QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

腾讯手游性能优化之路

SPEAKER / 何纯

主办方 Geekbang. InfoQ

手游市场现状

手游增速放缓

竞技类和MMO占主导

3.0时代精细化运营

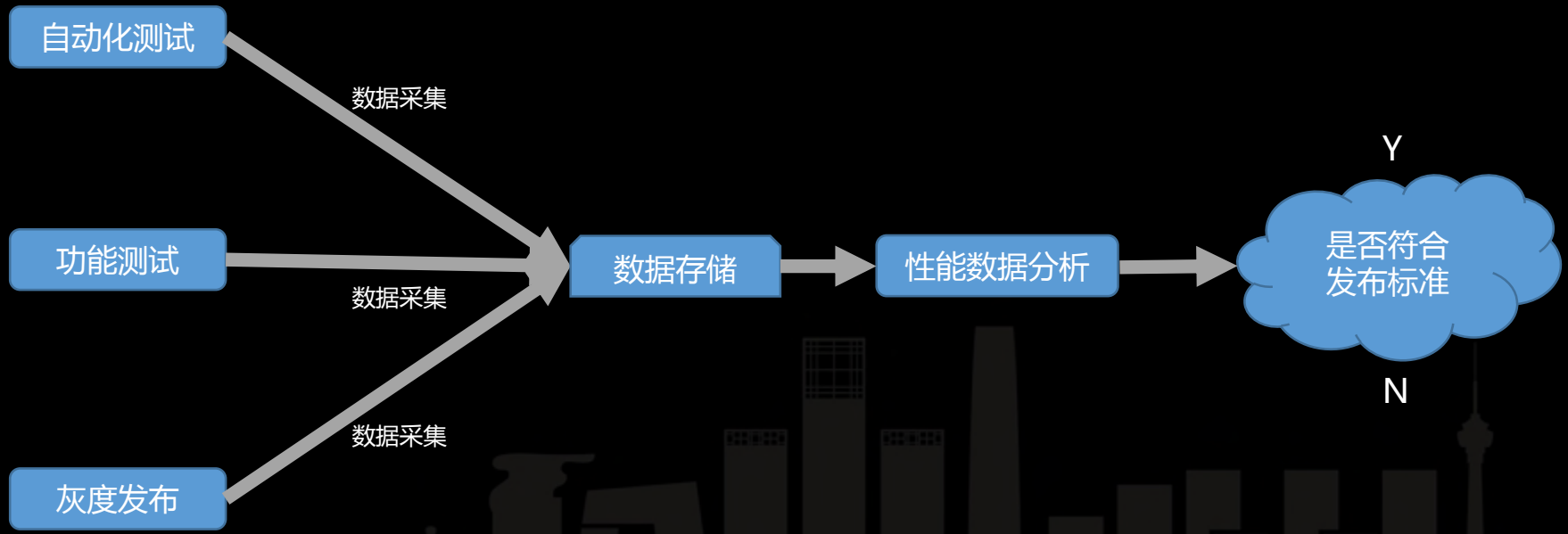
重度

手游市场现状


大家对手机GPU（游戏）性能的要求怎么样？
已结束
5969
参与人数
投票选项 单选
没什么要求，因为基本不怎么玩手机游戏 771(12.9%)
有一定要求，要能流畅运行简单的3D手游 1364(22.9%)
要求比较高，要能流畅运行高画质3D手游 1529(25.6%)
虽然玩的不多，但最好是通吃所有3D手游 2305(38.6%)
已结束


手机
10月18日 14:50 来自 微博 weibo.com
【性能6！】君知道，你们很多人都喜欢玩王者荣耀等moba游戏，玩6X/配置麒麟655芯片，64位8核，旗舰机16纳米制程工艺，拿一血，超神…游戏要玩得更6！尊享版4GB运行内存，64GB存储，最高支持128GB的Micro SD卡拓展，还有智能文件系统和代码级优化等，帮助解决手机卡顿问题。
收起
查看大图
向左旋转
向右旋转
64GB 存储

