New issue

Jump to bottom

Zip may call __iterator_get_unchecked twice with the same index #82291

Closed SkiFire13 opened this issue on Feb 19, 2021 · 6 comments · Fixed by #82292

Labels C-bug I-unsound P-critical T-libs

SkiFire13 commented on Feb 19, 2021 Contributor

Here `__iterator_get_unchecked` is called for potential side effects until `self.index == self.a.size()`, ignoring however that it could have already been called in `next_back` with those indexes.

rust/library/core/src/iter/adapters/zip.rs
Lines 200 to 208 in 0148b97

```
200     } else if A::may_have_side_effect() && self.index < self.a.size() {
201         let i = self.index;
202         self.index += 1;
203         // match the base implementation's potential side effects
204         // SAFETY: we just checked that `i` < `self.a.len()`
205         unsafe {
206             self.a.__iterator_get_unchecked(i);
207         }
208         None
```

Playground link that demonstrates how this can be exploited to get two mutable references to the same data and cause an use-after-free bug.

1

SkiFire13 added the C-bug label on Feb 19, 2021

jackh726 added I-unsound T-libs labels on Feb 19, 2021

rustbot added the I-prioritize label on Feb 19, 2021

SkiFire13 mentioned this issue on Feb 19, 2021

Prevent specialized ZipImpl from calling `__iterator_get_unchecked` twice with the same index #82292

Merged

hameerabbasi commented on Feb 19, 2021 Contributor

Assigning `P-critical` as part of the WG-prioritization discussion on Zulip.

hameerabbasi added P-critical and removed I-prioritize labels on Feb 19, 2021

SkiFire13 mentioned this issue on Feb 19, 2021

Fix underflow in specialized ZipImpl::size_hint #82289

Merged

the8472 commented on Feb 19, 2021 Contributor

The gift that keeps on giving 🤣

Maybe it is time to get rid of `TrustedRandomAccess` as it exists today and replace it with a slightly safer (but still unsafe) variant? One that imposes the additional requirement on the caller that it must bring the iterator back into a safe state after it is done iterating.

The upside would be eliminating a lot of code in `ZipImpl` and enabling the same optimization for `vec::IntoIter<T> where T: Drop` and similar sources.
While the downside would be that external iteration (i.e. `for` loops and manual calls to `next()`) would cease to benefit from `TrustedRandomAccess` optimizations and only internal iteration methods (including `for_each`) would continue to do so.

SkiFire13 commented on Feb 19, 2021 Contributor Author

I don't know, I feel like that would just make more complex to upheld the invariants. IMO a good improvement over the current `__iterator_get_unchecked` would be separating forward and backward iteration, this way it becomes easier to keep track of the state and I think would also make it possible to implement them for `vec::IntoIter<T> where T: Drop`

the8472 commented on Feb 19, 2021 Contributor

I don't see how separating the forward and backward state would help with a drop implementation. The issue is that the source (`IntoIter`) currently does not have access to that state, instead the consumer drives the iteration by direct access through `__iterator_get_unchecked` and never informs the source about it.

SkiFire13 commented on Feb 19, 2021 • edited ▾ Contributor Author

It would help because it allows the source iterator to keep track of its state by updating it when the method is called. Anyway, looks like someone already tried this approach back in 2016 and it resulted in worse optimizations, but maybe LLVM got better in the meantime, or maybe that implementation could have been better.

Anyway I don't think this is the right place to discuss this, a topic on zulip would probably be better.

the8472 commented on Feb 20, 2021 Contributor

> a topic on zulip would probably be better.

https://rust-lang.zulipchat.com/#narrow/stream/219381-t-libs/topic/Improving.20TrustedRandomAccess.20and.20its.20Zip.20specialization/near/227088066

JohnTitor added a commit to JohnTitor/rust that referenced this issue on Mar 6, 2021

`Rollup merge of rust-lang#82292 - SkiFire13:fix-issue-82291, r=m-ou-se` ··· `fbdc900`

JohnTitor added a commit to JohnTitor/rust that referenced this issue on Mar 6, 2021

`Rollup merge of rust-lang#82292 - SkiFire13:fix-issue-82291, r=m-ou-se` ··· `3242730`

m-ou-se added a commit to m-ou-se/rust that referenced this issue on Mar 6, 2021

`Rollup merge of rust-lang#82292 - SkiFire13:fix-issue-82291, r=m-ou-se` ··· `5e91bab`

Dylan-DPC-zz pushed a commit to Dylan-DPC-zz/rust that referenced this issue on Mar 6, 2021

`Rollup merge of rust-lang#82292 - SkiFire13:fix-issue-82291, r=m-ou-se` ··· `4c7e3ce`

bors closed this as completed in `1d5b2dc` on Mar 7, 2021

---

the8472 mentioned this issue on Jan 28

Use TrustedRandomAccess for loop desugaring #93243

Closed

the8472 mentioned this issue on Feb 22

Stacked borrows fails on `{ChunksMut,ChunksExactMut}::__iterator_get_unchecked()` #94231

Closed

Assignees

No one assigned

Labels

C-bug I-unsound P-critical T-libs

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Prevent specialized ZipImpl from calling `__iterator_get_unchecked` twice with the same index
SkiFire13/rust

5 participants