



Unisoc Confidential For hiar

性能调试指导手册

文档版本 V1.1
发布日期 2021-02-07

版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For hiar

紫光展锐（上海）科技有限公司



前言

概述

本文档从用户关注的性能指标，影响性能指标的因素，工程师分析性能问题的工具和方法等方面进行了全面介绍。

读者对象


本文档主要适用于对 Android 11.0 以及 Android10.0 性能进行调试相关的研发及测试人员。

缩略语

缩略语	英文全名	中文解释
IPA	Intelligent Power Allocation	智能功率分配
AMS	Activity Manager Service	Activity 管理服务
CPU	Central Processing Unit	中央处理器
GPU	Graphics Processing Unit	图形处理器
DDR	Double Data Rate	双倍速率
DVFS	Dynamic Voltage and Frequency Scaling	动态电压频率调整
LMKD	Low Memory Killer Daemon	低内存终止守护进程
PSI	Pressure Stall Info	内核压力失速信息
LMK	Low Memory Killer	低内存终止进程
LMFS	Low Memory Force Stop	低内存强制终止进程

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0	2020-10-28	第一次正式发布
V1.1	2021-02-07	增加 OS 版本 Android 11.0

关键字

CPU、GPU、性能。

Unisoc Confidential For hiar

目 录

1 影响因素	1
1.1 CPU	1
1.1.1 CPU 硬性指标	1
1.1.2 Power Hint	1
1.1.3 IPA & Thermal	2
1.1.4 CPU Set	2
1.2 GPU	2
1.3 多后台	3
1.3.1 LMKD + VMPRESSURE	3
1.3.2 LMK 参数	3
1.3.3 LMFS	3
1.3.4 进程关联	4
1.4 内存	4
1.4.1 内存主频	4
1.4.2 内存大小	4
1.4.3 swappiness	5
1.4.4 ZRAM	5
1.4.5 低内存设备配置	5
1.4.6 PinnerServer	5
1.4.7 LMKD 介绍	6
1.5 编译	7
1.6 IO	7
1.7 Kernel 配置	8
1.8 动画&截屏	9
1.9 App Compactor	9
1.10 PSI	9
1.11 Fastkill	9
2 常见性能问题分析	11
2.1 系统整体卡顿	11
2.2 开关机问题	11
2.2.1 开机	11
2.2.2 关机	12
2.3 应用启动	12
2.4 流畅性问题	13

图目录

图 1-1 原理框图	6
图 2-1 关机流程	12

Unisoc Confidential For hiar

1 影响因素

常见的性能指标包含以下方面：

- 执行时间：开关机时间、应用冷/热启动时间、应用退出、数据复制/删除、特定操作。
- 流畅性：动画、页面滑动、游戏帧率。
- 耗电：性能和功耗统筹考虑。
- Antutu 跑分和 google 性能指标。

影响系统性能的因素有很多，主要有 CPU、GPU、DDR/EMMC 等硬件规格，也有多后台策略、编译配置、软件代码等软件因素。

1.1 CPU

1.1.1 CPU 硬性指标

CPU 型号、CPU 主频和 CPU Cache 很大程度上能决定系统的基础性能，在设定项目性能目标，选取对比机时需要提前考虑。

1.1.2 Power Hint

进入某个场景时，通过相关接口通知 Power Hint 进行相应处理，如调整 CPU、DDR 的变频策略等，以达到提升性能或省电的目的，提升用户体验。

触发 Power Hint 有三种方法：

- 通过 PowerManagerService 的 Binder 接口 powerHint()
- 通过 PowerManager 中扩展接口 PrfmLock, Deprecated
- 通过 PowerHALManager, Power HIDL HAL 接口

Power Hint 场景定义参见 hardware/interfaces/power/1.0/types.hal，常见场景有 PowerHint.LAUNCH、PowerHint.INTERACTION。

