# 微服务架构

Huayulei_2003@hotmail.com
2018/05/15

# 目录

# 概述：目标和业务需求

**整体目标：**

1. 规范性：建立服务快速开发、部署、运维管理、持续开发、持续集成的流程规范
2. 自动化：开发代码并提交到代码库，简单配置，服务就会自动集成、自动部署
3. 高可用：服务采用集群部署、多节点同时服务
4. 扩展性：服务易于水平扩容和缩容
5. 高性能：既能满足现有阶段性能需求又能支撑业务量快速发展的下一阶段

**兼容现有和未来业务需求**

- 支持开发语言的多样性，如：php、golang、python、c/c++等
- 支持数据库的多样性，如：MySQL，Mongodb，Postgresql，Redis，Memcached
- 支持接口协议的多样性，如：http、https、http2、gRPC、websocket
- 支持系统容量和日活跃用户在未来阶段的增长（1000W日活）

# 概述：高效技术团队

高效技术团队的技术特征：

- 微服务架构

微服务和SOA架构是目前主流，并且微服务更加流行、高加高效

- 容器和容器编排技术

大量使用容器技术和容器编排技术，其中以docker + kubernetes

- DevOps理念深度落地

有效的版本管理、高度敏捷CI/CD体系，高速软件迭代交付

- 自动化程度比较高

大量应用工具和系统自动化管理，尽量减少人力成本并保证高效率和质量

# 微服务：微服务概述

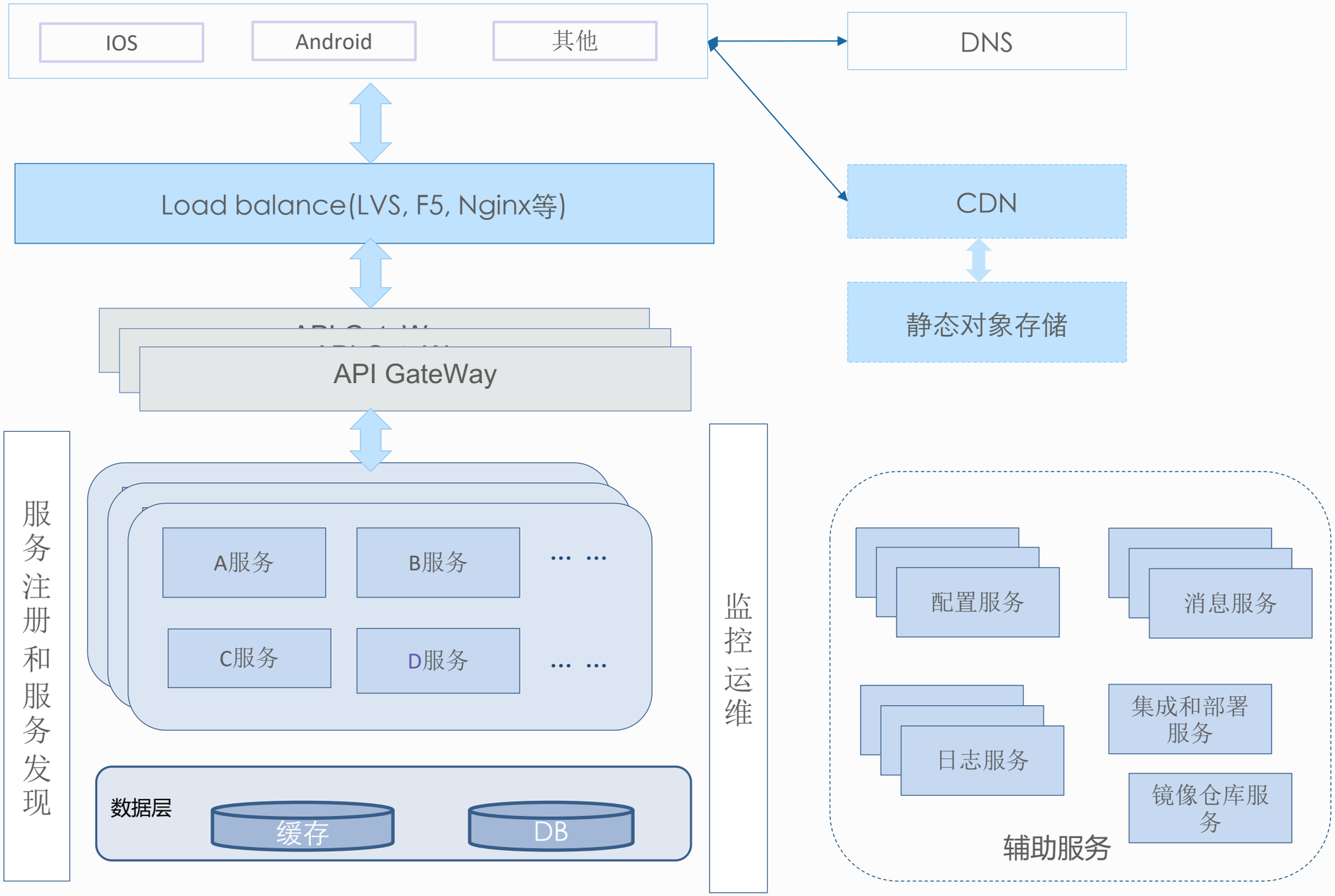
