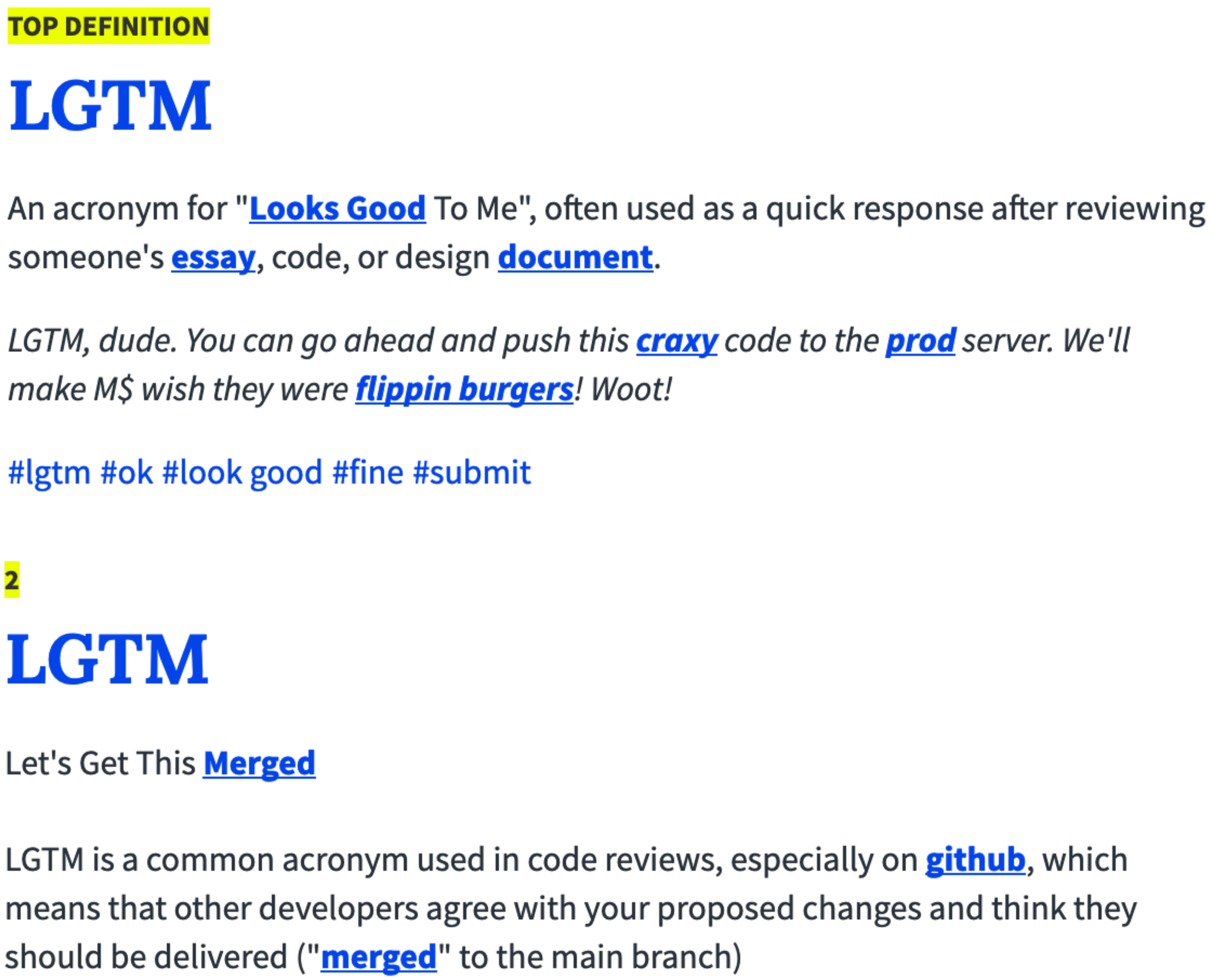
SHIP IT SHIP IT GOOD

17-356/17-766 SOFTWARE ENGINEERING FOR STARTUPS

isr institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

Zeeshan Lakhani

lgtm



TOP DEFINITION

LGTM

An acronym for "**Looks Good** To Me", often used as a quick response after reviewing someone's **essay**, code, or design **document**.

*LGTM, dude. You can go ahead and push this **craxy** code to the **prod** server. We'll make M$ wish they were **flippin burgers**! Woot!*

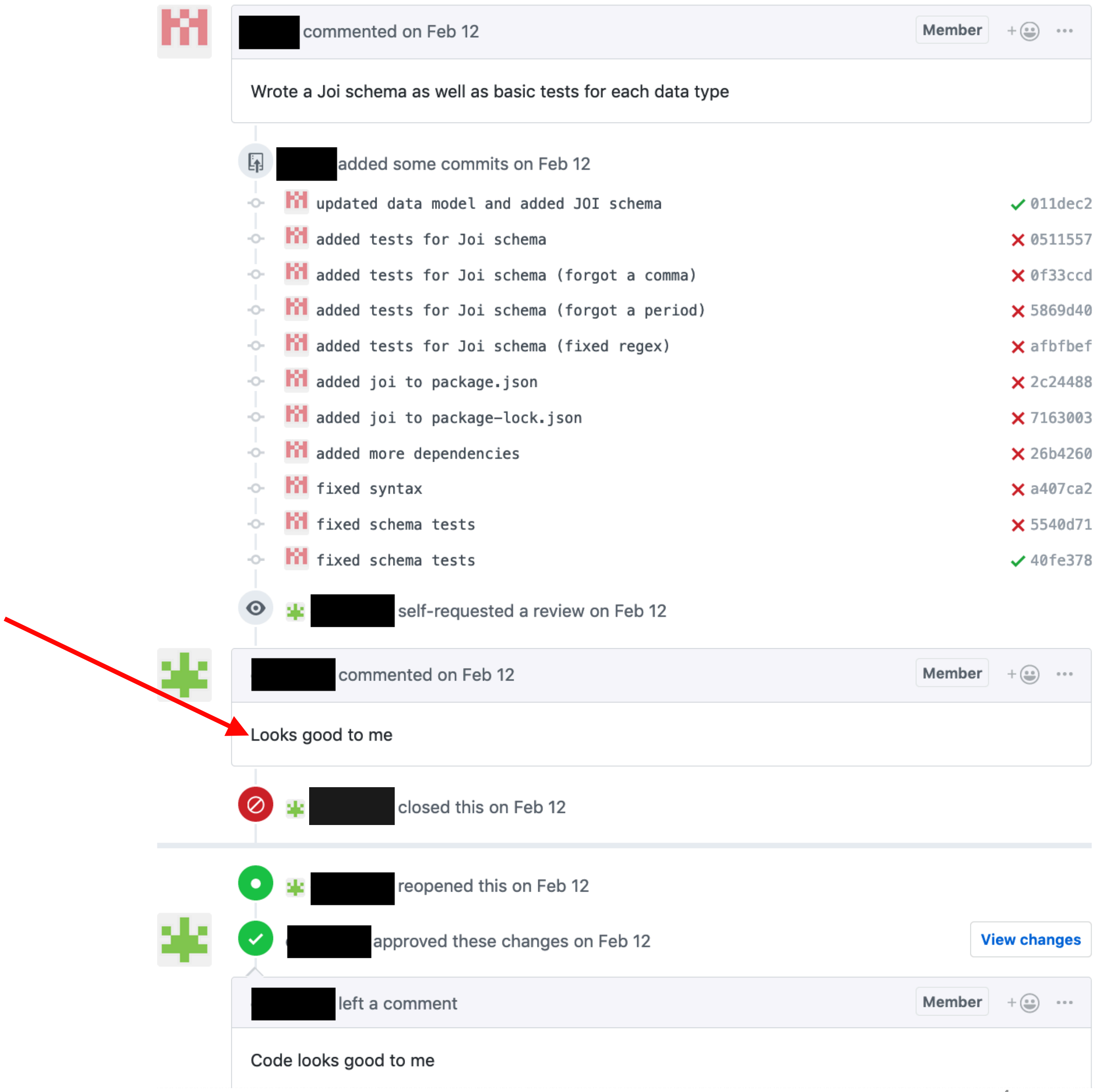
#lgtm #ok #look good #fine #submit

2

LGTM

Let's Get This **Merged**

LGTM is a common acronym used in code reviews, especially on **github**, which means that other developers agree with your proposed changes and think they should be delivered ("**merged**" to the main branch)


