Throughput Maximization in Mobile Cloudlets

Supervisor:
Associate Professor Weifa Liang
Qiufen Xia

Yan Wang
U5155825

COMP 8790 Software Engineering Project
Research School of Computer Science
Semester 1, 2013
Abstract

In mobile cloud computing (MCC), cloud service providers offer provide small-scale cloudlets which are in close quarters for mobile users to easy access their rich resources. Hence, there is no need to send most requests of mobile users to the cloud; instead, they can be processed locally. Even though the cloudlet is flexible, its shortcomings of limited resources and low processing abilities become the bottleneck. Therefore, how to make full use of the limited resources becomes a hot topic. In this report, we consider the online request admission control issue in a cloudlet and an optimization problem to maximize the system throughput. We will focus on a novel admission cost model which to present the consumptions of limited resource, and an Markov chain prediction mechanism to predict the dynamic occupation of the system based on an efficient control algorithms for online request admissions. Finally, we conduct experiments to evaluate the performance of the proposed algorithms. The numerical results are presented to illustrate the proposed algorithms have a good performance on the resources occupation prediction.
Acknowledgements

First of all, I would like to express my very great appreciation to my supervisor Professor Weifa Liang for his valuable and constructive suggestions during the planning and development of this research work. I would never complete my study and research work without his high level knowledge in academic areas, patient guidance, and encouragement and help when I suffered in the dilemma. I would also like to thank Ms Qiufen Xia, for her advice and assistance in leading me into research project. My grateful thanks are also extended to Professor Peter Christen for his kind consideration to keep my progress on schedule.

I would like to extend my thanks to all the stuff in Research School of Computer Science that I could have this wonderful and meaningful research experience in this project work.

Special thanks to my friends, Weiwei Hou, Yongxuan Fu, Jingjing Tan and Hao Tian for their generous help during the project.

Last but not least, I would like thank my parents, Mr.Dianzhu Wang and Mrs.Xiaoyun Wu for their support and encouragement throughout my study.
Chapter 1  Introduction

1.1 Overview

The latest projection from the UN’s International Telecommunications Union says that global mobile subscriptions will “outnumber people” during 2014, going beyond 7 billion\(^1\), and one fourth of which is using mobile internet\(^2\). Smart phones, tablets are becoming the essential part of diversified life of a huge population. As they are not only for making calls and sending texts anymore, people create their social life, enjoy the various entertainment apps, and even manage their business. However, mobile devices are not efficient as computers, especially on the storage, memory and complex computational calculation. In this case, the mobile cloud computing (MCC) platform is established to fill these gaps. Users can pay the providers for the online service in the MCC, for more storage or complicated tasks.

MCC provides a wide variety of service through the wireless communication, such as Wifi, 3G and so on, which bring out a another problem—unstable communication transmission. Hence, the cloudlet, a small scale of cloud with less but enough resources, offers well-connected and rusted service to the nearby mobile users. The best situation for the users is to achieve real-time interactive response and get the missions done rapidly. Unfortunately, due to the limited resources and in the cloudlet, not all the request from the users can be accepted. Here comes the challenge, which should be admitted or rejected.

1.2 Objectives

In this report, I will focus on improve the current algorithms for the online request admission problem and predict the occupation rates of the limited resources in cloudlets with Markov chain model. My objective is to maximize the throughput of the access point of the cloudlet and make sure the system can maintain a good performance for the users. The resource requested from users, like bandwidth, storage, memory, CPU time, as well as occupation duration, will be considered in the algorithms. With the constrains of the resources, the novel admission cost model with different priorities will be proposed. And the algorithms will be experimented through the simulation to evaluate the performance.

1.3 Structure

The report is organized in 6 chapters as follows. Chapter 2 introduces the related
works and literatures, as well as Chapter 3 focuses on the system model and problem definitions. In Chapter 4, we propose the approved online request admission algorithms with Markov chain prediction, followed by the performance evaluation of the proposed algorithms conducted in Chapter 5. The final conclusion and future work are presented in Chapter 6.
Chapter 2    Background

2.1 Mobile Cloud Computing

As a major application model in the era of the Internet, Cloud Computing has become a significant research topic of the scientific and industrial communities since 2007. Commonly, cloud computing is described as a range of services which are provided by an Internet-based cluster system. Such cluster systems consist of a group of low-cost servers or Personal Computers (PCs), organizing the various resources of the computers according to a certain management strategy, and offering safe, reliable, fast, convenient and transparent services such as data storage, accessing and computing to clients.

