# 重度MMO转小游戏经验分享

广州乐牛-林康亮

# 目录

CONTENTS

# 00 叠个甲！！

# 01 | 游戏概况

# 游戏概况-九灵神域

## 仙侠类MMO

## 小游戏H5-资源概况

3D场景接近100个
特效接近2万多个
动作对象1万多个
UI接近3万多张
音效1千多个
Lua文件150MB左右
....

# 游戏概况

## 小游戏H5-当前的内存运行数据

当前内存占用：

选服&选角场景 450MB左右->

新手场景600MB左右->

主城场景800MB左右

内存异常退出分析

| | |
|---|---|
| 1.91% | 45.4分钟 |
| 1.46% | 50分钟 |
| 1.19% | 42.7分钟 |
| 1.64% | 47.5分钟 |
| 0.63% | 50.6分钟 |
| 1.1% | 45分钟 |
| 1.2% | 42.1分钟 |
| 1.04% | 53分钟 |
| 1.5% | 40.2分钟 |
| 1.62% | 52.2分钟 |
| 0.8% | 46.2分钟 |
| 0.4% | 49.6分钟 |

| Process... | Process Name | Responsible Process | User Name | % CPU | CPU Time | # Threads | Memory |
|---|---|---|---|---|---|---|---|
| 899 | com.apple.WebKit.WebContent (899) | launchd (1) | mobile | 25.1% | 29.54 s | 23 | 449.25 MiB |
| 401 | WeChat (401) | launchd (1) | mobile | 3.2% | 3.52 min | 96 | 400.88 MiB |
| 0 | Unknown (0) | Unknown (0) | root | 5.5% | 6.30 min | 382 | 159.33 MiB |
| 844 | com.apple.dt.instruments.dtsecurity (8... | launchd (1) | root | 0.0% | 1.08 s | 2 | 114.31 MiB |

Live Processes

| Process... | Process Name | Responsible Process | User Name | % CPU | CPU Time | # Threads | Memory | Ki |
|---|---|---|---|---|---|---|---|---|
| 899 | com.apple.WebKit.WebContent (899) | launchd (1) | mobile | 68.3% | 45.04 s | 25 | 606.94 MiB | ar |
| 401 | WeChat (401) | launchd (1) | mobile | 12.0% | 3.56 min | 89 | 453.63 MiB | ar |
| 0 | Unknown (0) | Unknown (0) | root | 6.9% | 6.33 min | 382 | 159.33 MiB | ar |
| 844 | com.apple.dt.instruments.dtsecurity (8... | launchd (1) | root | 0.0% | 1.09 s | 2 | 115.20 MiB | ar |
| 61 | SpringBoard (61) | launchd (1) | mobile | 0.2% | 33.26 s | 14 | 68.47 MiB | ar |
| 64 | backboardd (64) | launchd (1) | mobile | 6.6% | 3.90 min | 18 | 43.24 MiB | ar |

# 02 | Lua&资源优化

## Lua配置优化-概览

未优化前文件大小：150MB
优化后：100MB
内存优化前：600MB
内存优化后：280MB
最大单表配置物理文件大小：28MB
目标：Lua内存峰值控制在200MB内！