APP 启动时，AMS(ActivityManagerService)通过 Power Hint 触发系统提升 CPU、DDR 频率，加快 APP 的启动。

调试时若需要关闭 Power Hint，可以使用如下命令：

```
# setprop persist.sys.power.hint 0
# reboot
```


1.1.3 IPA & Thermal

IPA (Intelligent Power Allocation, 智能功率分配), 根据性能需求与温度情况, 进行功率/性能智能分配。在高配置的芯片平台上容易触发温控, 若触发温控, 导致 CPU 自动降频同时拔除某几个 CPU, 此时可在 Kernel log 中查询关键字 “target_freq” 进行确认, 示例如下:

```
620[03-19 10:51:42.380] <6>[ 1275.581924] c0 13195 cpu4 temp:70125 target_freq:624000 online:0 qos_cpu:0 power:0
621[03-19 10:51:42.680] <6>[ 1275.885395] c0 13195 cpu0 temp:71000 target_freq:2028000 online:4 qos_cpu:4 power:2138
626[03-19 10:51:42.870] <6>[ 1276.081762] c0 13195 cpu4 temp:71750 target_freq:624000 online:0 qos_cpu:0 power:0
629[03-19 10:51:43.174] <6>[ 1276.382496] c0 13195 cpu0 temp:74125 target_freq:1560000 online:4 qos_cpu:4 power:1781
632[03-19 10:51:44.071] <6>[ 1277.081860] c0 13195 cpu4 temp:75250 target_freq:624000 online:0 qos_cpu:0 power:0
636[03-19 10:51:44.190] <6>[ 1277.391903] c0 13195 cpu0 temp:73750 target_freq:2028000 online:4 qos_cpu:4 power:1824
640[03-19 10:51:44.390] <6>[ 1277.593074] c0 13195 cpu4 temp:74875 target_freq:624000 online:0 qos_cpu:0 power:0
641[03-19 10:51:44.690] <6>[ 1277.891888] c0 13195 cpu0 temp:74500 target_freq:1560000 online:4 qos_cpu:4 power:1738
642[03-19 10:51:44.885] <6>[ 1278.092586] c0 13195 cpu4 temp:74875 target_freq:624000 online:0 qos_cpu:0 power:0
646[03-19 10:51:45.190] <6>[ 1278.392602] c0 13195 cpu0 temp:75625 target_freq:1248000 online:4 qos_cpu:4 power:1643
649[03-19 10:51:45.390] <6>[ 1278.592253] c0 13195 cpu4 temp:74500 target_freq:624000 online:0 qos_cpu:0 power:0
651[03-19 10:51:45.681] <6>[ 1278.892278] c0 13195 cpu0 temp:75625 target_freq:1560000 online:4 qos_cpu:4 power:1642
654[03-19 10:51:45.890] <6>[ 1279.091931] c0 13195 cpu4 temp:75250 target_freq:624000 online:0 qos_cpu:0 power:0
657[03-19 10:51:46.190] <6>[ 1279.391853] c0 13195 cpu0 temp:74875 target_freq:1560000 online:4 qos_cpu:4 power:1690
674[03-19 10:51:46.460] <6>[ 1279.592307] c0 13195 cpu4 temp:75625 target_freq:624000 online:0 qos_cpu:0 power:0
679[03-19 10:51:46.691] <6>[ 1279.895833] c0 13195 cpu0 temp:75250 target_freq:1560000 online:4 qos_cpu:4 power:1659
681[03-19 10:51:46.890] <6>[ 1280.091791] c0 10996 cpu4 temp:74875 target_freq:624000 online:0 qos_cpu:0 power:0
685[03-19 10:51:47.690] <6>[ 1280.892122] c0 13195 cpu0 temp:75625 target_freq:1560000 online:4 qos_cpu:4 power:1633
687[03-19 10:51:48.090] <6>[ 1281.091927] c0 13195 cpu4 temp:76125 target_freq:624000 online:0 qos_cpu:0 power:0
```

调试时关闭 Thermal 的方法:

