Kdump
Smarter, Easier, Trustier

Vivek Goyal (IBM)

Neil Horman (RedHat)

Ken'ichi Ohmichi (NEC Soft)

Maneesh Soni (IBM)

Ankita Garg (IBM)

Agenda

- Introduction
- Relocatable bzImage
- Dump Filtering
- Kdump Initramfs
- Linux Kernel Dump Test Module (LKDTM)
- Device Driver Hardening
- Early Boot Crash Dumping

Introduction

- I don't want to ship an extra kernel
  - Relocatable bzImage
- I don't have space to save full dump
  - Dump Filtering
- You can't assume that root file system is not corrupted after a crash
  - Kdump initramfs

Extra kernel binary

- Fixed load address for bzImage and vmlinux (1 MB physical)
- Dump Capture kernel runs from a reserved memory area
- CONFIG_PHYSICAL_START to build a separate kernel image
- Distributions don't want to ship extra kernel binary

Relocatable bzImage

- Single kernel binary can be run from various physical addresses
- No need to build a separate dump capture kernel
- Jan Kratochvil kicked off discussion with some patches
- Eric W. Biederman posted CFT patches for x86 and x86_64

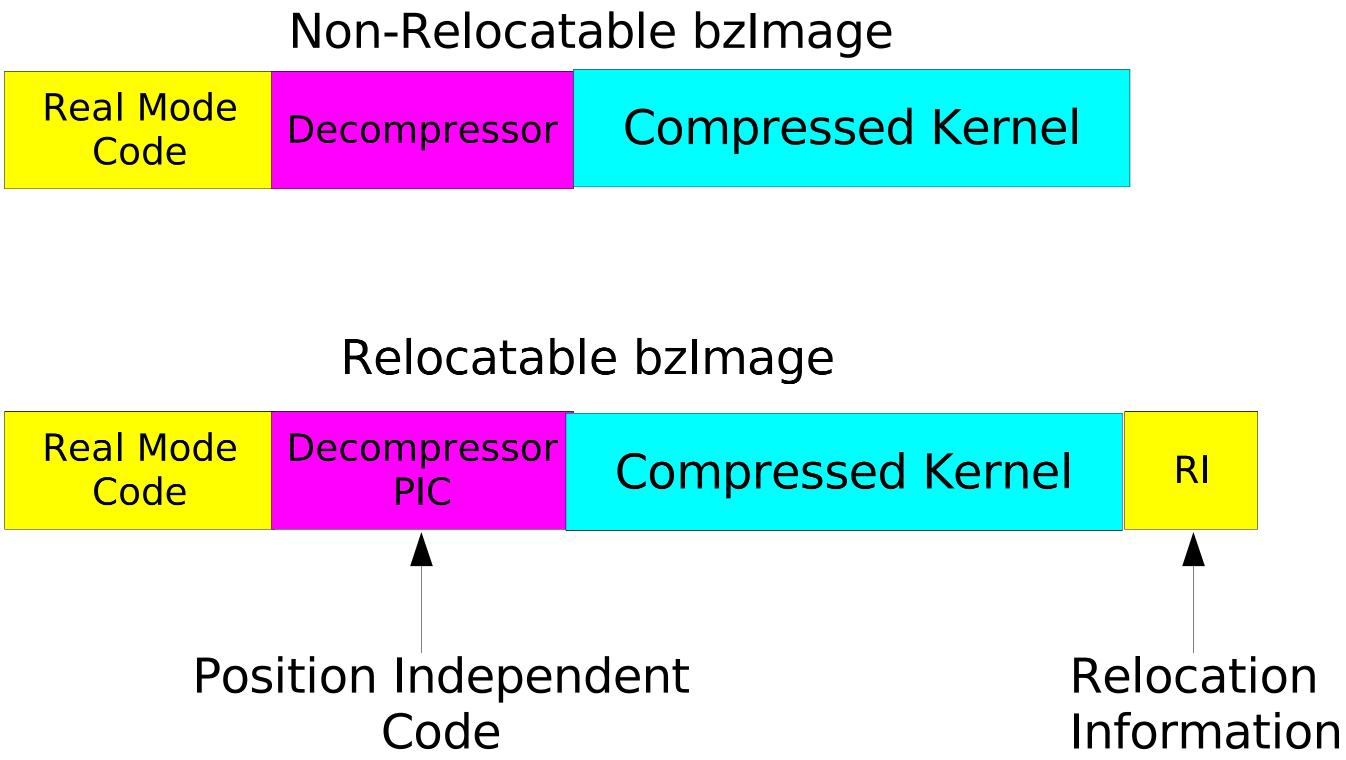
Design Approach

- Modify kernel text/data mapping at run time
  - Implemented for x86_64
- Relocate using relocation information
  - Implemented for i386

Design Details (i386)

- Linker generates relocation information
  - Compile vmlinux with *-shared* flag
    - Absolute symbol relocations
  - Compile vmlinux with --*emit-relocs* option
- Extract and process relocation information
  - Filter out relocations w.r.t. absolute symbols
- Pack relocation information into bzImage
- Process relocations at run time

