Project Submission Procedure:

- You can do the project in a group (upto 3 students are allowed in a group). One group of student will do one problem out of 4 problems.
- We will circulate a MS form and maximum of 12 groups are allowed to go for one problem. Every group will specify their preference in the order for all the four problems. https://forms.office.com/r/UJWGRiVb2i
- Send Source code of the implementation, Readme file about how to compile and run the code and platform used (linux, window, gcc, devc)
- Submit Latex/Word docuement of solution approach,
- Generate test case using random data: show the result for varying number task, machines/servers, other parameters etc.
- Deadline is 16th April 2024, Sumbit throght email: <asahu>AT<iitg.ac.in>
- Copy case lead to F grade.

P1: Placement and Scheduling Data Intensive Jobs in Edge-Cloud System

Consider a set of J jobs (arrived at time 0) that need to be processed by a cloud consisting of N physical machines (i.e., nodes) that are homogeneous. Each job j has a deadline dj and is required to access a set $\mathbf{C}_j$ of equal-sized chunks, we can assume each job j has $|\mathbf{Cj}|$ number of tasks accessing one data chunk each and can be run in parallel on the same machine or different machine. The chunks are stored in a distributed file system on the cloud. Each node is capable of hosting up to $\boldsymbol{B}$ data chunks and is equipped with S VMs which implies each node is able to simultaneously process S jobs. Let $\mathbf{C} = \cup \mathbf{C}_j$ be the set of all data chunks available at the central storage server, before processing the request we need to bring the data chunk to the physical machine. Replication of data chunks in different machines is allowed and data chunk needs to be placed only once at the beginning (before any job starts execution, once any one job starts the execution, we are not allowed to change the data placement) and during run time we can not place/replace/replicate the data chunk.

The time for each job j to process a required data chunk is unit time and it is the same for all the jobs. Completing a job j before the deadline d_j is equivalent to processing all the required chunks c of C_j before the deadline. Only one VM can access a data chunk in a given time slot of the same physical machine.

The problem aims to minimize the total number of active nodes ($\mathbf{N_a}$) to process all the jobs. The active node means the node stores at least one data chunk and is processed by at least one job. The number of active machines is always lower than the total number of machines which is $\mathbf{N_a < N}$.

P2: Trust Aware Scheduling of Online Tasks in FoG Nodes

Given a stream of tasks reaching the system online way where every task is associated with a user from a total of U users, we need to devise an approach to schedule tasks such that it minimizes the cost and maximizes the trusted reliability. Every task has arrival time, execution time, user information, and relative deadline of task ($\mathbf{d_j=a_j+K}$), where K is constant and K > p_j for all the tasks.

The underlying system has **M** homogeneous FoG/processing nodes and each FoG node m is associated with shared trust (can be represented as **ST[m]** of **ST[M]**, average user rating for the node/brand value of the node) and individual/direct trust from each user (**DT[u][m]** of **DT[U][M]**) for a node. Initially, all the user ratings is **0.5**. Trust is rated from 0 to 1 representing the probability of successful execution.

The individual trust of a node gets updated for every successful execution (trust increase) or unsuccessful execution (trust decrease) of the task. The **ST[m]** is the average of direct trust which is the s**um of DT[u][m]/U**. The cost of unit execution time on a node is determined by the shared trust value (Brand Value) of the node and this relation is not linear, $\mathbf{C = Base + BR*ST[m]^{2,}}$ where Base value is around 30% of brand constrant **BR**. We can associate a node with task failure probability and this failure rate changes over time. Unsuccessful execution of a task on a node depends on either (a) it missed the deadline or (b) get a failure during execution based on the failure probability of the node (this failure probablity of task can be modeled as FP=**1-exp (-f.t)** where ***f*** is failure rate $0 \leq f \ll 1$ of machine and ***t*** is the execution time of the task. *// modified problem statement FP **exp (-f.t)** updated to **1-exp (-f.t)***

Every task comes with reliability requirements and we need to schedule the task on machines such that it meets the reliability, meets the deadline constraints, and minimizes the cost.

