New issue

Jump to bottom

Memory Leak in crossbeam-queue ArrayQueue? (Latest git only, ver0.2.3 is not effected) #539

Closed **YoshikiTakashima** opened this issue on Aug 3, 2020 · 6 comments

Labels bug crossbeam-channel crossbeam-queue

YoshikiTakashima commented on Aug 3, 2020

Hi.

I'm Yoshi, a PhD student at CMU. We are currently evaluating a project that aims to automatically synthesize test cases for Rust libraries.

We ran the synthesized test cases with Miri, and it reports what appears to be a memory leak. The minimum example is:

```
{
let x_1 = ArrayQueue::<i32>::new(1);
}//destruct x_1


error: Undefined Behavior: incorrect layout on deallocation: allocation has size 64 and alignment 8, but gave size 16 and alignment 8
```

It looks like the drop destructor is not erasing enough memory.

You can also catch this with by running `cargo miri test` inside crossbeam-queue. One of your manually written test cases (`capacity`) will induce the same behavior.

The first commit to have this issue is 2180b820c846e5e6f2f11ccd1238198a5861dc08. You can confirm this by running `cargo miri test` on 1 commit before. It will not terminate because Miri is extremely slow, but you can see that it gets past `capacity` test case.

As for the root cause, I suspect that the drop function needs to be modified to match the changes to `new` , but I am yet to get a PR up.

In case this is intended behavior, or you would prefer if I focused on other parts of the code, please let me know.

Thanks
~Yoshi

cynecx commented on Aug 3, 2020 • edited Contributor

Yes, I just realized that the changes in #500 are not correct in terms of equivalence compared to the original code behavior.

The PR (#500) replaces:

```
let mut v = Vec::<Slot<T>>::with_capacity(cap);
```

with

```
let mut v: Vec<Slot<T>> = (0..cap)
    .map(|i| {
        // Set the stamp to `{ lap: 0, mark: 0, index: i }`.
        Slot {
            stamp: AtomicUsize::new(i),
            msg: UnsafeCell::new(MaybeUninit::uninit()),
        }
    })
    .collect();
```

The latter does **not** exactly allocate `cap` slot entries as `with_capacity` would do, instead with `cap=1` the current `Vec + collect` implementation would allocate 4 entries. When the `drop` implementation [1] would run, it would try to reconstruct the `Vec` with the `cap` as capacity although it is not the correct capacity, hence causing the memory leak.

[1]:

crossbeam/crossbeam-queue/src/array_queue.rs
Lines 426 to 429 in 5a68889

```
426     // Finally, deallocate the buffer, but don't run any destructors.
427     unsafe {
428         Vec::from_raw_parts(self.buffer, 0, self.cap);
429     }
```

caelunshun commented on Aug 3, 2020 • edited Contributor

This is fixed with #533. ~~Note that this issue was also present before #500 was merged, since Vec::with_capacity doesn't guarantee that the resulting allocation will have the exact capacity that was requested.~~

cynecx commented on Aug 3, 2020 • edited Contributor

@caelunshun Ah I didn't see that PR.

> since Vec::with_capacity doesn't guarantee that exact capacity.

The std doc says: "The vector will be able to hold exactly capacity elements without reallocating. If capacity is 0, the vector will not allocate.". Ähm, doesn't that kinda say that the capacity of the Vec will be exactly `cap` ?

caelunshun commented on Aug 3, 2020 — Contributor

Ah, looks like I was mistaken there. You're right, then; this issue was introduced in #500.

cynecx commented on Aug 3, 2020 — Contributor

@caelunshun Mhh. Maybe a simpler fix for #500 would have been something like:

```
// Instead of
let mut v = Vec::<Slot<T>>::with_capacity(cap);
// do this
let mut v = Vec::<MaybeUninit<Slot<T>>>::with_capacity(cap);
```

So that the actual initialization loop can be preserved.

YoshikiTakashima commented on Aug 3, 2020 — Author

Hi **@cynecx**, **@caelunshun**.

Thank you for the quick response.

I'll merge #533 into my fork and see if I can find more interesting issues.

Thank you for the help!

YoshikiTakashima closed this as completed on Aug 3, 2020

---

taiki-e mentioned this issue on Sep 6, 2020

Change crossbeam-channel's license to MIT OR Apache-2.0 #537

Merged

This was referenced on Oct 9, 2020

Run Miri on CI #578

Merged

Add advisory for UB in crossbeam-channel 0.4.3 rustsec/advisory-db#425

Merged

taiki-e added bug **crossbeam-channel** **crossbeam-queue** labels on Nov 4, 2020

CjS77 added a commit to tari-project/broadcast_channel that referenced this issue on Jun 7

`bug: update crossbeam-channel` ··· ✕ `659ca76`

CjS77 mentioned this issue on Jun 7

bug: update crossbeam-channel tari-project/broadcast_channel#4

Merged

CjS77 added a commit to tari-project/broadcast_channel that referenced this issue on Jun 7

`bug: update crossbeam-channel` ··· ✕ `60f9e2c`

CjS77 added a commit to tari-project/broadcast_channel that referenced this issue on Jun 7

`bug: update crossbeam-channel` ··· ✓ `eaa8b1a`

CjS77 added a commit to tari-project/broadcast_channel that referenced this issue on Jun 7

`bug: update crossbeam-channel (#4)` ··· ✓ `8aa6dd2`

Assignees

No one assigned

Labels

bug **crossbeam-channel** **crossbeam-queue**

Milestone

No milestone

Development

No branches or pull requests

4 participants