



UWA 发布
Unity 手游体检蓝皮书
2018-2019 年

侑虎科技（上海）有限公司
www.uwa4d.com
2019 年 8 月

写在前边

作为游戏行业的服务商，UWA（侑虎科技，详细介绍见文末，下统称 UWA）不仅为游戏开发者提供高效的性能优化工具，也致力于为行业提供更全面、更具体的信息和服务。为此，UWA 今天发布 2018-2019 年度手游蓝皮书，从**总体性能数据、引擎各模块开销、内存占用**等方面进行汇总分析，呈现 Unity 手游行业现状。

MMORPG 作为目前游戏市场上的头部力量，在 UWA 测评过的项目中，也占据了主要份额，因此我们以该类型的测评数据为例，为大家呈现 2018-2019 年的手游性能现状和发展趋势。

这是侑虎科技原创文章，欢迎转发分享，未经官方授权请勿转载。如果您有任何独到的见解或者发现也欢迎联系我们，一起探讨（QQ 群 793972859），也可扫码进入公司主页。



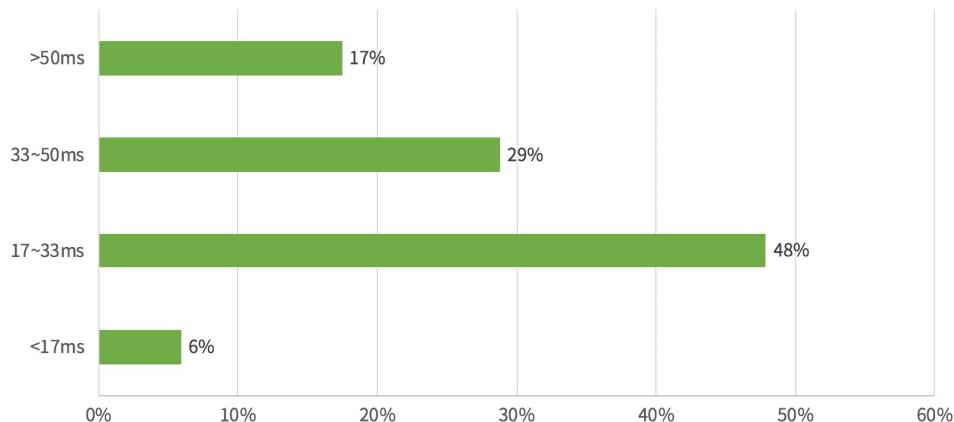
报告目录

写在前边.....	3
MMORPG 手游总体性能开销分析.....	5
MMORPG 手游 CPU 模块性能开销分析	8
一、渲染模块	8
二、逻辑代码	12
三、UI 模块.....	17
四、动画模块.....	22
五、物理模块.....	23
六、粒子模块	25
MMORPG 手游内存模块开销分析.....	28
一、内存泄露	28
二、总体内存	29
三、总体堆内存	30
四、纹理资源内存.....	31
五、网格资源内存.....	33
六、动画资源内存.....	34
七、Shader 资源内存	35
八、RenderTexture 资源内存.....	36
九、粒子系统资源内存	37
MMORPG 手游资源管理分析	39
UWA 对于 MMORPG 手游研发团队的建议.....	42



MMORPG 手游总体性能开销分析

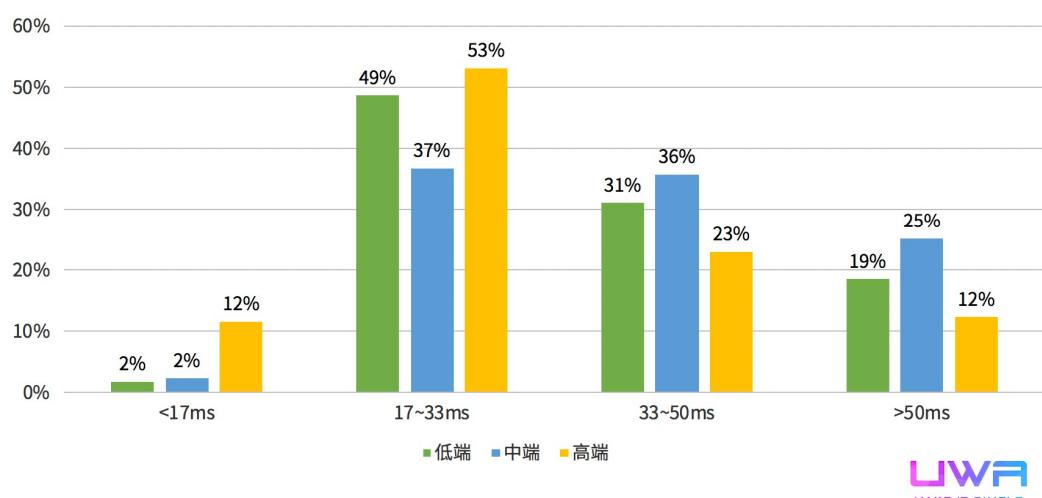
Android 设备CPU均值分布



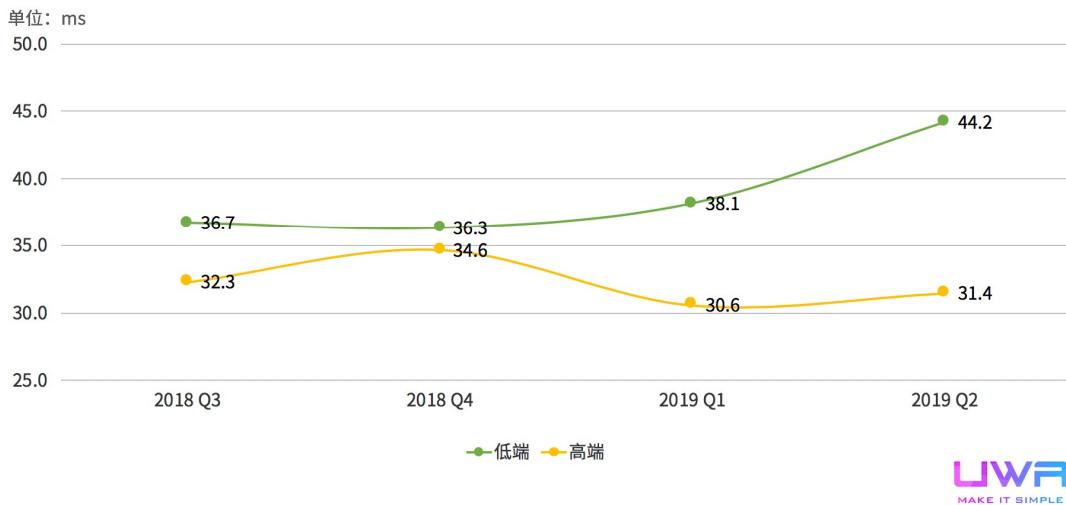
Android 设备的 CPU 均值主体范围为 16.4~72.8 ms，且主要集中在 17~33ms 和 33~50ms 两档。

按照 [UWA 官网](#) 的测评机型分类，我们将统计的性能数据分为低端、中端和高端；在趋势图中则直接以最具有代表性的低端和高端来展现。

Android设备CPU均值分布

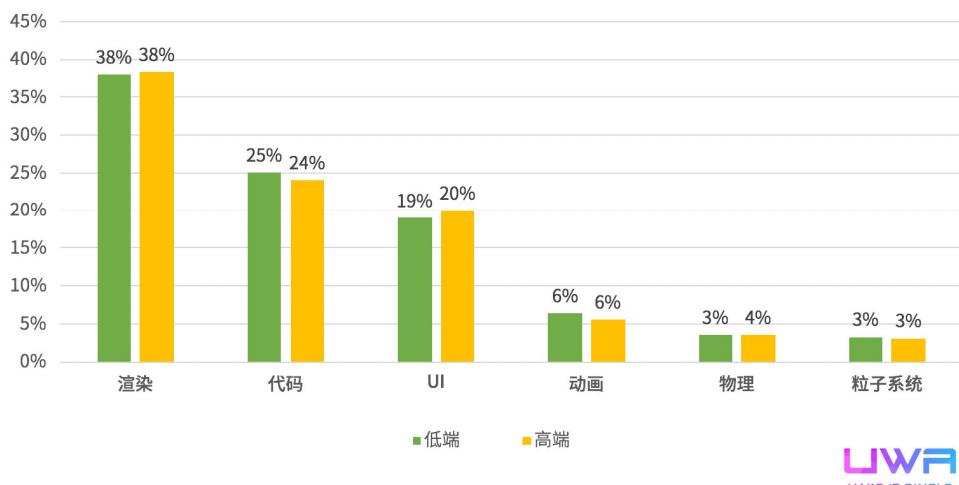


Android设备CPU均值分布趋势



- 1) 大部分 MMORPG 手游在中低端设备的性能开销较高，大于 33ms 的耗时占比范围在 50%~70%。
- 2) 在过去的 4 个季度中，我们看到高端设备上的性能走势较为平稳，但低端设备上的耗时在刚刚过去的一个季度中大幅飙升。这是因为在刚刚过去的一个季度，提测的超大场景的 MMO 新游戏明显增多。大量的 PBR 材质和图像后处理，使得低端设备上的开销大幅上升。这些新游戏中大多还未根据机型进行明显的分级，随着研发过程的推进，我们预计今年底在中低端设备上的整体性能耗时将会回落。

引擎模块CPU均值占用分布



无论是高、中和低端的 Android 设备，渲染、UI 和逻辑代码都是需要研发团队重点关注的，它们的消耗总和占了总开销的 80%。在绝大多数 MMO 游戏项目中，这三者都是研发团度需要关注的优化重点。

下面我们就来具体分析这些主流模块的开销情况。



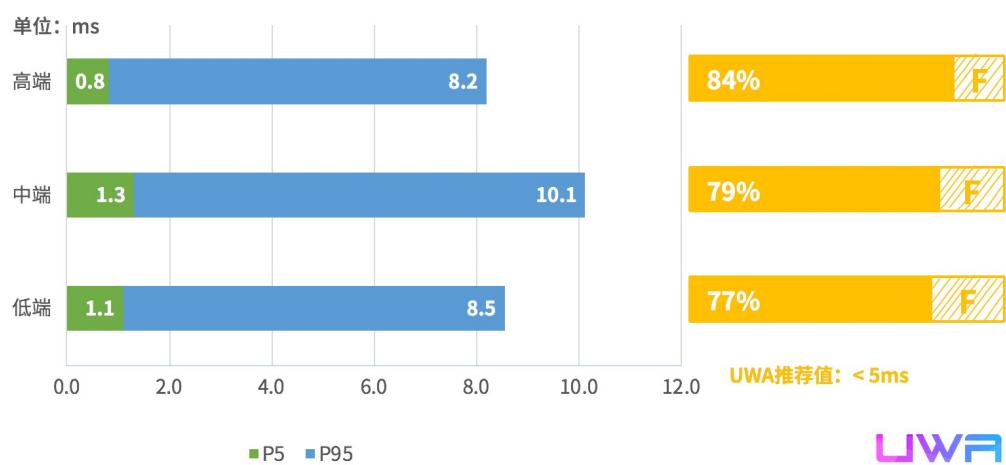
MMORPG 手游 CPU 模块性能开销分析

一、渲染模块

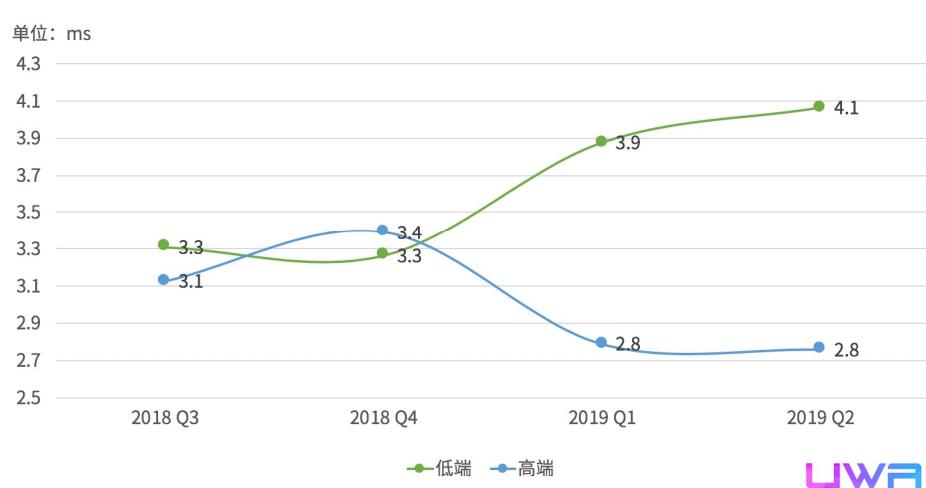
严重程度：地狱

为了能够更好地反映出各个性能参数的整体使用情况，我们统计了每种性能参数的主体使用范围，其范围区间是[5%~95%]，以下数据中 P5 代表 5%，P50 代表均值，P95 代表 95%。

