CUNY Academic Commons - Feature #10684

We should short-circuit wpdb charset check

2018-11-08 05:42 PM - Boone Gorges

Status:	Resolved	**Start date:**	2018-11-08
Priority name:	Normal	**Due date:**	
Assignee:	Boone Gorges	**% Done:**	0%
Category name:	Performance	**Estimated time:**	0.00 hour
Target version:	1.14.8		

Description

Looking over the processlist log, I see that there is some correspondence between (a) moments when there are many 'sleep' connections, and (b) SHOW FULL COLUMNS queries. The latter queries are, as far as I can see, almost always caused by wpdb::get_table_charset(), which fires very early in the WP bootstrap. WP provides a pre_ filter, which we can use to avoid the query: https://core.trac.wordpress.org/browser/tags/4.9.8/src/wp-includes/wp-db.php?marks=2550-2553,2563#L2536

But there are some potential issues.

1. It looks like our database table encoding is not consistent from table to table. Older tables, like wp_1_posts and wp_users, appears to have utf8_general_ci, while newer tables have multibyte support utf8mb4_unicode_520_ci. This is likely due to failed or incomplete db upgrade routines since 4.2.x. In order for us to be able to hardcode the encoding, we need to know the encoding; this likely means enforcing the same encoding across all tables. I'll do some tests in various places (like cdev) to see what can be done about this. If we can't convert for some reason, we'll instead need to have some way of recording the existing encoding for legacy tables, and building some sort of whitelist. See https://foliovision.com/2015/04/wordpress-database-upgrade-4-2, https://make.wordpress.org/core/2015/04/02/the-utf8mb4-upgrade/

2. The 'pre_get_table_charset' filter runs very early, too early even for an mu-plugin. We need to put the callback in a drop-in, which in our case would have to be object-cache.php or sunrise.php. Or we could introduce db.php, which would contain only the callback + an include statement for wp-db.php. See https://wordpress.stackexchange.com/questions/247109/where-is-the-proper-place-to-add-a-filter-for-pre-get-table-charset

3. It appears that there's a core bug forcing us to add another filter callback alongside pre_get_table_charset. See https://core.trac.wordpress.org/ticket/38921, https://wordpress.stackexchange.com/questions/247109/where-is-the-proper-place-to-add-a-filter-for-pre-get-table-charset

I'll begin to research.

History

#1 - 2018-11-08 10:45 PM - Boone Gorges

I've done some thinking and research about how this might be best effected, and I have a couple notes to jot down.

The simplest and perhaps best solution is to convert every single table to utf8mb4. For the last year+, newly created sites on the Commons have their tables created with a utf8mb4 charset, but older tables use a mix of encodings. Converting them all to utf8mb4 would make it very simple to have a basic, hardcoded callback for the 'pre_' filters discussed above. However, converting every table is problematic. Moving to utf8mb4 means that the max index length is shorter, meaning that many tables can't be automatically converted using ALTER TABLE. This is true of some core tables - as described in the make/core post linked above - and also for a fairly large variety of custom plugin tables. In each of these cases, we'd need to first drop and re-add the shorter index before converting. There's also the fact that converting is not guaranteed to preserve weird characters in all cases, which means that I would like to have a full-fidelity backup before getting started. But our database is mammoth, making this quite hard.

My thought on the other end of the spectrum was to assemble a matrix of **actual** table encodings, and for the filter callback to reference that matrix. This is hard, though. Encodings are not consistent across blogs (as noted, older blogs have different encodings, and even then they're not totally consistent, depending in part on the server environment at the time of table creation). Moreover, since we also have to filter column charsets, the matrix size gets much bigger. And, sadly, column encoding even within similar tables across blogs is not always consistent, making it hard to come up with general rules (like: blogs with ID < 2000 have table encodings utf8_general_ci; wp_terms tables with utf8_general_ci have such-and-such encoding for the 'name' column).

So I think we should do a more scaled-back version of the conversion project. Instead of converting all tables, we'll look for some targeted ones. The SHOW FULL COLUMNS queries in the processlist log are focused on, in rough order of frequency:
- options tables
- postmeta tables
- the wp_usermeta table
- statpress tables

It makes sense that these are on the list most, since they're some of the tables most likely to be updated on a fairly regular basis. So my next step is to write some scripts that crawl through the database, looking for tables of each of these types; uses mysqldump to get a backup of the table;

converts indexes if necessary; and then converts the table encoding. Once this is done, we can then write pre_ filters for these specific tables. This targeted approach should allow us to get the worst offenders.

