## 项目架构相关知识：

主流互联网应用的技术特点：

分布式 、高并发、集群、负载均衡、高可用。

分布式：一件事情拆开来做。

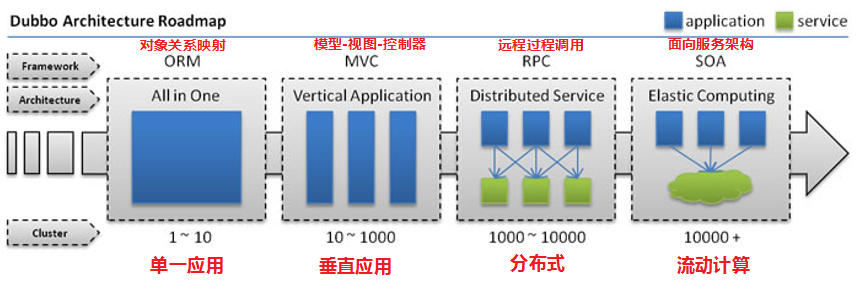
集群：一件事情大家一起做。

负载均衡：将请求平均分配到不同的服务器中，达到均衡的目的。

高并发：同一时刻，处理同一件事情的处理能力（解决方案：分布式、集群、负载均衡）

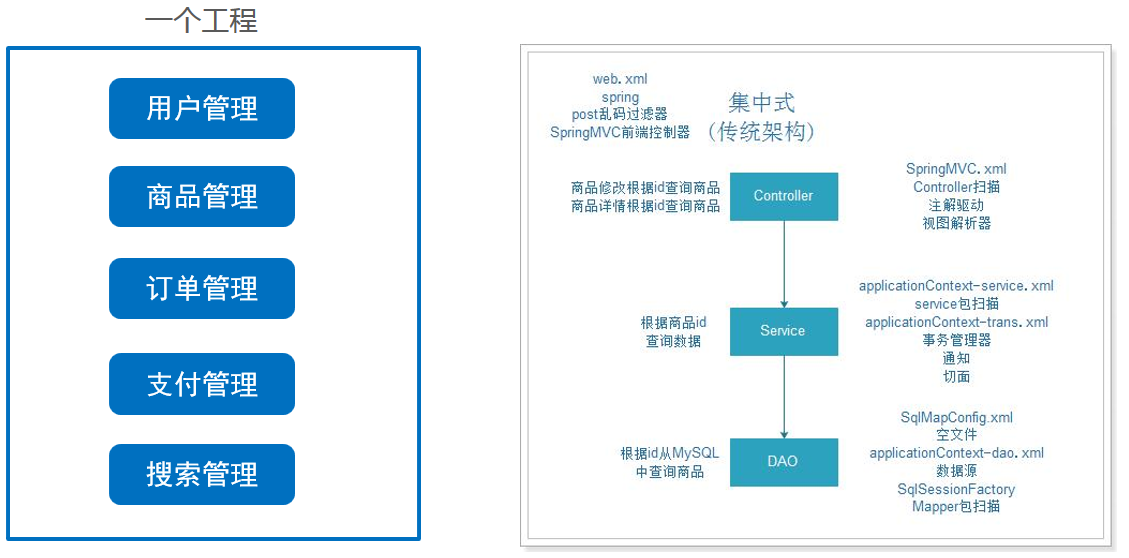
高可用：系统都是可用的。

### 应用架构的演进过程



### 单一应用架构（all in one）

当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。此时，用于简化增删改查工作量的数据访问框架(ORM)是关键。