The Mobile Cloud Computing (MCC) term was introduced after the concept of Cloud Computing. Basically, MCC refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device\(^3\). Regarding the definition, mobile applications move the computation power and storage from the mobile phones to the cloud.

It can be thought as a combination of the cloud computing and mobile environment. The cloud can be used for power and storage, as mobile devices don’t have powerful resources compared to traditional computation devices.

Today, there are already lots of good examples of MCC applications including Gmail and Google Maps.

2.2 Related work

Quality of Service (QoS) is a key issue in mobile cloud computing (MCC), and the admission control policy plays an essential part of it. However, QoS requirements are difficult to satisfy because of the high variability of Internet workloads. Hence, an appropriate admission control policy that can provide stable and efficient MCC environment for limited resources is crucial. In recent years, great research efforts have been spent on this hot topic.

Many existing works in literature focused on developing various admission control policies and resource allocation strategies. For example, Almeida et al.\(^4\) presents a self-managing technique that jointly addresses the resource allocation and admission control optimization problems in virtualized servers. Their solution is
designed taking into account the provider's revenues, the cost of resource utilization, and customers' quality of service (QoS) requirements, specified in terms of the response time of individual requests. Therefore, the solutions can satisfy QoS constraints while still yielding a significant gain in terms of profits for the provider, especially under high workload conditions. Hoang et al. \(^5\) formulate an optimization problem for dynamic resource sharing of mobile users in MCC hotspot with a cloudlet as a semi-Markov decision process (SMDP), which is a linear programming model. In the optimization model, QoS for different classes of mobile user is taken into account under resource constraints (i.e., bandwidth and server). Liang et al. \(^6\) formulate the resource allocation problem as a SMDP to capture the dynamics of user arrivals and departures, and an optimal decision is made to maximize the overall system rewards by striking the balance between the network utilities and costs of network resources.

However, the studies we mentioned above put much effort on the CPU resources and don't pay much attention to the other resources, like memory, storage and bandwidth. Even though with the fact that CPU intensive is the most common situation when dealing with the requests, because of other resources are not so “popular” as CPU is, but they also play an important role in the MCC system.

Markov model has been applied in many discipline research, especially Markov chain which is a mathematical system that undergoes transitions from one state to another, between a finite or countable number of possible states. For illustration, Xing et al. \(^7\) propose Hybrid-order tree-like Markov models to predict Web access precisely while providing high coverage and scalability. Oly et al. \(^8\) evaluate Markov models to represent the spatial patterns of I/O requests in scientific codes, and proposes three algorithms for I/O prefetching. Mostly Markov model is used to predict prefetching, it is also of great importance to predict the state of system's resource occupations, with which we can promote the performance and the use efficiency of the system.

We conclude from the former work, and propose a Markov chain model to predict the MCC system resources occupations. The aim of the work is to improve the proposed admission control algorithms that can efficiently admit requests according to the prediction system conditions as well as maximize the system throughput.
Chapter 3 Preliminaries

In this section, we first introduce the system model of mobile cloudlet computing and declare the limited resources. Then the problems of how to deal with the requests from various mobile users will be defined in the following part.

3.1 System model

A cloudlet is a new architectural element that arises from the convergence of mobile computing and cloud computing. It represents the middle tier of a 3-tier hierarchy: mobile device -- cloudlet -- cloud. A cloudlet is designed to bring the cloud closer with features of powerful, well-connected and safe.

A mobile cloudlet computing environment should consist of a set of local wireless mobile users and an abundant resources cloudlet as shown in figure 1. The cloudlet is also connected to a remote and more powerful cloud platform by the internet. Denote by \( \{u_i \mid 1 \leq i \leq N\} \) the set of local mobile users, where \( N \) is the number of mobile users. Most mobile users offload their computation work or large volume of data storage demand to the cloudlet, because mobile devices are not powerful as large-scale computers which have high-speed computing, large storage space and plenty of memory. When mobile users demand the resources of the cloudlet, they send requests in terms of amounts of resources they need to the cloudlet. Then, when the requests pass through the access point, as shown in Figure 1, the cloudlet makes decision whether to accept or reject the requests. Not all the requests can be accepted when the competition for the resources is fierce, because admission cost and ability to remain functions are taken consideration in cloudlet.
More specifically, we assume that time we deal with requests is not continuous, but slotted into several equal time slots and requests are conducted at the beginning of each time slot \( t \). Otherwise; the system will be buried in the process of the dealing with requests when the traffic is heavy. We further assume that system has no knowledge on the future requests arrival rates, and the prediction only focuses on the system resource occupation rate.