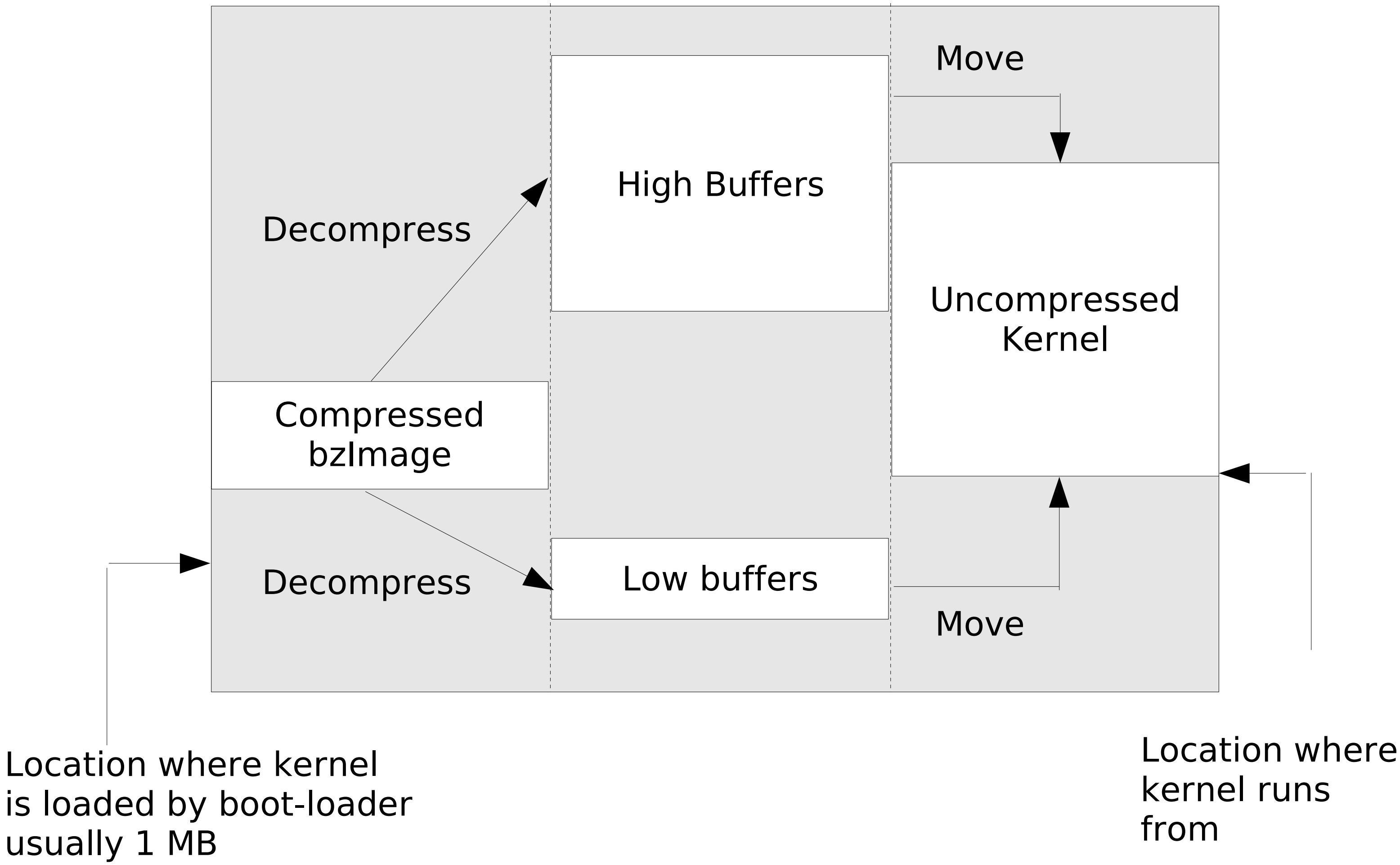
Design Details (i386)



Design Details (i386)

- Relocations increase the size of vmlinux by roughly 10%
- Relocations are discarded at run time
- Decompressor is compiled as position independent code (-fPIC)
- No relocation processing is done for decompressor
- Relocations are performed after decompression and control jumps to startup_32()

Old Decompression Logic



Move

High Buffers

Decompress

Uncompressed Kernel

Compressed bzImage

Decompress

Low buffers

Move

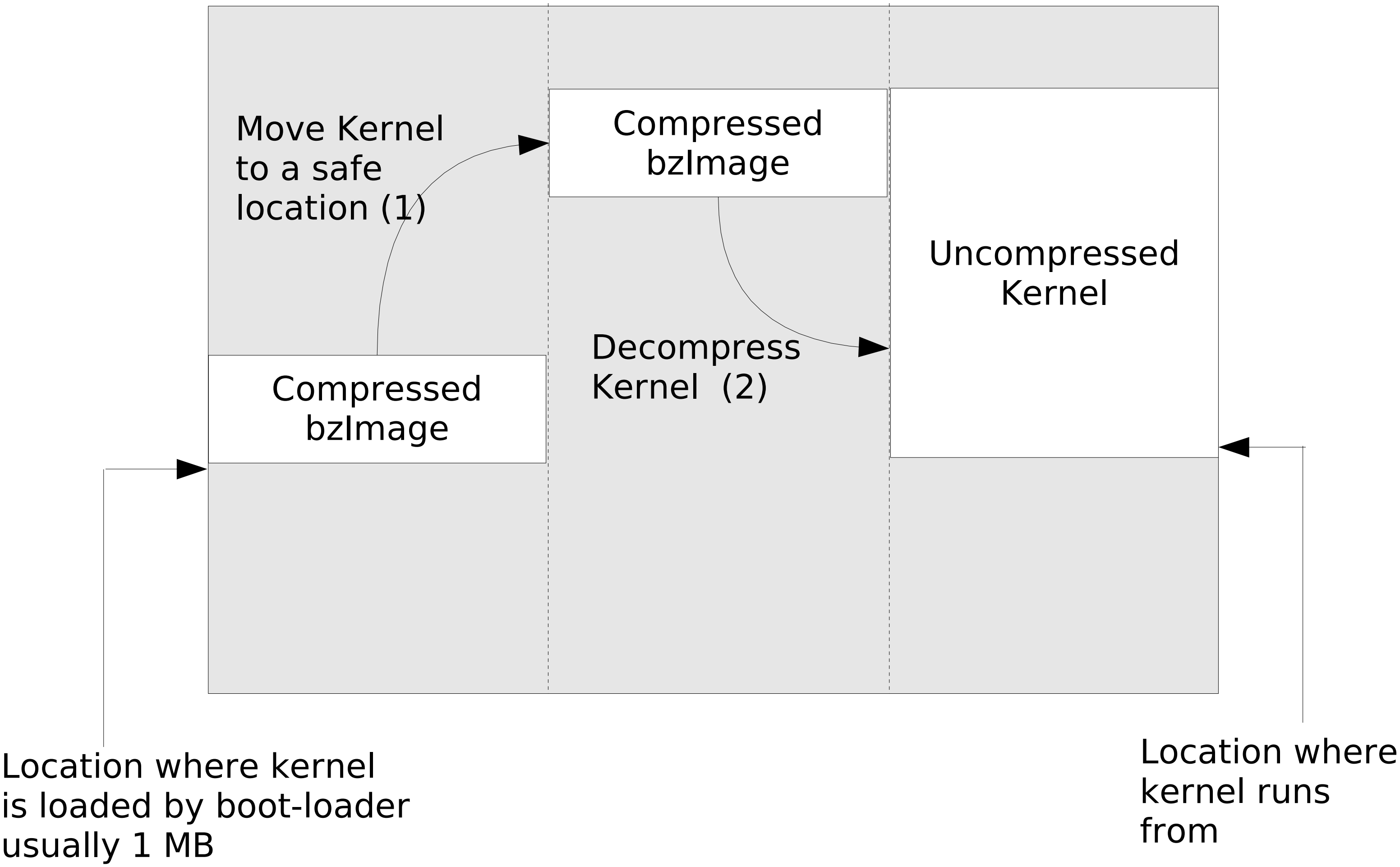
Location where kernel
is loaded by boot-loader
usually 1 MB

Location where
kernel runs
from

Old Decompression Logic

- Hard coded assumptions about low memory area for decompression
- Will overwrite other data if kernel execution needs to be bounded as in case of kdump
- Overall memory used for decompression is more than uncompressed size of kernel