架构优点：

架构简单，前期开发成本低、开发周期短，适合小型项目（OA、CRM、ERP）。

架构缺点：

全部功能集成在一个工程中

（1）业务代码耦合度高，不易维护。

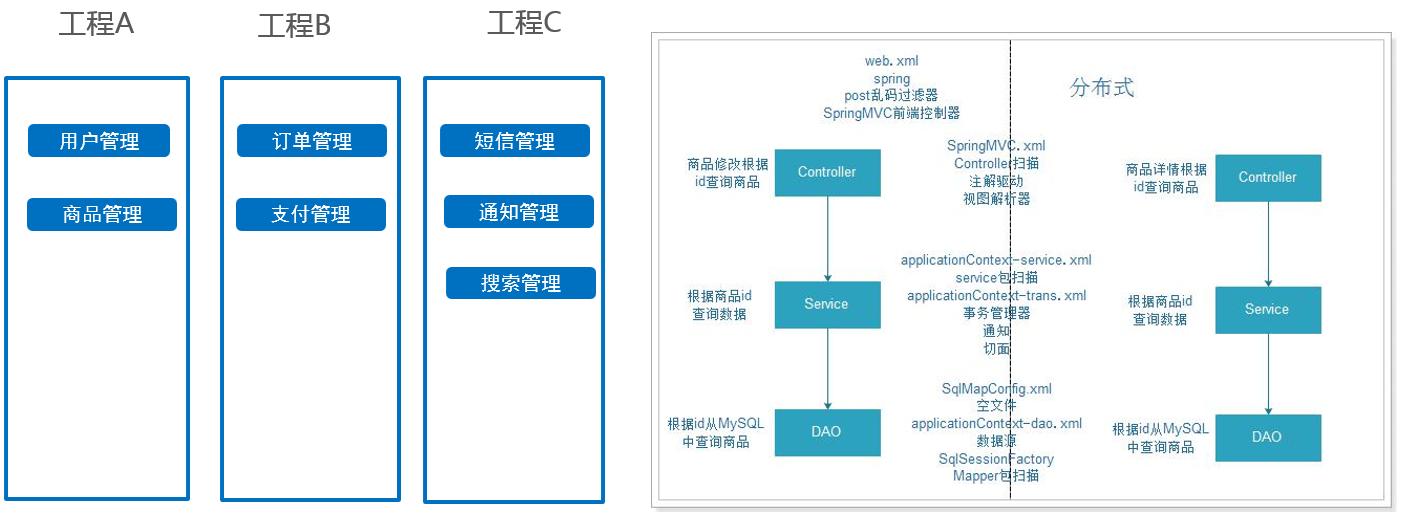
（2）维护成本高，不易拓展

（3）并发量大，不易解决

（4）技术栈受限，只能使用一种语言开发。

### 垂直应用架构

当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成互不相干的几个应用，以提升效率。此时，用于加速前端页面开发的Web框架(MVC)是关键。



