**1. Module Description**

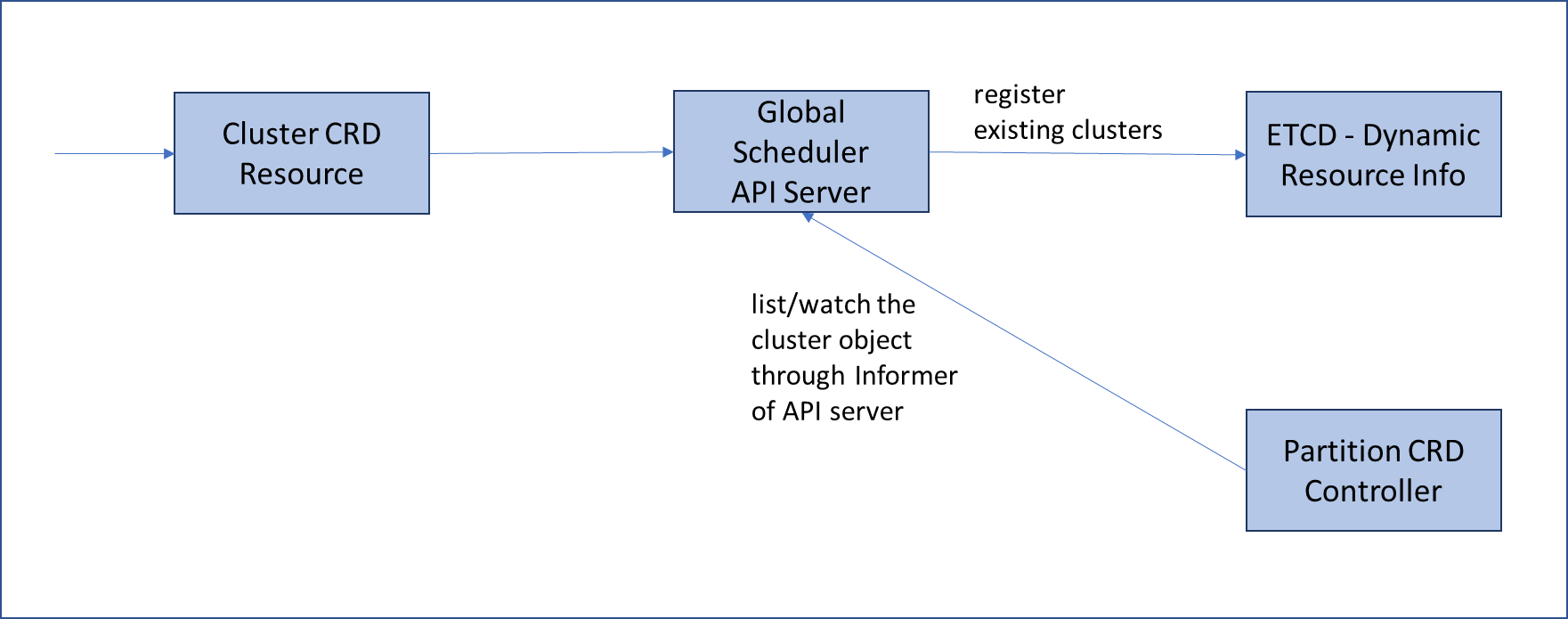
This module allows global resource scheduler register, deregister (or unregister), get and list existing clusters.

* Cluster is a set of nodes.
* Nodes are VM or host machines for running containerized applications in Pods.
* Namespace is a virtual cluster inside a Kubernetes cluster. There can be multiple namespaces inside a single Kubernetes cluster, and they are all logically isolated from each other.
* Relationship among objects:

Cluster > Namespace (multiple name space in a cluster) > Node (Host or VM) > Pod > Container

**2. Requirements**

* In milestone1, it supports Openstack cluster, Kubernetes cluster, and Arktos cluster. They share one single common data structure. It supports both CLI and REST APIs.
* Global resource controller does not create a cluster. It just register existing clusters which are created by other Kubernetes, Arktos, or Openstack API servers.



