**项目说明书**

**请注意：**

**此说明书将用于立项评估，请详细填写，根据填写内容会影响“研发配合度评分”。**

目录

[1. 背景 4](#_Toc528591070)

[2. 术语 4](#_Toc528591071)

[3. 系统概述 5](#_Toc528591072)

[3.1. 运营环境 5](#_Toc528591073)

[3.1.1. 服务器软件环境 5](#_Toc528591074)

[3.1.2. 服务器硬件环境 6](#_Toc528591075)

[3.1.3. 运营环境 7](#_Toc528591076)

[3.1.4. 所⽤外部组件 7](#_Toc528591077)

[4. 系统设计 8](#_Toc528591078)

[4.1. 系统整体介绍 8](#_Toc528591079)

[4.2. 游戏架构 8](#_Toc528591080)

[4.3. 游戏各模块分析 9](#_Toc528591081)

[4.3.1. 全局服务器 Global Server 9](#_Toc528591082)

[4.3.2. 登录/支付中转服务器 Barrack Server 9](#_Toc528591083)

[4.3.3. 核心逻辑服务器 Zone Server 9](#_Toc528591084)

[4.3.4. 区管理服务器 Manager Server 9](#_Toc528591085)

[4.3.5. 社交服务器 Relation Server 10](#_Toc528591086)

[4.3.6. 网关服务器 Gate Server 10](#_Toc528591087)

[4.4. 分区方式和扩容方式 10](#_Toc528591088)

[4.4.1. 扩容过程对用户的影响和扩容实现方式 10](#_Toc528591089)

[4.4.2. 分区分服的实现方式 10](#_Toc528591090)

[4.4.3. 是否支持合服，合服的实现方式 10](#_Toc528591091)

[5. 系统设计要点 11](#_Toc528591092)

[5.1. 安全策略 11](#_Toc528591093)

[5.1.1. 防外挂机制 11](#_Toc528591094)

[5.1.2. 防 DDoS 策略 11](#_Toc528591095)

[5.2. 负载均衡策略 12](#_Toc528591096)

[5.3. 容灾策略 12](#_Toc528591097)

[5.4. 回档策略 12](#_Toc528591098)

[5.5. 中手游容灾标准 13](#_Toc528591099)

[5.6. 过载保护策略 13](#_Toc528591100)

[5.7. 模块功能划分 14](#_Toc528591101)

[5.8. 数据扩容 14](#_Toc528591102)

[5.9. Cache机制 14](#_Toc528591103)

[6. 业务流程 14](#_Toc528591104)

[6.1. 登录流程 14](#_Toc528591105)

[6.1.1. 渠道用户验证流程如下： 14](#_Toc528591106)

[6.1.2. 游戏用户登录 16](#_Toc528591107)

[6.2. 创建角色流程 17](#_Toc528591108)

[6.3. PVE /PVP战⽃流程 18](#_Toc528591109)

[7. DB 分析 18](#_Toc528591110)

[7.1. DB存储分析 18](#_Toc528591111)

[7.2. DB负载分析 18](#_Toc528591112)

[8. 客户端 18](#_Toc528591113)

[8.1. 概况 19](#_Toc528591114)

[8.2. 设备支持 19](#_Toc528591115)

[8.3. 安全 19](#_Toc528591116)

[8.4. 弱网络处理 19](#_Toc528591117)

[8.4.1. 关键包丢包的处理 19](#_Toc528591118)

[8.4.2. 网络延迟增大的处理 20](#_Toc528591119)

[8.4.3. 断线重连的处理 20](#_Toc528591120)

[8.5. 版本和补丁的更新策略 20](#_Toc528591121)

[8.6. 机型适配 20](#_Toc528591122)

[8.6.1. UI 适配方案 20](#_Toc528591123)

[8.6.2. 运行适配方案 20](#_Toc528591124)

[8.7. 架构 20](#_Toc528591125)

[8.7.1. 架构图 21](#_Toc528591126)

[8.7.2. 关键模块说明 21](#_Toc528591127)

[9. 系统风险分析 22](#_Toc528591128)

[9.1. 罗列系统中存在的风险 22](#_Toc528591129)

[9.2. 异常情况下的游戏表现 22](#_Toc528591130)

# 背景

⼀款基于Unity 3D 引擎的移动平台上的MMORPG游戏。

# 术语

术语和定义

|  |  |
| --- | --- |
| **术语** | **解释** |
| BI 服 | 处理运营数据分析 |
| SDK服 | 处理与渠道之间登录/支付业务，对游戏服屏蔽不同渠道差异 |
| GM服 | GM管理后台（可以部署在提审服上节省运维成本） |
| MySql | MySql，保存用户核心数据 |
| Redis | Redis，负责保存社交类及排行榜相关业务 |
| FTP | 保存MySql/Redis全备数据，游戏服日志等 |
| ELK | 游服日志分析 |
| LogDB | 玩家行为日志埋点，用于GM后台查询玩家行为 |
| Zabbix | 服务器运行状态监控 |
| 资源服 | 下载区服列表及作为热更资源的CDN源站 |
| 异地FTP | 异地容灾备份FTP服务器（非必须） |
| 游服-Global | 全局服务器，负责游服集群管理，负责注册发现，负责GM转发等业务 |
| 游服-Barrack | 负责游戏内登录及支付业务，向SDK服发送请求，SDK与具体渠道交互完成后将结果返回给Barrack，Barrack将结果再转发给实际业务服务器进行后续操作。对于登录业务，Barrack还承担均衡负载功能，为即将登录的玩家均衡负载一个Gate供其连接 |
| 游服-Gate | 网关服务器，接收Client的TCP长连，负责登录验证，通信加解密，转发Client请求到具体Zone上 |
| 游服-Zone | 玩家养成/战斗等核心业务，挂载游戏内地图，一个游戏区可配置多个Zone承载玩家核心业务，支持动态伸缩 |
| 游服-Manager | 主要承担Zone服务器的均衡负载 |
| 游服-Relation | 承担社交类相关工作，如家族，组队等业务 |

