hwpoison

What is hwpoison?

Upcoming Intel CPUs have support for recovering from some memory errors (`MCA recovery`). This requires the OS to declare a page "poisoned", kill the processes associated with it and avoid using it in the future.

This patchkit implements the necessary infrastructure in the VM.

To quote the overview comment:

```
High level machine check handler. Handles pages reported by the
hardware as being corrupted usually due to a 2bit ECC memory or cache
failure.

This focusses on pages detected as corrupted in the background.
When the current CPU tries to consume corruption the currently
running process can just be killed directly instead. This implies
that if the error cannot be handled for some reason it's safe to
just ignore it because no corruption has been consumed yet. Instead
when that happens another machine check will happen.

Handles page cache pages in various states. The tricky part
here is that we can access any page asynchronous to other VM
users, because memory failures could happen anytime and anywhere,
possibly violating some of their assumptions. This is why this code
has to be extremely careful. Generally it tries to use normal locking
rules, as in get the standard locks, even if that means the
error handling takes potentially a long time.

Some of the operations here are somewhat inefficient and have non
linear algorithmic complexity, because the data structures have not
been optimized for this case. This is in particular the case
for the mapping from a vma to a process. Since this case is expected
to be rare we hope we can get away with this.
```

The code consists of a the high level handler in mm/memory-failure.c, a new page poison bit and various checks in the VM to handle poisoned pages.

The main target right now is KVM guests, but it works for all kinds of applications. KVM support requires a recent qemu-kvm release.

For the KVM use there was need for a new signal type so that KVM can inject the machine check into the guest with the proper address. This in theory allows other applications to handle memory failures too. The expection is that near all applications won't do that, but some very specialized ones might.

Failure recovery modes

There are two (actually three) modes memory failure recovery can be in:

vm.memory_failure_recovery sysctl set to zero:
: All memory failures cause a panic. Do not attempt recovery.

early kill
: (can be controlled globally and per process) Send SIGBUS to the application as soon as the error is detected This allows applications who can process memory errors in a gentle way (e.g. drop affected object) This is the mode used by KVM qemu.

late kill
: Send SIGBUS when the application runs into the corrupted page. This is best for memory error unaware applications and default Note some pages are always handled as late kill.

User control

vm.memory_failure_recovery

: See sysctl.txt

vm.memory_failure_early_kill

: Enable early kill mode globally

PR_MCE_KILL

: Set early/late kill mode/revert to system default

: arg1: PR_MCE_KILL_CLEAR:

Revert to system default

arg1: PR_MCE_KILL_SET:

arg2 defines thread specific mode

PR_MCE_KILL_EARLY:
Early kill
PR_MCE_KILL_LATE:
Late kill
PR_MCE_KILL_DEFAULT
Use system global default

Note that if you want to have a dedicated thread which handles the SIGBUS(BUS_MCEERR_AO) on behalf of the process, you should call prctl(PR_MCE_KILL_EARLY) on the designated thread. Otherwise, the SIGBUS is sent to the main thread.

PR_MCE_KILL_GET

return current mode

Testing

- madvise(MADV_HWPOISON, ...) (as root) - Poison a page in the process for testing
- hwpoison-inject module through debugfs `/sys/kernel/debug/hwpoison/`

  corrupt-pfn
  Inject hwpoison fault at PFN echoed into this file. This does some early filtering to avoid corrupted unintended pages in test suites.

  unpoison-pfn
  Software-unpoison page at PFN echoed into this file. This way a page can be reused again. This only works for Linux injected failures, not for real memory failures.

  Note these injection interfaces are not stable and might change between kernel versions

  corrupt-filter-dev-major, corrupt-filter-dev-minor

  Only handle memory failures to pages associated with the file system defined by block device major/minor. -1U is the wildcard value. This should be only used for testing with artificial injection.

  corrupt-filter-memcg

  Limit injection to pages owned by memgroup. Specified by inode number of the memcg.

  Example:

  ```
  mkdir /sys/fs/cgroup/mem/hwpoison

  usemem -m 100 -s 1000 &
  echo `jobs -p` > /sys/fs/cgroup/mem/hwpoison/tasks

  memcg_ino=$(ls -id /sys/fs/cgroup/mem/hwpoison | cut -f1 -d' ')
  echo $memcg_ino > /debug/hwpoison/corrupt-filter-memcg

  page-types -p `pidof init`   --hwpoison  # shall do nothing
  page-types -p `pidof usemem` --hwpoison  # poison its pages
  ```

  corrupt-filter-flags-mask, corrupt-filter-flags-value

  When specified, only poison pages if ((page_flags & mask) == value). This allows stress testing of many kinds of pages. The page_flags are the same as in /proc/kpageflags. The flag bits are defined in include/linux/kernel-page-flags.h and documented in Documentation/admin-guide/mm/pagemap.rst

- Architecture specific MCE injector

  x86 has mce-inject, mce-test

  Some portable hwpoison test programs in mce-test, see below.

References

http://halobates.de/mce-lc09-2.pdf
Overview presentation from LinuxCon 09
git://git.kernel.org/pub/scm/utils/cpu/mce/mce-test.git
Test suite (hwpoison specific portable tests in tsrc)
git://git.kernel.org/pub/scm/utils/cpu/mce/mce-inject.git
x86 specific injector

Limitations

- Not all page types are supported and never will. Most kernel internal objects cannot be recovered, only LRU pages for now.

--- Andi Kleen, Oct 2009