手游版本性能审核



《王者荣耀》真实性能优化效果

优化前

优化后

《王者荣耀》真实性能优化效果

优化前

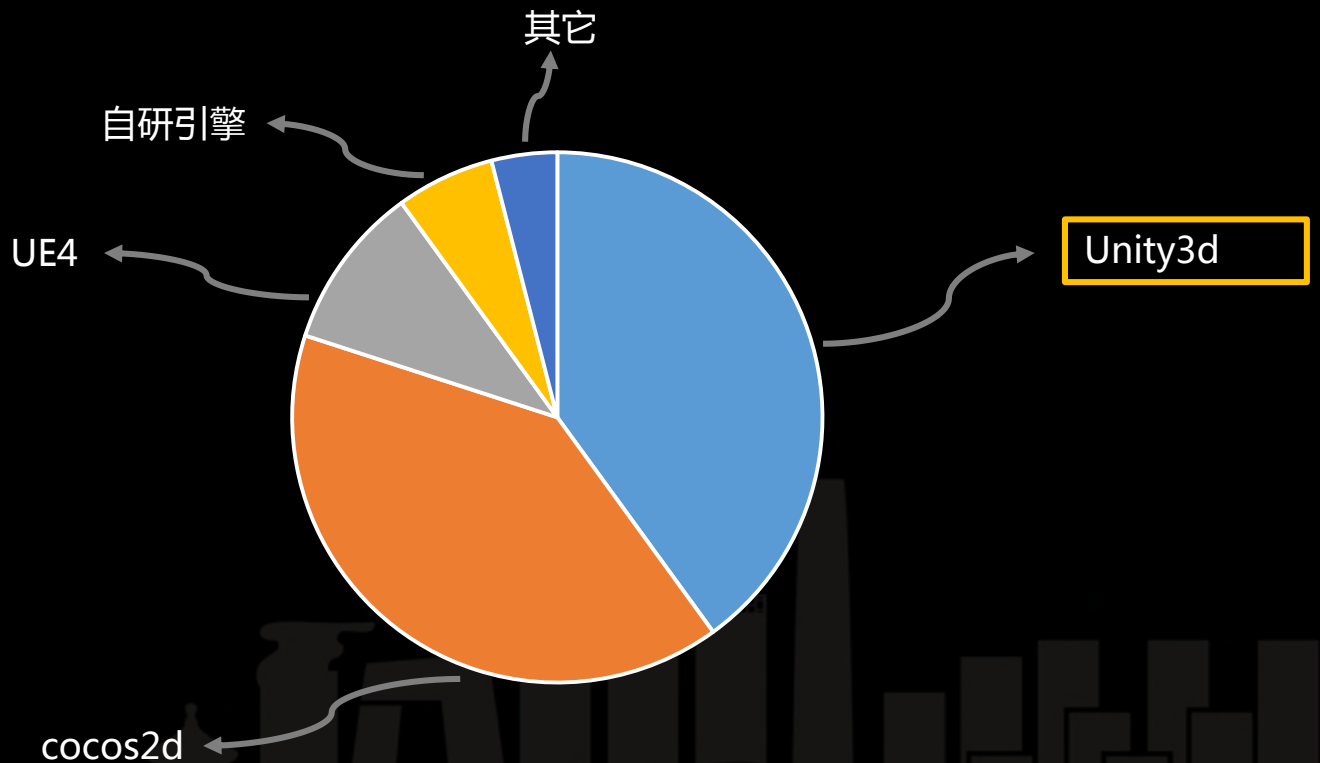
优化后

1 性能测试场景

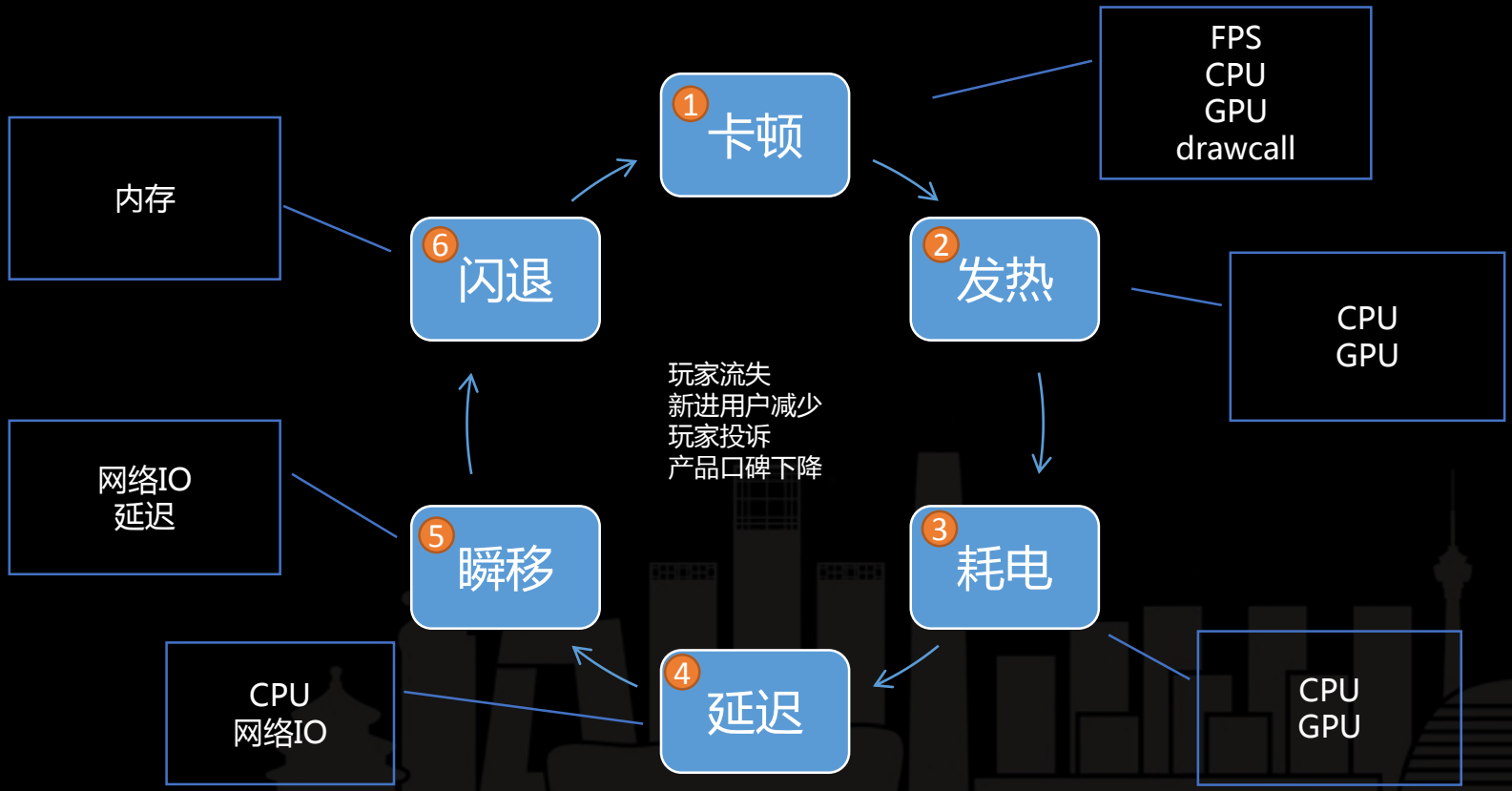
2 数据采集、分析、优化

3 腾讯手游发布标准

手游客户端引擎



手游客户端的性能问题


1 卡顿
FPS
CPU
GPU
drawcall
2 发热
CPU
GPU
3 耗电
CPU
GPU
4 延迟
CPU
网络IO
5 瞬移
网络IO
延迟
6 闪退
内存
玩家流失
新进用户减少
玩家投诉
产品口碑下降

