An Adaptive Replica Configuration Mechanism Based on Predictive File Popularity and Queue Balance in Mobile Edge Computing Environment

Mao-Lun Chiang
Chaoyang University of Technology
Hui-Ching Hsieh (luckyeva.hsieh@gmail.com)
Hsing Wu University https://orcid.org/0000-0002-2666-3115
Ting-Yi Chang
National Changhua University of Education
Wei-Ling Lin
Chaoyang University of Technology
Hong-Wei Chen
National Changhua University of Education

Research Article

Keywords: mobile edge computing, replica configuration, popularity, prediction.

Posted Date: November 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-851015/v1

An Adaptive Replica Configuration Mechanism Based on Predictive File Popularity and Queue Balance in Mobile Edge Computing Environment

Mao-Lun Chiang [1] , Hui-Ching Hsieh[2*] , Ting-Yi Chang[3], Wei-Ling Lin[4] and Hong-Wei Chen[3]

[1]Department of Bachelor Degree Program of Artificial Intelligence, National Taichung University of Science and Technology, Taiwan, R.O.C.
skyjustfly@gmail.com

[2*]Department of New Media Communication
Hsing Wu University
New Taipei City, Taiwan R.O.C.
luckyeva.hsieh@gmail.com (*Corresponding author)

[3]Department of Industrial Education and Technology
National Changhua University of Education
Changhua City, Taiwan ROC
tychang@cc.ncue.edu.tw

[4]Department of Information Management
Chaoyang University of Technology
Taichung City, Taiwan ROC
will.wlink@gmail.com

Abstract

In the current era of the Internet of Things (IoT), various devices can provide more services by connecting to the Internet. However, the explosive growth of connected devices will cause the cloud core overload and significant network delays. To overcome these problems, the Mobile Edge Computing (MEC) network is proposed to provide most of the computing and storage near the radio access network to reduce the traffic of the core cloud network and provide lower latency for the terminal.

Mobile edge computing can work with third parties to develop multiple services, such as mobile big data analysis and context-aware services. However, when there is a large amount of popular data accessed in a short period, the system must generate many replicas, which will not only reduce access efficiency but also cause additional traffic overhead. To improve the above problems, an Adaptive Replica Configuration Mechanism (ARCM) is proposed in this paper to predict the popularity of the file and make a replica to the low-blocking node. This method spreads the subsequent access workload by copying the popular file in advance to improve the overall performance of the system.

Keywords: mobile edge computing; replica configuration; popularity; prediction.

1. Introduction

In general, the cloud platforms can provide multiple service models, such as infrastructure as a service (IaaS) [26], platform as a service (PaaS) [9], and software as a service (SaaS) [18]. In the IaaS model, the computing resources and structures are provided to companies which are including servers, storage, the infrastructure of network topologies and virtual machines. Based on the IaaS model, users can scale on demand to provide more flexible and innovative services to balance the dynamic workloads. In the PaaS model, the vendors offer a development environment to application developers including operating systems, database, web server and programming-language execution environment. In the last kind of service, called SaaS, allows users to use the applications over the Internet on demand by its authority. However, in recent years, a large number of IoT applications have increased the load of the cloud core network and caused a long delay time, making it impossible for users to obtain better service quality.

To solve these problems, the MEC (Mobile Edge Computing) network is proposed in recent years to provide information technology (IT) and cloud services by the Radio Access Network (RAN) [16, 19]. Besides, the advantage of MEC network is close to mobile clients to provide ultra-low latency, large bandwidth, real-time computing, and flexible services by authorizing third-party applications [1, 5, 17], such as location tracking, mobile big data analysis, video optimized transmission, and context awareness.

Traditionally, the access method in the cloud environment is to upload data of local device to the server for further calculation. When the access workload of the server is getting higher, the system must duplicate more replicas to disperse the works. In the meantime, the duplication process may lead to insufficient storage, and the availability and access performance of the file will decline. Also, it usually requires higher costs to maintain hardware equipment [22, 30]. Therefore, proposing an effective replicating strategy is an important issue in the cloud environment. This kind of issue is relatively important in the MEC network. Basically, the MEC environment can provide localized content for users in the nearby service area, such as video streaming [3], AR / VR [28] and other services. While providing these kinds of services, the popular files may be requested frequently and repeatedly. Although the MEC network structure can reduce the response time of the service by deploying the server near to the end-users, it may also increase the time for accessing the data while there has no sufficient available data or congestion of service nodes. The mention problems will instead eliminate the original advantages of the MEC network.

In this paper, an effective data replication strategy has been proposed. The main purpose of the protocol is to improve the problem of insufficient availability while the high popularity replicas are under content localization. Furthermore, the congestion problem while the terminal devices requesting node services frequently will be improved, too. Here, the main idea of the proposed method is to predict the popularity of each data. When the popularity of the data is in high demand, it will be duplicated for access in advance. This mechanism can make the system responds to the request while the system is under environmental changes. Besides, the generated replicas will be allocated to the services nodes which are in low blocking status and having the most queuing space to provide services. The proposed protocol can enhance the availability, access efficiency and load balancing under the MEC environment.

The rest of this article is organized as follows. Section 2 describes the related works of replicating systems. Section 3 shows the details of our proposed protocol. Section 4 describes the experimental design. The experiments and analysis of our proposed protocol are illustrated in Section 5. Finally, the conclusion is presented in Section 6.

2. Related Works

In this section, the concepts of the mobile edge computing network will be introduced first. After that, some famous file replication strategies will be discussed. The comparisons of the advantages and the disadvantages of these strategies will be described in this section, too.

2.1 Mobile Edge Computing Network

The emergence of cloud computing has led computing technology into a new era. The main reason for it becoming more popular is because the operating modes can not only reduce the overhead costs of cloud providers but also can improve the scalability of the system. However, cloud data centers are usually far away from the terminal equipment and users. It is less conducive to the applications which require low latency while accessing. So far, some interesting research topics have been proposed in the MEC network [4, 6]. For example, the Internet of Things [20], Internet of Vehicles [12], and AR [7] are these kinds of applications. In other words, when the mention applications are applied under the cloud computing environment, it may lead to long transmission delays, Internet congestion or degradation of QoS, etc. To solve the above problems, the MEC network structure which is more closer to the mobile users has been proposed [13, 29, 31].

The concept of the MEC network is deploying the computing center near to end-users for data processing. This method can prevent a large amount of raw data being transmitted to the cloud data center. This deployment can help to reduce the load of the cloud data center and the response time for end-users while requesting services. Besides, the computing center under the MEC network is near to mobile users. Under such a structure, the system can provide context-aware services for local users by collecting RAN messages in the area [2] more efficiently. Unfortunately, localization of content may cause the popular data been requested at the same time. This circumstance will cause virtual machine nodes to become congestion. The access efficiency of the system will decrease also. Hence, proposing an efficient data replication strategy to improve the mention problems is also an important issue under the MEC network. Subsequently, the related works of data replication strategies proposed in the past will be introduced in the next sub-section.

2.2 Data Replication Strategies

In the past, scholars have proposed some data replication algorithm to improve the load balance or access efficiency for the system under different network architectures [11, 14-15, 21, 23-25, 32-33]. For example, to enhance the capability of the cloud storage systems, Qingson et al. proposed an efficient dynamic replication management scheme which is called CDRM [21]. The main idea of CDRM is to distribute the replicas to the nodes which have low blocking probability. Under such a mechanism, tasks can be processed more quickly, and service efficiency and load balancing can be improved. However, CDRM always selects the nodes with the lowest blocking probability as the service nodes, and this will cause the selected service nodes to be accessed all the time. Under such a circumstance, the workload of each services nodes will be unbalanced and the overall access efficiency will be decreased.

In the past, scholars have proposed a dynamic data replication algorithm DDRA [11] to improve the mentioned problem under the cloud computing network. The main idea of the DDRA algorithm is to provide more suitable service nodes for users depends on the blocking probability of the nodes and the queue space within the reference nodes. This mechanism can prevent tasks to be distributed into the congesting node and can also achieve load balancing for the system. However, DDRA will decide whether to increase a new duplication based on the ratio of file popularity to the number of replicas, and the threshold of the popularity is the average of all previous file accesses. Under such a circumstance, when there is a huge gap in the number of access times between the files, the files which are currently be popular, will be judged as not popular due to the original number of access times is smaller than the average threshold. Consequently, no new replica will be added by the system, and the workload of the nodes will keep increasing and the access efficiency will decrease. On the other hand, files with a large number of original access times will still cause unnecessary resource waste by adding new replicas because the current number of accesses still exceeds the threshold.

Wang proposed an adaptive file replication mechanism called PARM [15] in the cloud environment. Its main idea is to predict the popularity of the files for fast adjustment by applying the characteristics of atomic decay to the access times of files. However, since

PRAM does not set a stop-loss point for replica generation, this will cause popular files to be duplicated constantly, and the workload of the system will keep increasing.

To improve the problem of setting the threshold of popularity, scholars have proposed an adaptive file replication strategy called ADRM [22]. The main method is to duplicate the popular files by predicting the popularity of the archive. By applying the ADRM strategy, the number of replicas can be controlled with an appropriate ratio by setting the ratio of the number of the replica to avoid excessive resource consumption caused by excessive addition of replicas. Unfortunately, the ADRM strategy does not take the replica configuration into account, this will result in replicas not being deployed on nodes to handle the access workload.

In this paper, an adaptive replica configuration mechanism ARCM has been proposed to find out the files which have high popularity in advance. This can help to duplicate the file to disperse the workload generated by subsequent popular files beforehand. After that, when a file has been analyzed as a popular file and the ratio of the number of the replica is sufficient, the system will then move the archive from the high-blocking node to the low-blocking node to avoid excessive replica generation. Finally, the service node is selected to achieve load balancing by allocating replicas to the node which is low-blocking and has more space. The differences between the algorithms are shown in Table 1.

Table 1. The comparisons between ARCM, ADRM, DDRA, PARM, and CDRM.

	Popularity Prediction	**Replica Deployment**	**Replica move between nodes**	**Load Balancing**	**Environment**
ARCM	*O*	*O*	*O*	*O*	*MEC/Cloud*
ADRM[22]	*O*	*X*	*X*	*X*	*MEC/Cloud*
DDRA[11]	*X*	*O*	*O*	*O*	*Cloud*
PARM[15]	*O*	*X*	*X*	*X*	*Cloud*
CDRM[21]	*X*	*O*	*O*	*△*	*Cloud*

3. The Proposed Adaptive Replica Configuration Mechanism

In this paper, an Adaptive Replica Configuration Mechanism (ARCM) has been proposed to optimize the replica configuration in the MEC environment, and the system architecture is shown in Figure 1. Here, the MEC servers are responsible for responding to the users' requests, and the users will request and access the data from the MEC servers. Furthermore, each MEC server will in charge of allocating the file to the virtual machines

(VM) and collecting the historical access records. The role of the VMs is to manage the file for users to access. When the MEC server collects all the system information (including the access frequency of the file, the probability of the block node…) from VMs, the system will then execute the ARCM algorithm to allocate the file. The detailed procedure of the ARCM algorithm will be introduced as follows.

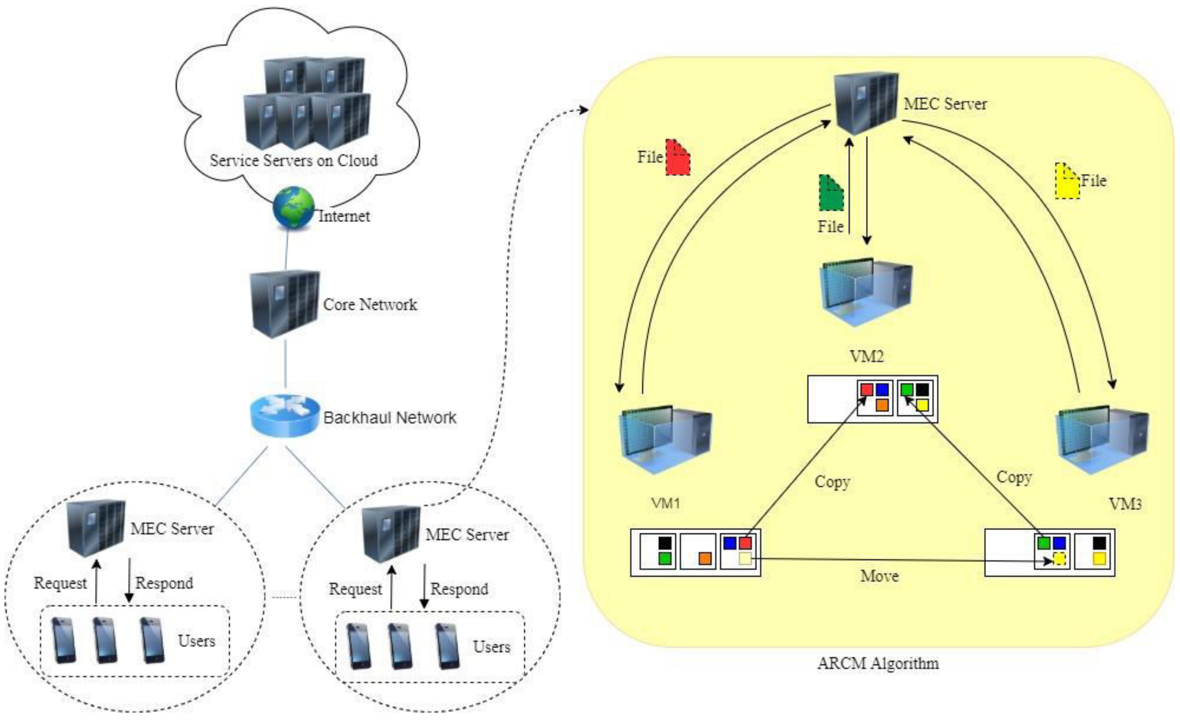


Figure 1. The System Architecture

In the ARCM mechanism, there are two key emphases in work: data Replication and service node selection. The goal of data replication is to calculate the popularity of the file to decide whether to replicate the file or not. This can help to improve the access efficiency for replica storage architecture. For the goal of service node selection, the system will select a proper node to place the replicas. This can help to increase the load balance of the overall system. The detail of the procedure will be described next.

3.1 Data Replication Strategy

To improve the access efficiency for replica storage architecture, the first key emphases in the work of ARCM is to adjust the number of replicas adaptively. Here system must get the time interval since last access for each file and then calculate the average time interval among all files, and predict the number of file access to evaluate whether the file will become popular or not in the future. Finally, the corresponding update operations are given according to the popularity of the replica and the ratio of the number of replicas, The operations include: adding more replicas, moving replicas, or deleting replicas. The details are described below.