# Lua&资源优化

## Lua-配置优化-常规优化

### 抽取默认项和Key值提取 + Table字符串化

```
34021501]={[3]=15,[6]=2339,[8]=511502,[9]="102,33847#104,705#243,2400#244,600#247,600",[14]="15阶·双鱼",[32]=34021501,[40]=750,},
34021502]={[3]=15,[6]=4678,[8]=511502,[9]="102,67693#104,1410#243,4800#244,1200#247,1200",[14]="15阶·双鱼",[32]=34021502,[40]=1200,},
34021503]={[3]=15,[6]=6432,[8]=511502,[9]="102,93078#104,1939#243,6600#244,1650#247,1650",[14]="15阶·双鱼",[32]=34021503,[40]=1500,},
34021504]={[3]=15,[6]=8771,[8]=511502,[9]="102,126925#104,2644#243,9000#244,2250#247,2250",[14]="15阶·双鱼",[21]=1200,[32]=34021504,[40]
34021505]={[3]=15,[6]=11695,[8]=511502,[9]="102,169233#104,3526#243,12000#244,3000#247,3000",[14]="15阶·双鱼",[21]=3600,[26]="{100155}",
34021506]={[3]=15,[6]=16373,[8]=511502,[9]="102,295043#104,6146",[14]="15阶·双鱼",[26]="{100155}",[32]=34021506,[40]=4200,},
34021507]={[3]=15,[6]=11695,[8]=511502,[9]="102,236926#104,4936#243,16800#244,4200#247,4200",[14]="15阶·双鱼",[21]=3600,[26]="{100155}",
34021601]={[3]=16,[6]=2751,[8]=511602,[9]="102,38994#104,812#243,2560#244,640#247,640",[14]="16阶·双鱼",[32]=34021601,[40]=800,},
34021602]={[3]=16,[6]=5502,[8]=511602,[9]="102,77987#104,1625#243,5120#244,1280#247,1280",[14]="16阶·双鱼",[32]=34021602,[40]=1280,},
34021603]={[3]=16,[6]=7565,[8]=511602,[9]="102,107232#104,2234#243,7040#244,1760#247,1760",[14]="16阶·双鱼",[32]=34021603,[40]=1600,},
34021604]={[3]=16,[6]=10316,[8]=511602,[9]="102,146226#104,3046#243,9600#244,2400#247,2400",[14]="16阶·双鱼",[21]=1500,[32]=34021604,[40
34021605]={[3]=16,[6]=13755,[8]=511602,[9]="102,194968#104,4062#243,12800#244,3200#247,3200",[14]="16阶·双鱼",[21]=4500,[26]="{100165}",
34021606]={[3]=16,[6]=19257,[8]=511602,[9]="102,344146#104,7169",[14]="16阶·双鱼",[26]="{100165}",[32]=34021606,[40]=4480,},
34021607]={[3]=16,[6]=13755,[8]=511602,[9]="102,272955#104,5686#243,17920#244,4480#247,4480",[14]="16阶·双鱼",[21]=4500,[26]="{100165}",
```