客户端性能测试

大量机器人同屏释放技能

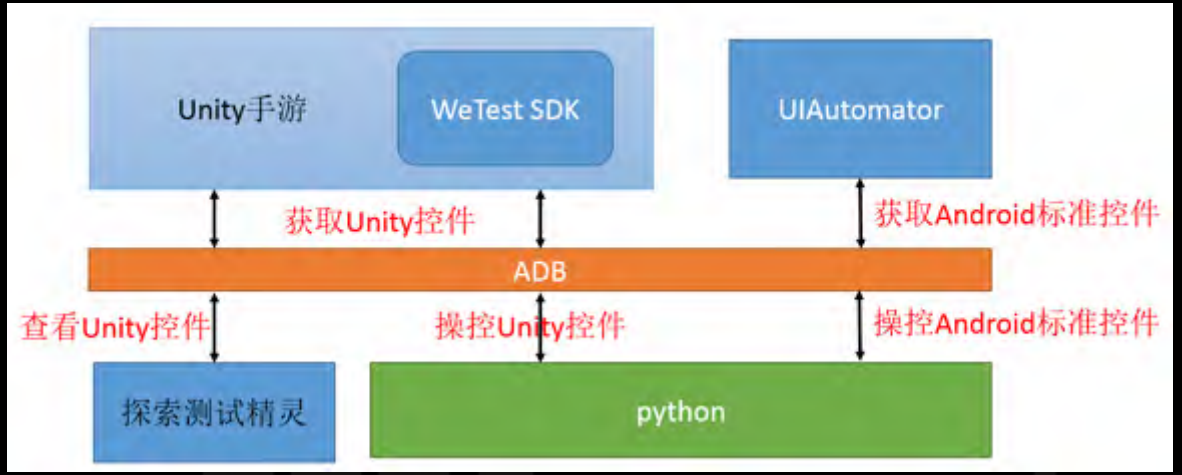
单人游戏

多人组队游戏

自动化测试

Unity游戏自动化框架 - GAutomator

1. 盲点
2. 基于unity控件的盲点
3. 能写脚本的自动化



- 王者荣耀、穿越火线、火影忍者 等游戏

1 性能测试场景

2 数据采集、分析、优化

3 腾讯手游发布标准

数据如何获取？

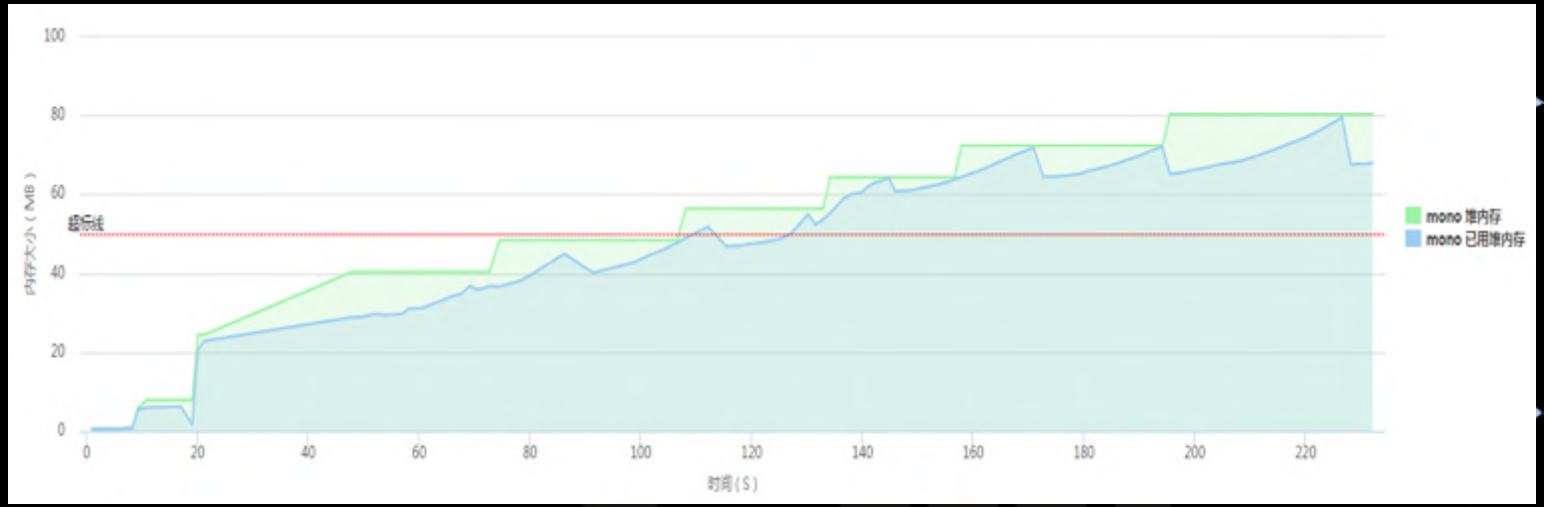
PSS内存	/proc/self/smaps
网络流量	/proc/net/xt_qtaguid/stats
mono内存	mono_stack_walk_no_il、mono_object_get_size
drawcall	glDrawArrays、glDrawElements、……
FPS	swapbuffer
CPU	/proc/stat、/proc/self/stat
三角形数量	glDrawArrays、glDrawElements、……
GPU	/sys/class/kgsl/kgsl-3d0/gpubusy

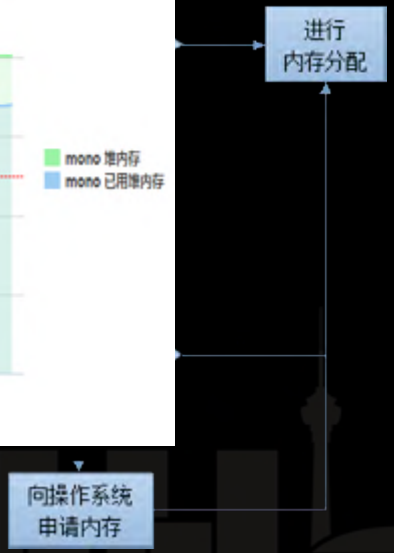
Unity简介 – 卡顿原因

- 某一帧耗时过长、导致图像输出延迟。

  - 资源加载
  - 低效逻辑函数
  - 主线程 IO
  - GC
  - 网络波动

Unity – GC


100
80
60
40
20
0
内存大小（MB）
虚拟线
0
20
40
60
80
100
120
140
160
180
200
220
时间（S）
mono 堆内存
mono 已用堆内存


进行
内存分配
向操作系统
申请内存

Unity – GC


```
lass myObj;
yObj A;


    A = new myObj();

    myObj B;
    B = new myObj();

    GC.collect();
}
```

