Zynq (MPSoC) crashkernel

A fallback solution for Linux crashes and kernel panics on Zynq embedded device

Neki Džemaili
CMS-DAQ group

Acknowledgements: P. Žejdl & M. Dobson

Meanwhile on a poor
sys-admins server at 03:00

Table of Contents

Our repositories are released on GitLab!!

- Crashkernel repository
- CentOS 8 rootfs for Zynq MPSoC

Based on the CentOS 7 rootfs documentation from Ralf :D

Overview

Second kernel that starts after main system kernel crashes / panics

- Skips the BootROM and bootloader(s).
- Creates a **dump** of the machine memory for post-mortem analysis.
- Analysis using **Crash utility** provided by Red Hat.
- Uses a small **INITRAMFS**.
- **Reboots** after creating the dump.

Uses **kdump** and **kexec:**

- **Kdump:**
  - Service to setup the crashkernel.
  - Uses the kexec-tools to load the crashkernel and start it after a crash / panic.
  - Also a utility that creates the dump in the crashkernel.
- **Kexec:**
  - Binary tools
  - System call
  - Uses the `KEXEC_ON_CRASH` flag when loading the crashkernel into memory.
  - Flag instructs kexec to start the crashkernel automatically on a system crash.

Kernel configuration (for Zynq devices)

```
$ petalinux-config -c kernel
```

```
Kernel Features  --->
  [*] kexec system call
  [*] Build kdump crash kernel
  [*] Randomize the address of the kernel image
File systems  --->
  Pseudo filesystems  --->
    [*] /proc file system support
    [*]   /proc/kcore support
    [*]   /proc/vmcore support
Kernel hacking  --->
  -*- Kernel debugging
      Compile-time checks and compiler options  --->
        [*] Compile the kernel with debug info
```

CONFIG_KEXEC
CONFIG_CRASH_DUMP
CONFIG_RANDOMIZE_BASE

```
CONFIG_RANDOMIZE_BASE:
Symbol: RANDOMIZE_BASE [=y]
Type  : bool
Prompt: Randomize the address of the kernel image
  Location:
    -> Kernel Features
  Defined at arch/arm64/Kconfig:1187
  Selects: ARM64_MODULE_PLTS [=y] && RELOCATABLE [=y]
```

RELOCATABLE is not directly accessible through menuconfig

CONFIG_PROC_FS
CONFIG_PROC_KCORE
CONFIG_PROC_VMCORE

CONFIG_DEBUG_KERNEL

CONFIG_DEBUG_INFO

Necessary for post-mortem analysis, else the Crash utility can't start properly

Crashkernel memory reservation

- **Reserved memory** is used by the crashkernel, its INITRAMFS and for the services that will be running.
- This memory is **not accessible** by main system kernel

Boot argument in the **device-tree** or in **U-Boot**

“crashkernel=X”

Meta-user layer in PetaLinux for adding custom device-trees

System-user.dtsi

```
/ {
    model = "CUSTOM BOARD CERN";
    compatible = "xlnx,zynqmp";

    chosen {
                xlnx,eeprom = &eeprom;
                bootargs = "earlycon console=ttyPS0,115200 clk_ignore_unused crashkernel=256M
 rd.earlykdump earlyprintk cpuidle.off=1 root=/dev/nfs ip=dhcp rw";
        };
};
```

```
...
[    0.000000] CPU features: detected: Kernel page table isolation (KPTI)
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 1033987
[    0.000000] Kernel command line: earlycon console=ttyPS0,115200 clk_ignore_unused
crashkernel=256M earlyprintk cpuidle.off=1 root=/dev/nfs ip=dhcp rw
[    0.000000] Dentry cache hash table entries: 524288 (order: 10, 4194304 bytes)
[    0.000000] Inode-cache hash table entries: 262144 (order: 9, 2097152 bytes)
...
```

Tests with different amounts of memory:

Memory	Test result
64 MB	Not able to mount rootfs
128 MB	
160 MB	
176 MB	Fails to start kdump service
192 MB	Works
256 MB	Works
512 MB	Works

