kdump: usage and internals

CFP, #LinuxCon, Beijing, June 19-20, 2017

Pratyush Anand(panand@redhat.com)

Dave Young(dyoung@redhat.com)

Overview

- **Kexec** is a mechanism to boot second kernel from the context of first kernel.

- Kexec skips bios/firmware reset stage thus reboot is faster.

- Kdump uses kexec to boot to a capture kernel when system panics.

Kernel: kexec_load()

- The kexec_load() system call loads a new kernel that can be executed later by reboot()
  - long kexec_load(unsigned long entry, unsigned long nr_segments, struct kexec_segment *segments, unsigned long flags);
- User space need to pass segment for different components like kernel, initramfs etc.
  - struct kexec_segment {

```
    void   *buf;        /* Buffer in user space */
    size_t  bufsz;      /* Buffer length in user space */
    void   *mem;        /* Physical address of kernel */
    size_t  memsz;      /* Physical address length */
};
```

Kernel: kexec_load()

- reboot(LINUX_REBOOT_CMD_KEXEC);
- kexec_load() and above reboot() option is only available when kernel was configured with CONFIG_KEXEC.
- Supported architecture:
  - X86, X86_64, ppc64, ia64, S390x, arm
  - arm64 (kernel/kexec, kexec-tools/kexec and makedumpfile are in upstream, kdump will be soon there)
- **KEXEC_ON_CRASH**
  - A flag which can be passed to kexec_load()
  - Execute the new kernel automatically on a system crash.
  - CONFIG_CRASH_DUMP should be configured

Kernel: kexec_file_load()

- CONFIG_KEXEC_FILE should be enabled to use this system call.
- It is an in-kernel way of segment preparation.
  - long kexec_file_load(int kernel_fd, int initrd_fd, unsigned long cmdline_len, const char __user * cmdline_ptr, unsigned long flags);
- User space need to pass kernel and initramfs file descriptor.
- Only supported for x86 and powerpc

User space: Kexec-tools

- Kexec-tools uses kexec_load()/kexec_file_load() and reboot() system call.
- Second kernel booting is mainly two stage process
  - Step 1: Load the second kernel in the memory from the context of first kernel
    - `kexec -l kernel-image --initrd=initrd-image --reuse-cmdline`
  - Step 2: Boot to the loaded kernel
    - `kexec -e`

User space: Kexec-tools

- Use -p for crash kernel load
  - `kexec -p kernel-image --append=command-line-options –initrd=initrd-image`
  - So When kernel crashes we boot to this loaded kernel.
    - `echo c > /proc/sysrq-trigger` : A test method to crash a kernel

Kdump: revisit

- OK...So..We have seen:
  - Kdump involves two different kernels.
  - When **primary**(production) **kernel** crashes, a pre-loaded new kernel boots which is called **capture**/crash **kernel**
  - A kernel to kernel boot loader called **kexec** helps in booting to the capture kernel.
- Capture kernel is kept mostly same as that of primary kernel, but could be different as well.
  - Kernel must be relocatable if they are same.

Kdump: revisit

- Capture kernel loads mostly different initramfs, but could be same as well.
- There may not be an initramfs at all.
- User space of capture kernel copies memory(dump) snapshot of primary kernel to the disk, and then reboots (to primary kernel).
- **Crash-utility/gdb** can analyse the **dump** snapshot after reboot.

