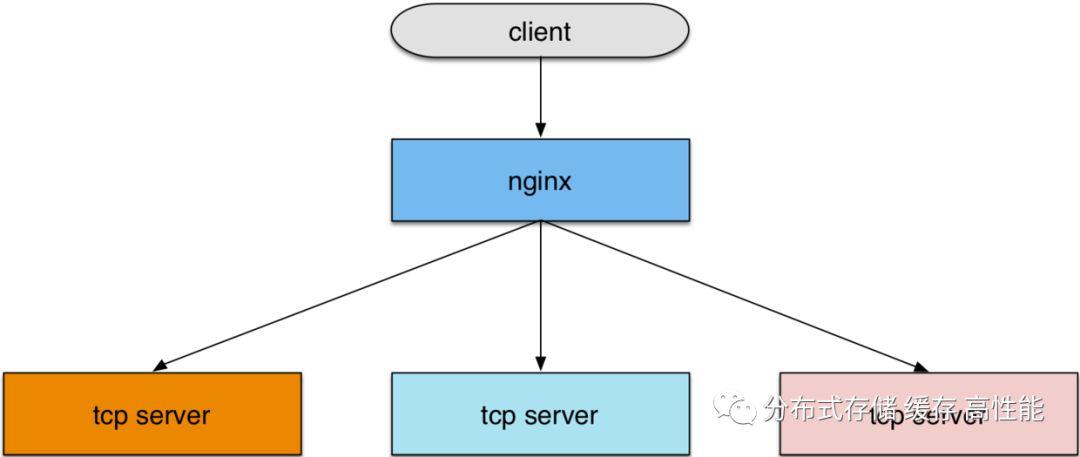
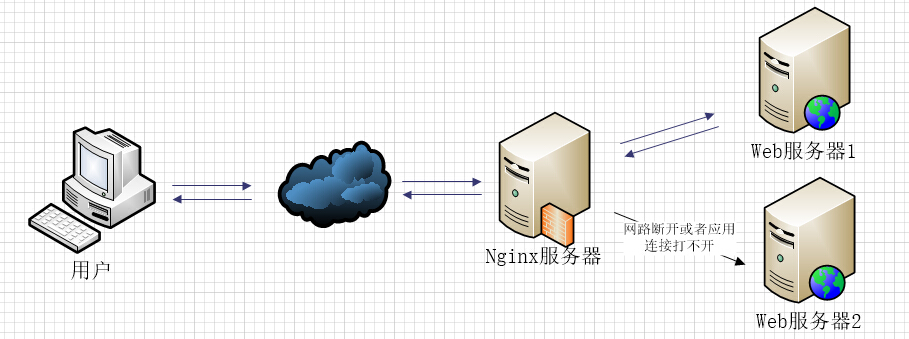
现有架构模式

