对于嵌入式设备，尽管在部署前会经历大量的测试和验证，但在使用现场有时候仍不可避免会出现意外情况，如 Kernel opps、panic。当出现类似情况时，系统日志往往无法及时写入 flash，重启后不能获得用于分析问题的关键信息。Ramoops 可以应对此类问题。当发发生 Kernel opps、panic时，它能够将相关日志保存到特定的内存区域，并在软重启后仍可以读取。文章将使用安装 Linux BSP v3.0 的 Apalis iMX6 计算机模块进行说明。

首先使用 Toradex Easy Installer 安装 Linux BSP v3.0。然后下载对应的 Linux 源码，分支为 toradex_4.14-2.3.x-imx。交叉编译工具是 gcc-arm-8.2-2019.01-x86_64-arm-linux-gnueabihf。

应用 Apalis iMX6 默认内核配置。

```
$ make apalis_imx6_defconfig
```

开启 ramoops 功能。

```
$ make menuconfig
File systems → Miscellaneous filesystems
<*>   Persistent store
support

   Choose compression algorithm (ZLIB)  ---
>
[*]     Log kernel console
messages
[*]     Log user space messages
<*>     Log panic/oops to a RAM buffer
```

```
    < >   ROM file system support
    <*>   Persistent store support
            Choose compression algorithm (ZLIB)  --->
    [*]     Log kernel console messages
    [*]     Log user space messages
    <*>     Log panic/oops to a RAM buffer
    < >   System V/Xenix/V7/Coherent file system support
    < >   UFS file system support (read only)
```

为了便于触发 kernel panic 开启 sysrq 功能。

```
Kernel hacking
[*] Magic SysRq key
0x1) Enable magic SysRq key functions by default
[*]   Enable magic SysRq key over serial
```

```
        printk and dmesg options  --->
    [ ] Compile-time checks and compiler options  --->
    [*] Magic SysRq key
    (0x1) Enable magic SysRq key functions by default
    [*]   Enable magic SysRq key over serial
    -*- Kernel debugging
        Memory Debugging  --->
```

最后重新编译内核以及内核模块。

```
$ make zImage LOADADDR=10008000
$ make modules
```

使用新的内核和模块重新启动 Apalis iMX6。Ramoops 在内核配置里又称为 PSTORE，使用下面命令查看之前的内核内置是否生效。

```
root@apalis-imx6:~# zcat /proc/config.gz |grep PSTORE
CONFIG_PSTORE=y
CONFIG_PSTORE_ZLIB_COMPRESS=y
# CONFIG_PSTORE_LZO_COMPRESS is not set
# CONFIG_PSTORE_LZ4_COMPRESS is not set
CONFIG_PSTORE_CONSOLE=y
CONFIG_PSTORE_PMSG=y
CONFIG_PSTORE_RAM=y
root@apalis-imx6:~# zcat /proc/config.gz |grep SYSRQ
CONFIG_MAGIC_SYSRQ=y
CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE=0x1
CONFIG_MAGIC_SYSRQ_SERIAL=y
```

此时，ramoops 还没有配置完成，需要在 device tree里创建对应的节点。在这之前先确定在内存中可以为 ramoops 预留的地址空间。在 Linux 运行下面命令。

```
root@apalis-imx6:~# cat /proc/iomem
00100000-00103fff : /soc/caam-sm@00100000
00120000-00128fff : 120000.hdmi_core
00130000-00133fff : galcore register region
……
02204000-02207fff : galcore register region
02400000-027fffff : 2400000.ipu
02800000-02bfffff : 2800000.ipu
10000000-4fffffff : System RAM
10008000-10cfffff : Kernel code
10e00000-10eeb3cf : Kernel data
```

RAM 的物理地址空间为 0x10000000-0x4fffffff，选择在 Kernel code 和 Kernel data 之外的 0x30000000 作为 ramoops 的起始地址，大小为 1MB。

在 arch/arm/boot/dts/imx6q.dtsi 的 reserved-memory 节点里添加 ramoops。

```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;

    /* global autoconfigured region for contiguous allocations */
    linux,cma {
        compatible = "shared-dma-pool";
        reusable;
        size = <0x14000000>;
        linux,cma-default;
    };

    ramoops@30000000{
            compatible = "ramoops";
            reg = <0x30000000 0x100000>;
            record-size = <0x10000>;
            console-size = <0x10000>;
            pmsg-size = <0x10000>;
    };
};
```

重新编译 device tree。

```
$ make imx6q-apalis-eval.dtb
```

使用新的 device tree 启动后，可以看到以下信息。

```
root@apalis-imx6:~# dmesg|grep ramoops
[    0.071682] pstore: Registered ramoops as persistent store backend
[    0.071707] ramoops: attached 0x100000@0x30000000, ecc: 0/0
```

下面命令分别设置系统在发生 kernel panic 时1 秒后自动重启，以及触发 kernel panic。

```
root@apalis-imx6:~# echo 1 > /proc/sys/kernel/panic
root@apalis-imx6:~# echo c > /proc/sysrq-trigger
```

在自动重启后，将 ramoops 挂载到 /home/root/pstore 目录，可以看到上次发生 kerne panic 时的日志。

```
root@apalis-imx6:~# mkdir -p /home/root/pstore
root@apalis-imx6:~# mount -t pstore psotre /home/root/pstore
root@apalis-imx6:~# ls pstore/
console-ramoops-0  dmesg-ramoops-0    dmesg-ramoops-1

root@apalis-imx6:~/pstore# tail -n 5 console-ramoops-0
[  856.337055] ffa0:                                     00be5898 00000000
00000020 76ed4bb4
[  856.345259] ffc0: 00be5898 00000020 00000002 00000001 76ed71c0 00be6828
00000001 7ed702e0
[  856.353460] ffe0: 00000000 7ed70138 76dd382d 76d88cd0 000f0010 ffffffff
[  856.360101]  r9:00be6828 r8:10c5387d r7:10c5387d r6:ffffffff r5:000f0010
r4:76d88cd0
[  856.385215] Rebooting in 1 seconds..

root@apalis-imx6:~/pstore# tail -n 5 dmesg-ramoops-0
<4>[  856.200454]  r9:00000000 r8:00000000 r7:00000002 r6:00d00440 r5:a909bf00
```

```
r4:a909bf00
<4>[  856.208233] [<80227be8>] (SyS_write) from [<80107d20>]
(ret_fast_syscall+0x0/0x54)
<4>[  856.215828]  r9:a975a000 r8:80107f24 r7:00000004 r6:76f5bda0 r5:00d00440
r4:0000006c
<0>[  856.223594] Code: e5834000 f57ff04e ebf07aaa e3a03000 (e5c34000)
<4>[  856.229847] ---[ end trace 583cc693cbfd2cb1 ]---
```

由于 ramoops 是将日志保存在内存里，如果模块是冷启动，即电源复位，那么相关的内容也不再保留。

总结

ramoops 可以使用较小的开销记录系统日志一般难以保存的 kernel panic 错误。由于存储于内存里面，因此在掉电后这些信息就不复存在。在 Linux 中还有许多其他的调试方法，如kdb, kdump, tracing 等，它们的使用特点和复杂程度也各有不同，用户可以根据需求加以选择。

参考

https://git.toradex.cn/cgit/linux-toradex.git/tree/Documentation/admin-guide/ramoops.rst?h=toradex_4.14-2.3.x-imx
https://lwn.net/Articles/501748/