Red Hat memory recommendations

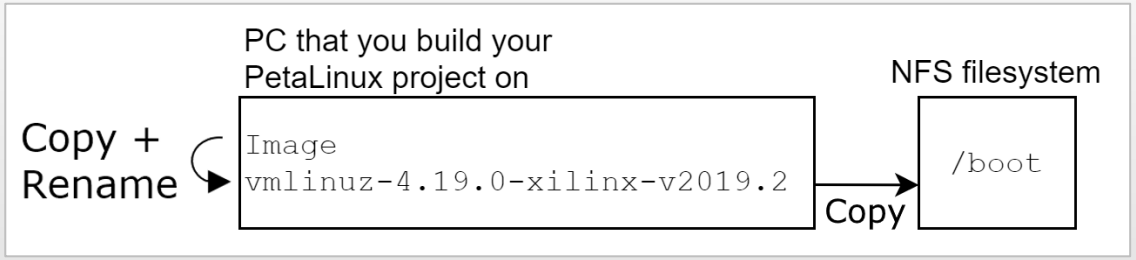
Final configuration steps

Installation of the **kexec-tools** package (provides the tools and system call)

```
$ yum install kexec-tools
```

(**NOTE:** We are using CentOS 8 rootfs for Zynq MPSoC)

Rename and copy the crashkernel image to the /boot directory of your filesystem



Enable the kdump service:

```
$ systemctl start kdump.service
$ systemctl enable kdump.service
$ systemctl status kdump.service
```

(**NOTE:** after enabling kdump,
Dracut will create an INITRAMFS for the crashkernel)

Causing a manual crash

Testing the crashkernel

```
[root@zcu102-lab40-r01-33 /]# echo 1 > /proc/sys/kernel/sysrq
[root@zcu102-lab40-r01-33 /]# echo c > /proc/sysrq-trigger
[  109.209712] sysrq: SysRq : Trigger a crash
[  109.213831] Unable to handle kernel NULL pointer dereference at virtual address 0000000000000000
...
... crash dump with call trace ...
...
[  109.440561] Starting crashdump kernel...
[  109.444469] Bye!

[    0.000000] Booting Linux on physical CPU 0x0000000001 [0x410fd034]
[    0.000000] Linux version 4.19.0-xilinx-v2019.2 (oe-user@oe-host) (gcc version 8.2.0 (GCC)) #1 SMP
Wed Mar 18 11:51:29 UTC 2020
...
[    4.922054] Run /init as init process
[    4.951268] systemd[1]: systemd 239 running in system mode. (...)
[    4.972778] systemd[1]: Detected architecture arm64.
[    4.977814] systemd[1]: Running in initial RAM disk.

Welcome to CentOS Linux 8 (Core) dracut-049-27.git20190906.el8_1.1 (Initramfs)!
...
```

Additional crashkernel configuration

Can be found in **/etc/kdump.conf**

Configuration option	**Default**
Dump-target support: • Raw device • SSH • NFS • Any device with a filesystem (E.g. SD-card)	No dump-target specified. Trying to use **ROOTFS** of the main system kernel
Path	`/var/crash/`
Core_collector	`makedumpfile -l --message-level 1 -d 31`
Kdump post- and pre-scripts	Not enabled
Extra binaries	Not enabled
Extra modules	Not enabled
Failure action	Reboot
Final action	Reboot
Force INITRAMFS rebuild	0
Etc…	

Crashkernel dump targets

Following dump-target configurations have been tested:

Main system kernel ROOTFS	Dump-target configuration	Result
SD-card	-	Successful, dump is saved at `/var/crash/` on SD-card
NFS	SD-card	
NFS	-	Fail, Dracut complains about `ip=` kernel parameter being set multiple times
NFS	Set to NFS	