[redacted] commented on Feb 12 Member

Wrote a Joi schema as well as basic tests for each data type

[redacted] added some commits on Feb 12

```
updated data model and added JOI schema            ✓ 011dec2
added tests for Joi schema                         ✗ 0511557
added tests for Joi schema (forgot a comma)        ✗ 0f33ccd
added tests for Joi schema (forgot a period)       ✗ 5869d40
added tests for Joi schema (fixed regex)           ✗ afbfbef
added joi to package.json                          ✗ 2c24488
added joi to package-lock.json                     ✗ 7163003
added more dependencies                            ✗ 26b4260
fixed syntax                                       ✗ a407ca2
fixed schema tests                                 ✗ 5540d71
fixed schema tests                                 ✓ 40fe378
```

[redacted] self-requested a review on Feb 12

[redacted] commented on Feb 12 Member

Looks good to me

[redacted] closed this on Feb 12

[redacted] reopened this on Feb 12

[redacted] approved these changes on Feb 12 View changes

[redacted] left a comment Member

Code looks good to me

institute for SOFTWARE RESEARCH

review this?

globals

```
public static int LONG_WORD_LENGTH = 5;
public static String longestWord;

public static void countLongWords(List<String> words) {
    int n = 0;
    longestWord = "";
    for (String word: words) {
        if (word.length() > LONG_WORD_LENGTH) ++n;
        if (word.length() > longestWord.length()) longestWord = word;
    }
    System.out.println(n);
}
```

return results, not effects

studies on code review

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft

Amiangshu Bosu*, Michaela Greiler†, and Christian Bird†
*Department of Computer Science
University of Alabama, Tuscaloosa, Alabama

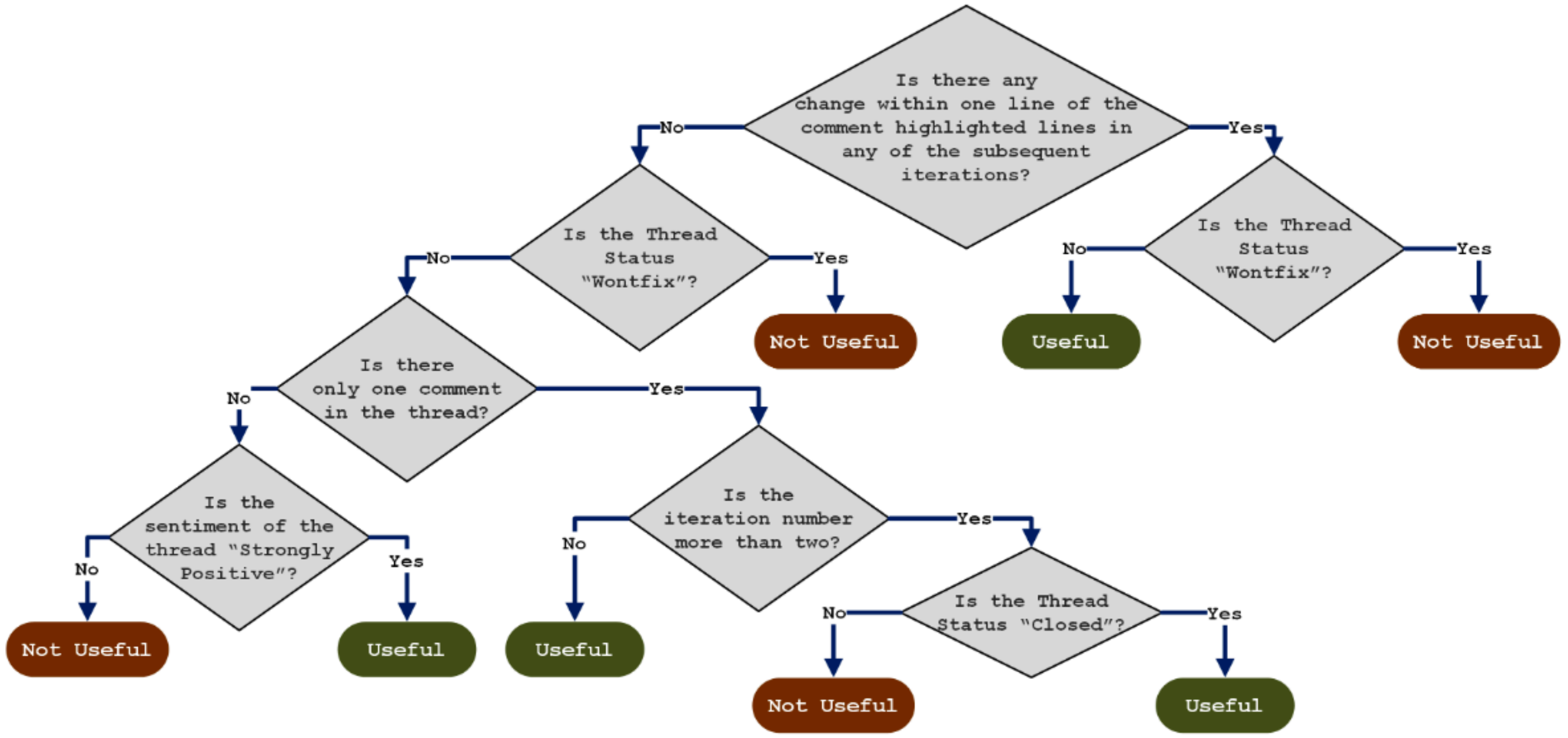
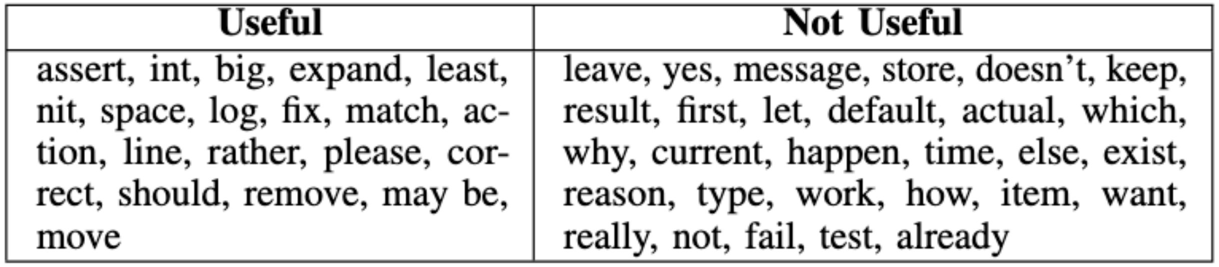

Is there any change within one line of the comment highlighted lines in any of the subsequent iterations?
No
Yes
Is the Thread Status "Wontfix"?
No
Yes
Not Useful
Is the Thread Status "Wontfix"?
No
Yes
Useful
Not Useful
Is there only one comment in the thread?
No
Yes
Is the sentiment of the thread "Strongly Positive"?
No
Yes
Not Useful
Useful
Is the iteration number more than two?
No
Yes
Useful
Is the Thread Status "Closed"?
No
Yes
Not Useful
Useful

