

OpenEuler kdump

Links

开源软件供应链点亮计划-开源之夏2024 (summer-ospp.ac.cn)

Documentation for Kdump - The kexec-based Crash Dumping Solution – The Linux Kernel documentation

Require Outcomes

- 1. 为 openeuler riscv 提供完善的 kdump 支持
- 2. 功能稳定，不影响系统稳定，不会对其他架构产生影响

Project current bg

openEuler 中的 kexec-tools 包含：

1. 自上游kexec-tools工具，
2. kdump相关的用户态工具、服务配置。如makedumpfile, kdumpctrl等。

其中，makedumpfile 在上游 v1.7.4 版本已经合入riscv架构支持，
kexec-tools工具的riscv支持仍未合入。经观察，其他主流发行版也都没有完整可用的riscv kdump相关工具包提供。需要以相对成熟的x86/arm64架构为参考标准，分析缺少的主要功能和存在的问题，在 riscv 架构下提供相应的实现。

(2) 已有的工作

目前 kexec-tools 工具通过合并来自上游社区贡献者的补丁以及升级 makedumpfile 版本，已经可以实现系统崩溃时启动kdump内核，并且抓取 vmcore 文件。

(3) 存在的不足

1. 加载 kdump 内核时，使用 initrd/initramfs 时加载失败
2. kexec_file_load 方式加载失败

(4) 希望改进的点

分析失败原因，参考x86/arm64架构的支持范围，提供 riscv 相应的 kdump 支持

(5) 最终项目实现的目标

为 openEuler riscv 提供完善的 kdump 支持，代码合入上游 kexec-tools 社区或 openEuler kexec-tools 仓库

Riscv OpenEuler qemu links

How to Build a RISC-V-based openEuler System, Deploy runc, and Run Containers

The default user name and password for the login are root and openEuler12#\$, respectively.

[config_relocatable - kernelconfig.io](#)

[Kexec - ArchWiki \(archlinux.org\)](#)

[Documentation for Kdump - The kexec-based Crash Dumping Solution - The Linux Kernel documentation](#)

Some useful info

- kdump need [config_relocatable](#) or custom dump-capture kernel

There are two possible methods of using Kdump.

1. Build a separate custom dump-capture kernel for capturing the kernel core dump.
2. Or use the system kernel binary itself as dump-capture kernel and there is no need to build a separate dump-capture kernel. This is possible only with the architectures which support a relocatable kernel. As of today, i386, x86_64, ppc64, arm and arm64 architectures support relocatable kernel.

Building a relocatable kernel is advantageous from the point of view that one does not have to build a second kernel for capturing the dump. But at the same time one might want to build a custom dump capture kernel suitable to his needs.

Following are the configuration setting required for system and dump-capture kernels for enabling kdump support.

```
.config - Linux/x86 6.9.0 Kernel Configuration
→ Search (relocatable) ━━━━━━━━ Search Results

Symbol: RELOCATABLE [=y]
Type : bool
Defined at arch/x86/Kconfig:2147
  Prompt: Build a relocatable kernel
  Location:
    -> Processor type and features
      (1)  -> Build a relocatable kernel (RELOCATABLE [=y])
Selected by [y]:
  - EFI_STUB [=y] && EFI [=y]
```

- [makedumpfile](#)

- 单纯fancy一点的cp，把kernel dump file从`/proc/vmcore`复制出来的时候可以有一些额外的筛选功能，属于是toolchain里的东西

Warm up!

- perform system crash

Linux Magic System Request Key Hacks

Documentation for sysrq.c

What is the magic SysRq key?

It is a 'magical' key combo you can hit which the kernel will respond to regardless of whatever else it is doing, unless it is completely locked up.

How do I enable the magic SysRq key?