Generated by Dracut

```
[    9.955976] systemd[1]: Started dracut ask for additional cmdline parameters.
[   10.000123] systemd[1]: Starting dracut cmdline hook...
[   10.101304] dracut-cmdline[1767]: dracut-8 (Core) dracut-049-27.git20190906.el8_1.1
[   10.123909] dracut-cmdline[1767]: Using kernel command line parameters: ip=12
8.141.174.208::128.141.174.1:255.255.255.0::kdump-eth0:none ifname=kdump-eth0:08
:00:30:f4:03:36 nameserver=137.138.16.5 nameserver=137.138.17.5 kdumpnic=kdump-e
th0 bootdev=kdump-eth0 root=nfs4:128.141.174.247:/rootfs/zcu102-lab40-r01-33:rw,
relatime,vers=4.2,rsize=4096,wsize=4096,namlen=255,hard,proto=tcp,timeo=600,retr
ans=2,sec=sys,clientaddr=128.141.174.208,local_lock=none,addr=128.141.174.247 if
name=eth0:08:00:30:f4:03:36 ip=eth0:static earlycon console=ttyPS0,115200 clk_ig
nore_unused earlyprintk cpuidle.off=1 ip=dhcp rw irqpoll nr_cpus=1 reset_devices
 cgroup_disable=memory udev.children-max=2 panic=10 swiotlb=noforce novmcoredd
[   10.584179] dracut-cmdline[1767]: Multiple ip= arguments: assuming rd.neednet=1
[   10.615208] dracut-cmdline[1767]: Warning: Empty autoconf value
[   10.711912] dracut: FATAL: Sorry, 'ip=eth0:static' does not make sense for multiple
interface configurations. es default to dhcp
[   10.731176] dracut: Refusing to continue
[   10.708579] systemd[1]: Shutting down.
```

Dracut configuration, Kdump and Dracut source code and scripts were researched to find a solution, but without any result.

Kdump-post script utilization

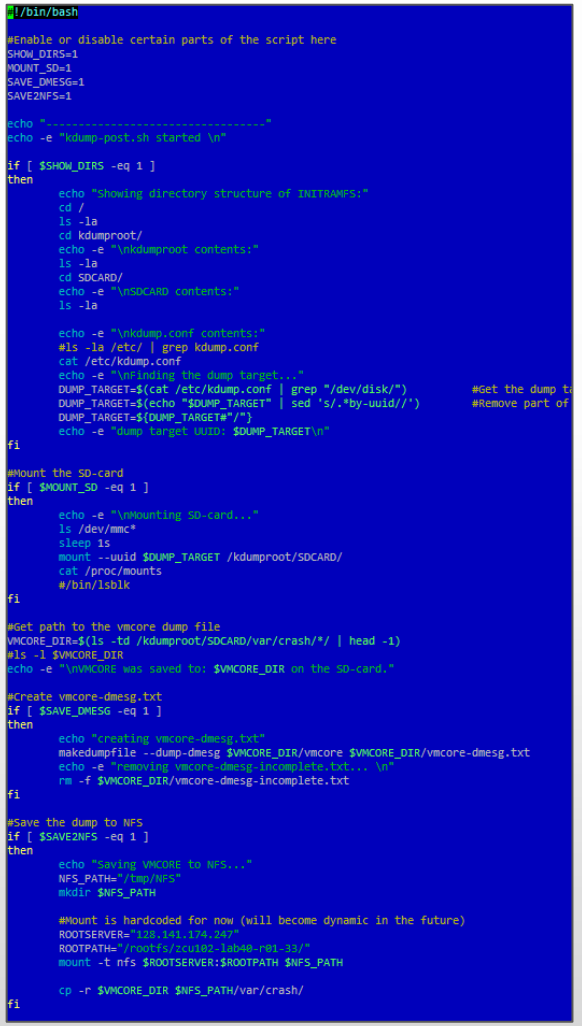
- Script that runs after saving of the dump by kdump
- Using it for workarounds

Current version of the kdump-post script:

- Saving of dmesg log
- Available in our repository

Ongoing work on the kdump-post script:

