IAS Project

TEAM REPORT

Scheduler and Server Life Cycle Manager

Submitted By :
Danish - 2018201016
Jay Krishna - 2019201019

Contents

1 Scheduler

1.1 Introduction

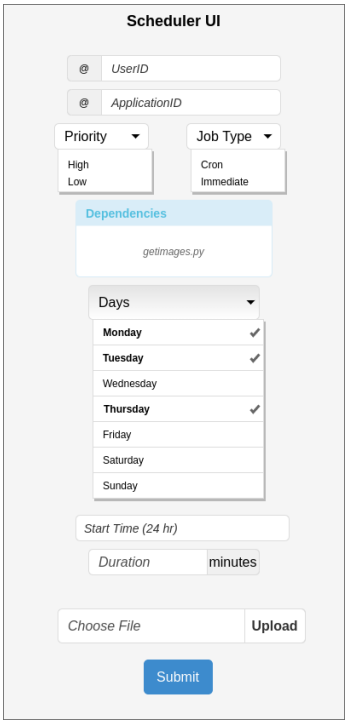
- Job Scheduler module is responsible to run/send jobs to deployment manager periodically at predetermined intervals for execution

- There are two sub-parts of the Module.
  1) Schedule Execution
  2) Schedule Termination

1.2 User Interface

The user-interface for scheduler will look like the figure as shown below. Application Developer can give scheduling data in three formats

- XML format
- Json format
- UI

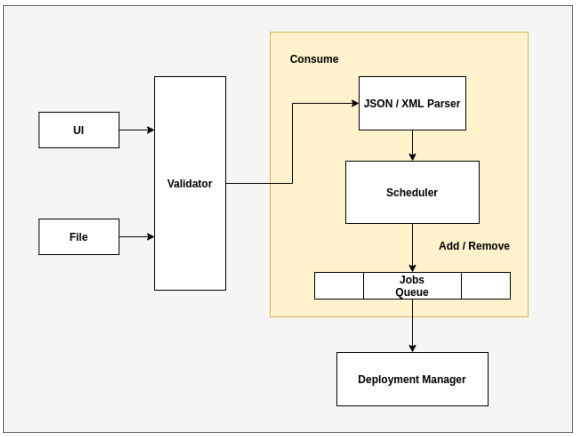


For the XML and Json file input, the validator will validate whether the files are in the predefined format.if not , it will show error to the user. if done using API call ,will return error message.

1.3 Data Flow

- The scheduler will receive the scheduling information in json/xml/ui format to get scheduling related metadata after the application is being accepted by Application Manager.
- The scheduler will validate scheduling config file with predefine format.
- If the given meta data is valid , then meta data will be stored in scheduling Queue.
- Each job stored in Queue will be then scheduled for specified time (start time) as given by the application manager in the meta-data.
- On the invocation of scheduled execution function for a particular job. Scheduler validates the dependencies taken from configuration file of application, job to kill the execution is scheduled (end time) as given in the meta data.Request for execution of algorithm is forwarded to the Deployment manager.
- Dependencies: Will specify the dependencies that must be run before running the algorithm strictly in the order they are required in. Scheduler will schedule the dependencies in the same order specified then will schedule the algorithm.
- On the invocation of scheduled kill function, request to kill the process is send forwared to deployment manager.

1.4 System Design



1.5 Technology used

KAFKA

Kafka is used for real-time streams of data, to collect big data, or to do real time analysis (or both). Kafka is used with in-memory microservices to provide durability and it can be used to feed events to CEP (complex event streaming systems) and IoT automation systems.Kafka is often used in real-time streaming data architectures to provide real-time analytics. Since Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system.

MongoDB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

Fluentd

Fluentd is an open source data collector for unified logging layer. Fluentd allowsyou to unify data collection and consumption for a better use and understanding ofdata.Fluentd treats logs as JSON, a popular machine-readable format.

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[3] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

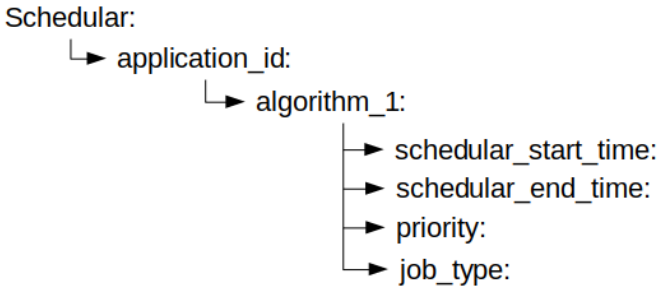
Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

1.6 Scheduling Data Input

Configuration file should contain following meta-information:

- Userid: unique user id for the user.
- Applicationid: unique id of the application of that user.
- Priority: This will specify if the algorithm is computation intensive or requires a high amount of resources.
  - If specified "high" the algorithm will run on a single node.
  - If specified "low"(default) the algorithm will run on shared nodes.
- Jobtype: This will be used by the scheduler to understand how to schedule the algorithm. Two choices are cron single:



  - If specified as a cron job scheduler will schedule it at periodic intervals.
    - Application Deployer can specify day/days at which the algorithm will be scheduled.
    - Also start time(in 24 hours format) and duration for which it should run(in minutes) will be specified.
  - If specified as immediate instance user have two choices:
    - If no start time given the scheduler will immediately schedule it to execution for the time duration specified.
    - If given a start time scheduler will schedule it at the given time for the time duration specified.

