零基础构建自己的服务治理框架

关于我

自我介绍

我是周梦康，6年的后端开发。
常用语言 `php` 、 `java` ，《深入 PHP 内核》作者之一。
目前就职于阿里云。

博客	**https://mengkang.net**
github	https://github.com/zhoumengkang
微博	http://weibo.com/zmkang
微信	zhoumengkang
邮箱	zhoumengkang@php.net

本次分享的视频讲解地址（扫描二维码即可查看）

为什么需要服务治理

远程调用的实现

跨语言通信的实现

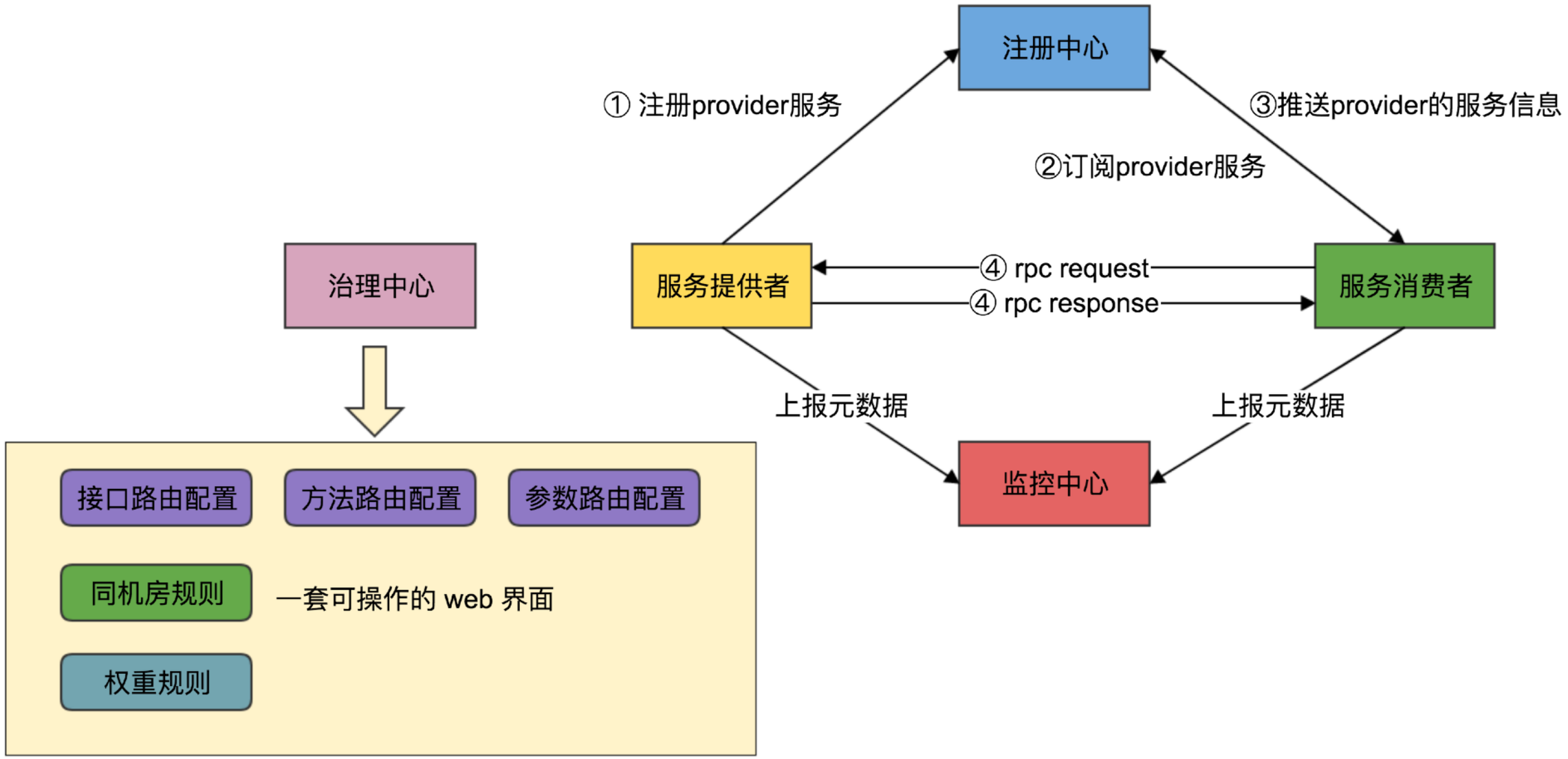
注册中心的实现

监控中心的实现

多维度的服务治理

RPC 协议的优化

服务治理框架图

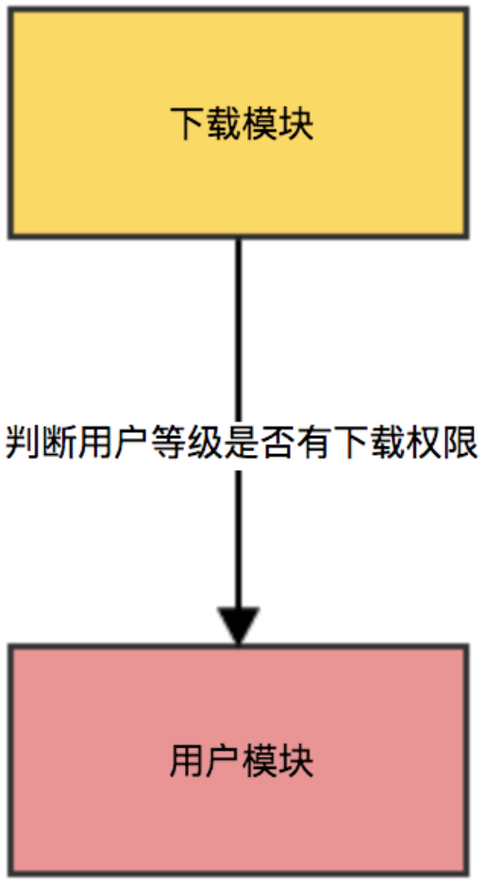

注册中心
① 注册provider服务
③推送provider的服务信息
②订阅provider服务
治理中心
服务提供者
④ rpc request
④ rpc response
服务消费者
上报元数据
上报元数据
监控中心
接口路由配置
方法路由配置
参数路由配置
同机房规则
一套可操作的 web 界面
权重规则

为什么要使用服务治理?

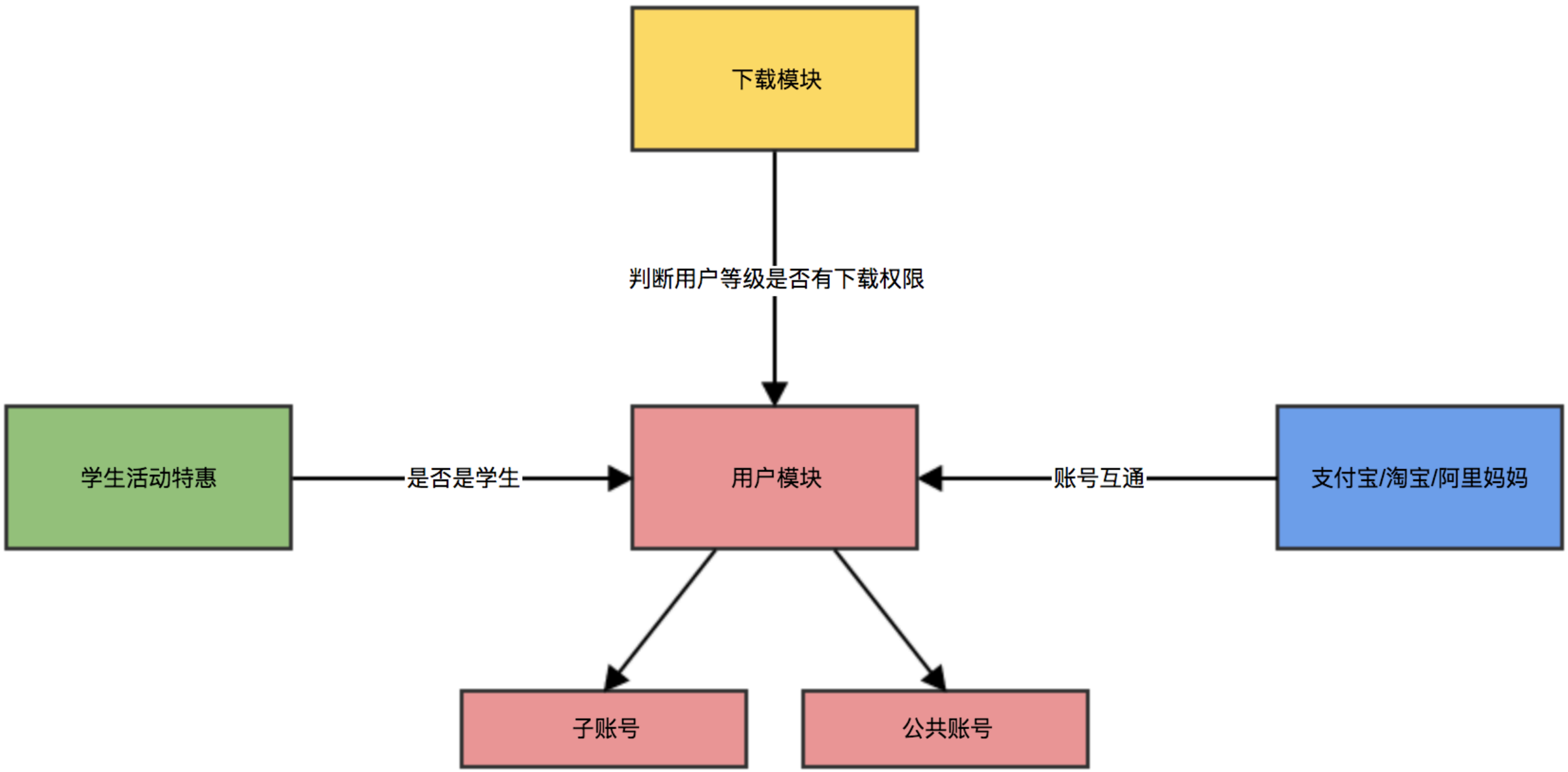
原本很简单的事情为什么要弄这么复杂?
服务治理能提高性能吗?
问题追踪更加复杂了，怎么解决?
我们学习这堂课之后都要切换到服务治理的架构方式上来吗?