P3: Data Quality Aware Profit Maximization in Sensor As a Service

The Sensor Cloud Service Provider (SCSP) provides a set of **K** sensor services **S**, each service is denoted by $\mathbf{s_k} \in \mathbf{S}$. The total number of services **K** is determined by multiplying the number of geographical locations by the type of sensor and the quality of sensor data. Every sensor service $\mathbf{s_k}$ associated with the amount of raw data $\mathbf{RD_k}$ to be collected from sensors and processed data size $\mathbf{PD_k}$, in general, $\mathbf{PD_k < RD_k}$. Every service s_k is associated with a cost of sensing $\mathbf{CS_k}$. **ES** is a set of **M** edge server nodes and each edge server node is denoted as $\mathbf{e_j} \in \mathbf{ES}$, **U** is a set of **N** users and each user is denoted by ui such that $\mathbf{u_i} \in \mathbf{U}$.

Users ui submit a request to the SCSP denoted as t_i, each with a tuple representation $< a_i, s_i, r_i, d_i, dr_i >$, which includes information about arrival time a_i, the requested sensor service s_i, the associated price r_i, deadline d_i, and data reliability requirement dr_i. Once the edge server e_j receives the task t_i from the user u_i through the SCSP, it collects sensor data and processes the task on the edge server ej or other edge servers. We are given the distance between edge servers **DEE[M][M]**, the distance between edge and users **DEU[M][N]**, and the distance between **DSE[K][M]**. We can safely assume transmission time is proportional to distances and amount of data. The number of requests may be higher than the number of users and a user may submit multiple requests at different times.

The price of the user task depends on the type of sensor service required and the task's deadline. The time required to complete a task depends on the time needed to sense the data (ST), process the data at the edge server (PT), and data transmission time (NT) to the user. So, the total task completion time is represented as execution time $\mathbf{e_i = ST + PT + NT}$.

The urgency Ui of the task t_i can be determined by the execution time and task deadline, as $\mathbf{U_i = e_i / d_i}$**.** The revenue of a task is determined by the price charged for sensing (CS_i), processing at the edge server (CP_i), and data transmission (CN_i) to the user assuming data processing happened in the requested edge server given by the SCSP. The revenue r_i of a task is represented as, **Revenue** $\mathbf{r_i = e_i * (1 + 2 * U_i^2)}$. The price for user service is the same as the revenue of the services and is known to both the user and the provider. Price is meant for the user and revenue is meant for the provider. If the processing of the sens data happens at different edge servers then we need to consider the transmission cost between the edge servers. Profit is revenue minus the cost of sensing, processing, and transmission. Profit will be calculated if the sensing task meets the deadline.

Devise an efficient approach to schedule the sensing and processing, such that the overall profit of SCSP is maximized.

P4: Task Scheduling in Edge Server Systems with limited Solar Energy and Infinite Batteries

Given the **M** edge servers each equipped with a solar panel of capacity **S** and a battery with infinite capacity. Assume a day has **T** time slots (for example T=24, 24 hours). Power generation for different time slot (jth) of the day at each edge server (ith) is given as **S[i][j]** and **S[i][j] ≤ S**. The number of tasks that arrives at edge servers (ith) at different time slot (jth) of the day is given as D[i][j]. **The values of S[i][j] and D[i][j] for all i and j are available at time 0.** *We know how much power generated at each edge server and how many number of tasks to arrive at each servers for all the time slot.*

The task needs to be executed in the same time slot and if the task executed it consumes a unit (or 1) amount utilization. We may choose to execute the task or not execute the task in the time slot.

The amount of power consumed in a time slot at an edge server is cube of the number of tasks executed in the time slot in that edge server. The compute capacity of each edge server is infinite. In a time slot, we may choose to store some power generated by the solar at the edge server or we may utilize some already stored power to execute some extra tasks or may not do anything with the battery. If we have stored power in the battery in earlier slots at the edge server then only we can use the battery up to the stored amount in the current slot.

The task can be migrated to another edge server without any cost. Battery power cannot be transferred from one edge server to another server.

Design an efficient approach to maximize the total number of task executions for the entire day by all the edge servers.