Fall 2017

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

Chin Tsai

San Jose State University

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 Compute Science
Dr. Teng Moh	Department of Compute Science
Dr. Thomas Austin	Department of Compute 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

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

LIST OF FIGURES

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 size 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 a 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.

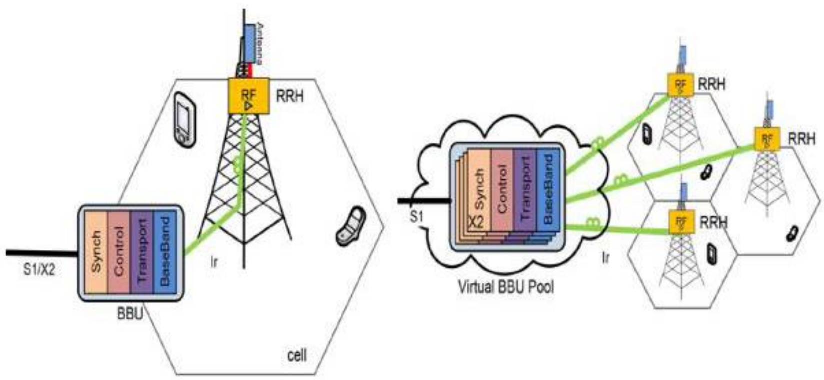

Figure 1. RAN (left) and CRAN (right) [1]

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.

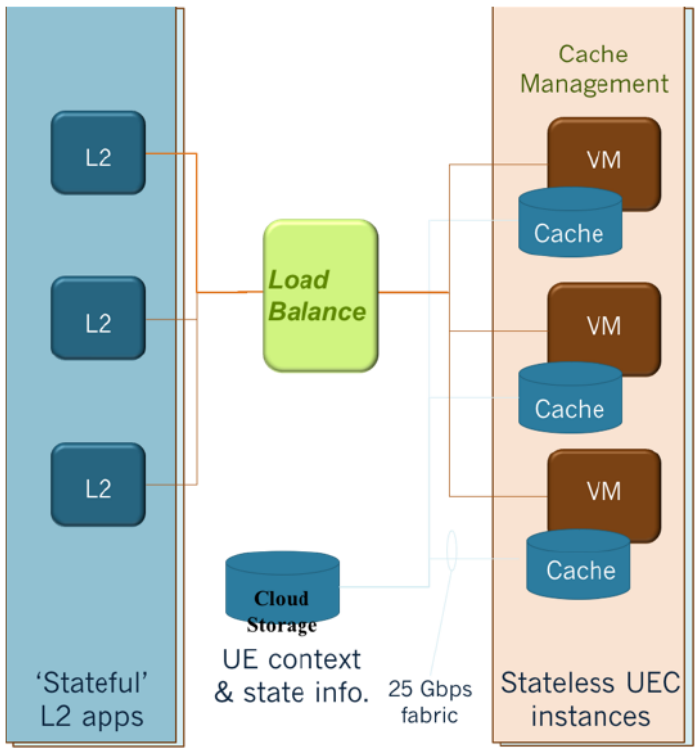

Figure 2. Project Overview

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.

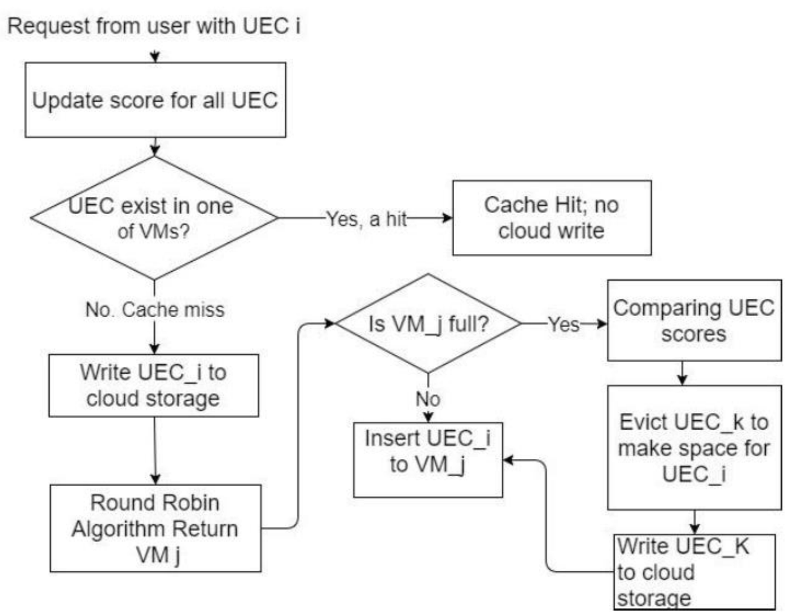

Figure 3. Cache Management Algorithm

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_i 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.

$$S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i1}} + \triangle\, \mathrm{u}) = S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i1}}) * \mathrm{e}^{-\mathrm{a}\triangle \mathrm{u}} \qquad (1)$$

$$S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i1}} + \triangle\, \mathrm{u}) = S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i1}}) * \mathrm{e}^{-\mathrm{a}\triangle \mathrm{u}} + 1 \qquad (2)$$

In EXD, each UEC’s score is determined by the time between requests. For both equations, $S_{\mathrm{i}}(u)$ represents the score of $\mathrm{UEC}_{\mathrm{i}}$ 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 $\mathrm{e}^{-\mathrm{a}\triangle \mathrm{u}}$ where Δu is the time elapsed since a $\mathrm{UEC}_{\mathrm{i}}$ is last requested at time u_{i1}. The longer the Δu, the higher the score reduction. The effectiveness of the Δu is determined by the value ‘a’. A small ‘a’ mean the exponent term has little effect on $S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i1}})$, 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 1. AHP Matrix

