亿级 PV 网站架构的技术细节与套路

亿级 PV 网站架构的技术细节与套路

后端架构的套路

性能优化的思路

服务治理的实现

日志收集与分析

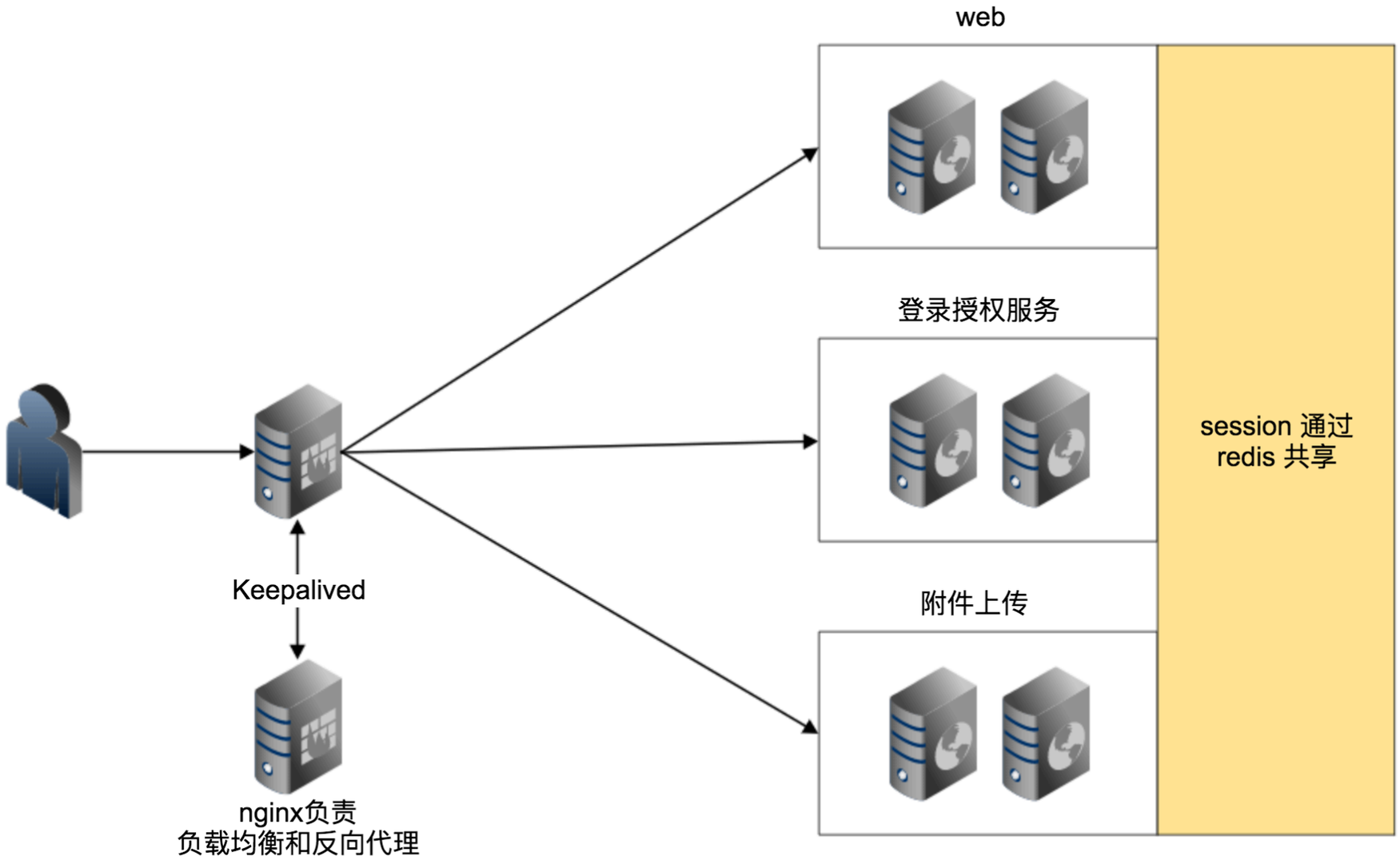
分布式计划任务

流量预估与压测

完善的监控系统

轻巧的发布系统

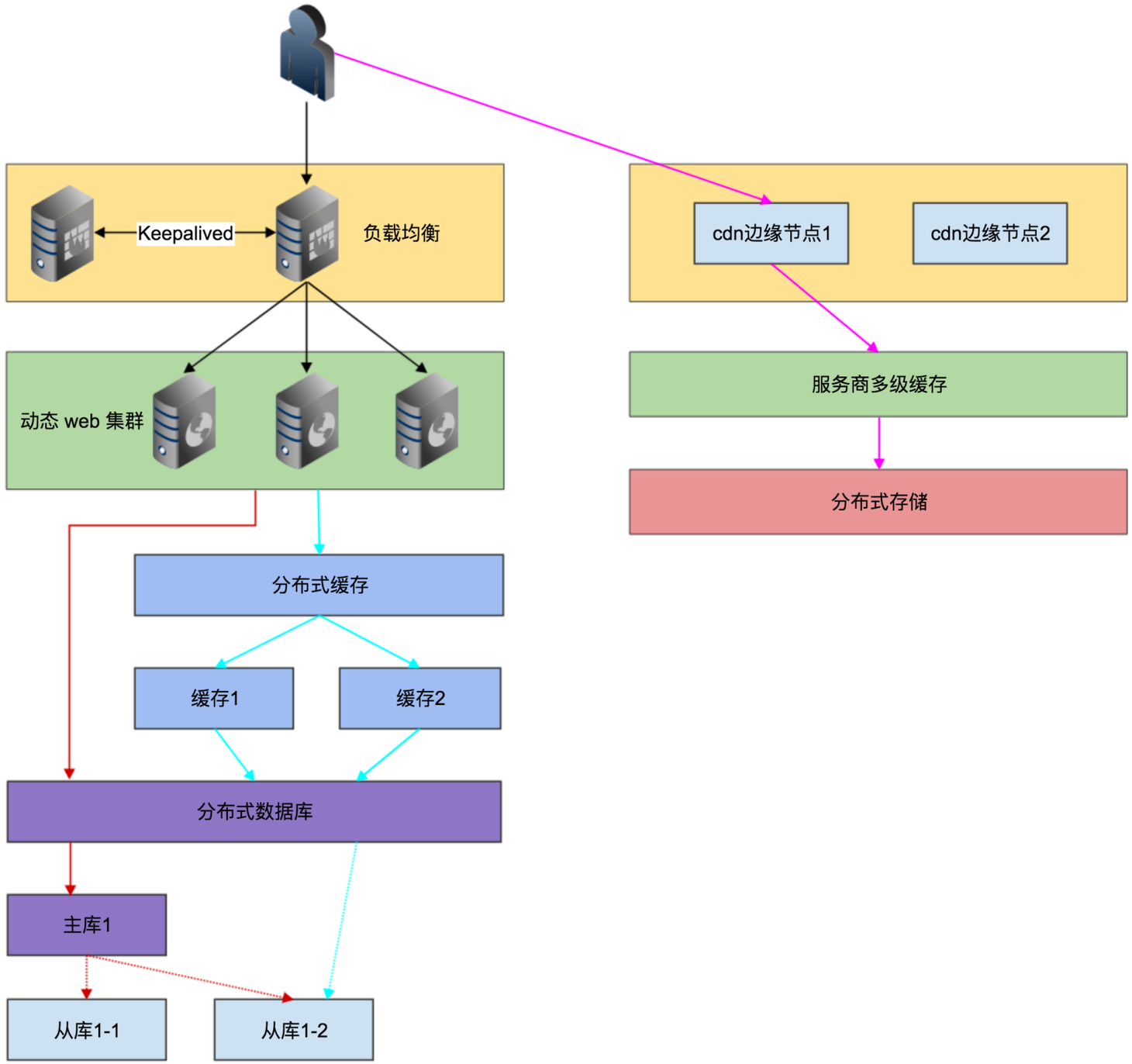
后端架构的套路（服务解耦）



为什么要业务拆分?

- 各请求耗时不一样，防止请求阻塞
- 功能属性、登录重启冷数据、重组feed流，计算密集；上传文件I/O密集
- 方便故障排查
- 代码不一定要拆分

后端架构的套路（负载均衡、动静分离、读写分离、缓存、分布式）



后端架构的套路

cdn 切片 https://mengkang.net/641.html

什么叫集群、分布式，分布式与集群有什么区别?

