# Database - Bug #7330

## Increase psCache size

05/08/2023 05:05 AM - Alexandru Lungu

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Alexandru Lungu | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |
| **billable:** | No | **case_num:** | |
| **vendor_id:** | GCD | **version:** | |

**Description**

**Related issues:**

| | |
|---|---|
| Related to Database - Bug #7351: Reduce SQL query string size of an INSERT IN... | **Closed** |
| Related to Database - Bug #7388: Create server configuration container for ca... | **Test** |

### History

#### #1 - 05/08/2023 06:05 AM - Alexandru Lungu

Consider extending the psCache size from 2048 to 65535. This was attempted before in #6829, but the memory consumption was too large. In the meantime, most of the H2 statements are generated using wildcard (e.g. tt.*).

- Check memory consumption on a large customer application (2048 vs 65535)
- Check which is the maximum and average cache key memory usage
- Check number of inserts / evictions in regard to hit/miss.

#### #2 - 05/09/2023 06:31 AM - Constantin Asofiei

Alexandru, for example, in an application, I saw ~8k JDBC prepared statements being created. If the actual used cache after increasing it to 65535 is still ~8k, then there are really 8k statements being prepared.

Also, I don't want to just hardcode the cache size, we need a directory configuration with a good default.

And something else thing to consider: any PreparedStatement instance is linked to a JDBC connection. Currently, the psCache is context-local. I wonder if it may help (for a multi-user scenario) to keep a global cache, and re-assigning the JDBC connection to the PreparedStatement, when is being actually used. The point here is to avoid the re-parse of the query.

#### #3 - 05/09/2023 06:32 AM - Constantin Asofiei

Constantin Asofiei wrote:

> And something else thing to consider: any PreparedStatement instance is linked to a JDBC connection. Currently, the psCache is context-local. I wonder if it may help (for a multi-user scenario) to keep a global cache, and re-assigning the JDBC connection to the PreparedStatement, when is being actually used. The point here is to avoid the re-parse of the query.

But the temp-tables are private to the H2 session, and I don't know how it will work if each session can have different definition of a temp-table.

#### #4 - 05/12/2023 04:57 AM - Alexandru Lungu

*- % Done changed from 0 to 50*

I've got a memory dump to check the status on the psCache on a large customer application.
Before doing a client run, the app-servers of the application consume 258kB - 305kB in psCache each. However, they are storing only 111 - 121 statements there. All statements are between 1.29kB - 7.42kB each. Therefore, the total of 1.130 statements stored by the app-servers add up to 2.681 kB. This is not of interest here.

For a client run, I see a full-cache (2047 entries) of 10.520kB (so ~10MB). Even though the largest statement is of ~48kB, these are isolated cases. In fact, the retaining distribution follows:

- 40kB - 50kB: 2 statements
- 30kB - 40kB: 3 statements
- 20kB - 30kB: 13 statements
- 10kB - 20kB: 37 statements
- 09kB - 10kB: 72 statements
- 08kB - 09kB: 23 statements
- 07kB - 08kB: 383 statements
- 06kB - 07kB: 149 statements
- other ~1300 are < 6kB

I've done an investigation on statements >15kB. These fall in one of the following 4 categories:

- 50% are INSERT SELECT statements which use the full field-list into the query string. This is one of the few cases where the full field-list is still embedded into the query string. Either the wildcard / direct-access can be used here. However, these are only some (around 20 / 2048) queries, so I don't think we need to panic and fix this straight-away.
- 25% are just large SELECT statements. Even if they use tt.* syntax, the wildcard is expanded into lots(> 100) ExpressionColumn instances.
- 25% are just large UPDATE statements. Each update statement stores a list of Parameter instances (> 100).
- I see only one statement which was cached even though it references a dropped table (and it wasn't evicted yet). It retains 41.57kB, where 15.87kB are the retained index nodes. I don't think this is something to worry about; the H2 index/table was already retaining memory before being dropped. Being continuously retained by this cache for a limited period of time reflects as a "delayed" memory free, not a memory spike.

Side observations:

- Most of the 383 statements 07kB-08kB are of SELECT-UNION kind, used for 2-unique index validation. Check #7323. I suspect most of the 2kB-3kB to be validation queries for tables with 1-unique index.
- Avoiding to expand wildcards to save time/memory is already discussed in #7321.

The results are meeting the expectations from #7026-170. I will redo my test with a 65535 psCache.

> Also, I don't want to just hardcode the cache size, we need a directory configuration with a good default.

I will add this.

> And something else thing to consider: any PreparedStatement instance is linked to a JDBC connection. Currently, the psCache is context-local. I wonder if it may help (for a multi-user scenario) to keep a global cache, and re-assigning the JDBC connection to the PreparedStatement, when is being actually used. The point here is to avoid the re-parse of the query.

This sounds adventurous. I thought of this before, but as you mentioned, this means caching "cross-session" prepared statements. There should be some sort of resource remapping from one session to another just to reuse the prepared statement. This should be carefully considered. For simple queries, we can use direct-access straight-away. For complex queries, we should dig deep into the query to remap any sub-select queries, all tables, indexes, etc. Some things may be hard to remap, like IndexCursor. The index selectivity seems the worse issue here, as one user may opt to query data in a faster way using another plan. This made me think that, after #6829, our prepared statements may still use a slow plan if cached for so long. I guess this is not visible yet because we have very simple queries (usually single-table).

#### #5 - 05/12/2023 06:02 AM - Alexandru Lungu

Retested with a psCache of capacity 65535.