# 系统架构图

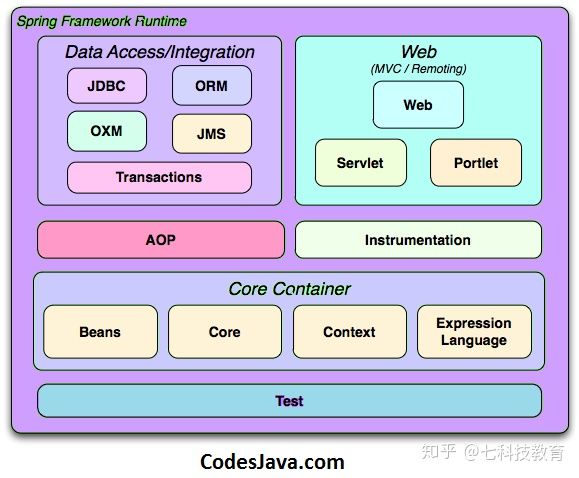
## 客户端》nginx 》





## ## 提供给 客户端的api接口

主要是 rest 接口springboot



## Cache 缓存

Spring cache和redis

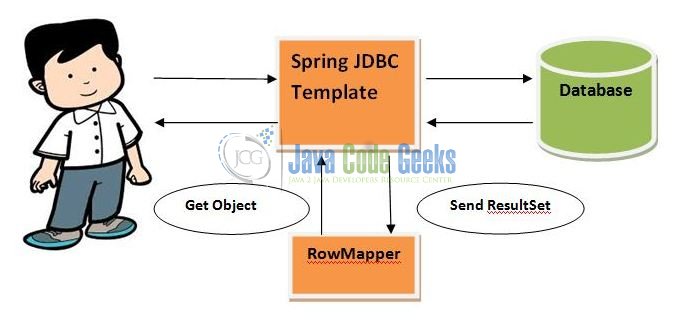
## ## database 数据库 与访问接口

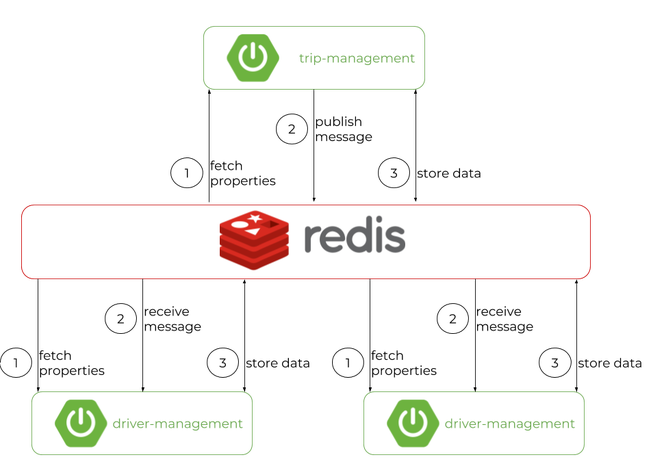
mybatis 5.7 以及以上

redis （nosql 数据库）

spirng jdbctemplete

mybatis ， redis驱动lib（socket接口）



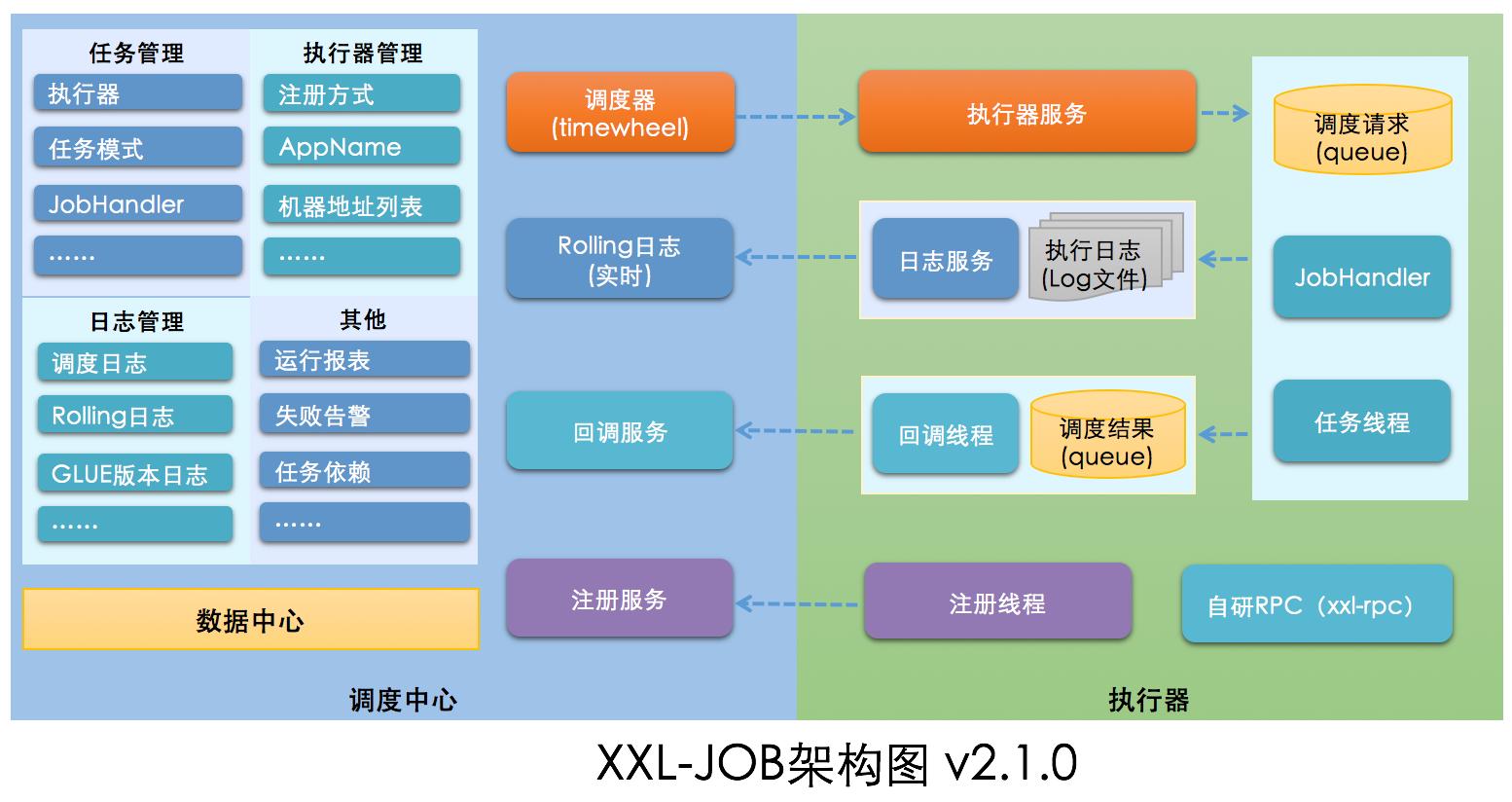


## ## msa微服务与负载均衡

nginx （ 目前）

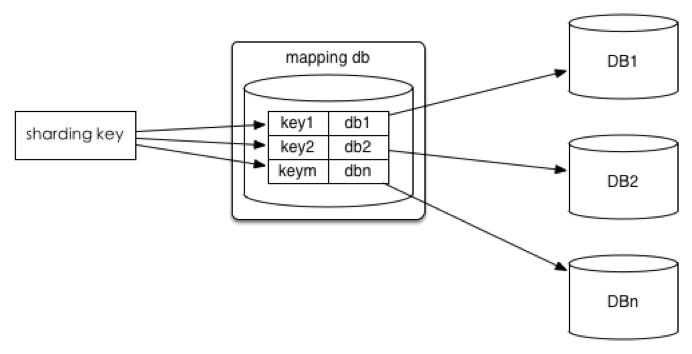
## ## timer定时调度

Xxljob



## ## 分库分表

自实现



# 架构改造总目标

架构简化简单容易运维测试

减少不必要的层次，一方面也提升了性能，减少了周转层次