In-Place Decompression



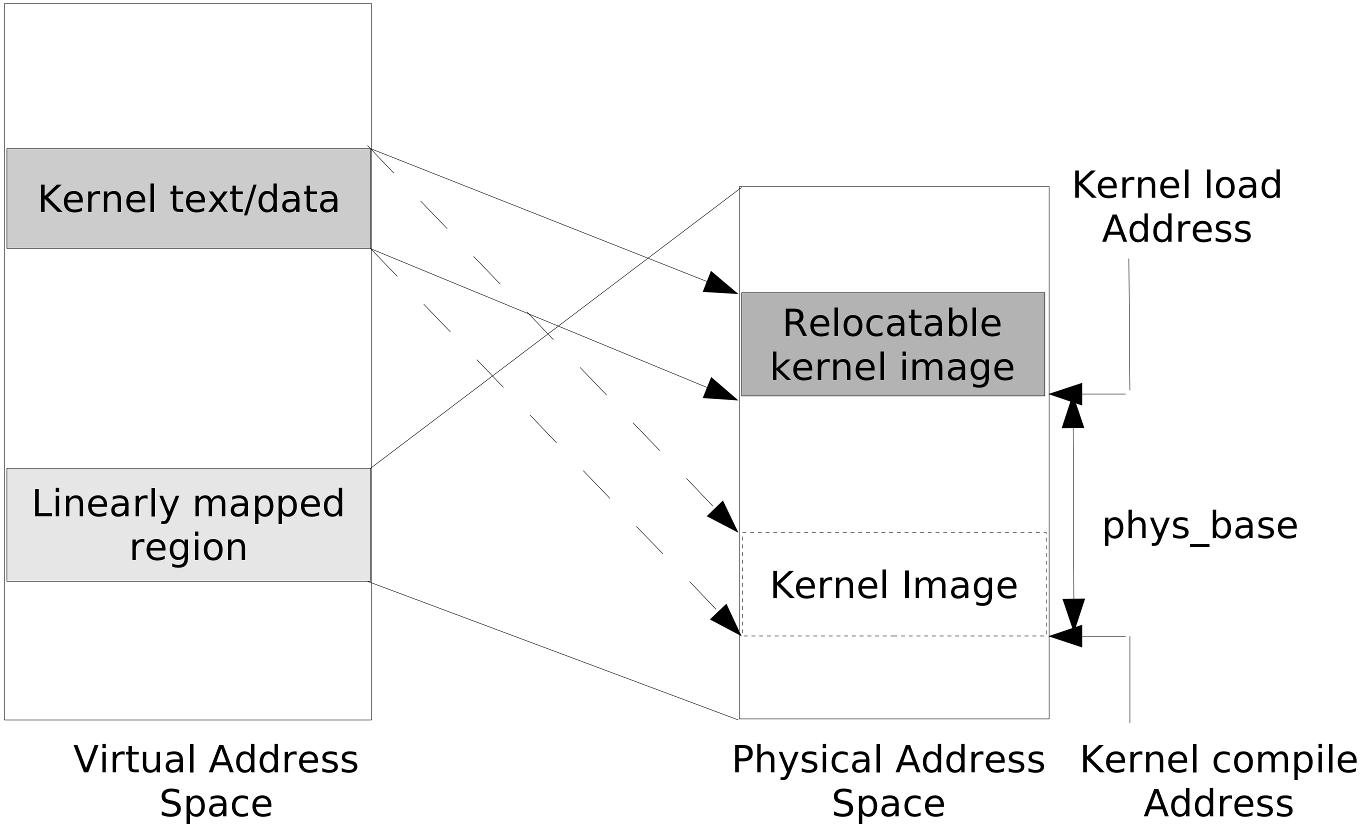
In-place Decompression

- Optimized overall memory used for decompression
- No hard codings regarding intermediate memory used for decompression

Design Details (x86_64)

- Kernel text and data region is separate from linearly mapped region
- Kernel text and data mappings are modified at run time
- Kernel location is determined at run time and kernel text/data mappings are updated

Design Details (x86_64)



__pa() Changes

- Modified __pa() to accommodate for the shift
- Old implementation

```
#define __pa(x)              (((unsigned long)(x)>=__START_KERNEL_map)?(unsigned long)(x) -
    (unsigned long)__START_KERNEL_map: (unsigned long)(x) – PAGE_OFFSET)
```

- New implementation

```
#define __pa(x)        __phys_addr((unsigned long)(x))
unsigned long __phys_addr(unsigned long x)
{
        if (x >= __START_KERNEL_map)
                return x - __START_KERNEL_map + phys_base;
        return x – PAGE_OFFSET;
}
```

bzImage Protocol Extension

- BzImage protocol extended (v2.05)
- Two new fields in bzImage header
  - relocatable_kernel

    Is Kernel Relocatable or Not
  - kernel_alignment

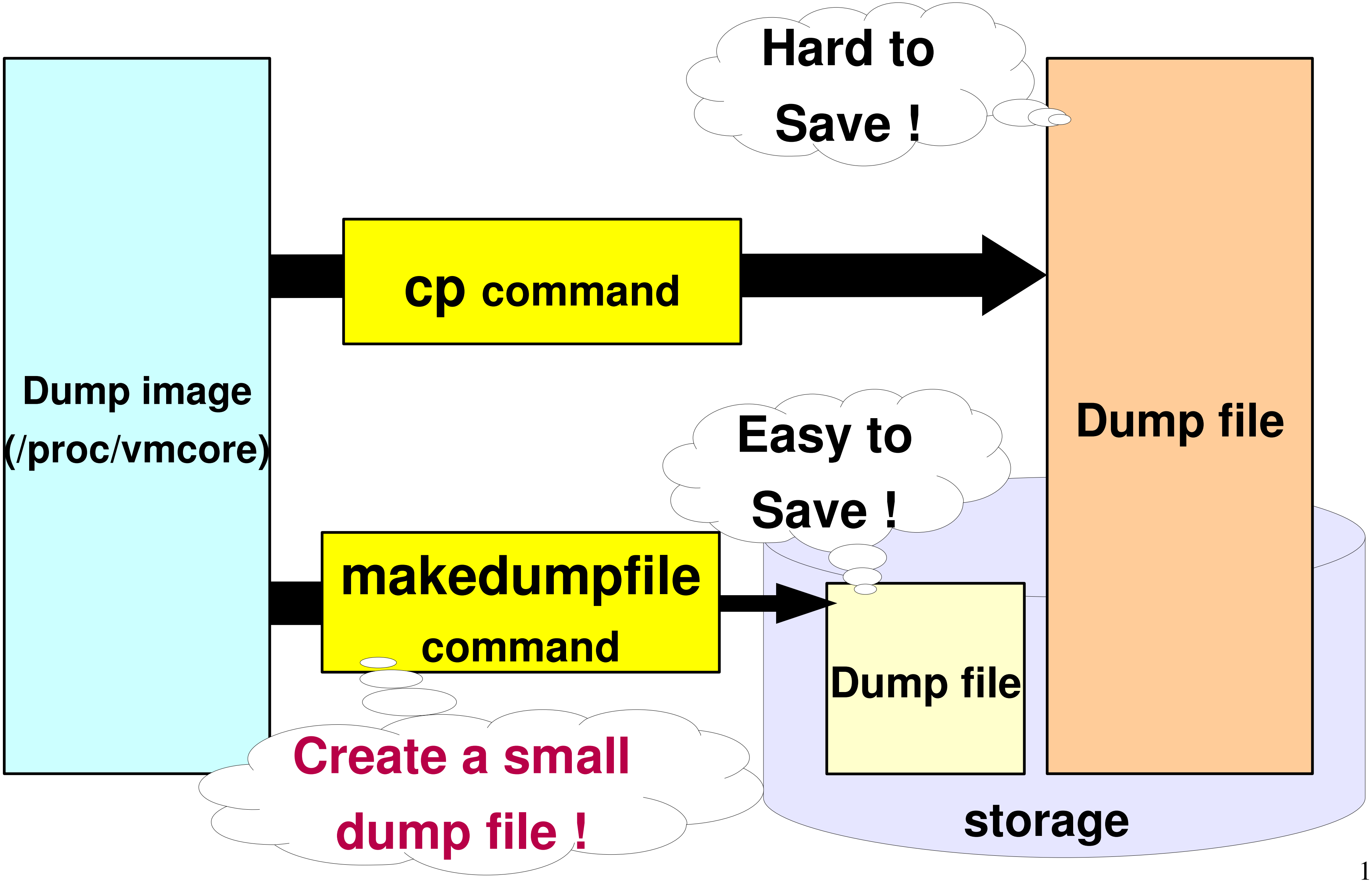
    Physical memory alignment required for kernel

