

Cloud Computing (CS466) Assignment

Title : Matching Theory Framework FOG Computing	
Student 1 Name: Amal Majunu Vidya	Student 1 Roll No.: 191CS107
Student 2 Name: Aakash Chandra	Student 2 Roll No.: 191CS132
Student 3 Name: Praneeth G	Student 3 Roll No.: 191CS235
Student 4 Name: Himanshu Kumar	Student 4 Roll No.: 191CS122

Introduction

Fog computing, fog networking, or fogging refers to an architectural approach that employs edge devices for performing significant computation tasks (known as edge computing), communication, and storage within a local network while using the internet backbone for routing. The primary objective of Fog Computing is to enhance efficiency and decrease the volume of data that needs to be transmitted to the Cloud for large-scale data analysis, processing, and storage.

Fog Computing (FC) is a novel concept that expands cloud computing to the periphery of the network. It involves a decentralized computing system that operates within a specific geographic location where Internet of Things (IoT) services are performed on smart devices called fog nodes (FNs). These FNs have computing, storage, and network capabilities and are situated at the edge of the network. These FN nodes can offload the computationally intensive tasks from the Edge Devices(ED) which improves the speed of services and responsiveness, and allows for mobility support for the devices.

The main idea of this paper is to provide an efficient algorithm for offloading the tasks which are computationally intensive from the EDs to the FNs using Matching Theory. It aims to reduce the time complexity under the assumption that the EDs will select a proper FN to offload its computationally intensive task.

Constraints

- The number of task types are preset. So each task that an ED sends will be one of those preset task types.
- An ED selects FN based on the transmission between the particular FN and itself, waiting time due to already computing tasks at FN and finally the runtime for the computation
- Each FN can have a preference towards the tasks which it can take. Similarly, an ED can have a preference over which FN to offload to based on the time it takes to perform the computation or based on the distance to the FNs since the sending rate might be slower.
- There is a limit on the number of tasks a FN can service in parallel.
- It is assumed that an ED is capable of sending only one task, and that each task can be compiled into a single packet of information for processing.
- FN is capable of comprehending the waiting time, when an ED queries. Similarly, even ED can comprehend the number of instructions it has, when FN queries.

Proposed Algorithm

1. Create the preference order of FN for each ED. This is inversely proportional to the sum of the cost of the communication link (*comCOST*) and the waiting time of the task before being processed if offloaded (*waitingTIME*).

$$preferenceFN = \frac{1}{comCOST + waitingTIME}$$

Using this, preference is calculated for each FN at each ED.

2. Create the preference order of tasks for each FN. This is inversely proportional to the sum of the cost of the communication link (*comCOST*), the waiting time of the task before being processed if offloaded (*waitingTIME*), and also the computing time required (*computeTIME*).

$$preferenceTask = \frac{1}{comCOST + waitingTIME + computeTIME}$$

Using this, preference is calculated for each task at each FN.

3. Edge devices propose its task to their preferred FN.
4. Each FN node accepts the proposals(task) till its maximum capacity to handle tasks that can be computed in parallel, is reached. Once the capacity is reached, further proposals are rejected using the Deferred Acceptance Algorithm (DAA), where, if the newly received proposal has higher preference, then the less preferred proposal among the already accepted ones is replaced.
5. Once the DAA is complete, each FN node conveys to the EDs the waiting time.
6. Using this waiting time ED updates its preference order of FNs and sends its proposals to new preferred FNs. Then each FN accepts the preferred ones among the received proposals, and rejects others.
7. Repeat step 5 and step 6 until all tasks are allocated.