# 系统概述

## 运营环境

### 服务器软件环境

|  |  |  |
| --- | --- | --- |
| **软件名称** | **版本** | **备注** |
| CentOS | 6.9以上 | 64bit |
| MySql | 5.7 | 不建议上8.0 |
| Redis | 5.0 | 建议5，不能低于4 |
| Windows Server | 2012 R2 |  |
| Nginx | 1.8 | 用于Http业务均衡负载 |
| ELK | 6.2.4 | 游服log收集与检索 |
| C# | .NetFrame 4.5以上 | 游戏服开发语言 |
| Java | 1.8 | SDK服务器开发语言 |
| Zabbix | 3.0.7 |  |

### 服务器硬件环境

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **服务器名称** | **型号** | **CPU** | **内存** | **硬盘** |
| BI服（自备） |  | 8 core | 32GB | 1T SSD |
| SDK服 |  | 8 core | 16GB | 512GB |
| GM服 |  | 1core | 2GB | 100GB |
| MySql |  | 8 core | 24GB | 512GB SSD |
| Redis |  | 4 core | 16GB | 512GB |
| FTP |  | 2 core | 4GB | 2T |
| ELK |  | 4 core | 8GB | 1T |
| LogDB |  | 4 core | 8GB | 1T |
| Zabbix |  | 2 core | 4GB | 256GB |
| 资源服 |  | 2 core | 4GB | 128GB |
| 异地FTP |  | 2 core | 4GB | 2T |
| 游服Global+Barrack集群  （部署在同一物理机） |  | 4 core | 8GB | 100GB |
| 游服-Zone集群+Manager+Relation  （部署在同一物理机） |  | 8 core | 16GB | 100GB |
| 游服-Gate |  | 4 core | 8GB | 100GB |

### 运营环境

|  |  |
| --- | --- |
| **情况** | **结果** |
| 是否有开发人员熟悉 iOS 或者 Android 并且具备成熟的开发能力 | □ 是 |
| 是否之前有接入过其他平台SDK的经验?（ Android 和 iOS 均可），如果有请列出 | □ 是  渠道 :AppStore，国内主流安卓渠道（应用宝/小米/360/硬核联盟等100+渠道） |
| 完备的运营后台 | □ 是 |
| 游戏是否为分区或分服游戏 | □ 是 |
| 是否支持合服 | □ 是 |
| 合服操作，需要的时间 | □ 小于 1 小时 |
| 玩家行为异常告警机制 | □ 是 |
| 核心玩法通信协议 | TCP |
| DDoS 时可屏蔽 UDP 协议 | □ 是（无基于UDP的业务，可以挂高防直接清洗UDP数据报） |
| 整个系统的可用性是否不依赖于某单个server的可用性 | □ 是（只有全局服务器Global是单点，不负责游戏具体业务，Global出现问题不影响线上玩家，不影响已开放的服务器，但无法开设新服，Global机器出现问题可以在线迁移，将Global域名绑到新机器后集群内其他服务器会自动发现重新注册） |
| 影响整个系统可用性的服务（上个问题为否时填写） |  |
| 完善的监控 | ☑ 黑盒监控（Zabbix） |

### 所⽤外部组件

|  |  |
| --- | --- |
| **外部组件名** | **备注** |
| 融合 SDK | U8 融合 SDK企业版 |
| UWA | Unity性能分析工具 |
| 腾讯 bugly | 崩溃监测 |
| 神策 | BI统计平台 |
| Test Bird | 测试平台 |