【打开拨号盘】→【输入*##83781#*##】→【Debug&LOG】→【Thermal】→【Thermal Switch 输入&IPA】

1.1.4 CPU Set

CPU Set 用于控制前台或者后台应用使用 CPU 的 id, 一般在有大小核的项目上使用, 能有效在性能与功耗之间进行平衡。

例如设置 write/dev/cpuset/background/cpus 0-3 表示后台应用只能使用 cpu0 至 cpu3。

可以在 init.rc(/system/core/rootdir/)中增加以下内容配置 cpuset。

on init

```
# Use all CPUs during bootup (should be changed when system is up)
copy /sys/devices/system/cpu/possible /dev/cpuset/foreground/cpus
copy /sys/devices/system/cpu/possible /dev/cpuset/top-app/cpus
copy /sys/devices/system/cpu/possible /dev/cpuset/background/cpus
copy /sys/devices/system/cpu/possible /dev/cpuset/system-background/cpus
write /dev/cpuset/foreground/boost/cpus 0
```

1.2 GPU

如果调试时无法确定 GPU 主频是否影响了性能, 可以手工把 GPU 固定到最高频率, GPU 主频所在目录如下:

/sys/module/mali/drivers/platform:mali/60000000.gpu/devfreq/60000000.gpu

- available_frequencies: 可以支持的所有频率, 如 600000000


```
echo 600000000 > min_freq
echo 600000000 > max_freq
```

- cur_freq: GPU 当前频率

imagination GPU 与 mali 平台的设备节点不同，调试频率的节点在类似此目录：

```
/sys/module/pvrsrvkm/drivers/platform:pvrsrvkm/600000000.gpu/devfreq/600000000.gpu
```

同时可以通过 cat 如下节点，查看 GPU 负载来判断某些 GPU 不支持的卡顿场景。

```
# cat /d/pvr/status
GPU Utilisation: 2%
```

1.3 多后台

确保前台流畅度的同时增强后台容纳能力是低内存优化的主要目标。一方面内存中要尽可能维持较多的后台，增强后台容纳能力，尽最大可能保证应用交互无内存异常现象，如无法启动、闪退，尽可能多的让应用热启动而不是冷启动增强用户体验；另一方面在增加多后台的同时要保证前台应用的流畅度。增强多后台能力：

- 对系统内存中不可回收部分进行最大程度裁减以扩大应用可用内存空间，例如裁剪 reserved、mlock、常驻进程、高优先级进程。
- 采用高比率的内存压缩技术增加实际可用内存空间，例如压缩匿名页的 ZRAM。

增强多后台下的前台流畅度，按 CPU、RAM、IO 紧张程度，内存是最为首要的因素。

在内存耗尽的情况下，前台应用的启动和应用内各个功能的操作由于没有内存，内存分配选择慢速分配从而导致 block，D 状态(不可中断的睡眠状态)大幅增加，前台出现卡顿、黑白屏、无响应等，这种状态下的内存回收对 CPU 和 IO 的消耗极大，最终导致系统卡死，因此需要在系统卡死之前提前进行内存回收。

必须按照一定规则控制后台进程的数量，需要 AMS 先判定各应用进程优先级，将优先级最低的进程最先杀死，同时压制低内存下的应用进程组内自启动和组间关联自启，避免低内存下的内存压力反复。

1.3.1 LMKD + VMPRESSURE

LMKD: 收到 VMPRESSURE 通知后按照当前内存压力值和水位值进行 kill 相关进程。

VMPRESSURE: 当前已扫描的内存中不可回收内存的比例: $pressure = [1 - (reclaimed/scanned)] * 100$, scanned 表示扫描的 memory page 个数，reclaim 表示当前扫描的内存页数中可回收的 memory page 个数。

1.3.2 LMK 参数

LMK 参数 sys.lmk.minfree 和 sys.lmk.adj 等目前通过动态调整，参见如下代码位置：
frameworks/base/services/core/java/com/android/server/am/processlist.java