[Picture1] Flow of registration of a cluster

2.1 [API Implementation](https://github.com/futurewei-cloud/global-resource-scheduler/issues/23)

(1) CLI: command line APIs like kubectl

* Register cluster, Unregister cluster, List cluster, Get cluster

(2) REST APIs: REST WEB APIs

* Register cluster, Unregister cluster, List cluster, Get cluster

2.2 [Create the CRD Controller code framework via code-generator](https://github.com/futurewei-cloud/global-resource-scheduler/issues/25)

* Global resource scheduler defines a cluster CRD definition.

2.3 [Implement the controller logic](https://github.com/futurewei-cloud/global-resource-scheduler/issues/27)

* List/watch the cluster object and scheduler object through Informer.
* Run consistent hashing algorithm to set the scheduler-cluster binding/association.
* Save the binding to ETCD

1. **Data Structure**

type **Cluster** struct {

apiversion:

kind:

Name:

*// Spec is the custom resource spec*

    Spec ClusterSpec

}

*// MyResourceSpec is the spec for a MyResource resource*

type **ClusterSpec** struct {

*// this is where you would put your custom resource data*

ipAdrress string

GeoLocation   GeolocationInfo

Region RegionInfo

Operator OperatorInfo

    flavors []FlavorInfo

storage []StorageSpec

EipCapacity int64

CPUCapacity int64

MemCapacity int64

ServerPrice int64

HomeScheduler string

}

type **FlavorInfo** struct {

FlavorID string

TotalCapacity int64

}

type **StorageSpec** struct {

TypeID string //(sata, sas, ssd)

StorageCapacity int64

}

type **GeolocationInfo** struct {

city string

province string

area string

country string

}

type **RegionInfo** {

region string

AvailabilityZone string

}

type **OperatorInfo** {

operator string

}

1. **APIs Design**

4.1 CLI APIs

* register cluster -filename FILENAME
* unregister cluster -name CLUSTERNAME
* unregister cluster -id CLUSTERID
* list clusters
* get cluster -name CLUSTERNAME
* get cluster -id CLUSTERID

4.2 Case Study of CLI commands

* Kubrnetes

Format: kubectl create -f FILENAME

Example: kubectl create -f ./cluster1.yaml

* Openstack

Format: openstack cluster create --profile {FILENAME} {CLUSTERNAME}

Example: openstack cluster create --profile qstack c3

* qstack: profile file name
* c3: cluster name

**3.3 REST APIs & Error Codes Design**

|  |  |  |  |
| --- | --- | --- | --- |
| **Group** | **API Name** | **Method** | **Request** |
| Cluster | registerCluster | POST | /globalscheduler/v1/clusters |
|  | unregisterClusterById | DELETE | /globalscheduler/v1/clusters/id/{clusterid} |
|  | unregisterClusterByName | DELETE | /globalscheduler/v1/clusters/name/{clusterid} |
|  | listCluster | GET | /globalscheduler/v1/clusters |
|  | getClusterById | GET | /globalscheduler/v1/clusters/id/{clusterid} |
|  | getClusterByName | GET | /globalscheduler/v1/clusters/name/{clusterid} |

(1) Register Cluster

* Method: POST
* Request: /globalscheduler/v1/clusters
* Request Parameter:
* Response: cluster profile
* Normal response codes: 201
* Error response codes: 400, 409, 412, 500, 503
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clusters

Body:

{

"cluster\_profile": {

"cluster\_name": "cluster1",

“cluster\_spec”: {

“ipAdrress”: “10.0.0.3”,

“GeoLocation”:   {

“city”: “Bellevue”,

“province”: “Washington”,

“area”: “West”,

“country”: “US”

},

“Region”: {

“region”: “us-west”,

“AvailabilityZone”: “us-west-1”

