Matrix Distributed Framework

架构设计说明书

作 者：杨成琳

版 本：V1.0.0.1

电子邮箱：[794867067@qq.com](mailto:794867067@qq.com)

GitHub ：https://github.com/PowerYangcl

前言

Matrix Distributed Framework(下文将简称为：Matrix)，可以理解为矩阵分布式架构运行环境。这套代码是基于阿里巴巴的开源框架Dubbo，再次高度封装后的核心运行时底层环境。这个框架名字来自一个科学命题和一部相关的电影《黑客帝国》，Matrix也是电影中矩阵的意思。奥地利数学家哥德尔在1931年发表了题为《论<数学原理>及有关系统的形式不可判定命题》的论文，其中提出这样一个观点，在任何数学系统中，只要其能包含整数的算术，这个系统的相容性就不可能通过几个基础学派所采用的逻辑原理建立。简单地说，就是在任何矩阵系统中，总有真理是游离于逻辑之外的，这些真理就叫做歌德尔命题。而这些“真理”在我们日常的程序设计中，可以看做是一个个出现的“异常”，发现异常，不断捕捉、不断完善，系统也跟着不断升级；

Matrix使用到的技术有：Spring4.2.6、Mybatis3.4.0、Redis5.0、Hystrix1.5.12、Zookeeper3.4、Dubbo2.6.0、Ehcache2.7.0、Fastjson、Mycat、Druid、Maven和Jsp/Html/Css3/Javascript/Jquery等。Matrix是一个完全的微服务架构解决方案，同时它也是业务线中各个微服务项目的核心底层；Matrix提供了通用的权限系统、多级缓存、API内置网关、独立的文件服务器、内嵌的Dubbo控制台、缓存信息路由、RocketMq消息队列，Hystrix服务熔断器、分布式定时任务等等。

* **权限系统**

复杂的主子权限系统，由总线控制台(Leader)控制，子系统还可以再次分配权限、角色，类似于淘宝中的商铺；还有数据权限，比如组织机构

* **多级缓存**：

系统一级缓存使用Ecache，二级缓存使用Redis；包含了Redis防雪崩设计、规则制定等等。这种优化后的缓存解决方案，一级缓存的数据读取速度可以在200纳秒以下，是使用单一传统Redis集群作为缓存解决方案的3万倍到10万倍。1毫秒(ms)=100,0000纳秒(ns)，服务器从Redis读取数据返回给服务器通常在6毫秒到20毫秒之间。这里给出的200纳秒以下，只是悲观值，当前最新的DDR4内存条的读取速度是2.8纳秒以下，所以硬件如果足够好，效率能提高到30万倍以上，但大多数初创公司无法承担这么昂贵的硬件成本。

* **内置网关**：

Matrix提供完整的API解决方案，系统内置了API网关相关的功能，包含：API信息树、API所属项目、单个API对指定域名跨域、API签名验签、API整体项目跨域、完整的接口测试、第三方请求者信息管理等等。

* **文件服务器**：

支持jpeg,jpg,png,bmp,docx,doc,xlsx,xls,pdf,html,zip,rar,ppt,pptx,csv,json等常见文件的上传与访问