We assume that the cloudlet provides \( K \) different resources. \( C_k \) represents the capacity of the resource \( k \), with \( 1 \leq k \leq K \). Let \( H(t) = \langle H_1(t), ..., H_K(t) \rangle \) be the amounts of resources occupied by the admitted requests at time slot \( t \). For the proposed algorithms, assume that \( H(t) \) is given, and then we use Markov chain model to predict the amounts of occupied resources to replace \( H(t) \). Donate by \( A(t) = \langle A_1(t), ..., A_K(t) \rangle \), is the available amounts of resources in the cloudlet at the time slot \( t \), with \( A_k(t) = C_k - H_k(t) \) for all the \( 1 \leq k \leq K \). Let \( r_i(t) = \langle r_{i,1}(t), ..., r_{i,k}(t); \tau_i \rangle \) be the amounts of resources demanded by the request \( r_i(t) \) at the time slot \( t \). Specifically, \( r_{i,k}(t) \) represents the amounts of resource \( k \) that the request demand and \( \tau_i \) is the occupation period of the requested resources. \( \langle r_1, ..., r_N \rangle \) is the sequence of the requests arriving at the access point of cloudlet and \( N \) denote the total number of the requests.

There will be two scenarios of the requests arriving. Firstly, we assume that requests arrive one by one at each time slot, which means that in one time slot only one request is evaluated. The admission strategy is that we not only consider whether the available resources will meet the demand of the requests but also the
admission cost that the system will pay for admitting the request. Secondly, there are more than one request arrive at the access point during one time slot. We treat them together at the beginning of each time slot as a batch problem.

3.2 Markov Models

3.2.1 Markov Chain

A Markov chain is a mathematical system that undergoes transitions from one state to another, between a finite or countable number of possible states\(^\text{10}\). In Markov chain, the next state depends only on the current state and not on the sequence of events that preceded it. The term "Markov chain" refers to the sequence of states such a process moves through. The term “transitions” means the changes of state of the system, and the probabilities associated with various state-changes are called transition probabilities. A transition matrix describes the probabilities of the transitions that happen between the states. Particularly, all the states of the process should be included in the transition matrix; otherwise, the chain will get stuck at some point (state) because that it cannot transit to the next state. Since the system changes randomly, it is almost impossible to predict the actual future state at a given point or state. However, what matters is that statistical properties of the system's future can be predicted. In many applications, it is these statistical properties that are important.

Let’s give an example on Markov chain. As we can see in Figure 2, it is a hypothetical stock market with states of bull market, bear market, and stagnant market. According to the Figure 2, it is a 90% probability that a bull week is followed by another bull week, 7.5% by a bear wee, and 2.5% by a stagnant week. We conclude the transition matrix for this example in Table 1.
Figure 2: state transitions of the hypothetical stock market

<table>
<thead>
<tr>
<th></th>
<th>Bull M</th>
<th>Bear M</th>
<th>Stagnant M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bull M</td>
<td>0.9</td>
<td>0.075</td>
<td>0.025</td>
</tr>
<tr>
<td>Bear M</td>
<td>0.15</td>
<td>0.8</td>
<td>0.05</td>
</tr>
<tr>
<td>Stagnant M</td>
<td>0.25</td>
<td>0.25</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1: Markov transition matrix for hypothetical stock market

From Table 2, we can get the transition matrix:

\[
P = \begin{bmatrix}
0.9 & 0.075 & 0.025 \\
0.15 & 0.8 & 0.05 \\
0.25 & 0.25 & 0.5
\end{bmatrix}
\]

Let \( \pi(n) \) present the market state at \( n \). Hence the next state is \( \pi(n + 1) \) with \( \pi(n + 1) = \pi(n)P \). So we can conclude that the distribution of stock market at time \( n + 1 \) given the distribution for time \( n \).

\[
\pi(n + 1) = \pi(n) * P = (\pi(n - 1) * P) * P = \pi(1) * P^n
\]

From this formula, we can calculate whichever state we want with a given start state. Let’s give an enumeration for the stock market states with the start \([0,1,0]\)(start with a bear market).

