Fall 2017

Cache Management and Load Balancing for 5G Cloud Radio Access Networks

Chin Tsai
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Computer Sciences Commons

Recommended Citation
DOI: https://doi.org/10.31979/etd.sgq8-ctcm
https://scholarworks.sjsu.edu/etd_projects/562

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.
Cache Management and Load Balancing for 5G Cloud Radio Access Networks

Chin Tsai

San Jose State University

Fall 2017
CACHE MANAGEMENT AND LOAD BALANCING FOR 5G CLOUD RADIO ACCESS NETWORKS

A writing project

Presented to

The Faculty of Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Chin Tsai

December 2017
The Designated Thesis Committee Approves the Thesis Titled

CACHE MANAGEMENT AND LOAD BALANCING FOR 5G CLOUD RADIO ACCESS NETWORKS

by

Chin Tsai

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2017

Dr. Melody Moh Department of Computer Science
Dr. Teng Moh Department of Computer Science
Dr. Thomas Austin Department of Computer Science
ABSTRACT

CACHE MANAGEMENT AND LOAD BALANCING FOR 5G CLOUD RADIO ACCESS NETWORKS

by Chin Tsai

Cloud radio access network (CRAN) has been proposed for 5G mobile networks. The benefit of a CRAN includes better scalability, flexibility, and performance. The paper introduces a cache management algorithm for a baseband unit of CRAN and load balancing algorithms for virtual machines load within the CRAN. The proposed scheme, exponential decay (EXD) with analytical hierarchy process (AHP), increases hit rate and reduces network traffic. The scheme also provides preferential services for users with a higher service level agreement (SLA). Finally, the experiment shows the proposed load balancing algorithm can reduce the virtual machines’ (VM) queue size and wait time.
# TABLE OF CONTENTS

LIST OF ACRONYMS ........................................................................................................ vi
LIST OF TABLES ............................................................................................................. vii
LIST OF FIGURES ......................................................................................................... viii
1. Introduction .................................................................................................................. 1
   1.1 Background ............................................................................................................. 1
   1.2 Project Overview ................................................................................................... 2
   1.3 Related Work ........................................................................................................ 4
2. Cache Management ..................................................................................................... 5
   2.1 General Cache Management Algorithm ............................................................. 6
   2.2 EXD Scoring Function ......................................................................................... 8
   2.3 EXD Scoring Function with AHP ................................................................. 9
   2.4 Experiment Setup for the Cache Management Algorithm ............................ 11
   2.5 Result - Cloud Writes and Network Traffic ............................................... 12
   2.6 Result – Cache Hit Rates for Various Service Levels ................................... 15
3. Load Balancing .......................................................................................................... 18
   3.1 Load Balancing Algorithms ............................................................................... 19
   3.2 Experiment Setup for the Load Balancing Algorithms ............................... 20
   3.3 Result ................................................................................................................ 21
4. Conclusion ................................................................................................................ 26
5. References ................................................................................................................. 27
LIST OF ACRONYMS

5G - 5th Generation
ACS - Ant Colony System
BBU - Baseband Unit
CRAN - Cloud Radio Access Network
EXD - Exponential Decay
EXD-AHP - Exponential Decay with Analytical Hierarchical Process
FLC - Fuzzy Logic Controller
FRLS - Fuzzy Rule-based Reinforced Learning System
GOL - Generic Online Learning
IoT - Internet of Things
LFU - Least Frequently Used
RAN - Radio Access Network
RL - Reinforcement Learning
RRH - Remote Radio Head
SDN - Software-Defined Networking
SLA - Service Level Agreement
UE - User Equipment
UEC - User Equipment Context
VM - Virtual Machine
LIST OF TABLES

Table 1. AHP Matrix .............................................................................................................. 10
Table 2. Experiment Parameter Values .................................................................................. 11
Table 3: Simulated Algorithms .............................................................................................. 12
Table 4. 5G UE events break down ...................................................................................... 19
Table 5. Eight LB Algorithms for CRAN .............................................................................. 19
Table 6. Experiment Parameter Values .................................................................................. 20
Table 7. Average and standard deviation of queue size for each UEC .................................. 23
Table 8. Average and standard deviation of total service time .............................................. 23
LIST OF FIGURES