},

“Operator”: {

“operator”: “globalscheduler”,

},

    “flavors”: [ {“FlavorID”: “small”, “TotalCapacity”: 5},

{ “FlavorID”: “medium”, “TotalCapacity”: 10},

{ “FlavorID”: “large”, “TotalCapacity”: 20},

{ “FlavorID”: “xlarge”,“TotalCapacity”: 10},

{ “FlavorID”: “2xlarge”, “TotalCapacity”: 5

}],

“storage”: [ {“TypeID”: “sata”, “StorageCapacity”: 2000},

{ “TypeID”: “sas”, “StorageCapacity”: 1000},

{ “TypeID”: “ssd”, “StorageCapacity”: 3000},

}],

“EipCapacity”: 3,

“CPUCapacity”: 8,

“MemCapacity”: 256,

“ServerPrice”: 10,

“HomeScheduler”: “scheduler1”

}

}

}

Response:

{

"cluster\_profile": {

"cluster\_id": "3dda2801-d675-4688-a63f-dcda8d327f51",

"cluster\_name": "cluster1",

“cluster\_spec”: {

“ipAdrress”: “10.0.0.3”,

“GeoLocation”:   {

“city”: “Bellevue”,

“province”: “Washington”,

“area”: “West”,

“country”: “US”

},

“Region”: {

“region”: “us-west”,

“AvailabilityZone”: “us-west-1”

},

“Operator”: {

“operator”: “globalscheduler”,

},

    “flavors”: [ {“FlavorID”: “small”, “TotalCapacity”: 5},

{ “FlavorID”: “medium”, “TotalCapacity”: 10},

{ “FlavorID”: “large”, “TotalCapacity”: 20},

{ “FlavorID”: “xlarge”,“TotalCapacity”: 10},

{ “FlavorID”: “2xlarge”, “TotalCapacity”: 5

}],

“storage”: [ {“TypeID”: “sata”, “StorageCapacity”: 2000},

{ “TypeID”: “sas”, “StorageCapacity”: 1000},

{ “TypeID”: “ssd”, “StorageCapacity”: 3000},

}],

“EipCapacity”: 3,

“CPUCapacity”: 8,

“MemCapacity”: 256,

“ServerPrice”: 10,

“HomeScheduler”: “scheduler1”

}

}

}

(2) Unregister Cluster By Id

* Method: DELETE
* Request: /globalscheduler/v1/clusters/id/{id}
* Request Parameter: String cluster\_id
* Response: cluster\_id
* Normal response codes: 200
* Error response codes: 400, 412, 500
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clusters/3dda2801-d675-4688-a63f-dcda8d327f51

Response:

deleted: 3dda2801-d675-4688-a63f-dcda8d327f50

(2) Unregister Cluster By Name

* Method: DELETE
* Request: /globalscheduler/v1/clusters/name/{name}
* Request Parameter: String cluster\_name
* Response: cluster\_name
* Normal response codes: 200
* Error response codes: 400, 412, 500
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clutsers/name/cluster1

Response:

Deleted: cluster1

(6) List Clusters

* Method: GET
* Request: v1/clusters
* Request Parameter:
* Response: clusters’ list
* Normal response codes: 200
* Error response codes: 400, 409, 412, 500, 503
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clusters

Response:

{

[

{

"cluster\_id": "3dda2801-d675-4688-a63f-dcda8d327f51",

"cluster\_name": "cluster1",

“cluster\_spec”: {…}

},

{

"cluster\_id": "3dda2801-d675-4688-a63f-dcda8d327f52",

"cluster\_name": "cluster2",

“cluster\_spec”: {…}

}

…

]

}

(7) Get ClusterById

* Method: GET
* Request: v1/clusters/id/{id}
* Request Parameter: string cluster\_id
* Response: cluster profile
* Normal response codes: 200
* Error response codes: 400, 412, 500
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clusters/3dda2801-d675-4688-a63f-dcda8d327f50

Response:

{

"cluster\_profile": {

"cluster\_id": "3dda2801-d675-4688-a63f-dcda8d327f51",

"cluster\_name": "cluster1",

“cluster\_spec”: {

“ipAdrress”: “10.0.0.3”,

“GeoLocation”:   {

“city”: “Bellevue”,

“province”: “Washington”,

“area”: “West”,

“country”: “US”

},

“Region”: {

“region”: “us-west”,

“AvailabilityZone”: “us-west-1”

