

主播端功耗优化

“老冰棍专项”

职责：技术主R

时间：2023.Q2

”



主播端功耗优化 - 背景

- 春季盛典，现场主播需要用冰袋降温才能正常直播，测试同学需要人工给主播每小时换一次冰袋
- 内部9人连线功能上线，产品体验，出现多次发热死机
- 线上发热数据

	极限发热	极限发热	严重发热	严重发热
指标	Android	iOS	Android	iOS
大盘	0.9%	4.2%	5.2%	51.2%
RTC场景	1.23%	6.5%	6.8%	64.3%

“ 极限发热 > 严重发热 > 轻微发热 ”

“ 极限发热下，核心功能存在可用性问题 ”

“ 这是4月份的数据，随着气温升高，发热现象会更严重 ”



主播端功耗优化 - 挑战

“ 功耗优化技术栈状态 ”

- 应用开发层业界功耗优化经验非常少，和厂商交流，他们的硬件功耗测试设备动辄几十万，整个功耗实验室入门门槛千万级别
- 软件层功耗相关的工具不够完善，**Profile**和定位功耗问题存在较大阻碍
- 功耗是个综合指标，涉及到cpu, gpu, display, radio, audio等多个硬件，对技术深度和广度要求都比较高
- 优化了功耗，并不等于解决了发热问题，还要考虑到主播使用习惯和使用环境的影响
- 直播和大型3D游戏比较类似，都是长时间 × 高频 × 高复杂度运算
 - 游戏用户通过高性能设备，水冷，强力风扇来解决；显然大部分手机直播用户不具备这些条件

“ 思考 ”

- 如果一般的性能优化是登山，那么功耗优化更像是徒手攀岩
- 用俊哥的话讲：要是不难，还花这么多薪水雇你干啥！



→ 1) 功耗原理分析和Profile

2) 渲染优化

3) 极限发热下的热缓解

手机发热原理分析

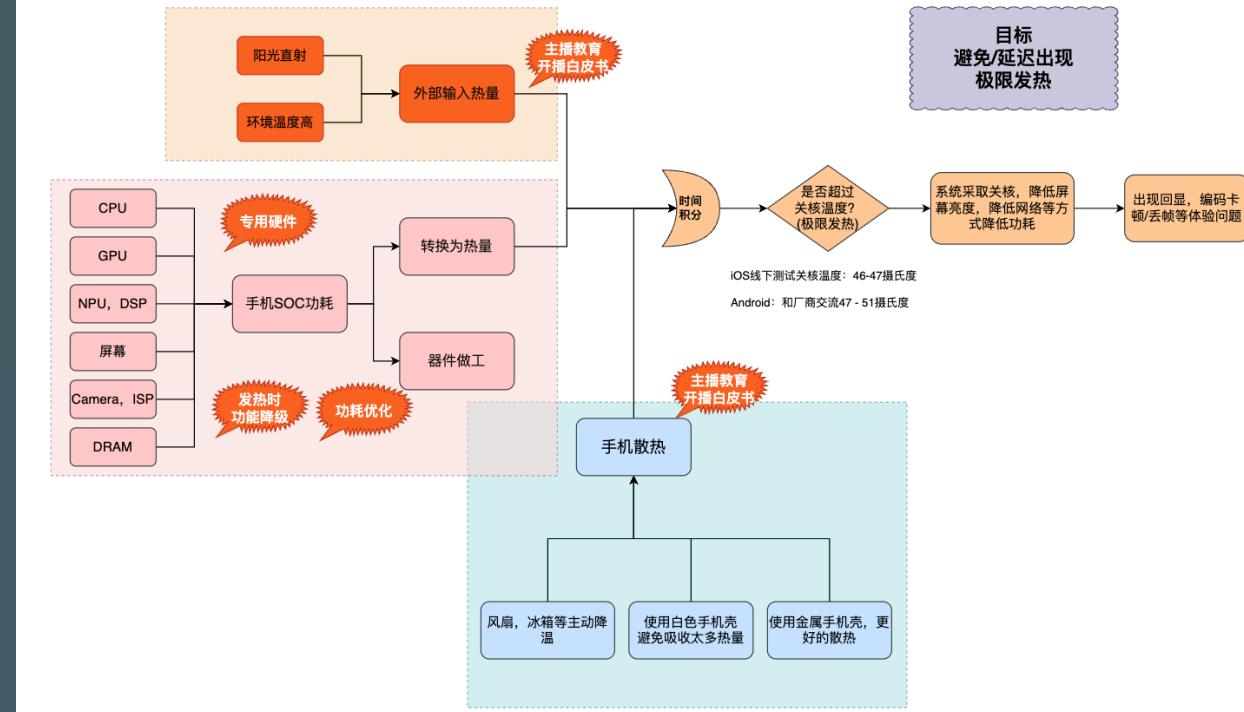
$$\text{手机发热} = \int_0^T (P_{SoC\text{功率}} - P_{做工\text{功率}} + \dot{Q}_{\text{外部输入}} - \eta_{\text{散热效率}}) dt$$

“ 观察 ”

- iOS/Android系统都有说明，手机过热会进行**CPU降频/关核**，降低网络性能，降低屏幕亮度等手段进行降温，极端情况会导致发热关机
- 工信部过热红线标准是外壳**48摄氏度**
- 通过外部降温手段，多人**RTC**等高耗场景，即使**CPU锁核**，也可以持续稳定且流畅运行（辅助说明，设备算力不是瓶颈）

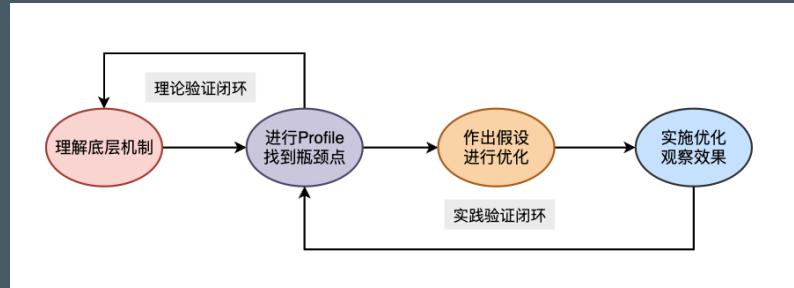
“ 洞察 ”