Figure 1. RAN (left) and CRAN (right) [1] ................................................................. 2
Figure 2. Project Overview ......................................................................................... 3
Figure 3. Cache Management Algorithm ..................................................................... 6
Figure 4: Total writes & network traffic, cache size 1250 MB .................................. 13
Figure 5: Total writes and network traffic, cache size of 2500 MB ......................... 14
Figure 6: Total writes and network traffic, cache size of 3750 MB ......................... 14
Figure 7: Total writes & network traffic, cache size 5000 MB ............................... 15
Figure 8: Hit rates of different service levels with cache size of 1250 MB ............ 16
Figure 9: Hit rates of different service levels with cache size of 2500 MB ............. 17
Figure 10: Hit rates of different service levels with cache size of 3750 MB ............ 17
Figure 11: Hit rates of different service levels with cache size of 5000 MB .......... 18
Figure 12. PDF of queue size ..................................................................................... 21
Figure 13. PDF of Total Service Time ..................................................................... 22
Figure 14. Queue size vs Occurrence in CDF ......................................................... 24
Figure 15. Total Service Time vs Occurrence in CDF ............................................. 24
Figure 16. Box plot of queue size ............................................................................ 25
Figure 17. Box plot of total service time ................................................................. 26
1. Introduction

1.1 Background

The emergence of new video encoding technologies, content-centric communication, and Internet of Things (IoT) has rapidly increased the demand on higher capacity cellular network. 5th Generation (5G) addresses the challenge by introducing new technologies. Cloud Radio Access Network (CRAN) is one of those technologies that provides centralized computation, scalability, and resource management to support many devices all at once [1].

A Radio Access Network (RAN), which provides a connection between user devices and core mobile networks, consists of evolved Node B (eNodeB) and user equipment (UE). A Traditional eNodeB consists of a remote radio head (RRH) and a baseband unit (BBU) with a RRH on each eNodeB as shown on the left side of Figure 1. The RRH’s job is to transmit and receive wireless signal as well as to amplify signal for transmission. A BBU is responsible for transforming IP packets into digital baseband signal and processing baseband signal from the RRH [2]. In a CRAN, baseband units of eNodeBs are pooled together as shown on the right side of Figure 1. The pool is made up of virtual machines (VM) to process user requests; this reduces power consumption, increases scalability, and reduces delay [1]. An example of these benefits would be a LTE handover. A LTE handover occurs when a user moves from one eNodeB to another. Its purpose is to transfer a user equipment context (UEC), which holds subscription
information of the user, to the current eNodeB where the user is located. In a traditional LTE network, UECs are stored in a local baseband unit of an eNodeB; thus, it takes time to transfer UECs between eNodeB. In a CRAN many handover steps are now internal processes of the BBU pool [3]; this significantly reduces handover latency. Furthermore, having an active user’s UEC in a cache is very important to maintain acceptable user experience as the UEC is required for many LTE procedures.

![Figure 1. RAN (left) and CRAN (right) [1]](image)

### 1.2 Project Overview

Since the memory size of a BBU pool is limited, it is important to provide a memory management scheme to improve the mobile users’ experience. This paper provides a cache management algorithm for a cache memory in a CRAN baseband unit pool. The characteristics of this cache algorithm include preferential eviction and a reduced cache miss penalty. Three scoring functions are implemented and are used for cache eviction; they are least frequently used (LFU), exponential decay (EXD) [5], and exponential decay with analytical hierarchical process (EXD-AHP) [6]. To spread the load of VMs
while processing requests, some load balancing algorithms are introduced to reduce queue size and total service time of VMs.

The overview of the project is shown in Figure 2.