\[
\pi(2) = \pi(0) * P^2 = \begin{bmatrix}
0 \\ 1 \\ 0
\end{bmatrix}
\begin{bmatrix}
0.9 & 0.075 & 0.025 \\
0.15 & 0.8 & 0.05 \\
0.25 & 0.25 & 0.5
\end{bmatrix}^2
= \begin{bmatrix}
0.2675 \\ 0.6637 \\ 0.0687
\end{bmatrix}
\]

\[
\pi(3) = [0.3575 \ 0.5682 \ 0.0743]
\]
\[
\pi(10) = [0.625 \ 0.3125 \ 0.0625]
\]
\[
\pi(n) = [0.625 \ 0.3125 \ 0.0625] \ (n > 10)
\]

After 10 times transitions, the steady-state probabilities indicate that 62.5% of weeks will be in a bull market, 31.25% of weeks will be in a bear market and 6.25% of weeks will be stagnant.

What happens when we start from a bull market, the steady-state probabilities are the same. Independent of the start state, the Markov process converges to a stationary distribution. Therefore, no matter what we begin with, the probability of states in system will stay the same level, as we mentioned before “statistical properties of the system's future”.

8
3.2.2 Markov prediction model

To capture current state of occupied resources in the cloudlet, we have chosen the Markov chain model to model occupation information of all the resources. A Markov model with $N$ states can be described by a transition matrix $P$. Each value $P_{ij}$ of this $N$ by $N$ Matrix represents the probability of a transition to state $j$ from the current state $i$. Because the transition probabilities depend only on the current state, the transition path to the current state does not affect future state transitions; this characteristic is known as the Markov property.\(^{12}\)

Given the online cloudlet resources, as we chosen before – CPU, Memory, Storage, Bandwidth, we create a transition matrix for each one of them. $N$ that presents $N$ states for the resources occupation will depend on the active request unit size. For example, the storage we have in the cloudlet is 100GB, and the most common requests reach the access point of the cloudlet is asking for 30MB, then the states number $N$ may be set between [100,300].

Consider one illustrative example. In the Table 2, vertical and horizontal scale represent the current state and the next state. Assume that system has 9 states. Once a state transit from one to another, the value in the corresponding cell will plus one. All the empty cells will be filled in with 0. Then the most possible state after the current state will be calculated by the formula below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: states transition matrix

Let $S'$ be the most possible next state after current state $S$, and $s = \{ s_1, ..., s_n \}$ be the all the possible state. $p = \{ p_1, ..., p_n \}$ is the counts for each state denote that how many times the transition happened between $S \rightarrow s_i$. \[ \]
\[ S' = \frac{\sum_{i=1}^{n} P_i s_i}{\sum_{i=1}^{n} p_i} \]  

(1)

### 3.3 Problem definitions

We can treat the online request admission problem as a simple question, whether to accept or reject the requests that arrive at the access point of cloudlet. Mobile user \( u_i \) sends its request \( r_i(t) = (r_{i,1}(t), ..., r_{i,K}(t); \tau_i) \) to the cloudlet to demand a special amount of resources at time slot \( t \). Specifically, \( r_{i,k}(t) \) represents the amounts of resource \( k \) that the request demand and \( \tau_i \) is the occupation period of the requested resources. We assume the requests arrive one by one at this stage, and the occupied resources are already known by the system.

The online request throughput maximization problem is to make the system throughput maximized during a specified time period \( T \). The core of the problem is whether to admit or reject the users’ requests to maximize the throughput. What is the system throughput? The system throughput is percentage that admitted requests from all the requests that arrive at the cloudlet. If a user request is admitted by the cloudlet, it will be implemented and the resources demanded will be occupied until the request is finished. Otherwise, the request will be rejected immediately in the current time slot and user can send the request again in the future. There is no relevance between the current request and former admission conditions.

The online batch request throughput maximization problem is to maximize the system throughput when dealing with multiple requests with one time slot.

The system throughput maximization problem is an efficiency-cost trade-off problem. The cloudlet should maintain a stable environment for computing while admit users’ request as much as possible.
Chapter 4  Algorithms

4.1 Structure of algorithms

In this section, we will focus on the algorithms for the online request throughput maximization problem. Figure 3 below shows the flow diagram of the basic frame of the algorithms.

We assume that \( r_i(t) = \langle r_{i,1}(t), ..., r_{i,K}(t); \tau_i \rangle \) is the request we deal with right now, and \( r_{i,k}(t) \) denote the resource \( k \) that the request demands and \( \tau_i \) is the occupation period of the requested resources. First, we should check if the resources that \( r_i(t) \) demand are available in the cloudlet. If the resources are not available, even only one of them, the request will be rejected immediately. Second,
check if the cost of the request is beyond the threshold we set for the cloudlet. If so, reject the request, otherwise, admit. The cost model will be introduced in the following part.

