openQA Project - action #35914

Changes to Job::duplicate

2018-05-04 13:58 - szarate

Status:	Resolved	**Start date:**	2018-05-04
Priority:	High	**Due date:**	
Assignee:	coolo	**% Done:**	0%
Category:	Feature requests	**Estimated time:**	0.00 hour
Target version:	Done		
Difficulty:	hard		

Description

This is a long story biting us from time to time.

Apparently changes in https://github.com/os-autoinst/openQA/pull/1623, made a problem that was underlying in our code a bit more obvious.

Now, according to mkravec:

```
There is ~10% chance that CaaSP cluster will have incomplete job. If incomplete happens, then:
- before this change it failed in "organized" way - jobs started cloning and loosing dependencies,
 so they never recovered and eventually it all died
- now it all goes "kaboom" and weird things happen
For example:
- https://openqa.suse.de/tests/1661011#settings - how does this job have 2 children QAM-CaaSP-admi
n
- https://openqa.suse.de/tests/1661007 was cloned but lost 1 dependency during that
- https://openqa.suse.de/tests/1652849 how can this job find 8 new dependencies after being cloned
 (it recovered & passed fine at at end)
- https://openqa.suse.de/tests/1652962 how can incomplete job have higher ID than one that passed
(1652905) - causing incomplete result being displayed in result overview
I reschedule ISO when this happens.
```

It was discussed during a meeting, that a possible solution was:

- When a Cluster Job is posted (via iso), create a clusterID
- Add said clusterID to all of the jobs spawned that belong to the same cluster
- When a user wants to restart one job from the cluster, look for all of the jobs with the same clusterID, and restart them (all or nothing)

In the meantime: the Job::dublicate function needs to stop being so smart. And refactored.

AC1: Cluster jobs are no longer displaying the behaviour described by mkravec or Ettore. (I.e jobs with missing or misconfigured dependencies)
AC1.1: ClusterID is introduced and openQA/Scheduler are using it automatically
AC1.2: When a single job, from a cluster is restarted/cancelled, the whole cluster behaves as a Borg Collective, and restarts or gets cancelled.
AC1.3: Complexity of Job::duplicate is reduced
AC2: Job::duplicate function is refactored
AC3: Proper unit tests for cases with chained, and parallel with multiple dependencies are written

Related issues:

Related to openQA Project - action #34504: [tools][sporadic] Job's auto_dupli...	**Resolved**	**2018-04-09**
Precedes openQA Project - coordination #32851: [tools][EPIC] Scheduling redesign	**Resolved**	**2018-05-05**

History

#1 - 2018-05-04 13:58 - szarate

Quoting Ettore

dasantiago wrote:

> My changes only adds a dependency that was missing, so because of that change it shouldn't lose anything.

Yes and no, as it's not only adding, before it was just skipping - if you look closely at https://github.com/os-autoinst/openQA/pull/1623/files#diff-85ae48e70a5c110c9e439c3a5ea28d5fR759 you can also skip other child deps (see next()) and ignore the other conditions down while cycling, that could call duplicate (recursively, again) and i suspect that can bring also to loose duplicated jobs dependencies; even if unpleasant, duplicate() looks buggy, to fix properly this we would need to rewrite it from scratch

> For example:
> https://openqa.suse.de/tests/1661011#settings
>
> This probably was introduced with my change, but the losing deps? :-(
>
> Is there any multimachine test environment that i can use for my needs? Or is there any way to simulate a multi machine environment?

#2 - 2018-05-04 14:09 - szarate

- Description updated

- Difficulty set to hard

I think we can start just with AC1, which is easy enough to be implemented in one sprint (Hopefully)

#3 - 2018-05-04 14:10 - szarate

- Related to action #34504: [tools][sporadic] Job's auto_duplicate fails to duplicate job dependencies added

#4 - 2018-05-09 07:49 - szarate

- Assignee set to szarate

#5 - 2018-05-09 08:07 - dasantiago

szarate wrote:

> Quoting Ettore
>
> dasantiago wrote:
>
> > My changes only adds a dependency that was missing, so because of that change it shouldn't lose anything.
>
> Yes and no, as it's not only adding, before it was just skipping - if you look closely at https://github.com/os-autoinst/openQA/pull/1623/files#diff-85ae48e70a5c110c9e439c3a5ea28d5fR759 you can also skip other child deps.

The next is to avoid duplicating jobs already duplicated. Please check the if condition https://github.com/os-autoinst/openQA/pull/1623/files#diff-85ae48e70a5c110c9e439c3a5ea28d5fR750

Also if you compare with the previous commit, it's the same condition that launches the next.

The function implements the depth first search algorithm https://en.wikipedia.org/wiki/Depth-first_search

#6 - 2018-05-09 09:27 - EDiGiacinto

dasantiago wrote:

> szarate wrote:
>
> > Quoting Ettore
> >
> > dasantiago wrote:
> >
> > > My changes only adds a dependency that was missing, so because of that change it shouldn't lose anything.
> >
> > Yes and no, as it's not only adding, before it was just skipping - if you look closely at https://github.com/os-autoinst/openQA/pull/1623/files#diff-85ae48e70a5c110c9e439c3a5ea28d5fR759 you can also skip other child deps.
>
> The next is to avoid duplicating jobs already duplicated. Please check the if condition https://github.com/os-autoinst/openQA/pull/1623/files#diff-85ae48e70a5c110c9e439c3a5ea28d5fR750
>
> Also if you compare with the previous commit, it's the same condition that launches the next.

