**SERVICE MIGRATION DOCUMENT**

**METHOD 1: Here the client migrates the service request from cluster 1 to cluster 2**

**Client Code**

Check the server status, pre-process the data to be processed by the server, and send it to the server to receive the result. Mention the IP & port number of both the servers.

• **timer (start, end)**

Calculates and returns the time interval between start and end.

**• request\_to\_edge (ml\_type, client\_id)**

Sends the request to the edge server and receives a response.

**• scalar\_connect (ws, port)**

1. Process on data for server whose ML Type is Scalar
2. After connecting the WebSocket for communication with the server, initial 40 Scalar data are transmitted for Model Ready Test and the result value is received.
3. If it is in the Model Ready state, the data after the initial 40 data are divided by 40 and sent to receive the result of performing the ML task in the server and process it.
4. From the second transmission, the end-to-end required time, data processing time, and bandwidth are calculated and output in the process of processing each 40 data.

• **main** (Flow starts from here)

1. connects to request\_to\_edge and the response from the server is stored in resp
2. In the response json file, if migration is not 1, then the processing happens in the same cluster.
3. service\_port is fetched & ml\_type is checked.
4. Event loop (Web Socket communication) occurs connecting the socket Ip and the service\_port of that particular ml\_type
5. In the response json file, if migration is 1, then the requests are sent to other cluster.
6. connects to request\_to\_edge1. Gets the service\_port. Event loop occurs in the similar manner as mentioned above.

**Server Code**

(Here two edge servers are used. Both the edge servers run almost similar code)

Checks availability and overall management of edge server deployment based on the request received from the client

**• create\_deployment\_object (ml\_type)**

1. Initially, set container Resource Requirement according to ML Type
2. Set the node selector in the Kubernetes Pod and set the scheduling of Xavier and tx2 H/W in the edge server.
3. Container and pod template are created, and deployment object is created and returned based on them.
4. # Scalar Type Resource Setting - Request=3417969 KiB (Memory)

* Maximum Limit=3906250 KiB (Memory)

1. # Image Type Resource Setting - Request = 4394531 KiB (Memory)

* Maximum Limit = 4882812 KiB (Memory)

• **service\_request**

1. After processing the Kubernetes basic settings, set the ML usable capacity of Xavier and tx2 H/W.
2. The deployment created through create\_deployment\_object is determined through the presence or absence of the deployment created in the existing namespace.
3. # If there is no existing deployment - create a namespace and add the created deployment

# If you have an existing deployment - check the available ML capacity set in advance and add the created deployment if it is within the allowable range.

1. If the ML is out of the usable capacity, the created deployment is not added to the existing namespace. Return the service port, enabling migration to be 1 to the client.
2. Now, client uses the nearby alternate edge server by sending the request there.

**• main**

Initialize and start the Flask server

**METHOD 2: Here the cluster 1 migrates the service request to cluster 2**

**Client Code**

Check the server status, pre-process the data to be processed by the server, and send it to the server to receive the result. Mention the IP & port number of server 1

• **timer (start, end)**

Calculates and returns the time interval between start and end.

**• request\_to\_edge (ml\_type, client\_id)**

Sends the request to the edge server and receives a response.

**• scalar\_connect (ws, port)**

1. Process on data for server whose ML Type is Scalar
2. After connecting the WebSocket for communication with the server, initial 40 Scalar data are transmitted for Model Ready Test and the result value is received.
3. If it is in the Model Ready state, the data after the initial 40 data are divided by 40 and sent to receive the result of performing the ML task in the server and process it.
4. From the second transmission, the end-to-end required time, data processing time, and bandwidth are calculated and output in the process of processing each 40 data.

• **main**

Flow starts from here.

1. connects to request\_to\_edge and the response from the server is stored in resp
2. In the response json file, if migration is not 1, then it prints that the request is not migrated.

Else if migration is 1, then the client comes to know that the request is migrated to other edge servers.

1. Then, in the response file, service\_port is fetched & ml\_type is checked.
2. Event loop (Web Socket communication) occurs connecting the socket Ip and the service\_port of that particular ml\_type

**Server Code**

Here, two edge servers are used. Following is the flow for server one.

Checks availability and overall management of edge server deployment based on the request received from the client

**• create\_deployment\_object (ml\_type)**

1. Initially, set container Resource Requirement according to ML Type
2. Set the node selector in the Kubernetes Pod and set the scheduling of Xavier and tx2 H/W in the edge server.
3. Container and pod template are created, and deployment object is created and returned based on them.
4. # Scalar Type Resource Setting - Request=3417969 KiB (Memory)

* Maximum Limit=3906250 KiB (Memory)

1. # Image Type Resource Setting - Request = 4394531 KiB (Memory)

* Maximum Limit = 4882812 KiB (Memory)

• **service\_request**

1. After processing the Kubernetes basic settings, set the ML usable capacity of Xavier and tx2 H/W.
2. The deployment created through create\_deployment\_object is determined through the presence or absence of the deployment created in the existing namespace.
3. # If there is no existing deployment - create a namespace and add the created deployment

# If you have an existing deployment - check the available ML capacity set in advance and add the created deployment if it is within the allowable range.

1. If the ML is out of the usable capacity, the created deployment is not added to the existing namespace. In that case, the requests are sent to other cluster.
2. Create another function request\_to\_edge2 in which edge server 1 sends the same request to server 2. It gets the service port of that cluster and returns it to the client.

**• main**

Initialize and start the Flask server