- 如果没有极限发热，大部分手机的绝对性能正常情况够用
- 主播端功耗高本身不是问题，功耗导致的极限发热才是问题
- 不要只局限于技术视角，工程师要利用一切有效的手段来解决问题



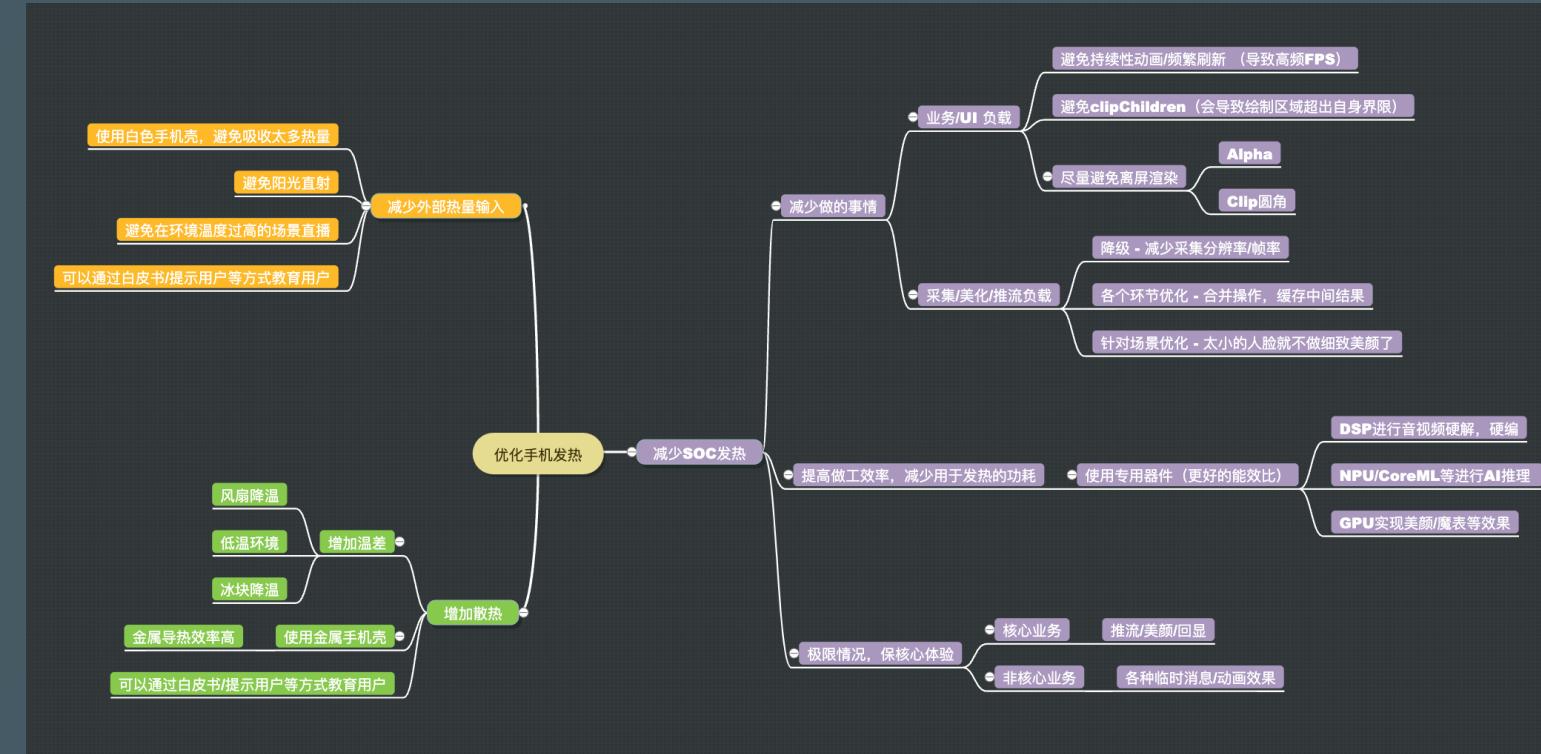
功耗优化整体思路

“方法论”



“洞察”

- 功耗优化：降低功耗负载
- 负载匹配：专用器件有更高的能效比
- 极限发热：优先保核心体验



硬件功耗分析：静态

“洞察”

- 功耗和发热与硬件息息相关，iOS功耗控制的更好，Android的散热水平更强
- 传统的优化集中于CPU，但是CPU功耗水平较低
- Camera, Screen, Audio, Video值得深入挖掘

“挑战”

- 关于CPU的优化经验/工具是最丰富的，但是在功耗优化中不能承担起大梁
- 其它器件的优化，需要深度的专业知识，对技术栈的挑战比较高

元器件	状态	解释	功耗
battery	battery.capacity	总电池容量（以毫安时为单位）	3000mA
camera	camera.avg	针对典型相机应用，相机子系统的平均电量消耗。	600mA
	camera.flashlight	相机闪光灯模块开启时消耗的平均电量。	500mA
screen	ambient.on	屏幕在低电耗/微光/始终开启模式（而非关闭模式）下消耗的额外电量。	0.5mA
	screen.on	屏幕以最低亮度开启时消耗的额外电量。	100mA
cpu	screen.full	与处于最低亮度的屏幕相比，当屏幕处于最高亮度时消耗的额外电量。	800mA
	cpu.idle	CPU（和 SoC）处于系统挂起状态时系统消耗的总电量。	
	cpu.awake	CPU 处于调度空闲状态（内核空闲循环）时消耗的额外电量；系统没有处于系统挂起状态。	
audio	cpu.active	CPU 以不同速度运行时消耗的额外电量。	小核心： $10-30mA * 4 = 40-120mA$ 大核心： $25 - 60mA * 4 = 100-240mA$ 150 - 360mA之间
	audio	通过 DSP 进行音频解码/编码时消耗的额外电量。	100mA
video	video	通过 DSP 进行视频解码时消耗的额外电量。	150mA