	L1	L2	L3	L4	Weight W_{AHP}
L1	1	5/1	5/1	5/1	0.579
L2	1/5	1	5/1	5/1	0.281
L3	1/5	1/5	1	5/1	0.102
L4	1/5	1/5	1/5	1	0.043

$$S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i}1} + \triangle\,\mathrm{u}) = S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i}1}) * \mathrm{e}^{-\mathrm{a}\triangle\mathrm{u}} + 1 + W_{AHP} \qquad (3)$$

$$S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i}1} + \triangle\,\mathrm{u}) = S_{\mathrm{i}}(\mathrm{u}_{\mathrm{i}1}) * \mathrm{e}^{-\mathrm{a}\triangle\mathrm{u}} + W_{AHP} \qquad (4)$$

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 $-\ln(u)/\lambda$ 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 2. Experiment Parameter Values

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

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

Acronyms	Cache Management Algorithms	UEC Scoring
LFU	Least Frequently Used	UEC request frequency
EXD [5]	Based on Exponential Delay	Equations (2) and (1)
EXD-AHP+1 (newly proposed)	Enhancing EXD with AHP	Equations (3) and (1)
EXD-AHP (newly proposed)	Enhancing EXD with AHP	Equations (4) and (1)

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, 3,750MB, and 5,000 MB.

$$Network\ Traffic = \frac{Total\ Writes * 200 * 8 * 1000}{Simulation\ Time} \qquad (5)$$

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.