即时修改生效 热部署随时修改配置和代码即时生效。。

尽可能减少编译打包

提升启动速度 尽可能控制在几秒内启动

频繁变动业务配置化 方便修改 提升扩展性

稳定性与性能提升，

项目启动解耦合，可以方便不同顺序方便启动不会互相影响，解除启动顺序依赖

# 功能系列增加

## 分布式的文件存储

# 架构系列 简化与提升扩展性系列

实现接口的通用性 多功能 尽可能业务无关性

频繁变动的业务配置化 脚本化

尽可能使用标准的协议

同样目标使用简单技术实现

尽可能使用现有的通用组件

# 架构改造 提升稳定性

多路冗余微服务

负载均衡配置前移，减少单点

使用简单技术实现功能与性能

# 架构改造 提升性能系列

## 比较重要的性能提升三件套 简化（层次架构避免过多中转），多路分散（**微服务msa负载均衡**、分区，分库），cache

cache （mq cache等）协调高性能组件和低性能组件的不匹配。

冷热数据分离 把热数据放入cache更快读写。。

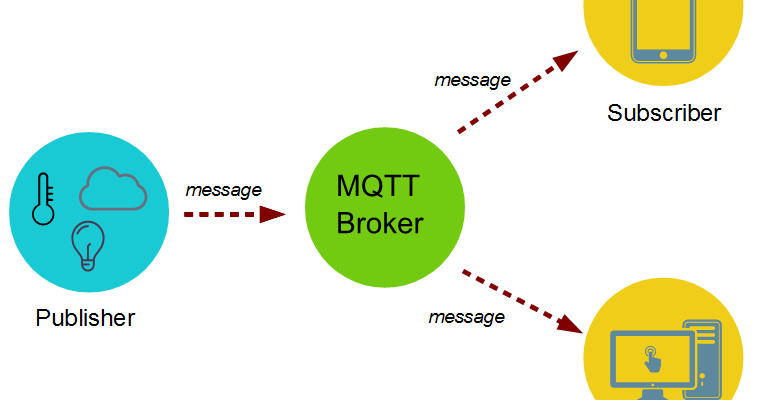
时间分散削峰填谷，可以吧高性能组件产生数据等待低性能组件消费完毕，削峰填谷。。