集群是物理形态，分布式是工作方式。

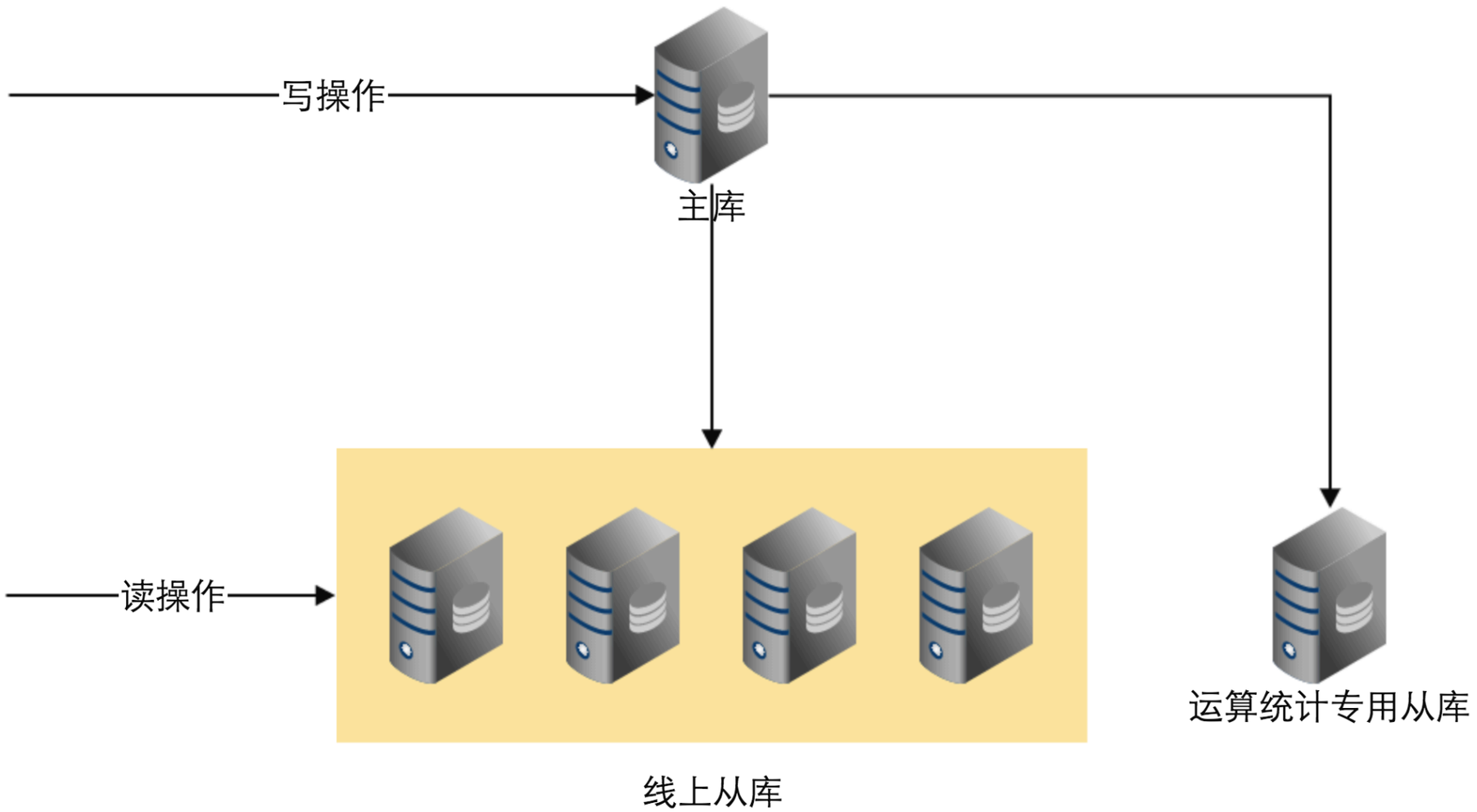
只要一堆机器放在那里，就是集群。

分布式将任务放在多个物理隔离的节点上进行。

分布式中各个子节点互不通信，统一受管控中心管理调度。

分布式管控中心指定路由、负载均衡，发现并剔除故障设备，方便扩容。

实践举例：实现一个分布式数据库



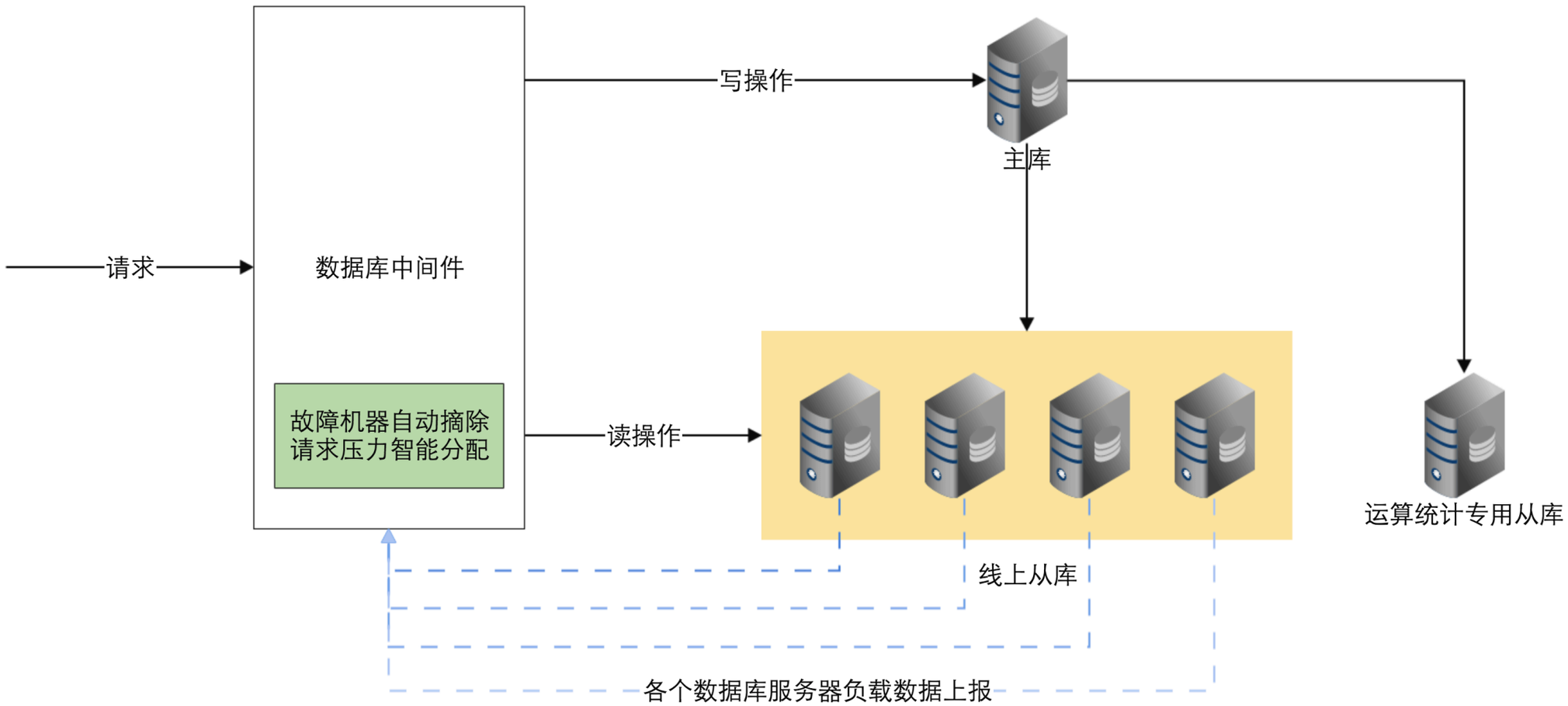
实践举例：实现一个分布式数据库

```php
define('USER_DB','user_db');
define('RANK_DB','rank_db');
```

```php
$tableConfig = [];
$tableConfig['user_fans'] = [
    'table_num' => 10,
    'db_group'  => USER_DB,
];

return $tableConfig;
```

```php
$dbConfig = [];

$dbConfig[USER_DB] = [
    'write' => [
        'host'     => '',
        'port'     => '',
        'dbname'   => '',
        'username' => '',
        'password' => '',
    ],
    'read'  => [
        [
            'host'     => '',
            'port'     => '',
            'dbname'   => '',
            'username' => '',
            'password' => '',
        ],
        [
            'host'     => '',
            'port'     => '',
            'dbname'   => '',
            'username' => '',
            'password' => '',
        ],
    ],
];

return $dbConfig;
```

实践举例：实现一个分布式数据库



实践举例：实现一个分布式数据库

php 函数 sys_getloadavg + redis 来实现负载的动态分配

```
uptime|awk '{print $10,$11,$12}'|awk -F',' '{print $1}'
```

