CUNY Academic Commons - Bug #9974

Network options cache is messing things up

2018-06-28 02:07 PM - Boone Gorges

Status:	Resolved	**Start date:**	2018-06-28
Priority name:	Normal	**Due date:**	
Assignee:	Boone Gorges	**% Done:**	0%
Category name:	WordPress (misc)	**Estimated time:**	0.00 hour
Target version:	1.13.8		

Description

There seems to be a bug affecting some network options and the notoptions cache in get_network_option(). Namely, certain options that are **not** empty are still having their option values written into this cache, with the result that empty values are being returned from get_site_option() incorrectly. I don't have time to debug this at the moment, so I've put a hotfix in place that causes the notoptions cache to be skipped (0 == 1 on line 1222 of wp-includes/option.php). This came up while investigating #9970, and I'm almost certain it's related to #9949.

Ray, I'll be on the road for a few days. If you get a few minutes to try debugging this, it would be helpful. Thank you!!

Related issues:

Related to CUNY Academic Commons - Bug #10075: Problem with domain mapping/re...	**Resolved**	**2018-07-30**
Related to CUNY Academic Commons - Bug #10087: Site Maintenance Message	**Duplicate**	**2018-08-01**
Has duplicate CUNY Academic Commons - Bug #10049: Files not uploading to site...	**Duplicate**	**2018-07-24**

History

#1 - 2018-07-02 04:14 PM - Raymond Hoh

Forgot to update this thread last Friday.

I tried to duplicate the problem on the dev site, but couldn't.

Here's the code I was testing:

```
add_action( 'get_header', function() {
    if ( empty( $_GET['ray-option-test'] ) ) {
        return;
    }

    add_filter( 'default_site_option_ray_not_exist', 'ray_dump_notoptions' );
    add_filter( 'default_site_option_admin_email', 'ray_dump_notoptions' );

    get_site_option( 'ray_not_exist' );
    var_dump( get_site_option( 'upload_filetypes' ) );
    die();
}, 999 );

function ray_dump_notoptions( $retval ) {
    $notoptions_key = "1:notoptions";
    $notoptions = wp_cache_get( $notoptions_key, 'site-options' );
    var_dump( $notoptions );
    return $retval;
}
```

One thing I found completely weird while looking into #9949 was the object cache for an item was different from the expected result on production.

In the WPMU Domain Mapping plugin, I made a mod to cache domain mapping results from the DB:
https://github.com/cuny-academic-commons/cac/blob/1.13.x/wp-content/plugins/wordpress-mu-domain-mapping/domain_mapping.php#L596

On the New Labor Forum site, the following call:

wp_cache_get(2045, 'wpmu-dm')

Was not returning rows from the WPMU Domain Mapping DB table. Instead, it was returning the following object:

```
array(1) {
    [0]=> object(stdClass)#3521 (1) { ["GET_LOCK('blc_lock-blog-2045', 0)"]=> string(1) "1" }
```

```
}
```

The GET_LOCK portion is from the Broken Link Checker plugin. It uses MySQL's GET_LOCK and RELEASE_LOCK commands: https://github.com/cuny-academic-commons/cac/blob/1.13.x/wp-content/plugins/broken-link-checker/includes/wp-mutex.php#L19

I've since cleared the object cache -- wp_cache_delete(2045, 'wpmu-dm') -- and the cache is now returning the expected result.

Could this be related to Cavalcade running the Broken Link Checker cronjob and somehow stomping on different cache keys? Or could this be a conflict with running GET_LOCK / RELEASE_LOCK and sessions? Could this be the cause for the multiple DB connections problem as well?

#2 - 2018-07-03 12:41 PM - Boone Gorges

Thanks, Ray.

The differing cache values could indeed be related. I wonder if it's due to WP connecting differently to Memcached when launched via WP-CLI (as during a Cavalcade run) vs during a regular Apache visit. Could you maybe test that?

The Broken Link Catcher thing could definitely be linked to MySQL connections. It's worth digging into the Cavalcade logs - now that we have them! - to see if MySQL restarts from the last week correspond to BLC cron jobs. Whether this is connected to sessions, I'm less sure about, but it's possible, since we use the database to store session data. See #9865. As far as the cache corruption, it's not obvious to me that BLC would be a cause - seems more likely to be a symptom.

I'm currently away but I can do some more investigation along the above lines when I return, if you don't get around to it first.

