openQA Project - action #94667

coordination # 80142 (Blocked): [saga][epic] Scale out: Redundant/load-balancing deployments of openQA, easy containers, containers on kubernetes

coordination # 92854 (New): [epic] limit overload of openQA webUI by heavy requests

Optimize products/machines/test_suites API calls

2021-06-24 14:01 - tinita

Status:	New	**Start date:**	2021-06-24
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Feature requests	**Estimated time:**	0.00 hour
Target version:	future		
Difficulty:			

Description

Motivation

The SQL to fetch the data is inefficient. For each item in the result set another SELECT FROM job_settings is executed. For https://openqa.suse.de/admin/test_suites that means about 2500 SELECTs currently.
Also the costly DateTime columns are fetched, although they are not needed.

Acceptance criteria

- AC1: the number of SQL queries doesn't depend on the number of items in the table anymore
- AC2: Less columns are fetched

Suggestions

- All three /api/v1/products, /api/v1/machines, /api/v1/test_suites are handled by the same code in lib/OpenQA/WebAPI/Controller/API/V1/Table.pm
- Enable DBIC_TRACE=1 environment variable to see all SQL
- https://github.com/os-autoinst/openQA/pull/3969 has a similar optimizations
- This could be a good task for someone not so familiar with DBIx::Class yet

History

#1 - 2021-06-24 14:02 - tinita

- Description updated

#2 - 2021-06-24 14:11 - tinita

- Description updated

#3 - 2021-06-24 16:19 - tinita

- Description updated

#4 - 2021-06-24 16:19 - tinita

- Description updated

#5 - 2021-08-27 14:46 - tinita

- Status changed from Workable to New

Set to New because not estimated yet

#6 - 2021-09-10 15:02 - okurz

- Subject changed from Optimize products/machines/test_suites API calls to [easy][beginner] Optimize products/machines/test_suites API calls

#7 - 2021-09-13 10:55 - tinita

- Subject changed from [easy][beginner] Optimize products/machines/test_suites API calls to Optimize products/machines/test_suites API calls

I had a look into this on the weekend, since the actual change is simple. But then a test failed unexpectedly:
t/40-script_load_dump_templates.t
So I don't consider it a beginner task anymore.

The order of the products/machines seems to be different than before, and this is also visible in the WebUI.
So the join with the *_settings table results in a different order (which means so far we simply relied on the default postgresql order).

Since I will be on vacation I'll post the actual patch for prefetching the settings here:

```
diff --git a/lib/OpenQA/WebAPI/Controller/API/V1/Table.pm b/lib/OpenQA/WebAPI/Controller/API/V1/Table.pm
index a8591f721..f86e9116c 100644
--- a/lib/OpenQA/WebAPI/Controller/API/V1/Table.pm
+++ b/lib/OpenQA/WebAPI/Controller/API/V1/Table.pm
@@ -108,15 +108,30 @@ sub list {
         }
         if ($have) {
             for my $par (@$key) {
-                $search{$par} = $self->param($par);
+                $search{"me.$par"} = $self->param($par);
             }
         }
     }

     my @result;
+    my $sql_field = {
+        Machines   => 'machine',
+        TestSuites => 'test_suite',
+        Products   => 'product',
+    }->{$table};
     eval {
         my $rs = $self->schema->resultset($table);
-        @result = %search ? $rs->search(\%search) : $rs->all;
+        @result = $rs->search(
+            \%search,
+            {
+                # 'prefetch' would select too much unneeded columns
+                # https://metacpan.org/pod/DBIx::Class::ResultSet#collapse
+                join      => 'settings',
+                collapse  => 1,
+                '+select' => [
+                    qw(settings.id settings.key settings.value),
+                    "settings.${sql_field}_id"
+                ]});
     };
     my $error = $@;
     if ($error) {
```