I get a psCache size with only 4470 statements, so according to #7330-2, half the new prepared statements could have been cache hits. The total size retained is 28.860kB, so it preserves the retaining per size (for 2048 I got 10.520kB). The largest entry retains 58.440kB. The distribution here is:

- 50kB - 60kB: 3 statements
- 40kB - 50kB: 9 statements
- 30kB - 40kB: 10 statements
- 20kB - 30kB: 31 statements
- 15kB - 20kB: 46 statements
- 10kB - 15kB: 93 statements
- other ~4278 are < 10kB

This is acceptable from my POV, (<5% statements have more than 10kB). The total size is also decent (~28MB), but can be improved. I analyzed the statements with >30kB:

- 40% are still INSERT SELECT with very long query strings (over tables with ~240 fields)
- 31% are SELECT over dropped tables. This is concerning as the expire cache policy won't take effect, so tables are retained in the cache. However, note that only some nodes from the traversed index are retained (not the whole index/table). In fact, only the path from the root to the cursor node is retained (if any). I have some examples where only 2kB/33kB represent the retained index nodes.
- 18% are just very large SELECT statements (even with 300 fields)
- 11% are UPDATES with several Parameter instances

#### #6 - 05/12/2023 06:10 AM - Alexandru Lungu

*- % Done changed from 50 to 80*

As a conclusion here, I think it is optimal to either extend the cache to 4096 (and expire 400 entries and lose some cache hits) and such we set the 30MB limit for one context; or, we can go with 8196 to save all cache hits, but risk having 50-60MB psCache per context. I would go with the second option and rely on the fact that the further H2 optimizations will succeed in lowering the total retention size.

Please advice! I will go ahead with the directory.xml option to make the psCache size configurable. I will commit it to 7026d.

#### #7 - 05/12/2023 08:21 AM - Eric Faulhaber

Alexandru Lungu wrote:

> As a conclusion here, I think it is optimal to either extend the cache to 4096 (and expire 400 entries and lose some cache hits) and such we set the 30MB limit for one context; or, we can go with 8196 to save all cache hits, but risk having 50-60MB psCache per context. I would go with the second option and rely on the fact that the further H2 optimizations will succeed in lowering the total retention size.
>
> Please advice! I will go ahead with the directory.xml option to make the psCache size configurable. I will commit it to 7026d.

I agree with your proposal. Nice research!

#### #8 - 05/12/2023 08:58 AM - Alexandru Lungu

Constantin, I am ready to commit the "ps-cache-size" option support. Shall I set a default of 2048 (as it was now) and let you configure it to 8196, or should I set it by default to 8196?
Right now, I don't know how this increase will affect other customers. On one hand, letting this set on 2048 won't affect other customers in any way. By setting it to 8196, it **may** start consuming more memory, but **may** perform better out-of-the-box.

#### #9 - 05/12/2023 09:00 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Constantin, I am ready to commit the "ps-cache-size" option support. Shall I set a default of 2048 (as it was now) and let you configure it to 8196, or should I set it by default to 8196.
> Right now, I don't know how this increase will affect other customers. On one hand, letting this set on 2048 won't affect other customers in any way. By setting it to 8196, it **may** start consuming more memory, but **may** perform better out-of-the-box.

2048 seems low to me, for long-running servers. I would default it to 8196, especially as you mention now the PS's heap usage is reduced.

#### #10 - 05/12/2023 09:02 AM - Greg Shah

Please default to 8192.  If memory is an issue, customers can reduce it and take the performance hit.  I prefer faster by default.

#### #11 - 05/12/2023 09:33 AM - Alexandru Lungu

*- Status changed from WIP to Review*

*- % Done changed from 80 to 100*

Committed 7026d/rev. 14566 Set default to 8196. Configuration option: persistence/ps-cache-size as integer.

#### #12 - 05/12/2023 09:33 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Committed 7026d/rev. 14566 Set default to 8196. Configuration option: persistence/ps-cache-size as integer.

Eric: I'm inclined to add a container node where to configure all these cache sizes.  I expect to add more persistence-related cache configs to directory.xml.

#### #13 - 05/12/2023 09:41 AM - Greg Shah

Why 8196 instead of 8192?

#### #14 - 05/12/2023 09:52 AM - Alexandru Lungu

Greg Shah wrote:

> Why 8196 instead of 8192?

I don't have a proper excuse for that :) Fixed it now in 7026d/rev. 14568.

#### #15 - 05/12/2023 11:16 AM - Eric Faulhaber

Constantin Asofiei wrote:

> Alexandru Lungu wrote:
>
> > Committed 7026d/rev. 14566 Set default to 8196. Configuration option: persistence/ps-cache-size as integer.
>
> Eric: I'm inclined to add a container node where to configure all these cache sizes.  I expect to add more persistence-related cache configs to directory.xml.

That makes sense, they should be consolidated into one location.

#### #16 - 05/15/2023 07:10 AM - Alexandru Lungu

*- Related to Bug #7351: Reduce SQL query string size of an INSERT INTO SELECT FROM added*

#### #17 - 05/29/2023 03:25 AM - Alexandru Lungu

*- Related to Bug #7388: Create server configuration container for cache sizes added*

#### #18 - 05/29/2023 03:27 AM - Alexandru Lungu

*- Status changed from Review to Test*

Merged 7026d to trunk as rev. 14587.
Created #7388 for further work on cache size configurations.
This can be closed.

#### #19 - 05/29/2023 09:58 AM - Greg Shah

*- Status changed from Test to Closed*