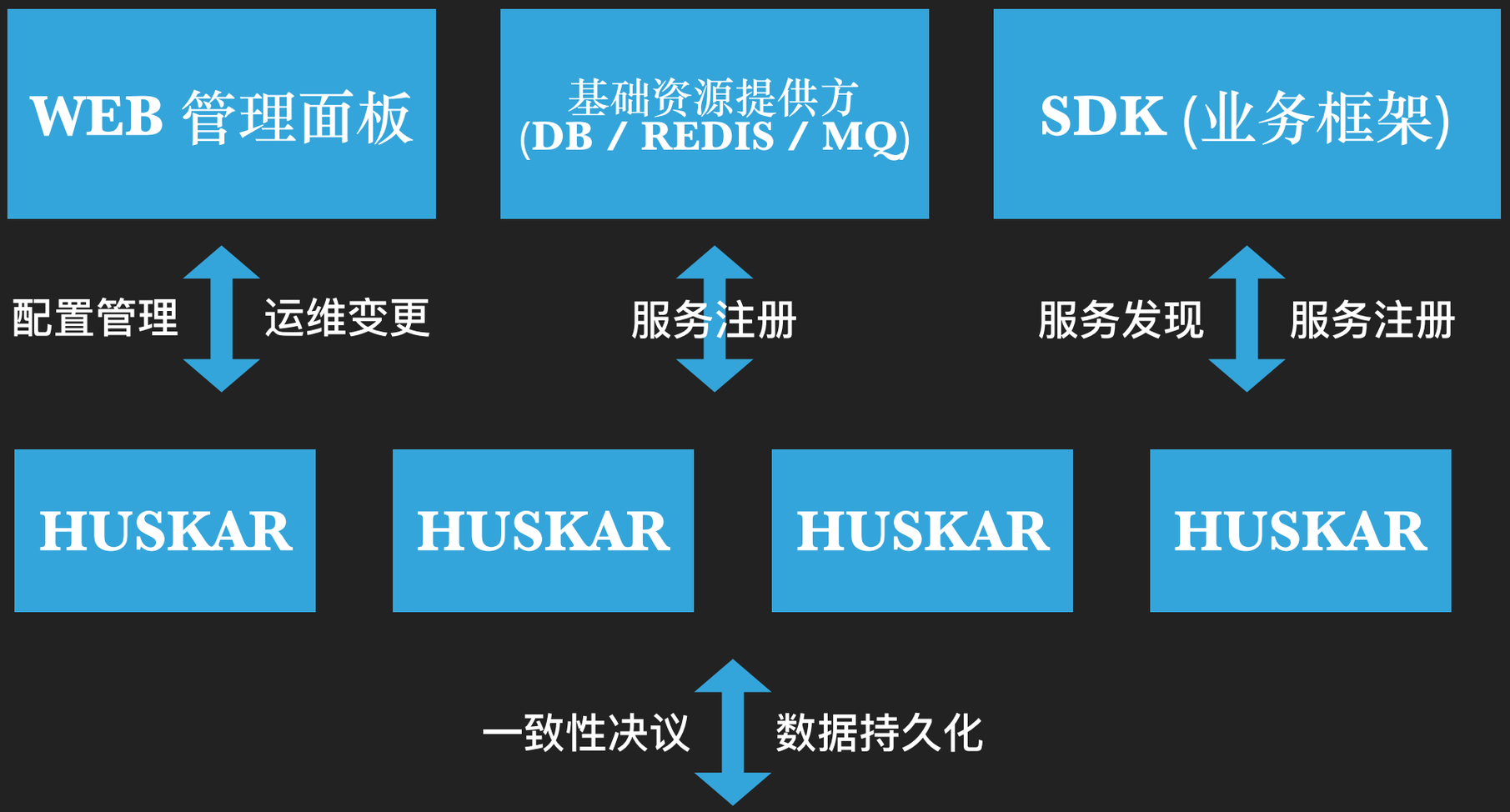


- 功能：提供服务注册和发现
- 设计：基于 ZooKeeper 的弱状态的中间件
- 基线：高可用、容易 scale、有 SOA 感知和控制能力
- 技术栈：Python + Gevent + Kazoo + Apache Curator

为什么要有 HUSKAR

- 最早的 Huskar 仅是一个 ZooKeeper SDK
- 多语言出现，客户端软负载的模式依赖可靠的服务发现
- ZooKeeper 客户端数量不宜太多（踩坑经验）
- 需要满足公司 SOA 的新需求，提供流量控制能力
- 需要能应对故障，可快速降级，也可快速 hotfix

