DEVLINK大会

PHP游戏开发

1、傲慢与偏见

- 偏见的产生
  - 历史：端游 -> 页游 ->手游
  - 印象：运营后台、官网等边缘业务

- 实际的案例
  - 社交类：开心农场、超级店长
  - 页游类：猛将无双、女神联盟
  - 手游类：大掌门、城市精灵

- 性价比
  - 招人： 相对容易
  - 优点： 快速迭代，热更新
  - 对比： 1php vs 3~4客户端

适合什么类型的游戏?

- 页游到手游
  大部分页游和手游都适合
- 游戏类型
  社交、卡牌、slg、rpg等

与WEB开发的差异

- 单机极致
  - web的无状态演变出很成熟的横向扩展能力
  - 游戏更考验单机能力的最大化
- 缓存不易
  - web可以做各种数据静态化处理
  - 游戏基本是动态请求处理
- 读写比例
  - web读写比例较大
  - 游戏基本每个请求基本都有数据的读写

百万级DAU游戏的开发经验

- 配置优化很重要
  配置代表数值，游戏的基石之一

- 配置极致的缓存 之 三级火箭

  进程内的缓存
  本机共享内存
  分布式共享内存

- 代码片断

```
public function getByCache($cacheKey) {

    //进程内的缓存
    if (!empty(self::$readCache[$cacheKey])) {
        return self::$readCache[$cacheKey];
    }

    //本地缓存
    if(!empty($localCache)) {
        self::$readCache[$cacheKey] = $localCache->get($cacheKey);
    }

    //网络缓存
    if(!empty($netCache)) {
        self::$readCache[$cacheKey] = $netCache->get($cacheKey);
        $localCache->set($cacheKey, self::$readCache[$cacheKey]);
    }
```

一些坑

- 缓存单条数据，而不是整张表
  一个大数组的反序列化的开销非常大

- Yac 有概率取到错误数据
  不是取不到数据，而是错误数据

- 如何更新
  flush共享内存
  配置文件用redis不同的库 （selectDb）

游戏数据的特点

- 数据量小
- 快
- 数据相对独立

选用REDIS

- Hash结构非常适合游戏的用户数据结构
  一个用户的数据是一个hash，如：
  userid_111 : {
  info: {}, bag:{}, item:{}....
  }
- 性能彪悍、可持久化
- 结构非常多，zsort做排行，key-value做cache等