1.3.3 LMFS

LMFS: 低内存情况下，在 LMK 和 LMKD 中被杀的进程，AMS 收到应用进程被杀的通知后将应用进程组 force-stop，与防应用后台自启动功能配合使用，避免出现低内存状态下应用进程组内自启动和进程组间关联自启动导致内存状态进一步恶化。

目前判断 LMFS 被杀需要过滤掉以下情况：

在前台、系统 app、cts、输入法、白名单、wakelock。

白名单的策略在资源文件 low_memory_killer_tracker_whitelist 中进行定义，一般设置微信、qq 等常用重要的 APK 为白名单，不让进行 LMFS。

1.3.4 进程关联

自启动控制：防止第三方应用进程组后台自启动，一个第三方应用进程组唤醒另一个第三方进程组，避免在低内存状态下使内存更为紧张。

1.4 内存

1.4.1 内存主频

可以通过以下方法查询 DDR 运行频率，设置 DDR 最高频率：

```
echo 0 > /sys/class/devfreq/scene-frequency/sprd_governor/auto_dfs_on_off
echo 933 > /sys/class/devfreq/scene-frequency/sprd_governor/scaling_force_ddr_freq //固定ddr频点
cat /sys/class/devfreq/scene-frequency/sprd_governor/ddrinfo_cur_freq //查看当前ddr频点
cat /sys/class/devfreq/scene-frequency/sprd_governor/ddrinfo_freq_table //查看支持的ddr频点
```

1.4.2 内存大小

Memtotal 是 Android 部分可用内存，即硬件总内存除去 cp 侧设备驱动独占的内存。对于 512MB 的项目，该部分内存至关重要。通过 cat /proc/meminfo 可以查询到 Memtotal 和其他内存信息。

分析 memory reserved 时，可以通过命令 cat /sys/kernel/debug/memblock/reserved 查看，查看结果如下：

```
sp7731e_1h10:/ # cat /sys/kernel/debug/memblock/reserved
0: 0x80004000..0x80007fff
1: 0x80008280..0x80bea623
2: 0x85400000..0x85435128
3: 0x85500000..0x856c1937
4: 0x87800000..0x879fffff
5: 0x88000000..0x8845a7ff
6: 0x89600000..0x8b8fffff
7: 0x94000000..0x95ffffff
8: 0x9e524000..0x9fdfffff
9: 0xaf6c0000..0xaf7b7fff
10: 0xaf7b8800..0xaf7b883b
```

其中前一系列代表起始物理地址，后一系列代表结束地址，两者差值即为占用空间。

Reserved 的配置在对应的 dts 与 dtsi 文件中设置，如 SC7731E 上 0x89600000 代表的是 modem 的起始地址。其中 0x80008280 较为特殊，其代表的是 Kernel 代码段的大小，该行的大小表示 Kernel image 占用的内存大小。用户 Kernel 的配置以及驱动的配置，可能对 Reserve 造成影响，需要重点关注。

此外，影响用户可用内存的一个重要因素就是常驻内存的大小，由于一些应用的特性要求，需要常驻内存。但是对于低内存配置项目，不能有太多应用设置为常驻内存。可以通过 dumsys meminfo 查看：

```

67,656K: Persistent
47,833K: com.android.systemui (pid 897)
11,260K: com.android.phone (pid 849)
4,318K: com.android.se (pid 1433)
4,245K: com.android.modemnotifier (pid 1448)
25,431K: Foreground

```

1.4.3 swappiness

swappiness 的值在 init.ram.rc (init.ram.gms.rc) 中定义:

```
write /proc/sys/vm/swappiness 150
```

最大值为 200, 如果是 100, 表示匿名页和 filecache 有同样的优先级。

```
anon_prio = swappiness;
file_prio = 200 - anon_prio;
```

调试时可以通过以下节点修改:

```
/proc/sys/vm/swappiness
```

说明

此参数经过展锐测试, 不建议修改。

1.4.4 ZRAM

ZRAM 大小可以通过以下代码进行修改, 其中 90%是物理内存的比率:

```
device/sprd/./common/fstab.enableswap_go zramsize=90%
```

