Database - Bug #7731

performance improvements for FQLPreprocessor

08/21/2023 02:03 PM - Constantin Asofiei

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

Description

History

#1 - 08/21/2023 02:09 PM - Constantin Asofiei

When something is inlined, the FQLPreprocessor does not use the cache at all. What I noticed is that FQLPreprocessor.parse is not dependent on any state other than the actual FQL. We can keep in memory the HQLAst and just create a duplicate once it is initially parsed.

On an additional note: we need to consider if we should use a LRUCache instead of a HashMap, for both this HQLAst cache and the existing FQLPreprocessor.cache, as state can add up fast (maybe with a default size increased to 16k or 32k). The FQLPreprocessor.cache is already at 2155 just after running two complex tests. But this is something which should be checked after running more complex functional tests in a customer app.

#2 - 08/31/2023 04:22 AM - Constantin Asofiei

Alexandru, I think this can help - can you take a look at it?

#3 - 09/04/2023 06:05 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Dănuț Filimon

Danut, please do some investigation here - you have some consistent background with FQLPreprocessor and inlining :) I will also do some work on this in parallel.

For to start we need to understand this scenario better, so please do a test-case that reproduces the issue. Eventually, transform it into a profiling test. We need to see in the end how much our work will improve this.

#5 - 09/06/2023 10:34 AM - Dănuț Filimon

If we create a HQLAst cache and I assume that we want to access and populate it in FQLPreprocessor.parse(), won't it be possible to have multiple similar fqls? Take tt01.f1 < 5 and tt01.f1 < 6 for example, there will be two different entries in this case and I don't think it is correct to keep such precise statements in the cache.

#6 - 09/06/2023 10:36 AM - Constantin Asofiei

The cache is meant to be on the FQL string, before the parse is performed - there is no buffer reference (i.e. semantics) in hql.g, just syntax processing. So in your case, as the FQL string is different, there will be different cache ~~settings~~ entries.

#7 - 09/11/2023 07:57 AM - Dănuț Filimon

The FQLPreprocessor will always use the cache when the dialect is P2JH2Dialect because of the inline = inline && dialect.isQueryRangeParameterInlined();, which will always return make the inline equal to false. Another thing I noticed is that when using a dynamic query, another FQLPreprecessor is created and then cached for dynamic use (this happens in AdaptiveCOmponent.prepareHQLPreprocessor).

One problem with the cache implementation is that when we have two dynamic queries, one with pt1.f1 < 1 and one with pt1.f1 < 2. Both will have a substitution parameter extracted before making it into the FQLPreprocessor and the HQLAst will not be created with SUBST tree, but with NUM_LITERAL, then cached. When preprocessing the second where clause, the cache will return the first cached HQLAst which has the **text** value **1** instead of **2**. We can also consider cases where the substitution parameter is of a different type (e.g. LOGICAL), in this case the HQLAst will be different from the one we already cached and will create another one.

#8 - 09/11/2023 08:07 AM - Dănuț Filimon

Dănuț Filimon wrote:

> One problem with the cache implementation is that when we have two dynamic queries, one with pt1.f1 < 1 and one with pt1.f1 < 2. Both will have a substitution parameter extracted before making it into the FQLPreprocessor and the HQLAst will not be created with SUBST tree, but with NUM_LITERAL, then cached. When preprocessing the second where clause, the cache will return the first cached HQLAst which has the **text** value **1** instead of **2**. We can also consider cases where the substitution parameter is of a different type (e.g. LOGICAL), in this case the HQLAst will be different from the one we already cached and will create another one.

This statement should be ignored since I found out what caused the problem. The HQLAst needs to be duplicated when adding it into the cache, not just when it is retrieved.

#9 - 09/11/2023 08:13 AM - Constantin Asofiei

Danut, just to be clear, I'm referring to the HQL AST returned by this code:

```
        HQLLexer lexer = new HQLLexer(reader);
        HQLParser parser = new HQLParser(lexer);

        parser.expression();
        root = (HQLAst) parser.getAST();
        root.fixups(null, null);
```

which exists in both FQLPreprocessor.parse (which is on the code path you mention for FQLPreprocessor.get()) and in FQLPreprocessor(List<RecordBuffer> bound, List<RecordBuffer> definition, String where) constructor, which is in the code path for FQLPreprocessor.translate().

