Chamilo LMS - Feature #7510

Database queries take too long to respond - Implement option for MYSQL_CLIENT_COMPRESS

04/02/2015 14:36 - Yannick Warnier

<table>
<thead>
<tr>
<th>Status:</th>
<th>Feature implemented</th>
<th>Start date:</th>
<th>04/02/2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>Yannick Warnier</td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td>Database &amp; API changes</td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>1.10.0</td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Complexity:</td>
<td>Normal</td>
<td>SCRUM pts - complexity:</td>
<td>?</td>
</tr>
</tbody>
</table>

**Description**

Torkil recently reported on IRC that one of his Chamilo portals was getting very slow with only 600 users connected simultaneously, in particular when loading a page with many social network messages.

In his case, the web server and database server were on different servers, which meant that data coming back from the database server could be slowed down by the badnwidth or a firewall in the middle.

After a few tests, I suggested he tried client database compression (a possibility in MySQL and MariaDB). The tests proved very efficient and the issue was fixed (thanks Torkil!).

This shown the possibility to have that as a setting in Chamilo, so that anyone with issues of this kind could easily fire up compression to see if that was fixing the issue.

So I did some tests on a local machine (so not too much load) and figured it wouldn't hurt to add a suggested setting for that in configuration.dist.php.

You now (since 1.10.0) have the possibility to enable compression (and I also added a clearer way to enable persistent connections) from configuration.php:

```php
// Persistent connections may have profound effects (not always positive) on your database server. Use with care.
//$_configuration['db_persistent_connection'] = false;
// For separate web and DB servers, reduce the bandwidth used by compressing data returning from the DB server. By default, it is ignored. Uncomment the following to enable compression.
//$_configuration['db_client_flags'] = MYSQL_CLIENT_COMPRESS;
```

**Associated revisions**

Revision 3cd5b4bc - 04/02/2015 15:16 - Yannick Warnier

Add DB compression flag configuration - refs #7510

**History**

#1 - 04/02/2015 15:14 - Yannick Warnier

- % Done changed from 30 to 50

On a local machine with a cuad-core, these are the results of an "ab -c 20 -n 500" on the homepage.

A word of warning: these results are not very relevant, neither in terms of compression as we would need queries with a lot of data, nor in terms of persistent connections as we would need a realistic user behaviour rather than "just" testing the homepage, but I still find them interesting.

Without any optimization:

Requests per second: 19.26 [#/sec] (mean)

With (only) persistent connections enabled (load average of 10 by the end of the test):

Requests per second: 17.18 [#/sec] (mean)
With (only) compression (load average of 10 by the end of the test):

Requests per second: 18.65 [#/sec] (mean)

With compression and persistent connections (load average of 10 by the end of the test):

Requests per second: 18.43 [#/sec] (mean)

So, in conclusion, there is no significant improvement on a small local server. It is likely that compression will only be useful when used with separate web and database servers.

The persistent_connection option is likely to be useful only for servers with an average number of users all using complex pages with a lot of queries in each single page.

The settings are optional though, so feel free to test them and return here to report.

#2 - 05/02/2015 01:14 - Yannick Warnier
- Subject changed from Database queries take too long to respond to Database queries take too long to respond - Implement option for MYSQL_CLIENT_COMPRESS
- Status changed from Assigned to Feature implemented
- % Done changed from 50 to 100

I have tested both settings on a relatively high-load server (600 users in the last 50 minutes) and it showed no notable/clear performance improvement. It showed no issue either.

These are settings that must be used carefully in very specific cases. Closing the task.

#3 - 07/04/2015 22:57 - Torkil Zachariassen

For the record. The original issue was permanently eliminated by moving the external database server onto the same subnet as the webserver. Being on separate networks the network failed to transmit chunks larger than than 8MB, and would "freeze". This was why the compression helped us somewhat by keeping the chunks smaller than the arbitrary limit of 8MB. Thus compression might be of some use in case of network issues.

After upgrading to 1.9.10, and having moved the database server, we use the following in configuration.php, as no changes have been made to database.lib.php or database.mysqli.lib.php yet, and we do not really need compression any longer. (Note the change of variables: 'persistent' rather than 'db_persistent_connection' and 'client_flags' rather than 'db_client_flags')

```php
// Extra database options for chamilo-lms 1.9.10, cf: https://support.chamilo.org/issues/7510
// Persistent connections may have profound effects (not always positive) on your database server. Use with care.
//$_configuration['persistent'] = false;
// For separate web and DB servers, reduce the bandwidth used by compressing data returning from the DB server. By default, it is ignored. Uncomment the following to enable compression.
//$_configuration['client_flags'] = MYSQL_CLIENT_COMPRESS;
```

In 1.10 the form presented above by Yannick should be used