版本 1.0
20191128

英雄传说多人在线网游服务器后端

项目白皮书

主讲老师：海江
场外支持：范兴
马士兵教育

英雄传说多人在线网游服务器后端

马士兵教育二期课程 第一个项目

项目概述

名称	《Hero Story》英雄传说游戏服务端
项目版本	第一版
预计授课时长	12-18 小时（6-9 次课）效果驱动，视具体效果进行课时延长
主讲老师	海江
部分代码	范兴
项目简介	《Hero Story（英雄传说）》

该游戏是一款欧美卡通风格的 MMORPG 游戏，目前正在研发中，2020 年准备发布在 Google Play 上，进军欧美市场。
该游戏技术框架基于国内知名网页游戏 《回到三国志》，
《回到三国志》 上线腾讯应用中心，并创下每个月 2000 万流水的骄人战绩。

《Hero Story（英雄传说）》在此基础之上做了大量升级和优化，具体内容如下：
前端：
- 采用 Cocos2D 技术，可以导出 Html5、Android、iOS 版本；
- 全部动画效果采用 Spine 骨骼动画技术；
- 本地资源以增量方式更新，版本更新速度加快；

后端：
- 采用 Netty 网络通信框架，同时支持 Http、Socket、WebSocket；
- Google Protobuf 处理消息数据；
- 通过自定义的 DataLifyCycle 机制来更新数据库对象，大幅降低数据库写入压力；
- 通过"注解 + 反射 + Javassist"动态生成字节码，降低工作量并极具提升程序性能；
- 部分功能模块引入 Groovy 技术，实现修改线上 Bug 无需重启服务器；
- 通过 Dubbo、RocketMQ 切分业务系统，实现微服务架构；
- 通过 ELK（ElasticSearch + Logstash + Kibana）收集和分析日志；
- 通过 Jenkins + Ansible 实现自动化的多机部署；