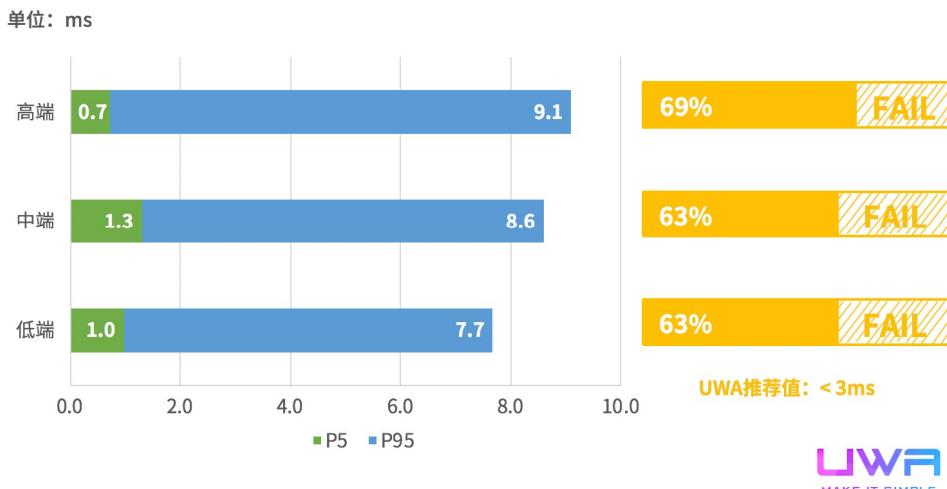
半透明渲染CPU主体范围（P5~P95）分布



半透明渲染的CPU耗时趋势



不透明渲染CPU主体范围（P5~P95）分布



不透明渲染的CPU耗时趋势



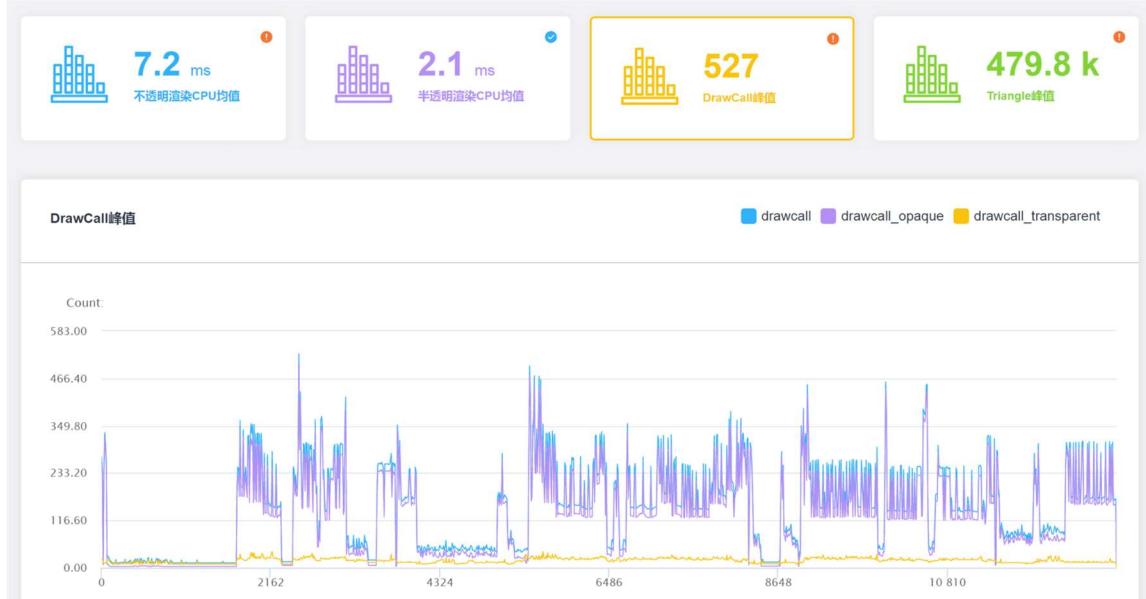
在过去的 4 个季度中，半透明渲染和不透明渲染在高端设备上的 CPU 耗时起伏较大，特别是不透明渲染，在 2019 Q1 升高到 3.8ms，但在 2019 Q2 阶段则大幅下降，降低到 3.1ms。但在低端设备上，无论是不透明渲染还是半透明渲染，其耗时都呈现上升趋势。这确实与 MMO 游戏的体量越来越大相关。在 2019 年的大型 MMO 游戏中，我们关注到了两种现象值得研发团队后续关注：

1) GPU Instancing 技术的大量使用

该技术可有效降低 Draw Call 的占用，从而对渲染模块的 CPU 端压力起到一定的缓解作用。但是，在使用该技术时，需要注意机型的测试，一些低端设备虽然支持 OpenGL ES 3.0，且 `SystemInfo.supportsInstancing` API 返回也为 True，但经过测试时，其底层并没有按照真实的 GPU Instancing 功能来进行

渲染，而是通过逐个 Draw Call 来进行渲染，所以，虽然开启了 GPU Instancing 功能，但其渲染耗时并没有下降。

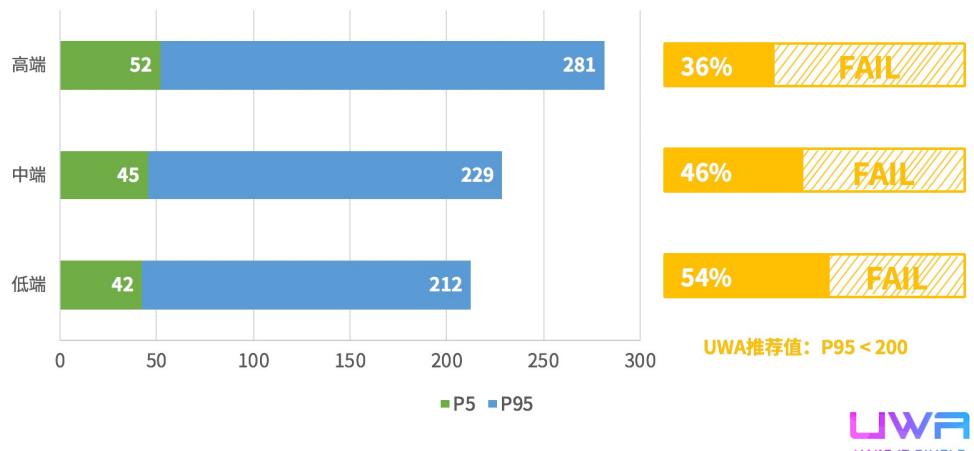
那么如何解决？我们今年更新了 [UWA GOT Online](#) 中 Draw Call 的统计方式，直接获取真实渲染底层的 Draw Call 调用情况并进行展示，同时对不同渲染模式的 Draw Call 进行区分，这样 GPU Instancing 功能是否起作用，通过报告直接看，一目了然。



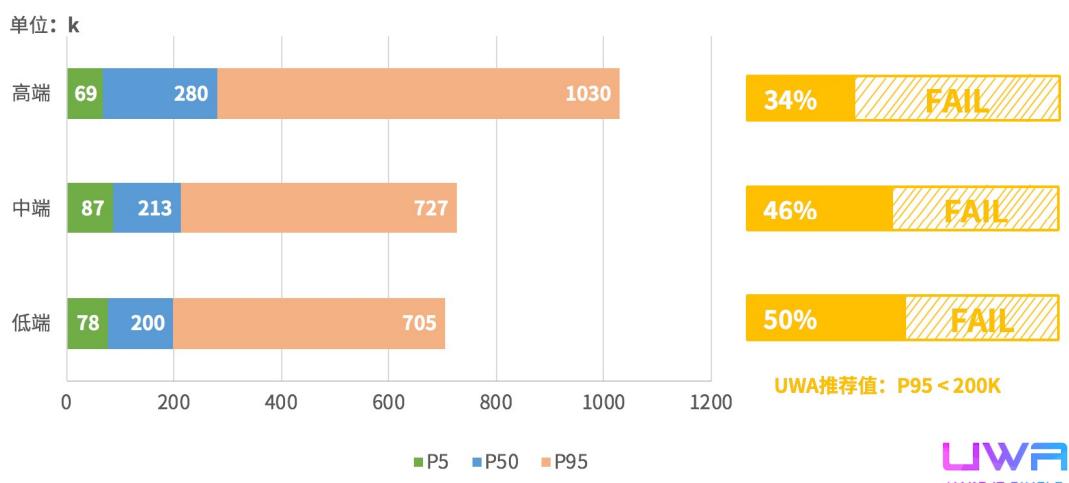
2) GPU 端压力越来越大

PBR、图像后处理的大量使用会导致 GPU 端的压力越来越大，甚至影响到了 CPU 端的 Draw Call 无法正常传输。这其中也跟不同的机型和不同的芯片型号相关，有些则反映到每个 Draw Call 的提交耗时随着 GPU 端的压力而无法增大，而且表现方式则是渲染线程等待 GPU、主线程等待渲染线程的情况，具体耗时因芯片和具体渲染情况的不同，则是会表现在 Mesh.Render、RenderLoop 的自身开销和 Camera.Render 的自身开销中。在部分 Unity 版本中，由于引擎一些自身设计原因，也会体现到粒子系统的 UpdateAll 相关函数中。可以说，主线程等待时间的体现方式因芯片的不同、具体使用情况的不同和引擎版本的不同，可谓是五花八门，受篇幅限制，在这里不一一进行说明，但归根原因主要还是 GPU 压力过大。对此，对于中低端设备上 GPU 端 OverDraw、Bandwidth 和 ALU 的控制是接下来重型 MMO 研发团队需要特别注意的点，特别是还有出海计划的 MMO 游戏。

Draw Call数量主体范围 (P5~P95) 分布



Triangle数量主体范围 (P5~P95) 分布



平均Triangle峰值使用趋势

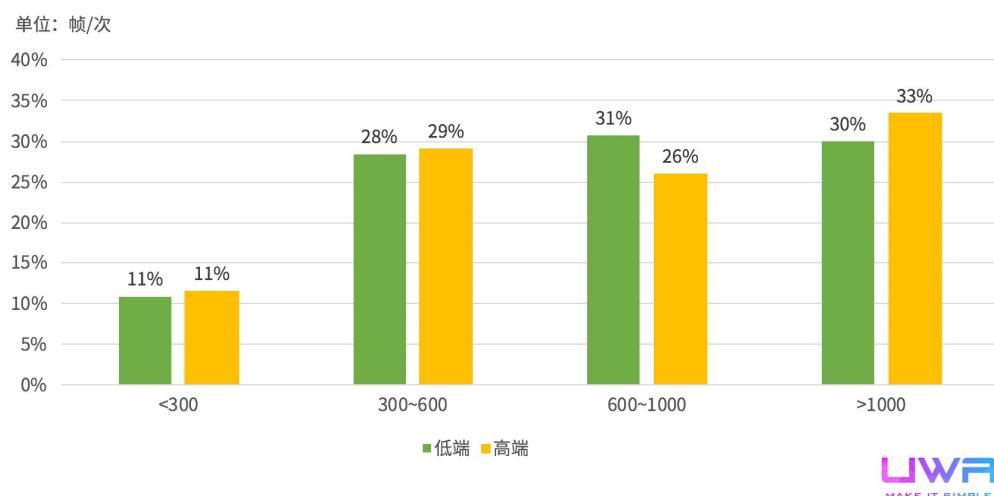


随着 Android 设备整体性能的不断提升，我们对于不同档次机型的 Draw Call 数和面片数量推荐值，正在进行大力分析和研究。就目前的数据分析结果表明，Draw Call 是渲染模块 CPU 端的重要杀手，材质切换次之，然后才是面片数量。而面片数量的对于性能的影响主要是局限在 GPU 端。具体的分析说明请查看：《万物皆数-深挖 UWA Benchmark 之渲染篇》

