EP2200 Project - Mobile Computation Offloading

Jan Hendrik Dahlhues — dahlhues@kth.se

February 23, 2025

1 Introduction

This project involves modeling two mobile cloud offloading systems. Initially, a theoretical analysis will be conducted to identify key parameters. This will be followed by a numerical evaluation to assess model performance.

2 System description

The systems consist of mobile users (MUs) connected to a access point (AP) that transmits computationally intensive tasks to a cloud computing infrastructure. Tasks can either be executed locally on the MU or offloaded to the cloud, which processes them in a first-come, first-served manner. Model 1 assumes negligible transmission delay, while Model 2 explicitly considers data transmission to the cloud.

Number of MUs	K
Task arrival rate	$\lambda_i = \lambda, \forall i \in \mathcal{K}$
Iput data size	$\sim exp(\mu_i^D)$, $\mu_i^D = \mu^D, \forall i \in \mathcal{K}$
Task complexity	$\sim exp(\mu_i^L)$, $\mu_i^L = \mu^L, \forall i \in \mathcal{K}$
MU computational capability	$F_i = F$, deterministic for $\forall i \in \mathcal{K}$
Cloud computational capability	F_0, deterministic
Number of servers in the cloud	m
Uplink rate	$R_i = R$, deterministic for $\forall i \in \mathcal{K}$
Probability to perform the computation locally	$p_i^l = p^l, \forall i \in \mathcal{K}$
Probability to offload the task to the cloud	$p_i^0 = p^0 = 1 - p^l, \forall i \in \mathcal{K}$

Figure 1: List of parameters.

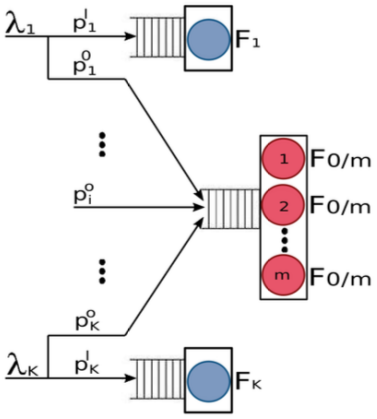

Figure 2: An example of Cloud-Offloading Model 1.

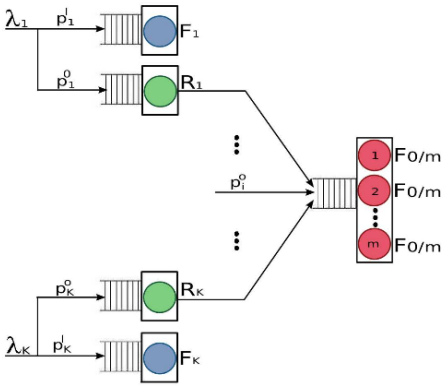

Figure 3: An example of Cloud-Offloading Model 2.

3 Theoretic Analysis

Following figure shows a state transition diagram for a system with only one MU for both: Model 1 and Model 2:

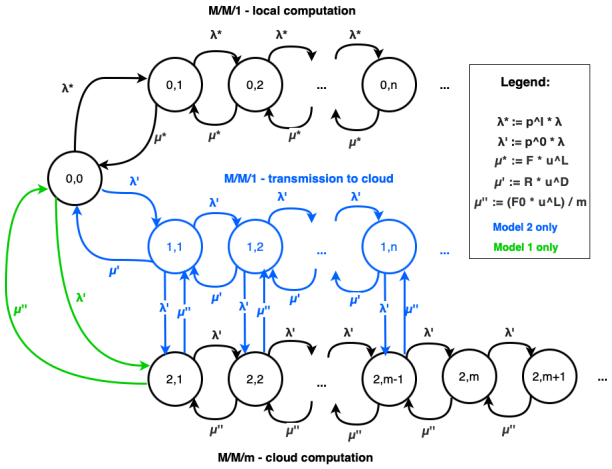

Figure 4: Model 1 & 2 - state transition diagram (only one MU)

3.1 Theoretic Analysis - Cloud-Offloading Model 1

3.1.1 Task execution and representation

For MUs, tasks arrive according to a Poisson process (M), have exponentially distributed service times (M), and are processed by a single server, therefore the Kendall notation for the task execution at the MU can be represented as M/M/1. The cloud has m servers and there is a single buffer at the cloud to store all incoming tasks. The task execution there can be represented as M/M/m. Since we have K MU's and one cloud system, the entire system can be represented in $K + 1$ dimensions.