前期：编写模块化代码

业务举例：根据用户等级下载附件



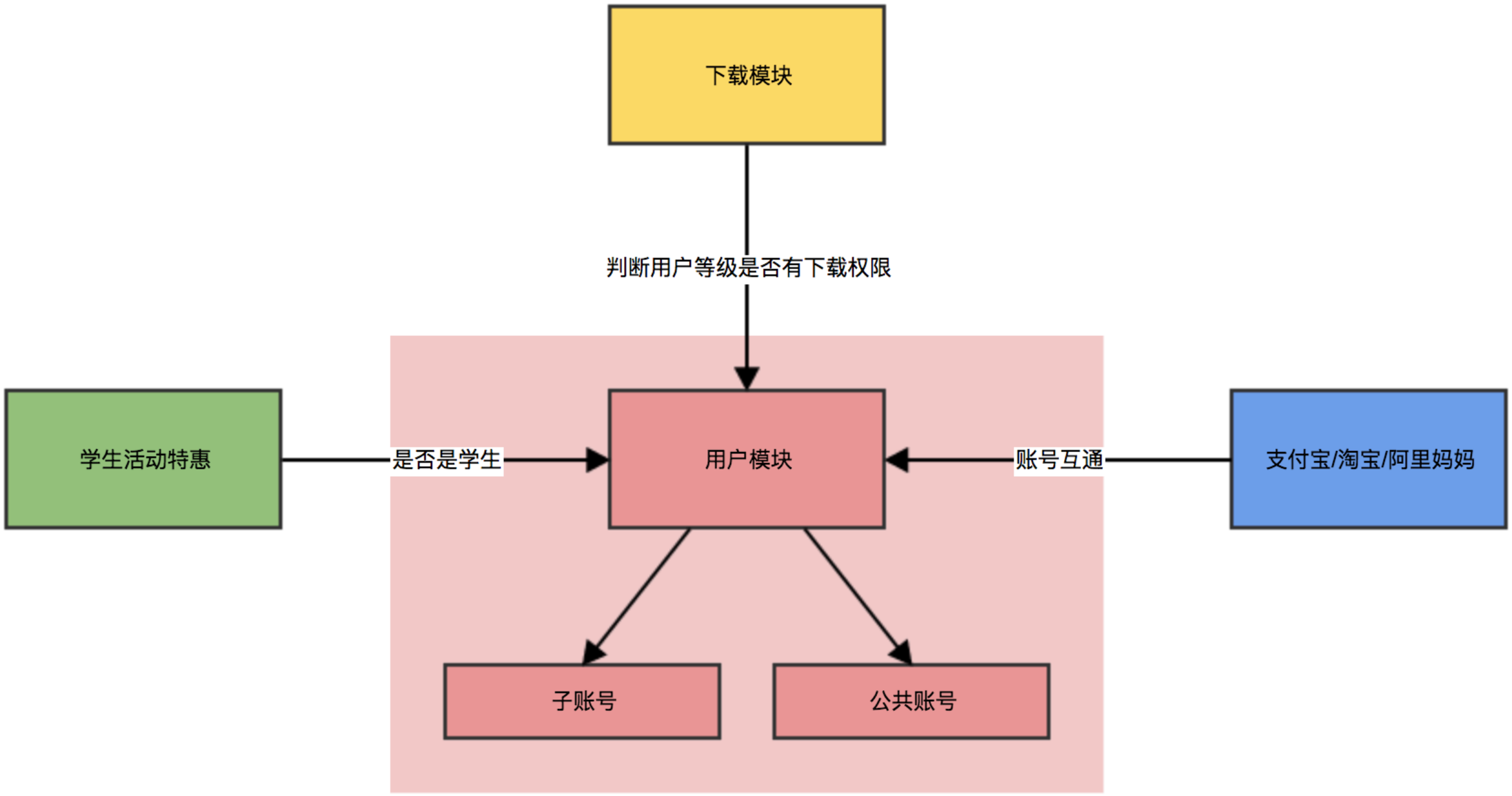
前期：编写模块化代码


下载模块
判断用户等级是否有下载权限
学生活动特惠
是否是学生
用户模块
账号互通
支付宝/淘宝/阿里妈妈
子账号
公共账号

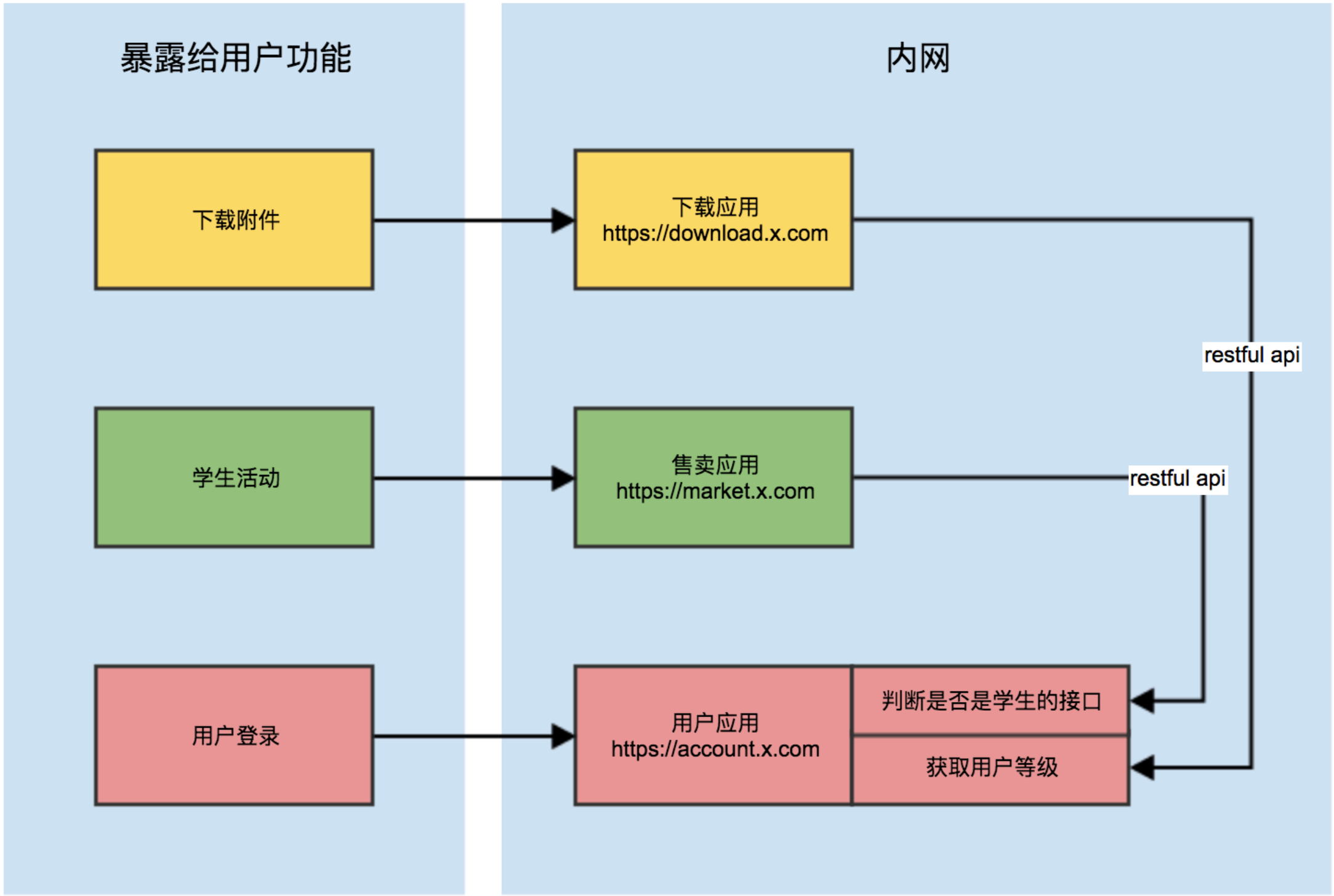
指导思想

高内聚低耦合

中期：服务拆分


下载模块
判断用户等级是否有下载权限
学生活动特惠
是否是学生
用户模块
账号互通
支付宝/淘宝/阿里妈妈
子账号
