Dynamic weights

Authors	Yongqiao Wang<yqwyq@cn.ibm.com>
Revision	0.1
Status	Reviewing

Table of Contents

Epic: MESOS-4189

Motivation

Mesos currently uses a static list of weights, specified when the master is started (via the `--weights` flag) and it requires that all the masters be rebooted to change the weights as resource allocation priority changes (e.g., a new high-priority framework installed with a new role, cluster operator needs to set a bigger value for the weight of this new role).

This document discusses Mesos support for dynamic weights at runtime. It proposes to add a new endpoint **/weights** to update/show weights of roles with the authorized principals, and the non-default weights will be persisted in registry.

High Level Proposals

In the short-term, a static list of weights (via the `--weights` flag) can still be specified when the master is started to initialize the weights configuration, and operator can change the weight of a role or configure weight for more roles (using default value 1.0) via the HTTP endpoint /weights

at runtime with the authorized principals, and those non-default (non 1.0) weights will be persisted in registry log.

In the long-term, the `--weights` flag will be removed, and weights can only be configured by /weights endpoint at runtime.

Weights management

Handling a /weights request at the Master endpoint can be logically decomposed into the following stages:

- Validate the request (JSON message format).
- Authenticate the request.
- Authorize the request.
- Check if the operation can be granted
    - Weight should > 0;
    - In explicit roles, role should exist in role whitelist or weight hashmap.
- Construct `protobuf` message based on the request.
- Persist weight changes in the registry.
- Update the Master's weights hashmap.
- Notify the allocator about weight changes.

We will introduce a new class to handle the /weights request:

```
/**
 * Inner class used to namespace the handling of /weights requests.
 *
 * It operates inside the Master actor. It is responsible for
 * validating, persisting and handle /weights requests, and
 * exposing weights status.
 * @see master/weight_handler.cpp for implementations.
 */
class WeightHandler
{
  public:
    explicit WeightHandler(Master* _master) : master(_master)
    {
      CHECK_NOTNULL(master);
    }
    process::Future<process::http::Response> get(
        const process::http::Request& request) const{}

    process::Future<process::http::Response> update(
        const process::http::Request& request) const{}

  private:
```

```
    Master* master;
};
```

Persisting non-default weights

To consider the Mesos master recovery and failover case, the non-default weights need to be persisted in registry:

- In the first boot, the first leading master initializes the replicated log with the weights specified by command-line flags(`--weights`). The flags values are only useful to bootstrap the cluster, after which point the registry becomes the source of truth.
- At runtime, the weights replicated log can only be updated by the /weights endpoint.
- For Mesos master restart/failover case, if the weights replicated log exists, then we will use the registry values and ignore the `--weights` flags, and also log a warning in Mesos master that the flags values are being ignored.
- For the future works, `--weights` flag will be removed, then after the Mesos master quorum is achieved, operator can send an update weights request to do a batch configuration for weights.

To support this, we shall:

- Introduce a Weight message in registry.proto;
- Introduce UpdateWeights operations;
- Recover weights from the registry on failover to the Master's weights hashmap;
- Extend RegistrarTest with weight-specific tests.

WeightInfo protobuf

For weight information display and persistent, we will define a WeightInfo protobuf in `mesos.proto`:

```
/**
 * Describes a weight, which is used to display and
 * persist the weight information.
 */
message WeightInfo {

  // Weight used to indicate resource allocation priority.
  required double weight = 1;

  // Related role name.
  required string role = 2;

}
```

Extending the Allocator interface

In the default DRF allocator, weight determines the resource allocation priority, so the Master needs to notify an allocator about weight update.

We propose the following approach for updating the `Allocator` interface:

```
class Allocator
{
public:
  ...
  ...

 /**
  * Updates weights.
  *
  * Updates the weight of the specified roles, and the
  * allocator is invoked to allocate the resources based
  * on the updated weights later.
  */
  virtual void updateWeights(
      const hashmap<string, double>& weights) = 0;

  ...
  ...
};
```

HTTP Endpoints

We propose to have a single REST-like endpoint with multiple http verbs to distinguish between different actions. For now, we propose to name the endpoint `/weights`;

HTTP status codes

Status	Reason
`200 OK`	/weights request granted.
`204 No Content`	GET Request to non-existing endpoint.
`400 Bad Request`	JSON message incorrect.
`401 Unauthorized`	Failed to authenticate/authorize

`405 Method not allowed`	Only GET and PUT method are allow in /weights endpoint.
`503 Temporarily Not Available`	Request cannot be processed at the moment

Show weights

NOTE: Because currently weights information have already be shown with /roles endpoint, so we will leave it out of the MVP.

To query all configured weights, operator can send an HTTP **GET** request to the `/weights` HTTP endpoint like this:

```
curl -X GET http://<master-ip>:<port>/weights
{
    "weights":[
        {
            "weight":"2.0",
            "role":"role1"
        },
        {
            "weight":"5.0",
            "role":"role2"
        }
    ]
}
```

NOTE: This request will only return all the non-default weights.

To get the weight for a particular role, operator can send an HTTP GET request to the `/weight/<role>` HTTP endpoint like this:

```
curl -X GET http://<master-ip>:<port>/weight/role1
```

NOTE: If the weight of the specified role never been set (via `--weights` flag or /weight endpoint) before:

- When using implicit roles, then the default value(1.0) will be returned.
- When using explicit roles(--roles flag has been specified when master is started), if this specified role does not exist in the role whitelist, then we will return a `204 No Content` code, otherwise return the default value(1.0).

Update weights

To update the weight of some roles, operator can send an HTTP **PUT** request to the `/weight` HTTP endpoint like this:

```
$ curl -d weights="[
    {
        "weight": 2,
        "role": "role1"
    },
    {
        "weight": 6,
        "role": "role2"
    }
]" -X PUT http://<master-ip>:<port>/weight
```

NOTE: This request support to update weight of multiple roles at one time, it is more useful when `--weights` removed, and operator can do batch configuration for weights.

NOTE: If the weight of a role is updated to the default value of 1.0, then this role's weight will disappear from the result list of /weights GET request.

NOTE: If the weight of the specified role has never been set (via `--weights` flag or /weights endpoint) before:

- When using implicit roles, this request will always be successful.
- When using explicit roles(--roles flag has been specified when master is started), if any one of those specified roles does not exist in the role whitelist, then we will return a `204 No Content` code, and others also will not be updated.

Future Works

Deprecating the `--weights` command-line flag: for backward compatibility, `--weights` flag will be still supported in the initial phase (goal: Mesos 0.27 release), and after one deprecation cycle (~six months, Mesos ~0.33 release), an error will be raised if the `--weights` command-line flag is specified.