3.1.1 Calculate Average Time Difference

To adjust the number of replicas adaptively, the first thing is to decide whether the file is popular or not. To do so, the system must calculate the time interval since last access for each file and then calculate the average time interval among all files, and the related procedure can be shown as formula (1) [15]. Here, Ave_{time} represents the average time interval of all files, T_i represents the *i-th* access time and *n* represents the total number of accesses. When the newest average time interval is less than the last time interval, it means that the file has a popular trend. In contrast, it means that the data is not popular.

$$\boldsymbol{Ave_{time}} = \frac{\sum_{i=1}^{n}(T_i - T_{i-1})}{n} \qquad \textit{(1)}$$

3.1.2 Predict the Number of File Access

In the ARCM mechanism, the concept of atomic decay [8] will be applied to predict the number of file accesses. The goal of this prediction is also to evaluate whether the file is popular or not. The procedure can be described as formula (2) [15], where Pre_{access} is the number of files accesses predicted, and PA_i represents the *i-th* stage visits. Then, the file access times predicted in this stage are compared with the predicted access times in the previous stage. If there are more predicted access times in this stage, it means that the file has a popular trend. Conversely, it means that the file is not popular.

$$\boldsymbol{Pre_{access}} = \sum_{i=1}^{n}(PA_i * 2^{1-i}) \qquad \textit{(2)}$$

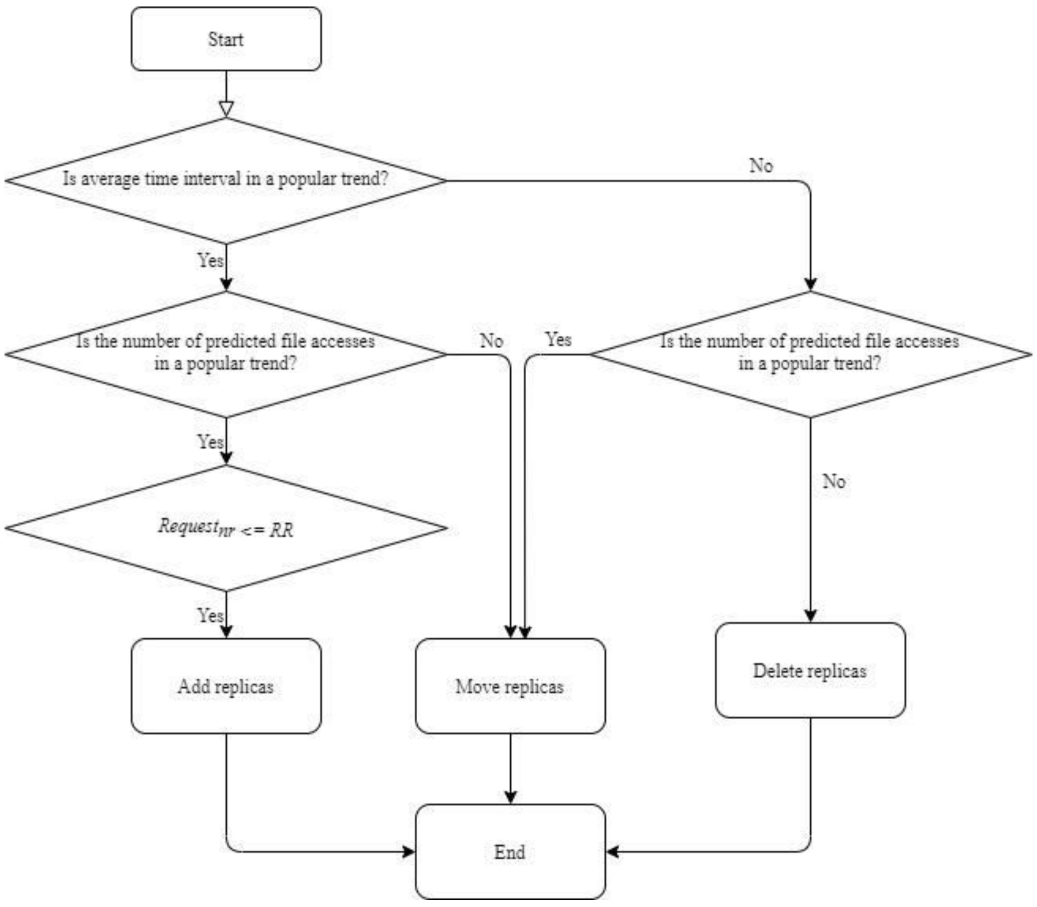
3.1.3 Replica Update Strategy

When the results of the average time interval and the number of predicted file accesses are in a popular trend, the system will further apply the Ratio of the number of Replica (RR) as the threshold for deciding whether to add more replicas or not. This is because maintaining the number of replicas based on RR value will help the system to get better access efficiency [11]]. Here, the system will apply the formula (3) to calculate the current requested ratio of the number of replicas $Request_{nr}$ and then compare with the RR value [11]. In formula (3), nr_i indicates the number of current requested replicas of file and tnr_i is the total number of replicas. When the current requested ratio of the number of replicas is smaller than the RR threshold, the system will add more replicas. In contrast, if the current requested ratio of the number of replicas is greater than or equal to the RR threshold, it means that the number of replicas is sufficient. Under such a circumstance, the system will move the replicas from the node which has the highest blocking probability to the node with the lowest blocking probability.

Furthermore, when the result of the average time interval is in a popular trend and the number of predicted file accesses is not in a popular trend, the system will also move the replicas from the node which has the highest blocking probability to the node with the lowest blocking probability.

For the last situation, when the results of the average time interval and the number of predicted file accesses are both not in a popular trend, the system will delete the replicas from the node which has the highest blocking probability. Noticeably, while removing the replicas, the system must ensure that the number of replicas is kept at least 3 copies to maintain basic usability. The overall flowchart is shown in Figure 2.

$$\boldsymbol{Request_{nr}} = \frac{\boldsymbol{nr_i}}{\boldsymbol{tnr_i}} * \mathbf{100}\% \qquad (3)$$


Start
Is average time interval in a popular trend?
No
Yes
Is the number of predicted file accesses in a popular trend?
No
Yes
Is the number of predicted file accesses in a popular trend?
Yes
No
$Request_{nr} <= RR$
Yes
Add replicas
Move replicas
Delete replicas
End
Figure 2. Data Replication Strategy

3.2 Service Node Selection Strategy

In the previous sub-section, the concepts of deciding whether the file is popular or not, and the conditions about add, move or delete files are introduced. Next, another key emphasis in the work of ARCM that is filtering the proper service nodes to place the replicas will be given. This procedure can help to ensure the quality of service and the load balancing of the overall system. Here, the procedure can be divided into two phases and the details are shown as follows.

3.2.1 Anti-Blocking Phase

To avoid replicas being allocated to the nodes which are in congested, users must spend more time to access the file, and the new task must wait until the service node is available. This procedure will increase the overall latency of the MEC environment. To improve these dilemmas, the blocking probability of each node is obtained by calculating the node's arrival rate and the request delay time [21]. Through this phase, the system can filter out the nodes which are in congested, and the nodes that are more suitable for services will be selected.

To calculate the arrival rate, formula (4) has been provided in ARCM. Here, p_j represents the popularity of the file being accessed, r_j represents the number of replicas, and λ is the real arrival ratio in all requests. Through this formula, the system can calculate the arrival rate of each node λ_i.

$$\lambda_i = \frac{p_j}{r_j}\lambda \qquad (4)$$

After getting the arrival rate, the system can bring the result into formula (5) to get the Block Probability (BP). In formula (5), τ_i represents the delay time which refers to the time for the terminal device to read the replica, and c_i represents the number of memory blocks divided by the node. To make the system more realistic, the system will apply the M/M/1 [21] rule to simulate the arrival rate for each task. When the memory blocks are full of tasks, new tasks must wait in the queue, and this situation is called blocked.

$$\boldsymbol{BP_i} = \frac{(\lambda_i\tau_i)^{c_i}}{c_i!}\left[\sum_{k=0}^{c_i}\frac{(\lambda_i\tau_i)^k}{k!}\right]^{-1} \qquad (5)$$

After calculating the blocking probability of each node through formula (5), the system will set up *AvgBP* value as the low blocking probability and this value will be the threshold for deciding whether the node is congested or not. To sum up, the system can select the nodes which are under low-blocking for services. This can help to decrease the latency while users are requesting for services and to improve the performance of the overall system.

3.2.2 Reference Queue Balance Phase

After selecting the node whose block probability is lower than the average value, the system will find out the node which has most queue space as the preferentially serving node from the nodes with a lower average blocking probability. Here, the method in this phase will apply the concepts of Reference Queue (RQ) proposed by Chiang et al.[11]. By preferentially assigning replicas to the nodes with most queue space, tasks will evenly be distributed and be processed quickly. This will help to improve access efficiency and achieve better load balancing. Figure 3 shows the entire flow of the service node selection strategy.

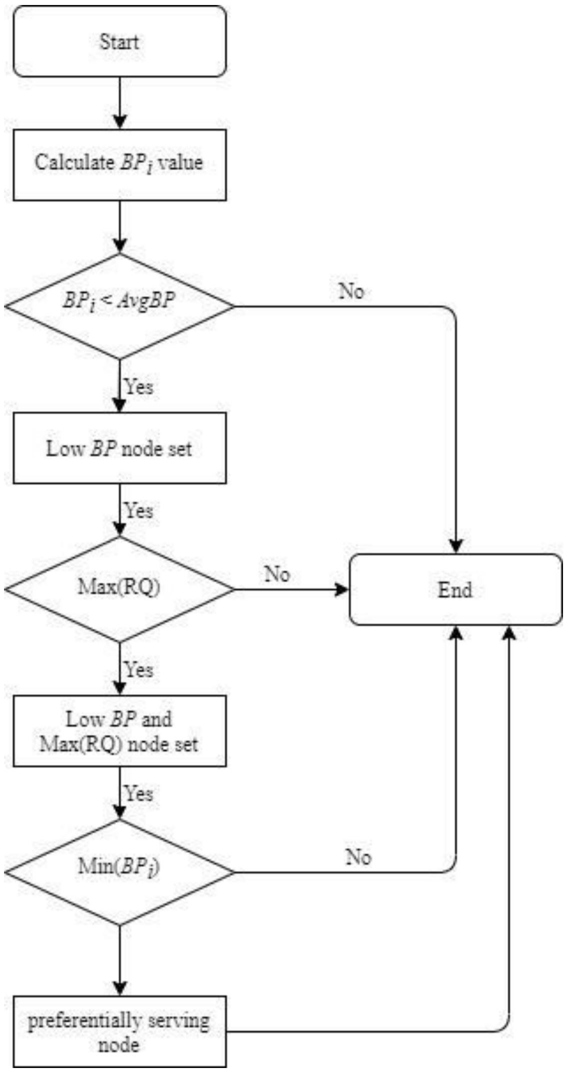


Figure 3. Service Node Selection Strategy

4. Example

Basically, by applying the ARCM protocol, the system can get a more efficient file replication strategy to maintain the availability of replicas, and the replica access efficiency can be improved under the MEC environment. To help to understand the proposed ARCM protocol, an example has been given in this section.

Once the terminal devices are connected to the MEC environment for requesting the file access services, the system will search for the related data according to the request and then set up the initial number of replicas base on the type of the file. At this time, the system must manage the record about the files which are accessed by nodes, and each record will be compared to the data accessed in the past. Then, the system will dynamically adjust the number of replicas by analyzing the popularity of the file and calculating the replica ratios. Finally, the system will generate a new replica and move it to the appropriate service node. The assumptions for the example are shown in Table 2:

Table 2. The assumptions for the example.

Assumptions	**Values**
Time period	5 minutes
Data node number	N_1 ~ N_8
Access file ID	A ~ D

When the terminal device is connected to the MEC environment to request access for the service, the system will analyze the type of the file and allocate numbers of replica according to the file classifications. If the files have a higher usage rate and require a longer time for storage, the system will allocate more number of replicas for the files. In contrast, for the files that have a lower usage rate and do not require too much time for storage, the system will allocate fewer replicas for them. The related allocations are shown in Table 3. Besides, when the file size is greater than 64MB, it will be stored in Block Level [27]. In contrast, it will be stored in File Level when the file size is less than 64MB [10]. By applying this mechanism, the system will access the files from different blocks to avoid the access delay caused by accessing the big file from a single node.

Next, when the terminal device accesses the files, the system will record the file ID, storage node, start timestamp, end timestamp, and the file size of each file according to the system storage format under the MEC environment. The format is shown in Table 4. When a time interval has passed, the system will generate a log analysis based on the number of accesses and the time interval between each accesses, and the related example is shown in Table 5.

Table 3. The allocations results of the file.

File ID	File Type	File classification	Number of Replicas
A	Trending information	Short-term storage	3
B	Weather information	Short-term storage	3
C	Natural disaster Information	Long-term storage	5
D	Traffic information	Short-term storage	3

Table 4. An example of the file storage format.

File ID	Node ID	Start Timestamp	Stop Timestamp	Size

A	N2	20180903192013	20180903265115	250MB

Table 5. Details of access records.

File ID	Node ID	Number of Accesses	Spacing Time
A	N2	5	7
A	N7	9	15
A	N5	7	13
B	N4	6	5
B	N6	4	4
B	N7	15	17
C	N8	9	8
C	N3	10	10
C	N5	6	5
C	N1	11	12
C	N2	14	14
D	N4	13	4
D	N5	7	13
D	N6	8	8

In the initial status, because the system has no historical data, the file will not be added, moved or deleted at this time, and the status of the record can be shown in Table 6. At the end of the second period of time, the system will compare the current access data (shown in Table 7) to the historical data. In terms of predicting the number of accesses, it will apply the atomic decay method [15], and the result is shown in Table 8. The comparison of the average access time interval is shown in Table 9.

Table 6. Access records for the first period of time.

File ID	Number of Accesses	Average Spacing Time	Come From
A	21	12.43	N2:5; N7:9; N5:7
B	25	12.04	N4:6; N6:4; N7:15
C	50	10.6	N8:9; N3:10; N5:6; N1:11; N2:14
D	28	7.39	N4:13; N5:7; N6:8

Table 7. Access records for the second period of time.

File ID	Number of Accesses	Average Spacing Time	Come From
A	6	12.26	N2:3; N7:3
B	18	11.47	N4:5; N6:3; N7:10
C	29	10.13	N8:7; N3:8; N5:6; N1:8

D	11	8.58	N4:5; N6:6

Table 8. Prediction of the number of accesses by the atomic decay method.

File ID	History	Now	Prediction
	Number of Accesses	Number of Accesses	Number of Accesses
A	21	6	$21*2^{-1}+6*2^{0}=16.5$
B	25	18	$25*2^{-1}+18*2^{0}=30.5$
C	50	29	$51*2^{-1}+29*2^{0}=54$
D	28	11	$28*2^{-1}+11*2^{0}=25$

Table 9. Comparison of the average access time interval.

File ID	History	Now
	Average Spacing Time	Average Spacing Time
A	12.43	12.26
B	12.04	11.47
C	10.6	10.13
D	7.39	8.58

After comparison, the system can analyze whether the file is in a popular trend or not. For the number of accessing the files, the ARCM algorithm applies the atomic decay method to make predictions and then compares the predicted value to the previous access record to analyze whether the file has a popular trend in access times. The results are shown in Table 9. In the results, we can see that the file classification of B and C are in a popular trend. Subsequently, the historical average access time interval will be used as the threshold value, and analyze whether the file is a popular trend or not based on the access time interval. The comparison result is shown in Table 11. Finally, to avoid adding too many replicas to the popular files, the system must maintain the number of replicas at a certain ratio by setting the replica ratio to avoid excessive waste of resources under the ARCM algorithm. This can help to achieve better access efficiency. Based on the experimental results proposed by Chaing et al, the system will also set the RR threshold value at 30% [11]. Subsequently, the system will calculate the current replica ratio of each file and compare it to the RR threshold value to analyze whether the number of currently requested file replicas is sufficient or not. The results are shown in Table 12. Finally, based on the results of Table 10 to Table 11, the system will decide to add, move or delete the replicas, and the related results are shown in Table 13.

Table 10. The predicting results.

File ID	Atomic decline prediction	Threshold	Popular Trend

A	16.5	21	X
B	30.5	25	O
C	54	50	O
D	25	28	X

Table 11. The analysis results based on the average access time interval.

File ID	History	Now	Popular Trend
A	12.43	12.26	O
B	12.04	11.47	O
C	10.6	10.13	O
D	7.39	8.58	X

Table 12. The result of setting the replicas ratio.

File ID	Number of Replicas	Ratio of Replicas	RR=30%
A	3	$\frac{3}{14}$*100%=21%	$Request_{nr}$<RR
B	3	$\frac{3}{14}$*100%=21%	$Request_{nr}$<RR
C	5	$\frac{5}{14}$*100%=36%	$Request_{nr}$>RR
D	3	$\frac{3}{14}$*100%=21%	$Request_{nr}$<RR

Table 13. The execution results.

File ID	Average Spacing Time	Prediction of the Number of File Accesses	RR=30%	Result
A	X	O		move
B	O	O	$Request_{nr}$<RR	add
C	O	O	$Request_{nr}$>RR	move
D	X	X		delete

So far, the system has finished the process of the data replication strategy. Next, the system will continue to filter the proper service nodes to place the replicas.

Here, the system will calculate the blocking probability of each node through formula (5), where the arrival rate λ is set to 0.2 which refers to the experimental results proposed by Qinsong et al. [21]. For the delay time τ_i, in order to help the system to compare the access performance more clearly, the setting environment will be consistent with the assumption of 1 second upon the request, read, and return of the files respectively. Therefore, the delay time of the homogeneous node is set to 3 seconds. Finally, the related arrival rate and blocking probability of each node are shown in Table 14 and Table 15, and the threshold value of low blocking probability *AvgBP* is 0.00026438.