Dump Filtering

(*makedumpfile*)

https://sourceforge.net/projects/makedumpfile

Purpose of Dump Filtering



Dump image (/proc/vmcore)

cp command

Hard to Save !

Dump file

makedumpfile command

Easy to Save !

Dump file

storage

Create a small dump file !

How to create a small dump file

- makedumpfile creates a small dump file by filtering out some or all of the following pages as unnecessary pages for the analysis.
    - Pages filled with zero
    - Free pages
    - Cache pages
    - User process data pages

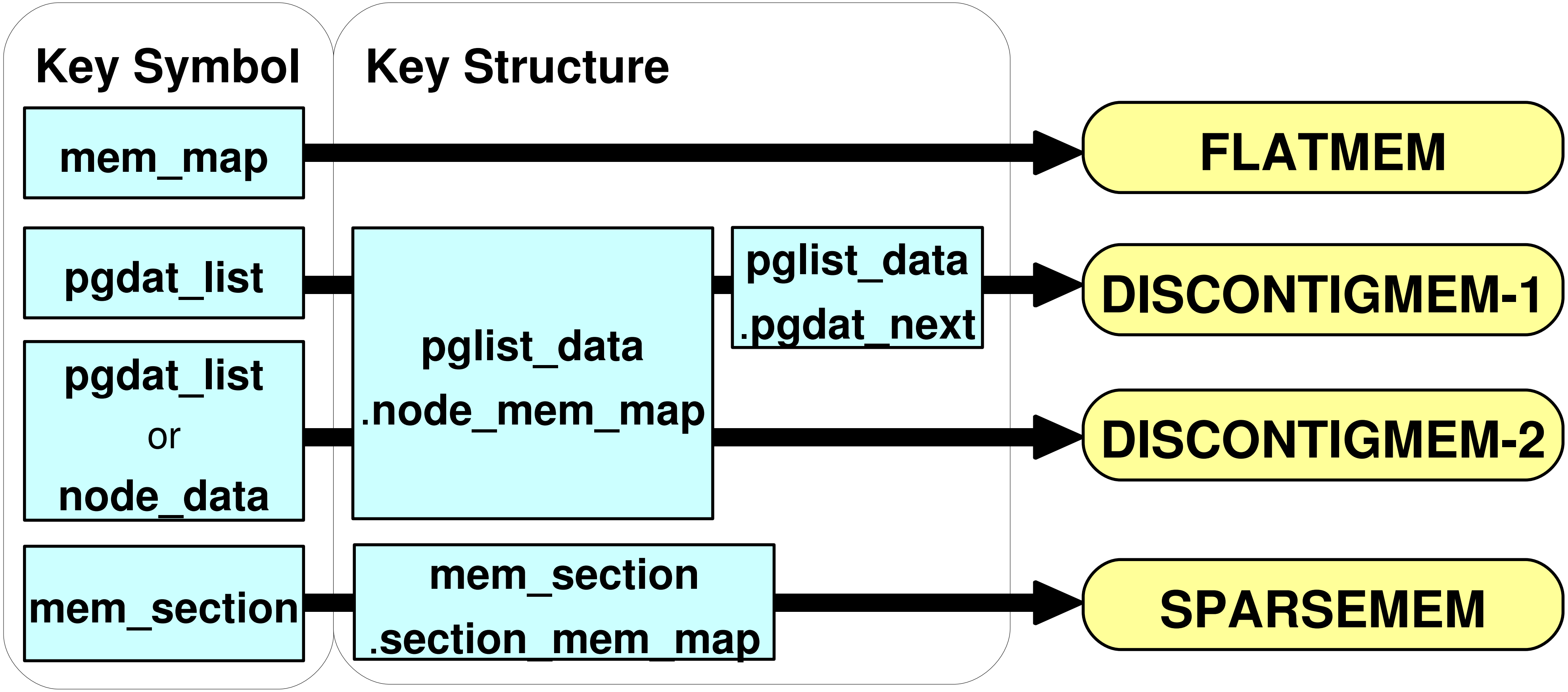
Implementation

- Each page type is distinguished by following
  - **Pages filled with zero**
    - Read each page
  - **Free pages**
    - Scan the member free_area in struct zone
  - **Cache pages** and **User process data pages**
    1. Determine the memory model
    2. Find out struct page
    3. Check attributes in struct page

