SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V10I8P106

Volume 10 Issue 8, 59-67, August 2023
© 2023 Seventh Sense Research Group®

Original Article

Nature Inspired Data Placement Strategy in Distributed Cloud Environment using Improved Firefly Algorithm

B. Prabhu Shankar[1], H. Najmusher[2], N. Rajkumar[3], R. Jayavadivel[4], C. Viji[5], S. M. Nandha Gopal[6]

[1,3,4,5]*Department of Computer Science & Engineering, Alliance College of Engineering and Design, Alliance University, Bangalore, Karnataka, India.*
[2,6]*Department of Computer Science & Engineering, HKBK College of Engineering, Bangalore, Karnataka, India.*

[1]*Corresponding Author : prabu2000@gmail.com*

Received: 09 June 2023 Revised: 13 July 2023 Accepted: 08 August 2023 Published: 31 August 2023

Abstract *- The execution of scientific applications needs high-processing computers and requires massive storage. This resulted in deploying applications in a distributed environment with high performance and extensive storage. Applications processed in cloud platforms face intolerable delays due to data movement across the centres. Optimized distribution of datasets among the global data centres has become an essential issue in the distributed cloud environment. This work proposes an improved data placement called IFA Data Placement (IFA-DP) method for a heterogeneous cloud environment. An effective and efficient optimal data placement strategy is proposed using a metaheuristic global optimisation firefly algorithm. The metaheuristic behavior of fireflies finds a better optimal solution. The primary aim of this work is to reduce the response time and execution cost, which is then proved by the simulation results. The access time of the Proposed IFA-DP is less by at least 2s compared to the existing methods.*

Keywords *- Data placement, Firefly algorithm, Metaheuristic optimization, Cloud computing.*

1. Introduction

With the advent of technology and the proliferation of digital devices and systems, a massive amount of data is being generated and accumulated at an unprecedented rate. Practical insights from the raw data have urged the researchers to find a proper data management architecture.

In contrast, improper data management will lead to heavy traffic in processing data. Since the data size is tremendously significant, the need of the hour is the storage capacity and computing resources for processing applications. Distributed cloud architecture provided a good solution for this problem. Since datasets are stored in the distributed cloud environment, I/O operations and application processing using these datasets become a time-consuming process.

The increasing volumes of data storage, ranging from terabytes to petabytes and complex data structures, have posed challenges in data management. The emergence of cloud systems has provided a solution for handling such data-intensive scenarios. Cloud systems support two types of applications: data-intensive applications and computer-intensive applications. The four layers of cloud architecture are the Application, service, management, and infrastructure. network architecture between the servers is different, and correlations exist between the datasets accessed by the processes.

Dataset placement is critical in determining a cloud system's performance and efficiency [1-4], particularly in data-intensive applications. The distribution of datasets across different storage resources in the cloud and managing their processes can significantly impact performance [5].

The network traffic data locality concept was introduced in the literature, where the data is moved to servers rather than brought data to processing nodes. Later a property of interest locality was used where only required parts of the dataset are moved, not the whole dataset. Later dataset grouping played a vital role in placing the datasets in the appropriate data centre.

There is a need for the cloud platform to intelligently place the dataset into the relevant data centres to ensure less dataset movement across the data centres during execution [6]. It is necessary to consider a heterogenous data centre platform and data correlation for a reasonable data placement strategy. Data placement methods are classified into static data placement and dynamic data placement. All file's service time and access rate are required for static placement algorithms. Dynamic data placement algorithms determine the best placement of files across multiple disks based on various factors such as data access patterns, workload characteristics, and system constraints [7, 8]. When multiple process requests for multiple datasets frequently, a correlation exists between the datasets.

Few data correlation placement methods are discussed in [9-12]. Replication effectively reduces data scheduling overhead and has garnered significant research interest in cloud computing and distributed systems [13]. The primary necessity of data placement is 1. To effectively manage the total cost of a storage system 2. Effective distribution of data into different storage devices.

Data-intensive applications require a proper data placement strategy to minimize data access costs and mitigate service delays. The placement of runtime datasets can be particularly challenging due to the dynamic nature of these datasets. Additionally, transferring intermediate datasets can pose bottlenecks when the storage and computational capacity are limited.

However, optimizing data placement in a cloud system can be complex, especially considering factors like data access cost, Providing data to a remote system, data query pattern, and data centre storage capacity. Most existing data placement techniques aim to store user's datasets closer to the data access requests. Finally, the goal is to minimize data access latency and improve overall system performance.

The following are the primary contributions of this work:

1. Considering the data movement history, a model is designed in a cloud computing environment for data placement mapped to minimize response time.
2. The proposed model improves the Firefly Algorithm for data placement. By placing the datasets with their accessibility nature, the proposed IFA-DP minimizes the data movement during the execution. In addition, one potential contribution of the work could be developing a pre-allocation strategy for datasets to data centres. This strategy aims to prevent the grouping of datasets on a single server, which can lead to access delays and performance issues. IFA-DP assures less data movement during execution and ensures storing of relevant data locally.

