EP2200 Course Project – 2017
Project II - Mobile Computation Offloading

1 Introduction

Queuing theory provides us a very useful mathematic tool that can be used to analytically evaluate the performance of different applications in the area of telecommunications, computing, traffic engineering, etc.

Computationally intensive applications, including augmented reality, natural language processing, face, gesture and object recognition, and various forms of user profiling for recommendations are increasingly used on mobile handsets. Many of these applications consume a significant amount of energy, which can be detrimental to battery life. Moreover, due to the limited computational power of the mobile handset the response time may be too long for good user experience.

Mobile cloud computing is a promising approach to extend the battery lifetime of mobile handsets and to ensure short response times. Mobile cloud computing allows to offload the computations through a wireless access point to a cloud infrastructure that has significant computational power. The computations are then performed in the cloud and the results are sent back to the mobile handset.

2 System Description - Mobile Computation Offloading

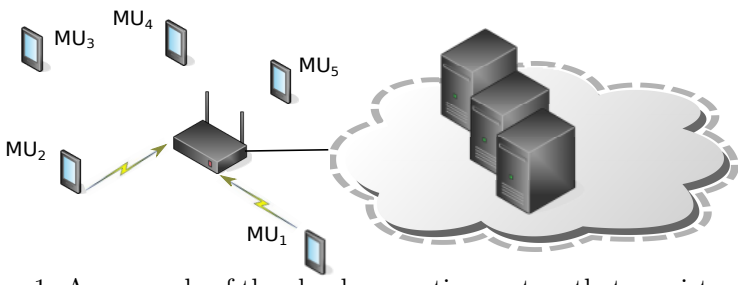

Figure 1: An example of the cloud computing system that consists of five MUs, one cloud and one AP.

2.1 System Definition

We consider a mobile cloud computing system as shown on Figure 1, consisting of mobile users (MUs) using mobile handsets, connected to a wireless access point (AP). MUs generate computationally intensive tasks, that can be executed locally at the MU, or can be offloaded to a cloud computing system. The cloud computing systems receives tasks from all the MUs, and processes them in arrival order. Once the computation is completed, the cloud sends back a short answer to the MU.

2.2 System Model

The cloud computing system consists of a set $\mathcal{K}=\{1, 2, ..., K\}$ of mobile users (MU) using mobile handsets and a cloud service.

Each MU generates computationally intensive tasks to perform. Tasks are characterized by the size D_i of the input data (e.g., in bytes), and by the number L_i of CPU cycles required to perform the computation. Furthermore, each MU is characterized by deterministic computational capability F_i and the computational capability of the cloud is also known and is denoted by F, where $F > F_i$.

If the MU decides to offload the computation to the cloud server, it has to transmit D_i amount of data pertaining to its task to the cloud server through the AP. The uplink rate R_i at which MU i transmits the input data of the task to the AP is given for each MU i. We assume that the AP and the cloud are connected by a high speed link and we do not model the time that is needed to transmit the data from the AP to the cloud. Moreover, results are transmitted back to the MUs with negligible delay.

We assume that the input data size D_i is exponentially distributed with mean $1/\mu_i^D$. Similarly, the number L_i of CPU cycles required to perform the computation is exponentially distributed with mean $1/\mu_i^L$. Note that the time required to perform a computation is given by the ratio of the required CPU cycles and the available computational capability.

We consider a homogeneous cloud computing system, in which all MUs are characterized by the same parameters (they have equal computational capability $F_i = F$, equal uplink rate $R_i = R$, equal mean data size $1/\mu_i^D = 1/\mu^D$ and mean task complexity $1/\mu_i^L = 1/\mu^L$). Furthermore, we assume that the tasks are generated at each MU $i \in \mathcal{K}$ according to a Poisson process with rate $\lambda_i = \lambda$.

Each MU $i \in \mathcal{K}$ performs a task locally with probability $p_i^l = p^l$, or offloads it to the cloud with probability $p_i^0 = 1 - p_i^l$. Again, we consider $p_i^l = p^l$. This is the parameter that the system can tune to optimize the performance.

We consider two different models of the cloud computing system: *Cloud-Offloading Model 1* and *Cloud-Offloading Model 2*.