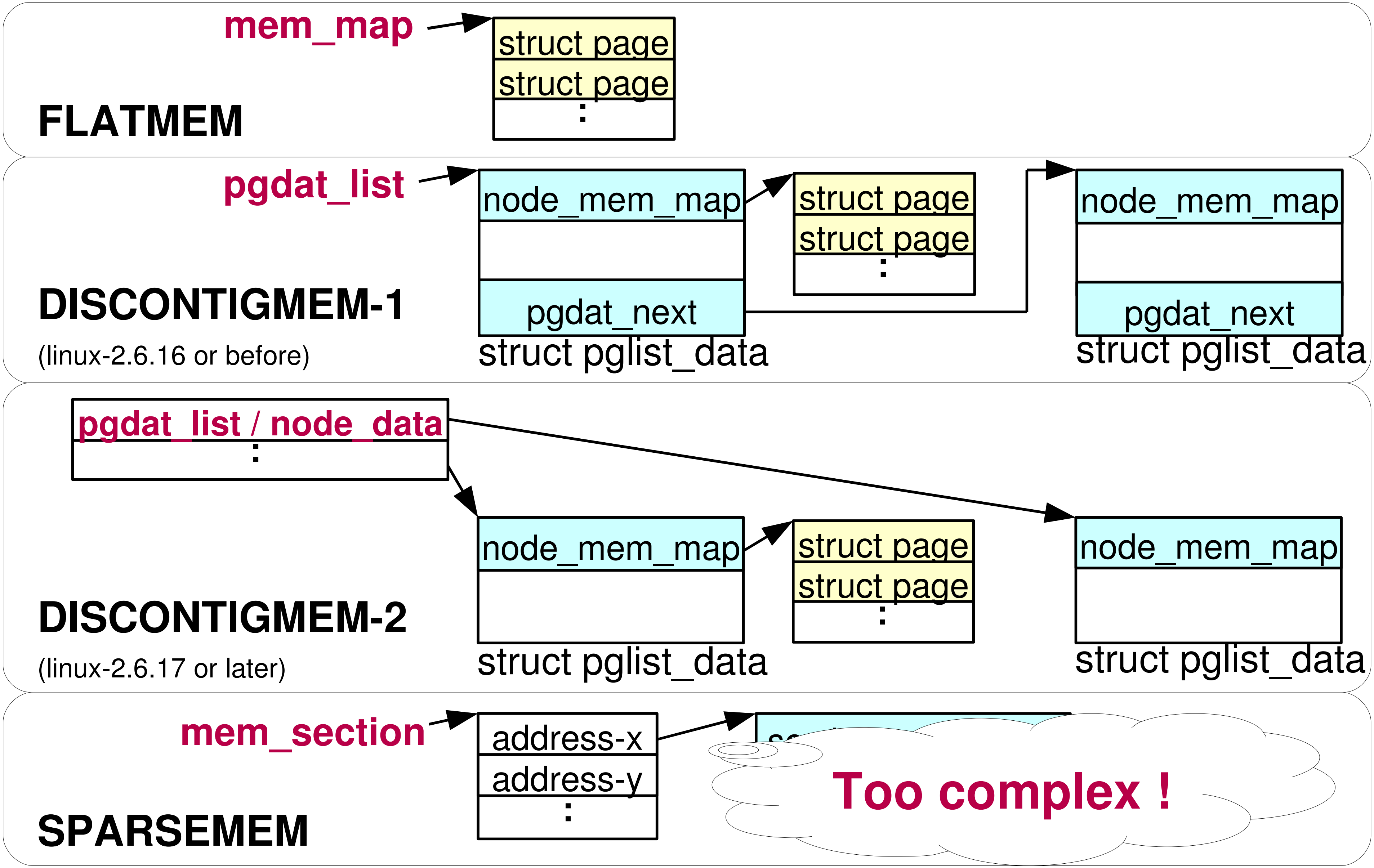
Next Slide

1. Determine the memory model

- Determine the memory model from the key symbols and structures in the vmlinux file containing debug information.



2. Find out struct page



mem_map

struct page
struct page
:

FLATMEM

pgdat_list

node_mem_map

pgdat_next

struct pglist_data

struct page
struct page
:

node_mem_map

pgdat_next

struct pglist_data

DISCONTIGMEM-1
(linux-2.6.16 or before)

pgdat_list / node_data
:

node_mem_map

struct pglist_data

struct page
struct page
:

node_mem_map

struct pglist_data

DISCONTIGMEM-2
(linux-2.6.17 or later)

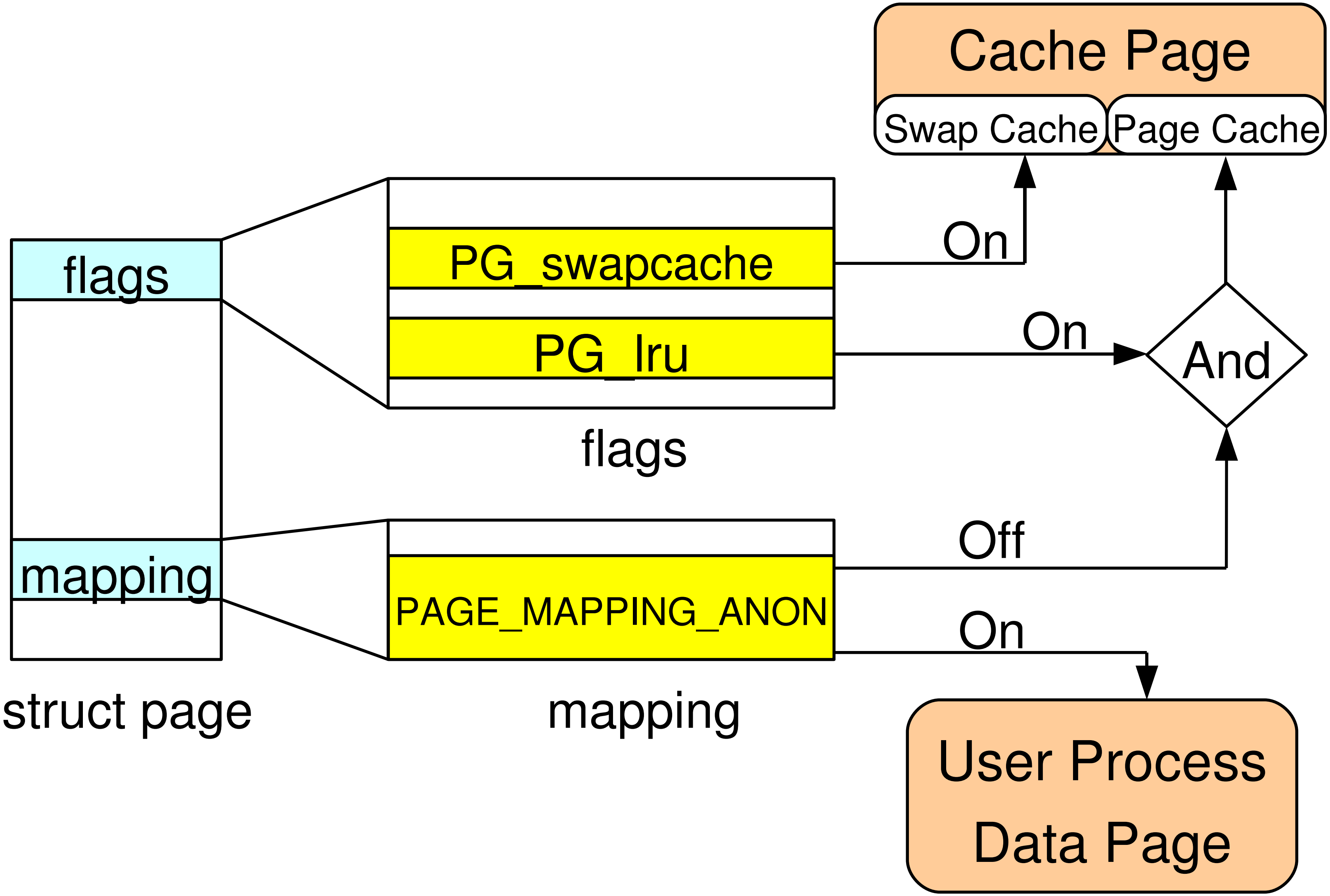
mem_section

address-x
address-y
:

Too complex !