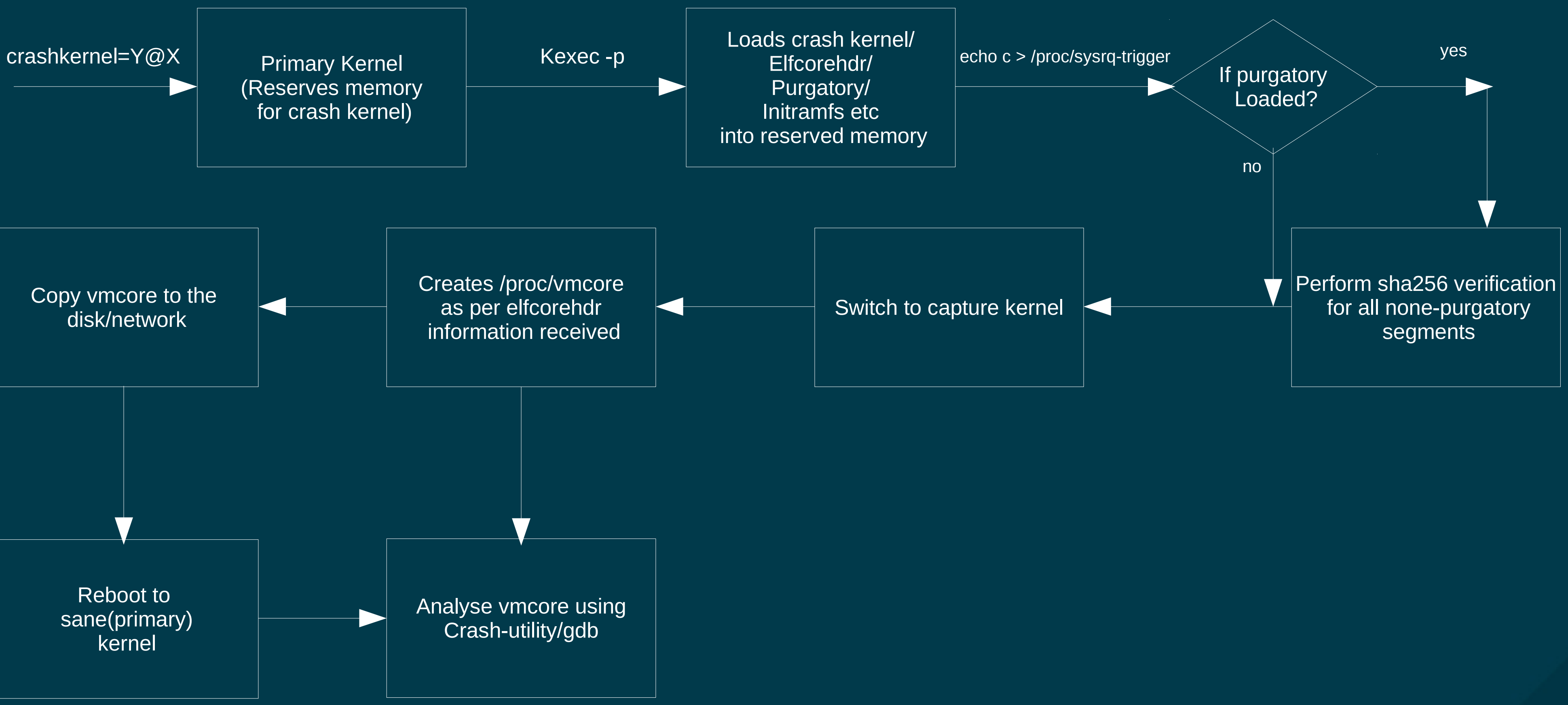
The Primary Kernel

- Needs reserved memory to load capture kernel.
  - Memory is reserved at kernel boot time using crashkernel=xM command line argument.
- When capture kernel is loaded:
  - It also creates elfcorehdr:
    - elfcorehdr stores necessary information about primary kernel’s core image.
    - Information is encoded in ELF format.
  - Can also create purgatory:
    - Purgatory does sha verification before switching to the new kernel.
- Can additionally load an initramfs as well by passing --initrd=initrd-image

The Capture Kernel

- Receives elfcorehdr as kernel cmdline/dtb
  - Arch dependent methods
  - But user do not need to bother, `kexec -p kernel_image` takes care of it.
- It creates a vmcore (/proc/vmcore) as per the core header information mentioned in elfcorehdr
- User space can copy this vmcore to the disk

Kdump : Complete Flow



Reserve Crash Kernel Memory

- crashkernel=size[KMG][@offset[KMG]]
  - Offset is optional, mostly not used.
- crashkernel=range1:size1[,range2:size2,...][@offset]
  - When size is dependent on available system RAM
- crashkernel=size[KMG],high
  - Allocate memory from top, could be above 4G
- crashkernel=size[KMG],low
  - Used only in conjunction with high
  - Allocates memory below 4G when using “high” has allocated above 4G.

Reserve Crash Kernel Memory

- Allocated memory region can be seen using:

  ```
  # cat /proc/iomem | grep "Crash kernel"
    15000000-34ffffff : Crash kernel
  ```
- Allocated memory region size can be seen using:

  ```
  # cat /sys/kernel/kexec_crash_size
  536870912
  ```
- How much memory is needed?
  - depends on initrd, machine IO devices complexity
  - Number of CPUs to be used in crash kernel
  - Usually 256M is good and works

Load Crash Kernel

- A typical command line to load crash kernel
  - kexec -p /boot/vmlinuz-`uname -r` --initrd=/boot/initramfs-`uname -r`kdump.img --reuse-cmdline
- Most of the arch provides options to:
  - reuse/assign/modify command line parameters for capture kernel
    - --reuse-cmdline
    - --command-line="root=/dev/sda1 ro irqpoll maxcpus=1 reset_devices"
    - --apend="irqpoll maxcpus=1 reset_devices"
  - Specify a new initramfs
    - --initrd=/boot/initramfs-`uname -r`kdump.img

Load Crash Kernel

- Can reuse initrd from first boot
  - --reuseinitrd
- See `man kexec` for more detail
- If a crash kernel is loaded

```
# cat /sys/kernel/kexec_crash_loaded
1
```

When Kernel crashes.....