You need to say "yes" to 'Magic SysRq key (CONFIG_MAGIC_SYSRQ)' when configuring the kernel. When running a kernel with SysRq compiled in, /proc/sys/kernel/sysrq controls the functions allowed to be invoked via the SysRq key. The default value in this file is set by the CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE config symbol, which itself defaults to 1. Here is the list of possible values in /proc/sys/kernel/sysrq:

- 0 - disable sysrq completely
- 1 - enable all functions of sysrq
- >1 - bitmask of allowed sysrq functions (see below for detailed function description):

```

2 = 0x2 - enable control of console logging level
4 = 0x4 - enable control of keyboard (SAK, unraw)
8 = 0x8 - enable debugging dumps of processes etc.
16 = 0x10 - enable sync command
32 = 0x20 - enable remount read-only
64 = 0x40 - enable signalling of processes (term, kill, oom-kill)
128 = 0x80 - allow reboot/poweroff
256 = 0x100 - allow niceing of all RT tasks

```

```

[root@openeuler ~]# cat /proc/sys/kernel/sysrq
0
[root@openeuler ~]# echo 1 > /proc/sys/kernel/sysrq
[root@openeuler ~]# cat /proc/sys/kernel/sysrq
1
[root@openeuler ~]# █

```

What are the 'command' keys?

Command	Function
b	Will immediately reboot the system without syncing or unmounting your disks.
c	Will perform a system crash by a NULL pointer dereference. A crashdump will be taken if configured.

- build rpm package

```
[root@openeuler rpmbuild]# tree
.
├── BUILD
├── BUILDROOT
└── RPMS
    ├── noarch
    │   └── kexec-tools-help-2.0.26-4.noarch.rpm
    └── riscv64
        ├── kexec-tools-2.0.26-4.riscv64.rpm
        ├── kexec-tools-debuginfo-2.0.26-4.riscv64.rpm
        └── kexec-tools-debugsource-2.0.26-4.riscv64.rpm
└── SOURCES
```

- install

```
Complete!
[root@openeuler rpmbuild]# rpm -i RPMS/riscv64/kexec-tools-2.0.26-4.riscv64.rpm
Created symlink /etc/systemd/system/multi-user.target.wants/kdump.service → /usr/lib/systemd/system/kdump.service.
[root@openeuler rpmbuild]# 

[root@openeuler rpmbuild]# kexec
Warning: No cmdline or append string provided
Warning: No dtb provided, using /sys/firmware/fdt
No kernel specified
kexec-tools 2.0.26
Usage: kexec [OPTION]... [kernel]
Directly reboot into a new kernel

-h, --help          Print this help.
-v, --version       Print the version of kexec.
-f, --force         Force an immediate kexec,
                   don't call shutdown.
-i, --no-checks    Fast reboot, no memory integrity checks.
-x, --no-ifdown    Don't bring down network interfaces.
-y, --no-sync      Don't sync filesystems before kexec.
-l, --load         Load the new kernel into the
                   current kernel.
-p, --load-panic   Load the new kernel for use on panic.
-q, --load-quick   Load the new kernel to quick kexec.
-u, --unload       Unload the current kexec target kernel.
                   If capture kernel is being unloaded
                   specify -p with -u.
-e, --exec         Execute a currently loaded kernel.
                   --exec-live-update Execute a currently loaded xen image after
                   storing the state required to live update.
-t, --type=TYPE    Specify the new kernel is of this type.
                   --mem-min=<addr> Specify the lowest memory address to
                   load code into.
                   --mem-max=<addr> Specify the highest memory address to
                   load code into.
                   --reuseinitrd   Reuse initrd from first boot.
                   --print-ckr-size Print crash kernel region size.
                   --load-preserve-context Load the new kernel and preserve
                   context of current kernel during kexec.
                   --load-jump-back-helper Load a helper image to jump back
                   to original kernel.
                   --load-live-update Load the new kernel to overwrite the
                   running kernel.
                   --entry=<addr>  Specify jump back address.
                   (0 means it's not jump back or
                   preserve context)
                   to original kernel.
-s, --kexec-file-syscall Use file based syscall for kexec operation
-c, --kexec-syscall   Use the kexec load syscall for compatibility
                   with systems that don't support -s (default)
-a, --kexec-syscall-auto Use file based syscall for kexec and fall
                   back to the compatibility syscall when file based
                   syscall is not supported or the kernel did not
                   understand the image
-d, --debug          Enable debugging to help spot a failure.
-S, --status         Return 1 if the type (by default crash) is loaded,
                   0 if not.

Supported kernel file types and options:
elf-riscv
image-riscv
  An RISC-V binary image, uncompressed, little endian.
  Typically an Image file.

Architecture options:
  --append=STRING     Append STRING to the kernel command line.
  --dtb=FILE          Use FILE as the device tree blob.
  --initrd=FILE        Use FILE as the kernel initial ramdisk.
  --command-line=STRING Use STRING as the kernel's command line.
  --reuse-cmdline     Use kernel command line from running system.
```