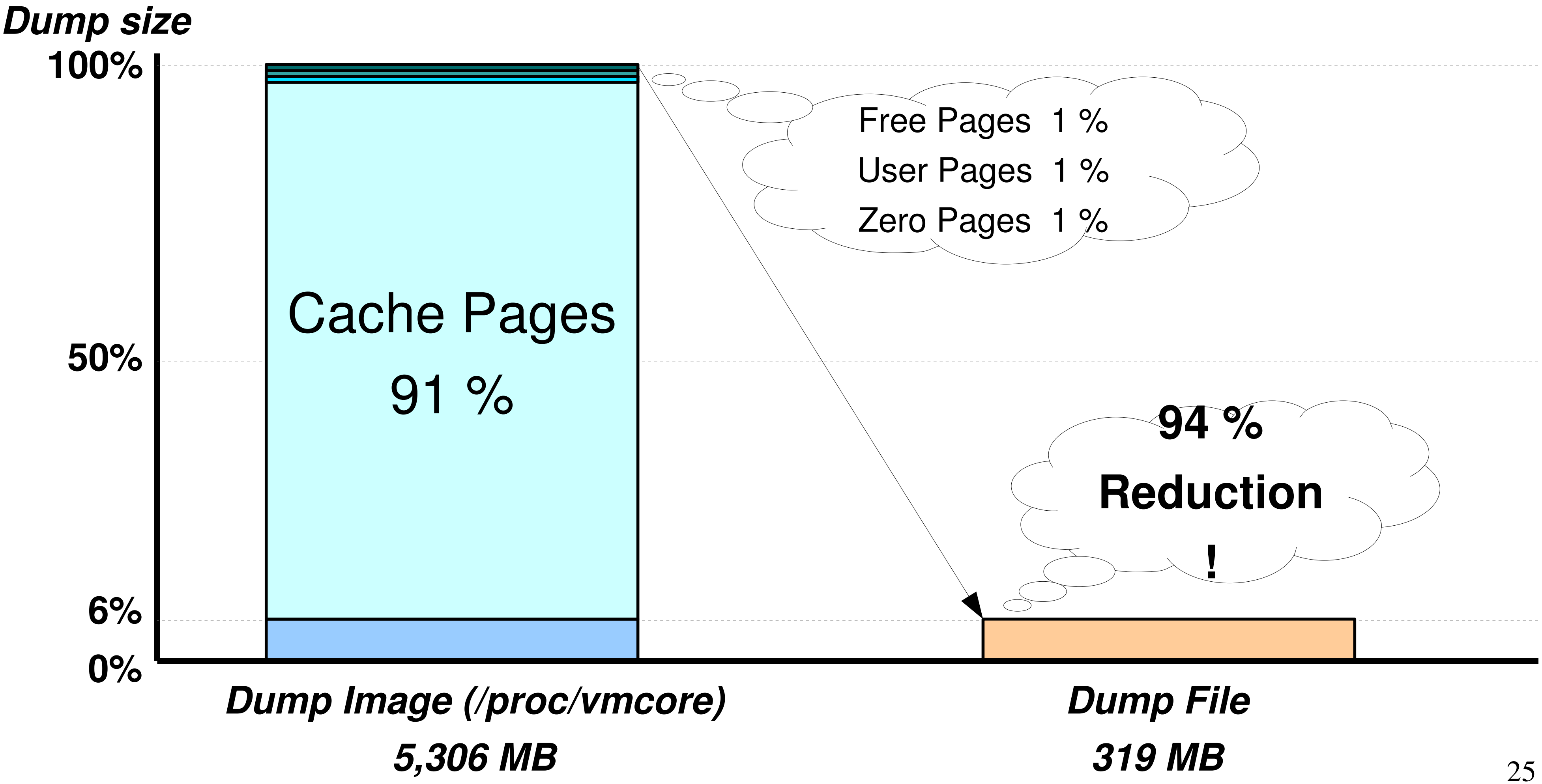
SPARSEMEM

3. Check attributes in struct page



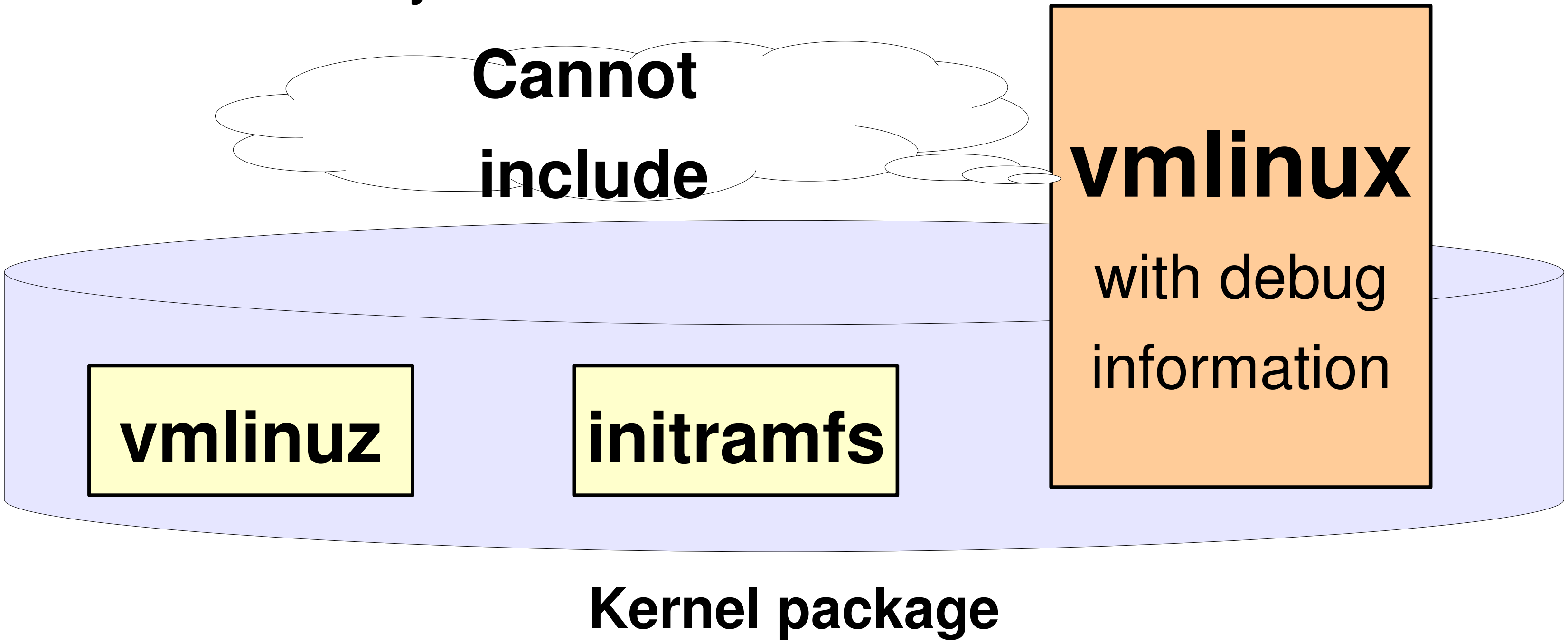
Reduction of Dump File

- x86_64, 5GB system memory
- Panic occurred during heavy I/O.