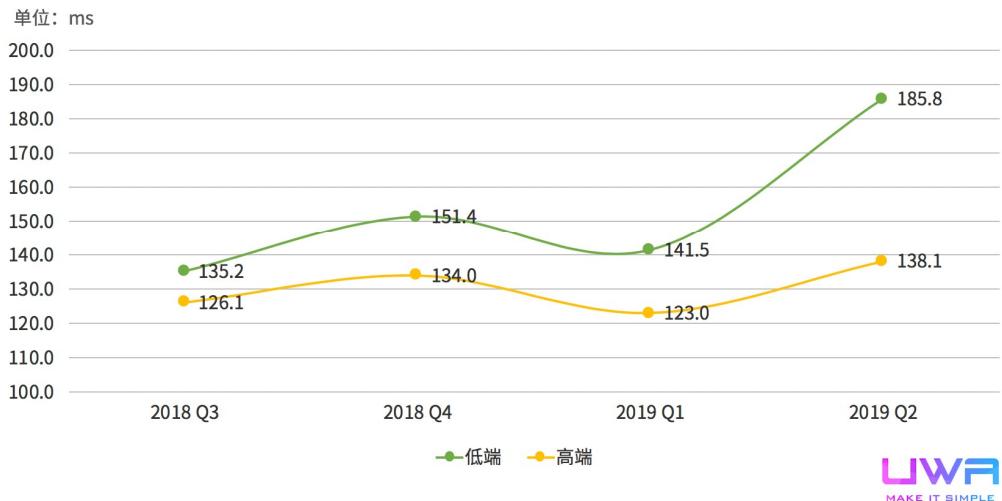
二、逻辑代码

严重程度：地狱

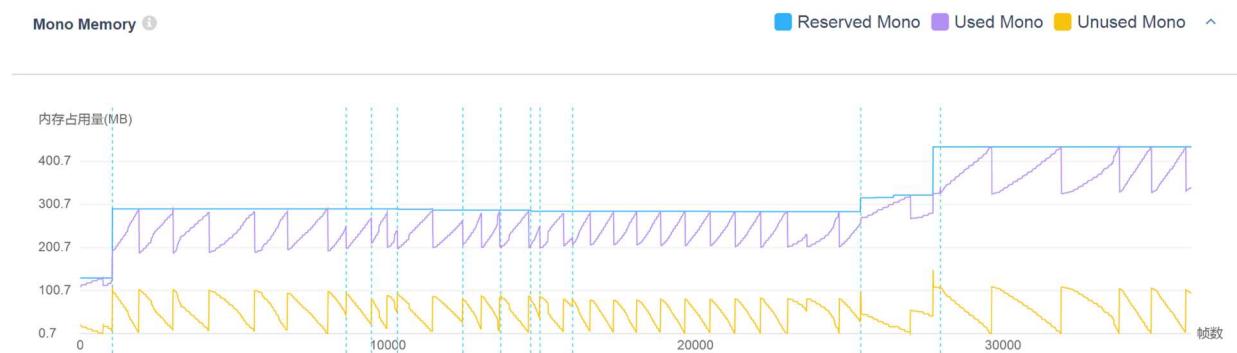
GC调用频率分布



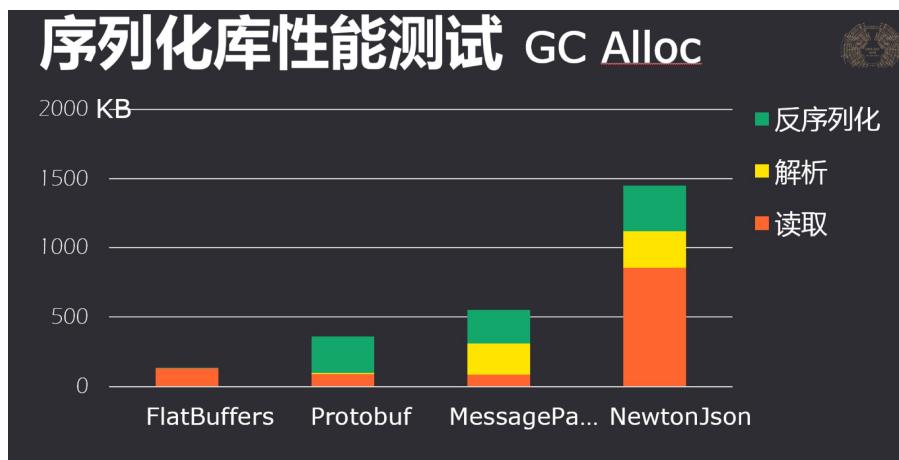
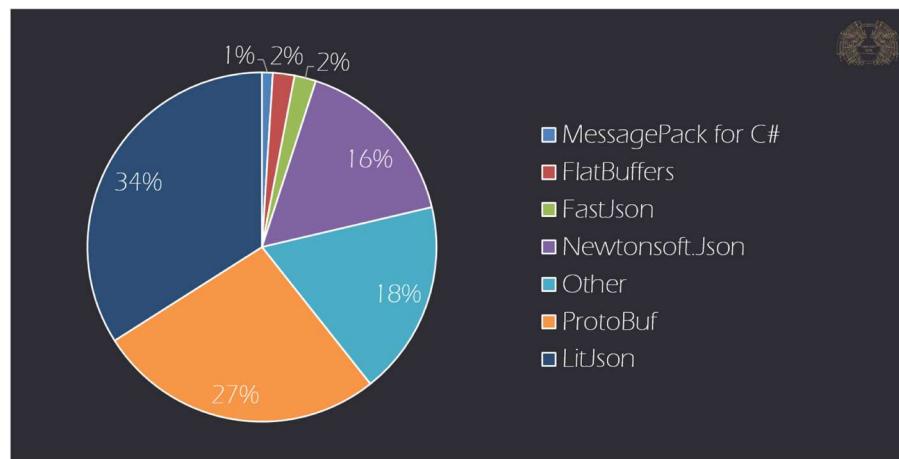
GC调用的平均耗时趋势



- 1) GC 触发频率很高，仍然是目前造成卡顿的主要原因之一，但是，今年可以将 GC 的触发频率控制在 1000 帧/次以上的项目占比达到了 33%，较之去年提高了将近 10 个百分点。
- 2) 随着 MMORPG 游戏越来越重度化，GC 平均耗时在逐步上升。对此，一方面，仍然建议研发团队对游戏运行时堆内存的分配通过 [UWA GOT Online](#) 来进行及时分析和优化。同时，目前项目中驻留 Mono 堆内存最大的一项则普遍是配置文件的内存占用所致。其表现情况是游戏启动后，一分钟之内的 Mono 堆内存就可以达到 50MB 以上的占用，如下图所示。



对此，我们在 [UWA DAY 2019](#) 上对目前市面上大量项目的配置文件使用情况进行了分析，如下图所示。

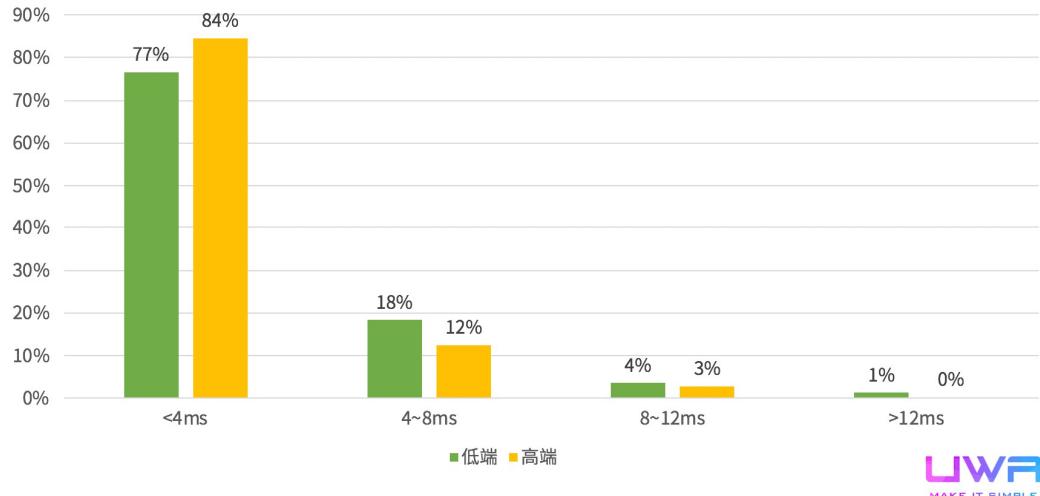


就目前来说，大部分游戏团中使用的配置文件读取方式都不够高效，且堆内存占用较高，我们在会上对主流使用框架进行了测评，建议配置文件堆内存分配较高的团队尽可能尝试使用 ProtoBuff 或者 FlatBuffers 等开源库，使得堆内存占用进一步降低。

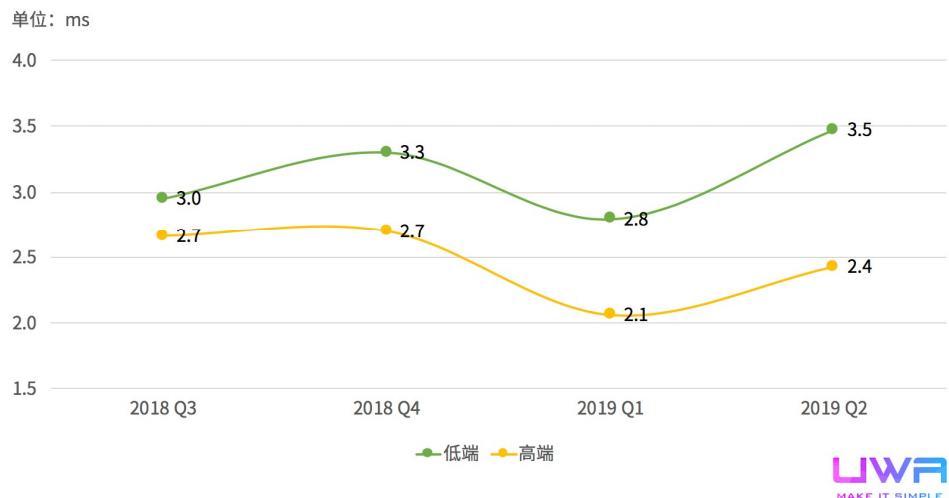
同时，Unity 引擎在 2018.3 版本中加入了对于 GC 调用的控制，研发团队可以尝试通过 API 开控制何时开启和关闭 GC.Collect，这对保持高帧率有很高需求的时刻（如：群战、打 Boss）有很大的帮助。另外，Unity 2019.1 版本以后

加入了 Incremental GC 功能，建议使用该版本以上研发的开发团队进行关注。

Instantiate 实例化耗时分布



Instantiate 实例化耗时走势

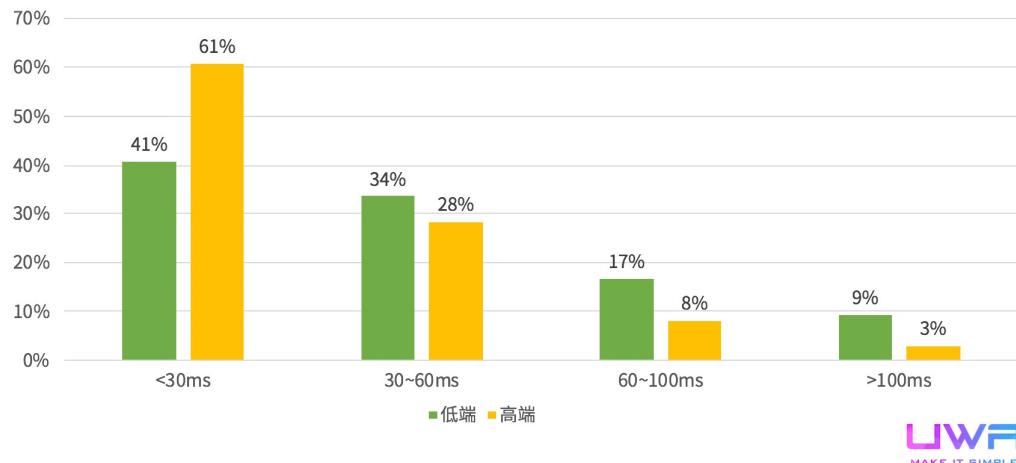


我们可以看到，在过去的四个季度中，Instantiate 耗时趋势较为平缓，低端设备中的耗时在 3.0~3.5ms 区间，高端设备中的耗时在 2.1~2.7ms 区间。