**微服务的优点：**

① 服务本身逻辑功能比较简单
② 对于一个服务，可以选择最好和最适合的语言和框架来开发
③ 服务之间本质上是松耦合的
④ 多个团队可以同时并行工作在不同的服务中
⑤ 可对某一服务持续发布，而对其他的影响相对较小
⑥ 可以水平扩容和缩容，灵活方便
⑦ 使用集群多点对外提供服务，提高了可用性。

**微服务架构的难点：**

① 微服务可能增加调用开销：比如网络
② 微服务对监控运维要求比较高
③ 服务间以接口访问、接口设计要求比较高
④ 分库、分表等数据拆分增加了系统的复杂度
⑤ 多重缓存、多数据库带来的数据一致性的挑战
⑥ 服务测试性的成本和复杂度上升

# 微服务：整体架构概述


IOS
Android
其他
DNS
Load balance(LVS, F5, Nginx等)
CDN
静态对象存储
API GateWay
服务注册和服务发现
A服务
B服务
… …
C服务
D服务
… …
监控运维
数据层
缓存
DB
配置服务
消息服务
日志服务
集成和部署服务
镜像仓库服务
辅助服务

# 微服务：API Gateway概述

GateWay优点：

① 服务发现与动态负载均衡：自动发现后端拆分、聚合、扩容、缩容的服务集群，当后端服务有所变化的时候，能够实现健康检查和动态的负载均衡。

② 流量控制：为每种类型的请求分配容量，当请求数量超过阀值时抛掉外部请求，限制流量，保护后台服务不被大流量冲垮；当新版本上线时，可以控制流量在新老版本之间的分配。