Table 14. The arrival rate for each node.

Node ID		λ_i
Node 1	File C: ((11/50)/5)*0.2 = 0.0088	λ_1 = 0.0088
Node 2	File A: ((5/21)/3)*0.2 = 0.0159 File C: ((14/50)/5)*0.2 = 0.0112	λ_2 = 0.0271
Node 3	File C: ((10/50)/5)*0.2 = 0.008	λ_3 = 0.008
Node 4	File B: ((6/25)/3)*0.2 = 0.016 File D: ((13/28)/3)*0.2 = 0.031	λ_4 = 0.047
Node 5	File A: ((7/21/3)*0.2 = 0.0222 File C: ((6/50)/5)*0.2 = 0.0048 File D: ((7/28/3)*0.2 = 0.0167	λ_5 = 0.0437
Node 6	File B: ((4/25)/3)*0.2 = 0.0107 File D: ((8/28)/3)*0.2 = 0.019	λ_6 = 0.0297
Node 7	File A: ((9/21)/3)*0.2 = 0.0286 File B: ((15/25)/3)*0.2 = 0.04	λ_7 = 0.0686
Node 8	File C: ((9/50)/5)*0.2 = 0.0072	λ_8 = 0.0072

Table 15. The blocking probability for each node.

Node ID	BP_i
Node 1	$\frac{(0.0088*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0088*3)^k}{k!}]^{-1}$ = 0.000003
Node 2	$\frac{(0.0271*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0271*3)^k}{k!}]^{-1}$ = 0.0000826
Node 3	$\frac{(0.008*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.008*3)^k}{k!}]^{-1}$ = 0.0000022
Node 4	$\frac{(0.047*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.047*3)^k}{k!}]^{-1}$ = 0.0004058
Node 5	$\frac{(0.0437*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0437*3)^k}{k!}]^{-1}$ = 0.0003294
Node 6	$\frac{(0.0297*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0297*3)^k}{k!}]^{-1}$ = 0.0001078

Node 7	$\frac{(0.0686*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0686*3)^k}{k!}]^{-1}$ = 0.0011826
Node 8	$\frac{(0.0072*3)^3}{3!}[\sum_{k=0}^{3}\frac{(0.0072*3)^k}{k!}]^{-1}$ = 0.0000016
AvgBP	(0.000003+0.0000826+0.0000022+0.0004058+0.0003294+0.0001078+0.0011826+0.0000016)/8 = 0.00026438

After calculating the average *AvgBP* value, the system can get the set of low blocking nodes whose BP values are smaller than the average from the system nodes. Subsequently, in addition to avoiding end users from choosing the congested nodes for services, the system must also consider whether tasks are evenly distributed. At this time, the system will also compare the queue space status of each node. Here, the node with low blocking probability and has the largest queue space will be the best service node. The results are shown in Table 16.

Assume that File A is currently predicted to be popular and has reached the condition for adding new replicas, the system will find out the set of low-probability blocked nodes whose blocking probability are lower than *AvgBP* (Node 1, Node 2, Node 3, Node 6, Node 8) and have the most queue space (Node 1, Node 3, Node 8) to place the replicas for File A. When more than two nodes meet the above conditions at the same time, the node with the smallest blocking probability will be selected as the serving node. In the overall example, a new replica for File A will be added to Node 8 for further services under the MEC environment.

Table 16. The blocking probability and queue space of each node.

Node ID	BP_i	RQ_i
Node 1	BP_1 = 0.000003	Free space of RQ = 4
Node 2	BP_2 = 0.0000826	Free space of RQ = 3
Node 3	BP_3 = 0.0000022	Free space of RQ = 4
Node 4	BP_4 = 0.0004058	Free space of RQ = 3
Node 5	BP_5 = 0.0003294	Free space of RQ = 2
Node 6	BP_6 = 0.0001078	Free space of RQ = 3
Node 7	BP_7 = 0.0011826	Free space of RQ = 3
Node 8	BP_8 = 0.0000016	Free space of RQ = 4

In this paper, the adaptive replica configuration mechanism which is called ARCM has been proposed to improve the access efficiency load balancing under the MEC environment. The main idea is to calculate the popularity of the file to decide whether to replicate the file or not and then the system will select a proper node to place the replicas. The related experiment and analyses will be given in the next section.

5. Experiments and Analysis

In this section, the environment for experiments, results and related analysis will be given to prove the performance of the ARCM mechanism.

5.1 Environment for Experiments

In this paper, the dynamic configuration algorithm experiments are simulated by Dev C++ under Windows environment. Furthermore, the CDRM, DDRA, PARM, ADRM, and Random algorithms are invoked to compare the performance in the Cloudsim for simulation. The experiments simulate the number of nodes from loosely environments (20 nodes) to densely environments (100 nodes) in the small MEC environment. Also, in the experiments, we simulate assigning different workloads (low workload, medium workload, and high workload) for nodes. Besides, in the experiments, the number of nodes is set to 20, 40, 60, and 100 respectively. In the low workload environment, 1000 tasks are assigned per cycle. In the medium workload environment, 5000 tasks are assigned per cycle, and 10,000 tasks are assigned per cycle in the high workload environment. To observe the task allocation status of each algorithm, the node capacity is set to be homogeneous, and each node has 5 task queue spaces. The remaining parameter settings are shown in Table 17 to Table 19. Finally, the proposed ARCM algorithm will compare to other algorithms through node utilization, Mean Average Deviation (MAD), the number of replicas, throughput, and completion time.

Table 17. The parameters for the experiment.

Item	**Value**
Number of Nodes	20、40、60、100
Task Queue	5
Workload	1000 Low Workload 5000 Medium Workload 10000 High Workload
File Type	A~E
Short-Term Storage	3 replicas
Long-Term Storage	5 replicas
Update Frequency	5 minutes
Ratio of Replicas	30%

Table 18. Cloudsim Parameter Settings.

Item	**Value**

VM	20~100
PEs	1
MIPS	1000(MAX)
RAM	512MB
Size	10000MB
BW	1000

Table 19. Experiment tools.

Tools	**Usage**
Dev C++	Dynamic replica configuration algorithm
Cloudsim	Build the service environment and experiment simulation
Microsoft Office Excel 2016	Record experiment results

5.2 Performance Analysis

In the sub-section, we will analyze the performance of each algorithm based on the different environments including loosely, ordinary and densely environments. Furthermore, we will also observe and analyze the performance and usage status of each algorithm under different workloads. The results are shown as follows.

5.2.1 Loosely Environment {Node=20; Workload=1000~10000}

Figure 4, Figure 5 and Figure 6 show the simulation results of low to high workload in the loosely environment when the number of nodes is set to 20. Here, the Random algorithm does not dynamically configure the number of replicas as the environment changes. Therefore, the node which owned the replica originally will have a higher node utilization rate. On the contrary, the node that does not been assigned with the replica will stay idle. For the CDRM algorithm, it also has a high utilization rate on certain nodes. The main reason is that nodes with better access efficiency are more likely to be assigned more tasks. Besides, for the PARM and ADRM algorithm, the simulation will distribute the replica randomly to understand the difference with other algorithms even when the PARM and ADRM algorithms are lack of replica configuration scheme. Finally, for the DDRA and ARCM algorithm, both of them consider the blocking probability and the number of queued nodes while assigning the tasks, hence these two algorithms are more balanced than CDRM in terms of node utilization.

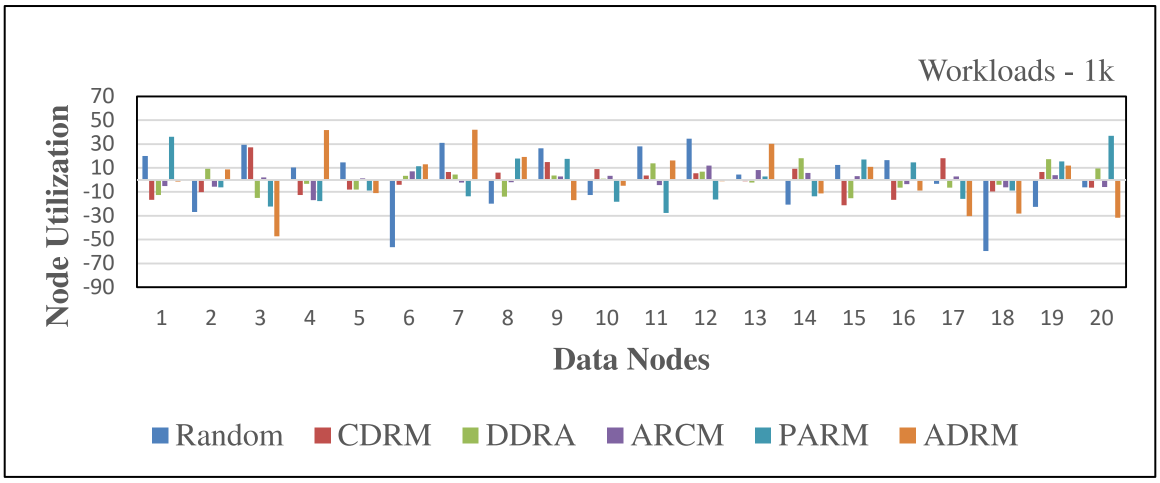


Figure 4. Node Utilization for 1000 workloads, Node = 20

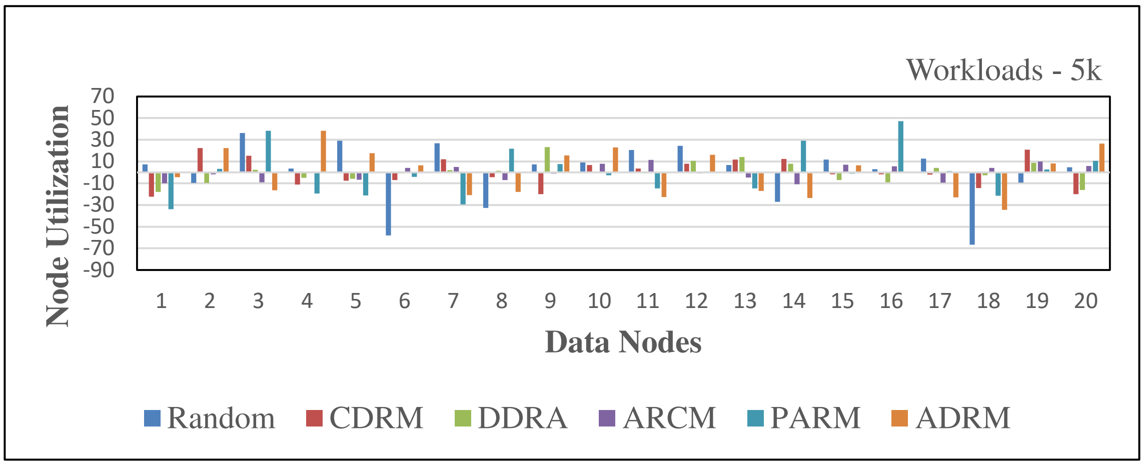


Figure 5. Node Utilization for 5000 workloads, Node = 20

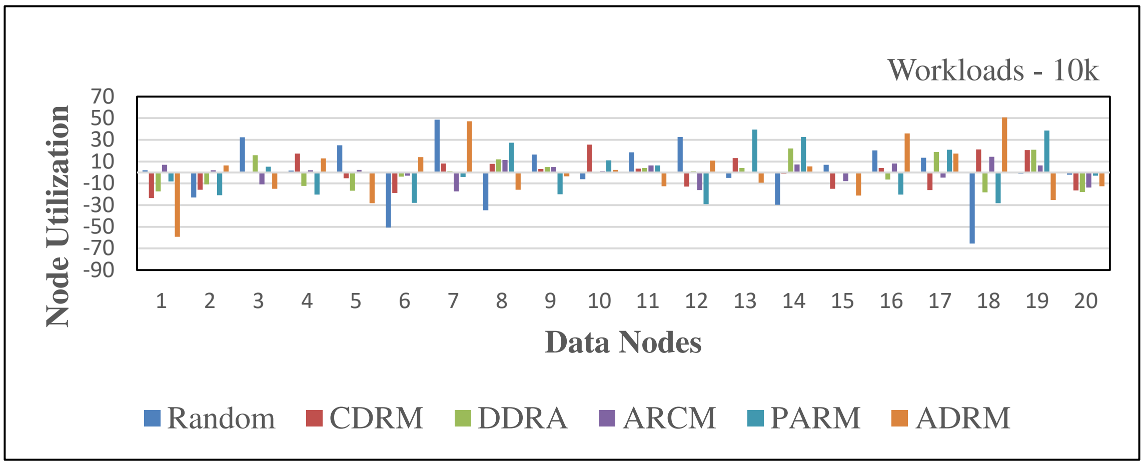


Figure 6. Node Utilization for 10000 workloads, Node = 20

To help to understand the status of the loading of the system clearly, we calculate MAD value through formula (6). In the formula, n represents the total number of nodes, and u_i represents the utilization rate of the nodes. When the MAD value is getting higher, it means that task distribution is uneven in the overall system. In contrast, the lower the MAD value, the better load balancing the system can achieve. From Figure 7 to Figure 9, it can be seen that the ARCM algorithm has the best load balancing. This is because that the ARCM algorithm considers the blocking probability and queuing space of each node while replacing the replicas. This can help to allocate the workload more balancing. In other words, the ARCM algorithm can adjust the number of replicas dynamically by predicting the popularity of files. When the file is analyzed as popular, the system can quickly process the workload generated by the file which has increasing popularity by adding more replicas in advance. This is the reason why the ARCM algorithm can allocate system nodes more balanced.

$$\sum_{i=1}^{n} |u_i - average(u_i)| \qquad (6)$$

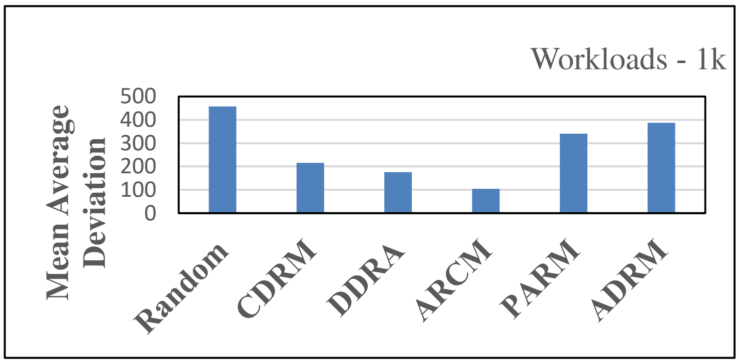


Figure 7. Mean Average Deviation for 1000 workloads, Node = 20

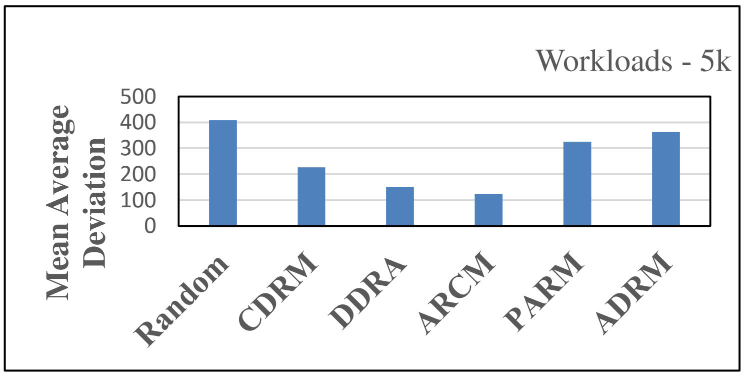


Figure 8. Mean Average Deviation for 5000 workloads, Node = 20

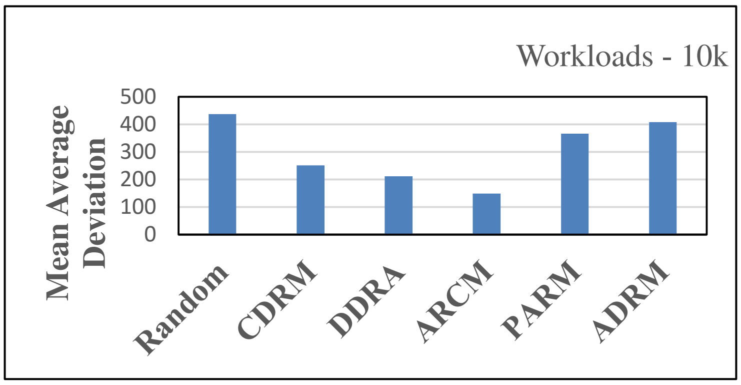


Figure 9. Mean Average Deviation for 10000 workloads, Node = 20

In Figure 10, we can see that the non-dynamic Random algorithm will not adjust the replicas as the environment changes under the low workload environment, and the number of replicas will remain at three. Besides, the default number of replicas in the CDRM algorithm is 1. Although the availability of replicas will be adjusted while executing the algorithm, the access efficiency will decrease due to the insufficient availability of replicas in the beginning. For the ARC and the ADRM algorithms, the average number of replicas been used are lower than the results of the DDRA algorithm. This is because that the replicas configuration settings of ARCM and ADRM are stricter than those of DDRA. Hence, the number of replicas is relatively stable. Furthermore, the ADRM algorithm ha better results in terms of the number of replicas than the DDRA algorithm, however, the performance of the node utilization rate is poor to the DDRA algorithm due to the incorrect replica configuration method. Finally, the PARM algorithm does not set up a stop-loss point for generating replicas, and this results in unlimited additions of popular replicas.