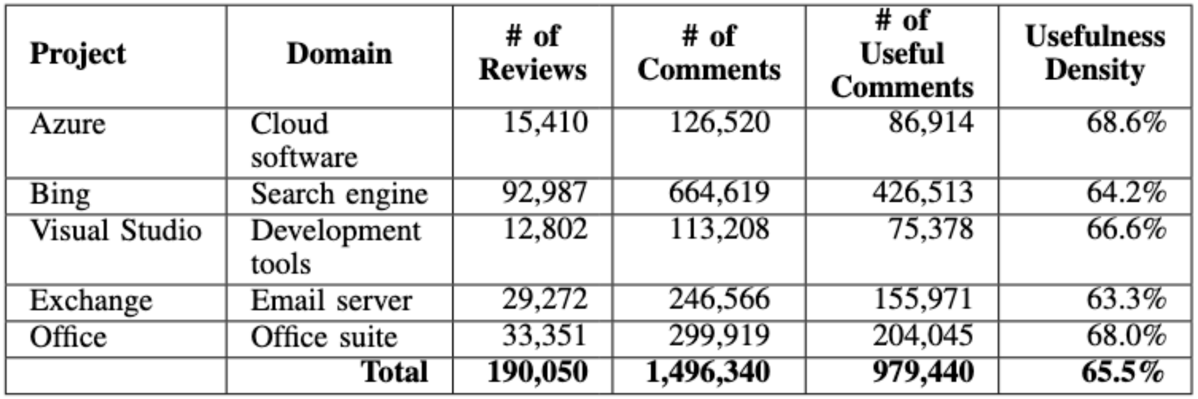
Fig. 5: Decision Tree Model to Classify Useful Comments

TABLE I: Keywords distribution

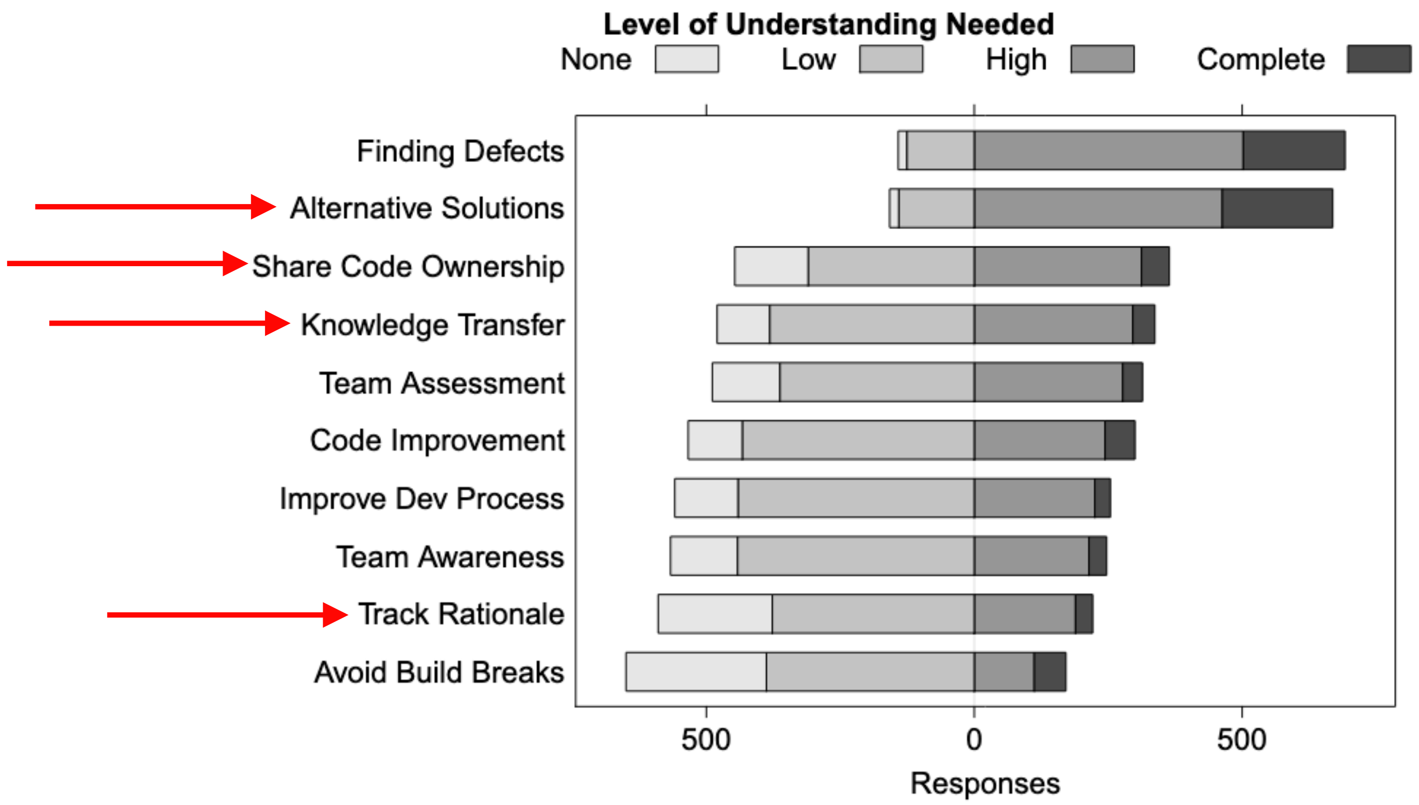


Useful	Not Useful
assert, int, big, expand, least, nit, space, log, fix, match, action, line, rather, please, correct, should, remove, may be, move	leave, yes, message, store, doesn't, keep, result, first, let, default, actual, which, why, current, happen, time, else, exist, reason, type, work, how, item, want, really, not, fail, test, already

TABLE II: Comment usefulness density



Project	Domain	# of Reviews	# of Comments	# of Useful Comments	Usefulness Density
Azure	Cloud software	15,410	126,520	86,914	68.6%
Bing	Search engine	92,987	664,619	426,513	64.2%
Visual Studio	Development tools	12,802	113,208	75,378	66.6%
Exchange	Email server	29,272	246,566	155,971	63.3%
Office	Office suite	33,351	299,919	204,045	68.0%
	Total	**190,050**	**1,496,340**	**979,440**	**65.5%**



Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review

code review @ google

Introduced to “force developers to write code that other developers could understand”

3 benefits found:

- checking the consistency of style and design
- ensuring adequate tests
- improving security by making sure no single developer can commit arbitrary code without oversight

Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. International Conference on Software Engineering

code review flow @ google

- creating
- previewing
- commenting
- addressing feedback
- approving (now it’s “lgtm”)

code breakdowns @ google

- distance (geographical vs organizational)
- social interactions (tone & power)
- review subject (i.e. design vs technical subject)
- context (urgent change vs “nice to have”)
- customization (i.e. arbitrary requirements)

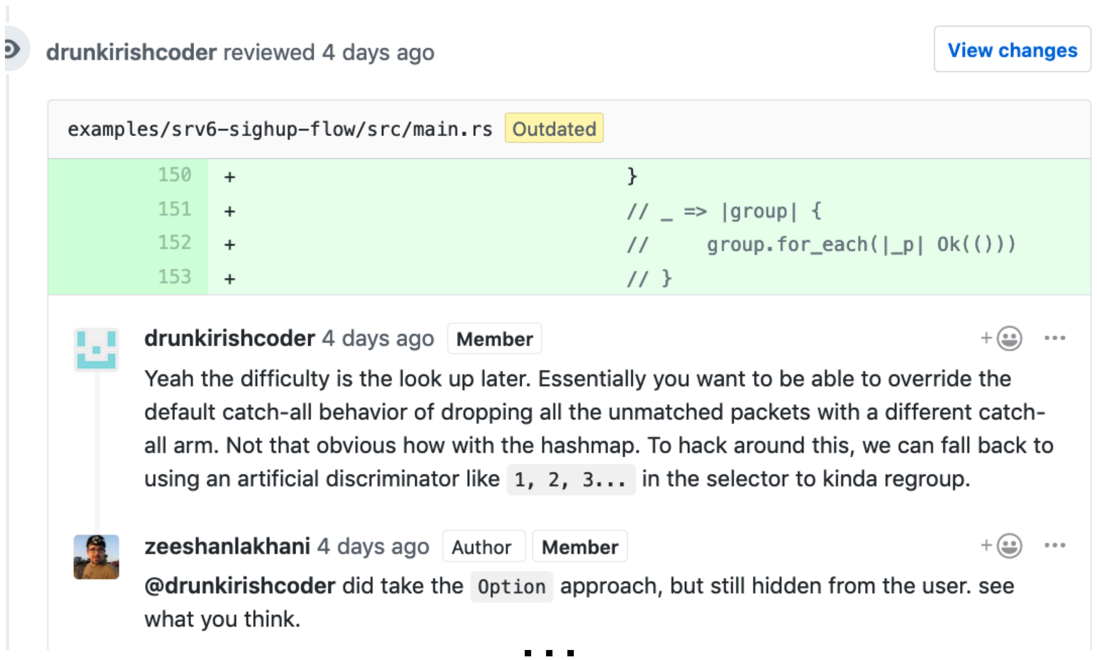
Finding 5. *Despite years of refinement, code review at Google still faces breakdowns. These are mostly linked to the complexity of the interactions that occur around the reviews. Yet, code review is strongly considered a valuable process by developers, who spend around 3 hours a week reviewing.*