#3 - 2018-07-10 10:33 AM - Boone Gorges

- Target version changed from 1.13.5 to 1.13.6

#4 - 2018-07-11 04:12 PM - Boone Gorges

- Subject changed from Network notoptions cache is messing things up to Network options cache is messing things up

An update that I ran into another version of this plugin today, which was related not to the notoptions cache, but to the regular cache fetch:

```
$cache_key = "$network_id:$option";
$value = wp_cache_get( $cache_key, 'site-options' );
```

For the time being, I've monkey-patched option.php to bypass the cache for site options.

It's worth noting that this specific incident corresponded with network or server issues at the GC. Is it possible that Memcache is somehow getting corrupted during these events?

#5 - 2018-07-11 04:14 PM - Boone Gorges

I've put some logging in place that will catch all attempts to put a value into the 'site-options' cache group. I'll check this periodically through the next day, to see if I can identify a plugin that might be messing with network cache values.

#6 - 2018-07-24 10:56 AM - Boone Gorges

- Category name set to WordPress (misc)

- Assignee changed from Raymond Hoh to Boone Gorges

- Target version changed from 1.13.6 to 1.13.7

I've been looking through the log but nothing is jumping out at me. Let's push this to another release and see if something crops up.

#7 - 2018-07-24 03:12 PM - Boone Gorges

- Has duplicate Bug #10049: Files not uploading to site gallery in JPG and PNG formats added

#8 - 2018-07-24 03:16 PM - Boone Gorges

#10049 is another instance of the same problem. In this case, the offending cache value was 1525292267:5956 (for cache key commons_wp:production:site-options:1:upload_filetypes
, which corresponds to get_site_option('upload_filetypes', 'site-options'). This value appears to be a timestamp followed by something else. This should be a clue, but I'm not sure how to follow it up.

I spent some time digging around in Memcached to identify an obvious problem, but the data all appears to be valid (ie, non-corrupt). The problem must be that there's a rogue plugin that's improperly setting cache values. (The correct value is in the database, and has not been changed.)

I'm going to start migrating network options like this over to 'pre_site_option_' filters, which are both better for performance and are not subject to this sort of cache problem. See https://github.com/cuny-academic-commons/cac/commit/b9dad3a5aae4ca4a5fddf290b55de392f2a6a2b4. Let's hope this problem doesn't begin to spread beyond network options, as it won't be possible to hardcode other kinds of values.

#9 - 2018-07-24 07:26 PM - Lisa Rhody

Thanks, Boone and Ray, for looking into it. I'll let the fellows know.
Best,
Lisa

#10 - 2018-07-24 08:16 PM - Raymond Hoh

> 1525292267:5956

This appears to be the syntax used for the _edit_lock post meta key. The integer after the colon is the user ID, so that's another clue.

#11 - 2018-07-25 03:08 PM - Boone Gorges

Hm - seems like it should be a helpful clue, but is in fact totally bewildering. It appears that the user in question has not logged in since 2018-06-14, so it's hard to see how a recent edit lock for that user would've triggered this recent problem. Also, edit_lock is stored in postmeta, so it's very hard to see how the cross-contamination would work. I wonder if there's something else in the system that uses a similar format?

I've pared back my logging a bit so that the next time this strikes, we might be able to find something....

#12 - 2018-07-30 07:40 PM - Boone Gorges

#10071 is another instance. Because the News site was down, I didn't leave the cache in place long enough to collect a bunch of data, but it appears that at least some of the site info stored in the cache was corrupt.

#10075 is yet another instance.

This is obviously unsustainable. For the time being, I'm disabling the object cache in WordPress. I will monitor to ensure that this doesn't result in too large a resource spike. I have some more theories about possible causes (see eg https://www.topsemtips.com/2011/09/php-and-memcached-returning-random-data/) but I do not have time at this moment to look into it.

#13 - 2018-07-30 07:41 PM - Boone Gorges

(Note to self that my working theory has to do with cache corruption between Memcached calls from CLI (Cavalcade) and Apache. Ray, if you have any brilliant ideas...)

#14 - 2018-07-30 08:06 PM - Raymond Hoh

I'm looking at Human Made's stack and they are using a fork of the Memcached object cache by tollmanz:
https://github.com/humanmade/hm-platform/blob/master/load.php#L106
https://github.com/humanmade/wordpress-pecl-memcached-object-cache/

Since their stack is also running Cavalcade, maybe we can try and use their memcached object cache fork to see if anything will magically clear up?

