

A33

内存配置说明

Confidential

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2014-07-30		初始版本

Confidential

目 录

A33 1

平板方案内存配置说明.....	1
1. 概述.....	1
1.1. 编写目的.....	1
1.2. 适用范围.....	1
1.3. 相关人员.....	1
2. 术语、缩略语及概念.....	2
2.1. 术语、定义、缩略语.....	2
2.1.1. 预留内存.....	2
2.1.2. vmalloc 区.....	2
2.1.3. ION.....	2
2.1.4. zram.....	2
2.2. 概念阐述.....	2
2.2.1. drop_cache 机制.....	2
2.2.2. lowmemorykiller 机制.....	2
3. KERNEL 相关配置.....	3
3.1. 通用内核配置.....	3
3.1.1. zram 配置.....	3
3.1.2. lowmemorykiller 配置.....	4
3.1.3. drop_caches 节点写使能.....	5
3.2. ION 预留区大小.....	5
3.2.1. 512M 方案.....	5
3.2.2. 1G 方案.....	6
3.3. vmalloc 区大小.....	6
3.3.1. 512M 方案.....	6
3.3.2. 1G 方案.....	6
4. ANDROID 相关配置.....	8
4.1. drop_cache 门限.....	8
2.1.1. 512M 方案.....	8
2.1.2. 1G 方案.....	8
4.2. lowmemorykiller 门限.....	9
2.1.3. 512M 方案.....	9
2.1.4. 1G 方案.....	9
2.2. zram disksize.....	9
2.2.1. 512M 方案.....	9
2.2.2. 1G 方案.....	9
4.3. dalvik heap 参数.....	9
2.2.3. 512M 方案.....	9

2.2.4. 1G 方案.....	10
-------------------	----

Confidential

1. 概述

1.1. 编写目的

介绍平板方案内存配置说明相关知识，供方案定制和开发人员参考。

1.2. 适用范围

适用于 A33 平台；

1.3. 相关人员

本文档的参考人员为 A33 方案定制或开发人员。。

Confidential

2. 术语、缩略语及概念

2.1. 术语、定义、缩略语

2.1.1. 预留内存

指专门用于 camera/gpu/ve 等模块的一块连续物理内存, 不被 linux 伙伴系统管理, 不能用作其他用途.

linux 标准函数不能分配超过 4M 的连续物理内存, 而硬件模块有时需要大于 4M 的连续物理内存. 预留内存就是为了解决这个问题.

2.1.2. vmalloc区

指 linux 内核虚拟地址空间中, 0xFF000000 之前的一段区间, 大小不能超过 976M;

这段区间用于物理内存的动态映射, io 虚拟地址, vmalloc 函数等. 与它相对应的是低端内存区, 即线性映射区;

2.1.3. ION

android 引入的内存管理框架, 在 kernel 实现, 主要用于应用层访问连续物理内存.

2.1.4. zram

即压缩内存机制. 在系统内存紧张时, 将不活动内存进行压缩, 并回写到一块压缩内存区域, 以提高内存利用率.

2.2. 概念阐述

2.2.1. drop_cache机制

指定期对文件系统缓存进行回收的机制. linux 会将空闲内存大量用于 fs 缓存, 这部分内存若不手动回收, 会导致系统空闲内存紧张, 影响效率.

android 原生不会 drop_cache, 我们在 framework 中添加了 drop_cache 机制, 每隔几秒回收一次.

哪些情况才需要回收? 根据系统总可见内存, 当前空闲内存, fs 缓存的大小和比例来定.

2.2.2. lowmemorykiller机制

指在系统空闲内存较少时, 杀死重要性低的应用, 回收内存的机制.

系统定义了若干个进程优先级, 比如前台应用优先级为 0, 后台应用优先级为 9~15, 数值越低, 优先级越大.

在每次进行内存回收时, 选取一个优先级最低, 且占用内存最多的应用杀掉.

内存回收的时机如何定? 根据系统总可见内存, 当前空闲内存和 fs 缓存大小来定.