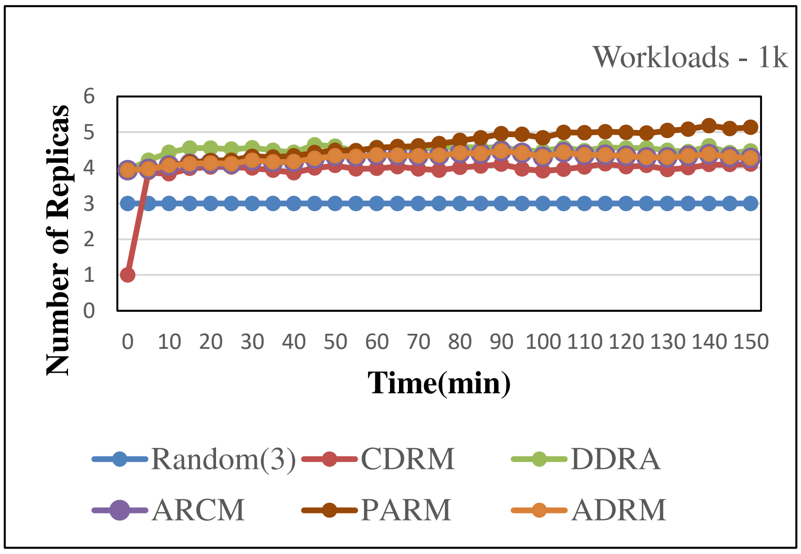


Figure 10. Number of Replicas for 1000 workloads, Node = 20

When the workload increases to 5,000, we can observe that the ARCM and DDRA algorithms become comparable. This is because that the DDRA algorithm only applies a single factor to determine the popularity. This factor is when the number of file access exceeds the average value, then the file will be judged as a popular file. Under such a circumstance, the system will add new replicas for this popular file. In contrast, the replicas will be deleted. Hence, the DDRA algorithm will easily lead the number of replicas to keep changing in the system. By comparison, the ARCM algorithm can more accurately predict environmental changes than DDRA. The related results can be seen in Figure 11

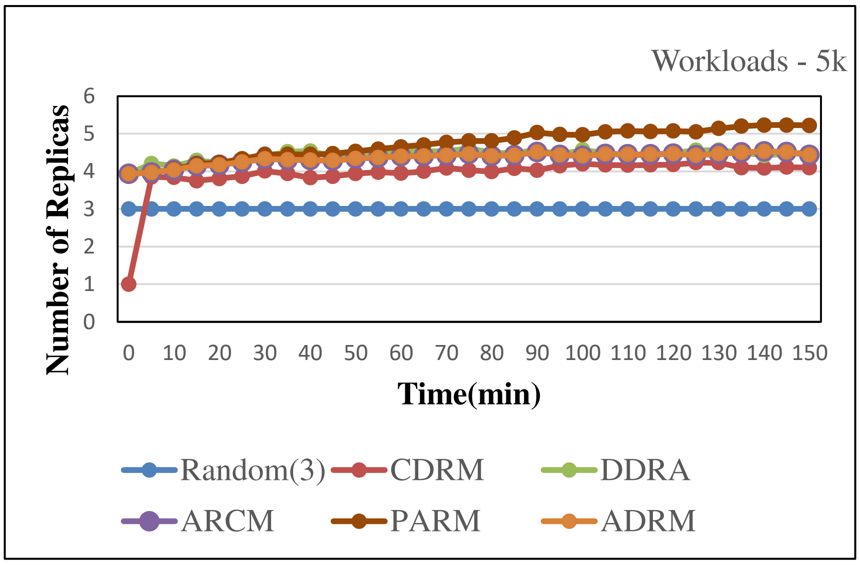


Figure 11. Number of Replicas for 5000 workloads, Node = 20

In Figure 12, it can be seen that the number of replicas used by the ARCM algorithm is less than that of DDRA at the beginning when the workload increases to 10,000. As time goes by, the number of replicas used by these two algorithms are almost the same. This is because the number of users' requests and the workload are getting higher. Thus, the differences in the popularity of each file will become more obvious. Therefore, the results of the popularity of the two algorithms are more likely to have the same situation.

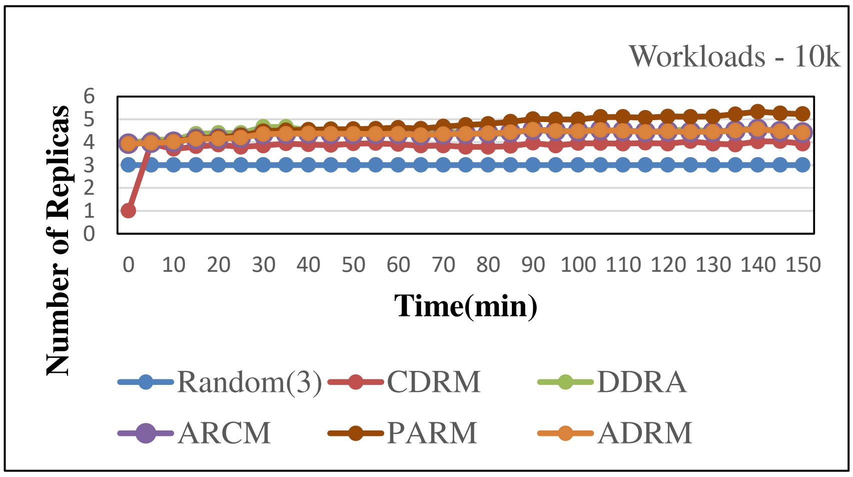


Figure 12. Number of Replicas for 10000 workloads, Node = 20

Figure 13 and Figure 14 show the simulation results of throughput and mean job time of node 20 from a low workload to a high workload environment. The results show that the ARCM algorithm has the best results in terms of throughput and completion time. As the MAD results shown in Figure 7 to Figure 9, it is known that the ARCM algorithm has better performance in terms of load balancing and can distribute the workload more evenly. Furthermore, the ARCM algorithm can help to improve the problem of delaying the overall working time by avoiding allocate the tasks on the specific nodes concentratedly. Finally, the experiment results show that the ARCM algorithm can improve the performance of throughput and the completion time for the loosely MEC environments.

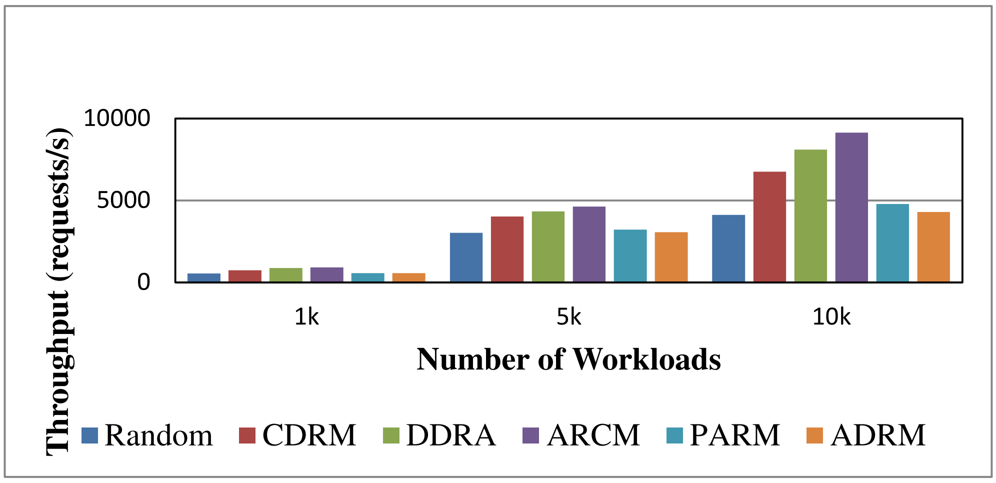


Figure 13. The throughput results of 20 nodes

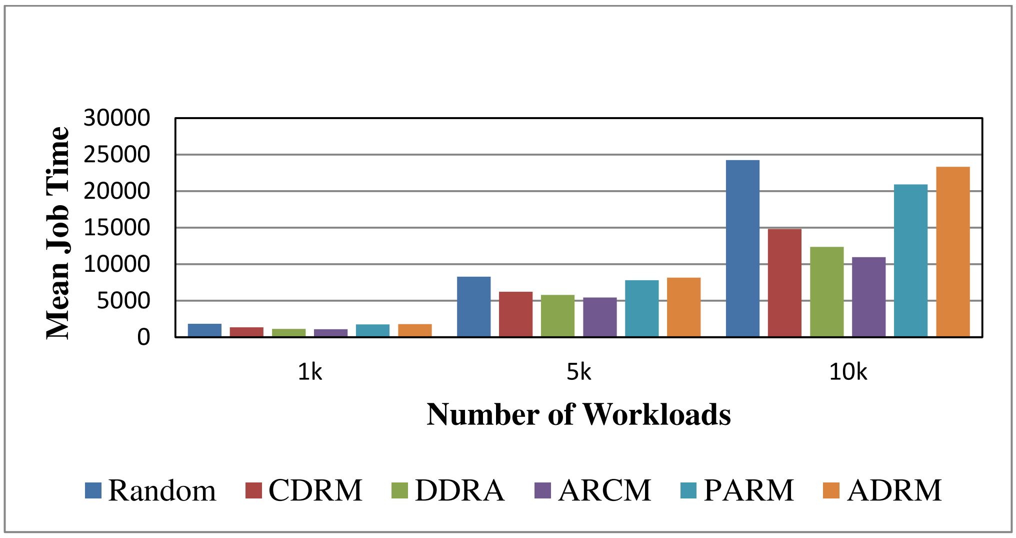


Figure 14. The mean job time results of 20 nodes

5.2.2 Ordinary environment {Node=40, 60;Workload=1000~10000}

In this sub-section, the performance and analysis results under the ordinary MEC environment when the workload is increased from 1,000 to 10,000 and the number of nodes is 40 or 60 respectively will be given, and the results can also be seen from Figure 15to Figure 36.

According to the results of Figure 15 to Figure 20 and Figure 26 to Figure 31, we can observe that since the non-dynamic Random algorithm only allocates a fixed number of replicas to nodes, and this Random algorithm also has the problem of uneven task allocation. The dynamic replica configuration CDRM algorithm provides faster services to end-users by assigning tasks to the nodes with the lowest blocking probability. Unfortunately, this cause that the nodes with better computing capabilities will always receive more task requests than others, and the system cannot reach the goal of load balancing. For the PARM and the ADRM algorithm, both of them also cannot reach load balancing due to the lack of proper replica configuration methods. As for the ARCM algorithm, it can predict the future trend for the files by analyzing the historical data. When a file is predicted as a popular trend, the system will add more replicas to the nodes and can distribute the workload balancing. Hence, the ARCM algorithm can achieve a better load balance.

Besides, the CDRM algorithm initially sets one replica for each file, which may easily lead to insufficient file availability in the early stage. For the ARCM algorithm, it has the procedure to classify the task at the beginning. If the files are in the type of short-term storage, the system will allocate three replicas, and if the files are in the type of long-term storage, five replicas will be allocated. Hence, in the low workload environments, the ARCM algorithm requires fewer replicas than the DDRA algorithm. In the middle workload environment, the performance of the ARCM and the DDRA algorithms are almost the same. Finally, in the high workload environment, the number of replicas of the ARCM and the DDRA algorithm is almost the same after the cycle of 60 minutes. The related results can be seen in Figure 21 to Figure 23 and Figure 32 to Figure 34.

According to the results shown in Figure 24 to Figure 25 and Figure 35 to Figure 36, when the MAD is getting lower, the system has better load balancing since the working capacity of the nodes is set to be the same. Therefore, it can be observed from the simulation results that compared to the DDRA and the CDRM algorithms, the ARCM algorithm can evenly distribute the workload and can avoid allocating the tasks on the specific nodes concentratedly. Therefore, ARCM has better throughput and completion time results.