#15 - 2018-07-30 08:32 PM - Raymond Hoh

This is what Human Made had to say about their memcached fork:
https://engineering.hmn.md/platform/plugins/

Here are some choice quotes:

> The Human Made fork of the original object cache includes important changes, including a fundamental change to how autoloading options works in WordPress. Concurrency problems in WordPress usually boil down to this bug (commonly known as the “alloptions” bug) in some shape or form.

I think the alloptions bug is very applicable to us. Worth a try.

> We also use Zack Tollman’s Memcached object cache on PHP 7, however this does not include the alloptions fix. Be extremely careful running this without extensive testing.

Interesting to note that they revert to the original version by tollmanz on PHP7. Something to flag for late August.

#16 - 2018-07-30 08:37 PM - Boone Gorges

Ah, very interesting. If you want to throw up the fork, feel free, and we can keep an eye on things. (Though part of the problem with "keep an eye on things" is that we only know there's a problem after it's hit us.)

#17 - 2018-07-31 12:36 PM - Raymond Hoh

I've added Human Made's memcached object cache plugin and flushed the cache just now. Let's see what happens.

Something to be aware of is wp_cache_flush() can only be used via wp-cli:
https://github.com/humanmade/wordpress-pecl-memcached-object-cache/blob/master/object-cache.php#L283

So if a plugin is using wp_cache_flush(), it will not work.

#18 - 2018-07-31 02:10 PM - Raymond Hoh

I think our global $memcached_servers variable in cac-env-config.php might need to change from localhost to the IP where memcached is located. I see that an IP was commented out, but I'm not sure if that IP is still correct or not.

I did some testing and found that the cache from the web server and from wp-cli is different.

For my example, I loaded my activity page, so I know that the activity object cache will be set from the web server.

Then, I grabbed the IDs of one of the activity items and tried to fetch the cache:
- on the web server, I ran var_dump(wp_cache_get(508604, 'bp_activity'));, and
- in wp-cli via ldv1, I ran wp eval 'var_dump(wp_cache_get(508604, "bp_activity"));'

The results are different, with the cache being correctly fetched from the web server and false being returned in wp-cli.

If I fetch the activity object in wp-cli with $a = new BP_Activity_Activity(508604);, then running the same wp eval call will result in the cache being set.

This doesn't explain the cache corruption as explained in some of the previous replies in this ticket though.

#19 - 2018-07-31 02:21 PM - Raymond Hoh

> Something to be aware of is wp_cache_flush() can only be used via wp-cli:
> https://github.com/humanmade/wordpress-pecl-memcached-object-cache/blob/master/object-cache.php#L283
>
> So if a plugin is using wp_cache_flush(), it will not work.

In Human Made's memcached plugin, I've commented out the section in wp_cache_flush() so we can continue to use this function outside of wp-cli for now.

#20 - 2018-07-31 05:16 PM - Raymond Hoh

I've disabled the object cache again due to #10075 reoccurring.

For now, until we can determine if the memcached IP is the problem, what I want to try is altering object-cache.php to bail out of the object cache if wp-cli is calling for it.

Something like this:

```
function wp_cache_get() {
    if ( defined( 'WP_CLI' ) ) {
        return false;
    }

    // rest of logic here
}
```

Do the same for all the other functions like wp_cache_delete() and wp_cache_set().

#21 - 2018-07-31 06:10 PM - Boone Gorges

Thanks, Ray. I have a sense that your recommendation - don't use the object cache in WP_CLI - is overly broad. If we disable cache setting via the CLI, then when (say) a post is updated via the CLI, a stale version will be in the cache, and will be fetched during the next normal pageload. This might not matter if we were using the CLI only occasionally, but all Cavalcade jobs run via the CLI, so it's likely to trigger a problem.

We switched to the use of 'localhost' for Memcached when our current environment was being provisioned so that we would be tolerant of our HA setup. That is, if we're running on LW3B, but this server were to fail and we were to fall back on LW3A, we can't have Memcached continue to contact LW3B. So, which I think you're right that localhost is part of the problem with CLI, I think we need to do something a bit more sophisticated - maybe similar to the "is this the active node" checking when firing up Cavalcade from WP:
https://github.com/cuny-academic-commons/cac/blob/6549cb12df859f844251d0cc7b386a6a2ac4c687/wp-config.php#L33 I need to think about this a bit more before offering anything concrete, but if you agree and you have any ideas for what it might look like, please feel free to share.

