CUNY Academic Commons - Bug #14276

Increase number of workers in Cavalcade

2021-04-02 05:34 PM - Raymond Hoh

Status:	Deferred	**Start date:**	2021-04-02
Priority name:	Normal	**Due date:**	
Assignee:	Raymond Hoh	**% Done:**	0%
Category name:	Cavalcade	**Estimated time:**	0.00 hour
Target version:	Not tracked		

Description

I wanted to check in on how Cavalcade was doing since the upgrade in #12240 and it looks like we are running into a backlog again.

Backlog was 1 hour behind, but I'm working to temporarily address this by removing some of the older items as mentioned in #12240. What this means is Cavalcade's workers cannot get through the job queue fast enough.

Boone, I think we might want to increase the number of workers used by Cavalcade. Currently, that number is set to 4 by default: https://github.com/humanmade/Cavalcade-Runner/blob/0dfb42d505e9cd870a11366c49ee680d327c961a/inc/class-runner.php#L36

Perhaps 6 or 8 workers might be preferable. If we wanted to change this, we would need to do this at the /bin/ script level. In the sample script provided by Cavalcade, that would be here: https://github.com/humanmade/Cavalcade-Runner/blob/0dfb42d505e9cd870a11366c49ee680d327c961a/bin/cavalcade#L30

The other way to address this is by patching the runner so it doesn't use the last updated nextrun timestamp with the interval as it does now when rescheduling jobs. See https://github.com/humanmade/Cavalcade-Runner/issues/51.

A workaround is to schedule a task to purge older waiting jobs that reschedule every hour or less than an hour on a daily or twice-daily basis.

Related issues:

Related to CUNY Academic Commons - Feature #12440: Upgrade Cavalcade	**Resolved**	**2020-02-18**
Related to CUNY Academic Commons - Bug #14299: Remove Site Health scheduled task	**Resolved**	**2021-04-09**

History

#1 - 2021-04-02 05:36 PM - Raymond Hoh

- Related to Feature #12440: Upgrade Cavalcade added

#2 - 2021-04-05 10:05 AM - Boone Gorges

Thanks for following up on this, Ray.

Increasing the number of workers seems reasonable to me, as long as Lihua doesn't have a problem with there being a bit more overhead in the form of PHP processes. To be clear, the change to the linked line will be:

```
$runner = Runner::instance( [ 'max_workers' => 8 ] );
```

Is that correct? In your GitHub comment, you describe this as a sort of "workaround", but IMO it's more accurate to say that an increased number of workers is a natural consequence of the very large number of sites and tasks on our installation. Even if we **didn't** have any problems with overlappingly-scheduled repeating events, we would still have a very large number of events that warrant more resources. Do you want to draft an email to Lihua about it?

It does seem like there's also a bug in Cavalcade here. The solution proposed in https://github.com/humanmade/Cavalcade-Runner/issues/51#issuecomment-489738426 seems correct to me. Calculating the new nextrun based on the current timestamp, rather than the old nextrun, feels like it would avoid a lot of problems. At the very least, the runner should be smart enough not to reschedule a recurring task if there are pending (and therefore duplicate) runs that are scheduled before the next time() + interval. Do you think it would be helpful if I chimed in on the GitHub ticket in favor of the time() solution? Your idea for a hook also seems fine, though it's a workaround for what is IMO a general bug in the system.

I don't have a problem with a regularly-scheduled cleanup task, if you think it'd help keep things tidy until there are broader fixes in place.

#3 - 2021-04-06 12:29 PM - Raymond Hoh

> Do you want to draft an email to Lihua about it?

I just checked Cavalcade and there doesn't appear to be a lag in the job queue at the moment.

It's been close to four days since I purged the trailing waiting jobs. Let's hold off on this until we can accurately determine what is causing the lag. I'm guessing certain jobs that occur on a weekly basis is the cause.

> $runner = Runner::instance(['max_workers' => 8]);

Yes, this looks correct to me.

> At the very least, the runner should be smart enough not to reschedule a recurring task if there are pending (and therefore duplicate) runs that are scheduled before the next time() + interval. Do you think it would be helpful if I chimed in on the GitHub ticket in favor of the time() solution?