4.2 Admission cost modeling

As a request may demand for several resources in the cloudlet, we give the cost model for resource $k$ as example. The admission cost is a convex function of the quantity of the resource, for which when user demand large amount of resource, the cost will increase exponentially. When the system of cloudlet is running under almost full load, the cost will be extremely high, because that system will take the risk of losing efficient operating ability. The unit admission cost of using resource $k$ with demand $r_{i,k}(t)$ by request $r_i(t)$ is defined as follows.

$$
\zeta(r_{i,k}(t), H_k(t)) = a_k \frac{H_k(t)}{c_k} \left( a_k \frac{r_{i,k}(t)}{c_k} - 1 \right)
$$

where $a_k > 1$ is a constant and $H_k(t)$ is the amount of resource $k$ that has already been occupied in the cloudlet by the requests admitted before. $C_k$ is the capacity of resource $k$ in the cloudlet.

From Eq. (2), $\frac{H_k(t)}{c_k}$ and $\frac{r_{i,k}(t)}{c_k}$ are both smaller than 1, which gives us that the range of $\zeta(r_{i,k}(t), H_k(t))$ is $(0, a_k - 1]$. We can see that when the system performs at a high load, which means $H_k(t)$ is bigger, the admission cost is higher. When the amounts $r_{i,k}(t)$ which request demands for resource $k$ increase, the admission cost will also increase.

Lots of work has been done for evaluate influence of different values of $a_k$ on the system throughput. For both algorithms we will introduce later, when $a = 4$, the system throughout gets a peak. Hence, we will use $a = 4$ for all the experiments in this report.

The admission cost for a request has a close relation with the occupation period $\tau_i$ for all resources. $\gamma(r_{i,k}(t), H(t))$ presents the admission cost for $r_i(t)$ at time slot $t$ as in Eq. (3)

$$
\gamma(r_{i,k}(t), H(t)) = \tau_i \cdot \sum_{k=1}^{K} \zeta(r_{i,k}(t), H_k(t) | r_{i,k}(t) \neq 0) = \tau_i \cdot \sum_{k=1}^{K} a_k \frac{H_k(t)}{c_k} \left( a_k \frac{r_{i,k}(t)}{c_k} - 1 \right)
$$
4.3 Admission policy

As we mentioned in the structure of algorithms, there is a threshold for the admission cost to maintain the high-quality performance of the system. That is, a threshold $B_k$ of the unit admission cost of using resource $k$ is given for each request $r_i(t)$. As we talked in the last part, the range of a unit admission cost is $(0, a_k - 1]$, thus the range of $B_k$ must be $(0, a_k - 1]$. Thus, a request $r_i(t)$ will be admitted if it meets the following constrains:

$\begin{align*}
(1) & \quad \zeta(r_{ik}(t), H_k(t)) \leq B_k \\
(2) & \quad \gamma(r_{i}(t), H(t)) \leq B 
\end{align*}$

Here comes a system throughput and resources occupied upper bound trade-off problem. If $B_k$ is smaller, which means the threshold is lower, fewer requests will be admitted by the cloudlet and the system throughput will decrease. However, if $B_k$ is bigger, more requests with larger resources demand will be admitted by the system, which directly lead to a reduction of the system throughput. Experiments are done to find the optimal result for $B_k$ and $B$. The system throughput achieve optimal situation with $B_k = B = 0.9$, while $a = 4$. Therefore, we will use these as our default setting in the performance evaluation.

4.4 Algorithm

4.4.1 Online request throughput maximization problem

As we mentioned in the algorithm structure part, the core of the problem is to decide whether to admit or reject users’ requests. In the Algorithm 1 described below, we assume that there is only one request arriving at the access point of the cloudlet per time slot. With each requests, $K$ resources are demanded which makes the online request throughput maximization problem a K-dimensional bin packing problem. In the meantime, K dimensions, like memory requirement and CPU occupation, can be reduced to one dimension bin packing problem by introducing the admission cost. We treat different attributes from different dimensions as different terms, and calculate the admission cost for each required resource. The Algorithm 1 is also referred to as Algorithm Online-OBO with predicted occupied resources.
Particularly, we use the prediction states, which are also called occupation resources, to replace the assumptive known occupation resources. Hence, the system get the current state of occupation resources from the prediction matrix based on the former state, instead of get the current occupation information from the cloudlet.