the cost of code review?

- review usefulness is negatively correlated with the size of code review
- significant time spent
- longer review time, less time & consistency to incorporate feedback
- can stall and affect other features/issues and, therefore, other team members

personal/related examples

something new



drunkirishcoder reviewed 4 days ago

View changes

examples/srv6-sighup-flow/src/main.rs Outdated

```
+                        }
+                        // _ => |group| {
+                        //     group.for_each(|_p| Ok(()))
+                        // }
```

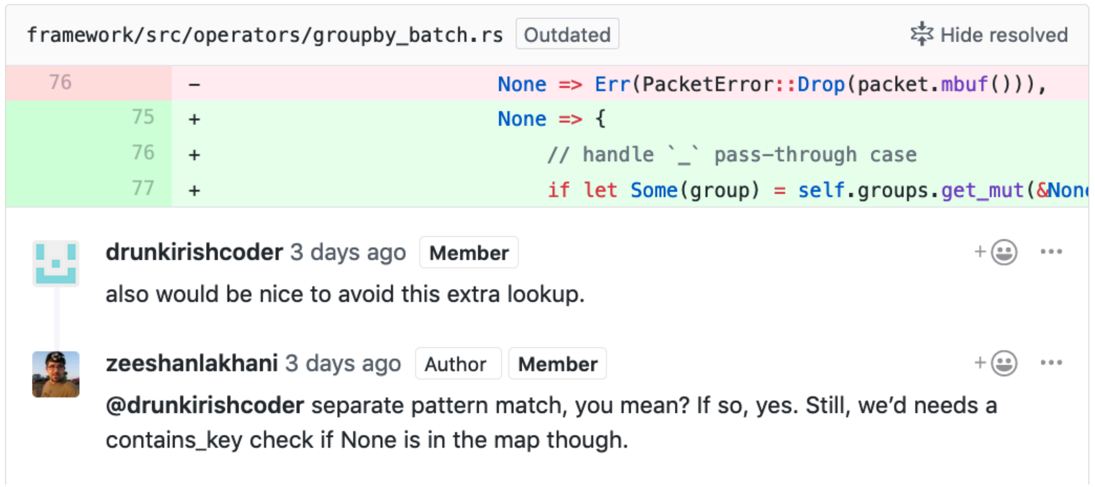
drunkirishcoder 4 days ago Member

Yeah the difficulty is the look up later. Essentially you want to be able to override the default catch-all behavior of dropping all the unmatched packets with a different catch-all arm. Not that obvious how with the hashmap. To hack around this, we can fall back to using an artificial discriminator like `1, 2, 3...` in the selector to kinda regroup.

zeeshanlakhani 4 days ago Author Member

@drunkirishcoder did take the `Option` approach, but still hidden from the user. see what you think.

...



framework/src/operators/groupby_batch.rs Outdated

Hide resolved

```
-                None => Err(PacketError::Drop(packet.mbuf())),
+                None => {
+                    // handle `_` pass-through case
+                    if let Some(group) = self.groups.get_mut(&Non
```

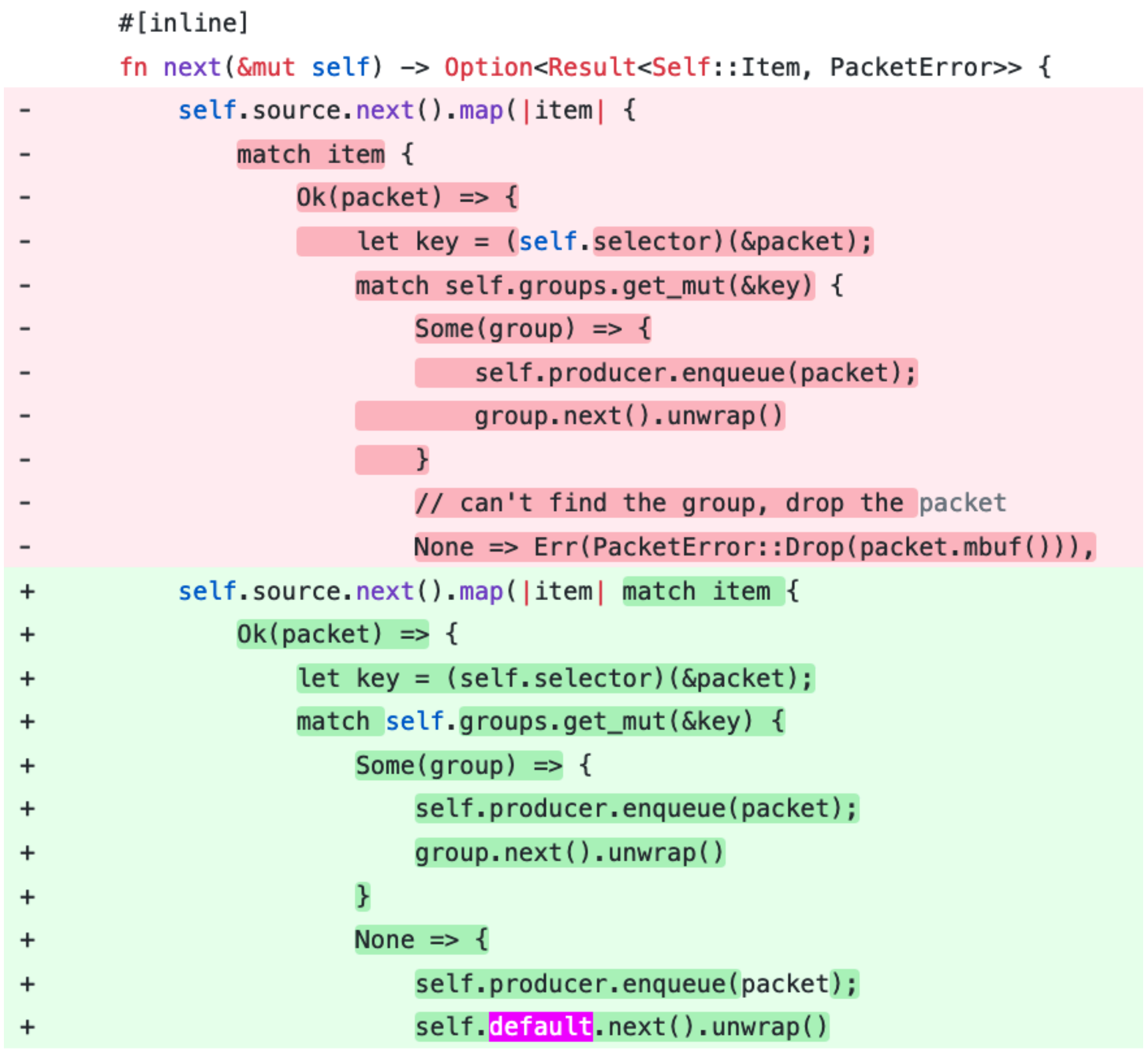
drunkirishcoder 3 days ago Member

also would be nice to avoid this extra lookup.