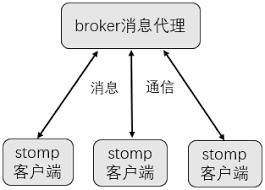
以及可以把低性能组件数据cache到一起，高性能组件一次处理。。

## 消息推送mq

支持Mqtt协议（常用）） stomp协议（常用）） xmpp协议（不常用）

其他消息推送私有协议建议不使用，开放协议优先比较好





## ##提供给 客户端的api接口改造

在nginx 和 web服务器之间增加msa微服务框架

暂时不变，后期可以根据需要增加一些rpc方面的接口，不只是rest

## ## msa微服务与负载均衡

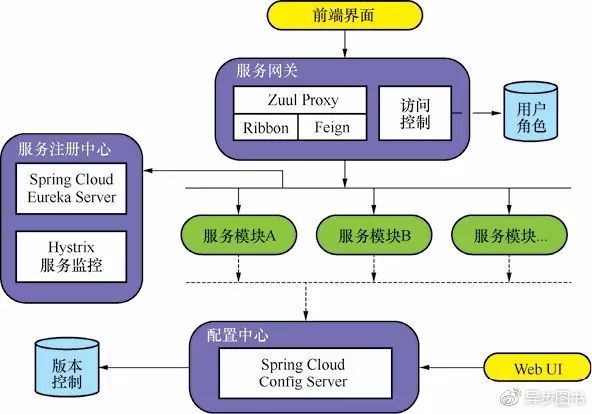
增加增加 springcloud来做微服务 经过讨论暂时不加dubbo这一类msa微服务框架

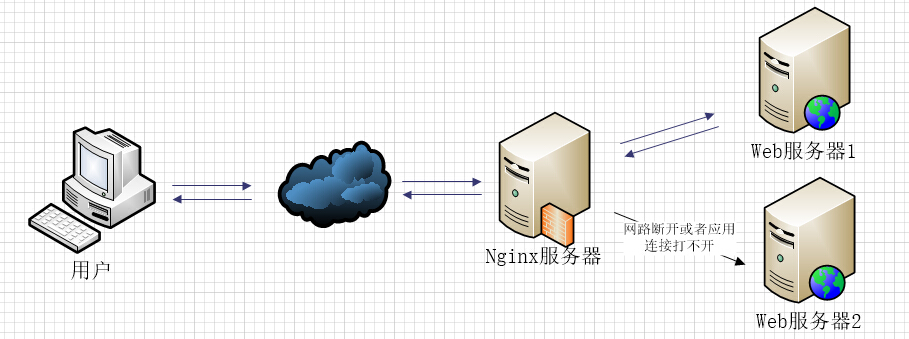
现有的nginx单点或也可以二次做负载均衡，进一步提升性能架构

## Cache 缓存

目前是 Spring cache和redis

可以适当增加些mybatis缓存。。 三重缓存增加更好的性能，以及稳定性。。某个缓存失效还有下一层兜底





## ## database 数据库 +nosql

目前的mysql + nosqldb(redis)