公共账号

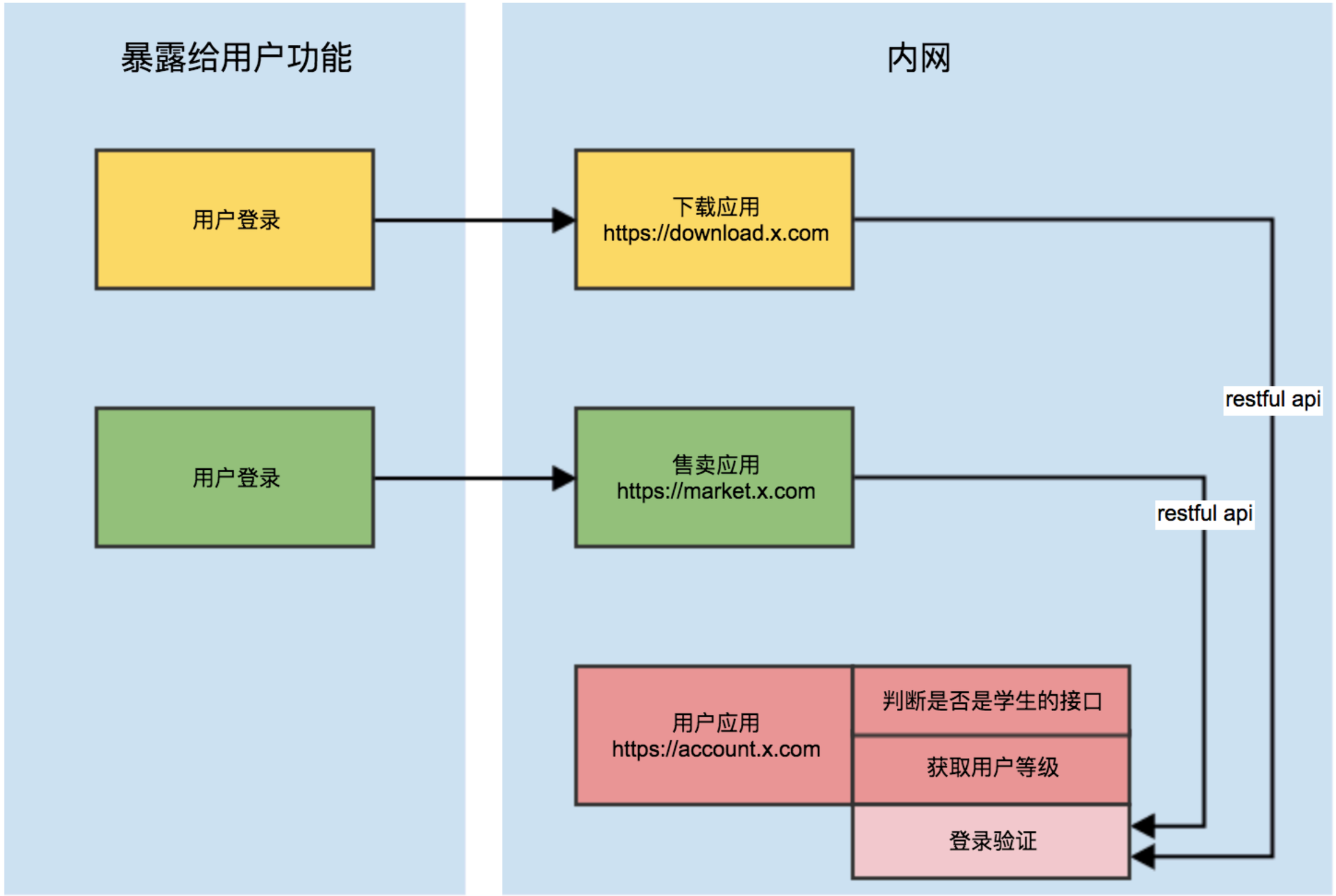
中期：各个独立运用成形


暴露给用户功能
内网
下载附件
下载应用
https://download.x.com
restful api
学生活动
售卖应用
https://market.x.com
restful api
用户登录
用户应用
https://account.x.com
判断是否是学生的接口
获取用户等级

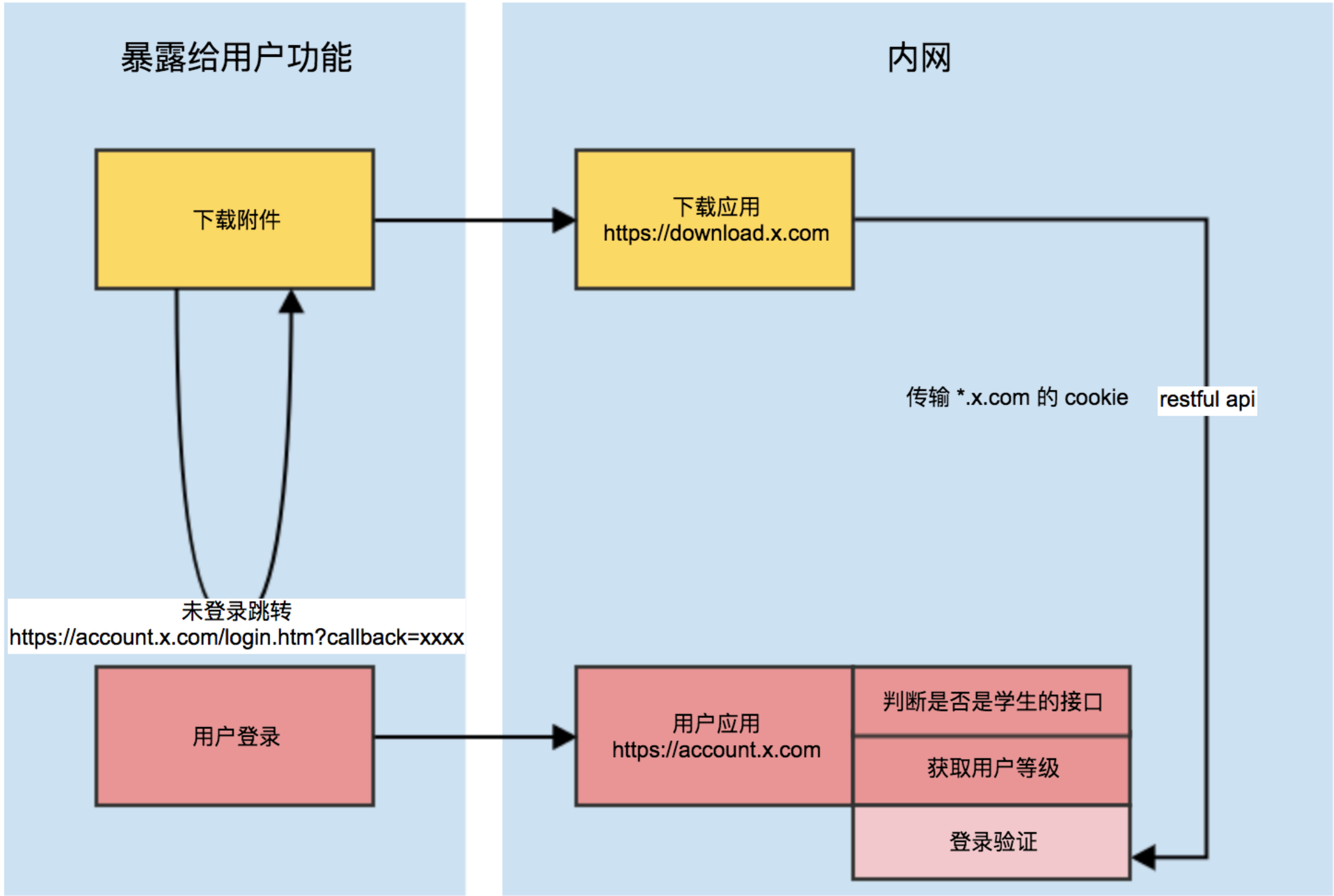
远程调用的实现

为什么登录独立处理?