We don't need to re-parse the FQL string if we already have a pristine HQLAst in the cache - when found in the cache, you can use root.duplicate() to get a copy of the pristine AST.

#10 - 09/11/2023 08:18 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Danut, just to be clear, I'm referring to the HQL AST returned by this code:
> [...]
> which exists in both FQLPreprocessor.parse (which is on the code path you mention for FQLPreprocessor.get()) and in FQLPreprocessor(List<RecordBuffer> bound, List<RecordBuffer> definition, String where) constructor, which is in the code path for FQLPreprocessor.translate().
>
> We don't need to re-parse the FQL string if we already have a pristine HQLAst in the cache - when found in the cache, you can use root.duplicate() to get a copy of the pristine AST.

Yes, this is exactly what I did. I use the fql.toFinalExpression() as key and create an entry in the LRUCache using the resulted HQLAst right after the fixups call.

#11 - 09/11/2023 08:27 AM - Dănuț Filimon

I want to test a customer application to see if the current size of the LRUCache (8192) is enough and increase it if necessary. At the same time, the cache is synchronized and a few tests should be made to make sure that there is a performance improvement. I already have a profiling test, but will look into expanding it (to test longer where clauses).

#12 - 09/11/2023 08:57 AM - Dănuț Filimon

Committed 7731a/rev.14733. Added the initial implementation for the HQLAst cache.

- The cache will be created using the CacheManager after deciding on it's size (currently 8192).

#13 - 09/11/2023 11:00 AM - Constantin Asofiei

Dănuț Filimon wrote:

> I want to test a customer application to see if the current size of the LRUCache (8196) is enough and increase it if necessary. At the same time, the cache is synchronized and a few tests should be made to make sure that there is a performance improvement. I already have a profiling test, but will look into expanding it (to test longer where clauses).

I don't think longer where clauses will show much - try to build a test having 100s/1000s of WHERE distinct clauses (non-dynamic). I think it can be as simple as FIND FIRST tt1 WHERE tt1.f1 > 1, and so on, tt1.f1 > 2, etc.

#14 - 09/12/2023 06:39 AM - Dănuț Filimon

Constantin Asofiei wrote:

> I don't think longer where clauses will show much - try to build a test having 100s/1000s of WHERE distinct clauses (non-dynamic). I think it can be as simple as FIND FIRST tt1 WHERE tt1.f1 > 1, and so on, tt1.f1 > 2, etc.

The problem with using simple FIND FIRST statements is that if I want to use a counter variable (tt1.f1 > counter) and have ~1k find statements then the counter will become a substitution parameter and the statement will be cached, meaning that the fql will only be parsed once/twice. While using a dynamic query, I can make sure that the fql is generated each time and easily test the cache access.

I profiled the changes and used dynamic queries with a counter that went up to 10k, the results were the following:

Test (5 runs)	Average (ms)
with cache (Cold)	11176.6
without cache (Cold)	11495.4
with cache (Warm)	4713
without cache (Warm)	4796

The improvements are ~**2.7%** and ~**1.7%** for the cold and warm tests. I also managed to test a large customer application and the astCache reached ~3k entries (3068). It seems that **8192** is too much and using **4096** for the cache size should be a better choice. Let me know what you think.

#15 - 09/12/2023 06:45 AM - Constantin Asofiei

Dănuț Filimon wrote:

> The problem with using simple FIND FIRST statements is that if I want to use a counter variable (tt1.f1 > counter) and have ~1k find statements then the counter will become a substitution parameter and the statement will be cached, meaning that the fql will only be parsed once/twice. While using a dynamic query, I can make sure that the fql is generated each time and easily test the cache access.

I agree. It's kind of hard to determine, even if you use a script to generate 1000 find first tt1 where tt1.f1 > 1 (2, 3,4, 5 to 1000), this is still using a small FQL length, which can be parsed pretty fast in HQLParser.parse.