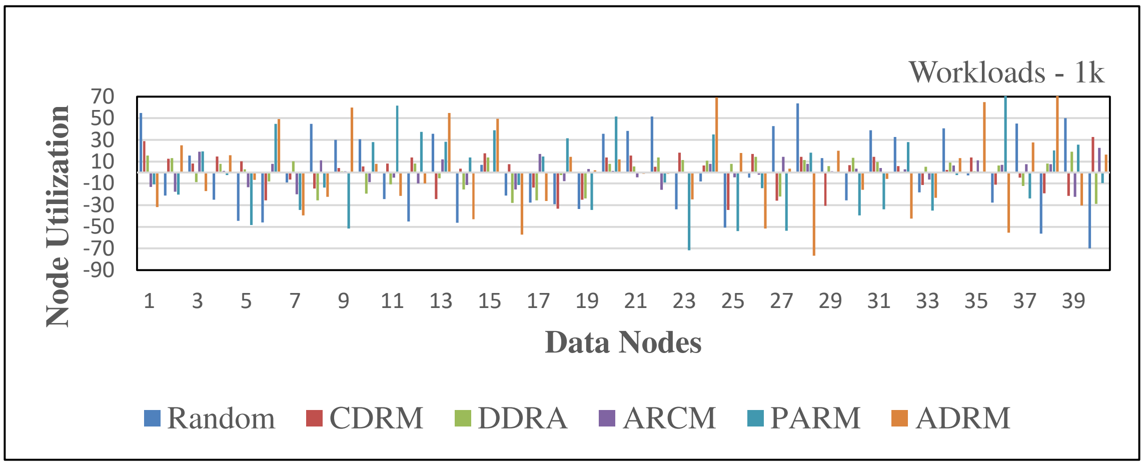


Figure 15. Node Utilization for 1000 workloads, Node = 40

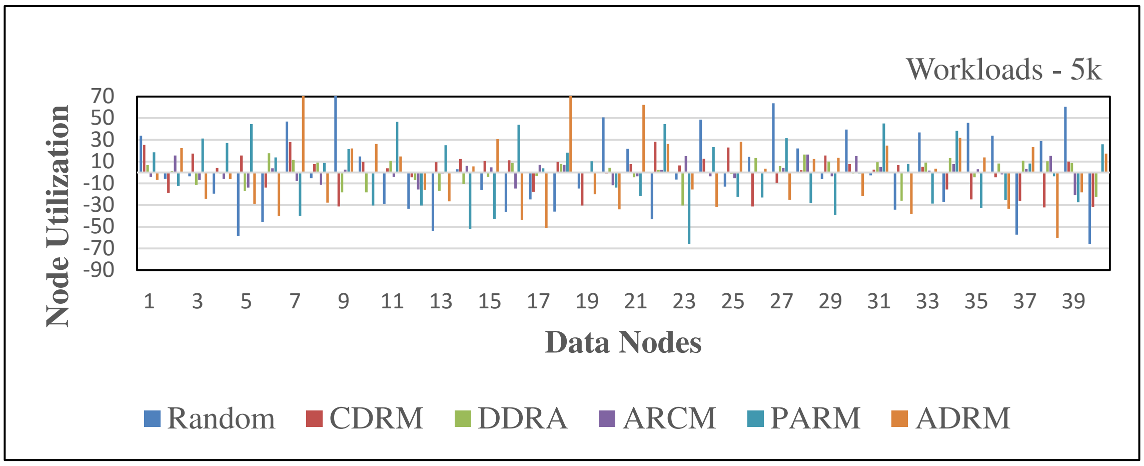


Figure 16. Node Utilization for 5000 workloads, Node = 40

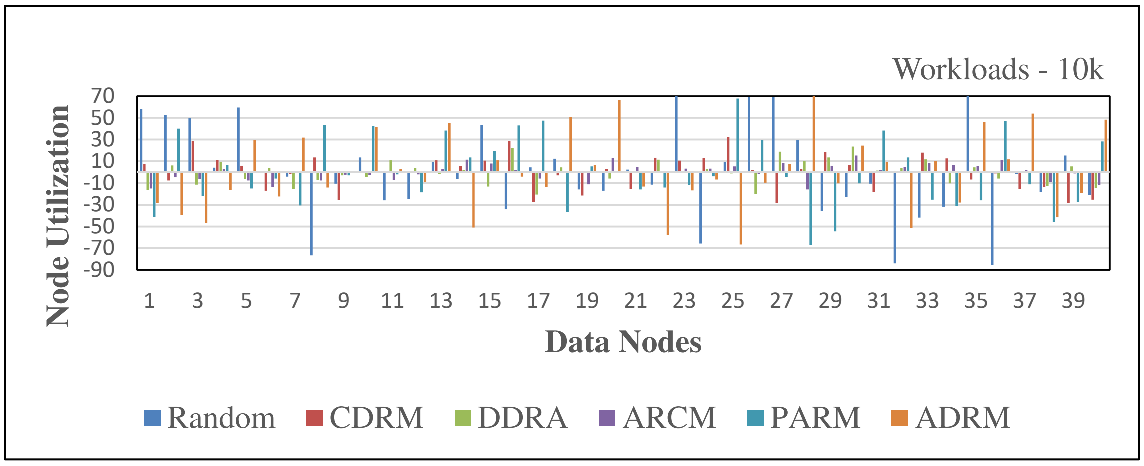


Figure 17. Node Utilization for 10000 workloads, Node = 40

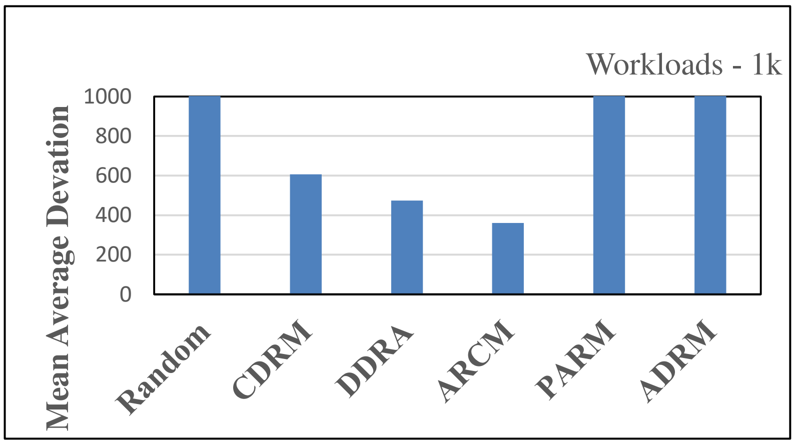


Figure 18. Mean Average Deviation for 1000 workloads, Node = 40

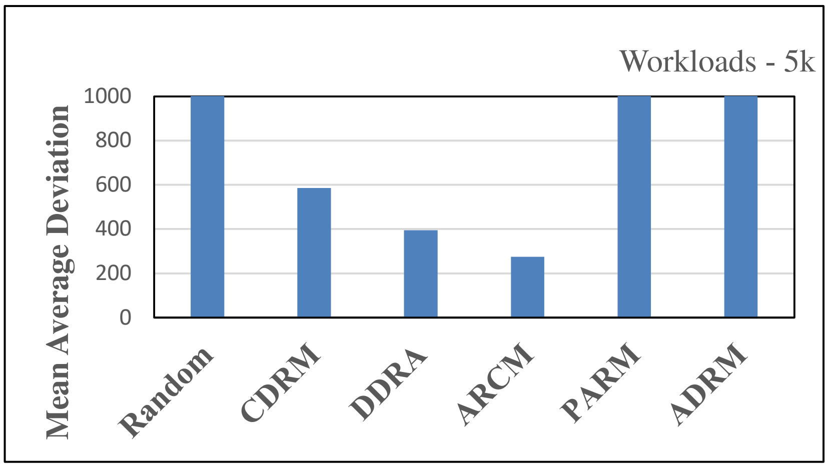


Figure 19. Mean Average Deviation for 5000 workloads, Node = 40

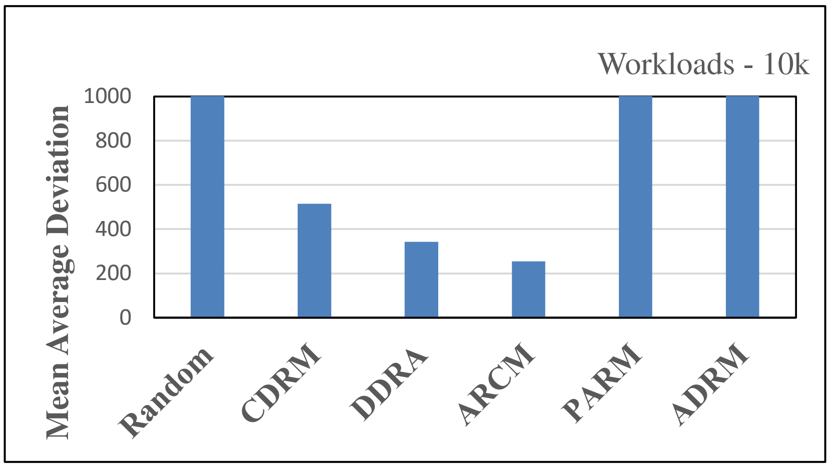


Figure 20. Mean Average Deviation for 10000 workloads, Node = 40

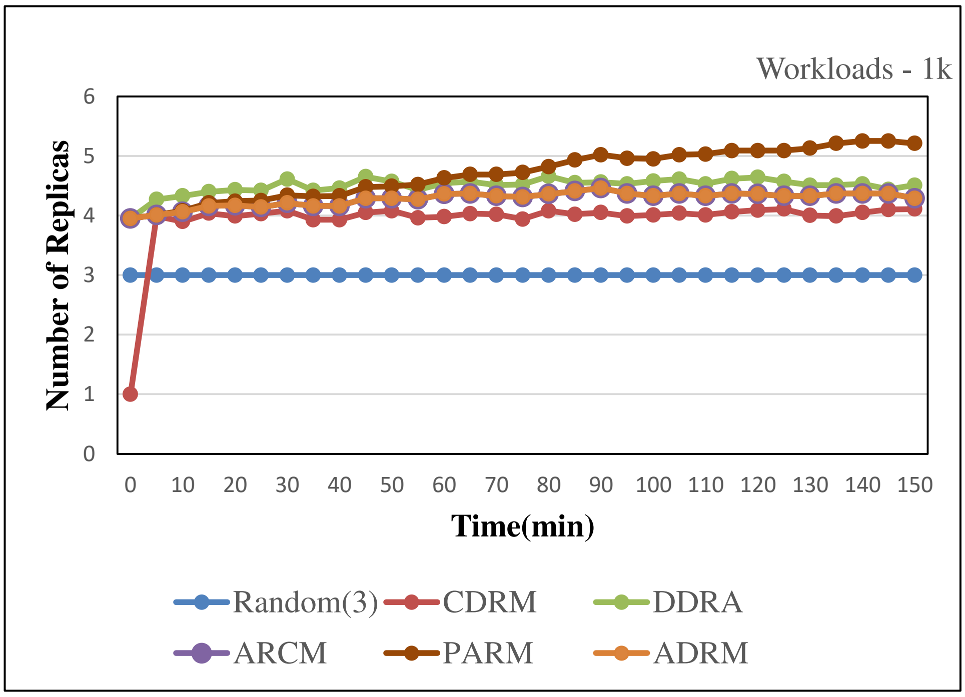


Figure 21. Number of Replicas for 1000 workloads, Node = 40

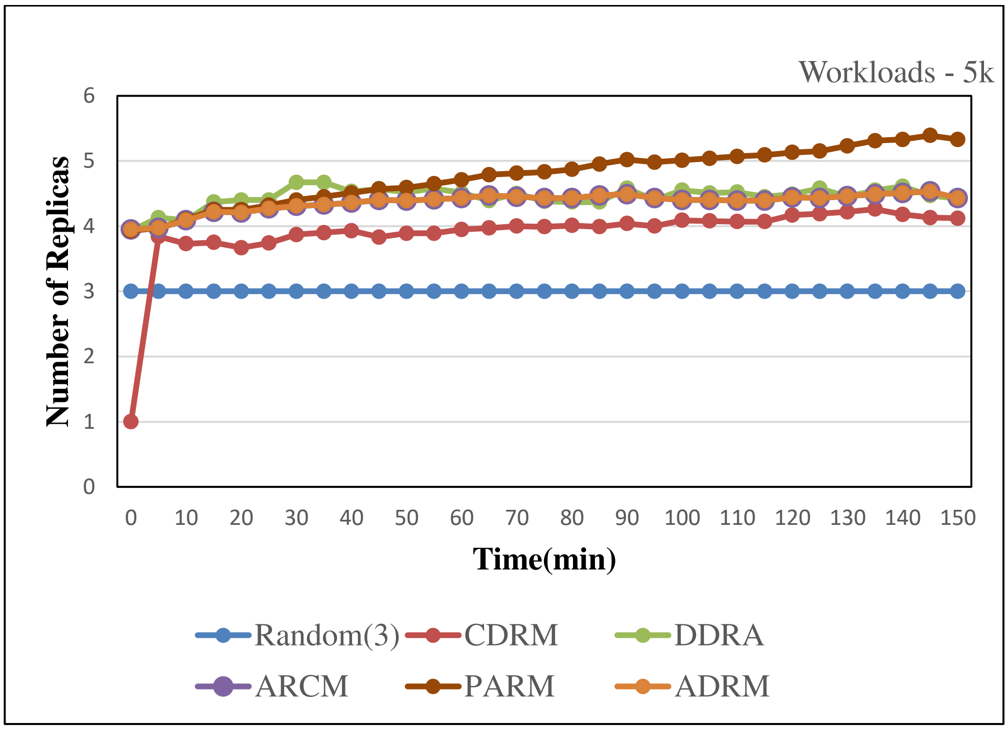


Figure 22. Number of Replicas for 5000 workloads, Node = 40

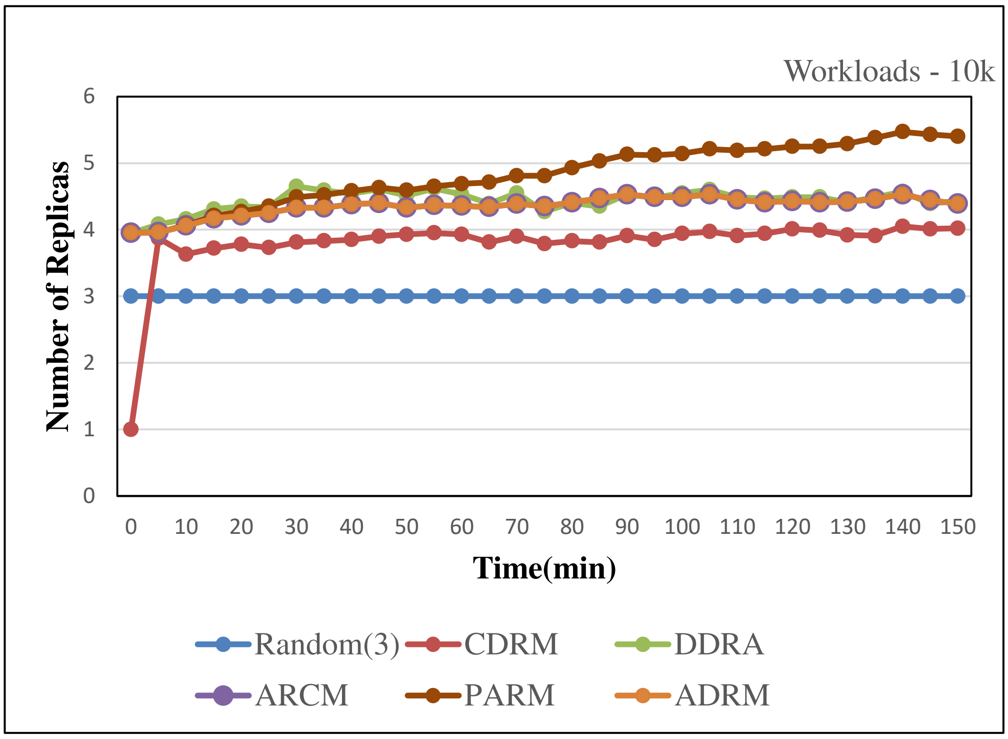


Figure 23. Number of Replicas for 10000 workloads, Node = 40

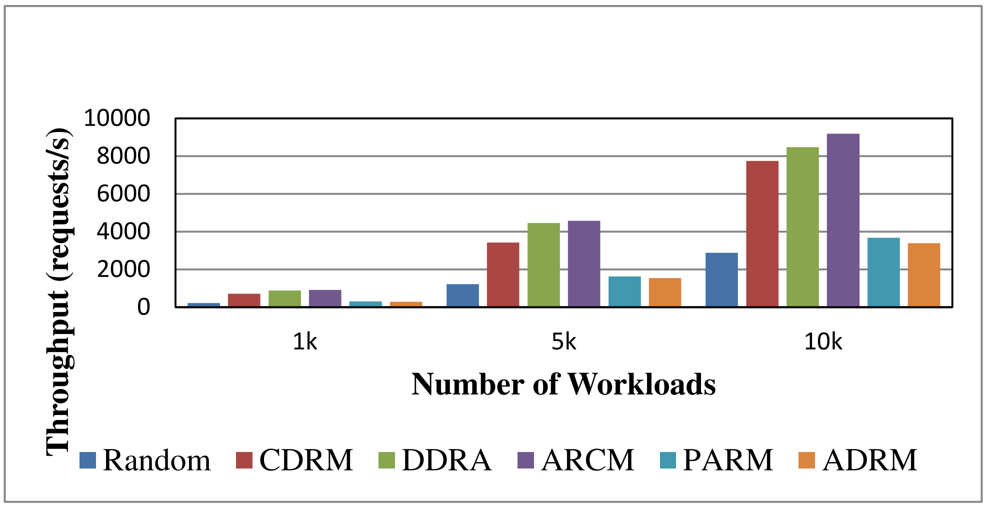


Figure 24. Throughput in 40 nodes

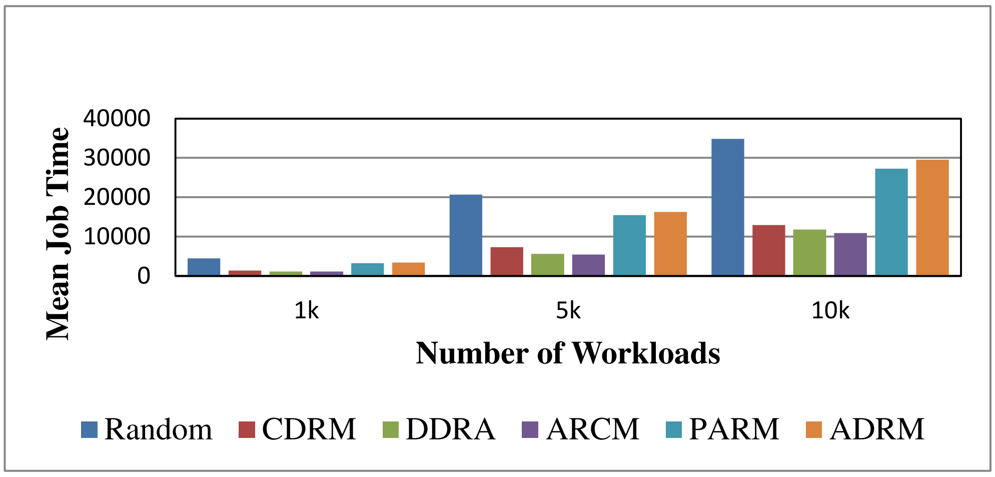


Figure 25. Mean Job Time in 40 nodes

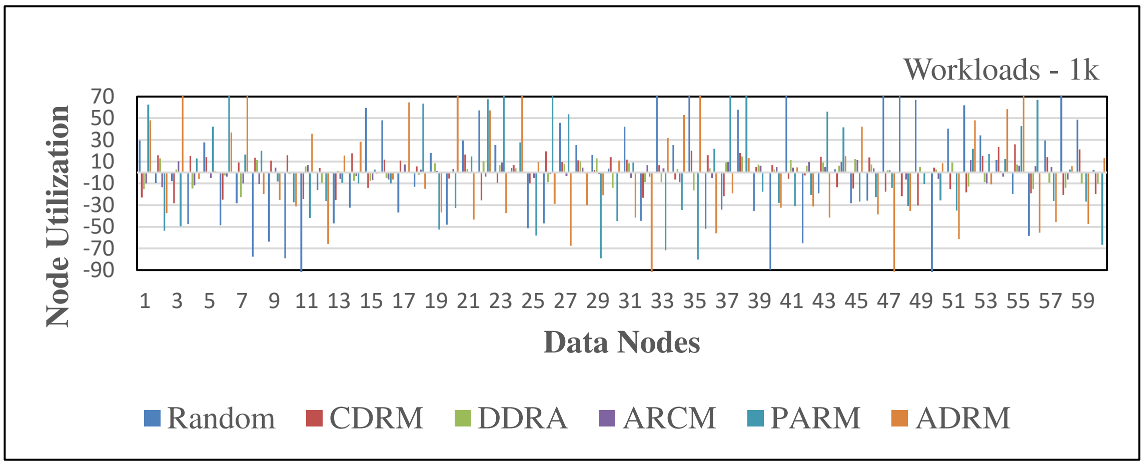


Figure 26. Node Utilization for 1000 workloads, Node = 60

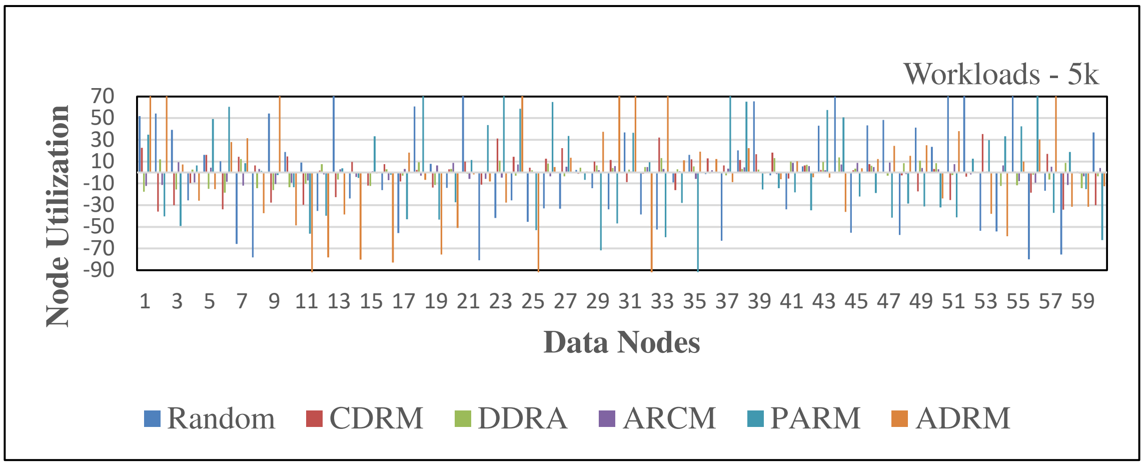


Figure 27. Node Utilization for 5000 workloads, Node = 60

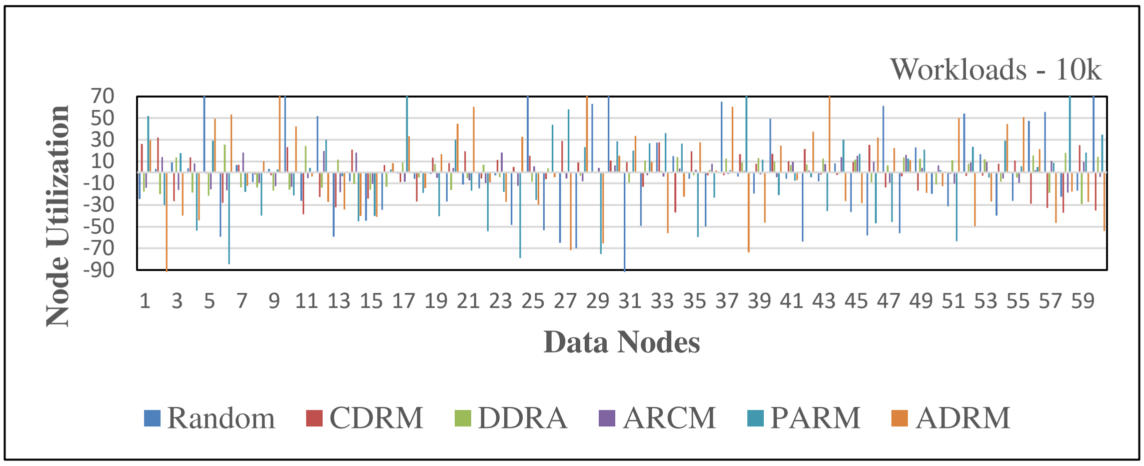


Figure 28. Node Utilization for 10000 workloads, Node = 60

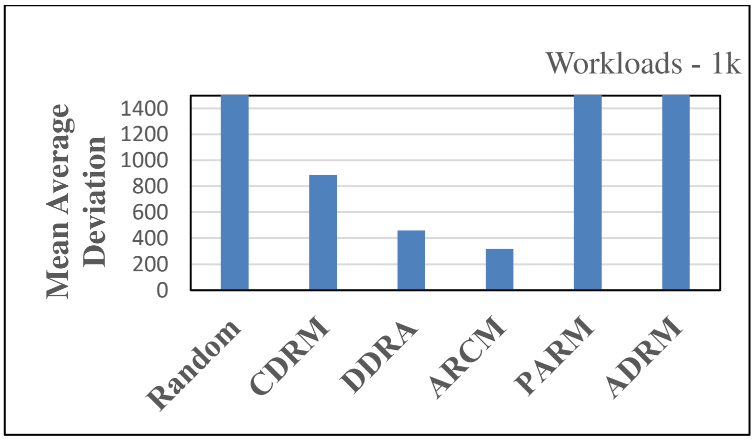


Figure 29. Mean Average Deviation for 1000 workloads, Node = 60

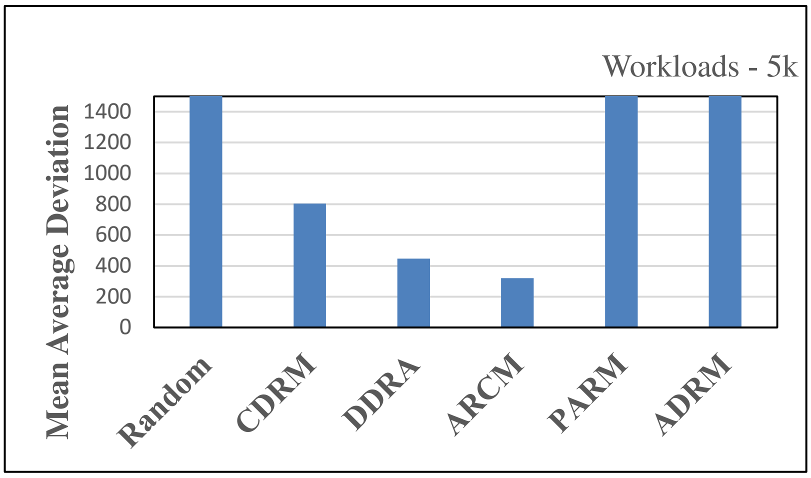


Figure 30. Mean Average Deviation for 5000 workloads, Node = 60

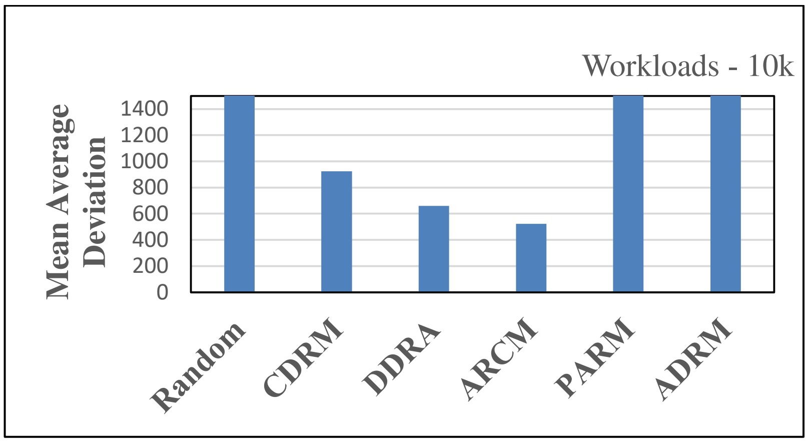


Figure 31. Mean Average Deviation for 10000 workloads, Node = 60

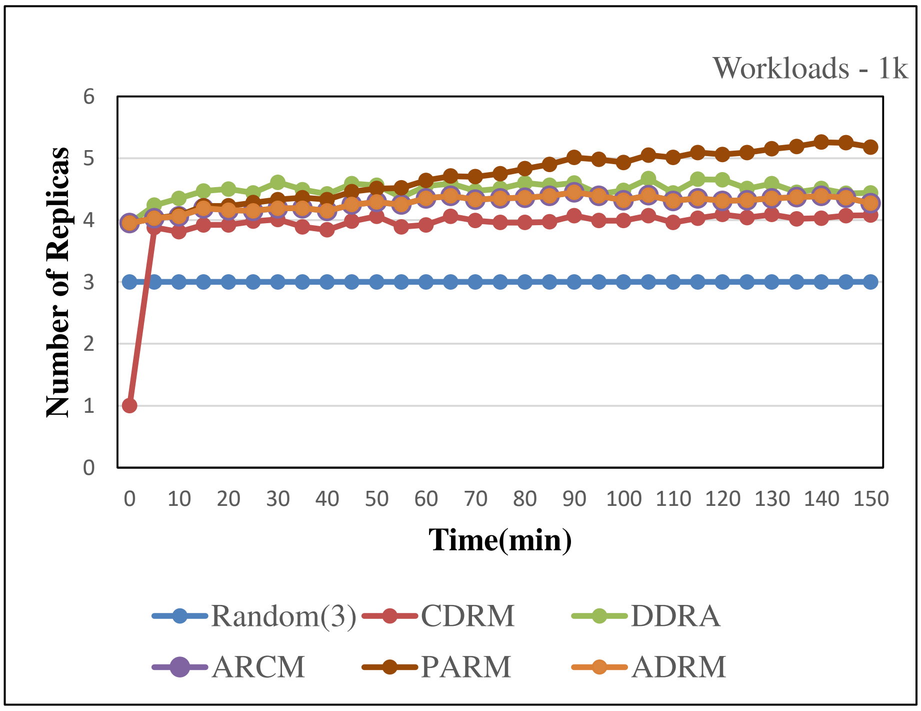


Figure 32. Number of Replicas for 1000 workloads, Node = 60

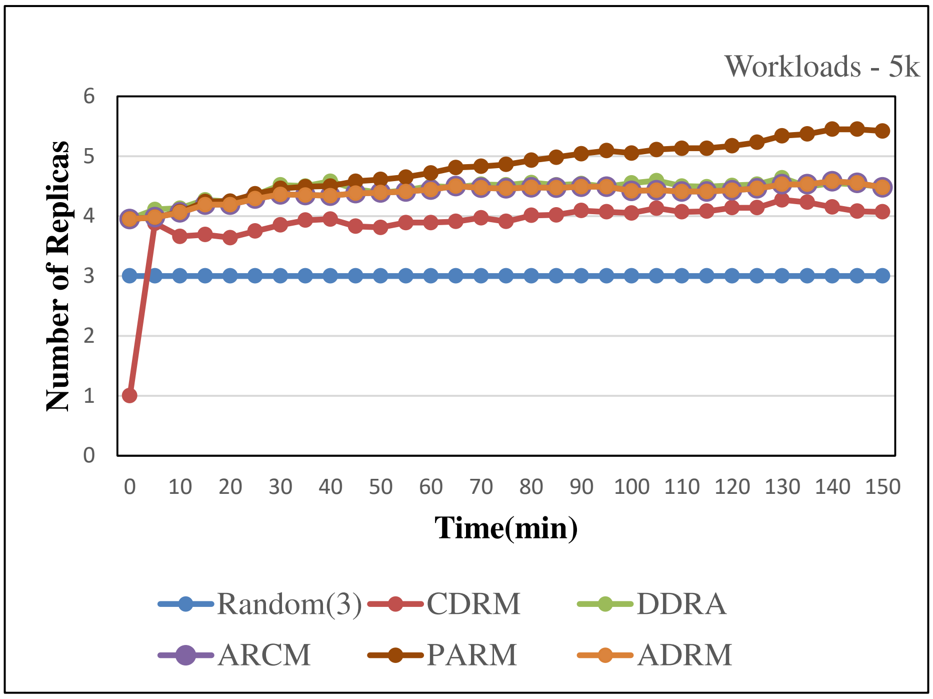


Figure 33. Number of Replicas for 5000 workloads, Node = 60

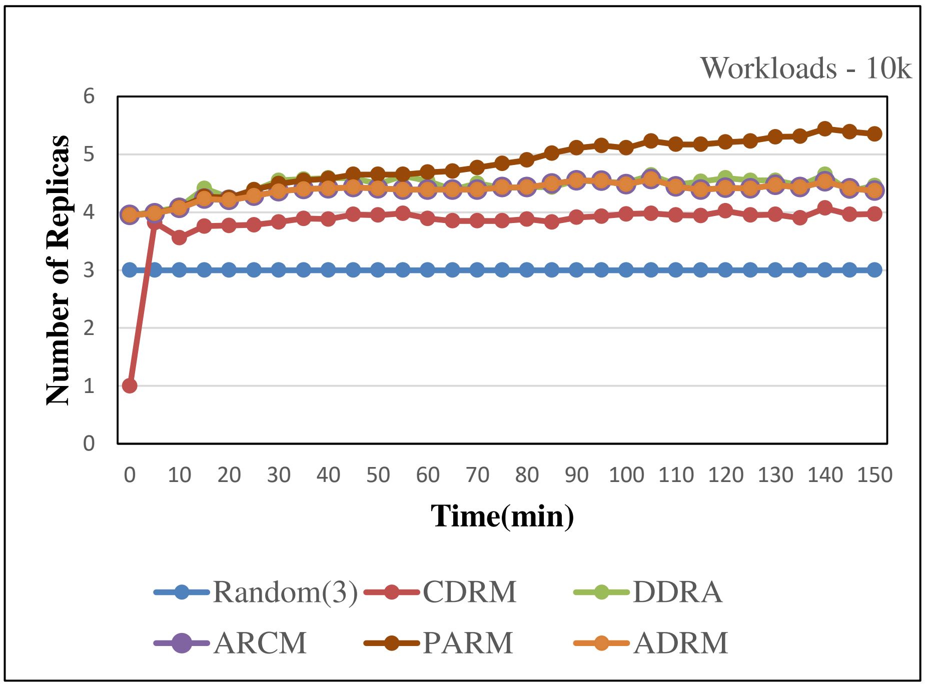


Figure 34. Number of Replicas for 10000 workloads, Node = 60

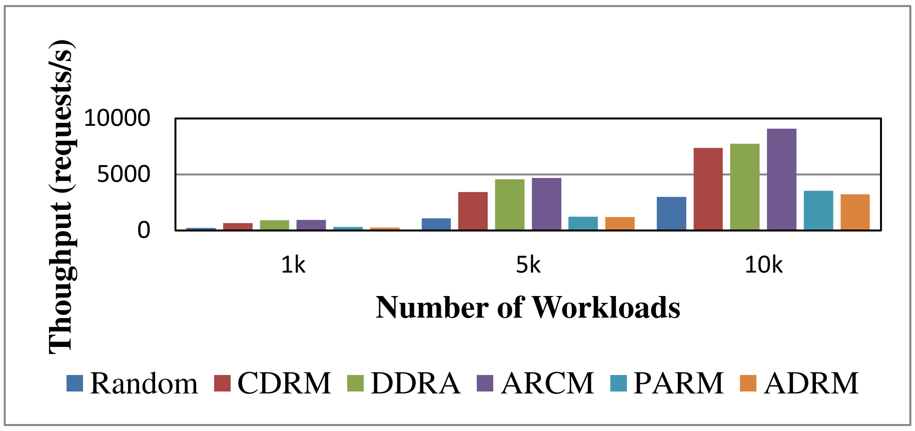


Figure 35. Throughput in 60 nodes

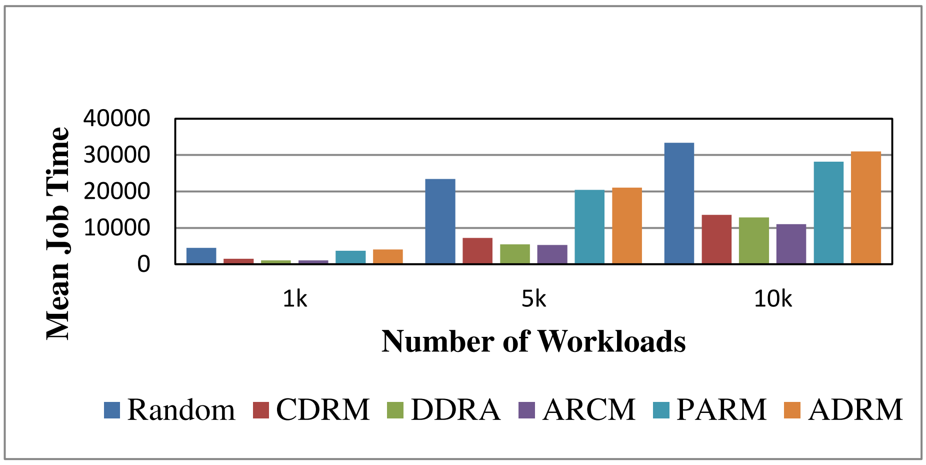


Figure 36. Mean Job Time in 60 nodes

5.2.3 Densely environment{Node=100; Workload=1000~10000}