3. kernel相关配置

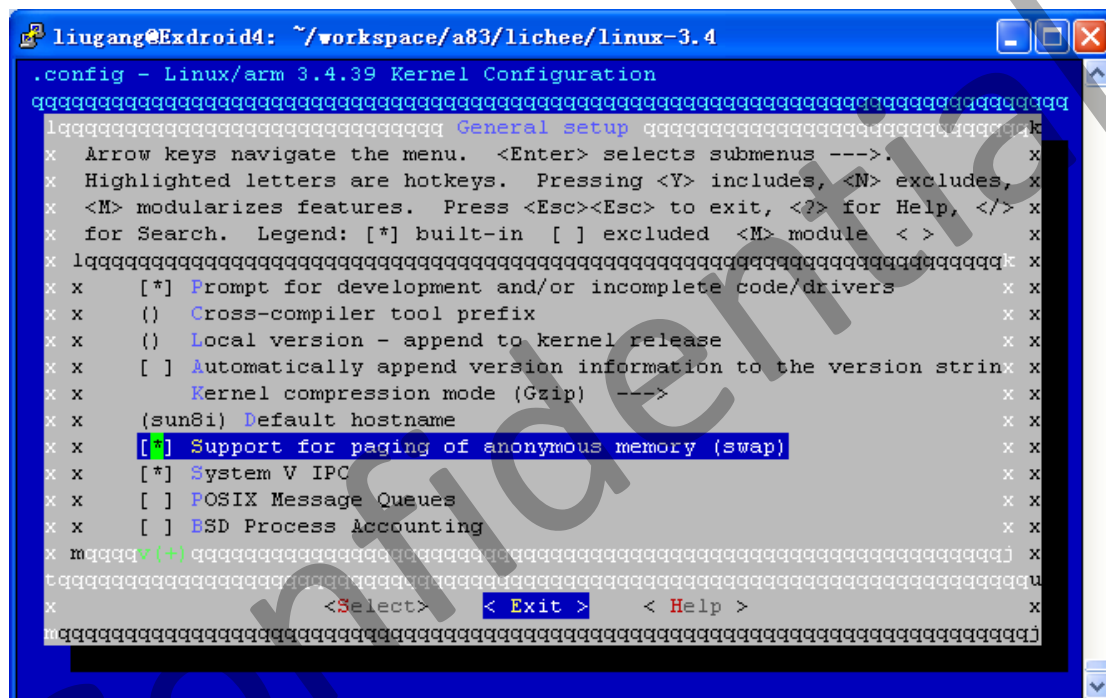
3.1. 通用内核配置

512M 和 1G 通用的内核 menuconfig 配置.

3.1.1. zram配置

(1) CONFIG_SWAP:

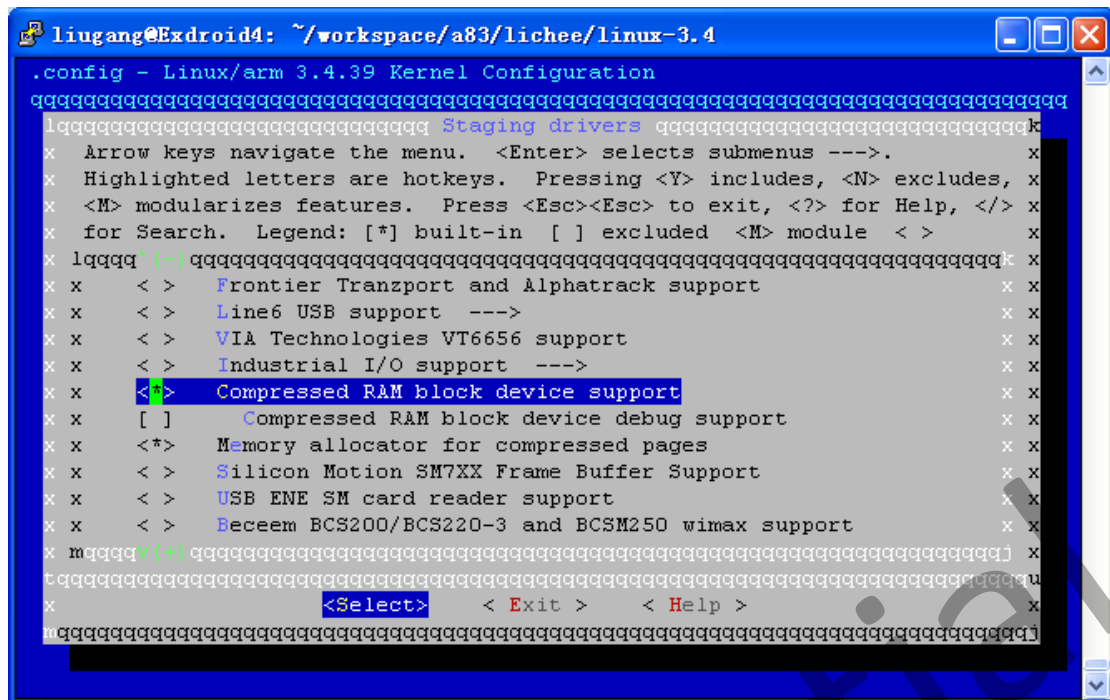
- 1 在 linux3.4 目录下, 输入 `make ARCH=arm menuconfig`
- 2 按照以下选项依次选择:
General setup ---> Support for paging of anonymous memory (swap)