For #7767, there are ~280k of HQLParser.expression calls, which in turn create some 94mm instances - this in turn affects garbage collector (especially on machines with few CPUs).

> I also managed to test a large customer application and the astCache reached ~3k entries (3068). It seems that **8192** is too much and using **4096** for the cache size should be a better choice. Let me know what you think.

Is this for #7767? I'd like to see the cache usage for this.

#16 - 09/12/2023 06:58 AM - Dănuț Filimon

Constantin Asofiei wrote:

> For #7767, there are ~280k of HQLParser.expression calls, which in turn create some 94mm instances - this in turn affects garbage collector (especially on machines with few CPUs).
>
> > I also managed to test a large customer application and the astCache reached ~3k entries (3068). It seems that **8192** is too much and using **4096** for the cache size should be a better choice. Let me know what you think.
>
> Is this for #7767? I'd like to see the cache usage for this.

It should be of help for #7767 since the cache is searched before the call to HQLParser.expression. I originally tested the application to see the max cache size it reached after browsing the application for ~40-50 minutes. I will test it right away but is the number of calls the only information that you want to know?

#17 - 09/12/2023 07:21 AM - Constantin Asofiei

Dănuț Filimon wrote:

> I will test it right away but is the number of calls the only information that you want to know?

The total number of cache hits (or misses). Set the cache number to something large (like 512k) and check the cache size after #7767 completes. Or check the value of HashMap.modCount for the LRUCache.cache field.

#18 - 09/12/2023 07:50 AM - Dănuț Filimon

Constantin Asofiei wrote:

> The total number of cache hits (or misses). Set the cache number to something large (like 512k) and check the cache size after #7767 completes. Or check the value of HashMap.modCount for the LRUCache.cache field.

So I should leave the cache size to **512k** instead of **8192** and wait for #7767 to be completed before actually checking what the actual size should be? Then besides the current task, what should I continue with to help with #7767?

#19 - 09/12/2023 07:58 AM - Constantin Asofiei

Dănuț Filimon wrote:

> So I should leave the cache size to **512k** instead of **8192** and wait for #7767 to be completed before actually checking what the actual size should be?

Yes. My worry is that if there are really 280k distinct FQLs via dynamic queries then this cache will not help. We will need to look into how to optimize hql.g, too.

> Then besides the current task, what should I continue with to help with #7767?

Let's discuss this at #7767.

#20 - 11/01/2023 06:28 AM - Constantin Asofiei

Constantin Asofiei wrote:

> Dănuț Filimon wrote:
>
> > So I should leave the cache size to **512k** instead of **8192** and wait for #7767 to be completed before actually checking what the actual size should be?
>
> Yes. My worry is that if there are really 280k distinct FQLs via dynamic queries then this cache will not help. We will need to look into how to optimize hql.g, too.

Dănuț, did you get a chance to measure this?

#21 - 11/01/2023 06:35 AM - Dănuț Filimon

Constantin Asofiei wrote:

> Constantin Asofiei wrote:
>
> > Dănuț Filimon wrote:
> >
> > > So I should leave the cache size to **512k** instead of **8192** and wait for #7767 to be completed before actually checking what the actual size should be?
> >
> > Yes. My worry is that if there are really 280k distinct FQLs via dynamic queries then this cache will not help. We will need to look into how to optimize hql.g, too.
>
> Dănuț, did you get a chance to measure this?

I did not. If I remember correctly, when I tested the customer application at the time the cache exceeded 4k entries but not by much. In the first place I did not notice the cache being maxed out at all and I got no performance improvement when testing #7767 (it even increased the time of the scenario by 5-10 seconds).

#22 - 11/01/2023 06:37 AM - Constantin Asofiei

OK, please set the cache size to 512k, run the test, look via the debugger and see what the actual cache size is after the test completes, and post here.

#23 - 11/01/2023 07:43 AM - Alexandru Lungu

Rebased 7731a to latest trunk. It is now at rev. 14811.