Algorithm 1 Algorithm for Online request throughput maximization problem with predicted occupied resources

**Input:** $B_k, B$, an arrival request $r_i(t)$, the prediction matrix for $H(t)$ of resources at time slot $t$, where $r_i(t) = (r_{i,1}(t), \ldots, r_{i,K}(t); \tau_i)$ and $1 \leq k \leq K$.

**Output:** Admit or reject request $r_i(t)$

1: $\gamma(r_i(t), H(t)) \leftarrow 0$; // The admission cost of request $r_i(t)$
2: get predicted occupation information $H(t)$ from prediction matrix based on the former state $S_n$ of resource; // apply prediction strategy to system
3: for each $r_{i,k}(t)$ in $r_i(t)$ do
4: Calculate $A_k(t) \leftarrow C_k - H(t)$ // available amount of resource $k$
5: if $r_{i,k}(t) > A_k(t)$ then
6: Reject request $r_i(t)$; EXIT;
7: else
8: if $r_{i,k}(t) \neq 0$ then
9: Calculate the cost $\zeta(r_{i,k}(t), H_k(t))$ by Eq. (2);
10: if $\zeta(r_{i,k}(t), H_k(t)) \leq B_k$ then
11: $\gamma(r_i(t), H(t)) \leftarrow \gamma(r_i(t), H(t)) + \tau_i \cdot \zeta(r_{i,k}(t), H_k(t))$;
12: else
13: Reject request $r_i(t)$; EXIT;
14: end if;
15: end if;
16: end if;
17: end for;
18: if $\frac{\gamma(r_i(t), H(t))}{|r_i(t)|} \leq B$ then
19: /* $S_{n+1}$ will be the former state for next time slot */;
Replace the former state $S_n$ with current $S_{n+1}$;
20: return Admit request $r_i(t)$;
21: else
22: Reject request $r_i(t)$; EXIT;
23: end if;
4.4.2 Online batch request throughput maximization problem

In this part, we will focus on the multiple request admissions at each time slot. The requests will arrive at the access point of cloudlet at any time, but we conduct them at the beginning of each time slot in as a batch process. Let $\Delta S(t)$ be the set of requests arrive at time slot $t$. After the decision-making by the admission control policy, a subset $\Delta S'(t) \subseteq \Delta S(t)$ will be admitted by the cloudlet. $\Delta S'(t)$ can be $\emptyset$, if all the requests are rejected by the system.

The Algorithm 2 described below is an elevated algorithm compared with Algorithm 1, by using a greedy strategy to solve the multiple request situations. The process of the admission control policy is as follows. $\Delta S(t)$ is the set of requests arriving at time slot $t$, and the system available resources $A(t)$ is calculated from the predicted occupied resources $H(t)$. The Algorithm 2 is also referred to as Algorithm Online-Batch with predicted occupied resources.

Initially, $\Delta S'(t) = \emptyset$, which means there is no admitted request. Then, for each request from $\Delta S(t)$, calculate its unit admission cost the same as Algorithm 1 using Eq. (2) and Eq. (3). If the request cannot meet the criteria, it is rejected immediately and removed from $\Delta S(t)$. Otherwise, it becomes a candidate of $\Delta S(t)$. When all the requests conducted, select the request with the minimum admission cost and put it into $\Delta S(t)$. If there are more than one request share the with the minimum admission cost, we pick the one with smallest occupation period $\tau_i$ and put it into $\Delta S'(t)$. Then let $\Delta S(t) = \Delta S(t) - \Delta S'(t)$ be the set of request for a new round of calculation. Before that, the system available resources are updated with the new occupation information from the newly admitted request. For example, we take $r_n$ as the newly admitted request, and $1 \leq n \leq |\Delta S(t)|$. System will update the new available resources after admitting one request by $A'(t) = (A_1(t) - r_{n,1}(t), ..., A_K(t) - r_{n,K}(t))$. Then, a new round of picking admitted request begins with the newly updated $\Delta S(t)$ and $A'(t)$. The procedure continues until $\Delta S'(t) = \emptyset$. At the end of algorithm, we will get a subset $\Delta S'(t) \subseteq \Delta S(t)$, which is the set of admitted request at the time slot $t$. $\Delta S'(t)$ can be $\emptyset$, if all the requests are rejected by the system.