In this figure, the requests are coming from layer 2 control applications and are sent to a BBU pool to be processed by VMs. Each request needs to go through a load balancer to be assigned to an optimal VM. The VM would process the user request and store the user’s UEC in its cache. If all caches are filled, the UEC is stored to a secondary cloud storage.
1.3 Related Work

There are many works on dealing with cache performance. Floratou et al. [5] introduced a cache algorithm for database applications. In this algorithm, all files in a cache were sorted according to their scores in descending order, and the first few lowest score files were evicted to make room for more important files. Each file’s score was determined by scoring functions such as exponential decay or the least recently used method. Finally, the paper also introduced an adapter algorithm which monitored the hit rate of the cache. The algorithm adjusted the parameters of the scoring functions based on the observations to increase the hit performance.

Podlipnig et al. [7] compiled a list of common cache replacement strategies. LFU with aging factor replacement algorithm was used as a base line comparison for the scoring functions presented in this paper.

Many load balancing schemes have been introduced for 5G and LTE networks. Gomes et al. [8] discussed a content migration technique between edge caches located in eNodeBs. In their scheme, a specified controller predicted mobile users’ movements and made decisions on migrating content to a new node. If a decision was made, analytic hierarchy process (AHP) was used to determine the best edge node for content migration. Finally, the authors pre-determined what content to migrate using content popularity. The authors successfully demonstrated the technique can reduce download latency and can increase hit rate at edge caches. Munoz et al. [9] used a fuzzy logic controller (FLC)
combined with a fuzzy rule-based reinforced learning system (FRLS) to solve the congestion problem for a femtocell, a small low-power cellular base station, in an office environment. Hisham et al. [10] provided a load balancing algorithm for micro and small cells; the algorithm increased the throughput for user equipment (UE) and reduced the up-link signal to noise ratio. Shahriari and Moh [11] applied generic online learning (GOL) based on reinforcement learning (RL) to a CRAN; the experiment demonstrated that the algorithm reduced cache misses and reduced communication load.

Many studies on load balancing for software-defined networking (SDN) were also introduced. Koushika and Selvi [12] combined their heuristic algorithm with a SDN controller to provide path and server load balancing within a network. Taking advantage of the OpenFlow controller, Zhang and Guo [13] used a dynamic load balancing algorithm for server clustering which distributed load to an optimal server. Using OpenDayLight, Sathyanarayana and Moh [14] adopted Ant Colony System (ACS) as their load balancing algorithm to achieve better network performance and resource utilization.

2. Cache Management

The proposed cache management algorithm would address 2 concerns: how to keep most requested UECs in a cache and how to keep UECs with a high SLA level in the cache. To address the first concern, the algorithm assigns a score to each UEC by using a scoring function. The second concern is addressed by using a weight calculated from
AHP. This section introduces the general cache management algorithm that utilizes the scoring function.

2.1 General Cache Management Algorithm

Four cache management algorithms are tested; however, the difference is only on their cache scoring function used. Figure 3 below is the flow chart of the general cache management algorithm which uses a scoring function to update UEC score.

![Figure 3. Cache Management Algorithm](image-url)
The detailed algorithm is also shown below. Note that even if a newly arrived $UEC_i$ has a score that is lower than the lowest score of a UEC in a cache, the UEC in the cache is still evicted to make room for the requested UEC. This is because the newly arrived $UEC_i$ belongs to an active user, and keeping an active user’s UEC in a cache is very important to maintain an acceptable user experience.