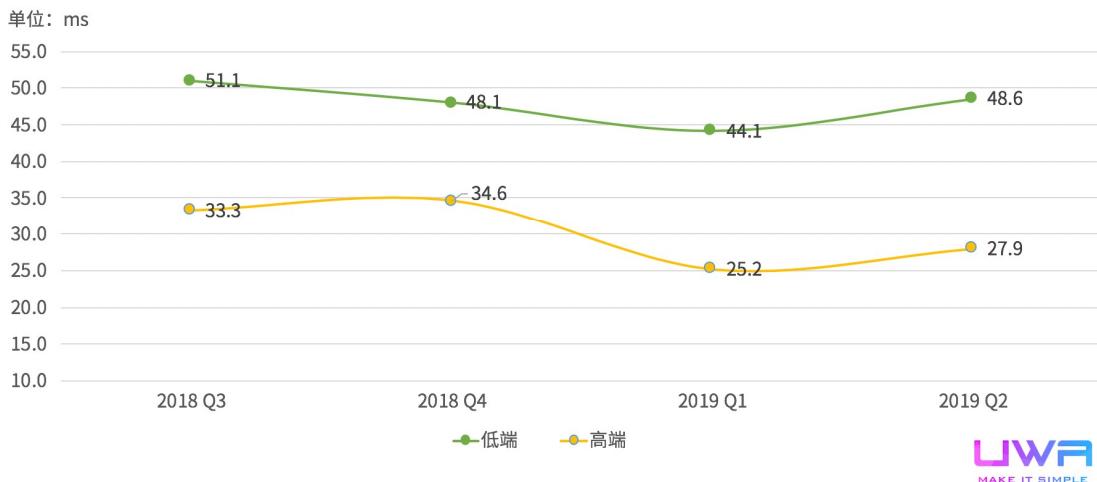
在我们驻场优化的项目中，我们看到项目运行时仍然会有大量的 GameObject 在频繁实例化和 Destroy。对此，我们建议研发团队重点查看 [UWA 线上性能报告中资源管理分类的“资源实例化/激活”页面](#)，通过它可以快速掌握到底具体哪些 GameObject 在不停地被执行实例化操作，从而更为高效地减少不必要的性能开销。

资源名称	类型	操作方式	操作次数	耗时(ms)
soldier_spear_0	GameObject	GameObject.Instantiate	525	195.03
BuffIcon	GameObject	GameObject.Instantiate	304	206.39
soldier_sword_0	GameObject	GameObject.Instantiate	254	88.85
soldierspear	GameObject	GameObject.Instantiate	215	87.58
agent_soldier_spear	GameObject	GameObject.Instantiate	212	719.10
HintRoad	GameObject	GameObject.Instantiate	205	38.55
ItemIcon	GameObject	GameObject.Instantiate	202	431.78
soldier_bow_0	GameObject	GameObject.Instantiate	166	50.49
soldiersword_mid	GameObject	GameObject.Instantiate	130	52.11
HintNormal	GameObject	GameObject.Instantiate	123	39.99

Shader.Parse CPU均值分布



Shader.Parse CPU均值耗时趋势



Shader.Parse 操作是比较突出的性能杀手，目前平均每次调用的 CPU 耗时主要分布在 22.5~56.5ms。这里需要研发团队最需要关注以下三点：

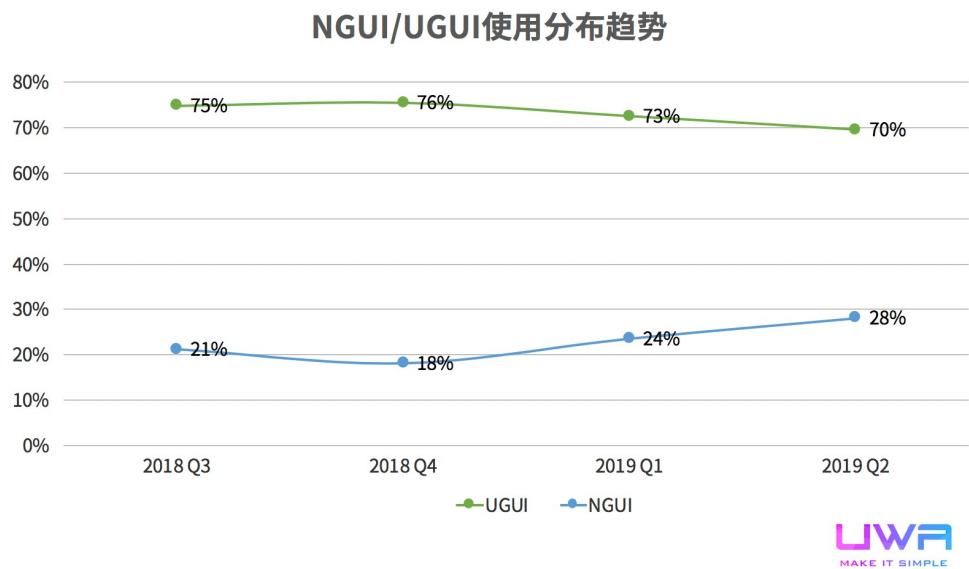
- 1) Shader 在游戏运行过程中是否存在冗余情况，即重复加载的情况；
- 2) 是否有自定义的 Shader 资源放入到 Always Included 中，如有，请尽量去除；
- 3) 移动端上 Standard Shader 的使用是否确实有必要。没有必要，则详细检测并删除。

三、UI 模块

严重程度：地狱



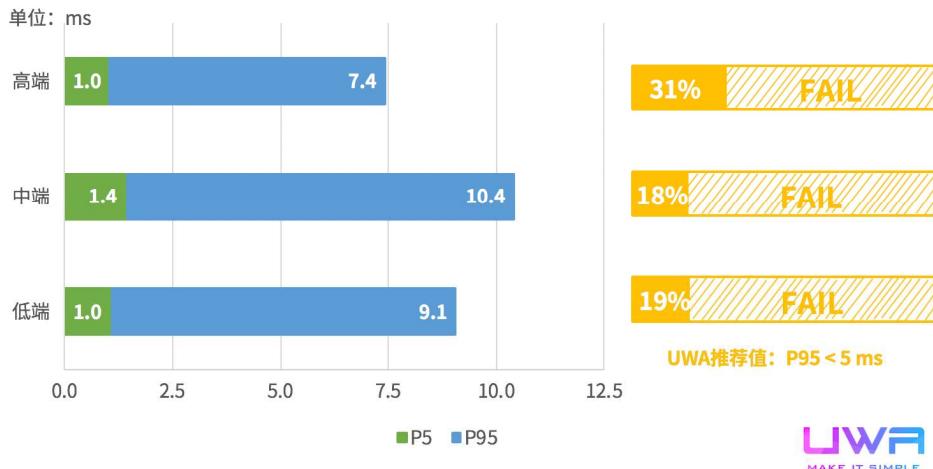
目前我们主要统计了 NGUI 和 UGUI 的占比以及详细的使用情况，在 UWA 测评过的项目中，还存在一些其它 UI 插件比如 FairyGUI，基于目前数量较少，并未加入统计。



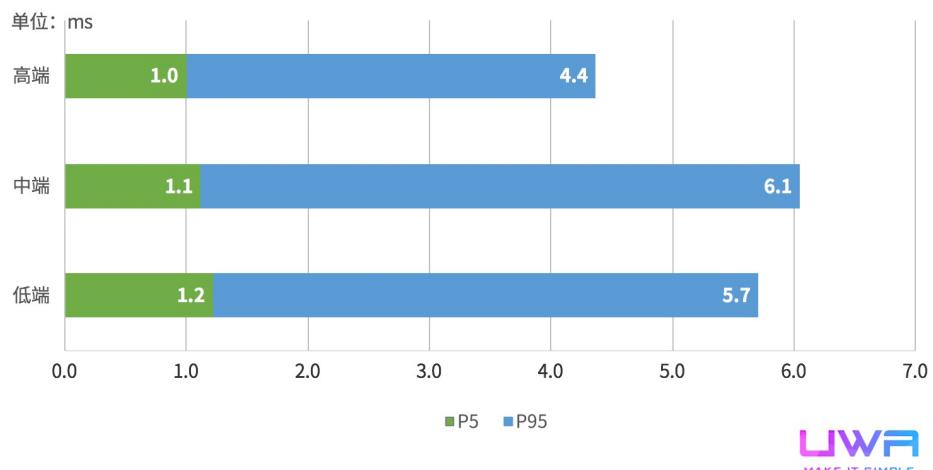
在去年的报告中，我们看到了 UGUI 的使用占比呈现反转的趋势，一跃达到 60%+ 的占比。而在刚刚过去的四个季度中，UGUI 的使用占比依然强势，最高时可达 76%。

下面我们将分别说明 NGUI 和 UGUI 的 CPU 耗时和堆内存占用情况。

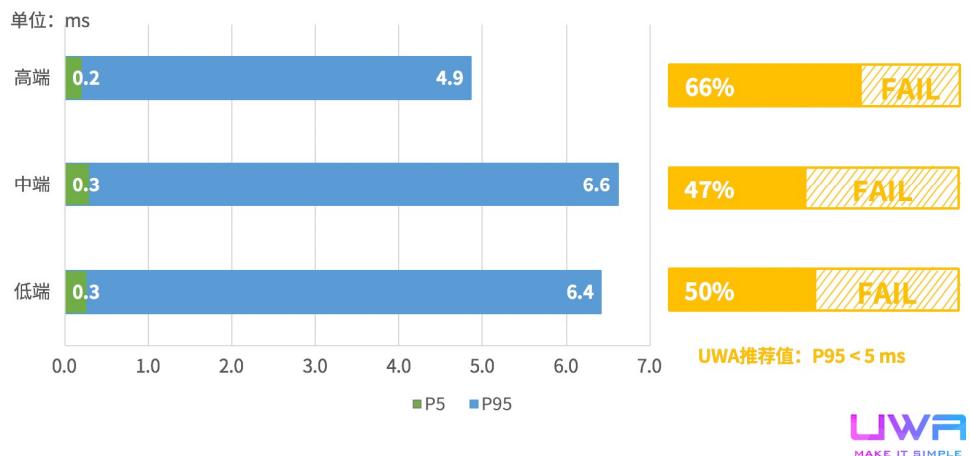
NGUI 的CPU耗时主体范围（P5~P95）分布



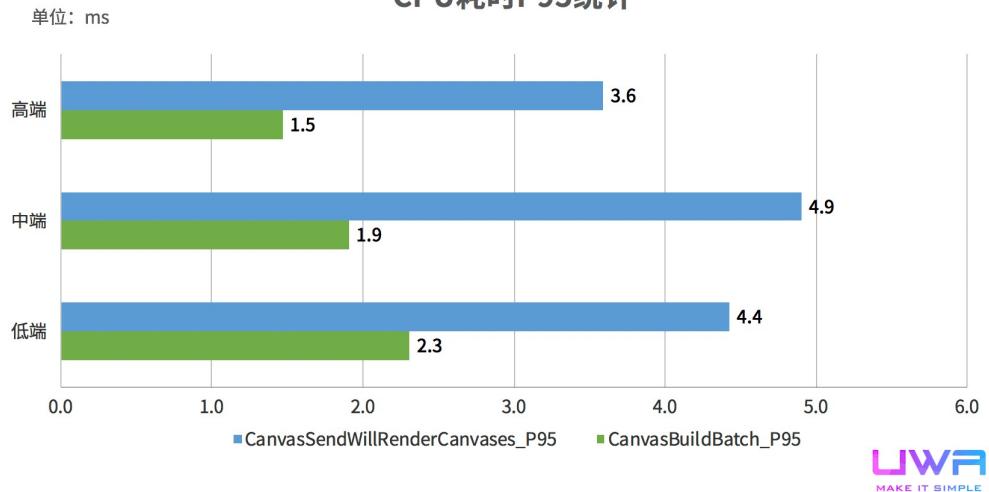
UIPanel.LateUpdate CPU 主体范围（P5~P95）分布



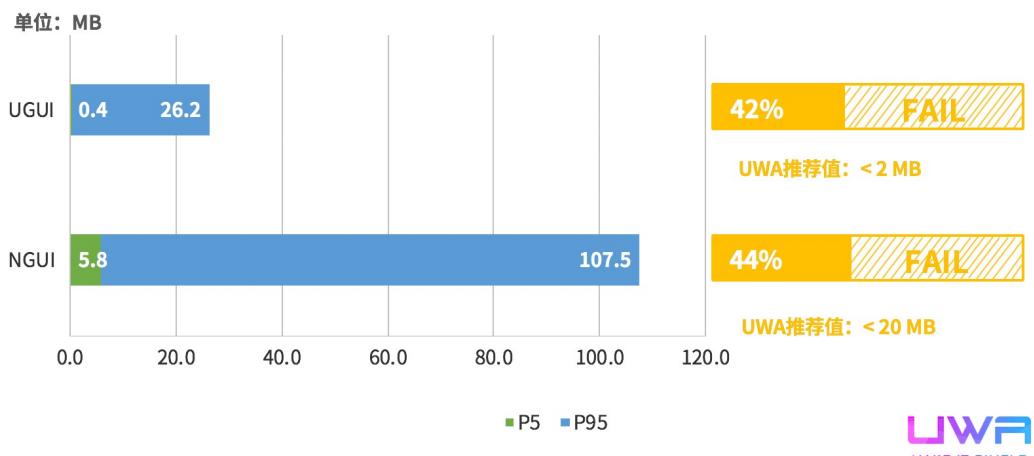
UGUI 的CPU耗时主体范围（P5~P95）分布



Canvas.BuildBatch/Canvas.SendWillRenderCanvas CPU耗时P95统计



NGUI/UGUI 堆内存主体范围 (P5~P95) 统计



UI 模块的性能开销依然很高，但相较于去年相比确实有进一步提升。这主要是在这一年多以来，各大研发团队对于 UGUI 的理解和掌握都在不断深入。当然，现在 UGUI 的明显性能问题主要有以下四方面：