- Prepare cpu registers for panic kernel (crash_setup_regs())
- Update vmcoreinfo note (crash_save_vmcoreinfo())
- shutdown non-crashing cpus and save registers (machine_crash_shutdown())
  - crash_save_cpu() saves registers in cpu notes
  - Might need to disable interrupt controller here
- Perform kexec reboot now (machine_kexec())
  - Load/flush kexec segments to memory
  - Pass control to the execution of entry segment

Purgatory

- Sha256 signature of none purgatory segments are calculated by kexec-tools/kernel and embedded into purgatory binary
- Purgatory code again re-calculates sha256 and compares to the value embedded into it
- Thus, it ensures the new kernel’s pre loaded data is not corrupted
- There are pre and post verification setup_arch() functions

Elf Program Headers

- Most of the dump cores involved in kdump are in ELF format.
- Each elf file has a program header
  - Which is read by the system loader
  - Which describes how the program should be loaded into memory.
  - `Objdump -p elf_file` can be used to look into program headers

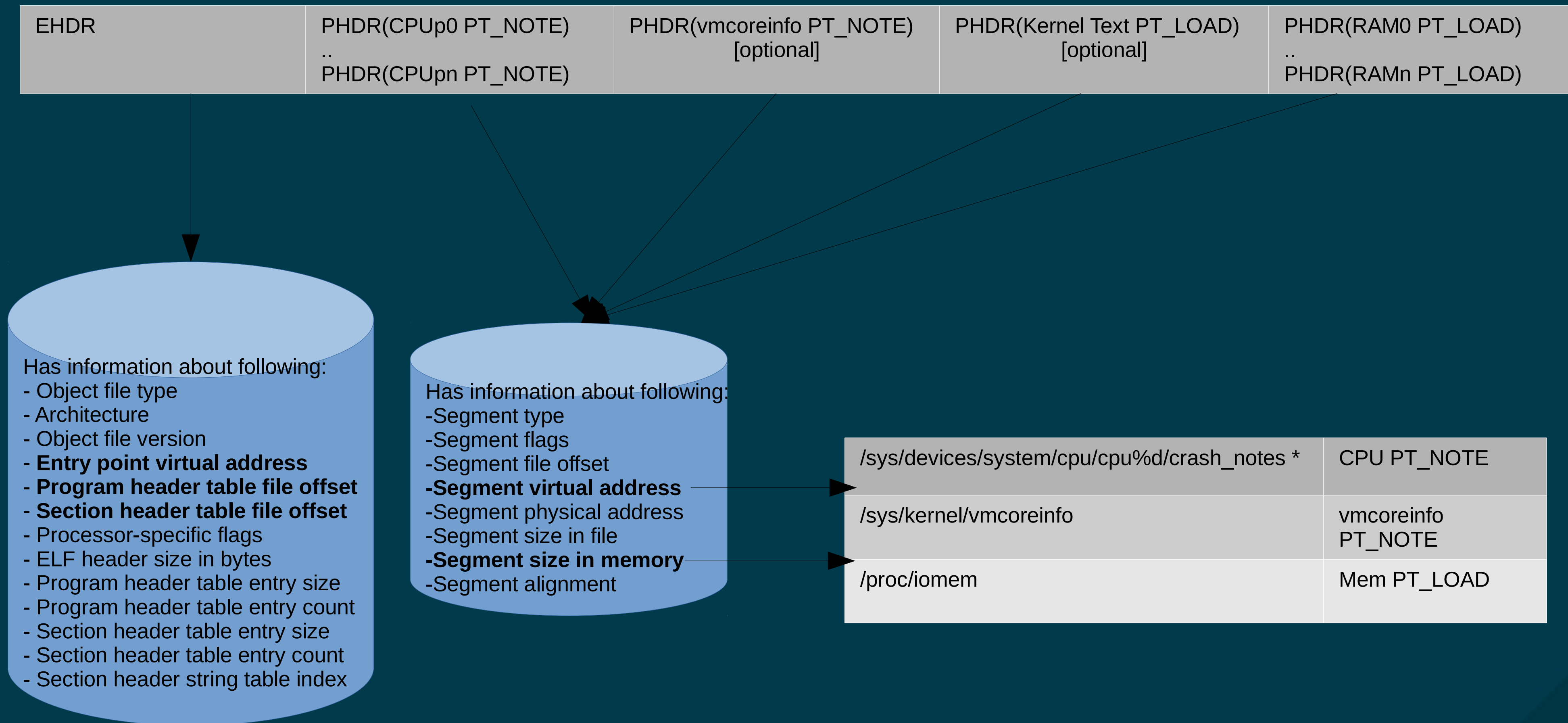
Elf Program Headers

```
# objdump -p vmcore

vmcore:     file format elf64-littleaarch64

Program Header:
    NOTE off    0x0000000000010000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**0
filesz 0x00000000000013e8 memsz 0x00000000000013e8 flags ---
    LOAD off    0x0000000000020000 vaddr 0xffff000008080000 paddr 0x0000004000280000 align 2**0 filesz
0x0000000001460000 memsz 0x0000000001460000 flags rwx
    LOAD off    0x0000000001480000 vaddr 0xffff800000200000 paddr 0x0000004000200000 align 2**0 filesz
0x000000007fc00000 memsz 0x000000007fc00000 flags rwx
    LOAD off    0x0000000081080000 vaddr 0xffff8000ffe00000 paddr 0x00000040ffe00000 align 2**0 filesz
0x00000002fa7a0000 memsz 0x00000002fa7a0000 flags rwx
    LOAD off    0x000000037b820000 vaddr 0xffff8003fa9e0000 paddr 0x00000043fa9e0000 align 2**0 filesz
0x0000000004fc0000 memsz 0x0000000004fc0000 flags rwx
    LOAD off    0x00000003807e0000 vaddr 0xffff8003ff9b0000 paddr 0x00000043ff9b0000 align 2**0 filesz
0x0000000000010000 memsz 0x0000000000010000 flags rwx
    LOAD off    0x00000003807f0000 vaddr 0xffff8003ff9f0000 paddr 0x00000043ff9f0000 align 2**0 filesz
0x0000000000610000 memsz 0x0000000000610000 flags rwx
private flags = 0:
```