vmlinux too large

- makedumpfile needs vmlinux file containing debug information, but the file is large (about 40MB).
  - Distributors cannot easily ship the file with the kernel package (about 10MB), and makedumpfile users needs to install the file by themself.



mkdfinfo file instead of vmlinux

- makedumpfile extracts necessary information (structure sizes, member offsets, etc.) from the vmlinux file, and outputs it to a mkdfinfo file. The file is small (about 1KB), and makedumpfile can use it instead of the vmlinux file.
  - **A distributor can ship it easily !**
  - Also a mkdfinfo file is small enough to be included into 2nd-kernel initramfs (**kdump initramfs**), so makedumpfile can run without mounting a root file system.

Kdump Initramfs Enhancements

Components of Kdump

- Kexec – kernel infrastructure that enables memory preserving reboot
- kdump kernel – Previously a separate kernel specially built for use in kexec reboot. Advent of relocatability patch allows for normal boot kernel to be used
- Kdump initramfs – Holds code/utils for use in crash recovery

Early Kdump Implementation

- Initramfs was minimal – Booted rootfs using nash
- Rootfs initscripts were responsible for dump capture
- Unsafe/Unreliable – Can't rely on integrity of crashed systems root filesystem when recovering crash dump.

Early kdump advances

- Moved rootfs utilities to task specific kdump initramfs
- Improved reliability
- Used very limited ram space very quickly – especially when you need to include DSO's
- Still used nash – limited flexibility

Kdump: Initramfs Goals

- Further reduce reliance on root filesystem for dump capture
- Increase number of dump targets and configuration flexibility
- Further limit the amount of memory required to store the kdump initramfs image

Solution: Busybox

- Large utilty set lets us have a complete environment in an initramfs
- Static linking reduces need for DSO's & confines memory requirements
- Scriptible environment lets us embed far more logic/intelligence in initramfs
- Gives us more configuration flexibility

Specific Improvements

- Use of msh to script dump process
- Ability to drop to shell in initramfs
- ifup/ifdown infrastructure allows for configuration of complex network environments (bonding/vlans)
- 70% reduction in most dump target configurations

Future Improvements

- Removal of remaining reliance on DSO's (ssh/scp)
- Removal of initramfs init script generation
- Overhaul of kdump configuration to support multiple/sequential dump targets
- Taking suggestions

Linux Kernel Dump Test Module (LKDTM)

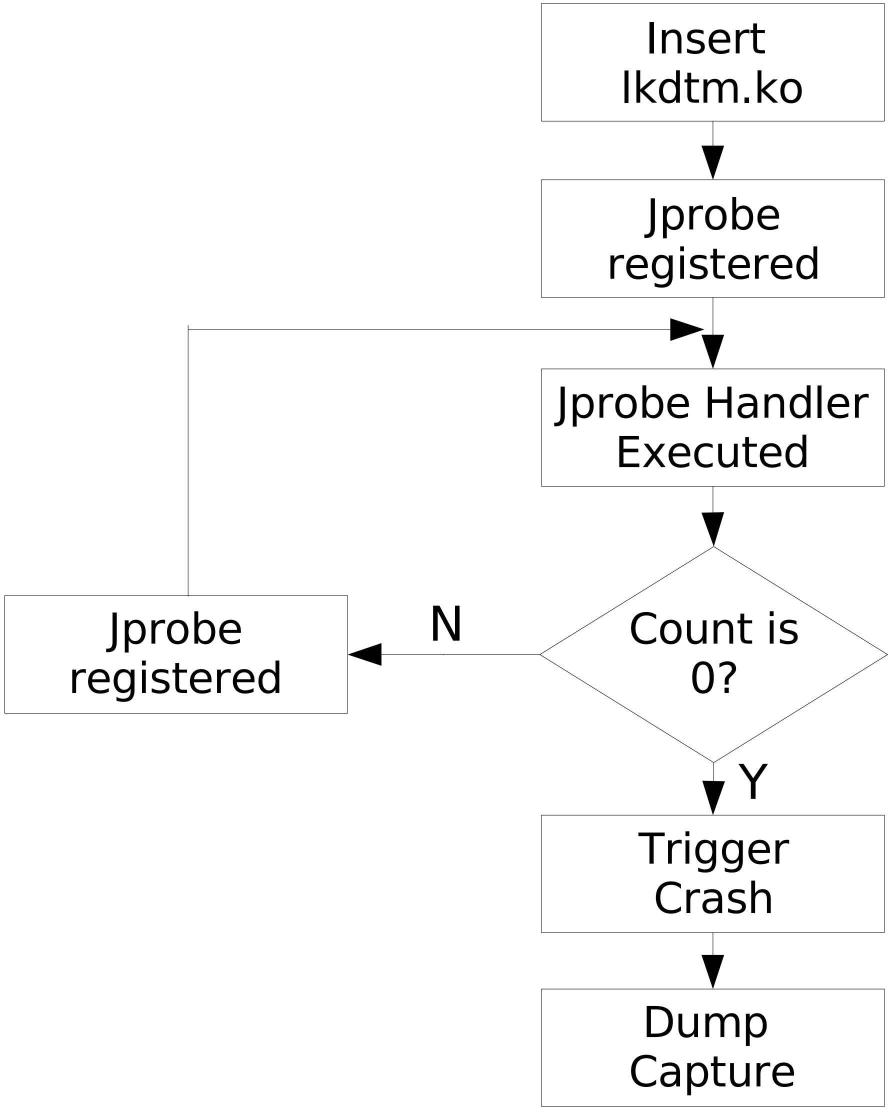
Kdump Testing (LKDTM)

- Important to test kdump in various scenarios
  - Crash in interrupt context
  - Crash in exception context
  - Invoke crash from various kernel code path
- Linux Kernel Dump Test Module (LKDTM)
- LKDTM facilitates inserting crash points at various kernel code paths

LKDTM

- LKDTM is based on LKDTT
- LKDTT uses Generalized Kernel Hooks Infrastructure; not mainline
- One needs to patch kernel for LKDTT
- LKDTM uses jprobes infrastructure

LKDTM Cont'd



Insert lkdtm.ko

Jprobe registered

Jprobe Handler Executed

Count is 0?