REDIS的问题

- 内存的需求无限增长
- 不合适做gm数据查询

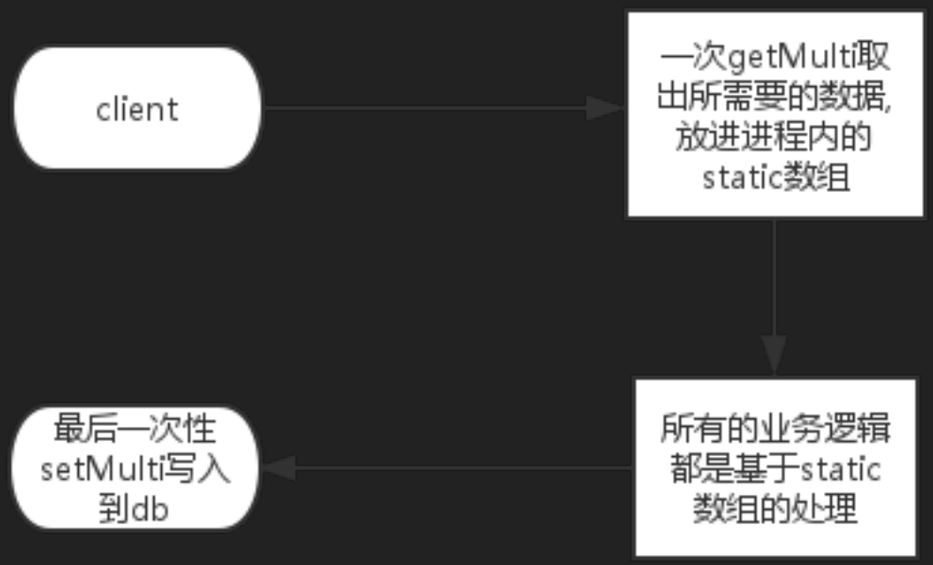
一些解决方案

- 改造redis：redis-storage
- 后台直接读取leveldb

5MS级别的业务请求处理时间

- 天下武功，唯快不破
- 不论业务的复杂度如何
- 一个api请求，只一次读取数据，一次写入数据

执行流程


client
一次getMulti取出所需要的数据，放进进程内的static数组
所有的业务逻辑都是基于static数组的处理
最后一次性setMulti写入到db

好处

- 非常快，基本可以达到5ms~10ms的处理时间要求
- 不产生脏数据，在业务逻辑任何一步处理失败
  都可以很安全的抛出异常，不会写入任何数据到db
- 业务逻辑简单，清析，不产生瓶颈

安全很重要

- 请求防伪造
  数据加密

- 请求防并发
  把并行请求变成并行请求
  1）同一用户
  2）每个请求到达之后，第一步先获取一个请求锁，处理完成之后再释放锁 （可以用redis的setnx机制）

一般外挂的处理

- 同一ip请求密度监控
  不可靠（同一局域网）
- 同一用户的请求密度监控
  不可靠 （人工的疯狂点击也应该存在）
- 同一用户每天的行为阈值监控
  可靠 （每天的产出是固定的）
- 社会工程学
  外挂有一定的规律性

一些小经验

- 和策划的配置的配合

    1）配置的规律化

    2）配置的自动化

配置的规律化

- ID的规律化
  举个例子：
  技能Id = 技能类别 + 等级
  600101 = 600100 + 1
  好处：通过id就可以获得很多有用的信息，不需要反查表
- 奖励规律化
  [[奖励类型，奖励标识, 奖励数量]]
  举个例子：
  [[1, 2, 200]] 表示 获得声望 金币 200个
  随机奖励：
  [[4, [[1,2,200]: 30, “”: 70]]] //表示[1,1,200] 30%的概率获得
- 条件规律化
  [[条件类型, 条件id, min, max]]
  举个例子：
  [[1,1, 20, 999]] //表示等级达到20级才能操作

配置的自动化

A	B	C	D	E	F	G	H	I	J	K	L	M
id	class	pid	sort		max	baseicon	midicon	topicon	des	pickcondition	completecondition	reward
id	类别	pid	排序		最顶级	图标	图标	图标	任务描述	接取条件	完成条件	奖励数据
1	1	1	1		1	2	2	2	2	3	3	3
1010101	1	101	1	训练师成长I	0	XunLianShiChengZhang1.png	5.png		训练师等级达到[c:4#5	[[1,1,1,999]]	[[1,1,5,999]]	[[1,4,5]]
1010102	1	101	2	训练师成长II	0	XunLianShiChengZhang1.png	10.png		训练师等级达到[c:4#1	[[1,3,1010101]]	[[1,1,10,999]]	[[1,4,10]]
1010103	1	101	3	训练师成长III	0	XunLianShiChengZhang1.png	15.png		训练师等级达到[c:4#1	[[1,3,1010102]]	[[1,1,15,999]]	[[1,4,15]]
1010104	1	101	4	训练师成长IV	0	XunLianShiChengZhang1.png	20.png		训练师等级达到[c:4#2	[[1,3,1010103]]	[[1,1,20,999]]	[[1,4,20]]

- 断线重连
- 粗暴型

  断线重新进入游戏
  好处: 不会造成数据不统一

- 友好型

  跟据当前操作来判定是否需要重加载游戏
  好处：体验更好
  坏处： 业务逻辑复杂、可能会导致数据不一致

Q&A

- 名字：王晶
- 网名：半桶水 （桶哥）
- github: https://github.com/shenzhe
- 微博：　不为人民服务
- 组织：　http://www.swoole.com