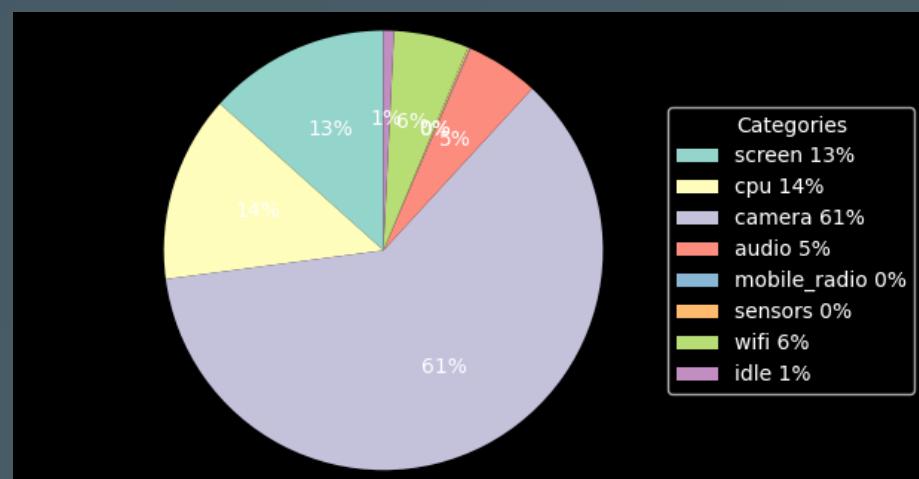
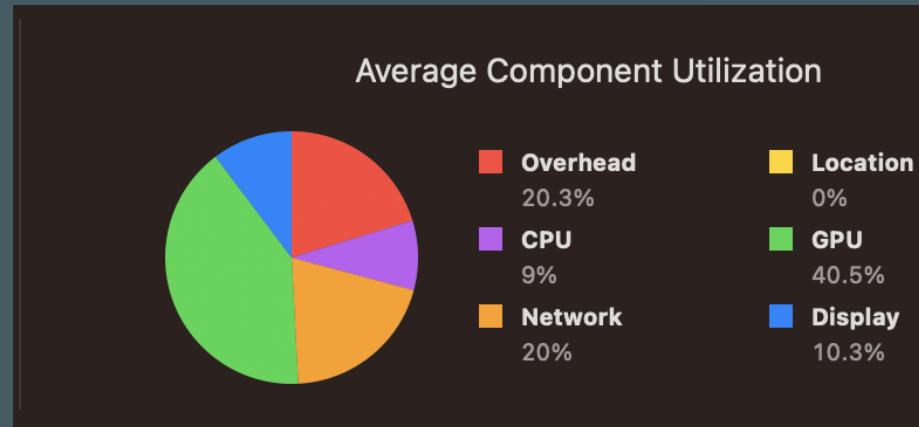
硬件功耗分析：动态

“ 右侧为双端开播50分钟不同器件功耗占比（上：iOS，下：Android） ”

“ 洞察 ”

器件	负载	优化点
GPU	Camera预览；特效；UI渲染；系统合成	降低UI刷新频率 减少重绘区域大小 特效算法/素材优化
Camera	CMOS采集；ISP处理	参数调优，降低采集分辨率
Network	推流；业务接口；长链接	batch 请求，减少冗余请求
Display	屏幕亮度；刷新率	降低UI刷新频率
CPU	业务逻辑；UI布局；AI推理	优化高频调用，AI推理使用 CoreML/NPU

“ 备注：Android因为工具支持问题，暂时缺少GPU数据，可以当作参考 ”



业务视角功耗分析

“洞察”

1. 主播端基准功能功耗就非常高
2. GPU增量主要来源于美颜/特效/UI渲染
3. CPU整体占用不高，且随着业务复杂化，增量较小（超预期）
4. 不同场景下，不同元器件的功耗占比变化不大

“直播间基础功耗主要来源”

- 器件的基底功耗 (camera/gpu/display都处于开机状态)
- 摄像头采集 (降低采集帧率/分辨率)
- 编码/推流 (采用硬编)
- UI渲染 (降低渲染频次等)

“观察和思考”

- 只有基础功能，普通室温下主播可以长时间稳定推流
 - 基底功耗在发热和散热的临界点附近
 - 要努力降低基础功耗
- 叠加RTC/大礼物特效后，很快会进入极限发热状态
 - 增量功耗的治理也很重要，也会有明显受益

	场景	进程CPU	系统CPU	GPU	功耗 (mW)	功耗占比
基准直播间	小微直播间无美颜10分钟 (Base)	8.4	23.6	17.2	2230.7	<ul style="list-style-type: none">• CPU: 9%• GPU: 37%• Network: 26%• Display: 6%• Overhead: 22%
+美颜	小微直播间 (有美颜) 10分钟	8.9	22.7	47.6	2401.2	<ul style="list-style-type: none">• CPU: 9%• GPU: 41%• Network: 23%• Display: 6%• Overhead: 21%
+中等直播间	中型直播间 信令回放	9.1	23.8	50.7	2421.7	<ul style="list-style-type: none">• CPU: 9%• GPU: 41%• Network: 23%• Display: 6%• Overhead: 21%
+辛巴直播间	大直播间 信令回放	9.2	24.8	53.3	2443	<ul style="list-style-type: none">• CPU: 9%• GPU: 41%• Network: 24%• Display: 6%• Overhead: 20%

安卓高/中/低端机型 Cpu功耗分析

“洞察”

1. **cpu功耗和频率是超线性关系**, 实测看厂商会倾向于满足性能诉求的情况下, 运行在相对低的频率上
2. 从整体算力上线来讲, **cpu算力是足够的**, 中端机大概也只占用了不到50%的峰值算力
3. 对用户体验的主要影响是手机发热, 导致的器件性能下降和用户体验问题