2.3 Cloud-Offloading Model 1

To model the execution of the task, we consider that each MU i has one server that has a computational capability F_i, and infinite buffer with tasks served in FIFO order. The cloud has m servers, and each of the servers has a computational capability F_0/m. There is a single buffer at the cloud to store all incoming tasks, the buffer has infinite capacity and implements FIFO service

order. Furthermore, to simplify the model we assume that the time needed to transmit the input data to the cloud is negligible compared to the task execution time. Figure 2 shows resulting queuing model for *Cloud-Offloading Model 1*.

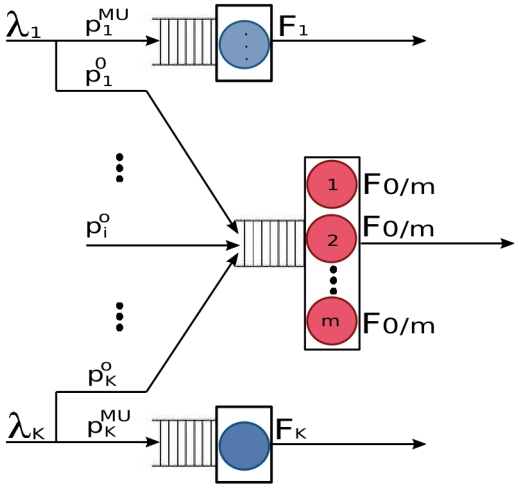

Figure 2: An example of *Cloud-Offloading Model 1*.

2.4 Cloud-Offloading Model 2

We model the execution of the tasks in the same way as in the case of *Cloud-Offloading Model 1*, but we also model the transmission of the input data to the cloud. We consider that each MU i has one transmitter that transmits data to the AP at a rate R_i, and we assume that the transmission buffer is infinite. Figure 3 shows the resulting queuing model for *Cloud-Offloading Model 2*.

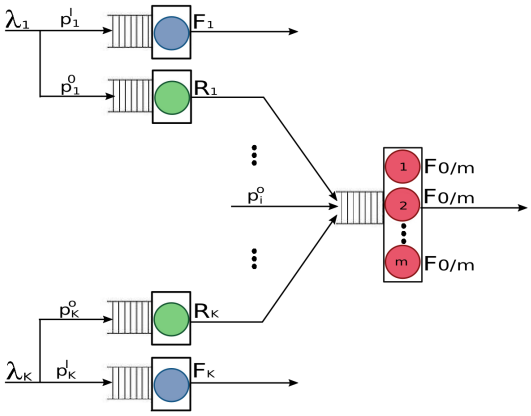

Figure 3: An example of *Cloud-Offloading Model 2*.

3 Performance Analysis

3.1 Theoretic Analysis

First you will need to derive analytic expressions that describe the performance of the cloud computing system, according to the two different models. We summarize the system parameters in Table 1.

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	$p_i^0 = p^0 = 1 - p^l$, $\forall i \in \mathcal{K}$

Table 1: List of parameters

3.1.1 Theoretic Analysis - Cloud-Offloading Model 1

Consider the queuing model for *Cloud-Offloading Model 1* shown in Fig. 2. Give the Kendall notation for the task execution at the MU and at the cloud.

Please derive:

- the mean task execution time at MU i and at the cloud server,
- the stability conditions for MU i and for the cloud, and the stability condition for the entire system,
- the mean number of tasks $\bar{N}_i$ in the system (at the MU or in the cloud) that have been generated by MU i,
- the mean time $\bar{T}_i$ that tasks generated by MU i spend in the system.

3.1.2 Theoretic Analysis - Cloud-Offloading Model 2

Consider the queuing model for *Cloud-Offloading Model 2* shown in Fig. 3. Give the Kendall notation for the task execution and for the data transmission at the MU and for the task execution in the cloud.

Please derive:

- the mean task execution time at MU i and at the cloud server, and the mean input data transmission time to the AP,
- the stability conditions for MU i and for the cloud, and the stability condition for the entire system,
- the mean number of tasks $\bar{N}_i$ in the system that are generated by MU i,
- the mean time $\bar{T}_i$ that tasks generated by MU i spend in the system.

3.2 Numerical Evaluation

Consider a homogeneous computing system that consists of $K = 10$ MUs and one cloud with computational capability $F_0 = 45 \cdot 10^9$ *Cycles/s* and $m = 15$ servers. The mean data sizes and the mean task complexities are the same for all MUs and equal to $1/\mu^D = 10^6$ *bits* and $1/\mu^L = 0.5 \cdot 10^9$ *Cycles/s*, respectively. The computational capability of each MU is $F = 0.6 \cdot 10^9$ *Cycles/s* and each MU transmits the data to the AP at a rate $R = 4 \cdot 10^6$ *bit/s*.