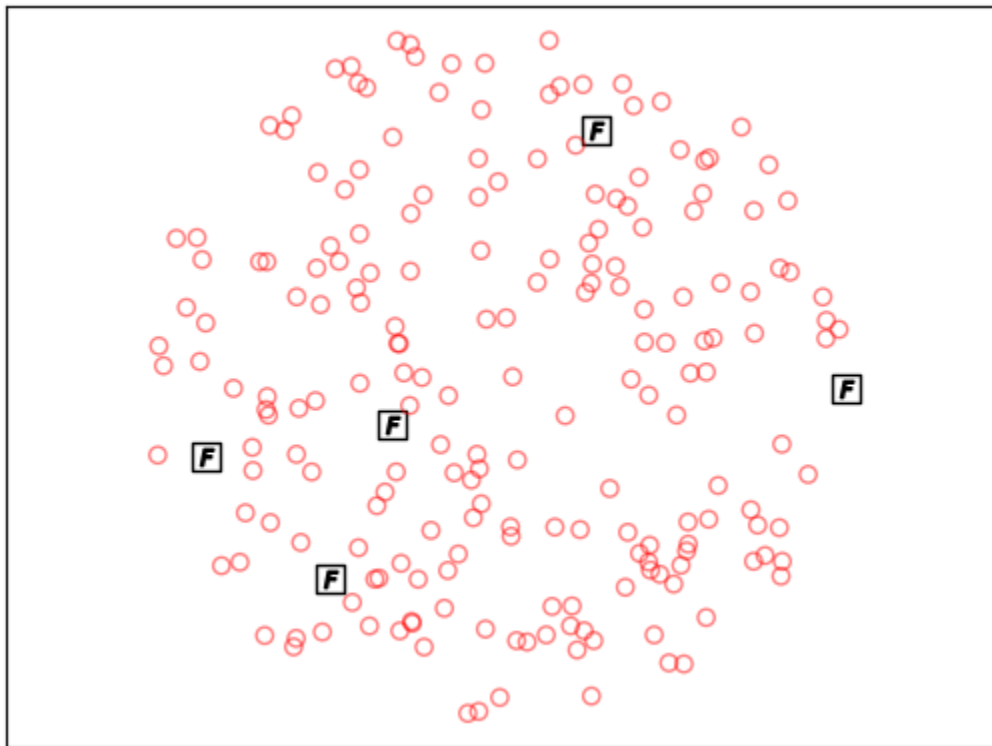
Implementation

Consists of 2 experiments :

- Experiment 1 - Number of tasks = 20, Number of Fog nodes = 5
- Experiment 2 - Number of tasks = 200, Number of Fog nodes = 5

Steps for implementing :

1. Initialize the
 - No. of FNs
 - No. of EDs
 - No. of Task types
2. Create an environment of FNs and EDs within a specified region



3. Assign different CPU speeds for different Fog Nodes
4. Make a plotting function to show the instantaneous task offloading to different FNs
 - Use different colors to differentiate between the tasks offloaded to different FNs
5. Implement the Proposed algorithm
6. Using the plotting function, print the environment every iteration for simulation.

Experimental Results

[Github repo](#)

As the algorithm is running, it can be seen that tasks from each edge device are being offloaded to a Fog node.