HUSKAR 如何构成


WEB 管理面板
基础资源提供方
(DB / REDIS / MQ)
SDK (业务框架)
配置管理
运维变更
服务注册
服务发现
服务注册
HUSKAR
HUSKAR
HUSKAR
HUSKAR
一致性决议
数据持久化
ZOOKEEPER
集群
ZOOKEEPER 多活复制中间件
可降级的 MYSQL 和 REDIS

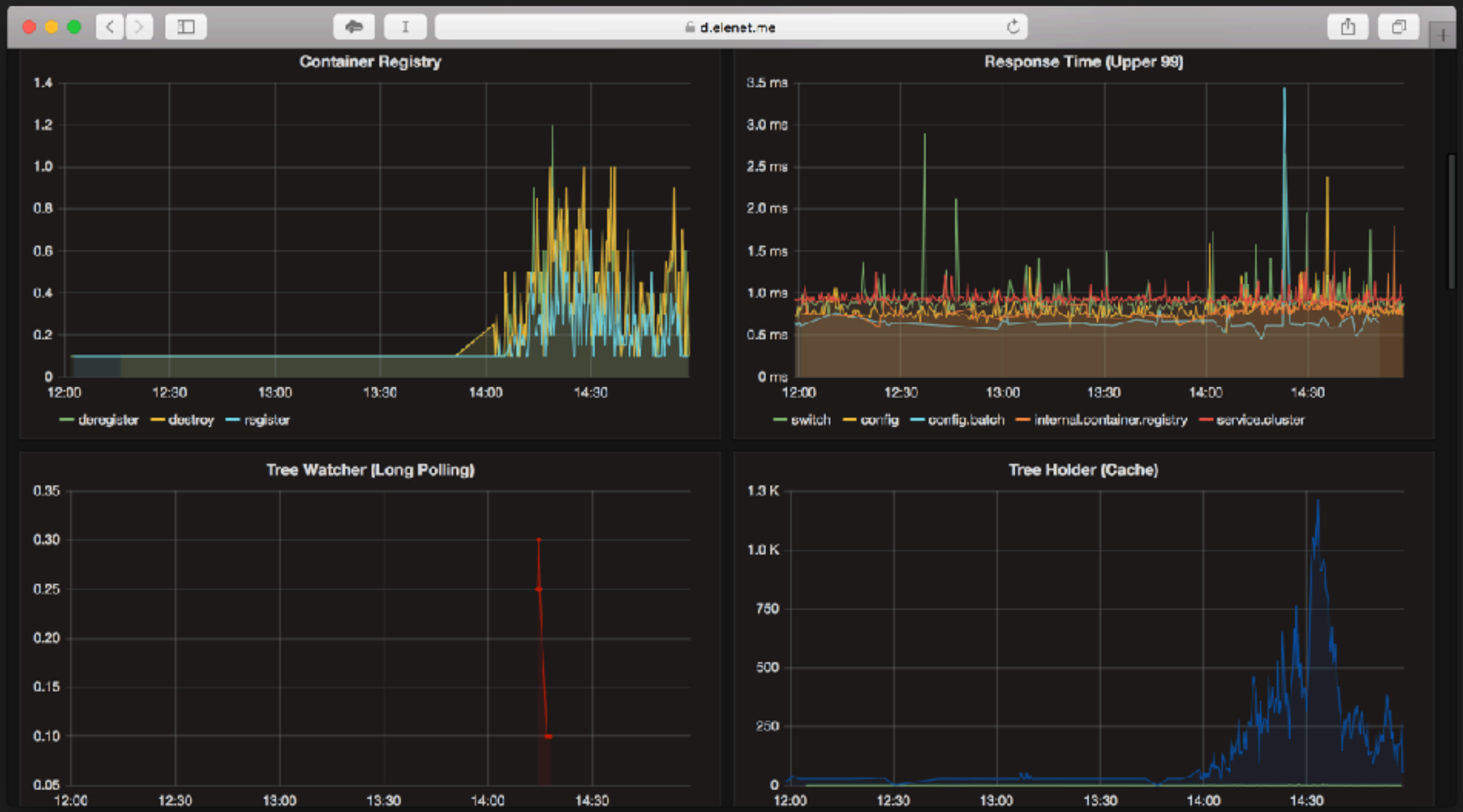
HUSKAR 如何做高可用

- 核心依赖（ZooKeeper）和非核心依赖分开
- 核心接口（注册、发现）和非核心接口（管理）分开
- ZooKeeper 故障、选举、GC 等：内存数据兜底
- 出现问题要先判断，不要先重启
- 接入层：HAProxy 双机双 VIP 互为备份 (keepalived)

