**Live Migration of Containers in Edge Computing Embedded Platform**

**Abstract:**

With the advent of IoT and an increase in embedded devices, there are many devices that may rely on a single web-service. It is vital that the service should be available to the user at any cost irrespective of the number of requests or the downtime of the servers. As the number of devices connected increases day-by-day, the network traffic developed because of these devices also increases. As a result, the network becomes over loaded with the traffic and the provides poor latencies to the applications. In-order to overcome these kind of problems, the solution of edge computing is introduced along with live migration. The edge computing helps to move the cloud management services for multi-tenant user applications using a distributed heterogeneous edge node. Live migration is transferring application instances across nodes without disconnecting the clients. It helps to solve many problems such as downtime during hardware maintenance or unexpected failure.

At present, the existing cloud management services does provide only the remote deployment of multi-tenant user applications on the cloud of edge nodes, but it does not have any facility to perform live migration of the nodes across the application.

In this project, the auto discovery of all the active end nodes to remotely deploy multi-tenant user application will be achieved along with the live migration of the nodes across the application. The platform uses container based virtualization to deploy and launch isolated, multi-tenant user applications.

**System Modules:**

The system can be majorly divided into 4 parts:

* User Interface
* Cloud
* Edge nodes.
* Live migration.

Each part of the system is further divided into sub-systems:

* CLOUD:
* Database.
* Messaging service.
* USER INTERFACE:
  + User dashboard.
* EDGE NODES:
* Container.

DATABASE:

The database is used to store the details related to the user such as the login credentials. The database also contains the data about the container that is allocated to the respective user, the images of the container etc. The database is consists of three major tables.

1. compute - the table information of the user and the respective containers.
2. image - this table has the images of the container for the edge devices to be connected.
3. device - this tables lists the devices present.

In-order to communicate the data between the REST and the messaging service, a python package that wraps the SQLAlchemy ORM package is implemented.

MESSAGING SERVICE:

The server and the client messaging or communication is done with the help of Web Application Messaging service(WAMP). The WAMP is an open standard websocket protocol, which has two different types of messaging patterns in a single protocol.

1. Remote Procedure calls.
2. Publish-subscribe.

USER DASHBOARD:

The User dashboard in the design is to create , start or stop a container. The user can also run and upload an application into the container. The dashboard consists of different web-pages

1. Sign/Register screen.
2. Screen to create a container, or upload an application or check the device connected.

Horizon, an openstack framework, is used for created the user dashboard.

DOCKER CONTAINERS:

A complete software package required to run a user’s application on the edge node by abstracting the underlying OS.

Swarm is a container Schedule used.

LIVE MIGRATION:

Live migration is the new concept of extended in this project. The live migration of containers are done on the top of the existing project titled “Edge computing embedded platform”(Abhishek Gurudutt, Chinmayi Divakara, Praveen Prabhakaran and Tejeshwar Chandra Kamaal).

The live migration of the containers present in the edge nodes are done using CRIU (Checkpoint and Restore In User-space). This allows the users to backup and restore any process lively in the user-space.

Below are the commands to perform live migration using CRIU

Initially, the CRIU has to be installed.

**apt-cache search criu**

**apt-get install criu**

**Dump:**

In-order to freeze the process and migrate it live , the dump command is used on the source side.

**criu dump -t <pid> --images-dir**

pid can be obtained used the following command in the base directory.

**ps aux | grep *filename***

The image of the process will be created.

**Copy:**

The image created must now be copied to the target destination from the source.

**scp -r <image-dir>/ <dst-ip>:**

The image will now be copied onto the target.

**Restore:**

The process now has to be restored on the target.

**sudo criu restore --t <pid> --images-dir <path-to-images>**

Now the process will be restored onto the target side.