- 很多卡顿是GC造成的

Unity – 如何减少GC

腾讯某MOBA手游、5v5 副本中平均15秒就产生一个GC。

产生GC的原因：

mono 内存申请；

手动调用GC.collect；

捕获GC：mono_profiler_install_gc

优化方法

小兵使用对象池pool；

Lua 调用C#时不要用object传递参数；

网络数据收发时用cache；

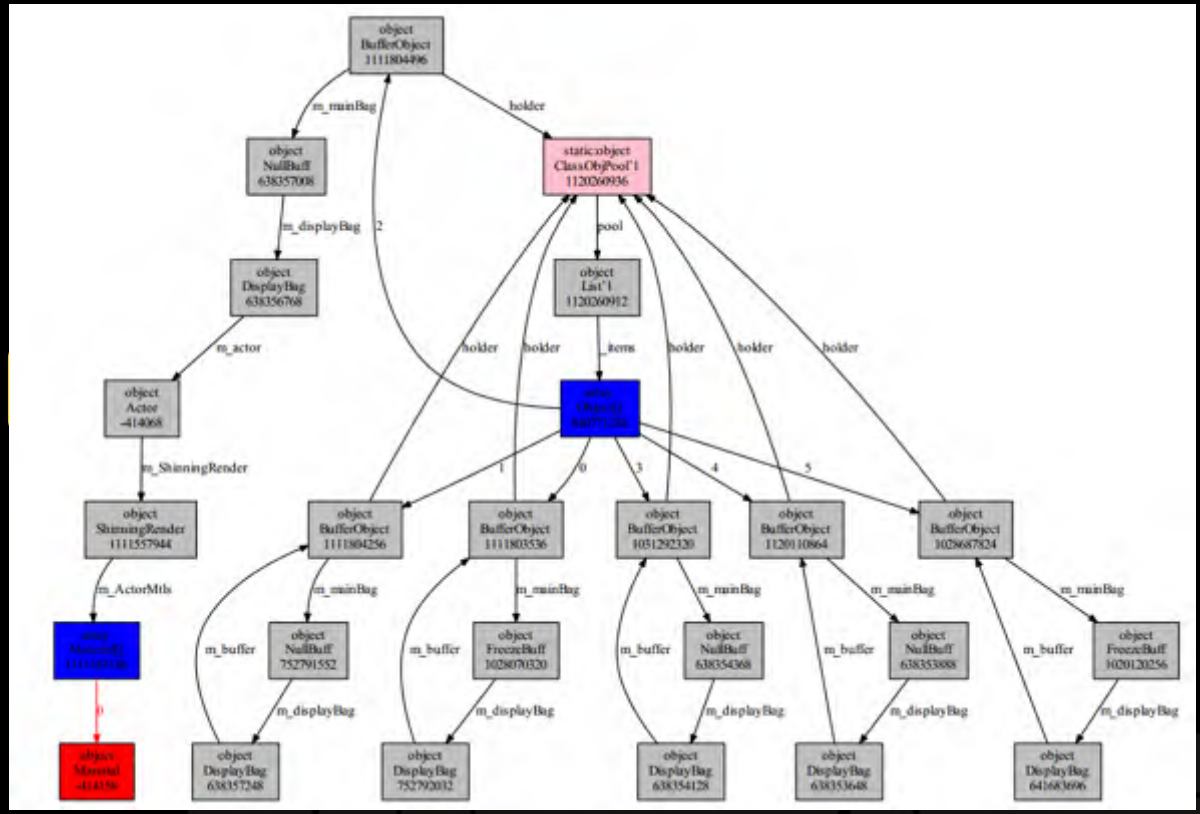
减少一部分UI的刷新频率；

玩家视野外的内容减少开销；

多添加析构用的abstract接口；

减少mono内存泄漏；

Unity – mono内存泄漏



Unity – mono内存快照对比

对比数据

快照间新增top50　快照间保留top50　下载对比报表

资源类型	资源大小(B)	对象堆栈
Byte[]	1622578	WWWManager/c__Iterator20:<>m__57 (object) WWWManager/c__Iterator24:MoveNext ()
Byte[]	484457	WWWManager/c__Iterator20:<>m__57 (object) WWWManager/c__Iterator24:MoveNext ()
Byte[]	249949	WWWManager/c__Iterator20:<>m__57 (object) WWWManager/c__Iterator24:MoveNext ()
String[]	131088	System.Array:Resize (object[]&,int,int) System.Array:Resize (object[]&,int) System.Collections.Generic.List`1:set_Capacity (int) System.Collections.Generic.List`1:GrowIfNeeded (int) System.Collections.Generic.List`1:Add (object) ByteReader:ReadCSV () NameSelector:LoadFilter (UnityEngine.TextAsset) NameSelector:OnInit () MainUILogicControl/c__IteratorCB:MoveNext ()

Unity – 资源

场景加载

大厅

PVE

PVP

Loading界面

从头到尾一个场景

动态 vs 静态

资源释放

自动释放

手动释放

Unity细分资源

- Texture2D
- GameObject
- Mesh
- AnimationClip
- AudioClip
- 资源重复率
- 关卡间保留资源
- 资源拷贝

