We just updated our Foreman install with ~14000 hosts to 1.8.2. When testing the install everything was fine, but as soon as users (~200) where starting to use the Frontend again, everything exploded, because of the reports dashboard widgets performance.

A single dashboard pageload took over 80 seconds on average because the DBquery didn't return before that, so 100+ users firing the same long running query against the db blocked the frontends completely and the DB wasn't too happy either.

So please add an additional index for that dashboard, others with "a lot" of hosts will probably run into that same problem, ymmv depending on number of hosts & report retention time.?

That was my prose, this is actual information from my DB colleague:

Data in 'reports': ~26GB

The query the widget performs on every Dashboard pageload:
```
SELECT "reports".* FROM "reports" WHERE (status <> 0) AND ("reports"."reported_at" >= '2015-06-12') ORDER BY "reports"."reported_at" DESC NULLS LAST LIMIT 6;
```

The pgsql 'explain' for that query said:
```
Limit  (cost=3278039.34..3278039.35 rows=6 width=538)
  ->  Sort  (cost=3278039.34..3282344.48 rows=1722055 width=538)
        Sort Key: reported_at
  (5 rows)
```

So the DB guy created a new index:
```
create index concurrently on reports (reported_at DESC NULLS LAST);
```

The pgsql 'explain' after that index now says:
```
Limit  (cost=0.43..19.91 rows=6 width=538)
  ->  Index Scan using reports_reported_at_idx on reports  (cost=0.43..5096157.42 rows=1569360 width=538)
  Index Cond: (status <> 0) AND (reported_at >= '2015-06-12 00:00:00'.::timestamp without time zone))
  (4 rows)
```

And now we again have Dashboard pagaloads in under a second.

**Associated revisions**

Revision 3dfaf19d - 06/25/2015 07:35 AM - Lukas Zapletal

Fixes #10879 - reported_at column is NOT NULL

Allows for better optimisation of queries on PostgreSQL with scoped_search.
Hello,

thanks for the report. Before I start digging, is it the NULL clause that makes the existing index index_reports_on_reported_at not effective?

Sorry I mean status field.

It looks like we need a composed index, we have the following indices:

```ruby
add_index "reports", ["host_id"], :name => "index_reports_on_host_id"
add_index "reports", ["reported_at", "host_id"], :name => "index_reports_on_reported_at_and_host_id"
add_index "reports", ["reported_at"], :name => "index_reports_on_reported_at"
add_index "reports", ["status"], :name => "index_reports_on_status"
```

Are you sure your recommendation create index concurrently on reports (reported_at DESC NULLS LAST) really solves the issue? I'd expect something like:

```ruby
add_index "reports", ["reported_at", "status"], :name => "index_reports_on_reported_at_and_status"
```

For the record the relevant bit is this one:

```ruby
def latest_events
  # 6 reports + header fits the events box nicely...
  Report.authorized(:view_reports).my_reports.interesting.search_for('reported > "7 days ago"').limit(6).includes(:host)
end
```

And it's get more complicated for non-admin users where we need to join tables. For admin users the query is simple (but still problematic).

It looks like scoped_search adds the "NULLS LAST" statement which makes our index non-effective. We can workaround this in our codebase like this:

```ruby
irb(main):008:0> Report.interesting.search_for('reported > "7 days ago"').limit(6).includes(:host)
  Report Load (0.5ms)  SELECT "reports".* FROM "reports" WHERE (status <> 0) AND (("reports"."reported_at" >='2015-06-16')) ORDER BY "reports"."reported_at" DESC NULLS LAST LIMIT 6
=> []
```

```ruby
irb(main):009:0> Report.interesting.where("reported_at > ?", 7.days.ago).limit(6).includes(:host)
  Report Load (0.7ms)  SELECT "reports".* FROM "reports" WHERE (status <> 0) AND (reported_at > '2015-06-15 09:14:05.855143') LIMIT 6
=> []
```
The problem can be with NULL records. I don't recall why we don't run NOT NULL for reported_at column. The new API V2 requires this field to be present as well as our model (:presence => true, :uniqueness => {:scope => :host_id}). Therefore I think the proper way to fix this is to make the column a non-null one. That should fix the scoped_search query builder as well and no extra index is necessary.

#4 - 06/22/2015 05:27 AM - Ohad Levy
Lukas Zapletal wrote:

The problem can be with NULL records. I don't recall why we don't run NOT NULL for reported_at column. The new API V2 requires this field to be present as well as our model (:presence => true, :uniqueness => {:scope => :host_id}). Therefore I think the proper way to fix this is to make the column a non-null one. That should fix the scoped_search query builder as well and no extra index is necessary.

any idea how reported_at could be null? it was never designed to be null afaik.

#5 - 06/22/2015 06:00 AM - Lukas Zapletal
Strange, even when I change reported_at to be non-null column, scoped search still adds "NULLS LAST" caluse:

```ruby
irb(main):011:0> Report.interesting.search_for('reported > "7 days ago"').limit(6).includes(:host)
  Report Load (0.5ms)  SELECT "reports".* FROM "reports" WHERE (status <> 0) AND ("reports"."reported_at" >= '2015-06-16') ORDER BY "reports"."reported_at" DESC NULLS LAST LIMIT 6 => []
irb(main):012:0> Report.interesting.where("reported_at > ?", 7.days.ago).order("reported_at DESC").limit(6).includes(:host)
  Report Load (0.6ms)  SELECT "reports".* FROM "reports" WHERE (status <> 0) AND (reported_at > '2015-06-15 09:59:51.578137') ORDER BY reported_at DESC LIMIT 6 => []
```

#6 - 06/22/2015 06:02 AM - Andreas Bogacki
Hi,
I am the DB guy mentioned in the OP.
I just created the index this way to get a production system back into working order.
I do not think that you need:

```ruby
add_index "reports", ["reported_at", "status"], :name => "index_reports_on_reported_at_and_status"
```

As you can see from the first explain sorting by reported_at triggers a full table scan which is expensive:
There is another index `index_reports_on_reported_at` on `reported_at` which cannot be used by the query due to DESC being used together with NULLS LAST in the query.