At any rate, as you note, I'm unclear on how the difference between Memcached server on CLI vs Apache could cause the **wrong** values to be put into a Memcached slot. I have a feeling that 'localhost' is a red herring here, and the real issue has something to do with the way that Cavalcade fires off new PHP workers to run its tasks. I have to put some more thought into how this idea could be tested.

#22 - 2018-07-31 08:50 PM - Raymond Hoh

> Thanks, Ray. I have a sense that your recommendation - don't use the object cache in WP_CLI - is overly broad. If we disable cache setting via the CLI, then when (say) a post is updated via the CLI, a stale version will be in the cache, and will be fetched during the next normal pageload.

Ahh, you're so right. That's what I get for typing my thoughts without thinking them through!

> I think we need to do something a bit more sophisticated - maybe similar to the "is this the active node" checking when firing up Cavalcade from WP

I wrote some code for IHC while trying to determine if wp-cli was on ldv2 to set the correct DB host in wp-config.php:

```
// ldv2 via web server.
$is_ldv2 = ! empty( $_SERVER['SERVER_ADDR'] ) && 'LDV2_IP_HERE' === $_SERVER['SERVER_ADDR'];

// ldv2 via wp-cli.
if ( ! $is_ldv2 ) {
    $is_ldv2 = ! empty( $_SERVER['SSH_CONNECTION'] ) && false !== strpos( $_SERVER['SSH_CONNECTION'], 'LDV2_IP
_HERE' );
}

if ( $is_ldv2 ) {
    define('DB_HOST', 'CUSTOM_DB_HOST' );
} else {
    define('DB_HOST', 'localhost');
}
```

To check if we're on wp-cli, I'm using $_SERVER['SSH_CONNECTION']. I know it's not exactly the same, but I list it here anyway.

> https://github.com/cuny-academic-commons/cac/blob/6549cb12df859f844251d0cc7b386a6a2ac4c687/wp-config.php#L33

Correct me if I'm wrong, but this delays the Cavalcade runner (by sleeping) until the active node is running?

> I have some more theories about possible causes (see eg https://www.topsemtips.com/2011/09/php-and-memcached-returning-random-data/) but I do not have time at this moment to look into it.

I started looking a bit into this as well. Setting the memcached instance to use a persistent ID might be something to look at as well:
https://davidwalsh.name/memcached-intro
http://hoborglabs.com/en/blog/2013/memcached-php
http://php.net/manual/en/memcached.construct.php

This currently isn't done in the memcached object cache plugin:
https://github.com/humanmade/wordpress-pecl-memcached-object-cache/blob/master/object-cache.php#L741

> At any rate, as you note, I'm unclear on how the difference between Memcached server on CLI vs Apache could cause the wrong values to be put into a Memcached slot.

Probably an overreaction, but would switching to Redis as an alternative be better?

#23 - 2018-08-01 11:42 AM - Boone Gorges

> Ahh, you're so right. That's what I get for typing my thoughts without thinking them through!

Sanity checks ftw!

I did some thinking about the Memcached server address (localhost) issue. The only time there's a mismatch between Memcached servers is when you launch WP-CLI **from ldv1**. I'm not sure what your SSH setup is like, but when I connect to my /home/bgorges directory when SSHing to LDV1, I have a soft link to the Commons production site: commons-production -> /PROD_WWW/html/commons. And /PROD_WWW/ appears to be some sort of link to the filesystem. But when you cd to this directory, you're still on LDV1. This is why the 'localhost' connection to Memcached doesn't work correctly. If you SSH to lw3b once connected to ldv1, and navigate to the web directory (/var/www/html/commons/www/), the Memcached connection is the same as the normal Apache one. Since the Cavalcade daemon runs on the web nodes - lw3b, lw3a - and **not** ldv1, I'm pretty certain that this is only an issue that arises if you or I manually use WP-CLI without first SSHing to lw3b. And it's for this reason that I think that the localhost issue is not the root of the problem here.

(I also spent some time trying out different ways to determine the proper hostname for the Memcached server, in a way that would be sensitive both

to LDV1 and to the HA setup on LW3A/B. I couldn't figure anything out. As far as I can tell, there is no way to determine, from LDV1, which web node is the active node.)