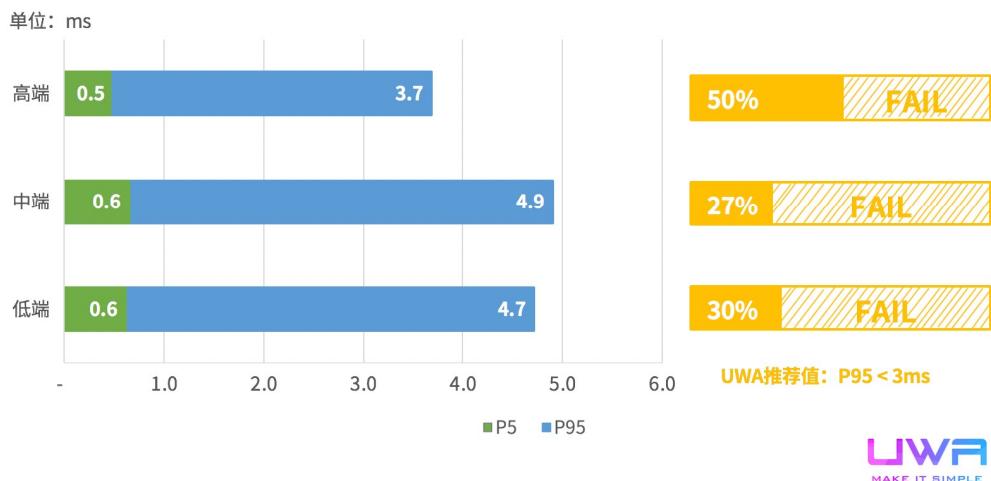
1. Draw Call;
2. 重建；
3. OverDraw；
4. 主线程阻塞。

针对以上问题，大家可以通过 [UWA 学堂](#) 中 UI 相关的相关技术问题和视频进行学习和了解，相信对帮助大家快速提升 UI 方面的技术理解大有裨益。

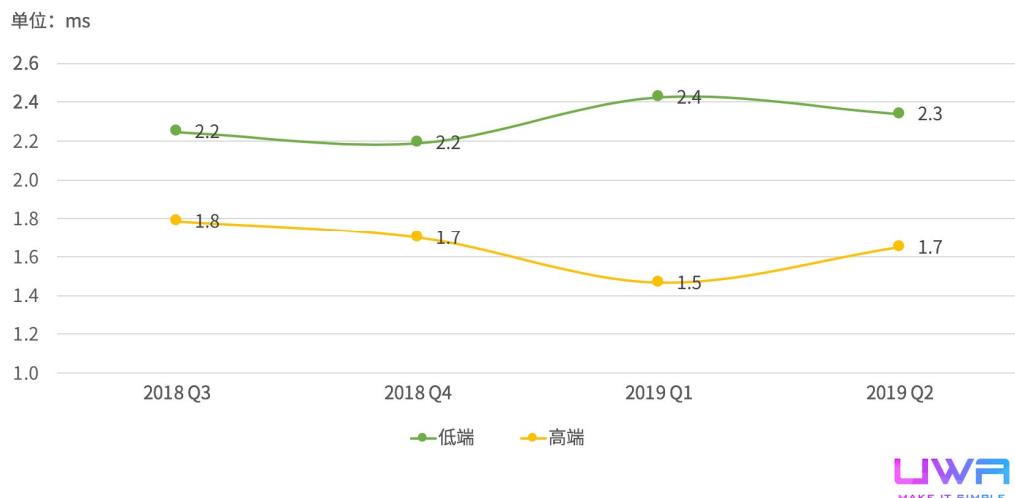
四、动画模块

严重程度：普通

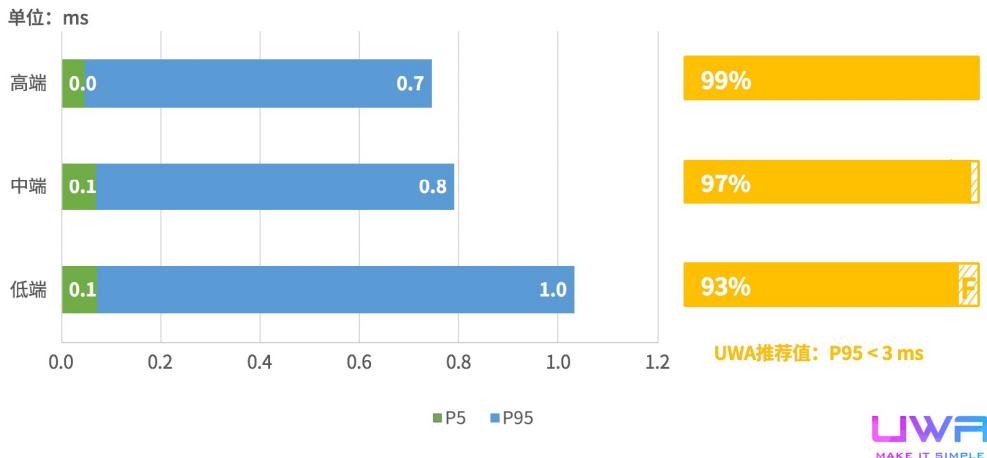
动画模块CPU主体范围（P5~P95）分布



动画模块耗时趋势



Mesh Skinning.Update CPU 主体范围 (P5~P95) 分布

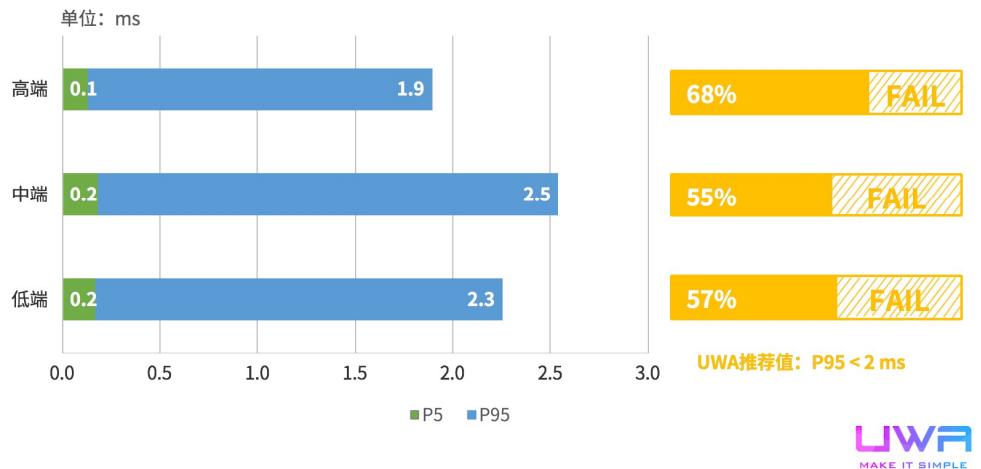


动画模块在最近一年来的性能表现一直较为平稳，这说明大家对于动画模块中应有的性能优化点掌握已经较为到位。虽然 MMO 项目中的角色、NPC 和怪物等数量持续有在增加，但在这些模型多数是通过 GPU Skinning 的方式来完成。这种方式对于实现场景中的大量同种怪物非常有效。

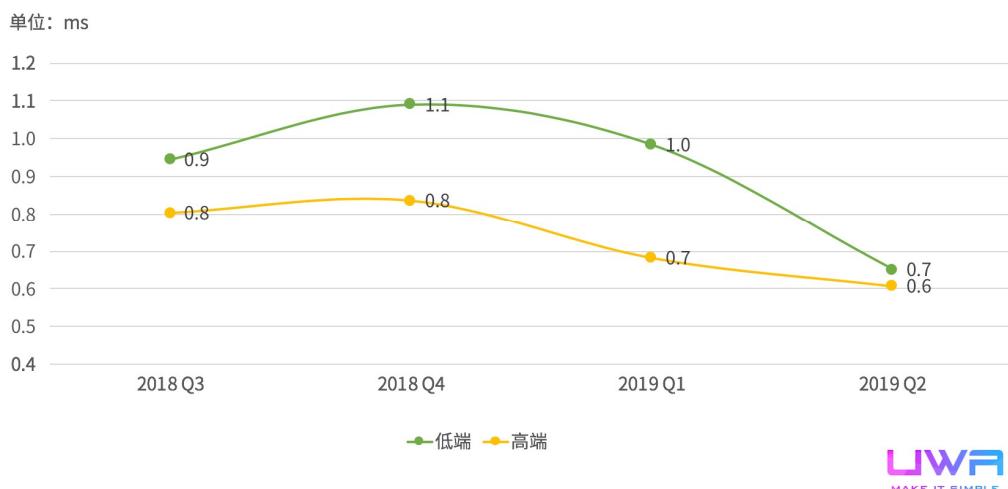
五、物理模块

严重程度：普通

物理模块CPU主体范围（P5~P95）分布



物理模块CPU均值趋势

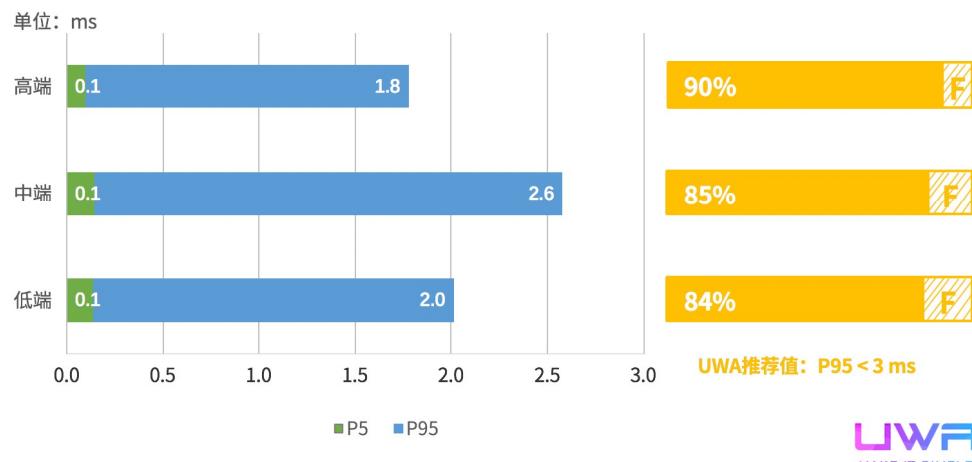


物理模块的性能普遍较好，且出现了持续下降的趋势。对于使用 Unity 2017.4 版本以后的团队，Auto Simulation 和 Auto Sync Transforms 是大家需要关心的参数，以避免不必要的性能耗时。