Unity – 如何优化资源

腾讯某著名FPS手游

某个版本的守护中心副本在低配机的内存峰值严重超标，大于350M，有crash风险。

获取资源信息：Resources.FindObjectsofTypeAll

新的贴图和特性过大，没有低精度资源；

动画资源没有释放；

音频资源没有释放；

6个小BOSS没有使用对象池；

优化方法

小块资源的预加载；

强化资源生命周期的概念，用完即删；

控制贴图大小，不要超过1024；

贴图要符合2的N次幂；

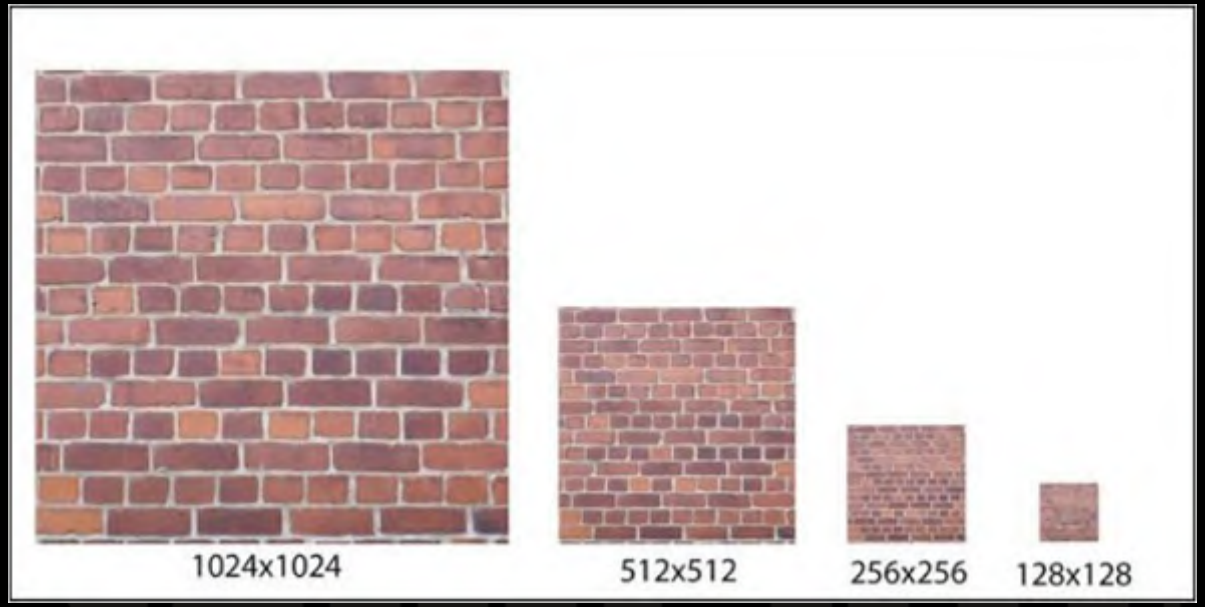
压缩、压缩、压缩；

高中低机型使用不同精度的资源；

避免无用的资源拷贝；

Unity – 如何优化资源

- 资源重复率
- 关卡间保留资源
- 资源拷贝
- 资源尺寸



Unity – 其它卡顿优化

某著名国战类RPG手游

大地图上的GC较少，但卡顿严重；

↓

获取函数开销

mono_stack_walk

mono_runtime_invoke

↓

网络IO太大；

资源加载过大；

函数使用不当；

优化方法

用“for”代替“foreach”；

进副本速度不能慢，副本内加载也不能大；

重复资源独立成包；

尽量少使用FindObjectsOfType函数；

大资源包要尽可能拆分；

进行数据包的合并与优化；

减少空的callback函数（update、fixedupdate）

Unity – 性能优化总结

01

游戏资源优化

压缩贴图

音频压缩

资源预加载

资源控制在建议值

02

渲染层优化

控制overdraw

Drawcall合并

定制化UI

模型面数小于3000

Alpha通道压缩

03

代码层优化

多用对象池

慎用DontDestroyOnLoad

控制mono堆大小

IL2CPP 或 纯C++

避免字符串连接操作

04

游戏策略优化

视野裁剪

分层特效

适当降低贴图精度

1 性能测试场景

2 数据采集、分析、优化

3 腾讯手游发布标准

腾讯手游TDR机型


A9/2G
500M
1200万
iPhone6S
一档机型
A8/1G
400M
900万
iPhone6
二档机型
A7/1G
360M
550万
iPhone5S
三档机型
IOS
安卓
4G内存
550M
1700万
R9/P9
一档机型
3G内存
450M
3800万
MI4/Mate8
二档机型
2G内存
350M
2900万
红米Note1S
三档机型

Unity游戏 – 性能标准

具体性能指标	标准值
CPU	GC单帧>2KB
	GC每帧>20B
	Time>33ms帧数占比<10%
单帧内存	ManagedHeap.UsedSize<30M
	Asset无重复资源
	总体内存峰值<150M
	总体Mono堆内存峰值<40Mb
Drawcall	峰值<250
三角形面片数	峰值<100000
VBO上传量	峰值5M
Skinned Mesh Renderer数量	峰值<50
Active Rigidbody	峰值<50
碰撞体数量	静态+动态峰值<100
纹理资源	峰值<50M
网格资源	峰值<20M
动画片段资源	峰值<15M
音频资源	峰值<15M
GC调用次数	尽量少

- **动态物体**
  - 控制面片数量：300-2000面片
  - 控制材质数量:1-3种
- **静态物体**
  - 网格顶点数少于500
  - 混合纹理数小于5
- **Texture2D贴图**
  - 贴图长宽不宜超过1024
- **音频文件**
  - 压缩音频文件时长不应小于1s，对于1s以下的音频文件建议采用非压缩的音频格式，压缩格式包括mp3、ogg等；

性能测试优缺点

收益

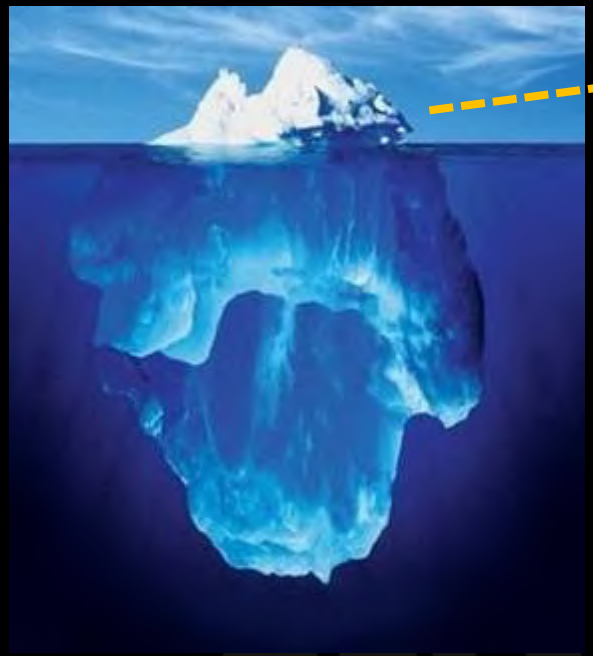
- ✓ 发现大部分场景的性能瓶颈
- ✓ 确保主流机型用户体验流畅
- ✓ 用户投诉降低
- ✓ 用户留存提升