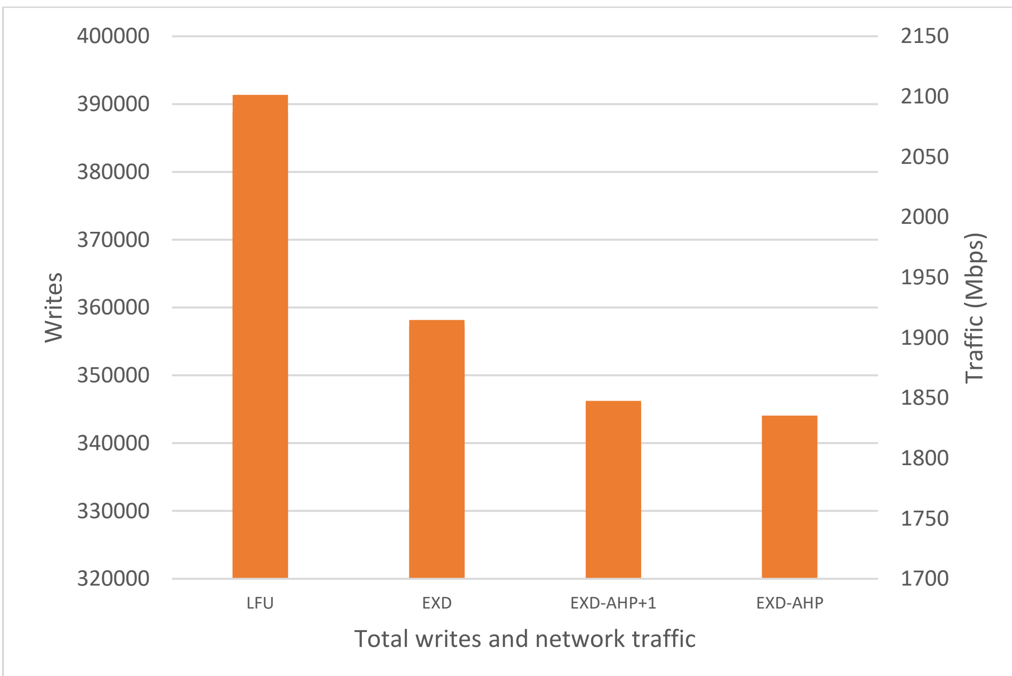

Figure 4: Total writes & network traffic, cache size 1250 MB

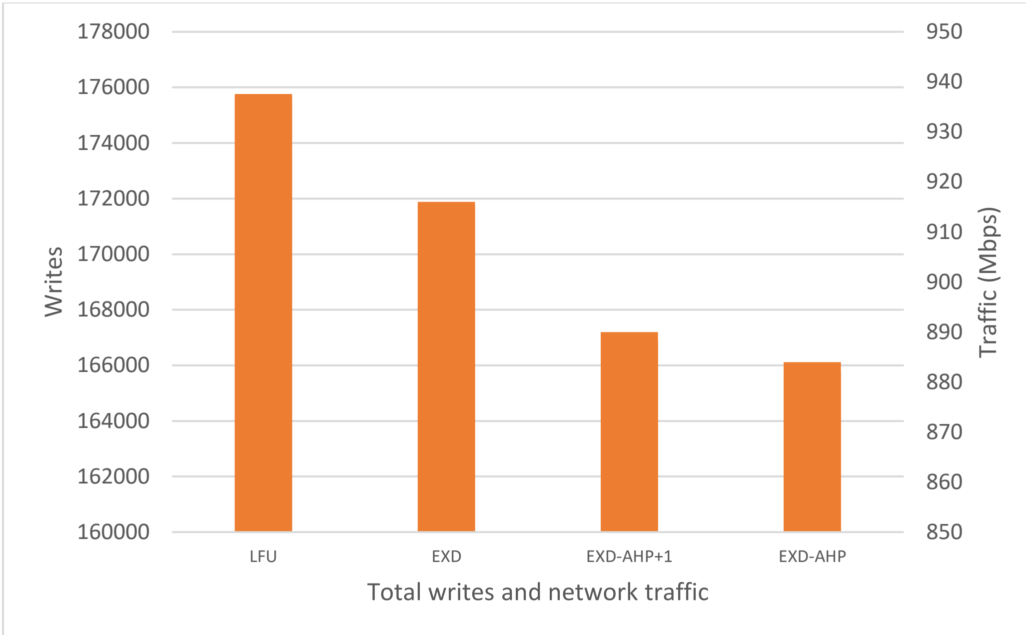

Figure 5: Total writes and network traffic, cache size of 2500 MB

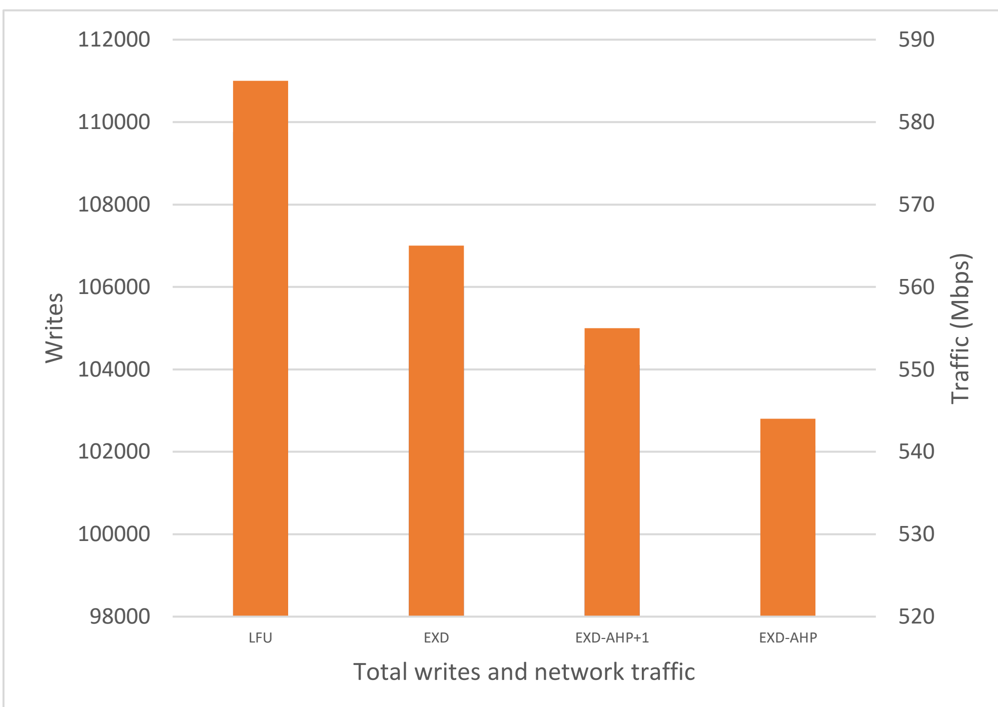

Figure 6: Total writes and network traffic, cache size of 3750 MB

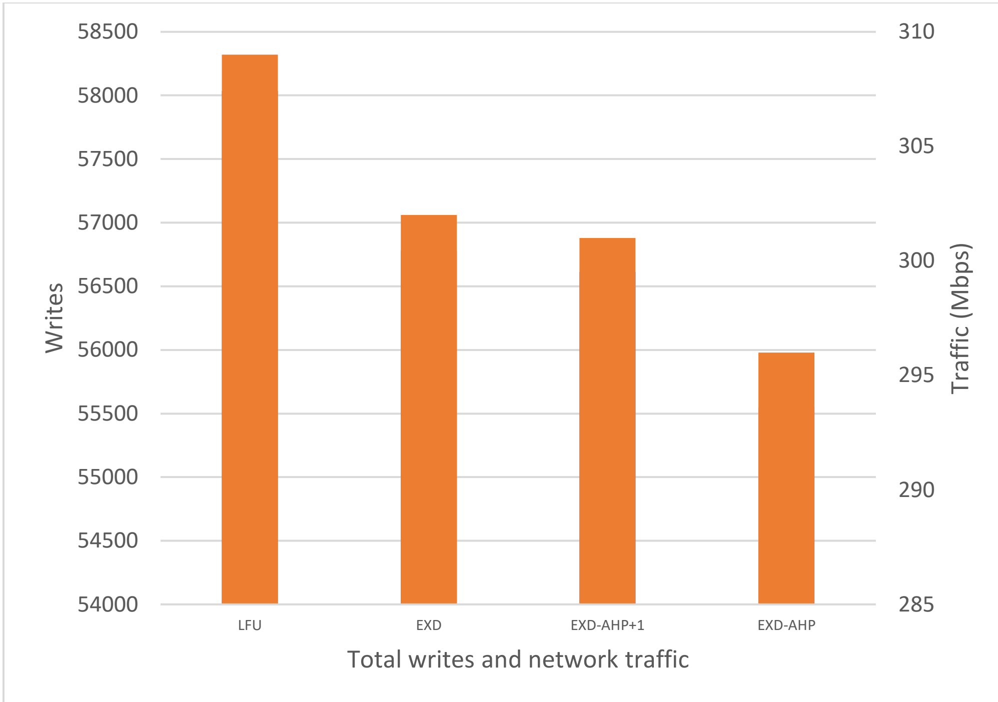

Figure 7: Total writes & network traffic, cache size 5000 MB

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.

$$L_i\ \ Cache\ Hit\ Rate = \frac{Total\ no.of\ L_i\ UEC\ cache\ hits}{Total\ no.of\ L_i\ UEC\ arrivals} \qquad (6)$$

