Database - Feature #6815

configure all cache sizes in the directory, and create documentation for them

10/06/2022 02:24 AM - Constantin Asofiei

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Bug #7388: Create server configuration container for ca...	**Test**

History

#1 - 10/06/2022 02:25 AM - Constantin Asofiei

In 6129a, the prepared statement cache in TempTableDataSourceProvider is still configured to 100 - this was not enough, I will increase it in 6129a to 65536. In 3821c, is configured to 1000.

We need to allow configuration of all caches in the directory, plus document them.

#2 - 10/26/2022 05:53 AM - Constantin Asofiei

This includes the query_cache_size=1024 for temp-tables in H2Helper.setCommonInMemoryProperties.

#3 - 07/05/2023 03:29 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon

This is already implemented in 7388a.

#4 - 07/05/2023 04:38 AM - Constantin Asofiei

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

#5 - 07/05/2023 04:39 AM - Constantin Asofiei

Alexandru Lungu wrote:

> This is already implemented in 7388a.

We need also the documentation.

#6 - 11/10/2023 10:57 AM - Alexandru Lungu

- Status changed from New to WIP

Danut, please point out to the documentation in this task and mark it as Review when finished.

#7 - 11/10/2023 11:38 AM - Constantin Asofiei

Alexandru, in #7669, I start seeing DynamicQueryHelper cache level #1 is full. Dumped 1 old entries - (some 750k message logs, this uses the default 65536, there is no config in directory.xml). Is there a way to get info when a cache size is small enough that it starts evicting entries aggresively? Maybe some JMX instrumentation? This may help to find some good values to tune the cache size.

OTOH, for this specific case, it may just be that the parsed predicate is 'one of a kind', in that it uses some unique value which is never reused, so caching it is moot anyway.

#8 - 11/14/2023 07:42 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, please point out to the documentation in this task and mark it as Review when finished.

The documentation for the cache sizes is available in Database Configuration. Currently the CacheManager stores and uses the cache sizes defined in the directory configuration, each cache size is retrieved internally when creating caches (LRUCache or Map). If we want to use the CacheManager to set the size for the property mentioned in #6815-2, we can simply make CacheManager.getCacheSize method public. Should I go ahead and do that?

#9 - 11/14/2023 07:53 AM - Alexandru Lungu

> OTOH, for this specific case, it may just be that the parsed predicate is 'one of a kind', in that it uses some unique value which is never reused, so caching it is moot anyway.

Maybe we can make them "more of a kind" :) I don't know if this is possible without context.

> Is there a way to get info when a cache size is small enough that it starts evicting entries aggresively? Maybe some JMX instrumentation? This may help to find some good values to tune the cache size.

I think we have some specific JMX on some caches ... or maybe not. Now that we use the CacheManager, I see a really neat implementation: add dynamic JMX information for each cache generated by the CacheManager. This way we can instrument all caches for fine tuning. This is not implemented, but can be done. I don't quite have the inspiration now to design and implement this, so I will ask Danut to tackle this.

In my head, having a fat JMX storing a map Cache Name -> Cache Statistics is the way to go. Take for example QueryProfiler that for each query is computing statistics like cache hit / miss / row count etc. I see cache statistics as storing the number of elements / evicts cache hit / etc. To avoid any potential overhead, maybe we can deliver InstrumentedLRUCache from cache manager if we want to do the instrumentation of simply LRUCache if in production. Constantin, any feedback on this?

#10 - 01/10/2024 07:12 AM - Dănuț Filimon

- Status changed from WIP to Review

- % Done changed from 0 to 70

Created **6815a** and committed **rev.14916**. As suggested by Alexandru in #6815-9, I've created a way to instrument cache through CacheManager. Similar to QueryProfiler, I created CacheProfiler and instrumented a total of 16 methods (number of calls and total execution time).

We also talked about expanding the CacheManager out of the persistence container if we want to configure caches that are outside of this scope. This includes modifying the searched containers (for cache sizes and instrumentation value) and documentation.

Alexandru, please review and modify the Done% if you think there more work to do. I plan to test a customer application and post the results here.

#11 - 01/11/2024 05:28 AM - Dănuț Filimon

Committed 6815a/rev.14917. Removed unnecessary methods and changes the container accessed for the instrumentation and cache-sizes. The container accessed is at the same level as the persistence container and is called cache-config.

As for the results of the instrumentation, I got the following:

Name	GET (count)	GET Time (ms)	PUT (count)	PUT Time (ms)
BufferManager	23964	8.76	297	0.18
DynamicTables Helper	2357	32.453	306	1.626
FQLHelperCache	2245427	479.051	1984	0.718
FQLPreprocessor	377640	159.429	4082	1.832
FQLPreprocessor ast	26339	22.65	3003	1.208
FastFindCache L2	2116970	92.151	-	-
FastFindCache L3	2075153	362.788	1700792	466.015
Persistence	1558366	362.994	14043	17.436
SortCriterion	382860	165.883	7463	1.7
TemporaryBuffer	7894	9.726	277	0.135
Session local/_temp/primary	5048192	5.048	-	-
Session local/<db>/meta	25419	2.605	-	-
Session local/<db>/primary	2911160	299.126	-	-
TempTableDataSourceProvider	4210094	608.273	4064	1.719
SourceNameMapper search	23	0.096	3	0.01
SourceNameMapper source	105954	11.696	-	-

There were more methods instrumented, but I just noticed an issue that invalidates some of the results so I solved it in **6815a/rev.14918**.