The condition `status <> 0` is already covered by the index `index_reports_on_status`.

The problem with degraded performance was gone right after the index was created.

Please let me know if you need any more information.

BTW: PostgreSQL 9.4.3 is used for this installation.

---

**#7 - 06/22/2015 06:08 AM - The Foreman Bot**
- Status changed from New to Ready For Testing
- Pull request https://github.com/theforeman/foreman/pull/2478 added
- Pull request deleted ()

**#8 - 06/22/2015 06:10 AM - Andreas Bogacki**

Lukas Zapletal wrote:

```
Strange, even when I change reported_at to be non-null column, scoped search still adds "NULLS LAST" clause:

  [...]
```

Is the code for scoped search checking columns for "not null" modifiers in the DB and generating different SQL statements based on that check? If not then changing the column in the database will have no effect on the SQL generated by the application.

**#9 - 06/22/2015 06:13 AM - Lukas Zapletal**

Hello,

I have created a patch that adds a migration to add "NOT NULL" on the created_at column. I believe PostgreSQL optimizer will then ignore "NULLS LAST" clause and use the index. I am not sure tho if this needs reindexing after "NOT NULL" was added. I am currently lacking of an instance with decent amount of reports.

**#10 - 06/22/2015 06:15 AM - Lukas Zapletal**
Yeah I was hoping that optimizer will ignore "NULLS LAST" when the column is actually "NOT NULL" :-(

It seems the gem always adds the clause: https://github.com/wvanbergen/scoped_search/blob/master/lib/scoped_search/query_builder.rb#L517-L521
(it is hardcoded)

#11 - 06/22/2015 06:28 AM - Lukas Zapletal

Ok I filed a patch against scoped_search not to add this clause when column is NOT NULL.

Until then we can modify our widget not to use scoped_search (my patch for review).

#12 - 06/22/2015 11:38 AM - Andreas Bogacki

Lukas Zapletal wrote:

Hello,

I have created a patch that adds a migration to add "NOT NULL" on the created_at column. I believe PostgreSQL optimizer will then ignore "NULLS LAST" caluse and use the index. I am not sure tho if this needs reindexing after "NOT NULL" was added. I am currently lack of an instance with decent amount of reports.

I just did check that on a copy of my production DB.

The PostgreSQL optimizer does not optimize away the NULLS LAST in the SQL statement.

foreman_prod=# \d+ reports
<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>integer</td>
<td>not null default nextval('reports_id_seq'::regclass)</td>
<td>plain</td>
</tr>
<tr>
<td>host_id</td>
<td>integer</td>
<td>not null</td>
<td>plain</td>
</tr>
<tr>
<td>reported_at</td>
<td>timestamp without time zone</td>
<td>not null</td>
<td>plain</td>
</tr>
<tr>
<td>created_at</td>
<td>timestamp without time zone</td>
<td>not null</td>
<td>plain</td>
</tr>
<tr>
<td>updated_at</td>
<td>timestamp without time zone</td>
<td>not null</td>
<td>plain</td>
</tr>
<tr>
<td>status</td>
<td>bigint</td>
<td></td>
<td>plain</td>
</tr>
<tr>
<td>metrics</td>
<td>text</td>
<td></td>
<td>extended</td>
</tr>
</tbody>
</table>

Indexes:
- "reports_pkey" PRIMARY KEY, btree (id)
- "index_reports_on_host_id" btree (host_id)
- "index_reports_on_reported_at" btree (reported_at)
- "index_reports_on_reported_at_and_host_id" btree (reported_at, host_id)
- "index_reports_on_status" btree (status)

Foreign-key constraints:
- "reports_host_id_fk" FOREIGN KEY (host_id) REFERENCES hosts(id)

foreman_prod=# explain SELECT "reports".* FROM "reports" WHERE (status <> 0) AND (("reports"."reported_at" >= '2015-06-12')) ORDER BY "reports"."reported_at" DESC NULLS LAST LIMIT 6;

---

Limit (cost=328846.53..328846.54 rows=6 width=539)
-> Sort (cost=328846.53..336132.00 rows=2914188 width=539)
  Sort Key: reported_at
  -> Seq Scan on reports (cost=0.00..276610.26 rows=2914188 width=539)
  Filter: ((status <> 0) AND (reported_at >= '2015-06-12 00:00:00'::timestamp without time zone))
Since this behaviour of the optimizer in PostgreSQL is unexpected/surprising we are trying to find out if this is a bug in the optimizer or if this is a shortcoming that can not be addressed.

For completeness I have also checked if the creation of the combined index add_index "reports", ["reported_at", "status"].:name => "index_reports_on_reported_at_and_status" does help and can confirm that it does not:

```sql
foreman_prod=# create index on reports (status, reported_at);
```

Valid solutions to fix the problem:

- either remove NULLS LAST from the SQL query in the application code
or create index concurrently on reports (reported_at DESC NULLS LAST);

#13 - 06/25/2015 07:36 AM - Dominic Cleal
- Assignee set to Lukas Zapletal
- Legacy Backlogs Release (now unused) set to 35

#14 - 06/25/2015 08:01 AM - Lukas Zapletal
- Status changed from Ready For Testing to Closed
- % Done changed from 0 to 100

Applied in changeset 3dfaf19d96933e51bca203477d94e4ad45adf23.