③ 灰度发布与AB测试：通过配置访问路由，以及访问权重，可实现灰度发布，或者AB测试。同时上线两套系统，通过切入部分流量的方式来测试和验证新系统。

④ 统一的接口协议和规范：使用统一的接口协议格式和接入规范、接入SDK、避免管理混乱造成的不可维护性。

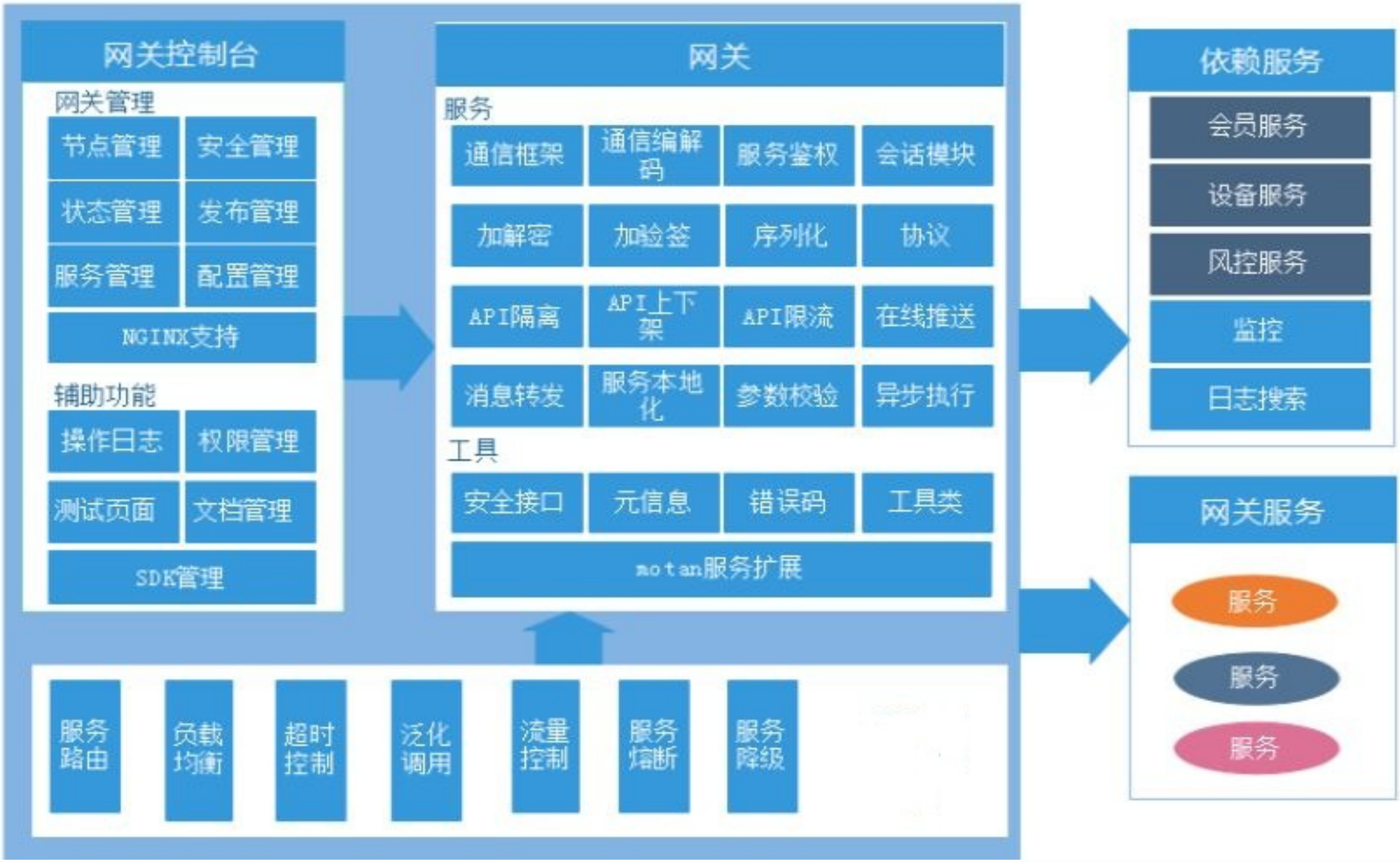
⑤ 身份认证和安全性控制：对每个外部请求进行用户认证，拒绝没有通过认证的请求，还能通过访问模式分析，实现反爬虫功能。