Figure 37 to Figure 47 are the results of the algorithms under the densely MEC environment. In this circumstance, with the number of the workload is getting higher, the number of replicas for the CDRM algorithms tends to increase. Due to the number of nodes is more than other algorithms, more replicas must be allocated to serve the requests. The ARCM algorithm uses the atomic decline method to predict the popularity of the files, and pre-configures popular replicas for the files to disperse the requested workloads. Hence, it can achieve high throughput and low completion time. For the ARCM and the DDRA algorithms, they control the number of replicas by setting the ratio of RR value to avoid generating too many replicas to occupy system space. Thus, these two algorithms can also achieve better results in terns of throughput and the completion time.

Based on the results shown in Figure 40 to Figure 42, we can observe that the non-dynamic Random has a higher MAD value than other algorithms. This means that it cannot improve the performance of the system configuration.

Furthermore, Figure 43 to Figure 45 shows the results of the status of the number of replica among each algorithm. Here, the CDRM algorithm only allocates one number of replica for the files at the beginning, and this will cause the problem of the insufficient number of available replicas in the early stage for the system. Hence, this algorithm is not able to distribute the workload of the replicas effectively and timely. For the DDRA algorithm, it improves the problem of setting the number of initial replicas and referred to the remaining queue space to achieve load balancing when replicas were configured. Hene, the DDRA algorithm has better performance than the CDRM algorithm. Besides, the PARM algorithm does not set the limitation of the number of replicas, and this will easily cause to add the replicas excessive. The ADRM algorithm improves the problem of DDRA on generating the popular replicas, hence, the ADRM algorithm uses less number of replicas than the DDRA algorithm. However, the ADRM algorithm lacks a proper replica configuration method, hence, the performance of the node utilization, throughput, and the completion time is slightly worse than the DDRA algorithm. Finally, the proposed ARCM algorithm requires more number of replicas than the DDRA algorithm, but it still has better performance in terms of the node utilization, MAD, throughput, and the completion time than other algorithms. This is because the ARCM algorithm adds the concept of prediction, and predicts the future status for each file by analyzing the trend of historical data. This can help to respond to the changes in the system. Hence, the overall performance of the proposed ARCM algorithm can get better performance for the MEC environment.