zeeshanlakhani 3 days ago Author Member

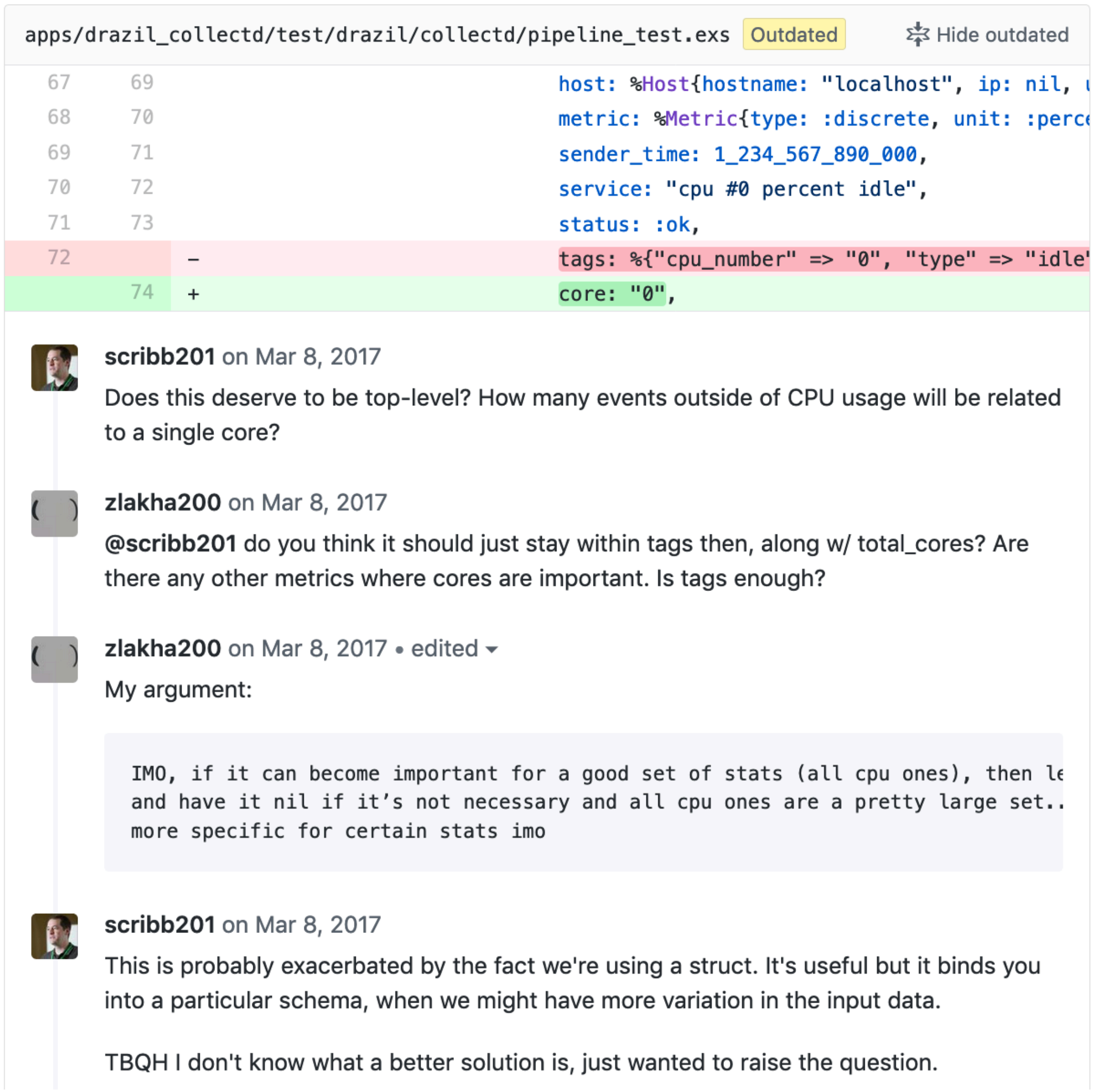
@drunkirishcoder separate pattern match, you mean? If so, yes. Still, we'd needs a contains_key check if None is in the map though.

isr institute for SOFTWARE RESEARCH



```
      #[inline]
      fn next(&mut self) -> Option<Result<Self::Item, PacketError>> {
-         self.source.next().map(|item| {
-             match item {
-                 Ok(packet) => {
-                     let key = (self.selector)(&packet);
-                     match self.groups.get_mut(&key) {
-                         Some(group) => {
-                             self.producer.enqueue(packet);
-                             group.next().unwrap()
-                         }
-                         // can't find the group, drop the packet
-                         None => Err(PacketError::Drop(packet.mbuf())),
+         self.source.next().map(|item| match item {
+             Ok(packet) => {
+                 let key = (self.selector)(&packet);
+                 match self.groups.get_mut(&key) {
+                     Some(group) => {
+                         self.producer.enqueue(packet);
+                         group.next().unwrap()
+                     }
+                     None => {
+                         self.producer.enqueue(packet);
+                         self.default.next().unwrap()
```

something old



apps/drazil_collectd/test/drazil/collectd/pipeline_test.exs Outdated Hide outdated

```
                                            host: %Host{hostname: "localhost", ip: nil, u
                                            metric: %Metric{type: :discrete, unit: :perce
                                            sender_time: 1_234_567_890_000,
                                            service: "cpu #0 percent idle",
                                            status: :ok,
-                                           tags: %{"cpu_number" => "0", "type" => "idle'
+                                           core: "0",
```

scribb201 on Mar 8, 2017

Does this deserve to be top-level? How many events outside of CPU usage will be related to a single core?

zlakha200 on Mar 8, 2017

@scribb201 do you think it should just stay within tags then, along w/ total_cores? Are there any other metrics where cores are important. Is tags enough?

zlakha200 on Mar 8, 2017 • edited

My argument:

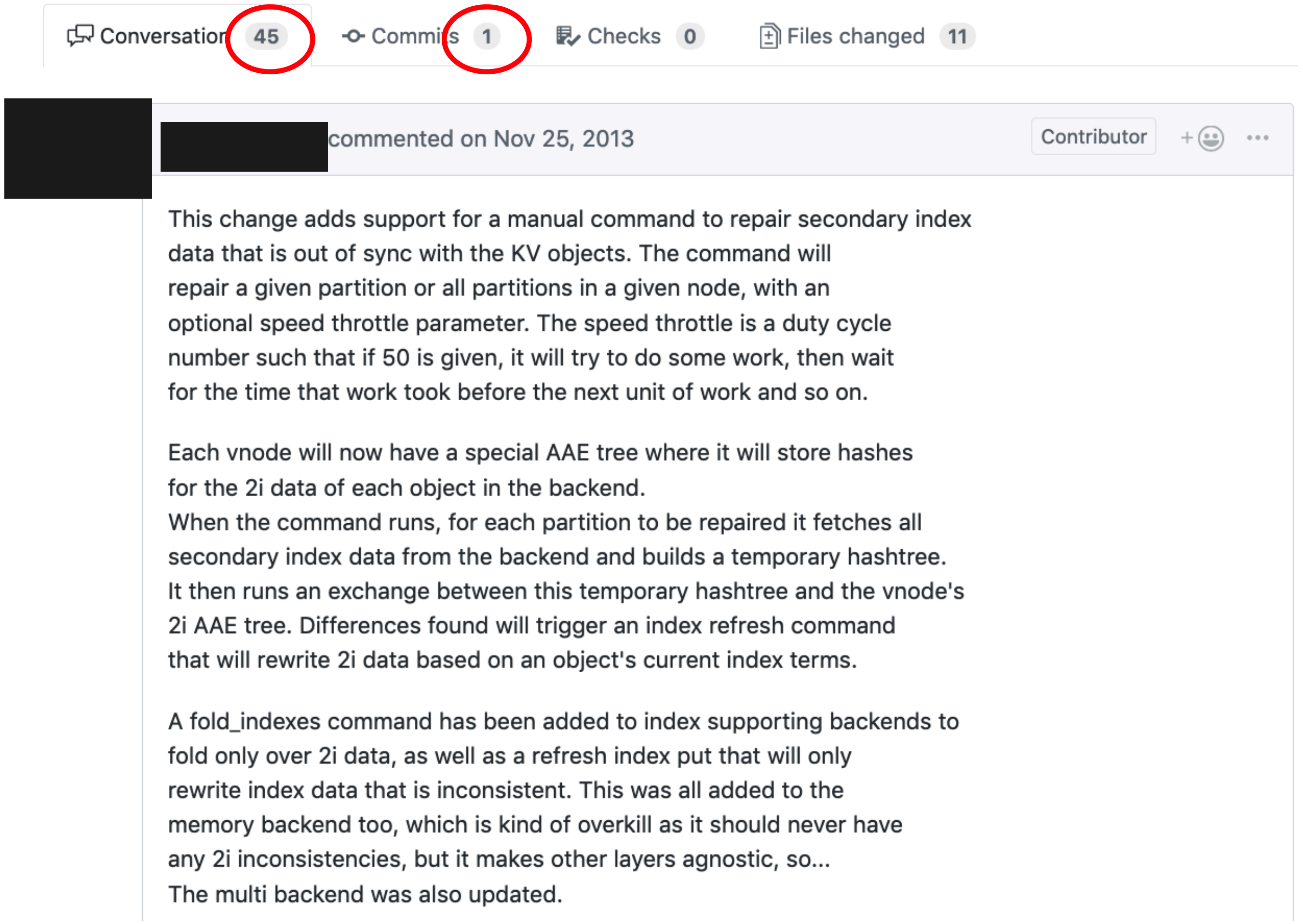
```
IMO, if it can become important for a good set of stats (all cpu ones), then le
and have it nil if it’s not necessary and all cpu ones are a pretty large set..
more specific for certain stats imo
```

scribb201 on Mar 8, 2017

This is probably exacerbated by the fact we're using a struct. It's useful but it binds you into a particular schema, when we might have more variation in the input data.

TBQH I don't know what a better solution is, just wanted to raise the question.

something borrowed

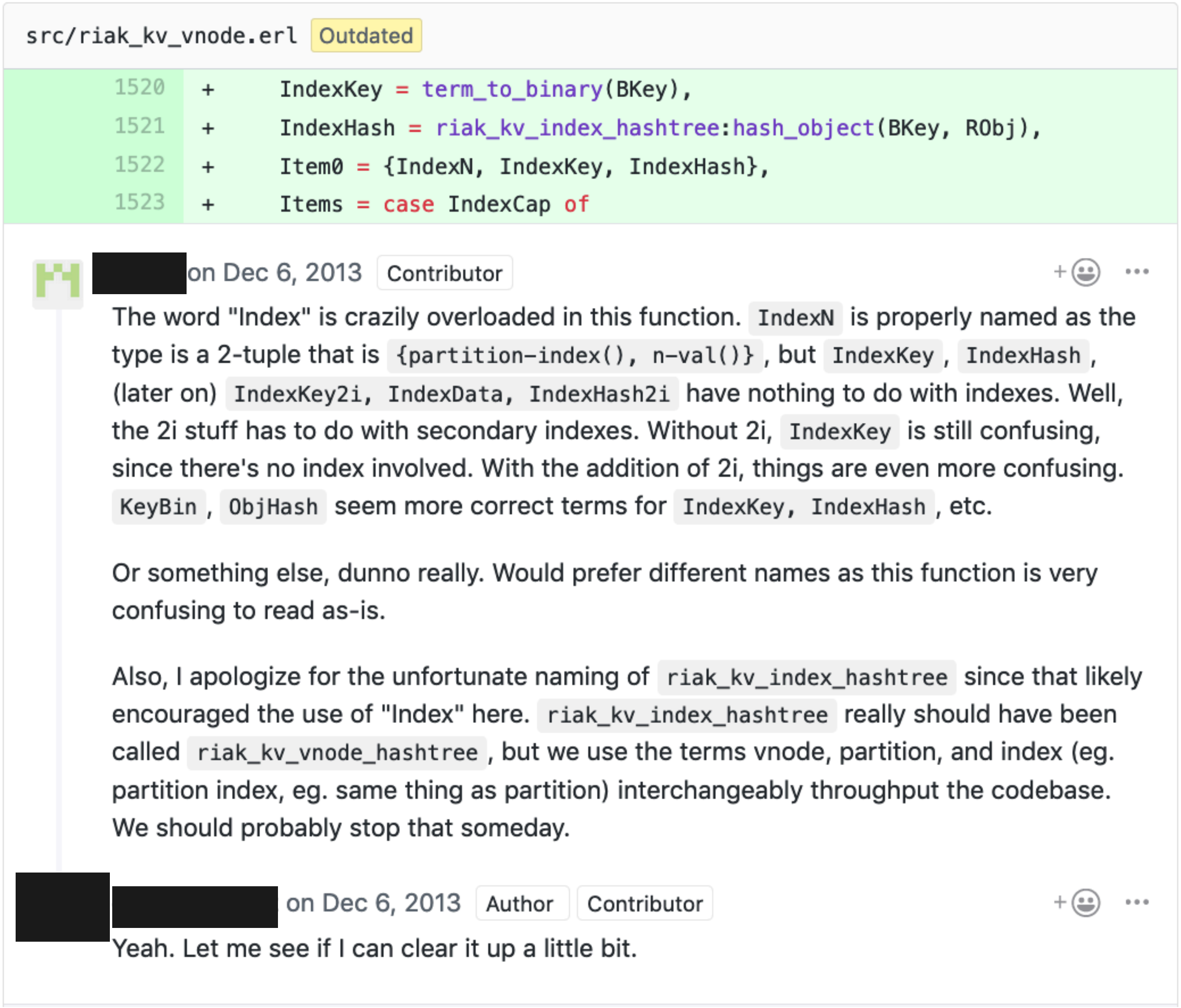


[redacted] [redacted] commented on Nov 25, 2013

This change adds support for a manual command to repair secondary index data that is out of sync with the KV objects. The command will repair a given partition or all partitions in a given node, with an optional speed throttle parameter. The speed throttle is a duty cycle number such that if 50 is given, it will try to do some work, then wait for the time that work took before the next unit of work and so on.

Each vnode will now have a special AAE tree where it will store hashes for the 2i data of each object in the backend.
When the command runs, for each partition to be repaired it fetches all secondary index data from the backend and builds a temporary hashtree. It then runs an exchange between this temporary hashtree and the vnode's 2i AAE tree. Differences found will trigger an index refresh command that will rewrite 2i data based on an object's current index terms.