⑥ 数据监控和日志分析：可收集数据和统计，为系统服务和整体架构优化提供数据支持。

GateWay缺点：

① 所有服务都流经Gateway，调用微服务性能有一定损耗。

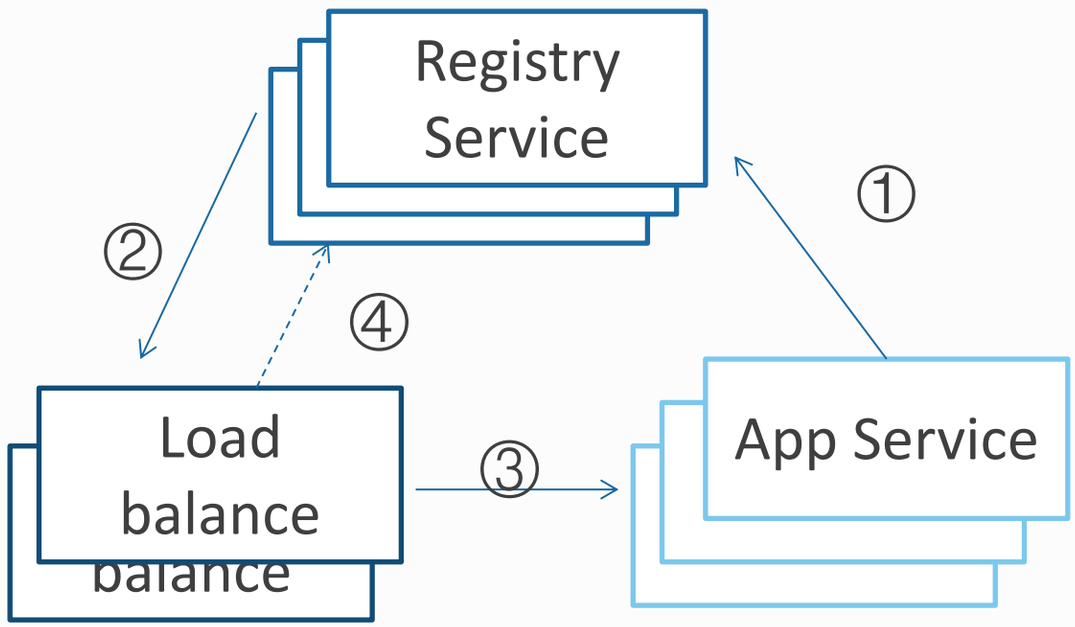
# 微服务：API Gateway逻辑功能



# 微服务：API Gateway开源项目

| 名称 | 开发语言 | 支持方 | star | 网址 |
| --- | --- | --- | --- | --- |
| zuul和zuul2 | java | Netflix | 5424 | https://github.com/Netflix/zuul/wiki |
| kong | c+lua | kong | 16627 | https://konghq.com/ |
| tyk | go | TykTechnologies | 3465 | http://tyk.io |
| fabio | go | eBay | 4824 | https://fabiolb.net |
| traefik | go | | 15925 | https://traefik.io/ |

# 微服务：服务注册发现与负载均衡




●app Service负载提供业务服务
●Registry service负责服务注册和服务健康检查
●Load balance负责服务发现和负载均衡。在存在API GateWay的场景下，该服务也要注册以提供给GateWay服务发现