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

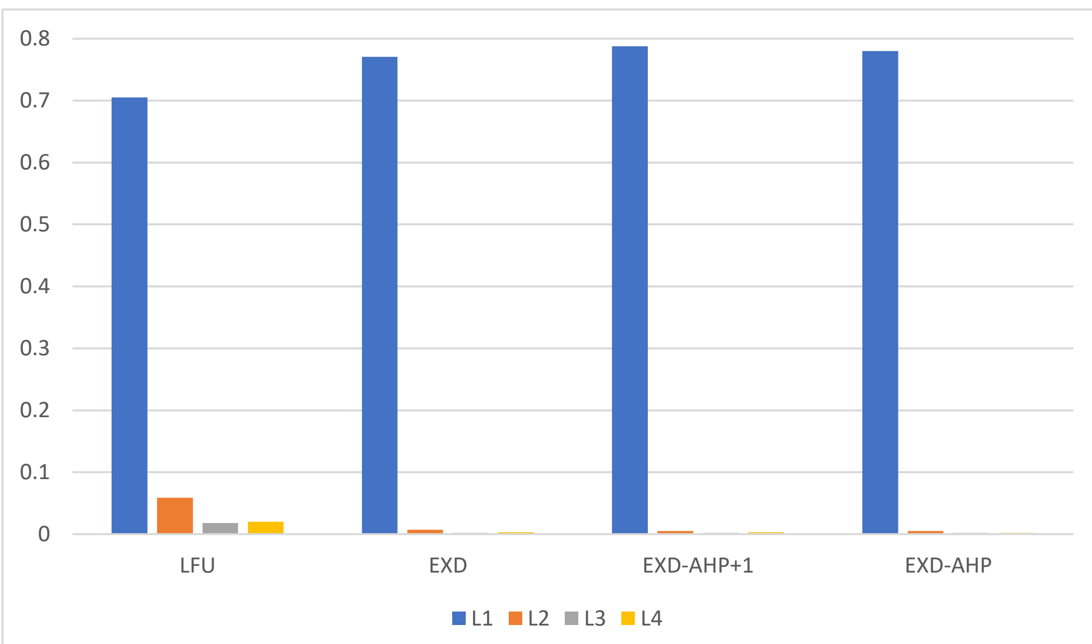

Figure 8: Hit rates of different service levels with cache size of 1250 MB

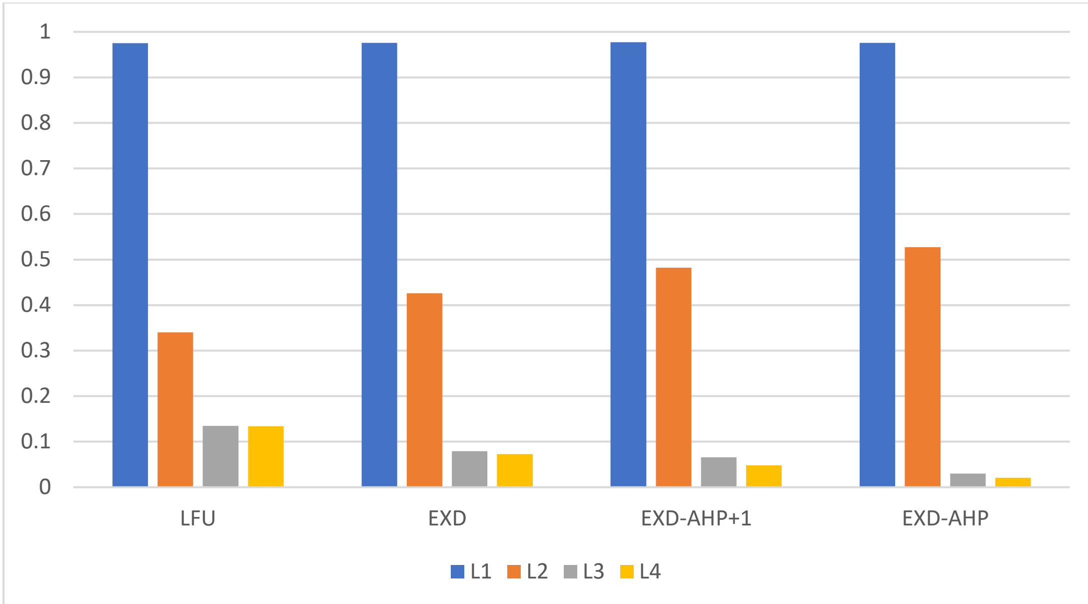

Figure 9: Hit rates of different service levels with cache size of 2500 MB

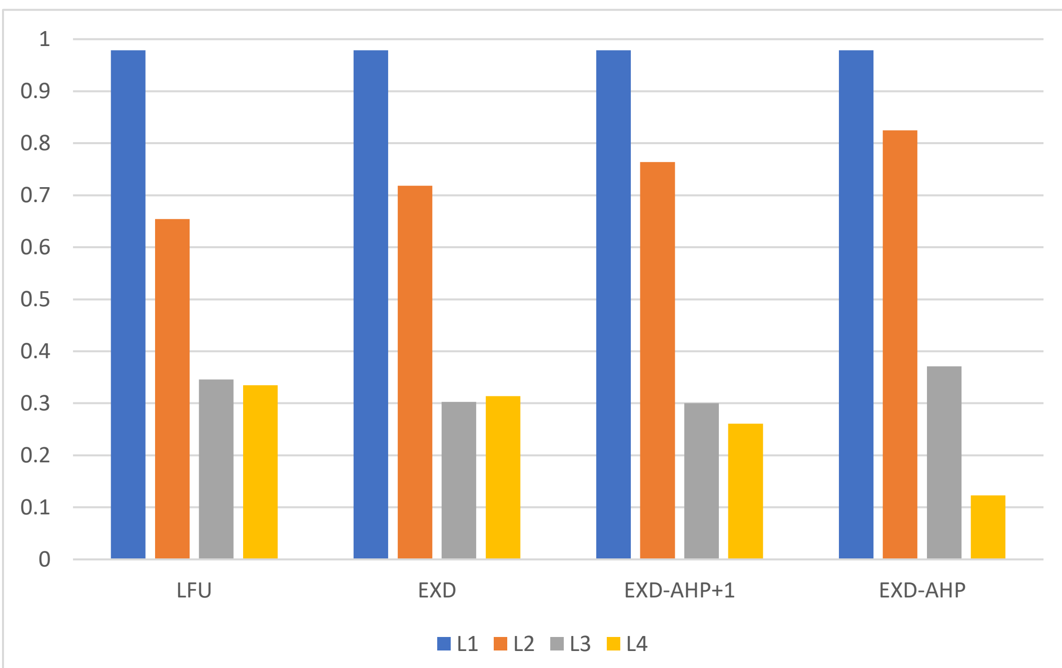

Figure 10: Hit rates of different service levels with cache size of 3750 MB

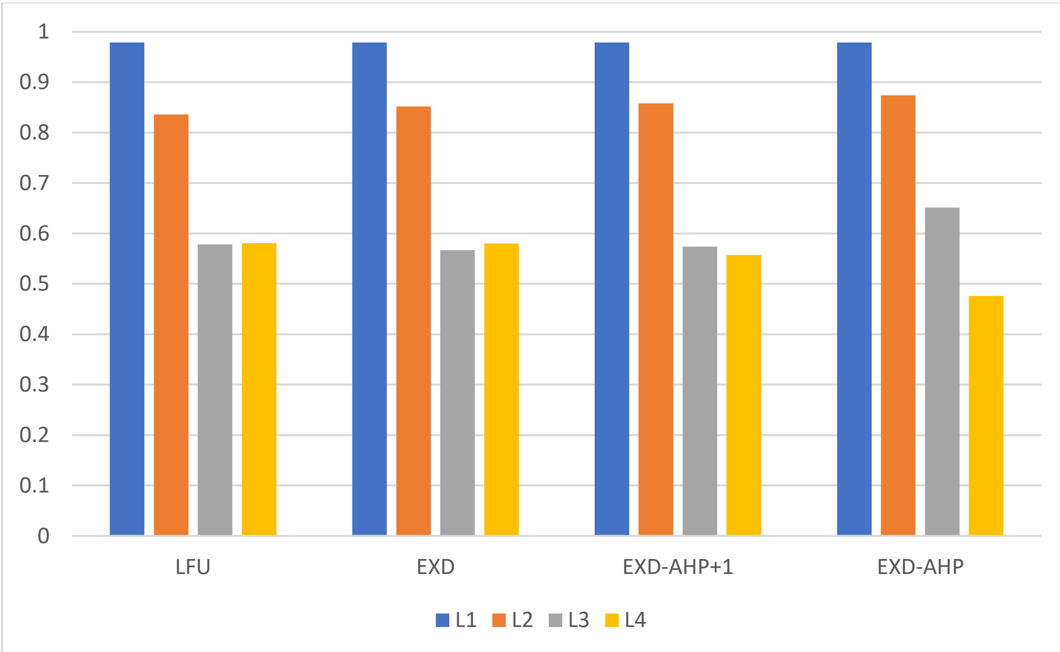

Figure 11: Hit rates of different service levels with cache size of 5000 MB

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

UE Events	Event arrival rate Events/eNB/second	Relative CPU Load
UE state transitions	750	1.00
Handovers	100	0.82
Tracking area updates (TAU)	30	1.24
Paging	500	0.26
Attach/detach	25	2.31
Actual Event Total	1400	

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

Acronym	Algorithm	Description
Five Basic LB Algorithms (listed in ascending order of complexity)		
RR	Round-robin	Round-robin of VM assignment.
RND	Random	Random VM assignment.
CPU	Based on CPU load	Assign to VM with min. cumulative CPU load.
S_{queue}	Based on shortest queue size	Assign to VM with shortest queue length.

S_{wait}	Based on shortest waiting time	Assign to VM with shortest waiting time.
Three Additional LB Algorithms (listed in ascending order of complexity)		
Ded	Dedicated	3 types of events with high occurrence (UE State transition, Handovers, Paging) each has own dedicated VM; all others share one VM.
Access	Based on min. number of UEC accesses	Assign to VM with min. number of UEC accesses.
Q_{CPU}	Based on queued time at CPU	Split all event types into 4 categories according their relative CPU load. The highest CPU load category will be assigned to VM with the shortest waiting time.

3.2 Experiment Setup for the Load Balancing Algorithms

CloudSim [15] is again 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 events arrival rate is 1,400 events per second with type distribution according to Table 4.

Table 6. Experiment Parameter Values

Parameter	Value
No. of VM	4
VM cache size	2,500MB
UEC record size	200 KB
UEC arrival process	Poisson
λ, mean UEC arrival rate	1,400 UEC/second
Total no. of distinct users	25,000
Total simulation time	5 minutes

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_{queue} and S_{wait}, on the other hand, seem to have much better results.