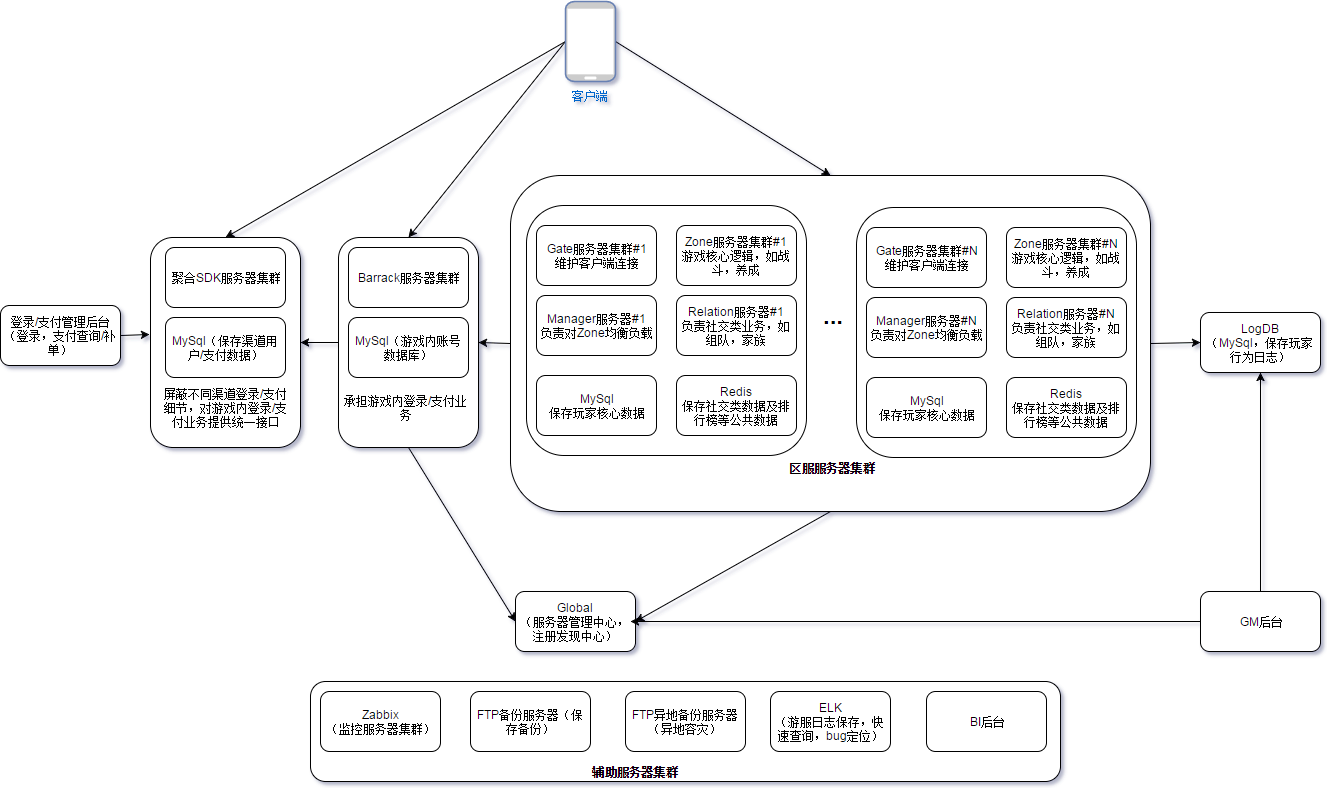
# 系统设计

## 系统整体介绍

|  |  |
| --- | --- |
| 游戏分区模式 | 一服一套Gate集群 + Zone集群+ Manager 单点 + Relation单点  整个集群共享Global + Barrack集群 |
| 前后台通讯类型 | Tcp + Protobuf + BlowFish |
| 连接类型 | 长连接 |

## 

## 游戏架构

****

## 游戏各模块分析

### 全局服务器 Global Server

* 整个集群只有一台Global服务器。
* 作为整个集群的管理中心，承担整个服务器的管理业务，如关闭某服务器，设置某服务器帧率，设置排队人数，设置白名单开启等等。
* 作为整个集群的注册中心，所有服务器启动后通过域名dns解析与Global建立连接。如Server A主动连接Global后，向其注册自身服务器信息，Global会根据配置的网络拓扑结构通知A它应该连接的其他服务器信息，也会将A的信息通知给集群内需要主动连接A的服务器集群，实现整个集群内的注册与服务发现。
* 承担GM后台与游服集群通信的转发业务，如GM后台冻结某个玩家，则客服人员通过访问GM，点击冻结后，GM后台发送协议到Global，Global根据具体的区服id和角色uid找到该大区的Manager服务器，发送冻结请求给Manager处理。

### 登录/支付中转服务器 Barrack Server

* 整个集群共享一组Barrack，支持动态伸缩。
* 承担玩家登录时与SDK服务器的转发和认证业务，认证通过后根据选择的服务器及对应的Gate集群状态，选择一个合适的Gate并新生成token，将玩家信息及token发给该Gate，将该Gate的ip和port，刚生成的token返回给玩家（更多细节详见6.1登录流程）。
* 承担玩家支付时与SDK服务器的转发与认证，并将支付结果转发给对应的Manager，Manager收到支付通知后，根据结果判断是否执行发币业务。

### 核心逻辑服务器 Zone Server

* 一个区下有一组Zone集群，支持动态伸缩。
* Zone挂载具体的游戏场景地图，负责在该游戏场景内的绝大部分业务，如战斗，任务，强化养成等等。
* Zone具体负责的游戏场景，通过配置实现。Zone1 挂载Map 1，分线为1-3的主城，Map Id为2，分线为1-4的新手村；Zone2挂载分线为4-6的主城，分线5-8的新手村。
* 一个区下有一组Zone集群，可以动态配置Zone的挂载地图信息及与其他服务器连接的端口后，在线动态伸缩。如在Zone1，Zone2有6个主城地图不够吃时，可以配置Zone3挂载主城分线为7-12，启动Zone3即可为后上线的玩家分担主城地图的压力。
* 所有战斗计算逻辑均在Zone上执行，自然能抵御封包攻击。

### 区管理服务器 Manager Server

* 每个区下有且只有一个Manager。
* Manager了解每个Zone上的人数，帧率，睡眠状态，了解每个Zone上的每个地图每条分线有多少人。当玩家申请进入某个Map时，Manager会根据整个集群内该Map的健康状况及挂载该Map的对应Zone的健康状况，找到一个合适的Map分配。
* 当玩家切图时，可能存在挂载目标地图的Zone与当前玩家所在的Zone不同的情况，Manager负责玩家在跨Zone情况下玩家数据的传输与管理。

### 社交服务器 Relation Server

* 每个区下有且只有一个Relation。
* Relation维护该区下所有在线玩家的信息，并且知道每个玩家在哪个Zone上。
* 负责在线玩家的社交业务，同一区下不同Zone上的玩家通过Relation进行交互，如好友，家族，组队等业务。
* 负责排行榜的缓存业务，定期动Redis捞取排行榜并缓存，减少Redis访问压力。

### 网关服务器 Gate Server

* 每个区下共享有一组Gate集群，支持动态伸缩。
* 负责与玩家的网络传输业务。
* 负责玩家通信的密钥管理，数据加解密业务。
* 负责将玩家的请求转发给玩家所在的Zone上处理，并将Zone的返回结果转发给玩家。

## 分区方式和扩容方式

### 扩容过程对用户的影响和扩容实现方式

* Barrack/Gate/Zone集群均可在线扩容，对线上用户透明无影响，只有对扩容后再进入游戏的用户进行均衡负载时路由到新扩容的服务器。
* 开新区一般在网关集群新加一组新区的Gate服务器，在游戏逻辑集群添加一组Zone集群+Manager+Relation即可。可通过部署工具配置相应物理机的ip后一键部署新服，对线上用户无影响，运营人员对新服测试通过后即可对外开放，玩家打开游戏，客户端会自动拉去服务器区服列表，玩家即可看到新区并登录游戏。

