Service Grid redesign. Phase 1. Implementation details.

- Overview
- Deployment process
- Topology/coordinator change
- Cluster activation/deactivation
- Deployment errors propagation

Overview

All nodes (servers and clients) are able to host services, but the client nodes are excluded from service deployment by default. The only way to deploy service on client nodes is to specify node filter in ServiceConfiguration.

All deployed services are identified internally by “serviceld” (IgniteUuid). This allows us to build a base for such features as hot redeployment and service versioning. It’s important to have an ability to identify and manage services with the same name, but a different version.

Deployment process

Service deployment is managed via DiscoverySpi and CommunicationSpi messages.

Users requests (deploy/undeploy) are represented as discovery custom message ServiceChangeBatchRequest which contains collection of actions intended changing services states. The action has to extend ServiceChangeAbstractRequest.

The requests are sent via DiscoverySpi, it allows us to get following guarantees:

- The request will be received by all the nodes in the topology. This allows continuing processing request if the coordinator failed and to obtain all needed metadata by one message to handle the request (service configuration, id, class etc.);
- A strict order of requests, to be able to validate request in case of duplication or conflicts in services configuration;
- The relation (order) between user requests and exchange of node joining process (which is also managed via DiscoverySpi). This allows sending a complete set of existing services metadata (also waiting for deployment) to joining node and not to lose any metadata (IgniteServiceProcessor#registeredServices);

Once the request is received, it is stored in a deployment queue as ServiceDeploymentTask to be processed in a separate thread because of requiring significant time. That means each request will be processed in order of queue.

The deployment queue is managed by ServiceDeploymentManager. Deployment worker (special thread) takes from the queue a deployment task and calls ServiceDeploymentTask#init to start the deployment process:

- The task performs actions of changing services states. In case of deploy request, each node calculates assignments independently using a deterministic function (IgniteServiceProcessor#reassign);
- Deployment results are represented as communication message ServiceSingleNodeDeploymentResultBatch, which is sent to coordinator via CommunicationSpi (p2p) once actions were performed. The message contains deployments errors and count of locally deployed instances of services related to the current deployment process;
- The coordinator aggregates the deployment results from the cluster. The result of the whole deployment process is represented as discovery custom message ServiceClusterDeploymentResultBatch which is built and sent to all nodes via DiscoverySpi once all single nodes results were received.
- Each node handles ServiceClusterDeploymentResultBatch, updates deployments information and complete initiators futures if needed, then finishes the deployment process;

The following events cause deployment process:

- Users deploy/undeploy requests;
- Affinity topologies change if affinity services exist;

Topology change events (EVT_NODE_JOINED/LEFT/FAILED);

Topology/coordinator change

Each topology change event (NODE_JOIN/LEFT/FAILED) causes deployment task. Assignments will be recalculated and applied for each deployed service if needed.

Services reassignment process takes into account previous assignments to avoid redundant redeployment.

If left/failed node had not deployed any services, then a deployment task will be finished without sending messages.

If a coordinator was changed during the service deployment process then all nodes send of ServiceSingleNodeDeploymentResultBatch to a new coordinator which continues the process as usual.

Cluster activation/deactivation

- On deactivation:
- local services are being undeployed;
- requests are not handling (including deployment / undeployment);

**On activation:**
- local services are being redeployed;
- requests are handling as usual;

### Deployment errors propagation

All errors occurred during service deployment exchange are propagated across the cluster and are available on any node. Current implementation covers the following errors' causes:

- Errors during assigned nodes definition, e.g. when failed to determine suitable nodes for deploy;
- Deployment errors, e.g. when failed to load service class;
- *Service*\#*init* errors, e.g. any users failures;