INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY

HYDERABAD

Internals of Application Server

Hackathon 2

Group: 3
Team : 2

Team Members:

Anchal Soni (2020201099)

Archit Gupta (2020201075)

Naman Jain (2020201080)

Contents

Scheduling Module

1. Introduction

1.1 Brief overview

In IoT Based Application Platform Scheduler is one of the integral modules which is responsible for scheduling algorithms to be run on the platform.

It will be responsible for handling the scheduling Requests, Start and STOP requests.

1.2 Artefacts and data structures

- Queue: A queue will be used to keep track of the start and end times of the algorithm to be run.

  Format:

Task 1 [app_id, Start_time, End_time, [sid(s),..]]	Task 2	...	Task n

- Task-packet: A structure in the form of a 2d list containing necessary information.
  1. It contains Application id uniquely identifying the application to be deployed.
  2. Start time of the process to be run
  3. End time of the process
  4. Sensor id(s) of the sensors needed to communicate during the execution.

  Format: [app_id, start_time, end_time, [sid1, sid2,...]]

  The scheduler will receive this validated task packet from the platform manager.

- Monitoring file: Scheduler will write its pulse after each time interval specified by the monitoring module. Monitoring module will read this common monitoring file to check if the scheduler is live or not.

- Log file: It is a state maintaining file used to maintain the state of the scheduler. Previous two records will be present at a time in the log file. In case of module failure this log file will be read to avoid data loss.

1.3 Active users in frame

There is no direct communication of users with this module but the active users in frame are as follows:

1. End user: Application developer will interact with the platform manager first for executing a service. The platform manager will then communicate with the scheduler and send a task packet to execute.

1.4 Use cases

Different use cases are mentioned below and described later in the document in detail

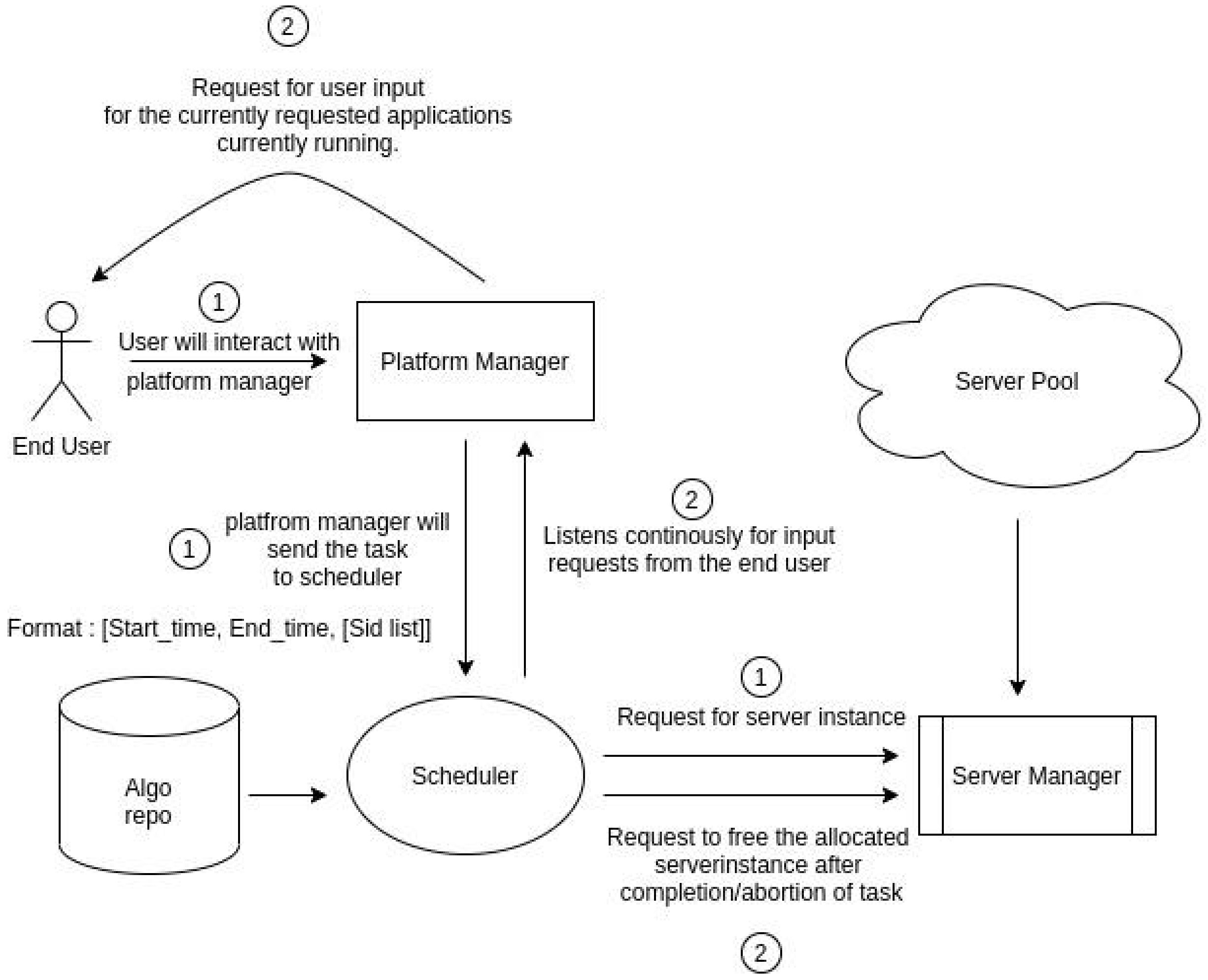
- Schedule algorithms based on the start time of task provided by the platform manager
- Request for server and get server instance(IP, PORT)
- Requests End-User through Platform manager to provide control inputs for our running Application.
- Write its latest record in a log file after each specified time interval. In case of module failure this log file will be read to avoid data loss.