前置技能

多线程与高并发
JavaSE
Maven
Git
IO NIO 反射

讲师介绍

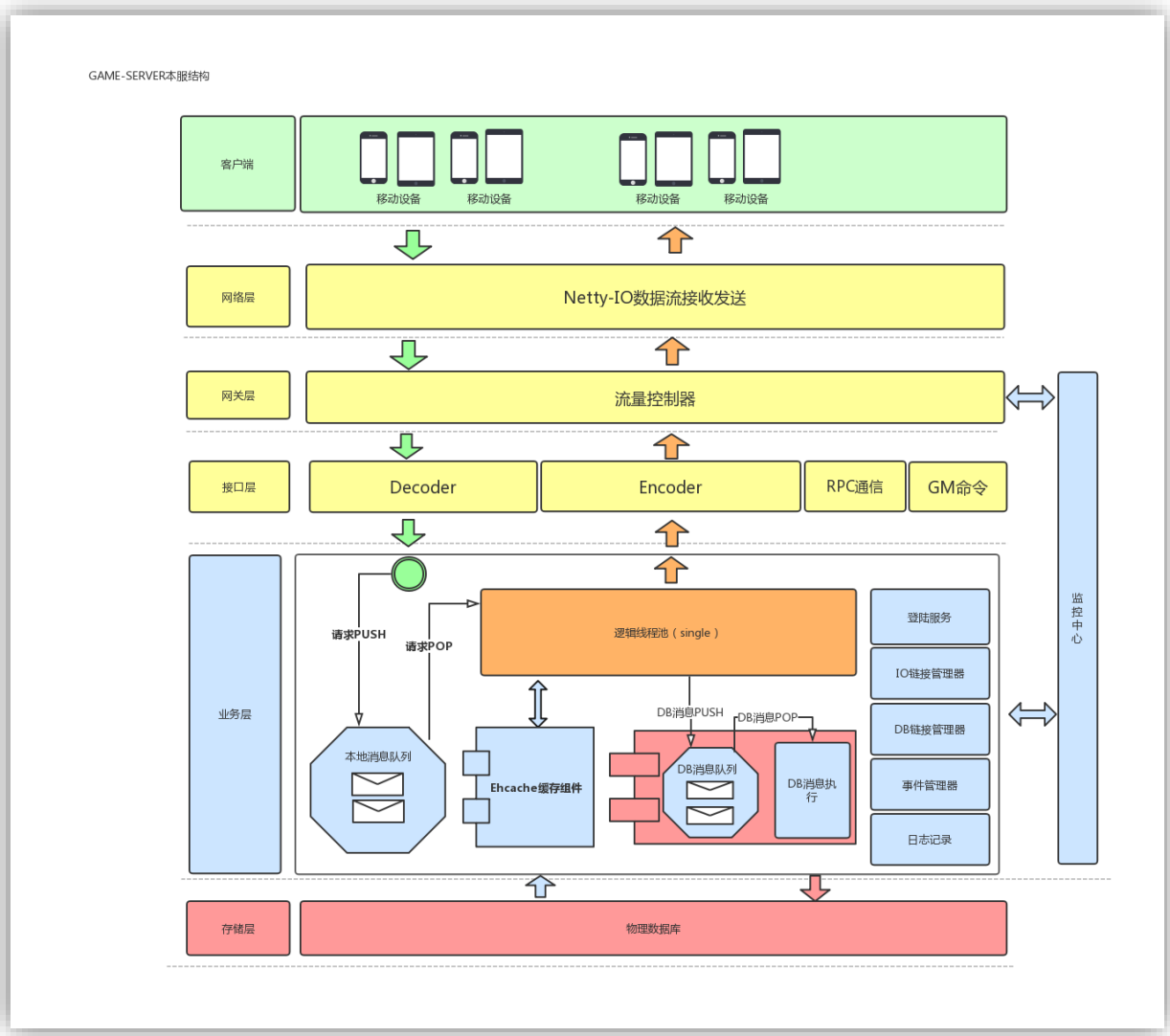
海江，10 年以上游戏开发经验
人人游戏技术经理，《钢铁元帅》服务端开发者，月营收达到千万级
掌趣科技技术总监，《回到三国志》服务器架构设计以及核心战斗系统实现者