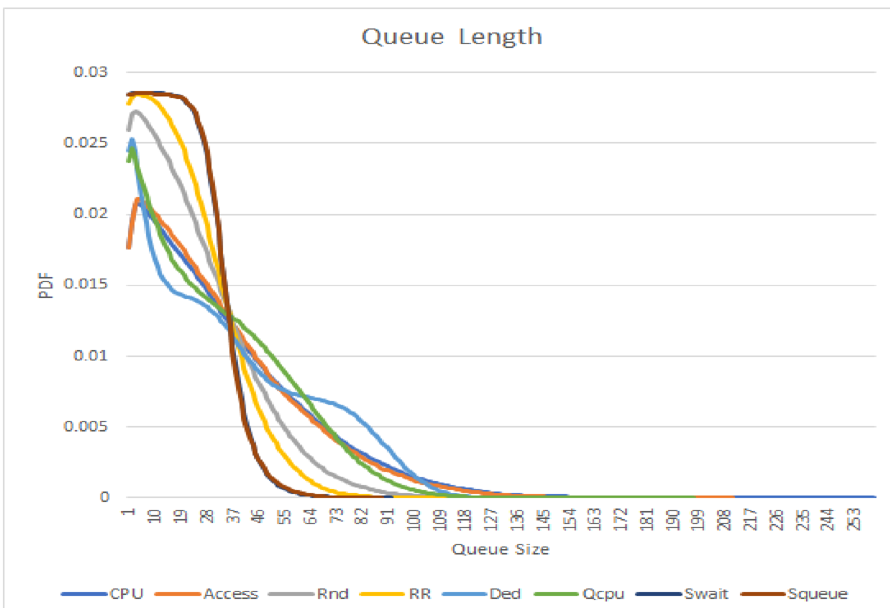

Figure 12. Probability Density Function (PDF) of Queue Size

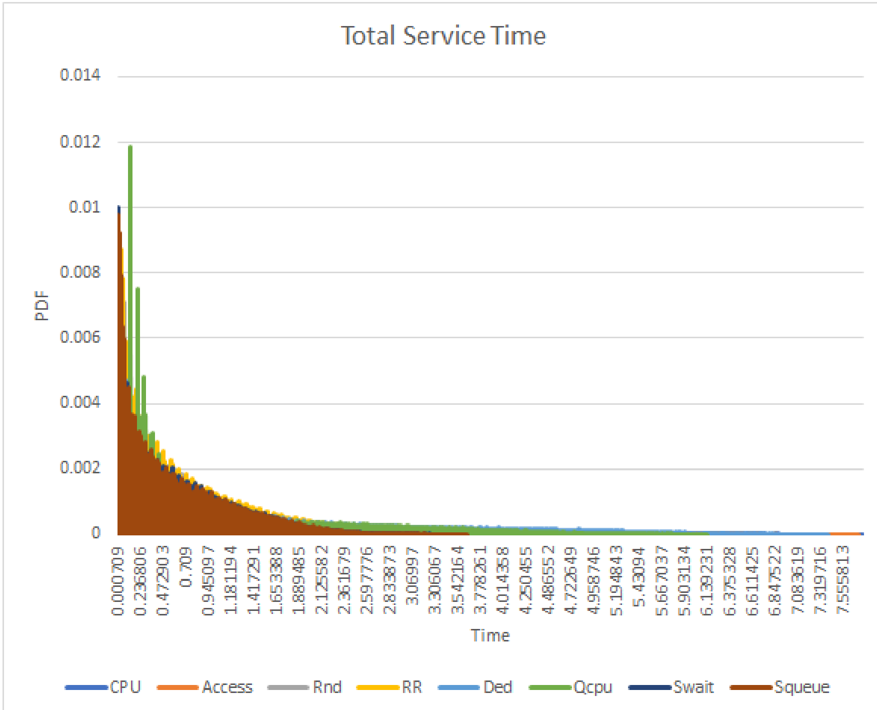

Figure 13. Probability Density Function (PDF) of Total Service Time

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_{queue} and S_{wait} have good performance compared to other algorithms. While both have similar performance, S_{queue} is less expensive than S_{wait}. Thus, S_{queue} is the preferable algorithm for VM load balancing.

Table 7. Average and standard deviation of queue size for each UEC

	RR	Rnd	CPU	S_{queue}	S_{wait}	Ded	Access	Q_{cpu}
Avg	21.192	24.387	34.970	18.995	18.990	36.245	33.495	31.801
StDev	1.007	1.119	3.022	0.5321	0.5545	0.888	2.122	1.9065

Table 8. Average and standard deviation of total service time

	RR	Rnd	CPU	S_{queue}	S_{wait}	Ded	Access	Q_{cpu}
Avg	1.163	1.366	1.913	0.920	0.894	2.103	1.862	1.759
StDev	0.221	0.309	0.482	0.153	0.149	0.420	0.480	0.316

