Suspect the MM subsystem when your Linux system hung up!?

<Dec 1 2017>

NTT DATA INTELLILINK Corporation
Tetsuo Handa

5 characteristics of Linux kernel's memory management subsystem.

1. MM subsystem does _NOT_ guarantee forward progress of memory allocation requests.
2. Theoretical problems are _NOT_ addressed unless they actually occurred in real life.
3. Problems triggered by intentional, malicious, or stress tests are _NOT_ addressed.
4. Developers do _NOT_ pay attention to users who cannot identify the real culprit by themselves.
5. Problems which cannot attract other developers are _NOT_ addressed.

The trigger which made me be involved in MM subsystem.

- CVE-2013-4312 and CVE-2016-2847 which I by chance found while doing "git bisect".
- The whole story of these vulnerabilities and/or topics up to Linux 4.8 are described at http://I-love.SAKURA.ne.jp/The_OOM_CTF.html .
  - 4 hours will not be sufficient for explaining whole of this page.
  - But I need to explain lightly because it is a prerequisite for understanding of today's talk. Subsequent pages are a small portion of subsequent deployment.

Almost 3 years have elapsed since the revelation of the "too small to fail" memory-allocation rule.

- Many of regressions caused by introduction of the OOM reaper in Linux 4.6 have been fixed.
- My goal in Linux 4.15 is that "We can prove that the system does not fall into OOM livelock situation as long as the OOM killer is invoked."

Almost 3 years have elapsed since the revelation of the "too small to fail" memory-allocation rule. (cont.)

- Some of dependencies which can lead to silent "Unable to invoke the OOM killer" problem have been fixed.
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/mm/vmscan.c?id=db73ee0d463799223244e96e7b7eea73b4a6ec31
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/mm/page_alloc.c?id=e746bf730a76fe53b82c9e6b6da72d58e9ae3565
- On the other hand, there are some worries remaining. Today, I pick up two of them.

Worries 1

- The OOM killer is not invoked for allocation requests without __GFP_FS flag. Therefore, GFP_NOIO / GFP_NOFS allocation requests have possibility of hanging up the system.
  - We can reproduce such hang up using artificial stress, but that problem will not be addressed unless it is proven to occur using real workloads.
    - http://lkml.kernel.org/r/201703031948.CHJ81278.VOHSFFFOOLJQMt@I-love.SAKURA.ne.jp
  - But it is a too much request for averaged users to prove that their systems hung up due to this problem.

Worries 1 (cont.)

- In order to avoid silent hang up, Linux 4.9 got warn_alloc() calls which "synchronously" prints messages when a memory allocation request took more than 10 seconds.
  - But since it was confirmed that concurrent warn_alloc() calls can hang up the system, warn_alloc() was reverted in Linux 4.15-rc1.
    - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/mm/page_alloc.c?id=400e22499dd92613821374c8c6c88c7225359980

Worries 1 (cont.)

- I have been proposing a watchdog which extends khungtaskd so that the system can print useful information "asynchronously" without locking up the system.
  - http://lkml.kernel.org/r/1495331504-12480-1-git-send-email-penguin-kernel@I-love.SAKURA.ne.jp
  - http://lkml.kernel.org/r/1510833448-19918-1-git-send-email-penguin-kernel@I-love.SAKURA.ne.jp
  - But since OOM livelock is the least attractive domain, I'm stuck with zero advocate. I'm strongly seeking for developers/users who can join this discussion. That's why I'm speaking today.

Worries 2

- Since there is no report of OOM livelocks on nommu environments, there is no anti OOM livelock mechanism for nommu environments.
  - But it is possible that the reason of no report is simply that ordinary users are not aware (or not spent time for debugging).
- Therefore, I'm seeking for someone who can volunteer for testing whether OOM livelocks can occur on nommu environments.

MM subsystem wants your help.

- Current development of MM is geared towards machines with TB class RAM.
  - E.g. inserting cond_resched() in order to avoid soft lockups, parallelizing initialization of memory in order to save boot time.
- Machines with small amount of RAM and/or nommu environments are lost in oblivion.
  - E.g. silently remove setting of MMF_OOM_SKIP from nommu environment due to never heard of OOM livelock reports with nommu environments, a proposal for making SRCU always available (eliminate CONFIG_SRCU option) and rewrite the core code.
- Embedded Linux users, please pay attention and respond to MM changes.

Global IT Innovator