- 架构优点：

（1）业务代码相对解耦

（2）维护成本相对易于拓展（修改一个功能，可以直接修改一个项目，单独部署）

（3）并发量大相对易于解决（搭建集群）

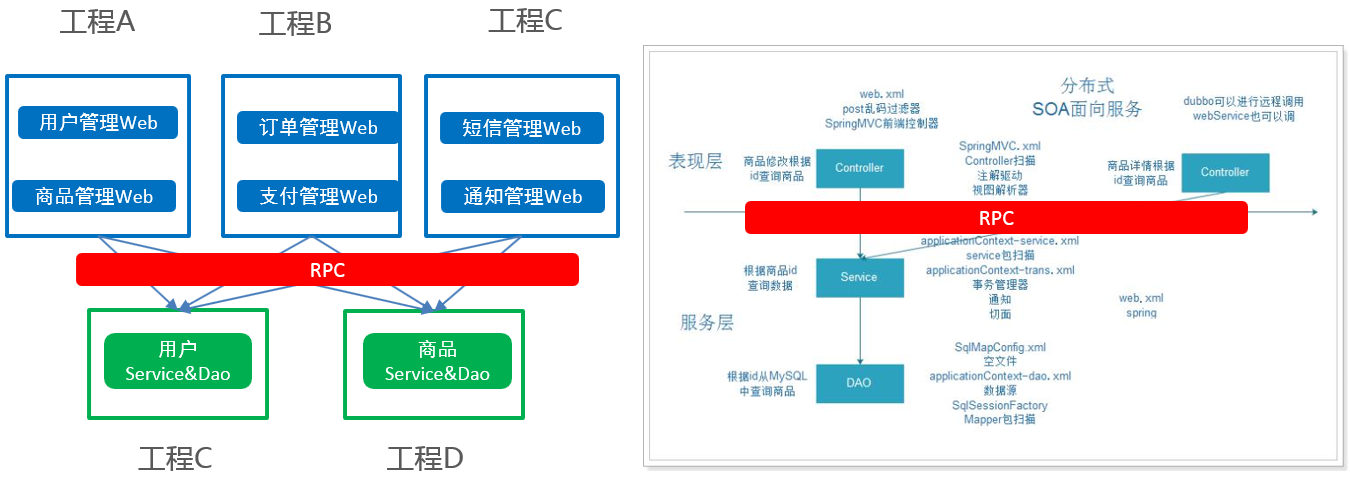
（4）技术栈可扩展（不同的系统可以用不同的编程语言编写）。

- 架构缺点：

代码之间存在数据、方法的冗余

### 分布式服务架构

当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，使前端应用能更快速的响应多变的市场需求。此时，用于提高业务复用及整合的分布式服务框架(RPC)是关键。



架构优点：

（1）业务代码完全解耦，并可实现通用

（2）维护成本易于拓展（修改一个功能，可以直接修改一个项目，单独部署）

（3）并发量大易于解决（搭建集群）

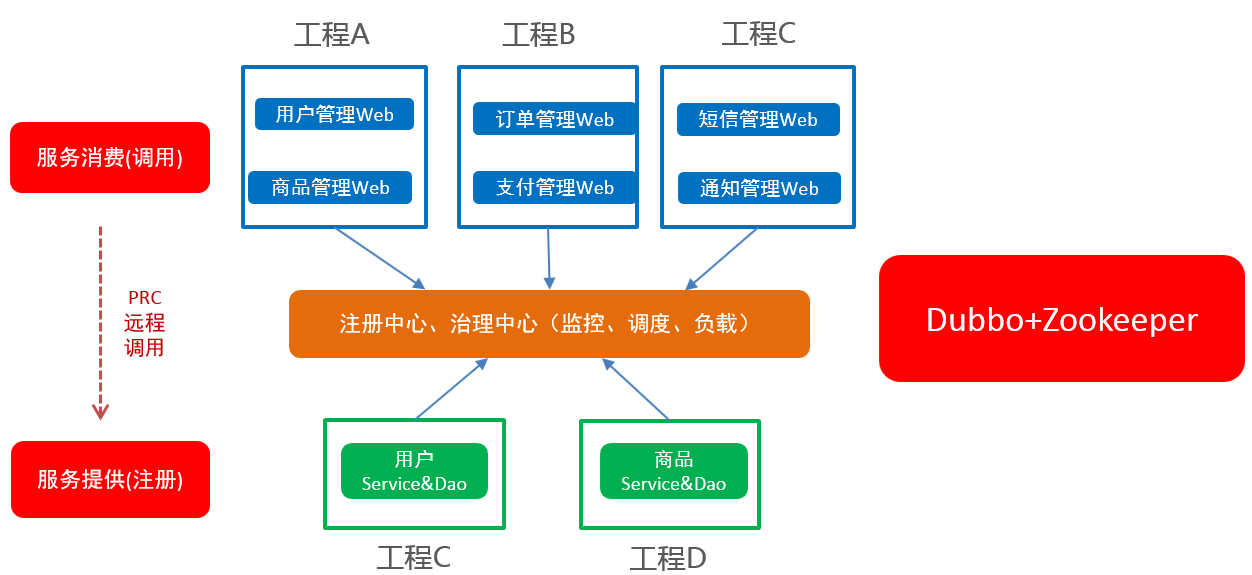
（4）技术栈完全扩展（不同的系统可以用不同的编程语言编写）。

架构缺点：

缺少统一管理资源调度的框架

### 流动计算架构（SOA）

当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问压力实时管理集群容量，提高集群利用率。此时，用于提高机器利用率的资源调度和治理中心(SOA)是关键。



资源调度和治理中心的框架：dubbo+zookeeper

架构优点：

（1）业务代码完全解耦，并可实现通用

（2）维护成本易于拓展（修改一个功能，可以直接修改一个项目，单独部署）

（3）并发量大易于解决（搭建集群）

（4）技术栈完全扩展（不同的系统可以用不同的编程语言编写）。

### 架构知识总结

1：单体架构

全部功能集中在一个项目内（All in one）。

2：垂直架构

按照业务进行切割，形成小的单体项目。

3：SOA架构

面向服务的架构（SOA）是一个组件模型,全称为：Service-Oriented Architecture，它将应用程序的不同功能单元（称为服务）进行拆分，并通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互.