3.1.2 Mean task execution times

The service intensity and task execution time are calculated as follows:
At MU i (M/M/1 system):

$$\mu_{MU} = F\mu^L \qquad \overline{x}_{MU} = \frac{1}{F\mu^L}$$

At cloud server (M/M/m system):

$$\mu_{cloud} = \frac{F_0\mu^L}{m} \qquad \overline{x}_{cloud} = \frac{1}{\frac{F_0\mu^L}{m}}$$

3.1.3 Stability conditions

For MU i: Task arrival rate * probability to perform the computation locally < task complexity * MU computational capability

$$\lambda p^l < \mu^L F$$

For cloud: Task arrival rate * probability to offload the task to the cloud < task complexity * cloud computational capability

$$\lambda(1-p^l)K < \mu^L F_0$$

For entire system: Task arrival rate * number of MUs < computational system capability * task complexity

$$\lambda K < p^l\mu^L F + (1-p^l)\mu^L F_0$$

3.1.4 Mean number of tasks $\overline{N}_i$ in the system that have been generated by MU i

For MU: In an M/M/1 queue, the number of tasks in the system follows a geometric distribution in steady state. To get the average task number in the system at the MU ($\overline{N}_{MU}$) , we also need to consider the computational capacity and the probability that the computation is performed locally:

$$\overline{N}_{MU} = \frac{\lambda_{MU}}{\mu_{MU} - \lambda_{MU}} \qquad \Rightarrow \qquad \overline{N}_{MU} = \frac{\lambda p^l}{F\mu^L - \lambda p^l}$$

For cloud: since the cloud is designed as an M/M/m system, we start calculating the average task number under service ($\overline{N}_{s,cloud}$) and in the queue ($\overline{N}_{q,cloud}$). We also consider the cloud offloading probability:

$$\overline{N}_{s,cloud} = \lambda_{cloud} * \overline{x}_{cloud} = \frac{\lambda_{cloud}}{\mu_{cloud}} = \frac{K\lambda(1-p^l)}{\frac{F0\mu^L}{m}} = a_{cloud}$$

$$\overline{N}_{q,cloud} = \sum_{k=m+1}^{\infty} (k-m)p^k = Dm(a_{cloud})\frac{a_{cloud}}{m - a_{cloud}}$$

where $Dm(a_{cloud})$ is given by the Erlang-C formula:

$$Dm(a_{cloud}) = P_{\text{wait}} = P(Q > 0) = \frac{\frac{(a_{cloud})^m/m!}{(1-(a_{cloud}/m)}}{\sum_{k=0}^{m-1} \frac{(a_{cloud})^k}{k!} + \frac{(a_{cloud})^m/m!}{1-(a_{cloud}/m)}}$$

Now we calculate the mean number of tasks in the cloud ($\overline{N}_{cloud}$):

$$\overline{N}_{cloud} = \overline{N}_{s,cloud} + \overline{N}_{q,cloud} = a_{cloud} + Dm(a_{cloud})\frac{a_{cloud}}{m - a_{cloud}}$$

The total mean number of tasks in the system ($\overline{N}$) is calculated as follows. To get the mean number of tasks $\overline{N}_i$ in the system that have been generated by MU i, we need to divide $\overline{N}$ by the amount of MU's (K):

$$\overline{N} = K * \overline{N}_{MU} + \overline{N}_{cloud} \qquad ; \qquad \overline{N}_i = \overline{N}_{MU} + \frac{\overline{N}_{cloud}}{K}$$

3.1.5 The mean time $\overline{T}_i$ that tasks generated by MU i spend in the system

Using Little's theorem: $\overline{N} = \lambda\overline{T}$

$$\Rightarrow \overline{T} = \frac{\overline{N}}{\lambda} = \frac{K * \overline{N}_{MU} + \overline{N}_{Cloud}}{\lambda} \qquad ; \qquad \overline{T}_i = \frac{\overline{N}_{MU} + (\overline{N}_{Cloud}/K)}{\lambda}$$

3.2 Theoretic Analysis - Cloud-Offloading Model 2

3.2.1 Task execution and representation

In Model 2, each MU is extended with a transmitter. The data transmission at the transmitter can be represented as M/M/1, since the input data size μ^D is exponentially distributed. The task execution notation at the MU and in the cloud is similar to Model 1 (for MU: M/M/1, for cloud: M/M/m). Since we have K MU's, K transmitter and one cloud system, the entire system can be represented in $2K + 1$ dimensions.

3.2.2 Differences between Model 1 and Model 2

Model 1 assumes instant data transmission, while Model 2 explicitly models transmission time. Also, Model 2 introduces queuing delay at the transmission phase.

