Arvados - Bug #11669

[Crunch2] crunch-dispatch-slurm hits scancel too much

05/10/2017 09:44 PM - Tom Clegg

Status:	In Progress	**Start date:**	05/10/2017
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	Arvados Future Sprints		

Description

Moved from #11626

Example:

```
2017-05-08_14:09:51.64626 2017/05/08 14:09:51 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:09:52.35643 2017/05/08 14:09:52 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:09:53.35654 2017/05/08 14:09:53 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:09:54.35664 2017/05/08 14:09:54 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:09:55.35680 2017/05/08 14:09:55 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:09:56.35509 2017/05/08 14:09:56 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:09:57.35538 2017/05/08 14:09:57 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:09:58.35661 2017/05/08 14:09:58 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:09:59.35681 2017/05/08 14:09:59 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:00.35669 2017/05/08 14:10:00 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:10:01.35804 2017/05/08 14:10:01 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:02.35522 2017/05/08 14:10:02 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:10:03.35664 2017/05/08 14:10:03 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:04.35882 2017/05/08 14:10:04 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:10:05.35906 2017/05/08 14:10:05 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:06.36192 2017/05/08 14:10:06 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:10:07.36220 2017/05/08 14:10:07 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:08.36188 2017/05/08 14:10:08 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
2017-05-08_14:10:09.36211 2017/05/08 14:10:09 Dispatcher says container
qr1hi-dz642-tjezupwylh67k2b is done: cancel slurm job
2017-05-08_14:10:10.35595 2017/05/08 14:10:10 container qr1hi-dz642-tjezupwylh67k2b
 is still in squeue after scancel
```

Related issues:

Related to Arvados - Bug #11626: [Crunch2] Still not propagating slurm errors...	**Resolved**	**05/08/2017**

History

#1 - 05/10/2017 09:47 PM - Tom Clegg

11669-tidy-scancel-race @ f29a53dab584dcae3e4a38a7623efa6d3b30a663

#2 - 05/10/2017 09:49 PM - Tom Clegg

- Description updated

#3 - 05/11/2017 02:56 PM - Peter Amstutz

Notes:

We've set up a race condition for ourselves. During normal operation, the last thing that crunch-run does is to mark the container as "Complete". If crunch-dispatch-slurm closes the updates channel (e.g. when the container goes into "Complete" state) before HasUUID() starts returning false, it will always start calling scancel.

It's possible the slurm job is lingering in "completing" state. I haven't noticed that but I haven't looked closely. It is conceivable that Arvados is propagating the container state quicker than slurm is propagating job state changes.

Issuing scancel on a non-existing job seems to exit 0.

It seems like it would make sense to check HasUUID() before calling scancel (instead of doing it after).

A one-second delay between calls to scancel seems very short (compared to, for example, the default 10 second message timeout of slurm). Looking at the logs, the scancel loop seems to resolve with 20 seconds.

We seem to have a pretty aggressive crunch-dispatch-slurm polling period configured (500ms).

#4 - 05/11/2017 05:57 PM - Tom Clegg

Sequence is

1. crunch-run updates container state to Complete or Cancelled
2. crunch-dispatch-slurm waits some amount of time ("delay A") for the slurm job to disappear from the queue by itself
3. crunch-dispatch-slurm decides it's waited long enough, and starts hitting scancel
4. crunch-dispatch-slurm waits some amount of time ("delay B") before retrying if scancel does not take effect immediately

In the current implementation, "delay A" is zero and "delay B" is 1 second.

There is an unavoidable race condition: no matter what we do, it will always be possible for the slurm job to disappear from the queue in between "check whether the job is still in squeue" and "hit scancel". Fortunately, this is a harmless race: nothing bad happens when we hit scancel on a job that has already completed.

Any given value for "delay A" or "delay B" is a compromise between responsiveness and resource usage. My understanding is:

- no matter how long delay A is, we will *sometimes* hit scancel when waiting a bit longer would have worked.
- nothing bad happens if we hit scancel when waiting a bit longer would have worked.
- when delay B is short, we run scancel lots of times.
- when delay B is short, we often log several "job is still in queue" messages for each container.

Not sure which of these is [anticipated to be] a problem.

> It seems like it would make sense to check HasUUID() before calling scancel (instead of doing it after).

AFAICT this would merely introduce a delay before the first call to scancel, without eliminating any races. The reason it's written the way it is is that in the "apiserver tells dispatch to cancel the job" case, it is better to hit scancel immediately, while in the container-finishing scenario, races are unavoidable anyway.

#5 - 05/12/2017 03:13 PM - Tom Clegg

We could reduce the number of scancel invocations by ~1 per container by calling scancel() immediately the very first time, and after that, checking HasUUID() after the 1s delay instead of before the 1s delay.

We could rate-limit scancel invocations using a global time.Ticker(time.Second/time.Duration(maxScancelPerSecond)) so we don't flood slurm, the process table, or our file descriptor allotment in situations where we're waiting for 1000 containers' slurm jobs to exit.

We could limit the total number of outstanding scancel processes at a given time by making a buffered channel "pool" and wrapping the invocation in "<-pool" and "pool<-struct{}{}" -- this would be more certain way to avoid flooding the process table and fd allotment.

We could make the delay between consecutive scancel invocations configurable. It might be better to have a longer delay after a non-zero scancel exit code, since that means either something is broken (and will probably still be broken 1 second later) or [slurm changes its exit code rules and] it's an error to scancel a job that is already gone (in which case we've done our work and there's no particular hurry to exit our goroutine). Aside: If there is no job named X then "scancel --job-name=X" amounts to "cancel an empty set of jobs" so exit 0 seems sensible.

We could make the delay between consecutive "scancel had no effect" log messages configurable. (Perhaps it's OK to make lots of scancel attempts but annoying to fill the logs with noise.)

#6 - 08/28/2017 10:14 PM - Tom Morris

- Target version set to Arvados Future Sprints