---

**Cache Management Algorithm**

1. For each request from user device with $UEC_i$
2. Calculate new score of $UEC_i$ using Equation (2);
3. Update the score of every entry in the cache using Equation (1);
4. if cache hit on one of the VMs then
5. update both content and score of the in-cache $UEC_i$ with those of this newly arrived $UEC_i$;
6. return;
7. else cache miss
8. write $UEC_i$ to the cloud storage;
9. select a VM using Round-Robin algorithm;
10. if the VM has cache space then
11. insert $UEC_i$ in VM’s cache;
12. return;
13. else no cache space, compare UEC scores
14. if score of $UEC_i$ greater than the min. score
15. evict from the cache the first lowest $E$ entries whose sum of scores is just lower than score of $UEC_i$;
16. write the evicted $E$ UEC’s to cloud storage;
17. else UEC, score lower than the min. score
18. evict minimum-scored UEC from cache;
19. write the evicted UEC to cloud storage;
20. insert $UEC_i$ in VM’s cache;
21. return;
2.2 EXD Scoring Function

The core of the cache management algorithm is the EXD scoring function. The function is used to keep most requested UECs in a cache and to keep less frequently requested UECs out of the cache. The EXD scoring function pair is listed below.

\[ S_i(u_{i1} + \Delta u) = S_i(u_{i1}) * e^{-a\Delta u} \quad (1) \]

\[ S_i(u_{i1} + \Delta u) = S_i(u_{i1}) * e^{-a\Delta u} + 1 \quad (2) \]

In EXD, each UEC’s score is determined by the time between requests. For both equations, \( S_i(u) \) represents the score of UEC \( i \) at time \( u \). Equation 1 is used to reduce the score of UECs currently in the cache that are not requested while Equation 2 is used to calculate the score of a requested UEC. Note that the amount of reduction from Equation 1 is determined by \( e^{-a\Delta u} \) where \( \Delta u \) is the time elapsed since a UEC \( i \) is last requested at time \( u_{i1} \). The longer the \( \Delta u \), the higher the score reduction. The effectiveness of the \( \Delta u \) is determined by the value ‘a’. A small ‘a’ mean the exponent term has little effect on \( S_i(u_{i1}) \), and vice versa. Equation 2 shares the same term, but the additional constant 1 term allows a UEC to have a higher score if it is frequently requested. The overall effect is the most requested UECs get higher scores while the least requested UECs get lower scores.
2.3 EXD Scoring Function with AHP

The EXD scoring function is modified to take the user SLA level into account. The motive behind this modification is to increase the hit rate of UECs with higher SLAs while preventing UECs with lower SLAs from getting higher scores. There are 4 SLA levels in this project; they are L1, L2, L3, and L4 where L1 is the best SLA level. The first step of this process is to set up a matrix as shown in Table 1. Each row in the matrix represents importance of a certain SLA level compare to other levels. For example, L1 is 5 times more important than L2, and L3 is 5 times less important than L1; the value chosen is arbitrary. Once the matrix is created, the weight vector can be calculated with the following steps:

1. Convert fractions to decimals
2. Square the result matrix
3. Sum up the rows of the matrix and get a vector
4. Normalize the result vector by dividing it with the sum of all elements in the matrix
5. Repeat steps 2 to 4 until the result no longer changes from the previous iteration

[7]
The result on step 5 is represented by the far-right column of Table 1. Each element of this vector represents the weight of the individual SLA level. These values are used to modify the existing EXD scoring function. The modification is shown in Equation 3 where $W_{AHP}$ is the individual SLA level calculated by AHP and is appended to the end of Equation 1. We also come up with Equation 4 where the constant 1 in Equation 2 is replaced by $W_{AHP}$.

<table>
<thead>
<tr>
<th></th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>Weight $W_{AHP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1</td>
<td>5/1</td>
<td>5/1</td>
<td>5/1</td>
<td>0.579</td>
</tr>
<tr>
<td>L2</td>
<td>1/5</td>
<td>1</td>
<td>5/1</td>
<td>5/1</td>
<td>0.281</td>
</tr>
<tr>
<td>L3</td>
<td>1/5</td>
<td>1/5</td>
<td>1</td>
<td>5/1</td>
<td>0.102</td>
</tr>
<tr>
<td>L4</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
<td>1</td>
<td>0.043</td>
</tr>
</tbody>
</table>

$$S_i(u_{i1} + \Delta u) = S_i(u_{i1}) \ast e^{-a\Delta u} + 1 + W_{AHP} \quad (3)$$

$$S_i(u_{i1} + \Delta u) = S_i(u_{i1}) \ast e^{-a\Delta u} + W_{AHP} \quad (4)$$
2.4 Experiment Setup for the Cache Management Algorithm