可以使用dubbo作为调度的工具（RPC协议）

4：微服务架构

将系统服务层完全独立出来，抽取为一个一个的微服务。

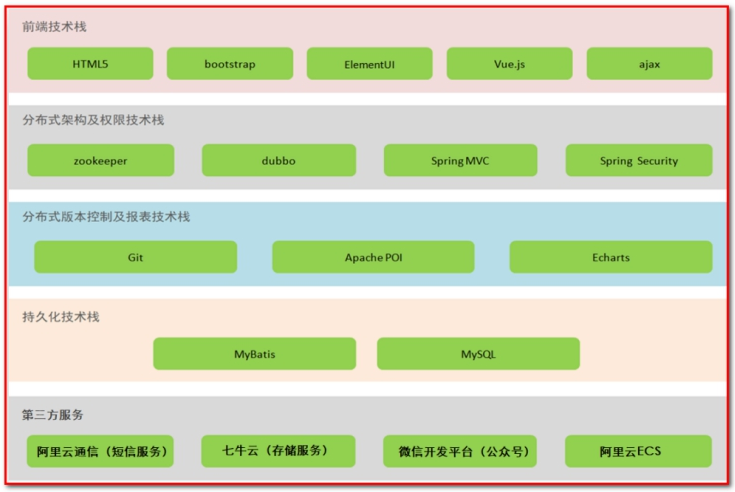
特点一：抽取的粒度更细，遵循单一原则，数据可以在服务之间完成数据传输（一般使用restful请求调用资源）。