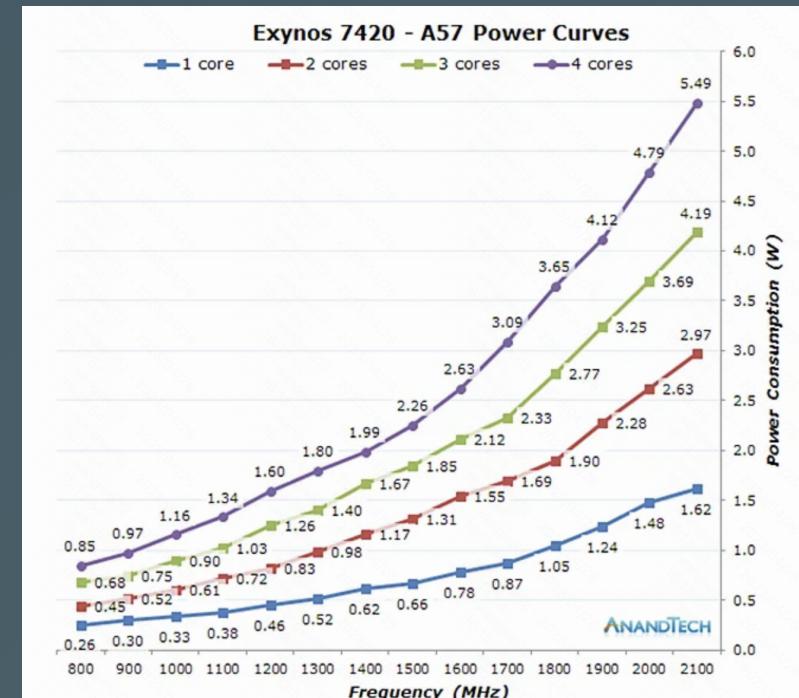
“思考”

- Cpu硬件功耗水平不高 + Cpu占用率适中 + Cpu没有明显的瓶颈点 => 不是功耗治理重点
- 挑战更严峻了: 研发工具最齐备, 经验最丰富的Cpu优化和本次功耗优化关系比较小了

做正确的事情比做会做的事情重要

真实可靠的**Profile**数据带来真正有效的洞察, 性能优化要避免纸上谈兵

直播场景	CPU使用率
低端机 +带美颜/魔表	60%
低端机 -无美颜/无魔表	53%
中端机+带美颜/魔表	44%
中端机+带美颜/魔表+辛巴信令回放	46%
中端机+带美颜/魔表+6人连线	48%



Profile -> 功耗治理方案

治理方向	分析	对用户影响	处理方案	团队
降低采集分辨率/帧率	减少Camera/GPU消耗	大	只有到极限发热才进行降级，正常case要保证体验	直播业务
美颜/特效算法优化	降低GPU消耗	中	尝试CoreML推理AI模型，全链路GPU处理等，上线实验验证	YTech
小人脸场景美化降级	降低GPU消耗	小	优化后Combo实验验证	YTech
CPU/网络优化	降低CPU/网络消耗	小	优化后Combo实验验证	直播业务
RTC硬编码/解码	降低CPU/GPU消耗	中	优化后Combo实验验证	音视频
有问题的特效优化	降低GPU消耗	小	优化后Combo实验验证	直播业务+YTech
UI渲染优化	降低CPU/GPU/Display消耗	小	优化后Combo实验验证	直播业务
极限发热下，业务降级	全面降低功耗水平	大	功能降级，单独实验	直播业务