登录方案1


暴露给用户功能
内网
用户登录
下载应用
https://download.x.com
restful api
用户登录
售卖应用
https://market.x.com
restful api
用户应用
https://account.x.com
判断是否是学生的接口
获取用户等级
登录验证

登录方案2


暴露给用户功能
内网
下载附件
下载应用
https://download.x.com
传输 *.x.com 的 cookie
restful api
未登录跳转
https://account.x.com/login.htm?callback=xxxx
用户登录
用户应用
https://account.x.com
判断是否是学生的接口
获取用户等级
登录验证

梳理用户应用对外的服务

UserService.getUserLevel
UserService.getStudentInfo
UserService.loginAuthCheck

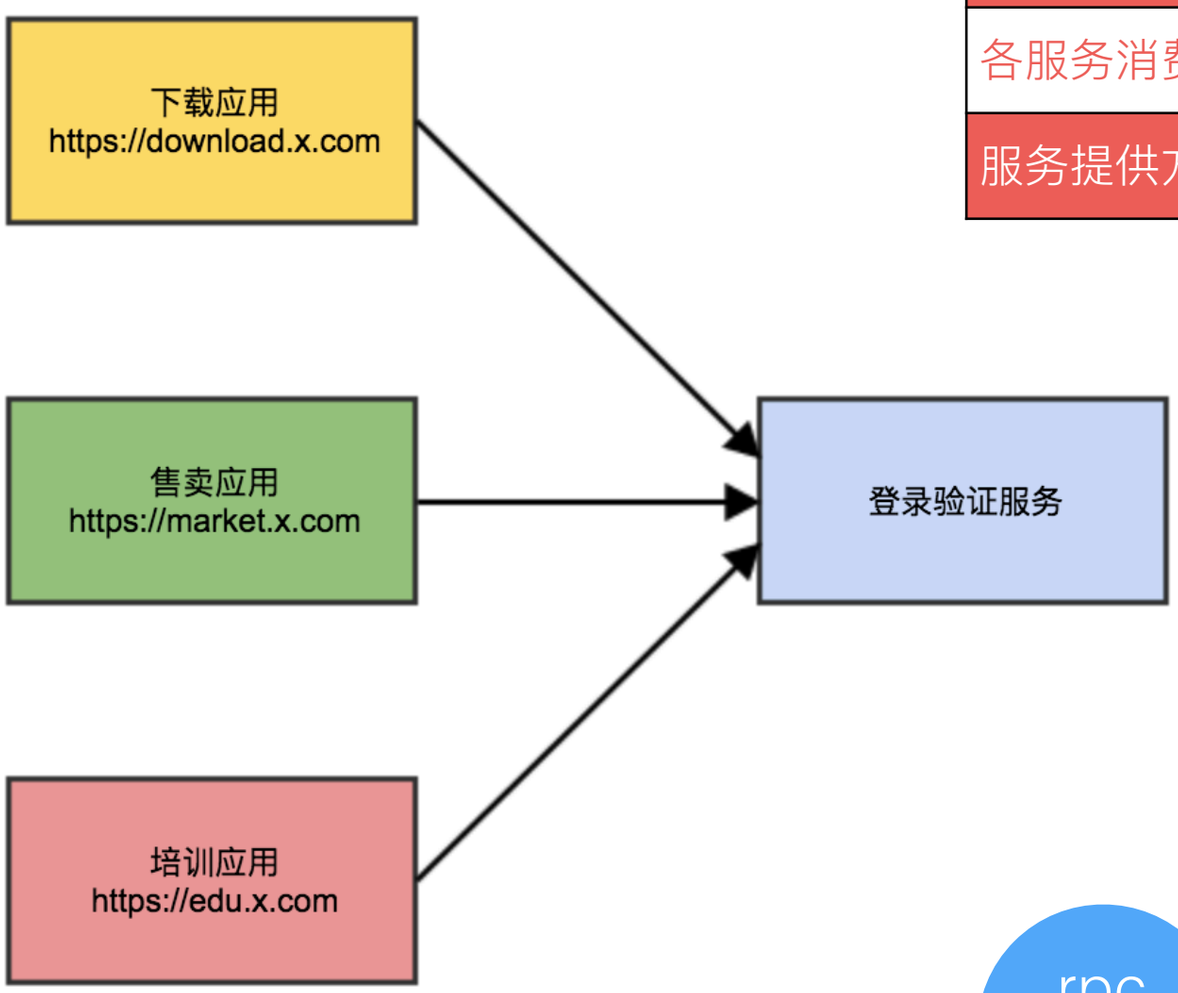
服务治理出现的原因

业务不断拓展，降低服务之间的耦合，从功能上拆分

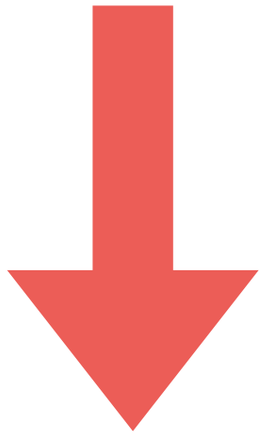
接口协议杂乱无章，沟通成本高

整个分布式系统稳定性保障

以用户登录认证为例



各服务消费方，调用服务的代码都非常类似，能否抽象通用sdk
各服务消费方，如何对自己调用的远程服务可用性进行监控
服务提供方，性能如何保障



rpc
框架

监控
中心

治理
中心

从理论到实践 - 原始版

A 服务器上

```php
$userService = new UserService();
$userService->getUserInfo($uid);
```

B 服务器想调用

```php
$client  = new \SDK\Client();
$request = new \SDK\UserService\Request\GetStudentInfoRequest();
$request->setUid($uid);
$request->setMethod("GET");
$response = $client->doAction($request);
```

从理论到实践 - 变型

```php
class Client
{
    private $url;
    private $service;

    private $rpcConfig = [
        "UserService" => "http://127.0.0.1:8081",
    ];

    public function __construct($service)
    {
        if (array_key_exists($service, $this->rpcConfig)) {
            $this->url = $this->rpcConfig[$service];
            $this->service = $service;
        }
    }

    public function __call($action, $arguments)
    {
        $content = json_encode($arguments);
        $options['http'] = [
            'timeout' => 5,
            'method'  => 'POST',
            'header'  => 'Content-type:application/x-www-form-urlencoded',
            'content' => $content,
        ];

        $context = stream_context_create($options);

        $get = [
            'service' => $this->service,
            'action'  => $action,
        ];

        $url = $this->url . "?" . http_build_query($get);

        $res = file_get_contents($url, false, $context);

        return json_decode($res, true);
    }

}

$userService = new Client('UserService');
var_export($userService->getUserInfo(103));
```

```php
class UserService
{
    public static function getUserInfo($uid)
    {
        // 假设以下内容从数据库取出
        return [
            'id'       => $uid,
            'username' => 'mengkang',
        ];
    }
}

$service = $_GET['service'];
$action = $_GET['action'];
$argv = file_get_contents("php://input");

if (!$service || !$action) {
    die();
}

if ($argv) {
    $argv = json_decode($argv, true);
}


$res = call_user_func_array([$service, $action], $argv);

echo json_encode($res);
```

从理论到实践 - 变型 - 小结

我们做了如下工作
使用 __call 调用了本地不存在的方法
使用了 http 协议进行传输
使用了 json 进行内容的序列化编码（同时 http 本身对 body 体做了二进制编码）

从理论到实践 - 跨语言调用 - java 客户端

第三方给的 sdk

```java
package net.mengkang.sdk;

public class User {
    private Integer id;
    private String username;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                '}';
    }
}
```

```java
package net.mengkang.sdk;

public interface UserService {
    User getUserInfo(Integer uid);
}
```

RPC 调用

```java
package net.mengkang.rpc;

import java.lang.reflect.Proxy;

public class RpcClient {
    public final Object proxy(Class type) {
        RpcClientInvocationHandler handler = new RpcClientInvocationHandler(type);
        return Proxy.newProxyInstance(type.getClassLoader(), new Class[]{type}, handler);
    }
}
```

```java
package net.mengkang.rpc;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONException;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;

class RpcClientInvocationHandler implements InvocationHandler{

    private Class service;

    public RpcClientInvocationHandler(Class clazz) {
        service = clazz;
    }

    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        Class returnType = method.getReturnType();
        String[] serviceName =  service.getName().split("\\.");

        String url = "http://127.0.0.1:8081?service=" + serviceName[serviceName.length - 1] + "&action=" + m

        String httpResponse = RpcRequest.doPost(url, args);

        Object res = null;

        try{
            res = JSON.parseObject(httpResponse, returnType);
        }catch (JSONException e){
            e.printStackTrace();
        }

        return res;
    }
}
```