CloudSim [15] is used for the experiment. 1 host and 4 VMs are created to simulate a BBU pool in a CRAN. In the experiment, 84,000 requests are sent to the host. The percentage of the SLA level within those requests are 52 percent for L1, 26 percent for L2, 13 percent for L3, and the rest are L4. The time between the request is modeled by equation -ln(u)/λ where u is a value between 0 and 1 and λ is the number of requests per second. In our experiment, λ is set to 1,400 requests per second. The detailed experiment parameters are shown in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of VM</td>
<td>4</td>
</tr>
<tr>
<td>VM cache size</td>
<td>1,250MB, 2,500MB, 3,750MB, and 5,000MB</td>
</tr>
<tr>
<td>λ, mean UEC arrival rate</td>
<td>1,400 UEC/second</td>
</tr>
<tr>
<td>UEC record size</td>
<td>200 KB</td>
</tr>
<tr>
<td>No. of distinct users</td>
<td>25,000</td>
</tr>
<tr>
<td>QoS levels</td>
<td>L1, L2, L3, and L4</td>
</tr>
<tr>
<td>𝑊_{AHP}</td>
<td>L1: 0.58; L2: 0.28; L3: 0.10; L4: 0.04.</td>
</tr>
<tr>
<td>QoS traffic distribution</td>
<td>L1: 52%; L2: 26%; L3: 13%; L4: 9%</td>
</tr>
<tr>
<td>EXD Parameter a</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>Total simulation time</td>
<td>5 minutes</td>
</tr>
</tbody>
</table>

Four types of weight functions are used during the experiment. The LFU scoring function is used as a baseline comparison for the proposed scoring functions.
Table 3: Simulated Algorithms

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Cache Management Algorithms</th>
<th>UEC Scoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFU</td>
<td>Least Frequently Used</td>
<td>UEC request frequency</td>
</tr>
<tr>
<td>EXD [5]</td>
<td>Based on Exponential Delay</td>
<td>Equations (2) and (1)</td>
</tr>
<tr>
<td>EXD-AHP+1</td>
<td>Enhancing EXD with AHP</td>
<td>Equations (3) and (1)</td>
</tr>
<tr>
<td>(newly proposed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXD-AHP</td>
<td>Enhancing EXD with AHP</td>
<td>Equations (4) and (1)</td>
</tr>
<tr>
<td>(newly proposed)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.5 Result - Cloud Writes and Network Traffic

This section evaluates the 4 scoring functions with respect to the number of cloud writes and the amount of network traffic where the network traffic is calculated using Equation 5 (Note that 200 kilobytes is converted to bits first). Figure 4 to Figure 7 show the results with cache sizes of 1,250 MB, 2,500 MB, 3,750 MB, and 5,000 MB.

\[
Network\ Traffic = \frac{Total\ Writes \times 200 \times 8 \times 1000}{Simulation\ Time} \quad (5)
\]

Each figure shows that of the proposed scoring functions, EXD-AHP, can achieve the smallest number of cloud writes and network traffic. In Figure 4, the network traffic for
EXD-AHP is about 250 Mbps less than that of LFU. In Figure 5, the saving for the same comparison is about 70 Mbps. The saving for the subsequent figures are 20 Mbps and 10 Mbps respectively. From this result, it is clear as cache size increases, the amount of network traffic reduced by EXD-AHP becomes less significant. This is because the larger the cache size, more UECs can be kept in the cache. The result is a higher hit rate and less network traffic.