特点二： 采用轻量级框架协议传输。（可以使用springcloudy）（http协议）

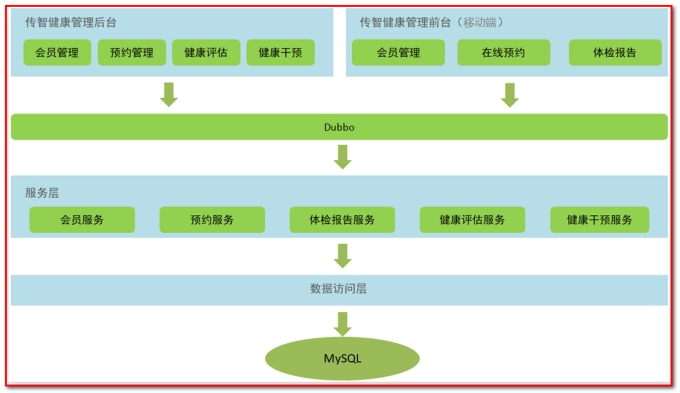
特点三： 每个服务都使用不同的数据库，完全独立和解耦。

## IMC项目概述

1. 项目介绍
2. 技术架构

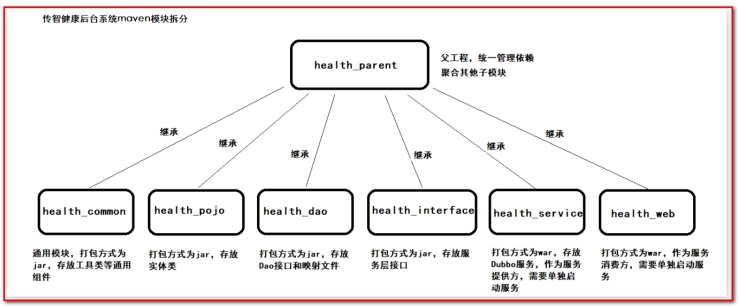


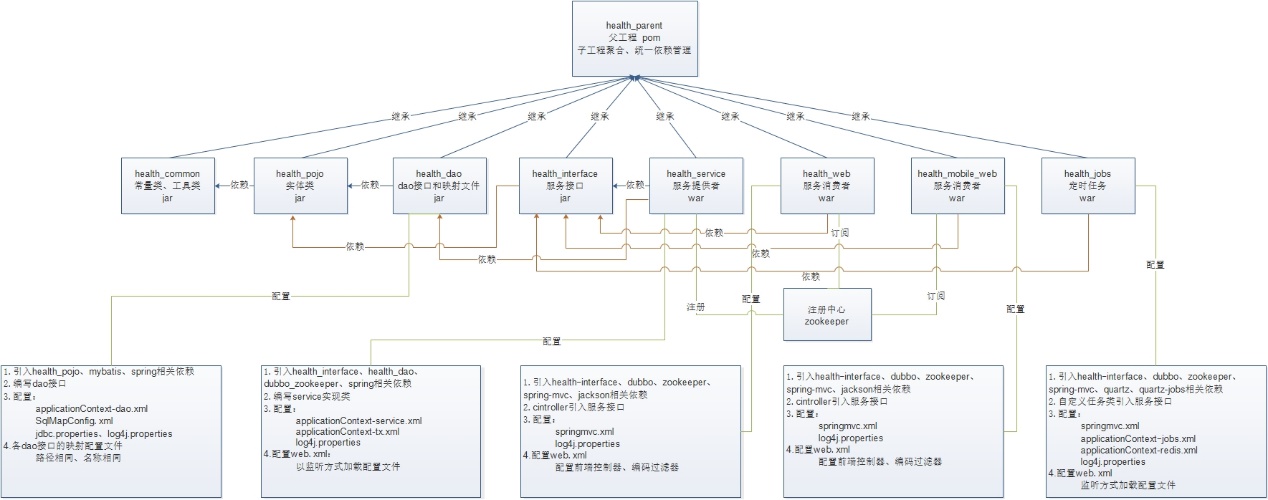
1. 功能架构



1. 项目结构

本项目采用maven分模块开发方式，即对整个项目拆分为几个maven工程，每个maven工程存放特定的一类代码，具体如下：





各模块职责定位：

health\_parent：父工程，打包方式为pom，统一锁定依赖的版本，同时聚合其他子模块便于统一执行maven命令

health\_common：通用模块，打包方式为jar，存放项目中使用到的一些工具类和常量类

health\_pojo：打包方式为jar，存放实体类和返回结果类等

health\_dao：持久层模块，打包方式为jar，存放Dao接口和Mapper映射文件等

health\_interface：打包方式为jar，存放服务接口

health\_service：Dubbo服务模块，打包方式为war，存放服务实现类，作为服务提供方，需要部署到tomcat运行

health\_web：打包方式为war，作为Dubbo服务消费方，存放Controller、HTML页面、js、css、spring配置文件等，需要部署到tomcat运行

SWAGGEr文档的使用:

+ 安装nodejs

+ 设置HS\_HOME 环境变量

HS\_HOME = F:\courses\changgou\资料\swagger\swagger-ui-master

+ 设置Path路径: 添加以下的路径

%HS\_HOME%\node\_modules\.bin

+ cd到F:\courses\changgou\资料\swagger\swagger-ui-master

+ 执行命令

hs -p 500

+ 浏览器中访问:

<http://127.0.0.1:500/dist/index.html>

+ changgou-parent 管理所有的版本统一 pom

+changgou-common 工具类 jar

+changgou-common-db 微服务使用到的jar

+changgou-service 聚合工程 管理微服务工程 pom

+ changgou-service-goods

+ changgou-service-order 依赖于API

+ changgou-service-user 依赖于api

+changgou-service-api 聚合工程 统一管理fegin pojo pom

+changgou-service-goods-api (有feign 有POJO) jar

+changgou-service-order-api(有feign 有POJO) JAR

+changgou-gateway 聚合工程 统一管理网关系统

+ changgou-gateway-admin 后台相关

+ changgou-gateway-web 前端相关

+ ....

+changgou-euereka-server

+changgou-web 聚合工程 统一管理web相关的微服务

+changgou-web-item

+changgou-web-manager: 调用订单微服务调用商品微服务 调用用户微服务聚合数据统一返回给页面。

+....

+..

. 