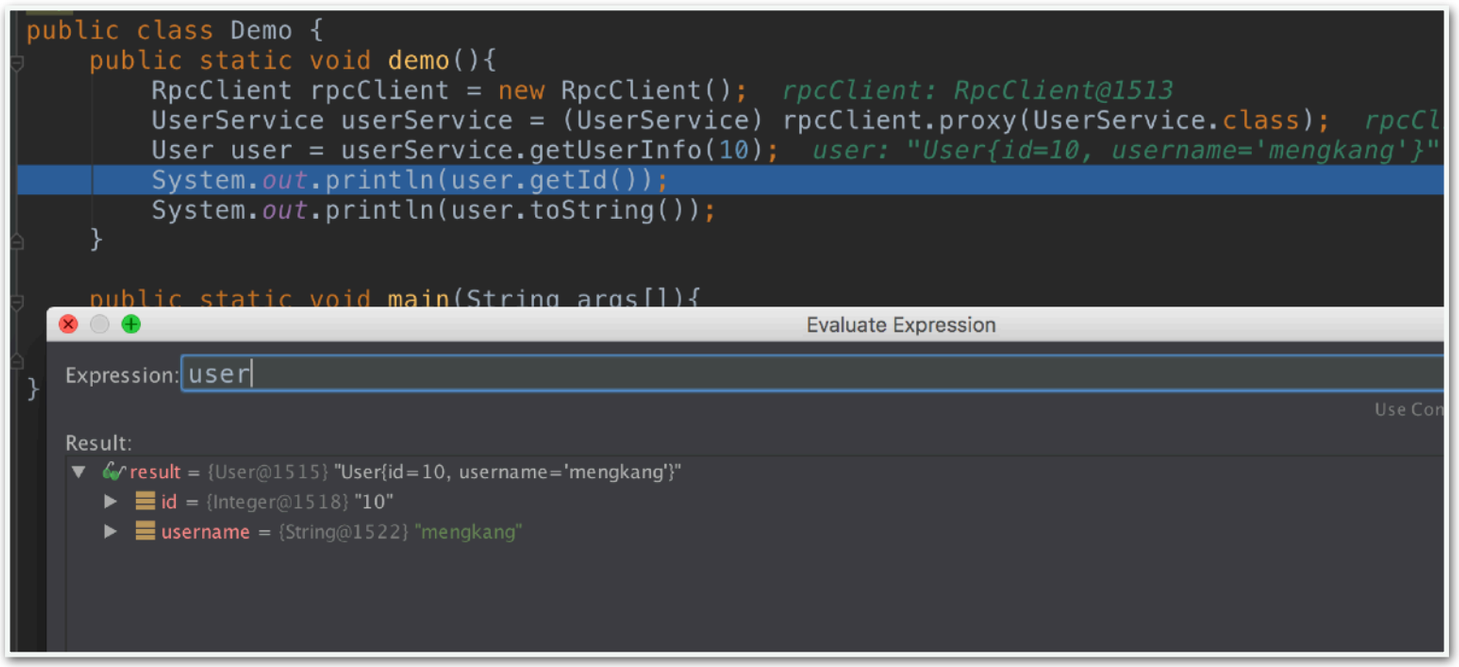
从理论到实践 - 跨语言调用 - java 客户端

```java
package net.mengkang;

import net.mengkang.rpc.RpcClient;
import net.mengkang.sdk.User;
import net.mengkang.sdk.UserService;

public class Demo {
    public static void demo(){
        RpcClient rpcClient = new RpcClient();
        UserService userService = (UserService) rpcClient.proxy(UserService.class);
        User user = userService.getUserInfo(10);
        System.out.println(user.getId());
        System.out.println(user.toString());
    }

    public static void main(String args[]){
        Demo.demo();
    }
}
```



从理论到实践 - Java 小结

Java 的优势

★ 更加本地化、更加透明

★ 比前面 PHP 更进一步，直接返回本地对象，远程无感知

从理论到实践 - 源码地址

PHP 客户端	https://gitee.com/zhoumengkang/soa-01-php-client
Java 客户端	https://gitee.com/zhoumengkang/soa-01-java-client
PHP 服务端	https://gitee.com/zhoumengkang/soa-01-php-service

从概念到实践 - 简单配置中心

服务消费者	https://gitee.com/zhoumengkang/soa-02-all/tree/master/consumer
服务提供者	https://gitee.com/zhoumengkang/soa-02-all/tree/master/provider
注册中心	https://gitee.com/zhoumengkang/soa-02-all/tree/master/config

从概念到实践 - 不足之处

```php
class Client
{
    private $url;
    private $service;

    private $rpcConfig = [
        "UserService" => "http://127.0.0.1:8081",
    ];

    public function __construct($service)
    {
        if (array_key_exists($service, $this->rpcConfig)) {
            $this->url = $this->rpcConfig[$service];
            $this->service = $service;
        }
    }

    public function __call($action, $arguments)
    {
        $content = json_encode($arguments);
        $options['http'] = [
            'timeout' => 5,
            'method'  => 'POST',
            'header'  => 'Content-type:application/x-www-form-urlencoded',
            'content' => $content,
        ];
```

复习 - rpc 的要素

地址	注册中心
接口	基于接口编程（java 里面体现）

rpc 客户端

PRC客户端初始化

服务调度器（软负载）

封包、拆包

request id

特点：全局唯一性

意义：调用链路追踪

协议定义

```
class Request
{
    private $requestId;
    private $token;
    private $service;
    private $action;
    private $parameters;

    //...
}

class Response
{
    private $requestId;
    private $code;
    private $body;

    //***
}
```

自定义 tcp 协议 - 以 swoole 为例

请求体长度	请求体
4字节	123字节
比如存 123	存 xxxxx....

```php
$server = new \swoole_server(SERVER_IP, SERVER_PORT);
$server->set(array(
    'worker_num'            => 1,
    'open_length_check'     => true,        // 开启协议解析
    'package_length_type'   => 'N',         // 长度字段的类型
    'package_length_offset' => 0,           // 第0个字节开始是包长度
    'package_body_offset'   => 4,           // 第4个字节开始计算长度
    'package_max_length'    => 2000000,     // 协议最大长度
));
$server->on('receive', "onReceive");

$server->start();

function onReceive($server, $fd, $from_id, $data){
    $head = unpack('N', substr($data, 0 , 4)); // 获取包头
    $body = substr($data, 4);  // 获取数据
    $data = json_decode($body,true);

    if ($data['action'] == 'configUpdate'){
        include_once dirname(ROOT)."/Rpc/Dispatcher.php";
        Rpc\Dispatcher::configUpdate($data['data']);
    }

    $server->close($fd);
}
```

rpc 并行化请求

```java
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.*;

/**
 * Created by zhoumengkang on 16/12/15.
 */
public class YarConcurrentClient {

    private static ExecutorService executorService;

    static{
        poolInit();
    }

    private static void poolInit(){
        executorService = Executors.newCachedThreadPool();
    }

    public void call() throws ExecutionException, InterruptedException {
        List<Future<String>> result =new ArrayList<Future<String>>();
        for (int i = 0; i < 4; i++) {
            Future<String> future = executorService.submit(new YarClientCallable(i));
            result.add(future);
        }
        for(Future<String> future:result){
            System.out.println("返回值: "+ future.get());
        }

    }

    public class YarClientCallable implements Callable<String> {

        private int seq;

        public YarClientCallable(int seq) {
            this.seq = seq;
        }

        public String call() throws Exception {
            System.out.println(Thread.currentThread().getName());
            Thread.sleep(3000);
            System.out.println("Weak up" + seq);
            return "完成" + seq;
        }
    }

}
```

```
Yar_Concurrent_Client {
    /* 属性 */
    static $_callstack ;
    static $_callback ;
    static $_error_callback ;
    /* 方法 */
    public static int call ( string $uri , string $method , array $parameters [, callable $c
    public static boolean loop ([ callable $callback [, callable $error_callback ]] )
    public static boolean reset ( void )
}
```

参考 https://mengkang.net/596.html

被屏蔽了的点

字节序+字节对齐	https://mengkang.net/1046.html
紧凑型数据结构跨语言的意义	https://mengkang.net/586.html

provider - 通用地址暴露所有接口列表

```php
define('ROOT', dirname(__DIR__));

function getCenter()
{
    return include ROOT . "/config/center.php";
}

function getProvider()
{
    $config = include ROOT . "/config/provider.php";
    $list = array_keys($config['services']);

    $res = [];
    foreach ($list as $service) {
        include ROOT . "/services/" . $service . ".php";
        $class = new ReflectionClass($service);
        $methods = $class->getMethods(ReflectionMethod::IS_STATIC
            | ReflectionMethod::IS_PUBLIC);

        foreach ($methods as $item) {
            $params = [];
            foreach ($item->getParameters() as $parameter) {
                $params[] = $parameter->getName();
            }
            $res[$service][] = [
                'method' => $item->name,
                'params' => $params,
            ];
        }
    }
    return $res;
}

function getConsumer()
{
    if (file_exists(ROOT . "/config/consumer.php")) {
        return include ROOT . "/config/consumer.php";
    }
    return [];
}

$res = [
    'center' => getCenter(),
    'provider' => getProvider(),
    'consumer' => getConsumer()
];

echo json_encode($res);
```

```
// 20171029101846
// http://localhost:8081/

{
  "center": {
    "ip": "0.0.0.1",
    "port": 10000
  },
  "provider": {
    "UserService": [
      {
        "method": "getUserInfo",
        "params": [
          "uid"
        ]
      },
      {
        "method": "isStudent",
        "params": [
          "uid"
        ]
      },
      {
        "method": "updateSchoolInfo",
        "params": [
          "uid",
          "school"
        ]
      }
    ]
  },
  "consumer": [
    "MarketService",
    "ScoreService",
    "MessageService"
  ]
}
```

