Arvados - Feature #8028

[Crunch2] Dispatch containers locally

12/16/2015 07:29 PM - Peter Amstutz

Status:	Resolved	**Start date:**	01/19/2016
Priority:	Normal	**Due date:**	
Assigned To:	Radhika Chippada	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2016-01-20 Sprint		

Description

This is a stub to support development and testing of other Crunch2 components. It isn't meant to be production-ready for this story, although in future work it may become part of a production setup and serve as a template for writing new dispatchers.

It's the "standalone worker" example described at Container dispatch, without the "determine local capacity" part: it just takes all containers that appear in the queue, and starts them immediately as child processes.

The implementation should emphasize simplicity: we don't want to load this up with features. Anything non-trivial probably needs to be done in a different layer anyway (e.g., API server or crunch-run).

Implementation

"crunch-dispatch-local", a Go program in source:services/crunch-dispatch-local/

- The main goroutine polls the queue every 10 (configurable with "-poll-interval 5s") seconds using a time.NewTicker(). When a new container appears, pass it to a "run" goroutine.
- The "run" goroutine starts "crunch-run" (configurable with "-crunch-run-command /usr/bin/crunch-run") and waits until [a] the crunch-run command exits, or [b] the container's priority changes to zero. If the latter happens, send TERM to the crunch-run command, and keep waiting until it exits.
- Emit the relevant logs mentioned at Container dispatch.

It's not very efficient for each "run" goroutine to poll the API server about its own container record, but it's not necessary to optimize this right now. (Ideally, when we are ready to optimize this, we'll go straight to websocket updates rather than batching our polling for all containers that are running now, etc.)

Subtasks:

Task # 8140: Review branches 8028-crunch-dispatch-local in arvados and arvados-dev repo...	**Resolved**

Related issues:

Related to Arvados - Feature #8128: [Crunch2] API support for crunch-dispatch	**Resolved**	**04/28/2016**

Associated revisions

Revision c62ae0ad - 01/20/2016 05:37 PM - Radhika Chippada

closes #8028
Merge branch '8028-crunch-dispatch-local'

Revision 7f05c28c - 01/20/2016 05:44 PM - Radhika Chippada

refs #8028
Merge branch '8028-crunch-dispatch-local'

Revision 7f05c28c - 01/20/2016 05:44 PM - Radhika Chippada

refs #8028
Merge branch '8028-crunch-dispatch-local'

History

#1 - 12/16/2015 07:35 PM - Peter Amstutz

- Description updated

#2 - 01/06/2016 08:33 PM - Tom Clegg

- Description updated

#3 - 01/06/2016 08:41 PM - Peter Amstutz

- Target version set to 2016-01-20 Sprint

#4 - 01/06/2016 08:47 PM - Radhika Chippada

- Assigned To set to Radhika Chippada

#5 - 01/07/2016 06:58 PM - Tom Clegg

- Description updated

- Category set to Crunch

#6 - 01/07/2016 09:41 PM - Radhika Chippada

- Category deleted (Crunch)

- Story points set to 2.0

#7 - 01/07/2016 09:42 PM - Radhika Chippada

- Category set to Crunch

Hmm, I wonder why the category was reset when I just set the story points!

#8 - 01/07/2016 09:47 PM - Radhika Chippada

- Status changed from New to In Progress

#9 - 01/13/2016 04:30 PM - Radhika Chippada

d3781de388530d06379974601247a6c044eee92e in branch 8028-crunch-dispatch-local

Implementation notes:

- main func with command line args for polling and crunch-run command
- Using ticker, poll for queued containers and invoke run goroutine for each
- (Upon discussing with Tom), any errors are logged but the program continues to run, without exiting. This is because, once we start one or more run procs, we need to wait for them complete. The program does end if there are any errors in args processing etc.
- (Upon discussing with Tom), after the crunch-run command completes, update the container state to "Complete" if it is still in "Running" state
- Log the events listed as applicable

Testing notes:

- Added containers fixtures, one in queued state and one in completed state (this one is not used in testing)
- Test the main method with args parsing
- Using the arvadostest.ServerStub implementation, verify the run goroutine functionality by verifying the contents of the log file

#10 - 01/19/2016 03:37 PM - Peter Amstutz

The "items available" is the total number of items that match the filter in the database. Because of paging, the number of items actually returned can be smaller. So you need to use len(containers.Items) here.

```
    for i := 0; i < containers.ItemsAvailable; i++ {
```

The dispatcher should probably record which containers UUIDs it has started and make sure they only are started once.

```
cmd := exec.Command(crunchRunCommand, "--job", uuid)
```

Crunch-run takes the first command line argument as the container UUID, so you can remove the "--job" from the runner command.

Crunch-run is responsible for moving the container from the "Queued" to "Running" state, so you can remove that part.

Please use cmd.Process.Signal(os.Interrupt) to terminate the process, so that crunch-run can shut down gracefully.

#11 - 01/19/2016 05:23 PM - Radhika Chippada

> Instead of containers.ItemsAvailable, use len(containers.Items)

Done

> The dispatcher should probably record which containers UUIDs it has started and make sure they only are started once.

Already doing this. However, noticed from testing log that I need to increase the poll-time in test slightly to ensure second run does not happen while the first one is still dispatching.

> you can remove the "--job" from the runner command.

Done

> Crunch-run is responsible for moving the container from the "Queued" to "Running" state, so you can remove that part

For the time being, I am leaving this alone. We will remove / update this when locking api is available and we have better state management in place.

> Please use cmd.Process.Signal(os.Interrupt) to terminate the process, so that crunch-run can shut down gracefully.

Done

#12 - 01/19/2016 08:35 PM - Peter Amstutz

Please add a graceful shutdown:

If the dispatcher receives SIGINT or SIGTERM:

1. stop the runQueuedContainers loop with the doneProcessing channel
2. send Interrupt to any running processes
3. wait for all the processes to finish.
4. only then, terminate

Update Test_doMain() and testWithServerStub() to use the "wait for processes to finish" capability instead of time.Sleep().

#13 - 01/20/2016 02:46 AM - Radhika Chippada

Added signal handling to dispatcher.

#14 - 01/20/2016 02:18 PM - Peter Amstutz

Thanks for turning this around.

I suggest refactoring the implementation a litle bit:

1. The signal handler just sends "true" on the doneProcessing channel
2. Make wg sync.WaitGroup a global variable
3. The run() function is responsible for calling wg.Add() and defer wg.Done()
4. the for{} loop in runQueuedContainers should break, not return when doneProcessing is ready
5. after the for{} loop in runQueuedContainers breaks, that's when it should send the interrupt to each entry in runningCmds
6. then runQueuedContainers should call wg.Wait()

Also, there should be a sync.Mutex protecting runningCmds map everywhere it is accessed.

#15 - 01/20/2016 05:40 PM - Radhika Chippada

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:c62ae0ad70080b1217dc478b4921441013d21db4.

#16 - 01/20/2016 05:55 PM - Nico César

updated ci.curoverse.com, diff was 1e48d4f..7f05c28