说明

此参数经过展锐测试, 不建议修改。

1.4.5 低内存设备配置

关闭在低内存设备上表现非常差的某些内存密集型功能: ro.config.low_ram=true。

1.4.6 PinnerServer

为了提高应用启动速度, SC7731E 的 PinnerServer 上将部分重点服务启动后放在内存中。以 Pike21h10 项目为例:

```
Sprdroidq_trunk/device/sprd/pike2/sp7731e_1h10/overlay/frameworks/base/core/res/res/values/config.xml
```

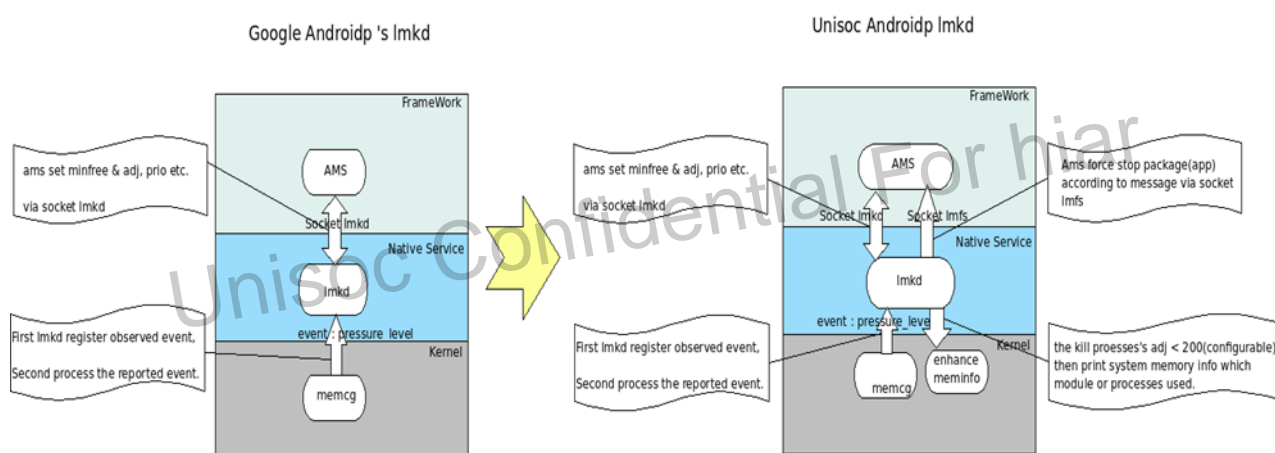
需要核对该部分做出的调整是否必要。

```
<string-array translatable="false" name="config_defaultPinnerServiceFiles">
  <item>"/system/lib/libjavacrypto.so"</item>
  <item>"/vendor/lib/hw/gralloc.sp7731e.so"</item>
  <item>"/system/lib/libhidltransport.so"</item>
  <item>"/system/framework/arm/boot-core-libart.oat"</item>
  <item>"/system/framework/arm/boot-conscrypt.oat"</item>
  <item>"/system/framework/arm/boot-core-libart.vdex"</item>
  <item>"/system/framework/arm/boot-ext.vdex"</item>
  <item>"/system/framework/arm/boot.vdex"</item>
  <item>"/system/framework/arm/boot-framework.vdex"</item>
  <item>"/system/priv-app/SprdContacts/oat/arm/SprdContacts.odex"</item>
  <item>"/system/priv-app/SprdContacts/oat/arm/SprdContacts.vdex"</item>
  <item>"/system/priv-app/SprdContactsProvider/oat/arm/SprdContactsProvider.odex"</item>
  <item>"/system/priv-app/SprdContactsProvider/oat/arm/SprdContactsProvider.vdex"</item>
</string-array>
```

