Database - Bug #7452

Analyze memory of the value cache from FWD-H2 and eventually increase size

06/20/2023 03:59 AM - Alexandru Lungu

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Alexandru Lungu	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Bug #7454: Make ValueStringIgnoreCase the default gener...	**Closed**
Related to Database - Bug #7448: Optimize FWD-H2 ValueTimestampTimeZone and m...	**Rejected**

History

#1 - 06/20/2023 04:14 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Alexandru Lungu

FWD-H2 has a Value.softCache that stores the values used by H2. Every bit of information stored by H2 is wrapped by such Value: integers, strings, dates, etc. I recently done a cache ratio statistic and reached ~~~66% (this was retrieved after #7363)~~ 51%. The cache is very fast, but doesn't consider LRU policy. It is just some kind of fixed size hash-map (array with hash index). This way it is faster (avoiding LRU overhead), but it has a worst-case scenario (two values with the same hash being continuously used).

The default is set on 1024. My statistic shown ~~549.859 cache hits and 209.659 cache misses (this was retrieved after #7363)~~ 471.874 cache hits and 230.256 cache misses on a customer application. I think these number are incomparable with the cache size, so there is room for improvement. My intend is to reach ~90% without sacrificing too much of memory (also considering #7448).

- I intend to do a test with 2048 and 4096 sizes just to check if larger caches actually produce significant performance bonuses.
- From my POV, private static SoftReference<Value[]> softCache really means one instance per JVM, right? This is inside H2. Thus, I wonder if multiple embedded in-memory connections share this same cache instance. Otherwise, maybe H2 is doing some hacks so that this softCache is in fact per-connection. Is this even possible?
    - If this is per connection, which i doubt, then increasing the size is quite dangerous.
    - If this is not per connection, then we can increase the size by much more, considering we have only one such instance.
    - Note that H2 also has OBJECT_CACHE_MAX_PER_ELEMENT_SIZE, which is the maximum size of a cached element. We we fear of caching objects that are too large, we can reduce this threshold. Currently, the default is 4096. E.g. strings with more than 4096 characters are not cached. This means that this softCache can retain only 8MB at most.
- As of #7388, we can consider making the cache size configurable.

#2 - 06/20/2023 06:06 AM - Alexandru Lungu

This is a statistic on the value cache hit ratio depending on the cache size.

Size	Hit	Miss	Ratio
1024	465.070	230.449	51%
2048	483.245	214.366	57%
4096	492.072	200.855	60%

The improvement is clearly there, but it is quite under the expectations. I will do some analysis on why are the misses so many. From some initial observations:

- Each time we insert a value that is STRING_IGNORE_CASE, we use INSERT prepared statement with setString. For each such ValueString parameter, a ValueStringIgnoreCase is generated in the store. Therefore, we will face 2 cache misses for new string values that should be case-insensitive. This is also a performance issue, as we need 2 FWD-H2 values for each FWD persisted character value. **Maybe we can presume that each setString is for a case-insensitive value, so that ValueStringIgnoreCase will be the common case. AFAIK, there are way more case-insensitive values than case-sensitive ones in customer applications.** - #7454

#3 - 06/20/2023 06:11 AM - Alexandru Lungu

- Related to Bug #7454: Make ValueStringIgnoreCase the default generated value for setString in FWD-H2 added

#4 - 06/20/2023 06:11 AM - Alexandru Lungu

- Related to Bug #7448: Optimize FWD-H2 ValueTimestampTimeZone and maybe avoid caching added

#5 - 06/20/2023 09:16 AM - Greg Shah

> AFAIK, there are way more case-insensitive values than case-sensitive ones in customer applications.

Very true.

#6 - 06/27/2023 06:40 AM - Alexandru Lungu

I will pend the effort here on analyzing cache size changes until #7454 is finished.