### 分区分服的实现方式

* 一个区下：一组Gate + 一组Zone + 单一Manager + 单一Relation

### 是否支持合服，合服的实现方式

* 支持合服。

合服方式为编写合服工具，以将2服合并到1服为例：

* MySql数据合并如下：

1. 1服和2服的数据库备份。
2. 先筛选出2服符合合服条件的用户群（长期流失玩家的数据不合并到1服），将该用户群数据直接select into到1服数据库（角色uid会根据服务器进行偏移，不同区的player uid不同，直接插入不会产生主键冲突）。
3. 根据运营需要将部分公共类数据重置，保证公平。
4. 配置服务器列表路由配置，2服登录的玩家可以选择1服或2服的登录入口，统一进入1服的游戏内。

* Redis数据合并如下：

1. 1服和2服的Redis备份。
2. 清理2服的公共类数据，比如排行榜等数据。
3. 将2服的AOF文件cat到1服AOF文件尾。
4. 1服重新执行一遍AOF。
5. 1服清理公共类数据，如排行榜。

* 合服后当然有开白名单做回归测试等操作，此处不再赘述。

# 系统设计要点

## 安全策略

### 防外挂机制

* 通信协议加密机制。
* 使用 PB协议，使用 BlowFish加密，动态生成BlowFish密钥。使用非对称私钥对BlowFish密钥进行加密后通知Client，Client用公钥解出BlowFish密钥后进行数据包加密。
* 服务器校验机制。
* 战斗同步方式为状态同步，战斗计算在后端进行，客户端负责演出计算结果。
* 对于封包攻击导致BlowFish解密失败，达到一定次数后录入MySql，供客服人员处理。

### 防 DDoS 策略

* 上云平台高防，清洗SDK服及Barrack集群对外域名的流量。
* 多个服的网关服务器Gate在实际生产环境中可以部署在一台物理机，减少对外暴漏ip数量，可以将该ip也上高防。服务器内部通信走内网。
* 在上高防之前可能存在DDoS攻击导致云厂商触发保护机制，沉默某些Gate集群对外ip，针对此情况可直接新申请云主机，一键部署新Gate集群，受影响的玩家只需再次登录即可进入游戏。整个游戏除了被沉默的Gate ip，其他游戏服不受影响，可以在线迁移。被沉默的Gate对应的Zone+Manager+Relation不受影响，新Gate启动后会自动发现，自动注册，继续服务。
* 在16年圣诞期间，我们港台服遭受了峰值30Gb流量的DDoS，持续近一天，除被沉默的新服ip要重新部署一遍后，其他服务没有重启，老服玩家不受影响。在17年初港台服再次遭到DDoS攻击，收到云平台预警消息后，20分钟内切换到高防代理，基本无影响。

## 负载均衡策略

* Zone集群负责承载该分区的核心业务，Manager根据每个Zone的状态及每个Zone下该地图的负载情况将新登录或进入地图的玩家分配到适合的Zone上。如果某个区玩家人数猛增，可以临时配置新Zone挂载对应地图后一键部署启动，在线扩容。
* 每个区有Gate集群作为网管服务器承担Client的连接。Gate会定期上报当前状态（帧率，睡眠时间，在线人数）给Barrack集群，Barrack集群在验证玩家登录同时会分配一个Gate供其连接，如果某个区的玩家猛增，可以临时增加该区的Gate，在线扩容。
* Barrack服务器无状态，Client在使用SDK登录时访问域名，Barrack可挂在Nginx后进行均衡负载，同样可以对Barrack在线扩容，Barrack承担的业务以转发为主（使用.Net FrameWork 异步线程池），基本上无扩容需求。

## 容灾策略

* MySql主从备份（如有需要可做vip主主备份或MHA），每日增量备份，每周一次全备，异地备份保存。
* Redis 选用AOF持久化策略，主从备份，每日备份，异地备份保存。数据量到达一定规模可平滑扩展为集群。
* 游戏服务器对持久化容灾策略：当与MySql或Redis网络连接出现断开，且2分钟内均无法重连成功则自动退出进程，防止玩家回档过多。
* 上一款MMORPG（迷城物语）运营期间（国内20万DAU，海外10万+DAU），除因内网抖动导致MySql断连且无法重连导致程序退出外，没有出现后端崩溃情况。

## 回档策略

* 回档时间需根据恢复数据量及备份集大小等因素决定。
* MySql回档策略，保留10天binlog，十天内的数据可通过binlog和备份集回滚到指定时刻。
* Redis回档策略，不支持回退策略。只能通过解析aof文件，查看执行错误命令之前是否有处理相同key的命令，再确定是否对其修改，由于重写触发的原因，实际操作不可控。
* MySql与Redis每天会做一次备份，如果直接使用备份做回档，通常不会超过30分钟。

## 中手游容灾标准

|  |  |  |
| --- | --- | --- |
| 不可接受 | 单点风险：单个程序失效导致超过 10% 的玩家无法游戏 | ☑ 不存在 |
| 意外停服风险：故障切换时需要停止所有区服并重启以重新刷新配置 | ☑ 不存在 |
| 长时间回档风险：因为数据变更未及时落地存储，导致可能发生 10 分钟以上的数据丢失 | ☑ 不存在 |
| 重要数据异常风险：任何可能导致玩家重要数据（例如货币、高价值装备）错误的风险 | ☑ 不存在 |
| 可接受 | 分布式架构中，单个程序失效可能导致不超过 10% 的玩家无法游戏 | ☑ 不存在 |
| 单个程序或机器故障无法自动切换，需要人工完成故障切换 | ☑ 不存在 |
| Cache 延迟写方案，意外故障导致不超过 10 分钟的数据回档 | ☑ 不存在 |
| 完备 | 不存在以上风险，各系统有全面的风险应对预案和故障自动切换措施 | ☑ 符合 |