The paper structure you described is as follows: Section 2, Literature Review, presents a comprehensive review of existing literature. Section 3 presents the details of the proposed data placement strategy.

Section 4 presents the results of simulations conducted to evaluate the effectiveness of the proposed data placement strategy. Finally, Section 5 summarizes the key findings of the research.

2. Literature Review

Stork is an advanced data placement scheduler offering comprehensive features for queuing, scheduling, managing, and monitoring job placement in a distributed computing environment. One of its key capabilities is the use of checkpointing techniques, which automate the process of saving the state of a job at specific intervals.

This ensures that if a job fails or needs restarted, it can resume from the most recent checkpoint rather than starting from scratch. Checkpointing provides a significant advantage in data processing as it reduces the need for manual intervention and improves overall job efficiency. Data placement during runtime is also adapted by this system [1].

Shyamala Doraimani and Adriana [9] proposed that Workload modelling plays a crucial role in data management within science grids, particularly when preserving time locality for data presaging. Time locality refers to the concept that data accessed closely in time will likely be reaccessed shortly. By understanding and leveraging this property, workload modeling techniques can optimize data placement and scheduling to improve overall system performance.

Zhao et al. [14] proposed an algorithm to reduce load balancing [15] and data movement between the data centers. However, specific ineffective fragments (presumably referring to data sets that cannot be efficiently stored in data centres) are excluded from the coding process, likely to avoid placing them in unsuitable locations.

To evaluate the fitness of each gene (placement), the authors defined a fitness function that considers both the dependency between data sets and the load balance across data centres. This fitness function measures how well a particular gene performs regarding efficient data placement. The genetic algorithm optimizes the genes to minimize data movement, aiming for an optimal solution that maximizes efficiency and reduces unnecessary data transfers.

According to the results presented by the authors, their proposed method outperformed other existing methods in workload modelling and data management. A genetic algorithm-based mathematical data placement technique was developed by Wei Guo et al. in [16].

The population size (G), mutation rate (Pm), and crossover rate (Pc) are all computed. Everyone's fitness value is obtained following the formation of the initial population BG. F = 1/(Bt) represents the fitness value. Individuals are ultimately selected using a roulette wheel [17]. The selected matrix undergoes crossover and mutation. [18, 19]. An individual (solution) in the population does not adhere to the storage capacity requirements; it is abandoned or considered infeasible. In their proposed method, Zhuo et al. [20], in a distributed computing environment, map and reduce tasks to bring together the key and values of the original data and achieve data locality [21, 22].

The authors implemented CORP in Hadoop 2.4.0 to evaluate the system's performance. The authors proposed a model for evaluating cost, effect, and variance from the sample datasets. Finally, the model improves the process and gradually reduces the execution time. CORP proposed model proved in terms of average data transmission of the entire distributed computing system.

Qing Zhao et al. [23] addressed the challenges of data placement in heterogeneous cloud environments and implemented new data placement techniques to store the data in heterogeneous clouds. Firstly, the authors proposed data dependency-based data clustering and recursive partitioning methods to consider the volume of the dataset and place the dataset in an optimal position. Secondly, the heuristic tree-to-tree data placement techniques reduce unnecessary data movement in high bandwidth channels. At last, Simulation results proved that the proposed strategy minimizes the amount of data transmission and execution time in a heterogeneous cloud.

Lizheng Guo, in his work [24], implemented the Particle Swarm Optimization (PSO) Algorithm to reduce the transmitting time and total execution cost. The proposed model described seems to involve Particle Swarm Optimization (PSO) with crossover, mutation, and local search algorithms for data placement in a cloud platform. PSO is a metaheuristic optimization technique that takes inspiration from the swarm behavior of particles. The final simulation result was to gain optimal solutions compared to existing CM-PSO and L-PSO methods, achieving less computational time.

Manmohan Chaubey and Erik Saule, in their work [25], investigated how dataset replication copes with the inaccuracies of processing time. The primary objective is to examine the problem you described involves scheduling independent tasks in a parallel system to minimize the makespan. However, the task execution times are only known as a multiplicative factor. The problem is divided into 2 phases: the offline and the online.

Data placement takes place in the offline phase, and the tasks are scheduled in the online phase. Therefore, three different kinds of strategies were investigated at different degrees of replication factors. There are I) no replication, II) replication is everywhere, and III) replication is in groups. The paper proposes approximation and theoretical lower bound algorithms and investigates the trade-off between the number of replicas and the guarantee on make span.

The algorithm proposed by Agarwal et al. [26] focuses on dynamically migrating Frequently used data to be stored in different subsets of a memory unit in an array format. The main aim of this work is to transfer frequently used data to load to active disks, and the rest of them can be shifted to energy-saving mode. Yang et al. [27] proposed an approach that helps the data placement in dynamic applications and efficiently handles the energy in a dynamic cloud model. The authors used a virtual machine for application encapsulation, resulting in the schedule and migrating the data in live applications[28].

The authors implemented algorithms to address specific challenges in a dynamic cloud environment. Let us break down the two algorithms (i) a Bin packing algorithm likely used to determine the optimal allocation of applications or workloads and (ii) an energy-aware heuristic algorithm that optimises resource allocation and utilization in a dynamic cloud environment by considering varying resource demands.

3. Data Placement Strategy in Cloud

Cloud computing offers services over the Internet through various applications and web-based tools [29] and its different models, namely public cloud and private cloud. While cloud computing has become increasingly popular for various applications, it is essential to note that cloud architectures alone may not always make optimal decisions regarding data processing, storage, and management [30]. Cloud data management can be complex due to the large volume of heterogeneous data and diverse data structures. Different data types, such as structured, semi-structured, and unstructured, pose challenges regarding organization, processing, and analysis within the cloud environment.

The variety of data types requires careful consideration of data storage mechanisms, access methods, and appropriate data processing techniques to manage and analyze the data in the cloud effectively. To overcome these challenges, organizations often employ specialized data management techniques, including data integration, preprocessing, indexing, and analysis algorithms, to handle the diverse nature of data in the cloud architecture. The data scheduling architecture is shown in Figure 1.

In a cloud environment, processors or CPUs can be heterogeneous, meaning they have different processing capabilities or capacities. This heterogeneity can arise due to variations in CPU clock speed, number of cores, cache size, memory bandwidth, and other architectural differences [31].

The cost of executing a task can vary based on the processor to which it is assigned and the diverse bandwidth available within the cloud nodes. These complexities and variations in cloud computing environments require careful consideration and planning for efficient data storage, processing, and management to ensure optimal performance and cost-effectiveness.

Figure 1 shows different processes and the datasets d1 through d10. Frequently accessed datasets are placed in data centres dc1 through dc4. Five different processes and ten datasets are depicted in the figure, where each process can execute successfully only with the availability of different datasets. If the required datasets are placed in different datasets, execution costs and time will be high. As shown in the figure, placing the frequently accessed datasets together in a data centre is wise. The proposed IFA-DP involves placing the dataset in an appropriate data centre in three stages. Stage 1 identifies the affinity degree of the datasets. The second stage involves the grouping of frequently accessed datasets together. Stage 3 focuses on correctly placing grouped datasets in suitable data centres. This step is critical to optimize data management and resource utilization in a cloud or distributed computing environment. The workflow architecture of the proposed method is depicted in Figure 2.

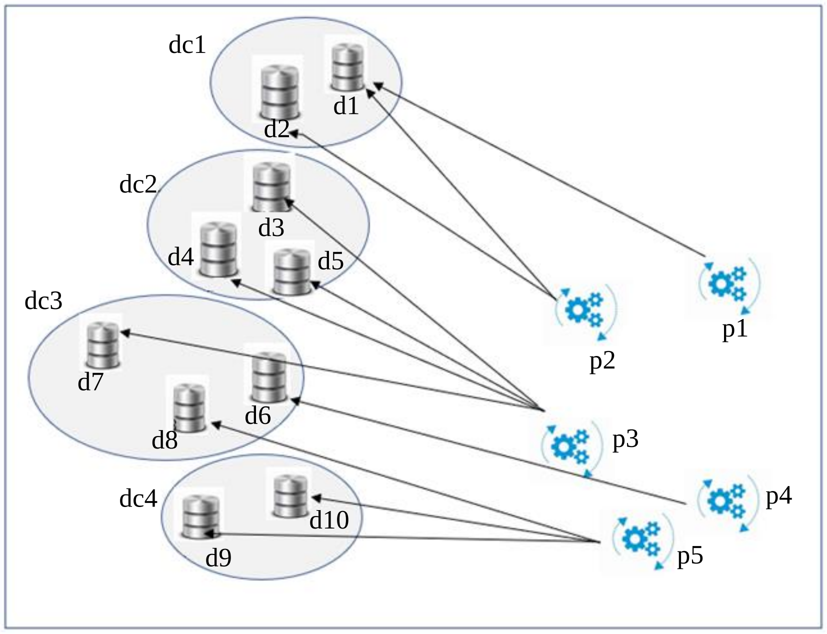

Fig. 1 Data scheduling architecture

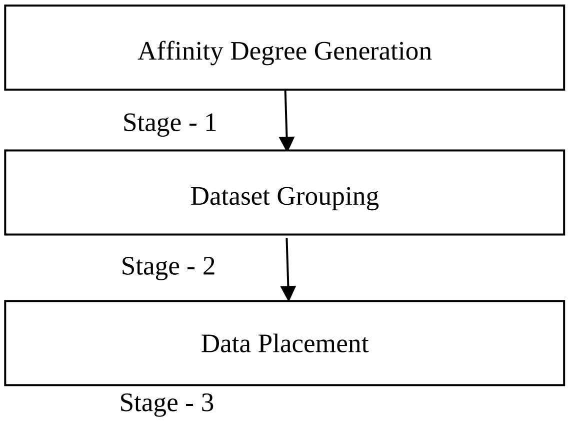

Fig. 2 Workflow of proposed IFA-DP

3.1. Affinity Degree Generation

IFA-DP mainly analyses node access and data movement history. Based on the analysis found, the best data placement strategy. One aspect of this analysis involves calculating the affinity degree for the datasets. During the execution of a process, if a dataset stays in a data centre without movement, then the dataset gains the highest affinity degree. The degree between the dataset and the data centre is directly proportional, and there is the slightest possibility of movement in the future. Affinity degree can be calculated using equation 1.

$$AD_{ij}=f_n(n_{ij}/\ tot_{access}*(1-MLE_{ij})) \quad (1)$$

AD_{ij} is the affinity degree of dataset i with node j. n_{ij} represents the total number of times node j has accessed dataset i; the total number of times the dataset has been accessed is given by tot_{access}. The history of access frequency during a period t is defined as the ratio of the total number of times dataset i was accessed by node j to the total number of times dataset i was accessed. MLE_{ij} is the likelihood of dataset movement in the future. MLE_{ij} is based on the history of data movements.

3.2. Dataset Grouping

Initially, the access likeliness of the datasets is identified, and then they are grouped to form a Data Grouping Matrix (DGM). Access likeliness Alike of dsi and dsj is the total number of processes that utilize dsi and dsj datasets. Access likeliness of all the datasets can be calculated using equations 2,3 and 4.

$$\text{dsacci}= \sum(p_{1,}p_{2,} \ldots .p_n), \tag{2}$$

Where $p_{1,}p_{2,} \ldots .p_n$ are the processes that access dsi.

$$\text{dsaccj}= \sum(p_{1,}p_{2,} \ldots .p_n), \tag{3}$$

Where $p_{1,}p_{2,} \ldots .p_n$ are the processes that access dsj.

$$\text{Alike}(i,j) = \text{dsacci} \cap \text{dsaccj} \tag{4}$$

DGM is formed from the Access likeliness values of the dataset. The values in the DGM represent the total number of times the datasets have been accessed together. Clustered DGM (CDGM) is formed from DGM by grouping similar values. DGM is converted to CDGM using the Bond Energy Algorithm (BEA). This clustered matrix helps to decide the placement of dataset groups. When the datasets are, the process is repeated until all the datasets are grouped. Finally, highly associated sub-matrices are identified as optimal submatrices from CDGM.

3.3. Data Placement Strategy with Improved Firefly Algorithm

3.3.1. Conventional Firefly Algorithm

The Firefly Algorithm was proposed by Yang [27] at the University of Cambridge. The Firefly Algorithm is a population-based optimization algorithm inspired by the behavior of fireflies. Fireflies move towards brighter fireflies, simulating the attraction between fireflies based on their relative brightness. The behavior of fireflies in the algorithm can be linked to two main hypotheses observed in the natural world. In optimization terms, this can be interpreted as the fireflies being attracted to promising solutions considered "better" in the objective function. The second hypothesis suggests that fireflies use their flashes to attract potential prey. In the context of the Firefly Algorithm, this can be understood as fireflies moving towards brighter fireflies, representing the search for better solutions that can potentially overcome the current solution's drawbacks. By leveraging these principles, the Firefly Algorithm aims to iteratively improve the population of potential solutions by simulating fireflies' movement and attraction behaviour.

Yang introduced the following conditions in the Firefly Algorithm: (1) The brightness of a firefly represents the quality or fitness of a potential solution. The attraction is negatively correlated with the Euclidean distance squared. (2) Fireflies exhibit random movements to investigate the search area. This randomness ensures a degree of global exploration and helps to prevent the algorithm from becoming stuck in local optima. Using these conditions, the Firefly Algorithm iteratively updates the positions of fireflies in the search space, with each firefly adjusting its position based on the attractiveness of other fireflies and incorporating random movements. Through this iterative process, the algorithm aims to converge towards better solutions and optimize the given objective function. It is worth noting that the specific implementation details of the Firefly Algorithm, such as the parameter settings, movement rules, and termination conditions, can vary depending on the problem at hand and the preferences of the algorithm designer.

Algorithm 1. Firefly Algorithm

```
Begin
  Generate the initial population of fireflies (xi, i = 1 to n)
  Calculate light intensity I for each firefly based on its
  fitness
     Set parameters (MaxGeneration , γ , α )
     Repeat for t = 1 to MaxGeneration:
        For each firefly i = 1 to n:
           For each firefly j = 1 to n:
              If Ij > Ii:  # Brighter fireflies attract
                 Calculate distance r between fireflies i and j
                 Calculate attractiveness via exp (-γ * r^2)
                 Generate random movement factor β
                 Update the position of firefly i
                 Evaluate new fitness and update light
                 intensity Ii
        Rank fireflies based on light intensity.
end
```

The brightness of a maximization problem can be proportional to its objective function's value. Similar to how the fitness function in genetic algorithms is constructed, other kinds of brightness can also be defined. Algorithm 1 depicts the basic firefly Algorithm.

Initially, the data placement is mapped with the Firefly algorithm. In the workflow model, the data centers represented as DC = dc1, dc2, dc3, …., dcm and a dataset represented as DS= ds1, ds2, ds3…dsn.

The proposed mapping is as follows:

1. In the data placement problem, the number of dimensions in the optimization problem (d) can be mapped to the number of datasets (n).
2. The location of each firefly (xi) in the Firefly Algorithm can be mapped to a possible solution to the data placement problem.
3. Intensity (I) to the fitness of data scheduling solutions.
4. The attraction of low-intensity fireflies to those with higher intensity can be mapped to changing non-optimal data schedules to more optimal schedules in the data placement problem [32].

Finally, new solutions are obtained, and light intensity is updated. The best-fitting fireflies are chosen for the subsequent iteration. The best firefly is chosen as the most appropriate solution.

Figure 3 shows how the low-intensity firefly moves towards the high-intensity firefly. Moreover, obtain a new solution in the proposed model. Figure 4 illustrates the general architecture of the proposed data placement method (IFA-DP). Assuming that data placement and scheduling problem with a total of m data centres and n tasks, and we have a population of fireflies with a maximum population size of maxp, we can associate each firefly in the initial population with a possible solution in the data placement and scheduling problem, each firefly's location is defined as a vector of d members.

The Firefly xi = {1,2,2,3,1,4,3} represents scheduling. Datset1 is accessed by task 1, dataset 2 and 3 are accessed by task 2; dataset 4 is accessed by task 3, dataset 5 is accessed by task 1and so on. Calculate the fitness of the initial population by defining intensity i for fireflies at position xi. Determine the brightness attraction.

Considering fireflies i and j, firefly i should move towards firefly j if i^th fitness is more significant than j^th firefly. The conventional Firefly Algorithm is enhanced by selecting k random positions of low-intensity firefly and replacing the positions with the corresponding positions of j^th firefly. Thus, i^th firefly with low intensity moves towards with high intensity of j^th firefly. R denotes the distance between the i^th and j^th.

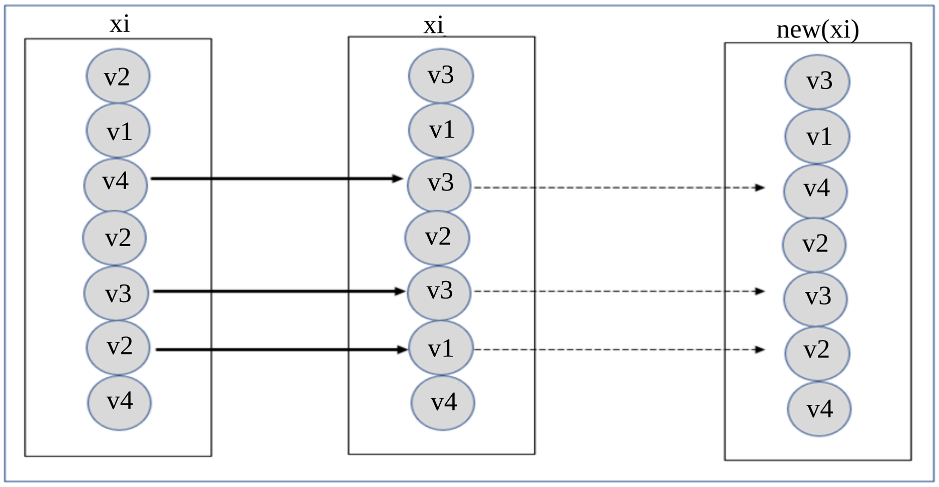

Fig. 3 Proposed firefly movement

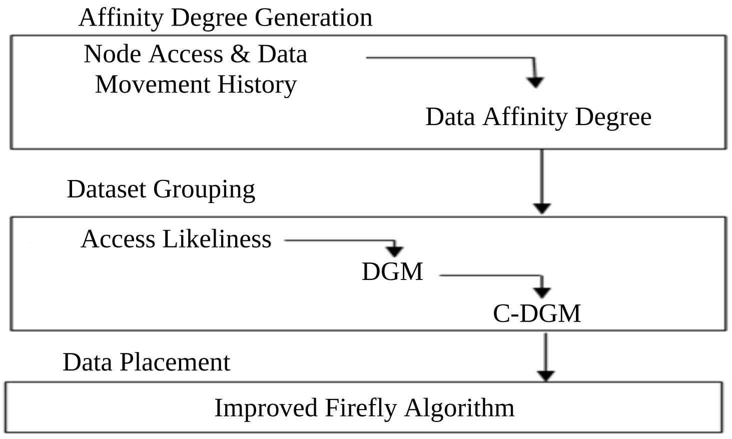
Affinity Degree Generation
Node Access & Data Movement History
Data Affinity Degree
Dataset Grouping
Access Likeliness
DGM
C-DGM
Data Placement
Improved Firefly Algorithm

Fig. 4 Proposed data placement strategy using improve firefly algorithm

The three stages of the proposed system are shown in the figure above. The first stage is the affinity degree generation stage, which involves calculating the affinity degree of the datasets using the previous data movement and node access history. Stage two involves finding the access likeliness of the datasets that resulted in grouping and clustering the similarly valued data. Finally, the clustered data is significantly placed in the appropriate data centres using the proposed improved Firefly Algorithm.

4. Results and Discussion

Simulations followed by the execution of actual applications are performed in the Cloud Sim framework. Metrics used to evaluate the proposed system are execution cost and access time. The proposed method IFA-DP is compared with existing methods Data-g Rouping – Aware (DRAW) data placement scheme [33] and MADP [34]. Parameters for the simulation are tabulated in Table 1. Details of the simulation analysis are described in the following sections. Utility values are also compared in this experiment.

Table 1. Simulation parameters

Entity	Quantity
Data Centre	1
Hosts in DC	500
Ram Capacity	16/64 GB
Processing Elements	4/8
Processing capacity of Processing Element	90/120/150/225 MIPS
Process length/instructions	600000 to 200000000 MI
Data set size	100 to 700 in steps of 100

Figure 5 compares the average response time for existing DRAW, MADP, and proposed IFA-DP for different scales of datasets. It is evident from the figure that the proposed IFA-DP shows less response time for all dataset scales. The response time of DRAW and MADP seems to be similar for most of the dataset scales.

Table 2. Comparison of an average response time of different dataset scales for DRAW, MADP, and proposed IFA-DP

No of Datasets	DRAW	MADP	IFA-DP
100	2	1.5	0.5
200	1.7	1.8	1
300	2.1	3	2
400	3.5	2.5	2.3
500	2.2	3.3	1.5
600	3.6	2.6	1.8
700	3	2.4	1

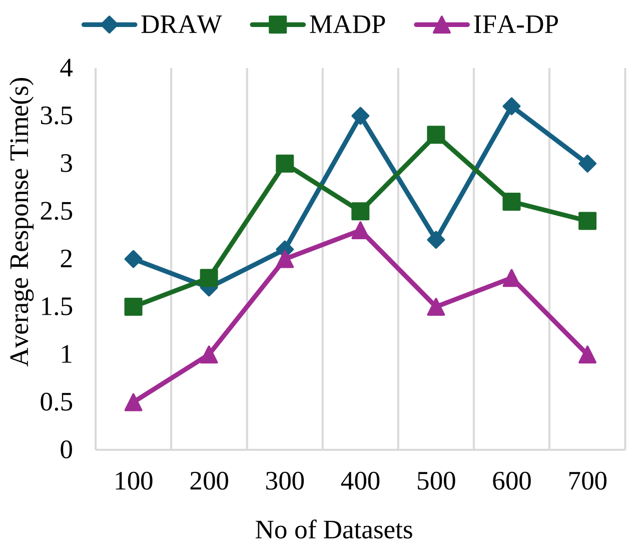

Fig. 5 Comparison of an average response time of different dataset scales for DRAW, MADP, and proposed IFA-DP

It is proven from Figure 6 that two solutions are obtained for each dataset scale. The goal solution is 2 for dataset scale 100,200, and 500. The goal solution is 1 for 700 datasets. The utility function used in the data placement and scheduling problem can obtain a balanced solution for arbitrary scales of datasets.

Table 3. Comparison of utility value for different ranges of datasets

No of Datasets	Solution 1	Solution 2
100	0.2	0.6
200	0.3	0.7
500	0.3	0.8
700	0.98	0.65

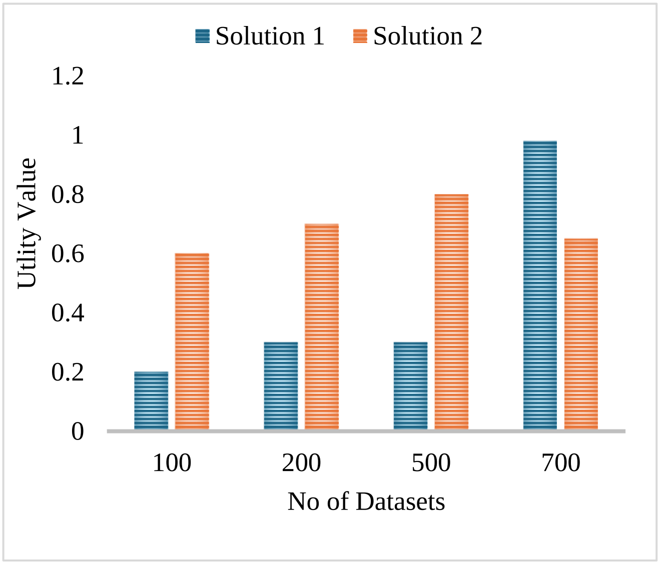

Fig. 6 Comparison of utility value for different ranges of datasets

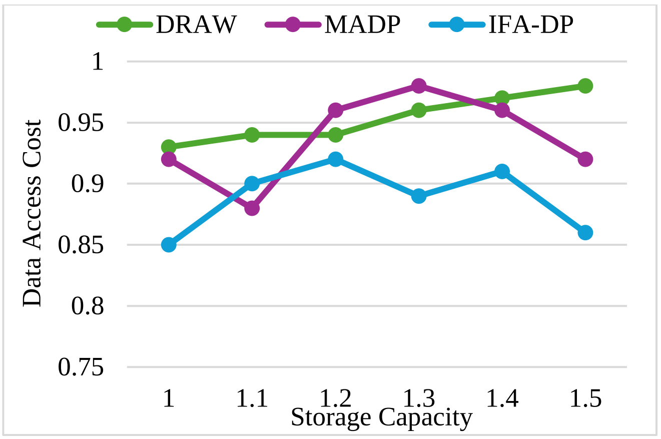

Fig. 7 Comparison of data access cost vs Storage capacity

Table 4. Comparison of data access cost vs Storage capacity

Storage Capacity	DRAW	MADP	IFA-DP
1	0.93	0.92	0.85
1.1	0.94	0.88	0.9
1.2	0.94	0.96	0.92
1.3	0.96	0.98	0.89
1.4	0.97	0.96	0.91
1.5	0.98	0.92	0.86

Figure 7 in comparing data access costs, the proposed IFA-DP (Firefly Algorithm-based Data Placement) technique outperforms existing data placement techniques. The cost values associated with data access are normalized. The reason might be that the proposed IFA-DP focuses on placing frequently accessed data items together in the data center. Thus, the performance of IFP-DP is superior to DRAW and MADP in terms of data access cost.

5. Conclusion and Future Enhancement

This cloud computing era challenges data storage and management since the available data is diverse, complex, and heterogeneous. Data placement is an important aspect that makes cloud computing possible. This paper addresses the data scheduling problem by proposing a nature-inspired data placement approach in a cloud system. The contribution of the work is as follows:

The affinity degree of the datasets is identified from which datasets are grouped using access likeliness. Finally, the grouped datasets are placed in the appropriate data centres using an improved Firefly algorithm.The proposed IFA-DP is compared with two existing data placement methods: DRAW and MADP. Simulation results prove that the proposed IFA-DP excels DRAW and MADP regarding response time and data access cost. Data access cost for IFA-DP is less for all the storage capacity comparing DRAW and MADP. Our future work will focus on replicating frequently accessed datasets for better response time. In addition, load balance can be investigated in further research.

References

[1] Tevfik Kosar, and Miron Livny, "A Framework for Reliable and Efficient Data Placement in Distributed Computing Systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 10, pp. 1146-1157, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[2] Huan Liu, and Dan Orban, "GridBatch: Cloud Computing for Largescale Data-Intensive Batch Applications," *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, Lyon, France, pp. 295-305, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ewa Deelman et al., "The Cost of Doing Science on the Cloud: The Montage Example," *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, Austin, Tx, USA, pp. 1-12, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[4] T. Kosar, and M. Livny, "Stork: Making Data Placement a First-Class Citizen in the Grid," *24th International Conference on Distributed Computing Systems*, Tokyo, Japan, pp. 342-349, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[5] Ewa Deelman, and Ann Chervenak, "Data Management Challenges of Data-Intensive Scientific Workflows," *2008 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, Lyon, France, pp. 687-692, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[6] S. Veerapandi, R. Surendiran, and K. Alagarsamy, "Enhanced Fault Tolerant Cloud Architecture to Cloud-Based Computing using Both Proactive and Reactive Mechanisms," *DS Journal of Digital Science and Technology*, vol. 1, no. 1, pp. 32-40, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Lili Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the Placement of Web Server Replicas," *Proceedings IEEE INFOCOM 2001,Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Anchorage, AK, USA, vol. 3, pp. 1587-1596, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[8] K. R. Pattipati, and J. L. Wolf, "A File Assignment Problem Model for Extended Local Area Network Environments," *International Conference on Distributed Computing Systems*, Paris, France, pp. 554-561, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[9] Shyamala Doraimani, and Adriana Iamnitchi, "File Grouping for Scientific Data Management: Lessons from Experimenting with Real Traces," *HPDC '08: Proceedings of the 17th International Symposium on High Performance Distributed Computing*, Boston, pp. 153-164, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[10] Gilles Fedak, Haiwu He, and Franck Cappello, "BitDew: A Programmable Environment for Large-Scale Data Management and Distribution," *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, Austin, TX, USA, pp. 1-12, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[11] Marcel Chibuzor Amaechi, Matthias Daniel, and Bennett E O, "Data Storage Management in Cloud Computing using Deduplication Technique," *SSRG International Journal of Computer Science and Engineering*, vol. 7, no. 7, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Zheng Pai et al., "A Data Placement Strategy for Data-Intensive Applications in Cloud," *Chinese Journal of Computers*, vol. 33, no. 8, pp. 1472-1480, 2010. [Google Scholar] [Publisher Link]