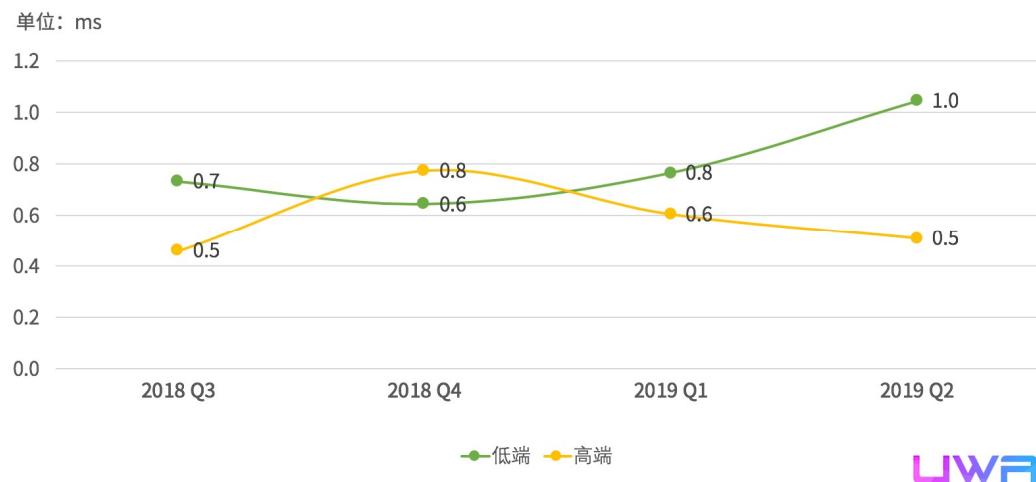
六、粒子模块

严重程度：普通

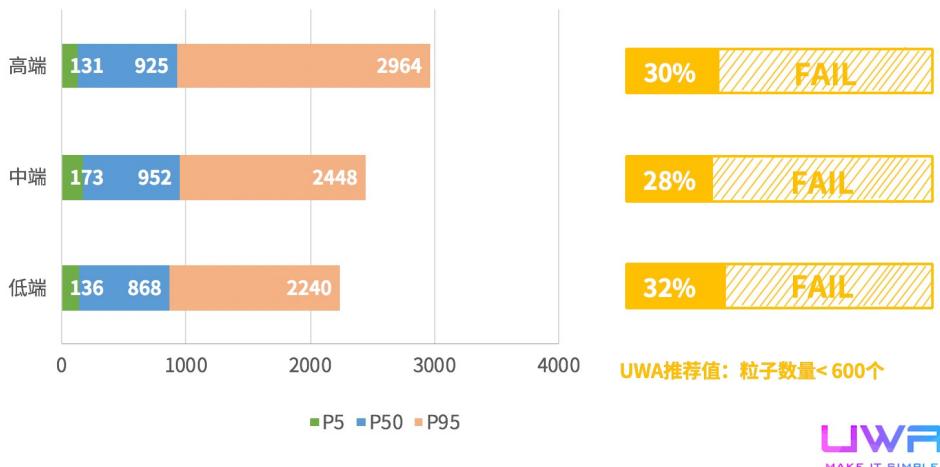
粒子系统CPU主体范围（P5~P95）分布



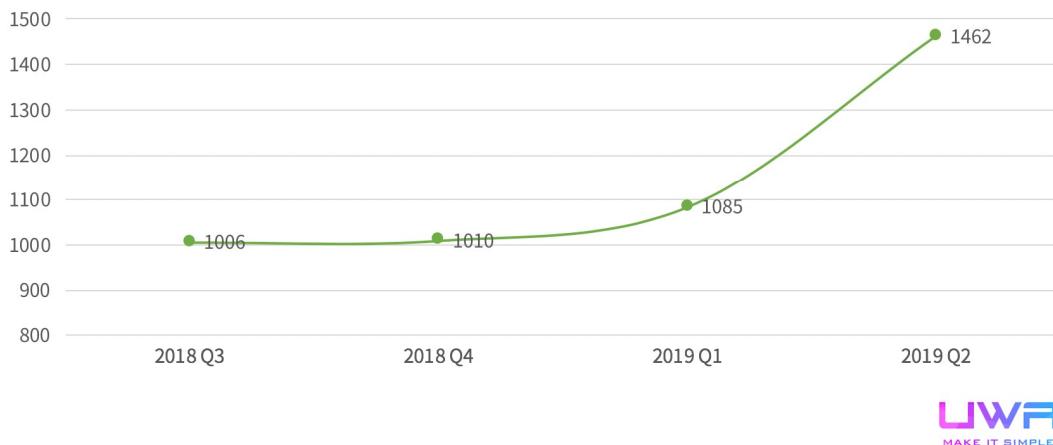
粒子系统CPU均值趋势



粒子系统数量主体范围 (P5~P95) 分布



粒子系统使用数量趋势



粒子系统的 CPU 开销普遍较低，但总体使用数量峰值在 2019 Q2 后大幅上升。我们依然建议研发团队尽可能将数量峰值控制在 600 以下（低端设备）和 1000 以下（中高端设备）。对此，建议研发团队经常通过以下两方面来检测自己的粒子特效使用情况：

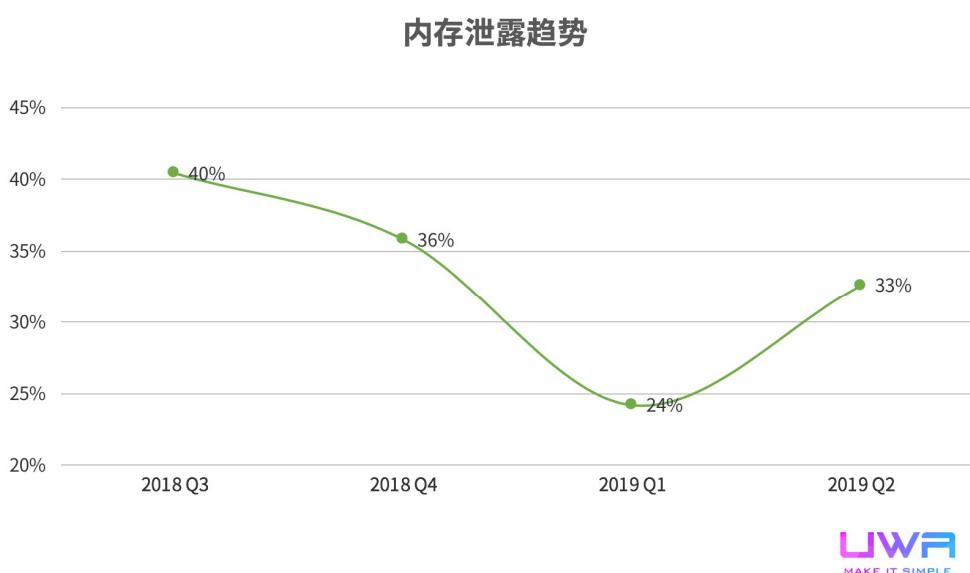
- 1) 粒子系统（特别是技能特效）的配置文件是否过量；
 - 2) 特效中是否含有长久不用的粒子系统。
-



MMORPG 手游内存模块开销分析

一、内存泄露

严重程度：噩梦



内存泄露问题在过去的一年中得到了很大的改善，虽然 2019 Q2 的泄露项目占比有所回升，达到了 33%，但相较于去年同期（2018 Q2 为 55%）仍然下降了 22%。

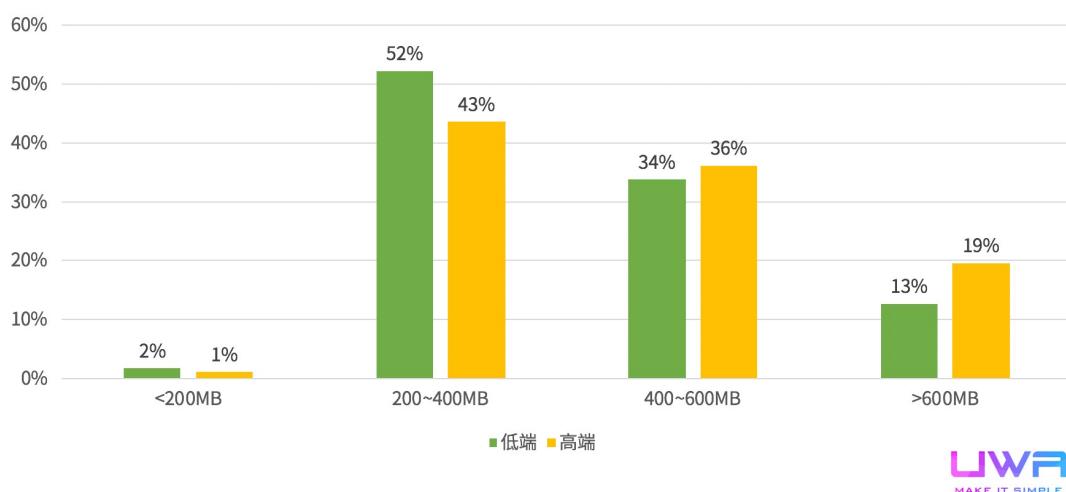
内存泄露的问题得以很好的解决，主要还是因为大家可更为直观、精准地查看其具体泄露的 C# 函数和 Lua 函数。对此，我们建议研发团队后续可更为频繁地进行检测和监控，即可进一步降低堆内存泄露的风险。研发团队可通过以下两种方式高效地对堆内存泄露函数进行分析和优化：

- 1) [UWA GOT Online 中的详细 Mono 堆内存分析和 Lua 性能分析报告](#);
 - 2) [UWA 线上深度测评中的详细 Mono 堆内存测评报告](#)。
-

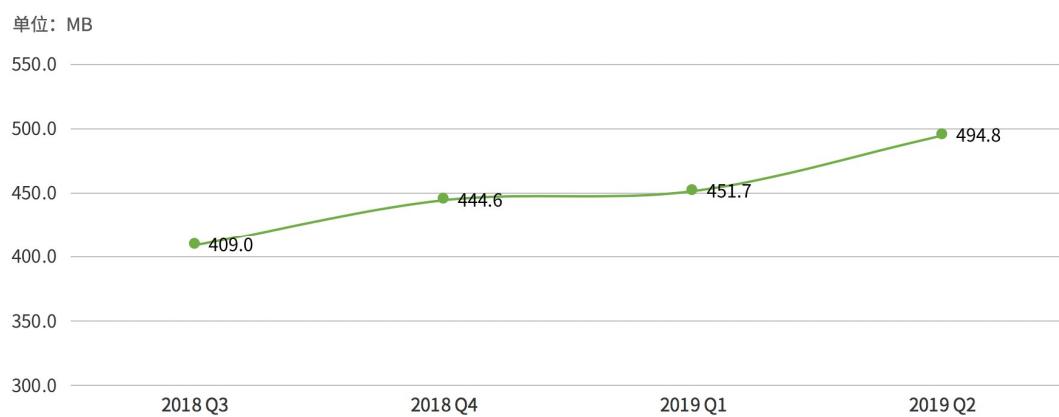
二、总体内存

严重程度：地狱

总体内存峰值分布



总体内存峰值趋势

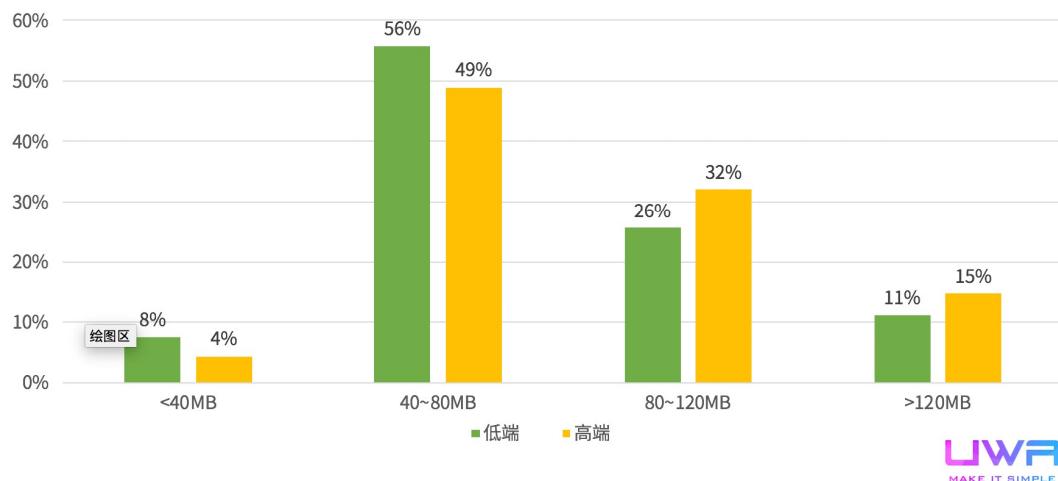


总体内存最近的一年里，峰值均值稳步提升，共计提升了 85MB。随着高端移动设备的普及、MMO 游戏的逐步重度化，内存逐步提升是一个必然的趋势。目前，项目中的主要内存瓶颈依然是资源和 Mono 堆内存。对此，我们将在接下来的文章中进行详细分析。

