SUSE

Kernel Livedump

Lukáš Hruška <lhruska@suse.cz>

Kernel Live Patching Developer

Livedump

- Create a consistent image of memory without stopping or restarting the machine
- Debugging complex issues found only on machines which run HA services
- Initially introduced by Yoshida Maasanori in 2012

Feature details

- restart not required =)
- copy-on-write
- consistent state
- x86 arch
- multiple target storage support
- output format supported by crash, drgn, …

Implementation

- decomposition into subproblems
  - wrprotect
    - make memory r/o
    - make sure the handler is called only once per PFN
  - livedump-core
    - generic module for livedump
    - sysfs
    - list of possible output destinations
  - memdump
    - Dumps into a raw block device

Implementation

wrprotect diagram

```
-------------------------
|          2MB          |   PMD
-------------------------
    |       |       |
    split_large_pages
    |       |       |
    V       V       V
-------------------------
|  4KB  |  4KB  |  4KB  |   PTEs
-------------------------
    ^      \     /
    |       \   /
_--------    V V
|PRESENT|  -------------
|  R/W  | ..| X | X+1 | ..  PFNs
|  SW1  |  -------------
|  ...  |
---------

SW1             1: PFN was originally R/W   0: PFN was originally R/O       1:1 per PTE N:1 per PFN
pgbmp_original  1: PFN will be part of dump 0: PFN won't be part of dump    N:1 per PTE 1:1 per PFN
pgbmp_save      1: PFN was still not saved  0: PFN was already saved        N:1 per PTE 1:1 per PFN

correct_error_code? = PF_PROT && PF_WRITE && !PF_USER && !PF_RSVD && !PF_INSTR
get_pfn             = lookup_address(addr) | __pa(addr) << PAGE_SHIFT

PF -> correct_error_code? -1-> get_pfn -> pfn in pgbmp_original -1-> pfn in pgbmp_save -1-> save -> unprotect
                         \                                     \                      \            /
                          -0-> not_processed                    -0-> not_processed     ------0-----
```

Queue size problem

- memdump’s handler currently saves content of a page on queue
- minimal required size of PF queue
- during stop machine state there are sensitive pages added to the queue
- dumping method also use some data structures which triggers another PF
  - might fail the whole dumping process
- starting calibration of min. required pages * const (?)
  - wrprotect → write to wrprotected var → tracing write finish on selected storage
    - memdump tp: block:block_rq_complete (wrprotected var's PFN == tp.pfn)
- unlock as many as possible known PFNs having data needed to perform the write op (?)

PF handling variants

- sometimes small inconsistency in dumped data might be OK
- two possible versions:
  - 100% consistent variant with possible performance impact
    - active waiting for space on queue (?)
    - dynamically resizing the PF queue till -ENOMEM (?)
    - sweep and PFs sharing same queue with sweep heuristic - % to use (?)
  - possible inconsistent variant with minimal performance impact
- 100% consistent variant with minimal performance impact possible to not complete (?)

TODOs

- vmap area support (only correct synchronization needs to be solved)
- interrupt instead of stop-machine state
- analysis on impact of 4kB split of pages on performance (TLB overhead)
  - huge page support / PT restoration
- other target storage support (file, network, …)

SUSE

Thank you

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

Frankenstrasse 146

90461 Nürnberg

www.suse.com