1) 功耗原理分析和Profile

→ 2) 渲染优化

3) 极限发热下的热缓解

UI刷新对功耗的影响

UI刷新对GPU的影响因素

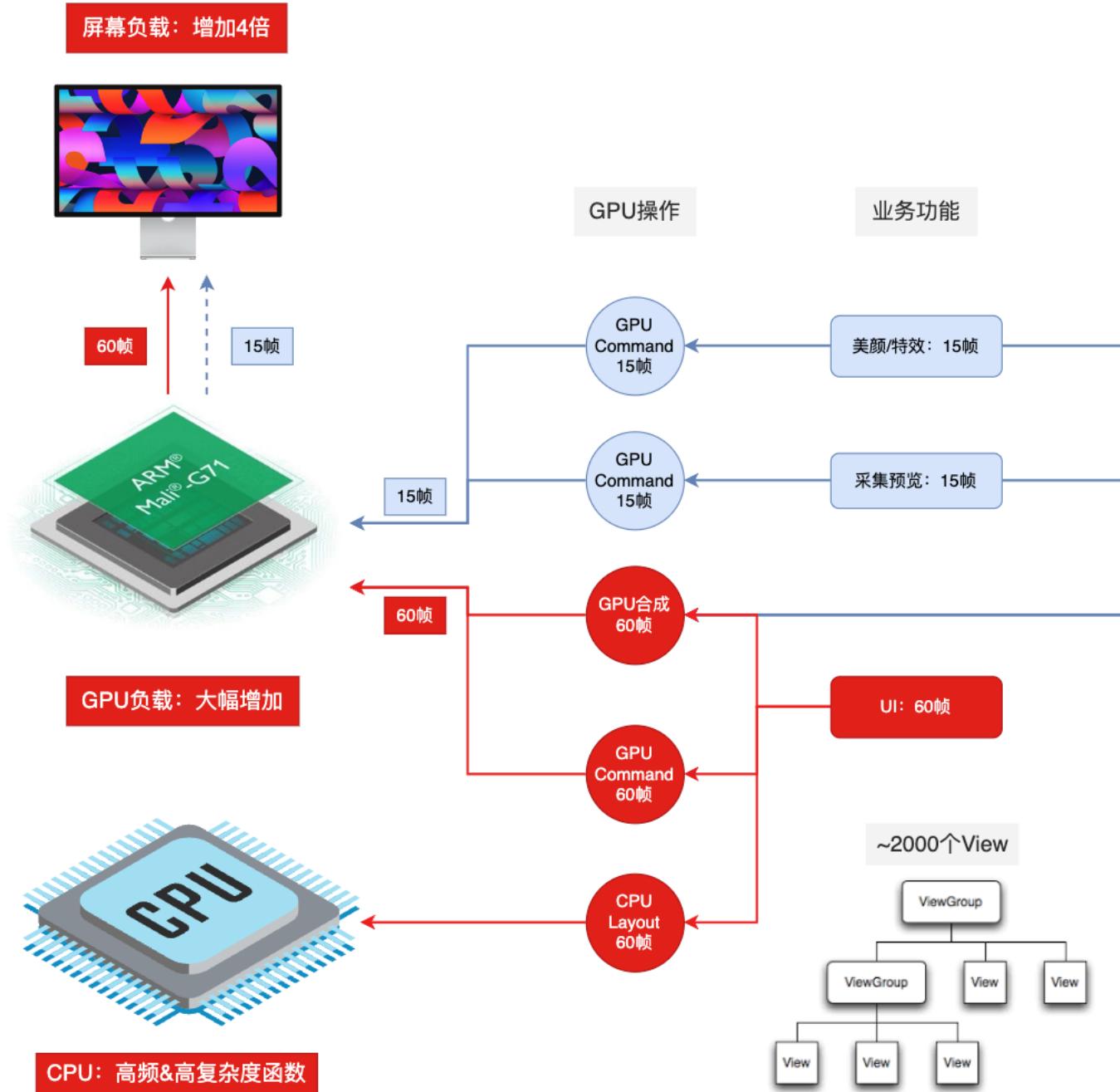
- 刷新率
- 刷新区域
- 层间合成

Oled 屏幕特性

- 支持动态刷新帧率，越接近静态的页面功耗越低
- 黑色区域几乎不耗电 (熄屏展示时钟)

如何定量的分析UI刷新对GPU的影响？

- 通过Demo App, 摸高实验证



UI刷新对功耗影响的定量验证

UI刷新率

刷新率	CPU	GPU	电流
10 FPS	5.73% (+35%)	0.08%	225mA (+18%)
1 FPS	3.68%	0.11%	185mA

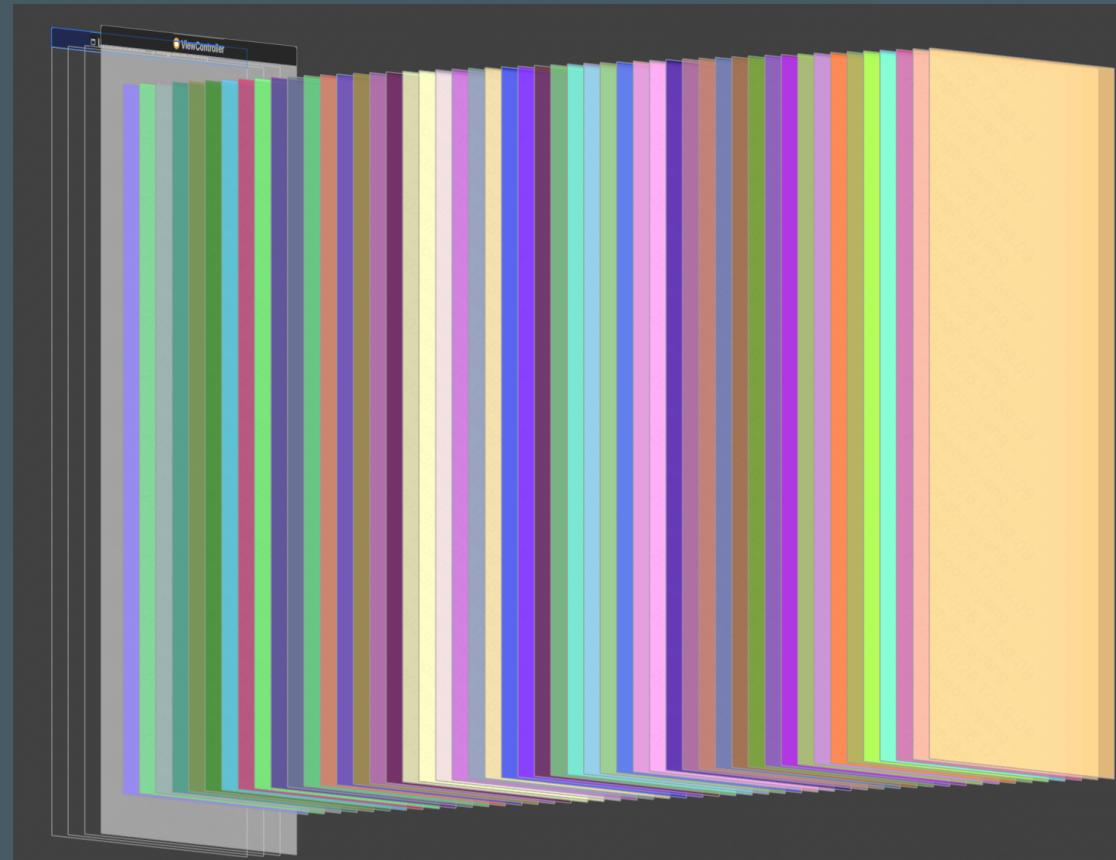
“ UI刷新率对功耗有非常大的影响 ”

层间合成

视图层级	SysCPU	GPU	电流	颜色透明度
1	5.1%	1.23%	233mA	0.7
10	5.01%	4.35% (253%)	232mA	0.7
50	4.88%	1.31%	203mA	1

“ 无Alpha合成的情况下，多层UI基本无消耗； ”

“ Alpha合成对功耗影响非常大（对设计效果影响也比较大） ”