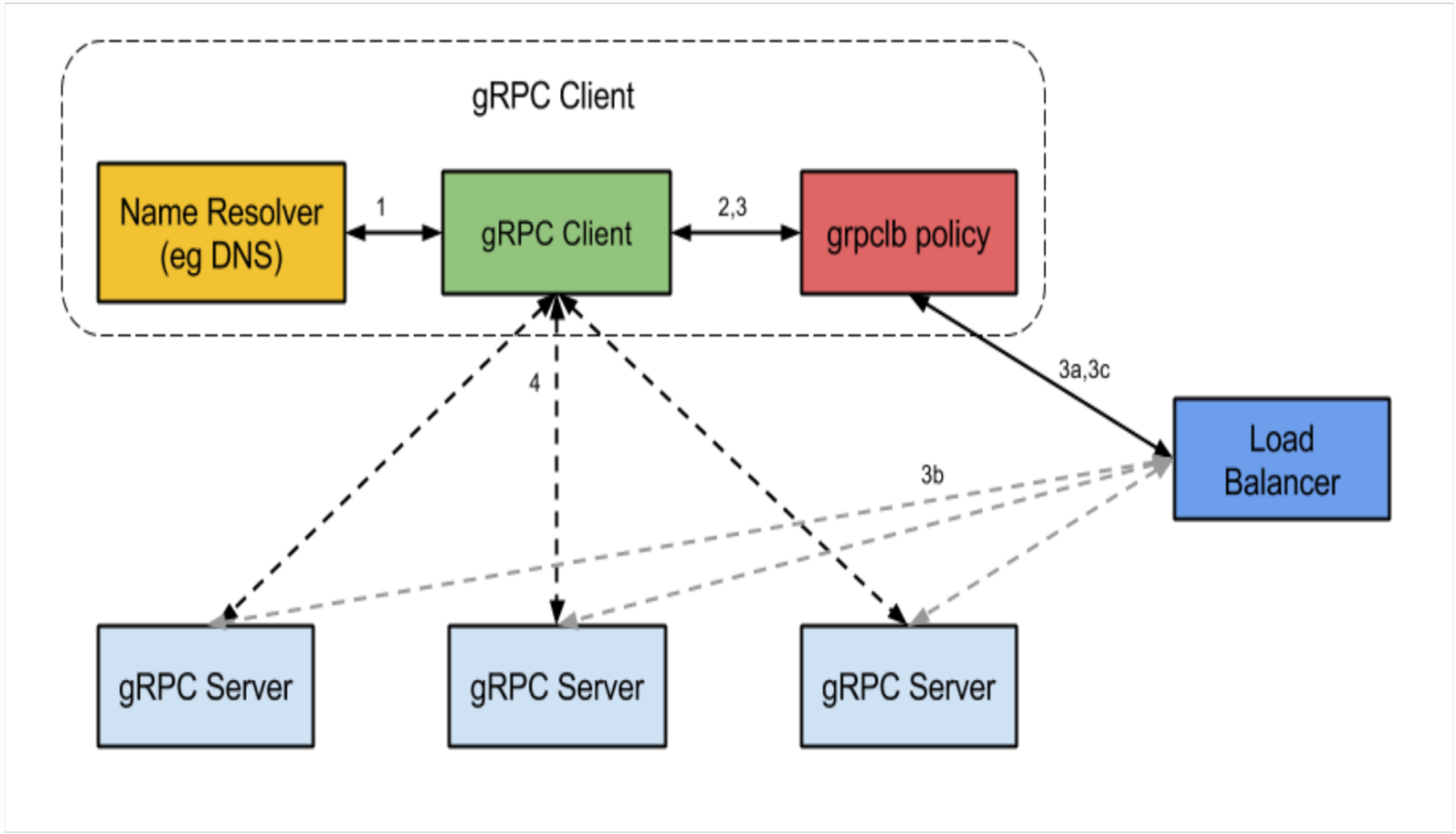
**主要优点：**

- 可扩展性：任务服务都可以水平扩容和缩容。
- 高可用：出现单点故障，不影响业务。
- 单一服务职责和功能分离：便于升级维护

**主要缺点：**

- 整体系统调用关系复杂，系统监控和故障恢复管理难度增大
- 服务交互增多，性能有所下降
- 此方案适合业务量级比较大、基础架构成熟的公司。

# 微服务：gRPC服务注册发现与负载均衡




Load-balancing policies fit into the gRPC client workflow in between name resolution and the connection to the server.