* **Dubbo控制台**：

系统迁移了Dubbokeeper的控制台项目，并且根据自身具体需求进行了重构。

* **缓存信息路由**：

配合Dubbo控制台使用

* **服务熔断**：

将Hystrix服务熔断器与Dubbo整合

* **分布式定时任务**：

基于quartz的分布式定时任务，包含配置界面、调度优化等等；在quartz的基础上进行了大量封装与二次开发，以此来满足企业的高可用需求

* **消息队列**：

系统使用RocketMq作为消息队列。

Matrix项目发起自2015年底，至今为止3年多，期间经历过10次以上的推翻式重构。Matrix诞生于互联网大潮之中，最初的目的是提供一个通用的企业服务总线的核心，让他像一个发动机一样，源源不断的为每一个业务线提供动力；从而大量节省程序设计者的开发时间，让公司的软件系统形成一个**生态环境**。对于很多非IT行业的人来说(俗话中的外行人)，觉得软件是个很简单的东西，只要我找到一堆程序员，再找2个产品，2个美工，几个测试人员；让他们把我的想法写成一个网站或写成一个SaaS (SaaS: Software-as-a-Service软件即服务)平台，外加安卓和IOS客户端，只要在创业老板正确的商业方向上前进，我们的公司就能迎来天使的各种A、B、C轮融资，直到上市。但现实中真实的情况远远不是这么简单的事。软件是一套工程，大学中单独有一个专业，名字就叫“软件工程”，与一个工程相关的，最容易让人联想到的是“复杂”，软件工程也有一套成体系的理论模型。各个行业的工程两个字都衍生自建筑行业，盖一栋大楼，需要详细的设计规划，包括：地基多深、给水排水、供暖、通风、楼内电网等等。软件工程相比较建筑工程，给水排水这些东西都是能看得到的，软件工程中还有很多很多是肉眼看不到的，这些才是最难的地方。中国是房地产大国，但却只是个软件小国。虽然中国有阿里、华为这样逆天的软件公司，但相比较西欧和北美的同行，中国并没有什么特别拿的出手的东西，比如操作系统。中国软件行业的情况更多的是在做外包，而中国的培训机构大多也是在做这件事：培训做外包的人。也就是说：真正能够达到设计者水平的人，在国内并不多。更多的是这样的人：你交给我任务，我照猫画虎，能完成任务。至于后面需要把老虎改成狮群，对不起，您提的需求改不了。

很多创业型的公司，或者大型集团新成立的事业部，急着招了一波人，加班加点按照老板的需求，用SpringMVC或者SpringBoot搭了一个项目，有后台有接口还有客户端。老板看到表面功能都按自己想法实现了，心里很高兴；而且老板的商业方向也是对的，一经推广在市场上很受欢迎！创业者的心里肯定十分兴奋，心中有成千上万的Idea等着去做、去扩展、去铺开！但是，往往系统不是您想改，想改就能改的了。2014年到2015年是互联网创业公司的大潮，涌现了各种各样的互联网公司。他们创业之初都是信心满满，的确每个创业者的商业方向其实都是对的。大家对于成功的渴望，让一个个的开发团队如雨后春笋般的出现。以团购网站为例，当时出现了很多比如：美团网、拉手网、窝窝团、高朋团购、团美网、58团购、大众点评，还有很多不知名的。但到今天，只有美团和饿了么留到了最后。这些失败的公司有自己运营的问题，但更多的是他们软件系统跟不上产品的迭代。进入A轮以后，投资人的热钱已经进来，他们有更多的想法需要你的平台来实现，但是公司的项目组却迟迟拿不出规划好的新功能，渐渐的也就被资本市场边缘化，最后倒闭关门。原因无外乎如下6点：