UI刷新对功耗影响的定量验证

UI渲染区域

刷新区域	SysCPU	GPU	电流
刷新一行	5.19%	0%	190mA
全部刷新	5.37% (+3.5%)	0.14% (+14%)	200mA (+5%)

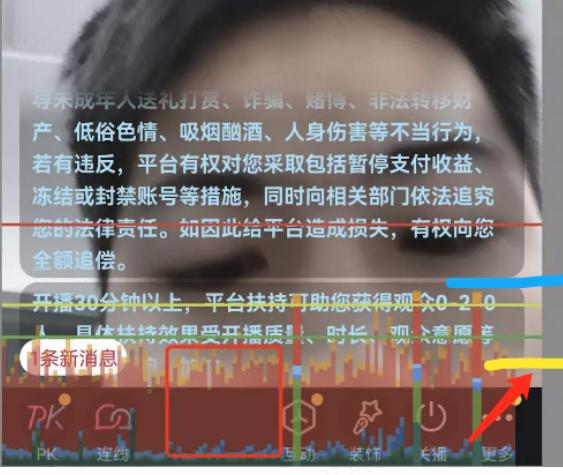
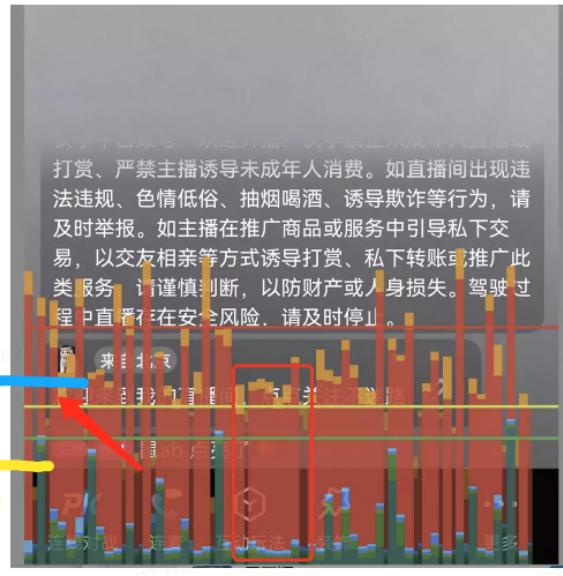
“每帧渲染脏区域不同对功耗也有相对大影响”

渲染问题定位和优化

$$\text{渲染复杂度} = FPS_{\text{刷新频次}} \times FrameComplexity$$

- 和竞品对比，快手渲染频次高，单帧渲染复杂度高，整体渲染功耗高很多
- 单帧的渲染复杂度来源于
 - 渲染命令的复杂程度
 - 单次渲染区域大小
- 单纯降低帧率预估可以有120mW以上的收益

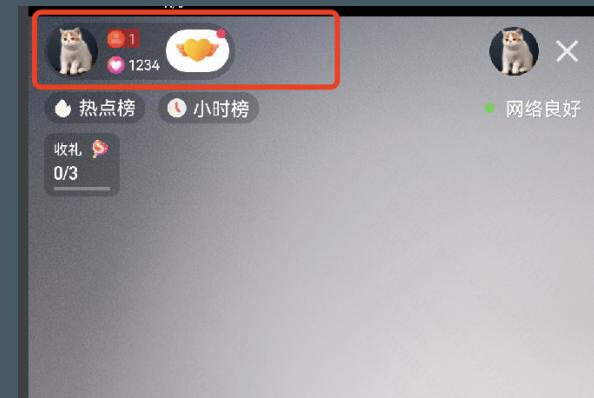
“整体收益空间比较大”

	抖音	快手
FPS	15	60
单帧 渲染 复杂 度		

Oppo 评估数据			
快手播放短视频	60fps	30fps	收益
	312mA	280mA	32mA 120-150mW

渲染问题定位和优化 - 典型渲染问题

- 典型问题: 在线人数/网络状态等导致高频且全局刷新
- 解决方案: 提供OptimizedTextView控件解决一类问题
- 优化后的效果



渲染优化整体策略

业务	线上表现/base组	命中实验
业务降级		
业务 - 热点榜单	<ul style="list-style-type: none">展示为不断轮播的跑马灯	<ul style="list-style-type: none">如果是热点主播，固定展示热点主播如果不是，固定展示热点榜单点击行为正常响应
业务 - 左上角运营挂件	<ul style="list-style-type: none">3秒轮播	<ul style="list-style-type: none">10秒轮播
业务 - 心愿单挂件	<ul style="list-style-type: none">3秒轮播	<ul style="list-style-type: none">10秒轮播
业务 - 右下角挂件	<ul style="list-style-type: none">5秒轮播	<ul style="list-style-type: none">10秒轮播
业务 - 小快提示	<ul style="list-style-type: none">用户未打开小快的情况下，默认展示小快引导	<ul style="list-style-type: none">如果用户没有打开小快，默认不展示小快引导
优化		
业务 - 网络状态	每秒刷新一次，导致全局刷新	状态变化时才刷新
业务 - 在线人数/点赞数量	每次刷新都是全局刷新	只有宽度变化才是全局刷新，大部分是内部变化
PK倒计时	导致全局刷新	只刷新倒计时区域
RTC 无内容变化，但是固定频率全局刷新	固定频率全局刷新	内容变化才刷新
聊天室Speaking动画	全局刷新	只刷新View区域
礼物数量/用户名	全局刷新	只刷新固定区域

渲染优化效果

“ 线下测试数据 ”

场景	进程CPU平均值	系统CPU均值	帧率均值 (HZ)	电流(mA)
base组	42.62	82.34	57.65	1011.89
实验组	32.70 (-23%)	69.34 (-19%)	15.19 (-70%)	887.08 (-14%)

“ 线上实验数据 ”

	秀场Base	秀场Exp	PK Base	PK Exp
FPS	51.3	41.3 (-19.5%)	52.3	36.5 (-30.2%)
SoC温度	50.0	49.4 (-1.3%)	56.4	55.0 (-2.4%)