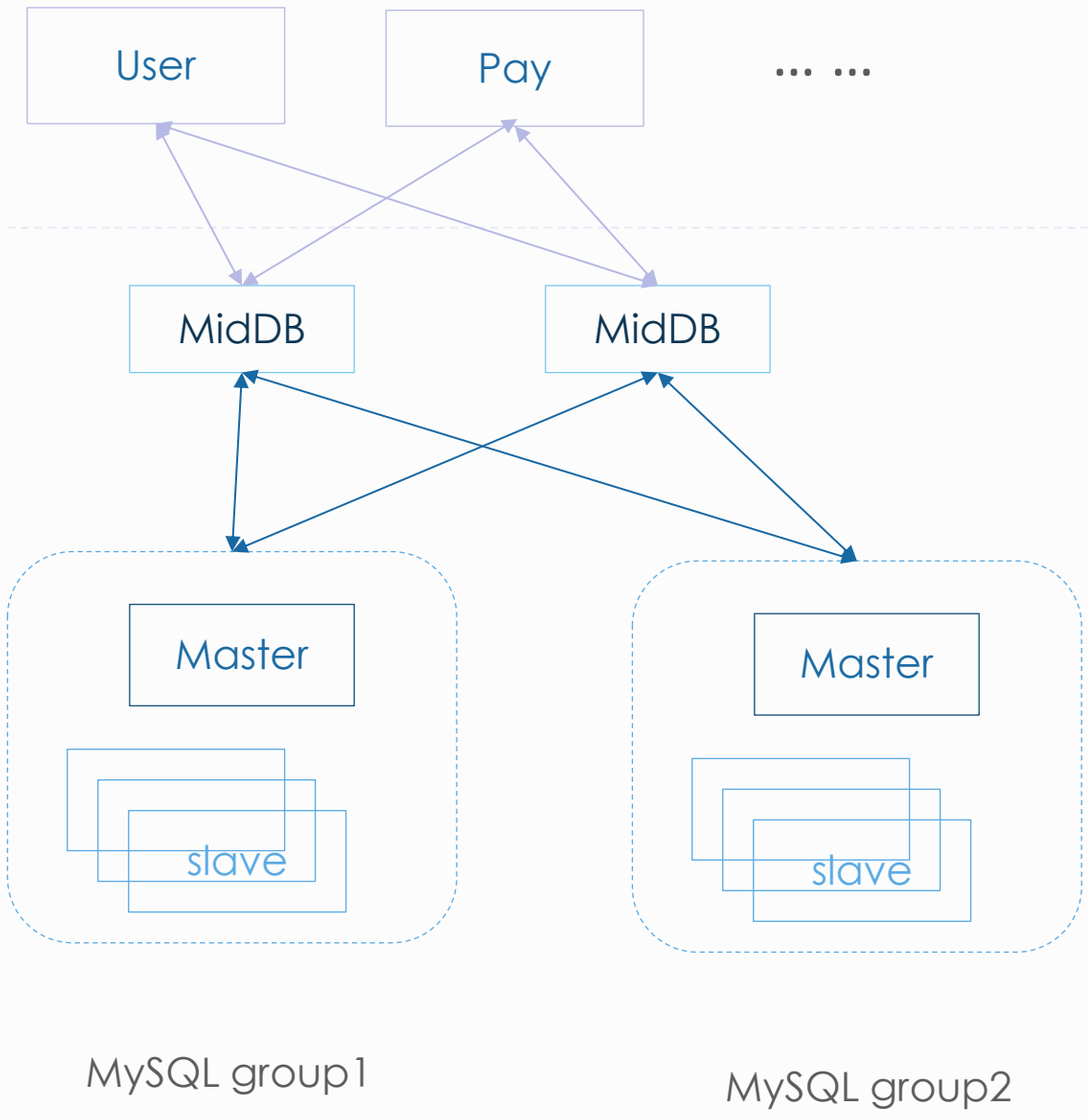
Our solution is： gRPC and Etcd 或 gRPC and consul

https://github.com/grpc/grpc/blob/master/doc/load-balancing.md

# 微服务：服务注册发现开源项目

| 名称 | 开发语言 | 支持方 | star | 网址 |
| --- | --- | --- | --- | --- |
| zookeeper | java | apache | 4618 | http://zookeeper.apache.org/ |
| consul | go | HashiCorp | 12475 | https://www.consul.io/ |
| etcd | go | coreos | 18845 | https://coreos.com/etcd |
| | | | | |

# 微服务：数据库中间层




数据库中间层MidDB主要功能：

1. 支持分库分表

① Hash模式

② 支持range模式

2. 支持读写分离

① Master写或强制读

② Slave负载均衡读，带权重

3. 其他功能：

① Master或Salve的上线、下线

② 支持SQL日志和慢日志查询

③ 支持Client连接管理

④ Metric日志监控和错误日志监控

# 微服务：数据库中间层开源项目

| 名称 | 开发语言 | 支持方 | star | 网址 |
| --- | --- | --- | --- | --- |
| Mycat-Server | java | | 4410 | http://mycat.org.cn |
| cobar | java | alibaba | 2251 | https://github.com/alibaba/cobar |
| Atlas | c | Qihoo360 | 3536 | https://github.com/Qihoo360/Atlas |
| kingshard | go | | 3564 | https://github.com/flike/kingshard |
| TiDB | go等 | pingcap | 13749 | https://pingcap.com |