OpenEuler kdump

```
-- No entries --
[root@openeuler rpmbuild]# systemctl status kdump.service
Unit kdump.service could not be found.
[root@openeuler rpmbuild]# systemctl status kdump
* kdump.service - Crash recovery kernel arming
  Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; preset: enabled)
  Active: failed (Result: exit-code) since Thu 2024-05-16 02:48:00 CST; 35s ago
    Process: 10207 ExecStart=/usr/bin/kdumpctl start (code=exited, status=1/FAILURE)
   Main PID: 10207 (code=exited, status=1/FAILURE)

May 16 02:48:00 openeuler systemd[1]: Starting Crash recovery kernel arming...
May 16 02:48:00 openeuler kdumpctl[10209]: No memory reserved for crash kernel
May 16 02:48:00 openeuler kdumpctl[10209]: Starting kdump: [FAILED]
May 16 02:48:00 openeuler systemd[1]: kdump.service: Main process exited, code=exited, status=1/FAILURE
May 16 02:48:00 openeuler systemd[1]: kdump.service: Failed with result 'exit-code'.
May 16 02:48:00 openeuler systemd[1]: Failed to start Crash recovery kernel arming.
[root@openeuler rpmbuild]# 

[root@openeuler rpmbuild]# cat /proc/cmdline
root=/dev/vda2 rw console=ttyS0 systemd.default_timeout_start_sec=600 selinux=0 highres=off earlycon no4lvl
[root@openeuler rpmbuild]# 
```

官方手册里是包含下一级boot的二进制内容的bios，-append没用，加不了crashdump command line option

rv openeuler repo:

<https://mirror.iscas.ac.cn/openeuler-sig-riscv/openEuler-RISC-V/preview/openEuler-23.09-V1-riscv64/repo/23.09/EPOL/main/riscv64>

编译6.6.30版本内核，手动加上command line option，qemu直接引导内核

```
.config - Linux/riscv 6.6.30 Kernel Configuration
+ Boot options
  Boot options
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters
  are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
  for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

  (root=/dev/vda rw console=ttyS0 crashkernel=128M) Built-in kernel command line
  Built-in command line usage (Use bootloader kernel arguments if available) --->
  --*- UEFI runtime support
  [*] Permit falling back to parsing riscv,isa for extension support by default
```

```
.config - Linux/riscv 6.6.30 Kernel Configuration
+ General setup + Kexec and crash features
  Kexec and crash features
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters
  are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
  for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

  [*] Enable kexec system call
  [*] Enable kexec file based system call
  [*] kernel crash dumps
```

```
cmd="qemu-system-riscv64 \
-nographic -machine virt \
-smp "$vcpu" -m "$memory"G \
-kernel "Image" \
-drive file=openeulerfs,format=raw,id=hd0 \
-object rng-random,filename=/dev/urandom,id=rng0 \
-device virtio-gpu \
-device virtio-rng-device,rng=rng0 \
-device virtio-blk-device,drive=hd0 \
-device virtio-net-device,netdev=usernet \
-netdev user,id=usernet,hostfwd=tcp::"$ssh_port"-:22 \
-device qemu-xhci -usb -device usb-kbd -device usb-tablet"
```

```
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Finished Network Manager Wait Online.
[ OK ] Reached target Network is Online.

openEuler 23.03
Kernel 6.6.30 on an riscv64

openeuler-riscv64 login:
```

- 编译openeuler kexec-tools 的rpm包，开kdump

```
[root@openeuler-riscv64 ~]# systemctl restart kdump
[root@openeuler-riscv64 ~]# mv /boot/Image.gz /boot/vmlinuz-6.6.30^c
[root@openeuler-riscv64 ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: enabled)
   Active: active (exited) since Thu 2024-05-16 23:06:11 CST; 11s ago
     Process: 427 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
      Main PID: 427 (code=exited, status=0/SUCCESS)

May 16 23:06:01 openeuler-riscv64 dracut[694]: No dracut internal kernel commandline stored in the initramfs
May 16 23:06:02 openeuler-riscv64 dracut[694]: *** Stripping files ***
May 16 23:06:02 openeuler-riscv64 dracut[694]: *** Stripping files done ***
May 16 23:06:02 openeuler-riscv64 dracut[694]: *** Creating image file '/boot/initramfs-6.6.30kdump.img' ***
May 16 23:06:08 openeuler-riscv64 dracut[694]: Using auto-determined compression method 'pigz'
May 16 23:06:08 openeuler-riscv64 dracut[694]: *** Creating initramfs image file '/boot/initramfs-6.6.30kdump.img' done
May 16 23:06:10 openeuler-riscv64 kdumpctl[2875]: Warning: No dtb provided, using /sys/firmware/fdt
May 16 23:06:11 openeuler-riscv64 kdumpctl[429]: kexec: loaded kdump kernel
May 16 23:06:11 openeuler-riscv64 kdumpctl[429]: Starting kdump: [OK]
May 16 23:06:11 openeuler-riscv64 systemd[1]: Finished Crash recovery kernel arming.
Lines 1-16 (END)
```