Elf Program Headers

- Most of the program headers involved in kdump are of types:
  - PT_NOTE (4): Indicates a segment holding note information.
  - PT_LOAD (1): Indicates that this program header describes a segment to be loaded from the file.

elfcorehdr


EHDR
PHDR(CPUp0 PT_NOTE)
..
PHDR(CPUpn PT_NOTE)
PHDR(vmcoreinfo PT_NOTE)
[optional]
PHDR(Kernel Text PT_LOAD)
[optional]
PHDR(RAM0 PT_LOAD)
..
PHDR(RAMn PT_LOAD)
Has information about following:
- Object file type
- Architecture
- Object file version
- Entry point virtual address
- Program header table file offset
- Section header table file offset
- Processor-specific flags
- ELF header size in bytes
- Program header table entry size
- Program header table entry count
- Section header table entry size
- Section header table entry count
- Section header string table index
Has information about following:
-Segment type
-Segment flags
-Segment file offset
-Segment virtual address
-Segment physical address
-Segment size in file
-Segment size in memory
-Segment alignment
/sys/devices/system/cpu/cpu%d/crash_notes *
CPU PT_NOTE
/sys/kernel/vmcoreinfo
vmcoreinfo PT_NOTE
/proc/iomem
Mem PT_LOAD

Crash notes

- A percpu area for storing cpu states in case of system crash
- Area is terminated by a null note
- Note Name: CORE
- Note Type: NT_PRSTATUS(1)
- Has information about current pid and cpu registers

vmcoreinfo

- This note section has various kernel debug information like struct size, symbol values, page size etc.
- Values are parsed by crash kernel and embedded into /proc/vmcore
- Vmcoreinfo is used mainly by makedumpfile application

vmcoreinfo

- include/linux/kexec.h has macros to define a new vmcoreinfo
  - VMCOREINFO_OSRELEASE()
  - VMCOREINFO_PAGESIZE()
  - VMCOREINFO_SYMBOL()
  - VMCOREINFO_SIZE()
  - VMCOREINFO_STRUCT_SIZE()
  - VMCOREINFO_OFFSET()
  - VMCOREINFO_LENGTH()
  - VMCOREINFO_NUMBER()
  - VMCOREINFO_CONFIG()

vmcore

- Starts with elfcorehdr
- Then all the data represented by different headers like crash notes, vmcoreinfo and memory dump follows.