3.2.1 Stability region

Plot the maximum task arrival rate $\lambda_{max}(p^l)$ for the three queuing nodes (execution node at the MU, transmission node at the MU and execution node at the cloud) that ensures the stability of the node, as the function of p^l, for $p^l = [0, 1]$. Plot the three curves in the same figure.

Next, for both system models, plot the maximum task arrival rate $\lambda_{max}(p^l)$ that ensures the stability of the entire system, as the function of p^l, for $p^l = [0, 1]$ with a step 0.1.

Which queuing node determines the maximum task arrival rate $\lambda_{max}(p^l)$ of the entire system as p^l increases? Compare the two system models.

3.2.2 Mean performance metrics with respect to λ

Plot $\bar{N}_i$, the mean number of tasks in the system that have been generated by MU i, as a function of the task arrival rate λ, where λ takes values from the interval $[0.1, 1.5]$ *tasks/s* with a step of 0.1. Plot the curves in the same figure for two values of $p^l \in \{0.2, 0.7\}$ and for both system models (in total four curves in the figure).

With the same set of parameters, plot $\bar{T}_i$, the mean time that the tasks generated by MU i spend in the system. Plot the curves in the same figure for two values of $p^l \in \{0.2, 0.7\}$ and for both system models (in total four curves in the figure).

Compare the two system models. How do the mean number of tasks $\bar{N}_i$ and the mean system time $\bar{T}_i$ change as λ increases? What happens when p^l increases? Give an intuitive explanation.

3.2.3 Mean performance metrics with respect to p^l

Plot for $\lambda = 1$ *task/s* the mean time $\bar{T}_i$ that the tasks generated by MU i spend in the system, as a function of probability p^l to perform the computation locally, where p^l takes the value from the interval $[0, 1]$ with a step of 0.1. Plot the curves in the same figure for three values of $K \in \{5, 15, 25\}$ and for both system models (in total six curves in the figure).

Compare the two system models. How does the mean system time $\bar{T}_i$ change as p^l increases? Discuss the results for different values of K.

4 Waiting Time Distribution - Extra Exercise for the Best

Until now we were working with mean performance metrics. Now we will look at the waiting time distribution experienced by the tasks in the cloud computing system.

Please derive the distribution of the waiting time of the tasks generated by MU i, that is, $F_W(t)$ in the case of *Cloud-Offloading Model 1*.

For the numerical analysis consider the same set of parameters as in Section 3.2.

Plot for $\lambda = 1$ *task/s* the CDF $F_W(t)$ of the waiting time as a function of $t = [0.1s, 10.1s]$ with a step $1s$ for three different values of probability $p^l \in \{0.2, 0.4, 0.7\}$. What is the probability that the waiting time W is less than 2.1 s for the different values of local computation probabilities? Based on the results, discuss which queuing node impacts the most the waiting time.

Next, plot for $p^l = 0.4$ the CDF $F_W(t)$ of the waiting time as a function of $t = [0.1s, 10.1s]$ with a step $1s$ for three different values of arrival intensity $\lambda \in \{0.1, 0.8, 1.5\}$. What is the probability that the mean waiting time W is less than 2.1 s for three different intensity values? How does the arrival intensity λ impact the waiting time?

5 Submission Instructions and Grading

- The submission deadline is given on the course webpage. You need to submit the project report through the course web, see Assignments (or Inlämningsuppgifter in Swedish). If you can not do this (you do not have kth e-mail), send the project report to **vfodor at kth.se**.
- You are allowed to solve the problem in groups, however, you have to prepare a project report on your own. Reports including the same text will be disqualified.
- You need to submit a report of a maximum of 4 printed pages. The report needs to contain a description of the solution of the problems, including detailed derivation of the formulas that used to calculate the results, and discussions on the insight from your results.
- You need to use software tools to get the results, we propose Matlab, but we accept all solutions, e.g., you can even program everything in C. You should not include your codes in the report.
- Check the grammar of the report. There are good tools available to do that. Make sure that performance graphs are possible to interpret. Give the dimensions and units of the axes.

Grading: pass or fail. To pass the moment, you need to show that you made a serious attempt to solve the problems. The best 20% of the submissions receive 5 extra points on the exam. Extra points will not be considered at the make-up exam or at later exams.

Would you have any questions, contact Viktoria at **vfodor at kth.se**.