1.5 Interaction with other modules

- Platform manager: Receives application run request from user and after validating request constructs a task packet to send it to scheduler.
- Server manager: Request for server instance. Server manager takes care of load in servers and accordingly allocates a server instance having least load. If all available servers reach a threshold, it allocates a new server instance.
- Platform init module: To reinitialise scheduler in other machines of the network in case of machine failure.

2. Block diagram and Flow

2.1 Block diagram:


2
Request for user input
for the currently requested applications
currently running.
1
User will interact with
platform manager
End User
Platform Manager
Server Pool
1
platfrom manager will
send the task
to scheduler
2
Listens continously for input
requests from the end user
Format : [Start_time, End_time, [Sid list]]
1
Request for server instance
Algo
repo
Scheduler
Server Manager
Request to free the allocated
serverinstance after
completion/abortion of task
2

2.2 Flow:

when the scheduler is initialised, it reads its latest state from a log file. The first time scheduler is initialised it reads an empty log file. During its run, after each particular time interval, the scheduler writes its state into the log file which in case of machine failure or if by any chance the scheduler is reinitialised, it can read its latest state and the data is not lost.

When the platform manager gets an end user request, it validates and prepares a task packet for the scheduler and sends it to the scheduler. The scheduler extracts the application id, start time , end time and sensor ids from the task packet and reads the actual code logic of the algorithm from the repository.

2.3 Technologies used:

Python version 3.8 : Programming language used for implementation of modules
Flask API : For Making Flask Servers.
Kafka : For Communication.
Pyro4 : Used for RPC between application instance and server instance.

3. Input to Scheduler

Input to the scheduler will be received from the platform manager in the form of a task packet. Once the scheduler extracts the application id sent by the application manager it will read application code from the application repository where the actual code for the application resides.

4. Output from Scheduler

IP and PORT received from the server manager will be sent to the platform manager. In case of machine failure an error report will be generated and sent to the platform administrator.

5. Fault tolerance

Three can be three types of fault possible:

1. When the system on which our module is running goes down. This type of situation is handled by a common monitoring and fault tolerating module which will supervise the machine condition manually and in such situations reports to the platform admin for initializing the scheduler module in some other suitable machine of the distributed system.

2. When the scheduler goes down: This fault is detected by the monitoring module. If the scheduler fails to write its pulse in the common monitoring file the monitoring module reports an error and requests the platform initialiser to reinitialise the scheduler in some other machine.

3. When the scheduler throws some unidentified error: monitoring module is responsible to check for these errors and resolve them.

4. When a server instance goes down: The fault tolerance module of the server will come into action and will request the server manager for a new server instance.

6. Test Cases

1. Validating the start and end time

API call can be made to start_end_time_validator providing the input JSON consisting of the various details including start time and end time of a process which will return Boolean after checking if the start and end time are valid and can be scheduled.

Input: Task - packet From Platform manager(2-D List Format)
Output: Boolean

2. Validating application

We will check all the sensor ids, if any sensor id received for an application is “False” then we must not schedule that application and inform the Platform Manager about the same.

3. Server Failure

The fault tolerance module will call the server manager in case of server failure and reassign a new instance of server to the current running process.

4. STOP Request

The end user can Stop the execution at any time if he wants by Giving suitable input from the user menu. It will END the application immediately and will free the server instance that it had occupied from the server manager.
Input: From Terminal by END-USER.