[Core] Improve hash collision avoidance in prefix caching #12621

New issue

Merged

comaniac merged 1 commit into `vllm-project:main` from `russellb:hash-collisions` 5 days ago

russellb commented last week Member

Prefix caching makes use of Python's built-in `hash()` function. As of
Python 3.12, the behavior of `hash(None)` has changed to be a
predictable constant value. This makes it more feasible that someone
could try exploit hash collisions.

The impact of a collision would be using cache that was generated using
different content. Given knowledge of prompts in use and predictable hashing
behavior, someone could intentionally populate the cache using a prompt
known to collide with another prompt in use. There doesn't seem to be much
value to an attacker in doing this, but it's certainly not ideal, and could
interfere with the accuracy of results for another user.

The invasiveness of this fix should be weighed against the severity of the
issue to determine whether this is worth fixing.

Reviewers

- mgoin ✓
- comaniac ✓
- WoosukKwon
- robertgshaw2-redhat
- njhill
- ywang96
- alexm-redhat
- zhuohan123
- youkaichao
- DarkLight1337

Assignees

No one assigned

Labels

ready v1

Projects

None yet

Milestone

No milestone

Using a hashing algorithm that is less prone to collision (like sha256, for
example) would be the best way to avoid the possibility of a collision.
However, it would have an impact to both performance and memory footprint.
An alternative is to continue to use `hash()`, but make it much more difficult
to predict the hash value.

What we want is that the starting hash value is randomized, which is the
behavior we got here prior to Python 3.12. An easy fix is to use a
string. Here we use `'None'` to still make it clear we're starting from
nothing, but with a string we'll get a different hash value each time
vllm runs. Note that within a given run, the value will remain the same.
This restores the safer hashing behavior from before.

Thank you very much to **@kexinoh** for reporting this concern privately so
that it could be evaluated for its severity prior to our decision to fix
this as a security enhancement.

The commit that changed this behavior for Python 3.12 is here:

- python/cpython@ `432117c`
- **gh-99540: Constant hash for _PyNone_Type to aid reproducibility** python/cpython#99541

Signed-off-by: Russell Bryant rbryant@redhat.com

Development

Successfully merging this pull request may close these issues.

None yet

7 participants

russellb requested review from **WoosukKwon**, **robertgshaw2-redhat**, **njhill**, **ywang96**, **comaniac**, **alexm-redhat**, **zhuohan123** and **youkaichao** as code owners last week

github-actions bot commented last week

👋 Hi! Thank you for contributing to the vLLM project.
Just a reminder: PRs would not trigger full CI run by default. Instead, it would only run `fastcheck` CI which starts running only a small and essential subset of CI tests to quickly catch errors. You can run other CI tests on top of those by going to your `fastcheck` build on Buildkite UI (linked in the PR checks section) and unblock them. If you do not have permission to unblock, ping `simon-mo` or `khluu` to add you in our Buildkite org.

Once the PR is approved and ready to go, your PR reviewer(s) can run CI to test the changes comprehensively before merging.

To run CI, PR reviewers can do one of these:

- Add `ready` label to the PR
- Enable auto-merge.

🚀

comaniac approved these changes last week View reviewed changes

comaniac left a comment Collaborator

LGTM

comaniac added the ready label last week

comaniac enabled auto-merge (squash) last week

mgoin commented last week Member

Maybe this is still an issue with other hashers, but is there a reason why we don't use blake3 for hashing in the text case? It is currently what we use in the multimodal case for performance reasons AFAIK https://github.com/vllm-project/vllm/blob/e3f7ff65e7a6c08cd354f7f333bce543a4f0607e/vllm/multimodal/hasher.py

comaniac commented last week Collaborator

> Maybe this is still an issue with other hashers, but is there a reason why we don't use blake3 for hashing in the text case? It is currently what we use in the multimodal case for performance reasons AFAIK https://github.com/vllm-project/vllm/blob/e3f7ff65e7a6c08cd354f7f333bce543a4f0607e/vllm/multimodal/hasher.py

AFAIK it's just because `hash` has decent performance for short texts, but yeah we could benchmark blake3 in this scenario and see if we should use it here too.

👍 2

mergify bot commented last week

This pull request has merge conflicts that must be resolved before it can be
merged. Please rebase the PR, **@russellb**.

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/syncing-a-fork

mergify bot added needs-rebase v1 labels last week

auto-merge was automatically disabled 5 days ago
Head branch was pushed to by a user without write access

russellb force-pushed the `hash-collisions` branch from **`3dc5cf0`** to **`f0efd13`** 5 days ago Compare

mergify bot removed the needs-rebase label 5 days ago

russellb requested a review from **DarkLight1337** as a code owner 5 days ago

mergify bot commented 5 days ago

This pull request has merge conflicts that must be resolved before it can be
merged. Please rebase the PR, **@russellb**.