后期可能会加些全文检索 es 和json文档数据库 mongodb 的后备计划

以后也可做数据库集群（读写分离 ，分区等，或者更大型数据库mssql oracle等备用选项 ）

## ## database 数据库 访问接口与数据访问接口

Sql接口

可以给nosql数据库（redis，es，mongodb）增加些简单sql方面api，更加简单的查询语句

以及用来简化查询较复杂的内存数据集合查询（list数组类）

程序api

可以增加些流行的mybatis和jpa等数据库api接口替换以前的 springtemplete数据库 api接口

Rest接口

增加通用的rest接口，特别是mysql，可将其默认的socket接口转换为rest，达到更加易用的目的。。es已经有默认的rest即可，

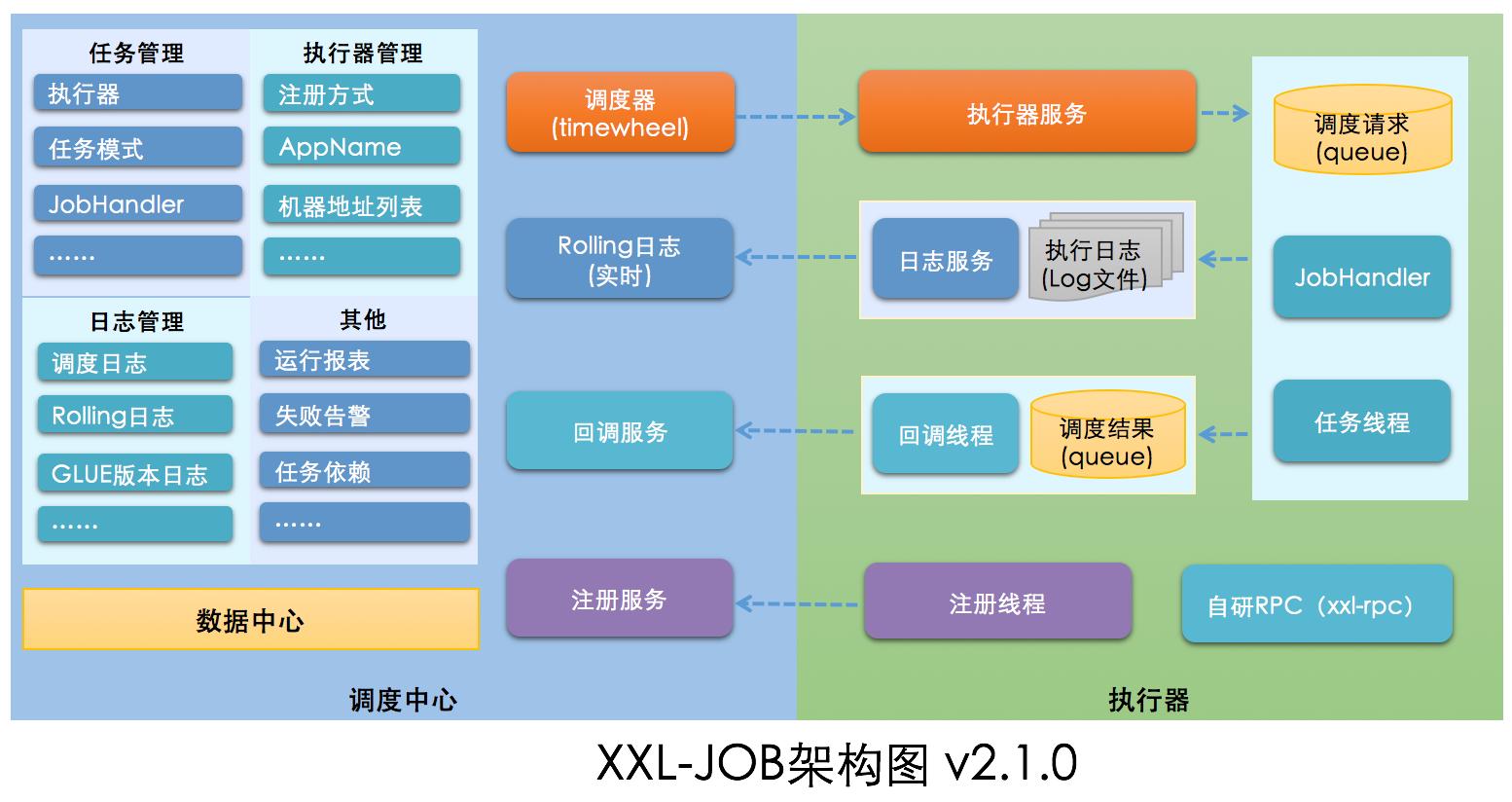
Redis可能也需要增加rest接口。。

## ## 分库分表

自实现

## ## timer定时调度

Xxljob。。。 后期根据需要可以附加linux crontab定时任务 +数据库定时任务



## ## 分库分表

自实现 动态数据源

Mycat sharejdbc Sharding-Proxy可以备选

## 日志 springcloud+elk

 由于微服务架构中每个服务可能分散在不同的服务器上，因此需要一套分布式日志的解决方案。spring-cloud提供了一个

用来trace服务的组件sleuth。它可以通过日志获得服务的依赖关系。基于sleuth，可以通过现有的日志工具实现分布式日志的采集。

这里使用的是ELK，也就是elasticsearch、logstash、kibana。

## sso联合登陆

## 数据同步触发机制

**不管什么数据库，实时数据同步工具，都是把自己模拟成一个从库，进行数据拉取和解析。** 具体来说，mysql是通过binlog进行同步；postgresql使用wal日志进行同步。

## 监控系统 总控

日志类监控

各类程序运行状态的监控，总控平台，ui集成