<table>
<thead>
<tr>
<th>Algorithm 2 Algorithm for Online batch request throughput maximization problem with prediction occupied resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A set of requests $\Delta S(t)$, $B_k, B$, the prediction matrix for $H(t)$ of resources at time slot $t$.</td>
</tr>
</tbody>
</table>
Output: A subset of admitted requests $\Delta S'(t) \subseteq \Delta S(t)$

1: get predicted occupation information $H(t)$ from prediction matrix based on the former state $S_n$ of resource; // apply prediction strategy to system
2: $\Delta S; \Delta S'(t) \leftarrow \emptyset; H'(t) \leftarrow H(t);$  
3: while $U \neq \emptyset$ do
4: $\text{min \_cost} \leftarrow \infty; // A variable indicating the minimum admission cost
5: /* A variable indicating the index of the request with the minimum admission cost */
     $i_0 \leftarrow \infty;$
6: for each request $r_i(t)$ in $U$ do
7: /* The admission cost by processing request $r_i(t)$ */
8: $\gamma(r_i(t), H'(t)) \leftarrow 0;$
9: for each $r_{i,k}(t)$ in $r_i(t)$ do
10: Calculate $A'_k(t) \leftarrow C_k - H'_k(t)$ // available amount of resource $k$
11: if $r_{i,k}(t) > A'_k(t)$ then
12: $U \leftarrow U - \{r_i(t)\};$ Reject request $r_i(t);$  
13: else
14: Calculate the unit admission cost
     $\zeta(r_{i,k}(t), H_k(t))$ of $r_i(t)$ by Eq. (2);  
15: if $\zeta(r_{i,k}(t), H_k(t)) > B_k$ then
16: $U \leftarrow U - \{r_i(t)\};$ Reject request $r_i(t);$  
17: else
18: $\gamma(r_i(t), H'(t)) \leftarrow \gamma(r_i(t), H'(t)) + \tau_i \cdot \zeta(r_{i,k}(t), H_k(t));$
19: end if;
20: end if;
21: end for;
22: if $\frac{\gamma(r_i(t), H'(t))}{\|r_i(t)\| \tau_i} > B$ then
23: $U \leftarrow U - \{r_i(t)\};$ Reject request $r_i(t);$  
24: end if;
25: if $\frac{\gamma(r_i(t), H'(t))}{\|r_i(t)\| \tau_i} < \text{min \_cost}$ then
26: $\text{min \_cost} \leftarrow \frac{\gamma(r_i(t), H'(t))}{\|r_i(t)\| \tau_i};$
27: $i_0 \leftarrow i;$
28: else
29: Select the request $r_{i'}(t)$ with a smaller occupation period between
the two requests, i.e., \( \min_{\text{cost}} \left\{ \gamma(r_i(t), H'(t)) \middle| r_i(t) \right\} \).

30: \( i_0 \leftarrow i' \);
31: \) end if;
32: \) end if;
33: \) end for

34: \( \Delta S'(t) \leftarrow \Delta S'(t) \cup \{r_{i_0}(t)\} \) where \( r_{i_0}(t) \) has the minimum admission cost;
35: \( U \leftarrow U - \{r_i(t)\} \);
36: Update the amounts of occupied resources \( H'(t) \) by taking the occupied resources by \( r_i(t) \) into \( H'(t) \);
37: \) end while
38: /* \( S_{n+1} \) will be the former state for next time slot */;
39: \) return \( \Delta S'(t) \subseteq \Delta S(t) \).

### 4.5 Training for Markov chain model

As we mentioned in Chapter 3, to capture current state of occupied resources in the cloudlet, we have chosen the Markov chain model to model occupation information of all the resources. In order to construct the state transition matrix, we pick the training samples for Markov chain model from the system real-time occupation information. The selectivity point we choose is the end of the time slot after the “finished” requests release their resources back to the cloudlet system. There are two reasons to choose this point to collect data:

- If the time point is at the beginning of the time slot, the occupation information about the system resources is updated after this time slot’s admit-reject decision. The occupation amounts of the resource are over estimated, and it will lead a decrease in system throughput because of the high occupation of illusion.
- The available information of the resource, which calculated from the resources capacity and resources occupation, is most representative.
Chapter 5    Performance Evaluation

In this chapter, we evaluate the performance of the Markov chain prediction in the proposed algorithms and investigate the impact of the training data size and the scale of the state division on the algorithm performance.