实践举例：实现一个分布式数据库

面试问题	如何保证读写分离?
加分指数	★☆☆☆☆
复现指数	★☆☆☆☆
实用指数	★★☆☆☆

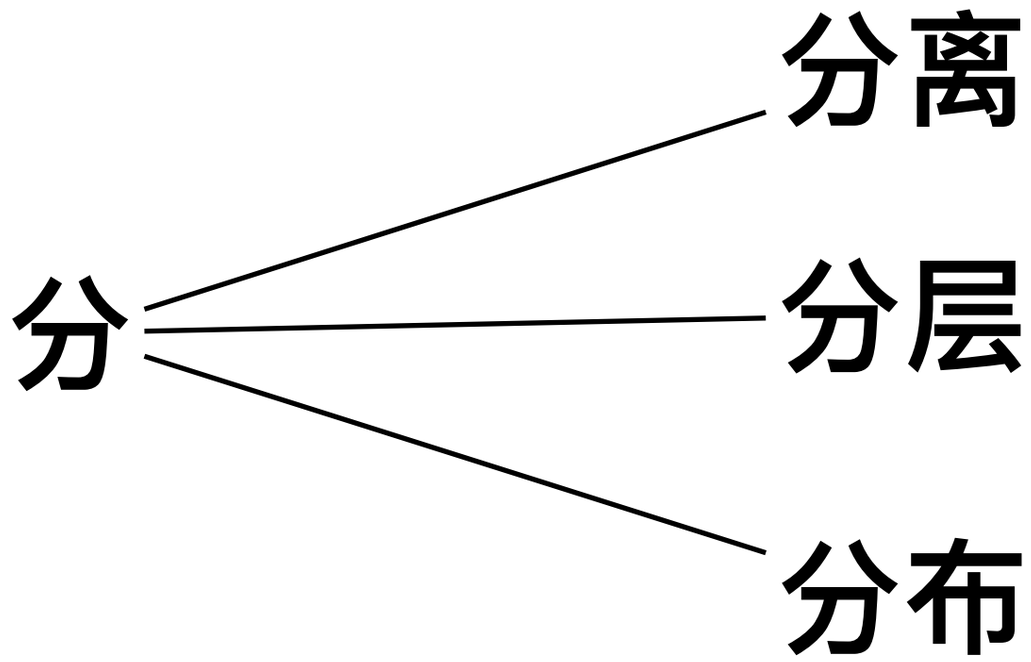
SQL 语句预处理，关键字判断
帐号权限配置，从库操作帐号只拥有读权限

说点细节：实现一个分布式数据库

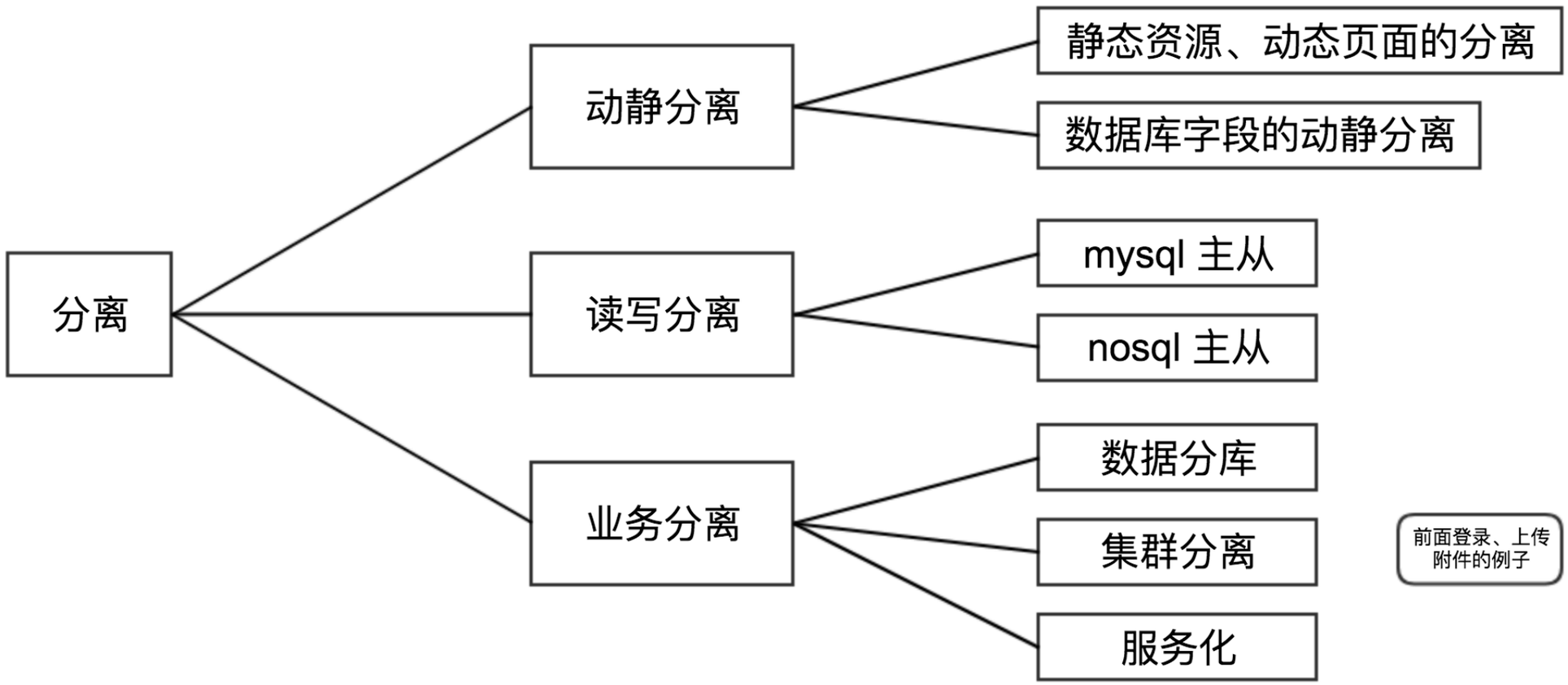
面试问题	数据库服务器负载高，如何排查如何解决?
加分指数	★★☆☆☆
复现指数	★★☆☆☆
实用指数	★★★★★

show processlist
mysql -uroot -e "show processlist;" |grep -i 'delete'|awk '{printf "mysql -uroot -e \"kill %d;\"",$1}'|sh
记得检查分析 mysql 慢日志

