Task Scheduling in Edge Servers with limited solar energy and unlimited battery

April 13, 2024

1 Problem Statement Description

Given , there are M edge servers each equipped with solar panel of capacity S and a battery with infinite capacity.Each day has T time slots. Power generation and number of tasks arrived for different time slot (j) at each edge server (i) are given by S[i][j] and D[i][j] and S[i][j]¡=S. The values of S[i][j] and D[i][j] for all i and j are available at time 0. The tasks need to be executed in the same time slot and if the task executed it consumes a unit amount utilisation.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.

The objective of the problem is to maximise the total number of task executions for entire day.

2 Approach with dynamic programming

2.1 Brute force approach

In this approach, Each state of dynamic programming depends on server number, time slot number, energy's of all servers(including their battery's) and remaining number of tasks available to be executed in each time slot.Here we start at the first server in first time slot and choose to execute 0,1,2.. tasks on it.Then energy is updated and next server is called with recursive call, which also does same function. If the last server is reached, it will call first server of next time step.Thus this will continue and number of tasks executed is stored (along with tasks executed by servers after it and all servers in latter time steps) and maximum number of tasks executed is chosen among all possibilities.But this approach might lead to exponential time complexity, if the sub problems are not repeating(which was observed to be the case in some situations).

2.2 Main approach with dynamic programming

In this approach we sum all the energy's generated in each server across all time slots and sort based on this total generated energy. As its always beneficial to execute tasks on machines with less energy than on more energy server(for example to execute 4 tasks, two energy 8 servers may be used instead of one server with energy 64, this saves more energy which may be used in latter time steps to execute more tasks).

In each time step, store all the tasks arrived in a single array, as these tasks can be distributed across different servers in each time step.

Then for each server (in the sorted order), distribute the tasks across all time steps (according to the available number of tasks and energy in each time step).As energy utilised/available only depends on whether the server has executed tasks in previous time steps, consideration of other servers is not relevant. Dynamic programming approach can be used for this, to maximise the number of tasks

executed in each server across all time steps. Each state depends on Time step, Energy available(which is bounded by T*S). After completing dp for single server, tasks in each time step can be updated and same process can be continued. This will maximise the number of tasks executed in each server across all time steps.The subproblems can be found to be repeating in this case, thus dynamic programming helps.

3 Other approaches

3.1 Greedy with sorting the servers based on energy in each time slot

In this approach in each time step, we can greedily sort the servers based on their total available energy (i.e energy generated and already available energy from battery).As executing 2 tasks take 8 units of energy but execution of 3 tasks take 27 units of energy , it is beneficial to execute tasks with as less energy as possible, as this will spare some energy which can may be later used to execute more tasks.For example, there are 3 servers with energy 8,8,27 and number of tasks available are 2.Instead of executing task in server with 27 energy, we can execute 2 tasks in server with energy 8.This will make more energy available in next time step, thus useful to maximise the number of tasks.

But this heuristic might fail in some cases, where energy productions in servers, follow different order.For example consider 2 servers with 3 time slots, with energy's (8,27),(0,8),(0,0) and number of tasks of 4,3,1, We can check that the above heuristic fails to give maximum possible number of task executions.

3.2 Greedy with reverse sorting the servers based on energy in each time slot

Here also, we can similarly sort the servers based on available energy's but in decreasing order.Thus allocate available tasks in this manner.As, most tasks are executed in initial stages, this could be heuristic.

This heuristic also fails, in scenarios where saving the energy instead of executing tasks, might lead to more task executions in future time steps.For example consider 2 servers with 2 time slots, with energy's (27,8),(0,0).Tasks in each time step be 5,2.If we save energy in first time step by executing only 4 tasks, we can execute 2 more tasks in second time step, otherwise we can only execute 5 tasks.