On another Github issue, rmccue notes that the current "nextrun + interval" solution is intentional: https://github.com/humanmade/Cavalcade/issues/74#issuecomment-453150620. So feel free to chime in on the ticket as well.

#4 - 2021-04-07 04:45 PM - Boone Gorges

rmccue notes there that the behavior is intentional because "the clock monotonically increases". But our problem is not non-monotonic (ie incorrectly ordered) scheduling, it's that we are ending up with so many tasks that we don't have enough workers to complete them. The tasks that **are** scheduled are properly consecutive, but they are scheduled in the past (and often **multiple times** in the past) so that they never catch up.

Am I capturing the issue correctly? I want to be sure I understand it :)

Either way, maybe we can hold off as you continue to gather info.

#5 - 2021-04-09 02:14 AM - Raymond Hoh

> Am I capturing the issue correctly? I want to be sure I understand it :)

Yes you are. I just wanted to note what rmccue said as Cavalcade's rescheduling logic hasn't changed despite the many reports on Github.

> Either way, maybe we can hold off as you continue to gather info.

I just checked the Cavalcade job queue and the lag started again, but it was only lagging by about 15 minutes. Since this started again on a weekly basis, I want to look at removing weekly cron jobs that are not really useful.

The main offender is WP's Site Health cron job. There are currently 8943 scheduled tasks for this job. To me, the Site Health page isn't useful to subdomain administrators since we manage plugin and theme updates via Git, or even to us as network admins since we can query for this type of info anytime. I would like to propose that we remove the Site Health page and its corresponding cron job. What do you think, Boone?

Or if we want to be conservative, we can keep the Site Health page for the main site only. Then, we can remove the majority of tests that the Site Health check runs; I think we could safely remove 95% of them.

#6 - 2021-04-09 10:22 AM - Boone Gorges

> To me, the Site Health page isn't useful to subdomain administrators since we manage plugin and theme updates via Git, or even to us as network admins since we can query for this type of info anytime. I would like to propose that we remove the Site Health page and its corresponding cron job. What do you think, Boone?

I'm 100% in favor of removing it altogether. Let's add it to the block list at wp-content/mu-plugins/cavalcade.php and delete pending jobs.

I'll leave a comment on the GitHub thread in favor of fixing this upstream in Cavalcade.

#7 - 2021-04-09 06:51 PM - Raymond Hoh

- Related to Bug #14299: Remove Site Health scheduled task added

#8 - 2021-04-09 06:58 PM - Raymond Hoh

> Let's add it to the block list at wp-content/mu-plugins/cavalcade.php and delete pending jobs.

Done in #14299. I've also removed the pending wp_site_health_scheduled_check jobs and cleared Cavalcade's cache.

#9 - 2021-09-14 10:44 AM - Boone Gorges

- Target version set to Not tracked

Ray, do you think we need to take any more steps here?

#10 - 2021-09-14 07:22 PM - Raymond Hoh

- Status changed from New to Deferred

I just checked the Cavalcade job queue and there is currently no job lag, so we should be good.

I did see a new cronjob that is probably not necessary -- wp_https_detection (new as of WP v5.7.0). It runs twicedaily on every site to check if HTTPS is properly set up: https://developer.wordpress.org/reference/functions/wp_schedule_https_detection/. Since HTTPS is enabled across all sites on the Commons, this task can be removed. I've just done so here:
https://github.com/cuny-academic-commons/cac/commit/19e1d57f35fdb12e0a79bf9eb3bb9286cafcbf00

I'm going to change this ticket's status to Deferred because we still might want to do this sometime down the road.

#11 - 2021-09-26 03:58 PM - Raymond Hoh

After the scheduled network maintenance today, there was a backlog of Cavalcade jobs building up. At around 9:30 ET, the backlog was about 10 mins.

So I started removing some jobs that were lagging:

wp db query 'select hook,`interval`,count(*) from wp_cavalcade_jobs where nextrun < utc_timestamp() and status = "waiting" group by hook order by count(*) desc'

I also deployed https://github.com/cuny-academic-commons/cac/commit/19e1d57f35fdb12e0a79bf9eb3bb9286cafcbf00 as a hotfix and purged all wp_https_detection jobs.