Project

Storage/Memory Structure and Subsystems

PREPARED FOR

IAS Project : IIIT-Hyderabad

PREPARED BY

Group1-Team 3:

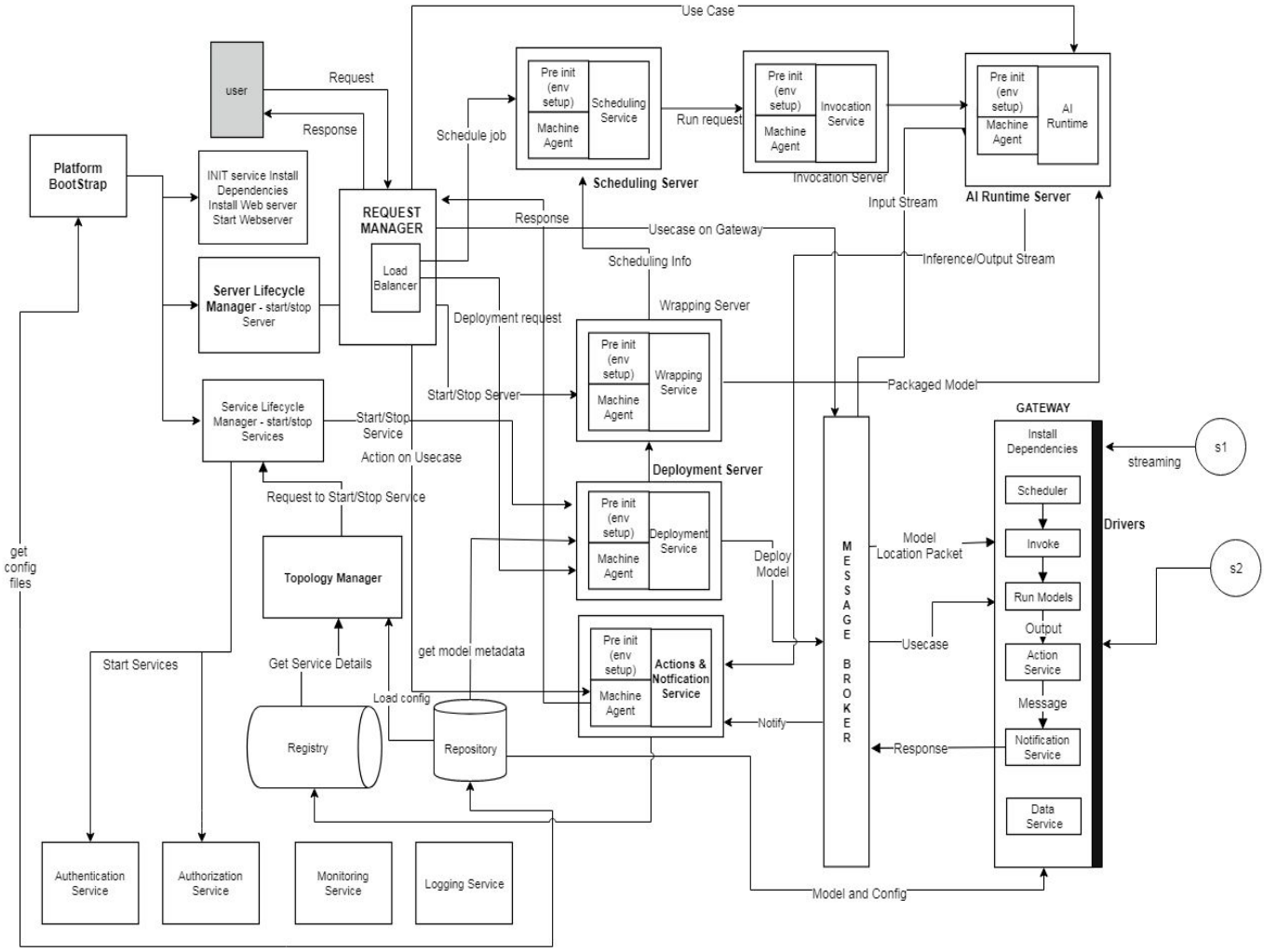
Amit Kumar

Rajesh Dansena

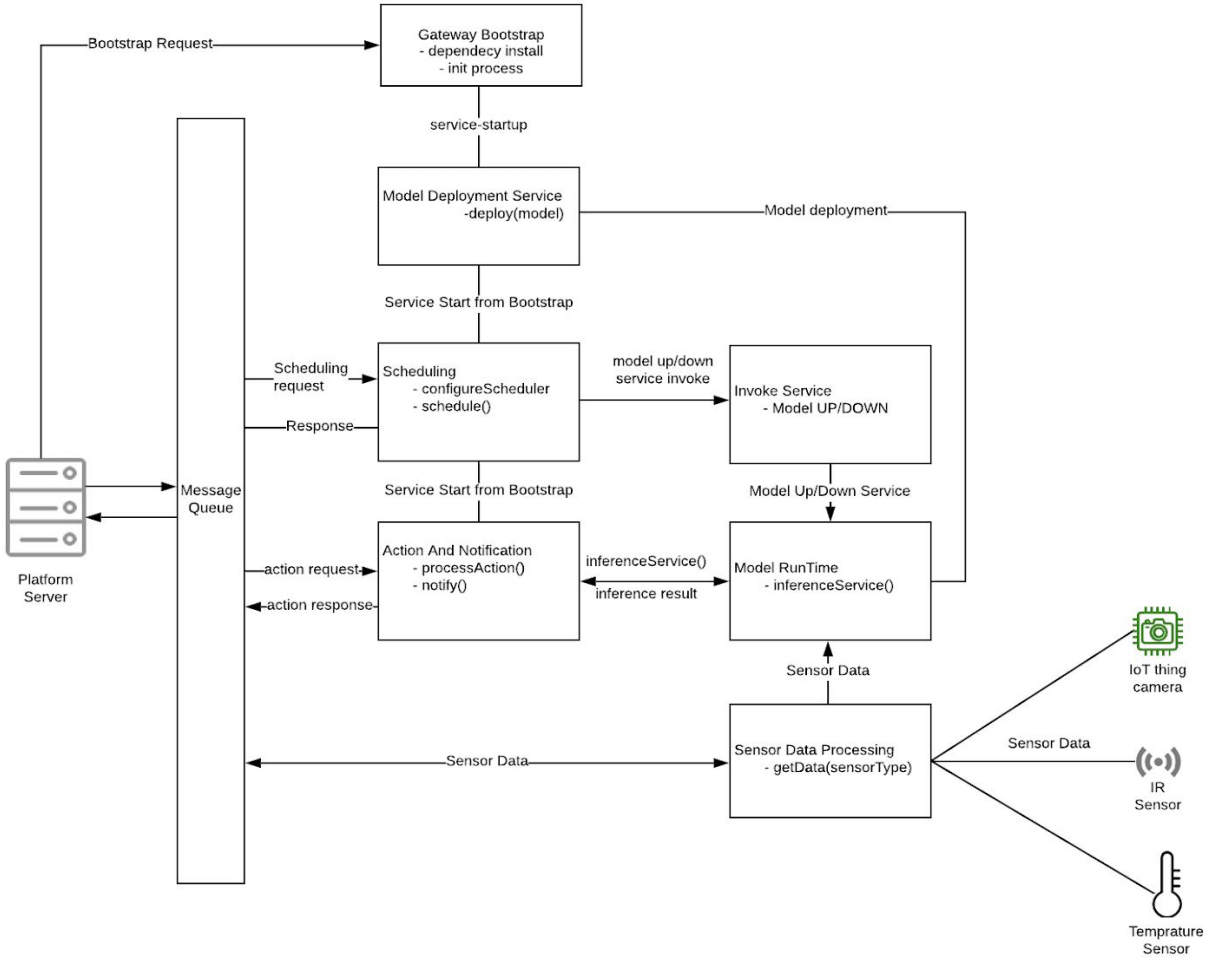
Document : V1.0

1. Block Diagram

1.1 Overall Project



1.2 Gateway


Bootstrap Request
Gateway Bootstrap
- dependecy install
- init process
service-startup
Model Deployment Service
-deploy(model)
Model deployment
Service Start from Bootstrap
Scheduling request
Scheduling
- configureScheduler
- schedule()
model up/down service invoke
Invoke Service
- Model UP/DOWN
Response
Service Start from Bootstrap
Model Up/Down Service
Message Queue
Platform Server
action request
action response
Action And Notification
- processAction()
- notify()
inferenceService()
inference result
Model RunTime
- inferenceService()
Sensor Data
IoT thing camera
Sensor Data
Sensor Data
Sensor Data Processing
- getData(sensorType)
IR Sensor
Temprature Sensor

2. Subsystems and APIs

2.1 Gateway Bootstrap

Sub-System Detail: This sub-system acts as agent to make the Gateway ready for Model deployment and other services as Notification and Action, Inference, Sensor Data processing and streaming etc. It has following APIs.

API:

- **installDependecy()** : This API is inoked by the cloud server and bootstrap installs all the dependency into the Gateway.
- **initProcess()** : Initializes all the sub-system across this Gateway Services. This helps to recover from a previous crash of the Gateway services. It sends the Service start request to all the Sub-systems in this Gateway and starts the services.