> Correct me if I'm wrong, but this delays the Cavalcade runner (by sleeping) until the active node is running?

This is the technique I'm using to ensure that Cavalcade jobs run only on the active web node; this is necessary because the web nodes share a single database server. Cavalcade-Runner is a long-running PHP process that checks every second to see if there are any pending jobs. I'm hooking into this check, as early as I can given the way that Cavalcade-Runner boots up; and if I determine that the daemon is not running on the active node, I return an empty "job", which tricks Cavalcade-Runner into thinking that there's nothing to do, so that it checks again in another second. The advantage of this technique is that, if HA kicks in and LW3A becomes the active node, the daemon on LW3B will immediately start returning "null" jobs, while LW3A will immediately start returning queued jobs. The alternative way of setting this up would be to start/stop the Cavalcade daemon when the active web node changes, but I decided it wasn't worth trying to hash the details of this process out with Lihua.

> I started looking a bit into this as well. Setting the memcached instance to use a persistent ID might be something to look at as well

I'm giving this a try. I've added getmypid() to this line https://github.com/humanmade/wordpress-pecl-memcached-object-cache/blob/master/object-cache.php#L741, and reactivate the HM object-cache.php. Let's see what happens.

#24 - 2018-08-01 11:43 AM - Boone Gorges

- Related to Bug #10075: Problem with domain mapping/redirection for CETLS site added

#25 - 2018-08-01 02:18 PM - Raymond Hoh

> The only time there's a mismatch between Memcached servers is when you launch WP-CLI from ldv1.

Ah ha, that would explain things!

I usually ssh into ldv1 and then run cd /PROD_WWW/html/commons/www before launching wp-cli. I didn't think about ssh'ing into lw3b, but it makes total sense now that you mention it.

Just ssh'd into lw3b to perform a quick test and can confirm your thoughts that memcached works.

So moral of the story is if I need to perform any wp-cli operations that affect the cache in anyway, I need to do them on lw3b.

> I'm giving this a try. I've added getmypid() to this line https://github.com/humanmade/wordpress-pecl-memcached-object-cache/blob/master/object-cache.php#L741, and reactivate the HM object-cache.php. Let's see what happens.

Fingers crossed!

#26 - 2018-08-01 03:29 PM - Raymond Hoh

- Related to Bug #10087: Site Maintenance Message added

#27 - 2018-08-01 03:41 PM - Raymond Hoh

#10087 is the same bug as #10071. Since the site was down, I also didn't have much of an opportunity to gather any data and just addressed the problem by flushing the cache.

It appears using the persistent ID for memcached isn't working. Boone, do you think making the persistent ID change requires rebooting memcached?

#28 - 2018-08-01 03:55 PM - Boone Gorges

I've moved object-cache.php to object-cache.php.hm, since the fix didn't work.

> It appears using the persistent ID for memcached isn't working. Boone, do you think making the persistent ID change requires rebooting memcached?

I don't think so. getmypid() returns the pid of the Apache or CLI process serving the specific request. Since pretty much every request gets a new process, it should take effect immediately.

Until we have a better idea, I guess we have to leave object caching disabled.

#29 - 2018-08-07 06:30 PM - Raymond Hoh

So here's an interesting tidbit.

I have an hourly cronjob for RBE that checks if the IMAP connection is still connected to the inbox. If the IMAP connection no longer exists, I get an email saying that I need to reconnect. I usually get one email, but now with object caching disabled, I get four.

The emails were all sent within 2 seconds of one another and the client-ip sending the emails all vary within three distinct GC IPs (146.96.x.x) (I got the IP via the Received-SPF email header). Not sure if Cavalcade workers can have different IPs, but thought it might be somewhat related to the Cavalcade concurrency problems we were talking about on the call today.

Just a wild theory.

#30 - 2018-08-08 10:56 AM - Boone Gorges

Hm, that's an interesting data point. I'd forgotten that you'd done work to move the RBE persistence check into the object cache. Does having the cache disabled mess up RBE in this way, or does it fall back on the database (or a file, I can't remember which)?

Regarding the different IPs - how are these emails being sent? Via WP's mail system? I think the Commons server sends mail directly, right? Do any of the IP addresses match the public production IP 146.96.128.200? How about the internal IP of lw3b, 146.96.128.162? The fact that you got **four** seems suspect to me, since, to my knowledge, there are only two active server nodes in the HA cluster - lw3a and lw3b.

