Image download progress

Introduction

When containers are scheduled on a Mesos cluster there is currently no information about the progress of image downloads. The container task will have the TaskState STAGING while the image is downloaded and the end-user does not have accurate information about when downloading will be finished. Consequently, it is hard for the end-user to decide to restart the task or wait if a task remains in STAGING for a long time.

This design document describes a design which adds image download progress to Mesos.

User stories

1. As a Mesos framework author, I want Mesos to provide a way to see periodic image download progress in terms of percentage downloaded and download speed

Terminology

Short	**Description**	**Example**
Image	An image that gets downloaded before a container task runs	nginx:latest
Image Provisioner	Mesos component which sets up a root filesystem for a given image	src/slave/containerizer/mesos/provisioner/provisioner.cpp
Image Provider	Mesos component which downloads and unpacks images	src/slave/containerizer/mesos/provisioner/docker/registry_puller.cpp

Design exploration

- *Option 1:* The Mesos fetcher logs every percent download progresss and sends status updates for each progress, setting two extra fields, `total` and `remaining`
    - Pros:
        - We can implement a libcurl progress function and hook this in `fetcher.cpp::downloadWithNet`
        - We can call `Slave::statusUpdate` from the progress function and set the three fields on the `TaskStatus` object
- *Option 2:* Create a Mesos hook which monitors download progress and sets the three `'total', 'percentageDownloaded' and 'megaBytesDownloaded'`.
    - Pros:
        - Potentially less invasive
    - Cons:
        - Hook modules have to deployed on the cluster separately which is extra work for operators.

It is preferred not to use modules or hooks but to add this to the Mesos core.

Architecture

slave.cpp

- Add an `/images` endpoint which accepts an image name like `library/nginx`
- If the image is being downloaded it returns a 202 and a JSON representation of the `total` and `remaining` number of Megabytes to be downloaded. If it does not exists it returns a 404. If it exists it returns a 200.
- The `/images` endpoint finds the `FetcherProcess` for the image and calls progress which returns the JSON representation.

containerizer.cpp

- Whenever `fetcher->fetch` is called, add an entry to a map with the image name and the `FetcherProcess`

fetcher.cpp

- Add a libcurl progress function which updates the `total` and `remaining` fields on the `FetcherProcess`
- Add a method progress which returns a JSON representation of total and remaining that can be called from the slave images endpoint
- When downloading is completed or fails remove the FetcherProcess from the map

TODO: Status updates

Open Questions

- How to capture the download speed?
- What kind of new TaskStatus::REASON_ types do we have to create to communicate download failures?
- Look into performance impact of status updates. See https://issues.apache.org/jira/browse/MESOS-6941