Database - Bug #7167

Associating records from opened buffers to new sessions is slow

03/03/2023 07:21 AM - Alexandru Lungu

Status:	Closed	**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

Related issues:

Related to Database - Feature #4021: better use of session-related resources ...	**New**
Related to Database - Bug #7334: Reclaim used sessions to improve performance	**Test**
Related to Database - Bug #8196: Reduce number of AbstractTempTable._hasRecor...	**Test**

History

#1 - 03/03/2023 07:45 AM - Alexandru Lungu

There are several moments when Persistence$Context.getSession is used (usually by query execution, persistence record loading or transaction starting).

The performance hit was already noticed in #5919-59 and #5237-5, but FWD still faces today a full BufferManager.activeBuffers iteration to retrieve the records which are not associated with a new session. In a large customer application POC:

- ~1.700 sessions are created
- ~40.000 active buffers actually have a record which should be associated with the new session
- ~250.000 active buffers don't have a record at all, so no need to associate
- ~2.250.000 active buffers are not matching the persistence (e.g. are temporary buffers and the session is for a permanent database)

For the application I am testing, this means:

- 98% of the active buffers are iterated, but not associated.
- on average, there are ~1500 active buffers on a session creation
- on average, 23 buffers should be found and associated
- on average, 147 buffers are empty
- on average, 1323 buffers are not for the right persistence

Having 1.700 sessions closed and re-created may be a problem in the first place (also noted in #5919-59), as we are losing record caches. Note that the session to the temp database has a very long lifespan (having a volatile nature). This issue mostly refers to the sessions over the persistence database.

#2 - 03/16/2023 06:04 AM - Alexandru Lungu

- Related to Feature #4021: better use of session-related resources such as PreparedStatements and caches added

#3 - 03/16/2023 06:17 AM - Alexandru Lungu

After a bit of investigation, the database sessions com.goldencode.p2j.persist.orm.Session are closed quite eagerly. Also, the ones closed are mostly sessions for the persistent database. Each time the session count reaches 0, the session is closed. A session close also clears the cache and closes the underlying connection.

Eric, I see #4021 is something near this problem, implying that the session shouldn't be closed that often. This way, we shouldn't address the BufferManager.activeBuffers iteration bottleneck, but the session opening/closing issue. I will test if FWD is doing better now without closing sessions, comparing to your empirical test in #4021.

#4 - 07/10/2023 07:17 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Dănuț Filimon

This is basically fixed by the reclaimable session Danut implemented recently in #7334. This is in trunk now.

#5 - 11/07/2023 05:06 PM - Eric Faulhaber

Can this task be closed?

#6 - 11/08/2023 03:46 AM - Alexandru Lungu

Eric, in theory, this can be closed. However, I am waiting for some performance tests to check that BufferManager.activeBuffers is no longer a problem. Reaching you back asap.

#7 - 11/10/2023 11:02 AM - Alexandru Lungu

- % Done changed from 0 to 100

- Status changed from WIP to Internal Test

With some slim analysis, I still see BufferManager.activeBuffers in top 50 performance items (being called ~100 times in a POC). Most probably, we need to increase the session reclaiming time. By default, the reclaiming time is 1s. I will increase the time to 2s / 5s / 10s and see if there are any major changes. In theory, this can be configurable from directory.xml, so there is nothing more to be worked here per-se.

Still (I think this was in another task I will search), we need to see if we can reclaim Sessions but switching connections to avoid starving.

#8 - 11/22/2023 05:08 AM - Alexandru Lungu

Danut, please re-test the large POC and see if why we are still having spikes on BufferManager.activeBuffers. Please attempt to check with 10s of session lifespan first. After, do some debugging and understand why are we still expiring sessions - are they so sparsely used?

#9 - 11/22/2023 05:09 AM - Alexandru Lungu

Also, please remake the statistic on #7167-1 now with 1s of lifespan vs 10s of lifespan.

#10 - 11/23/2023 09:09 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, please re-test the large POC and see if why we are still having spikes on BufferManager.activeBuffers. Please attempt to check with 10s of session lifespan first. After, do some debugging and understand why are we still expiring sessions - are they so sparsely used?

I retested the POC and got the following results for the session lifespan:

	1s lifespan (ms)	10s lifespan (ms)	Difference (%)
average of last 20 runs	16200.6	15305.4	-5.525%
total average	15799	14789.8	-6.387%

EDIT: The results of the tests are the average of 5 cold runs.

Also, please remake the statistic on #7167-1 now with 1s of lifespan vs 10s of lifespan.

I also retested the scenario for the case when the reclaiming is disabled and obtained:

Counts	Reclaiming disabled	1s lifespan	10s lifespan
No matching persistence	285708435	5874605	3564224
No records to associate	31051569	26227813	26226169
Records associated	2517852	614027	613197
Sessions created	124912	1127	89
Sessions reclaimed	0	123767	124805

#11 - 01/05/2024 03:56 AM - Alexandru Lungu

Danut, I don't think that the shift to 10s looks OK with these statistics. Mind that we "lock" a connection from the pool while we keep the session "reclaimable", so 1s is enough in the current context to avoid starving. Mind that we already faced starving in some applications with 1s (locking the connection pool). If we are to increase the lifespan to **10s**, we need to take care of connection releasing first. I think there is a task for it already - close connection when session enters the "reclaiming" state (please search and relate). We can expect for some performance downgrade from "unpreparing" the statements, but we still have the DMO cache preserved. **This is lead 1**

It is curious why "No records to associate" and "Records associated" are similar, even if there are ~90% less sessions created. After some investigation, activeBuffers is also used in AbstractTempTable._hasRecords. I don't have a tracker on this path, so please investigate further how often is activeBuffers hit from AbstractTempTable._hasRecords. If you see heavy traffic there, please create a new task on that (as it is not related to this caption: "Associating records from opened buffers to new sessions is slow". **This is lead 2**

#12 - 01/15/2024 03:56 AM - Dănuț Filimon

I've got the results of how often is activeBuffers hit from AbstractTempTable._hasRecords, and how often it is hit from other sources.

Counts	from AbstractTempTable	from BufferManager/other
Number of calls	11312	1188
Nomatching persistence	35345444	24577866
Norecords to associate	26237166	1974
Records associated	601546	1123

Most of the records are associated through AbstractTempTable._hasRecords, as seen above.

EDIT: The test was done with POC with default session lifespan.

#13 - 01/15/2024 04:03 AM - Dănuț Filimon

- Related to Bug #7334: Reclaim used sessions to improve performance added

#14 - 01/15/2024 04:25 AM - Dănuț Filimon

- Related to Bug #8196: Reduce number of AbstractTempTable._hasRecords calls to avoid BufferManager.activeBuffers added

#15 - 01/30/2024 10:01 AM - Eric Faulhaber

- Status changed from Internal Test to Closed

I understand the work begun here is being continued in #8196.