#31 - 2018-08-09 01:34 PM - Raymond Hoh

For RBE, I'm no longer using the object cache to check for IMAP persistence. Disabled this a long time ago as we ran into bugs, so went back to the regular file_exists() check. I understand that this isn't totally ideal, but we haven't had any reports of duplicate emails for awhile now.

I've pasted the email headers for the four emails I received here:
https://gist.github.com/r-a-y/bf92a0cf5e5c02dc7bc9e5044001f752

If you check the Received-SPF header, the client-ip is different for three of the emails (IPs end in 87-89; subdomain names appear to be proofmaster1, proofsender1, proofagent1 respectively). They do not match the public production IP. However, if you check the Received email header, lw3b is listed.

I'm not sure if my IP theory has any legs though. I checked some older emails from before the Cavalcade switch and emails tend to rotate around those IPs in the Received-SPF header.

However, the multiple firing issue still needs to be looked at. I'm wondering if we need to amend the snippet that's added to wp-config.php to try and limit the Cavalcade runner from running. To determine if a cron job is running more than once, we can probably add some logging to any cron job script to see if it is firing more than intended.

#32 - 2018-08-09 01:54 PM - Raymond Hoh

> However, the multiple firing issue still needs to be looked at. I'm wondering if we need to amend the snippet that's added to wp-config.php to try and limit the Cavalcade runner from running.

This appears to be just limited to RBE.

I checked the Cavalcade jobs queue with wp cavalcade jobs --site=1 --limit=500 --status=waiting and I see four separate jobs for the same bp_rbe_schedule task. So this appears to be related to RBE only.

Could be how I'm using wp_next_scheduled() on 'init', which runs on every pageload. If there is a fair bit of traffic at the time of the wp_next_scheduled check, the DB write to the wp_cavalcade_jobs DB table could be slow enough that multiple jobs can be scheduled. I guess this is a minor, valid issue that isn't specific to RBE.

Unfortunately, Cavalcade doesn't have a CLI command to remove jobs, but I just removed them with a wp db query CLI command. Sorry for diverging from the initial report!

#33 - 2018-08-13 02:00 PM - Boone Gorges

> Could be how I'm using wp_next_scheduled() on 'init', which runs on every pageload. If there is a fair bit of traffic at the time of the wp_next_scheduled check, the DB write to the wp_cavalcade_jobs DB table could be slow enough that multiple jobs can be scheduled. I guess this is a minor, valid issue that isn't specific to RBE.

Ah, interesting. It's possible that it's connected, especially since RBE connections run really long and thus would hold open a Memcached connection.

To bring this ticket up to date: we've been communicating with Lihua about possible issues. He's changed our Memcached configuration so that, instead of lw3b and lw3a acting as fallbacks for each other on both web nodes, each node now speaks only to the Memcached service that runs on localhost. We're hopeful that this will at least make debugging easier.

For the moment, object caching is re-enabled and we should monitor closely for recurrences of the issue. If they do come up, let's be sure to collect

detailed info - specifically, what info was incorrectly stored in which cache key - so that we can set up further debugging. My next recourse is to begin recording stack traces for every Memcached **insert** that goes through wp_cache_set(), so we can see whether the issue is on the "setting" end. If this fails, we may have to start (selectively) recording stack traces for Memcached **gets**.

#34 - 2018-08-14 10:55 AM - Boone Gorges

- Target version changed from 1.13.7 to 1.13.8

#35 - 2018-08-14 11:57 AM - Boone Gorges

To ease further debugging - and ensure that we're tracking customizations to the object cache drop-ins - I've introduced a bit of abstraction for enabling/switching cache drop-ins. https://github.com/cuny-academic-commons/cac/commit/01009b89df0d3dc8437be084dbd52e597892e008

So:

```
define( 'CAC_CACHE_ENGINE', 'memcached-hm' );
```

will load the drop-in wp-content/object-cache/object-cache-memcached-hm.php. This lets you have a different drop-in locally, or none at all.

This is not yet implemented on the production site, but it will be after the 1.13.8 release.

#36 - 2018-08-29 05:08 PM - Boone Gorges

- Status changed from New to Resolved

Tentatively, the underlying problem appears to be fix by changes to the memcached configuration on the Commons. Namely, the daemon now runs in such a way that the secondary web server is not used as part of the Memcached cluster. I'm going to mark resolved and cross my fingers that I never have to deal with this problem again :-/