# 微服务：监控概述

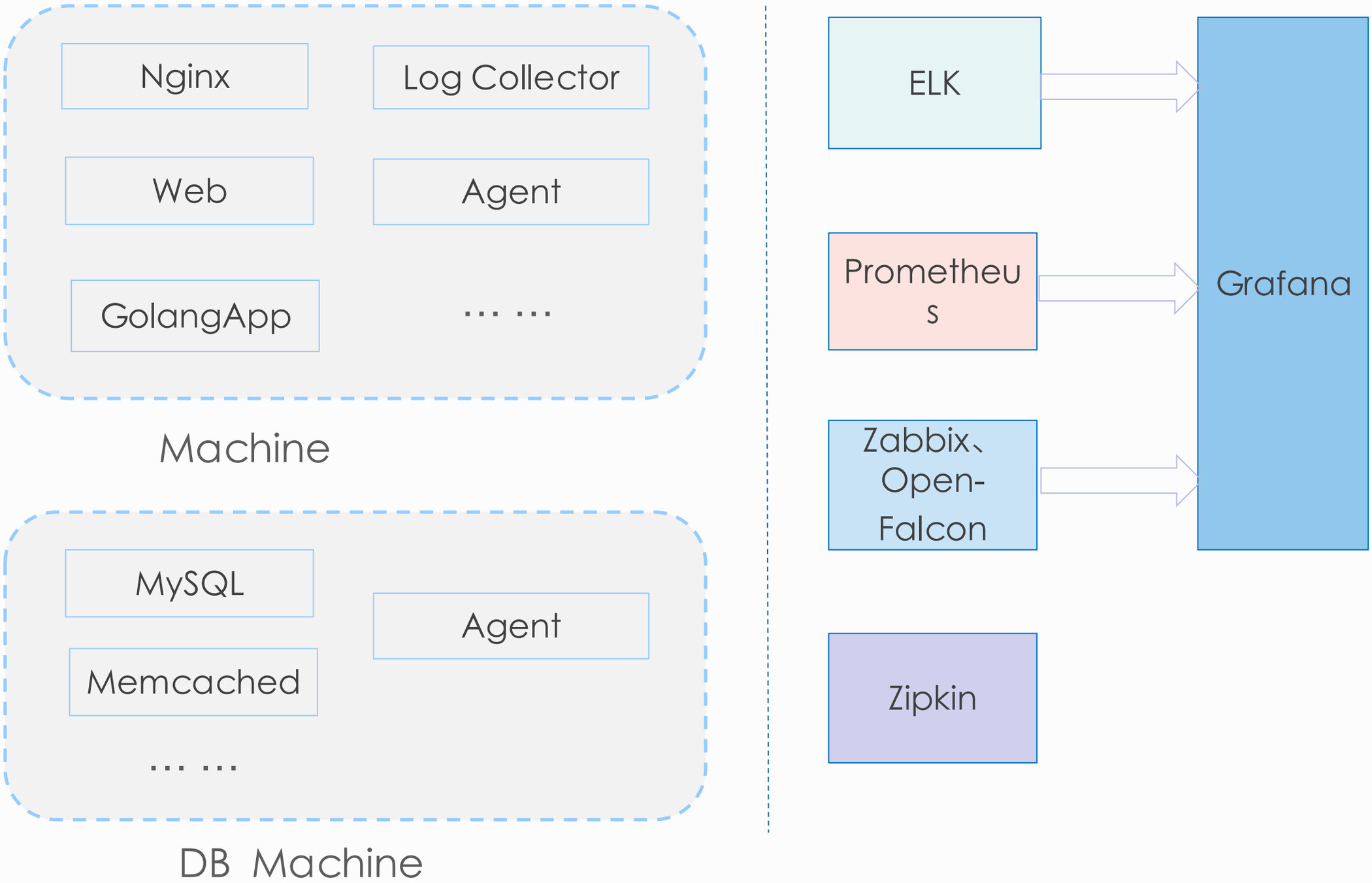
**选型原则：**

- 稳定性：监控服务不能造成线上服务不稳定
- 性能：监控服务应尽量减少对硬件资源占用，不会影响线上服务性能
- 效率：尽量使用成熟，很多企业实践过的方案，文档和社区支持比较完善

**主要描述：**

- 基础监控：使用Zabbix服务监控、Open-Falcon服务监控
- Metric监控：使用prometheus监控
- 服务状态监控：使用收集脚本上传Zabbix Agent监控、Open-Falcon Agent服务监控
- 调用链监控（可选）：推荐使用zipkin
- 故障定位监控（可选）：推荐使用sentry

# 微服务：监控架构


Nginx
Log Collector
Web
Agent
GolangApp
… …
Machine
MySQL
Agent
Memcached
… …
DB Machine
ELK
Prometheus
Zabbix、Open-Falcon
Zipkin
Grafana

# Q&A

Thank you very much !