A fold_indexes command has been added to index supporting backends to fold only over 2i data, as well as a refresh index put that will only rewrite index data that is inconsistent. This was all added to the memory backend too, which is kind of overkill as it should never have any 2i inconsistencies, but it makes other layers agnostic, so...
The multi backend was also updated.


src/riak_kv_vnode.erl Outdated

```
+       IndexKey = term_to_binary(BKey),
+       IndexHash = riak_kv_index_hashtree:hash_object(BKey, RObj),
+       Item0 = {IndexN, IndexKey, IndexHash},
+       Items = case IndexCap of
```

[redacted] on Dec 6, 2013 Contributor

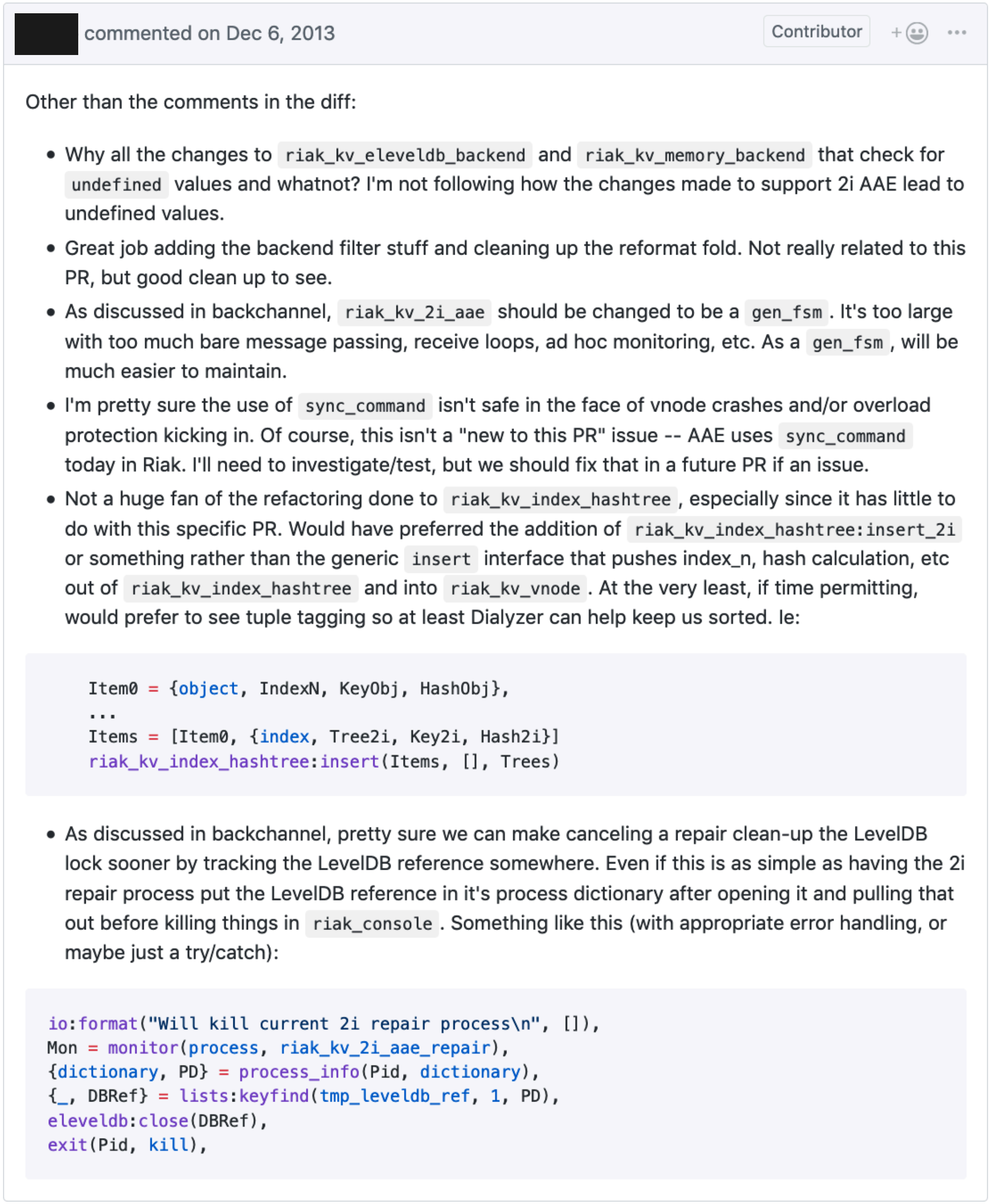
The word "Index" is crazily overloaded in this function. `IndexN` is properly named as the type is a 2-tuple that is `{partition-index(), n-val()}`, but `IndexKey`, `IndexHash`, (later on) `IndexKey2i, IndexData, IndexHash2i` have nothing to do with indexes. Well, the 2i stuff has to do with secondary indexes. Without 2i, `IndexKey` is still confusing, since there's no index involved. With the addition of 2i, things are even more confusing. `KeyBin`, `ObjHash` seem more correct terms for `IndexKey, IndexHash`, etc.

Or something else, dunno really. Would prefer different names as this function is very confusing to read as-is.

Also, I apologize for the unfortunate naming of `riak_kv_index_hashtree` since that likely encouraged the use of "Index" here. `riak_kv_index_hashtree` really should have been called `riak_kv_vnode_hashtree`, but we use the terms vnode, partition, and index (eg. partition index, eg. same thing as partition) interchangeably throughput the codebase. We should probably stop that someday.

[redacted] [redacted] on Dec 6, 2013 Author Contributor

Yeah. Let me see if I can clear it up a little bit.



[redacted] commented on Dec 6, 2013

Contributor

Other than the comments in the diff:

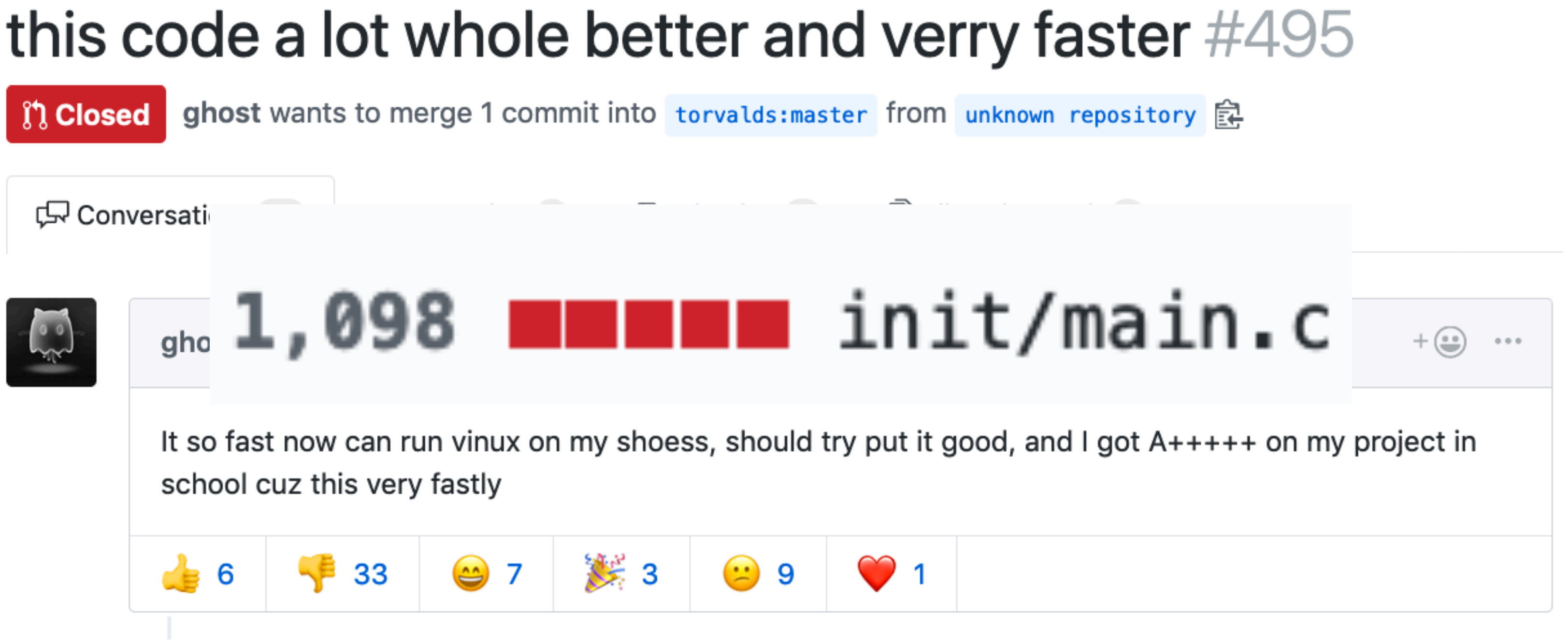
- Why all the changes to `riak_kv_eleveldb_backend` and `riak_kv_memory_backend` that check for `undefined` values and whatnot? I'm not following how the changes made to support 2i AAE lead to undefined values.
- Great job adding the backend filter stuff and cleaning up the reformat fold. Not really related to this PR, but good clean up to see.
- As discussed in backchannel, `riak_kv_2i_aae` should be changed to be a `gen_fsm`. It's too large with too much bare message passing, receive loops, ad hoc monitoring, etc. As a `gen_fsm`, will be much easier to maintain.
- I'm pretty sure the use of `sync_command` isn't safe in the face of vnode crashes and/or overload protection kicking in. Of course, this isn't a "new to this PR" issue -- AAE uses `sync_command` today in Riak. I'll need to investigate/test, but we should fix that in a future PR if an issue.
- Not a huge fan of the refactoring done to `riak_kv_index_hashtree`, especially since it has little to do with this specific PR. Would have preferred the addition of `riak_kv_index_hashtree:insert_2i` or something rather than the generic `insert` interface that pushes index_n, hash calculation, etc out of `riak_kv_index_hashtree` and into `riak_kv_vnode`. At the very least, if time permitting, would prefer to see tuple tagging so at least Dialyzer can help keep us sorted. Ie:

```
    Item0 = {object, IndexN, KeyObj, HashObj},
    ...
    Items = [Item0, {index, Tree2i, Key2i, Hash2i}]
    riak_kv_index_hashtree:insert(Items, [], Trees)
```