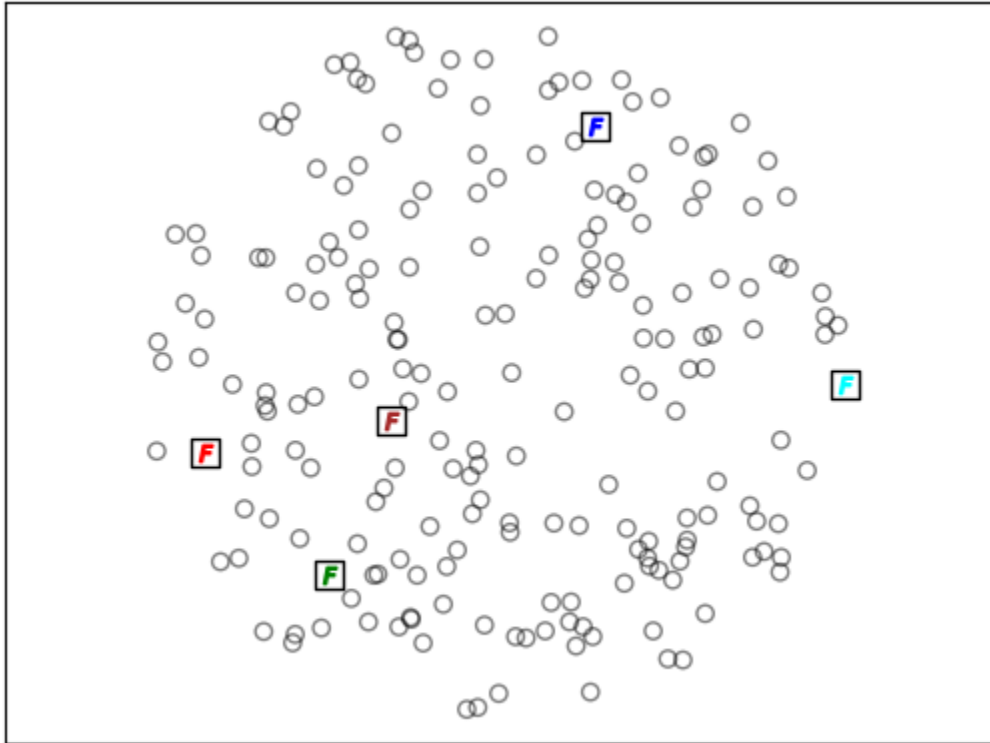


Fig1. Initial Plot - when no tasks are allocated

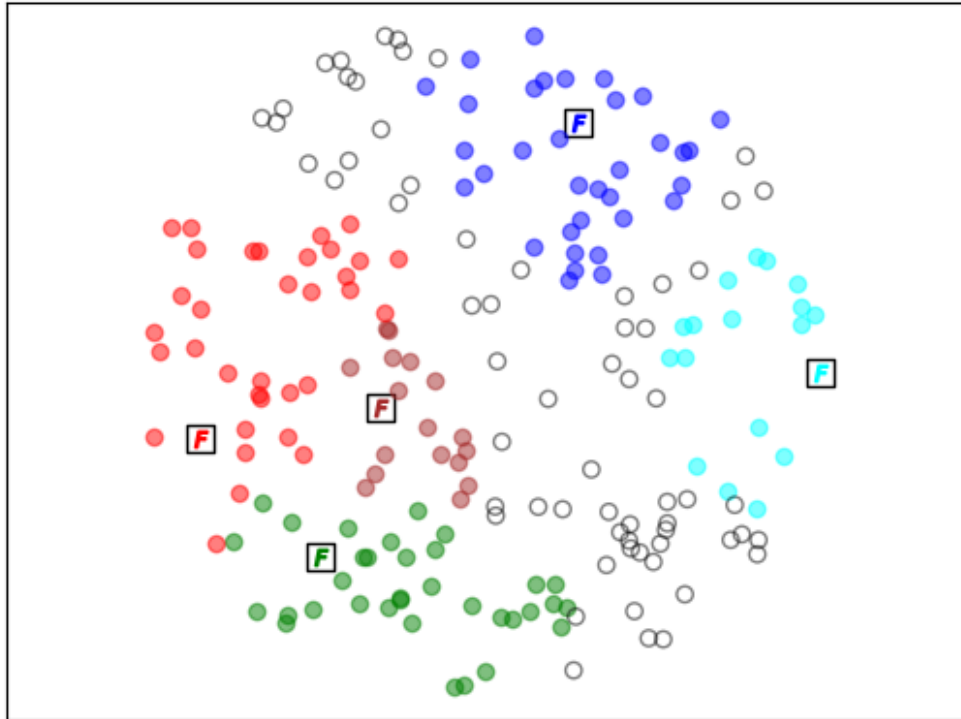


Fig2. One instance of Simulation where tasks are offloaded

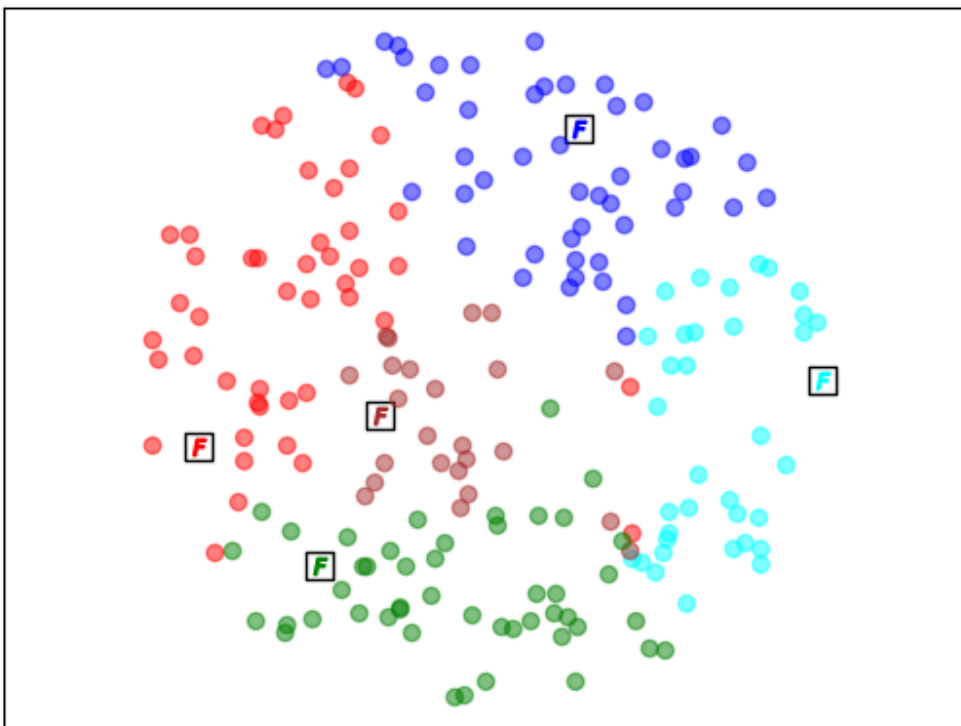



Fig3. Final plot of Simulation

Note:

- - Task “not yet offloaded”
-  - Fog Node assigned to “red” color
- - Task “offloaded to red” Fog Node