1、公司没有成熟的软件生态环境。仓促间组建了一个项目开发团队，一切从0开始；一切软件工程方法论、项目管理措施均不适用。

2、项目仓促开展，产品设计并不完善，底层结构不合理，但代码已经写成，很难再改。

3、由于开发进度过快，各个模块耦合性极高，代码质量很差，几乎谈不上扩展性，新的产品需求迭代、变更无从谈起。

4、加班加点，工程师身体和精神两个层面过度疲劳，从人性角度看，很难控制自身代码质量；高度的工作压力下，完成任务为第一要务，其他都会忽略。

5、工程师能力水平参差不齐，代码风格各异，没有统一的约束(至少从框架代码上看)和指挥，更像是一群乌合之众。

6、人员流动。这是一个很严重的问题，如果这个离职的工程师，代码质量极高，注释丰富，结构设计合理，那么情况还好。否则对项目组简直就是灾难。

这些是市场上99%“互联网公司”的通病，也是这些公司无法在资本市场崛起的原因。对比大公司，初创公司在人才储备上已经处于很大的劣势，技术框架上更没法与大公司相比，那么能带来优势的也就只有创业Leader的商业方向了。其实，在绝大多数看上去还不错，挺光鲜亮丽的公司中，他们的软件系统也是千疮百孔，产品的更新迭代也是难以为继，光是权限设计就能难倒一大片公司。因为系统在设计之初就没有规划，自然也就不成体系；到了后期有人想去改代码，但是高度耦合的代码会让所有人失去改的勇气，因为改好了是你应该做的，改不好那就是你能力不行，是你的责任。慢慢的破窗效应在项目中出现，一个恶性循环也就开始了。

Matrix正是为了解决这个问题而设计，让您的软件系统形成生态环境。只有形成这样的软件生态环境，软件工程管理的方法论、瀑布模型、敏捷开发模型才能生效，否则无论项目经理怎么对项目开发进行排期，项目都会严重延期，因为什么基础条件都没有，开发的工程师自然也无法高效的推进手中的任务进度。使用Matrix作为您公司软件架构的底层后，会为您的IT团队带来如下好处：

1、节省大量的底层技术的预研时间。Matrix对大型分布式系统常用的技术做了高度的抽象和封装，在多个项目中已经使用，技术解决方案成熟、稳定、可靠。

2、业务快速开发。系统提供完善的分页(灵活性高，扩展性强)、多系统-主子权限(权限模块独立且具备通用性)、开发者示例、弹窗等等功能。

3、完善的restful api接口功能，灵活、独立，支持Https。

4、软件项目变的可控，能够按照预定的成本、进度、质量顺利完成，项目经理可以对人员(People)、产品(Product)、过程(Process)和项目(Project)进行分析和管理。

5、开发人员不用天天加班，跳出恶性循环，显著降低人员流动性。他们只需要完成简单的增删改查即可；更多的去关注自己的业务逻辑，无需在深入底层框架的开发。

6、大量缩短项目开发周期。屏蔽底层技术难点，让一切变得简单。

7、项目弹性高。业务项目继承Matrix的模块化设计也会体现出高内聚、低耦合的特点。当我们一个业物不需要了，直接删掉这部分项目的代码，不会影响其他项目的运行。

8、项目灵活性高、扩展性强。基于Maven，以及路径同源性的特点，让项目高度灵活，扩展性极强。

9、技术框架比肩中大型互联网公司。Matrix至今为止经历过10个以上大型分布式项目的验证；大量的核心底层技术在实际场景中被抽象、封装和沉淀下来。

如果您现在的情况是创业公司、上市公司新成立的事业部、大型集团的子公司新成立的IT事业部或者现在的部门要重构所有的代码，那么您可以选择Matrix作为技术框架的底层，这样可以直接提升开发团队的技术积累水平、项目微服务化+业务分离、节约大量的科研投入时间、极大的缩短项目开发周期。尤其是节约的时间和被缩短的项目开发周期将会让您的公司在残酷的市场竞争中取得绝对优势；因为您的技术积累要优于同行，产品的开发、迭代速度还比他们快，市场形势一旦有变化您就可以拿出应对的产品功能去适应市场竞争。虽然最终能不能赢到最后会有很多外在因素限制，但是成功的机会会显著提升。也就是说，真正影响公司最终走向的变成了您的商业方向和运营管理方式，技术这块您看不到的“坑”都会让Matrix填平，技术部门不会成为公司成长路线上掣肘的那一个，而是变成了您最得力的助手，他会火力全开，成为您的急先锋。

Matrix适用于如下商业方向：大数据平台、区块链、电商平台(规模类似于唯品会、U购物、国美、有赞)、互联网SaaS平台、教育行业、移动医疗、新闻、网约车。不适用于人工智能、视频网站；这两个行业没有涉足过，无法保证一定可用。其中大数据平台、电商、互联网SaaS、移动医疗和网约车5个行业，有真实案例。