指导并参与多个项目的服务器架构设计
组织多次公司内部的技术培训
掌趣科技金牌讲师

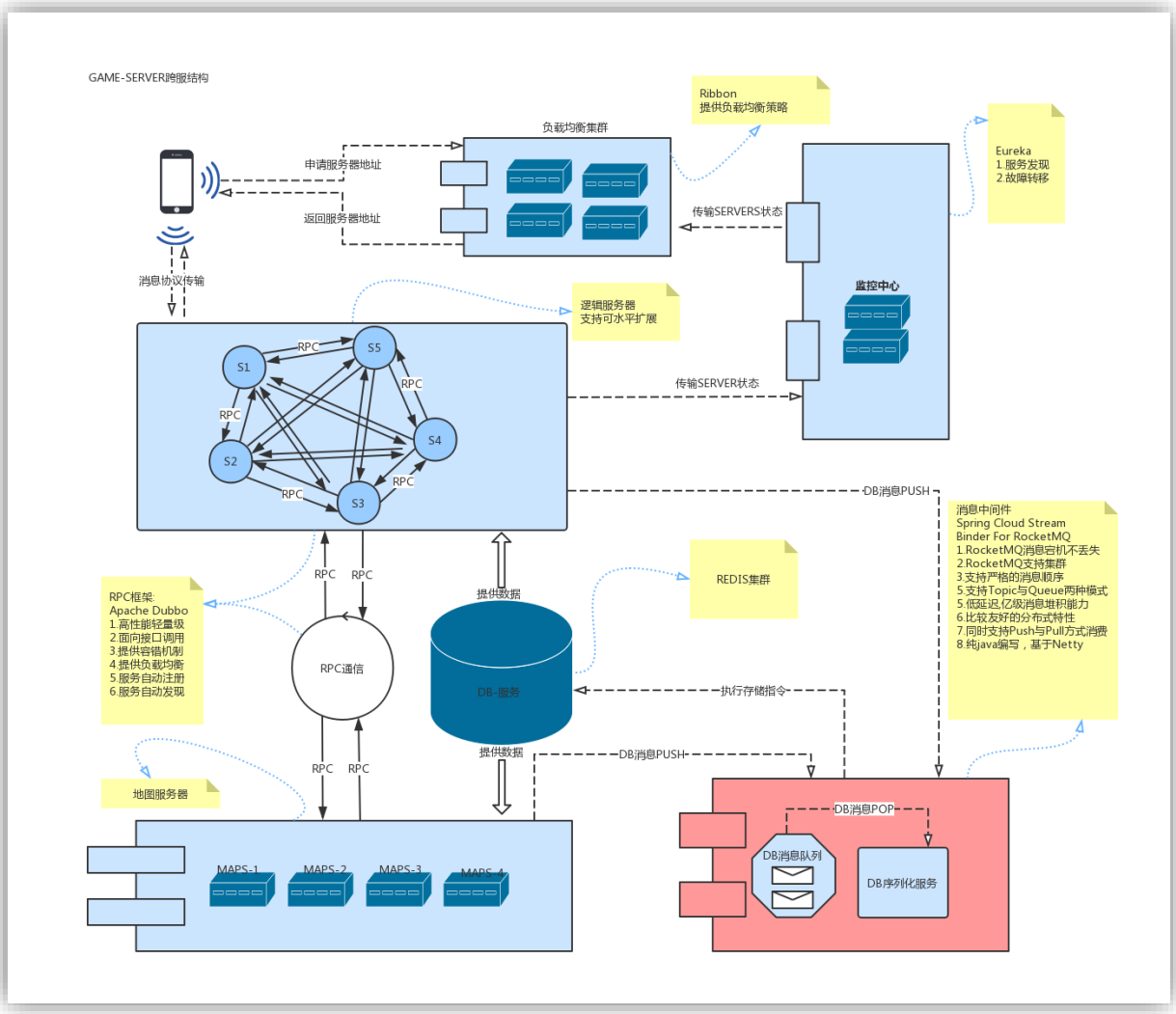
范兴，10 年以上游戏从业经验
参与研发多款 SLG，MMOARPG 等多类型游戏
历任服务端主程，服务端技术经理，技术总监，JAVA 技术专家等职位
《塔防三国志》技术经理，注册玩家超过一亿，DAU 超过 600 万，总流水超过 12 亿
擅长游戏底层架构设计与实现