1. Saving dumps to NFS
2. Dynamic mount of SD-card, based on kdump.conf
3. Showing additional debug info
4. Modular script:
   - Enable and disable specific functionalities of the kdump-post script



```
#!/bin/bash

#Enable or disable certain parts of the script here
SHOW_DIRS=1
MOUNT_SD=1
SAVE_DMESG=1
SAVE2NFS=1

echo "-----------------------------------"
echo -e "kdump-post.sh started \n"

if [ $SHOW_DIRS -eq 1 ]
then
        echo "Showing directory structure of INITRAMFS:"
        cd /
        ls -la
        cd kdumproot/
        echo -e "\nkdumproot contents:"
        ls -la
        cd SDCARD/
        echo -e "\nSDCARD contents:"
        ls -la

        echo -e "\nkdump.conf contents:"
        #ls -la /etc/ | grep kdump.conf
        cat /etc/kdump.conf
        echo -e "\nFinding the dump target..."
        DUMP_TARGET=$(cat /etc/kdump.conf | grep "/dev/disk/")          #Get the dump t
        DUMP_TARGET=$(echo "$DUMP_TARGET" | sed 's/.*by-uuid//')        #Remove part of
        DUMP_TARGET=${DUMP_TARGET#"/"}
        echo -e "dump target UUID: $DUMP_TARGET\n"
fi

#Mount the SD-card
if [ $MOUNT_SD -eq 1 ]
then
        echo -e "\nMounting SD-card..."
        ls /dev/mmc*
        sleep 1s
        mount --uuid $DUMP_TARGET /kdumproot/SDCARD/
        cat /proc/mounts
        #/bin/lsblk
fi

#Get path to the vmcore dump file
VMCORE_DIR=$(ls -td /kdumproot/SDCARD/var/crash/*/ | head -1)
#ls -l $VMCORE_DIR
echo -e "\nVMCORE was saved to: $VMCORE_DIR on the SD-card."

#Create vmcore-dmesg.txt
if [ $SAVE_DMESG -eq 1 ]
then
        echo "creating vmcore-dmesg.txt"
        makedumpfile --dump-dmesg $VMCORE_DIR/vmcore $VMCORE_DIR/vmcore-dmesg.txt
        echo -e "removing vmcore-dmesg-incomplete.txt... \n"
        rm -f $VMCORE_DIR/vmcore-dmesg-incomplete.txt
fi

#Save the dump to NFS
if [ $SAVE2NFS -eq 1 ]
then
        echo "Saving VMCORE to NFS..."
        NFS_PATH="/tmp/NFS"
        mkdir $NFS_PATH

        #Mount is hardcoded for now (will become dynamic in the future)
        ROOTSERVER="128.141.174.247"
        ROOTPATH="/rootfs/zcu102-lab40-r01-33/"
        mount -t nfs $ROOTSERVER:$ROOTPATH $NFS_PATH

        cp -r $VMCORE_DIR $NFS_PATH/var/crash/
fi
```

Dmesg saving issue

The dmesg does not get saved properly during dump collection

Utilize **kdump-post script**
To store dmesg log
manually on SD-card

```
[   18.088805] EXT4-fs (mmcblk0p2): recovery complete
[   18.093604] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
kdump: saving to /kdumproot//SDCARD//var/crash//127.0.0.1-2020-05-12-09:15:35/
[   18.148724] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
kdump: saving vmcore-dmesg.txt
No program header covering vaddr 0xffffff80092bafe0found kexec bug?
kdump: saving vmcore-dmesg.txt failed
kdump: saving vmcore
Copying data                                          : [100.0 %] |           eta: 0s
[   28.492641] random: crng init done
[   28.496053] random: 7 urandom warning(s) missed due to ratelimiting
[   28.411485] rngd[1745]: Enabling JITTER rng support
[   28.411592] rngd[1745]: Initalizing entropy source jitter
kdump: saving vmcore complete
```