## 过载保护策略

* 开服导量阶段有排队机制，如1服人数到达3000人开启排队，每3秒放入一位玩家，到达3500人时进入满员状态，每下线一位玩家才会放入一位玩家。对3000和3500人数的设定可通过后台随时修改。排队机制也可通过后台随时开启和关闭。在1服排队过程中，当然不会影响2服正常放人。
* 可以一键部署新服，临时增开新区。当然更建议先将新服部署好，只针对内部人员可以开放测试，测试通过后可根据导量情况随时对外开放。
* 针对MMO广播的小包整合，比如通知AOI内人物/宠物/坐骑移动，出现/消失等，同一帧内合并发送，减少碎包。
* 如果过载导致网络延迟，针对待发送缓冲队列的缓冲数量，有对应的机制对非重要协议进行强制丢弃AOI广播包。如待发送队列到达500，则丢弃1/4的AOI广播包，队列达到700则丢弃1/2的AOI广播包，到达1000则丢弃所有AOI广播包。

## 模块功能划分

* 功能模块逻辑上是独立的，比如登录，登出，切图，战斗（技能，buff，宠物，坐骑），寻路，AI，任务，养成，组队，好友，家族。

## 数据扩容

* MySql采用一个区一个Database实例，不同区可以在不同物理机上部署，当然也可以在同一台物理机部署节省运维费用。如果在后期合服导致某个服的Database数据量过大，游戏服支持逻辑层分片，针对具体数据库表做路由，不需依赖MySql分片中间件。
* Redis采用一个区一个实例，不同区可以在不同物理机上部署，当然也可以在同一台物理机部署节省运维费用。如果在后期合服时某个Redis数据量过大，可以将该Redis集群化即可。Redis4.0以上支持在线自动分片。
* 实际上针对MMO的业务数据量（线上已有40多个服合成一个服），单个MySql或Redis毫无压力，无分片必要。

## Cache机制

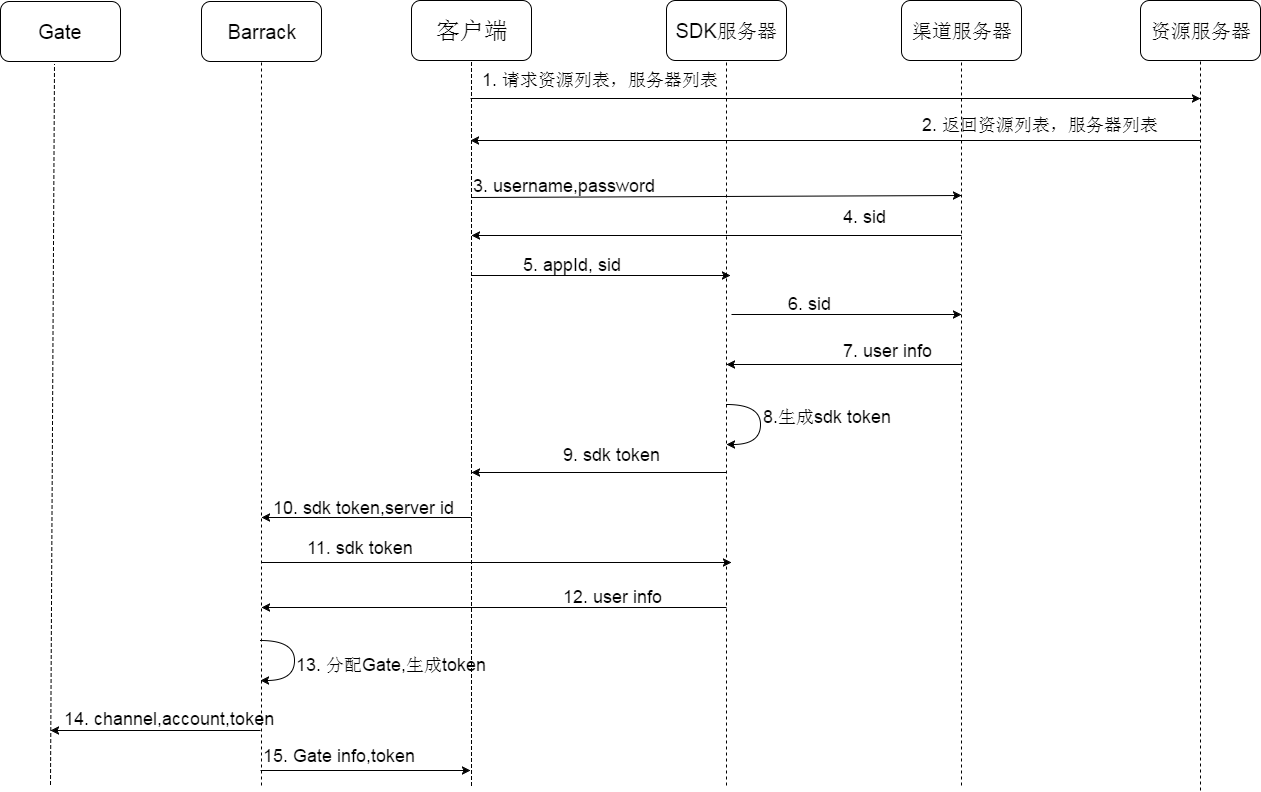
* 社交相关数据保存在Redis中（如好友列表，排行榜）。
* 玩家下线，Zone会保存玩家数据15分钟，15分钟内再次登录会被路由到同一个Zone内，无需访问MySql加载玩家数据。
* 针对重要数据，比如钻石，vip经验，物品（获得，删除，堆叠数量变化，属性变化）等敏感数据实时同步。
* 针对非重要数据且频繁变更数据，如金币，经验，部分计数器（如任务计数器），在下线，切图，升级，及定期（5分钟）情况下持久化。

# 业务流程

## 登录流程

* 登录流程分为两大部分，第一部分为**渠道用户验证**，即玩家使用渠道帐号登录渠道，游戏服务器向渠道认证该用户的合法性；第二部分为**游戏用户登录**，即玩家登录真实游戏世界。

### 渠道用户验证流程如下：



1. 客户端向资源服务器请求资源列表和服务器列表。
2. 资源服务器返回资源列表和服务器列表，客户端根据资源列表判断是否需要热更，强更；解析服务器列表向玩家展示服务器分区信息。
3. 客户端向渠道发送渠道的帐号和密码。
4. 渠道服务器验证该用户合法性，通过后返回sid。
5. 客户端向SDK服务器发送app id， sid。
6. SDK服务器转发至该渠道。
7. 渠道服务器验证sid通过后，返回SDK服务器该用户的用户信息。
8. SDK服务器为该用户生成sdk token。
9. SDK将sdk token发送至客户端。