10:21:31.648-1: 190248.22167969

10:26:03.209-1: 61211.114257812

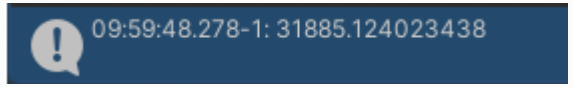
单表190MB
优化后
单表61MB

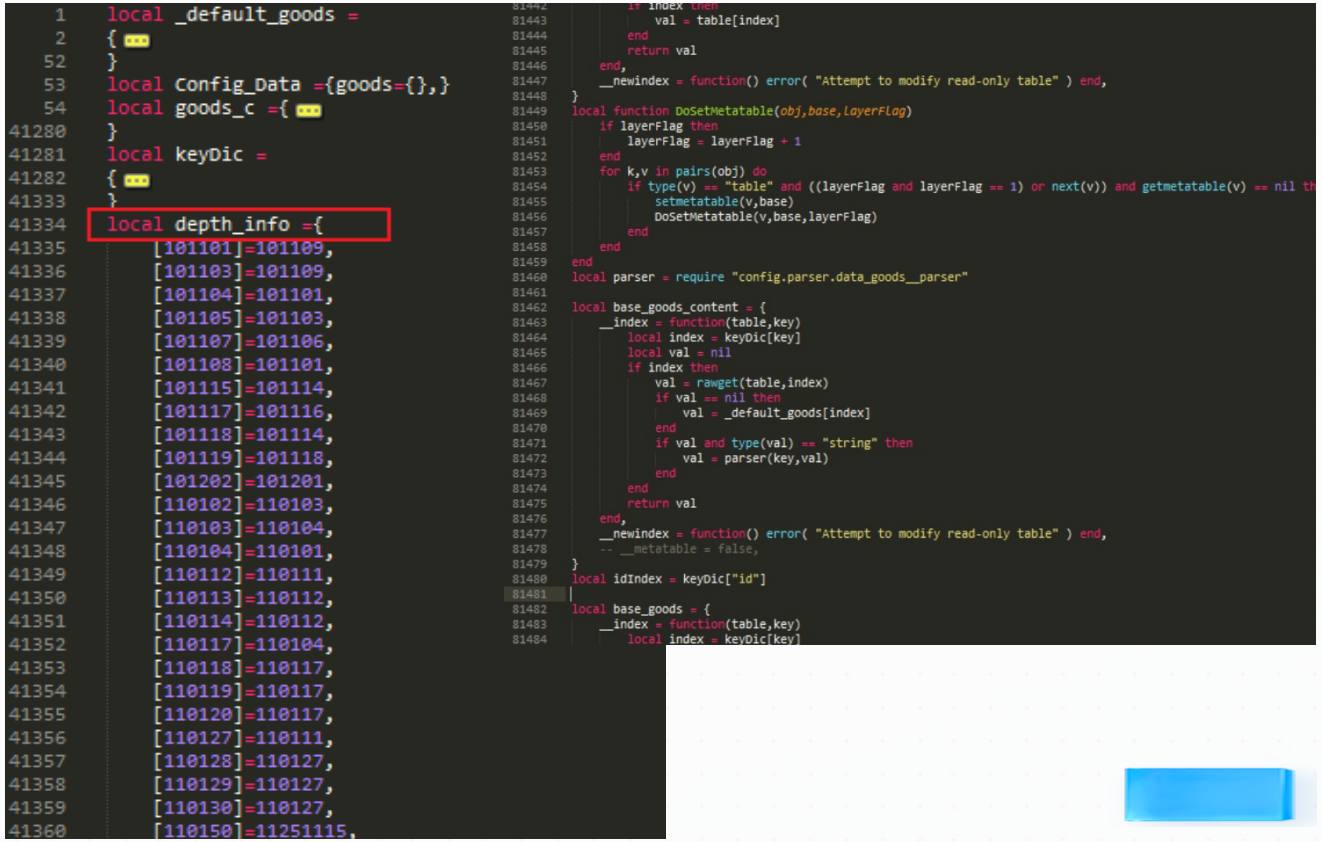
```
local _default_goods =
{
    [1]=0,
    [2]=0,
    [3]=0,
    [4]=0,
    [5]=0,
    [6]=0,
    [7]=0,
    [8]=120800,
    [9]="",
    [10]="{0,[]}",
    [11]=1,
    [12]=0,
    [13]="",
    [14]="每日充值礼包",
    [15]=0,
    [16]="",
    [17]="",
    [18]=0,
    [19]="",
    [20]="",
    [21]="",
    [22]=0,
    [23]="",
    [24]=0,
    [25]=0,
    [26]="1,3",
    [27]="{}",
    [28]=0,
    [29]=0,
    [30]="",
    [31]="",
    [32]=11,
    [33]=500008,
    [34]=0,
    [36]=0,
    [37]=0,
    [38]=1,
    [39]=1,
    [40]=0,
    [41]=0,
    [42]=1,
    [43]=0,
    [44]=0,
    [45]=100,
    [46]=5,
    [47]=0,
    [48]=0,
    [49]="",
    [50]="1,2",
}
local Config_Data ={goods={},}
local goods_c ={ ...
}
local keyDic =
{
    String=17,
    activity=2,
    add_attr=23,
    bag_order=22,
    batch_flag=18,
    bind=25,
    broadcast=10,
    can_sell=1,
```

## Lua-配置优化-行依赖

行依赖方式是通过递归将重复信息进行提取
配套自动生成的解析代码

序列化table+抽取公共项+行依赖
单表内存占用31MB



## Lua-配置优化-二进制索引

在行依赖的基础上将内容抽离保存为二进制，Lua只保留Key信息

Main.lua | data_goods.lua

```
52  }
53  local Config_Data ={goods={},}
54  local goods_c ={
55      [101101]=322,
56      [101102]=165238,
57      [101103]=356655,
58      [101104]=511845,
59      [101105]=694649,
60      [101106]=887709,
61      [101107]=1099127,
62      [101108]=1291131,
63      [101109]=1485241,
64      [101111]=1710951,
65      [101113]=1894761,
66      [101114]=2079632,
67      [101115]=2284404,
68      [101116]=2474900,
69      [101117]=2681721,
70      [101118]=2874750,
71      [101119]=3070329,
72      [101201]=3263073,
73      [101202]=3312692,
74      [110101]=3339740,
75      [110102]=3583313,
76      [110103]=3755857,
77      [110104]=3928423,
78      [110111]=4112326,
79      [110112]=4344657,
```