},

“Operator”: {

“operator”: “globalscheduler”,

},

    “flavors”: [ {“FlavorID”: “small”, “TotalCapacity”: 5},

{ “FlavorID”: “medium”, “TotalCapacity”: 10},

{ “FlavorID”: “large”, “TotalCapacity”: 20},

{ “FlavorID”: “xlarge”,“TotalCapacity”: 10},

{ “FlavorID”: “2xlarge”, “TotalCapacity”: 5

}],

“storage”: [ {“TypeID”: “sata”, “StorageCapacity”: 2000},

{ “TypeID”: “sas”, “StorageCapacity”: 1000},

{ “TypeID”: “ssd”, “StorageCapacity”: 3000},

}],

“EipCapacity”: 3,

“CPUCapacity”: 8,

“MemCapacity”: 256,

“ServerPrice”: 10,

“HomeScheduler”: “scheduler1”

}

}

}

(1) Get Cluster By Name

* Method: GET
* Request: /clusters/name/{name}
* Request Parameter:
* @PathVariable String cluster\_name,
* Response: cluster profile
* Normal response codes: 200
* Error response codes: 400, 412, 500
* Example

Request:

http://127.0.0.1:8080/globalscheduler/v1/clusters/cluster1

Response:

{

"cluster\_profile": {

"cluster\_id": "3dda2801-d675-4688-a63f-dcda8d327f51",

"cluster\_name": "cluster1",

“cluster\_spec”: {

“ipAdrress”: “10.0.0.3”,

“GeoLocation”:   {

“city”: “Bellevue”,

“province”: “Washington”,

“area”: “West”,

“country”: “US”

},

“Region”: {

“region”: “us-west”,

“AvailabilityZone”: “us-west-1”

},

“Operator”: {

“operator”: “globalscheduler”,

},

    “flavors”: [ {“FlavorID”: “small”, “TotalCapacity”: 5},

{ “FlavorID”: “medium”, “TotalCapacity”: 10},

{ “FlavorID”: “large”, “TotalCapacity”: 20},

{ “FlavorID”: “xlarge”,“TotalCapacity”: 10},

{ “FlavorID”: “2xlarge”, “TotalCapacity”: 5

}],

“storage”: [ {“TypeID”: “sata”, “StorageCapacity”: 2000},

{ “TypeID”: “sas”, “StorageCapacity”: 1000},

{ “TypeID”: “ssd”, “StorageCapacity”: 3000},

}],

“EipCapacity”: 3,

“CPUCapacity”: 8,

“MemCapacity”: 256,

“ServerPrice”: 10,

“HomeScheduler”: “scheduler1”

}

}

}

1. Case Study
   1. Openstack Cluster

https://docs.openstack.org/senlin/latest/user/clusters.html#creating-a-cluster

$ openstack cluster create --profile qstack c3

+------------------+--------------------------------------+

| Property | Value |

+------------------+--------------------------------------+

| config | {} |

| created\_at | None |

| data | {} |

| dependents | {} |

| desired\_capacity | 0 |

| domain\_id | None |

| id | 60424eb3-6adf-4fc3-b9a1-4a035bf171ac |

| init\_at | 2015-05-05T13:35:47Z |

| location | None |

| max\_size | -1 |

| metadata | {} |

| min\_size | 0 |

| name | c3 |

| node\_ids | |

| profile\_id | bf38dc9f-d204-46c9-b515-79caf1e45c4d |

| profile\_name | qstack |

| project\_id | 333acb15a43242f4a609a27cb097a8f2 |

| status | INIT |

| status\_reason | Initializing |

| timeout | 3600 |

| updated\_at | None |

| user\_id | 0b82043b57014cd58add97a2ef79dac3 |

+------------------+--------------------------------------+

openstack --os-identity-api-version 3 dataprocessing cluster create [-h]

[-f {html,json,shell,table,value,yaml}]

[-c COLUMN]

[--max-width <integer>]

[--noindent] [--prefix PREFIX]

[--name <name>]

[--cluster-template <cluster-template>]

[--image <image>]

[--description <description>]

[--user-keypair <keypair>]

[--neutron-network <network>]

[--count <count>] [--public]

[--protected] [--transient]

[--json <filename>] [--wait]