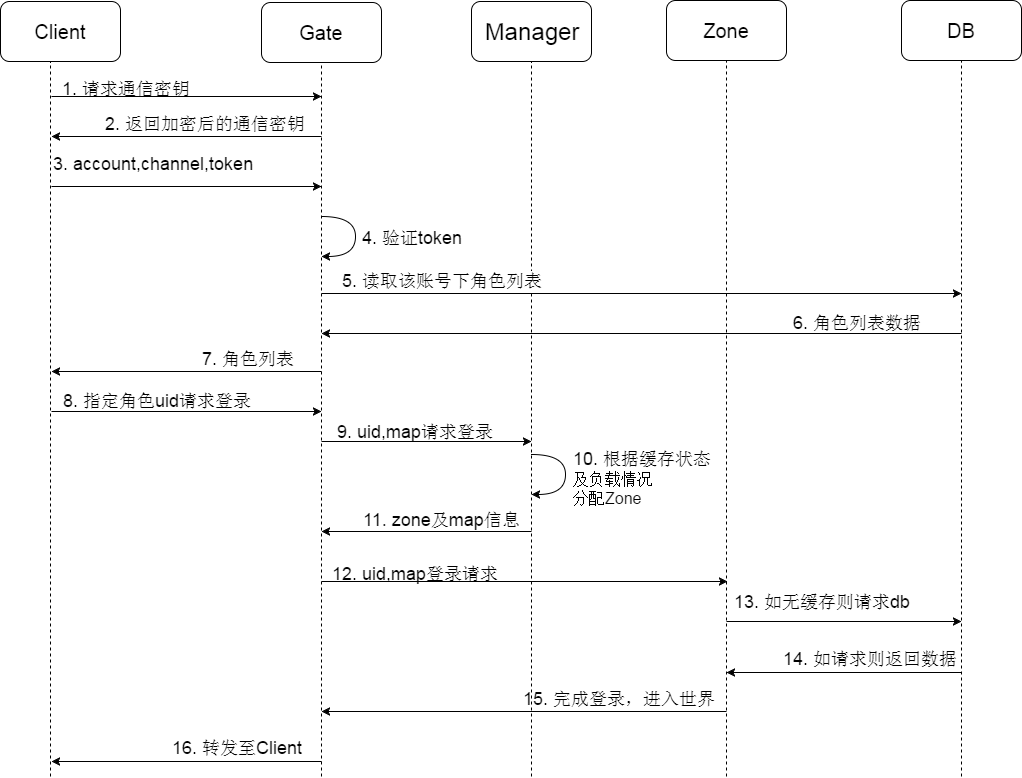
返回的sdk token同时也返回了账号信息，客户端此时应弹出服务器列表，并根据缓存情况展示上次登录服务器。

如果因更换设备或者大更导致没有缓存，可在点击最近登录服按钮时向后端发送查询最近登录服请求。

1. 客户端选取要登录的区服，点击登录，向Barrack发送token及server id
2. Barack将该token转发至SDK服务器验证合法性。
3. SDK验证通过后，向Barrack返回该用户的信息。
4. Barrack收到SDK验证通过信息后，为该玩家生成登录token及分配一个Gate。
5. Barrack将该玩家的渠道，渠道帐号和登录token发送至该Gate。
6. Barrack将新分配的Gate对应的ip，port与登录token返还给客户端。

至此，客户端已经有了接下来要去连接的Gate信息及登录验证token；而同时Gate服务器也有了该玩家的帐号，渠道及登录token信息。接下来玩家即可连接该Gate登录到游戏世界。