makedumpfile

- It compresses /proc/vmcore data
- Excludes unnecessary pages like:
  - Pages filled with zero
  - Cache pages without private flag (non-private cache)
  - Cache pages with private flag (private cache)
  - User process data pages
  - Free pages
- Needs first kernel’s debug information to exclude unnecessary pages

makedumpfile

- Debug information comes from either VMLINUX or VMCOREINFO
- Can also erase any specific sensitive kernel symbol
- Output can either be in ELF format or kdump-compressed format
- Typical usage:
  - makedumpfile -l --message-level 1 -d 31 /proc/vmcore makedumpfilecore
    - -d is the compression level

Analysing crash

- gdb
- Crash-utility
    - Have physical view of memory
    - Typical usage:
        - crash vmlinux vmcore
        - If vmcore is corrupted and we are not in crash shell, then crash can be started in minimal mode (pass –minimal)
            - Only few commands are available in minimal mode
        - Type help for command list in crash shell

Analysing crash : An example

- bt/log/dmesg: can tail the point of crash and cpu register values at that time

crash> bt

[…]

PC: ffff0000084b7984 [sysrq_handle_crash+36]

LR: ffff0000084b85b0 [__handle_sysrq+296]

[…]

X2: 0000000000040a00   X1: 0000000000000000   X0: 0000000000000001

#6 [ffff8003d3ac7cc0] __handle_sysrq at ffff0000084b85ac

#7 [ffff8003d3ac7d00] write_sysrq_trigger at ffff0000084b8a24

Analysing crash : An example

- Want to see the code at crash point

  crash> dis ffff0000084b7984

  0xffff0000084b7984 <sysrq_handle_crash+36>: strb w0, [x1]

- What went wrong:
  - bt says x1=0x0 and w0=0x1
  - Code was trying to write 0x1 at address 0x0, and it crashed

Kdump: The Fedora way

- Fedora has some scripts to take care of various use case scenarios.
- Configurations files:
  - /etc/sysconfig/kdump:
    - Initrd rebuild is not needed after any configuration change, like:
      - KDUMP_COMMANDLINE_APPEND: append arguments to the current kdump commandline
      - KEXEC_ARGS: any extra argument which we want to pass to kexec command
      - KDUMP_IMG: to specify image other than default kernel image

Kdump: The Fedora way

- /etc/kdump.conf:
  - Values which can affect initrd rebuild, like:
    - **core_collector**: specifies the command to copy the vmcore.
    - **path**: file system path where vmcore will be saved
    - kdump_pre/post: script/command which need to run before and after vmcore save
    - default: if something goes wrong then what to do (reboot |halt|poweroff|shell|dump_to_rootfs)
    - extra_modules: if you want to add any extra kernel modules in initrd
    - extra_bins: any extra binary file

Kdump: The Fedora way

- /proc/sys/kernel/sysrq:
    - Need to write 1 to enable test crash using `echo c > /proc/sysrq-trigger`
- Start/stop/status kdump service:
    - systemctl start kdump
    - systemctl stop kdump
    - systemctl status kdump

Debugging Kdump issues

- `Kexec -p kernel_image` did not succeed
  - Check if crash memory is allocated
    - cat /sys/kernel/kexec_crash_size
      - Should have none zero value
    - cat /proc/iomem | grep "Crash kernel"
      - Should have an allocated range
    - If not allocated, then pass proper “crashkernel=” argument in command line
    - If nothing shows up then pass -d in the kexec command and share debug output with kexec mailing list.

Debugging Kdump issues

- Do not see anything on console after last message from first kernel (like “bye”):
  - Check if `kexec -l kernel_image’ followed by `kexec -e` works
  - Might be missing some arch/machine specific options
  - Might have purgatory sha verification failed. If your arch does not support a console in purgatory then it is very difficult to debug.
  - Might have second kernel crashed very early
    - Pass some earlycon/earlyprintk option for your system to the second kernel command line
    - Share dmesg log of both 1st and 2nd kernel with kexec mailing list.

What next

- shrink memory use for kdump initramfs
- move distribution initramfs code to upstream
- simplify kdump setup
- Kdump support for arm64 coming soon
- kexec_file_load() support for unsupported arch

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos