Journal of Theoretical and Applied Information Technology
15th February 2020. Vol.98. No 03

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REPLICATION STRATEGIES BASED ON MARKOV CHAIN MONTE CARLO AND OPTIMIZATION ON CLOUD APPLICATIONS

AWS ISMAIL ABU EID[1], WSW AWANG[2], MZARINA[3], AH Zakaria[4]

[1]University Sultan Zainal Abdin, Faculty Informatics and Computing, Camps Besut, Terengganu, Malaysia.
[2]University Sultan Zainal Abdin, Faculty Informatics and Computing, Camps Besut, Terengganu, Malaysia.
[3]University Sultan Zainal Abdin, Faculty Informatics and Computing, Camps Besut, Terengganu, Malaysia.
[4]University Sultan Zainal Abdin, Faculty Informatics and Computing, Camps Besut, Terengganu, Malaysia.
E-mail: [1]Aws.abu.eid@gmail.com,[2]suryani@unisza.edu.my, [3]zarina@unisza.edu.my, [4]aznida@unisza.edu.my

ABSTRACT

This work positioned a dynamic replication strategy capable of meeting tenant availability and performance criteria concomitantly. The Monte Carlo BAT Optimization MCBO model is a strategy geared towards calculating the optimal path to update distributed replicas on cloud sustaining high data availability. In general, replica creation is prompted in two circumstances: a specific number of replicas is not achieved or in case of unsatisfactory response time objective. Following this, it is necessary for the replication process to be successful in order to design the MCBO model for determine optimal path. Data replication and query scheduling were combined for ensuring the replica placement in a load-balancing manner while handling tenant budget. The experimental outcomes revealed significant improvement for the availability and performance following the use of the model.

Keywords: *Data Replication, Cloud Enviroment, Mcbo Model, Bat Algorithm, Optimization*

1. INTRODUCTION

Wide-ranging and large-scale incorporation of Internet services and big data in the current climate has rendered the cloud as the optimum answer for the burgeoning need for storage due to its provision of illimitable capacity, high availability, and swift access time. The cloud computing paradigm itself is highly popularized in today's industrial and academic sectors, attracting scholarly attention due to the possibility of extensive benefits for the industry and communityalike Ali et al .2015 [1]. For cloud providers, resources reutilization can be done following their release by specific users, which leads to their enhanced usage Rittinghouse, and Ransome 2017 [2]. This allows the establishment of multiple large-scale data centresin geographically disseminated sites. Following this, data replicationis upheld as an effectual approach for faulttolerance provisions, minimized end-user latency, and reduced data exchange via a network. Consequently, replica management has emerged as a challenging field for these providers.

In general, cloud computing is an up-and-coming ideal offering service such as computing, communication, and storage resources over a network. Various cloud applications and their service provision are stuck in gridlock due to communication resources. This renders data replication a favourableanswer as it positions data (e.g. databases) closer to the consumers (e.g. cloud applications) Boru et al. 2015 [3], as well as alleviating issues of network delays and bandwidth utilization. As a result, distributed storage is highly demanded due to the voluminous amount of data handled and disseminated by application services on the Internet in providing for multiple tenants [4]. Contextually, data replication is deemed a popular technique that assures availability and performance by distributing many data copies across differing locations Selvi et al. 2015[5]. It improves the likelihood for a minimum of one copy being accessible in case of failures and in consideration of various objectives, such as reduced storage cost and enhanced fault-tolerance and access delays Kumari and Kaur (2018) [6]. Thus, replication is unsurprisingly a crucial element of cloud computing applications in which the services are utilized by a multitude of clientele that accesses their data from varying locations at a low frequency of latencies.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Various works in the literature have described data replication in cloud systems Milani et al .2016, Tabet et al. 2017 [7,8]. Such strategies have been positioned to achieve an increased level of data availability and load balancing Limam et al. 2016 [4], enhanced performances Tos et al.2016, Xiong et al. 2011 [9,10], and minimized bandwidth consumption Kloudas et al. 2015 [11]. However, the outlined goals are shown to be discordant: for instance, data replication assures their availability, but this may occur in the expense of inter-site communication. This results in an overloaded network, thus impacting the performance. Furthermore, a majority of the strategies dismiss replication cost and provider profit both.

As such, this work designs, applies, and positions the Monte Carlo Markov Chain (MCMC) technique in producing random samples (paths) in a cloud environment in order to undertake the outcome assessment using a meta-heuristic algorithm (BAT-algorithm) one-test-at-a-time. Utilizing MCBO as a novel technique, this study uses an optimal path that is implementable for solving one of the commonest replication-related issues in the cloud. Consequently, this study aims proposed the method to reduce the time and resource consuming, which will discuss the details of method stages in section three. In brief, by generating the likelihood of distinct results in a random variables method, input analysis, and test suite iteration. Additionally, the process of attaining the best outcomes is further aided by calibrating pertinent MCBO variables.

This work is organized according to the following structure: section two assesses correlated studies undertaken regarding replica placement and migration in the cloud, while the consequent section details the proposed answer to the issue. Then, section four presents the simulation outcomes obtained in evaluating the approach and its performance. Lastly, section five offers a comprehensive conclusion.

2. LITERATURE REVIEW AND RELATED WORK

Various works have previously positioned approaches for data replication in cloud systems. For example, Kloudas et al. 2015 study [11] have detailed a scheduler termed as Pixida in order to achieve reduced data movement across resource-limited links. This is achieved by the introduction of Silo, which is a new abstraction method crucial for modeling the Pixida's scheduling objectives as a graph partitioning problem. Furthermore, the authors reveal that the pre-existing graph partitioning problem formulations are not mapped to the way substantial data jobs function, thus rendering the solutions incongruent with opportunities for data movement prevention. Consequently, a new graph partitioning problem has been formulated and a novel algorithm is proposed as a solution. Pixida has been further integrated into the Spark in which experimental outcomes show that it attains traffic reduction up to nine-fold using the links in comparison with current schedulers. Meanwhile, Tos et al.2016 [9] have positioned an approach for data replication aimed for a satisfactory performance assurance for the tenants while concomitantly securing cloud provider profitability. This strategy offers an estimated response time for any queries and the expenses impacting the aforementioned profitability.

In another instance, Boru et al. 2015 [3] have designed models meant as a solution for energy consumption and bandwidth demand posed by database access in cloud computing datacentres. Furthermore, the work has detailed an effectual energy replication strategy according to the models, thus resulting in an enhanced Quality of Service (QoS) and minimised communication delays. The resulting assessment generated via comprehensive simulations has revealed performance and energy efficiency tradeoffs, which subsequently served as guidance in designing upcoming data replication solutions.

Next, study Bonvin et al .2010 [12] have detailed a scattered key-value store named Skute in which virtual nodes function autonomous agents, which makes decisions to the advantage of the data owners without external control. Here, the economic model positioned revolves around a virtual economy. These nodes will make rental payments to other nodes for replica hosting with regard to storage utilisation and query load. Moreover, they are revenue-generating in nature according to the number of queries answered. The study positioned has taken into consideration the availability first followed by the net benefit via replica placement to nodes according to their economic fitness. In Skute, performance guarantees are not deemed as a crucial component of the SLA implemented in it but a reduction in the average query load per node is recorded and obtained over time.

The study by Sakr and Liu, 2012 [13] has detailed an SLA-focused provisioning approach for cloud databases, which is consumer-centric in nature.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Here, the database servers are subjected to scaling in and out per the SLA requirements; being the primary SLA goal for decision-making processes, the total execution time of transactions is thus selected. This strategy requires close monitoring of the cloud system, while the cloud providers comprehensively describe application-specific rules to ensure adaptive resource scaling. Although the benefits of SLA-aware provisioning for scaling are known, the economic outcomes of replication for cloud providers have not been detailed.

Janpet and Wen, 2013 [14] have proposed a strategy for data replication geared towards reduced data access time, which is achieved by identifying the shortest access path to data objects. The work has modelled the access frequency, delay, and replication budget in order to yield the node that is the closest and most suited for replica placement. In particular, the replication budget is predefined in nature and merely implemented as a limiting factor for the users towards ensuring a regulated number of replicas. However, the work did not address in-depth regarding the economic correlation between the users and cloud provider. Regardless, the work has shown that a closer placement between data objects and the nodes with high access frequency enhances the response time.

Next, SWORD has been detailed by Kumar et al. 2014 [15], which is a workload-aware data placement and replica selection scheme. The work has positioned a novel metric named query span, which is the average amount of nodes implemented for query execution. This strategy is aimed towards reducing the query span in achieving reduced communication overhead, resource consumption, energy footprint, and transaction costs. The authors have thus claimed that SWORD handles performance degradation via increments of data repartitioning. Additionally, provider profit is not emphasised in this work, but the study effectiveness has been proven via an experimental analysis conducted for the measurement of query span and transaction times.

Meanwhile, Zhang et al. 2014 [16] have depicted an auction model in the implantation of a replica placement policy, which is geared towards meeting the availability aspect in a large-scale cloud storage environment. If the coveted level of availability is not sustained, the placement of a new replica is established by holding bidding. The bidding price is reliant upon different node properties, such as failure probability, network bandwidth, and space availability. The objective function does not incorporate the response time, but the authors have reported improved performance and satisfactory availability during the experiments.

Besides, Sousa and Machado 2012 [17] have offered Replica, which is a strategy for elastic multitenant database replication. It considers the performance SLA and makes adjustments for the number of replicas in an elastic manner by observing the system implementation. Any workload changes and variations can be dealt with by targeting the transactions towards replicas having sufficient resources. An experimental work has juxtaposed RepliC with a rule-based scaling scheme, which yielded outcomes indicative of the strategy meeting the QoS satisfactorily with minimum SLA violations.

Boru et al. 2015 [18] have described an approach for data replication with an emphasis on enhancing the energy efficiency of cloud data centres. Here, the strategy advances energy consumption, bandwidth implementation, and network delays at the levels of inter-data centre and intra-data center a like. The data centre power utilisation and bandwidth consumption of database operations are modelled accordingly in this work, whereby recurring assessments are undertaken in determining the replication decision. They will also estimate the power and bandwidth utilisation of the replicas in future periods. A simulation study undertaken has thus revealed that a closer placement between replicas yields improved power consumption and response time. Regardless, the economic rewards are not emphasised.

3. PROPOSED APPROACH

The approach proposed in this work is termed as the MCBO model, which is subjected to four phases during the building process in order to attain the goals of reduced time and resource consumption when updates for the cloud replicas are required. This is shown in Figure 3.3 accordingly. Meanwhile, Figure 1 displays all four stages and their processes as follows:

Implement the MCMC Algorithm (see Algorithm 1) to generate more random walks (i.e. more iterations). A random walk can be described as a mathematical object otherwise known as a stochastic or random process, which defines a route made up of sequential random moves on certain mathematical space, such as the integers. (see figure 1)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1. Assess and obtain the outcomes for every iteration yielded from the MCMC algorithm (Phase 1) with random walks(i.e. paths), parameter time, and location. Software tools can be utilized to save such outcomes, such as gridsim or excel.
2. Generate the MCBO test suite (see Algorithm 2), which implements the Bat Algorithm (BA) as the basic algorithm in establishing the optimal path (i.e. random walk). This is dependent upon the starting position and the time taken for random walks as indicated by the MCMC method.
3. The MCBO strategy will save the optimal random walk (i.e. path) that is utilisable in the scheduler policy in order to update the replica on the cloud.

4. RESULT:

The model proposed and its phases yield results, which will be detailed accordingly across four sub-sections.

4.1 Run of the MCBO model

The current study was associated with the random disrupted strategy for the selection of an initial location and the amount of step required for the task completion. By establishing the starting location for the MCBO model, the scale and the limits utilised in this work could be identified. Accordingly, the scale of 1:100 was implemented, indicative of every 1 is equivalent to 100 kilometres on the ground. Furthermore, preventing a coordinate X or y negative signal was achieved by working on the first quarter (x, y) of the Cartesian coordinate system or Universal Transverse Mercator (UTM). Our boundary for scaling from 0 to 100 that mean we work in positive area. Distribution system x axis and y axis Cartesian level (0,100). Next, one could proceed to the subsequent phase in the model using the BAT algorithm in establishing the optimal path orvariables (i.e. Start location, time (Steps)). This allowed a measurement of the model effectiveness towards minimising the time and resource consumptions upon the use ofthe update operation for the replicas in the cloud.

Start Location: The first location value is arbitrarily established in the first iteration.

Time (Steps): The number of steps recorded upon the model movement from the start location until the end of the domain border.

4.1.1 Run Random Walk Algorithm

This phase consisted of two principal sections under the umbrella of the processing. The first section detailed the output for five iterations generated by the random walk algorithm and the extraction of the (Start Location, Times (step)). Then, the second section implemented the random walk algorithm to generate outputs for 9000 iterations.

4.1.1.1 Outputs for five iterations generated by Random Walk Algorithm.

The outputs were established using the random walk algorithm (i.e. start location, current location, next location to achieve the final position (100), and how long it takes (i.e. steps) to achieve the final position (100)).

- Iteration Number 1 :

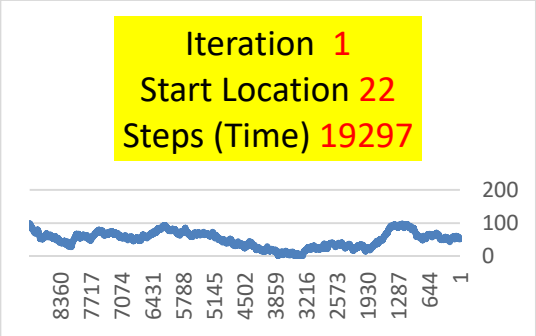


Figure 2: Iteration Number 1.

The route obtained by the random walk algorithm is displayed in Figure 3.4.It included the location of two required values at the start point 22 and the steps necessary to cover boundary 19297.

$I_1 = \{L1, T1\}$, represented numerically as {[22] , [19297]}.

- Iteration Number 2.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

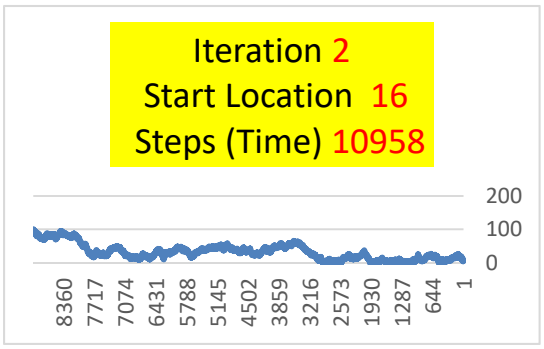
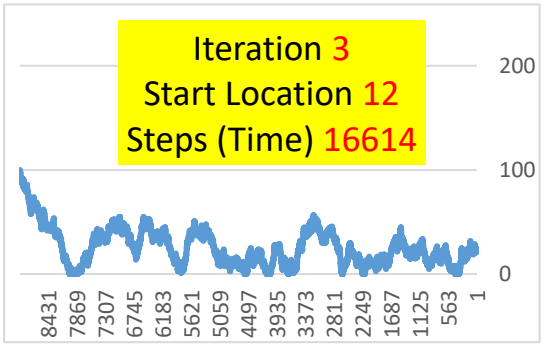


Figure 3: Iteration Number 2.

The route produced by the random walk algorithm is illustrated in Figure 3.It included the location of two required values at the start point 16 and the steps necessary to cover boundary 10958.

- $I_2 = \{L2, T2\}$, represented numerically as $\{[16], [10958]\}$.



The route established by the random walk algorithm is depicted in Figure 4.It included the location of two required values at the start point 12 and the steps necessary to cover boundary 16614.

- $I_3 = \{L3, T3\}$, represented numerically as $\{[16], [10958]\}$.
- **Iteration Number4.**

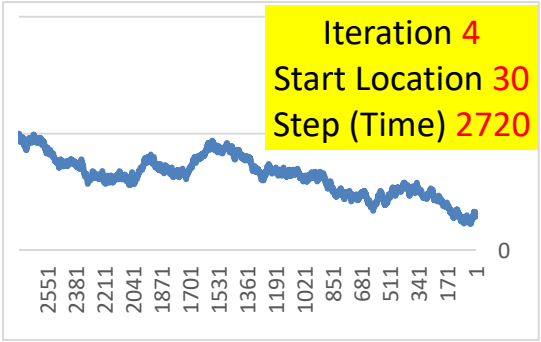


Figure 5:Iteration Number 4.

The route obtained by the random walk algorithm is illustrated in Figure 5.It included the location of two required values at the start point 30 and the steps necessary to cover boundary 2720.

- $I_4 = \{L4, T4\}$, represented numerically as $\{[30], [2720]\}$.
- **Iteration Number 5.**

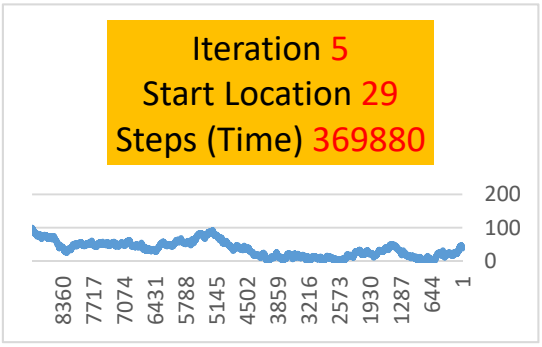


Figure 6:Iteration Number 5.

The route yielded by the random walk algorithm is revealed in Figure 6. It included the location of two required values at the start point 30 and the steps necessary to cover boundary 2720.

- $I_5 = \{L5, T5\}$, represented numerically as $\{[29], [369880]\}$.

4.1.1.2 The output for 9000 iterations Generate by Random Walk Algorithm

This section details the output obtained when 9000 iterations are subjected to the random walk algorithm. Every iteration was equipped with their respective (Start location, Time (step)) in order to achieve the final position (100) in the domain. Here, the start location could be replicated due to its arbitrary nature, thus requiring all 9000 iterations to be categorised in order to define the values of the start location and the average time (steps) for each of these locations. Table 1 displays the output values for all 9000 iterations with information such as start location, repeated, summation, average, and rounded average values included.

4.1.2 Analysis and Result Collection

This section describes the second phase of the MCBO model. The route obtained by the random

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

walk algorithm is depicted in Figure 7, detailing the outcomes for all 9000 iterations. The information was utilised to assess the outcomes obtained and extract the values of tworequired values of the start location and the steps necessary to cover boundary.

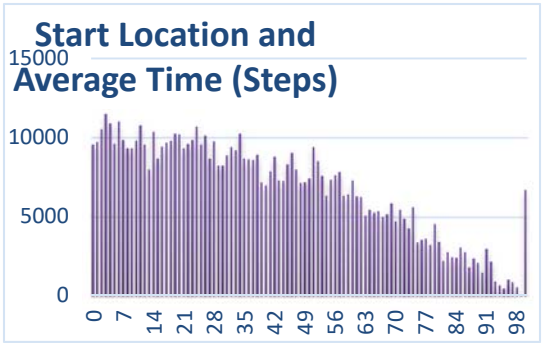


Figure7: Graph of the start location and time (step) to reach the domain boundaries

The outputs from 9000 iterations were divided into 20 sets and extracted depending on the Start location (0-99), as {0,1,5,10,15,20,25,35,35,40,45,50,55,60,65,70,75, 80,85,90,95,99} in establishing the candidate sets. Table 2 depicts the candidate set in which each set has iterations that yielded the time (steps) required to attain the upper boundary. The particular number of steps were input into the BAT algorithm in order to establish the optimal time (step).

During this particular phase, the representative outcomes for three candidate sets were used to clearly delineate the input for the BAT algorithm.

- **Candidate Set Number 1:**(see table 3)
- **Candidate Set Number 2:** (see table 4)
- **Candidate Set Number 3**: (see table 5)

4.1.3 The Use of BAT Algorithm

The current phase depicts the representative components as Numerical Legal ValuesI_i = {Li, Ti}, which is represented numerically using Table 2. This was done by extracting the candidate sets {[start location], [(Time)]} from Table 1. Typically, defining the sets of candidate group required the BAT algorithm to undergo three phases in order to assess the optimal Solution for each candidate set. This is depicted accordingly inTable6.

Table 6: BAT population for candidate sets

Size (N)	Start location	Steps (Time)
Set 1 [0]	[X1,X2.X3,.. X86]	[7334, 10925, 5896,..,5224]
Set 2 [5]	[X1,X2.X3,.. X101]	[3607, 8885, 24832,..,8857]
Set 3 [10]	[X1,X2.X3,.. X83]	[4739, 9186, 16212,… ,3660]
Set 4 [15]	[X1,X2.X3,.. X101]	[6077, 15842, 3263,…,10881]
Set 5 [20]	[X1,X2.X3,.. X92]	[31936, 3792, 5692,…,10948]
…	…	
…	…	
Set 20 [99]	[X1,X2.X3,.. X81]	[1, 83, 37,…,5]

The phases can be viewed as seen below:

Phase 1.

The variable for the BAT algorithm is determined using Table 7, wherethe size of the BATalgorithm size (N) is established using the mean population of the algorithm.Meanwhile, the candidate sets for each set indicate the iteration for each set and the value of each iteration in order to obtain the solution = $\sum(time)/Popoultion$

Table 7: Bat input value

Input Name	Values	Describe
Bat populations	20 sets	Each set has a specific number of iteration and time
Solutions	Defined by BAT Algorithm	Rounded (Average(Time))
Lower boundary	0	Lower boundary
Upper boundary	3	Upper boundary
Tolerance	0.001	Tolerance value

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Phase 2. Calculate the **solution** for Bat populations. (See figure 8)

The aforementioned outputs of the BAT algorithm were made up of population, solution, speed measurement (velocity), and frequency.

4.1.3.2 Solution for Candidate Sets Generated by BAT Algorithm

Each set was subjected in the BAT algorithm in order to obtain the solution. Table 8 displays the solution obtained for the candidate sets.

4.1.4 Friedman Test

The Friedman test was used to assess the significant values and best position of the candidate sets. The results obtained are shown in Table 9.

Friedman test for collecting the results of MCMC

In this subsection, the test check was undertaken to all 20 sets (refer to Chapter 3) across all candidate sets in order to establish the significance or insignificance of the outcomes.

Friedman test for Candidate Set Zero-location

Table 8. Descriptive Statistics for zero-location

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Start Location	86	.00	.000	0	0
Time	86	9587.52	6562.659	1170	27707

- **Friedman Test**

Table 9. Friedman rank for zero-location

Ranks	
Start Location	1.00
Time	2.00

Table 10. Friedman test for zero-location

Test Statistics[a]	
N	86
Chi-Square	86.000
df	1
Asymp. Sig.	.000

The outcomes obtained showed that the Chi-square value of 86 at the level of 0.01, which was indicative of the error that occurred every 100 iterations at one time.

5. CONCLUSION

This work positioned a strategy for data replication strategy geared for a concomitant effect of meeting the availability and performance criteria both. To ensure resource preservation, the MCMC method was utilised inproducing random samples (i.e. paths) in a cloud environment for outcome assessment by implementing a meta-heuristic algorithm (i.e. BATalgorithm) one-test-at-a-time. The use of a novel model termed as MCBO, one could establish the optimal path. MCBO benchmarking involves benchmarking of current approaches as well as the accompanying statistical analysis.MCBO performance achieved results that were statistically significant. Building on the current content in this chapter, the next chapter will summarize all findings, conclude and comment on contributions, and provide a roadmap for possible future research in this direction.

6. FUTURE WORK

Since the application of MCBO presented in this study is still a prototype, the realization of automated test replication strategies based on Markov Chain Monte Carlo and Optimization on Cloud Applications will be a clear starting point for future work. Many MCBO attributes (i.e. input-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

output interaction, candidate sets and Scale) were especially to be included.

REFERENCES

[1] Ali, M., Khan, S. U., &Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information sciences*, *305*, 357-383.

[2] Rittinghouse, J. W., &Ransome, J. F. (2017). *Cloud computing: implementation, management, and security*. CRC press.

[3] Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., &Zomaya, A. Y. (2015, June). Models for efficient data replication in cloud computing datacenters. In *2015 IEEE International Conference on Communications (ICC)* (pp. 6056-6061). IEEE.

[4] Limam, S., Mokadem, R., &Belalem, G. (2019). Data replication strategy with satisfaction of availability, performance and tenant budget requirements. *Cluster Computing*, 1-12.

[5] Selvi, M. S. A. E., &Anbuselvi, R. (2015, March). An Analysis of Data Replication Issues and Strategies on Cloud Storage System. In *International Journal of Engineering Research & Technology (IJERT), NCICN-2015 Conference Proceedings, pp18-21*.

[6] Kumari, P., & Kaur, P. (2018). A survey of fault tolerance in cloud computing. *Journal of King Saud University-Computer and Information Sciences*.

[7] Milani, B. A., &Navimipour, N. J. (2016). A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications*, *64*, 229-238.

[8] Tabet, K., Mokadem, R., Laouar, M. R., &Eom, S. (2017). Data replication in cloud systems: a survey. *International Journal of Information Systems and Social Change (IJISSC)*, *8*(3), 17-33.

[9] Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., & Bora, S. (2016, July). A performance and profit oriented data replication strategy for cloud systems. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (pp. 780-787). IEEE.

[10] Xiong, R., Luo, J., Song, A., Liu, B., & Dong, F. (2011, September). QoS preference-aware replica selection strategy using MapReduce-based PGA in data grids. In *2011 International Conference on Parallel Processing* (pp. 394-403). IEEE.

[11] Kloudas, K., Mamede, M., Preguiça, N., & Rodrigues, R. (2015). Pixida: optimizing data parallel jobs in wide-area data analytics. *Proceedings of the VLDB Endowment*, *9*(2), 72-83.

[12] Bonvin, N., Papaioannou, T. G., &Aberer, K. (2010, June). A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In *Proceedings of the 1st ACM symposium on Cloud computing* (pp. 205-216). ACM.

[13] Sakr, S., & Liu, A. (2012, June). Sla-based and consumer-centric dynamic provisioning for cloud databases. In *2012 IEEE Fifth International Conference on Cloud Computing* (pp. 360-367). IEEE.

[14] Janpet, J., & Wen, Y. F. (2013, March). Reliable and available data replication planning for cloud storage. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)* (pp. 772-779). IEEE.

[15] Kumar, K. A., Quamar, A., Deshpande, A., &Khuller, S. (2014). SWORD: workload-aware data placement and replica selection for cloud data management systems. *The VLDB Journal—The International Journal on Very Large Data Bases*, *23*(6), 845-870.

[16] Zhang, H., Lin, B., Liu, Z., &Guo, W. (2014, September). Data replication placement strategy based on bidding mode for cloud storage cluster. In *2014 11th Web Information System and Application Conference* (pp. 207-212). IEEE.

[17] Sousa, F. R., & Machado, J. C. (2012, November). Towards elastic multi-tenant database replication with quality of service. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing* (pp. 168-175). IEEE Computer Society.

[18] Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., &Zomaya, A. Y. (2015). Energy-efficient data replication in cloud computing datacenters. *Cluster computing*, *18*(1), 385-402.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Indexed

Algorithm 1: MCMC method(Metropolis-Hastings Algorithm)

Input: Number of iteration **(N)**; Current Position (**L**);Current Position **(U)**; //distribution limits (L,U)
Output: **Random Walk path(series)** ,**StepsS** //time to cover the path

1: Define n, l_{lower}, , U_{upper}
2: Initialise CurrentMin , CurrentMax; Step S ; Current Position **P**
3: Randomly Initialise R_i
4: **while**$(number\ of\ itteration < N)$**do**
5: { **while** (Current Position **P** **!=L&&**Current Position **P** **!= U) do**
6: Generate Randomly R_i =Random.math();
7: If (R_i<Mid(L,U))
8: Current Position **P** ++
9: **If** (Current Position **P** >Current Max)
Current Max= Current Position
10: **End if**
11: Else
12: Current Position **P** - -
13: **If** (Current Position **P** <Current Min**)**
14: Current Min= Current Position
15: **End if**
16: **End if**
17: Step **S** ++;
18: **End while**
19: **End while**

Algorithm 1: MCMC method(Metropolis-Hastings Algorithm)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm 2: Bat-inspired Algorithm (BA)	
Input: objective function $f(x_i), x_i = (x_{i_1}, \dots, x_{i_D})^T$. **Output**: Best fitness x_*.	
1:	Define n, T_{max}, $Q_i \in [Q_{min}, Q_{max}]$;
2:	Randomly Initialise x_i, $velocity_i$, $Q_i$for $i = 1, 2, \dots, n$;
3:	Initialise pulse rates r_i and the loudness A_i;
4:	**while**$(ts < \mathrm{T}_{max})$**do**
5:	**for** each bat $\mathrm{n_i}$**do**
6:	Use movement equations to generate fresh alternatives by changing frequency, updating speed and place (4-2 to 4-4);
7:	**if**$(rand(0,1) > r_i)$**then**
8:	In the present population, select the best solution ; generate a local solution for the best solution ;
9:	Select the best solution in the current population ; produce a local solution for the best solution ;
10:	**End**
11:	By flying randomly, generate a fresh solution ;
12:	**For Every Location in 9000 do**
13:	Calculate Time (steps) For same Location ;
14:	**Find the solution =** $\sum(\boldsymbol{time})/\boldsymbol{Popoultion}$**;**
15:	**End for**
16:	Extract the result solution in the current population ;
17:	**End**
18:	**End**
19:	Processing and visualisation of outcomes ;

Algorithm 2: Bat-inspired Algorithm (BA)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

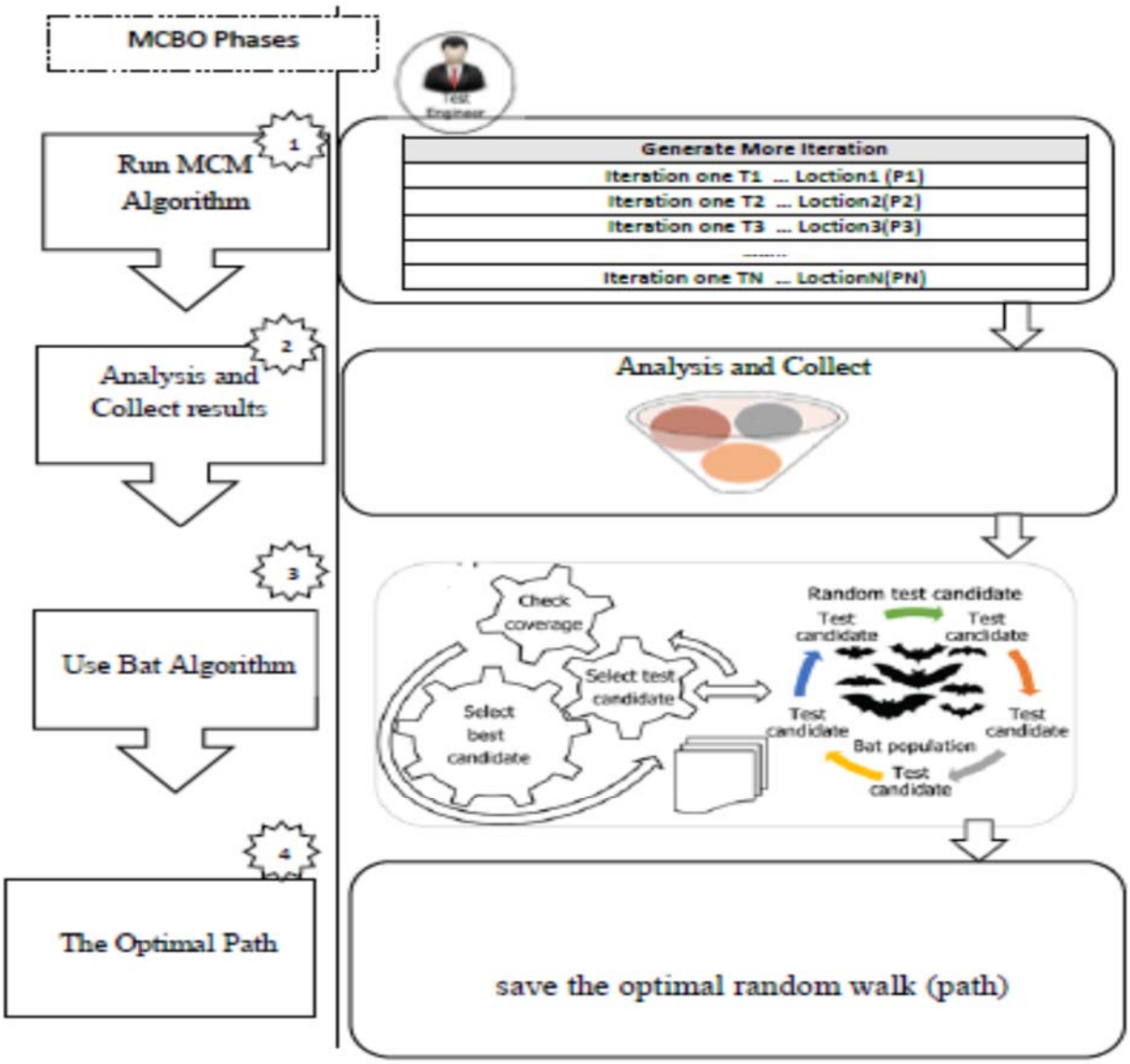


Figure 1: The overview of the Proposed Model.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: The output for 9000 iterations

Start location	Repeat	Total (Time)
0	86	824527
1	78	757359
2	103	1087452
3	90	1035219
4	85	928084
5	101	969729
6	74	816235
7	92	905551
8	98	915197
9	89	829879
10	83	814755
11	94	1015051
12	89	852482
13	74	592313
14	98	1017524
15	101	876881
16	85	801942
17	91	880056
18	95	930762
19	87	892180
20	92	940185
21	88	821449
22	104	999976
23	91	895071
24	88	943704
25	89	850592
26	89	902834
27	82	711740
28	92	898882
29	96	790917
30	100	824688
31	95	843790
32	80	753680
33	84	772957
34	87	891337
35	108	938496
36	86	742475
37	96	825666
38	72	641618
39	85	608481
40	84	579951
41	78	613216
42	82	722078
43	90	657173
44	97	703898
45	98	814998
46	92	833236
47	91	727959
48	85	606631
49	96	690201
50	91	673508
51	84	791455
52	92	782037
53	84	635281
54	95	598865
55	79	580065
56	70	533887
57	97	756999
58	89	561937
59	91	582123
60	83	605273
61	78	491410
62	83	518481
63	89	454433
64	98	533737
65	91	480878
66	95	508514
67	88	442326
68	78	405510
69	114	664363
70	87	411019
71	91	493271
72	91	447483
73	86	367529
74	81	451840
75	78	266186
76	85	302429
77	113	413895
78	91	293423
79	84	384836
80	97	332289
81	78	173995
82	95	265213
83	78	190857
84	102	246959
85	86	264135
86	93	257961
87	73	131197
88	81	194620
89	99	208662
90	108	159157
91	83	247811
92	111	240007
93	98	90067
94	110	74258
95	92	45690
96	86	90888
97	100	87061
98	103	57463
99	81	6543
Total	**9000**	**60088883**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Candidate Set Generation from Random Walk Algorithm.

Set	Start location	Repeat
1	0	86
2	5	101
3	10	83
4	15	101
5	20	92
6	25	88
7	30	100
8	35	108
9	40	84
10	45	98
11	50	91
12	55	79
13	60	83
14	65	91
15	70	87
16	75	78
17	80	97
18	85	86
19	90	108
20	99	81

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3: Result for start location 0 using Random Walk Algorithm

Iteration	Start location	Steps (Time)
1	0	7334
2	0	10925
3	0	5896
4	0	27707
5	0	22032
6	0	4569
7	0	1879
8	0	11946
9	0	4823
10	0	5471
11	0	16486
12	0	7948
13	0	4758
14	0	4060
15	0	7415
16	0	11097
17	0	21091
18	0	7776
19	0	6602
20	0	11956
21	0	17456
22	0	3561
23	0	18493
24	0	3365
25	0	10826
26	0	2949
27	0	20403
28	0	17551
29	0	13243
30	0	8717
31	0	3538
32	0	2851
33	0	20190
34	0	2777
35	0	9278
36	0	11602
37	0	20026
38	0	3510
39	0	5775
40	0	4163
41	0	5840
42	0	23899
43	0	2343
44	0	1170
45	0	7168
46	0	6087
47	0	10590
48	0	3744
49	0	8091
50	0	25241
51	0	3285
52	0	7795
53	0	7825
54	0	2777
55	0	1513
56	0	10512
57	0	17116
58	0	19573
59	0	6771
60	0	12867
61	0	14057
62	0	2106
63	0	2451
64	0	2698
65	0	15810
66	0	13741
67	0	6920
68	0	9019
69	0	4420
70	0	15720
71	0	3677
72	0	11216
73	0	23224
74	0	13682
75	0	7004
76	0	17485
77	0	3033
78	0	18209
79	0	5865
80	0	11873
81	0	9041
82	0	11721
83	0	1809
84	0	6000
85	0	2270
86	0	5224

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4: Result for start location 5 using Random Walk Algorithm

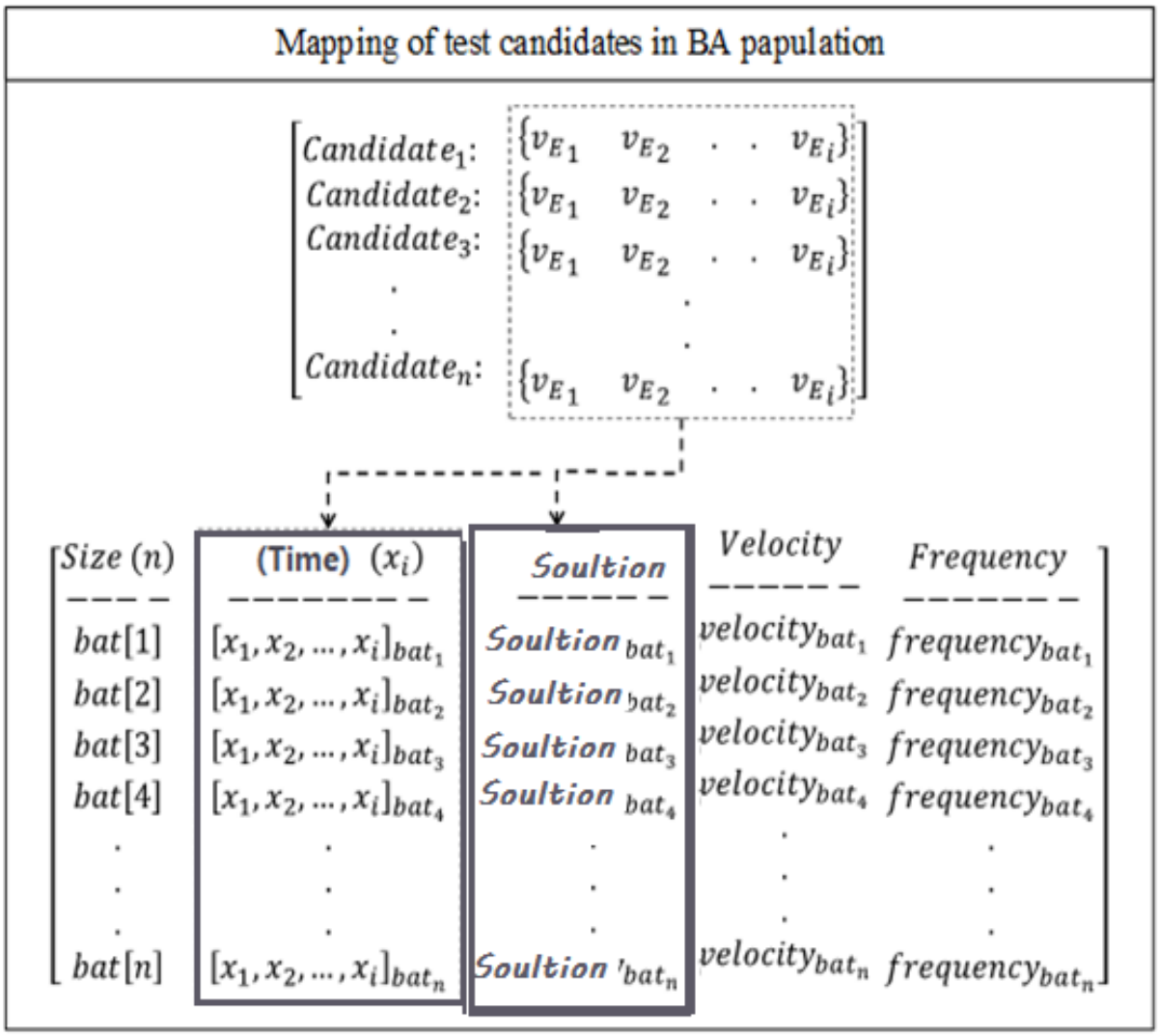
Iteration	Start location	Steps (Time)
1	5	3607
2	5	8885
3	5	24832
4	5	8795
5	5	22174
6	5	5000
7	5	32249
8	5	3130
9	5	7325
10	5	15868
11	5	14381
12	5	5242
13	5	24064
14	5	3680
15	5	2806
16	5	12647
17	5	17070
18	5	3063
19	5	7231
20	5	5018
21	5	2591
22	5	3223
23	5	7078
24	5	4089
25	5	17023
26	5	7213
27	5	13782
28	5	5684
29	5	3394
30	5	6650
31	5	3573
32	5	14196
33	5	1881
34	5	20285
35	5	4018
36	5	3638
37	5	2171
38	5	12589
39	5	3716
40	5	13704
41	5	11062
42	5	6435
43	5	6244
44	5	15542
45	5	2136
46	5	11445
47	5	9775
48	5	3380
49	5	15579
50	5	7307
51	5	24665
52	5	25484
53	5	2896
54	5	6439
55	5	29566
56	5	10228
57	5	9424
58	5	33144
59	5	2690
60	5	24692
61	5	4293
62	5	8212
63	5	12140
64	5	12701
65	5	5012
66	5	14230
67	5	17438
68	5	1337
69	5	11824
70	5	9558
71	5	1657
72	5	15635
73	5	3921
74	5	5995
75	5	10603
76	5	7211
77	5	7433
78	5	4353
79	5	16967
80	5	8085
81	5	4520
82	5	842
83	5	4222
84	5	9628
85	5	6786
86	5	2483
87	5	2492
88	5	10661
89	5	3603
90	5	12510
91	5	15247
92	5	12325
93	5	6148
94	5	15722
95	5	9071
96	5	9155
97	5	1459
98	5	4977
99	5	8593
100	5	6125
101	5	8857

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 5:Result For Start Location 10 Using Random Walk Algorithm

Iteration	Start location	Steps (Time)
1	10	4739
2	10	9186
3	10	16212
4	10	39767
5	10	1572
6	10	10465
7	10	16032
8	10	18894
9	10	4018
10	10	6936
11	10	4418
12	10	12637
13	10	4880
14	10	10186
15	10	18069
16	10	17822
17	10	9898
18	10	17342
19	10	8369
20	10	12242
21	10	15280
22	10	10539
23	10	2855
24	10	22746
25	10	13834
26	10	2014
27	10	8263
28	10	5052
29	10	4624
30	10	19851
31	10	7053
32	10	9981
33	10	1223
34	10	3924
35	10	5385
36	10	11062
37	10	20798
38	10	45646
39	10	6667
40	10	5220
41	10	1682
42	10	5605
43	10	5757
44	10	4935
45	10	9366
46	10	4597
47	10	6588
48	10	9697
49	10	5349
50	10	3398
51	10	6118
52	10	6679
53	10	7671
54	10	4392
55	10	5023
56	10	10894
57	10	2478
58	10	2438
59	10	21808
60	10	9537
61	10	10199
62	10	1692
63	10	9597
64	10	4458
65	10	5264
66	10	5818
67	10	12227
68	10	6677
69	10	19447
70	10	3531
71	10	7192
72	10	1316
73	10	12743
74	10	12352
75	10	13031
76	10	14225
77	10	24112
78	10	21333
79	10	7984
80	10	11710
81	10	2132
82	10	4342
83	10	3660

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 8: Solution for Candidate Set Generated by the BAT algorithm

Set	xi	Total (Time)	Average (time)	Rounded (average(Time))
0	86	824527	9587.523256	9588
5	101	969729	9601.277228	9601
10	83	814755	9816.325301	9816
15	101	876881	8681.990099	8682
20	92	940185	10219.40217	10219
25	89	850592	9557.213483	9557
30	100	824688	8246.88	8247
35	108	938496	8689.777778	8690
40	84	579951	6987.361446	6987
45	98	814998	8316.306122	8316
50	91	673508	7401.186813	7401
55	79	580065	7342.594937	7343
60	83	605273	7292.445783	7292
65	91	480878	5284.373626	5284
70	87	411019	4724.356322	4724
75	78	266186	3412.641026	3413
80	97	332289	3425.659794	3426
85	86	264135	3071.337209	3071
90	108	159157	1473.675926	1474
95	92	45690	496.6304348	497
99	81	6543	80.77777778	81