#24 - 11/01/2023 08:58 AM - Dănuț Filimon

Constantin Asofiei wrote:

> OK, please set the cache size to 512k, run the test, look via the debugger and see what the actual cache size is after the test completes, and post here.

After running the scenario from #7767, the cache size is **150**.

#25 - 11/01/2023 09:02 AM - Constantin Asofiei

Dănuț Filimon wrote:

> Constantin Asofiei wrote:
>
> > OK, please set the cache size to 512k, run the test, look via the debugger and see what the actual cache size is after the test completes, and post here.
>
> After running the scenario from #7767, the cache size is **150**.

Thanks. Is the cache configurable via directory.xml? I would leave it to 8192 as default and tune it on a case-by-case basis.

#26 - 11/01/2023 03:25 PM - Dănuț Filimon

Constantin Asofiei wrote:

> Is the cache configurable via directory.xml? I would leave it to 8192 as default and tune it on a case-by-case basis.

I committed **7731a/rev.14812** where I made the astCache configurable. Please note that there already exists a cache that can be configured for FQLPreprocessor, so you'll have to use a discriminator. Here is the configuration that you need to place in persistence/cache-size container:

```
<node class="container" name="08">
    <node class="string" name="class-name">
        <node-attribute name="value" value="com.goldencode.p2j.persist.FQLPreprocessor"/>
    </node>
    <node class="integer" name="size">
        <node-attribute name="value" value="8192"/>
    </node>
    <node class="string" name="discriminator">
        <node-attribute name="value" value="ast"/> <!-- important, the value must be "ast" as specified here -
->
    </node>
</node>
```

For any additional information feel free to look at Database Configuration.

- astCache is initialized at server bootstrap
- changed cache size from 8196 to 8192 (overlooked it when I first committed)
- fixed history entries

#27 - 11/02/2023 09:46 AM - Greg Shah

Is this ready for review?

#28 - 11/02/2023 10:17 AM - Dănuț Filimon

Greg Shah wrote:

> Is this ready for review?

Yes.

#29 - 11/02/2023 11:30 AM - Greg Shah

Please set the status and %Done accordingly. That is the "event" that triggers a code review.

#30 - 11/02/2023 11:33 AM - Alexandru Lungu

- % Done changed from 0 to 100

- Status changed from WIP to Review

It seems Danut has stepped away from "events" and "triggers" for a while :)

#31 - 11/02/2023 12:12 PM - Greg Shah

Ovidiu/Constantin: Please review.

#32 - 11/02/2023 03:36 PM - Ovidiu Maxiniuc

- Status changed from Review to Internal Test

Review of 7731a/14811-2.

A straightforward implementation. I am OK with the code. Good job!

#33 - 11/15/2023 09:32 AM - Alexandru Lungu

What internal tests should be done here, Danut? Are you able to do some smoke tests on large applications?

I can take this for a regression test round + performance test round. Please let me know if I should set-up anything before performance testing - is the default value large enough to initiate performance tests now?

#34 - 11/15/2023 09:37 AM - Dănuț Filimon

Alexandru Lungu wrote:

> What internal tests should be done here, Danut? Are you able to do some smoke tests on large applications?
>
> I can take this for a regression test round + performance test round. Please let me know if I should set-up anything before performance testing - is the default value large enough to initiate performance tests now?

If you want to test the implementation, you need to setup the cache size as specified in #7731-26. The current size of the cache is enough to run the performance tests. I can also do a few smoke tests on a large application.

#35 - 11/27/2023 07:35 AM - Constantin Asofiei

I think the changes are OK. I've tested with a large POC performance run, and there was a visible improvement.

#36 - 11/27/2023 11:52 AM - Greg Shah

- Status changed from Internal Test to Merge Pending

This can merge to trunk after 7669a.

#37 - 11/28/2023 09:14 AM - Greg Shah

You can merge now.

#38 - 11/28/2023 09:37 AM - Alexandru Lungu

- Status changed from Merge Pending to Test

Branch 7731a was merged into trunk as rev. 14844 and archived.

#39 - 11/28/2023 09:43 AM - Dănuț Filimon

Database Configuration was updated with the necessary information for configuring the cache size of astCache.