1.4.7 LMKD 介绍

LMK(Low Memory Killer)功能由 Framework 的 AMS 部分，Native 层的 LMKD 部分，Linux 内核的 vmpressure 或 PSI 或 lowmemorykiller 模块这三大组成部分组成。Unisoc 基于以上模块又扩展了 LMFS 和 enhance memory information 两个模块，原理框图如图 1-1 所示。

图 1-1 原理框图



各子模块的作用简介如下：

- **AMS**
 - 计算 LMKD 杀进程用的 moomAdj 数组以及对应的 minfree 水位值数组。
 - 动态调整进程的实时 oom_adj 值：根据进程的状态，如所包含组件状态等信息，自动计算并调整相应进程的 adj 值，来动态更新其重要程度。
- **LMKD**：设置 AMS 传输下来的 adj，minfree 和 prioperty，在 psi mode 和 vmpressure mode 下 kill 进程。
- **Vmpressure**：通常和 memcg 一起使用，感知内存分配时的压力并上报 LMKD。
- **PSI**：感知系统 stall 时的压力并上报 LMKD。
- **LMFS**：Android 进程被 kill 后由 LMFS 上报 AMS，根据白名单决定 force stop 是否含被杀进程的 package

- enhance meminfo: adj 0（在 psi mode 和 vmpressure mode 下可设定）的进程被杀时会打印系统的内存信息。

1.5 编译

DEXOPT

dex2oat(speed, speed-profile, quicken,)

1GB 内存及以上配置的项目内存不是特别紧张，可以将部分预编译选项设置为 speed 模式。1GB 项目需要确保使用如下 mk:

build/make/target/product/go_defaults_common_speed_compile.mk

对于系统软件有编译选项调整，在对应的 board mk 中，通过白名单的方式进行添加。

以 sp7731e_1h10_native.mk 代码为例:

PRODUCT_DEXPREOPT_SPEED_APPS+= \

SprdContacts\

SprdContactsProvider\

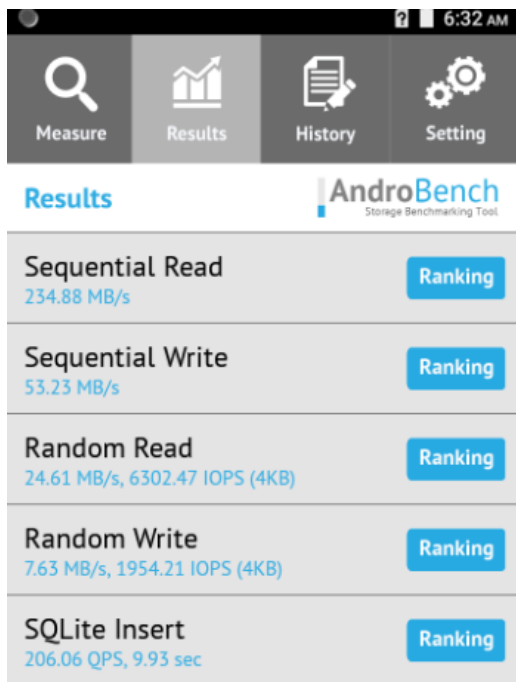
```
<string-array translatable="false" name="config_defaultPinnerServiceFiles">
  <item>"/system/lib/libjavacrypto.so"</item>
  <item>"/vendor/lib/hw/gralloc.sp7731e.so"</item>
  <item>"/system/lib/libhidltransport.so"</item>
  <item>"/system/framework/arm/boot-core-libart.oat"</item>
  <item>"/system/framework/arm/boot-conscrypt.oat"</item>
  <item>"/system/framework/arm/boot-core-libart.vdex"</item>
  <item>"/system/framework/arm/boot-ext.vdex"</item>
  <item>"/system/framework/arm/boot.vdex"</item>
  <item>"/system/framework/arm/boot-framework.vdex"</item>
  <item>"/system/priv-app/SprdContacts/oat/arm/SprdContacts.odex"</item>
  <item>"/system/priv-app/SprdContacts/oat/arm/SprdContacts.vdex"</item>
  <item>"/system/priv-app/SprdContactsProvider/oat/arm/SprdContactsProvider.odex"</item>
  <item>"/system/priv-app/SprdContactsProvider/oat/arm/SprdContactsProvider.vdex"</item>
</string-array>
```