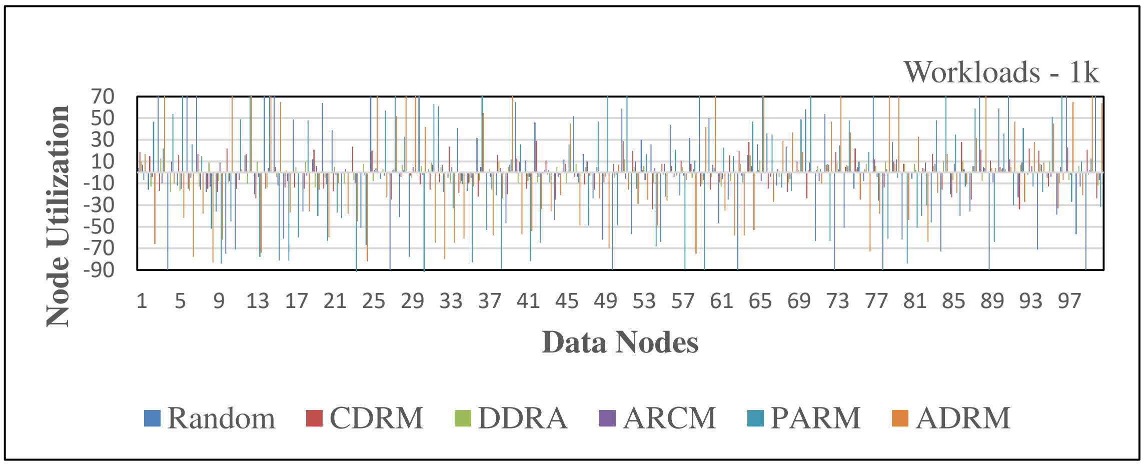


Figure 37. Node Utilization for 1000 workloads, Node = 100

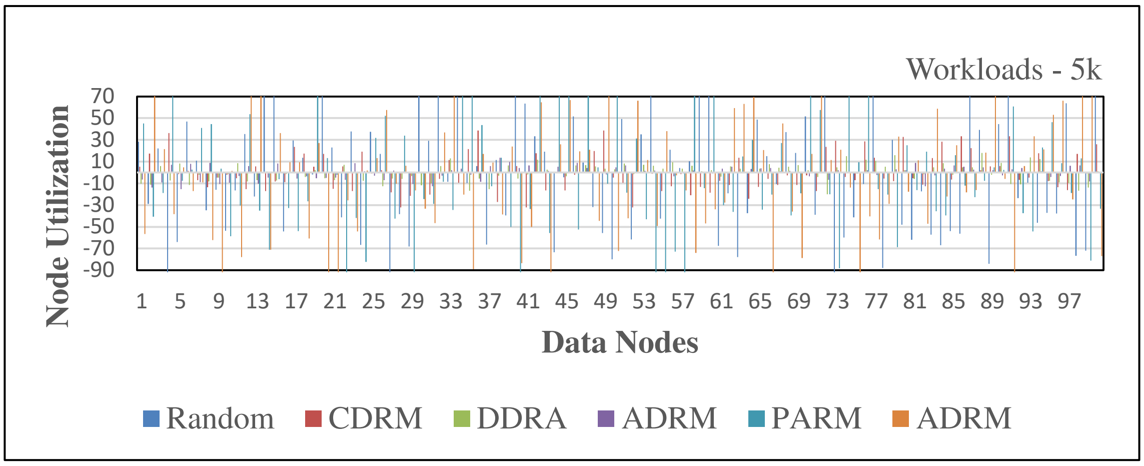


Figure 38. Node Utilization for 5000 workloads, Node = 100

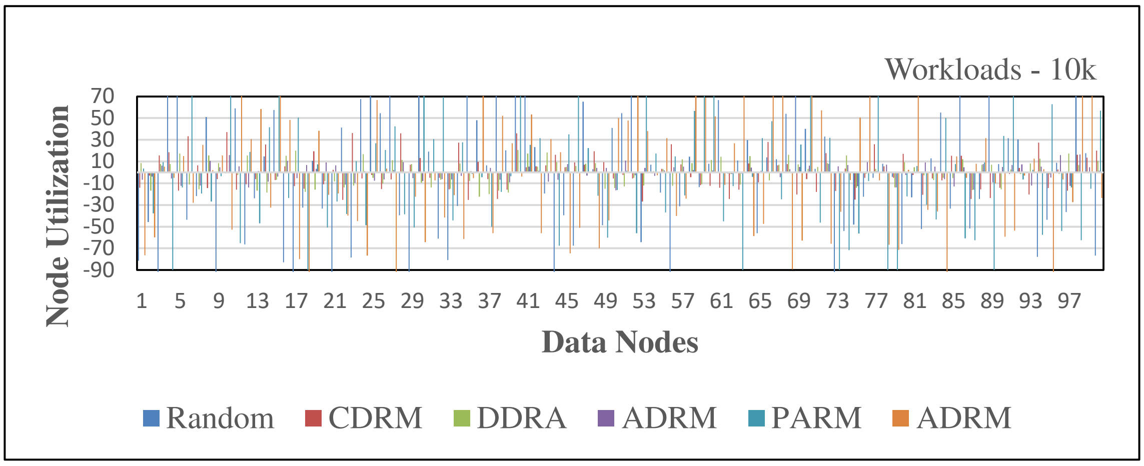


Figure 39. Node Utilization for 10000 workloads, Node = 100

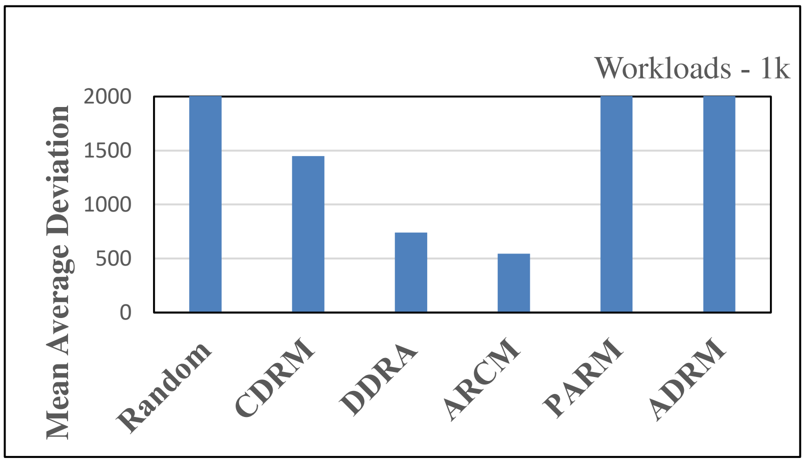


Figure 40. Mean Average Deviation for 1000 workloads, Node = 100

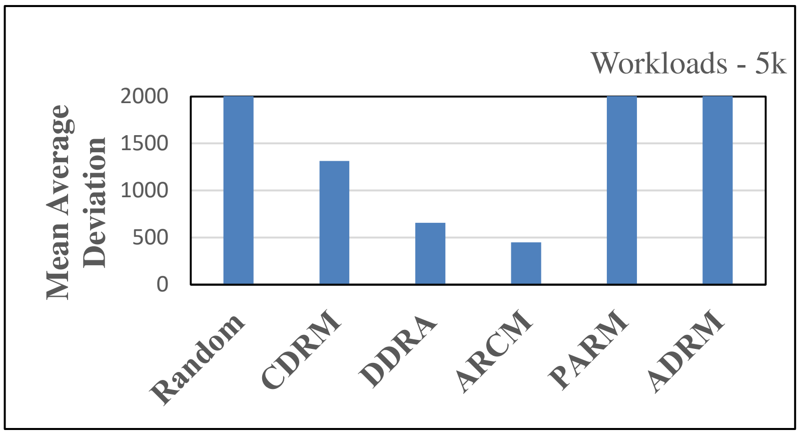


Figure 41. Mean Average Deviation for 5000 workloads, Node = 100

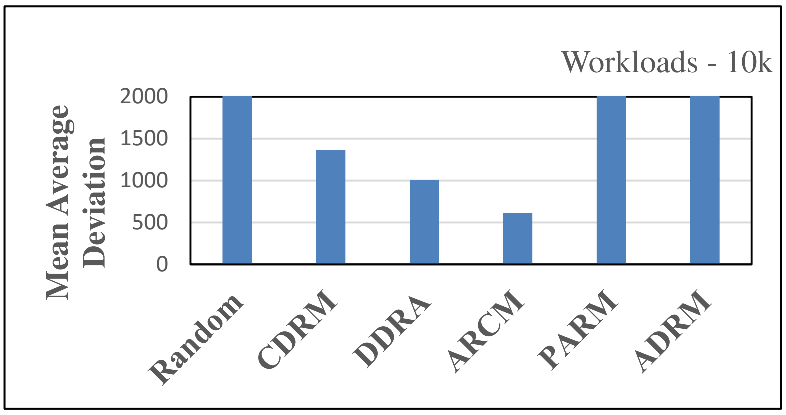


Figure 42. Mean Average Deviation for 10000 workloads, Node = 100

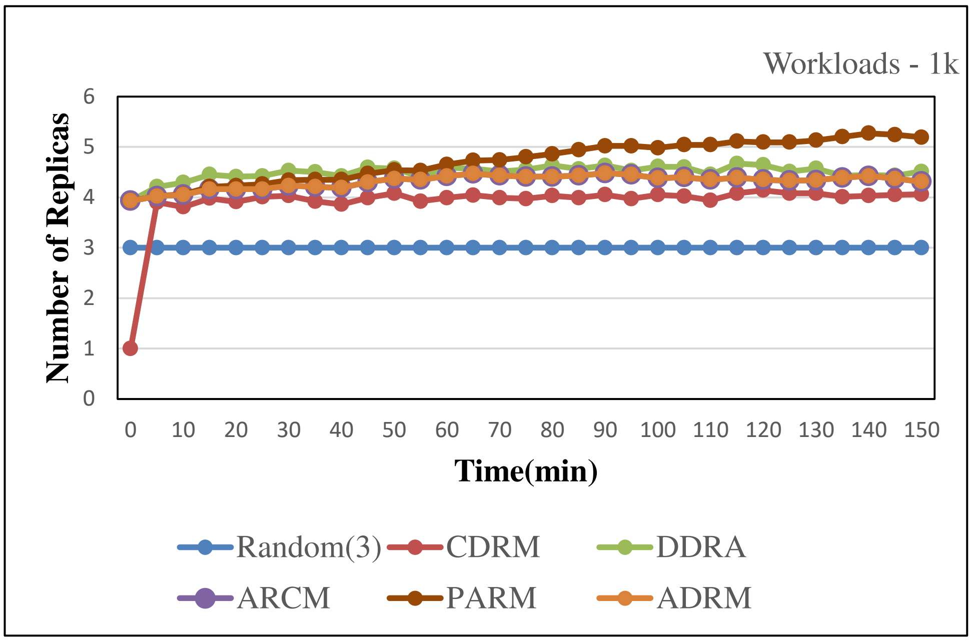


Figure 43. Number of Replicas for 1000 workloads, Node = 100

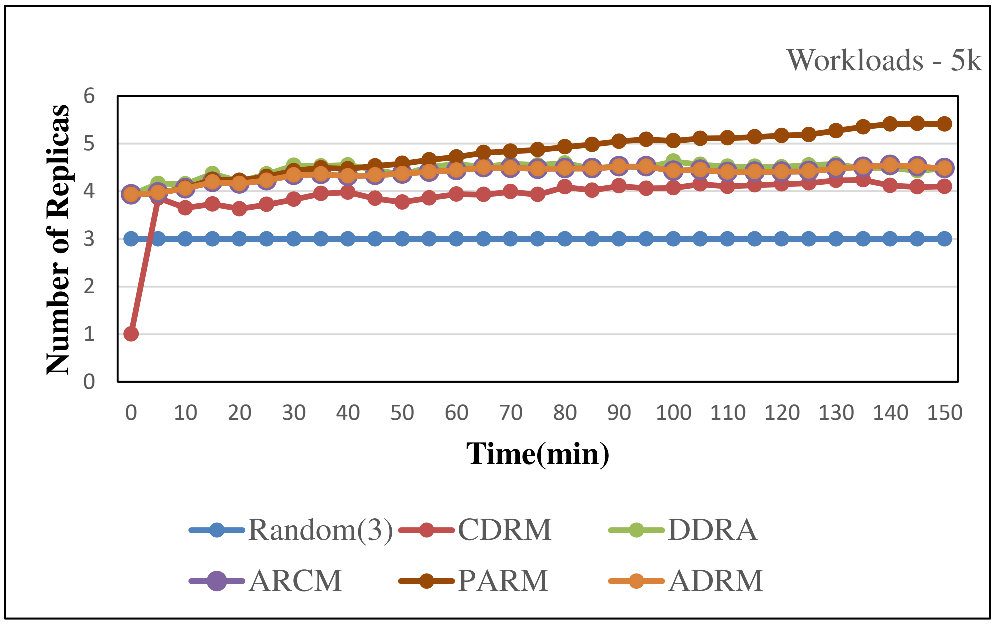


Figure 44. Number of Replicas for 5000 workloads, Node = 100

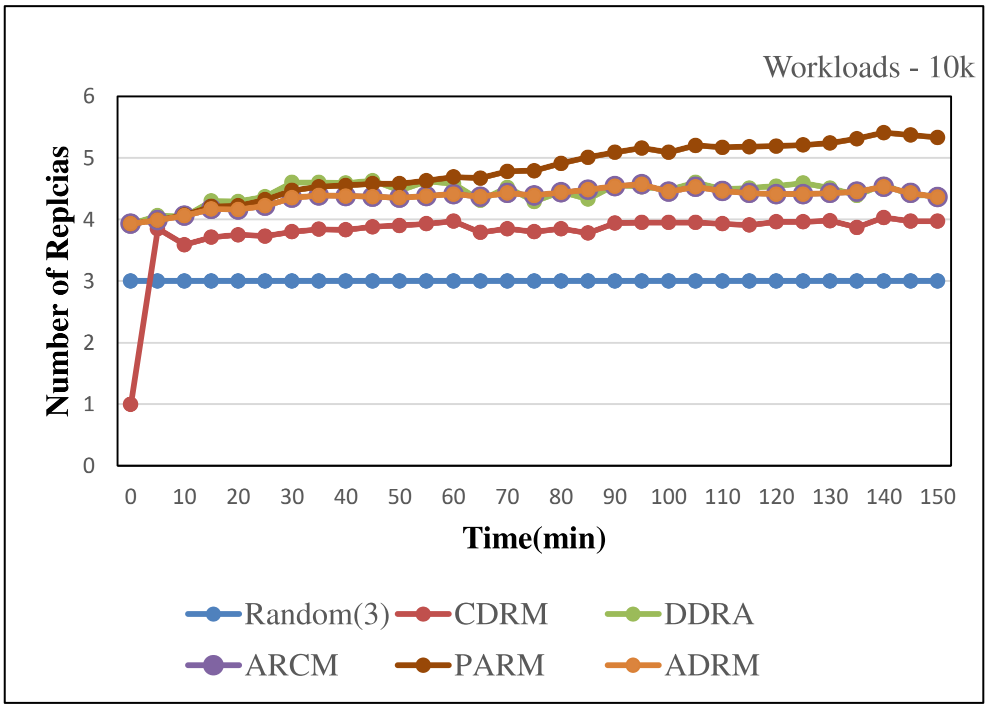


Figure 45. Number of Replicas for 10000 workloads, Node = 100

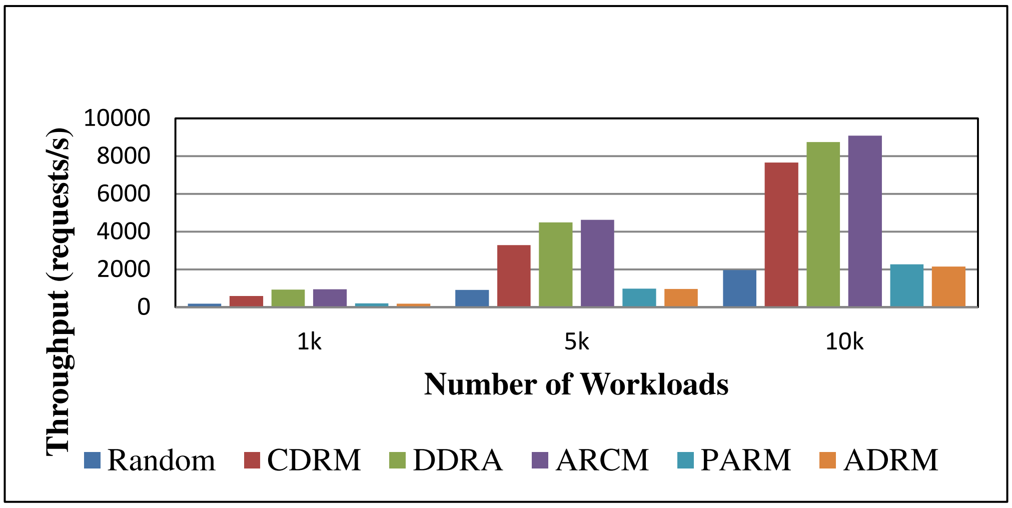


Figure 46. Throughput in 100 nodes

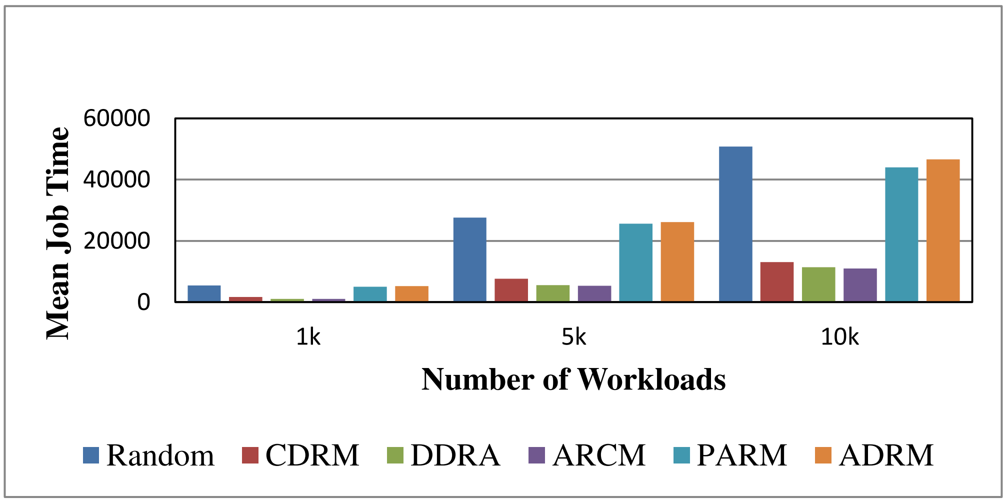


Figure 47. Mean Job Time in 100 nodes

6. Conclusions

Mobile edge computing is an emerging field that handles most of the computing and storage by deploying the MEC servers near to the radio access networks. This network architecture can help to reduce the load of the cloud core and can provide users with low-latency, high-bandwidth, and more diverse application services.

In this paper, an adaptive dynamic replication mechanism call the ARCM algorithm is proposed under the MEC environment to improve the performance of the configuration on popular file. Basically, the proposed ARCM mechanism applies the concepts of atomic decay and the access time interval to predict which file will be popular, and then refers to the replica ratio for replication. Finally, the system can add the new replica to the appropriate service nodes to effectively disperse the workload for the nodes. The experimental results show that the system throughput can be improved and has a better completion time under loosely or densely environments. Therefore, the ARCM algorithm can improve the access efficiency, quality of service, and can maintain better load balancing in the overall MEC environment.

- Conflict-of-interest statement:
  Here is our paper entitled "An Adaptive Replica Configuration Mechanism Based on Predictive File Popularity and Queue Balance in Mobile Edge Computing Environment" that is submitted to you. To the best of our knowledge, the named authors have no conflict of interest, financial or otherwise.
- Funding details:
  The manuscript "An Adaptive Replica Configuration Mechanism Based on Predictive File Popularity and Queue Balance in Mobile Edge Computing Environment" has no funding received.
- Informed consent statement:
  There is no informed consent statement in the research.
- Author contribution:
  1) Mao-Lun Chiang: conceived of the presented idea, verified the analytical methods.
  2) Hui-Ching Hsieh: conceived of the presented idea, verified the analytical methods, wrote the manuscript.
  3) Ting-Yi Chang: verified the analytical methods, wrote the manuscript.
  4) Wei-Ling Lin: carried out the experiment.
  5) Hong-Wei Chen: carried out the experiment.
- Hereby, I consciously assure that for the manuscript, the following is fulfilled:
  1) This material is the authors' own original work, which has not been previously published elsewhere.
  2) The paper is not currently being considered for publication elsewhere.
  3) The paper reflects the authors' own research and analysis truthfully and completely.
  4) The paper properly credits the meaningful contributions of co-authors and co-researchers.
  5) The results are appropriately placed in the context of prior and existing research.
  6) All sources used are properly disclosed (correct citation). Literally copying of text must be indicated as such by using quotation marks and giving proper reference.
  7) All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.

I agree with the above statements and declare that this submission follows the policies of Solid State Ionics as outlined in the Guide for Authors and in the Ethical Statement.

Reference

[1] A. Ahmed and E. Ahmed (2016) A Survey on Mobile Edge Computing. International Conference on Intelligent Systems and Control (ISCO), pp. 1-8.

[2] A. Reznik, R. Arora, M. Cannon, L. Cominardi, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella, C. Turyagyenda and Z. Zheng (2017) Developing Software for Multi-Access Edge Computing. ETSI White Paper, No. 20.

[3] A. Younis, T. X. Tran and D. Pompili (2019) On-Demand Video-Streaming Quality of Experience Maximization in Mobile Edge Computing. IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1-9.

[4] Aral, A., Brandic, I., Uriarte, R.B. et al. (2019) Addressing Application Latency Requirements through Edge Scheduling. J Grid Computing, vol.17, pp. 677-698.

[5] Artesyn Embedded Technologies (2017) Multi-Access Edge Computing Solving Tomorrow's Problems Today. Multi-Access Edge Computing White Paper.