不足

- 测试环境覆盖机型不足；
- 用户行为不能100%模拟；
- 游戏场景无法100%覆盖；
- 多人游戏玩法覆盖不足；

➢ 解决方案：线上性能监控和分析（APM）

手游APM - why？

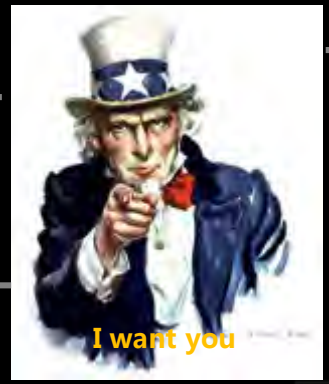


测试环境

线上运营环境

2000多款机型
更多的场景
复杂的操作
险恶的网络
第三方APP兼容
玩家投诉
……

手游APM – 产品功能



手游APM

云端控制

实时监控

发现问题

风险预警

数据分析

手游APM – 数据分析

数据纬度

- FPS
- 内存
- CPU
- drawcall
- 三角形数量
- mono heap
- 场景加载时长
- GPU信息
- OpenGL ES 版本
- 电量
- 温度
- 网络流量
- 联网方式

FPS

- 均值
- 方差
- 卡顿
- 抖动
- 低帧率
- 分段

APP区别？