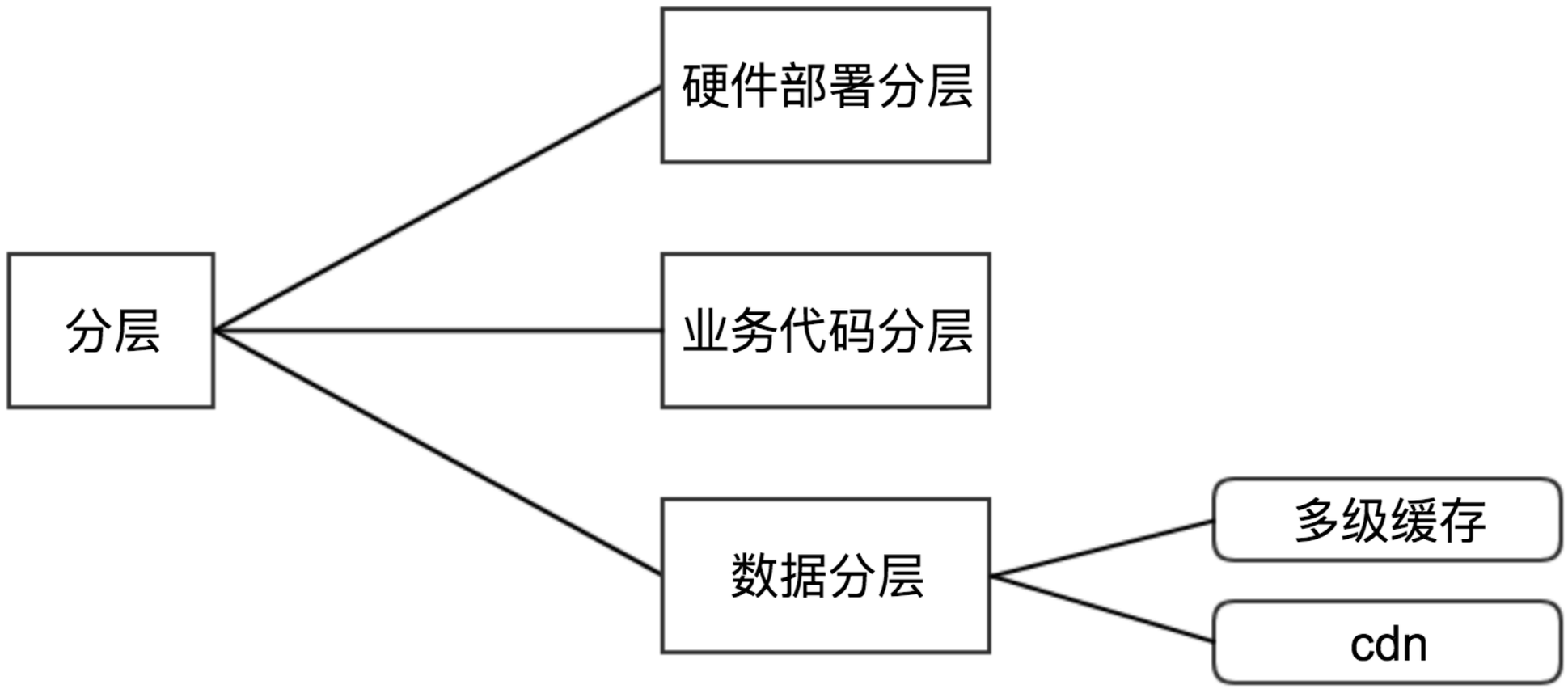
架构和性能优化的核心原则



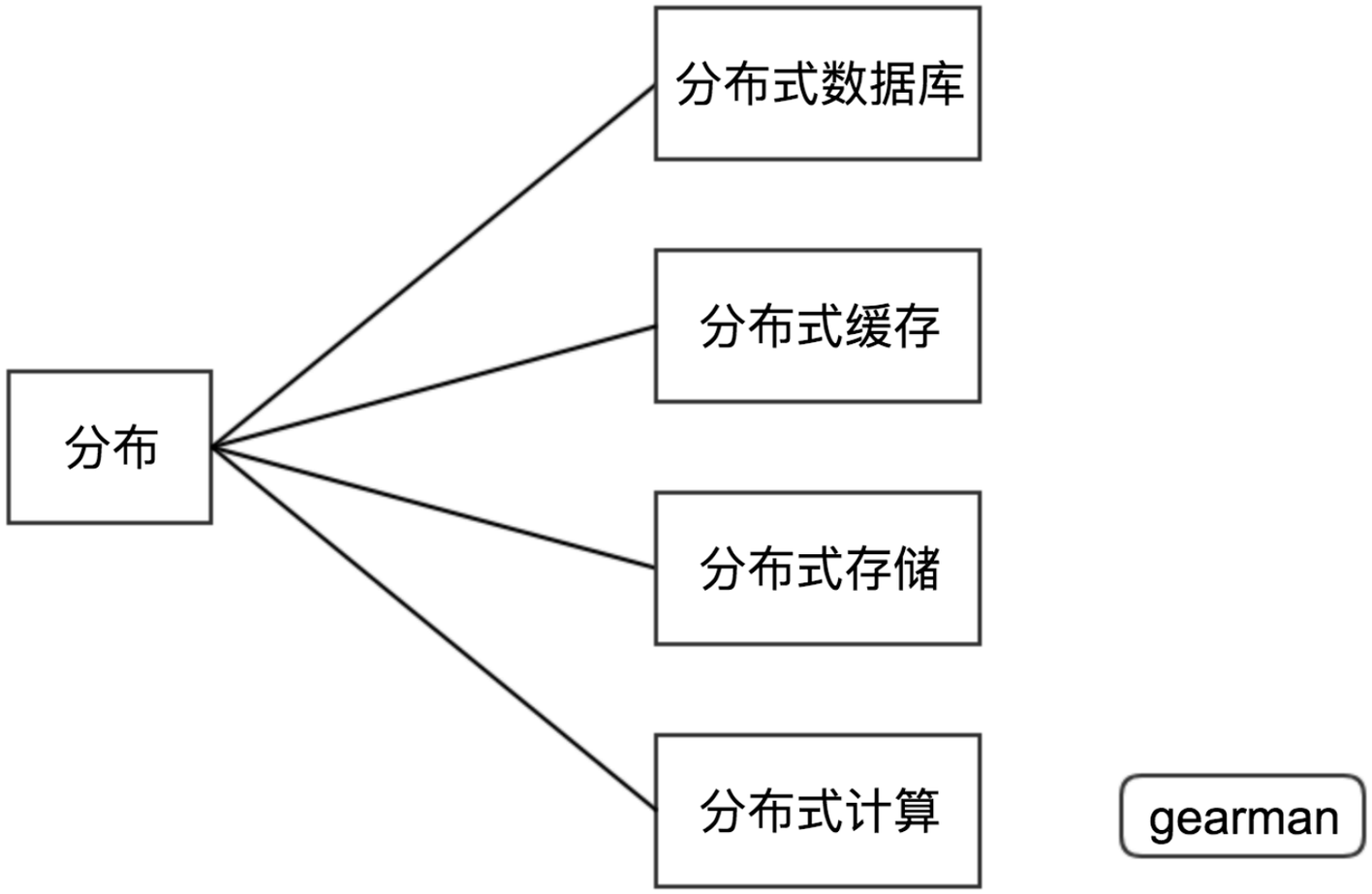
架构和性能优化的核心原则



架构和性能优化的核心原则

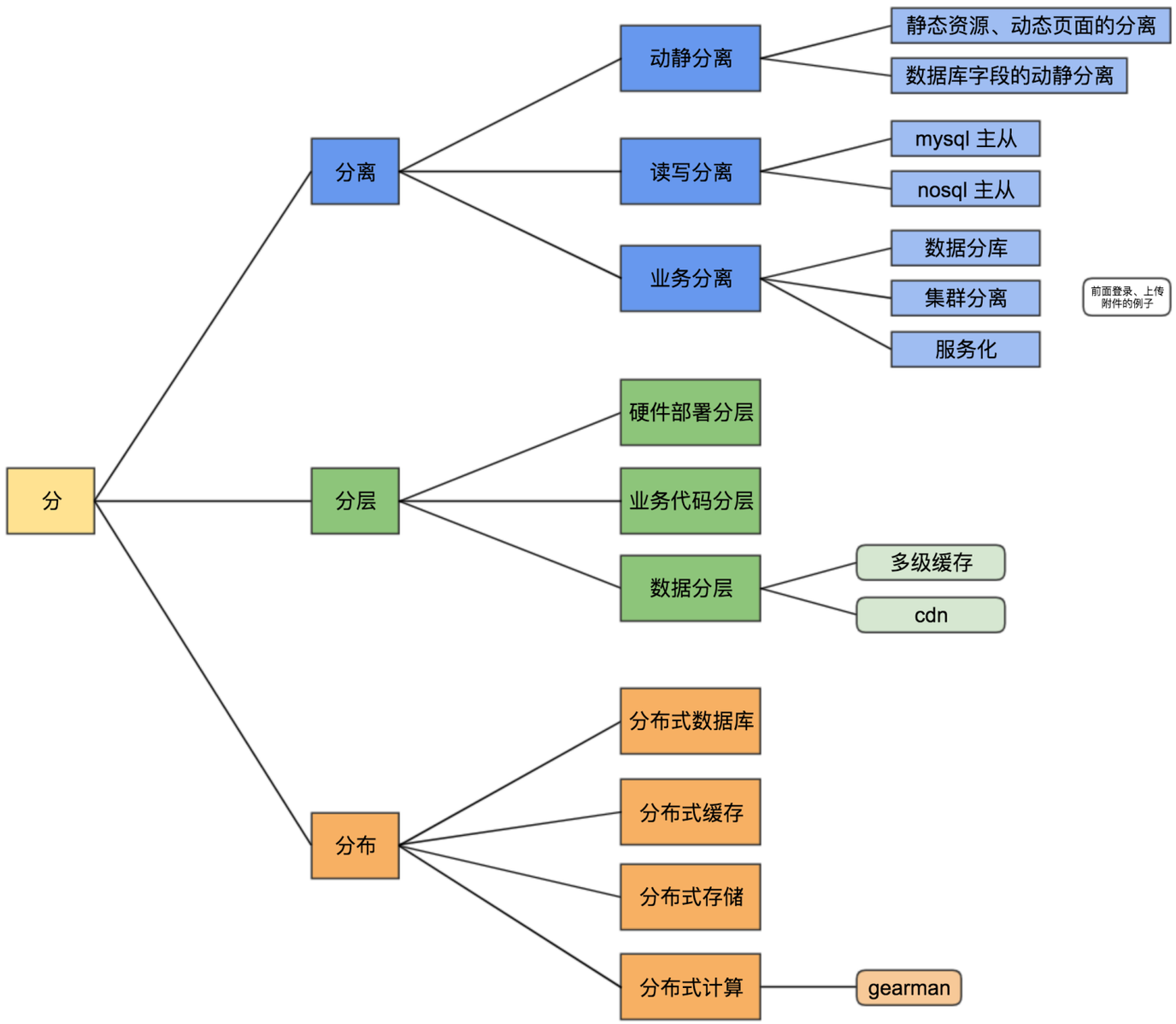


架构和性能优化的核心原则

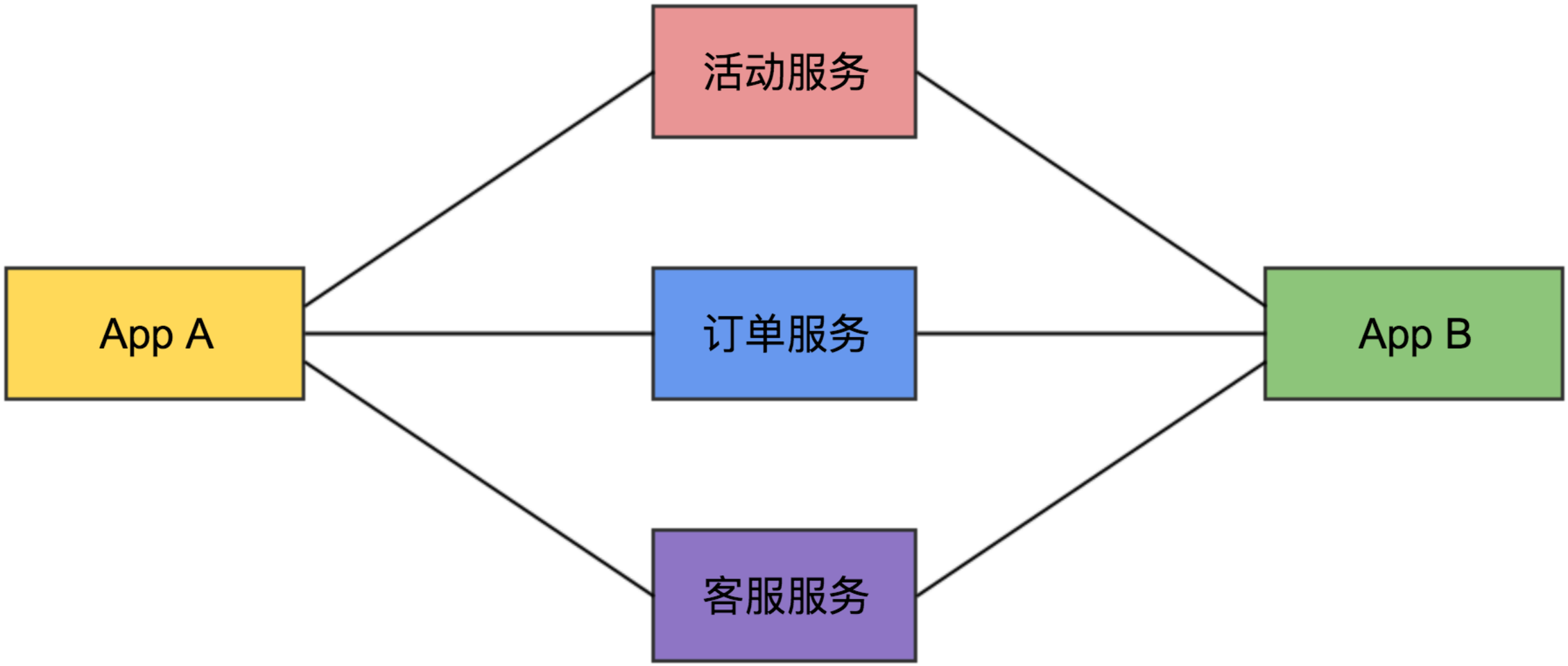


gearman使用举例：https://mengkang.net/369.html

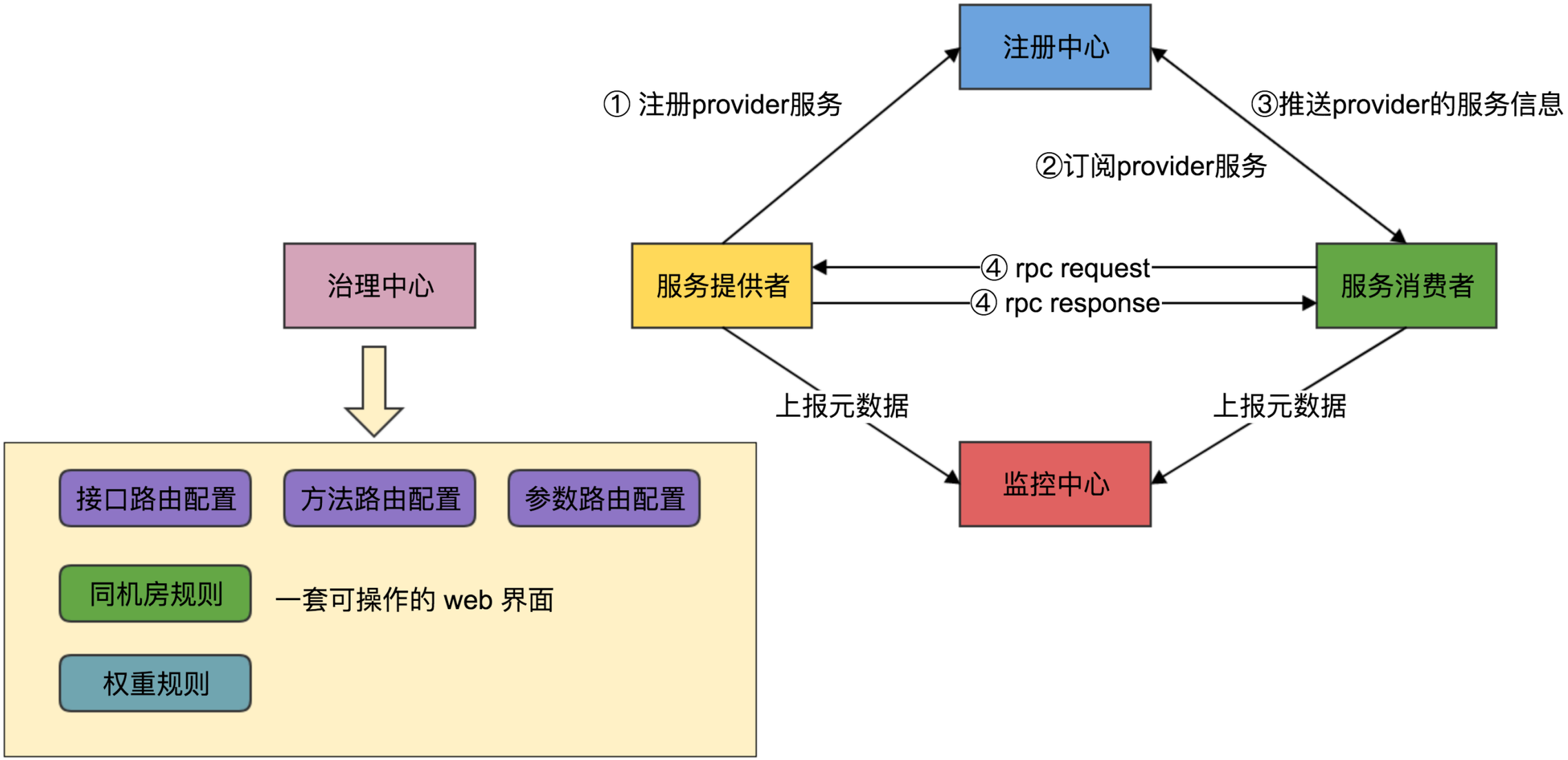
架构和性能优化的核心原则（面试加分利器）



服务治理是如何出现的？为什么需要服务治理？



服务治理的原理

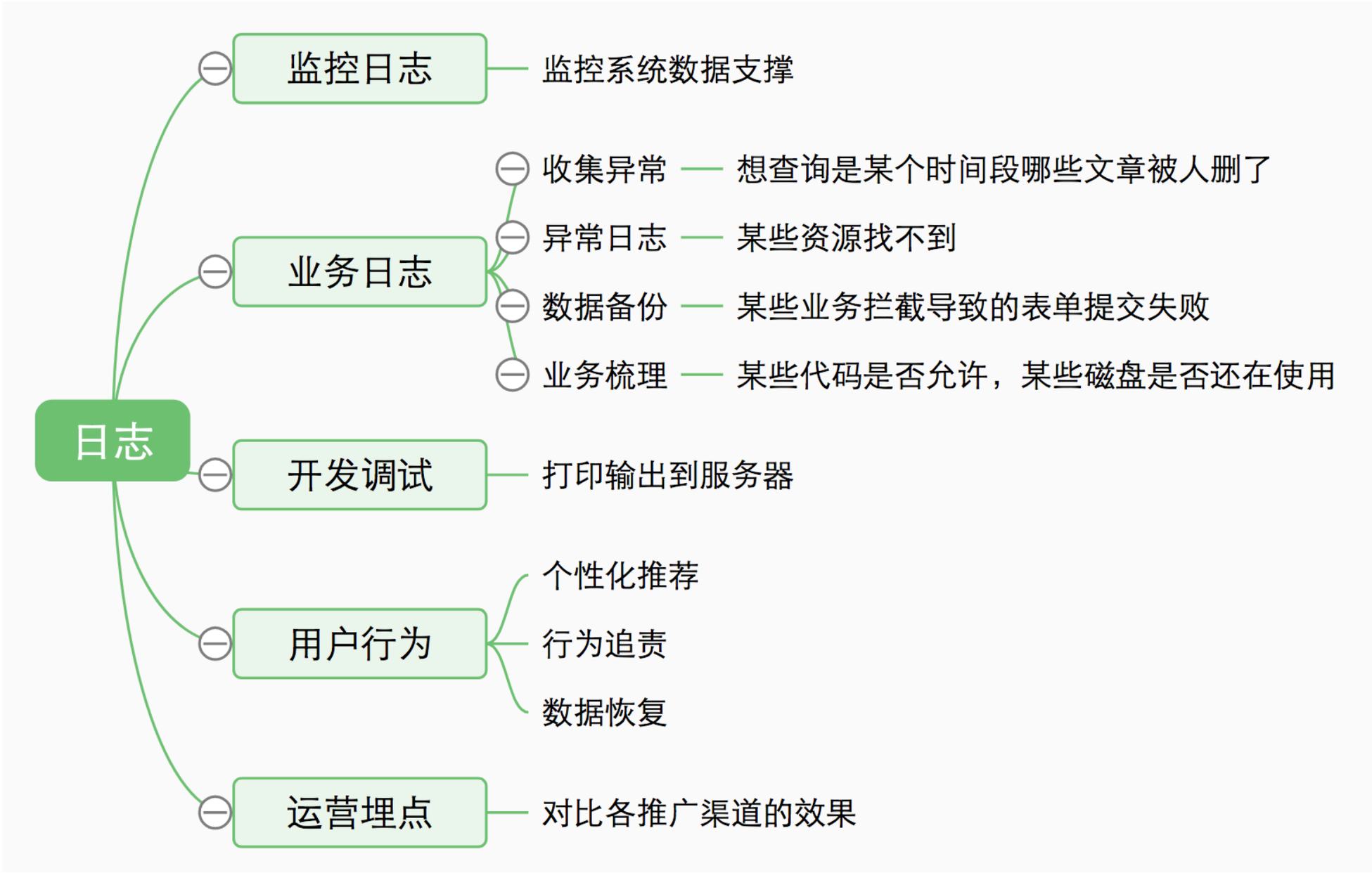

注册中心
① 注册provider服务
③推送provider的服务信息
②订阅provider服务
治理中心
服务提供者
④ rpc request
④ rpc response
服务消费者
上报元数据
上报元数据
监控中心
接口路由配置
方法路由配置
参数路由配置
同机房规则
一套可操作的 web 界面
权重规则

服务治理的核心之一 RPC

RPC 的学习 https://mengkang.net/580.html

yar java client https://github.com/zhoumengkang/yar-java-client

日志的分类


日志
监控日志
监控系统数据支撑
业务日志
收集异常
想查询是某个时间段哪些文章被人删了
异常日志
某些资源找不到
数据备份
某些业务拦截导致的表单提交失败
业务梳理
某些代码是否允许，某些磁盘是否还在使用
开发调试
打印输出到服务器
用户行为
个性化推荐
行为追责
数据恢复
运营埋点
对比各推广渠道的效果