说明

添加 speed 编译后，会增加对内存的消耗。

1.6 IO

EMMC 的读写性能对开机速度，应用的启动有较大影响，一般通过 AndroBench apk 与参考机对比 EMMC 读写性能。AndroBench 对 EMMC 的读写性能测试如下:



通过如下命令可以确认 EMMC 的 SDIO 主频，一般需要在 HS200 的水准。如有明显问题，需要 EMMC 模块相关人员参与分析。

```
cat /sys/kernel/debug/mmc0/clock
```

```
sp7731e_1h10:/ # cat /sys/kernel/debug/mmc0/clock
200000000
```

1.7 Kernel 配置

Kernel 配置对性能影响较大，并且 google 对 Kernel 的配置存在要求，因此建议除驱动配置外的其他配置尽量与平台保持一致。在 check 时请勿开启多余 debug 选项。

由于平台代码有 usrdiff 文件协助 defconfig 管理 Kernel 配置，所以直接读取对应机器上的.config 文件与参考机进行对比更为准确。输入如下命令：

```
adb pull /proc/config.gz
```

```
C:\Users\qiao.wang\Desktop>adb pull /proc/config.gz
515 KB/s (24707 bytes in 0.046s)
C:\Users\qiao.wang\Desktop>
```

1.8 动画&截屏

为了排除动画和截屏对应用启动等场景的性能测试影响，可以关闭动画和截屏。

动画关闭方法：

【手机菜单】→【设置】→【系统】→【开发者选项】→【窗口动画缩放 过渡动画缩放 动画程序时长缩放】

截屏关闭方法：

【打开拨号盘】→【输入*##83781#*##】→【Debug&LOG】→【性能工具】→【Starting Window】

1.9 App Compactor

App Compactor 的具体逻辑如下：

1. Kernel 合入一组 patch，允许用户空间对特定进程的内存进行回收。
2. AMS 主动压缩进程来提升后台进程的并发性。

功能开关：

AppCompactor.java中的Boolean DEFAULT_USE_COMPACTOR = false;

此功能由于会影响热启动的时间，默认处于关闭状态。如果不关注应用热启动，建议在太内存项目上开启，优化冷启动时间。

1.10 PSI

Android 11.0 和 Android 10.0 支持新的 LMKD 模式，使用 PSI 监视器来检测内存压力。内核中的 PSI 测量由于内存不足而导致任务延迟。这些延迟会直接影响用户体验，因此延迟代表了内存压力严重性的指标。

由于 vmpressure 信号（由内核生成，用于内存压力检测并由 LMKD 使用）通常包含大量误报，因此 LMKD 必须执行过滤以确定内存是否真的有压力。这会导致不必要的 LMKD 唤醒并使用额外的计算资源。使用 PSI 监视器可以更精确地检测内存压力，最大限度地减少额外的计算资源占用。

1.11 Fastkill

为了在启动某些大型应用时（特别是对分配连续内存段有特殊需求的应用）有充足的内存，以达到加快启动速度的目的，在 Android 11.0 和 Android 10.0 上实现了启动前的快杀功能。原理是通过上层在启动前下发快杀命令到 LMKD，LMKD 选择杀掉一些进程，以达到释放内存优化大应用启动时间的效果。

配置文件：sprd_performance_config.xml，例如：

```
<item name="FastKill">
    <string-array name="pkgName">
        <string>com.android.camera2</string>
    </string-array>
</item>
```



```
<hintType>null</hintType>
```

```
</item>
```