Main.lua | data_goods.lua | data_goods_content.lua

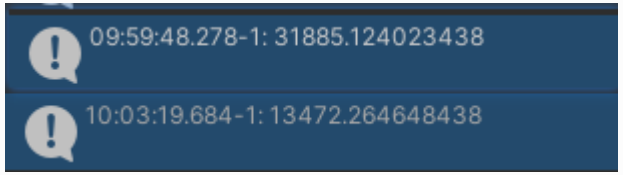
## Lua-配置优化-二进制索引

对应的解析代码，在Lua层传递偏移值读取

```lua
local sltIds = {[118407]=1,[120501]=1,[120502]=1,[120503]=1,[120901]=1,[121901]=1,[121902]=1,[121903]=1,[11221231]=1,[11221271]=1,}
local tcontentPath = "Lua/config/Content/data_goods__content.lua"
local base_root_default = {
    __index = function(table,key)
        local index = keyDic[key] or key
        local val = nil
        if index then
            val = rawget(table,index)
            if val then
                return val
            else
                val = goods_c[index]
            end
        end
        if val then
            local toffset,tlen
            if sltIds[index] then
                toffset,tlen = g_getoffsetandlen(val)
            else
                toffset = bit.rshift(val,9)
                tlen = bit.band(val,g_low_9_bits_mask)
            end
            local tcontent = ToLuaUtils.ReadLuaConfigContent(tcontentPath,toffset,tlen)
            val = loadstring("return " .. tcontent)()
            if val then
                setmetatable(val,base_goods)
                DoSetMetatable(val,base_goods_content,1)
                Config_Data.goods[index] = val
            end
        end
        return val
    end,
}

setmetatable(Config_Data.goods,base_root_default)

DoSetMetatable(_default_goods,base_default)
setmetatable(_default_goods,base_default)
return Config_Data
```




内存占用从31MB -> 13MB

# Lua&资源优化

## Lua-配置优化-代码化公式

**通过自定义配置表读取逻辑，实现定制化的配置优化，将配置表中有规律的部分运行时还原，生成配置的时候也还原校验**

```lua
--等级转成 x阶x星 默认都从1阶1星 = 1级 开始
local function GetStepStarFromLv(lv,systemNum)
    local systemNum =  systemNum or 10
    local step =  math.ceil(lv / systemNum)
    local star = lv % systemNum
    star = (star == 0) and 10 or star
    return step ..","..star
end

local function GetLvSkill(type,lv,skillAssist)
    local lvSkill = ""
    local useLv = nil

    --print("into GetLvSkill  "..type..  "..lv)
    if skillAssist and skillAssist[type] then
        local curTypeSkill = skillAssist[type]
        for skillLv,skillValue in pairs(curTypeSkill) do
            if lv >= skillLv  and (not useLv or skillLv > useLv ) then
                lvSkill = skillValue
                useLv = skillLv
            end
        end
    end
    return lvSkill
end

--自定义读取内容
local function CustomGet(funTable)
    local curValue = funTable.outValue
    local baseTable = funTable.baseTable
    local curTable = funTable.curTable

    local key = funTable.key
    if key == "name" then
       curValue = baseTable[curTable.type][1].name
    elseif key == "model" then
        curValue = baseTable[curTable.type][1].model
    elseif key == "step" then
        curValue = GetStepStarFromLv(curTable.lv)
    elseif key == "skill" then
        local Config_Data = funTable.Config_Data
        local skillAssist = Config_Data.skillAssist
        curValue = GetLvSkill(curTable.type,curTable.lv,skillAssis
    end
    return curValue
end
--开放接口
local function _Get(funTable)
    return CustomGet(funTable)
end
return  Get
```

```lua
--data_surface__upLevelConfig_CustomSet
--自定义修改配置
local CustomSet = {}
local hadStep = nil
local hadSkill = nil
local function CustomConfig(config)
    --local parser = require "lua.LDB.parser.data_surface__upl
    local skillAssist = {}  --技能辅助表
    for typeKey,v in pairs(config.upLevelConfig) do
        local tempSkill = nil
        for lvKey,value in ipairs(v) do
            if lvKey > 1 then
                value.name = nil
                value.model = nil
            end
            if hadStep then --留一个数据 做项提取
                value.step = nil
            else
                hadStep = true
            end
            --处理技能
            if value.skill ~= tempSkill then
                tempSkill = value.skill
                skillAssist[typeKey] = skillAssist[typeKey] or
                skillAssist[typeKey][lvKey] = value.skill
            end
            if hadSkill then --留一个数据 做项提取
                value.skill = nil
            else
                hadSkill = true
            end
        end
    end
    config.skillAssist = skillAssist
    return config
end
```

```lua
--自定义校验
local customGetFunc = require "LuaConfigExtract.CustomGet.data_surface__upLevelConfig_CustomGet"
local function CustomVerify(value)
    local baseConfig = value.baseConfig
    local customConfig = value.customConfig
    local checkNum = 0
    local falseNum = 0
    for bigKey,localValue in pairs(baseConfig) do
        for typeKey,v in pairs(localValue) do
            if typeKey ~= 10 and typeKey ~= 6 and typeKey ~= 13 then --排除不需要的外显
                for lvKey,v2 in pairs(v) do
                    for itemName,v3 in pairs(v2) do --遍历每项

                        local CheckData =
                        {outValue = customConfig[bigKey][typeKey][lvKey][itemName],
                         baseTable = customConfig[bigKey],curTable = customConfig[bigKey][typeKey][lvKey],key = itemName,
                         Config_Data = customConfig}
                        local customValue = customGetFunc(CheckData)
                        if v3 ~= customValue   then
                            print( string.format("Error!! verify was not pass !\n ErrorPoint: typeKey:%s lvKey:%s itemNa
                            falseNum = falseNum + 1
                        end
                        checkNum = checkNum + 1
                        if falseNum >= 50  then --差异太多就直接跳过
                           return false
                        end
                    end
                end
            end
        end
    end
    if falseNum > 0 then
        return false
    end
    print(" The verify was pass  ! !  checkNum :"..checkNum)
    return true
end
```

