redmineorg-copy202205 - Vote #77631

#lock_nested_set very slow on mysql with thousands of subtasks

2022/05/09 18:01 - Admin Redmine

ステータス:	Reopend	**開始日:**	2022/05/09
優先度:	通常	**期日:**	
担当者:		**進捗率:**	0%
カテゴリ:	Performance_53	**予定工数:**	0.00時間
対象バージョン:	Candidate for next major release_32	**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/23318	**status_id:**	8
category_id:	53	**tracker_id:**	1
version_id:	32	**plus1:**	0
issue_org_id:	23318	**affected_version:**	
author_id:	123866	**closed_on:**	
assigned_to_id:	1	**affected_version_id:**	
comments:	18		

説明

I have a complex hierarchy of around 15000 issues in redmine, where an issue of this set could potentially have 3000 subtasks.

When doing CRUD operations on such issue, I notified significant slow downs, caused by the @lock_nested_set@ function.

Here is the profiling for the query actually run in @lock_nested_set@:

```
SELECT `issues`.`id` FROM `issues` WHERE (root_id IN (SELECT root_id FROM issues WHERE id IN (70395,70389)))
Y `issues`.`id` ASC FOR UPDATE;

+-------+
|       |
.........
| 70371 |
| 70373 |
| 70375 |
| 70377 |
| 70379 |
| 70381 |
| 70383 |
| 70385 |
| 70387 |
| 70389 |
| 70391 |
| 70393 |
+-------+
2932 rows in set (2.70 sec)

mysql> show profile for QUERY 1;
+--------------------------------+----------+
| Status                         | Duration |
+--------------------------------+----------+
| starting                       | 0.000025 |
| Waiting for query cache lock   | 0.000004 |
| checking query cache for query | 0.000081 |
| checking permissions           | 0.000003 |
| checking permissions           | 0.000004 |
| Opening tables                 | 0.000038 |
| System lock                    | 0.000017 |
| init                           | 0.000056 |
| optimizing                     | 0.000013 |
| statistics                     | 0.000023 |
¦ preparing                                          ¦ 0.000009 ¦
¦ executing                                          ¦ 0.000002 ¦
¦ Sorting result                             ¦ 0.000005 ¦
¦ Sending data                                    ¦ 0.000049 ¦
¦ optimizing                                       ¦ 0.000015 ¦
¦ statistics                                     ¦ 0.000047 ¦
¦ preparing                                          ¦ 2.690165 ¦
¦ end                                                      ¦ 0.000009 ¦
¦ query end                                          ¦ 0.000067 ¦
¦ closing tables                             ¦ 0.000010 ¦
¦ freeing items                               ¦ 0.000038 ¦
¦ logging slow query                      ¦ 0.000002 ¦
¦ logging slow query                      ¦ 0.000163 ¦
¦ cleaning up                                        ¦ 0.000003 ¦
+--------------------------------+----------+
24 rows in set (0.00 sec)
```

It takes around 3 seconds to execute the whole query.

I think the main problem is with the nested SELECT statement. If I execute it separately, then paste its results directly into the main query, the query is much faster:

```
mysql> SELECT root_id FROM issues WHERE id IN (70395,70389);
+---------+
¦ root_id ¦
+---------+
¦     45083 ¦
+---------+
1 row in set (0.00 sec)

SELECT `issues`.`id` FROM `issues` WHERE (root_id IN (45083))  ORDER BY `issues`.`id` ASC FOR UPDATE;

+-------+
¦          ¦
.........
¦ 70371 ¦
¦ 70373 ¦
¦ 70375 ¦
¦ 70377 ¦
¦ 70379 ¦
¦ 70381 ¦
¦ 70383 ¦
¦ 70385 ¦
¦ 70387 ¦
¦ 70389 ¦
¦ 70391 ¦
¦ 70393 ¦
+-------+
2932 rows in set (0.01 sec)
```

I am not an expert in sql queries, and don't want to break anything... Shouldn't we use a JOIN instead?

```
Environment:
    Redmine version                    3.3.0.stable
    Ruby version                          2.2.2-p95 (2015-04-13) [x86_64-linux]
    Rails version                      4.2.6
    Environment                           development
    Database adapter                 Mysql2
```

journals

FYI, I was able to change the statement used in @lock_nested_set@

from:

```
self.class.reorder(:id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?))", sets_to_lock).lock
```

to:

```
self.class.reorder(:id).joins("INNER JOIN #{self.class.table_name} t2 ON #{self.class.table_name}.root_id = t2.root_id").where("t2.id
N (?)", sets_to_lock).distinct.lock.ids
```

so far the later returns instantly, with the same results as the former. I don't know though how it affects the locking mechanism.