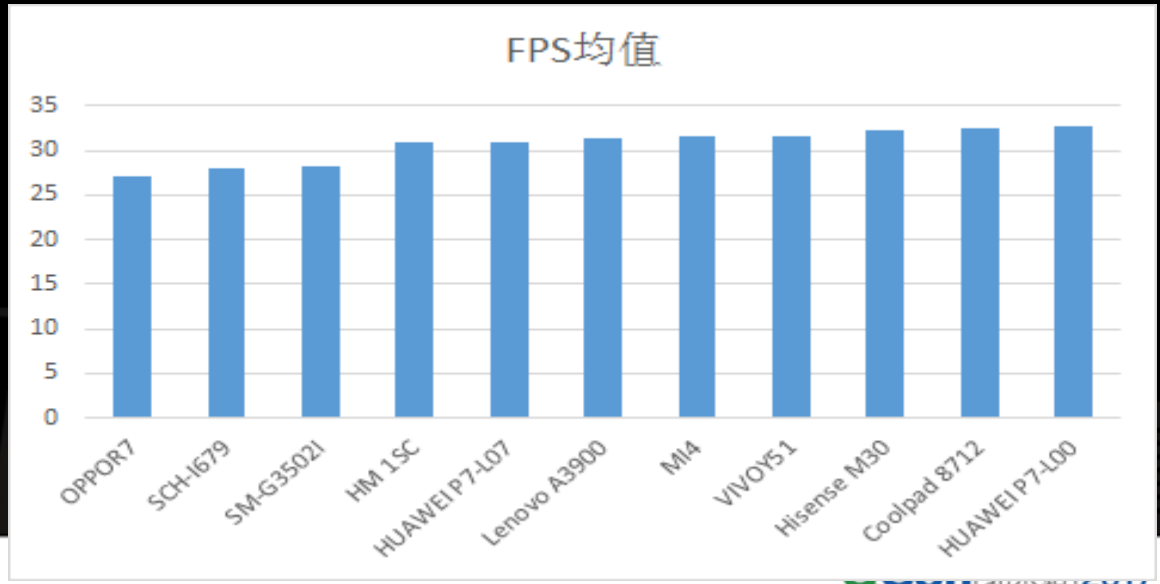
手游APM – 卡顿原因

手游APM – 性能分析

选项纬度 ➢ 版本 ➢ 时间 ➢ 机型 ➢ 场景 ➢ 画质 ➢ 平台

数据纬度 • FPS • CPU • 内存 • drawcall • GPU • 温度 • 电量

- ✓ 观察连续多个版本之间的性能走势。
- ✓ 发现某个版本中性能最差的场景。
- ✓ 发现Top100中性能较差的机型。



手游APM – 低帧率分析

场景1：低帧率20%

场景2：低帧率10%

场景3：低帧率50%

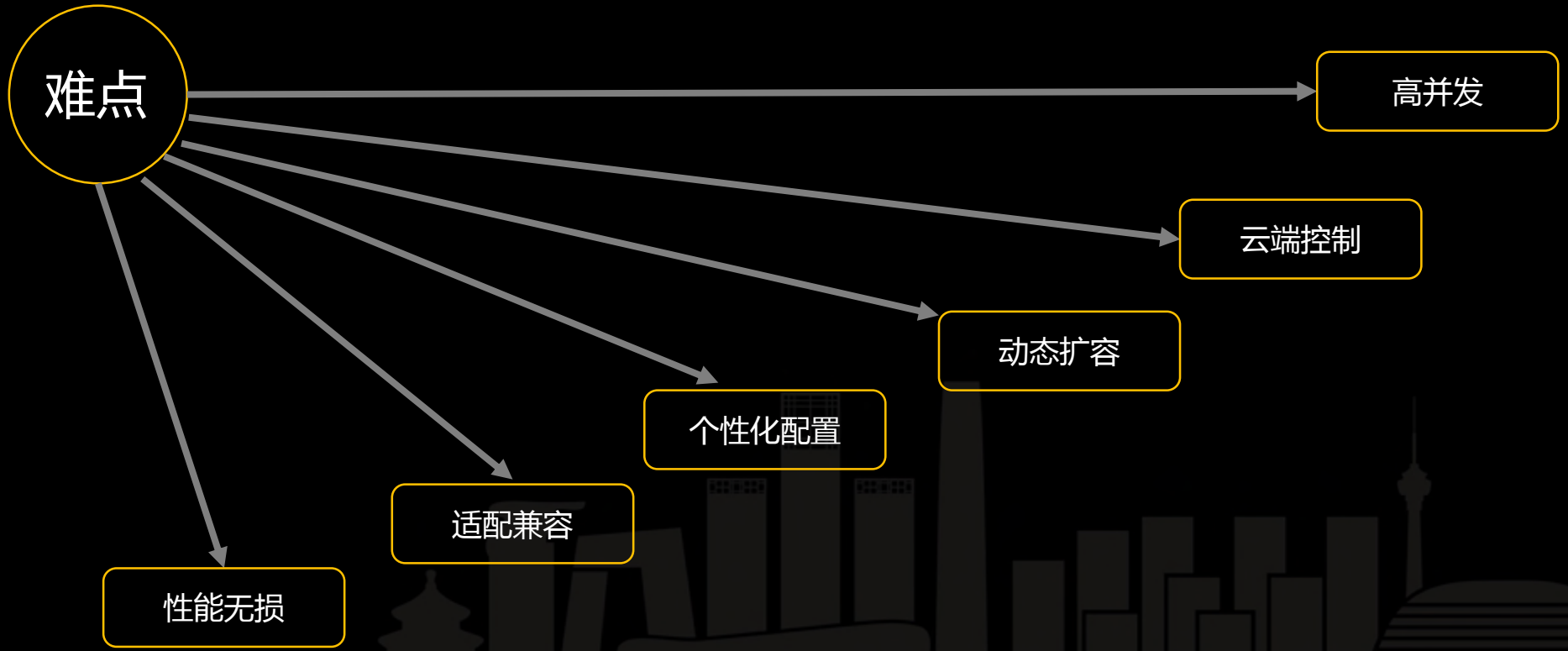
场景4：低帧率80%

场景5：低帧率20%

场景6：低帧率15%

手游APM – GC分析

手游APM – 技术难点



手游APM – 技术难点

性能无损

1个数据采集线程
常驻内存1.0~1.5MB
IO缓冲技术
核心场景内无流量

高并发