服务查询

我做的demo http://localhost:8082/

信息的同步

推

网站私信

拉

不活跃用户feed 流

拉的弊端：权重修改，什么时候拉

服务查询

控制台

- 服务查询
- 服务报表
- 服务测试
- 服务治理
  - 权重规则
  - 机房规则
  - 服务鉴权
  - 服务路由
  - 服务归组

具体信息

应用名	xxx
所属人	周梦康
开发负责人	周梦康

服务器列表

服务提供者　服务消费者

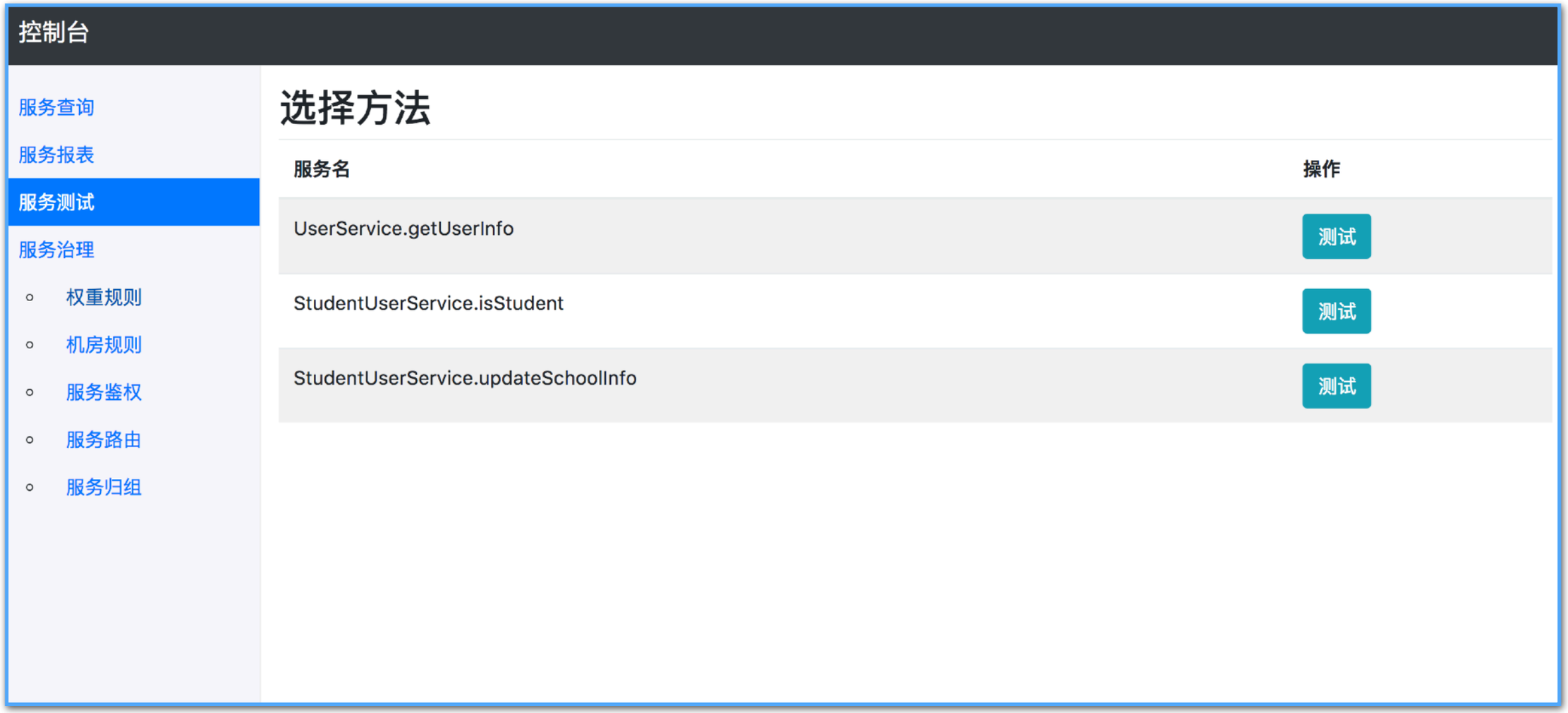
ip	端口	序列化方式	超时时间
0.0.0.0	8081	json	2000ms
0.0.0.0	8082	json	2000ms

元数据

方法的测试


控制台
服务查询
服务报表
服务测试
服务治理
权重规则
机房规则
服务鉴权
服务路由
服务归组
搜索结果
UserService
服务器列表
服务
ip
端口
应用名
操作
UserService
0.0.0.0
8081
user-center
选择方法测试
UserService
0.0.0.0
8082
user-center
选择方法测试

方法的测试


控制台
服务查询
服务报表
服务测试
服务治理
权重规则
机房规则
服务鉴权
服务路由
服务归组
选择方法
服务名
操作
UserService.getUserInfo
测试
StudentUserService.isStudent
测试
StudentUserService.updateSchoolInfo
测试

软负载 - 权重规则

控制台

- 服务查询
- 服务报表
- 服务测试
- 服务治理
  - 权重规则
  - 机房规则
  - 服务鉴权
  - 服务路由
  - 服务归组

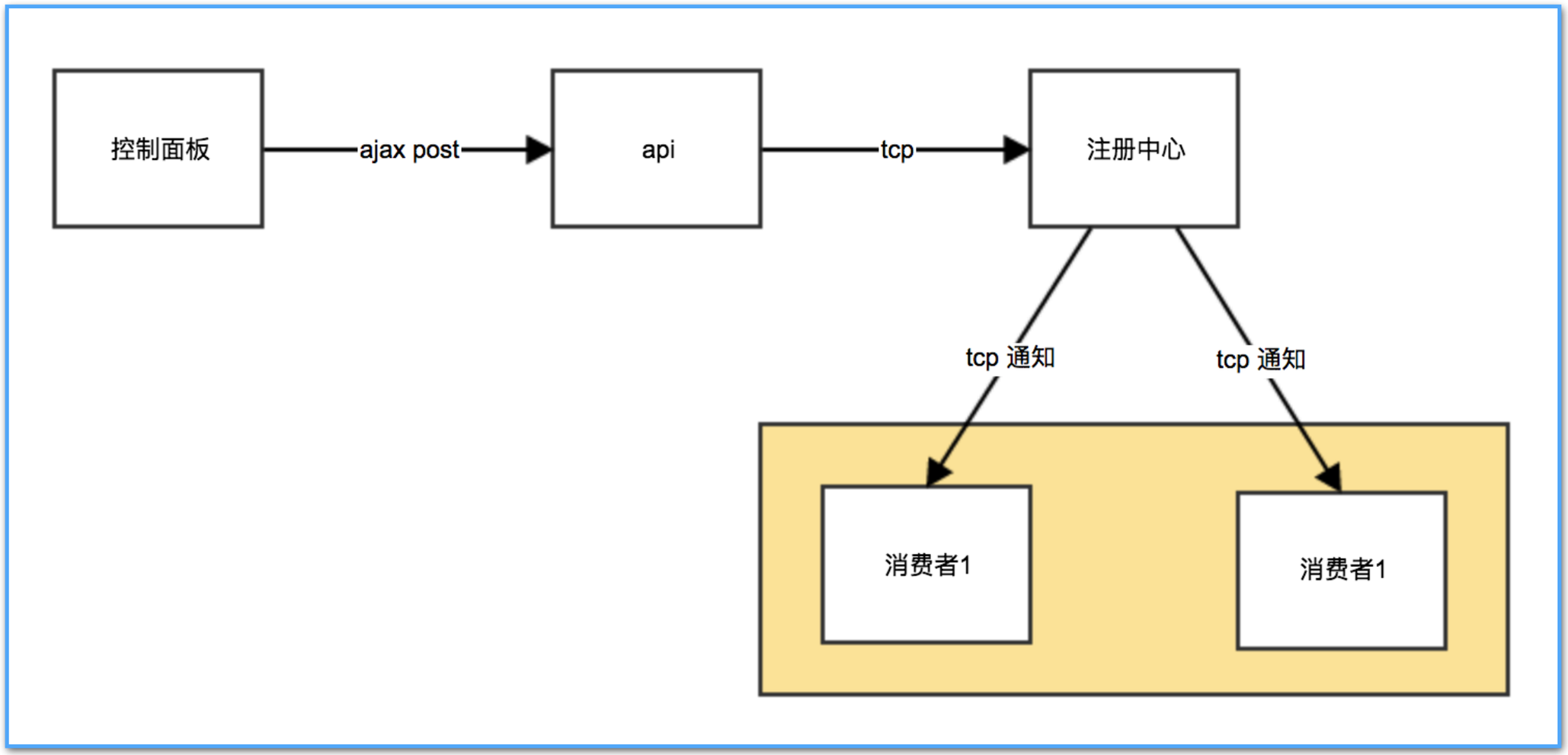
搜索结果

UserService

服务器列表

服务	ip	端口	应用名	操作
UserService	0.0.0.0	8081	user-center	权重: 104 提交
UserService	0.0.0.0	8082	user-center	权重: 103 提交