## Lua-配置优化-协议化

优化后内存下降，但是CPU耗时增加！！
特别是遍历操作，全展开后遍历耗时非常恐怖

回归传统H5做法，将部分大表迁移到后端存储，通过协议返回

优点：前端不需要加载存储大表，按需从后端返回，性能大幅提升
缺点：需要前后端改动大量的逻辑代码适配，耗费人力

## Lua-加载优化

通过NativeArray的特性将LuaBuffer直接传递到C层

```
public unsafe static void* UnsafeReadFile(string fileName, ref int bufferSize)
{
    fileName = fileName.Replace(".lua", "").Replace('/', '_').ToLower();
    TextAsset luaAsset = null;
    foreach (var ab in m_LuaABList)
    {
        if (ab.Contains(fileName))
        {
            luaAsset = ab.LoadAsset<TextAsset>(fileName);
            break;
        }
    }
    var nativeArray = luaAsset.GetData<byte>();
    var arrayPtr = nativeArray.GetUnsafePtr();
    bufferSize = nativeArray.Length;

    m_RemoveLuaAssetQueue.Enqueue(luaAsset);
    m_RemoveNativeArrayQueue.Enqueue(nativeArray);

    return arrayPtr;
}
```

```
[MonoPInvokeCallbackAttribute(typeof(LuaCSFunction))]
public unsafe static int LoadFile(IntPtr L)
{
    try
    {
        string fileName = LuaDLL.lua_tostring(L, 1);
#if UNITY_EDITOR
        byte[] buffer = LuaFileUtils.Instance.ReadFile(fileName);
#else
        int bufferSize = 0;
        var buffer = LuaFileUtils.Instance.UnsafeReadFile(fileName, ref bufferSize);
#endif

        if (buffer == null)
        {
            string error = string.Format("cannot open {0}: No such file or directory", fileName);
            error += LuaFileUtils.Instance.FindFileError(fileName);
            throw new LuaException(error);
        }

#if UNITY_EDITOR
        if (LuaDLL.luaL_loadbuffer(L, buffer, buffer.Length, fileName) == 0)
#else
        if (LuaDLL.unsafe_luaL_loadbuffer(L, buffer, bufferSize, fileName) == 0)
#endif
        {
            return 1;
        }

        LuaDLL.lua_pushnil(L);
        LuaDLL.lua_insert(L, -2);  /* put before error message */
        return 2;
    }
```