日志收集

Logstash+Elasticsearch+Kibana

sentry

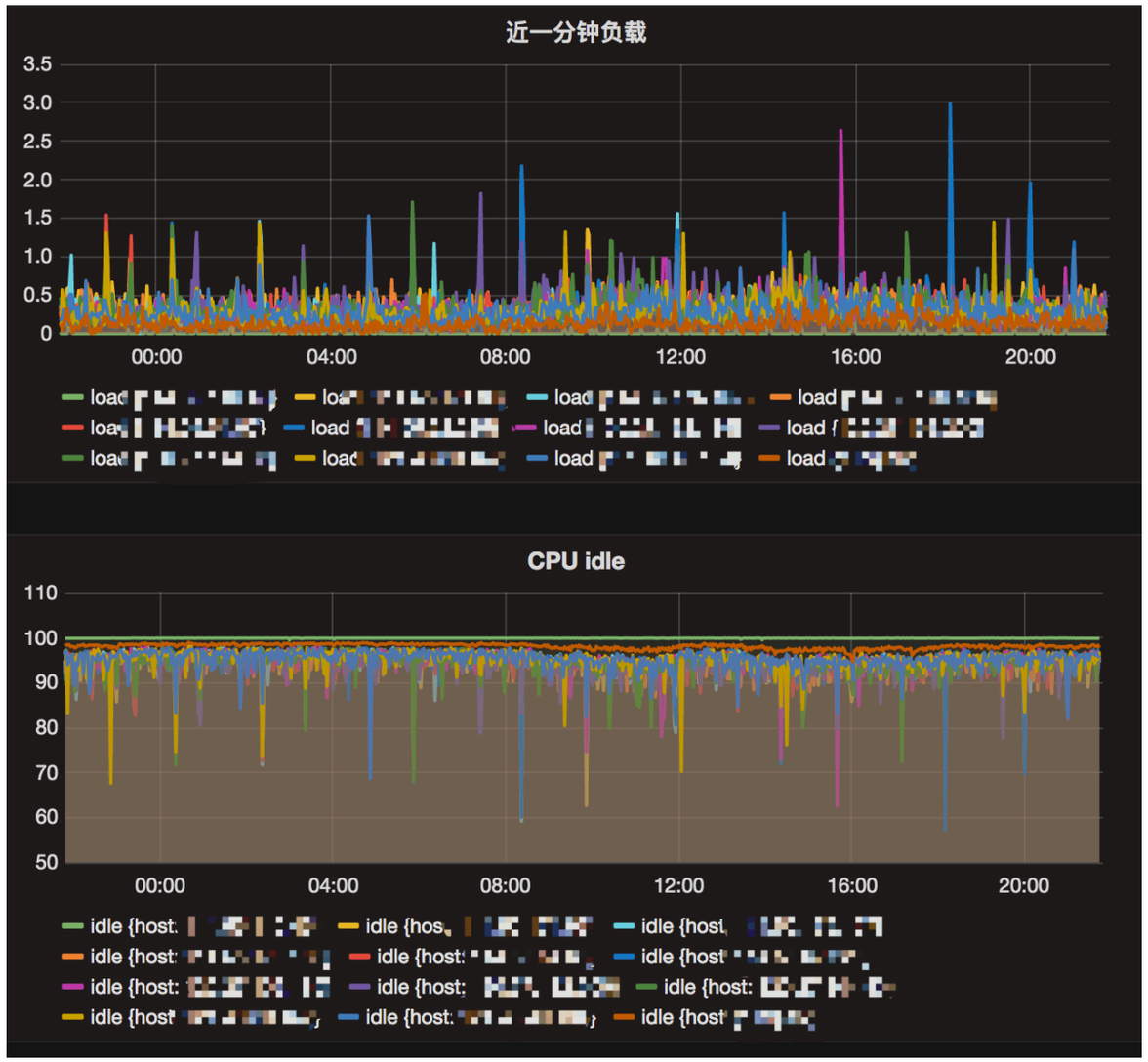
日志分析面试题

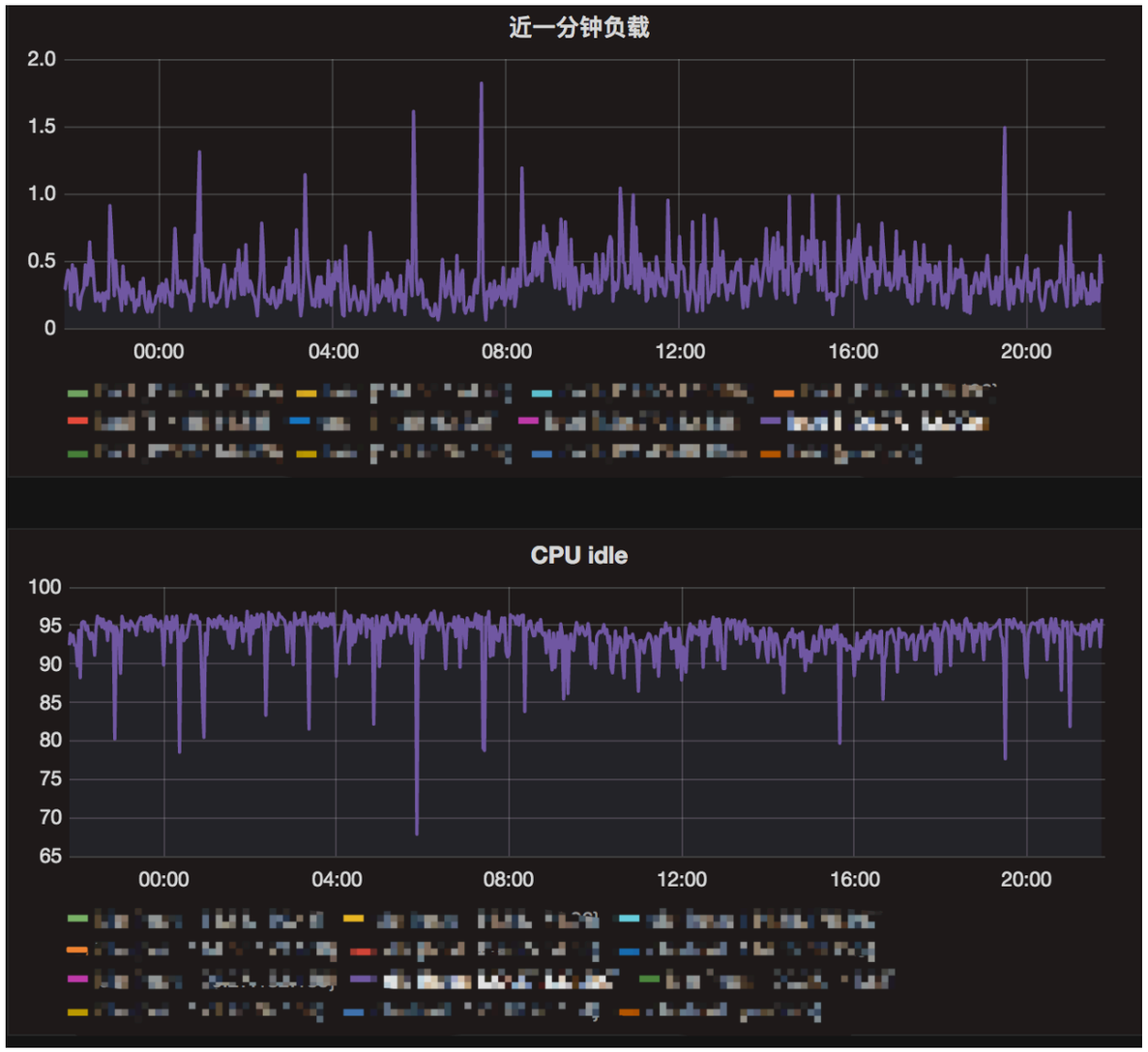
从 nginx 日志里面找出请求耗时最长的100条请求

```
log_format  main  '$remote_addr^A$remote_user^A[$time_local]^A$request'
'^A$status^A$body_bytes_sent^A$http_referer'
'^A$http_user_agent^A$http_x_forwarded_for^A$request_time';
```

```
cat x.log|awk -F '^A' '{print $10,$4}'|sort -r|head -n 10
```

完善的监控系统


近一分钟负载
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
00:00
04:00
08:00
12:00
16:00
20:00
load
CPU idle
110
100
90
80
70
60
50
idle {host:


近一分钟负载
2.0
1.5
1.0
0.5
0
00:00
04:00
08:00
12:00
16:00
20:00
CPU idle
100
95
90
85
80
75
70
65

完善的监控系统

老牌的监控系统 zabbix

现代监控 grafana + influxdb https://mengkang.net/836.html

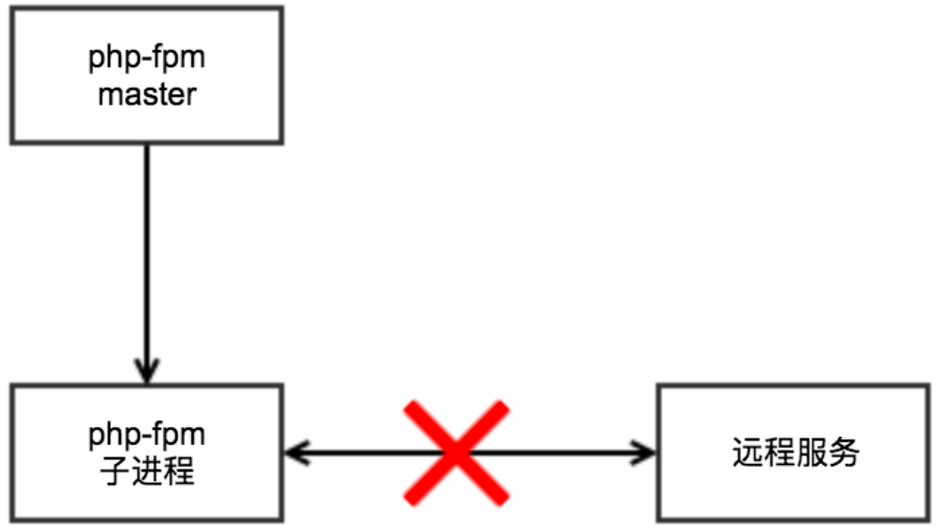
都需要监控哪些数据呢?

系统层：CPU、内存、负载、网卡、I/O 等

应用层：QPS、api 响应时长、redis 内存使用量、队列任务数、php-fpm 进程数、mysql 线程数

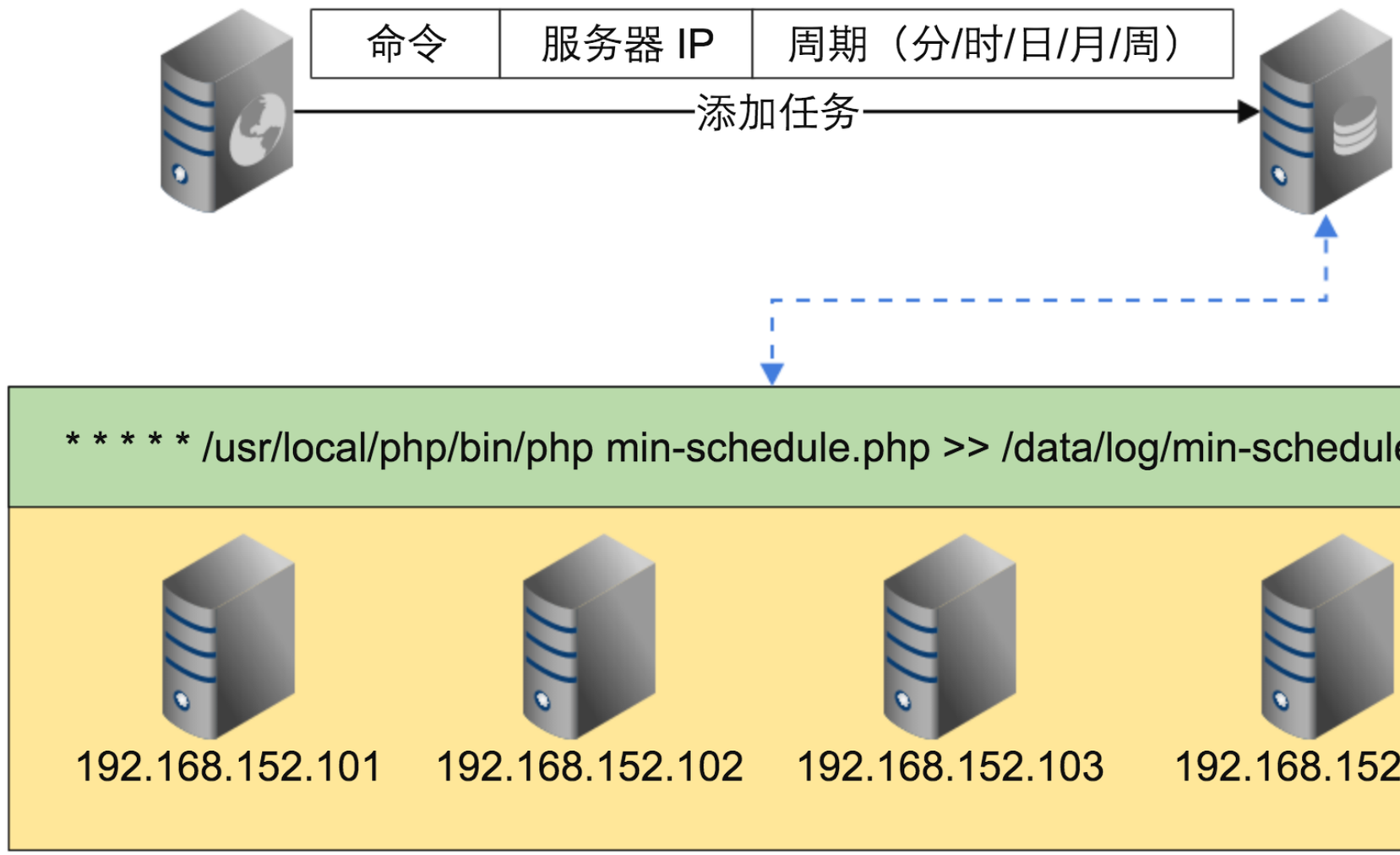
健康巡查：dns 解析、ip 是否访问、硬盘、各种基础服务

案例1：网络I/O阻塞导致的连锁的反应



反思
1. 发现服务不可用时，缓存服务状态10分钟，新请求，直接异常处理。
2. 最好是服务状态能实时更新，正好与前面的服务治理相呼应。
3. 服务健康检查要做好。监控报警要提早。

分布式计划任务的实现


命令
服务器 IP
周期（分/时/日/月/周）
添加任务
* * * * * /usr/local/php/bin/php min-schedule.php >> /data/log/min-schedule.log
192.168.152.101
192.168.152.102
192.168.152.103
192.168.152.104

为什么这么多计划任务服务器?