1) 功耗原理分析和Profile

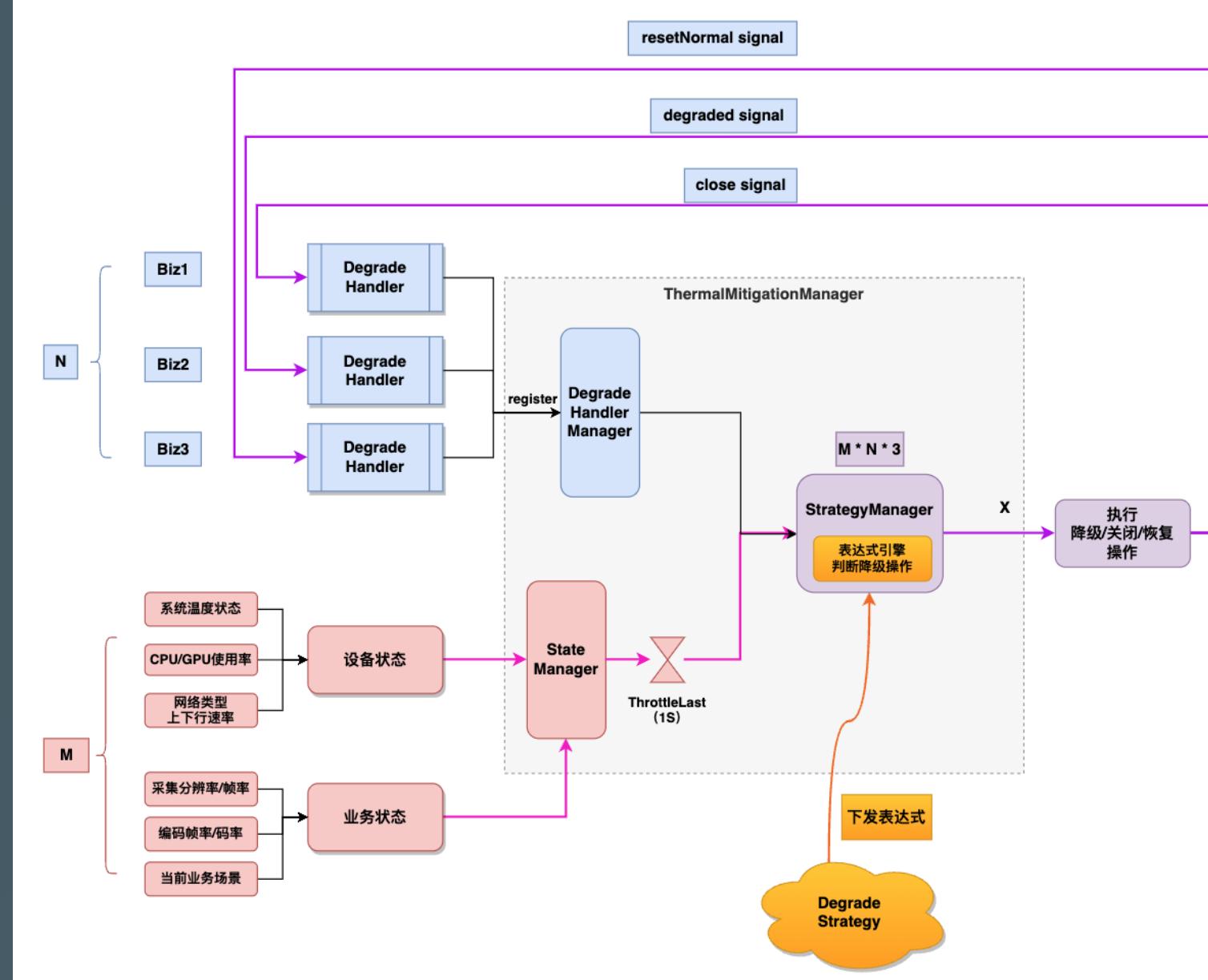
2) 渲染优化

→ 3) 极限发热下的热缓解

功耗调度方案 - 热缓解框架

“核心设计理念”

1. 只设计Normal, Degraded, Close三级降级状态，避免给业务带来过高的复杂度
2. 降级参考信号丰富，会采集基础性能数据，系统发热状态，编码采集帧率等等，可以实现灵活的降级判断标准
3. 采用自定义表达式引擎作为降级配置，降级的可解释性和灵活度都大幅提升