Figure 4: Total writes & network traffic, cache size 1250 MB
Figure 5: Total writes and network traffic, cache size of 2500 MB

Figure 6: Total writes and network traffic, cache size of 3750 MB
2.6 Result – Cache Hit Rates for Various Service Levels

This section shows how well the 4 scoring functions provide different levels of support for different service levels (L1, L2, L3, and L4) in terms of cache hit rates. The cache hit rate for each level is defined in Equation 6.

\[
L_i \text{ Cache Hit Rate} = \frac{\text{Total no. of } L_i \text{ UEC cache hits}}{\text{Total no. of } L_i \text{ UEC arrivals}}
\]  \hspace{1cm} (6)
Figures 8 to 11 show the hit rates for cache sizes of 1,250 MB, 2,500 MB, 3,750 MB and 5,000 MB respectively. When the cache size is very small only L1 receives a good hit rate as shown in Figure 8. This is due to the small cache size. Once the cache size starts to increase, the hit rates of the various service levels start to show up.

While Figure 10 and Figure 11 show clear distinction between the L1 and L2 hit rates for each scoring function, there is no clear distinction for the L3 and L4 hit rates except for EXD-AHP. In some cases, the hit rate for L4 is higher than that of L3. However, EXD-AHP shows the clear distinction between the 2 service levels.

![Figure 8: Hit rates of different service levels with cache size of 1250 MB](image-url)
Figure 9: Hit rates of different service levels with cache size of 2500 MB

Figure 10: Hit rates of different service levels with cache size of 3750 MB
Figure 11: Hit rates of different service levels with cache size of 5000 MB

3. Load Balancing

Since user requests are processed by VMs in a BBU pool, a load balancing scheme is required to reduce queue size and total service time. Total service time is defined as the total time a UEC spent in a VM. This includes the time the UEC needs to wait before being processed plus the UEC processing time. There are many different UE events in 5G as shown in Table 4. Note that each event has its own relative CPU load and arrival rate. The total events arrival rate is 1,400 events per second.
Table 4. 5G UE events break down

<table>
<thead>
<tr>
<th>UE Events</th>
<th>Event arrival rate</th>
<th>Relative CPU Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>UE state transitions</td>
<td>750</td>
<td>1.00</td>
</tr>
<tr>
<td>Handovers</td>
<td>100</td>
<td>0.82</td>
</tr>
<tr>
<td>Tracking area updates</td>
<td>30</td>
<td>1.24</td>
</tr>
<tr>
<td>(TAU)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paging</td>
<td>500</td>
<td>0.26</td>
</tr>
<tr>
<td>Attach/detach</td>
<td>25</td>
<td>2.31</td>
</tr>
<tr>
<td>Actual Event Total</td>
<td>1400</td>
<td></td>
</tr>
</tbody>
</table>

3.1 Load Balancing Algorithms

Eight load balancing algorithms are tested. Table 5 shows the description of each load balancing algorithm. The algorithms are listed in ascending order of the computation complexity.

Table 5. Eight LB Algorithms for CRAN

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Five Basic LB Algorithms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(listed in ascending order of complexity)</td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td>Round-robin</td>
<td>Round-robin of VM assignment.</td>
</tr>
<tr>
<td>RND</td>
<td>Random</td>
<td>Random VM assignment.</td>
</tr>
<tr>
<td>CPU</td>
<td>Based on CPU load</td>
<td>Assign to VM with min. cumulative CPU load.</td>
</tr>
<tr>
<td>Squeue</td>
<td>Based on shortest queue size</td>
<td>Assign to VM with shortest queue length.</td>
</tr>
</tbody>
</table>
20

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of VM</td>
<td>4</td>
</tr>
<tr>
<td>VM cache size</td>
<td>2,500MB</td>
</tr>
<tr>
<td>UEC record size</td>
<td>200 KB</td>
</tr>
<tr>
<td>UEC arrival process</td>
<td>Poisson</td>
</tr>
<tr>
<td>$\lambda$, mean UEC arrival rate</td>
<td>1,400 UEC/second</td>
</tr>
<tr>
<td>Total no. of distinct users</td>
<td>25,000</td>
</tr>
<tr>
<td>Total simulation time</td>
<td>5 minutes</td>
</tr>
</tbody>
</table>
3.3 Result

Figure 12 and Figure 13 are probability density functions for the queue size and total service time for the VMs in the simulation. The figures demonstrate that the CPU, Access, Qcpu, and Ded load balancing algorithms have much higher occurrence for longer queue size and longer total service time. S\text{queue} and S\text{wait}, on the other hand, seem to have much better results.