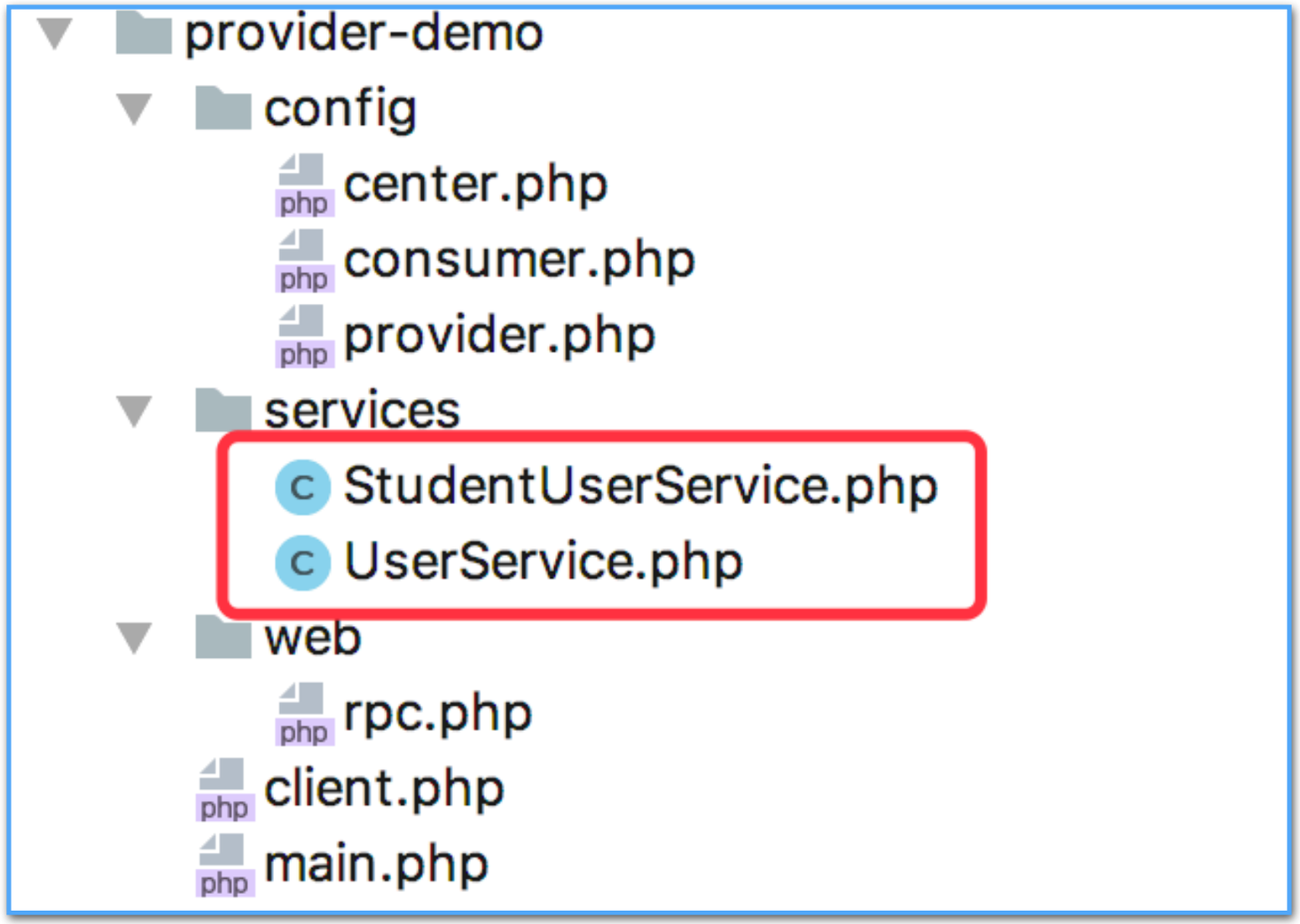
软负载 - 权重规则



软负载 - 权重规则 - 适用场景

与 nginx 类似
机器本身性能差异、配置不一致、一台物理机，一台虚拟机
压测引流、性能摸高

软负载 - 接口路由



如果想对用户中心做接口级别路由，其中提供用户中心的机器有：192.168.1.2、192.168.1.3
希望调用StudentUserService的全部路由到192.168.1.3

软负载 - 方法路由

```php
class UserService
{
    public static function getUserInfo($uid)
    {
        // 假设以下内容从数据库取出
        return [
            'id'       => $uid,
            'username' => 'mengkang',
        ];
    }

    public static function updateUsername($uid,$name){
        // 数据入库操作...
        return true;
    }
}
```