(2) CONFIG_ZRAM

ZRAM 设置方法:

- 1 在 linux3.4 目录下, 输入 `make ARCH=arm menuconfig`
- 2 按照以下选项依次选择:
Device Drivers ---> Staging drivers
---> Compressed RAM block device support
---> Memory allocator for compressed pages



3.1.2. lowmemorykiller配置

lowmemorykiller 设置方法:

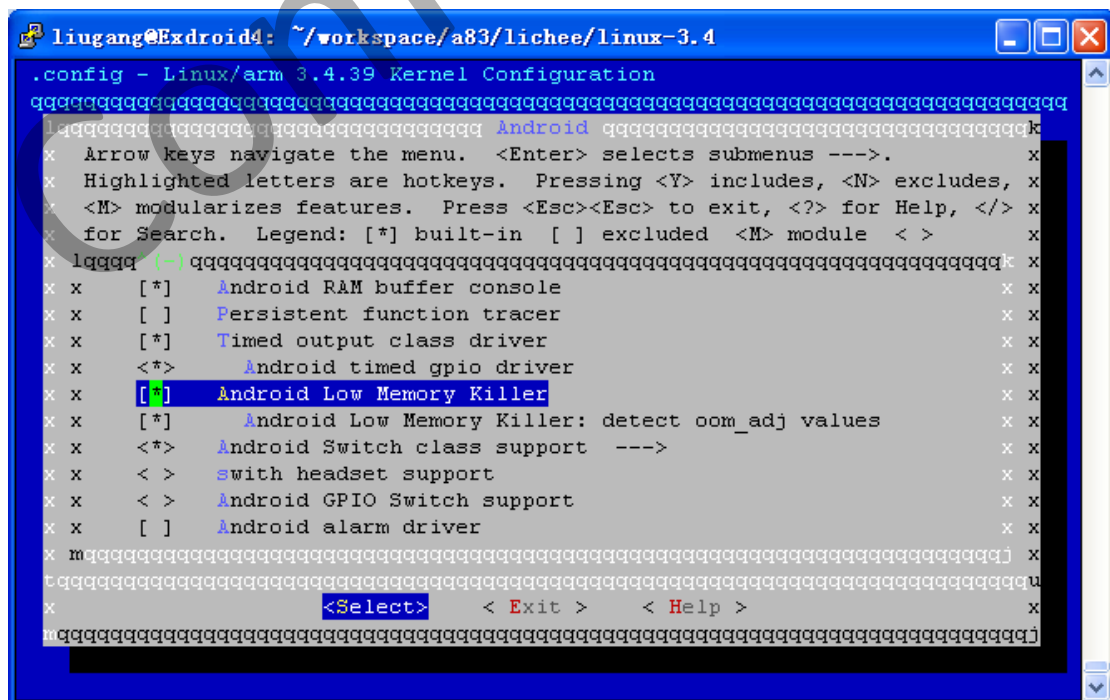
- 1 在 linux3.4 目录下, 输入 make ARCH=arm menuconfig
- 2 按照以下选项依次选择:

Device Drivers ---> Staging drivers

---> Compressed RAM block device support -> Android

---> Android Low Memory Killer

---> Android Low Memory Killer: detect oom_adj values



3.1.3. drop_caches节点写使能

```
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 49f47258..48ba6b0 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -1167,7 +1167,7 @@ static struct ctl_table vm_table[] = {
        .procname      = "drop_caches",
        .data           = &sysctl_drop_caches,
        .maxlen         = sizeof(int),
-       .mode           = 0644,
+       .mode           = 0666,
        .proc_handler    = drop_caches_sysctl_handler,
        .extra1          = &one,
        .extra2          = &three,
```

3.2. ION预留区大小

ION 预留多少合适? 需根据实际需要来定, 不同方案不一样, 估值依据如下:

1. 几个主要模块的内存消耗: GPU, VE, CAMERA, DISPLAY;
2. 规格场景的内存消耗: mirecast, 3D 游戏, CTS/GMS;

预留大小如何确定? 一般分两步:

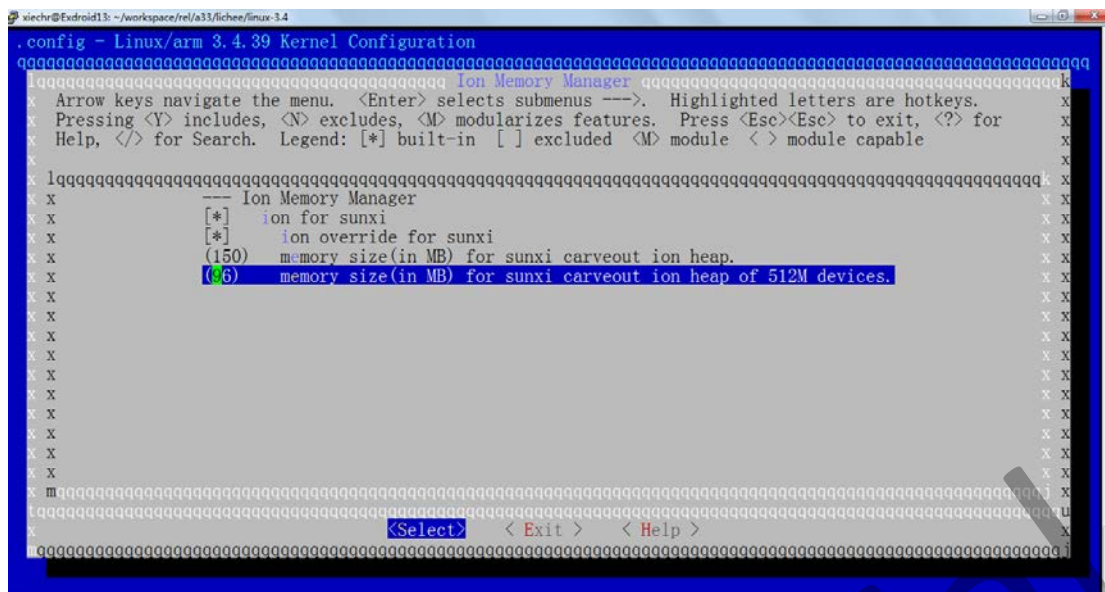
1. 根据规格, **估算**主要场景下, GPU/VE/CAMERA 等各模块消耗的内存, 计算出总和 total;
2. 采用**试凑法**, 将预留大小设为 total, 测试主要场景下, 是否有 ION 申请失败的打印.
3. 若有申请失败情况, 则逐渐加大预留量, 比如每次增加 16M, 直到所有场景压力测试通过为止.