Detailed Configuration file is as follows:

```xml
<Services>
    <service>
        <name>service1</name>
        <filename>service1.py</filename>
        <priority>low</priority>

        <dependencies>
            <dependency>service2</dependency>
            <dependency>service3</dependency>
        </dependencies>
        <sensor_input>
            <input>
                <type>BY_ID</type>
                <id>12345</id>
            </input>
            <input>
                <type>BY_LAT_LON</type>
                <lat>1234.12</lat>
                <lon>24.213</lon>
                <floor>2</floor>
                <radius>12</radius>
                <count>ALL</count>
            </input>
        </sensor_input>
        <scheduling_info>
            <file_name>scheduling1.xml</file_name>
        </scheduling_info>
    </service>
</Services>
```

Scheduling File Format

```xml
<schedules>
    <schedule>
        <!-- Schedule 1 Info -->
    </schedule>
    <schedule>
        <!-- Schedule 2 Info -->
    </schedule>
</schedules>
```

Different Types Of schedules
SINGLE INSTANCE

```xml
<schedules>
    <schedule>
        <type>SINGLE_INSTANCE</type>
        <start_date>12-12-2020</start_date>
        <time>
            <start>10:00</start>
            <end>12:00</end>
        </time>
    </schedule>
</schedules>
```

DayWise

```xml
<schedule>
    <type>DAYWISE</type>
    <day>MONDAY</day>
    <time>
        <start>10:00</start>
        <end>12:00</end>
    </time>
</schedule>
```

Periodic

```xml
<schedule>
    <type>PERIODIC</type>
    <start_date>12-12-2020</start_date>
    <period>10</period>
    <time>
        <start>10:00</start>
        <end>12:00</end>
    </time>
</schedule>
```

```
<sensors>
    <sensor>
        <name>BTF-14</name>
        <gateway>ip:port</gateway>
        <sensor_id>SI12</sensor_id>
        <data_type>VECTOR</data_type>
        <size> 20 </size>
        <address>
            <area> iiit </area>
            <building_name>nilgiri</building_name>
            <room_no>D12</room_no>
        </address>
        <geo_location>
            <lat>1234.242</lat>
            <lon>854.21</lon>
            <floor_no>4</floor_no>
        </geo_location>
        <type>INPUT_OUTPUT</type>
    </sensor>
</sensors>
```

1.7 Data Structures

Queue

All the incoming requests for scheduling are stored in the queue as there may be multiple requests at a time and on the fifo basis they are stored. Then every request is popped and scheduled according to the scheduling start time defined in meta data file.

Heap

All the jobs scheduled for execution are stored in the min heap on the basis of start time,and after every period of time the heap is accessed and top of the process is checked for scheduling .if the current time is greater than or equal to start time of that job then request is made to deploy management for the execution.

2 Server Life Cycle Manager

2.1 Introduction

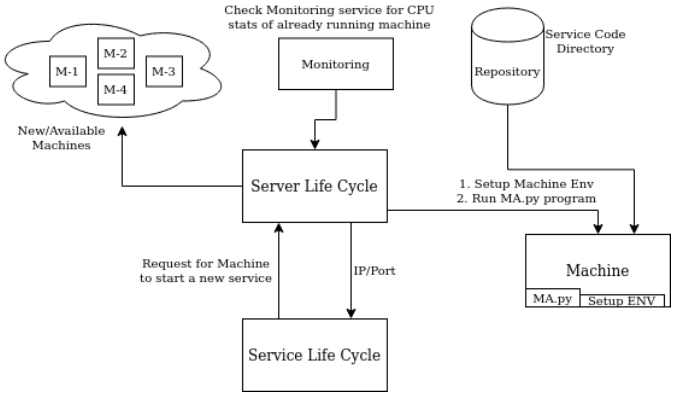
Server Life Cycle Manager is responsible for allocating machine to service life cycle manager while maintaining optimal load among all running machines.

2.2 Data Flow

- The server manager will receive request from service life cycle manger to allocate an instance.

- Server life cycle manger will ping monitoring service to get current load of different servers.
- It will then decide which server to allocate to particular service.
- If no servers can handle the request it will query the registry to get list of free available instances,setup environment on that, run machine agent code and return that to service life cycle manager

2.3 System Design



2.4 Data Structure

Dictionary

To maintain mapping between IP,port load of respective server.