3.2.3 Mean Task Execution and Transmission Time

At MU and cloud server: equal to Model 1
For transmission:

$$\mu_{TM} = R\mu^D \qquad\qquad \overline{x}_{TM} = \frac{1}{R\mu^D}$$

3.2.4 Stability conditions

MU and cloud: equal to Model 1
For transmission *i*:

$$\lambda(1-p^l) < R\mu^D$$

For entire system: Adding transmission to the system

$$\lambda K < p^l\mu^L F + (1-p^l)\mu^L F_0 + (1-p^l)R\mu^D$$

3.2.5 Mean number of $\overline{N}_i$ in the system that have been generated by MU *i*

For transmission: In an M/M/1 queue the number of tasks in the system follows a geometric distribution in steady state:

$$\overline{N}_{TM} = \frac{\lambda_{TM}}{\mu_{TM} - \lambda_{TM}} \qquad \Rightarrow \qquad \overline{N}_{TM} = \frac{\lambda(1-p^l)}{R\mu^D - \lambda(1-p^l)}$$

For the mean number of tasks in the system, we add the mean number of tasks at the transmitter:

$$\overline{N} = K * \overline{N}_{MU} + K * \overline{N}_{TM} + \overline{N}_{Cloud} \qquad ; \qquad \overline{N}_i = \overline{N}_{MU} + \overline{N}_{TM} + \frac{\overline{N}_{cloud}}{K}$$

3.2.6 The mean time $\overline{T}_i$ that tasks generated by MU *i* spend in the system

Using Little's theorem: $\overline{N} = \lambda\overline{T}$

$$\Rightarrow \overline{T} = \frac{\overline{N}}{\lambda} = \frac{K * \overline{N}_{MU} + K * \overline{N}_{TM} + \overline{N}_{Cloud}}{\lambda} \qquad ; \qquad \overline{T}_i = \frac{\overline{N}_{MU} + \overline{N}_{TM} + (\overline{N}_{Cloud}/K)}{\lambda}$$

3.3 Numerical Evaluation

3.3.1 Stability region

If $p^l = 1$(all tasks are executed locally), the execution node at the MU determines the maximum arrival rate. When $p^l = 0$ (all tasks are offloaded), the execution node at the cloud (in m Model 1) and the transmission node (in Model 2) determines the maximum arrival rate. In Model 1, the stability is primarily determined by the cloud execution node. The bottleneck shifts from the MU execution node (high p^l) to the cloud execution node (low p^l). In Model 2, the transmission node becomes a critical bottleneck.

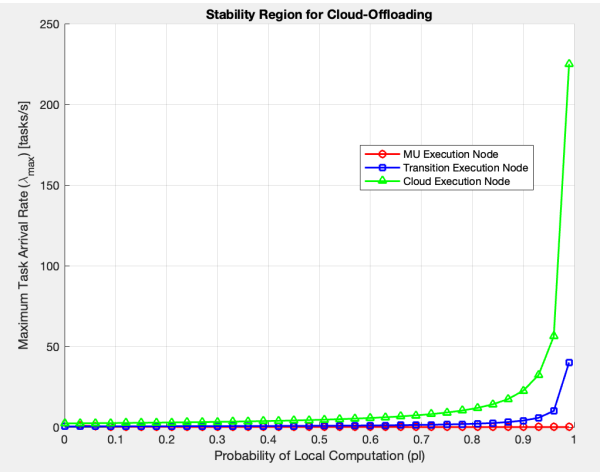

Figure 5: $\lambda_{max}(p^l)$ that ensures the stability of the node, as the function of p^l (steps: 0.03).

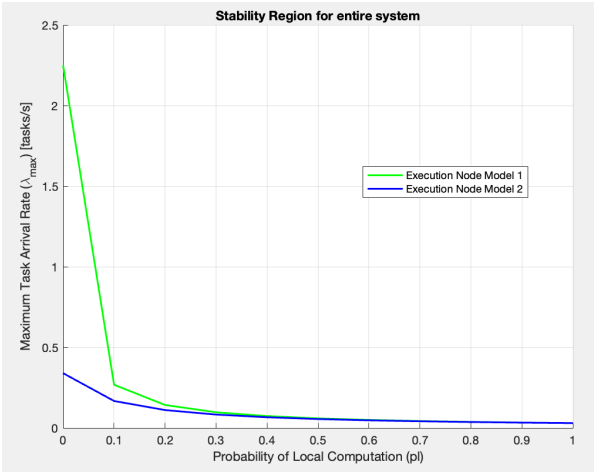

Figure 6: $\lambda_{max}(p^l)$ that ensures the stability of the entire system, as the function of p^l (steps: 0.1).