# Lua&资源优化

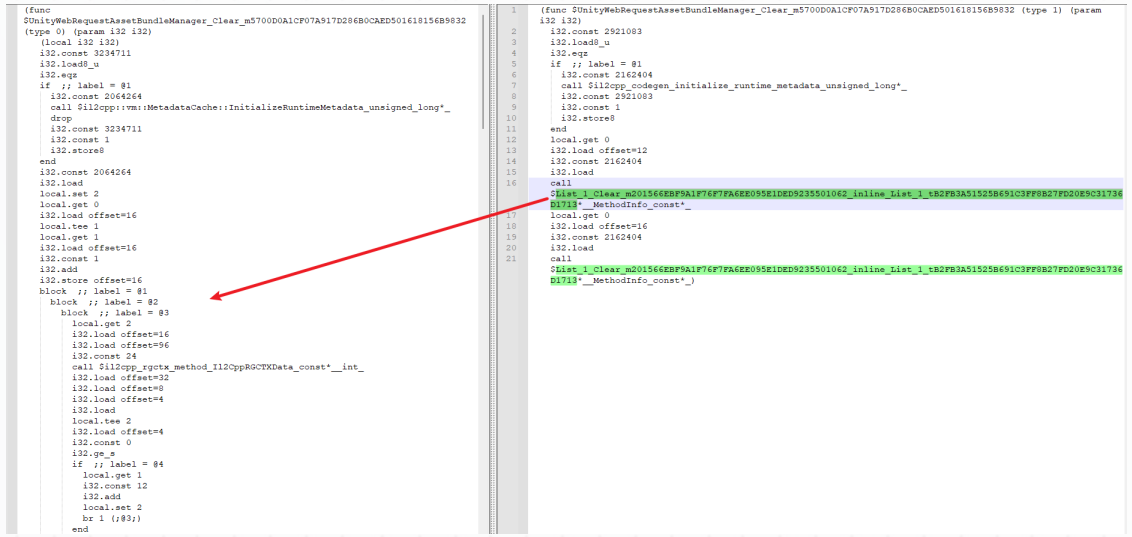
## WASM代码优化

通过工具搜索和匹配Lua代码，将多余的Wrap文件剔除

定制link.xml，只保留必须的代码库

通过twiggy工具排序找出大的代码块

使用wasm工具链来查看转换代码



IL2CPP Size + CodeOptimization Size

位置: ıe\minigame5.0_stable_202403202030\wasmcode

大小: 3.98 MB (4,181,999 字节)

占用空间: 3.98 MB (4,182,016 字节)

IL2CPP Size + CodeOptimization Speed

位置: ıe\minigame5.0_stable_202403121612\wasmcode

大小: 4.34 MB (4,557,852 字节)

占用空间: 4.34 MB (4,558,848 字节)

## 慎用纹理拆分

微信的纹理拆分工具和Unity的AutoStreaming

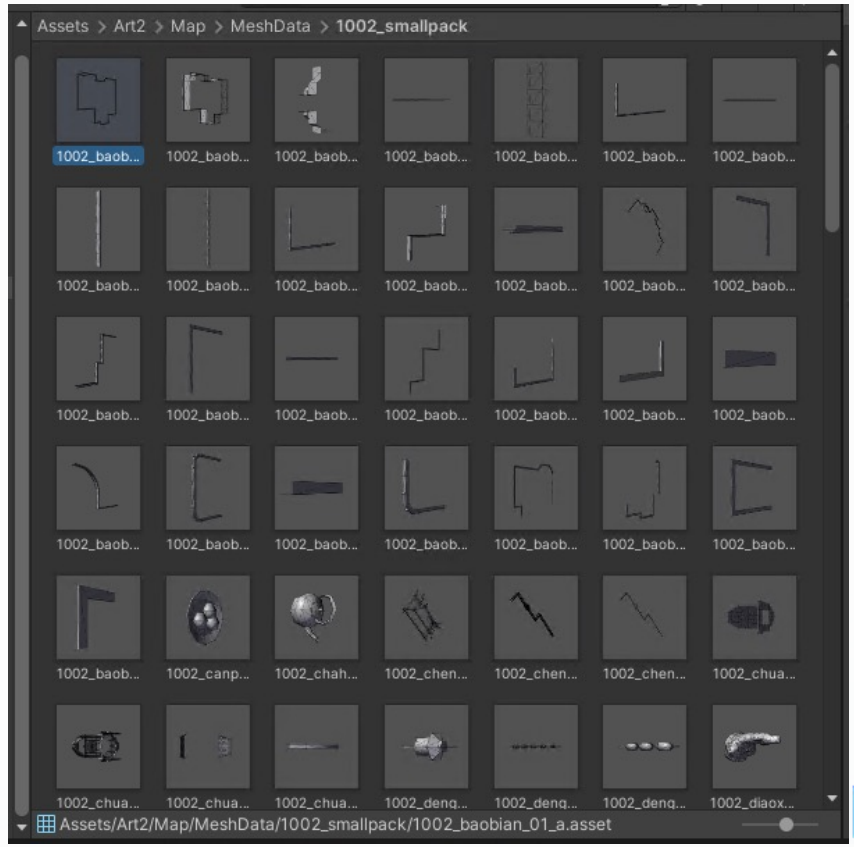
优点：降低AssetBundle的内存占用（WXAssetBundle已解决），分平台生成不同的纹理格式
缺点：大量的文件占据下载和I/O资源，能否彻底去掉AB包的结构直达Image加载?