- 一台机器多个用处，比如主用途是 lvs 的备用机、图片存储服务器。
- 不要因为是后台运行就完全考虑性能，比如查询不带条件，注意内存消耗。

怎么保证执行单一入口的文件里面的任务列表是非阻塞的?

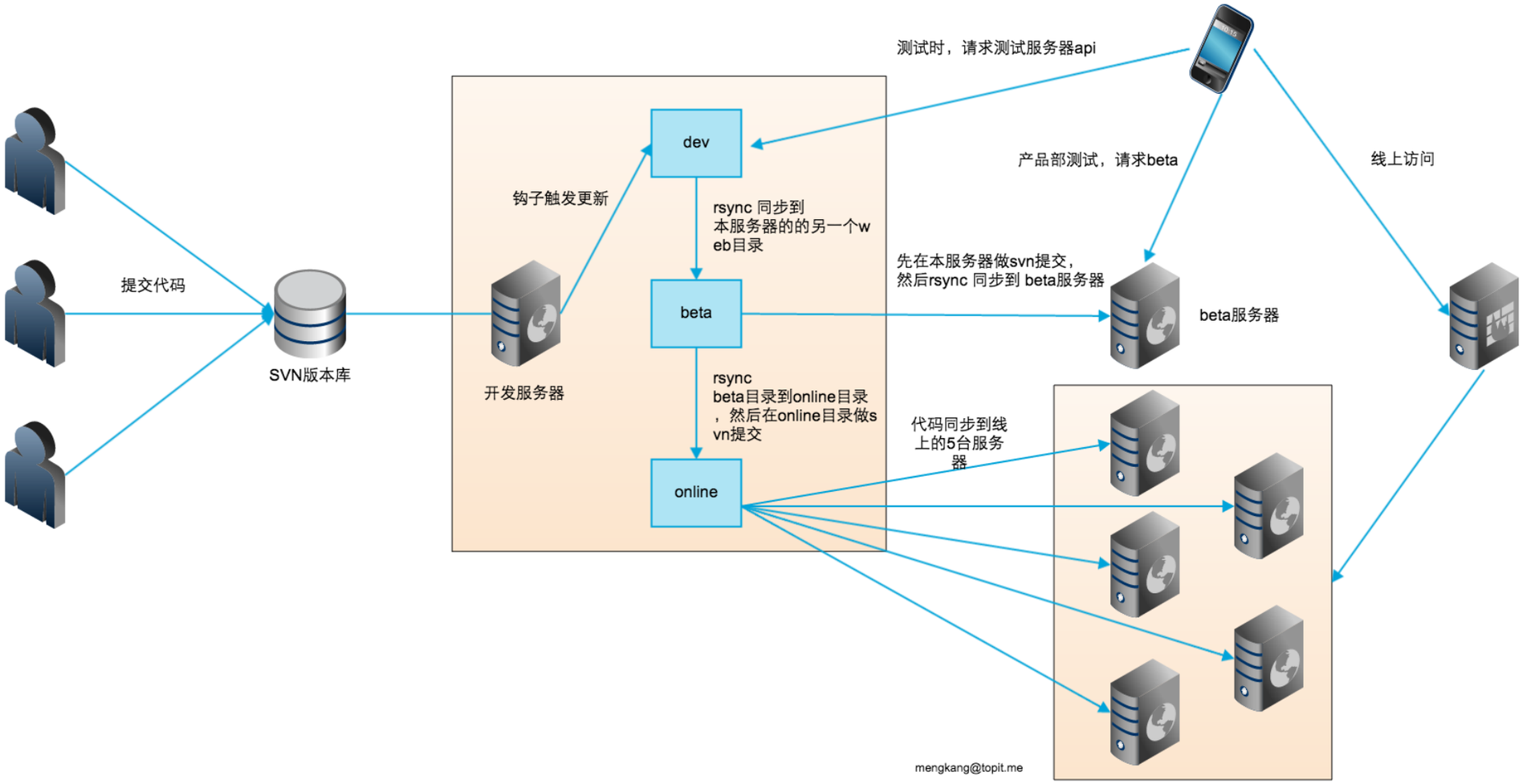
popen，不要用 fread 来接收返回的数据

性能的压测

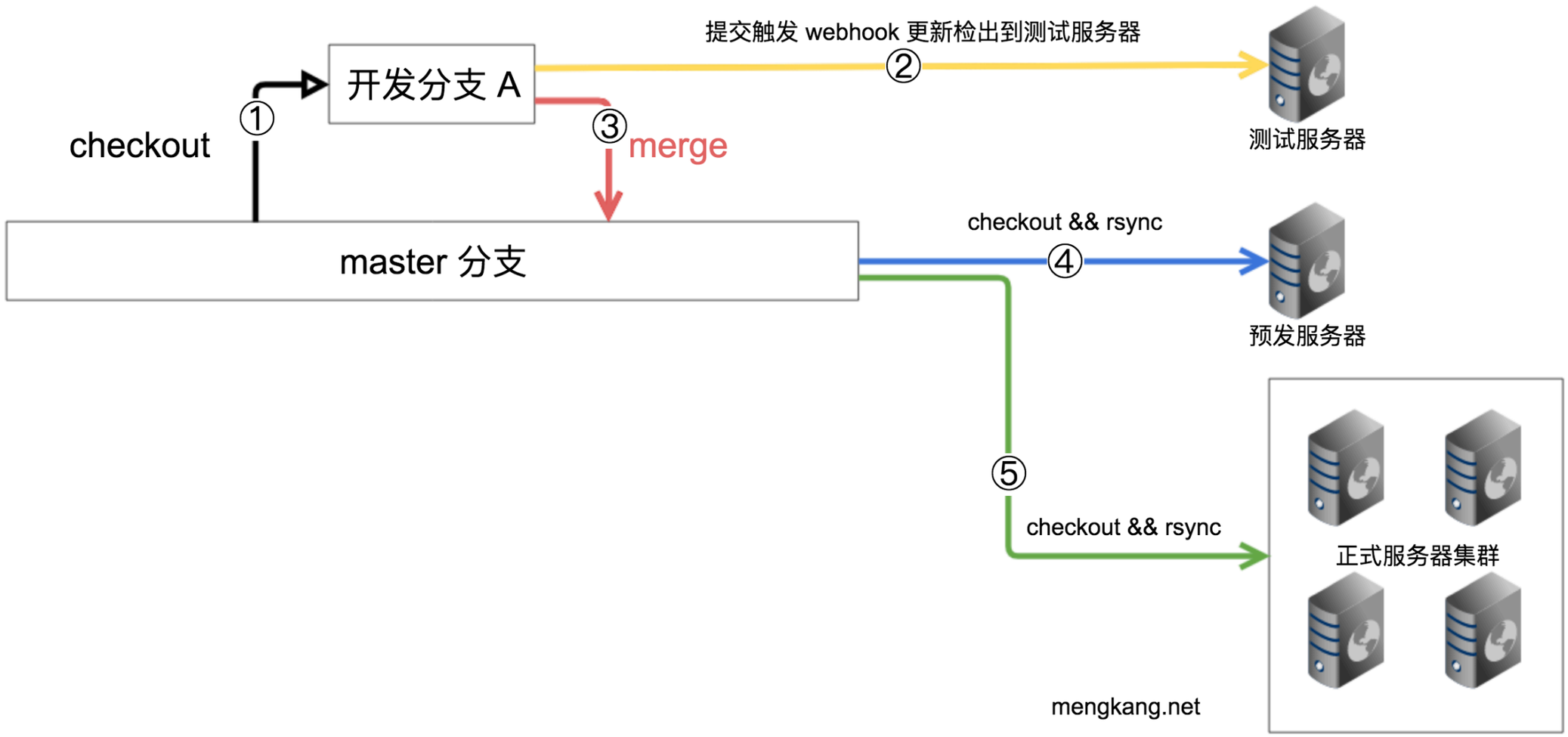
不谈响应时间的吞吐量都是耍流氓 http://coolshell.cn/articles/17381.html

搭建一个基于 git 的发布系统

古老方法：基于 svn 的代码发布系统



搭建一个基于 git 的发布系统




```
git fetch && git checkout $branch -- && git pull --progress --no-stat -v origin $branch
```