服务器持续优化
DB主从分离
均衡负载
动态扩容
排队上报

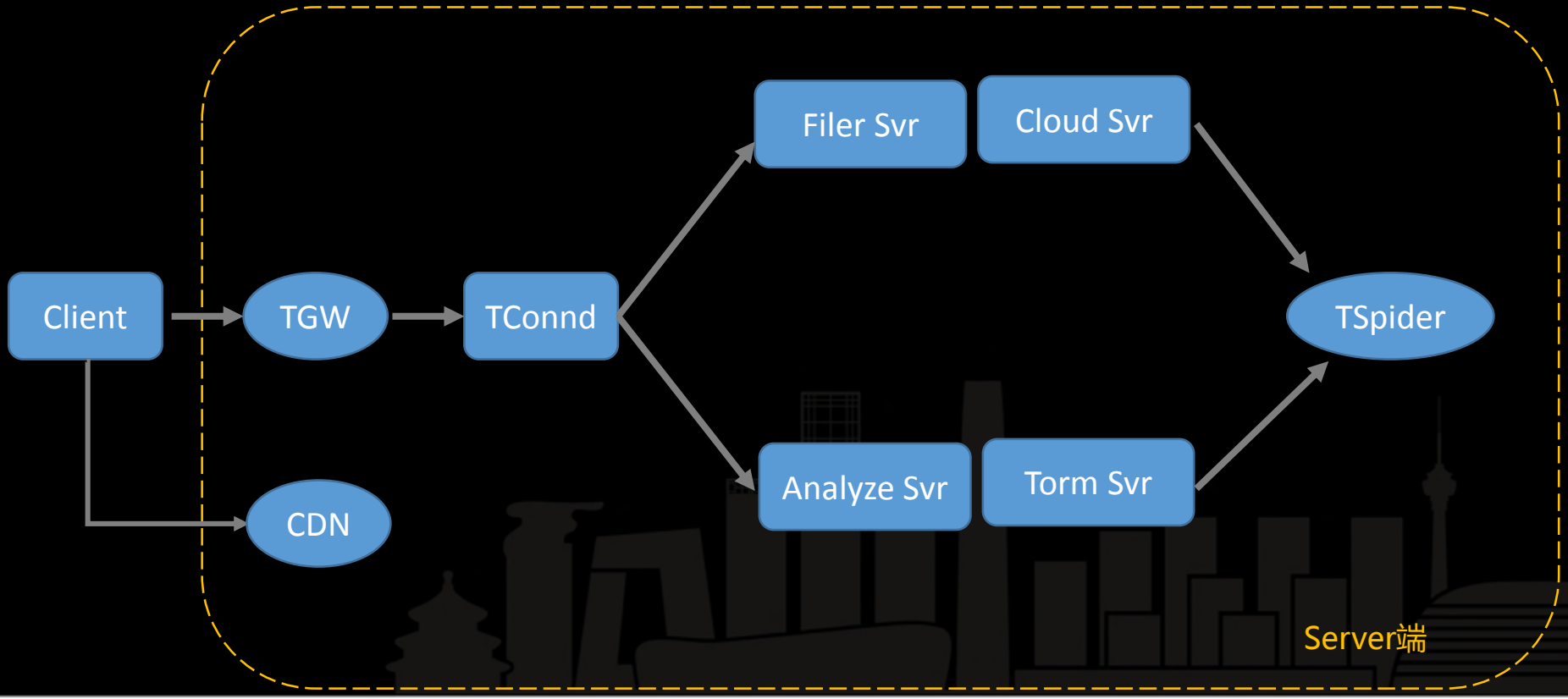
适配兼容

其它SDK兼容
安卓版本兼容
OpenGL版本兼容

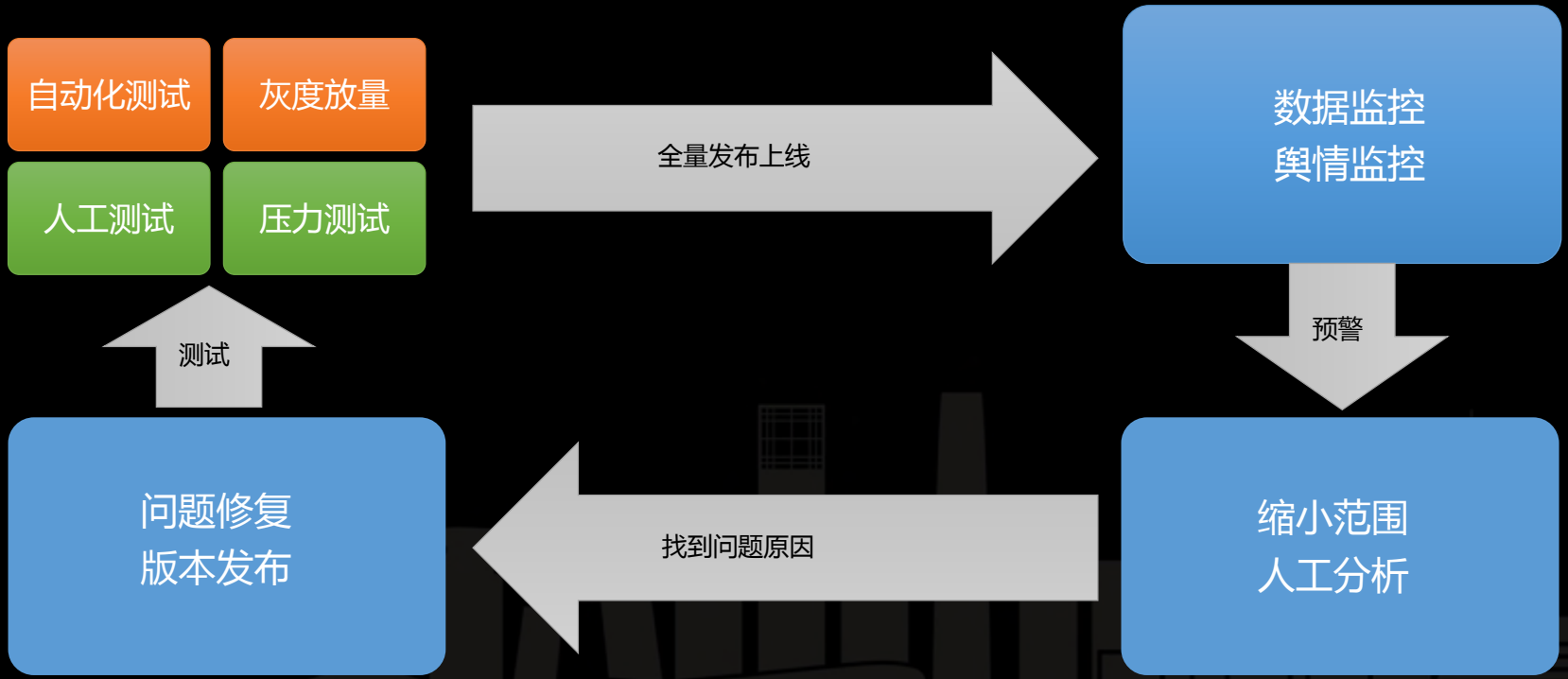
手游APM – 云端控制

灰度	支持概率、手机品牌、手机厂商、IP 地址
过滤	全关全开、APM 版本、游戏版本、手机芯片架构、手机厂商、手机品牌、平台、操作系统版本

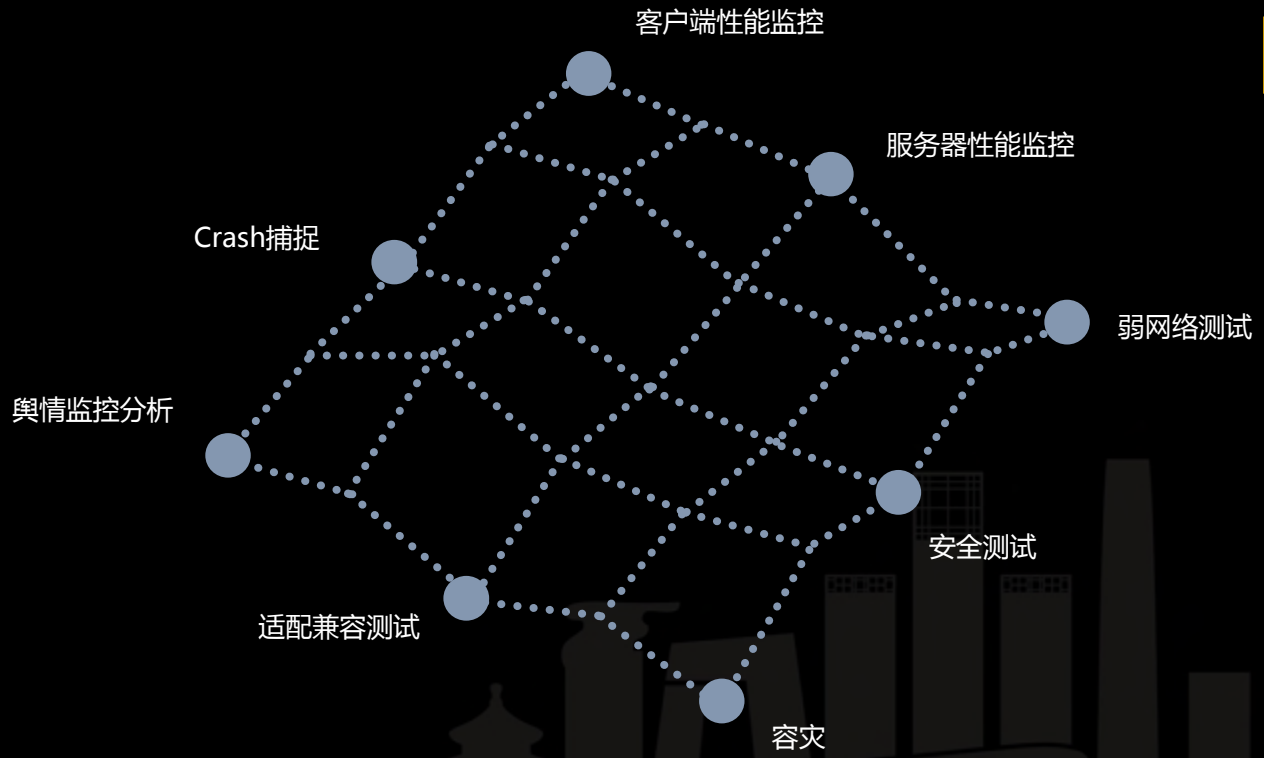
手游APM – 性能上报流程



性能全链路


自动化测试
灰度放量
人工测试
压力测试
全量发布上线
数据监控
舆情监控
预警
缩小范围
人工分析
找到问题原因
问题修复
版本发布
测试

产品质量全覆盖



织一张网 筑一面墙

Do the right thing, get things done.

Thanks !