5.1 Simulation environment

We assume that a mobile cloudlet environment that consists of \( n \) servers\(^{15} \). The cloudlet provides four resources for users to demand: CPU, memory, disk storage and bandwidth. The capacities of resources are shown in Table 3. Particularly, bandwidth capacity of the wireless access point is for the cloudlet, and other three resources capacity are for each server.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2.99GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>8 GB</td>
</tr>
<tr>
<td>Storage</td>
<td>1024 GB</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>75 Mbps</td>
</tr>
</tbody>
</table>

Table 3: capacities of resources in cloudlet\(^{16} \)

As we mentioned in the Chapter 4: admission policy, we assume that \( B_k = B = 0.9 \) and \( a = 4 \) as default settings. In the previous work done on the evaluation of system performance, these parameters help the system achieve the maximized throughput both for algorithms Online-OBO and Online-Batch without prediction resources occupation.

We assume that each time slot lasts 10 seconds\(^{17} \) and the default system monitoring time period is \( T = 8,000 \) time slots. We further assume that, for online request throughput maximization problem, there is only one request arrives at the access point of cloudlet at each time slot. For online batch request throughout maximization problem, the amounts of requests arrives at each time slot is between the range of \([2, 10]\). All the requirements of resources are generated randomly within special ranges according to Amazon Small Instance\(^{18} \) settings. The requirements for resources are listed below in Table 4.

<table>
<thead>
<tr>
<th>Type</th>
<th>Algorithm-OBO</th>
<th>Algorithm-Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU power(GHz)</td>
<td>([1,6])</td>
<td>([1,6])</td>
</tr>
<tr>
<td>Memory(MB)</td>
<td>([1,400])</td>
<td>([1,400])</td>
</tr>
</tbody>
</table>

18
### Table 4: Parameters of requests

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage (GB)</td>
<td>[0.06,1.2]</td>
</tr>
<tr>
<td>Bandwidth (Mbps)</td>
<td>[0.05,1.5]</td>
</tr>
<tr>
<td>Maximum occupation period (time slots)</td>
<td>20</td>
</tr>
</tbody>
</table>

#### 5.2 Markov chain prediction performance evaluation

In this part, we first evaluate the performance of the proposed Online-OBO with predicted occupied resources and the ideal Online-OBO under different monitoring periods. From Figure 4, we can see that the ideal Online-OBO achieves a very high system throughput during all the monitoring periods. For example, the system throughput of Online-OBO is around 10% higher than the Online-OBO with predicted occupied resources. Both algorithms have good stabilization. The reason behind is that we set strict constraints to the requests, and the requests with large amounts of demand for resources are rejected to maintain the high throughput of the system. What’s more, the request arrives at access point of cloudlet one by one releasing the pressure of the system computing. The Algorithm Online-OBO with predicted occupied resources cannot capture all the occupation information exactly as the real system situation. The state transition matrix of resource occupation can only predict the most possible state after the current state.

![Figure 4: The system throughput of Online-OBO and with predicted of occupied resources](image-url)
Now, we apply the Markov chain model prediction to the online batch request throughput maximization problem. From Figure 5 we can see that system throughput of Online-Batch is around 80%, which is almost 15% higher than the Online-OBO with predicted occupied resources. The trends of two lines, represent the system throughput are not so similar, because they are not tested with the same random input requests. Hence, we focus on the average of the system throughput of the two algorithms.

Figure 5: The system throughput of Online-Batch and with predicted of occupied resources

We now study the impact of the training data size on the system throughput. As shown in Figure 6, \( T \) denote the time period that training data set covers, which is 24,48,96 hours. We can conclude that the larger the training data set, the more possible that prediction approximates the real conditions. However, the large training sample don not make a big difference on improving the throughput of the system. The possible reason may be that the random requests we conduct for the testing are similar, even after a long period of time. That makes the strengths of large training sample less remarkable.
Figure 6: The impact of training data size on the system throughput of Online-OBO with predicted occupied resources.
Chapter 6 Conclusions and future work

In this report, we introduce the Markov chain model into the prediction of occupation information for cloudlet system. Combined with the proposed algorithms, we can see the occupation resources prediction are representative and have applicability in the mobile cloudlet system throughput maximization problem.

The training samples we use in the experiment are the random requests we conduct are similar, even after a long period of time. Hence, there are a lot of states for occupied resources never happened in the experimental observations. What we want to do is to optimize the state transition matrix to be more pragmatizing.

What also came to my mind is that we can improve the Markov chain model from state-to-state to state-to-multistate. The one-to-multi structure may represent the actual state transition better.