搭建一个基于 git 的发布系统（发布脚本）

https://github.com/zhoumengkang/lecture/blob/master/01/release.sh

注意：发布到线上的时候，记录commit id 并邮件通知大家，方便回滚

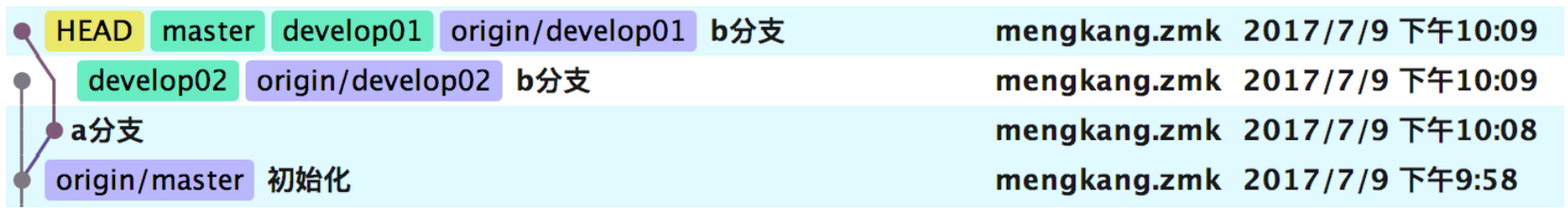
严格来说，应该在合同并确定发布之后，应该删除远程分支

搭建一个基于 git 的发布系统（严格模式）

特殊场景

A同学新建了a分支对直播模块进行迭代开发（后发布）
B同学新建了b分支对读书模块进行迭代快发（先发布）
A同学发现一个历史遗留的严重Bug，想尽快发布

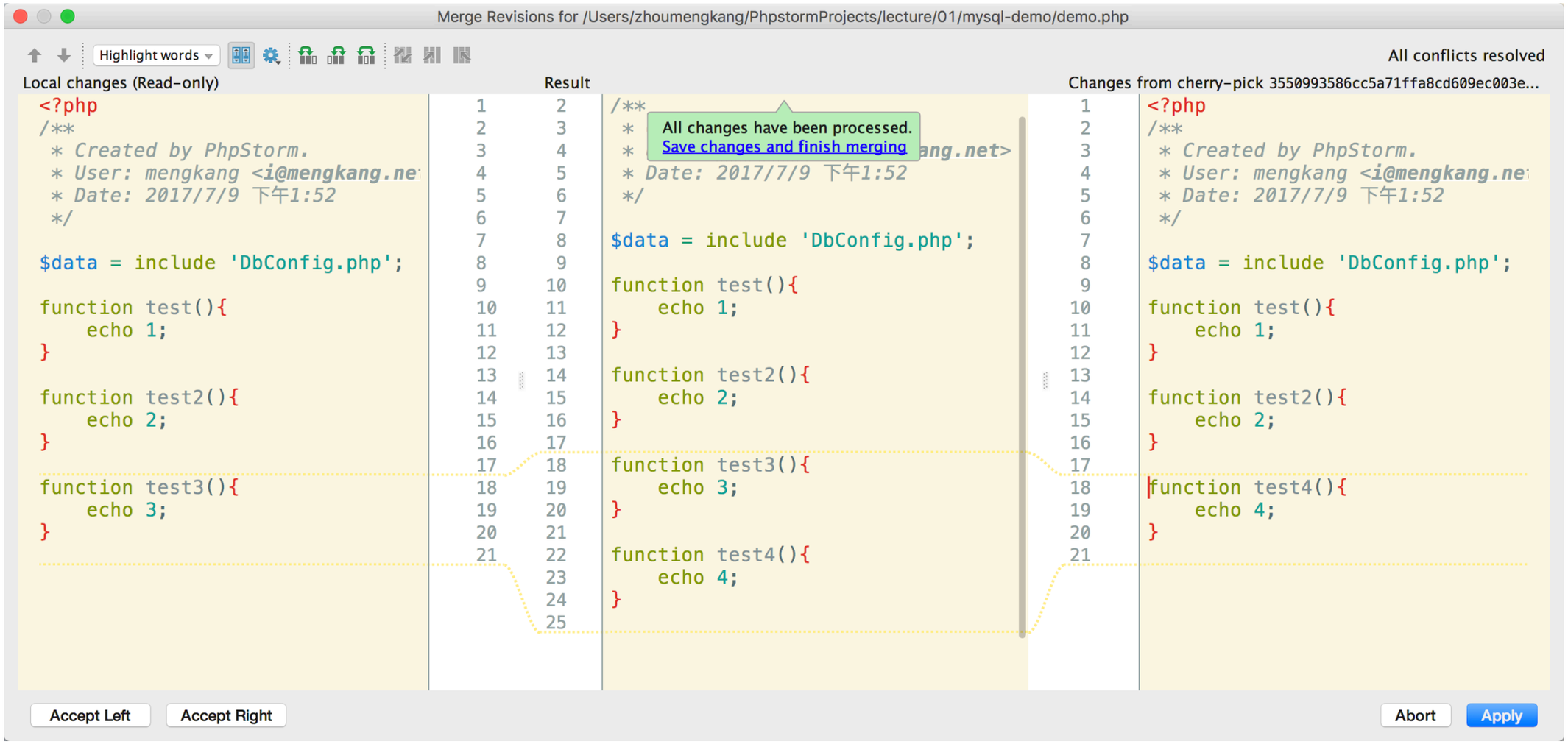
cherry-pick



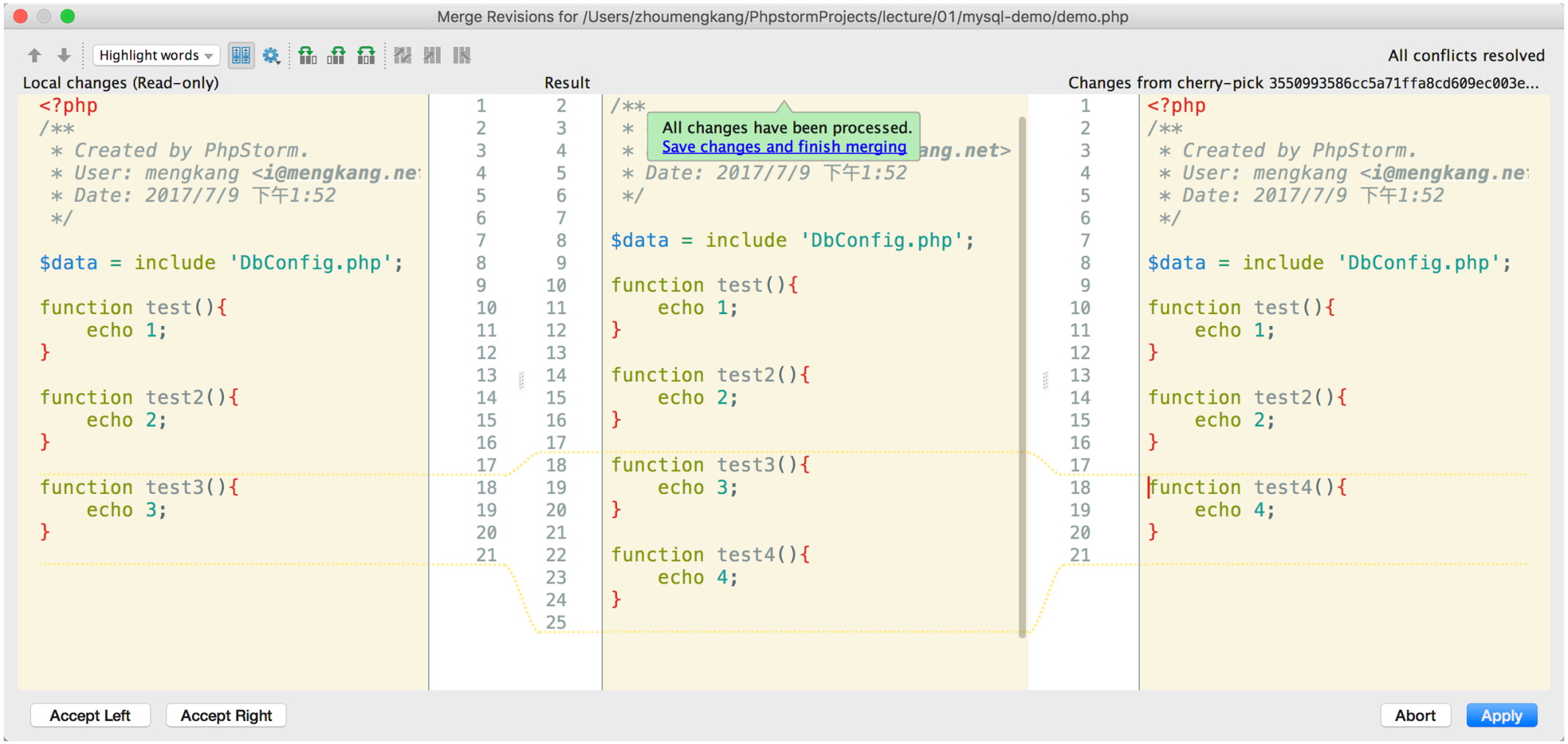
特殊场景


Merge Revisions for /Users/zhoumengkang/PhpstormProjects/lecture/01/mysql-demo/demo.php
Highlight words
No changes. 1 conflict
Local changes (Read-only)
Result
Changes from cherry-pick 3550993586cc5a71ffa8cd609ec003e...
<?php
/**
* Created by PhpStorm.
* User: mengkang <i@mengkang.net>
* Date: 2017/7/9 下午1:52
*/
$data = include 'DbConfig.php';
function test(){
echo 1;
}
function test2(){
echo 2;
}
function test3(){
echo 3;
}
function test4(){
echo 4;
}
Accept Left
Accept Right
Abort
Apply

特殊场景


Merge Revisions for /Users/zhoumengkang/PhpstormProjects/lecture/01/mysql-demo/demo.php
Highlight words
All conflicts resolved
Local changes (Read-only)
Result
Changes from cherry-pick 3550993586cc5a71ffa8cd609ec003e...
All changes have been processed.
Save changes and finish merging
Accept Left
Accept Right
Abort
Apply

特殊场景


Merge Revisions for /Users/zhoumengkang/PhpstormProjects/lecture/01/mysql-demo/demo.php
Highlight words
All conflicts resolved
Local changes (Read-only)
Result
Changes from cherry-pick 3550993586cc5a71ffa8cd609ec003e...
All changes have been processed.
Save changes and finish merging
Accept Left
Accept Right
Abort
Apply