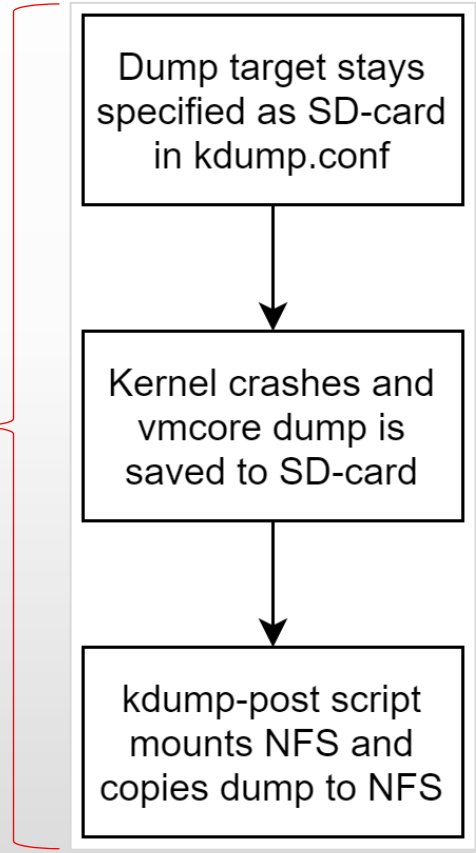
```
/dev/mmcblk0  /dev/mmcblk0p1  /dev/mmcblk0p2
[   29.249530] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
total 47020
-rw------- 1 root root 48141084 May 12 09:15 vmcore
-rw-r--r-- 1 root root        0 May 12 09:15 vmcore-dmesg-incomplete.txt
VMCORE is stored in: tmp/SDCARD/var/crash/127.0.0.1-2020-05-12-09:15:35/
creating vmcore-dmesg.txt

The dmesg log is saved to tmp/SDCARD/var/crash/127.0.0.1-2020-05-12-09:15:35//vmcore-dmesg.txt.

makedumpfile Completed.
[   29.611961] systemd[1]: Shutting down.
```

NFS dump workaround

Utilize the **kdump-post script**



Early kdump

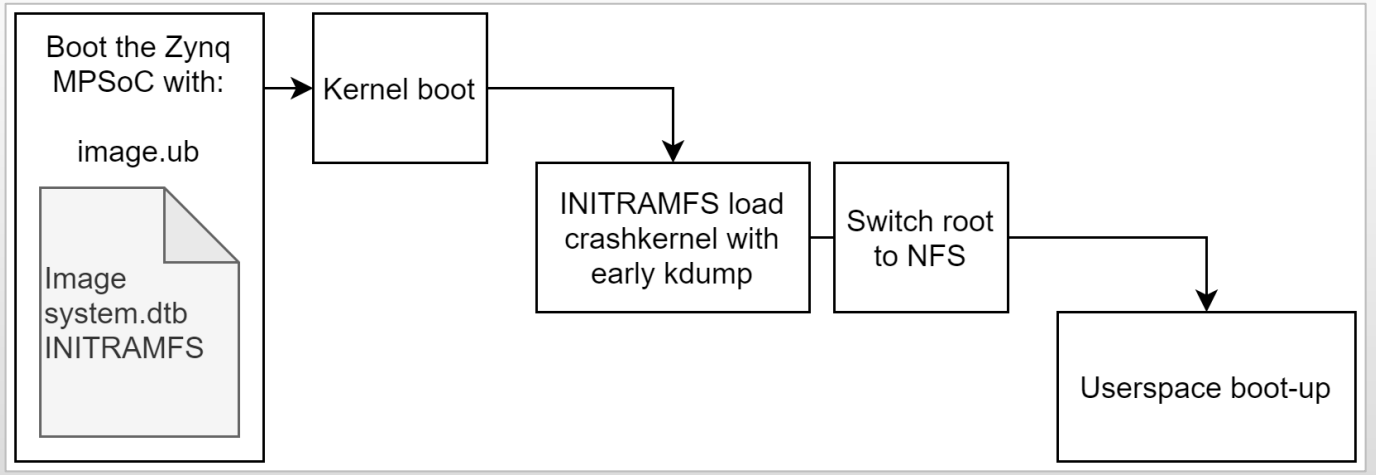
Feature added in CentOS 8 / RHEL 8

- Loads crashkernel into memory earlier
- Allows the kdump service to be started before any other service is started
- Crashkernel can kick in if a service causes a crash / panic during booting

Good reason
For using CentOS 8 ☺

Workings of early kdump:

1. Boots the main system kernel with an INITRAMFS
2. INITRAMFS contains the early kdump module that starts the kdump.service and loads the crashkernel into memory



Red Hat guide on setting up early kdump

Summary

Current version of the crashkernel: → [Crashkernel repository](#) available on GitLab ☺

- SD-card dump target support
- Saving of dmesg log

Next version of the crashkernel:

- NFS dump target support *(works already)*
- Dynamic SD-card mount, based on kdump configuration *(works already)*
- Option to enable additional debug info *(works already)*
- Modular kdump-post script *(works already)*
- Early kdump support for Zynq MPSoC *(needs further testing)*

Thank you, any questions?