Passed all tests.

The following is a diff of changes made by Stephane Evr.

```
Index: lib/redmine/nested_set/issue_nested_set.rb
===================================================================
--- lib/redmine/nested_set/issue_nested_set.rb    (revision 15663)
+++ lib/redmine/nested_set/issue_nested_set.rb    (working copy)
@@ -158,7 +158,7 @@
           self.class.reorder(:id).where(:root_id => sets_to_lock).lock(lock).ids
         else
           sets_to_lock = [id, parent_id].compact
-          self.class.reorder(:id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?
lock.ids
+          self.class.reorder(:id).joins("INNER JOIN #{self.class.table_name} t2 ON #{self.class.table_name}.root_id = t
ere("t2.id IN (?)", sets_to_lock).distinct.lock.ids
         end
       end
```

-- -- Go MAEDA wrote: > Passed all tests. Did you run the tests with mysql? Because I get an error with Postgresql with the patch applied (FOR UPDATE not allowed with DISTINCT). Tests pass without @.distinct@ -- Jean-Philippe Lang wrote: > Did you run the tests with mysql? Because I get an error with Postgresql with the patch applied (FOR UPDATE not allowed with DISTINCT). > Tests pass without @.distinct@ Sorry, I run the tests only with sqlite. -- Patch committed without the @.distinct@ call, thanks. -- MySQL "IssueNestedSetConcurrencyTest#test_concurrency" and "IssueNestedSetConcurrencyTest#test_concurrent_subtasks_creation" fail. http://www.redmine.org/builds/logs/build_trunk_mysql_ruby-2.3_3063.html It may be related with #19344. -- -- r15891 reverted. -- I found an alternative solution which passes the tests, although not as fast as with a JOIN. Here, I simply group by issue id in the original subquery: SELECT `issues`.* FROM `issues` WHERE (root_id IN (SELECT root_id from issues WHERE id IN (96457,96455) *GROUP BY id*)) ORDER BY `issues`.`id` ASC; So the statement would be rewritten as :

```
self.class.reorder(:id).where("root_id IN (SELECT root_id FROM #{self.class.table_name} WHERE id IN (?) GROUP BY id)",
lock).lock.ids
```

I finally found how to refactor the JOIN Statement and keep the initial performance improvements. Please find attached a patch against the master branch. This passed the issue_nested_set_concurrency_test on MySQL.

I ran the test 10 times and all passed

Reverted r16053 because of SQL error with PostgreSQL.

PostgreSQL does not accept the DISTINCT in the subquery:
http://www.redmine.org/builds/logs/build_trunk_postgresql_ruby-2.3_3124.html

One option would be to have a statement specfic to MySQL just like we already have one for SQL Server.

Jean-Philippe Lang wrote:

> PostgreSQL does not accept the DISTINCT in the subquery:

http://www.redmine.org/builds/logs/build_trunk_postgresql_ruby-2.3_3124.html

One option would be to have a statement specfic to MySQL just like we already have one for SQL Server.

Perhaps this is the solution. I don't know if PostgreSQL is affected by this problem.

Based #note-13 patch.
This patch reduces #19344 test failure times on my CentOS7 mariadb-5.5.52-1.el7.x86_64.
(Not always test pass. About 50% time test passes.)

---

related_issues

relates,New,19344,MySQL 5.6: IssueNestedSetConcurrencyTest#test_concurrency : always fails

履歴

#1 - 2022/05/10 17:07 - Admin Redmine

- カテゴリ を Performance_53 にセット

- 対象バージョン を Candidate for next major release_32 にセット