- enable sysrq key

```
.config - Linux/riscv 6.6.30 Kernel Configuration
+ Kernel hacking + Generic Kernel Debugging Instruments
Generic Kernel Debugging Instruments

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module capable < > module capable
```

[*] Magic SysRq key (0x1) Enable magic SysRq key functions by default [**] Enable magic SysRq key over serial () Char sequence that enables magic SysRq over serial [*] Debug Filesystem Debugfs default access (Access normal) ---> [] KDB: kernel debugger ---- [] Undefined behaviour sanity checker ----
--

- trigger kernel crash, but kernel panic while loading crash kernel

OpenEuler kdump

```
[ 0.475134] usbhid: USB HID core driver
[ 0.475716] riscv-pmu-sbi: SBI PMU extension is available
[ 0.476259] riscv-pmu-sbi: 16 firmware and 18 hardware counters
[ 0.476510] riscv-pmu-sbi: Perf sampling/filtering is not supported as sscof extension is not available
[ 0.478650] NET: Registered PF INET6 protocol family
[ 0.486522] Segment Routing with IPv6
[ 0.486863] In-situ OAM (IOAM) with IPv6
[ 0.487455] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[ 0.490327] NET: Registered PF PACKET protocol family
[ 0.491749] 9pnet: Installing 9P2000 support
[ 0.492235] Key type dns_resolver registered
[ 0.531111] debug_vm_pgtable: [debug_vm_pgtable] : Validating architecture page table helpers
[ 0.537203] clk: Disabling unused clocks
[ 0.556008] /dev/root: Can't open blockdev
[ 0.556859] VFS: Cannot open root device "" or unknown-block(0,0): error -6
[ 0.557137] Please append a correct "root=" boot option; here are the available partitions:
[ 0.557630] fe00      4096000 vda
[ 0.557711]   driver: virtio_blk
[ 0.558391] List of all bdev filesystems:
[ 0.558597]   ext3
[ 0.558654]   ext2
[ 0.558788]   ext4
[ 0.558877]   vfat
[ 0.558976]   msdos
[ 0.559064]   iso9660
[ 0.559168]
[ 0.559458] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)
[ 0.559901] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 6.6.30 #5
[ 0.560203] Hardware name: riscv-virtio,qemu (DT)
[ 0.560496] Call Trace:
[ 0.560746] [<ffffffffffff80005b1c>] dump_backtrace+0x1c/0x24
[ 0.561034] [<ffffffffffff80885fb0>] show_stack+0x2c/0x38
[ 0.561225] [<ffffffffffff80889200>] dump_stack_lvl+0x3c/0x54
[ 0.561413] [<ffffffffffff80892034>] dump_stack+0x14/0x1c
[ 0.561595] [<ffffffffffff808863ba>] panic+0x104/0x2b
[ 0.561797] [<ffffffffffff80a0156e>] mount_root_generic+0x1d6/0x284
[ 0.562029] [<ffffffffffff80a01816>] mount_root+0x202/0x234
[ 0.562289] [<ffffffffffff80a01a42>] prepare_namespace+0x1fa/0x256
[ 0.562539] [<ffffffffffff80a01002>] kernel_init_freeable+0x258/0x27a
[ 0.562811] [<ffffffffffff8089367c>] kernel_init+0x20/0x10a
[ 0.563027] [<ffffffffffff8089b7de>] ret_from_fork+0xe/0xic
[ 0.563674] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0) ]---
```

```
[root@openeuler-riscv64 ~]# echo c > /proc/sysrq-trigger
```