Unisoc Confidential For hiar

2 常见性能问题分析

2.1 系统整体卡顿

Userdebug 版本系统的整体性能受编译方式、调试功能、log 输出开关等因素影响，所以不能作为性能问题的测试版本。如果系统整体卡顿可以检查以下几个方面：

- 检查 loglevel

如果 loglevel 设置过高，会影响系统整体性能。

cat proc/sys/kernel/printk, 输出 14 1 7 为正常，第一个 1 代表的 Kernel 的 log 等级。

- 检查内存参数是否正常

cat proc/meminfo, 观察 MemTotal 的值，以及其中的其它参数。

- 排查 Kernelconfig 是否与平台版本有较大差异

adb pull /proc/config.gz。

- 检查 ZRAM 是否开启

cat proc/meminfo, 观察 swapTotal 和 swapFree, 为 0 则未开启。

- 检查 DDR 频点

cat /sys/class/devfreq/scene-frequency/sprd_governor/ddrinfo_freq_table。

- 检查内部 app 是否异常

dumsys meminfo, 查看是否有异常内存，或者是否有过多的常驻应用。

- 通过 getprop 获取系统属性，对比平台是否有特别属性控制错误。
- 使用跑分软件跑分，排查特定跑分原因，如 IO 跑分、GPU 游戏跑分等等。

2.2 开机问题

2.2.1 开机

开机分为下载后首次开机、正常开机、恢复出厂设置、BBAT 开机、校准开机等。

正常开机过程分为三大阶段：UBOOT, Kernel, 上层。

上层问题关注 log 中关键字 boot_progress_, 根据对比先找到耗时长过程，然后再具体分析。

开机过程中常见的影响有：

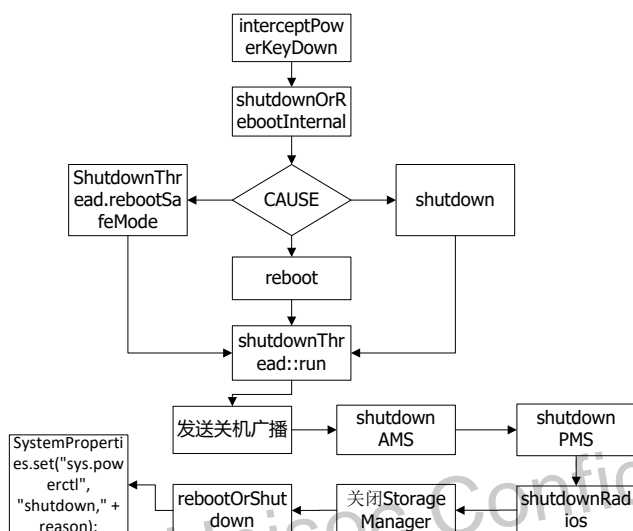
- Kernel mount 时间过长。

- 预置过多应用导致 packagemanager 加载慢。
- 开机过程中做 fs trim 导致加载慢。
- NAND 或者 EMMC 物理原因等等。
- 文件加密。
- 开机中添加了过多的常驻应用或启动过多 service。

2.2.2 关机

关机过程关注关机广播的处理，关机动画等。关机流程如图 2-1 所示。

图2-1 关机流程



2.3 应用启动

应用启动慢常见原因分析：

- 检查动画&截屏对启动的影响。
- Power Hint 是否正常启动，可在 log 中查询 POWERHAL 关键字，根据查询到的信息来判断是否正常启动。
- 通过 systrace 对比
 - 如果是 bindApplication 耗时长，检查加载的 apk、so 等文件大小是否有很大差异，检查 pinnerserver 配置。
 - 如果是 ActivityStart 耗时长，则检查资源文件差异，inflate、measure 过程是否有大的差异。
- 分析 systrace，检查应用启动期间是否有其他耗资源的后台线程在并行运行。
- 对于低内存配置的项目，则检查 LMK 行为和可用内存剩余情况。

2.4 流畅性问题

流畅性问题关注 GPU 的处理性能，通过分析 systrace，查看 dequeueBuffer、eglSwapBuffersWithDamageKHR 等操作 sleeping 时间是否过长。

Unisoc Confidential For hiar