HUSKAR 如何做高可用

- 降级次序 - 1：降级管理接口（最小模式）
- 降级次序 - 2：降级服务注册（只读模式）
- 降级次序 - 3：IDC 切换（另一个 IDC 顶替）
- 自动降级（熔断）比人工干预时效快
- 人工干预时，指标对于判断和决策大有帮助

HUSKAR 的内部指标 - 性能、活动信息

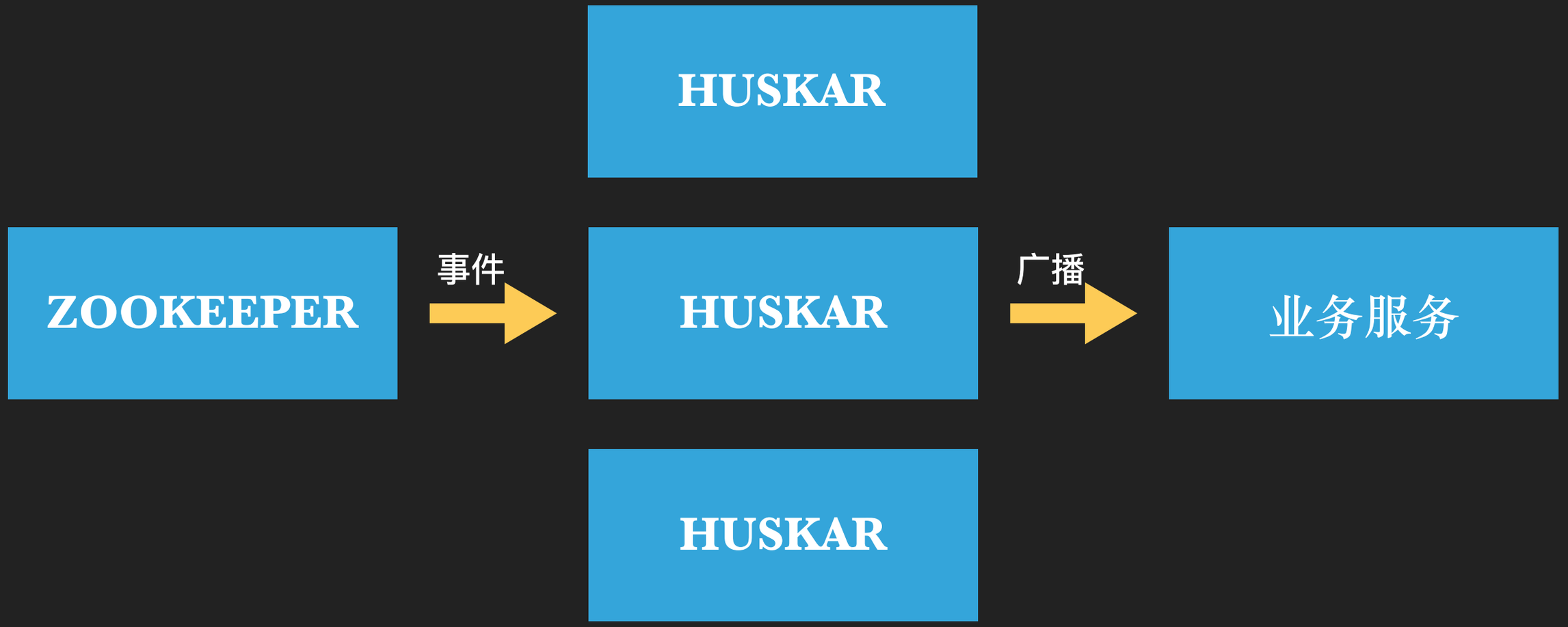


HUSKAR 的内部指标 - 节点状况

HUSKAR 的内部指标 - ZOOKEEPER 监控

如何 SCALE

- 服务注册的需求容量是静态的：取决于 IDC 规模
- 服务发现的需求容量是动态的：取决于调用拓扑



SOA 和 HUSKAR

- app_id: 服务的唯一标识，背后对应一个 codebase
- cluster: 一组服务实例，软负载的最小流量单元

- 注册：你是谁（token），你注册到哪（app_id + cluster）
- 发现：你是谁（token），你调谁（app_id + cluster）

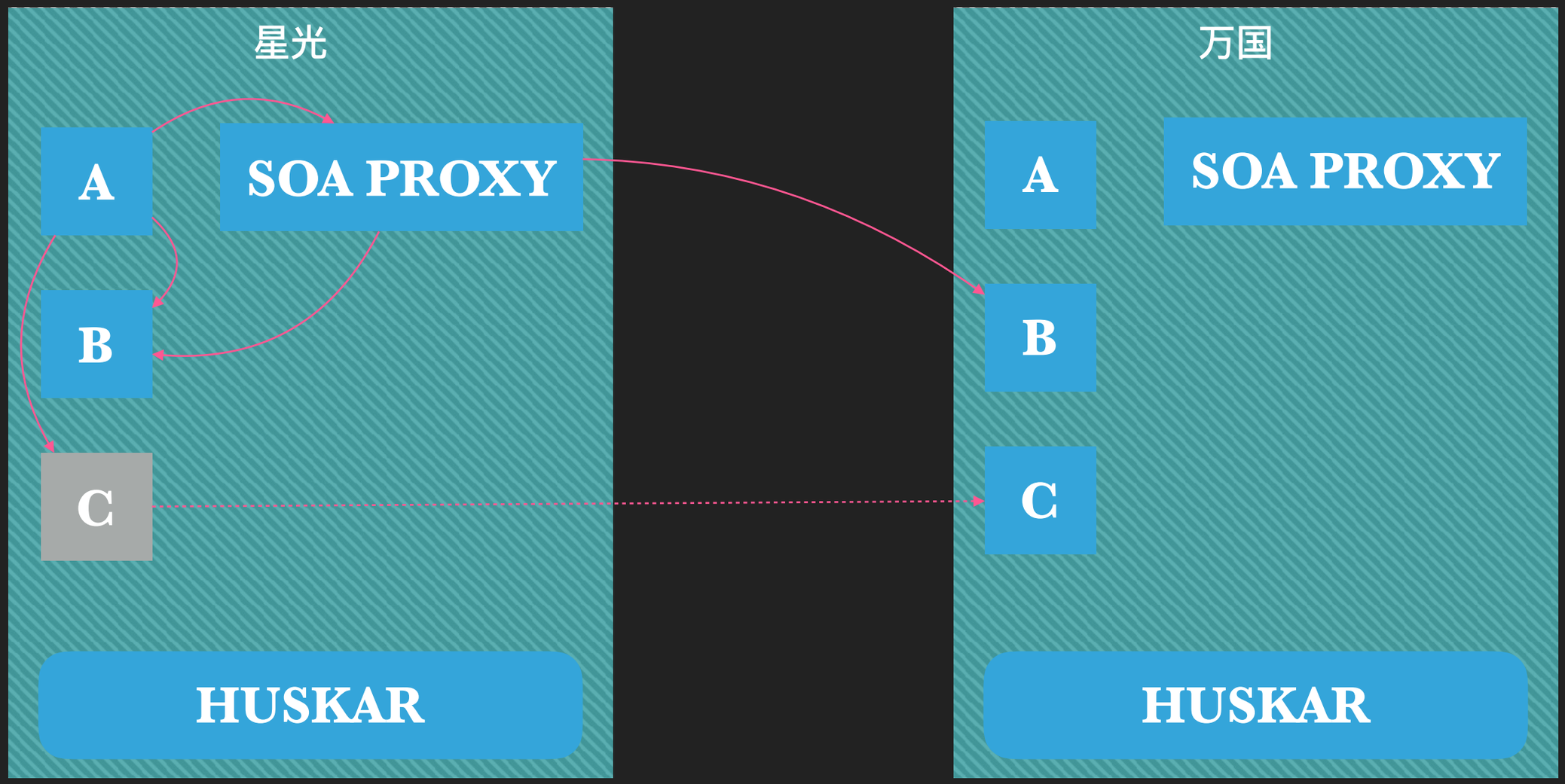
新服务接入 HUSKAR 需要做什么

- 说接入 Huskar 不如说是接入 SOA
- 服务提供方：提供服务（RPC） + 把自己注册上去
- 服务调用方：客户端软负载 + 调用 + 熔断限流降级 etc
- 我需要在我的机房部署 Huskar 吗？包办了
- 可能需要部署 SOA Proxy 等
- RD 对接框架，框架对接 Huskar

SOA 和多活

- Huskar 本身是一个多活服务，各 zone 对等
- app_id + cluster, cluster 会需要加一个 zone 前缀
- 多活服务调用
  - 不跨 zone 调用多活服务：直接调
  - 跨 zone 调用非多活服务：通过 SOA Proxy (业务路由)
  - 调用 global zone 服务：直接调

SOA 和多活



下一步 HUSKAR 会做什么

- 托管路由：调用方只需要提供 app_id 不需要提供 cluster
- 基础资源：Database db = DatabaseRegistry.get("book")

- 如何兼容北京中心新框架：
  短期保持向后兼容，长期共同演进

Q&A