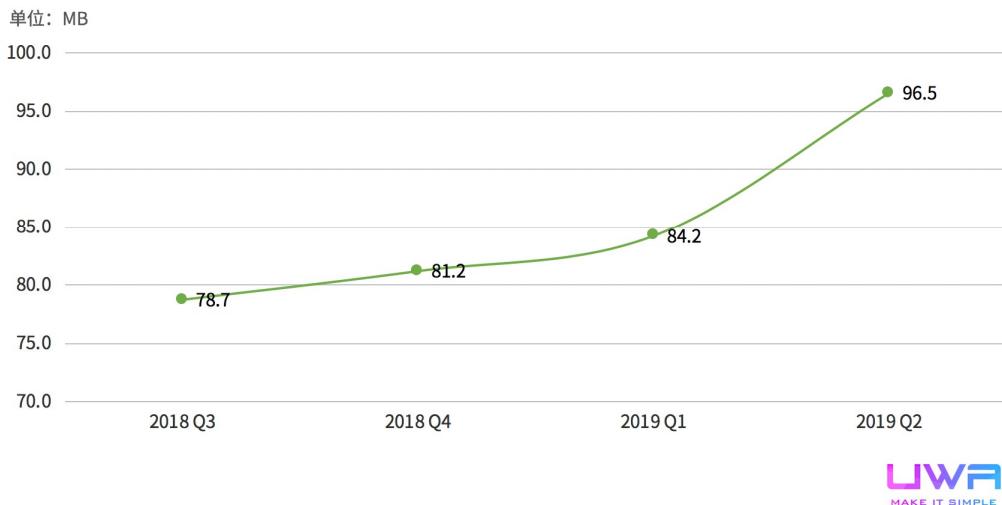
三、总体堆内存

严重程度：地狱

总体堆内存峰值分布



总体堆内存峰值使用趋势



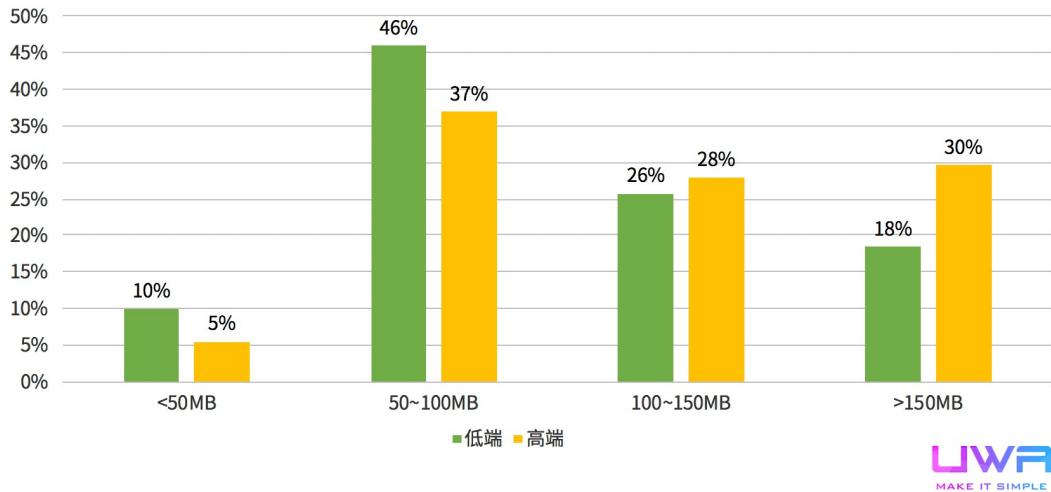
总体堆内存在最近一年里上升趋势非常明显，这主要是游戏的重度化所致。但另一方面，也是研发团队对于 Mono 堆内存分配的疏于管理所致。就目前而言，Mono 堆内存过高主要是因为配置文件的序列化库使用不当所致，这是 UWA 在接下来一年中所要攻坚的重点，让研发团队能够更为合理地对序列化信息进行管理。

项目的内存占用很大一部分来自于资源的使用，下面我们将对项目中主流资源的使用情况进行分析。

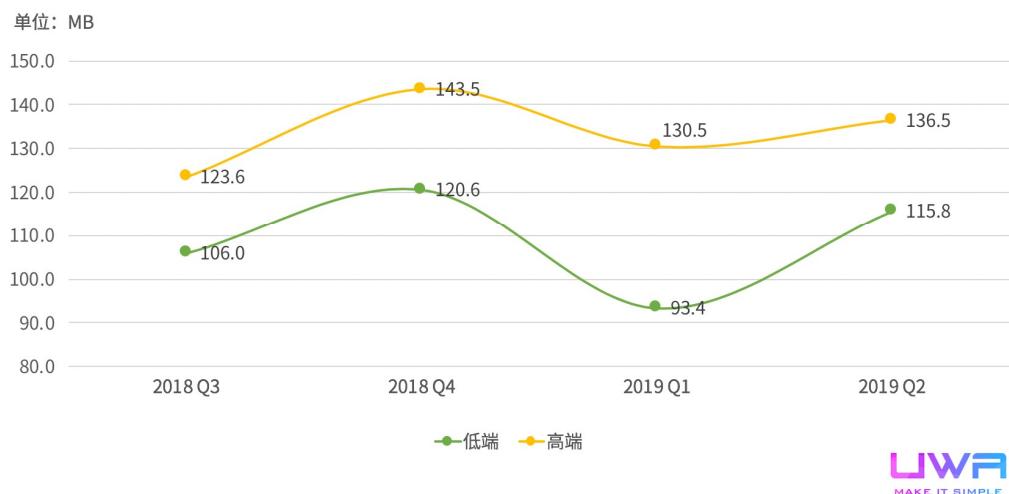
四、纹理资源内存

严重程度：地狱

纹理资源内存峰值分布



纹理资源内存峰值使用趋势

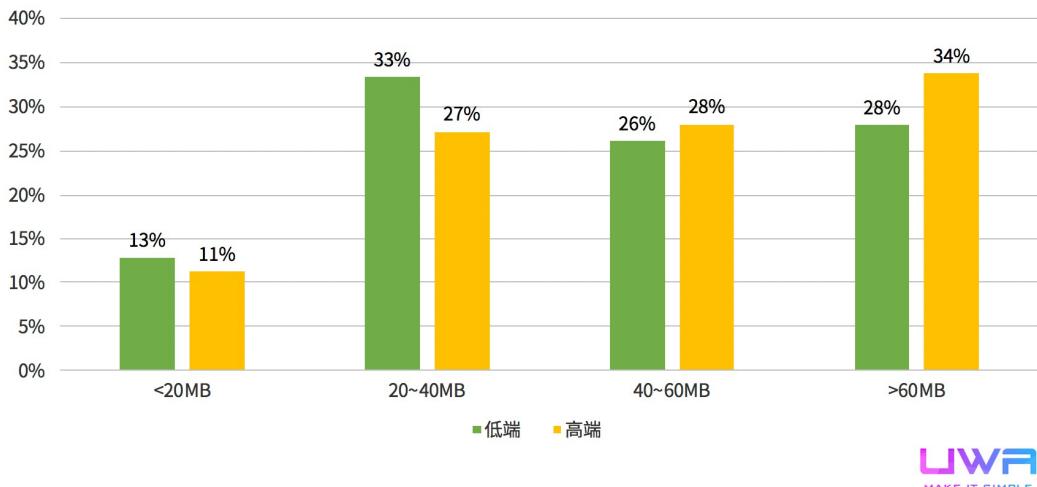


如上图所示，对于纹理资源来说，在过去的 4 个季度中，其内存占用处于震荡阶段。目前，大多数团队对于纹理的使用格式都已经有了清晰的认识，建议研发团队对于纹理 Mipmap 使用情况、渲染利用率等进行进一步的检测和完善。

五、网格资源内存

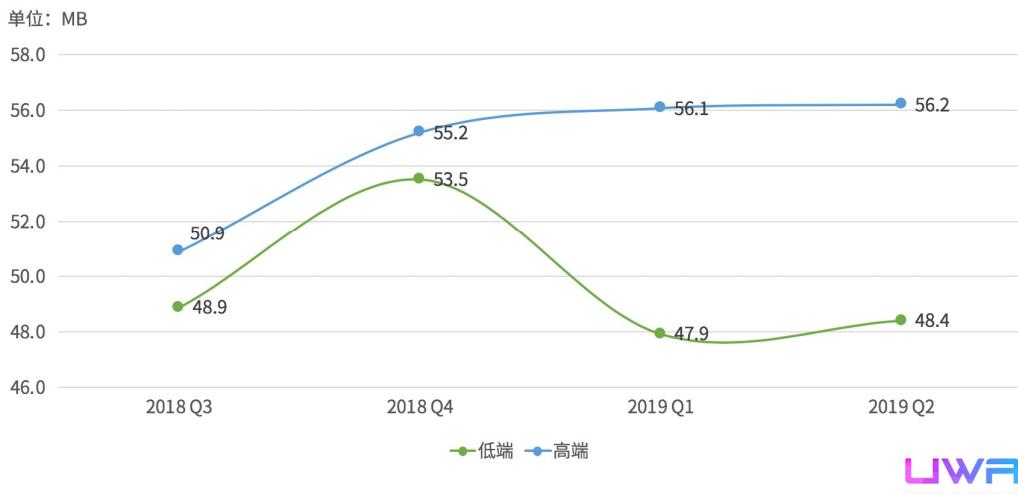
严重程度：地狱

网格内存峰值使用分布



UWA
MAKE IT SIMPLE

网格资源内存峰值使用趋势



UWA
MAKE IT SIMPLE

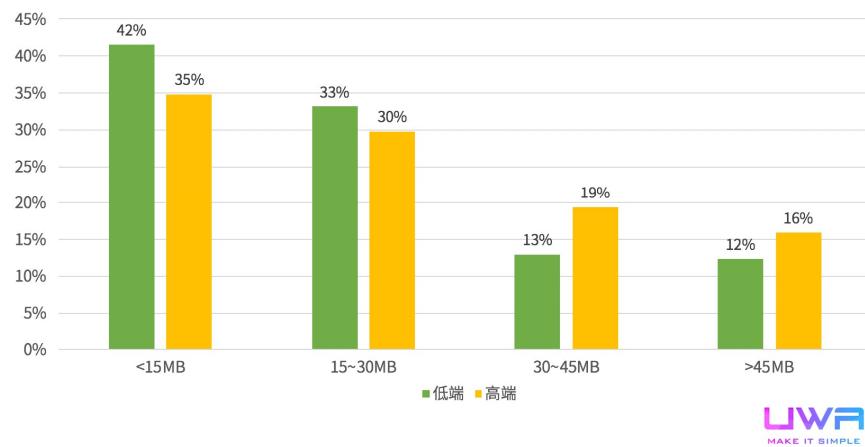
- 1) 网格资源内存峰值主要分布为 12.7~96.1 MB，且主要集中在 60MB 以内。
- 2) MMORPG 游戏的场景普遍较大、角色普遍较多，因此，其网格资源量明显