项目架构图

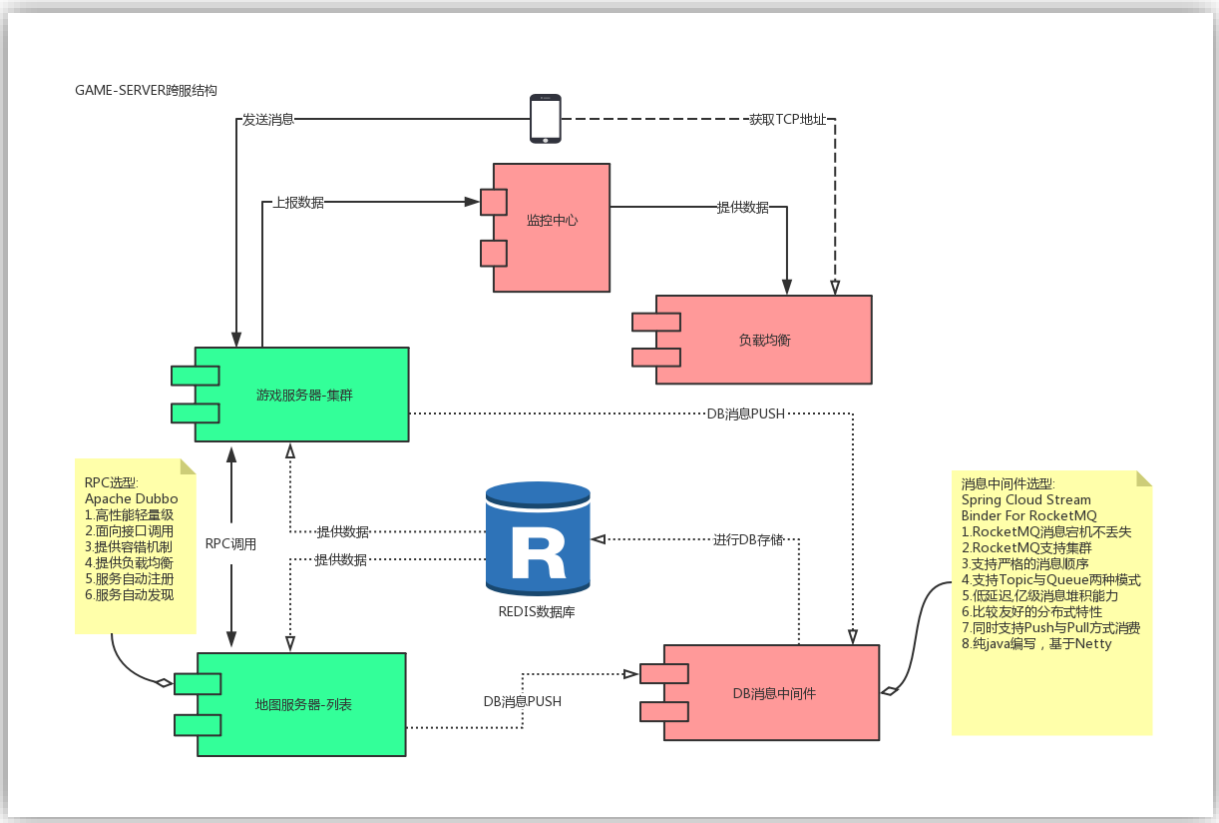
单服架构



跨服架构


GAME-SERVER跨服结构
Ribbon
提供负载均衡策略
负载均衡集群
申请服务器地址
返回服务器地址
Eureka
1.服务发现
2.故障转移
传输SERVERS状态
消息协议传输
监控中心
逻辑服务器
支持可水平扩展
RPC
S1
S2
S3
S4
S5
传输SERVER状态
DB消息PUSH
消息中间件
Spring Cloud Stream
Binder For RocketMQ
1.RocketMQ消息宕机不丢失
2.RocketMQ支持集群
3.支持严格的消息顺序
5.支持Topic与Queue两种模式
5.低延迟,亿级消息堆积能力
6.比较友好的分布式特性
7.同时支持Push与Pull方式消费
8.纯java编写，基于Netty
RPC框架:
Apache Dubbo
1.高性能轻量级
2.面向接口调用
3.提供容错机制
4.提供负载均衡
5.服务自动注册
6.服务自动发现
RPC通信
提供数据
REDIS集群
DB-服务
执行存储指令
地图服务器
MAPS-1
MAPS-2
MAPS-3
MAPS-4
DB消息POP
DB消息队列
DB序列化服务

简化版跨服架构



授课安排

第一节课：第一个游戏服务器程序——基于 Netty 实现		
		简单的自我介绍；
		给同学展示过去做过的项目： # 人人游戏——钢铁元帅； # 腾讯应用中心——回到三国志；
		介绍一下目前正在研发中的项目： # 项目白皮书
	有演示	演示我们将要学习的项目——给学生一个可见的确定的学习目标； # 演示地址 http://cdn0001.afrxvk.cn/hero_story/v1.3.1/index.html
		介绍游戏项目常用的开发工具和框架： # 前端：Unity、Cocos、Unreal、Egret、Spine、DragonBone； # 后端：C/C++、Java、Python、JavaScript、GoLang；
	手敲代码	创建 Maven 项目并添加 Netty 依赖包； # 需要增加阿里 Maven 镜像； # 解释一下 Maven 的节点配置； # 在 IDEA 中刷新 Maven，可以看到用户目录下的 .m2 文件；
	手敲代码	手写第一个游戏服务器程序（Netty 的第一个例子，要最简单的那种）；
		逐行解释每一句代码；
	有图	介绍 Netty 的大致运行原理，需要有图；
	有故事	用大白话，也就是用一个故事来解释 Reactor 模型（以一个小餐馆为例子，说明 BossGroup、WorkerGroup、事件驱动）；
		阐述客户端与服务器端的通信过程，让学生知道有“消息协议”这件事；
		Google Protobuf 入门： # Protobuf 安装； # Protobuf 协议格式（需要讲述一下消息的嵌套）； # Protobuf 命令生成 Java 代码；
		《Hero Story（英雄传说）》消息协议文档；
	手敲命令行	生成 Java 代码；
	手敲代码	增加解码器；
		拦截 UserEntryCmd 指令，打印用户信息；