- As discussed in backchannel, pretty sure we can make canceling a repair clean-up the LevelDB lock sooner by tracking the LevelDB reference somewhere. Even if this is as simple as having the 2i repair process put the LevelDB reference in it's process dictionary after opening it and pulling that out before killing things in `riak_console`. Something like this (with appropriate error handling, or maybe just a try/catch):

```
io:format("Will kill current 2i repair process\n", []),
Mon = monitor(process, riak_kv_2i_aae_repair),
{dictionary, PD} = process_info(Pid, dictionary),
{_, DBRef} = lists:keyfind(tmp_leveldb_ref, 1, PD),
eleveldb:close(DBRef),
exit(Pid, kill),
```

something ewww



this code a lot whole better and verry faster #495

Closed ghost wants to merge 1 commit into `torvalds:master` from `unknown repository`

Conversati

gho

```
1,098 ■■■■■ init/main.c
```

It so fast now can run vinux on my shoess, should try put it good, and I got A+++++ on my project in school cuz this very fastly

👍 6 👎 33 😄 7 🎉 3 😕 9 ❤️ 1

my favorite issue

#KV679

what makes up a good review?

constructive code reviews!

the reviewee & the reviewer

where sync & async come together

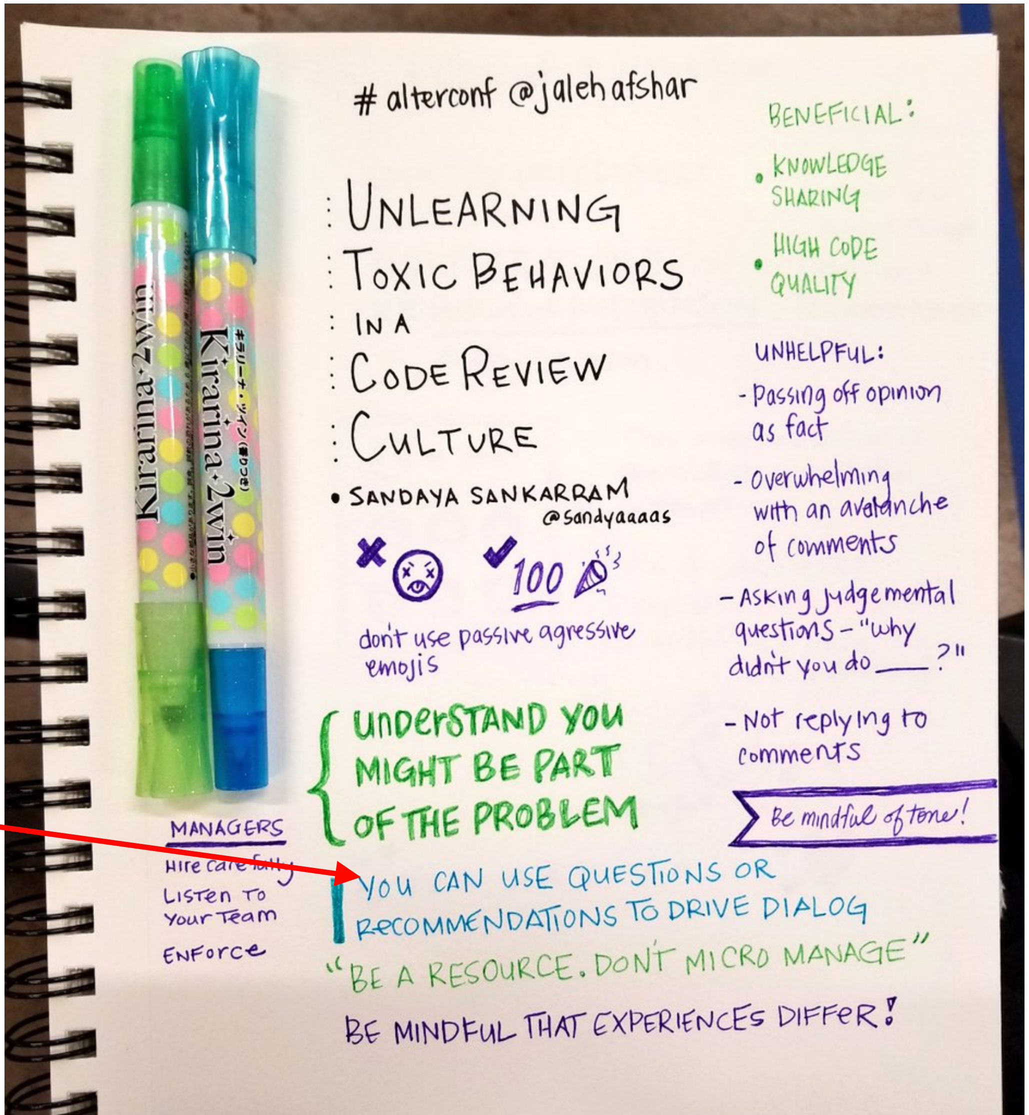
- gitter
- github issues / zenhub
- be distributed while in person
- commit histories and “git blame”

avoiding bad code review experiences

- linters/formatters (i.e. no nitpicking in reviews!)
- style guides
- static analysis
- **no** large commits
- comments, docstrings, naming
- valuable & readable commit messages
- coverage tools & ci

institute for SOFTWARE RESEARCH

unlearning bad practices



alterconf @jalehafshar

UNLEARNING TOXIC BEHAVIORS IN A CODE REVIEW CULTURE

• SANDAYA SANKARRAM @sandyaaaas

✗ ✓ 100

don't use passive agressive emojis

UNDERSTAND YOU MIGHT BE PART OF THE PROBLEM

YOU CAN USE QUESTIONS OR RECOMMENDATIONS TO DRIVE DIALOG

"BE A RESOURCE. DON'T MICRO MANAGE"

BE MINDFUL THAT EXPERIENCES DIFFER!

MANAGERS

Hire carefully

Listen to your Team

Enforce

BENEFICIAL:

- KNOWLEDGE SHARING
- HIGH CODE QUALITY

UNHELPFUL:

- Passing off opinion as fact
- Overwhelming with an avalanche of comments
- Asking judgemental questions - "why didn't you do ___?"
- Not replying to comments

Be mindful of tone!

exercise!

based on Small-Group Code Reviews For Education by Philip Guo

goto: https://github.com/CMU-17-356/codereview