[13] Dharma Nukarapu et al., "Data Replication in Data Intensive Scientific Applications with Performance Guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1299-1306, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zhao Er-Dun et al., "A Data Placement Strategy Based on Genetic Algorithm for Scientific Workflows," *2012 8th International Conference on Computational Intelligence and Security*, Guangzhou, China, pp. 146-149, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[15] Nishu Rana, and Pardeep Kumar, "Random Walk-Based ACO Load Balancing Algorithm for Cloud Computing Environment," *International Journal of P2P Network Trends and Technology*, vol. 8, no. 6, pp. 8-15, 2018. [Publisher Link]

[16] Wei Guo, and Xinjun Wang, "A Data Placement Strategy Based on Genetic Algorithm in the Cloud Computing Platform," *2013 10th Web Information System and Application Conference*, Yangzhou, China, pp. 369-372, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[17] Thomas Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[18] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1998. [Google Scholar] [Publisher Link]

[19] Kalyanmoy Deb et al., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[20] Zhuo Tang et al., "A Data Skew Oriented Reduce Placement Algorithm Based on Sampling," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1149-1161, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Mohammad Javad Abbasi, and Mehrdad Mohri, "Scheduling Tasks in the Cloud Computing Environment with the Effect of Cuckoo Optimization Algorithm," *SSRG International Journal of Computer Science and Engineering*, vol. 3, no. 8, pp. 1-9, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[22] Neha Thakkar, and Rajender Nath, "Discrete Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environment," *International Journal of P2P Network Trends and Technology*, vol. 8, no. 6, pp. 1-7, 2018. [Publisher Link]

[23] Qing Zhao, Congcong Xiong, and Peng Wang, "Heuristic Data Placement for Data-Intensive Applications in Heterogeneous Cloud," *Journal of Electrical and Computer Engineering*, vol. 2016, pp. 1-8, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[24] Lizheng Guo et al., "A Particle Swarm Optimization for Data Placement Strategy in Cloud Computing," *Information Engineering and Applications*, pp. 946-953, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[25] Manmohan Chaubey, and Erik Saule, "Replicated Data Placement for Uncertain Scheduling," *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, Hyderabad, India, pp. 464-472, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[26] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi, "Big Data and Cloud Computing: Current State and Future Opportunities," *EDBT/ICDT '11: Proceedings of the 14th International Conference on Extending Database Technology*, Uppsala, pp. 530-533, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[27] Xin-She Yang, "Firefly Algorithms for Multimodal Optimization," *Stochastic Algorithms: Foundations and Applications*, pp. 169-178, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[28] S. Veerapandi, R. Surendiran, and K. Alagarsamy, "Live Virtual Machine Pre-copy Migration Algorithm for Fault Isolation in Cloud Based Computing Systems," *DS Journal of Digital Science and Technology*, vol. 1, no. 1, pp. 23-31, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[29] Michael Armbrust et al., "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[30] Peter Mell, and Timothy Grance, "*The NIST Definition of Cloud Computing*," Thesis, National Institute of Standards and Technology, 2011. [Google Scholar] [Publisher Link]

[31] Eduardo Pinheiro, and Ricardo Bianchini, "Energy Conservation Techniques for Disk Array-Based Servers," *ICS '04: Proceedings of the 18th Annual International Conference on Supercomputing*, pp. 68-78, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[32] B. Prabhu Shankar, and S. Chitra, "Optimal Data Placement and Replication Approach for SIoT with Edge," *Computer Systems Science and Engineering*, vol. 41, no. 2, pp. 661-676, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[33] B. Prabhu Shankar et al., "Energy-Efficient Data Offloading using Data Access Strategy-Based Data Grouping Scheme," *SSRG International Journal of Electronics and Communication Engineering*, vol. 10, no. 5, pp. 28-37, 2023. [CrossRef] [Publisher Link]

[34] T. V. V. Satyanarayana et al., "A Secured IoT-Based Model for Human Health through Sensor Data," *Measurement: Sensors*, vol. 24, p. 100516, 2022. [CrossRef] [Google Scholar] [Publisher Link]