N

Jprobe registered

Y

Trigger Crash

Dump Capture

LKDTM Crash Points

- IRQ Handling with IRQ disabled
  - Insert probe at __do_IRQ
- IRQ Handling with IRQ enabled
  - Insert probe at handle_IRQ_event
- BLOCK IO
  - Insert probe at ll_rw_block
- Timer Processing
  - Insert probe at hrtimer_start

Type of crash event and Usage

- Panic
- BUG
- Exception
- Stack Overflow
- Usage

```
#modprobe lkdtm cpoint_name=<> cpoint_type=<>
  [cpoint_count= {>0}] [recur_count={>0}]
```

Kdump Testing Automation

- Testing automated and test scripts merged with Linux Test Project (LTP)
- Scripts make use of LKDTM to trigger crash
- Currently works with SLES10 and RHEL5
- Usage

```
# ./setup

# ./master run.
```

Device Driver Initialization

- Continues to be a pain point
- Fix the problem at driver level
- Driver should reset the device in second kernel
- Command line parameter “reset_devices” can help driver

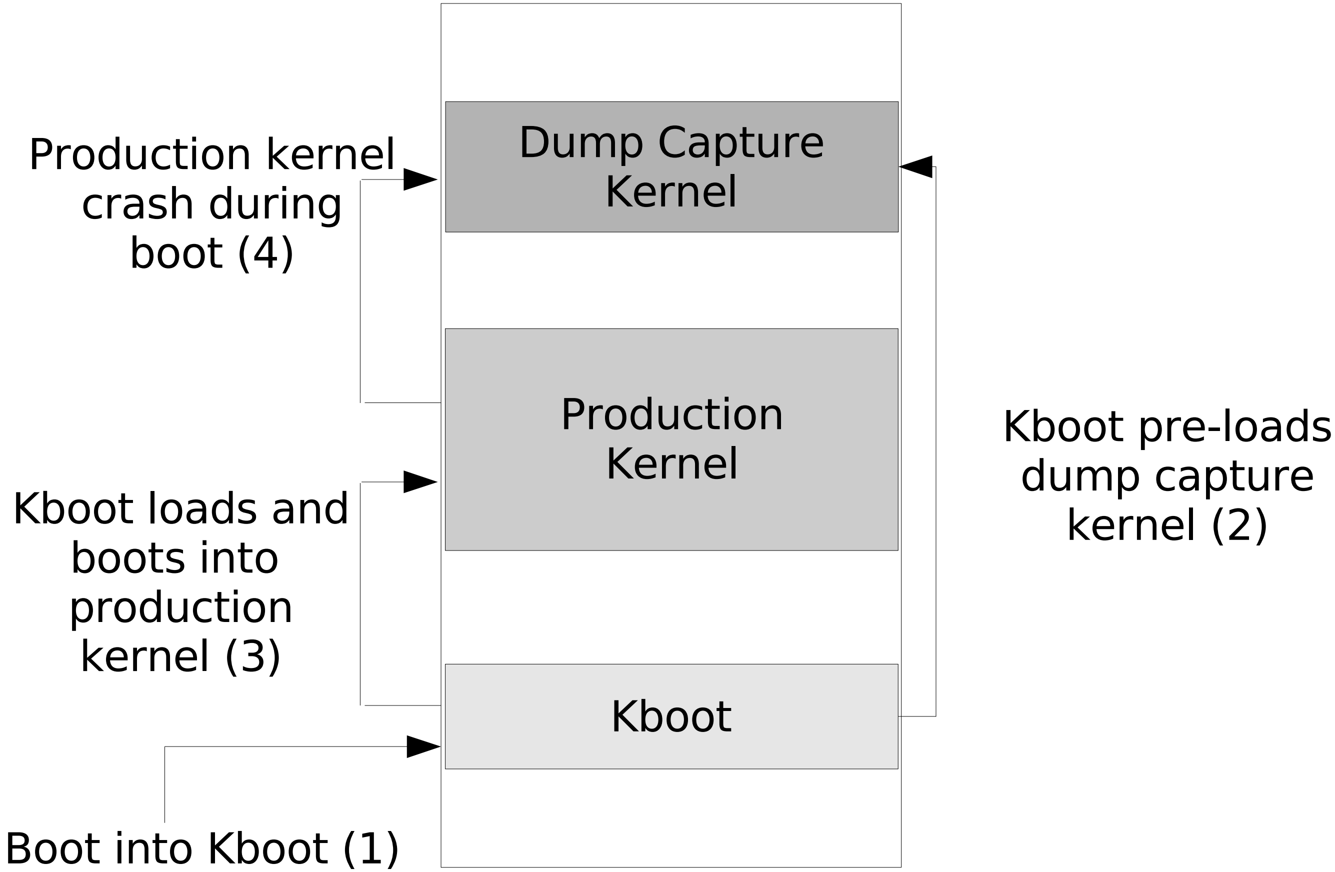
Device Driver Initialization (Cont'd)

- Drivers that have been fixed
  - aacraid; megaraid; mptsas; ibmveth; ibmvscsi
- Driver with reported pending issues
  - cciss

Device Driver Initialization (Cont'd)

- Possibly PCI Express hot reset functionality can be used to reset the device
- Use of EEH infrastructure on Power to reset the device?

Early boot crash dumping


Dump Capture Kernel
Production Kernel
Kboot
Production kernel crash during boot (4)
Kboot pre-loads dump capture kernel (2)
Kboot loads and boots into production kernel (3)
Boot into Kboot (1)

Questions?

Appendix

- makedumpfile usage:

  ```
  # makedumpfile [-c|-E] -d dump_level [-x vmlinux|-i mkdfinfo] /proc/vmcore dumpfile
  ```

  - Creating the compressed dump file (Readable only with **crash**)

    ```
    # makedumpfile  -cd 31  -x  vmlinux  /proc/vmcore  dumpfile
    ```

  - Creating a dumpfile in ELF format (Readable with **gdb** and **crash**)

    ```
    # makedumpfile  -Ed 31  -x  vmlinux  /proc/vmcore  dumpfile
    ```

  - Using a mkdfinfo file instead of a vmlinux file

    ```
    # makedumpfile  -cd 31  -i  mkdfinfo  /proc/vmcore  dumpfile
    ```

  - Creating a mkdfinfo file from a vmlinux file

    ```
    # makedumpfile  -g  mkdfinfo  -x  vmlinux
    ```

Legal Statement

-
-
-
- This work represents the view of the authors and does not necessarily represent the view of IBM, RedHat or NEC Soft.
-
-
-

Legal Statement Cont'd

- Fedora is a registered trademark of Red Hat in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.
- References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.
- This document is provided "AS IS" with no express or implied warranties. Use the information in this document at your own risk.