#12 - 01/11/2024 05:39 AM - Alexandru Lungu

What other statistics can be achieved? I suspect the ratio is the most important one, so lets get it done. Also, we need a size, so we can understand in what circumstances were that statistics achieved.
Secondly, please address the export function. We want to have an easy to read format (table-like). Maybe use the Textile format we also use for Redmine (for easy copy-paste).
Mind that I didn't review the changes yet; please address the features above first.

Finally, we need to segregate the GET into GET hit and GET miss. We also need the timings separated. Also, we need different options for "instrument" and "profile". We can allow instrumentation without considerable overhead, but profiling may introduce too much of an overhead. Lets have them enabled separately.

#13 - 01/11/2024 05:48 AM - Dănuț Filimon

- Status changed from Review to WIP

Alexandru Lungu wrote:

> What other statistics can be achieved? I suspect the ratio is the most important one, so lets get it done. Also, we need a size, so we can understand in what circumstances were that statistics achieved.
> Secondly, please address the export function. We want to have an easy to read format (table-like). Maybe use the Textile format we also use for Redmine (for easy copy-paste).
> Mind that I didn't review the changes yet; please address the features above first.
>
> Finally, we need to segregate the GET into GET hit and GET miss. We also need the timings separated. Also, we need different options for "instrument" and "profile". We can allow instrumentation without considerable overhead, but profiling may introduce too much of an overhead. Lets have them enabled separately.

Currently only the count and execution time are monitored, the ratio can be calculated during export (there is no method for exporting the results atm). I'll start working on the suggested points (the export, GET hit and GET miss ratio should be a must) and I especially like the idea of using the Textile format.

You can put the review on hold for the moment.

#14 - 01/12/2024 03:15 AM - Dănuț Filimon

I added monitoring for the configured and the in use cache size (latest size of the cache), GET hit/miss and modified the export method to print the results in the Textile format but the resulted table is too big to be posted on Redmine due to the number of instrumented methods. Can we consider redirecting the output to a csv file by adding an additional parameter to toggle between the formats or disregard the Textile format altogether?

Alexandru Lungu:

> Also, we need different options for "instrument" and "profile".

I think instrumentation should always be enabled so that method calls can be counted, profiling should only monitor the execution time of the method and can be an additional parameter and when it is used, to calculate the average method call time based on the counted calls.

#15 - 01/12/2024 03:50 AM - Alexandru Lungu

Dănuț Filimon wrote:

> I added monitoring for the configured and the in use cache size (latest size of the cache), GET hit/miss and modified the export method to print the results in the Textile format but the resulted table is too big to be posted on Redmine due to the number of instrumented methods. Can we consider redirecting the output to a csv file by adding an additional parameter to toggle between the formats or disregard the Textile format altogether?

Why is it too big? Shouldn't it be similar to #6815-11, but with some extra columns?
Anyway, you have only the possibility to use print with our FWD JMX. CSV can do to be easily imported in Spreadheets.

> I think instrumentation should always be enabled so that method calls can be counted, profiling should only monitor the execution time of the method and can be an additional parameter and when it is used, to calculate the average method call time based on the counted calls.

The instrumentation implies overhead which is not desired in production systems. Instrumentation (and profiling) is meant to be used by us / system administrators that want to tune the caches in testing environments. Thus, please disable the instrumentation by default.

- the CacheManager should easily return bare caches when instrumentation is disabled
- the CacheManager should deliver instrumented caches when instrumentation is enabled,
- the cacheManager should deliver instrumented caches with profiling enabled only when both instrumentation and profiling are enabled.

I think these settings can go in the directory.xml together with the sizes. After the size definition, we can enable instrumentation and profiling eventually (but they are disabled by default). **Also**, if you have a cache configuration in directory.xml, but no size, let the default size kick in.

#16 - 01/12/2024 04:01 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Why is it too big? Shouldn't it be similar to #6815-11, but with some extra columns?
> Anyway, you have only the possibility to use print with our FWD JMX. CSV can do to be easily imported in Spreadheets.

Not only get and put methods are instrumented, methods like containsKey, containsValue, getOrDefault, putAll, putIfAbsent, remove, replace are methods that are also instrumented. If you also want to create an "Average" between call count and execution time we will have additional columns. I am currently looking into removing some of the methods that are not called based on previous instrumentation results.

> The instrumentation implies overhead which is not desired in production systems. Instrumentation (and profiling) is meant to be used by us / system administrators that want to tune the caches in testing environments. Thus, please disable the instrumentation by default.
>
> - the CacheManager should easily return bare caches when instrumentation is disabled
> - the CacheManager should deliver instrumented caches when instrumentation is enabled,
> - the cacheManager should deliver instrumented caches with profiling enabled only when both instrumentation and profiling are enabled.
>
> I think these settings can go in the directory.xml together with the sizes. After the size definition, we can enable instrumentation and profiling eventually (but they are disabled by default). **Also**, if you have a cache configuration in directory.xml, but no size, let the default size kick in.

I took it in account and this is already handled.

#17 - 01/12/2024 04:04 AM - Alexandru Lungu

Danut, please merge some of the statistics in one: get, getOrDefault can be merged. The same as put, putAll and putIfAbsent. The others can be standalone: containsKey, containsValue, remove and replace.

#18 - 01/12/2024 05:29 AM - Dănuț Filimon

- Status changed from WIP to Review

- % Done changed from 70 to 100

Committed 6815a/rev.14919. Reduced the number of instrumented methods (based on results from a large application, those methods were not used at all), combined similar methods, separated instrumentation and profiling. Profiling can be configured in the cache-config container, through profiling (similar to instrumentation, it is a boolean).

I just remembered that the Redmine tables are really responsive and I've seen large tables being posted so we can ignore my previous suggestion of using a csv (we can add it if we really want to).

We also received a fix for the POC so I'll resume work in other issues and also run the instrumentation.