Yes, i believe the point is the recursiveness impact of your changes

> The function implements the depth first search algorithm https://en.wikipedia.org/wiki/Depth-first_search

Which is one of the basis of graph search algorithms, i would not guess anything different to going on there - but i guess you already know that since you are linking me a wikipedia page :)

If it did correctly then we didn't see such problems - also, we should cover it with extracting the graph algorithm then from the schema / and having separate test for the specific function, but i'm afraid it's not just a simple plain implementation so i know it's harder than it sounds.

Also as you can see yourself from DFS, it can be implemented in iterative way instead of using recursion :)

But in graph algorithms there are a lot of ways to achieve what we need here, like dynamic programming techniques would help a lot to reduce mostly the domain of our problem since we don't have big graphs to handle - what i'm saying here is simply: we should refactor it, decouple it from schema so we can actually test it, and then apply it to the real schema.

#7 - 2018-05-09 10:51 - dasantiago

EDiGiacinto wrote:

> Which is one of the basis of graph search algorithms, i would not guess anything different to going on there - but i guess you already know that since you are linking me a wikipedia page :)

I just added more info into the ticket, as someone might have trouble to understand the algorithm/code.

> If it did correctly then we didn't see such problems

This is were i don't totally agree with you. I'm not saying that the code is perfect and it doesn't contains bugs, but the issue existed for some time and since this seems to happen only in MM jobs, I'm a bit more inclined to think that there's a problem somewhere else - But it's only my opinion

> Also as you can see yourself from DFS, it can be implemented in iterative way instead of using recursion :)

Everything can be implemented without using recursion. Using recursion is just convenient in some cases. Looping through graph nodes is one of those cases :-)

> we should refactor it, decouple it from schema so we can actually test it, and then apply it to the real schema.

This is where i disagree the most. Because it seems that we should immediately rewrite the code. However i believe that before doing any changes we should start by understanding the issue. Which we don't. Doing changes before understanding what's wrong is usually not a good idea.

#8 - 2018-05-09 12:45 - EDiGiacinto

dasantiago wrote:

> EDiGiacinto wrote:
>
> > Which is one of the basis of graph search algorithms, i would not guess anything different to going on there - but i guess you already know that since you are linking me a wikipedia page :)
>
> I just added more info into the ticket, as someone might have trouble to understand the algorithm/code.
>
> > If it did correctly then we didn't see such problems
>
> This is were i don't totally agree with you. I'm not saying that the code is perfect and it doesn't contains bugs, but the issue existed for some time and since this seems to happen only in MM jobs, I'm a bit more inclined to think that there's a problem somewhere else - But it's only my opinion

I'm not saying that it's bugfree the whole function, but just that recursion imho bite your change in unwanted sides.

> > Also as you can see yourself from DFS, it can be implemented in iterative way instead of using recursion :)
>
> Everything can be implemented without using recursion. Using recursion is just convenient in some cases. Looping through graph nodes is one of those cases :-)

Of course it's convenient - iteration here possibly would have the function easier to read. (not even talking about performance impact here)

> > we should refactor it, decouple it from schema so we can actually test it, and then apply it to the real schema.
>
> This is where i disagree the most. Because it seems that we should immediately rewrite the code. However i believe that before doing any changes we should start by understanding the issue. Which we don't. Doing changes before understanding what's wrong is usually not a good idea.

I feel here you are scared about touching the code? If something requires to be re-thought, then i think we should do when necessary - imho this code section calls a proper rewrital/implementation.

Indeed refactoring is not about changing the same function, but write it from scratch with a good testing unit right on the side of it (TDD ftw) , not later in time. Extracting into functions in this case just make it easier to test it.

#9 - 2018-05-09 12:46 - szarate

- Status changed from New to In Progress

> This is where i disagree the most. Because it seems that we should immediately rewrite the code. However i believe that before doing any changes we should start by understanding the issue. Which we don't. Doing changes before understanding what's wrong is usually not a good idea.

This is already too expensive timewise, to keep investing time trying to figure out and understanding the issue in detail, we already have the hint that it's hidden within that recursion call. Regardless of the algorithm used, regardless of the complexity of the issue, this method already has many code smells :) which is reason enough to at least do some code changes.

So I'm focusing on: Adding tests, extracting and decoupling for now. BFS, DSF, DAG use analysis is part of another take on this ticket.

And in the end... All things must be reborn

#10 - 2018-05-09 12:47 - szarate

- Precedes coordination #32851: [tools][EPIC] Scheduling redesign added

#11 - 2018-05-23 07:45 - szarate

So I managed to extract the lookup with tests passing only, but as soon as I try to remove the recursion by simply building the dependency tree in one shot, it all goes down south pretty fast.

#12 - 2018-05-25 12:17 - coolo

- Assignee changed from szarate to coolo

I'm almost there: https://github.com/os-autoinst/openQA/pull/1663

#13 - 2018-05-26 05:07 - coolo

- Status changed from In Progress to Resolved

I removed the recursive nature of duplicate, which now that I looked into it was very dangerous as the DB transactions happened per job within that recursion, but there was no global rollback - but dependencies were already added partly.

I also removed some - let's say - optimizations that reused scheduled jobs for the new cluster jobs, which lead to unclear dependencies of the chain (especially if partly jobs failed to duplicate).

I also revisted one decision that makes restarting multimachine jobs very hard in practise: done parents were excluded from restarting. I don't know what use case lead to that decision, but in our daily practise we rely on a multimachine test to have multiple machines running in parallel - so we currently have to be very careful always to restart the parent. I fixed this and now the full cluster is restarted. Test cases have been adapted, but many test cases in 05-scheduler-dependencies.t lost their full sense - as they tested old (IMO broken) behaviour.

#14 - 2018-06-15 11:43 - szarate

- Target version changed from Current Sprint to Done