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

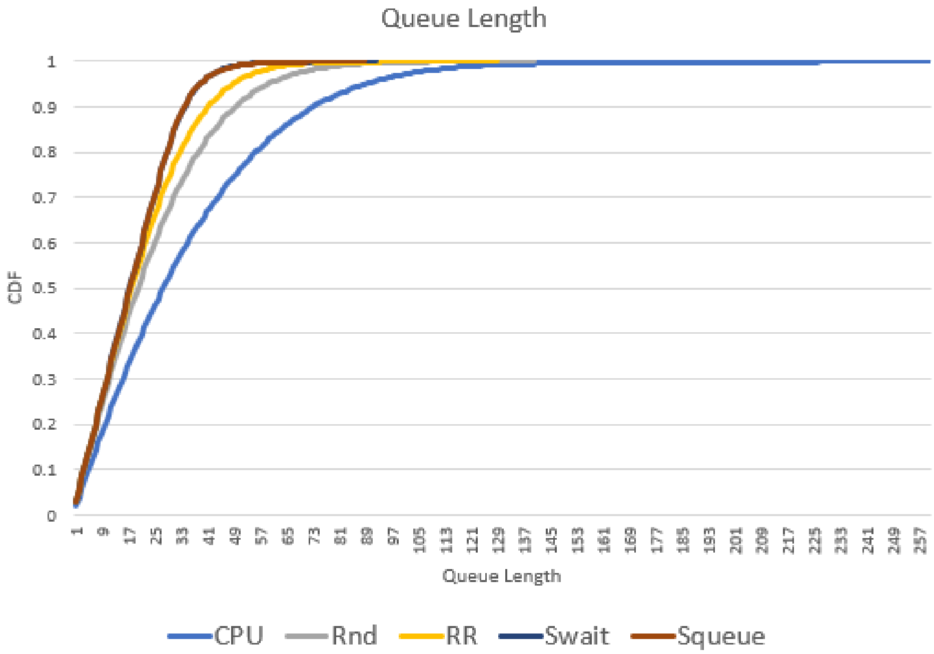

Figure 14. Cumulative Density Function (CDF) of Queue Size vs Occurrence

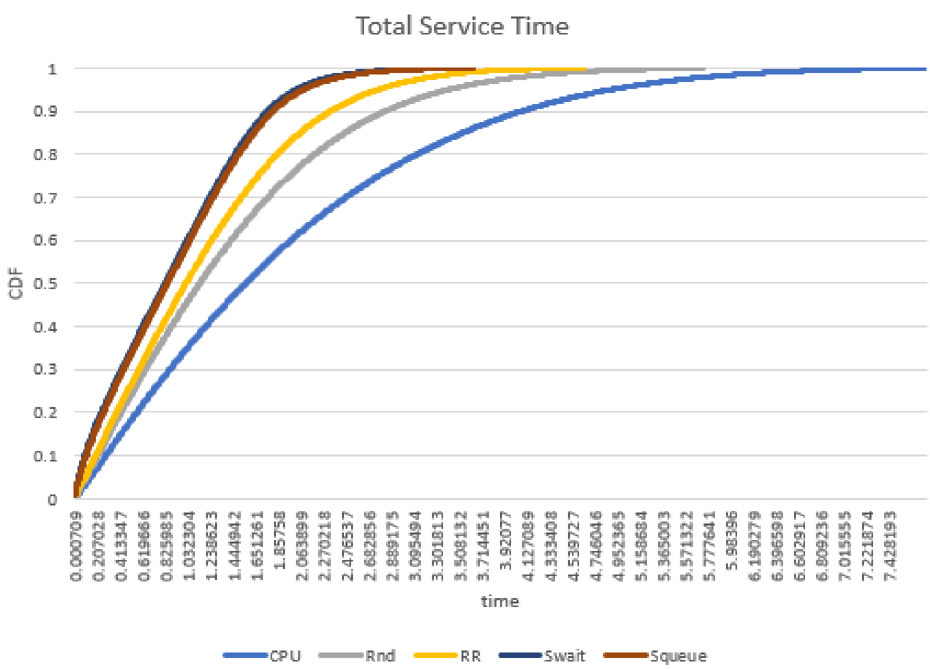

Figure 15. Cumulative Density Function (CDF) of Total Service Time vs Occurrence

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_{queue} and S_{wait} both have better performance as they have the smallest maximum queue size and total service time.