[6] Baraki, H., Jahl, A., Jakob, S. et al. (2019) Optimizing Applications for Mobile Cloud Computing Through MOCCAA. J Grid Computing, vol. 17, pp. 651-676.

[7] D. Schmalstieg and T. Höllerer (2017) Augmented reality: Principles and practice. IEEE Virtual Reality (VR), pp. 425-426.

[8] E. Kaxiras (2003) Atomic and Electronic Structure of Solods. New York: Cambridge University Press.

[9] G. Lawton (2008) Developing Software Online With Platform-as-a-Service Technology. Computer, vol. 41, issue 6. (DOI: 10.1109/MC.2008.185)

[10] H. Yu, G. Xiqian, Y. Lan, H. Ren and Y. Chen (2012) The Research about File-Level Trace Technology. International Conference on Computational and Information Sciences (ICCIS), pp. 1131-1134.

[11] Hsieh, H., Chiang, M. (2019) The Incremental Load Balance Cloud Algorithm by Using Dynamic Data Deployment. Journal of Grid Computing vol. 17, pp. 553-575. https://doi.org/10.1007/s10723-019-09474-2

[12] J. Lim, H. Yu, K. Kim, M. Kim and S. Lee (2017) Preserving Location Privacy of Connected Vehicles With Highly Accurate Location Updates. IEEE Communications Letters, pp. 540-543.

[13] K. Ko, Y. Son, S. Kim and Y. Lee (2017) DisCO: A Distributed and Concurrent Offloading Framework for Mobile Edge Cloud Computing. International Conference on Ubiquitous and Future Networks (ICUFN), pp. 763-766.

[14] K. Shvachko, H. Kuang, S. Radia and R. Chansler (2010) The Hadoop Distributed File System. IEEE Symposium on Mass Storage Systems and Technologies (MSST),

pp. 1-10.

[15] K.Q. Yan, S.C. Wang, Y.H. Su, S.Z. Chen (2015) Popularity-based Rapid Consistency Strategy on Data Replica in the Cloud Computing Environment. International Conference on Innovation and Management (IAM 2015W), Singapore, February 3-6, pp. 28.

[16] M. T. Beck and M. Maier (2014) Mobile Edge Computing: Challenges for Future Virtual Network Embedding Algorithms. International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 3-8.

[17] M. T. Beck, M. Werner, S. Feld and T. Schimper (2014) Mobile Edge Computing: A Taxonomy. International Conference on Advances in Future Internet, pp. 48-54.

[18] P. Buxmann, T. Hess and S. Lehmann (2008) Software as a Service. Wirtschaftsinformatik, vol. 50, issue 6, pp. 500-503.

[19] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong and J. Yao (2013) 5G on the Horizon: Key Challenges for the Radio-Access Network. IEEE Vehicular Technology Magazine, vol. 8, issue 3.

[20] P. Yadav and S. Vishwakarma (2018) Application of Internet of Things and Big Data towards a Smart City. International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), pp. 1-5.

[21] Q. Wei, B. Veeravalli, B. Gong, L. Zeng and D. Feng (2010) CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster. IEEE International Conference on Cluster Computing (CLUSTER), pp. 188-196.

[22] Mao-Lung Chiang, Bo-Ren Huang, Hong-Wei Chen, Hui-Ching Hsieh, Ting-Yi Chang (2019) Study of Adaptive Dynamic Replication Mechanism in Mobile Edge Computing Environment. IEEE International Conferene on Applied System Innovation, Japan, April 11-15.

[23] R. S. Chang, H. P. Chang and Y. T. Wang (2008) A Dynamic Weighted Data Replication Strategy in Data Grids. Proceedings of the IEEE/ACS International Conference on Computer Systems and Application*s*, pp. 414-421.

[24] Rafique, A., Van Landuyt, D. & Joosen, W. (2018) PERSIST: Policy-Based Data Management Middleware for Multi-Tenant SaaS Leveraging Federated Cloud Storage. J Grid Computing, vol. 16, pp. 165–194.

[25] Rahmani, A., Azari, L. & Daniel, H. (2017) A File Group Data Replication Algorithm for Data Grids. J Grid Computing, vol. 15, pp. 379–393.

[26] S. Bhardwaj, L. Jain and S. Jain (2010) Cloud Computing: A Study of Infrastructure as a Service (IaaS). International Journal of Engineering and Information Technology, vol. 2, no. 1.

[27] S. Ghemawat, H. Gobioff and S. T. Leung (2003) The Google file system. Proceeding of the nineteenth ACM symposium on Operating systems principles, pp. 29-43.

[28] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci and A. E. Saddik (2019) Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet. IEEE MultiMedia, vol. 26, issue 1, pp. 21-30.

[29] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella (2017) On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. IEEE Communications Surveys & Tutorials, vol. 19, issue 3, pp. 1657-1681.

[30] T. Yeh and Y. Tu (2018) Enhancing Data Availability through Automatic Replication in the Hadoop Cloud System. International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 86-93.

[31] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub and E. Benkhelifa (2016) The Future of Mobile Cloud Computing: Integrating Cloudlets and Mobile Edge Computing. International Conference on Telecommunications (ICT), pp. 1-5.

[32] Yi-Min Wang, Hsuan-Fu Yu (2010) Adjustable File Replication Strategy for Hierarchical Data Grid. International Journal of Advanced Information Technologies (IJAIT), vol. 4, no. 2, pp. 35-43. (DOI： 10.30111/IJAIT.201006.0005)

[33] Zhang, B., Wang, C., Zhou, B.B. et al. (2018) DCDedupe: Selective Deduplication and Delta Compression with Effective Routing for Distributed Storage. J Grid Computing vol. 16, pp. 195-209.