3.3.2 Mean performance metrics with respect to λ

The mean number of tasks and also the mean time spent in the system increase with λ. As the probability of local computation increases, the mean number of tasks and the mean system time grow at a faster rate. Since Model 2 also considers the transmission, the mean number and time of tasks are slightly higher in this Model. At specific points (plotting $\lambda = 0.5$ for $p^l = 0.7$ and plotting $\lambda = 1.5$ for $p^l = 0.2$), the system gets unstable since ρ_{MU} gets bigger than 1.

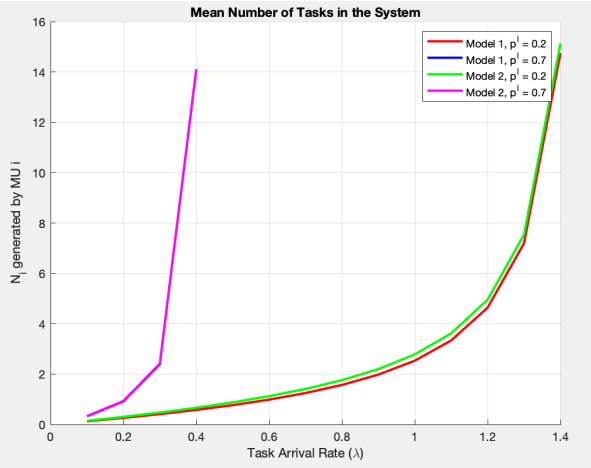

Figure 7: $\overline{N}_i$ generated by MU i, as a function of λ from interval [0.1,1.5] tasks/s (steps: 0.1)) $p^l \in (0.2, 0.7)$.

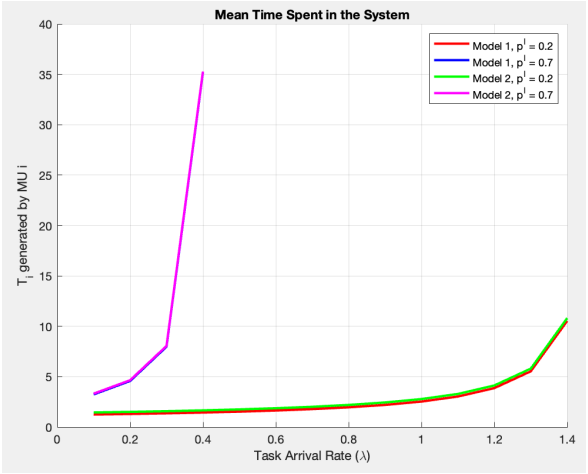

Figure 8: $\overline{T}_i$ generated by MU i, as a function of λ from interval [0.1,1.5] tasks/s (steps: 0.1)) $p^l \in (0.2, 0.7)$.

3.3.3 Mean performance metrics with respect to p_l

The mean system time T_i increases when more tasks are performed locally. If the number of MU's increases, the mean system time gets slightly higher. In general, the mean time in Model 2 is higher since it also consider the transmission times. At a specific point (while trying to plot $p^l = 0.3$), the system gets unstable ($\rho_{MU} > 1$). If there are 25 MU's, the system is already unstable ($\rho_{cloud} > 1$) at $p^l = 0$, so the curve for $K = 25$ cannot be plotted.

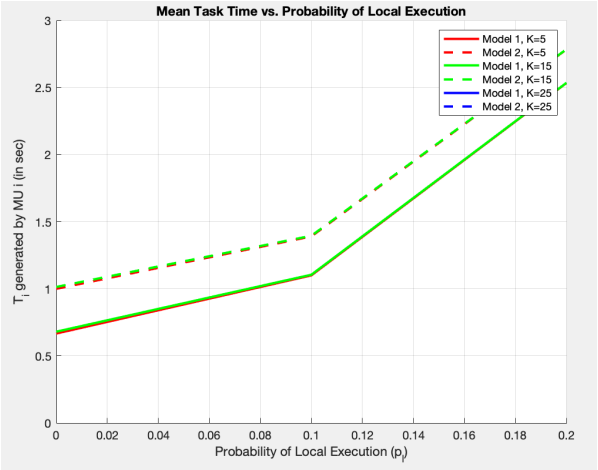

Figure 9: T_i generated by MU i as a function of p_l from interval [0,1] (steps: 0.1) to perform the computation locally