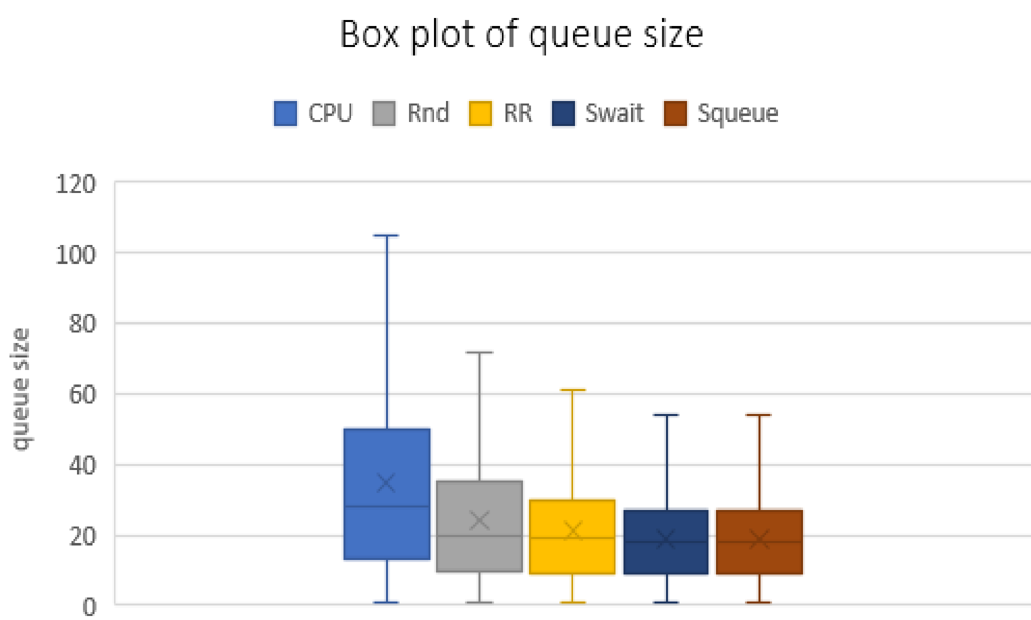

Figure 16. Box plot of queue size

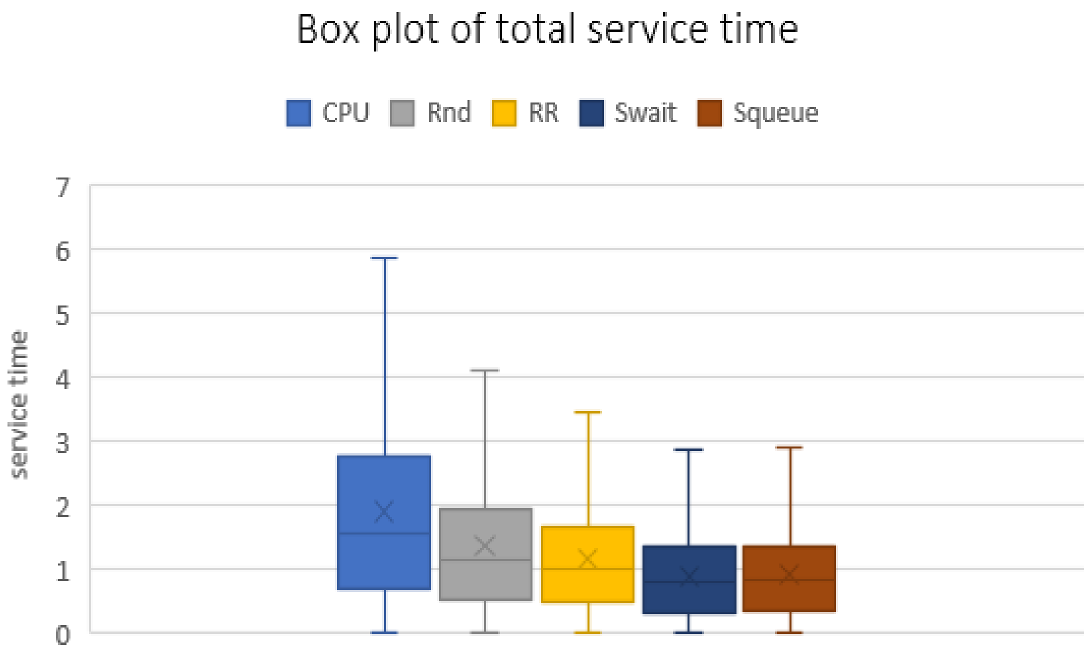

Figure 17. Box plot of total service time

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_{queue} is the preferable algorithm due to it being less expensive compared to S_{wait}.

5. References

[1] Checko, Aleksandra, Henrik L. Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S. Berger, and Lars Dittmann. "Cloud RAN for mobile networks—A technology overview." IEEE Communications surveys & tutorials 17.1 (2015): 405-426.

[2] Watanabe, Kimio, and Mamoru Machida. "Outdoor LTE Infrastructure Equipment (eNodeB)." FUJITSU Sci. Tech. J 48.1 (2012): 27-32.

[3] Liu, Liang, Feng Yang, Richard Wang, Zhenning Shi, Alan Stidwell, and Daqing Gu. "Analysis of handover performance improvement in cloud-RAN architecture."

[4] Communications and Networking in China (CHINACOM), 2012 7th International ICST Conference on. IEEE, 2012.

[5] Floratou, Avrilia, Nimrod Megiddo, Navneet Potti, Fatma Özcan, Uday Kale, and Jan Schmitz-Hermes. "Adaptive Caching in Big SQL using the HDFS Cache." Proceedings of the Seventh ACM Symposium on Cloud Computing. ACM, 2016.

[6] R. Saaty, "The analytic hierarchy process—what it is and how it is used," Mathematical Modelling, vol. 9, no. 3–5, pp. 161 – 176, 1987.

[7] Podlipnig, Stefan, and Laszlo Böszörmenyi. "A survey of web cache replacement strategies." ACM Computing Surveys (CSUR) 35.4 (2003): 374-398.

[8] Gomes, Andre, Torsten Braun, and Edmundo Monteiro. "Enhanced caching strategies at the edge of lte mobile networks." IFIP Networking Conference (IFIP Networking) and Workshops, 2016. IEEE, 2016.

[9] Pablo Muñoz; Raquel Barco; José María Ruiz-Avilés; Isabel de la Bandera; Alejandro Aguilar, "Fuzzy Rule-Based Reinforcement Learning for Load Balancing Techniques in Enterprise LTE Femtocells", IEEE Transactions on Vehicular Technology, pp. 1962–1973, vol. 62, 2013.

[10] Hisham Elshaer; Federico Boccardi; Mischa Dohler; Ralf Irmer, "Load & Backhaul Aware Decoupled Downlink/Uplink Access in 5G Systems", IEEE International Conference on Communications (ICC), pp. 5380–5385, 2015.

[11] B. Shahriari and M. Moh, "Generic Online Learning for Partial Visible Dynamic Environment with Delayed Feedback - Online Learning for 5G C-RAN Load-Balancer," *Proc. of the International Conference on High Performance Computing and Simulation (HPCS)*, Genoa, Italy, July 2017.

[12] Koushika, A. M., and S. Thamarai Selvi. "Load Balancing Using Software Defined Networking in cloud environment." *2014 International Conference on Recent Trends in Information Technology (ICRTIT),* IEEE, 2014.

[13] Zhang, Hailong, and Xiao Guo. "SDN-based load balancing strategy for server cluster." *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2014.

[14] S. Sathyanarayana and M. Moh, Joint Route-Server Load Balancing in Software Defined Networks using Ant Colony Optimization, *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS)*, Innsbruck, Austria, July 2016.

[15] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." *Software: Practice and experience* 41.1 (2011): 23-50.