#2 - 2018-11-09 12:32 PM - Boone Gorges

- File convert-tables.php added

Ray, could I ask you to do a review of what I've done so far? Two things:

1. I wrote a script that converts all options tables to utf8mb4. It's attached. You can see that it has some failsafes built into it, including table-level backups. It's also flexible enough to be used in the future for other tables. I ran it on cdev and it converted all 2800+ tables without any errors at all, so I feel pretty good about it. But it definitely needs a review.

2. Here's the filter changeset: https://github.com/cuny-academic-commons/cac/commit/d7176df9aab1d78e0dbbd0692d1c063dff61f3d9

#3 - 2018-11-09 04:13 PM - Boone Gorges

wp_bp_activity is another high-impact place to make this change, since it's a single table, and is updated frequently.

#4 - 2018-11-12 11:02 AM - Raymond Hoh

> Ray, could I ask you to do a review of what I've done so far?

Thanks for the well-researched post, Boone.

The CLI script looks good and since you've already tested it on cdev with no errors, it should be safe to run on production. The Foliovision article notes that running this sort of script should be done when the site isn't really busy.

I also agree that converting tables based on the number of hits in the process log is a good and safe way to show if the charset conversion will improve performance. I'd say go ahead when you feel comfortable to pull the trigger, Boone!

#5 - 2018-11-13 10:50 AM - Boone Gorges

- Target version changed from 1.13.13 to 1.14.1

Thanks for the sanity check, Ray!

Last night, I ran the conversion script for _options tables, and then put https://github.com/cuny-academic-commons/cac/commit/6f4bbc956c3a08f8ada8bd45f9dae881f2e1504b in production as a hotfix. It seems to have worked - SHOW FULL COLUMNS queries are appearing in the processlist log with about half the frequency.

I've just run the conversion on wp_bp_activity and added to the short-circuit filter: https://github.com/cuny-academic-commons/cac/commit/84f00f98601af11b91196203f1a06bbe23c0db91

I'll save a tarball containing table backups from this set of conversions for a few weeks. options-20181112.tar.gz in my ldv2 home directory.

Bumping this ticket to the next milestone to handle another round of offenders. I plan on tackling:
- wp_usermeta
- statpress tables
- postmeta
- commentmeta

Some of these will require rebuilding indexes, so will require handling on a per-table basis.

#6 - 2018-11-17 05:59 PM - Boone Gorges

I've run the scripts on the following tables:

- postmeta https://github.com/cuny-academic-commons/cac/commit/976af0726f21be8d8318039d92fdf099991d8c29
- statpress https://github.com/cuny-academic-commons/cac/commit/60beb92038394a49fac52c9bbee6611cce5bbf58

#7 - 2018-11-27 12:39 PM - Boone Gorges

- Target version changed from 1.14.1 to 1.14.2

#8 - 2018-12-05 07:36 AM - Boone Gorges

Cavalcade tables converted and added to whitelist in https://github.com/cuny-academic-commons/cac/commit/87d774e91886143b557ecfc8a6464bfbe3ec9e6b

#9 - 2018-12-05 10:20 AM - Boone Gorges

- Target version changed from 1.14.2 to 1.15

Commentmeta tables backed up and converted. Relevant changeset:
https://github.com/cuny-academic-commons/cac/commit/64a2b2a671ec3c326f1f7f35d9fb7d4732777a82

This covers all the cases I referenced above. Going to punt this to 1.15 for future analysis.

#10 - 2019-03-01 05:36 PM - Boone Gorges

- Target version changed from 1.15 to 1.14.8

I've just run the upgrade routine for `posts` and added the short-circuit
https://github.com/cuny-academic-commons/cac/commit/c328782837a3553793a78b9ed6fd4d636f801a08

In my view, this is all that needs doing. Let's close the ticket.

#11 - 2019-03-12 10:33 AM - Boone Gorges

- Status changed from New to Resolved

Files

convert-tables.php	1.7 KB	2018-11-09	Boone Gorges