### 游戏用户登录



1. 客户端与Gate连接后，首先向Gate请求通信密钥。
2. Gate生成BlowFish密钥，并通过RSA私钥加密后返还给客户端。

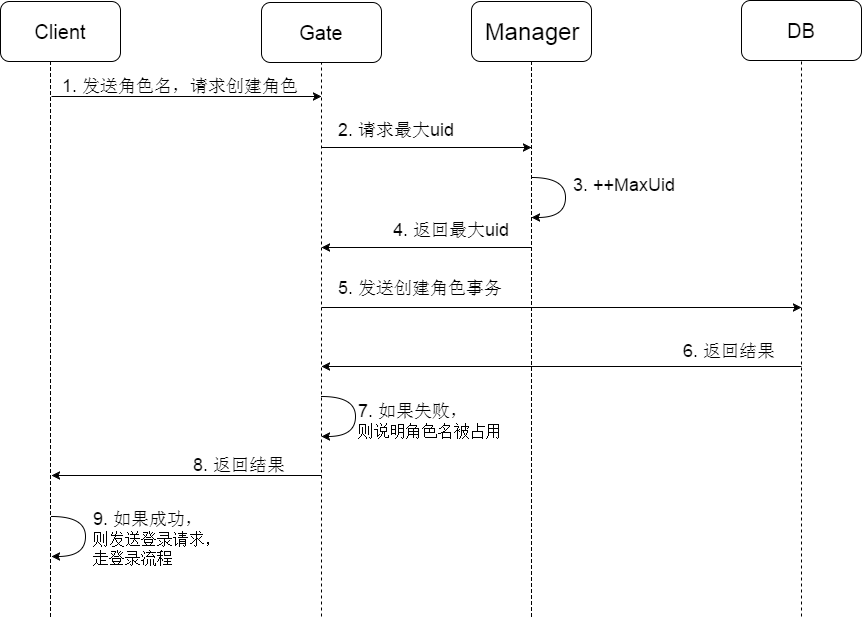
客户端通过RSA公钥接触BlowFish私钥后，后续通信均需要经过加密后传输。

1. 客户端发送帐号，渠道，登录token，请求登录。
2. Gate验证token合法性。
3. Gate从MySql拉取该帐号下的角色列表。

**说明：渠道帐号及渠道名组成游戏内的唯一帐号，如华为渠道下账号为10001的用户，在游戏内的帐号即为10001@huawei。如果Gate从MySql读取该帐号时，该帐号不存在则自动插入该帐号，实现注册帐号的功能。关于注册功能不再赘述**

1. MySql将角色列表信息返还给Gate。
2. Gate将角色列表信息发送给客户端。
3. 玩家选举某个角色登录，将该角色的uid发送给Gate。
4. Gate根据该uid及上次下线的map id，向Manager发送分配Zone请求。
5. Manager分配一个Zone（如果有保存该玩家离线缓存的Zone则直接分配该Zone，如果没有则根据Zone集群状态分配一个适合的Zone）。
6. Manager将该Zone及对应的map信息返还给Gate。
7. Gate向该Zone为该玩家发送进入世界请求。
8. Zone从db读取该玩家数据（Zone会判断是否有该玩家上次的离线缓存，如果有的话则直接执行步骤15，无需向db请求数据）。
9. MySql向Zone返回该玩家数据。
10. Zone创建该玩家对象，进入游戏世界，完成登录，并将结果数据返还给Gate。
11. Gate将数据转发给客户端，至此玩家完成登录，进入游戏世界。

## 创建角色流程



1. 客户端向Gate发送创角请求。
2. Gate向Manager请求当前角色最大uid。
3. Manager将MaxUid自增。
4. Manager将自增后的MaxUid返回给Gate。
5. Gate将创建角色的所有sql封装为一个事务，发送给MySql。
6. MySql执行事务，返回结果。除去系统级别错误（如磁盘空间不足等），应用层执行该事务失败的唯一可能就是用户名用户名重复。
7. Gate处理MySql执行结果。
8. Gate将创建角色的结果通知客户端。
9. 客户端判断创角结果，如果成功则发送登录请求，即可进入世界。

## PVE /PVP战⽃流程

* 如果战斗的目标在同一地图（如野外或可以PVP的PK区），直接战斗即可。
* 如果需要进入指定地图或创建副本拉入，则玩家Player所在的当前Zone，称之为CurZone，向Manager请求将该玩家拉入指定Map（副本也是一种Map），Manager根据当前Zone集群状态选出一个Map及对应的Zone，称之为DestZone，将结果返回给CurZone。
* 如果Manager分配的DestZone与CurZone相同，则CurZone直接创建该地图（如有必要，比如创建副本）或者直接拉入（如指定PVP地图）。
* 如果DestZone与CurZone不同，则CurZone将该Player数据通过Manager转发给DestZone，DestZone根据数据创建相同的Player，之后创建地图或直接将该Player拉入指定地图后即可开始战斗。

# DB 分析

## DB存储分析

待开发后期压测。

## DB负载分析

待开发后期压测。

# 客户端

## 概况

|  |  |
| --- | --- |
| 游戏引擎 | Unity 3D |
| 编程语言及用途 | C#（核心战斗），lua（UI） |
| 异常上报 | Bugly |
| 数据采集 | 服务端采集数据，客户端无数据采集 |
| 流畅操作网络延迟要求 | □ 小于 200 ms |
| 核心玩法 10 分钟内流量使用 | □ 小于 1200 KB |
| 客户端 crash 率 | □ 小于 3% |
| 热更新策略 | □ 强制更新 |

## 设备支持

列举游戏支持的最低机型配置和推荐机型软硬件配置。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **平台** | **硬件最低配置** | **硬件推荐配置** | **系统最低版本** | **系统推荐版本** |
| iOS | iPhone5 | iPhone7以上 | iOS 9.0 | iOS 11.0 |
| Android | 1G内存 双核CPU | 2G内存以上 4核CPU以上 | Android4.4 | Android7.0 |

## 安全

|  |  |
| --- | --- |
| **项目** | **现有处理方案** |
| 关键内存加密 | 主要数据保存和验证都使用服务器，客户端无内存加密 |
| 关键配置加密 | 无 |
| 完整性校验 | 相关文件有md5校验，网络通信使用BlowFish |
| 结果校验 | 服务器端验证 |
| 代码混淆、加密 | 打包使用IL2CPP |

## 弱网络处理

### 关键包丢包的处理

* TCP协议本身是保证传输的数据完整性不会丢数据，如果有数据丢失只能是发送和解析过程出现问题。客户端发送消息时会保存到队列中依次发送，如果发送失败会进行一定次数重发，如果超过重发次数上限则舍弃包。接收时如果解析失败，直接舍弃相关包数据。

### 网络延迟增大的处理

* 人物移动客户端会进行预演，如果网络延迟过大，人物依旧可以移动，收到服务器消息后进行校验，如果客户端位置和服务端位置差别较大，则将人物位置重置到服务端保存的位置。
* 网络延迟大时释放技能收到服务器返回才释放成功，否则释放失败。

### 断线重连的处理

* 登录界面连接不到服务器时，重复尝试连接服务器，连接到服务器后显示登录按钮。
* 游戏中断线：自动重连，一定时间内重连成功则可以继续进行游戏，重连失败后，弹出提示框，关闭后返回到登录界面。

## 版本和补丁的更新策略

* 热更新：支持更新配置文件、坐骑、宠物等。每次启动游戏进行强制更新，使用本地资源版本和线上资源版本进行对比，下载相关资源，如果下载过程中断，或者下载文件md5校验失败，有相应提示，并可以重新下载相关资源
* 安装包更新：安装包更新可以支持多个版本的包进入游戏，也可以限制某一个版本的包必须进行强更包才可以进入游戏。