服务调用按照方法控制其路由到固定的几台或者几组机器。
比如需要读写分离

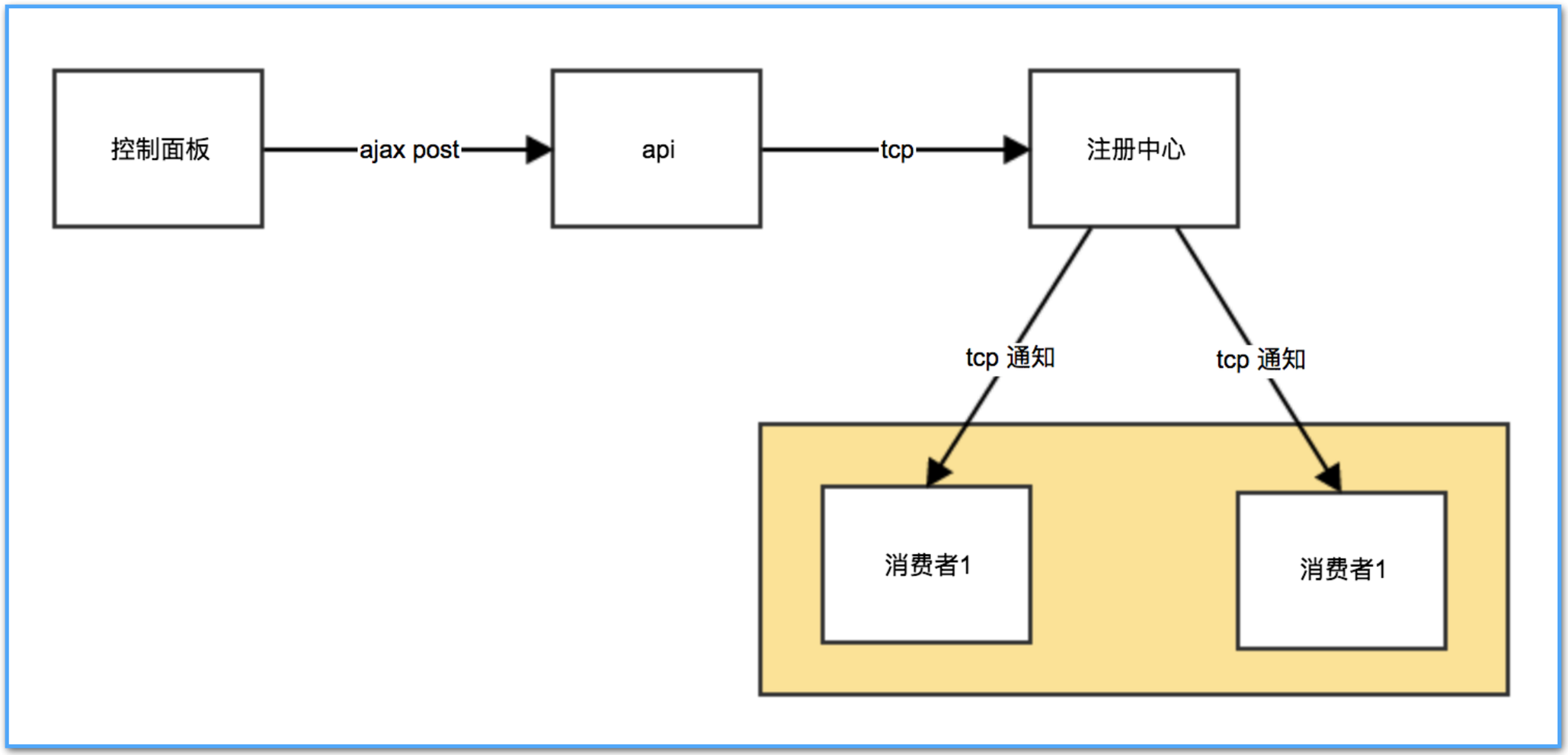
软负载 - 参数路由

服务调用按照参数传入的实际值，路由到固定的几台机器或者几组机器。
比如根据一个响应时间的参数要求，路由不同性能的机器组，有共享组，有独享组，类似会员专享

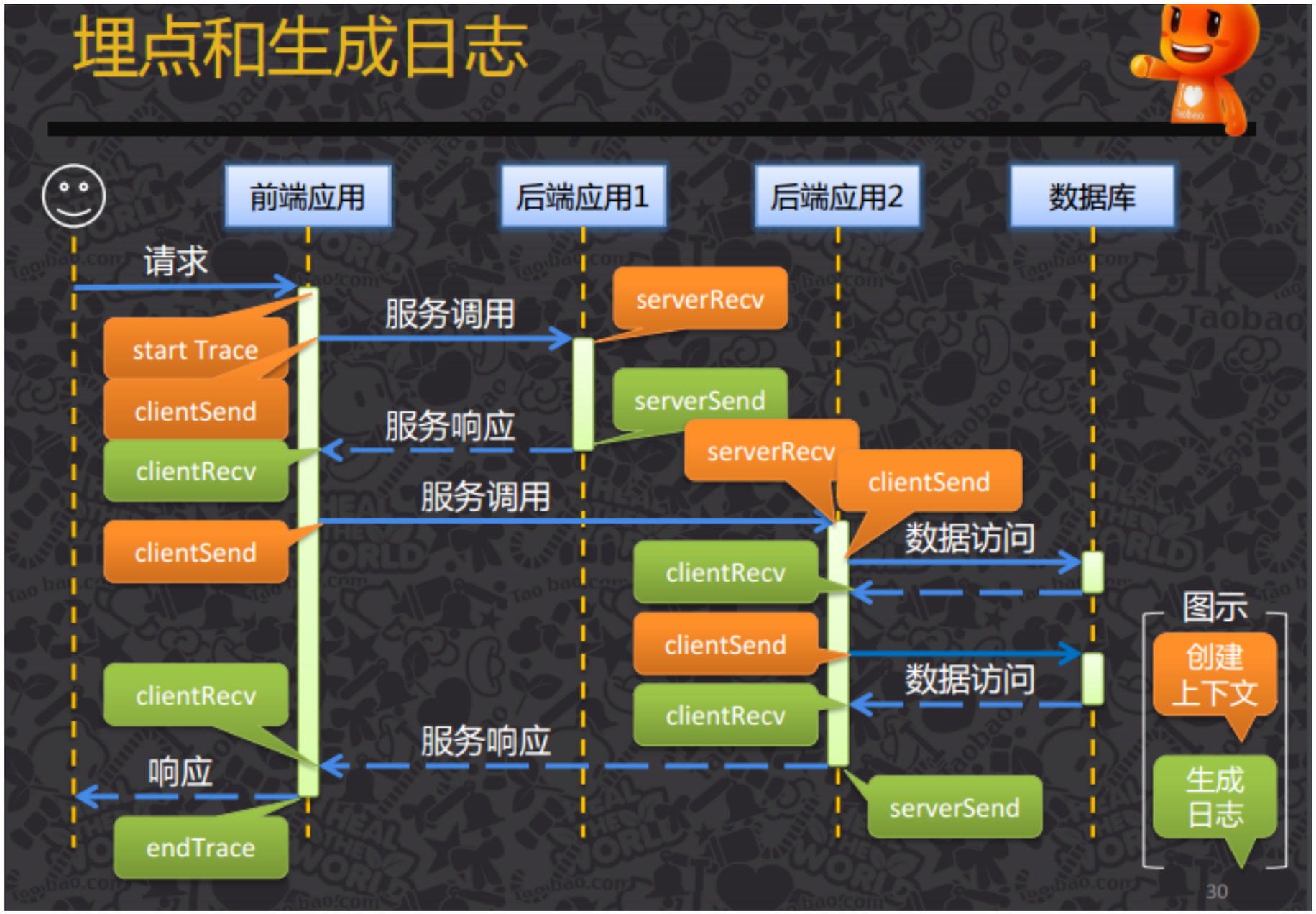
服务治理 - 服务鉴权

只有授权应用才能使用
比如查询用户的手机号

服务治理 - 优雅下线



服务治理 - 分布式调用跟踪



图片来源 https://yq.aliyun.com/articles/58408

服务治理 - 分布式调用跟踪

图片来源 https://wenku.baidu.com/view/1cf0f1c3d1f34693daef3e82.html

服务治理与微服务的结合运用

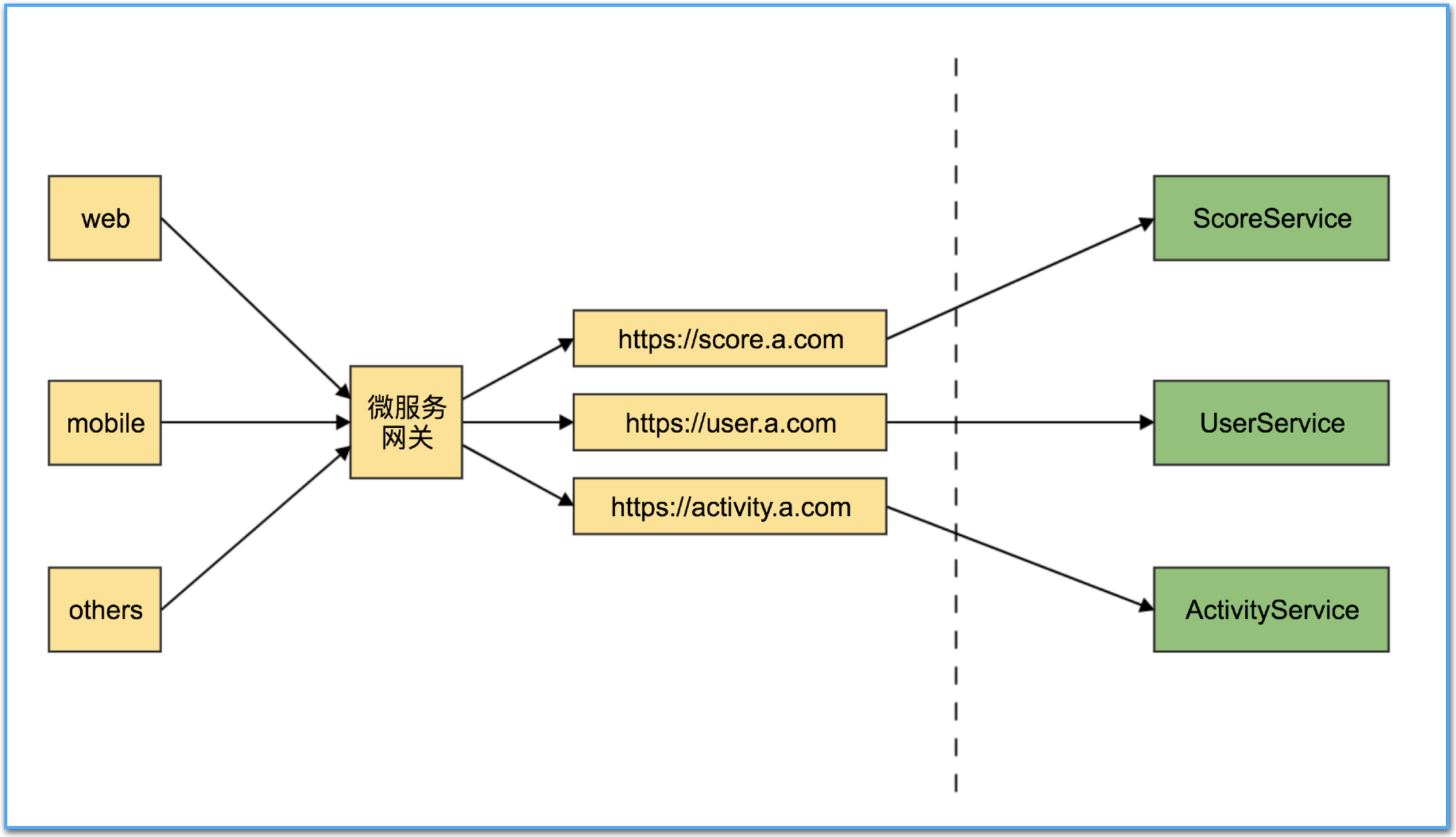
微服务架构

服务网关 + restful api

服务治理

注册中心 + 接口编程

服务治理与微服务的结合运用



服务治理 - 服务鉴权

代码地址：https://gitee.com/zhoumengkang/soa-03-all
原文：https://segmentfault.com/l/1500000011300619

谢谢