目前只保留了少部分的UI分离，其余的都没使用!

## 自定义抽离Mesh

将网格信息单独抽离出来，下载后直接赋值，剥离AB的加载

其它类型的资源可同理制作



# Lua&资源优化

## 加载的优化

异步和同步加载接口配合使用，减少构建下载消耗，提高缓存文件的提取效率

```
WXCustomReadFileSync(filePath) {
  const fs = wx.getFileSystemManager();
  try {
    const res = fs.readFileSync(filePath);
    if (typeof res !== 'string') {
      cacheArrayBuffer(filePath, res);
      return `${res.byteLength}`;
    }
    return res;
  }
  catch (e) {
    if (e.message) {
      return e.message;
    }
    return 'fail';
  }
},
```

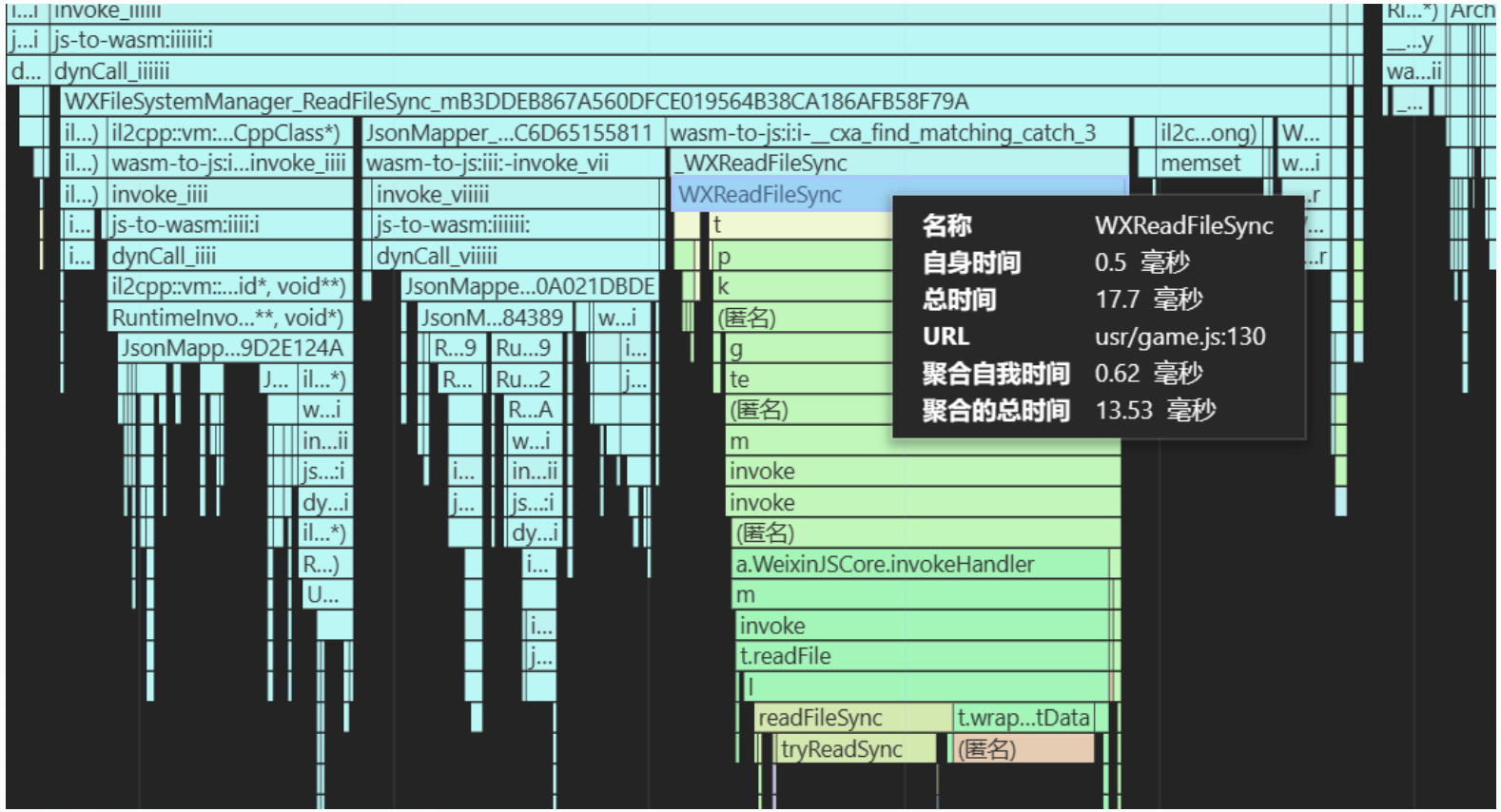
```
private byte[] ReadWxLocalFileSync(string path)
{
    try
    {
        var sb = new StringBuilder();
        sb.Append(GetUseDataPath());
        sb.Append(path);
        var realPath = sb.ToString();

        var bytes = wxFileSystemManagerEx.CustomReadFileSync(realPath);
        if (bytes != null)
        {
            return bytes;
        }
        return null;
    }
    catch (Exception ex)
    {
        return null;
    }
}
```

```
private AssetBundle LoadAssetBundleSync(string path)
{
    var bytes = ReadWxLocalFileSync(path);
    if (bytes != null)
    {
        return AssetBundle.LoadFromMemory(bytes);
    }
    return null;
}
```