## 机型适配

### UI 适配方案

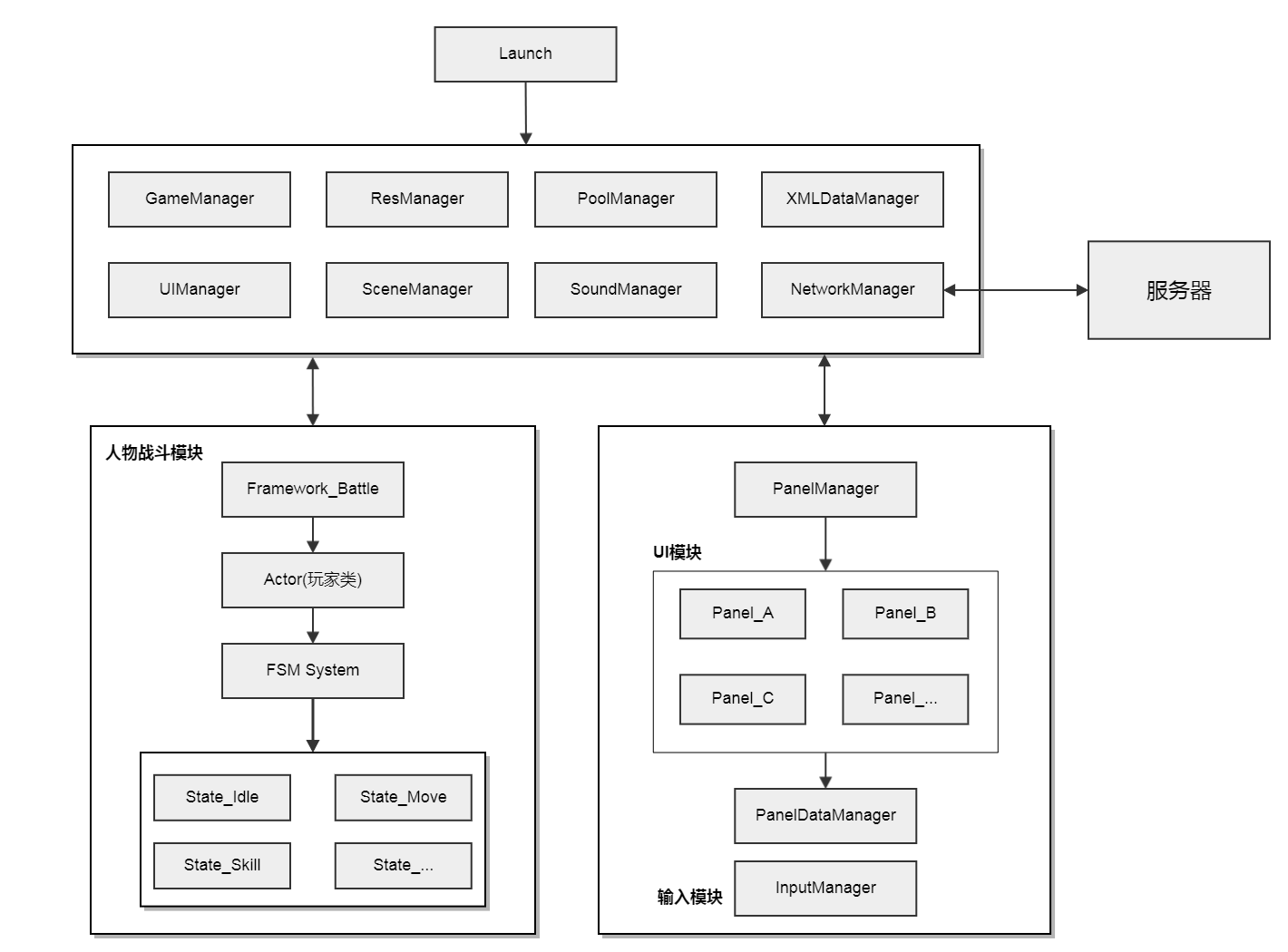
* 设计分辨率为1280\*720，主界面图标根据设备分辨率不同浮动边缘显示，全屏界面以1280\*720为基准，四周黑底处理，非全屏界面直接按比例缩放，居中显示。全屏背景图尺寸为1280\*960，不同设备进行裁剪处理。

### 运行适配方案

* 安卓6.0以上版本使用动态权限申请。
* 对于不同机型进行高、中、低三种模式运行游戏，保证游戏流畅运行。

## 架构

### 架构图



### 关键模块说明

* GameManager：游戏管理模块，所有游戏逻辑总管理模块。
* ResManager：资源加载管理模块。
* UIManager：UI管理模块。
* NetworkManager：网络模块。
* InputManager：处理鼠标、键盘、触屏输入。
* 人物战斗模块：Actor中保存人物信息和FSM，控制人物行动。
* UI模块：PanelManager控制各个功能面板打开和关闭，玩家的操作通过。PanelDataManager处理后和服务端进行通信。
* NGUI：UI插件。
* Protobuf：服务器通信数据格式。
* ShaderManager：渲染模块。

# 系统风险分析

## 罗列系统中存在的风险

* 游戏服务器运行在Windows环境，建议发行商选取高质量云服务商，确保其提供稳定的Windows云主机。

## 异常情况下的游戏表现

* Global宕机，不能新开任何服务器进程，当然也不能开新区，GM部分功能无法使用，游戏内业务不受影响。
* Barrack宕机，正在该Barrack处理的登录和支付业务中断，集群内其他Barrack会承担该Barrack业务，玩家后续操作不受影响。
* Gate宕机，连接该Gate的玩家掉线，且无法断线重连。集群内的Zone会感知到Gate掉线，将立即保存连接该Gate的玩家数据，不会导致数据回档。连接集群内其他Gate的玩家不受影响。同一区下的其他Gate会承担该Gate业务，该区下后续登录的玩家会被分配到其他Gate上。
* Zone宕机，该Zone下玩家掉线。该Zone下的玩家一些非重要数据会回档不超过5分钟。同一区下的其他Zone会承担该Zone业务，该区下后续登录的玩家会被分配到其他Zone上。
* Manager宕机，集群内玩家无法切图，只能在当前地图内游戏，之后的玩家无法登录。
* Relation宕机，社交类服务受到影响，如无法打开排行榜，无法组队等。