![Queue Length](image)

Figure 12. Probability Density Function (PDF) of Queue Size
Figure 13. Probability Density Function (PDF) of Total Service Time

Table 7 and 8 show the average and standard deviation of the queue size and the total service time for the simulated algorithms. We can tell that $S_{\text{queue}}$ and $S_{\text{wait}}$ have good performance compared to other algorithms. While both have similar performance, $S_{\text{queue}}$ is less expensive than $S_{\text{wait}}$. Thus, $S_{\text{queue}}$ is the preferable algorithm for VM load balancing.
Table 7. Average and standard deviation of queue size for each UEC

<table>
<thead>
<tr>
<th></th>
<th>RR</th>
<th>Rnd</th>
<th>CPU</th>
<th>$S_{queue}$</th>
<th>$S_{wait}$</th>
<th>Ded</th>
<th>Access</th>
<th>$Q_{cpu}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>21.192</td>
<td>24.387</td>
<td>34.970</td>
<td>18.995</td>
<td>18.990</td>
<td>36.245</td>
<td>33.495</td>
<td>31.801</td>
</tr>
<tr>
<td>StDev</td>
<td>1.007</td>
<td>1.119</td>
<td>3.022</td>
<td>0.5321</td>
<td>0.5545</td>
<td>0.888</td>
<td>2.122</td>
<td>1.9065</td>
</tr>
</tbody>
</table>

Table 8. Average and standard deviation of total service time

<table>
<thead>
<tr>
<th></th>
<th>RR</th>
<th>Rnd</th>
<th>CPU</th>
<th>$S_{queue}$</th>
<th>$S_{wait}$</th>
<th>Ded</th>
<th>Access</th>
<th>$Q_{cpu}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>1.163</td>
<td>1.366</td>
<td>1.913</td>
<td>0.920</td>
<td>0.894</td>
<td>2.103</td>
<td>1.862</td>
<td>1.759</td>
</tr>
<tr>
<td>StDev</td>
<td>0.221</td>
<td>0.309</td>
<td>0.482</td>
<td>0.153</td>
<td>0.149</td>
<td>0.420</td>
<td>0.480</td>
<td>0.316</td>
</tr>
</tbody>
</table>

Figure 14 and Figure 15 are cumulative density functions for the queue size and the total service times for the VMs in the simulation. It is clear $S_{queue}$ and $S_{wait}$ have the best result since it has almost 0 chance that a VM will encounter a queue size of more than 100. Furthermore, the maximum wait time for both algorithms is much shorter compare to other algorithms.
Figure 14. Cumulative Density Function (CDF) of Queue Size vs Occurrence

Figure 15. Cumulative Density Function (CDF) of Total Service Time vs Occurrence
Finally, box plots of the queue length and total service time of these 5 load balancing algorithms are shown in Figure 16 and Figure 17. It is clear $S_{\text{queue}}$ and $S_{\text{wait}}$ both have better performance as they have the smallest maximum queue size and total service time.

![Box plot of queue size](image)

**Figure 16. Box plot of queue size**
4. Conclusion

The experiment concludes that EXD-AHP can achieve the lowest number of writebacks which translates to the higher hit rate. However, the number of reduced writebacks compared to the rest of the scoring functions is only significant if the cache size is small. Furthermore, EXD-AHP provides the capability of giving mobile users with a high SLA level better cache performance. For load balancing, $S_{queue}$ and $S_{wait}$ load balancing algorithms can reduce request time. The is supported by the fact they have the lowest occurrence of high queue size and have the lowest minimum service time.
Furthermore, $S_\text{queue}$ is the preferable algorithm due to it being less expensive compared to $S_\text{wait}$.

5. References