高于其它类型游戏。但从使用趋势上来看，其最近一年的使用趋势较为平稳，均值均在 50MB 上下浮动。

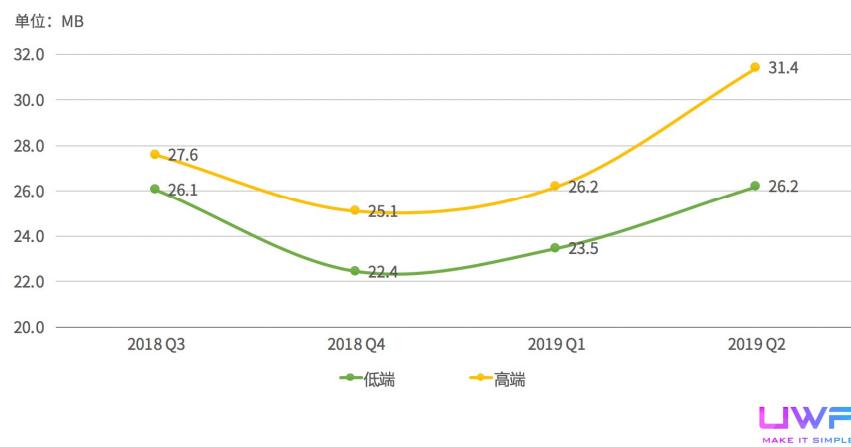
六、动画资源内存

严重程度：普通

动画资源内存峰值分布



动画资源内存峰值使用趋势

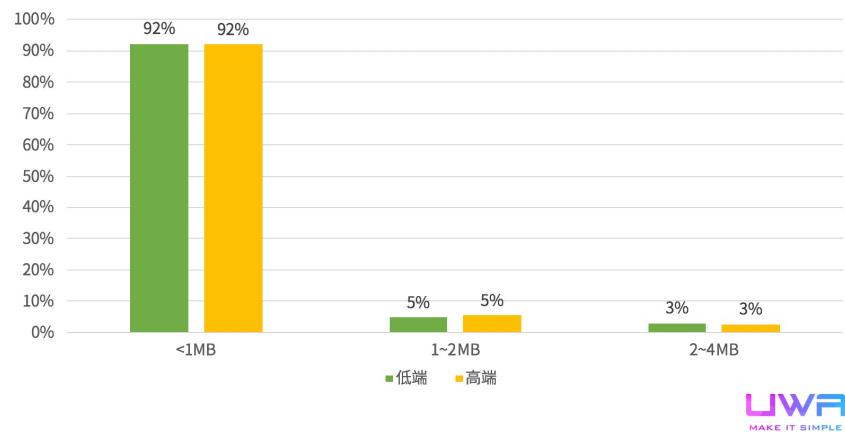


动画模块的内存使用趋势较为平稳，平均峰值内存长期控制在 20~30MB 区间之内。

七、Shader 资源内存

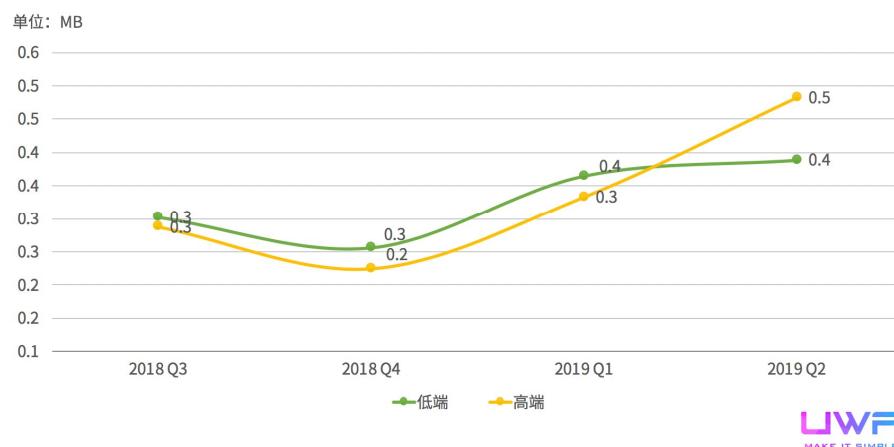
严重程度：普通

Shader资源内存峰值分布



UWA
MAKE IT SIMPLE

Shader资源内存峰值使用趋势



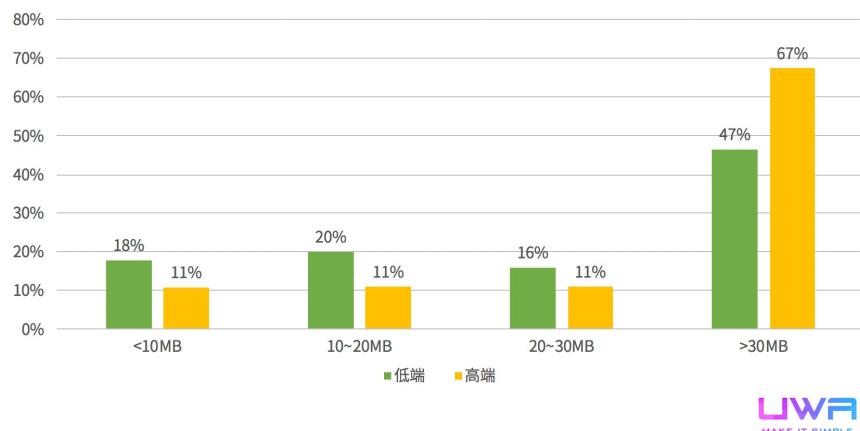
UWA
MAKE IT SIMPLE

Shader 内存占用的上升趋势较为明显，但本身内存占用很小。这里，UWA 仍然需要提醒的是，Standard Shader 在大家的项目中被经常误引入进来，建议大家在 [UWA 性能简报](#)中的“具体资源使用信息”中特别关注 Standard Shader 的使用情况。

八、 RenderTexture 资源内存

严重程度：噩梦

RenderTexture资源内存峰值分布



RenderTexture资源内存峰值使用趋势

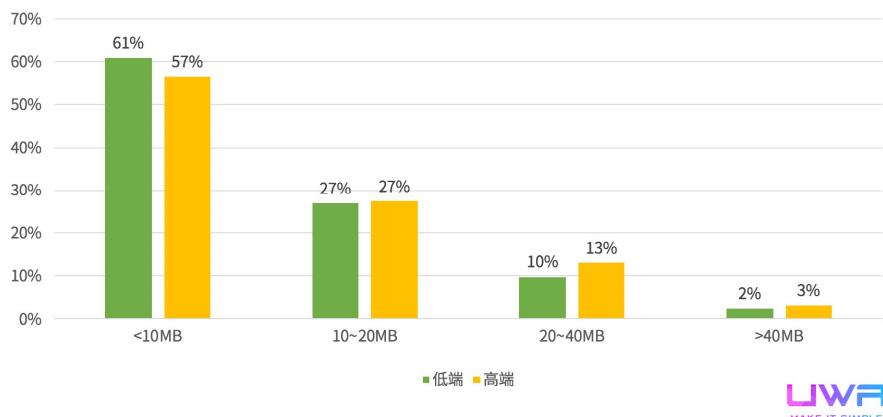


RenderTexture 的内存占用在进入 2019 年以后上升趋势明显，特别是在低端设备上，随着大家对于图像后处理效果的需求提升，后续很可能会出现 RenderTexture 使用程度大幅提升的情况，因此，建议研发团队对 RenderTexture 密切关注。

九、粒子系统资源内存

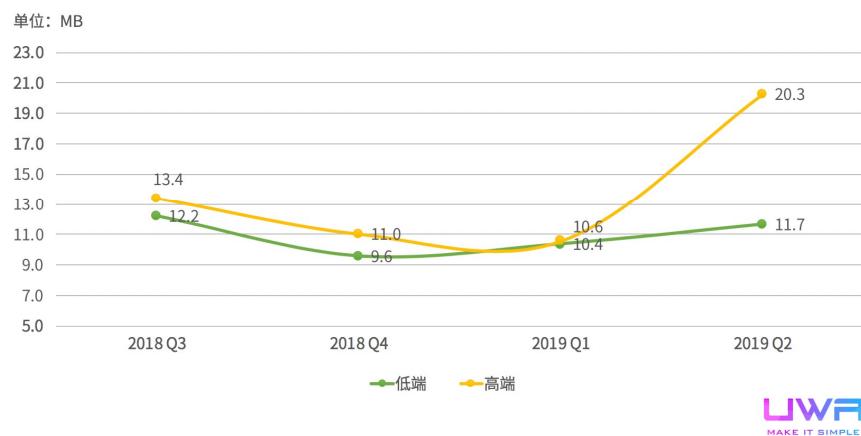
严重程度：噩梦

粒子系统资源内存峰值分布



LWA
MAKE IT SIMPLE

粒子系统内存峰值使用趋势



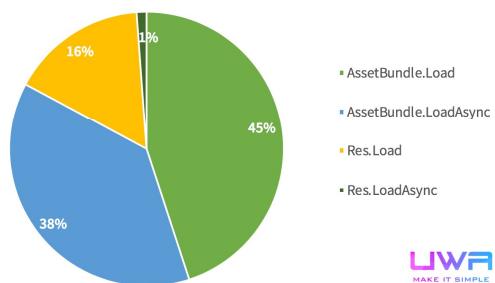
LWA
MAKE IT SIMPLE

粒子系统的使用数量过大，从而导致其内存占用过高。对此，其最有效的优化方法还是降低粒子系统的使用数量。

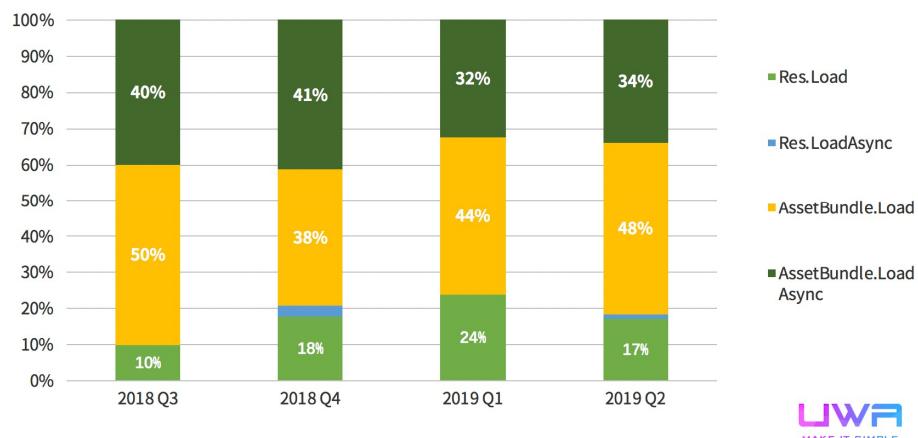


MMORPG 手游资源管理分析

资源加载方式调用占比



资源加载方式调用趋势

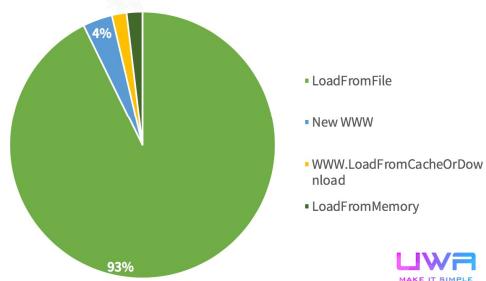


AssetBundle 加载方式 (Load 和 LoadAsync) 的使用占比在过去的一年中都保持在 80% 左右。毫无疑问，AssetBundle 加载方式是目前绝大多数研发团队的资源加载首选。

在接下来的一年中，我们建议大家使用 AUP (Async Upload Pipeline) 功能来尝试达到更高的加载效率。关于 AUP 中需要特别关注的技术点，建议查看

UWA DAY 2019 中的相关分享 [《Unity 引擎加载模块和内存管理的量化分析及优化方法》](#)。

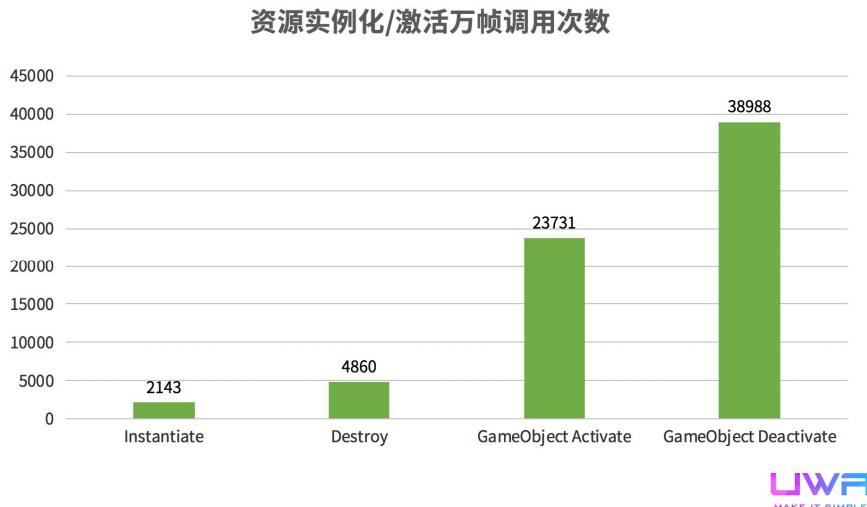
AssetBundle加载方式调用占比



AssetBundle加载方式调用趋势



LoadFromFile 加载方式一家独大，其使用占比从去年的 74% 增涨到今年的 94%。LoadFromFile+LZ4 的组合在移动游戏中加载性能优势明显，已经逐步成为了移动游戏资源加载的不二选择。



- 1) 对于 Instantiate/Destroy 的完善，研发团队时刻关注缓存池的使用是否合理；
- 2) 对于 Active/Deactive 的完善，则需要研发团队尽可能注意避免 UI 和动画角色的频繁调用，从而尽可能避免不必要的调用开销。



UWA 对于 MMORPG 手游研发团队的建议

- 1、警惕渲染模块、逻辑代码和 UI 模块这三大性能杀手！随着精品化 MMORPG 项目的日益增多，这三项在接下来依然是研发团队在性能优化时的重中之重！
- 2、内存泄露问题在今年的项目研发中得到了相当大的缓解，项目占比较之去年同期下降了 22%，在 [UWA 线上测评报告](#) 和 [UWA GOT 中的内存泄露分析功能](#) 的支持下，我们相信研发团队不仅可以快速检测和定位堆内存的具体泄露问题和出处，还可以高效对其进行完善和修复。
- 3、在内存优化方面，Mono 堆内存、纹理、网格、RenderTexture 和粒子系统依然是大家接下来需要关注的重点。特别是 Mono 堆内存部分，对于序列化信息的管理已经刻不容缓，在接下来的一年，UWA 会尽可能为各大研发团队解决和监控这一内存难题。
- 4、资源加载方面，LoadFromFile(Async)已经是移动游戏项目中的主流加载方式，AssetBundle.Load(Async)同样也是资源加载的主流方式。接下来的一年中，LoadAsync 方式和 AUP 功能将在各大团队中使用，这一方面建议国内各大研发团队开始着手进行学习和研究。

5、GPU 性能问题在过去的一年中逐步凸显，在不少超重度 MMO 游戏中，GPU 已经俨然成了游戏项目的主要瓶颈，对此，建议研发团队对于 OverDraw、BandWidth 和 ALU 进行密切关注，UWA 在驻场深度优化过程中，就这三方面的性能瓶颈不断进行分析和研究，后续待成熟后会向大家进行有针对性的分享。

*本报告中的数据最终解释权归 UWA 所有。

关于侑虎科技

侑虎科技（上海）有限公司成立于 2015 年 8 月，主要开发并销售移动应用分析与优化工具，并提供开发移动应用产品的技术支持和咨询服务，目前提供 1) 性能测评与优化 2) 资源检测与分析 3) UWA GOT Online 三大工具，帮助开发者在短时间内大幅度提升性能表现；同时其搭建的知识分享型社区使广大开发者收益。

UWA 博客：blog.uwa4d.com

UWA 问答：answer.uwa4d.com

UWA 开源库：lab.uwa4d.com

UWA 学堂：edu.uwa4d.com