2.2 Deployment Service

Sub-System Detail: This Sub-System has all the deployment related services like Model deployment and Inference Service startup. Once bootstrap installs all the dependencies, This service worker deploys the Machine Learning Model on Gateway and keep ready for inference.

API:

- **deploy(Model)** : Takes input as Model data structure which contains Model related information and Model URL from where Model is downloaded and deployed into the Gateway.
- **undeploy(Model)** : Takes input as Model data structure and undeploy any existing Model running/deployed on the Gateway. This API has been given as future scope to the platform to clear up the memory and Model running on the Gateway.

2.3 Scheduling Service

Sub-System Detail: Scheduling Service is responsible to schedule the Model deployment and Model GRPC service startup. There can be many scheduling type possibilities. Based on the User's scheduling input it does the scheduling of the Model on Gateway. Scheduling service is helpful to save the memory and processing usage of the system.

API:

- **configureScheduler(CongurationDetail)** : configures the scheduler before hand. This service takes configuration details like Model and system details where the

Model is deployed and details about the Invocation Service in the Gateway. Invocation Service is the System which is responsible to invoke the start and stop service. Scheduling Service does call the invocation service based on scheduled time.

- **schedule(SchedulingDetail)** : This service schedules the configured service based on the Scheduling details provided by the User or other system.

2.4 Action and Notification

Sub-System Detail: This Sub-System is responsible for doing some action or notifying user based on the User's given action condition. User gives the action file while deployment itself as part of the zipped packaged file. The action file is plugged in here and used to act and notify accordingly.

API:

- **processAction()** : This API is invoked on inference request. Here, inferencing happens based on user's inference input and on correct inference, action gets triggered from here.
- **notify()** : This API is to notify the user. Here user can be notified on email, sms or any configured way. This could be future scope of the platform and can be extended further.

2.5 Invoke Service

Sub-System Detail: Invoke Service invokes the system services of different sub-systems like service stra-up and stop, service recovery etc.

API:

- **serviceStart(ServiceType)** : This is to start a Service based on provided Service Type.
- **serviceStop(ServiceType)** : This is to stop a Service based on the provided Service Type.
- **pingService(ServiceType)** : This API is to check a service whether it is in UP or DOWN state.

2.6 Model RunTime

Sub-System Detail: This is where the Model runs and does expose the Model inference gRPC services for user inference request.

API:

- Note that this will have dynamic service APIs based on the User Model file and it's configuration details.

2.7 Sensor Data Processing

Sub-System Detail: This sub-system connects with sensors through their drivers and gets the raw data of streams. It processes the raw data into proper model input form and starts streaming it to the on-premise Model or to the Message Queue which is consumed by the Model deployed on the server.

API:

- **getData(SensorType)** : This API is provides the processed data. It takes Sensor Type as input and gives that sensor's data stream.

3. Storage/Memory Structure

Below are the list of data structures we have used in the application development :

- Python Library Data Structures :List, Dictionary, Set, User customized Objects.
- Scheduler Data Structures: runqueue (push_cpu,migration_thread,migration_queue)
- User Defined Data Structures : User defined Classes, Interface, Tree.

Storage Structure:

1. Sensor Information
    - Sensor Type
    - Sensor Location
    - Sensor ID
    - Gateway ID in which sensor is deployed
2. Gateway Information
    - Gateway ID
    - Gateway Name
    - Gateway IP
    - Gateway PORT
    - Sensor Lists
3. Server Information
    - Server ID
    - Server Instance Name
    - Server IP
    - Server PORT

    - Services information
4. Scheduling Information
    - Schedule Type
    - Schedule Start Time
    - Schedule Stop Time
    - Model to schedule
5. User Information
    - User name
    - User Id
    - User Password Salt
    - Registered Gateways
    - Models uploaded
6. Network Topology
    - Service Location
    - Number of services running
    - Server Instances
    - Server-Service mapping


```
{
"ModelList" :
[
        {
                "Modelname":"my_model1",
                "FileName":"iris.zip",
                "Type":"iris",
                "InputStream" : "192.168.43.54",
                "ModelPath":"iris/",
                "DeployIp":"192.168.43.110",
                "DeployPort":"8501",
                "DeployPassword":"dhawal@A1",
                "DeployUserName":"dhawal",
                "StartTime":"22:50",
                "EndTime":"22:51",
                "Repeat":"YES",
                "Count":3,
                "Interval":2,
                "Repeat_Period":10
        }


        ]
}
```