## 加载的优化

同步接口在中低端机型上有较大的加载耗时，需要分场景使用



## 优化部分总结

抽取Mesh独立打包，提高载入速度，降低AB包体

脱离UI预制件和图片的关联，独立管理文件依赖和打包

根据系统进行挑选，大部分系统可以不打图集

Update机制控制，根据不同逻辑来控制不同的更新频率，降低CPU损耗

参考AutoStreaming，独立拆分出更多的资源单独打包

参考团结引擎，优化IL2CPP元表加载，降低内存占用

# Lua&资源优化

## 坑点回顾

**缓存的坑**

安卓机器上，首次进入游戏开始缓存，当数量达到一定程度的时候，缓存的时间大幅度升高，需要控制首次游戏的缓存数量

**初始化的坑**

game.js-startGame之前调用SDK初始化会阻碍微信的初始化，导致较长时间的启动黑屏

**压缩工具的坑**

文件数量达到一定程度后会有分离失败的问题，主要集中在忽略文件的步骤，需要自行拆分命令调用（准备彻底弃用）

**iOS的坑**

长时间1.2G或以上内存占用，会导致WebContent内存释放不及时，引发重复闪退；
低版本性能模式兼容性不好；
碎片问题引发的发热等问题；

# 03 | 推进优化

# 推进优化

## 制作原则

- *尽量减量不减质*
- *不再兼容APP版本内容*
- 移除多余的界面背景特效
- 界面关闭按钮挪到左边
- 界面外显类（包括坐骑、主角、翅膀、各种TIPS等等）不加载不播放出场动作
- 场景不要随意同步到稳定版，同步前需要在群里先声明
- APP版本内容同步到H5需要严格对比重整，防止覆盖已有的优化处理
- 移除场景里外显的多余特效
- UI预制件不得存放过多的节点用于显示隐藏（用动态加载的方式替代）
- 界面进行美术/逻辑优化后把预制旧的不用的节点删掉
- 弹窗界面需要支持空白点击遮罩关闭
- 减少类似任务栏背景图这类半透过渡明显的图片，压缩后条纹感严重

# 推进优化

## 九灵神域-技术管理-问题分解


**将复杂问题拆分细化，快速推进。 不定期技术小组研讨会议，总结当前阶段的优化目标**

# 谢谢观看