ION 预留内存大小设置方法:

- 1 在 linux3.4 目录下, 输入 make ARCH=arm menuconfig
- 2 按照以下选项依次选择:
Device Drivers ---> Graphics support ---> Ion Memory Manager --->
memory size(in MB) for sunxi carveout ion heap.
memory size(in MB) for sunxi carveout ion heap of 512M device

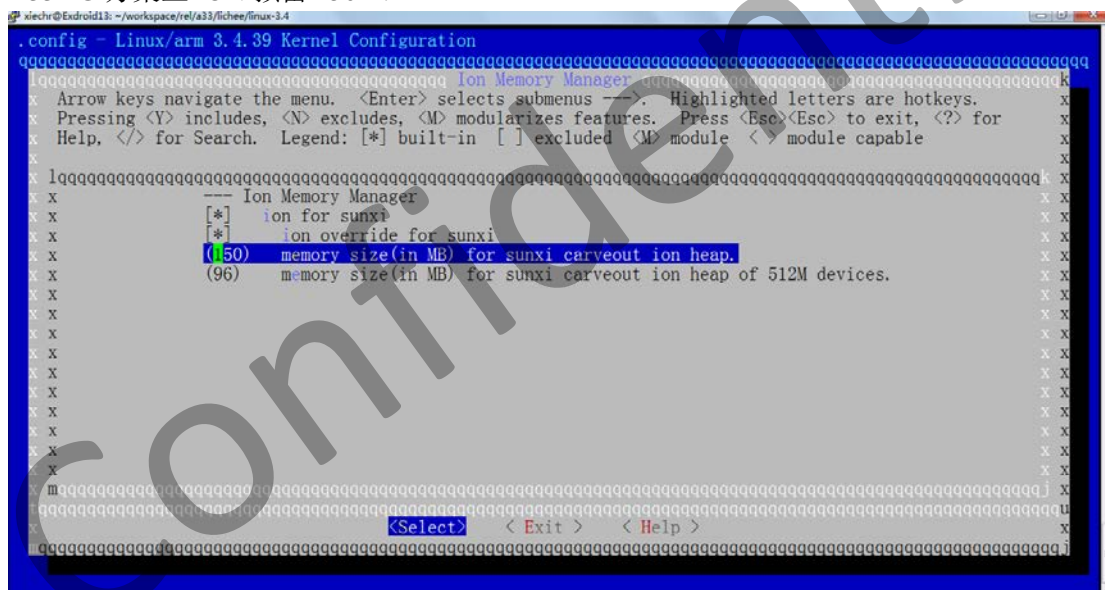
3.2.1. 512M方案

A33 512M 方案上 ION 预留 96M:



3.2.2. 1G方案

A33 1G 方案上 ION 预留 150M:



3.3. vmalloc区大小

3.3.1. 512M方案

512M 方案下, vmalloc 区默认大小为 496M, 从 0xD0000000 到 0xFF000000;

512M 方案使用默认配置即可, 无须改动.

3.3.2. 1G方案

1G 及以上方案中, vmalloc 区默认大小为 248M, 从 0xEF800000 到 0xFF000000;

在 ION 内存消耗较大的场景下, 比如 miracast/3D 游戏/GMS/CTS, 默认 248M 可能不能满足

需求, 因此建议将 vmalloc 区增大到 384M.

方法是在命令行增加"vmalloc=384m", 以下方式任选一:

1. 在方案 env.cfg 文件中, 增加"vmalloc=384m"信息:

lichee\tools\pack\chips\sun8iw5p1\configs\default\env.cfg:

```
setargs_nand=setenv bootargs console=${console} root=${nand_root} vmalloc=384M
init=${init} loglevel=${loglevel} partitions=${partitions}
setargs_mmc=setenv bootargs console=${console} root=${mmc_root} vmalloc=384M
init=${init} loglevel=${loglevel} partitions=${partitions}
```

上述 sun8iw5p1 对应 A33.

2. 在方案中增加"vmalloc=384m":

android\device\softwinner\astar-y3\BoardConfig.mk:

```
BOARD_KERNEL_CMDLINE += vmalloc=384M
```

将上述 astar-y3 替换为实际方案目录.