第二节课：让游戏中的用户互动起来		
	手敲代码	增加编码器；
	手敲代码	增加广播逻辑，增加 _channelGroup；
	手敲代码	完成 UserEntryCmd 指令逻辑，可以看到新来的用户；
		提问一个思考题:就是先进入游戏的用户,可以看到后进入游戏的用户。但是反过来,后进入游戏的用户却无法看到先进入游戏的用户。这是为什么?
		答案是还需要增加 "还有谁" 的逻辑（带同学看一下 Protobuf 文档）；
	手敲代码	增加 User 类；
	手敲代码	增加 Map<Integer, User> 字典；
	手敲代码	在 收到 UserEntryCmd 之后将用户塞到 Map<Integer, User> 字典中；
	手敲代码	拦截 WhoElseIsHereCmd 指令，是用户可以互相看到对方；
	手敲代码	增加移动逻辑；
	手敲代码	增加退场逻辑；
	手敲代码	增加攻击逻辑；
	手敲代码	增加减血逻辑；
	手敲代码	增加死亡逻辑；
		抛出一个问题：移动时存在 Bug；

第三节课：设计模式在游戏服务器中的应用		
		在解决移动 Bug 之前，先重构代码；
		带着同学们分析有哪些代码是看上去很相似的；
	手敲代码	重构——整理出 Broadcaster 广播员类；
	手敲代码	重构——整理出 UserManager 用户管理器类；
	手敲代码	重构——整理出命令处理器类，包括： # UserEntryCmdHandler； # WhoElseIsHereCmdHandler； # UserMoveToCmdHandler； # UserAttkCmdHandler；
	手敲代码	重构——提取接口 IUserCmdHandler 并使用泛型；
	手敲代码	重构——增加命令处理器工厂类；
	手敲代码	重构——增加消息识别器，并应用到消息编码器和消息解码器中；
		阶段性总结，工厂模式，给出 UML 图，并提出一个问题： # 普通程序员重构到这个程度已经是优秀！ # 但是作为架构师呢？
		结论：作为架构师，必须要考虑"工程"问题，最大程度降低开发难度，开发难度越低，适用性越广；
		引导并提出 Java 反射机制；
	手敲代码	引入 PackageUtil 类，并实现最终重构；
		（？）引申阅读，Java 反射 + javassist 提升 ORM 的效率（有时间的话可以讲这个）；