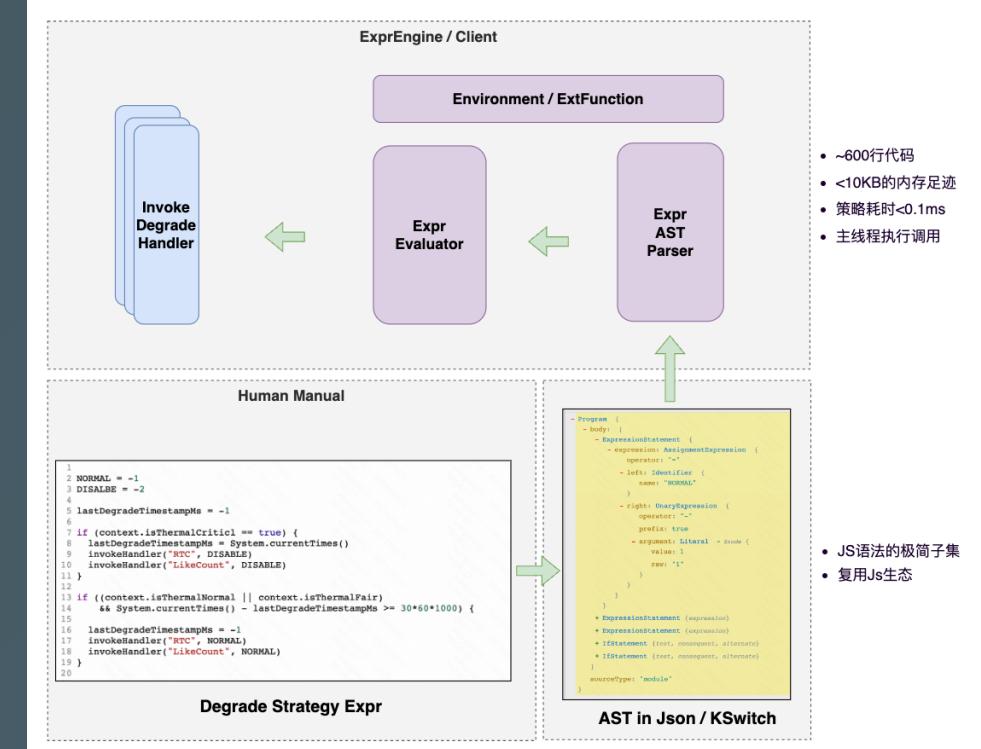


自定义表达式引擎实现灵活/可解释的降级策略下发

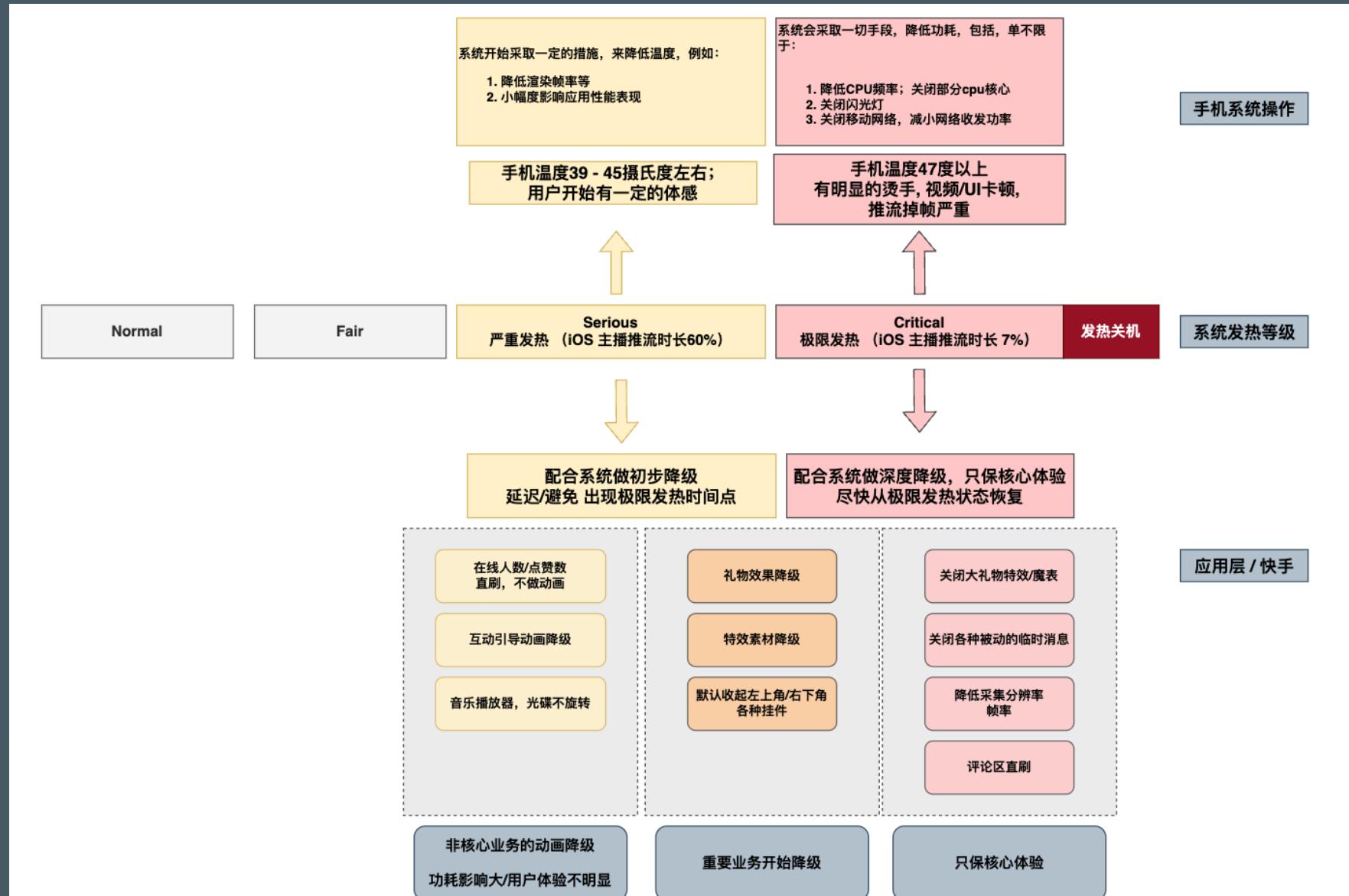
需要考虑的变量个数：

$$z = m_{\text{影响功耗的因素}} \times n_{\text{性能表现}} \times x_{\text{降级业务}} \times y_{\text{降级信号类型}}$$

传统方案		表达式引擎方案
抽象	固定的维度+加权平均值	预埋维度+下发表达式
灵活性	中	高
可解释性	低	高 (只关注少数几个重要维度即可)
方便AB实验	低	高 (可解释性高, AB更方便)
普适性	低 (很难迁移)	高 (只需更改策略, 即可适用于其它降级场景)
配置复杂度	低 (数学公式)	高 (代码逻辑)



热缓解策略 - 业务分级降级，保证核心体验



功耗治理 - 主播教育

“ 观察 ”

- 夏天在户外，普通刷视频手机都会严重烫手，进而关机（冰块比啥优化都好使）
- 户外主播 / 春夏秋季盛典活动等，不可避免的存在高温环境下直播

“ 方案 ”

- 大型活动主播开播保障白皮书
 - 风扇/冰块降温
 - 避免阳光直射
 - 不要佩戴手机壳
- 产品功能提示
 - 出现极限发热，主动提示用户散热技巧