4. android相关配置

4.1. drop_cache门限

2.1.1. 512M 方案

文件: android\frameworks\base\services\java\com\android\server\am\ActivityManagerService.java

```
private boolean flushCache(int nCache, int nFree){
    boolean nRet = true;
    if((nFree < 20000 && nCache > 70000) || (nFree < 10000 && nCache > 30000)){
        nRet = writeFile("/proc/sys/vm/drop_caches", "3");
    }
    if((nCache - nFree) > 80000) {
        nRet = writeFile("/proc/sys/vm/drop_caches", "1");
    }
    return nRet;
}
```

上述 nFree, nCache 分别表示空闲内存和 fs 缓存, 单位为 KByte;

2.1.2. 1G 方案

1G 方案目前没有使能 drop_cache 机制, 不需要配置.

注: 上述 flushCache 函数仅当 sys.mem.opt 为 true 才会执行, sys.mem.opt 在 system/core/init/property_service.c 里面配置:

```
if (get_dram_size() > 512) {
    property_set("dalvik.vm.heapsize", "384m");
    property_set("dalvik.vm.heapstartsize", "8m");
    property_set("dalvik.vm.heapgrowthlimit", "96m");
    property_set("dalvik.vm.heapminfree", "2m");
    property_set("dalvik.vm.heapmaxfree", "8m");
    property_set("sys.mem.opt", "false");
    property_set("ro.config.low_ram", "false");
} else {
    property_set("dalvik.vm.heapsize", "184m");
    property_set("dalvik.vm.heapstartsize", "5m");
    property_set("dalvik.vm.heapgrowthlimit", "48m");
    property_set("dalvik.vm.heapminfree", "512K");
    property_set("dalvik.vm.heapmaxfree", "2m");
    //aw use
    if(strcmp(buf,"true")){
        property_set("sys.mem.opt", "true");
    }
}
```

```

    }
    property_set("ro.config.low_ram", "true");
}

```

根据上述代码，大于 512M 的方案 sys.mem.opt 被设为 false，因此 flushCache 不起作用。

4.2. lowmemorykiller门限

2.1.3. 512M 方案

文件: android\frameworks\base\services\java\com\android\server\am\ProcessList.java

[A33 使用默认配置，暂不调整。](#)

2.1.4. 1G 方案

[A33 使用默认配置，暂不调整。](#)

2.2. zram disksize

2.2.1. 512M 方案

512M 方案上，zram disksize 设为 384M。

文件: android\device\softwinner\astar-y3\fstab.sun8i

[/dev/block/zram0 none swap defaults zramsize=402653184](#)

2.2.2. 1G 方案

1G 方案上，zram disksize 设为 512M。

文件: android\device\softwinner\astar-y3\fstab.sun8i

[/dev/block/zram0 none swap defaults zramsize=536870912](#)

4.3. dalvik heap参数

dalvik heap 参数会限制进程分配的内存大小，在总内存很多时，值越大则应用响应越快；但在总内存较少时，必须限制该值，以免应用占用过多内存，导致内核运行紧张。

2.2.3. 512M 方案

文件: android\system\core\init\property_service.c

```

static int enable_adaptive_memory(void)
{
    ...
    //for memory > 1024,
    if (get_dram_size() > 512) {
        property_set("dalvik.vm.heapsize", "384m");
        property_set("dalvik.vm.heapstartsize", "8m");
        property_set("dalvik.vm.heapgrowthlimit", "96m");
        property_set("dalvik.vm.heapminfree", "2m");
        property_set("dalvik.vm.heapmaxfree", "8m");
    }
}

```

```
property_set("sys.mem.opt", "false");  
property_set("ro.config.low_ram", "false");  
} else { ...
```

2.2.4. 1G 方案

文件: android\system\core\init\property_service.c

```
static int enable_adaptive_memory(void)  
{  
...  
    //for memory > 1024,  
    if (get_dram_size() > 512) {  
...  
        } else {  
            property_set("dalvik.vm.heapsize", "184m");  
            property_set("dalvik.vm.heapstartsize", "5m");  
            property_set("dalvik.vm.heapgrowthlimit", "48m");  
            property_set("dalvik.vm.heapminfree", "512K");  
            property_set("dalvik.vm.heapmaxfree", "2m");  
            //aw use  
            if(strcmp(buf,"true")){  
                property_set("sys.mem.opt", "true");  
            }  
            property_set("ro.config.low_ram", "true");  
        }  
        return 0;  
    }  
}
```