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/syncing-a-fork

mergify bot added the needs-rebase label 5 days ago

russellb force-pushed the `hash-collisions` branch from **65443ba** to **98b121a** 5 days ago Compare

mergify bot removed the needs-rebase label 5 days ago

mgoin approved these changes 5 days ago View reviewed changes

`[Core] Improve hash collision avoidance in prefix caching` ... ✓ a032ee8

russellb force-pushed the `hash-collisions` branch from **98b121a** to **a032ee8** 5 days ago Compare

comaniac merged commit **73b35cc** into `vllm-project:main` 5 days ago View details

46 checks passed

---

markmc commented 4 days ago Contributor

Does this change what's noted in the design doc for v1 ?

> Note 2: The above hash key structure is not 100% collision free. Theoretically it's still possible for the different prefix tokens to have the same hash value, but this should be nearly impossible to happen. Of course, contributions are welcome if you have an awesome idea to eliminate collusion entirely.

comaniac commented 4 days ago Collaborator

> Does this change what's noted in the design doc for v1 ?

No this PR mainly deals with the None case.

👍 1

russellb commented 4 days ago Member Author

> Does this change what's noted in the design doc for v1 ?
>
> > Note 2: The above hash key structure is not 100% collision free. Theoretically it's still possible for the different prefix tokens to have the same hash value, but this should be nearly impossible to happen. Of course, contributions are welcome if you have an awesome idea to eliminate collusion entirely.

I think that's still accurate. Collisions are still possible, but what I was trying to avoid here is making it feasible to predict those collisions because of predictable hashing behavior.

👍 1

nFunctor commented 4 days ago Contributor

Thanks for this PR **@russellb** . Perhaps if you have time for a semi-related question...

We observed a fairly weird effect during some intense generation, and perhaps it is related to this PR. The task involved a generation of summaries over an extensive batch, both prefix caching (the system prompts are fairly extensive) and n-gram speculative decoding were on. We ended up with "mixed summaries", eg a phrase like "London is the capital of" got replaced with "London is my favourite" (and both phrases existed in the inputs batch but for different batch indices).

I first thought that something went wrong with the speculative worker but I now start to think that perhaps it could have been due to prefix cache? Would you think the same? Apologies for the scarce details, unfortunately I am not yet sure if I can reproduce the exact circumstances of that generation experiment.

kexinoh commented 4 days ago via email

I am the discoverer of the problem, and we construct the phenomenon exactly as you say. Then I also need to add that hash(None) is not really a random value before Python3.12,

but rather a memory address value (which makes it less random). In the case of Python VM, None may be set to 0.

...

👍 1

russellb commented 3 days ago Member Author

> Thanks for this PR **@russellb** . Perhaps if you have time for a semi-related question...
>
> We observed a fairly weird effect during some intense generation, and perhaps it is related to this PR. The task involved a generation of summaries over an extensive batch, both prefix caching (the system prompts are fairly extensive) and n-gram speculative decoding were on. We ended up with "mixed summaries", eg a phrase like "London is the capital of" got replaced with "London is my favourite" (and both phrases existed in the inputs batch but for different batch indices).
>
> I first thought that something went wrong with the speculative worker but I now start to think that perhaps it could have been due to prefix cache? Would you think the same? Apologies for the scarce details, unfortunately I am not yet sure if I can reproduce the exact circumstances of that generation experiment.

Can you clarify if this was observed prior to this PR going in, or after? I want to make sure I didn't cause a regression.

If it was before, it's possible that you experienced a hash collision. Prior to this PR, using Python 3.12, that collision would be easily reproducible. After this PR, it would not. The conditions for a collision should be different every time vllm is run.

That doesn't remove the possibility for collisions, though. That would take more work. It's very interesting to hear that you may have observed this without going after it intentionally!

nFunctor commented 2 days ago Contributor

@russellb the issue happened with a docker build of vllm (0.6.5, and I believe all recent images run on 3.12) so it was observed before the PR. I have not done the tests with the nightly build/docker from source yet. Never seen such things happen outside docker in my python 3.11 venv.

I am not sure I can go into the significant detail about the setup where the bug was observed but here are some elements:

- AWQ checkpoint of Llama 3.1 70B instruct running lots of requests. The GPU KV cache is often near 100%. The server is queried by a collection of workers whose load can vary.
- Prefix caching, chunked prefill and n-gram speculative decoding on.
- The issue involved multiple entries in queue getting confused (content switch) at a common word combo ("London is").

I would add that in my experience AWQ/Marlin-powered models have a rare tendency to produce wrong answers that consist of repeating strings, up to max tokens (I hope to find time to document this issue at some point, it is not easy to reproduce). From what you say it should not be related at all, but thought I'd mention it as a known issue with the setup; the latter is overall prone to some numerical instability even without the hash collision.

👍 2

ilayathalapathy-3719 commented yesterday

@russellb Do we need a CVE for it and have you requested one already?

fxmarty-amd pushed a commit to fxmarty-amd/vllm that referenced this pull request yesterday

`[Core] Improve hash collision avoidance in prefix caching (vllm-proje…` `93b6e6e` ⋯

russellb commented yesterday Member Author

> **@russellb** Do we need a CVE for it and have you requested one already?

A CVE was assigned and is reflected here: GHSA-rm76-4mrf-v9r8