第四节课：游戏中的移动和攻击——如何设计消息、如何处理多条线程读写同一数据		
		阐述目前移动消息存在的 Bug；
		如何解决？（帧同步？这样会给服务器带来极大压力）；
		给同学们出一道简单的数学问题（追及问题），属于比较轻松的互动环节； # A 点韩梅梅家； # B 点李雷家； # 傍晚韩梅梅从 A 点（也就是自己家里）出发去 B 点（也就是去李雷家）找李雷； # 韩梅梅晚上找李雷要做什么，着不是要问的问题； # 问：如果你在晚上 8:00 的时候出现在韩梅梅的必经之路上，那么你会在什么位置上与韩梅梅相遇？ # 注意：这个问题本身是有缺陷的；
		重新设计消息，并作出解释；
	手敲代码	修改代码并做测试；
		再看看攻击中的跨线程问题，将线程名字打印出来；
		跟同学们一起分析问题；
	手敲代码	写多线程并发测试修改数据；
		提出 synchronized 方案；
		提醒同学注意，这样可能导致性能下降并可能出现可怕的"死锁"；
	手敲代码	写一个 Synchronized 导致死锁的例子；
		提出 Atomic 方案；
	手敲代码	写一个 Atomic 代码例子；
		提醒同学注意，这样会导致代码臃肿，问题还是不能得到有效解决；
		最后提出游戏项目的解决方案，以及游戏项目和 Web 项目对多线程情况处理的差异： # 游戏项目，多用单线程处理业务，模型简单； # Web 项目，多用多线程处理业务，因为请求（HttpRequest）无先后顺序而且又必须满足及时响应（HttpResponse）的需求；
	手敲代码	增加主线程；
		展示线程示意图；
		（？）讲一道 Google 的面试题（建议讲讲这个，开阔思路）；

第五节课：用户登录——游戏中的数据库读写方式		
		带同学看一下登录界面并了解一下登录消息；
	操作数据库	创建 t_user 数据表；
	手敲代码	引入 MyBatis 包并增加登录逻辑；
		IO 读写是在哪个线程完成的？答案是主线程；
		跟同学们一起分析这样做会不会有问题？（有！如果一下子涌进很多用户，那么就排队处理，最后服务器就被 IO 堵死了）； # 新用户登录不进来； # 老用户在游戏里也都动不了了；
		怎么解决呢？——通过一个专门的 IO 线程来解决；
	手敲代码	增加 AsyncOperationProcessor 和 IAsyncOperation；
		但问题还是没有根本解决！ # 新用户还是登录不进来； # 老用户倒是正常了；
		原因是 IO 是单线程的，所以 IO 必须扩展成多线程的；
		但是扩展出多线程后，有一个问题是必须要保证的，就是： # A 用户的 IO 请求一定都要在同一个线程里执行！这样就不会导致 A 用户的数据库数据发生脏读写；
	手敲代码	给 IAsyncOperation 增加 BindId；
	手敲代码	改造 AsyncOperationProcessor 增加多线程并增加绑定逻辑；

第六节课：利用消息队列完成排行榜功能；		
		介绍一下需求场景；
		提出设计思路： # 记录玩家的胜利次数，刷新排名数据，在 UserAttkCmdProcessor 中实现； # 提出质疑，如果策划增加需求怎么办； # 业务逻辑就会因此变复杂、排行榜逻辑会洒在各个角落； # 给出一个完善的设计思路；
		给出设计图并解释；
		消息协议设计；
		引入 Redis，并介绍将要用到的功能： # set、get； # zrange、zrevrange；
		引入 RocketMQ，并介绍要用到的功能；
		敲代码实现生产者；
		敲代码实现消费者，RankApp；
		接通客户端消息；

第七节课：游戏服务器日志处理；		
		介绍一下需求场景；
		引入 LogStash；

第八节课：打包和部署游戏服务器；		
		第一次部署，需要登录远程服务器创建目录： # 创建并介绍目录结构，一定要使用相同的目录结构； # 需要添加 .ssh 公钥；
		Linux Shell 入门： # git clone； # mvn； # scp； # 然后登录到每一个远程服务器上执行解压缩命令（这一步也可以另外写脚本来实现）；
		游戏服务器的更新流程；
		引入 Ansible： 介绍 Ansible 的基本使用命令； 介绍 Ansible-Playbook（host、.yml 及语法）；
		运行 ansible；
		有没有一个可视化的界面？
		介绍 Jenkins： # 安装，下载 .war 文件，并放到 Tomcat 服务器上； # 启动 Jenkins，启动前需要输入一个 Key； # 创建第一个 Project；
		添加 CheckBox 组件，可以指定更新某个服务器；

参考资料

1. 关于 Netty：http://www.sohu.com/a/272879207_463994