Cluster mechanism extension

Cluster mechanism extension refers to the extension of the iServer cluster functions, including generic spatial service clusters and domain spatial service clusters, which can better satisfy users' needs.

- Extending load balancer

- Extending cluster filter

In addition to the cluster mechanism, SuperMap iServer can build distributed domain spatial service clusters as well as ordinary domain spatial service clusters.

Distributed DSS clusters cluster similar DSSs that provide different services. For example, in the weather query DSS, different weather query DSSs provide weather reports for different cities, one can cluster them together to provide weather reports for all cities.

- Introduction to filter

- Filter configuration

- Construct distributed weather report DSS cluster

Introduction to filter

To implement distributed DSS clusters, SuperMap iServer provides the filter mechanism on the cluster server. Before load balancing, it filters the DSS nodes
(including the DSSs provided by the cluster server itself) using the filter mechanism and saves only the nodes that can provide correct DSSs.

SuperMap iServer provides the

com.supermap.services.cluster.client.spi.ClusterServiceFilter interface. The user needs to implement this interface to filter DSS nodes. The main methods of ClusterServiceFilter are the following:

```java
boolean filter(ServiceInfo serviceInfo, HttpServletRequest request);
```

This method is used to filter DSS nodes according to client requests and the information of each DSS node. The ServiceInfo.additions attribute is an array of Strings, which corresponds to the service ability information of the DSSs (extracted by NameMapping). If it returns true, the DSS node should be filtered out (i.e. not used); if false, the node can be used to provide DSS services (after load balancing, one of all the remaining nodes will provide the service).

Additionally, when implementing ClusterServiceFilter, one needs to add the @ClusterServiceFilterType notes. @ClusterServiceFilterType has two mandatory elements: componentType and interfaceType. They limit the type of service components and the type of service interfaces that a filter can work with, that is, the filter can only effect the service instances that have the specified type combination. componentType should be assigned as the name of the service component implementation type.
interfaceType should be assigned as the name of the service interface implementation type.

Example usage:

```java
@ClusterServiceFilterType(componentType = "Temperature", interfaceType = "JaxrsServletForJersey")
public class CityNameFilter implements ClusterServiceFilter {
    public boolean filter(ServiceInfo serviceInfo, HttpServletRequest request) {
        ...
    }
}
```

**Filter configuration**

Filters need to be configured on the cluster server. In the iserver-system.xml file (located in %SuperMap iServer_HOME%/webapps/iserver/WEB-INF), add a child node to `<clusterServiceFilters/>`, which is a child node of `<clusterService/>`:

```xml
<cluster>
    ...
    <clusterService>
        ...
        <clusterServiceFilters>
            <ClusterServiceFilter name="cityNameFilter" class="com.supermap.sample.cluster.CityNameFilter" />
        </ClusterServiceFilter>
        ...
    </clusterService>
</cluster>
```

**Note:** One can implement multiple filters under the `<clusterServiceFilters/>` node at the same time. They will be executed in turn.

 `<ClusterServiceFilter/>` may also have two optional attributes: componentName and interfaceName, that can further specify the scope of the filter.
Construct distributed weather report DSS cluster

In the weather query DSS cluster example,

com.supermap.sample.cluster.CityNameFilter implements the ClusterServiceFilter interface, that can filter the weather report DSS nodes, using the following codes:

```java
public class CityNameFilter implements ClusterServiceFilter {
    public boolean filterServiceInfo(ServiceInfo serviceInfo, HttpServletRequest request) {
        String cityName = null;
        if (serviceInfo != null) {
            String requestCityName = request.getParameter("cityName");
            if (requestCityName != null) {
                cityName = requestCityName;
            }
        }
        if (cityName == null) {
            return false;
        }
        String filterServiceInfo = serviceInfo.getServiceInfo().getCityName();
        if (filterServiceInfo != null) {
            String[] cityNames = filterServiceInfo.split(";
            for (int i = 0; i < cityNames.length; i++) {
                if (cityNames[i].equals(cityName)) {
                    return true;
                }
            }
        }
        return false;
    }
}
```

Deploy the weather query DSS on three SuperMap iServer servers--C1, S1, and S2. C1 only provides weather reports for "Paris", S1 "Tokyo", and S2 "Beijing" (when configuring TemperatureProvider, use C1.txt, S1.txt, and S2.txt respectively).

Using C1 as the cluster server, S1 and S2 as child nodes, one can construct clusters (see build domain spatial service clusters), by configuring filters on C1, i.e. com.supermap.sample.cluster.CityNameFilter. Open the configuration file of C1, services.xml, and add a child node to the node <clusterServiceFilters/> as follows:

```xml
<clusterServiceFilters>
    <ClusterServiceFilter name="cityNameFilter" class="com.supermap.sample.cluster.CityNameFilter"/>
</clusterServiceFilters>
```

Build the cluster structure as follows:
When someone accesses C1 requesting the weather report for "Beijing", that is, accessing the following link:

"http://C1:8090/iserver/services/Temperature/restjsr/Beijing", C1 will filter all the DSS nodes in the cluster, get the DSS node that can provide weather report services for "Beijing"--here it is S2, and have S2 respond to the client request (i.e. the request of the weather report for "Beijing"). The result is as follows:

Because of the filtering, when the DSS cluster respond to the request of the weather report for "Beijing", it will not use the weather services of C1 and S1; therefore, the accuracy is guaranteed.